POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale in Ingegneria del Cinema e dei Mezzi di Comunicazione

Tesi di Laurea Magistrale

Studio delle tecniche di animazione e di Motion Capture facciale per la realizzazione di un prodotto di animazione indipendente

Nascita, sviluppo e distribuzione sul mercato internazionale dell'Add-On di Blender *ReveRig*

Politecnico di Torino

Relatore
prof. Riccardo Antonio Silvio Antonino

Candidato
Nicolò Vigna
matricola: s257735

Anno Accademico 2020/2021

Ringraziamenti

Vorrei ringraziare il Prof. Riccardo Antonino, relatore di questa tesi, e Robin Studio S.r.l.s., azienda con la quale ho collaborato, per avermi dato l'opportunità di entrare a far parte dell'ambizioso progetto *Reverie Dawnfall*, permettendomi di approfondire tematiche a me care e di crescere sia a livello professionale che personale.

Ringrazio la vasta community di Blender che ha creduto in *ReveRig* e ha sostenuto il progetto.

Un ringraziamento speciale alla mia famiglia, in particolare a mia madre e mio padre: è grazie al loro sostegno, sia morale che economico, e ai loro sacrifici mai scontati se oggi sono riuscito a raggiungere questo importante traguardo.

A mia sorella, che tra litigate ed abbracci si è rivelata la miglior compagna in questo viaggio chiamato vita.

Agli amici di una vita e ai miei amici d'università più cari, grazie per aver sopportato le mie ansie e per aver reso questo percorso indimenticabile.

Abstract

La tesi ha l'obiettivo di fornire un resoconto del lavoro svolto in collaborazione con l'azienda Robin Studio S.r.l.s., in particolare per l'individuazione delle migliori tecnologie e metodologie da impiegare nella realizzazione delle animazioni facciali dei personaggi della serie d'animazione indipendente *Reverie Dawnfall*. Importante spazio viene dato alla discussione di *ReveRig*, l'Add-On di Blender nato e sviluppato all'interno di questo lavoro di tesi e oggi distribuito a livello internazionale.

Il documento di tesi si divide in quattro fasi principali. La prima fase ha lo scopo di introdurre il concetto di animazione facciale offrendo un panorama generale sulle principali complessità e problematiche che governano questo settore.

La seconda è una fase di ricerca delle tecniche per l'animazione facciale attualmente esistenti. Tale ricerca comprende le tecniche per la manipolazione della geometria (es. blendshape), le tecniche per l'acquisizione dei dati (es. motion capture) e le tipologie di rigging facciale. Ciascuna delle tecniche individuate viene contestualizzata in base alla tipologia di prodotto e di contesto dentro le quali ricade la produzione. Particolare attenzione viene rivolta verso le tecniche adatte per una produzione indipendente.

Con la terza fase si entra nel dettaglio del progetto *Reverie Dawnfall*. Inizialmente viene individuata la situazione e il contesto dalla quale il lavoro di tesi ha avuto origine. Grazie all'individuazione del contesto è possibile definire gli obiettivi da raggiungere e i limiti da superare, elementi fondamentali per poter definire le migliori tecniche, strumenti e il miglior workflow da introdurre nella produzione indipendente di *Reverie Dawnfall*. Da qui emerge la necessità di utilizzare programmi open source come Blender e Papagayo-NG, inoltre diventa centrale l'utilizzo della tecnologia di Mocap markerless offerta da ARKit.

La quarta e ultima fase racchiude tutto il lavoro di ricerca e di sperimentazione atto ad adattare strumenti precedentemente esistenti alla pipeline di produzione di *Reverie Dawnfall*, ma soprattutto rivolto alla creazione ex novo dell'Add-On (plug-in) di Blender chiamato *ReveRig*, avente scopo di offrire all'utente una serie di strumenti fondamentali per la realizzazione dell'animazione facciale.

L'obiettivo ultimo del lavoro di tesi è quello di individuare tecniche e strumenti già esistenti e di creare strumenti appositi da poter integrare all'interno dalla già avviata produzione indipendente della serie animata *Reverie Dawnfall* per la realizzazione delle animazioni facciali, mantenendo al contempo uno sguardo costante al mercato indipendente internazionale.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni l'utilizzo della computer grafica si è affermato in modo preponderante all'interno dell'industria dell'intrattenimento e non solo. Sempre di più sono i prodotti che racchiudono al loro interno personaggi realizzati ed animati completamente in digitale. A partire dai primi lavori di Frederic I. Parke [1] nel 1972, l'animazione facciale ha ricoperto un ruolo sempre più importante nell'ambito dell'animazione digitale. La sua complessità la rende ancora oggi un importante elemento di studio e di ricerca, tanto che, nonostante il rapido avanzamento tecnologico in questo settore, realizzare un'animazione facciale credibile rimane uno degli aspetti più complicati da soddisfare.

Grazie al veloce avanzamento tecnologico, oggi anche studi indipendenti hanno la possibilità di utilizzare strumenti avanzati un tempo impensabili. Questo incentiva anche piccole realtà a realizzare prodotti animati una volta legati unicamente al mondo delle grosse case di produzione.

Il lavoro di tesi è svolto in collaborazione con Robin Studio S.r.l.s., uno studio di produzione e animazione con sede a Torino co-fondato nel 2017 da Riccardo Antonino, docente del corso di Effetti Speciali presso il Politecnico di Torino e relatore di questa tesi. All'interno di questo contesto totalmente indipendente, sempre nel 2017, prende forma e si sviluppa il progetto *Reverie Dawnfall*, una serie d'animazione 3D cyberpunk. Il seguente lavoro di tesi nasce dall'esigenza, da parte dello Studio Robin, di approfondire le ricerche e lo studio relativo all'animazione facciale per i personaggi della serie animata in modo da identificare strumenti, tecniche e processi di lavoro da poter implementare nella pipeline di produzione già avviata. Tale ricerca ha permesso di ideare e sviluppare l'Add-On di Blender *ReveRig*.

La motivazione principale che ha portato alla scelta di questo argomento di tesi è la mia personale passione per il mondo degli Effetti Speciali (VFX) e della Computer Grafica (CG) in generale. Tale passione mi ha spinto a intraprendere come percorso di studi universitario il corso di laurea triennale in Ingegneria del Cinema e dei Mezzi di Comunicazione, e successivamente come proseguimento il corso di laurea magistrale. Nei cinque anni di studi questa passione si è alimentata e consolidata. Inoltre, questo lavoro di tesi risulta perfetto per soddisfare la mia esigenza di unire una parte più tecnica ed ingegneristica ad una parte, invece, più creativa. Il tutto risulta ancora più interessante in quanto il lavoro è stato svolto in collaborazione con l'azienda Robin Studio S.r.l.s., questo mi ha dato modo di scontrarmi con problemi e limiti reali presenti in questo settore, che altrimenti difficilmente sarei riuscito ad individuare.

L'obiettivo posto dall'azienda e perseguito grazie a questo lavoro di tesi consiste nell'identificare tecniche, strumenti e processi di lavoro relativi all'animazione facciale, da poter inserire in modo sostenibile all'interno della pipeline di produzione della serie animata *Reverie Dawnfall*. Tale proposta di tesi ha un forte carattere sperimentale, carattere che contraddistingue l'intero progetto.

Il lavoro di tesi si va ad inserire in un progetto più ampio e già in fase di produzione, diventa quindi necessario rispettare i vincoli, gli stili e le metodologie utilizzate dallo Studio Robin. In particolare, il processo individuato per l'animazione facciale dovrà adattarsi ad una produzione fortemente indipendente, caratterizzata da basso budget e scarse risorse in termini di personale specializzato e di tempo a disposizione. Pertanto, il lavoro di tesi da me svolto si propone di individuare delle tecniche e delle tecnologie che vadano a ridurre significativamente i tempi necessari per la realizzazione delle animazioni facciali dei personaggi, principali e non, andando quindi ad abbattere i costi di produzione. Il lavoro viene svolto basandosi principalmente sulle esigenze di produzione dell'azienda ma viene costantemente rivolto uno sguardo a quella che è l'industria italiana ed internazionale, in particolar modo facendo riferimento alla community di Blender.

Il lavoro svolto si divide principalmente in due fasi. Nella prima fase si è effettuata una ricerca approfondita relativa alle principali tecniche e tecnologie attualmente disponibili per il mercato dell'animazione facciale, sia per produzioni *Major* (es. Pixar, WesaDigital, Industrial Light & Magic...) che per produzioni indipendenti, con evidente concentrazione per le seconde, con lo scopo di individuare quelle in grado di apportare maggiori vantaggi alla produzione seriale di *Reverie Dawnfall*. La seconda fase, caratterizzata da una forte natura sperimentale e da un processo trial&error, mi ha portato ad adattare strumenti già esistenti alla pipeline di produzione di *Reverie Dawnfall*, ma soprattutto a creare ex-novo appositi strumenti e funzionalità, culminati nella progettazione, sviluppo e distribuzione sul mercato internazionale dell'add-on di Blender *ReveRig*. ReveRig è appositamente pensato per agevolare il processo di animazione facciale all'interno di Blender.

Outline

La tesi si suddivide un due parti principali ed è articolata in dodici capitoli:
La parte I, composta dai capitoli 2 - 6, racchiude l'ampio lavoro di ricerca effettuato in merito allo stato dell'arte ed altri elementi teorici fondamentali per comprendere appieno la materia.

- **Capitolo 2**: ha lo scopo di introdurre il concetto di animazione facciale offrendo un panorama generale sulle principali complessità e problematiche che governano questo settore. Vengono fornite le basi anatomiche del volto umano; si discute il problema dell'Uncanny Valley e come questo possa essere evitato; viene presentato il sistema di codifica facciale FACS definendone potenziali e limiti; vengono riportati i dodici principi che ancora oggi guidano il lavoro dell'animatore; vengono fornite le indicazioni principali per effettuare una corretta modellazione 3D del volto ed infine viene riassunto il flusso di lavoro che porta alla realizzazione di un'animazione facciale introducendo i capitoli successivi.
- **Capitolo 3**: vengono analizzate le principali tecniche di deformazione della geometria poligonale mettendo a confronto tecniche basate su: interpolazione, blendshape, deformazioni bone-based, parametrizzazione, modelli basati sui muscoli e modelli basati su pseudo-muscoli. Per ciascuna di queste tecniche si evidenziano vantaggi, svantaggi e il

principale campo di applicazione, in particolare prestando maggiore attenzione all'utilizzo di blendshape.
- **Capitolo 4**: un breve capitolo per definire in modo chiaro in cosa consiste il rigging facciale e come questo si compone. Particolare attenzione viene rivolta all'utilizzo dell'interfaccia utente (UI) e alle sue differenti conformazioni.
- **Capitolo 5**: vengono descritte ed analizzate le principali tecniche di acquisizione dati. In particolare, ci si sofferma sull'animazione tradizionale basata sui keyframe, sull'animazione guidata da testo o da audio e sulle tecniche di motion capture. Importante spazio viene dato anche al processo che porta a realizzare un'animazione facciale all'interno dei film di animazione "stile Pixar".
- **Capitolo 6**: vengono presentati i principali fattori che influenzano la scelta di determinati strumenti e tecniche piuttosto che altri. In particolare viene dato maggior rilievo alla tipologia di prodotto e alla tipologia di contesto dentro la quale la produzione dell'opera nasce e si sviluppa.

Con la parte II, composta dai capitoli 7 - 13, si entra nello specifico di quanto realizzato per il progetto *Reverie Dawnfall*. Dopo una prima parte dove viene presentato il progetto e gli obiettivi da raggiungere, i capitoli rimanenti riportano il cuore di questo lavoro di tesi, ovvero il mio personale apporto al progetto di *Reverie Dawnfall* e più in generale alla community di Blender nel campo dell'animazione facciale, ovvero l'Add-On *ReveRig*.

- **Capitolo 7**: viene presentato il progetto *Reverie Dawnfall* e la situazione dalla quale il lavoro di tesi ha avuto origine. Particolare spazio viene dato ai limiti che lo Studio Robin mi ha chiesto di superare e quindi gli obiettivi da raggiungere.
- **Capitolo 8**: vengono presentati i due strumenti principali che verranno utilizzati all'interno della pipeline di produzione della serie animata, ovvero Blender e Papagayo, mostrandone limiti e potenziale.
- **Capitolo 9**: viene trattato il cuore del processo di animazione facciale individuato. Viene presentata la tecnologia ARKit, alla base della tecnica di acquisizione dati individuata, ovvero il motion capture markerless, e il metodo semi-automatizzato per generare le blendshape necessarie.
- **Capitolo 10**: viene presentato il lavoro di ricerca e di sperimentazione relativo alla realizzazione semi-automatizzata dell'animazione della lingua. Si approfondisce il funzionamento e la struttura del programma open source Papagayo-NG, viene definito uno nuovo set di fonemi/visemi per l'animazione della lingua, viene descritto il lavoro di modifica realizzato al codice sorgente del programma ed infine viene definito il workflow da seguire all'interno della produzione seriale.
- **Capitolo 11**: tratta in modo approfondito le motivazioni, lo sviluppo e l'implementazione, all'interno della pipeline di lavoro, dell'add-on di Blender *ReveRig*, nato e sviluppato all'interno di questo progetto di tesi. Questo capitolo più di tutti racchiude le conoscenze teoriche e pratiche apprese durante il percorso di studi e il lavoro di tesi. Viene inoltre dato spazio ai primi risultati relativi alla pubblicazione online di *ReveRig* e alle opportunità che questo ha comportato.
- **Capitolo 12**: vengono presentati i risultati ottenuti con l'implementazione del workflow, delle tecniche e delle tecnologie descritte nei capitoli precedenti di questo documento di tesi, per la produzione di *Reverie Dawnfall*.

- **Capitolo 13**: una breve riflessione sui risultati raggiunti grazie alla tesi e una panoramica degli sviluppi futuri del progetto *Reverie Dawnfall* e dell'add-on *ReveRig*.

Grazie al seguente lavoro di tesi è stato possibile studiare a fondo l'industria dell'animazione facciale 3D e quindi individuare le tecnologie, gli strumenti e le metodologie più adatte ad essere inserite all'interno di un contesto indipendente, in particolar modo soddisfacendo le richieste e le esigenze dello studio Robin per la realizzazione della serie animata *Reverie Dawnfall*. Per soddisfare tali necessità, all'interno di questo lavoro di tesi, è nato e si è sviluppato l'Add-On *ReveRig*, strumento apprezzato sia da Robin Studio S.r.l.s. sia dalla cummunity internazionale di Blender. I risultati, approfonditi nel capitolo 13, evidenziano come grazie a *ReveRig* è stato possibile intercettare e soddisfare delle esigenze ampiamente diffuse all'interno del mercato indipendente dell'animazione computerizzata.

Intorno a questo settore si è generato molto interesse con conseguente rapido avanzamento tecnologico, questo apre importanti opportunità per i futuri sviluppi di *ReveRig* e per la realizzazione di *Reverie Dawnfall*.

Parte I

Teoria e stato dell'arte

Capitolo 2

Introduzione all'animazione facciale computerizzata

L'animazione facciale computerizzata è un'area della computer grafica (CG) che incapsula metodi e tecniche per generare e animare il volto di un personaggio, in particolare per quanto riguarda la creazione delle diverse espressioni facciali.

Ogni giorno, in un modo o nell'altro, entriamo in contatto con altri esseri umani e di conseguenza con i loro volti. Pertanto, siamo altamente "addestrati", sia a livello evolutivo che a livello sociale, a riconoscere ed analizzare la più piccola sfumatura nel volto del nostro interlocutore. È principalmente per questo motivo che creare un'animazione facciale convincente e visivamente interessante, come ad esempio far sorridere o far sbattere le palpebre al nostro personaggio, risulta tutt'altro che un compito semplice. Richiede, invece, un'ottima conoscenza e comprensione del complesso sistema anatomico che si nasconde dietro la comparsa di ogni singola espressione facciale, una buona conoscenza dei principi e delle tecniche di animazione e la conoscenza dei sistemi e delle tecnologie attualmente a disposizione.

Questo capitolo ha lo scopo di fornire una panoramica generale sulle principali complessità e problematiche che governano questo settore, offrendo al lettore le basi per comprendere meglio il lavoro che si è svolto. Nella sezione 2.1 si fornisce una veloce panoramica sulle basi dell'anatomia del volto umano; nella sezione 2.2 viene trattato e approfondito il concetto di Uncanny Valley e come questa possa essere evitata; nella sezione 2.3 viene presentato il sistema di codifica facciale FACS definendone potenziali e limiti; nella sezione 2.4 si riportano i principi sui quali si basa l'animazione, indispensabili per la realizzazione di animazioni facciali stilizzate; nella sezione 2.5 viene trattata la fase di modellazione 3D ed infine la sezione 2.6 è il punto di partenza per l'introduzione ai capitoli 3, 4 e 5.

2.1 Basi anatomiche

Sebbene non sia scopo di questa tesi la realizzazione di un volto umano fotorealistico, come vedremo meglio nella parte II del documento, è comunque bene introdurne le basi anatomiche. Infatti, avere una visione globale del funzionamento del volto umano risulta indispensabile per avere maggior consapevolezza del problema che si sta affrontando. Queste conoscenze saranno

utili nella determinazione delle tecniche e dei metodi che si andranno ad utilizzare e saranno di enorme aiuto nel guidare l'animatore nella fase di animazione facciale vera e propria.

L'intera sezione, per contenuto e per struttura, fa ampio riferimento a [2, capitolo 3] che a sua volta è stato realizzato facendo riferimenti, a volte diretti, a manuali anatomici come [3]. La seguente sezione è da intendersi come un'introduzione ed una semplificazione dell'anatomia del volto umano. Per il lettore che vuole approfondire tali argomenti, l'elenco completo dei riferimenti è visionabile nella bibliografia.

2.1.1 Il cranio

Il cranio è di particolare interesse per la sintesi facciale perché fornisce la struttura su cui sono posti i muscoli e la pelle, inoltre, fornisce la forma complessiva del viso. Si divide principalmente in due parti: il *neurocranio* ovvero la porzione superiore del cranio, quella che contiene l'encefalo e alcuni dei principali organi di senso, e lo *splancnocranio* (o scheletro della faccia). Lo scheletro del viso è diviso a sua volta in tre parti principali: quella superiore è costituita dalle ossa nasali e dalle orbite oculari, la parte centrale dalla mascella, le cavità nasali e il naso, mentre la parte inferiore dalla mandibola. Le proporzioni tra queste parti vanno considerate nella fase di modellazione. In totale comprende 14 ossa nella parte anteriore del cranio e, ad eccezione di mandibola e vomere, ogni osso è presente in coppia fornendo così simmetria al viso.

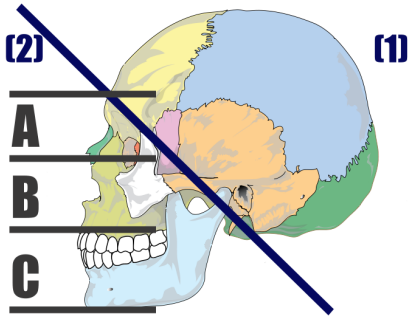


Figura 2.1: Divisione approssimativa del cranio. (1) neurocranio, (2) splancnocranio. (A) la parte superiore composta dalle ossa nasali e dalle orbite oculari, (B) la mascella, e (C) la mandibola.

2.1.2 Muscoli della faccia, testa e collo

La maggioranza dei muscoli umani è associata ad una qualche forma di movimento perché spesso, questi sono legati a due ossa o organi. Tuttavia, alcuni muscoli facciali, in particolare quelli legati alle espressioni facciali, sono da un lato legati ad un osso e dall'altro direttamente alla pelle. Infatti, presentano sempre due estremità, la parte statica chiamata *origine*, e la parte mobile chiamata *inserzione*.

I muscoli della faccia

Oltre alle espressioni facciali, alcuni muscoli intervengono durante la masticazione, deglutizione e nel parlato o durante l'apertura chiusura degli occhi. I muscoli delle espressioni facciali

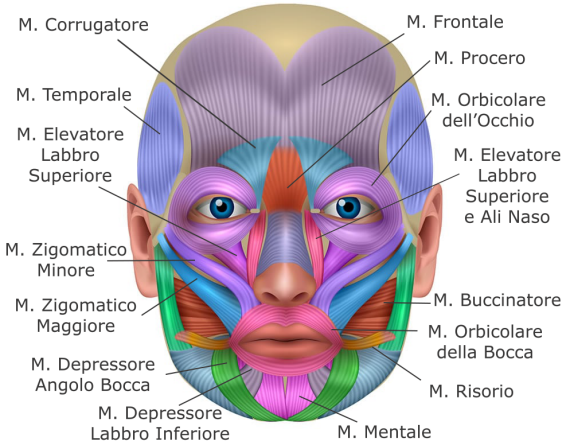


Figura 2.2: Principali muscoli della faccia. Visone frontale del volto.

sono tutti superficiali, ovvero si attaccano, almeno da un lato, allo strato di grasso sottocutaneo e alla pelle. Questi muscoli agiscono in maniera sinergica e non indipendente, per questo motivo è difficile creare dei confini precisi tra i vari muscoli.

Possono essere raggruppati secondo l'orientamento delle singole fibre muscolari e possono essere suddivisi nella parte superiore e inferiore del viso. Si possono distinguere tre tipi di muscoli per i movimenti primari: muscoli *lineari/paralleli* che tirano il punto di inserzione verso il punto di origine, come ad esempio lo zigomatico maggiore; muscoli *circolari/radiali* che si contraggono radialmente verso un centro immaginario come l'orbicolare dell'occhio; e i muscoli *laminari* che si comportano come array di muscoli lineari.

I muscoli facciali sono per convenzione suddivisi in tre grandi gruppi: i muscoli del gruppo orbitale, i muscoli del gruppo nasale e i muscoli del gruppo orale.

Muscoli del gruppo orbitale

Muscolo orbicolare dell'occhio si estende attorno all'orbita oculare e sulla palpebra superiore. Ha origine nella parte nasale dell'osso frontale, le fibre creano un ampio strato che circonda l'orbita dell'occhio.
AZIONE: svolge un importante ruolo nella protezione dell'occhio, permette di chiudere ed aprire le palpebre. La parte orbitale può agire in maniera indipendente, la sua attività genera rughe ai margini esterni dell'occhio. La parte del muscolo relativo alle palpebre permette di gestire un controllo fine, tali azioni possono avere un forte significato nella comunicazione non verbale.

Muscolo corrugatore del sopracciglio è un muscolo molto piccolo che si trova sopra il muscolo orbitale dell'occhio, nel margine mediale dell'arcata sopracciliare.
AZIONE: contraendosi determina uno spostamento in basso e medialmente del sopracciglio. Produce, insieme all'orbicolare dell'occhio, rughe verticali sulla fronte.

Muscoli del gruppo nasale Nell'essere umano tali muscoli sono meno sviluppati rispetto ad altri esseri viventi.

Muscolo procero un muscolo piccolo a forma piramidale, in questo gruppo è quello che si trova più in alto. L'inserzione è sulla pelle situata tra le due sopracciglia.
AZIONE: deprime l'estremità mediale del sopracciglio, producendo rughe trasversali sul ponte nasale.

Muscolo nasale uno dei più grandi di questo gruppo, si trova sui lati del naso. Si divide in due parti, una parte trasversale (muscolo costrittore) ed una alare (dilatazione narici).
AZIONE: queste due parti lavorano per restringere e dilatare le narici.

Muscolo depressore del setto nasale si trova tra il naso e il labbro superiore, l'origine si trova sulla fossa incisiva della mascella, l'inserzione di divide tra setto nasale e parte alare del muscolo nasale.
AZIONE: aiuta la parte alare del muscolo nasale ad allargare l'apertura nasale.

Muscoli del gruppo orale I muscoli del gruppo orale sono numerosi e sono parte importante dei muscoli delle espressioni facciali. Si possono dividere in due gruppi, uno apre le labbra mentre l'altro le chiude.

Muscolo orbicolare della bocca consiste in numerosi strati di fibre muscolari che circondano l'orifizio orale. In parte è costituito da fibre derivate da altri muscoli che convergono sulla bocca. L'origine è tra mascella e mandibola e l'inserzione prende contatti con pelle e mucosa delle labbra.
AZIONE: permette un controllo quasi infinito delle labbra. Viene utilizzato nel parlato e nella comunicazione non verbale, è anche molto utile per la masticazione.

Muscolo buccinatore muscolo sottile e ampio che forma gran parte della guancia (tra mascella e mandibola); ha origine tra i processi alveolari di mascella e mandibola e l'articolazione temporo-mandibolare. L'inserzione si fonde con quella del muscolo orbicolare della bocca.
AZIONE: comprime le guance contro i denti, lavora in sinergia con la lingua in fase di masticazione.

Muscolo elevatore del labbro superiore e dell'ala del naso ha origine nel processo frontale superiore della mascella ed inserzione nella cute sulla parte laterale delle narici e del labbro superiore.
AZIONE: solleva e inverte il labbro superiore e rende più profondo la parte superiore dei solchi naso-labiali; lo scivolamento mediale dilata la narice.

Muscolo elevatore del labbro superiore piatto e largo è diviso in tre parti rispettivamente con origine nell'osso dell'orbita, zigomatico e della mascella. L'inserzione è incorporata nel labbro superiore.
AZIONE: solleva il labbro superiore approfondendo i solchi naso-labiali.

Muscolo zigomatico maggiore ha origine sull'osso zigomatico e inserzione a livello della pelle del labbro superiore.
AZIONE: solleva il labbro superiore, a volte mostrando i denti. Partecipa alla creazione del sorriso.

Muscolo elevatore dell'angolo della bocca più in profondità rispetto ai muscoli zigomatici, l'origine si trova a livello della fossa canina della mascella e l'inserzione si trova sul modiolo delle labbra.
AZIONE: solleva l'angolo della bocca, mostrando i denti e approfondendo il solco naso-labiale.

Muscolo risorio ha origine sulla fascia parotidea e inserzione nel modiolo delle labbra.
AZIONE: agisce sull'angolo della bocca muovendolo lateralmente e verso l'esterno. Concorre alla creazione del sorriso.

Muscolo mentale situato sulla punta del mento, ha origine sulla sezione della mandibola corrispondente al mento e inserzione nel tessuto sottocutaneo del mento.
AZIONE: agisce come elevatore della pelle del mento e induce la protrusione del labbro inferiore.

Muscolo elevatore dell'angolo della bocca chiamato anche muscolo canino, ha origine nella fossa canina della mascella e inserzione sul modiolo delle labbra.
AZIONE: eleva il modiolo della bocca in direzione quasi verticale.

Muscolo depressore dell'angolo della bocca ha origine sul tubercolo della mandibola e inserzione sul modiolo delle labbra.
AZIONE: deprime lateralmente il modiolo e l'angolo della bocca aprendo la bocca, caratteristica dell'espressione della tristezza.

Muscolo depressore del labbro inferiore l'origine si trova sulla linea obliqua della mandibola, l'inserzione è legata alla pelle del labbro inferiore fondendosi con l'inserzione dell'orbicolare della bocca.
AZIONE: tira il labbro inferiore verso il basso e lateralmente, esempio durante la masticazione.

Muscoli della mandibola

I movimenti della mandibola sono complessi, i muscoli in azione lavorano in gruppo per garantire movimenti bilanciati. I movimenti base della mandibola sono: elevazione, depressione, protrazione e ritrazione.

Muscolo massetere muscolo ampio e spesso, ha origine nell'arco zigomatico e inserzione sulla superfice della mandibola.
AZIONE: potente muscolo utilizzato per elevare la mandibola durante la masticazione, lavora con altri muscoli per protrarre e ritrarre la mandibola.

Muscolo pterigoideo interno ed esterno il muscolo pterigoideo interno ha origine nella fossa pterigoideo e inserzione sulla superficie della mandibola. Il muscolo pterigoideo esterno ha due teste, una ha origine sulla superficie infra-temporale dell'osso sfenoidale e l'altra sulla superficie della placca pterigoideo laterale.
AZIONE: agiscono per elevare, protrarre e ritrarre la mandibola. Quando entrambi si contraggono dallo stesso lato, si ha uno spostamento del mento verso quella direzione. Questi movimenti sono importanti per la masticazione.

Muscolo temporale è un grande muscolo che ha origine nella fossa temporale, sulla superfice laterale del cranio, mentre ha inserzione sul processo coronoideo della mandibola.
AZIONE: muscolo meno potente del muscolo massetere ma principale responsabile dell'elevazione della mandibola e in misura minore di una sua ritrazione.

Muscolo digastrico composto da due parti carnose collegate da un tendine.
AZIONE: uno dei principali responsabili dell'abbassamento della mandibola, controlla la posizione della laringe. Quando la mandibola è sollevata, aiuta nella ritrazione della stessa.

Muscolo genioioideo ha origine nella mandibola e concorre alla muscolatura del pavimento della bocca.
AZIONE: agisce per elevare la lingua e deprimere la mandibola.

Muscolo miloioideo forma il pavimento muscolare della bocca.
AZIONE: questo muscolo eleva la lingua e il pavimento della bocca, e in una certa misura deprime la mandibola.

Muscolo platisma è un muscolo grande e largo e copre la maggior parte della porzione anteriore e laterale del collo.
AZIONE: tende la cute del collo e abbassa la mandibola.

Articolazione temporo-mandibolare / ATM o TMJ (Temporo mandibular Join) È l'articolazione che si trova tra mandibola e cranio, svolge la funzione di articolare il movimento complesso della mandibola. Fondamentale per la masticazione e la fonazione. Oltre ai movimenti di elevazione, depressione, protrusione e ritrazione già visti per la mandibola, si considerano anche i movimenti asimmetrici e laterali.

Muscoli della lingua

La lingua è composta interamente da muscoli, nervi e vasi sanguigni. I muscoli sono divisi in due gruppi, muscoli intrinseci che si trovano all'interno della lingua e muscoli estrinseci che hanno origine fuori dalla lingua e sono responsabili della sua posizione all'interno della bocca.

La lingua ha una capacità straordinaria di cambiare forma e posizione dovuto al ruolo che ricopre durante la masticazione, la deglutizione e il parlato. Inoltre, gioca un ruolo fondamentale nel dare forma alla cavità orale durante il parlato[1].

Muscoli della fronte

Tali muscoli non hanno attaccamenti con le ossa ma hanno origine dalla radice del naso e dalla pelle del sopracciglio e inserzione nel cuoio capelluto.

AZIONE: la loro ampia struttura e l'orientamento verticale delle fibre crea rughe orizzontali quando si ha contrazione. La contrazione agisce per sollevare le sopracciglia. L'azione frontale può essere suddivisa in parte interna ed esterna.

2.1.3 Pelle

La pelle copre l'intera superficie esterna del corpo umano ed è un'interfaccia altamente specializzata tra il corpo e l'ambiente circostante. Sebbene sia più o meno simile su tutto il corpo, il suo aspetto e la sua consistenza variano molto a seconda dell'età, dovrebbe essere quindi modellata in modo accurato per trasmettere le sue caratteristiche fisiche. Il tessuto facciale ha spessore variabile, ad esempio intorno agli occhi è sottile mentre intorno alle labbra è più spesso. Anche il colore varia a seconda di molti fattori, in particolare risulta, per un soggetto caucasico, di un colore più rosato dove è sottile e più giallastro dove di spessore maggiore. Ovviamente altri fattori come età, sesso e salute concorrono a determinarne l'aspetto. Le rughe sono dovute in parte alla perdita di elasticità e in parte alla perdita del tessuto grasso entrambe dovute all'invecchiamento. La pelle umana ha una struttura a strati, composta da epidermide, lo strato più esterno composto da cellule morte, dal derma e dal tessuto adiposo o sottocutaneo.

[1] L'animazione della lingua per la serie *Reverie Dawnfall* verrà trattata approfonditamente nel capitolo 10.

Linee della pelle La struttura a strati della pelle non è omogenea né isotropa. Queste caratteristiche furono elaborate prima da Langer nel 1861, tramite osservazione di cadaveri, e poi nel 1951 da Kraissl, definite su individui in vita. Le linee di Kraissl, a differenza di quelle di Langer che si basavano sull'orientamento del collagene, indicano le linee di massima tensione che tipicamente sono ortogonali alla direzione di azione delle fibre del muscolo sottocutaneo. Tali linee descrivono per buona parte le rughe del viso.

2.1.4 Occhi

Le componenti principali dell'occhio sono:

Sclera lo strato più esterno del bulbo oculare, consiste in un denso mantello di collagene.

Cornea membrana sottile, sporgente e trasparente inserita nella fessura circolare al polo anteriore della sclera.

Iride diaframma regolabile attorno all'apertura centrale della pupilla. È il meccanismo che regola automaticamente la quantità di luce che attraversa l'occhio. Il suo colore può variare notevolmente da individuo a individuo. Consente di regolare in maniera volontaria e involontaria la visione da vicino e da lontano.

Retina lo strato più interno del bulbo oculare, è un'espansione del nervo ottico. Per ovvie ragioni non ha rilevanza in fase di modellazione.

Quando si creano modelli digitali di una faccia, bisognerebbe prestare molta attenzione ai dettagli degli occhi. Gli occhi dovrebbero convergere e non dovrebbero fissare il vuoto, la pupilla dovrebbe essere in grado di dilatarsi e dovrebbe essere fatta attenzione al riflesso delle luci sulla superficie dell'iride e della cornea; inoltre l'occhio non è completamente sferico ma la cornea crea una leggera curvatura.

Movimento degli occhi

Gli occhi hanno la capacità di reagire molto rapidamente agli stimoli esterni, è importante comprendere i movimenti principali per poterli applicare nel modo più completo all'interno di un modello digitale.

Vergenza quando gli occhi mettono a fuoco un oggetto muovono simultaneamente entrambi i bulbi, la posizione dell'oggetto a fuoco determina la convergenza.

Saccade movimento volontario che si osserva durante l'esplorazione della scena visiva. Sono movimenti molto rapidi che coinvolgono entrambi gli occhi, si tratta di un movimento molto comune.

Riflesso vestibolo-oculare movimento involontario di tipo compensatorio in seguito al movimento della testa. È finalizzato a mantenere l'immagine sulla retina quando si muove la testa.

Riflesso opto-cinetico movimento involontario dell'occhio, si verifica quando la testa rimane ferma ma gli occhi seguono un oggetto.

2.2 Il problema dell'Uncanny Valley

L'essere umano è stato evolutivamente addestrato a riconoscere la più piccola sfumatura nei volti umani. Questo è un aspetto cruciale della nostra vita sociale, come afferma Daniel

C. Krawczyk: «il riconoscimento facciale è un'abilità fondamentale che si sviluppa presto e supporta le nostre capacità sociali» [4]. Durante le interazioni con altri esseri umani siamo alla costante ricerca di indizi su quello che è il loro stato d'animo e su quali sono le loro intenzioni. Per questo motivo, anche se il più delle volte non ce ne rendiamo conto, il nostro cervello analizza minuziosamente ogni minima variazione nei volti dei nostri interlocutori.

A livello percettivo si può verificare, inoltre, il fenomeno chiamato *pareidolia*[2]; questo fenomeno risulta spesso sfruttato nell'industria cinematografica, e non solo, per la creazione di personaggi, più o meno umanoidi, in grado di generare un forte legame empatico con il pubblico. Alla pareidolia si può ricondurre la facilità con la quale riconosciamo volti che esprimono emozioni in segni estremamente stilizzati quali le emoticon. Si ritiene che questa tendenza sia stata favorita dall'evoluzione, poiché consente di individuare situazioni di pericolo anche in presenza di pochi indizi.

Siccome siamo così ben addestrati nel leggere i volti, risulta estremamente difficile ricreare in maniera sintetica (con l'utilizzo di robotica o tramite CGI) delle facce che sembrino ed agiscano a tutti gli effetti come un volto umano. Si rischia quindi di precipitare in quella si chiama *Uncanny Valley*.

2.2.1 Uncanny Valley

Letteralmente "valle perturbante", è un concetto introdotto per la prima volta nel 1970 da Masahiro Mori [5] per descrivere le reazioni che le persone hanno di fronte a robot umanoidi che cercano di emulare il più possibile l'aspetto e il comportamento umano.

Come è possibile vedere nel grafico in figura 2.3, più un robot assomiglia ad un essere umano nell'aspetto e nel comportamento più risulta interessante, questo però solo fino ad un certo punto. Quando un robot diventa "troppo" antropomorfo ma non abbastanza da essere paragonabile ad un vero essere umano, nello spettatore si genera una strana sensazione che va dal senso di straniamento, alla paura, fino al senso di nausea e di repulsione con conseguente calo di empatia.

È importante notare che l'uncanny valley si riferisce solo a esseri umanoidi o animali e non ad oggetti inanimati. Bisogna, inoltre, precisare che al momento la validità scientifica di questa teoria è ancora dibattuta, tuttavia, come lo stesso Mori afferma, non deve essere interpretata come una rigida legge scientifica ma più come una valida linea guida nel processo di design [6].

Con l'avvento della *CGI* (computer-generated imagery), tale termine è diventato di uso comune all'interno dell'industria cinematografica e videoludica. Sempre di più sono le opere artistiche che inseriscono al loro interno personaggi o creature realizzate utilizzando la Computer Grafica e il rischio di precipitare nell'Uncanny Valley è sempre dietro l'angolo. Prodotti che a livello tecnico risulterebbero delle vere e proprie conquiste, risultano poi creare nello spettatore un senso di straniamento che non permette al pubblico di godersi l'opera.

---

[2] «Processo psichico consistente nella elaborazione fantastica di percezioni reali incomplete, non spiegabile con sentimenti o processi associativi, che porta a immagini illusorie dotate di una nitidezza materiale (per es., l'illusione che si ha, guardando le nuvole, di vedervi montagne coperte di neve, battaglie, ecc.)». Pareidolìa. (Consultato il 25/02/2021). In *Vocabolario on line - Treccani.* URL: https://www.treccani.it/vocabolario/pareidolia/#:~:text=%E2%80%93%20Processo%20psichico%20consistente%20nella%20elaborazione,neve%2C%20battaglie%2C%20ecc.

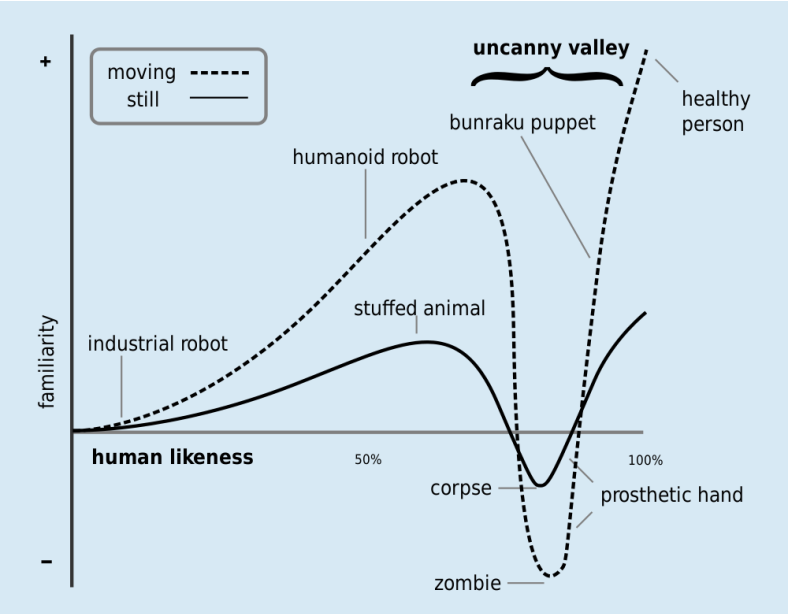


Figura 2.3: L'uncanny valley. Opera modificata tratta da Wikimedia Commons. Licenza Attribution-ShareAlike 3.0.

Nonostante tale fenomeno racchiuda tutti gli aspetti e comportamenti di un personaggio, dalla postura durante il ciclo di camminata al dettaglio dell'epidermide, i problemi si verificano, come accennato precedentemente, principalmente nella rappresentazione dei volti umani.

Si possono verificare situazioni in cui a livello grafico si è in grado di realizzare un volto statico estremamente fotorealistico, quasi indistinguibile da un volto vero, ma nel momento in cui avviene una simulazione di un comportamento (es simulazione di un dialogo), il realismo di tale animazione risulta insufficiente se accostato al realismo grafico. Questo crea una dissonanza tra le aspettative che vengono create dall'elevato fotorealismo e l'effettivo risultato dell'animazione; tale dissonanza è ciò che causa l'effetto di straniamento che fa affondare un prodotto nell'Uncanny Valley.

Anche se lo sviluppo tecnologico in merito avanza molto velocemente, risulta estremamente complesso replicare fedelmente ogni piccola variazione nelle espressioni di un volto umano. Come abbiamo visto nella sezione 2.1, i muscoli che intervengono nel generare una determinata espressione facciale sono molteplici e anche una minima variazione può generare espressioni e significati completamente diversi. Il problema quindi si crea quando al cervello giungono informazioni vaghe, ambigue e contraddittorie.

Siccome l'utilizzo di CGI all'interno di prodotti audiovisivi negli ultimi anni è aumentato notevolmente, aumentano sempre di più gli esempi che mostrano l'effetto dell'Uncanny Valley in azione.

Esempi nella cinematografia

Una delle prime volte in cui l'industria cinematografica è diventata consapevole di cosa fosse l'uncanny valley fu nel 1988 quando venne presentato il cortometraggio di animazione

prodotto dalla Pixar e diretto da John Lassater, *Tin Toy*. Il cortometraggio è considerato un'opera rivoluzionaria per le tecniche innovative che ha introdotto, tra le quali l'introduzione di un personaggio umano creato interamente in digitale. È stato accolto positivamente non solo da ricercatori e ingegneri ma anche dalla critica, tanto da essere il primo cortometraggio di animazione computerizzata a vincere l'Oscar. Tuttavia, arrivarono molte critiche per la resa del personaggio *Billy*, che risultava così poco credibile da creare fastidio e inquietudine. Furono proprio queste recensioni negative che portarono l'industria cinematografica a tenere seriamente in considerazione il concetto di uncanny valley.

Facendo un salto avanti, nel 2001 viene rilasciato sul mercato *Final Fantasy: The Spirit Within*, di Hironobu Sakaguchi. Fu il primo lungometraggio foto-realistico interamente generato in computer grafica e che, con suoi 137 milioni di dollari, ha avuto il primato di film più costoso ispirato ad un videogioco. Il film è composto da 141 964 frame, ciascun frame ha richiesto 90 minuti per essere renderizzato; in totale ci sono voluti quattro anni e 200 membri dello staff per completarlo. Il film è anche riconosciuto per essere uno dei primi prodotti *mainstream* ad essere realizzato con l'utilizzo della *Motion Capture*. Come abbiamo visto per *Tin Toy*, anche in questo caso il lungometraggio fu accolto positivamente per quanto riguarda il lato tecnico e il realismo che è riuscito a raggiungere ma sfortunatamente non ha incontrato l'approvazione del pubblico occidentale, che anzi ha trovato il "falso realismo" dei personaggi fastidioso e inquietante, facendolo precipitare nell'uncanny valley. Il risultato è uno dei maggiori flop commerciali nella storia del cinema.

Sempre nello stesso anno uscì il lungometraggio *The Mummy Returns* diretto da Stephen Sommers. Si tratta di un lungometraggio in *live action* al quale sono stati aggiunti VFX digitali. L'effetto di uncanny valley si può percepire nelle scene in cui è presente il personaggio de *The Scorpion King*, in questo caso il volto dell'attore Dwayne Johnson è stato rimpiazzato da una sua ricostruzione 3D in digitale. Il risultato ottenuto è un esempio perfetto per dimostrare la validità dell'uncanny valley. Si ha un volto umano in CGI che fallisce nel tentativo di raggiungere il foto-realismo massimo. Tale fallimento risulta ancora più evidente in quanto lo spettatore ha un confronto diretto e continuo tra elementi reali e digitali. A peggiorare la situazione è il fatto che si è cercato di riprodurre il volto di Dwayne Johnson (The Rock), attore molto noto e apprezzato dal grande pubblico, e per questo ancora più difficile da "falsificare".

Uno degli esempi più utilizzati quando si parla di uncanny valley è sicuramente *Polar Express* di Robert Zemeckis, film di animazione prodotto nel 2004. Fu il primo cartone animato realizzato in CGI utilizzando la tecnica della performance capture. Anche se a differenza di *Final Fantasy: The Spirit Within* questo film non fece un flop al botteghino, le critiche si concentrarono principalmente sul character design e l'animazione dei personaggi che li hanno fatti precipitare nell'uncanny valley. I movimenti, le espressioni e lo sguardo dei personaggi rendono la visione piuttosto fastidiosa e inquietante. I principali problemi si sono riscontrati nel dover scalare i dati raccolti durante le riprese in motion capture ai modelli 3D dei bambini. Infatti, per l'animazione dei bambini presenti nel film sono stati utilizzati attori adulti, tra cui Tom Hanks; ciò ha introdotto enormi problemi di proporzioni nel momento del *retargeting*, causando nel pubblico quella sensazione di fastidio. Infine, non erano presenti marker per intercettare il movimento della

lingua e degli occhi che quindi sono stati animati manualmente, quando questi key-frame sono stati integrati con i dati catturati tramite *Mocap* si è ottenuto un qualcosa di disturbante.

Nonostante alcune critiche affermavano che *Polar Express* fosse un esperimento fallito, il film ha totalizzato un buon incasso e sicuramente ha introdotto delle innovazioni tecniche nel settore. Questo ha convinto Robert Zemeckis a realizzare ulteriori prodotti con questa tecnica, tra cui *La Leggenda di Beowulf* (2007) e *Christmas Carol* (2009). Purtroppo, a differenza della prima opera l'accoglienza fu più modesta mentre i problemi di credibilità nelle animazioni rimasero. Fu però il lungometraggio animato *Mars Needs Moms* (2011) che portò l'abbandono della tecnica di motion capture da parte del regista. Infatti, fu il peggiore flop ottenuto al botteghino per un prodotto Disney.

Prodotti più recenti che sono entrati più o meno nell'uncanny valley sono *Le avventure di Tintin - Il segreto dell'unicorno* (2011) di Steven Spielberg, *Cats* (2019) di Tom Hooper, *The Lion King* (2019) di Jon Favreau, *Alita - Angelo della battaglia* (2019) di Robert Rodriguez e *Beyond the Aquila Rift* (2019) ottavo episodio della serie animata *Love, Death & Robots* prodotta da Netflix.

Come è possibile evincere dagli esempi sopracitati, la causa principale che fa precipitare un personaggio nell'uncanny valley è la ricerca ossessiva del fotorealismo. Ricerca che si va a scontrare con i limiti tecnici che ancora oggi incidono sulle produzioni cinematografiche. Negli esempi presentati si può notare come la ricerca del fotorealismo risulti controproducente, in quanto da un lato è l'elemento principale che fa lievitare i costi di produzione e dall'altro il suo mancato raggiungimento, a causa dei limiti tecnici e delle limitate risorse economiche a disposizione, crea un prodotto non godibile.

Il senso di fastidio generato da questi personaggi antropomorfi causa un distaccamento empatico tra lo spettatore e il personaggio, al pubblico quindi viene limitata la possibilità di immergersi completamente nel racconto del prodotto e ne viene limitata l'esperienza finale. Questo spesso si ripercuote sugli incassi al botteghino.

Come evitare la fossa

A meno che il nostro intento sia proprio quello di realizzare personaggi che creino disagio, sarebbe buona cosa cercare di evitare di finire nella fossa dell'uncanny, perché questo comporterebbe, come abbiamo visto, perdite economiche e andrebbe ad inficiare il potenziale narrativo dell'opera. Come è stato precedentemente anticipato, i problemi si verificano principalmente quando si vuole raggiungere un elevato grado di realismo. Quando si realizza un prodotto ad alto realismo, caratteristiche non ordinarie come, ad esempio, espressioni umane negli animali (*The Lion King* - 2019) o occhi sproporzionati rispetto al volto (*Alita* - 2019) risultano più fastidiosi che non se fossero applicati a prodotti a basso realismo o a prodotti stilizzati. Quindi *render* fotorealistici non dovrebbero essere accompagnati da aspetti non umani.

Il punto debole nella raffigurazione di esseri umani/umanoidi sono quasi sempre gli occhi. Come dice un vecchio proverbio «gli occhi sono lo specchio dell'anima». È proprio sugli occhi, infatti, che si concentra maggiormente lo sguardo dello spettatore, ed è dagli occhi che si riescono a recepire gran parte delle informazioni relative ad un'espressione facciale. Risulta quindi fondamentale evitare quelli che in gergo vengono chiamati "*dead eyes*", letteralmente

"occhi morti". Un esempio di dead eye lo possiamo trovare nel film *Polar Express*. Nonostante i progressi tecnologici risulta tuttora molto complesso realizzare occhi realistici.

Risulta spesso vantaggioso reindirizzare le risorse, che si utilizzerebbero per creare raffigurazioni accurate di esseri umani ed animali, verso la creazione di personaggi e storie più coinvolgenti ed accattivanti.

Il problema dell'eccessivo realismo dei personaggi può essere evitato ricorrendo alla stilizzazione e all'applicazione del fenomeno chiamato *neotenia* [3].

In pratica consiste nell'inserire in personaggi adulti, caratteristiche infantili per renderli più gradevoli alla vista. Caratteristiche giovanili comunemente utilizzate sono gli occhi grandi e tondi, una testa grande e tondeggiante, pelle morbida, nasi piccoli e così via. Importante precisare che queste caratteristiche non dovrebbero essere applicate a personaggi realistici ma a personaggi a basso realismo e/o stilizzati. Si ritiene che questo senso di gradevolezza che si prova di fronte a caratteristiche infantili sia dovuto al nostro innato istinto protettivo nei confronti di bambini e cuccioli di animali.

Questa tecnica è molto utilizzata nell'industria dell'animazione sia digitale che tradizionale, basti pensare ai personaggi dei manga giapponesi o ai protagonisti dei film Disney. Per capire le potenzialità di queste tecniche basta confrontare i personaggi presenti nei lungometraggi d'animazione digitale *Polar Express* e *Gli Incredibili*, entrambi prodotti nel 2004 (vedi figura 2.4). Risulta evidente come il processo di stilizzazione dei personaggi ne *Gli Incredibili* abbia reso il prodotto estremamente più godibile e per nulla fastidioso.

È quindi possibile evitare di entrare nell'uncanny valley rinunciando ad un eccessivo realismo e accuratezza dei personaggi per privilegiare la stilizzazione e il legame empatico con lo spettatore.

Figura 2.4: Ricerca del fotorealismo vs stilizzazione. A sinistra il *Controllore* (*Polar Express*) ©2004 Warner Bros. Ent. A destra *Mr. Incredibile* (*Gli Incredibili*) ©2004 Disney Enterprises Inc./Pixar Animation Studios.

[3] Il termine neotenia indica «il fenomeno per cui un organismo raggiunge la maturità sessuale conservando i caratteri larvali o giovanili, in conseguenza di fattori ambientali o genetici». Neotenia. (Consultato il 22/02/2021). In *Enciclopedia on line - Treccani*. URL: https://www.treccani.it/enciclopedia/neotenia/.

2.3 Facial Action Coding System (FACS)

Il Facial Action Coding System (FACS) è un sistema completo e anatomico per descrivere tutti i movimenti facciali visivamente distinguibili. È stato presentato per la prima volta nel 1978 da Paul Ekman e Wallace V. Friesen e poi successivamente aggiornato nel 2002 [7].

In questo sistema vengono considerati tutti e i soli i movimenti facciali osservabili, vengono quindi ignorati quei movimenti impercettibili e altri possibili fenomeni non legati al movimento (es flusso sanguigno). Tali movimenti facciali sono ricondotti all'attivazione e all'azione di singoli muscoli o gruppi muscolari. I singoli movimenti muscolari, e le singole posture facciali riconoscibili vengono definiti come *Action Unit* (AU), ovvero unità di azione. Ogni AU si riferisce alla più piccola azione consentita, quindi non è possibile suddividere ulteriormente una action unit. Queste possono essere legate ad un valore di intensità e possono essere utilizzate per studiare tutte le possibili azioni facciali umane.

Nel FACS in totale sono classificate 66 AU [2, capitolo 2] di cui le principali sono 44, ci sono poi 14 azioni che descrivono i cambiamenti nell'orientamento della testa e della direzione dello sguardo, il tutto arricchito da alcune azioni descrittive (AD) per i comportamenti più grossolani.

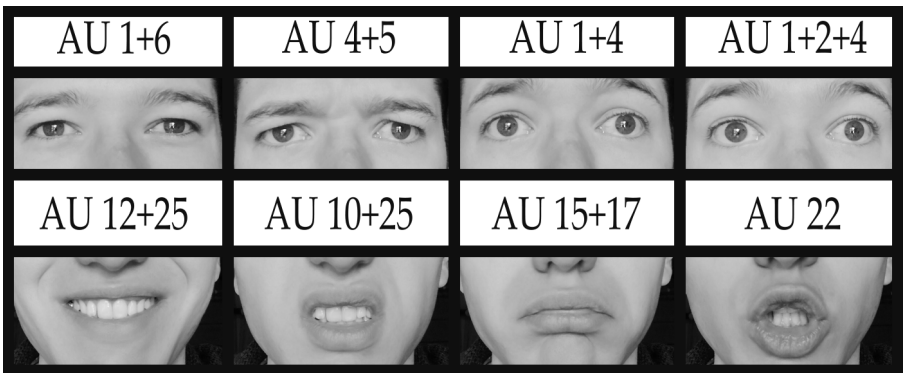


Figura 2.5: Esempio di alcune action unit in azione

Il sistema FACS è stato sviluppato andando a determinare quanto ogni muscolo volontario della faccia fosse in grado di generare cambiamenti visibili, i muscoli involontari non sono stati presi in considerazione. Il rapporto muscoli e action unit però non è uno a uno, infatti diverse AU possono essere caratterizzate dal movimento dello stesso muscolo o dello stesso gruppo di muscoli (es muscoli della fronte che intervengono nel movimento delle sopracciglia). Sebbene il sistema risulti completo e affidabile per le azioni relative alla parte alta del volto (fronte, sopracciglia e palpebre), non include la totalità delle azioni visivamente distinguibili nella parte bassa del volto, in quanto tra il movimento della mascella e la deformazione delle labbra si potrebbero generare un numero estremamente elevato di azioni. Tuttavia, risulta un ottimo compromesso tra affidabilità e semplicità di applicazione.

Il FACS è pensato unicamente per la descrizione dei movimenti facciali, mentre per quanto riguarda l'indicazione su ciò che un movimento può significare bisogna rivolgersi all'Emotion FACS (EMFACS).

Sebbene il FACS non sia stato creato appositamente per essere utilizzato nella computer animation, questo sistema descrittivo è utilizzato ampiamente come base per il controllo delle espressioni facciali nell'animazione 3D. Bisogna però precisare alcuni limiti che incorrono nell'utilizzo di tale sistema come base per l'animazione facciale. Per prima cosa, come accennato precedentemente, il FACS si limita a descrivere i movimenti facciali e non il loro significato; inoltre, non è *time-based* e i movimenti facciali sono definiti unicamente a partire da una posa base. In secondo luogo, il FACS serve a descrivere le espressioni facciali e non il parlato, i movimenti che si vanno a creare durante ogni singolo fonema, unità base del parlato, non sono incorporati nel sistema.

Nonostante questi limiti l'aspetto più interessante del FACS consiste proprio nel fatto che ogni azione facciale viene scomposta in singole unità, ciascuna delle quali può essere considerata indipendente da tutte le altre. Si è in grado, quindi, tramite accurata combinazione, di generare una gamma quasi infinita di espressioni facciali partendo dalle singole AU.

Tutto ciò risulta incredibilmente vantaggioso se applicato all'animazione digitale. Infatti, come vedremo nel capitolo 3, uno dei metodi più utilizzati per l'animazione dei volti in CG consiste nel definire un set di espressioni base (*blendshape*) e di combinarle per ottenere il risultato desiderato.

2.4 I 12 principi dell'animazione

Frank Thomas e Ollie Johnston due dei *Nine Old Men*, ovvero degli animatori storici della Walt Disney, hanno delineato quelli che tutt'ora sono considerati i dodici principi dell'animazione e li hanno pubblicati nel libro *The Illusion of Life: Disney Animation* del 1981 [8]. Questi principi sono ancora oggi alla base della maggior parte dei prodotti di animazione, sia tradizionale (2D) sia computerizzata tantoché Jhon Lasseter, uno degli animatori e registi Pixar più conosciuti e apprezzati, legò nel 1987 i principi dell'animazione tradizionale alle tecniche della computer animation [9].

I dodici principi che verranno descritti di seguito possono essere applicati, come per tutti gli oggetti e forme animabili, anche all'animazione facciale. È tuttavia importante ricordare che in questo contesto si sta parlando di prodotti di animazione e cartoni animati, quindi non di animazioni facciali fotorealistiche.

Squash & Stretch Principio fondamentale per dare fisicità ad un oggetto. Serve per simulare il peso e il volume dei personaggi durante i movimenti, andando a comprimere ed espanderne il corpo, mantenendone però il volume. Utilizzato nelle espressioni facciali per enfatizzare determinate azioni, comporta una deformazione del volto.

Anticipation Nei prodotti di animazione è fondamentale aumentare la comprensibilità di ciò che sta succedendo. Per questo motivo prima del verificarsi di ogni azione lo spettatore dovrebbe essere avvertito e preparato. Il principio di anticipazione si occupa propriamente di questo, consiste nel creare un'azione opposta a quella che sarà l'azione principale. Ad esempio, prima di far spalancare gli occhi al nostro personaggio, questi verranno fatti chiudere, in questo modo si avvisa lo spettatore che sta per succedere qualcosa e il tutto risulta più credibile e naturale.

Staging Bisogna essere chiari nel presentare il soggetto della nostra azione. Se ogni movimento del personaggio ha uno scopo ben definito è importante che lo spettatore non si distragga.

Per focalizzare l'attenzione su ciò che vogliamo si fa utilizzo delle giuste inquadrature dei giusti movimenti di camera e della giusta illuminazione.

Straight Ahead & Pose to Pose Si riferisce a due differenti tecniche utilizzate per creare un'animazione. Straight ahead prevede che l'animatore disegni ogni singola posa in ordine temporale, quindi dal primo frame fino alla fine della sequenza. In questo modo si ottengono animazioni più fluide. Pose to Pose prevede di definire delle pose chiave o keyframe e le pose intermedie vengono estrapolate tramite tecniche di interpolazione. Ad oggi è la tecnica più comunemente utilizzata in computer animation.

Follow Through & Overlapping Action Principio che fa riferimento alla fisica. In un oggetto in movimento alla quale sono collegati degli elementi ma liberi di muoversi in maniera indipendente (braccia, capelli, vestiti...), questi seguiranno il movimento dell'oggetto a cui sono legati ma nel momento in cui l'oggetto termina la sua azione o si dovesse verificare un cambio repentino del movimento, allora le articolazioni continueranno a muoversi seguendo la loro traiettoria per un certo lasso di tempo. Per la buona riuscita di questo principio è fondamentale la gestione delle tempistiche.

Ease-In & Ease-Out Anche in questo caso si tratta di un principio che fa riferimento alla fisica. Ogni oggetto nel modo reale è dotato di una sua massa ed è soggetto ad inerzia. Ciò significa che qualsiasi movimento sarà caratterizzato da una fase di iniziale di accelerazione e una fase finale di decelerazione. Nell'animazione tradizionale tale effetto era dato dal differente numero di frame disegnati, in computer animation tale principio viene soddisfatto tramite il controllo delle curve di interpolazione (IPO).

Arcs Tutti i movimenti naturali tendono a compiere traiettorie ad arco o traiettorie paraboliche e molto raramente delle linee rette. Ad esempio, quando un personaggio ruota la testa questa si abbasserà leggermente durante il movimento descrivendo, in questo modo, un arco.

Secondary Action Ad un'azione principale è bene che seguano delle azioni secondarie, in modo da creare maggiore naturalezza, organicità e vivacità all'animazione. Tuttavia, le azioni secondarie non devono andare a prevalere su quella principale, altrimenti si crea un conflitto per l'attenzione dello spettatore con conseguente riduzione della comprensione di ciò che sta succedendo.

Timing Ogni azione è caratterizzata dal tempo in cui viene compiuta. La corretta gestione delle tempistiche è fondamentale per dare l'illusione che gli oggetti rispettino una fisica. Azioni più veloci denotano una sensazione di urgenza, mentre azioni più lente trasmettono rilassatezza ed armonia. Il timing è fondamentale nell'animazione facciale perché consente di dare il giusto spazio alle emozioni che si vogliono trasmettere.

Exaggeration Spesso è utile esagerare un'azione per renderla più visibile e comprensibile, bisogna però fare attenzione a non minarne la credibilità. Nella realizzazione delle espressioni facciali questo principio è fondamentale, soprattutto in situazioni in cui il personaggio si trova distante dalla camera e quindi il volto risulta poco visibile. In generale, comunque, basare le espressioni facciali unicamente su reference di attori reali potrebbe creare delle azioni meccaniche che stonerebbero su un personaggio animato. Sebbene possa sembrare assurdo, esagerare le espressioni facciali di un personaggio animato in modo accorto permette di creare un'animazione più credibile.

Solid Drawings Un animatore deve sempre tenere in considerazione il volume e il peso di un personaggio, anche quando si tratta di animazione 2D. Con l'animazione 3D questo principio viene automaticamente soddisfatto.

Appeal L'obiettivo finale deve essere quello di dare vita e unicità al soggetto dell'animazione. Tutto ciò che si vede e sente a schermo deve andare a concorrere alla caratterizzazione del personaggio. Quindi il tutto deve essere reso più accattivante e interessante. Sotto questo principio può ricadere anche quello che viene definito *twinning*, ovvero bisogna evitare il più possibile la simmetria, sia nelle pose che nei movimenti. Spesso i modelli 3D per semplicità vengono realizzati simmetrici per risparmiare tempo, per questo motivo l'asimmetria dovrà essere apportata in fase di animazione. Ad esempio, nell'animazione facciale le espressioni realizzate dovrebbero creare un volto parzialmente asimmetrico. Movimenti asimmetrici risultano più naturali, organici e ne aumentano la credibilità.

2.5 Modellazione facciale

Come si è potuto evincere dalla sezione 2.1, il volto umano è un sistema estremamente complesso e replicare in modo completo tale complessità attraverso un modello 3D è un obiettivo che ancora oggi deve essere raggiunto. Fortunatamente, in molti campi, tra cui anche quello dell'animazione computerizzata, è possibile realizzare modelli facciali che siano solo un'approssimazione del comportamento e della struttura anatomica del volto umano, ottenendo comunque risultati credibili.

Nella fase di modellazione, ogni decisione che viene presa andrà ad influenzare il risultato in fase di animazione, per questo la realizzazione del modello facciale è un processo molto delicato che deve tenere in considerazione molti fattori, in particolar modo la topologia facciale.

Uno studio più approfondito relativo alle tecniche e gli strumenti utilizzati per la modellazione di personaggi 3D, ed in particolar modo focalizzato sulla realizzazione dei personaggi per la serie animata *Reverie Dawnfall*, è stato fatto nel 2018 dal collega Michele Cannata [10]. Per questo motivo ci si limiterà a riportare alcuni concetti fondamentali che devono essere presi in considerazione nella realizzazione del modello poligonale del volto.

Oggi giorno la tecnica sicuramente più utilizzata prevede di approssimare il volto umano tramite una superfice poligonale (es in figura 2.6). Anche nei casi in cui il modello iniziale venga realizzato con tecniche differenti, come ad esempio la scultura 3D, il modello finale viene quasi sempre ricondotto ad una superfice poligonale. Per questo motivo comprendere le regole base da seguire nella realizzazione della topologia poligonale risulta fondamentale per una buona animazione. Con il termine *topologia*, in questo contesto, si vuole indicare come i vertici dei poligoni sono connessi tra loro e, quindi, come vanno a comporre la superficie geometrica.

Con una buona ed intelligente topologia, è possibile realizzare un volto in grado di deformarsi in espressioni facciali realistiche, ma se si realizza una topologia incorretta allora tali errori risulteranno evidenti in fase di animazione; si possono ottenere quindi deformazioni non volute e movimenti non organici. Inoltre, una buona topologia poligonale permette anche di ridurre il numero di vertici utilizzati, limitando in questo modo la complessità di calcolo e rendendo più semplice apportare successive modifiche, senza però inficiare sul risultato finale.

Di seguito si riportano alcuni punti chiave, tratti da [2, capitolo 4] da tenere in mente quando si approssima un volto umano con dei poligoni:

- I poligoni devono essere disposti in modo da approssimare la superfice del viso.
- I poligoni devono essere disposti in modo da consentire alla faccia di deformarsi in modo naturale.

- Deve essere possibile utilizzare la stessa topologia poligonale per approssimare tutte le espressioni facciali desiderate. Come vedremo meglio nel capitolo 3, non è consigliato creare *mesh* differenti per le diverse espressioni.
- La densità dei vertici dovrebbe essere distribuita seguendo la curvatura della superfice. Aree come naso, bocca e occhi necessitano maggiore densità di informazioni rispetto ad aree con minore curvatura come ad esempio la fronte, le guance e il collo.
- Utilizzare il minor numero possibile di poligoni per ottenere il risultato desiderato.
- I lati dei poligoni dovrebbero coincidere con le pieghe naturali del viso, come ad esempio nelle fossette ai lati della bocca.
- Se si vogliono realizzare pieghe dove la superfice è liscia è necessario suddividere tale superfice in più poligoni in modo da avere delle normali al piano separate. Questo permetterà di creare effetti di ombreggiatura.
- È bene evitare o limitare al minimo i punti in cui un vertice condivide più di quattro lati.

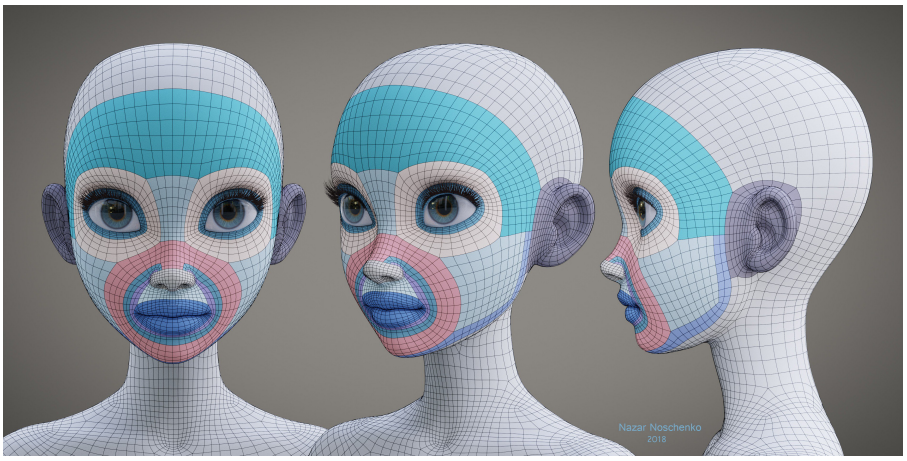


Figura 2.6: Topologia poligonale di un volto. Opera di Nazar Noschenko (2018).

In fase di modellazione realizzare un volto simmetrico risulta estremamente vantaggioso e permette di risparmiare molto tempo, in quanto è sufficiente creare solo una metà del volto e l'altra viene automaticamente creata tramite *mirroring*. Tuttavia, se l'obiettivo è realizzare un modello realistico o replicare il volto di una persona reale, allora, procedere con il mirroring potrebbe non restituire il risultato desiderato, in quanto, anche se con percentuali differenti, tutti i volti umani sono in parte asimmetrici. Per quanto riguarda la realizzazione di personaggi non fotorealistici e stilizzati, procedere tramite simmetria in fase di modellazione è sicuramente vantaggioso, purché l'asimmetria sia data nella realizzazione delle espressioni facciali, perché un volto asimmetrico risulta più naturale ed organico. Sotto quest'ultimo caso rientrano anche i personaggi realizzati per la serie animata *Reverie Dawnfall*.

2.6 Panorama sulle fasi dell'animazione facciale

L'animazione facciale computerizzata può essere divisa in due grandi compartimenti. Il primo gruppo, strettamente legato alla fase di modellazione, racchiude tutte le *tecniche di manipolazione della geometria* che, come si vedrà nel dettaglio nel capitolo 3, sono necessarie per generare sul

volto digitale le espressioni facciali desiderate. Il secondo compartimento, invece, comprende le *tecniche di acquisizione dei dati* per l'animazione facciale [11]. Tali tecniche, che verranno trattate in modo approfondito nel capitolo 5, riguardano tutti quei metodi e quegli strumenti utili per creare i dati relativi all'animazione facciale.

Sebbene questi due gruppi racchiudono operazioni che verranno svolte tipicamente in fasi di lavorazione differenti e quindi possibilmente da professionisti diversi, è necessario che le tecniche adottate consentano alle due fasi di dialogare. Tipicamente, per facilitare la relazione tra le diverse operazioni si fa utilizzo di appositi *rig facciali*. L'operazione di *rigging* verrà trattata nel capitolo 4.

Come presentato in figura 2.7, già in fase di modellazione è necessario individuare ed implementare dei metodi che consentano di deformare come desiderato la geometria del volto. Dopo di che, in una fase successiva si andranno a creare i dati utili per realizzare l'animazione facciale. Tali dati possono essere creati manualmente o acquisiti tramite tecniche più o meno sofisticate (si veda il capitolo 5). Infine, i dati vengono applicati per definire il comportamento delle possibili deformazioni del volto realizzando, quindi, l'animazione facciale vera e propria.

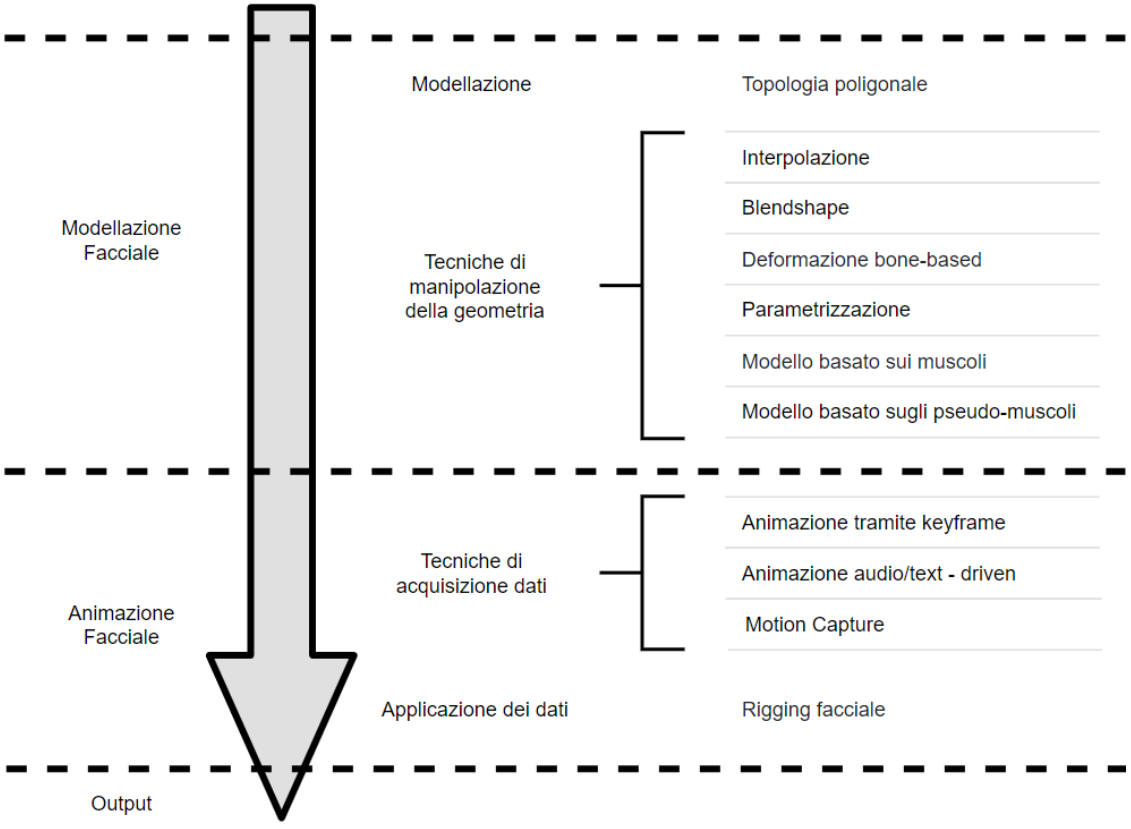


Figura 2.7: Panorama sulle fasi dell'animazione facciale computerizzata

Capitolo 3

Tecniche di manipolazione della geometria

L'obiettivo di tutte queste tecniche è quello di manipolare la superfice poligonale del volto, in modo da creare le espressioni facciali desiderate all'interno di una sequenza temporale. L'industria della computer animation oggi giorno è molto vasta, comprende al suo interno produzioni di videogiochi a bassa ed alta fedeltà, prodotti d'animazione stilizzati, prodotti cinematografici che richiedono estremo realismo e molto altro. Risulta evidente che ognuno di questi campi abbia esigenze differenti e per questo motivo gli approcci utilizzati possono essere molteplici. Le tecniche che sono qui di seguito presentate variano in termini di complessità di realizzazione e in termini di qualità del realismo nel risultato ottenuto.

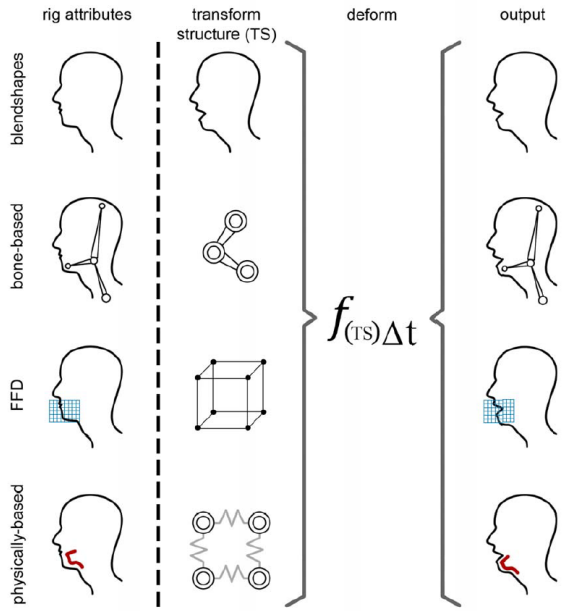


Figura 3.1: Differenti tecniche di deformazione facciale. Figura tratta da [12].

3.1 Interpolazione

Probabilmente è la tecnica più utilizzata per l'animazione facciale, perché offre un approccio semplice ed intuitivo. L'interpolazione è un processo che permette di generare un valore a partire dai valori circostanti. Tipicamente una funzione di interpolazione ha lo scopo di determinare, dati due keyframe, il valore degli in-between frame, ovvero i valori intermedi. Solitamente, per semplicità viene adoperata un'interpolazione lineare, ovviamente tale concetto base può essere applicato ad ogni dimensione. Nel caso ci fossero più di due key frame allora un'interpolazione bilineare è in grado di fornire un risultato migliore rispetto a quella lineare. Data l'enorme semplicità e l'intuitività di questo approccio, le tecniche di interpolazione sono molto utilizzate in computer animation, specialmente nell'animazione facciale.

F. I. Parke [1] ha introdotto per primo tale approccio nell'animazione facciale, nel suo primo esempio erano state definite due espressioni chiave e dopo di che, tramite la manipolazione nel tempo di un singolo parametro, ovvero il coefficiente di interpolazione, si andava a variare l'espressione facciale passando fluidamente tra le due espressioni precedentemente definite. Sebbene l'interpolazione sia una tecnica facile da implementare, la sua capacità di raggiungere elevati gradi di realismo nelle espressioni facciali è fortemente limitata. Un miglioramento si può ottenere dividendo la faccia in regioni indipendenti, tuttavia questo rischia di creare animazioni poco organiche. Nonostante tali limiti, come vedremo nella sezione 3.2, la tecnica di interpolazione è alla base dell'animazione facciale improntata sulle *blendshape* (shape interpolation), una delle tecniche, ancora oggi, più utilizzate nell'animazione digitale.

3.2 Blendshape

Definizione Le blendshape sono dei modelli facciali lineari in cui i vettori di base non sono ortogonali tra loro ma rappresentano le singole espressioni facciali [13]. I vettori di base vengono chiamati *blendshape*, *morph tarhet*, *shape key* o anche semplicemente *shape*. All'interno di questo documento utilizzeremo principalmente il termine blendshape sia per indicare le singole espressioni facciali, sia per indicare la tecnica di deformazione in generale.

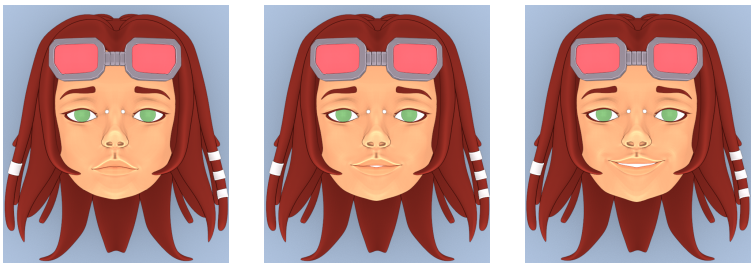


Figura 3.2: Deformazione geometria tramite blendshape. Sinistra: posa neutra o di base; destra: espressione di sorriso specialmente nella parte inferiore del volto; nel mezzo: interpolazione tra le due espressioni.

L'utilizzo delle blendshape è molto diffuso grazie alla loro semplicità di utilizzo e alla loro intuitività, e nonostante i loro limiti, che vedremo più approfonditamente più avanti, sono la tecnica più diffusa e più utilizzata nell'industria cinematografica. Alcuni esempi in cui si sono

utilizzate le blendshape su dei personaggi principali sono *Il curioso caso di Benjamin Button*, *Il signore degli anelli*, *King Kong*, *Final Fantasy: The Spirit Within*, *Star Wars* e *Stuart Little* [13].

Come funzionano La tecnica delle blendshape si basa sull'interpolazione, per questo motivo a volte può essere indicata come *shape interpolation*. Con questo approccio l'espressione facciale finale viene realizzata andando a combinare linearmente un certo numero di blendshape che rappresentano determinate deformazioni (si veda la figura 3.2 per un esempio). Ad esempio, per ricreare un volto sorridente si possono combinare una blendshape che deforma la parte superiore del volto (occhi e sopracciglia) e una blendshape che si occupa di deformare la parte inferiore andando a sollevare gli angoli della bocca per creare un sorriso. In questo modo è possibile creare un numero elevato di combinazioni tra le varie blendshape andando a gestire i vari parametri di influenza, chiamati anche pesi o *slider*.

Secondo quanto riportato da Lewis [13] ci sono due grandi vantaggi nell'utilizzare le blendshape:

- La parametrizzazione utilizzata ha un valore facilmente identificabile: il peso (weight). Questo offre all'animatore una relazione diretta ed intuitiva tra il parametro peso e l'influenza che una blendshape avrà sul risultato finale.
- Aiutano l'animatore a gestire meglio il modello, evitando di creare deformazioni non volute. In questo modo risulta più facile creare animazioni più organiche e credibili.

Sebbene tale tecnica permetta di creare un numero indefinito di espressioni da poter combinare insieme, realizzare un personaggio digitale in grado di coprire una vasta gamma di espressioni facciali consiste in un lavoro molto lungo, in certi casi potrebbe richiedere più di un anno di lavoro svolto da professionisti specializzati [13]. Ad esempio, per il personaggio di *Gollum* nel film *Il Signore degli Anelli* le blendshape presenti nel modello finale hanno raggiunto il numero totale di 946, di cui però la maggior parte sono state introdotte per correggere deformazioni non volute, generate dall'interferenza tra due o più blendshape, mentre le blendshape originali erano 64 [14]. Il problema dell'interferenza verrà trattato meglio in seguito.

Due varianti della tecnica basata sulle blendshape sono l'approccio "*whole-face*" (globale) e l'approccio "*delta*" (locale) [15].

Approccio whole-face l'espressione facciale che si ottiene come risultato può essere vista come una semplice somma vettoriale. In pratica, un modello è formato da n blendshape, ognuna delle quali è composta da un numero di vertici pari a p, infine ogni vertice è rappresentato da tre valori x,y,z. Ogni blendshape, quindi, può essere espressa come un vettore $\mathbf{b}_k$ di lunghezza $3p$, ottenuto andando a ordinare le coordinate dei vertici in maniera consistente tra tutte le blendshape del modello (es xxx...yyy...zzz... o xyzxyzxyz...). L'espressione facciale finale può essere quindi ottenuta con:

$$\mathbf{f} = \sum_{k=0}^{n} w_k \mathbf{b}_k \tag{3.1}$$

o utilizzando la notazione matriciale

$$\mathbf{f} = \mathbf{B}\mathbf{w} \tag{3.2}$$

Dove $\mathbf{f}$ è un vettore $3p \times 1$ contenente l'espressione facciale risultante, $\mathbf{B}$ è una matrice $3p \times n$ contenente le n blendshape, $\mathbf{b}_k$ è la singola blendshape e $\mathbf{w}$ è un vettore $n \times 1$ contenente i pesi (weight) per ciascuna blendshape.

Approccio delta con questo approccio, partendo da quanto visto sopra, si definisce una blendshape $\mathbf{b}_0$ che rappresenti la posa base, tipicamente il volto a riposo, mentre tutte le blendshape $\mathbf{b}_k$ restanti vengono sostituite dalla differenza $\mathbf{b}_k - \mathbf{b}_0$.

$$\mathbf{f} = \mathbf{b}_0 + \sum_{k=1}^{n} w_k(\mathbf{b}_k - \mathbf{b}_0) \tag{3.3}$$

(dove $\mathbf{b}_0$ è l'espressione facciale neutrale). In forma matriciale può essere indicata come

$$\mathbf{f} = \mathbf{b}_0 + \mathbf{B}\mathbf{w} \tag{3.4}$$

Tipicamente i pesi per ciascuna blendshape vengono limitati tra [0,1], ma in alcune DDC (digital content creation suites) come Maya e Blender, si ha possibilità di modificarne il range.

In entrambi i casi, i valori intermedi vengono trovati tramite tecniche di interpolazione lineare. Come è facile intuire, visto che la posizione dei vertici dell'espressione facciale finale è data dalla combinazione pesata delle diverse blendshape, risulta fondamentale che la topologia poligonale, e quindi il numero di vertici e come tali vertici sono collegati tra loro, rimanga invariata. Questo è un fattore che dovrà essere tenuto in considerazione sia nella creazione del rigging facciale sia nella scelta della tecnica di acquisizione dati.

Come accennato precedentemente, l'utilizzo di blendshape non è privo di limiti e criticità. Uno dei maggiori problemi nell'utilizzare le blendshape è dato dalla loro *interferenza* [15]. Questo è dovuto al fatto che le varie blendshape non sono ortogonali tra loro, ciò significa che i parametri (pesi) che le definiscono non sono indipendenti. Può succedere, infatti, che alcune blendshape si sovrappongano creando effetti di competizione o di rinforzo. In pratica la sovrapposizione è dovuta al fatto che un certo numero di vertici risulterà presente nel vettore di due o più blendshape, e quindi la posizione finale dei vertici in questione sarà determinata dall'influenza delle blendshape che li contengono. Tale problema si può limitare definendo delle aree di influenza per ciascuna blendshape (es sopracciglio-destro), tuttavia molte espressioni facciali, per risultare più organiche e credibili, necessitano in alcuni punti di una parziale sovrapposizione di diverse blendshape. Inoltre, se consideriamo che i modelli utilizzati nell'industria cinematografica raggiungono le 100 blendshape, risulta evidente che sia quasi impossibile evitare del tutto le sovrapposizioni. Infatti, come riportato precedentemente, per il modello di *Gollum* la maggior parte delle blendshape create serviva per andare a gestire le diverse interferenze create dall'uso di 64 blendshape originali.

Per ridurre il numero di interferenze è preferibile usare l'approccio locale e suddividere il più possibile in aree specifiche il potenziale di azione di ciascuna blendshape, andando a definire per ciascun vertice il peso che ne indichi il fattore d'influenza, un approccio intuitivo è la tecnica del *weight painting* dove i pesi vengono "pitturati" sui vertici. Uno dei metodi per definire le regioni di influenza si basa sul FACS, come vedremo meglio nel capitolo 9, questo permette anche di avere una parametrizzazione semantica, avendo una relazione quasi uno a uno tra blendshape e azione muscolare, rendendo il processo di animazione più intuitivo. Tuttavia, il solo utilizzo delle blendshape potrebbe dare risultati non soddisfacenti, per questo motivo non è raro utilizzare

le blendshape come base per poi integrarle con metodi più sofisticati, come quelli descritti di seguito.

3.3 Deformazione Bone-Based

Tale tecnica, chiamata anche *skeletal animation*, è stata alla base della definizione del processo di *rigginig*. Nata originariamente per andare a gestire l'animazione dei movimenti del corpo di personaggi 3D, rimane tuttora una delle tecniche più utilizzate per questo scopo, ma oggi è ampiamente adottata anche per andare a determinare le deformazioni dei volti digitali. Tale approccio si basa sul creare un modello gerarchico di corpi articolati, detti ossa (*bones*), chiamato scheletro o armatura, da mettere in relazione con la *mesh* che si vuole deformare. Tale fase viene chiamata *skinning* [12]. Il modello gerarchico può avere differenti gradi di complessità in base alle esigenze, ad esempio andare a gestire la deformazione di una mano richiede maggiore complessità e un numero maggiore di ossa rispetto a gestire il movimento di una gamba.

Spostando l'attenzione dal corpo al volto umano, ci si rende conto che il meccanismo da adottare dovrà essere differente, in quanto, a differenza del corpo, le espressioni facciali sono in piccolissima parte definite dal movimento di ossa, come l'apertura della mascella, ma sono per la maggior parte dei casi determinate dall'azione dei muscoli sottocutanei (vedi capitolo 2.1). Per questo motivo definire una gerarchia di ossa per la faccia risulta un compito complicato. Ciò che tendenzialmente viene fatto, è posizionare le ossa in modo che vadano a simulare il comportamento di un determinato muscolo o gruppo muscolare; per individuare correttamente le regioni di influenza si fa spesso riferimento al FACS.

Ogni osso ha potenzialmente 6 gradi di libertà (DOF, degree of freedom), perciò, dopo aver definito i punti in cui posizionare le ossa, è necessario anche andare eventualmente ad inserire dei vincoli sui movimenti che un determinato osso può compiere, in modo da prevenire che in fase di animazione vengano effettuate deformazioni non volute. Dopo aver definito la struttura gerarchica, la posizione delle ossa e averne definito i DOF, è possibile procedere con lo *skinning*. In questa fase è necessario definire il range di influenza di ogni osso, ovvero bisogna indicare quali vertici della mesh verranno influenzati dal comportamento di un determinato osso e andarne a definire anche il valore di influenza, chiamato peso. Più il peso sarà elevato e più la posizione di un vertice sarà influenzata dall'azione di un determinato osso. Il processo di assegnazione dei pesi viene chiamato *weight painting*.

La skeletal animation, accompagnata con una buona interfaccia utente, consente all'animatore di lavorare in modo intuitivo e semplice e di avere un controllo fine sulle deformazioni applicate. Per questo motivo è una tecnica molto utilizzata, e integrata con l'utilizzo delle blendshape consente di creare espressioni facciali complesse e credibili.

3.4 Parametrizzazione

La tecnica di parametrizzazione nell'animazione facciale è stata introdotta per andare a superare i limiti presentati con la tecnica di semplice interpolazione. L'obiettivo è quello di creare un modello facciale in grado di generare un'ampia gamma di espressioni grazie al controllo di un numero limitato di parametri [2, capitolo 5]. Il vantaggio introdotto è che tale tecnica permette di avere un controllo esplicito su una specifica configurazione facciale.

Il modello ideale (detto *universale*) di parametrizzazione dovrebbe consentire di generare qualsiasi espressione e ogni possibile volto, semplicemente andando a definire un set specifico di parametri [2, capitolo 7], tuttavia ad oggi tale modello universale non è ancora stato raggiunto. Infatti, anche in questo caso, si presentano alcuni limiti. Il primo è dovuto al fatto che la scelta dei parametri è strettamente correlato alla topologia poligonale della faccia e, quindi, una parametrizzazione globale non è possibile. Un altro limite importante da considerare è dovuto al fatto che non è possibile avere una soluzione arbitraria in caso di conflitto venutosi a creare dall'azione contemporanea di due parametri che vanno ad agire sugli stessi vertici. Quando si viene a verificare un conflitto di questo genere, il risultato finale viene compromesso a causa di possibili distorsioni indesiderate e movimenti innaturali.

Per ovviare a questo problema si può organizzare il modello in modo che ogni parametro vada ad influenzare una specifica area della mesh, questo però determina la comparsa di una visibile divisione netta dei movimenti lungo i confini delle regioni interessate. Infine, la realizzazione di un'animazione realistica necessita di andare a settare manualmente ogni singolo valore dei parametri coinvolti. La presenza di questi limiti ha portato a sviluppare nuovi metodi come il modello basato sulla fisica e il modello basato sugli pseudo muscoli.

3.5 Modello basato sui muscoli (o basato sulla fisica)

Questo approccio si basa principalmente su come la forza dei muscoli facciali venga propagata e quindi come agisca nella deformazione del volto. Spesso tali modelli, per il riferimento anatomico del volto, si basano sul FACS. Il modello basato sulla fisica si può dividere in tre categorie: *spring mesh muscle* (muscolo a maglia elastica), *vector muscle* (muscolo vettoriale) e *layered spring mesh muscle* (muscolo a maglia elastica stratificata) [16].

Spring Mesh Muscle Presentato da Platt nel 1985 [17], le forze che vengono applicate alla mesh elastica tramite archi muscolari sono in grado di generare espressioni facciali realistiche. I muscoli vengono suddivisi in blocchi funzionali in aree specifiche della struttura facciale. I vari blocchi, che nel modello definito da Platt sono 38, sono interconnessi grazie a una rete di molle. Le action unit vengono generate applicando una forza muscolare che a sua volta comporta una deformazione della rete di molle.

Vector Muscle Un modello di ampio successo è stato quello proposto da Waters nel 1987 [18]. Questo approccio utilizza i campi di moto in specifiche aree di influenza. Per ogni muscolo è necessario definire la direzione del vettore, l'origine e il punto di inserzione. Il limite di questo modello è che ogni muscolo deve essere accuratamente posizionato in modo da rispettare l'anatomia umana, inoltre non vi è un sistema per automatizzare tale procedimento e un posizionamento errato può causare un'animazione innaturale e può determinare la comparsa di deformazioni non volute.

Layered Spring Mesh Muscle Modello proposto da Terzopoulos e Waters nel 1990 [19], consiste in una struttura a tre livelli di mesh deformabile, corrispondenti a pelle, tessuto adiposo e muscoli connessi a loro volta alle ossa. Un sistema di molle elastiche collega ogni livello e ogni punto della mesh. Permette di raggiungere un elevato realismo, tuttavia richiede parecchie risorse computazionali e richiede molto lavoro per essere applicato ad ogni specifico personaggio.

3.6 Modello basato sugli pseudo-muscoli

I modelli basati sui muscoli, come si è visto, sono in grado di creare animazioni realistiche andando ad approssimare l'anatomia umana, ma definire tutti i parametri necessari per simulare uno volto umano specifico può richiedere enormi risorse in termini di tempo. Gli pseudo muscoli consentono di deformare la mesh facciale andando a simulare i muscoli reali, senza però considerare la complessa struttura anatomica sottostante. La deformazione tipicamente viene applicata solo allo strato superficiale del volto, ovvero la pelle, mentre la forza muscolare viene simulata tramite l'utilizzo di *splines*, *wires* e *Free Form Deformation* [20].

Abstract muscle action (AMA) Con questa tecnica si hanno una serie di parametri che permettono di controllare le procedure AMA. Le procedure AMA sono simili, ma non uguali, alle *action unit* definite dal FACS. Lavorano su un'area specifica del volto e ogni procedura approssima l'azione di un singolo muscolo o di un gruppo di essi. Le espressioni facciali vengono realizzate andando a controllare un gruppo di procedure AMA.

Free Form Deformation (FFD) È una delle tecniche più popolari ed utilizzate, permette di deformare oggetti volumetrici andando a manipolare dei punti di controllo che fanno parte di una griglia tridimensionale (3D lattice). In pratica si va a stabilire un sistema di riferimento locale, ovvero una griglia di coordinate, che vada a mettersi in relazione con l'oggetto che si vuole deformare, sia nella sua interezza ma anche solo per una specifica porzione. Dopo aver legato la porzione di oggetto con la griglia di coordinate, per deformare l'oggetto non si agirà più sui vertici dell'oggetto stesso, ma si andrà a manipolare la griglia di coordinate. In questo modo, i vertici dell'oggetto deformato verranno mappati nuovamente nel sistema globale. L'enorme vantaggio introdotto consiste che l'animatore non si troverà più a dover manipolare direttamente i vertici dell'oggetto, che nella maggior parte dei casi possono contare un numero elevato di elementi, ma potrà controllare un numero decisamente più limitato di punti di controllo (vertici della griglia di coordinate). Tuttavia, le deformazioni possibili sono limitate a quelle che possono essere applicate alla griglia locale. La mappatura tra i due sistemi di riferimento, quello dell'oggetto da deformare e quello della griglia di coordinate, viene effettuata grazie al prodotto tensoriale trivariato del polinomio di Bernstein. Le FFD possono deformare molti tipi di primitive, come superfici poligonali, quadratiche, parametriche ed implicite; e modelli solidi.

Le *Extended Free Form Deformation* (EFFD) permettono di creare delle griglie a forma cilindrica, che consentono una maggiore flessibilità nella deformazione rispetto a quelle cubiche.

Le *Rational Free Form Deformation* (RFFD) sono un'estensione ulteriore delle FFD, in pratica ogni punto di controllo ha incorporato un parametro che ne specifica il peso, aggiungendo così maggiore possibilità di controllo e flessibilità. Nel caso in cui i pesi di tutti i punti di controllo fossero equalizzati, allora la RFFD diventa una FFD.

Le FFD (EFFD, RFFD) hanno l'enorme vantaggio di aumentare il livello di astrazione nel controllo della deformazione, in quanto la manipolazione dei punti di controllo non è più strettamente dipendente dalla superfice specifica che si vuole deformare. Tuttavia, questo approccio non permette un controllo dettagliato sulla creazione di rughe e pieghe del volto e non consente una simulazione accurata del comportamento dei muscoli.

Kalra et al. [21] hanno presentato una tecnica per simulare l'azione dei muscoli utilizzando le RFFD suddivise in regioni specifiche del volto, basate su aree anatomiche. Per ogni regione, viene definita una griglia di controllo, quindi l'estensione e la compressione della pelle in una

specifica regione di interesse è simulata attraverso la manipolazione dei punti di controllo e dei rispettivi pesi in una specifica griglia. Sebbene il risultato non sia estremamente realistico, andare a gestire i punti di controllo risulta molto più semplice e intuitivo rispetto manipolare dei muscoli vettoriali.

Muscoli spline Sebbene i modelli poligonali siano molto utilizzati, spesso hanno delle limitazioni nel mantenere una superficie levigata e flessibile; una superficie piana determinata da un numero minimo di vertici non può essere convertita in una superfice curva se non attraverso una tecnica di *sub-division*. Per andare incontro a questi problemi si possono utilizzare i muscoli spline; le superfici basate sulle spline garantiscono un elevato ordine di continuità, tipicamente sopra il C2. Ovvero, nel punto di congiunzione tra due curve si ha uguaglianza di derivata prima e derivata seconda. Inoltre, le trasformazioni affini sono determinate da un piccolo insieme di punti di controllo e non da tutti i vertici della mesh, introducendo così un vantaggio a livello computazionale. Esempi di curve utilizzate sono le Catmull-Rom spline o spline cardinali e le B-Spline. Per ridurre ulteriormente il numero dei punti di controllo si può utilizzare un modello gerarchico delle spline; l'integrazione di spline organizzate a livello gerarchico con una deformazione basata sulla simulazione dei muscoli è in grado di generare un'ampia gamma di espressioni facciali [16].

Capitolo 4

Rigging facciale

Il compito dell'animatore è quello di dar vita ad un modello 3D andando ad agire su specifici parametri. Per questo motivo è fondamentale fornire all'animatore un set di strumenti che rendano l'intero processo intuitivo e il più semplice possibile. Il rigging facciale è il processo che porta alla creazione di strumenti per il controllo dell'animazione facciale e della loro interfaccia, consente quindi all'animatore di poter lavorare con un maggior livello di astrazione. A formare un rig facciale intervengono tre componenti principali: i metodi di deformazione della geometria descritti nel capitolo 3, l'interfaccia utente (UI) e il modello 3D (mesh) [22]. Il processo di rigging «è un processo iterativo che richiede artisti di esperienza» [12], infatti è necessario considerare moltissimi fattori, prima di tutti la morfologia e topologia della mesh sulla quale viene basato il rig. Poi è necessario avere già definito quale o quali tecniche di deformazione si vogliono implementare, infatti ogni tecnica ha le sue caratteristiche e quindi anche l'interfaccia che verrà utilizzata potrebbe variare da caso a caso. Inoltre, a determinare come un rig deve essere formato e quello che sarà il suo comportamento, intervengono anche le metodologie con le quali i dati dell'animazione vengono acquisiti e applicati al rig stesso. Le principali tecniche di acquisizione e applicazione dati verranno descritte nel capitolo 5. Il rig, quindi, si può considerare come uno strumento di passaggio tra i dati di ingresso relativi alle deformazioni che si vogliono applicare al modello, e l'output finale, ovvero l'animazione stessa. Dato l'alto livello di complessità, nelle produzioni di medie/grandi dimensioni, il lavoro di rigging dei personaggi viene svolto da professionisti specializzati chiamati *rigger* [12].

4.1 Interfaccia Utente (UI)

L'interfaccia utente è un insieme di strumenti e controlli messi a disposizione dell'utente per rendere l'animazione facciale più immediata ed intuitiva; con questo scopo a volte torna utile legare più deformazioni con un unico parametro di controllo [22]. Le UI possono variare molto tra di loro, ma è possibile individuare due tipologie principali che spesso vengono combinate insieme: le UI *window-based* e le UI *viewport 3D* [12].

UI window-based offrono all'animatore un set di controlli in cui è possibile andare ad inserire direttamente il valore necessario. Possono essere di diversa natura ma tipicamente si presentano come un insieme di slider e bottoni. Si chiamano window-based in quanto tali

slider sono posizionati all'interno di apposite finestre, tipicamente fornite dal software di animazione stesso.

UI viewport 3D tali interfacce fanno utilizzo degli oggetti 3D, messi a disposizione dal programma di animazione, con funzione di controller; ovvero il loro movimento o rotazione su un determinato asse è legato al controllo di un determinato parametro. In questo modo l'animatore può agire sul rig direttamente nello stesso spazio di lavoro nel quale si verifica l'animazione. Ci sono due tipologie, la versione 2D fa utilizzo di oggetti planari organizzati su uno stesso piano, solitamente circoscritti da un contenitore rettangolare, e tipicamente vengono disposti a lato del modello che si vuole animare. La versione 3D, invece, consiste nell'utilizzare oggetti 2D o 3D, come ad esempio cubi, sfere o ossa; tali controller vengono messi in relazione con il modello del volto. Una soluzione tipica è posizionare i controller direttamente sull'area della mesh che andranno a deformare.

Attraverso il rigging è quindi possibile creare un'apposita interfaccia, che vada ad elevare il livello di astrazione, ponendosi da tramite tra l'applicazione dei dati di input e il risultato finale, semplificando notevolmente il compito dell'animatore. Solitamente il rigging facciale comprende più metodi di deformazione della geometria in modo da avere una possibilità di controllo maggiore. Una soluzione molto adottata consiste nel combinare la deformazione bone-based con l'utilizzo di blendshape e di FFD, in modo da integrare la flessibilità offerta dalla skeletal animation con l'espressività delle blendshape [12].

Come si vedrà meglio nella parte II di questo documento, per l'animazione facciale dei personaggi di *Reverie Dawnfall* è previsto un sistema di rig facciali che fa uso principalmente di blendshape e di deformazione bone-based.

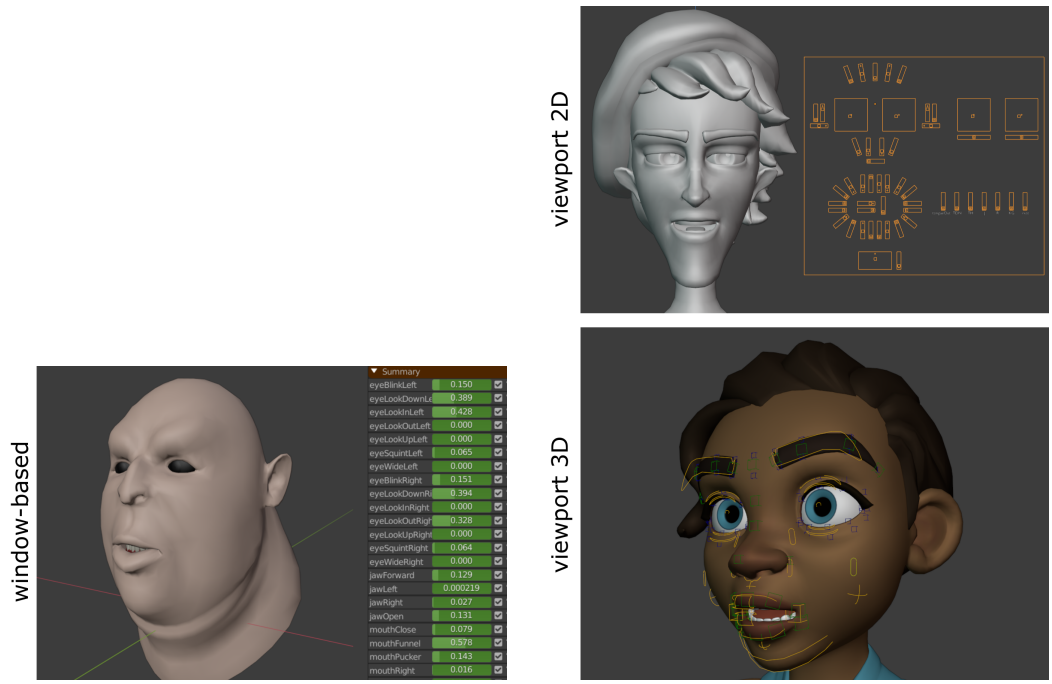


Figura 4.1: Esempi delle diverse tipologie di interfacce per il rigging facciale

Capitolo 5

Tecniche di acquisizione dati

Dopo aver definito, nel capitolo 3, le tecniche che consentono la manipolazione e la deformazione del volto per andare a creare le diverse espressioni facciali, è necessario individuare quelle che sono le tecniche utilizzate per definire la deformazione che si vuole ottenere. Come visto con le tecniche di manipolazione geometrica, anche in questo caso le diverse tecniche riportate di seguito presentano delle caratteristiche che le rendono più adatte in certi contesti produttivi piuttosto che in altri.

Prima di tutto è necessario definire cosa si intende con il termine *keyframe*, in italiano fotogramma chiave. I fotogrammi chiave sono alla base dell'animazione computerizzata, indicano quei fotogrammi che vengono disegnati singolarmente e che rappresentano quelle che in animazione vengono chiamate pose chiave. Dopo di che il software di animazione andrà a definire i valori intermedi, compresi tra due keyframe (*in-between frame*), tramite delle curve di interpolazione (IPO). Come vedremo in seguito, la quasi totalità delle tecniche utilizzate nella computer animation fa utilizzo di keyframe.

5.1 Animazione tramite keyframe

Si tratta sicuramente della tecnica più diffusa in computer animation, come abbiamo visto i keyframe sono l'elemento base su cui si basa l'animazione digitale e questa tecnica ne prevede un controllo diretto. La manipolazione diretta dei fotogrammi chiave permette il massimo controllo sull'animazione, tuttavia la realizzazione di un'animazione credibile consiste in un processo manuale che richiede grandi quantità di tempo e animatori con elevate competenze, «approssimativamente, un'animazione professionale richiede un keyframe all'incirca ogni tre fotogrammi» [13].

Trattandosi di una manipolazione diretta, è una tecnica che prevede un basso grado di automazione. Tale approccio può essere applicato a ciascun *AVAR* (Animation Variable), ovvero a tutti quegli attributi o parametri collegati al soggetto dell'animazione. Per quanto riguarda l'animazione facciale è possibile applicare un'animazione al rigging facciale che a sua volta è in grado di controllare la deformazione del volto tramite blendshape e altre tecniche viste nel capitolo 3 ma anche andare a manipolare qualsiasi altro parametro presente nell'oggetto che si va ad animare (es il colore).

Sebbene, come vedremo per le tecniche successive, esistano metodi che permettono una sostanziale automazione del processo di animazione e quindi un cospicuo risparmio di tempo, l'animazione "tradizionale" tramite keyframe è ancora molto utilizzata e spesso preferita, perché consente un controllo fine sull'intero processo di animazione e quindi permette di creare prodotti di elevata qualità e personaggi con grande espressività. Infatti, se consideriamo i principali studi di animazione 3D come Pixar, Disney, DreamWorks e altri, la quasi totalità dei personaggi comparsi su schermo, è stata animata manualmente tramite keyframe. Questo consente di realizzare animazioni facciali estremamente espressive, visto anche che trattandosi, il più delle volte, di personaggi stilizzati o cartoni animati, l'esagerazione dei movimenti e delle espressioni è appositamente ricercata (vedi capitolo 2.4).

Tale enfatizzazione ed esagerazione dell'azioni spesso risulta limitata quando si procede con tecniche di acquisizione dati che vedremo in seguito, come il motion capture e l'animazione guidata dall'audio (audio-driven).

5.1.1 Animazione facciale nei film d'animazione

La sezione che segue ha origine su quanto ho appreso frequentando il workshop *Facial animation for feature animated film* di Victor Navone [23]. Tuttavia, non è da intendersi come una mera riproposizione del contenuto del workshop; si tratta invece di una linea guida su quelli che sono gli elementi cardine nella creazione di espressioni facciali stilizzate. Ci tengo a precisare che ciò che verrà presentato di seguito è solo uno dei tanti metodi che possono essere adottati quando ci si approccia all'animazione facciale, tuttavia ritengo sia ottimo per comprendere le meccaniche e la complessità che si celano dietro questo mestiere. Il lettore interessato a questo argomento può trovare il workshop a cui faccio riferimento nella bibliografia.

Victor Navone, Supervising Animator per lo studio di animazione Pixar, è entrato a lavorare in Pixar nel 2000 e, a partire da *Monster & Co.*, ha partecipato come animatore alla realizzazione di quasi tutti i lungometraggi Pixar, tra i quali ricordiamo *Alla ricerca di Nemo*, *Cars*, *Inside Out* e *Toy Story 4*, e alla produzione di diversi cortometraggi animati. Vincitore nel 2008 del "Visual Effects Society award for Outstanding Animated Character in an Animated Motion Picture" per il lungometraggio *WALL-E* [24, 25].

La realizzazione dell'animazione facciale in prodotti di animazione computerizzata "stile Pixar" può essere suddivisa in differenti fasi, come riportato di seguito, e si fonda sui 12 principi dell'animazione visti nel capitolo 2.4.

Studio

Come prima cosa è necessario prendere confidenza con il personaggio che si andrà ad animare. È una fase necessaria anche per animatori esperti in quanto ogni modello 3D ha le sue caratteristiche, i suoi vincoli e un suo specifico rig; il rigging nelle produzioni di grandi dimensioni viene effettuato da professionisti appositi. In questa fase l'animatore procede a testare il rig per verificarne i limiti e le potenzialità e per prendere confidenza con lo strumento. Ovviamente lavorare con modelli simili permette una più facile transizione tra un personaggio e l'altro, per questo motivo spesso si cerca di utilizzare il più possibile delle strutture standard.

Pianificazione delle inquadrature

Prima di procedere con l'animazione vera e propria è necessario stabilire come una scena dovrà svolgersi, quali inquadrature utilizzare e l'intenzione che si vuole trasmettere. Fondamentale è avere a disposizione delle reference. Tipicamente è un procedimento a sé stante che a sua volta si divide in fasi, partendo dalla sceneggiatura, passando dallo storyboard per poi arrivare alla realizzazione di un animatic. Spesso si vanno a realizzare anche dei video di riferimento per avere un'idea più completa sui movimenti che dovrebbe compiere un personaggio. Come vedremo successivamente, è fondamentale avere a disposizione la traccia audio definitiva che verrà utilizzata come base su cui costruire l'intera animazione. Infatti, oltre ovviamente a realizzare la corretta sincronizzazione labiale, dalla traccia audio, grazie al tono di voce e alle pause del discorso, è possibile estrarre importanti informazioni sull'intenzione e sull'umore di fondo generale che si vuole dare all'animazione. È pertanto possibile andare a sviluppare l'intera animazione facciale ma anche i movimenti del corpo. Grazie a tutte queste informazioni si è in grado di definire le inquadrature finali. Avere in mente in modo chiaro le inquadrature permette di ottimizzare il lavoro, in quanto inquadrature a campo largo hanno esigenze differenti da dei primi piani. Nel primo caso si potrà essere meno precisi sui dettagli dell'espressione facciale e sui micromovimenti del volto per dare maggior risalto ai macro-movimenti, andando spesso ad enfatizzarli ed accentuarli, ben al di là di movimenti realistici, per renderli maggiormente comprensibili dallo spettatore. Nel secondo caso invece sarà necessario un lavoro maggiore sui dettagli e sulla gestione delle espressioni fini. Inoltre, la definizione delle inquadrature permette di concentrarsi principalmente su quello che si vedrà dal punto di vista della camera e, quindi, su quello che sarà il risultato finale, senza doversi preoccupare di non creare strane deformazioni nella parte di volto non visibile.

Blocking della faccia

La fase di *blocking* serve per impostare le inquadrature e i movimenti di camera definiti precedentemente e creare i movimenti approssimativi del corpo e della testa. Anche se si sta animando la faccia, è utile introdurre dei movimenti generali per il corpo, anche se non inquadrato, in quanto aiuta a dare maggior naturalezza all'animazione globale. Dopo aver creato il movimento generale per il corpo e per la testa, si procede con il blocking degli occhi, sia per quanto riguarda la direzione dello sguardo sia per quanto riguarda la chiusura delle palpebre (*blink*). I blink spesso si verificano tra i maggiori cambiamenti di attitudine e di movimento, ma anche quando si ha un cambio di direzionalità dello sguardo. In questo caso la chiusura delle palpebre svolge anche i compiti di *staging* e di *anticipazione* visti nel capitolo 2.4. Per gestire al meglio questa fase risulta utile fare riferimento al materiale video di reference introdotto precedentemente, infatti non solo ogni personaggio ha le sue peculiarità, il suo stile e i suoi *tic* ma le tempistiche nell'animazione dello stesso personaggio possono variare molto all'interno di una stessa scena.

Movimento degli occhi Gli occhi hanno tipicamente un movimento balistico, ciò significa che si muovono in modo molto veloce e preciso, non si ha quasi mai la fase di accelerazione/decelerazione e la fase di assestamento. Inoltre, a differenza della maggior parte degli oggetti in natura, le pupille raramente si muovono su archi ma si muovono con linee rette.

Sbattimento delle palpebre (blink) Le palpebre superiori sono quelle che percorrono la maggior parte del percorso, mentre quelle inferiori tipicamente hanno un movimento solo leggermente accennato. Per quanto riguarda le tempistiche da adottare, in generale si possono considerare: due frame per la chiusura, due frame per il mantenimento della chiusura delle palpebre e ulteriori otto frame per la loro apertura. Ovviamente si tratta di indicazioni generali che vanno adattate al singolo caso specifico. Introdurre un leggero *squash & stretch* agli occhi nel momento della chiusura serve ad enfatizzarne l'azione, inoltre introdurre un leggero ritardo nell'apertura della parte interna crea una maggiore organicità all'animazione. Nel caso si tratti di blink involontari conviene evitare di muovere anche le sopracciglia, cosa che al contrario va attuata nel caso la chiusura degli occhi fosse un'azione volontaria del personaggio. Infatti, in questo caso, un movimento coordinato delle sopracciglia permette di enfatizzare l'espressione e l'intenzionalità del personaggio.

Espressioni facciali Risulta utile posizionare i cambiamenti delle espressioni facciali prima di un movimento del corpo o della testa, così da creare un'anticipazione al movimento stesso. Per rendere tutto più interessante è bene ragionare anche in base al contrasto che si andrà a verificare tra due espressioni consecutive. In questa fase si vanno a definire delle pose base per sopracciglia, bocca, denti e naso andando ad utilizzare come base le posizioni della chiusura delle palpebre precedentemente create. Per le sopracciglia bisogna ricordarsi che il loro movimento non consiste solo nell'andare verso l'alto o il basso ma hanno anche un movimento convergente verso il centro del ponte nasale, inoltre il movimento delle sopracciglia può comportare un movimento, seppur leggero, delle palpebre superiori. La posizione delle sopracciglia è utile per accentuare la direzionalità dello sguardo, per questo motivo spesso vengono posizionate ad altezze differenti. Per quanto riguarda i denti, è opportuno evitare delle situazioni in cui sono parzialmente visibili, quindi o si mostrano in modo chiaro o si nascondono con l'utilizzo delle labbra, questo per evitare di distrarre lo spettatore.

Per la fase di blocking è consigliato inserire i keyframe necessari tutti negli stessi fotogrammi così da avere uno spazio di lavoro ordinato e pulito, rendendolo più facilmente manipolabile nelle fasi successive.

Animazione della parte superiore del volto

Sopracciglia Sebbene i movimenti e le posizioni che le sopracciglia possono assumere siano molti, il meccanismo che li genera è piuttosto semplice. Come visto nel capitolo 2.1 i muscoli che concorrono al movimento delle sopracciglia sono principalmente i muscoli frontali che trascinano le sopracciglia verso l'alto, e il muscolo corrugatore che le tira verso il basso e verso il ponte nasale. Tipicamente la parte esterna del sopracciglio è meno complessa e tende a seguire il movimento della parte centrale e quella mediana. In fase di animazione è importante considerare il sopracciglio come un oggetto unico e, quindi, è consigliato pensare come il movimento di una sua sezione possa influenzare quelle restanti. Inoltre, è importante considerare che il muscolo corrugatore è collegato ad entrambi i lati e quindi una sua azione dovrà andare ad influenzare almeno in parte entrambe le sopracciglia, bisogna immaginarsi una linea continua che vada ad unire le due sopracciglia in modo da evitare posizioni troppo sconnesse tra loro. Solitamente, a meno di esigenze particolari del modello da andare ad animare, i rig facciali presentano tre o quattro parametri a sopracciglio per controllarne il movimento.

Palpebre Avendo già creato l'animazione degli occhi e della loro chiusura, in questo passaggio ci si può concentrare sulla forma e sulla posizione che le palpebre avranno durante la scena. La parte iniziale (la parte inferiore) della palpebra superiore determina enormemente l'espressione e l'intenzione generale del personaggio. Variando lo spazio che intercorre tra la pupilla o la fine dell'iride e l'inizio della palpebra si possono ottenere espressioni estremamente differenti tra di loro e per questa ragione è importante prestarci molta attenzione. Se la distanza tra palpebra e iride è molta, e quindi è visibile parte della sclera, si otterrà un'espressione molto energica, come ad esempio di paura o di sorpresa. Man mano che la palpebra scende si otterrà un'espressione più naturale, fino ad un'espressione normale o di base quando la palpebra andrà a toccare l'inizio dell'iride. Nel momento in cui si inizia a coprire parzialmente l'iride, l'espressione che si andrà a generare sarà sempre più legata alla rilassatezza, alla tristezza o alla stanchezza. Quando la palpebra scende a coprire più della metà della pupilla allora si avrà un'espressione di sonnolenza o incoscienza. Data l'enorme espressività generata dalla posizione della palpebra è necessario prestare molta attenzione. Il rapporto di distanza tra pupilla e palpebra superiore deve essere mantenuto anche durante il movimento dell'occhio. Molti rig facciali permettono di compensare automaticamente questo movimento. Invece la palpebra inferiore è soggetta a movimenti e ad azioni molto più limitati e non è necessario che segua in modo preciso il movimento della pupilla.

Durante l'animazione delle palpebre è quindi necessario tenere a mente che la palpebra superiore è influenzata sia del sopracciglio sia dalla posizione della pupilla, mentre la palpebra inferiore è influenzata principalmente dal movimento della guancia, a sua volta influenzato dall'azione dell'angolo della bocca, e in parte dal movimento della pupilla.

Animazione della parte inferiore del volto e sincronizzazione labiale (lip sync)

Dopo aver definito l'intenzione emotiva grazie alla fase di blocking e all'animazione di occhi e sopracciglia, in questa fase ci si può concentrare su uno dei punti cardine dell'animazione facciale, ovvero la sincronizzazione labiale con la traccia audio. Realizzare una corretta sincronizzazione è fondamentale per rendere l'animazione più organica e dare maggior naturalezza alle scene di dialogo e per favorirne la comprensione da parte dello spettatore. Data la complessità che deriva dal dover gestire l'animazione delle labbra, guance, naso, lingua e della mascella, la creazione di un corretto *lip sync* è un compito estremamente complesso e richiede ottime competenze in materia.

Mascella Tendenzialmente la fase di sincronizzazione labiale inizia andando a gestire l'apertura della mascella durante la produzione dei differenti fonemi. Si creano le pose principali e dopo di che si va ad agire sulle curve di interpolazione tra i vari keyframe, in modo da non avere linee rette ma andare a gestire l'*ease-in* e l'*ease-out* in modo da impostare le differenti velocità di movimento.

Angoli della bocca Il movimento degli angoli della bocca tendenzialmente si svolge su due assi, su e giù, dentro e fuori. Tipicamente si dovrebbe avere già effettuato in fase di blocking il movimento su/giù e quindi ora ci si può concentrare maggiormente nella caratterizzazione dei vari fonemi. Il movimento dentro/fuori è determinato quasi unicamente dai fonemi prodotti ed in particolar modo dalle vocali pronunciate. L'interpolazione tra gli in-between frame non deve essere lineare ma è necessario gestire accelerazione e decelerazione. La precisione è fondamentale e quindi il numero di keyframe inseriti può essere molto elevato.

Labbra La forma delle labbra è determinata principalmente dal fonema pronunciato ed in particolar modo dalle consonanti. Ad esempio, nel caso in cui venga pronunciata una [b], [m] o [p] le labbra dovranno congiungersi e rimanere unite per almeno due frame, mentre nel caso in cui vengano pronunciate le consonanti [f] o [v] il labbro inferiore dovrebbe andare a toccare la parte inferiore dei denti superiori. Bisogna, inoltre, considerare che si sta realizzando un prodotto di animazione e che quindi può essere utile accentuare alcuni movimenti per rendere tutto più comprensibile, anche andando a rinunciare ad un movimento realistico.

Naso Ciò che va ad influenzare maggiormente il comportamento del naso è la sua unione con il labbro superiore e la posizione dell'angolo della bocca, per questo è importante mantenere coerente la sua distanza con il labbro superiore ed evitare movimenti scoordinati.

Lingua L'animazione della lingua è uno strumento fondamentale per migliorare la comprensibilità di un dialogo da parte dello spettatore. I movimenti principali della lingua si hanno in corrispondenza di fonemi che comprendono [d], [l], [r], [t] e il th[1]. Risulta utile concentrarsi solo sui movimenti principali e rendere questi movimenti puliti e comprensibili, aumentandone la durata d'azione e andando a gestire l'*ease-in* ed *ease-out*. Ovviamente risulta controproducente animare la lingua in istanti in cui non risulta visibile.

Appeal e asimmetria Trattandosi, in questo caso, di un'animazione di un personaggio stilizzato è importante andare a considerare i dodici principi dell'animazione visti nel capitolo 2.4. È bene spingere la nostra animazione verso uno stile 2D, andando ad ignorare in parte l'anatomia del volto in favore del risultato finale. L'obiettivo è quello di ottenere un risultato interessante e credibile dal punto di vista della camera, per questo motivo, spesso, ci si può permettere di creare deformazioni assolutamente non organiche alla parte nascosta del volto pur di ottenere il risultato voluto su schermo. Ulteriori deformazioni non realistiche che spesso vengono applicate sono lo *squash & stretch*, queste risultano molto importanti per enfatizzare determinate azioni e per rendere alcune espressioni più leggibili e comprensibili. Un altro fattore importante da considerare è l'asimmetria; tutti i volti umani sono in parte asimmetrici e riproporre un'asimmetria alle espressioni facciali renderà tutta l'animazione più organica e naturale. Come linea guida si può considerare che tipicamente le espressioni spontanee, solitamente legate a stati emotivi come gioia e paura, sono simmetriche, mentre le espressioni volontarie, come ad esempio rabbia e confusione, sono tendenzialmente asimmetriche.

Commento

Le fasi sopra riportate vanno considerate come delle valide linee guida e non come delle regole fisse. Inoltre, il processo di animazione tramite keyframe raramente sarà lineare ma il più delle volte ci si troverà a dover aggiustare e modificare ciò che era stato fatto precedentemente. Come si è potuto evincere da quanto sopra riportato, realizzare un'animazione facciale con questo approccio risulta un lavoro estremamente complesso che richiede ingenti quantità di risorse in termini di tempo, competenze e denaro. Tuttavia, il risultato che si andrà ad ottenere sarà di estrema qualità. Per questo motivo è l'approccio più diffuso all'interno di studi di animazione

[1] In inglese il th può essere pronunciato principalmente in due modi: /ð/ (come in this) e /θ/ (thing).

di grandi dimensioni, mentre studi di natura più indipendente, tra cui *Robin Studio S.r.l.s.*, tipicamente cercano soluzioni alternative.

5.2 Animazione guidata da testo o da audio (Text/Audio-Driven)

Come si è potuto vedere nella sezione 5.1.1, realizzare l'animazione del parlato tramite un approccio diretto con fotogrammi chiave, sebbene il risultato possa essere eccellente, comporta livelli di complessità molto elevati. Per questo motivo sono stati studiati e proposti differenti approcci che consentono di (semi)automatizzare tale processo.

Le tecniche di animazione *Text* e *Audio-Driven* vanno a soddisfare tali esigenze. Tali approcci prevedono, per l'appunto, di andare a guidare l'animazione del parlato tramite dei dati raccolti attraverso testo scritto o attraverso file audio, questi dati verranno infatti elaborati e poi utilizzati per realizzare l'animazione. I dati più utilizzati come base di partenza con questo approccio sono relativi ai *fonemi*[2] e *visemi*. I visemi altro non sono che la controparte visiva dei fonemi, in pratica consistono nelle posizioni che un volto assume durante la pronuncia di un determinato fonema.

Generalizzando si può dire che tali tecniche suddividono i dati di ingresso (testo o audio) in fonemi, tali fonemi vengono legati ad un certo istante temporale, dopo di che si crea una mappatura tra i fonemi e i rispettivi visemi che dovranno essere applicati al modello per realizzare l'animazione finale. Il range di suoni che può essere generato dall'apparato fonatorio umano è incredibilmente vasto, per questo motivo è stato necessario classificare e mappare tali suoni con dei simboli univoci. L'alfabeto più completo e più utilizzato nell'analisi del parlato è sicuramente l'*IPA* (International Phonetic Alphabet) la cui ultima versione è stata pubblicata nel 2015 [26].

Tali approcci possono essere raggruppati in modelli basati sulla linguistica e modelli basati sul machine learning [27].

5.2.1 Modelli basati sulla linguistica

Questi modelli basano l'intero approccio sui fonemi e visemi. Sotto questo gruppo rientrano anche i primi studi e le prime tecniche di animazione guidata dall'audio proposti. Nei primi approcci tipicamente i fonemi venivano estrapolati dalla sorgente e poi direttamente convertiti in visemi. Queste tecniche possono risultare utili se si vuole animare un cartone animato tradizionale (2D) o un qualcosa di stilizzato, in quanto non è necessaria una rappresentazione realistica del parlato [28], tuttavia risultano inadatte se applicate a prodotti con maggiore realismo [2, capitolo 9]. Questo perché non esiste una corrispondenza uno a uno tra fonemi e visemi, infatti alcuni fonemi possono essere rappresentati tramite un unico visema; ad esempio i fonemi [b], [m] e [p] possono essere rappresentati tutti tramite la semplice chiusura delle labbra. Tutto ciò può creare confusione nello spettatore [2, capitolo 9].

[2] In linguistica il fonema è «un segmento fonico-acustico non suscettibile di ulteriore segmentazione in unità dotate di capacità distintiva». Fonema. (Consultato il 22/02/2021). *Enciclopedia on line - Treccani*. URL: https://www.treccani.it/enciclopedia/fonema/#:~:text=Il%20termine%20si%20%C3%A8%20affermato,d'ordine%20pi%C3%B9%20complesso)..

Un altro importante fattore da considerare è che, sebbene sia possibile individuare i singoli fonemi, nella realtà la postura che un volto assume in uno specifico istante non può semplicemente essere determinato dal fonema corrispondente perché entrano in gioco altri fattori, come il contesto in cui tale fonema viene pronunciato, ovvero i fonemi che lo precedono e che lo seguono, ma anche l'intonazione e lo stato emotivo del soggetto. Infatti, nel parlato si verifica una sovrapposizione dei suoni prodotti e quindi i confini tra i singoli fonemi o gruppi di fonemi sono molto più sfumati [2, capitolo 9]. Questa sovrapposizione può essere indicata con il termine *coarticolazione* o evento coarticolatorio [29]. In pratica, sia il suono prodotto che la postura dell'apparato fonatorio sono strettamente legati al contesto, ad esempio la postura corrispondente alla produzione della vocale [a] può variare a seconda che la consonante che la precede sia una [b] o una [q].

Tale fenomeno presenta un elevata complessità ma deve essere tenuto in considerazione nel caso in cui si vogliano realizzare delle animazioni facciali di maggiore realismo e precisione.

Una delle soluzioni maggiormente adottate è il modello proposto da Cohen e Massaro [30] che sfrutta il concetto di dominanza. In questo modello ad ogni fonema viene associata una funzione di dominanza e viene legato a un gruppo di valori dei parametri di controllo, che verranno utilizzati per definire come andare a deformare la mesh. La funzione di dominanza indica l'influenza che un visema avrà nella determinazione dell'espressione facciale finale in un determinato istante. Infine, le funzioni di dominanza dei vari fonemi, vengono unite tenendo in considerazione anche la velocità del parlato, andando quindi a definire il risultato finale.

Tali metodi hanno il vantaggio di permettere un elevato controllo sull'intero processo e sul corretto risultato, soprattutto nella produzione delle posizioni relative ai fonemi [b], [m], [p], [f] e [v]. Tuttavia, hanno come svantaggi la complessità dell'intero processo, la difficoltà nell'ottenere un risultato convincente per suoni non previsti dagli alfabeti fonetici, ma soprattutto non prevedono l'animazione completa del volto ma si limitano a definire la deformazione di labbra, mascella e a volte della lingua [27], quindi l'aspetto emotivo dell'espressione facciale, che risiede per la maggior parte nella sezione superiore del volto, dovrà essere gestito con altre tecniche, rischiando di ottenere un risultato sconnesso e poco organico.

5.2.2 Modelli basati sul machine learning

Tali tecniche fanno utilizzo di machine learning e reti neurali per andare ad utilizzare direttamente l'audio senza dover fare un'analisi esplicita della struttura del parlato [27]. Per poter funzionare è necessario "addestrare" la rete neurale; una delle tecniche più diffuse consiste nell'andare a stimare l'*Hidden Markov Model* (HMM), basato su quanto raccolto da fonti audio e video. In alternativa, il machine learning viene sfruttato per identificare in maniera più precisa la coarticolazione ed effettuare una mappatura tra testo e fonemi e successivamente tra fonemi e visemi.

Esempi recenti di ricerche in questa direzione sono state effettuate da NVIDIA nel 2017 [27] (si veda la figura 5.1), e nel 2019, basandosi su principi simili, è stato pubblicato il modello VOCA (Voice Operated Character Animation) [31].

Sebbene le ricerche in questo campo avanzino, nella maggior parte dei casi, compresi quelli sopra citati, il target di riferimento non è l'industria cinematografica per prodotti di animazione 3D, bensì sono rivolti maggiormente al settore videoludico. Questo si può evincere dal fatto che tali metodi sono costruiti in modo da poter animare in maniera diretta, a partire da un file

audio, i vertici di specifici modelli 3D, e non forniscono la possibilità di intervento da parte dell'animatore, cosa che tendenzialmente risulta non necessaria all'interno di un videogioco.

Tutto ciò pone un enorme limite all'implementazione di tali tecniche all'interno del settore dell'animazione computerizzata.

5.2.3 Stato emotivo

Un fattore importantissimo e determinante per il risultato finale è lo stato emotivo del soggetto. Infatti, sebbene, come abbiamo visto precedentemente, sia possibile realizzare un corretto lip sync basandosi unicamente sui fonemi pronunciati, questo darà come risultato un'animazione globale del volto meccanica e non organica. Infatti, oltre a gestire in modo corretto il movimento di bocca e lingua, durante il parlato si verificano moltissimi altri cambiamenti sul volto, e molti di questi dipendono dallo stato emotivo con cui vengono pronunciati. Anche se in alcune situazioni è possibile animare le restanti parti manualmente, ad esempio la parte superiore del volto, è vantaggioso adottare metodi che permettano di trasmettere in modo automatico lo stato emotivo insieme alla creazione del lip sync. Tipicamente lo stato emotivo viene analizzato in maniera separata ed aggiunto prima della produzione dell'output (si veda la figura 5.1).

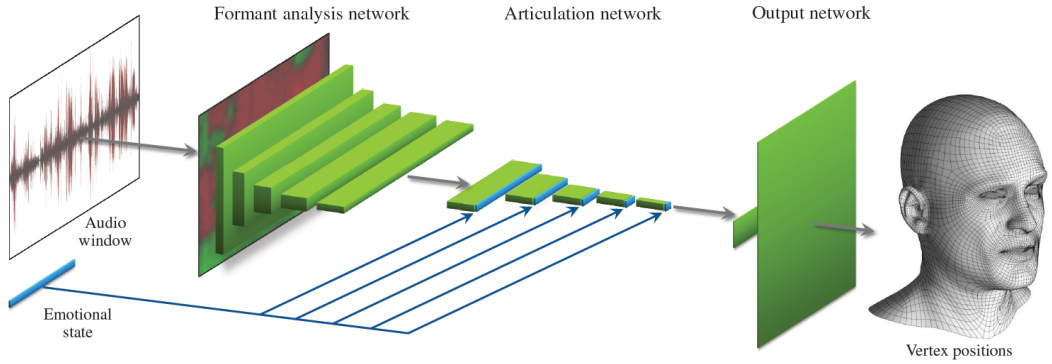


Figura 5.1: Schema della rete neurale profonda presentata in [27]. La rete richiede come dato d'ingresso circa mezzo secondo di traccia audio e restituisce la posizione 3D dei vertici di una mesh con topologia fissa . La rete accetta anche un input secondario che descrive lo stato emozionale.

5.3 Motion Capture

Come si è potuto leggere nelle sezioni 5.1 e 5.2, effettuare un'animazione facciale è un lavoro estremamente complesso, che richiede la creazione e la corretta gestione temporale di un numero molto elevato di parametri; per questa ragione molti sono stati gli sforzi per trovare possibili alternative che andassero ad agevolarne il lavoro. Una possibile soluzione consiste nel trasferire i dati raccolti dalla registrazione di un attore sul personaggio digitale da animare.

Il *motion capture*, nella versione abbreviata *Mocap* e in italiano "cattura del moto", è il processo di codifica del moto dal mondo reale a quello digitale [22]. Utilizzato principalmente nell'industria dell'intrattenimento, ma non solo, per registrare il movimento di un attore e applicarlo ad un personaggio digitale. Nel caso in cui si effettui la registrazione, oltre che del

corpo, anche delle espressioni del volto, allora il motion capture viene anche indicato con il termine *performance capture* [22]. In generale, con queste tecnologie è possibile applicare i dati raccolti dalla performance dell'attore sia a personaggi realistici ma anche a creature non antropomorfe o non umane. La fase nella quale i dati catturati vengono applicati al modello digitale si chiama *retargeting* (vedi pagina 57).

Un attore particolarmente conosciuto per le sue performance in mocap è sicuramente Andy Seriks, colui che ha dato vita a personaggi come *Gollum* nella trilogia de *Il Signore degli Anelli*, *King Kong* (2005), *Cesare* nel franchise de *Il pianeta delle scimmie* e molti altri.

Oltre a poter utilizzare il motion capture per inserire alcuni personaggi all'interno di film in live action, questi strumenti possono essere utilizzati per realizzare intere produzioni, come dimostrato da pellicole quali *A Christmas Carol* o *Polar Express* già presentati nel capitolo 2.2.

L'utilizzo del mocap risulta ideale laddove l'obiettivo sia realizzare animazioni (semi) realistiche [22].

Le tecniche di mocap utilizzate per registrare le performance del corpo e del volto possono essere differenti e tipicamente si preferisce procedere con la loro registrazione in maniera disgiunta, effettuando due sessioni di ripresa distinte o andando ad utilizzare differenti strumenti di cattura, ad esempio utilizzando un *Head Mounted Camera* (HMC) per la registrazione del volto. Per questo motivo in questo contesto verranno affrontati principalmente gli strumenti e i metodi per il mocap facciale. Uno studio più approfondito delle tecniche di cattura del movimento del corpo e dell'applicazione dei dati acquisiti, sui personaggi della serie animata *Reverie Dawnfall*, è stato effettuato dalle mie colleghe Coarezza Melissa e Abate Stefania all'interno dei rispettivi lavori di tesi di laurea Magistrale [32, 33].

L'obiettivo del mocap facciale è registrare le più piccole deformazioni nell'espressione del viso e trasmettere tali deformazioni ad un personaggio digitale [22]. Siccome i volti umani possono variare molto tra di loro e data l'esigenza di registrare i più fini movimenti, è necessario adottare delle tecniche sofisticate. Bisogna, inoltre, far presente che anche con gli strumenti di cattura più precisi sul mercato, sarà comunque necessario un lavoro più o meno lungo di rifinitura dei dati e di correzione dell'animazione, effettuato preferibilmente da animatori esperti, in modo che si applichi al meglio sul personaggio computerizzato.

Le tecniche di cattura utilizzate si possono in generale suddividere in due gruppi: quelle che fanno utilizzo di marker (*marker based*) e quelle che non ne fanno uso (*markerless*) [22].

5.3.1 Motion Capture marker-based

Con questo approccio, i marker vengono posti sul volto dell'attore in modo tale che ne vengano registrati i movimenti. L'utilizzo dei marker rende l'operazione di tracciamento molto più semplice in quanto i punti sono facilmente distinguibili; di contro si ha che il numero di marker utilizzati potrebbe risultare insufficiente e un loro aumento potrebbe compromettere la performance attoriale a causa del loro ingombro. Per riprodurre nel modo più fedele i dati raccolti sarà necessario riprendere il volto con almeno due camere con punti di vista differenti, in modo da ricostruirne la posizione in uno spazio 3D tramite tecniche come la triangolazione.

Come indicato in [22], i marker possono essere di diversa natura, di seguito si vanno a descrivere le principali tipologie.

Marker riflettenti per sistemi ottici

Sono marker di un materiale che in particolari condizioni risulta riflettente, vengono utilizzati nei sistemi ottici in combinazione con appositi filtri per le lenti della camera. In questo modo i marker risulteranno estremamente visibili e quindi facilmente tracciabili.

L'accuratezza del sistema dipende dal numero di camere utilizzate e dalla dimensione del volume da esse coperto, più il volume sarà grande, maggiore sarà il numero di camere necessarie. Data la piccola dimensione dei marker, un volume di cattura minore è consigliato. Per questo motivo solitamente non vengono adoperate le stesse camere utilizzate per la cattura del corpo, in quanto il volume sarebbe troppo ampio e il risultato sarebbe inaccurato. Tuttavia, ridurre il volume di cattura richiederebbe di registrare il volto in un secondo momento, andando ad incidere sul budget finale.

Sebbene siano sufficienti due camere per ricostruire la posizione di un punto in un sistema tridimensionale, data la forma e la dimensione dei marker, essendo tutti uguali tra loro, il software potrebbe fare confusione e, dato il verificarsi di possibili occlusioni, è consigliato aumentare il numero di camere fino ad ottenere una copertura totale e precisa del volto. Questo incide negativamente sul budget finale.

Un altro problema dei marker ottici è che, oltre a essere costosi, devono essere applicati manualmente e spesso possono cadere in fase di ripresa, inoltre la precisione dei dati raccolti dipende da dove i marker sono posizionati. È importante mettere un numero maggiore di marker laddove si verificano le deformazioni più importanti (es bocca e occhi) e tenere in considerazione sia la conformazione dell'attore che del personaggio da animare.

Marker con makeup

I marker in questo caso vengono realizzati con del trucco e tipicamente hanno una forma circolare. In questo modo si evita il problema di possibile caduta dei marker durante le riprese e si limita anche l'ingombro prodotto sul volto dell'attore, permettendone quindi una performance più naturale, inoltre, risultano meno costosi. Tuttavia, la fase di tracciamento risulta più complessa in quanto la visibilità, data dal contrasto tra i marker e lo sfondo, è minore rispetto a quelli ottici, inoltre, essendo direttamente applicati sul volto, potrebbero andare a deformarsi seguendo lo strato di pelle, tali deformazioni potrebbero determinare delle imprecisioni in fase di tracciamento.

Utilizzare i marker realizzati con il trucco consente anche di poter adoperare delle camere "normali" e non più quelle utilizzate per i sistemi ottici. Il vantaggio consiste nel fatto che queste camere sono più comuni, tipicamente meno costose e possono essere anche molto compatte, tanto da poter usare delle head-mounted camera riducendo significativamente il volume di cattura e quindi aumentando la precisione con cui i punti vengono catturati. Inoltre, catturare i dati con un formato video, permette di avere delle importanti reference in fase di animazione.

Importante però, in questo caso, prestare attenzione ad una corretta e costante illuminazione del volto. I fattori da tenere in considerazione sono il numero di camere utilizzate, il frame rate di ripresa e la compressione dei dati [22].

Posizionare i marker

Sia nel caso si utilizzino marker riflettenti sia che si utilizzino marker realizzati con makeup, il tempo che sarà necessario per andare ad applicarli correttamente al volto dell'attore inciderà

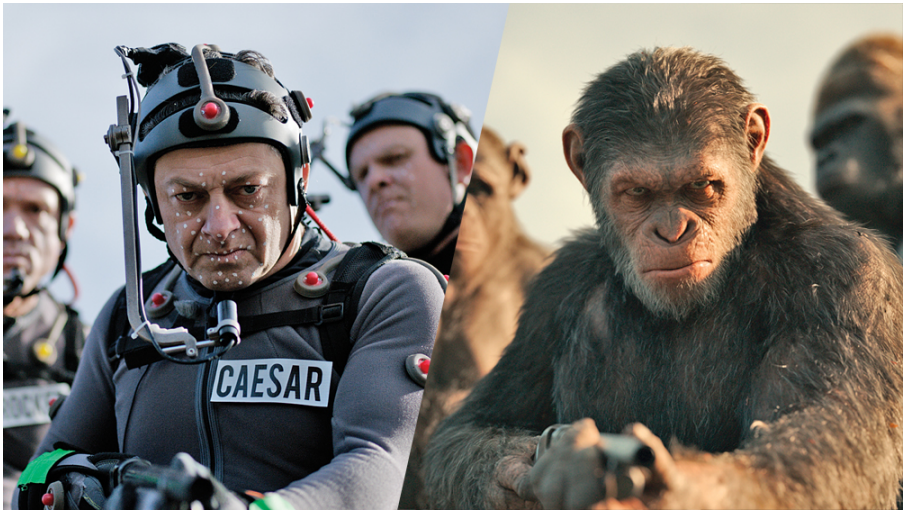


Figura 5.2: Sistema di motion capture marker-based (marker con makeup). Per gentile concessione di Peter Caranicas.

sull'organizzazione delle riprese. Posizionare in modo corretto i marker risulta fondamentale in quanto i dati raccolti dipendono strettamente dalla quantità e dalla posizione in cui i marker si trovano.

Un approccio tipico è posizionarli con un *metodo basato sui muscoli*, che tipicamente si rifà a quanto indicato dal FACS (vedi capitolo 2.3). In questo modo, anche se attori differenti hanno volti differenti, sarà possibile identificare e catturare più o meno precisamente lo strato muscolare sottostante, comune a tutti. In questo caso sarà necessario realizzare un modello ed un rig in grado di gestire tale metodo.

Siccome il processo di applicazione dei marker è un lavoro da svolgere manualmente, che richiede molto tempo ed è determinante per la corretta riuscita della cattura dati, in produzioni di grandi dimensioni è utile creare delle *maschere* in cui sono presenti dei buchi per i marker in modo che il loro posizionamento risulti più immediato e costante nel tempo.

Distribuzione densa e casuale dei marker

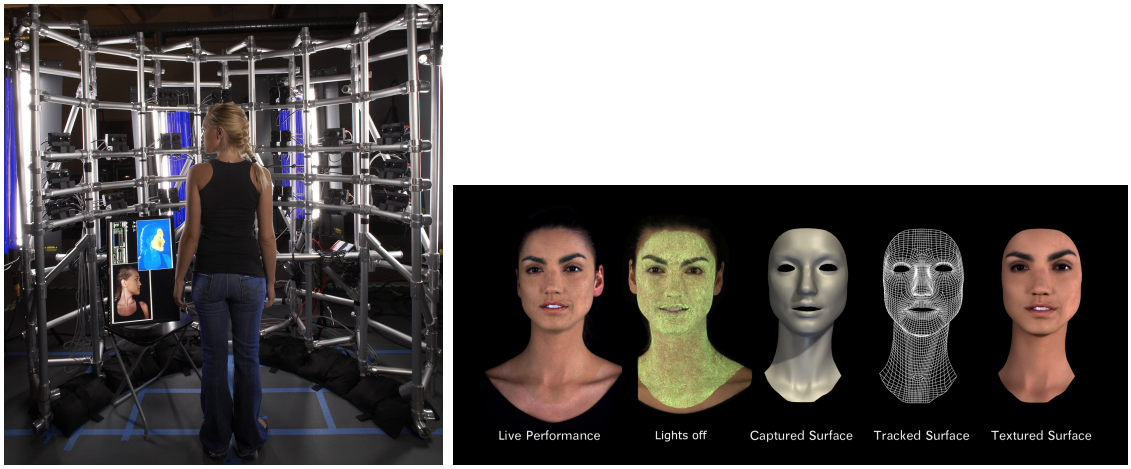
L'applicazione dei marker, come abbiamo visto, può essere un processo molto lungo. Una tecnica più recente, a metà strada tra i sistemi marker-based e quelli markerless, diventata popolare dopo essere stata utilizzata per la realizzazione del film *Il curioso caso di Benjamin Button* (2008), consiste nell'applicare sul volto dell'attore una texture densa e con distribuzione casuale in modo da andare a creare un numero molto elevato di punti (marker) che possano essere successivamente tracciati. Tale texture può essere di diversa natura, tipicamente consiste in uno strato di trucco di materiale riflettente o fosforescente in modo che sia facilmente visibile in certe condizioni di illuminazione.

Il vantaggio di utilizzare un makeup fosforescente è dovuto al fatto che in questo maniera diventa possibile separare in modo netto e facile il volto dallo sfondo, rendendo il processo di tracciamento più preciso.

La fase di cattura deve essere effettuata in un ambiente apposito, tipicamente si utilizza una sorta di gabbia aperta sulla quale vengono montate un numero elevato di luci e camere ad alta

risoluzione (figura 5.3a). Da un lato si ha il vantaggio che l'attore ha maggiore possibilità di azione e movimento, seppur sempre limitato al volume di cattura, ma dall'altro lato l'attrezzatura necessaria risulta piuttosto costosa e difficilmente sostenibile da una produzione indipendente.

Con questa tecnica si è in grado di generare modelli con un numero di vertici anche superiore a 100 mila [22], il che permette di ottenere un volto estremamente realistico, preciso e pieno di dettagli. Tuttavia, il problema maggiore è dovuto a possibili incoerenze temporali della topologia della mesh 3D generata, ciò comporta un enorme limite in fase di post produzione in quanto tipicamente per questioni di retargeting si ha necessità di avere una topologia poligonale costante nel tempo. Basti pensare che il sistema più utilizzato per gestire le deformazioni fa utilizzo di blendshape, le quali, come visto nella sezione 3.2, hanno necessità di basarsi su un'unica topologia della mesh. Uno dei sistemi più popolari è sicuramente quello di MOVA®Contour [34], utilizzato per la prima volta nella realizzazione del film *Il curioso caso di Benjamin Button* sopra citato.



(a) Attrice davanti a una griglia densa di speciali luci fluorescenti e camere sincronizzate.

(b) Cinque fasi del sistema MOVA Contour Reality Capture, che danno vita a un personaggio animato fotorealistico ad alta risoluzione.

Figura 5.3: Sistema di mocap con distribuzione densa e casuale dei marker. Per gentile cortesia di MOVA®Contour®Reality Capture [34].

5.3.2 Motion Capture Markerless

L'utilizzo di marker facilita sicuramente la fase di tracciamento in quanto il software sarà in grado di riconoscere in maniera precisa i punti da tracciare; tuttavia, le risorse richieste in termini di tempo di preparazione e in termini di hardware utilizzato sono molto elevate, andando ad incidere negativamente sul budget. Per questo motivo non è possibile, o comunque non risulta vantaggioso, utilizzare tecniche di mocap con marker per determinate produzioni, ad esempio per produzioni indipendenti.

Per questi motivi sono state sviluppate delle tecniche di cattura che non fanno utilizzo di marker, ma se da un lato offrono una soluzione ai problemi sopra indicati, dall'altro richiedono tecniche e software più sofisticati per andare ad effettuare una cattura corretta. Tali tecniche basano l'intero processo su quelle che sono le caratteristiche salienti del volto, come occhi, bocca e sopracciglia.

Figura 5.4: Sistema di mocap markerless per produzioni indipendenti.

Active Appearance Model (AAM)

Uno dei metodi di mocap markerless più utilizzato nell'industria dell'intrattenimento è l'*Active Appearance Model* (AAM) [35]. Basato su dei modelli statistici che vanno ad analizzare le principali caratteristiche dei volti, dopo di che va a generare un modello deformabile che va a mettersi in relazione con il volto catturato, in modo che i movimenti vengano tracciati con facilità.

Basandosi su dei modelli statistici, risulta fondamentale, inizialmente effettuare una fase di *training* andando a specificare, manualmente o con un processo semi-automatizzato, la posizione dei punti salienti del volto (sopracciglia, labbra, occhi, naso...) e andando ad etichettarli. Tipicamente nell'effettuare tale fase si vanno ad utilizzare frame di espressioni facciali che replicano le espressioni facciali principali indicate dal FACS. In questo modo è possibile avere delle pose chiave certe su cui basare l'intero approccio di cattura. Tuttavia, la fase di training può richiedere molto lavoro e precisione, inoltre i punti caratteristici di un volto sono limitati e presenti solo in alcune zone piuttosto che in altre. Per questo motivo diventa complicato analizzare il movimento di parti come le guance o il mento.

Pictorial Structure Model (PSM)

Un modello simile all'AAM è il *Pictioral Structure Model* (PSM) [36]. Infatti, anche questo si basa su un modello statistico relativo a posizione e movimento dei punti caratteristici del volto. Anche in questo caso è necessaria una fase di training ma il vantaggio introdotto consiste nel fatto che il PSM è in grado di calcolare la posizione dei punti salienti per ogni frame senza dover necessariamente aver conoscenza della posizione nei frame precedenti. In pratica non è più necessario inizializzare il processo e il tracciamento risulta molto più preciso.

5.3.3 Tracciamento

Le tecnologie di motion capture, sia che facciano utilizzo di marker sia che non ne facciano uso, sono caratterizzati da una fase cruciale per il risultato finale, ovvero il tracciamento (*tracking*).

Questa fase consiste nel definire per ogni istante temporale, tipicamente ogni frame, la posizione in cui si trovano i punti analizzati, in questo modo sarà possibile ricostruirne il movimento.

A differenza di quanto accade per la cattura del movimento del corpo, in questo caso si vuole registrare la deformazione del volto e non la sua posizione nello spazio. Ciò significa che la posizione dei punti catturati non dovrà essere definita nel sistema di coordinate globale della scena, cosa che invece avviene per la cattura del corpo, ma sarà necessario definire un sistema locale; altrimenti i punti del volto analizzati saranno soggetti oltre che alla deformazione del volto stesso, anche al movimento della testa nello spazio di cattura. Per questo motivo è fondamentale analizzare il movimento dei punti su una testa fissa nel tempo e nello spazio. Una possibile soluzione potrebbe consistere nel bloccare con qualche supporto la testa dell'attore in modo che ne sia impedito il movimento ma, per ovvie ragioni, questa è un approccio sconsigliato in quanto la performance attoriale ne sarebbe gravemente compromessa.

Tipicamente quello che si fa è effettuare un processo di *stabilizzazione* [22] in modo da andare ad eliminare il movimento della testa tramite tecniche sofisticate. Un metodo comunemente usato consiste nel definire tre, o più, punti fissi su cui poi basare il sistema di riferimento locale a cui gli altri punti catturati andranno a fare riferimento. I punti più utilizzati sono la punta del naso e la posizione delle spalle, tuttavia, sfortunatamente non esistono punti completamente rigidi, ma tutti sono soggetti a possibili deformazioni e movimenti non previsti. Per questo motivo si fa utilizzo di specifici software che sfruttano modelli statistici per ottenere una stabilizzazione migliore.

Anche nel caso si faccia utilizzo di head-mounted camera, che essendo montate sulla testa dovrebbero avere una ripresa stabile del volto andando già ad escludere in movimento stesso della testa, si richiede una fase di stabilizzazione in quanto il sistema potrebbe essere soggetto a possibili oscillazioni o slittamenti dovuti ai movimenti dell'attore in fase di ripresa.

5.3.4 Retargeting

Una volta che i dati della performance attoriale sono stati catturati, stabilizzati e tracciati si può procedere con il trasferire tali dati sul personaggio digitale che si vuole animare. Tale fase si chiama *retargeting*. Sebbene alcune tecniche viste sopra permettano di creare mesh poligonali ad elevatissima risoluzione, spesso queste risultano incoerenti a livello temporale, ovvero la loro topologia potrebbe variare ad ogni frame. Questo risulta inadatto nella maggior parte delle situazioni in cui si voglia effettuare l'animazione facciale.

Infatti, in fase di animazione, nella quasi totalità dei casi, si fa utilizzo di quella che si può chiamare topologia universale, ovvero una topologia fissa nel tempo, la cui animazione sarà data da una deformazione dei suoi vertici. Una delle tecniche di deformazione della geometria più utilizzate quando si parla di motion capture facciale è l'utilizzo di blendshape (vedi sezione 3.2).

In fase di retargeting si è soliti non applicare direttamente l'animazione alla mesh del volto ma si preferisce agire sul rig facciale. Un modo per approcciarsi con il rig facciale è chiamato *direct drive*. Consiste nel legare in maniera diretta i dati relativi al movimento dei marker al rig e quindi al metodo di deformazione.

Tale approccio presenta molti limiti; prima di tutto sarà necessario avere una relazione uno a uno tra marker catturato e parametro del rig altrimenti si andrebbero a perdere importanti informazioni raccolte in fase di registrazione. Poi, tale metodo può andare bene nel caso in cui la conformazione del volto dell'attore e del personaggio da animare siano uguali (rapporto 1:1),

mentre, come accade nella maggior parte dei casi, andare ad animare volti differenti risulta più complesso in quanto sarà necessario ricreare uno specifico rig per gestire le differenze nelle proporzioni.

Se ci si trova in questa situazione bisogna adottare tecniche di *expression cloning*, ovvero quando il volto "sorgente" e la mesh "target" hanno proporzioni significativamente differenti, ad esempio quando un attore deve animare un personaggio bambino (come per i film *Polar Express* e *Il curioso caso di Benjamin Button*) oppure quando deve animare creature non umane (come in *Avatar* e per il personaggio di *Gollum*) [13].

Un approccio di utilizzo più comune consiste nell'utilizzare un *modello comportamentale* [22], che si occupa di tradurre i dati raccolti in fase di registrazione in un formato che possa essere facilmente utilizzabile dal rig facciale. In pratica, algoritmi di varia natura vengono utilizzati per trasformare i dati relativi alle coordinate spaziali dei marker in grandezze scalari, come ad esempio la percentuale di spostamento tra due marker vicini.

Il modello comportamentale è utilizzato in quasi tutte le situazioni in cui non si fa utilizzo di marker, ad esempio nel sistema *ARKit* i dati acquisiti vengono trasformati in grandezze scalari che vanno a definire l'influenza di determinate blendshape. Sicuramente uno degli approcci più diffusi di modello comportamentale si basa sul FACS o comunque sulla struttura muscolare del viso.

5.3.5 Commento

In generale le tecniche di mocap markerless risultano meno precise rispetto a quelle marker based. Per questo motivo nelle grandi produzioni e laddove sia richiesto un elevato grado di realismo, i sistemi che utilizzano marker, sono da preferire. Tuttavia, in molte situazioni non è necessario ottenere dei volti realistici ma piuttosto l'obiettivo è animare dei personaggi animati stile cartoon o comunque stilizzati. In queste situazioni l'utilizzo di sistemi markerless risulta sicuramente adeguato se non vantaggioso. Inoltre, oggi giorno si diffondono sempre di più tecnologie che si basano su mocap markerless rivolti ad un mercato consumer. Sebbene, solitamente, la loro precisione non sia al livello degli strumenti utilizzati dalle grandi produzioni, è possibile ottenere, nel caso in cui l'obiettivo sia un prodotto stilizzato, delle animazioni più che soddisfacenti senza dover investire ingenti somme di denaro.

Tecnologie come *ARKit*, sviluppata originariamente da Stefien Bouaziz et al. [37] e successivamente acquistata da Apple Inc. [38, 39], sono un enorme potenziale per gli studi di animazione indipendenti, dentro i quali ricade anche *Robin Studio S.r.l.s.* Per questo motivo, come vedremo nella parte II, tali strumenti sono alla base del processo di animazione facciale per la serie animata *Reverie Dawnfall.*

Capitolo 6

Prodotto e Contesto

Come si è potuto vedere nei capitoli 3 e 5, le tecnologie e i metodi che possono essere utilizzati per realizzare un'animazione facciale sono molteplici ed ognuno con le proprie specifiche caratteristiche. Per questo motivo è indispensabile identificare preventivamente la tipologia di prodotto che si vorrà realizzare e il contesto produttivo nel quale il lavoro dovrà essere svolto. Questi sono fattori determinanti sulla scelta delle tecniche e degli strumenti che si andranno ad utilizzare.

6.1 Tipologia di Prodotto

I prodotti che possono essere realizzati avvalendosi della computer animation (CA) sono i più disparati. Da prodotti di animazione "stile Pixar", a personaggi fotorealistici come nella pellicola *Il curioso caso di Benjamin Button*, fino all'animazione di personaggi all'interno di videogiochi. Risulta ovvio che ogni settore abbia le proprie esigenze e per questo motivo è necessario individuare gli strumenti che meglio soddisfano tali richieste.

Nel caso si voglia realizzare un personaggio fotorealistico, ad esempio per essere inserito all'interno di una pellicola cinematografica girata per la maggior parte con attori in live action, sarà fondamentale avere un modello facciale con un numero elevato di poligoni così da creare un elevato numero di dettagli, come rughe e altre imperfezioni. Inoltre, in questo caso sarà necessario realizzare un'animazione facciale il più realistica possibile, quindi utilizzare sistemi di motion capture ad alta definizione potrebbe essere una soluzione.

Nel caso, invece, l'obiettivo sia animare un personaggio all'interno di un videogioco, allora probabilmente l'approccio migliore sarebbe utilizzare tecniche di animazione guidate dall'audio o dal testo, in quanto l'animazione dovrebbe essere effettuata in real time, e l'utilizzo di tecnologie di motion capture potrebbero risultare ingombranti.

Per quanto riguarda i personaggi stilizzati, tra i quali rientrano anche i personaggi della serie animata *Reverie Dawnfall*, un approccio "classico" si basa sull'utilizzo di blendshape e di rig facciali piuttosto complessi, che vengono animati manualmente tramite keyframe.

Tuttavia, entrano in gioco altri fattori ascrivibili al contesto dentro il quale la produzione viene portata avanti.

6.2 Tipologia di contesto: Produzioni Major e Indie

Con tipologia di contesto intendiamo l'insieme di tutti quei fattori che caratterizzano un determinato ambiente di lavoro. In modo approssimativo si possono distinguere due gruppi dentro i quali le produzioni possono ricadere: produzioni *Major* e produzioni *Indie*.

Con questi termini non si vogliono distinguere i prodotti che vengono realizzati in base al mercato a cui si rivolgono; quindi prodotti *mainstream*, che si rivolgono ad un pubblico più ampio possibile, e prodotti di *nicchia*, che si rivolgono ad un pubblico con determinati gusti e caratteristiche e quindi tendenzialmente limitato nei numeri. Tuttavia, non è raro che produzioni major tendano a produrre opere mainstream mentre quelle indie tendano a realizzare prodotti di nicchia.

La suddivisione tra produzioni major e indie si basa principalmente su come l'azienda, o studio di animazione, abbia organizzato il proprio sistema produttivo, e sul budget messo a disposizione per realizzare il prodotto.

6.2.1 Budget investito sul progetto

Il budget a disposizione per la realizzazione del prodotto è forse uno dei fattori che più di tutti incide sulle scelte di produzione e quindi anche sul risultato finale.

Una produzione si dice *Major* quando gli investimenti, sia da parte dello studio di animazione, sia da parte di chi commissiona l'opera, sono elevati. In questo gruppo rientrano praticamente tutti gli studi di animazione più popolari e conosciuti, tra i quali: Pixar Animation Studios, Walt Disney Animation Studios, DreamWorks Animation, Illumination Entertainment, Sony Pictures Animation e molti altri (solo per citare gli studio che si occupano di animazione digitale).

Tipicamente, in produzioni di grandi dimensioni si realizzano opere che richiedono un elevato budget per essere portate a termine. Il budget servirà, escluse le azioni di marketing, per andare a retribuire un numero elevato di professionisti e per effettuare ricerca e sviluppo.

Siccome il budget di produzione è molto elevato, tipicamente le Major tendono a realizzare prodotti per la grande distribuzione in modo da garantirsi un buon rientro economico. Per fare un esempio, il lungometraggio animato *Toy Story 4* (2019) partendo da un budget di 200 000 000 $ e ha totalizzato un incasso globale di 1 073 394 593 $ [40].

Di contro, una produzione si dice *Indie* o indipendente, quando non può contare su grandi capitali da poter investire. In generale gli studio indipendenti sono caratterizzati da scarse risorse economiche personali e nella maggior parte dei casi, per poter realizzare un prodotto si vedono costretti a rivolgersi ad investitori esterni attraverso sponsor, concorsi pubblici, crowdfunding o altri mezzi. Per questo motivo, tipicamente, il budget a disposizione risulta imparagonabile a quello che hanno a disposizione le Major. Tuttavia, non è da escludere che tramite investimenti esterni si riescano a raccogliere le risorse utili per realizzare un prodotto di alta qualità.

Diventa difficile delineare una linea netta tra le tipologie di produzione, perché uno studio indipendente potrebbe trovarsi a realizzare un prodotto che a tutti gli effetti comporta una produzione di grandi dimensioni. Ma dentro la categoria Indie rientrano anche tutti quei prodotti realizzati da studi indipendenti e con risorse limitate.

Un esempio tutto italiano è sicuramente *Gatta Cenerentola* (2017) prodotta da MAD Entertainment con appena 1,2 milioni di euro [41].

6.2.2 Organizzazione del personale e della pipeline di lavoro

L'organizzazione della pipeline di lavoro e del personale impiegato dipende strettamente dalla dimensione della produzione e quindi dal budget che si ha a disposizione. Produzioni Major possono contare su una struttura e su un'organizzazione solida. In questa tipologia di produzioni tipicamente si contano impiegati centinaia di professionisti, e grazie a contratti lavorativi di breve durata, risulta facile assumere nuovo personale, prolungarne o terminarne il contratto, a seconda delle necessità produttive. Questo, per l'azienda, offre enormi vantaggi, in quanto da un lato ha a disposizione una struttura di partenza molto solida, composta di uffici e attrezzature varie, classica di aziende multinazionali, e dall'altro lato si ha l'agilità e la flessibilità che caratterizzano le start-up.

Nelle produzioni ad alto budget la pipeline di lavoro è estremamente suddivisa in settori specifici, detti anche dipartimenti; tali fasi di produzione procedono almeno a livello teorico consecutivamente, nella pratica si verifica solitamente una parziale sovrapposizione di alcune fasi in quanto spesso diventa necessario apportare modifiche ed aggiustamenti e ciò comporta di dover tornare alla fase precedente. Data la suddivisione netta della pipeline, anche il personale impiegato sarà, tendenzialmente, specializzato in un unico settore.

La pipeline di produzione, escludendo la preproduzione dove l'idea del prodotto prende forma e la post-produzione dove il prodotto viene "impacchettato" per la distribuzione, si può riassumere nelle seguenti fasi:

Modellazione 3D in questa fase l'animatore realizza i modelli 3D di ambienti e personaggi a partire da delle reference, tipicamente tramite superfici geometriche poligonali. Programmi tipicamente utilizzati sono Autodesk Maya e 3D Studio Max. Un'alternativa open source, come vedremo nel dettaglio nel capitolo 8.3, è Blender.

Texturing in questa fase un texture artist crea e applica le texture a modelli e personaggi. Si vanno a definire i colori e le superfici degli oggetti modellati precedentemente.

Rigging in questa fase si vanno a realizzare i rig per tutti i personaggi. Si vanno a preparare i modelli per la fase di animazione, si crea l'armatura o scheletro e la si applica ad un determinato personaggio, andandone a definire le possibili deformazioni.

Animazione in questa fase l'animatore dà vita agli oggetti e ai personaggi presenti nella scena. Consiste in una delle fasi di produzione che richiedono più tempo e che risultano determinanti per il risultato finale.

Simulazione della fisica in questa fase si vanno ad animare tutte quelle componenti che sono soggette alla fisica e che quindi richiedono, solitamente, elevate risorse computazionali per essere simulate. Esempi di simulazione avvengono per i vestiti, capelli, acqua, folle, sistemi particellari ecc.

VFX sebbene molti elementi possano essere gestiti tramite simulazione della fisica, a volte risulta conveniente ottenere lo stesso risultato andando ad utilizzare effetti visivi in modo da limitare la complessità computazionale.

Lighting in questa fase si va a definire ed applicare l'illuminazione alla scena. Tramite l'illuminazione è possibile creare una specifica atmosfera, dare profondità di campo all'inquadratura e rendere un soggetto più evidente rispetto ad un altro.

Rendering in questa fase si va a generare quello che sarà l'output finale della pipeline di produzione. Tipicamente si va a creare un file immagine per ogni frame; per un maggior controllo in fase di post-produzione (compositing) il frame può essere suddiviso in differenti livelli.

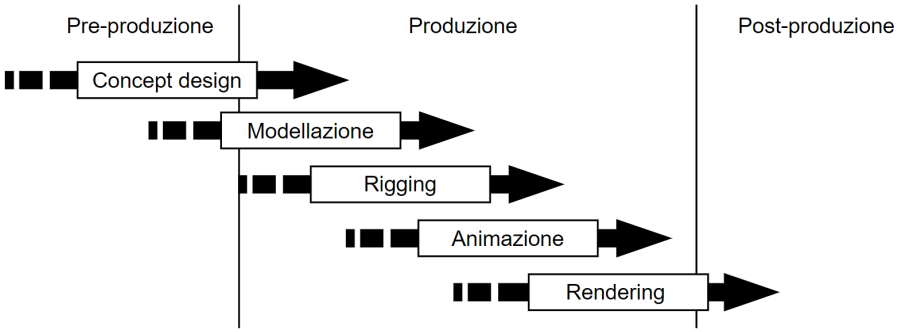


Figura 6.1: Fasi principali della pipeline di produzione per un prodotto di animazione 3D.

Nelle produzioni Major, ciascuna delle fasi sopra descritte, vedrà impiegati degli specifici professionisti. Infatti, il personale risulterà estremamente specializzato su uno specifico compito. Questo garantisce da un lato all'azienda di avere professionisti con competenze elevate, e dall'altro consente ai professionisti di non dover imparare una moltitudine di programmi e di compiti ma di specializzarsi in uno solo. Inoltre, ciò consente di attribuire a ciascun lavoratore un grado di specializzazione per un determinato settore. I gradi di specializzazione vanno dallo stagista, al grado *junior*, *senior*, fino ad arrivare al *supervisor* e al capo dipartimento.

Una produzione indipendente, il cui budget sarà limitato, difficilmente si troverà nelle condizioni di poter assumere un numero elevato di professionisti specializzati per ciascuna fase. Per questo motivo all'interno di questo contesto si preferisce la figura del *generalist*, ovvero un professionista che ha buone competenze per tutte, o quasi, le fasi della pipeline di produzione. Questo consente di avere a libro paga un numero più ridotto di lavoratori, tuttavia richiede una necessaria rivisitazione della pipeline di lavoro, andando a unire differenti fasi e andando a ridurre o eliminarne altre. Tutto ciò implica, nella maggior parte dei casi, la realizzazione di un prodotto meno "complesso" rispetto a quelli realizzati da studi major.

Questo, in qualche modo, spinge gli studi indipendenti a trovare strade alternative per la realizzazione del prodotto. Tale contesto, quindi, è tipicamente caratterizzato da un approccio altamente sperimentale che porta a trovare strumenti, tecniche e linguaggi innovativi. Infatti, la riduzione di complessità citata prima non comporta di per sé una invalidità del risultato finale, anzi, non sono rari i casi in cui l'ingegno tecnico ed artistico abbia premiato un'opera indipendente. Un esempio evidente è il lungometraggio animato, già sopra citato, *Gatta Cenerentola* che tra i molti riconoscimenti ricordiamo due David di Donatello nella 63-esima edizione per "Miglior produttore" e "Migliori Effetti Digitali", e nel 2018 il Nastro d'Argento "Per la qualità, l'innovazione e il coraggio produttivo" [42].

Come già riportato precedentemente, il seguente lavoro di Tesi è stato svolto in collaborazione con l'azienda torinese *Robin Studio S.r.l.s.* per andare ad individuare le migliori tecniche e il miglior workflow per la realizzazione delle animazioni facciali per una serie di animazione chiamata *Reverie Dawnfall*. Quindi, il contesto dentro il quale il mio lavoro e le mie ricerche sono state svolte, è completamente ascrivibile ad una produzione indipendente. Questo, come vedremo meglio nella parte II, ne ha determinato l'intero sviluppo ed esito.

Parte II

Produzione Indipendente

Dopo aver discusso, nella parte I del documento, dello stato dell'arte nel campo dell'animazione facciale e aver introdotto tecniche, strumenti, definizioni e termini utili in questo settore, in questa seconda parte del documento si andrà a trattare in maniera esaustiva quella che è stata la componente sperimentale di questo lavoro di tesi. Andando da prima ad individuare in maniera chiara il contesto dentro il quale il lavoro di tesi è stato svolto, definendo quindi di cosa tratta il progetto *Reverie Dawnfall* e la situazione dalla quale questo lavoro è partito (capitolo 7). Dopo di che verrà trattato quello che è stato il mio personale apporto al progetto di *Reverie Dawnfall* e più in generale alla community di Blender nel campo dell'animazione facciale, andando a definire gli strumenti e le tecnologie già esistenti, introdotte nel workflow produttivo, e gli strumenti appositamente creati all'interno di questo lavoro (capitoli 8, 9 e 10), ponendo particolare attenzione alla descrizione di *ReveRig* (capitolo 11).

Capitolo 7

Reverie Dawnfall

7.1 Studio Robin

Il lavoro di Tesi è stato svolto in collaborazione con *Robin Studio S.r.l.s.*, uno studio di produzione e animazione con sede a Torino co-fondato nel 2017 da Riccardo Antonino, docente del corso di Effetti Speciali presso il Politecnico di Torino e relatore di questa tesi. Da quanto riportato nella cartella stampa di Reverie Dawnfall [43]:

> L'obiettivo di Robin, crasi di "Robe Incredibili", è quello di rendere straordinaria la storia di ogni oggetto o persona che viene rappresentata sullo schermo. Per farlo, lo studio sperimenta tecnologie all'avanguardia per la produzione e la post produzione. Lo studio fa della comunicazione culturale il suo core business e cura la comunicazione d'immagine e le video-scenografie dell'artista Arturo Brachetti e collabora con enti museali torinesi quali il Museo Egizio e la Pinacoteca Agnelli, oltre che alla produzione di cortometraggi e spettacoli teatrali indipendenti

All'interno di questo contesto totalmente indipendente nasce e si sviluppa il progetto *Reverie Dawnfall.*

7.2 Progetto Reverie Dawnfall

All'interno della cartella stampa del progetto [43] la serie animata *Reverie Dawnfall* è descritta nel seguente modo:

> *Reverie Dawnfall*, ovvero "Il calare dell'alba su un sogno ad occhi aperti" è una serie d'animazione 3D cyberpunk che pone l'accento sulle tematiche dei disturbi neuro-psichiatrici, dell'esplorazione spaziale e del reale, dell'ecologia e le terribili conseguenze dell'ignorare lo sviluppo sostenibile. Protagonista della storia è un gruppo di ragazzi, disabili come la quasi totalità della popolazione mondiale, alle prese con un interrogativo che è al contempo letterale e filosofico: e se questo non fosse il mondo giusto? La serie è ambientata su di un pianeta dall'ecosistema al collasso e in rotazione sincrona rispetto alla propria stella. Un emisfero è quindi perennemente in ombra e uno illuminato. La città teatro della vicenda sorge sulla linea di demarcazione tra le due zone ed è pertanto in perenne illuminazione crepuscolare. Questo dà vita ad una palette di colori che si distacca sia dal cupo notturno dei cyberpunk stile *Blade Runner*, sia i saturi toni caldi dei post-apocalittici alla *Mad Max*.

Il progetto nasce nel 2017 a cura dello studio Robin, in un contesto totalmente indipendente, infatti, al momento è auto-prodotto dallo studio stesso. «Questa serie nasce quindi dal forte desiderio di realizzare un prodotto multimediale animato fortemente competitivo nel panorama televisivo nazionale, con l'aspirazione di espandersi anche al di fuori del confine del nostro Paese»[32]. Al momento attuale, la serie animata prevede un arco narrativo che si va a delineare attraverso tre stagioni, composte da una decina di episodi ciascuna della durata di circa 20 minuti, che vedono come protagonista una giovane e brillante studentessa di etimologia di nome *Nadya Sinkamen*. La prima stagione è già in fase di scrittura, infatti, oltre averne redatto la storyline generale, sono già state ultimate le sceneggiature dei primi episodi, mentre le restanti sono in fase di stesura.

7.2.1 Situazione di Partenza

Come indicato precedentemente, il progetto di *Reverie Dawnfall* nasce nel 2017 con il lavoro di tesi di Melissa Coarezza e Michele Cannata. In questa prima fase è stata finalizzata l'idea della serie animata e sono state effettuate le prime ricerche e i primi lavori relativi alla modellazione dei personaggi [10] e alle tecniche di animazione tramite motion capture [32]. Dal lavoro dei due colleghi è stata realizzata, inoltre, una prima versione del teaser trailer della serie animata.

Nel 2019, dato l'avanzamento tecnologico e le nuove potenzialità rese disponibili dalla nuova versione di Blender (versione 2.8), lo studio Robin ha deciso di formare un nuovo gruppo di lavoro con lo scopo di effettuare ulteriori sperimentazioni e ricerche in diversi campi del settore della computer animation. Come risultato di questo lavoro è stata realizzata una nuova versione del teaser trailer di *Reverie Dawnfall*. In questa seconda fase, il cui lavoro è alla base di quanto sviluppato per questa tesi, hanno preso parte diversi colleghi che hanno approfondito, tramite i loro lavori, differenti aspetti e problematiche caratteristiche di una produzione indipendente.

Il lavoro da loro svolto consiste in: modellazione degli ambienti 3D svolto da Ilaria Ginepro [44], illuminazione e shading procedurale svolto da Giacomo Balma e Andrea Lorusso [45], rigging di un personaggio orientato al motion capture di Stefania Abate [33] e simulazione di vestiti e capelli di Nicola Figari Barberis [46].

Da questa seconda fase, quindi, è emerso il materiale e gli strumenti con i quali il seguente lavoro di tesi è incominciato. In particolare, i personaggi principali del trailer, *Nadya* e *Jameela*, che risultavano già riggati come descritto in [33]. Per quanto riguarda l'animazione facciale, e quindi ciò che più è inerente a questo lavoro di tesi, le ricerche effettuate nelle fasi precedenti possono essere trovate nei seguenti documenti di tesi [32, 33]. Da questi emerge che già nel 2017

il problema di come l'animazione facciale dovesse essere gestita era stato per lo meno considerato. Con il lavoro di Coarezza è stato individuato il metodo con il quale andare ad animare il corpo dei personaggi della serie. In particolare, data la natura indipendente del progetto, si è deciso di utilizzare per l'intera produzione della serie animata il sistema di *Motion Capture* inerziale fornito dall'azienda Rokoko, tramite l'utilizzo delle loro Smartsuit Pro.

Da questa scelta è nata l'idea di utilizzare sistemi di motion capture anche per andare ad animare il volto dei personaggi. Dopo un primo tentativo fallimentare nel 2017 che prevedeva l'utilizzo di una GoPro con Faceware Technologies, nel 2019 per il secondo trailer si è deciso di fare affidamento su quello che allora era il neonato sistema di motion capture facciale di Rokoko. Tuttavia, il lavoro svolto non era abbastanza soddisfacente, tanto che, all'interno del secondo trailer, non sono presenti scene di dialogo e le animazioni facciali presenti risultano non completamente all'altezza del progetto. I motivi che principalmente limitavano il potenziale del sistema di performance capture di Rokoko, come vedremo approfonditamente in seguito, erano relativi alla mancanza di un'interfaccia che permettesse una facile manipolazione dei dati catturati, e la difficoltà nel dover andare a creare tutte le blendshape necessarie.

Per queste ragioni Robin Studio, nella persona di Antonino Riccardo, relatore di questa tesi, ha deciso di rivolgersi a me per andare ad effettuare ricerche più approfondite e ulteriori sperimentazioni nel campo dell'animazione facciale, con l'obiettivo di individuare le tecnologie, i sistemi e il workflow più adeguati da inserire all'interno della pipeline di produzione già sviluppata dai miei colleghi. È in questo contesto, quindi, che ha avuto inizio questo progetto di tesi.

Tale contesto è stato determinante all'interno di tutto il processo di lavoro. Ad influire maggiormente sul risultato finale rientra sicuramente la tipologia del contesto di lavoro, caratterizzato da una forte carattere indipendente e quindi anche caratterizzato da una necessaria flessibilità. Ma determinante è stato entrare all'interno di un progetto già in corso d'opera, infatti è stata fondamentale una fase iniziale di apprendimento della pipeline di lavorazione già individuata e una fase di approfondimento del lavoro che fino ad allora era stato svolto. Questo ha creato dei vincoli e dei paletti sui quali si è costruito il seguente lavoro. Alcuni esempi di paletti da rispettare riguardano il programma utilizzato, Blender, e come i file di progetto sono stati organizzati al suo interno.

7.3 Individuazione di Obiettivi e Limiti

Dopo aver definito nella parte I del documento le tecniche che sono alla base dell'animazione facciale nel settore della computer animation, è arrivato ora il momento di individuare quelle che meglio si integrerebbero all'interno della pipeline di produzione della serie animata *Reverie Dawnfall*. Come già indicato nel capitolo 6, i fattori da considerare nella scelta sono principalmente relativi alla tipologia di prodotto e alla tipologia di contesto.

Tipologia di prodotto Come riportato precedentemente, il lavoro di tesi è rivolto principalmente alla serie animata *Reverie Dawnfall*. Tale prodotto, sebbene per i temi trattati non sia direttamente rivolto ad un pubblico giovanissimo, è caratterizzato da uno stile grafico di stampo fumettistico, e i personaggi risultano stilizzati sia nelle forme che nelle texture. Per avere un'idea più chiara dello stile utilizzato si può fare riferimento alla figura 7.1 nel quale vengono presentati alcuni frame estratti dall'ultimo trailer realizzato (2020). Risulta evidente, quindi, che data la

natura "cartoon" del prodotto, non sarà necessario andare ad utilizzare metodi sofisticati per ricreare un'animazione facciale del tutto realistica, anzi l'animazione facciale sarà più improntata sullo "stile Pixar" andando quindi a fare uso dei dodici principi dell'animazione, come visto nella parte I.

Tipologia di contesto Da quanto visto ad inizio capitolo 7, il progetto è nato ed è portato avanti da uno studio di animazione indipendente. Ciò significa che le risorse a disposizione sono limitate, sia in termini di budget sia in termini di personale e di competenze impiegati. Questo pone dei vincoli evidenti sulle tecniche e sugli strumenti che sarà possibile adottare.

Figura 7.1: Fotogrammi tratti dal teaser trailer di *Reverie Dawnfall*.

Dall'individuazione della tipologia del prodotto e del contesto, le esigenze che emergono possono essere così riassunte:

- Animazione facciale stilizzata ("stile Pixar"), facendo riferimento ai 12 principi dell'animazione, che si integri al meglio con l'animazione del corpo dei personaggi effettuata con sistemi di mocap;
- Limitare il più possibile le risorse impiegate nell'intero processo di animazione facciale (budget, tempo, personale, competenze, software e hardware).

Andare ad individuare il miglior workflow che soddisfacesse tutte le esigenze è stato un compito arduo, in quanto queste risultano in parte in contrasto tra di loro. Realizzare un'animazione "stile Pixar" richiede tipicamente un approccio basato sui keyframe e, come abbiamo visto nel capitolo 5.1.1, la quantità di lavoro necessario per ottenere un risultato soddisfacente è incredibilmente elevata, inoltre richiede animatori con ottime competenze in materia.

Un approccio del genere risulta quindi impensabile all'interno di una produzione indipendente, in quanto, date le lunghe tempistiche, si verificherebbe un aumento inevitabile dei costi. Inoltre, risulterebbe difficile avere a disposizione un animatore specializzato che si occupi solamente di quel compito, mentre sarebbe più comune che ad occuparsi dell'animazione sia una figura generalista, che seppur competente, potrebbe non garantire un risultato soddisfacente.

Dall'altra parte un approccio basato sui sistemi di motion capture potrebbe portare significativi vantaggi nel velocizzare il processo, tuttavia sono da escludere tutte quelle tecniche che richiedono strumentazione troppo costosa e che richiedono personale tecnico apposito per garantirne il corretto funzionamento. Si vanno quindi ad escludere le tecniche di Mocap marker-based, in quanto, oltre all'attrezzatura costosa, richiedono anche tempistiche di preparazione importanti, cosa non accettabile all'interno di un contesto indipendente in cui la flessibilità è fondamentale.

La soluzione è nata basandosi sulle ricerche svolte dai miei colleghi per l'animazione del corpo dei personaggi. Come già riportato, si è scelto di utilizzare delle Smartsuit Pro di Rokoko che consistono in un sistema di motion capture di tipo inerziale. Da questo punto è diventato naturale per i miei colleghi fare utilizzo del sistema di mocap facciale offerto dalla stessa azienda, seppur con risultati incerti. Seguendo la stessa linea di pensiero è stata individuata nelle tecniche di motion capture facciale markerless parte della soluzione. Infatti, tali tecnologie consentono di velocizzare notevolmente il processo di animazione, senza però risultare economicamente insostenibili come le tecniche marker-based.

Però, come verrà spiegato nel capitolo 9, l'utilizzo di tecniche di mocap facciale non è privo di problematiche; da un lato la precisione di cattura delle espressioni facciali è limitata ma, come abbiamo visto prima, per noi è una limitazione parziale in quanto non ricerchiamo il realismo, dall'altro lato non sono pensate per realizzare animazioni stilizzate.

Oltre a queste considerazioni, è stato necessario identificare delle tecniche che bene si integrassero con la pipeline di lavoro già sviluppata e con gli strumenti di lavoro precedentemente individuati. Ovvero le tecniche di animazione facciale individuate si devono integrare completamente con l'utilizzo di Blender, e devono consentire, qualora fosse necessario, di apportare modifiche al loro funzionamento in modo da adattarle meglio al processo produttivo di *Reverie Dawnfall*.

Date le premesse sopra riportate è stato possibile individuare le tecnologie e gli strumenti da inserire all'interno del workflow per l'animazione facciale di personaggi stilizzati per un prodotto di animazione indipendente. Nei capitoli 9, 10 e 11 verranno presentati i risultati della mia ricerca, per ogni strumento e tecnica individuata ne verranno riportati potenzialità e limiti e come quest'ultimi sono stati superati. In particolare nel capitolo 8 si introducono i principali software utilizzati nella realizzazione di *Reverie Dawnfall*; il capitolo 9 tratta gli strumenti per l'animazione facciale, viene presentata la tecnologia ARKit, le applicazioni che su di essa si basano e l'Add-On di Blender Faceit; il capitolo 10 discute le tecniche e gli strumenti individuati per semi-automatizzare l'animazione della lingua; infine nel capitolo 11 viene presentato l'Add-On di Blender *ReveRig* appositamente creato per offrire all'animatore una serie di strumenti utili per ottimizzare l'intero workflow, mantenendo al contempo l'attenzione verso le esigenze del mercato internazionale, in particolar modo sulla community di Blender.

Capitolo 8

Strumenti principali

In questo capitolo introduciamo i principali software che verranno utilizzati all'interno della pipeline di lavoro. In particolare, si vanno a presentare Blender e Papagayo, i due software sui quali questo lavoro di tesi si appoggia maggiormente. Sebbene all'interno della realizzazione di un'animazione facciale intervengano anche altri strumenti, in questo capitolo si vanno ad introdurre solo quei programmi che ci hanno consentito di apportare modifiche sostanziali al codice sorgente in modo tale da adattarli nel migliore dei modi alle esigenze della produzione di *Reverie Dawnfall*. La possibilità di poter modificare parte del funzionamento di tali programmi è garantita dalla loro natura *open source*.

Per maggior chiarezza, prima di introdurre i due strumenti sopra citati, si definiscono alcuni termini che sono ricorrenti all'interno di questo documento di tesi.

8.1 Open source

Letteralmente "sorgente aperta", indica un software non protetto da copyright, il cui codice sorgente è rilasciato con una licenza che lo rende modificabile da chiunque[1]. In informatica, il codice sorgente è quella parte del software che i programmatori possono manipolare per modificare il comportamento del programma.

Oggi il significato del termine *open source* è utilizzato anche in modo più generico, per definire una filosofia e un sistema di valori che celebrano lo *scambio aperto*, la partecipazione collettiva, la trasparenza, la meritocrazia e lo sviluppo della comunità.

8.2 GNU General Public License

La GNU General Public License (spesso indicata con l'acronimo GNU GPL) è una licenza fortemente *copyleft* per *software libero* stesa originariamente nel 1989 da Richard Stallman per il

[1]Open source. (Consultato il 26/02/2021). In *Vocabolario on line - Treccani*. URL: https://www.treccani.it/vocabolario/open-source/#:~:text=open%20sources%20%E2%80%B9...,utenti%20e%20quindi%20liberamente%20modificabile.

sistema operativo GNU. Essa consente agli utenti finali di utilizzare, condividere e modificare il software da essa protetta [47].

Con software libero (*free software*) la Free Software Foundation (FSF) si riferisce al concetto di libertà e non al prezzo. Per questo motivo è consentito distribuire il software anche dietro pagamento, tuttavia la GPL garantisce che chiunque possa ricevere il codice sorgente. Un aspetto cruciale del software libero è che garantisce agli utenti di cooperare tra loro.

È una licenza copyleft, questo vuol dire che le opere derivate possono essere distribuite solo sotto gli stessi termini di licenza. Questo garantisce che tutte le versioni derivate che saranno distribuite dovranno essere libere. In questo modo si evita di trovarsi a competere con una versione modificata del proprio lavoro che sia diventata proprietaria.

La GPL non obbliga a pubblicare una versione modificata. Ognuno è libero di apportare modifiche al software e di utilizzarlo privatamente senza mai renderlo pubblico, questo vale anche per le organizzazioni e le società. Però, se la versione modificata viene resa pubblica la GNU GPL obbliga a rendere disponibile il codice sorgente del programma modificato e di rilasciarlo a sua volta sotto GPL.

La FSF detiene i diritti di copyright sul testo della GNU GPL ma non sul software da essa coperto. La sua versione più recente è *GNU GPL v3* (29 giugno 2007).

Come vedremo di seguito, gran parte degli strumenti già esistenti che sono stati utilizzati e gli strumenti appositamente realizzati in questo contesto, ricadono sotto la GNU GPL. Questo ci ha consentito di adattare software esistenti alle specifiche esigenze legate alla tipologia di progetto e all'ambiente di lavoro indipendente, e hanno permesso di agevolare la sperimentazione caratterizzata da un processo *trial & error*.

8.3 Blender

Blender è una suite di creazione 3D gratuita, *open source* e multipiattaforma, disponibile per dispositivi Linux, Windows e Mac. Supporta l'intera Pipeline 3D – modellazione, rigging, animazione, simulazione, rendering, compositing e motion tracking oltre a video editing e game design [48].

Nasce nel 1995, inizialmente destinato a diventare il software 3D dello studio di animazione olandese NeoGeo co-fondato nel 1988 da Ton Roosendall. Nel 1998 Roosendall fondò una nuova azienda, chiamata Not a Number (NaN), dedicata allo sviluppo e alla vendita di Blender. Purtroppo, a causa di difficoltà economiche gli investitori decisero di chiudere il progetto. Fortunatamente il supporto della community convinse Ton Roosendall a dar vita alla fondazione no-profit Blender Foundation con lo scopo di continuare lo sviluppo di Blender e di rilasciarlo sul mercato sotto la licenza *GNU General Public License* [49]. Lo sviluppo del software non si è mai fermato e periodicamente vengono rilasciati aggiornamenti e nuove versioni. L'ultima versione disponibile, al momento della stesura di questo documento, è la *2.93*.

Perché Blender

La sua natura open source e la sua gratuità rendono Blender un ottimo strumento in produzioni a basso budget. Un errore che spesso si può commettere è associare il costo di un software al suo valore e le sue potenzialità effettive, ed è un errore in cui si può inciampare quando si parla di Blender. Sicuramente le versioni più vecchie del software (*2.79* e precedenti) avevano

alcune problematiche relative all'usabilità e facilità di utilizzo se comparate ad altri software 3D come Maya, Cinema 4D e ZBrush. Tuttavia, dal rilascio della versione *2.8* nel luglio del 2019, la distanza dagli altri software 3D si è costantemente ridotta. Blender è a tutti gli effetti uno strumento professionale che poco ha da invidiare ai suoi competitor.

La figura 8.1 rappresenta una comparazione delle ricerche via web per le principali DDC (Digital Content Creation suites) negli ultimi cinque anni. Le statistiche delle ricerche via web possono dare un'indicazione dell'attività della community online e della relativa popolarità. Come si può notare Blender conta la community più attiva e in continua crescita, mentre i competitor sono in leggera ma continua decrescita.

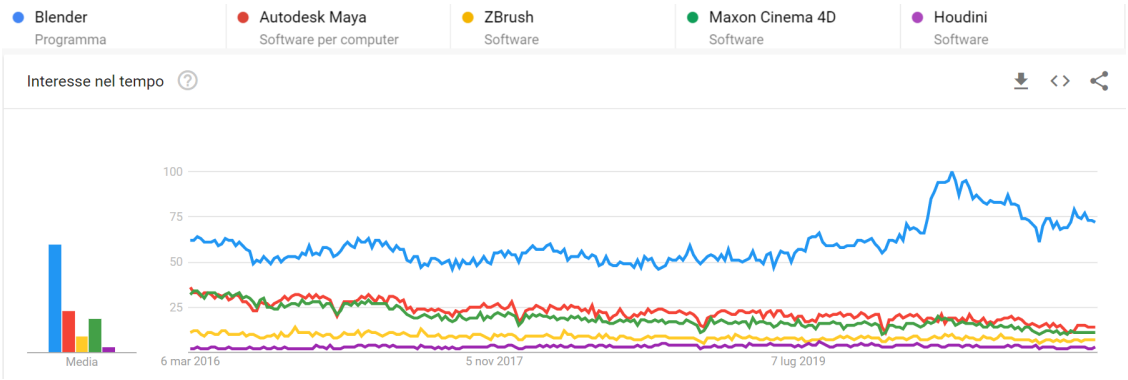


Figura 8.1: Confronto delle ricerche via web tra Blender e 4 delle principali DCC nel periodo marzo 2016 - marzo 2021. Fonte Google Trends (`https://trends.google.com/trends/`).

Un enorme vantaggio che offre Blender consiste nel fatto che è in grado di supportare l'intera pipeline 3D di produzione. Questo risulta incredibilmente vantaggioso all'interno di contesti indipendenti e a basso budget, dove tipicamente si va a premiare una figura *generalist* (con capacità orizzontali), piuttosto che professionisti specializzati verticalmente in un solo campo.

Tipicamente negli studi *Major*, dove, come abbiamo visto nel capitolo 6, le risorse sia a livello economico sia a livello di competenze risultano abbondanti, si hanno software dedicati per ogni fase della pipeline produttiva. Inoltre, anche le figure professionali sono estremamente specializzate su un'unica fase della pipeline; questa suddivisione del lavoro evita ai professionisti di dover imparare ad utilizzare in modo approfondito la moltitudine di software utilizzati all'interno dell'intera filiera.

Tuttavia, trasporre tale organizzazione all'interno di studi di animazione *Indie* risulta estremamente complesso a causa delle limitate risorse a disposizione. Per questo motivo diventa vantaggioso adottare un software 3D in grado di lavorare sull'intera pipeline di produzione, così da limitare i costi dovuti all'acquisto/abbonamento di un numero elevato di software e permettere a figure generaliste di poter lavorare agevolmente sull'intera filiera senza dover imparare differenti strumenti. La gratuità di Blender offre sicuramente un vantaggio aggiuntivo.

Infine, forse l'elemento più interessante consiste nella sua natura open source. Ciò significa che il codice sorgente del software è accessibile a tutti, consentendo potenzialmente a chiunque di introdurre modifiche nel programma in modo da adattarlo alle singole necessità ed esigenze.

Sono proprio questi motivi che hanno spinto i colleghi che mi hanno preceduto, a scegliere Blender come software principale durante la fase di produzione del teaser trailer di *Reverie*

Dawnfall, ed è per gli stessi motivi che ritengo che tale strumento debba essere utilizzato per l'intera produzione della serie animata.

Come vedremo nel capitolo 11, la natura open source di Blender verrà sfruttata per creare nuove funzionalità appositamente pensate per agevolare il lavoro di animazione facciale.

8.3.1 Blender API

Come altri software 3D dedicati, Blender dispone di un Application Programming Interface (API) che consente a tutti gli utenti di automatizzare funzioni di accesso o processi che non sono esposti all'interfaccia utente (UI) e di creare strumenti personalizzati. Blender utilizza Python 3.x per le Blender Python API, ha un interprete Python incorporato che viene caricato all'avvio del programma e rimane attivo mentre Blender è in esecuzione. L'interprete Python viene utilizzato per disegnare l'interfaccia utente e per strumenti interni. Blender inoltre fornisce i suoi moduli Python come `bpy` e `mathutils` al suo interprete, ciò consente di importarli facilmente all'interno di qualsiasi script per avere così accesso a tutti i dati, classi e funzioni di Blender [50, 51]. Questo consente a tutti gli effetti di adattare il software alle proprie esigenze.

Nella documentazione Blender 2.91.0 Python API [50] vengono indicate tutte le funzionalità che possono essere realizzate tramite script.

- Modificare tutti i dati accessibili da interfaccia utente (Scene, Mesh, ecc.);
- Modificare le preferenze dell'utente, i temi e le key-map;
- Personalizzazione delle impostazioni per le esecuzioni dei tool (strumenti);
- Creazione di elementi di UI come pannelli, menu e intestazioni;
- Creazione di nuovi strumenti;
- Creazione di strumenti interattivi;
- Creazione di nuovi motori di rendering che si integrino con Blender;
- Iscrizione alle modifiche ai dati e alle loro proprietà;
- Definizione di nuove proprietà per i dati presenti in Blender;
- Creazione di geometrie tramite Python

e le funzionalità che (ancora) non sono presenti:

- Creazione di nuove tipologie di spazio
- Assegnazione di proprietà personalizzate a tutti i tipi.

I due modi più comuni per eseguire script Python in Blender sono o tramite il *Text Editor* interno a Blender o inserendo i comandi direttamente all'interno della console Python. Gli script in Python possono essere sia eseguiti singolarmente sia impacchettati come *add-on* per essere scaricati all'avvio e agevolare ulteriormente il loro utilizzo. Tipicamente l'opzione add-on viene intrapresa quando si realizzano funzionalità opzionali o quando si creano funzionalità complesse che richiedono molto codice.

Le differenze tra add-on e script creati internamente è che gli add-on devono contenere una variabile `bl_info` che viene letta da Blender per avere metadati come il nome dell'add-on, l'autore, la categoria e altro. Inoltre, queste informazioni vengono mostrate nella schermata *User Preferences* > *Add-on*. Quando gli add-on vengono pubblicati, devono essere condivisi sotto una licenza compatibile con la GPL, si è comunque liberi di vendere il servizio di download.

Sebbene il Text Editor interno a Blender risulti molto utile, quando si intendono realizzare funzionalità complesse può risultare più comodo utilizzare dei veri e propri ambienti di sviluppo.

L'estensione Blender Development[2] per l'IDE (Integreated Development Environmet) di Visual Studio Code (VSC) consente di collegarsi al processo interno di Python di un eseguibile di Blender (`blender.exe`) e utilizzare le API per eseguire script e caricare add-on. Questo permette di eseguire il debug sia dentro VSC che dentro Blender in tempo reale, rendendo così l'intero processo più reattivo e veloce.

8.3.2 Shape Key

In Blender ciò che fino ad ora sono state chiamate *blendshape* vengono indicate sotto il nome di *Shape Key* (chiavi di forma) [52]. Le Shape Key possono essere create tramite l'apposito pannello accessibile dalla tab *Object Data* delle Proprietà (es. la tab *Mesh* per oggetti avente mesh). Sebbene le shape key possono essere applicate a qualsiasi oggetto dotato di vertici, come mesh, lattice, curve e superfici, per semplificare la lettura si farà riferimento unicamente alle mesh.

Le chiavi di forma possono essere create manualmente tramite la manipolazione della mesh in *Edit Mode* o in *Sculpt Mode*. Tramite il parametro *Value* (influence) è possibile controllare l'intensità delle deformazioni effettuate sull'oggetto.

Non è consentito modificare la topologia della mesh, aggiungendo o rimuovendo vertici, per questo motivo è consigliato creare le shape key necessarie solo dopo aver definito in modo completo l'oggetto che si andrà a deformare. Al momento della creazione della prima shape key verrà automaticamente generata una shape key di nome *Basis* che, come dice il nome stesso, rappresenta lo stato dei vertici dell'oggetto nella posizione originale e verrà utilizzata come base alla quale le nuove shape key faranno riferimento.

In Blender le shape key relative ad una mesh vengono organizzate in una struttura a *stack*[3]. Lo stack può essere di tipo *Relative* (relativo) o *Absolute* (assoluto).

Relative è la tipologia più comune in quanto viene utilizzata per il controllo di muscoli, delle articolazioni e dell'animazione facciale. È basato sull'approccio "*delta*" visto nella sezione 3.2. Ogni shape key è definita in relazione alla forma *Basis* o ad un'altra shape key. Il risultato, chiamato *Mix*, è ottenuto come effetto cumulativo di ogni chiave di forma e del suo valore. Il parametro *Value* rappresenta il peso della fusione tra una shape key e la sua chiave di riferimento. Tipicamente tale valore va da 0 a 1, dove un valore di 0 indica un'influenza del 100% della chiave di riferimento mentre un valore di 1 indica un'influenza totale della shape key.

Absolute utilizzato principalmente per deformare un oggetto in differenti forme nel tempo. È basato sull'approccio "*whole-face*" visto nella sezione 3.2. Ogni shape key definisce la forma dell'oggetto ad uno specifico tempo, *Evaluation Time*, definito dal parametro *Value*. Il risultato è un'interpolazione tra due chiavi di forma consecutive dato l'Evaluation Time.

---

[2] L'estensione Blender Development in VS Code può essere scaricata al seguente indirizzo: https://marketplace.visualstudio.com/items?itemName=JacquesLucke.blender-development.

[3] In informatica, una struttura a stack è una lista lineare con elementi ordinati in cui è possibile effettuare, nella stessa estremità, due operazioni principali: *Push*, per aggiungere un elemento e *Pop* per rimuovere l'ultimo elemento aggiunto. Stack. (Consultato il 22/02/2021). *Enciclopedia on line - Treccani*. URL: https://www.treccani.it/enciclopedia/stack/.

Nel proseguimento di questo documento di tesi verrà utilizzato principalmente il termine *blendshape*, in modo da utilizzare una terminologia di più ampia portata; tuttavia, in situazioni in cui si faccia diretto riferimento all'utilizzo degli strumenti interni a Blender o laddove fosse necessaria maggiore scorrevolezza del testo, il termine blendshape verrà alternato con *shape key*.

8.3.3 Driver

I driver sono un modo per controllare i valori delle proprietà mediante una funzione o un'espressione matematica [53]. Ciò significa che è possibile slegare l'animazione di una proprietà dal numero di frame. I driver sono uno strumento molto potente, e quindi molto utilizzato, nella costruzione dei rig e nel determinare le trasformazioni applicate alle ossa e l'influenza delle shape key, dei vincoli e dei modificatori. Come vedremo nel capitolo 11, l'utilizzo dei driver sarà alla base del funzionamento dell'add-on *ReveRig*.

8.3.4 Linking

Questa funzione permette di riutilizzare materiali, oggetti e *data-block* provenienti da un altro file `.blend`. Questo consente di mantenere una migliore organizzazione dei file e di creare delle librerie condivise su più file di lavoro. Quando si effettua il *Link* si crea un riferimento ai dati presenti nel file d'origine, in questo modo tutte le modifiche apportate al file di origine verranno automaticamente applicate nei file in cui è stato creato il Link [54]. Tuttavia, non sarà possibile modificare direttamente l'oggetto a livello locale se non attraverso degli *Oggetti Proxy* (*Object* > *Relations* > *Make Proxy...*).

La creazione di un oggetto proxy consente di apportare modifiche localmente su un oggetto collegato ad una libreria esterna. In questo modo tali modifiche non si ripercuoteranno sul file originale ma rimarranno a livello locale. Tuttavia, le manipolazioni possibili rimangono limitate a quelle consentite dall'oggetto proxy creato.

Un esempio molto frequente riguarda modelli dotati di armatura (rig). L'armatura può essere convertita in oggetto `_proxy` e manipolata per animare un personaggio, ma non sarà possibile andare a modificare la mesh del personaggio stesso e quindi non si avrà la possibilità di accedere alle sue blendshape. Questo sistema permette di creare una struttura di file molto organizzata e permette di rendere ciascun file più leggero, in quanto gli oggetti presenti non sono copiati all'interno del file stesso ma sono dei link a librerie esterne.

Per questi motivi, si è deciso di adottare tale struttura per la produzione della serie animata *Reverie Dawnfall*. Nel capitolo 11 verranno approfondite le limitazioni che tale sistema si porta dietro e come tali limiti sono stati superati.

8.3.5 Creazione di un Add-On: concetti base

Per poter comprendere meglio come l'Add-On *ReveRig* sia stato realizzato, è necessario chiarire quali sono le componenti base che vanno a creare un add-on tipo. Di seguito verranno presentati i principali strumenti messi a disposizione da Blender e da Python, utilizzati per costruire nuove funzionalità. Non si ha comunque la pretesa di descrivere dettagliatamente tutte le possibili combinazioni che possono essere create con il codice ma ci si limiterà a darne una visione generale maggiormente improntata su quanto realizzato per ReveRig.

Prima di tutto bisogna specificare che un add-on altro non è che un modulo Python con dei requisiti che consentono a Blender di poterne leggere le informazioni e di poter creare delle interfacce grafiche[4]. Qualsiasi sia la finalità di un add-on, in esso vi sono alcuni elementi che risultano obbligatori, o per lo meno la loro mancanza rende l'intero codice non utilizzabile. Il codice 8.1 riporta uno degli esempi di add-on più semplice che si possa realizzare.

```
bl_info = {
  "name": "My Test Add-on",
  "blender": (2, 80, 0),
  "category": "Object",
}

import bpy

def register():
  print("Hello World")

def unregister():
  print("Goodbye World")

if __name__ == "__main__":
  register()
```

Codice 8.1: Codice base per un add-on in Blender

Da tale esempio si possono estrarre gli elementi indispensabili nella creazione di un add-on. In particolare, si ha la variabile `bl_info`, ovvero un dizionario contenente i metadati dell'add-on, come nome dell'autore, titolo, versione e descrizione dell'add-on, come già indicato nella sottosezione 8.3.1. Dopo di che ci sono due funzioni: `register` e `unregister`. La prima viene chiamata solo quando l'add-on viene abilitato ed è utilizzata per tutto ciò che dovrà essere registrato per far funzionare correttamente il codice, tipicamente operatori, pannelli e menu. La funzione `unregister`, invece, viene chiamata solo quando l'add-on viene disabilitato, e ha il compito di annullare la registrazione tutto ciò che è stato registrato nella funzione precedente. Per poter utilizzare le funzionalità messe a disposizione da Blender è necessario importare il modulo `bpy` (`import bpy`). Le ultime due righe del codice 8.1 servono per consentire all'utente di lanciare lo script direttamente dal *Text Editor* senza dover necessariamente installare l'add-on. Tipicamente viene inserito al fondo del codice.

L'esempio sopra riportato non ha un vero e proprio effetto su Blender; per creare nuove funzionalità tipicamente si fa utilizzo di apposite funzioni e di apposite classi. Le classi principalmente utilizzate nella creazione di *ReveRig*, oltre a quelle create per specifiche funzionalità interne al codice, sono di due tipi: *operatori* e *pannelli*. Le prime estendono quanto definito in `bpy.types.Operator` e consentono di creare e di eseguire le funzionalità desiderate. Le seconde estendono quanto definito in `bpy.types.Panel` e vengono utilizzate per definire l'interfaccia grafica che verrà mostrata all'interno di Blender. All'interno del codice si è deciso di nominare tali classi rispettando la seguente convenzione: `UPPER_CASE_{SEPARATOR}_mixed_case`, dove

[4]Fonte: Blender 2.92 Manual. (Ultimo aggiornamento il 25/02/2021). URL: https://docs.blender.org/manual/en/latest/advanced/scripting/addon_tutorial.html.

il separatore è composto da due lettere che indicano il tipo della classe: `OP` per gli operatori e `PT` per i pannelli (es DMANAGER_PT_Brows).

All'interno di un pannello, grazie all'utilizzo del metodo `draw()`, un operatore può essere rappresentato come un bottone. All'interno degli operatori e in particolar modo all'interno del metodo `execute()` è possibile chiamare tutte le funzioni necessarie per il corretto svolgimento dell'operazione desiderata.

Per maggiore chiarezza del codice, l'add-on ReveRig è stato organizzato in modo da facilitare sia la fase di sviluppo sia quella relativa alla successiva manutenzione. In particolare, si è deciso di dividere lo spazio di lavoro come segue (vedi il codice riassuntivo 8.2):

- ad inizio pagina viene definito il dizionario `bl_info` e vengono importati tutti i moduli necessari.
- la seconda sezione del codice comprende tutte le funzioni che sono state implementate per compiere le operazioni necessarie.
- la terza sezione inizia e si conclude con la definizione delle classi *Operator* e delle classi relative alla creazione di nuove proprietà.
- la quarta sezione racchiude tutte quelle classi di tipo *Panel* utilizzate per creare l'UI dell'add-on.
- in fondo al codice compaiono le funzioni `register` e `unregister`.

```
#Prima sezione: metadati e importazione dei moduli necessari
bl_info = {...}
import bpy, bmesh, math, re, time

#Seconda sezione: Metodi e Funzioni
def createDriversFunction(...): ...
def createDrivers(...): ...
def insertKeyFrames(...): ...
def copyFCurvesAnimation(...): ...
def lipsyncer(context): ...
def createTongueAnimation(...): ...
def remapTongueValue(...): ...
def insertKeyFrames_ActiveObj(...): ...

#Terza sezione: Definizione Operatori e classe MyProperties
class FR_OP_ShapekeySliderCreator(bpy.types.Operator): ...
class FR_OP_ShapekeyDriverCreator(...): ...
class ANIM_OP_CopyFCurvesAnimation(...): ...
class ANIM_OP_ImportPlotLipsyncer(...): ...
class ANIM_OP_RemapTongue(...): ...
class ANIM_OP_InsertRestPoseKeyFrame(...): ...
class ANIM_OP_InsertSingleKeyFrame(...): ...
class MyProperties(bpy.types.PropertyGroup): ...

#Quarta sezione: Definizione dei Pannelli per creare il Layout dell'add-on
class CSD_PT_CreateArmaturePanel(bpy.types.Panel): ...
class CA_PT_CopyFCurvesAnimation(...): ...
class CA_PT_ImportTongueAnimation(...): ...
class DMANAGER_PT_ui(...): ...
class DMANAGER_PT_Parent(DMANAGER_PT_ui): ...
class DMANAGER_PT_Children(...): ...

```

```
#Ultima sezione: funzioni register e unregister
def register(): ...
def unregister(): ...
if __name__ == "__main__":
  register()
```

Codice 8.2: Organizzazione generale del codice realizzato per ReveRig

Oltre aver dato l'organizzazione all'impaginazione del codice come riportato nel codice 8.2, per migliorare ulteriormente l'usabilità del codice, sia durante l'intera fase di sviluppo ma anche per garantirne un facile utilizzo in seguito, si è deciso di utilizzare il *principio della singola responsabilità*. Ovvero ogni operatore racchiude al suo interno tutto ciò che è necessario per portare a termine un compito e questo soltanto. In questo modo è possibile attribuire con maggiore certezza le responsabilità di un eventuale problema nell'esecuzione del programma. Consente di testare in maniera più mirata le diverse componenti senza dover agire sul resto del codice, riducendo significativamente il lavoro necessario per la manutenzione dello stesso. Per la creazione di add-on con elevata complessità si può decidere di suddividere le diverse responsabilità in differenti moduli Python. Per l'add-on ReveRig, sebbene conti più di 2000 righe di codice, si è deciso di creare un unico modulo e di organizzarlo come visto nello schema 8.2. Questo ha permesso di utilizzare direttamente il *Text Editor* di Blender per sviluppare e testare il codice.

Una descrizione più completa ed esaustiva su come l'add-on ReveRig sia stato sviluppato e come possa essere utilizzato all'interno della pipeline di lavoro, verrà trattata nel capitolo 11.

8.4 Papagayo e Papagayo-NG

Papagayo è un programma di *lip-sync* (sincronizzazione labiale), sviluppato da Lost Marble, progettato per aiutare l'animatore 2D ad allineare i fonemi (in particolare le forme che assume la bocca) con l'audio registrato. Semplifica il processo di sincronizzazione labiale dei personaggi animati rendendo il processo molto intuitivo e semplice; è necessario digitare le parole pronunciate nell'audio (o copiarle/incollarle dal copione dell'animazione), quindi posizionarle correttamente sopra la forma d'onda dell'audio [55].

Il programma è nato principalmente per integrarsi con il software di animazione 2D Moho, ma è possibile esportare il risultato ottenuto come file `.dat` e quindi, grazie all'utilizzo di un add-on, è possibile utilizzarlo anche con Blender. Papagayo è gratuito e disponibile per Windows e Mac OS X. È licenziato sotto *GNU General Public License*, quindi è possibile scaricare il codice sorgente e apportare delle modifiche allo stesso, è inoltre possibile ridistribuire il software modificato a patto che venga reso disponibile il codice sorgente.

Papagayo è disponibile in due versioni, quella più vecchia la 1.2 e l'ultima realizzata, la versione 2.0b1. Tuttavia, quest'ultima è ancora in fase beta e, come si può evincere dalla figura 8.2 tratta dalla pagina GitHub dalla quale è possibile scaricare il codice sorgente[5], questa versione non è più stata modificata dopo il luglio 2014.

[5]Il codice della versione 2.0b1 del programma Papagayo può essere scaricato alla pagina GitHub avente il seguente URL: `https://github.com/LostMoho/Papagayo`. Allo stesso indirizzo è possibile consultare il grafico riportato in figura 8.2.

Figura 8.2: Grafico riportante i contributi apportati al master del programma Papagayo 2.0b1 nel periodo 13/07/2014 - 1/03/2021[5].

I mancati aggiornamenti fanno pensare che il progetto sia stato accantonato dagli sviluppatori originali. Infatti, dopo aver provato ad installare il programma, si sono verificati differenti problemi che ne hanno compromesso l'utilizzo.

Tuttavia, un'iniziativa indipendente nata con lo scopo di produrre animazioni 2D open source, Morevna Project, ha deciso di mantenere ed estendere Papagayo in una nuova versione chiamata *Papagayo-NG* [56]. Come è possibile evincere dalla figura 8.3, tratta dalla pagina GitHub del codice sorgente[6], in questo caso il programma è in continuo sviluppo e aggiornamento.

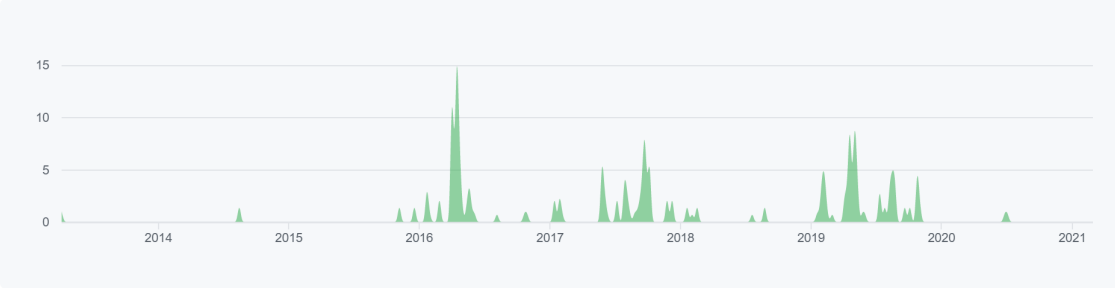


Figura 8.3: Grafico riportante i contributi apportati al master del programma Papagayo-NG nel periodo 07/04/2013 - 1/03/2021[6].

Per questo motivo si è scelto di utilizzare Papagayo-NG all'interno della pipeline di lavoro nella realizzazione della serie animata *Reverie Dawnfall*. Il suo funzionamento e la sua implementazione all'interno di questo progetto verranno approfonditi nel capitolo 10.

[6]Il codice sorgente del programma Papagayo-NG può essere scaricato alla pagina GitHub di Morevna Project avente URL: `https://github.com/morevnaproject-org/papagayo-ng`. Allo stesso indirizzo è possibile consultare il grafico riportato in figura 8.3.

Capitolo 9

Animazione facciale

In questo capitolo andremo a trattare il cuore del processo di animazione facciale individuato. Da quanto appreso nella parte I di questo documento, l'animazione facciale si può dividere in tecniche di manipolazione della geometria e tecniche di acquisizione e applicazione dati. Tali componenti sono strettamente legate tra loro tanto che la scelta di una tecnica influenza necessariamente anche la seconda.

Il seguente lavoro è partito dall'individuazione della tecnica di acquisizione dati; questa consiste nell'utilizzo di strumenti che fanno uso della tecnologia *ARKit* per effettuare un motion capture facciale di tipo markerless. Tale scelta, di conseguenza, ha determinato le tecniche di deformazione della geometria da andare ad applicare, infatti il mocap facciale con tecnologia ARKit si basa sull'utilizzo di *blendshape*.

Tale approccio, come vedremo nel dettaglio in seguito, ha posto dei limiti, che in parte sono stati superati, riguardanti la creazione e l'utilizzo delle blendshape stesse e riguardanti i dati raccolti dal sistema di performance capture. Infatti, tali sistemi non consentono l'acquisizione di dati rispetto al movimento della lingua, cosa che risulta un limite importante specialmente in situazioni in cui vi è un'inquadratura stretta sul volto del personaggio. Inoltre, si è reso necessario integrare a tale sistema degli strumenti che consentono una successiva manipolazione dei dati acquisiti, in quanto la necessità di apportare aggiustamenti all'animazione all'interno di una produzione seriale risulta elevata, sia perché i dati catturati potrebbero essere in parte compromessi, sia perché a livello registico potrebbero essere richieste delle modifiche.

9.1 Apple ARKit

Per rendere il processo di animazione facciale più veloce e più adatto ad un contesto indipendente si è deciso di adottare una tecnica di acquisizione dati basata su una tecnologia di motion capture markerless: *ARKit*. La tecnologia che si cela dietro tali strumenti rappresenta benissimo il progresso tecnologico nel settore della performance capture markerless in real-time. ARKit, infatti, si basa sul lavoro svolto da Soufien Bouaziz [37] per il software commerciale Faceshift, utilizzato in produzioni cinematografiche di alto livello come *Star Wars: Il risveglio della Forza* prima di essere acquisito da Apple Inc. nel 2015 [38, 39].

Nel 2017, Apple lancia sul mercato l'iPhoneX, smartphone dotato per la prima volta di una camera RGB-D, chiamata anche *TrueDepth camera*, appositamente pensata per integrarsi con il

neonato sistema di realtà aumentata di Apple [57]. Con l'uscita sul mercato della generazione iPhone 12 Pro, e la versione ARKit 4, ulteriori miglioramenti sono stati effettuati sia a livello hardware che software, infatti tali dispositivi hanno integrato un LiDAR Scanner che consente di catturare informazioni sulla profondità più dettagliate.

Come funziona

Tramite l'*Apple ARKit Development kit*, i programmatori di applicazioni *iOS* sono in grado di accedere e di utilizzare i dati acquisiti e messi a disposizione da ARKit per produrre esperienze di Realtà Aumentata (AR), tramite i dispositivi compatibili. Per quanto riguarda il mocap facciale il sistema ARKit, tramite la camera fontale dei dispositivi idonei, è in grado di registrare informazioni relative alla posizione, orientamento, topologia e all'espressione di un determinato volto [58]. Siccome posizione e orientamento del volto, nel nostro caso sono già catturati tramite tuta inerziale SmartSuit Pro, la parte che maggiormente ci interessa è relativa alla cattura delle espressioni facciali.

ARKit individua delle specifiche caratteristiche del volto, basandosi sul FACS, e tramite dei coefficienti appositi, va a descrivere come queste aree del volto vengono modificate e deformate, ovvero le *blendshape*. I dispositivi idonei all'utilizzo di ARKit sono in grado di registrare in tempo reale le espressioni facciali di un utente e tali espressioni vengono gestite secondo un modello basato sulle blendshape.

In pratica, fa uso di un dizionario in cui le chiavi sono i nomi delle blendshape e il valore è un numero floating point che va da 0 (neutrale) a 1 (massimo movimento) [59]. In totale ARKit suddivide le espressioni facciali in 52 blendshape, ciascuna delle quali andrà ad influenzare una specifica area del volto e andrà ad identificare uno specifico movimento (tabella 9.1 e figura 9.1).

Tabella 9.1: Le 52 blendshape definite da ARKit [59]

Area del volto	Nomi delle blendshape
Occhi (14)	eyeBlinkLeft, eyeLookDownLeft, eyeLookInLeft, eyeLookOutLeft, eyeLookUpLeft, eyeSquintLeft, eyeWideLeft, eyeBlinkRight, eyeLookDownRight, eyeLookInRight, eyeLookOutRight, eyeLookUpRight, eyeSquintRight, eyeWideRight
Mascella (4)	jawForward, jawLeft, jawRight, jawOpen
Bocca (23)	mouthFunnel, mouthPucker, mouthLeft, mouthRight, mouthRollUpper, mouthRollLower, mouthShrugUpper, mouthShrugLower, mouthClose, mouthSmileLeft, mouthSmileRight, mouthFrownLeft, mouthFrownRight, mouthDimpleLeft, mouthDimpleRight, mouthUpperUpLeft, mouthUpperUpRight, mouthLowerDownLeft, mouthLowerDownRight, mouthPressLeft, mouthPressRight, mouthStretchLeft, mouthStretchRight
Sopracciglia (5)	browOuterUpRight, browDownRight, browInnerUp, browDownLeft, browOuterUpLeft
Guance (3)	cheekPuff, cheekSquintLeft, cheekSquintRight
Naso (2)	noseSneerLeft, noseSneerRight
Lingua (1)	tongueOut

La fase di trasferimento dei dati raccolti dal sistema di mocap al modello da animare, come abbiamo già visto nella sezione 5.3.4, si chiama *retargeting*. Con l'utilizzo della tecnologia ARKit, il retargeting deve essere gestito, per l'appunto, tramite un sistema di blendshape. Ciò significa che per ottenere la resa migliore, il modello che vogliamo andare ad animare deve essere costituito

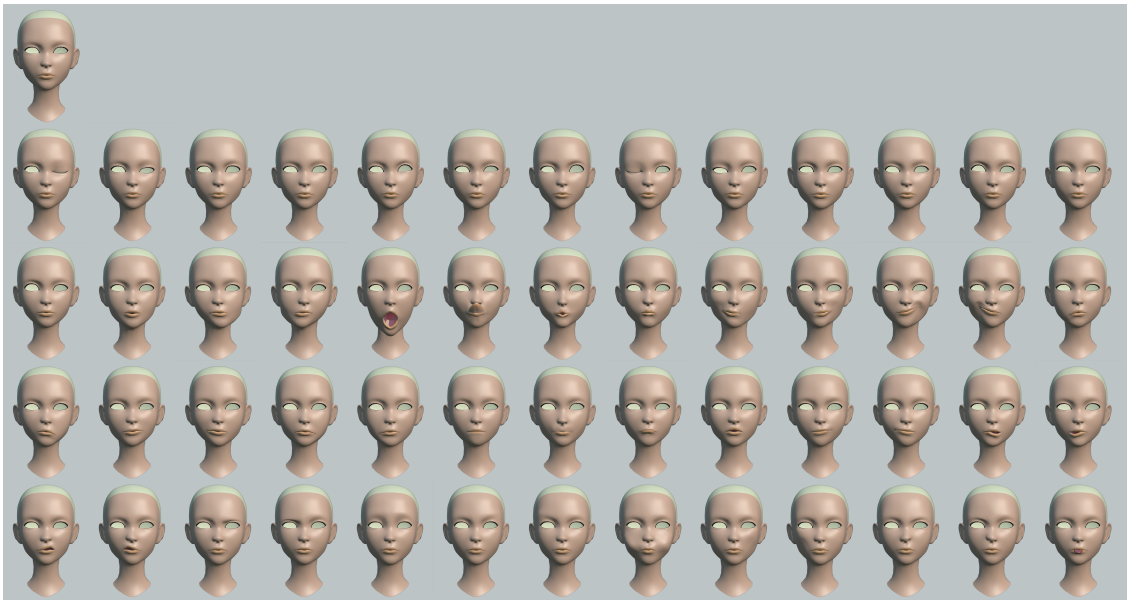


Figura 9.1: Le 52 blendshape definite da ARKit

dalle stesse blendshape che il sistema di mocap è in grado di registrare, ovvero deve essere dotato delle 52 blendshape riportate nella tabella 9.1. Per funzionare in modo corretto, tali blendshape, oltre ad avere la giusta nomenclatura, dovranno corrispondere nella maniera più corretta possibile alle blendshape individuate da ARKit (figura 9.1).

Prendiamo ad esempio la blendshape `eyeBlinkLeft`, che già dal nome si intuisce corrispondere alla chiusura dell'occhio sinistro; se nel nostro modello la blendshape avente lo stesso nome invece di prevedere una corretta deformazione dell'occhio sinistro, per assurdo andasse a deformare parte della bocca, allora si verificherebbe un'incoerenza tra dato acquisito e deformazione effettiva.

Sebbene questo sia un esempio limite, non risulta invece raro che nel nostro modello 3D, la deformazione corrispondente ad una determinata blendshape non corrisponda in modo fedele a quanto indicato da ARKit; questo comporta la comparsa di evidenti deformazioni non volute e di un'animazione complessiva poco organica. Tutto ciò va assolutamente evitato, risulta per tanto essenziale prestare estrema attenzione in fase di creazione delle blendshape.

L'utilizzo delle blendshape pone quindi un forte limite, sia perché creare 52 blendshape consiste in un lavoro estremamente lungo e in un processo di *trial & error*, sia perché anche una piccola imprecisione nella definizione di una blendshape potrebbe creare interferenze non volute. Come si vedrà nella sezione 9.2.1, tali limiti possono essere superati tramite l'utilizzo di appositi strumenti, in particolare con l'utilizzo dell'add-on di Blender *Faceit*.

Dall'altro lato però, l'utilizzo delle blendshape consente di trasferire l'animazione registrata su qualsiasi tipologia di volto, indipendentemente dalle proporzioni e dalla topologia poligonale, purché, ovviamente, le blendshape corrispondano (vedi figura 9.2).

Di seguito si vanno a riassumere i vantaggi e gli svantaggi che possono derivare dall'utilizzo di un sistema di motion capture facciale markerless basato sulla tecnologia ARKit:

Figura 9.2: Animazione applicata a modelli con topologie differenti

Pro:

- Tale approccio consente di velocizzare il processo di animazione e di renderlo alla portata anche di animatori non specializzati.
- Non necessita di grossi investimenti in termini di software e hardware.
- Non risulta necessario effettuare le *take* di motion capture in uno studio di registrazione o in un ambiente in particolare, questo svincola il processo di animazione dalla disponibilità di ambienti di ripresa.
- Essendo un approccio markerless, non prevede un grosso lavoro di preparazione ma risulta sufficiente avere a disposizione uno smartphone e la giusta illuminazione per poter catturare l'animazione facciale.
- Non è necessaria una figura tecnica specializzata per effettuare le registrazioni, la cattura delle espressioni facciali può essere effettuata in maniera semplice e veloce da chiunque.
- Permette di andare ad animare una vasta gamma di volti umanoidi, purché dotati delle giuste blendshape.

Contro:

- I dispositivi che possono fare uso di tecnologia ARKit sono limitati, in particolare ad ora si limitano ai dispositivi Apple dotati di TrueDepth camera[1]. Ciò significa che se lo studio non ne possiede neanche uno sarà necessario effettuare un acquisto, e trattandosi di prodotti Apple il prezzo di mercato tipicamente è medio-alto (1000 € circa a seconda del tipo e delle condizioni del dispositivo). Tale spesa risulta comunque minore rispetto all'utilizzo della maggior parte dei sistemi di motion capture visti precedentemente.
- L'utilizzo di 52 blendshape crea un collo di bottiglia nella riproposizione di tali deformazioni per ciascun personaggio che si vuole animare.
- I dati acquisti potrebbero non garantire un'animazione soddisfacente, soprattutto se il risultato che si cerca è più vicino ad uno stile *cartoon*.
- Non fornisce strumenti che permettano di andare ad agire in maniera semplice ed intuitiva sui dati raccolti.

Quanto detto fino ad ora è relativo alla tecnologia messa a disposizione da Apple per gli sviluppatori di applicazioni iOS. La fase successiva consiste nell'identificare quale o quali applicazioni utilizzare nella pipeline di produzione. Siccome la produzione di *Reverie Dawnfall* si

[1] Al momento della stesura di questo documento i dispositivi Apple dodati di TrueDepth camera sono: iPhone X, XR, XS, XS Max, 11, 11 Pro, 11 Pro Max, iPhone12...; iPad Pro di terza o quarta generazione

basa sull'utilizzo di Blender, risulta fondamentale scegliere un'applicazione in grado di dialogare in maniera più o meno diretta con tale programma. Da questa esigenza sono state individuate due applicazioni, *Face Capture* di Rokoko e *Face Cap* di Bannaflak; entrambe si basano sulla tecnologia ARKit ed entrambe consentono di utilizzare i dati raccolti all'interno di Blender. Nelle sezioni 9.1.1 e 9.1.2 si andrà a descriverle brevemente e a riportarne pregi e difetti.

9.1.1 Face Capture di Rokoko

L'applicazione Face Capture di Rokoko è stata la scelta effettuata dai miei colleghi durante la realizzazione del secondo trailer di *Reverie Dawnfall*. Il motivo principale che ha spinto in questa direzione è il fatto che l'animazione dei personaggi viene effettuata tramite una tuta provvista di sistemi inerziali fornita dalla stessa azienda. Siccome sia i dati catturati tramite tuta, che i dati catturati tramite applicazione vengono elaborati all'interno dello stesso software, *Rokoko Studio*, questo consente di avere una pipeline di produzione integrata ed unica. Tuttavia, l'effettiva integrazione tra movimento del corpo e cattura facciale è limitata in quanto rimangono sostanzialmente due componenti separate.

Il vero vantaggio si verifica in fase di registrazione in quanto è possibile avere un riscontro immediato, tramite Rokoko Studio, di ciò che si sta catturando. Ma, visto il necessario lavoro di pulizia che richiede l'animazione del corpo, tale vantaggio risulta comunque poco sfruttato; anche perché all'interno di Rokoko Studio non si ha la possibilità di agire sui dati relativi al volto, quindi il lavoro di animazione vero e proprio verrà comunque effettuato su Blender.

Per quanto riguarda il costo, l'applicazione risulta scaricabile gratuitamente ma per attivarne le funzionalità è necessario effettuare l'abbonamento all'add-on di Rokoko Studio che consente, per l'appunto, di dialogare ed importare i dati dal dispositivo mobile alla workstation. Tale abbonamento prevede un costo mensile di 39 $ se si sottoscrive annualmente, altrimenti 49 $ mensili, da aggiungere al costo di Rokoko Studio [60], che però per quanto riguarda Robin Studio è già stato effettuato.

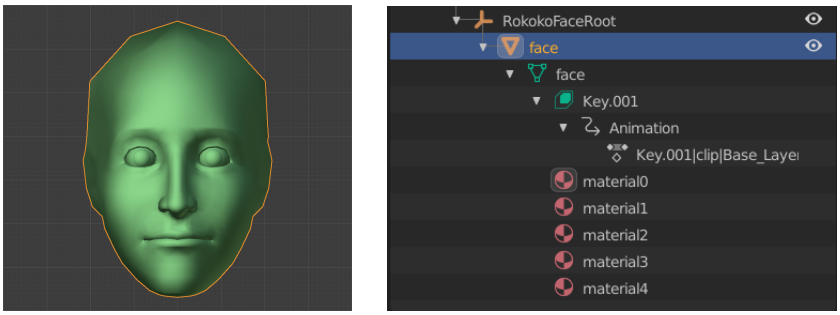
Una volta importati i dati catturati all'interno di Rokoko Studio è possibile decidere come tali dati dovranno essere esportati. Nel nostro caso, i dati relativi all'animazione facciale devono essere esportati in formato `FBX` (binario) in modo da essere letti facilmente all'interno di Blender.

Tali dati consistono in una mesh predefinita di un volto, dotata delle 52 blendshape individuate da ARKit, per ciascuna blendshape e per ciascun frame acquisito, sono riportati i valori di influenza nel range [0,1]. In questo modo risulta possibile trasferire l'animazione delle blendshape del file `FBX` sul personaggio che si vuole animare.

La figura 9.3 mostra in 9.3a il modello del volto generato da Rokoko e in 9.3b la struttura dell'oggetto che viene importato. I dati che a noi interessano sono contenuti dentro la mesh nominata *face*.

9.1.2 Face Cap di Bannaflak

Seguendo gli stessi principi base visti prima, questa applicazione si fonda sull'utilizzo della tecnologia ARKit. Le differenze principali rispetto all'applicazione di Rokoko è che, in questo caso, non risulta necessario mediare attraverso un software proprietario ma l'applicazione consente di esportare direttamente i dati catturati sotto forma di file `FBX ASCII`. Questo, tuttavia, ci obbliga ad effettuare una conversione di tale file in quanto Blender non è in grado di aprire file



(a) Modello 3D generato **(b)** Outliner interno a Blender

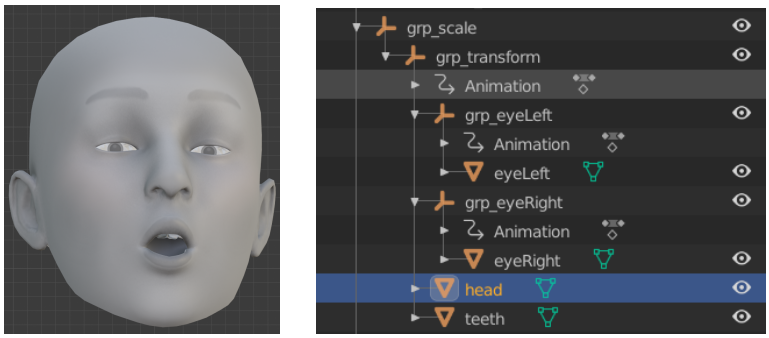
Figura 9.3: Oggetto fbx creato con Face Capture di Rokoko

`ascii-fbx`. Una soluzione semplice e gratuita consiste nell'utilizzare il programma Autodesk FBX Converter che, per l'appunto, consente di convertire l'ascii-fbx in un file FBX binario.

FaceCap consente di registrare ed esportare i dati anche gratuitamente tramite una versione di prova. Tuttavia, tale versione limita a pochi secondi (5s) la dimensione dei file catturati; se si vuole pieno accesso alle sue funzionalità sarà necessario effettuarne l'acquisto, il costo si aggira sui 60 $ [61]. Risulta quindi un ottimo investimento.

Oltre ad esportare il file `FBX` contenente la mesh, le blendshape e l'animazione, consente di registrare e sincronizzare una traccia audio (formato `WAV`) e di esportare i dati in un formato di testo.

Nella figura 9.4 viene mostrato in 9.4a il modello 3D che viene creato da FaceCap e in 9.4b la struttura dell'oggetto importato all'interno di Blender. I dati relativi all'animazione delle espressioni facciali sono contenuti dentro la mesh *head*. I restanti dati sono principalmente relativi al movimento della testa all'interno del volume di cattura, dati che a noi non servono in quanto già acquisti tramite la tuta Smartsuit Pro di Rokoko.



(a) Modello 3D generato **(b)** Outliner interno a Blender

Figura 9.4: Oggetto fbx creato con Face Cap di Bannaflak

9.2 Creazione delle 52 blendshape

Come già discusso nel capitolo 3.2 tale approccio per l'animazione facciale porta con sé alcune problematiche tra cui l'insorgenza di possibili interferenze e le lunghe tempistiche necessarie alla creazione delle blendshape stesse. Ricordiamo che «creare un modello sufficientemente dettagliato può richiedere fino ad un anno di lavoro per un modellatore esperto» [13] necessitando di molte correzioni e rifiniture.

Ovviamente noi, dato il contesto indipendente, non ci troviamo nella condizione di poter impiegare così tanto tempo per la realizzazione di un set accettabile di blendshape per ogni personaggio, per questo motivo risulta fondamentale ottimizzare tale processo. Il limite che maggiormente hanno riscontrato i miei colleghi quando si sono confrontati con l'animazione facciale riguardava, per l'appunto, proprio la creazione delle 52 blendshape necessarie per il corretto funzionamento del sistema di motion capture facciale basato su ARKit.

Per *Nadya* e *Jameela*, i due personaggi principali presenti nel trailer realizzato dai miei colleghi, sono stati adottati due approcci differenti per l'animazione facciale, ma entrambi improntati sull'utilizzo delle blendshape. Per Nadya erano state create le cinquantadue blendshape, necessarie per l'uso del Face Capture di Rokoko, manualmente. Tuttavia, questo ha determinato l'insorgenza di molte interferenze e di una mancata corrispondenza tra l'espressione registrata e quelle trasferita sul personaggio di Nadya [33]. In figura 9.5 si possono notare alcune delle interferenze causate da un'incorretta creazione delle blendshape e viene mostrato come «nessuna parte del volto sembra avvicinarsi all'espressione della take di mocap» [33].

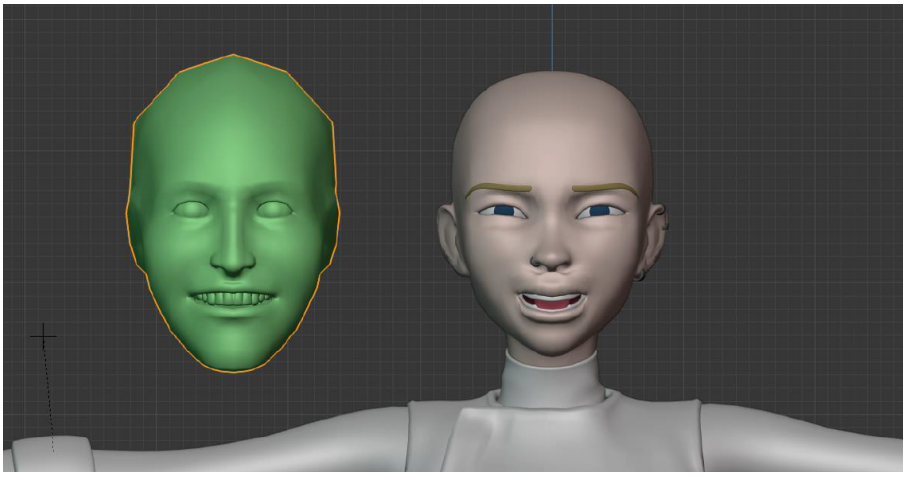


Figura 9.5: Interferenze tra blendshape sul volto di Nadya. Quest'immagine è tratta da [33].

È stato quindi necessario un lungo lavoro di ripulitura, che ha portato alla modifica delle blendshape già create, alla creazione di blendshape aggiuntive appositamente create per contrastare alcune interferenze e al rifacimento completo di altre. Tuttavia, nonostante questo lungo lavoro di rifinitura, l'animazione facciale risultante non era ancora soddisfacente, il che ha comportato un intervento manuale per andare ad ottenere il risultato desiderato [33].

Data la quantità di lavoro necessaria per animare Nadya, per il personaggio di Jameela si era deciso di utilizzare un altro approccio. In questo caso sono state create solo cinque blendshape rappresentanti alcune delle espressioni principali. Sebbene Jameela, all'interno del

trailer, fosse un personaggio secondario, la sua mancata espressività traspare dallo schermo. Un tale approccio all'animazione facciale risulta non accettabile se utilizzato all'interno di un prodotto seriale come *Reverie Dawnfall*, dove i personaggi che compaiono in ogni episodio, tra quelli principali, secondari e le comparse, sono molteplici. Risulta necessario andare ad individuare delle alternative che consentano di abbattere le tempistiche e che consentano di creare animazioni credibili per il maggior numero di personaggi in scena.

Un'alternativa alla creazione manuale delle blendshape consiste nel rivolgersi a servizi esterni come Polywink, in particolare questo offre la possibilità di creare le 52 blendshape necessarie per utilizzare ARKit in 24 ore [62], tuttavia il costo per ottenere come risultato un oggetto FBX è di 299 € a personaggio, il che la rende una via impraticabile per una produzione seriale indipendente in cui i personaggi da animare sono molteplici.

Questo ci porta a *Faceit*, lo strumento individuato per porre soluzione ai limiti visti precedentemente.

9.2.1 L'uso dell'Add-On Faceit

Faceit è un add-on di Blender, creato da Fynn Braren, che consente di generare in maniera semi-automatica un set di blendshape per il viso di personaggi umanoidi senza utilizzare strumenti di modellazione o di sculpting [63]. È un tool appositamente pensato per creare tutte e cinquantadue le blendshape necessarie per utilizzare i sistemi di motion capture markerless con tecnologia ARKit.

Nonostante sia un add-on di Blender, al momento il download del codice è venduto a 78 $ per la versione singola e 289 $ per la versione Studio. Effettuare l'acquisto garantisce la possibilità di ricevere tutti i successivi aggiornamenti gratuitamente e di poter accedere al codice sorgente. Per questo lavoro di tesi sono riuscito a mettermi in contatto con Fynn Braren, che è risultato estremamente disponibile a fornirmi la versione v1.2.1 a titolo gratuito. Il costo, che potrebbe risultare elevato per un add-on di un software open source, risulta estremamente vantaggioso e competitivo, viste le potenzialità offerte.

Con Faceit si ha la possibilità di creare tutte le blendshape necessarie in circa un paio d'ore, andando a far risparmiare settimane di lavoro basato su un approccio *trial & error*. La velocità introdotta e la sua facilità di utilizzo, rendono Faceit uno strumento indispensabile all'interno della pipeline di produzione di *Reverie Dawnfall*. Ciò rende vantaggioso andare ad utilizzare il mocap facciale anche per quei personaggi non principali, che hanno pochi minuti a schermo e poche battute, ma che se animati con tecniche alternative potrebbero richiedere troppo tempo o l'animazione potrebbe risultare non sufficientemente espressiva.

Un altro vantaggio nell'utilizzare Faceit è che le blendshape create sono appositamente pensate per lavorare con la tecnologia ARKit, quindi si vanno ad eliminare alla radice eventuali problemi di mancata corrispondenza delle espressioni e problemi di interferenza. Inoltre, Faceit fornisce una documentazione dettagliata e completa sul suo funzionamento ed utilizzo, il che rende il processo facile ed intuitivo anche per chi ci si approccia per la prima volta.

Di seguito si vanno a descrivere i principali passaggi che sono stati svolti per la creazione delle 52 blendshape per i personaggi di Nadya e Jameela, della serie *Reverie Dawnfall*, utilizzando la versione v1.2.1 dell'add-on Faceit. Per maggiori dettagli sulle ultime versioni il lettore può fare riferimento alla documentazione di Faceit [64].

1. **Preparazione della geometria**
   Sebbene Faceit sia in grado di lavorare con una vasta gamma di possibili volti, per ottenere il miglior risultato possibile è bene partire con un modello 3D ben organizzato. In particolare, è fondamentale che il modello del volto abbia una corretta topologia poligonale (vedi capitolo 2.5) e che rispetti alcuni prerequisiti:

   - La mesh del volto deve essere una superfice continua e connessa;
   - Il modello di partenza deve essere in una posa neutrale (occhi aperti e bocca chiusa);
   - Il personaggio deve avere la geometria degli occhi, mentre denti, lingua e barba sono opzionali;
   - È consigliato dividere le diverse componenti del viso in oggetti separati;
   - Se il personaggio è già dotato di armatura, come nel caso di Nadya e Jameela, è necessario effettuare una fase di duplicazione del modello prima di procedere (fase chiamata *duplicate workaround*).

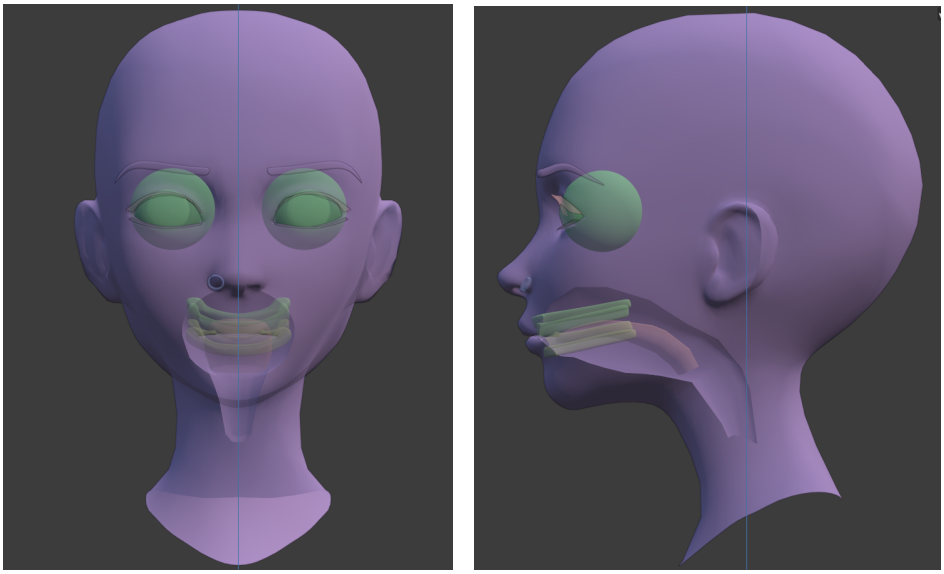


Figura 9.6: Modello del viso di Nadya scomposto in differenti oggetti. Le differenti geometrie sono rappresentate con colori diversi.

   Per evitare possibili problemi nelle fasi successive, è consigliato separare le diverse geometrie, che concorrono alla creazione del volto, in oggetti distinti. Ad esempio, si va a dividere la mesh del volto da quella della parte interna della bocca (arcata dentale superiore/inferiore e lingua) e dalla mesh degli occhi. Per separare la geometria si può procedere come indicato: *Edit Mode* > *Select Verticies* > *Mesh* (menu) > *Separate* > *By Loose Parts* oppure *Edit Mode* > *Select Vertices* > *P* (shortcut) > *Choose an Option*. In una fase successiva sarà comunque possibile riunire i diversi oggetti in un'unica geometria: *Object Mode* > *Select Objects* > *Ctrl + J*.

2. **Duplicate workaround**
   Siccome sia il modello di Nadya che il modello di Jameela sono già provvisti di un rig, sia per il corpo sia per il volto, in base a quanto indicato in [33], è stato necessario

precedere la creazione delle blendshape con una fase denominata *duplicate workaround*. Tale fase consiste nell'andare a duplicare le geometrie del modello 3D del personaggio, in modo tale che tutte le modifiche che verranno effettuate nelle fasi successive non vadano a compromettere il rig. Infatti, la procedura che porta alla creazione delle blendshape comporta una perdita di informazioni relative ai *vertex group* e alle *shape key* (blendshape) già create.

Siccome all'interno di Blender è possibile trasferire le shape key tra due modelli aventi la stessa topologia, diventa possibile andare ad effettuare il processo di creazione delle 52 blendshape sulla geometria duplicata e poi "copiare" le nuove blendshape sul modello originale, senza perdere alcun tipo di informazione. Per duplicare l'oggetto del volto basta, in *Object Mode*, selezionare tutti gli oggetti desiderati > *Shift+D*. Gli oggetti duplicati, per mantenere lo spazio di lavoro pulito, possono essere spostati in una nuova Collection, *M* > *New Collection*.

Ora è possibile applicare le modifiche necessarie ai duplicati; bisogna però fare attenzione a non modificare la topologia della mesh andando ad aggiungere o eliminare vertici in quanto questo impedirebbe di trasferire, al modello originale, le blendshape generate.

Un altro passaggio fondamentale consiste nel rimuovere tutti quei modificatori che deformano la geometria, come *Armature Modifier*[2]. Vanno inoltre eliminate le relazioni di parentela, selezionando gli oggetti interessati > *Alt+P* > *Clear and Keep Transformation*. Infine, se necessario, si procede con il separare le diverse geometrie che compongono la faccia come indicato nello step 1.

Ora è possibile proseguire normalmente con le altre fasi andando a lavorare sugli oggetti duplicati, al termine di tutto vi sarà un'ultima fase che consiste nel trasferire le blendshape create con i duplicati, sugli oggetti originali.

3. **Setup**
   Dopo essersi assicurati che il modello su cui si lavora rispetti quanto sopra riportato, si può procedere alla prima fase che prevede l'utilizzo vero e proprio dell'add-on Faceit. In questa fase, che ha atto nel pannello *Setup* dell'add-on, si andranno a registrare gli oggetti relativi al volto e si definiranno le diverse parti della faccia. Per prima cosa bisogna registrare l'oggetto principale che rappresenta la faccia, selezionando l'oggetto in questione e andando a cliccare su *Register Face Object*. Dopo di che vanno registrati tutti gli oggetti secondari (es occhi, bocca...) che sono soggetti a deformazione, andandoli prima a selezionare e poi andando a premere il bottone *Register Secondary Objects*.

   Ora si può procedere con il definire le diverse parti della faccia, specificando i vertici che ne fanno parte. Tali vertici verranno salvati all'interno di un apposito *Vertex Group*. Le parti della faccia previste da Faceit si dividono in *Eyes* (destro e sinistro), *Teeth* (superiori e inferiori, possono comprendere anche le rispettive gengive e tutte quelle parti che ne devono seguire il movimento in maniera rigida), *Tongue* e *Rigid* che riguarda tutte quelle superfici rigide che non devono deformarsi.

[2]La lista completa dei modificatori che deformano la geometria può essere consultata nel manuale di Blender al seguente indirizzo: https://docs.blender.org/manual/en/latest/modeling/modifiers/index.html#deform.

Per indicare i vertici bisogna entrare in *Edit Mode*, selezionare i vertici necessari e premere il rispettivo bottone (es *LeftEyeball*). Per altri oggetti, come ad esempio la barba e le ciglia, è sufficiente effettuarne la registrazione ma non è necessario definirne i vertici.

4. **Landmark**
   Dopo aver correttamente registrato e definito le varie componenti della faccia è possibile passare al secondo pannello dell'add-on denominato *Rig*. Il posizionamento dei landmark è uno step importantissimo, in quanto da questa fase dipende gran parte della qualità del risultato finale; per questo motivo è necessario prestare abbondante attenzione e procedere con precisione.

   Per prima cosa bisogna cliccare sull'operatore *Generate Landmarks*, questo farà apparire una sorta di maschera che dovrà essere posizionata e scalata sul volto del nostro personaggio, attraverso il movimento del mouse ed un click finale di conferma. In questo primo step non è necessario essere precisi ma l'obiettivo è agevolare il lavoro successivo. Faceit permette, inoltre, di indicare se la geometria su cui si lavora è simmetrica o asimmetrica, per quanto riguarda i personaggi della serie *Reverie Dawnfall* in fase di modellazione si è scelto un approccio simmetrico. Per il corretto svolgimento di questa fase è necessario che sia impostata la vista frontale (`Numpad 1`) e che ci si trovi in *Edit Mode*. Quindi i punti di riferimento andranno posizionati il più precisamente possibile seguendo quella che è la conformazione del viso; una volta sistemati i punti dalla vista frontale è possibile cliccare sull'operatore *Project Landmarks* che andrà a proiettare i punti sulla mesh del nostro personaggio.

   Da questo momento è possibile procedere con il preciso posizionamento dei landmark utilizzando delle viste prospettiche in modo da avere un controllo a 360 gradi. Come già detto, il corretto posizionamento dei punti di riferimento è fondamentale e determinante per la qualità del risultato finale; se non si è soddisfatti è possibile tornare indietro e riposizionarli.

   Nella figura 9.7 viene mostrato come i landmark sono stati posizionati per il personaggio di Nadya; ulteriori esempi possono essere trovati all'interno della documentazione [64].

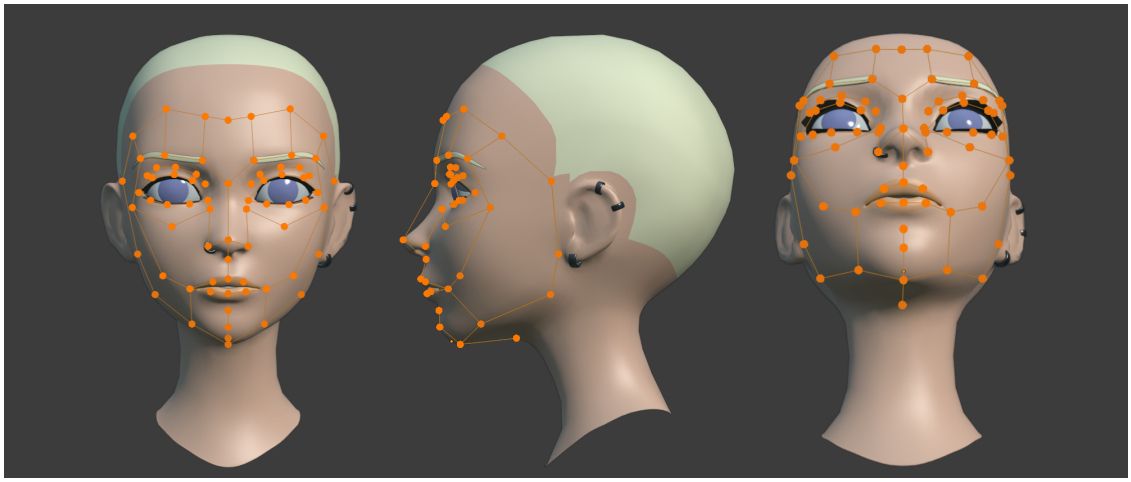


Figura 9.7: Posizionamento dei landmark per Nadya

5. **Rigging**
   Se si è soddisfatti del posizionamento dei landmark, si può procedere con la fase di rigging, sempre all'interno del pannello *Rig*. Premendo l'operatore *Generate Rig* si andrà a creare quella che in Blender viene chiamata *Armatura*, ovvero una struttura gerarchica composta da corpi articolati chiamati *bone*. La maggior parte delle ossa è posizionata esattamente dove erano stati posizionati i landmark mentre alcune ossa sono determinate tramite tecniche di interpolazione; questo è il motivo per il quale è fondamentale il corretto posizionamento dei punti di riferimento. Nel caso non fossimo soddisfatti del risultato, è possibile tornare alla fase precedente andando a cliccare su *Back to Landmarks*.

   Se invece siamo soddisfatti, possiamo procedere con la fase denominata *Bind*, vista nel capitolo 3.3 del documento con il nome di *skinning*, ovvero quel processo nel quale si va ad unire l'armatura (rig) all'oggetto che dovrà essere deformato. In Faceit questo processo è suddiviso in *Bind* (primario) e *Bind Secondary*.

   Con l'operatore *Bind* si vanno ad unire le geometrie registrate nella fase di Setup con l'armatura appena creata; da questo step vengono escluse le geometrie opzionali come quelle relative alla barba e alle sopracciglia che invece verranno unite con l'operatore *Bind Secondary*. Potrebbe essere necessario modificare manualmente i pesi assegnati ai vertici tramite tecniche di *weight painting*; questo, insieme alla fase di posizionamento dei landmark, sono le fasi che più necessitano un lavoro di *trial & error*, ma grazie alla semplicità di utilizzo garantita dall'interfaccia non risultano in processi lunghi e snervanti.

   Solo dopo aver effettuato il *Bind* è possibile procedere con l'operatore *Bind Secondary* per unire all'armatura le geometrie mancanti. Uno step opzionale riguarda l'utilizzo dell'operatore *Smooth Correct Modifier*, che per alcuni personaggi potrebbe migliorare la qualità della deformazione.

6. **Animation**
   Con la creazione del rig è stata generata anche una sequenza contenente tutte le 52 espressioni definite da ARKit, tali espressioni sono equispaziate da dieci frame in modo che sia possibile vederne in modo chiaro lo sviluppo, per un totale di 520 fotogrammi. Per vedere tale sequenza in azione è sufficiente mettere in play l'animazione; tramite il pannello *Animate*; Faceit fornisce un'interfaccia intuitiva che consente all'utente di "navigare" tra tutte le espressioni generate. In pratica si ha un link tra l'elemento della lista e il rispettivo keyframe nella timeline. In questa fase diventa possibile controllare quello che sarà il risultato finale, o per lo meno come ogni blendshape andrà ad agire da sola.

   Sebbene Faceit sia uno strumento molto potente, potrebbe generare delle espressioni non completamente soddisfacenti, per questo motivo si ha la possibilità di andare a editarle entrando in *Pose Mode*. Per modificare un'espressione è sufficiente posizionarsi sul keyframe che ne indica la massima influenza, andando a cliccare la blendshape che ci interessa della lista presente nel pannello *Animate*; poi in *Pose Mode* è possibile agire sulle ossa dell'armatura in modo da ottenere il risultato desiderato (tramite traslazione, rotazione e scalamento delle ossa). Dopo di che è possibile salvare le modifiche apportate andando ad aggiornare i keyframe delle ossa modificate.

   Molte espressioni facciali agiscono solo su un lato del volto, per agevolare il lavoro, Faceit offre la possibilità di copiare le modifiche fatte ad un'espressione di un lato del volto, sull'altra metà tramite l'operare *Mirror Shape*.

Per assicurarsi che le espressioni facciali siano effettivamente congrue con quanto previsto da ARKit, si consiglia di effettuare una fase di verifica in cui si va a confrontare ogni singola blendshape descritta e rappresentata nella documentazione ARKit [59] con quella generata tramite Faceit. Questa è una fase cruciale in quanto l'espressività del personaggio e in generale l'animazione facciale dipendono dalla corretta deformazione che apporta ogni blendshape. Nel caso un personaggio abbia delle caratteristiche peculiari, ad esempio un occhio più chiuso dell'altro, in questa fase è possibile andare a modificare le deformazioni apportate da ogni espressione. Tuttavia, questo potrebbe introdurre dei limiti qualora si decidesse, in un momento successivo, per ragioni stilistiche o pratiche, di modificare tale caratteristica. Si consiglia, quindi, di procedere con la creazione "classica" delle blendshape previste da ARKit e nel caso fosse necessario introdurre particolari caratteristiche, queste possono essere realizzate in fase di animazione tramite altri strumenti come, ad esempio, l'introduzione di blendshape apposite o andando a utilizzare dei rig facciali.

La fase di verifica delle blendshape potrebbe richiedere del tempo ma è uno step necessario per assicurarsi il miglior risultato possibile.

7. **Baking**
   Una volta che si è soddisfatti delle espressioni facciali generate tramite le differenti pose dell'armatura, è possibile andare a creare effettivamente le blendshape (shape key in Blender). Tale fase si chiama *baking* e viene effettuata andando nel pannello *Bake* dell'add-on e cliccando sul bottone *Bake ARKit Shape Keys*.

   Con l'operazione di bake verranno create tutte e 52 le blendshape previste da ARKit. Sebbene il risultato visivo dell'animazione delle singole espressioni possa essere soddisfacente, è fondamentale testare le blendshape create con un'animazione vera e propria, in modo da verificare la possibile insorgenza di eventuali interferenze o una scarsa organicità del risultato ottenuto. Infatti, è solo con un'animazione complessa, che prevede l'influenza di un elevato numero di blendshape contemporaneamente, che è possibile accorgersi di eventuali problemi.

   Nel caso insorgessero problemi, è sufficiente tornare alla fase di rigging e ripercorrere gli stessi passi. Una volta che le shape key sono state create rimane comunque possibile modificarle manualmente tramite un approccio tradizionale.

8. **Trasferire le blendshape alla geometria originale**
   Siccome nel caso dei personaggi di Nadya e Jameela, è stato necessario passare attraverso la fase di *duplicate workaround*, come passaggio finale è fondamentale trasferire il lavoro svolto fino ad ora sul duplicato, al modello originale. Tale trasferimento è possibile solo se non sono state apportate modifiche alla topologia poligonale degli oggetti, ovvero se non sono stati aggiunti o rimossi vertici e se non è stata modificata la modalità con la quale i vertici sono uniti tra loro. Se nello step 1 si è separato il modello originale in diverse geometrie allora potrebbe essere necessario riordinare in modo corretto i vertici che compongono le mesh altrimenti le blendshape trasferite potrebbero creare deformazioni non volute.

   Per trasferire le blendshape è necessario selezionare l'oggetto originale e tenendo premuto il tasto *Shift*, selezionare l'oggetto duplicato; dopo di che è sufficiente premere il bottone *Transfer Shape Keys* nel pannello *Bake* dell'add-on e le blendshape verranno automaticamente trasferite.

Sebbene a prima vista la descrizione delle fasi sopra riportate possa far pensare che tale processo richieda molto lavoro, la verità è che per ottenere un risultato accettabile con l'utilizzo di Faceit sono sufficienti circa un paio d'ore di lavoro. Ovviamente tali tempistiche possono variare a seconda del personaggio che si va a utilizzare e la precisione che si vuole ottenere; personaggi principali richiedono maggior precisione e quindi tempistiche maggiori rispetto a personaggi di sfondo in cui è possibile prestare meno attenzione.

Rimane comunque uno strumento potentissimo che permette di risparmiare settimane di lavoro e quindi anche di ottimizzare il budget a disposizione. Faceit diventa, quindi, uno strumento indispensabile all'interno della produzione della serie animata *Reverie Dawnfall*.

Capitolo 10

Animazione della lingua

Quando la scena prevede un dialogo effettuato da uno o più personaggi, e l'inquadratura permette di vedere in modo piuttosto ravvicinato il volto del personaggio (es primo piano, primissimo piano, dettaglio bocca) allora risulta "necessario" introdurre oltre all'animazione del volto, di labbra, denti e occhi anche i movimenti relativi alla lingua, questo perché rende la scena più organica e credibile. Infatti, anche se il più delle volte non ne siamo consci, il nostro cervello capta ed elabora anche le informazioni provenienti dal movimento della lingua, che risulta molto utile per migliorare la comprensione delle parole che un interlocutore sta pronunciando.

Il movimento della lingua, accompagnato dalla posizione delle labbra e di eventuali espressioni facciali, rende significativamente più semplice ed immediata la comprensione di ciò che viene detto. Questo è uno dei motivi per il quale tendenzialmente la comprensione del discorso migliora quando ci si trova faccia a faccia con l'interlocutore piuttosto che parlandoci al telefono. Anche all'interno di un prodotto di animazione, il movimento della lingua conferisce ad un personaggio maggiore credibilità e naturalezza, inoltre permette allo spettatore di avere dei riferimenti chiari relativi alle parole pronunciate all'interno di un dialogo, facilitando in questo modo la visione dell'opera e permettendo di vivere un'esperienza migliore.

Tale componente non era stata presa in considerazione dai miei colleghi nei lavori precedenti, dovuto anche al fatto che all'interno del prodotto da loro realizzato non erano presenti scene di dialogo importanti. Inoltre, come visto nel capitolo 9, gli strumenti fino ad ora individuati per realizzare l'animazione facciale, ovvero i sistemi di mocap facciale basati su ARKit, non consentono di catturare dati relativi al movimento della lingua ad eccezione di un'espressione denominata *tongueOut*, che però risulta di utilizzo estremamente limitato. Per questo motivo è stato necessario individuare strumenti e tecniche alternative per poter integrare l'animazione della lingua all'interno della pipeline di produzione. Tale processo di lavoro, infatti, sebbene possa essere in parte separato dall'animazione delle espressioni facciali, deve poter essere integrato nel workflow generale dell'animazione facciale e dovrebbe rispettare le stesse esigenze individuate nel capitolo 7.3, ovvero consentire un'animazione stilizzata e al contempo gravare il meno possibile sulle risorse a disposizione.

Un primo approccio potrebbe essere quello di animare la lingua tramite un processo basato sui keyframe. Tuttavia, sebbene le azioni che la lingua può compiere siano abbastanza limitate, almeno nei prodotti stilizzati dove non si vuole raggiungere il realismo, all'interno di un prodotto

seriale come *Reverie Dawnfall* la presenza di dialoghi abbonda e di conseguenza aumenta anche il lavoro relativo all'animazione della lingua. Quindi animare la lingua manualmente tramite keyframe, come visto nella sezione 5.1.1, è un approccio limitante e poco adatto al nostro scopo.

Dato che, fino a questo momento, i nostri personaggi sono animati quasi interamente con tecniche di motion capture, potrebbe venire in mente di adottare un metodo simile anche per andare ad animare la lingua. Tuttavia, come detto sopra, il sistema di mocap di ARKit non è in grado di catturare il movimento della lingua come da noi sperato; alcune alternative potrebbero arrivare da altri sistemi di mocap, ma come visto nel capitolo 5.3 nessuna delle tecniche viste consente di registrare in modo affidabile ciò che ci serve. Per questo motivo si sono cercate soluzioni alternative.

Come prima cosa si sono approfondite le tecniche di animazione guidate dall'audio, come riportato nel capitolo 5.2. Queste tecniche, per l'appunto, si basano su file audio o su file di testo per animare dei modelli 3D, in particolare per il nostro scopo risultano interessanti quelle tecniche che fanno utilizzo dei fonemi e dei visemi (la controparte visiva dei fonemi). Tuttavia, estrarre in modo automatico i fonemi da una traccia audio è una tecnica estremamente complessa e tuttora in sviluppo che, come abbiamo visto, fa uso del machine learning e di reti neurali. Questo ha comportato enormi difficoltà nel reperire materiale *open source* o prodotti consumer adattabili alle esigenze di una produzione indipendente. In particolare, non è stato possibile trovare sistemi audio-driven che fossero economicamente abbordabili e che consentissero di modificarne le funzionalità per meglio adattarle al nostro scopo. È stato necessario, rivolgersi nuovamente ad altre opzioni.

La soluzione attuale è giunta grazie all'utilizzo di *Papagayo-NG*, software che permette di semi-automatizzare il *lip sync* per la creazione di prodotti animati, tipicamente 2D. Sebbene una descrizione generale di questo programma sia già stata trattata nel capitolo 8.4, di seguito andremo ad esplorarne in maniera più approfondita il suo funzionamento e come la sua natura open source ci abbia consentito di adattarlo alle esigenze della produzione di *Reverie Dawnfall*.

10.1 Come funziona Papagayo-NG

Come visto nel capitolo 8.4, Papagayo-NG è la versione più aggiornata del programma Papagayo e, ad eccezione di qualche estensione, i principi sui cui si basano sono gli stessi. È un programma pensato per facilitare il lip sync in prodotti di animazione tipicamente 2D, quindi non è caratterizzato da un elevato realismo ma risulta sufficientemente affidabile da poter essere utilizzato per il nostro scopo.

Papagayo-NG può essere considerato un approccio ibrido, in quanto fa utilizzo sia di una traccia audio che di un file di testo ma questi verranno utilizzati con scopi differenti. I dati principali, ovvero i fonemi, vengono estrapolati dal file di testo che, quindi, deve riportare in modo corretto tutte le parole pronunciate all'interno del dialogo. La traccia audio, invece, non verrà utilizzata per estrarre i fonemi ma viene utilizzata come base sulla quale andare a "costruire" il testo. Ovvero, le parole presenti nel file di testo verranno presentate all'interno di un'apposita interfaccia (*Waveform View*), sovrapposte alla forma d'onda del file audio corrispondente (vedi figura 10.1). Tramite tale interfaccia è possibile disporre, prima le frasi e poi le singole parole, in modo che queste si allineino temporalmente al frame nella quale una determinata parola viene

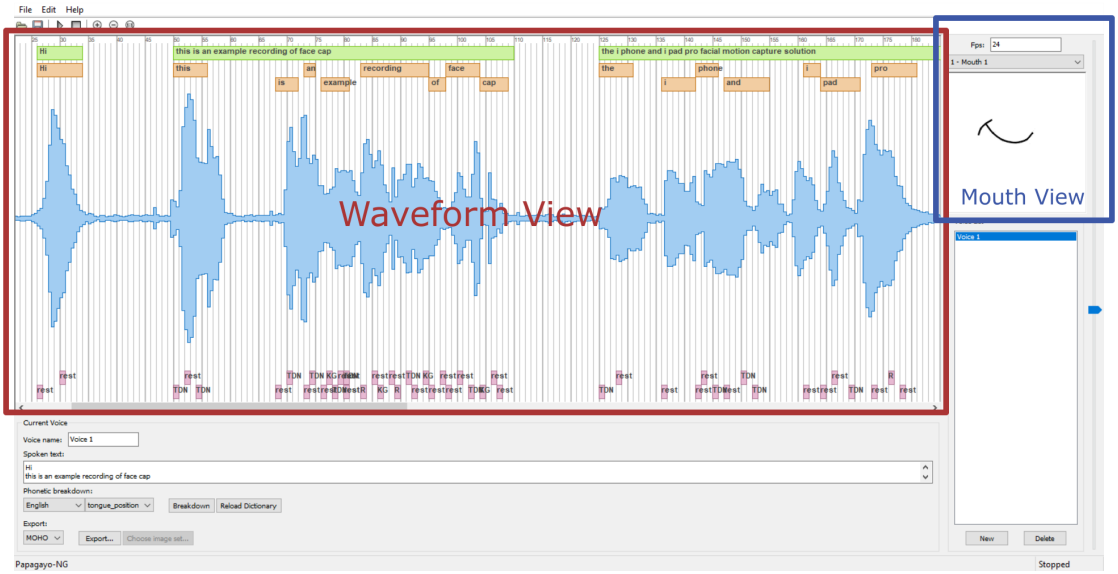


Figura 10.1: Interfaccia di Papagayo-NG

pronunciata all'interno della traccia audio. In questo modo è possibile avere una corrispondenza temporale tra traccia audio e fonemi individuati. Tale corrispondenza è fondamentale per andare ad esportare quanto ottenuto, nel nostro caso tramite file `.dat`; questo file sarà una sorta di dizionario dove ogni fonema viene legato al fotogramma nella quale viene pronunciato.

Con questo sistema diventa possibile utilizzare tale file all'interno di Blender, grazie all'utilizzo di specifici add-on. Un add-on che permette di importare i file dat provenienti da Papagayo-NG e di utilizzarli per agire su specifiche blendshape si chiama *Lipsync Importer & Blinker*, come vedremo nel capitolo 11 tale script non verrà utilizzato all'interno del workflow di *Reverie Dawnfall* ma sarà alla base di una specifica sezione di *ReveRig*, l'add-on creato all'interno di questo lavoro di tesi.

Quindi, l'idea alla base di questo approccio consiste nell'estrarre i principali fonemi da un file di testo riportante le parole del dialogo, associare tali fonemi a un dizionario di visemi e dargli un riferimento temporale basandosi sulla traccia audio, dopo di che importare i dati raccolti all'interno di Blender e infine utilizzare tali dati per animare il parametro di influenza delle blendshape corrispondenti ai visemi individuati.

Il primo limite da superare consiste nel fatto che Papagayo-NG è pensato per il lip sync e non per l'animazione della lingua, quindi sia i fonemi che i set di visemi presenti sono unicamente pensati per l'animazione della bocca. Diventa necessario definire un nuovo dizionario di fonemi e di visemi in modo che ci consenta di mappare i dati di ingresso in dati relativi all'animazione della lingua. Grazie al fatto che Papagayo-NG è distribuito con licenza compatibile a *GPL* si ha la possibilità di accedere al codice sorgente e di apportare le modifiche necessarie. Per far questo, prima è necessario dare una veloce indicazione di come il programma è strutturato.

10.2 Struttura del programma Papagayo-NG

Papagayo si basa sul *CMU Pronouncing Dictionary* (CMUdict), ovvero un dizionario open source creato dallo Speech Group della Carneige Mellon University (CMU) per essere utilizzato nel campo del riconoscimento vocale [65]. Il CMUdict fornisce una mappatura ortografica/fonetica per più di 134 000 parole inglesi; all'interno di Papagayo-NG si ha anche la possibilità di utilizzare testo in lingue differenti dall'inglese, come ad esempio l'italiano, ma bisogna precisare che si potrebbe incorrere in imprecisioni, mentre la versione inglese è quella che al momento fornisce la miglior corrispondenza.

All'interno della cartella con percorso `papagayo-ng/rsrc/languages/default` è possibile accedere ai dizionari CMU utilizzati, in particolare sono presenti una versione standard e una estesa del CMUdict, inoltre vi è un file `user_dictionary` che consente all'utente di poter aggiungere manualmente parole e termini non presenti nel CMUdict, come ad esempio i nomi dei personaggi presenti in *Reverie Dawnfall*. Ogni parola deve essere inserita in una propria riga, quindi prima si riporta il nuovo termine normalmente, dopo di che si vanno ad indicare i fonemi che compongono tale parola, i fonemi devono essere quelli utilizzati all'interno del CMUdict.

Parola	**Traduzione**
WOWZA	W AW1 Z AH0
GREEN	G R IY N

I fonemi utilizzati nel CMUdict sono in totale 39 e sebbene siano in numero minore rispetto a quelli individuati dall'*IPA* (International Phonetic Alphabet), consentono comunque di ottenere un risultato soddisfacente. Anzi, all'interno di Papagayo-NG, siccome 39 fonemi risultano difficilmente gestibili, si effettua un ulteriore passaggio che prevede di mappare i fonemi del CMUdict all'interno di un nuovo set di fonemi, tipicamente più ristretto. Di default Papagayo-NG utilizza il set di fonemi definito da Preston Blair; questo è un popolare set di visemi molto utilizzato per l'animazione facciale nei cartoni animati. Nel set di Preston Blair sono individuati solo 10 visemi e questi serviranno per mappare tutti i fonemi possibili [66]. Papagayo-NG mette a disposizione anche il set di fonemi individuato da Fleming & Dobbs, ma nessuno di quelli presentati risulta utilizzabile per gestire l'animazione della lingua.

L'operazione di mappatura e la definizione di questi set "ristretti" di fonemi viene effettuata in specifici file `.py` nominati come segue: `phoneme_name_of_the_set.py`, e tali file vengono richiamati all'interno di un ulteriore file python chiamato `phonemes.py`. Questo ci consente di creare un file apposito che vada a definire un nuovo set di visemi e che effettui la mappatura necessaria con i fonemi del CMUdict.

Per una migliore usabilità, l'interfaccia di Papagayo-NG prevede una finestra che consente di vedere un'anteprima dell'animazione generata (*Mouth View*); tale anteprima viene ricreata tramite immagini statiche, rappresentanti i rispettivi visemi, alternate tra loro. Papagayo-NG, all'interno della cartella `papagayo-ng/rsrc/mouths`, fornisce di default sette tipologie di immagini utilizzate per la pre-visualizzazione. Ciò significa che, nel momento in cui si andrà a creare un nuovo set di fonemi, sarà necessario creare anche una nuova cartella contenente le immagini relative ai nuovi fonemi generati.

Prima di modificare il codice sorgente di Papagayo-NG è necessario definire il nuovo set di fonemi/visemi che dovrà essere utilizzato per l'animazione della lingua.

10.3 Definizione del nuovo set di fonemi/visemi

Siccome la sincronizzazione labiale viene già effettuata tramite la tecnica di mocap facciale vista nel capitolo 9, ora ci si può concentrare unicamente sul movimento della lingua. Questo è un grosso vantaggio in quanto gestire l'animazione della bocca attraverso Papagayo-NG andrebbe a creare delle eccessive semplificazioni sui possibili visemi implementati. Perciò diventa possibile svincolare l'azione della lingua dal resto delle articolazioni dell'apparato fonatorio.

Nel mondo reale, i movimenti che può compiere la lingua all'interno della nostra bocca sono molteplici e a volte complessi, tuttavia non è scopo di questo progetto realizzare un'animazione eccessivamente realistica, anzi il prodotto, come visto in più occasioni, è caratterizzato da uno stile fumettistico e cartoon. Ciò ci consente di prendere in considerazione solo quei movimenti principali ed evidenti. Inoltre l'animazione della lingua risulta evidente solo laddove la bocca risulta sufficientemente aperta e dove il movimento è piuttosto marcato. Infine, come abbiamo visto nel capitolo 5.1.1, per personaggi stilizzati, non è necessario replicare tutti i movimenti della lingua ma basta riportare quelle che sono le azioni principali in modo che lo spettatore possa avere una comprensione maggiore della scena.

Basandosi su queste premesse, sebbene la posizione della lingua sia in parte influenzata anche dalla produzione delle vocali, si è individuato nella pronuncia delle consonanti il maggior apporto alla definizione della posizione della lingua. Di seguito vengono riportate le diverse posizioni delle articolazioni (POA, place of articulation), in particolare della lingua, durante la produzione delle principali consonanti. Un riferimento grafico a quanto riportato sotto può essere consultato a pagina 105 nelle figure 10.3 e 10.4.

Bilabiali nelle consonanti bilabiali il labbro superiore e quello inferiore si toccano. Per questo motivo il movimento della lingua all'interno della bocca può considerarsi ininfluente. Tali consonanti sono [p], [b] e [m] e nel CMUdict sono indicate come P, B e M.

Labiodentali nelle consonanti labiodentali il labbro inferiore va a toccare la parte inferiore dei denti superiori. Anche in questo caso la lingua non risulta visibile, pertanto non ne associamo alcun movimento in particolare. Tali consonanti sono [f] e [v] e nel CMUdict sono indicate come F e V.

Dentali nelle consonanti dentali la punta della lingua va a toccare la parte inferiore e posteriore dei denti superiori. Posizione tipicamente assunta nella pronuncia di termini inglesi che contengono i fonemi [θ] e [ð] indicati nel CMUdict come TH.

Alveolari nelle consonanti alveolari la punta della lingua si avvicina o tocca la cresta alveolare, ovvero quella parte di gengiva immediatamente dietro ai denti superiori. In questo gruppo rientrano le consonanti [t], [d], [n] che bloccano completamente il passaggio dell'aria e le [s], [z], [l] che sono caratterizzate da una minore pressione della lingua, tuttavia per i nostri fini possono essere considerate un gruppo unico. Nel CMUdict sono indicate come T, D, N, S, Z e L.

Post alveolari nelle consonanti post alveolari il comportamento è simile a quello delle alveolari, in questo caso la lingua va a spingere sulla cresta alveolare non solo con la punta ma anche con parte del corpo. In questo gruppo rientrano le consonanti [ʃ], [ʒ], [tʃ] e [dʒ]; data l'elevata somiglianza con le alveolari verranno definite dallo stesso visema.

Retroflesse nelle consonanti retroflesse la punta della lingua viene ruotata all'indietro all'interno della cavità orale. Una consonante retroflessa molto presente è la [ɹ] indicata nel CMUdict come R.

Palatali nelle consonanti palatali il corpo della lingua va a toccare il palato duro (parte anteriore). Esempio di consonante palatele è [j] presente nel CMUdict come JH.

Velari nelle consonanti velari il corpo della lingua va a toccare il palato molle (parte posteriore). In questa categoria rientrano [k], [g] e [ŋ] presenti nel CMUdict come K, G, Q e NG.

Glottidali nelle consonanti glottidali la lingua viene posizionata in modo da consentire un abbondante flusso d'aria, per questo motivo può essere considerata come una posizione a "riposo". Dentro questa categoria rientra la consonante [h], nel CMUdict H.

Da questa classificazione emerge la necessita di creare *sei nuovi visemi* con il compito di racchiudere tutti i fonemi presenti nel dizionario CMU. Siccome i termini utilizzati precedentemente per andare ad individuare le differenti consonanti sono poco intuitivi, per lo meno per chi non conosce la materia, si è deciso di etichettare i visemi del nuovo set con dei nomi più rappresentativi e di facile comprensione, utilizzando all'interno del nome le principali consonanti rappresentate. Nella tabella 10.1 si riporta il nuovo set di visemi, appositamente pensato per l'animazione della lingua, e la rispettiva corrispondenza con i fonemi del CMUdict.

Tabella 10.1: Nuovo set di fonemi/visemi

Nuovi fonemi	Tipologia di POA	Fonemi del CMUdict corrispondenti
TDN	alveolari e post-alveolari	T D N S Z J L SH CH DH ZH
TH	dentali	TH
J	palatali	JH
R	retroflesse	R ER
KG	velari	K G Q NG
rest	bilabiali, labiodentali, glottidali e altro	AA AE AH AO AW AY EH EY IH IY OW OY UH UW B M P HH F V W Y E21

Dopo aver definito il nuovo set di fonemi, si è creato un nuovo file denominato `phonemes_tongue_position.py`, si è aggiornato il file `phonemes.py` andando ad aggiungere `tongue_position` all'interno della lista `phoneme_sets`, infine si è creata una nuova cartella denominata `8-Tongue Position` (`papagayo-ng/rsrc/mouths`) contenente sei immagini corrispondenti ai nuovi visemi individuati. Il codice relativo al file `phonemes_tongue_position.py`, realizzato per effettuare la mappatura tra CMUdict e il nuovo set di fonemi `tongue_position`, e relativo al file `phonemes.py` può essere consultato nell'appendice A di questo documento.

Ora è possibile estrarre da un file di testo i fonemi indicati all'interno del set `tongue_position` e quindi creare il file `.dat` da importare all'interno di Blender. Di seguito andremo a riportare le fasi principali che caratterizzano il workflow interno a Papagayo-NG che porta alla creazione dei dati che poi verranno esportati in Blender.

10.4 Workflow di Papagayo-NG

Prima di incominciare a lavorare su Papagayo-NG è necessario avere registrato su traccia audio il dialogo che si vuole utilizzare. Sebbene sia possibile usare un audio appositamente creato per questa fase, il consiglio è di utilizzare quello che sarà l'audio definitivo all'interno del prodotto animato, o se questo non fosse possibile, di usare la traccia audio registrata durante la

fase di cattura delle espressioni facciali. In pratica ciò che serve è l'audio che corrisponda il più possibile a quanto verrà mostrato su schermo. Per quanto riguarda Papagayo-NG, la componente del file audio che più è determinante sul risultato finale è il suo sviluppo temporale, ovvero le pause e il ritmo all'interno del discorso. Infatti, Papagayo non è in grado di percepire altre informazioni come le differenti intonazioni o lo stato emotivo.

Per essere sicuri di poter importare correttamente il file audio all'interno del programma, la traccia audio dovrebbe essere salvata con un formato `WAV`; se si utilizza FaceCap per il mocap facciale, la traccia audio registrata sarà già nel formato corretto.

Dopo aver aperto Papagayo-NG, la prima cosa da fare è importare il file audio; per fare questo bisogna andare su *File* > *Open* (o con lo shortcut *Ctrl+O*) > si aprirà una finestra nella quale sarà possibile navigare all'interno del proprio sistema in modo da andare ad indicare il file che dovrà essere utilizzato, dopo aver selezionato il file corretto > premere *Open*. All'interno della *Waveform View* verrà mostrata la forma d'onda della traccia audio appena importata, tale forma d'onda risulterà campionata in base al frame rate indicato nell'apposita casella di input sopra la *Mouth View*, di default è settato a 24 fps ma è fondamentale utilizzare lo stesso frame rate che verrà poi utilizzato nella fase di animazione all'interno di Blender.

Per riprodurre l'audio è possibile premere il pulsante *Play*, oppure è possibile scorrere il mouse con il tasto sinistro premuto sopra la forma d'onda; in questo modo ci si può posizionare su uno specifico fotogramma.

Dopo aver importato il file `wav` possiamo notare che sia nella finestra *Current Voice* che in *Voice List* è stata automaticamente aggiunta una nuova voce denominata *Voice 1*, questo perché Papagayo-NG consente di gestire più voci che compaiono all'interno dello stesso audio; tuttavia, si consiglia di utilizzare tracce audio nelle quali compare un'unica voce in modo da rendere il processo di lavoro il più lineare ed intuitivo possibile. Infatti, la presenza di due o più voci differenti introduce un livello di complessità che sarebbe meglio evitare. Il *Voice name* può essere editato all'interno della finestra *Current Voice* nell'apposita casella di input.

Avendo importato il file audio è ora possibile procedere con l'inserimento del testo relativo a quanto detto nel dialogo. Il testo va inserito nell'apposita sezione denominata *Spoken text*; come si può intuire da questo termine il testo che andrà inserito dovrà corrispondere il più fedelmente possibile a quanto detto nella traccia audio.

Il tempo necessario a compiere questa fase dipende molto dalla lunghezza del dialogo e dal numero di parole contenute; tuttavia, se la performance catturata in fase di registrazione corrisponde a quanto indicato sulla sceneggiatura, allora diventa possibile effettuare una semplice operazione di *copia/incolla* risparmiando, in questo modo, molto lavoro. Nella maggior parte dei casi, però, risulta necessario apportare alcune modifiche dovute a possibili improvvisazioni o rivisitazioni effettuate in fase di ripresa. Si ricorda che *Reverie Dawnfall*, sebbene sia un prodotto italiano, è pensato per essere distribuito a livello internazionale, quindi sceneggiatura e prodotto finale saranno realizzati in lingua inglese; questo ci consente di utilizzare il CMUdict inglese, che al momento è quello più affidabile.

Durante l'inserimento del testo, per ottenere il miglior risultato possibile, si consiglia di seguire delle linee guida; in particolare bisogna omettere dal testo tutta la punteggiatura in quanto questa non partecipa alla definizione dei fonemi, anzi potrebbe creare distrazioni non volute. Si consiglia, inoltre, ogni volta che si verifica una pausa piuttosto evidente nel discorso, di procedere con la scrittura del testo partendo da una nuova riga in modo da separare in maniera

chiara le diverse frasi e le diverse parole. Come vedremo questo risulta molto utile nel momento in cui si dovrà procedere con il posizionamento dei fonemi nei vari fotogrammi.

Se all'interno del nostro testo compaiono parole inusuali, bisogna assicurarsi che esse siano presenti nei CMUdict utilizzati da Papagayo-NG, se così non fosse, è possibile procedere con l'aggiunta del nuovo termine all'interno dell'`user_dictionary` come visto nella sezione 10.2.

Una volta inserito correttamente il testo relativo a quanto pronunciato all'interno del file audio, è possibile procedere con la mappatura del testo nel set di fonemi da noi stabilito. Nella finestra *Current Voice*, sotto *Phonetic breakdown* è possibile definire la lingua utilizzata, nel nostro caso English e il set di fonemi da utilizzare, nel nostro caso `tongue_position`; inoltre è possibile indicare, nella finestra *Mouth View*, la tipologia di preview che vogliamo utilizzare, nel nostro caso `8 - Tongue Position`. Ovviamente tale set non è presente nella versione base di Papagayo-NG ma, come visto nella sezione 10.3, è stato creato all'interno di questo lavoro di tesi appositamente per la realizzazione di *Reverie Dawnfall*, quindi sarà necessario utilizzare la versione modificata di Papagayo-NG.

Impostati lingua e set di fonemi è possibile cliccare sul bottone *Breakdown*, qualora all'interno del testo fossero ancora presenti parole non riconosciute, comparirà una finestra di dialogo che ci inviterà ad applicare manualmente la mappatura dei fonemi.

Terminata la mappatura, all'interno della finestra *Waveform View* comparirà il testo scritto precedentemente in tre livelli di rappresentazione. Il primo livello, caratterizzato dal color verde, rappresenta il testo suddiviso nelle diverse frasi, ogni frase corrisponde al testo precedentemente scritto sulla stessa riga. Il secondo livello, di color salmone, crea una nuova casella per ogni parola presente nel testo; mentre l'ultimo livello, caratterizzato dal color viola, riporta i diversi fonemi individuati per ciascuna parola. Tale organizzazione a livelli risulta molto utile nella fase che seguirà, ovvero il posizionamento a livello temporale dei fonemi.

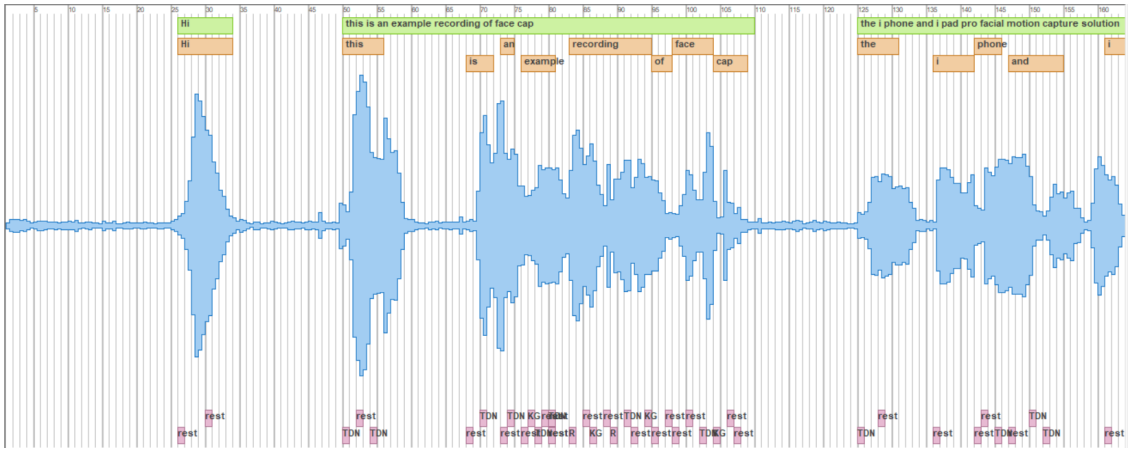


Figura 10.2: Waveform View

Come detto precedentemente, l'obiettivo è quello di creare una relazione tra i fonemi e i frame nei quali essi vengono espressi. Papagayo-NG non è in grado di determinare automaticamente tale corrispondenza temporale ma demanda il compito all'utente. Per agevolare il processo, viene

fornita la suddivisione a livelli vista sopra. Oltre a essere a livelli è anche una struttura gerarchica in quanto modificare la durata e la posizione di una frase, comporterà una corrispondente modifica delle parole comprese nella frase, mentre non si avrà alterazione delle restanti parti. Sfruttando lo stesso principio, i fonemi corrispondenti ad una parola, sono legati all'estensione temporale e alla posizione della parola stessa.

Questo consente di agire con diversi livelli di precisione, partendo dalle frasi, passando dalle parole ed infine, se necessario, andando ad agire direttamente sui fonemi. Tale suddivisione gerarchica è il motivo per il quale precedentemente si è consigliato di dividere su diverse righe di testo le varie componenti del discorso.

La fase di posizionamento è quella che richiede tempistiche maggiori, infatti, a seconda del livello di precisione che si vuole ottenere, sarà necessario procedere con un approccio *trial & error* e saranno tipicamente necessarie diverse modifiche ed aggiustamenti finali. Tuttavia, siccome in questo caso non stiamo andando ad animare le labbra del personaggio ma solo la sua lingua, è possibile procedere un po' più velocemente. Le tempistiche di questa fase variano molto in base alla lunghezza della traccia audio, alla complessità e al numero di parole presenti nel testo e al livello di precisione che si vuole ottenere. Il posizionamento corretto è facilitato da diversi elementi forniti dall'interfaccia del programma:

- la presenza della forma d'onda dell'audio che fornisce a livello visivo un'indicazione immediata delle pause e dell'attacco delle diverse parole.
- la struttura gerarchica a livelli che unisce frasi, parole e fonemi.
- la possibilità di avere un riferimento audio per i vari frame, in modo da verificare effettivamente la durata e la posizione di parole e fonemi.
- la possibilità di vedere una preview stilizzata dell'animazione ottenuta, in modo da verificare l'effettivo timing dei diversi fonemi.

Soddisfatti del posizionamento dei fonemi generati è possibile esportare il risultato in modo da poterlo utilizzare all'interno di Blender. Per fare questo, nella finestra *Current Voice*, sotto Export è necessario prima definire che tipologia di dati vogliamo esportare, le opzioni sono *MOHO*, *ALELO* e *Images*, per il nostro progetto bisogna selezionare *MOHO*. Dopo di che è possibile premere il bottone *Export...*, questo aprirà una nuova finestra che consentirà di nominare il file e di decidere dove andarlo a salvare.

Il file verrà creato nel formato `.dat` e sarà così formato: nella prima riga compare la scritta '`MohoSwitch1`'; dopo di che su ogni riga viene indicato, in ordine temporale di comparsa, il numero di frame e il fonema/visema relativo (es `1 rest`). Un esempio più completo di questo formato può essere consultato nell'appendice B.

Compiendo questi passi il file sarà esportato con successo; bisogna precisare che tale file DAT riguarda unicamente la corrispondenza frame-fonema e non il progetto totale sulla quale si è lavorato fino ad ora, pertanto per salvare il progetto di Papagayo-NG è necessario andare su *File* > *Save* (o *Save as*) oppure utilizzare lo shortcut *Ctrl+S*; il file verrà salvato con estensione `.pgo` e conterrà tutto il lavoro svolto all'interno di Papagayo-NG.

Dopo aver esportato con successo il file `.dat`, il lavoro su Papagayo-NG può considerarsi terminato e diviene possibile spostarsi su Blender in modo da integrare l'animazione della lingua all'interno del più ampio processo di animazione facciale. L'add-on di Blender *ReveRig*, che verrà utilizzato per gestire l'animazione della lingua, verrà trattato in modo approfondito nel capitolo

11 ma è necessario andare ad indicare alcuni prerequisiti necessari che il personaggio 3D, su cui si va a realizzare l'animazione, dovrà rispettare.

10.5 Blendshape della lingua in Blender

Per fare in modo che i dati raccolti all'interno del file dat precedentemente creato, possano essere utilizzati per animare la lingua di un personaggio, è necessario che il modello 3D di tale personaggio abbia configurate le blendshape corrispondenti ai visemi definiti dal set `tongue_position` (si veda la tabella 10.1), e che siano dotate della medesima nomenclatura.

Le sei blendshape previste (TDN, TH, J, R, KG, rest) devono essere create manualmente per ciascun personaggio, quest'operazione richiederà del tempo ma è un passaggio necessario se si vuole gestire l'animazione della lingua in maniera semiautomatica.

A seconda del personaggio su cui si lavora ci si potrebbe trovare in condizioni di partenza differenti, ovvero la geometria della lingua potrebbe essere unita ad altre geometrie come quella dei denti o dell'intero volto, oppure potrebbe corrispondere ad un oggetto a sé stante. Nel secondo caso non vi sono particolari impedimenti, mentre se ci si trova nella prima condizione sarà fondamentale lavorare unicamente sui vertici che compongono la geometria della lingua in modo da non influenzare il resto della mesh. Per far questo, un'opzione è quella di creare, nel caso non fosse già presente, un nuovo *Vertex Group* che corrisponda ai vertici della lingua, in modo da facilitarne la selezione in futuro.

Per aggiungere un *Vertex Group* ed assegnargli dei vertici è necessario selezionare la mesh, andare su *Properties* > *Object Data* > *Vertex Group Panel*, premere sul bottone contrassegnato dal simbolo + e rinominare il gruppo come desiderato (es Tongue); dopo di che bisogna entrare in *Edit Mode* e selezionare unicamente i vertici interessati, infine, sempre nel pannello *Vertex Group*, bisogna premere su *Assign*. In questo modo sarà possibile selezionare in modo pratico solo i vertici interessati in modo che le nuove shape key create non andranno a creare deformazioni in zone non desiderate, limitando così anche l'insorgenza di possibili interferenze.

È bene creare le blendshape relative ai movimenti della lingua solo dopo aver realizzato le 52 espressioni definite da ARKit (si veda la tabella a pagina 82), in quanto le nuove blendshape si baseranno anche sulla shape key *jawOpen* relativa all'apertura della mascella.

Dopo aver specificato in modo chiaro su quali vertici lavorare è possibile procedere con la creazione delle shape key. Le deformazioni generate dai vari visemi dovranno rispettare quanto indicato nella sezione 10.3; per avere un'idea più definita del risultato che si vuole ottenere si può fare riferimento alla figura 10.3. Tale figura rappresenta le sei blendshape individuate per l'animazione della lingua, nella loro massima deformazione, ovvero con influenza pari a 1. Come si può notare tali deformazioni appaiono estreme e troppo pronunciate, questo è dovuto al fatto che il movimento della lingua è in stretta relazione con l'apertura della bocca, infatti la deformazione massima per ogni blendshape è stata ottenuta impostando, come riferimento, la mascella completamente aperta (jawOpen con *value 1*) e ricreando la posizione corretta che la lingua deve assumere nelle rispettive blendshape. In figura 10.4 è possibile verificare che impostando i valori di jawOpen e della blendshape di riferimento a 1 si otterrà il risultato desiderato.

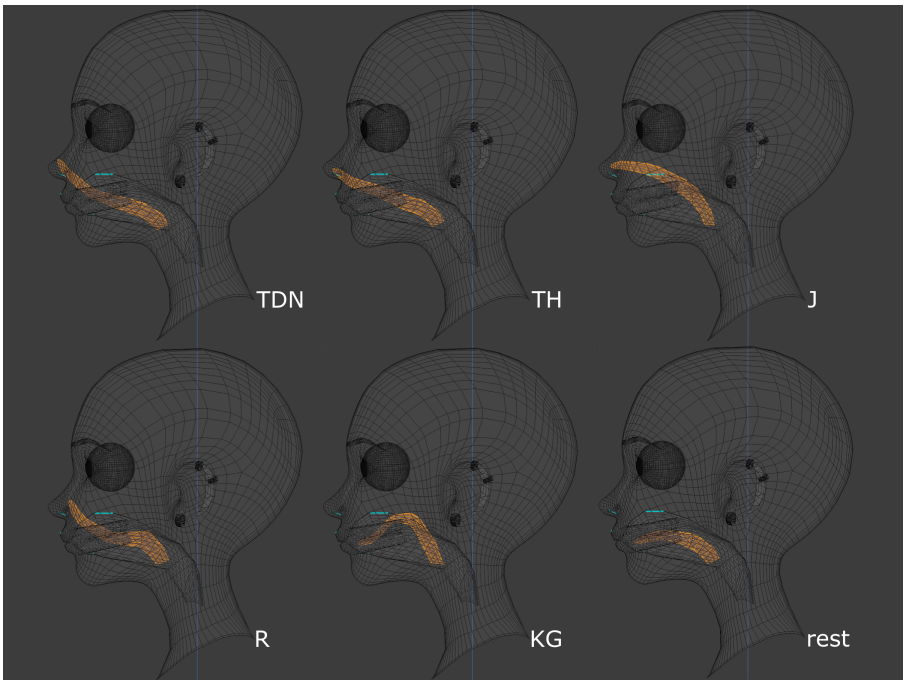


Figura 10.3: Le 6 blendshape per la lingua. In arancione viene mostrata la deformazione massima applicata per ciascuna blendshape.

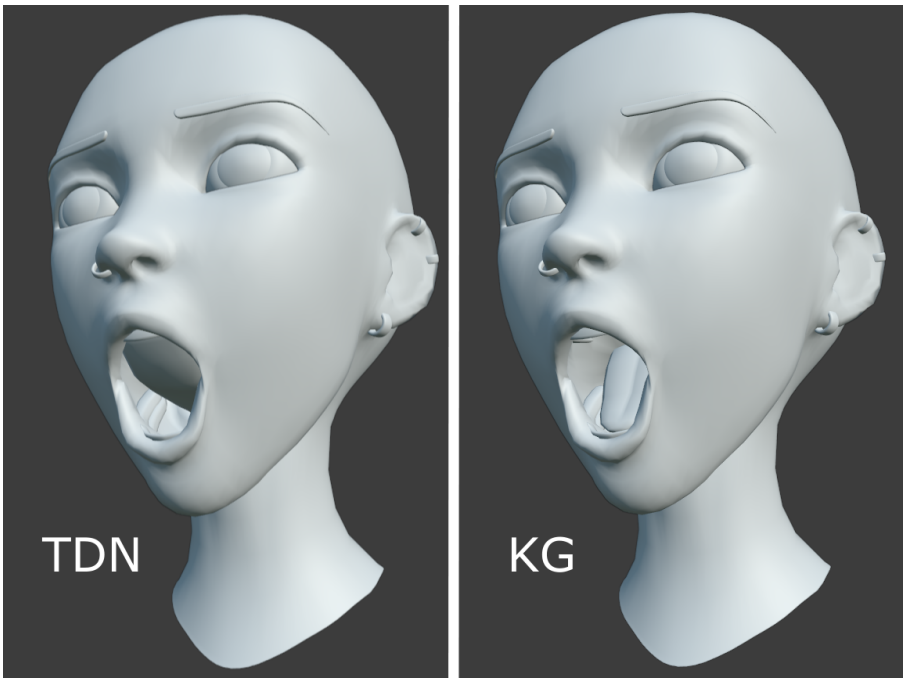


Figura 10.4: Risultato finale con la massima influenza della blendshape *jawOpen*. Nell'esempio le blendshape TDN e KG.

Tale dipendenza, tra apertura della mascella e blendshape della lingua, verrà trattata in maniera più dettagliata nel capitolo 11, per ora è sufficiente comprendere il meccanismo che porta alla corretta creazione delle sei blendshape relative all'animazione della lingua. La shape key denominata *rest*, corrisponde alla posizione di riposo della lingua, pertanto non prevede alcuna deformazione. Sebbene possa venire in mente di utilizzare la blendshape *Basis* al posto di *rest*, tale ragionamento non è corretto in quanto la blendshape *Basis* influenza anche vertici non appartenenti alla lingua. Per questa ragione è fondamentale creare, in ogni caso, una nuova blendshape denominata *rest* che agisca unicamente sulla geometria della lingua.

L'animazione della lingua diventa, quindi, parte integrante del workflow di lavoro nella realizzazione dell'animazione facciale per i personaggi della serie animata *Reverie Dawnfall*. Risulta comunque, almeno in parte, un lavoro separato dal resto del volto, questo consente di gestire in maniera migliore le risorse all'interno della produzione indipendente. Infatti, sebbene sia un processo semi-automatizzato, richiede comunque del lavoro manuale e va ad incidere sulle risorse sia a livello di personale che di tempo impiegati.

Essendo un processo in parte separato dal resto della pipeline di produzione, diventa possibile gestire in maniera più accurata le risorse a disposizione, in base alle necessità e al risultato che si vuole ottenere. Infatti, l'animazione della lingua diventa fondamentale quando l'inquadratura in una sequenza di dialogo risulta abbastanza stretta sul soggetto (es primi piani, primissimi piani e dettagli), mentre può risultare superflua in situazioni in cui il personaggio da animare si trova distante dalla camera e quindi il movimento della lingua non verrebbe percepito dallo spettatore. In quest'ultimo caso diventa possibile non procedere con l'animazione della lingua, andando a impiegare le risorse disponibili in fasi più significative della produzione. Questo introduce un enorme vantaggio sulla filiera produttiva, consentendo alla produzione maggiori margini di manovra.

Capitolo 11

ReveRig Add-On

Nei capitoli 9 e 10 sono state descritte le tecniche di acquisizione dei dati e della deformazione della geometria pensate per essere introdotte nella produzione di *Reverie Dawnfall*. Sebbene tali tecniche possano essere utilizzate, così come descritto precedentemente, senza andare ad effettuare ulteriori passaggi, per ottenere il miglior risultato possibile è necessario trovare soluzione a tutti quei limiti e quelle esigenze che l'utilizzo di tali procedimenti ha introdotto.

Il capitolo seguente tratta in modo approfondito le motivazioni, lo sviluppo e l'implementazione, all'interno della pipeline di lavoro, dell'Add-On di Blender *ReveRig*, nato e sviluppato all'interno di questo progetto di tesi. Viene dato, inoltre, spazio alle motivazioni che ci hanno spinto a distribuire a livello internazionale *ReveRig* e come questo abbia influenzato positivamente l'intero progetto, offrendo nuove possibilità per il futuro.

11.1 Esigenze

In questa sezione andremo ad individuare le principali esigenze e i principali limiti, nati dall'utilizzo delle tecniche di animazione delle espressioni facciali e della lingua, che hanno determinato la necessità di sviluppare degli strumenti appositi da poter essere inseriti all'interno del processo di lavoro. Tali limiti hanno diversa natura, alcuni consistono in un vero e proprio blocco nell'avanzamento dei lavori, altri riguardano la qualità del prodotto finale e altri ancora impediscono l'ottimizzazione delle risorse a disposizione. Per ciascuno dei limiti e delle esigenze sotto riportate, sono state individuate delle possibili soluzioni e appositi strumenti da poter inserire all'interno di Blender. In questo contesto ci limiteremo a descrivere le esigenze e i limiti da superare mentre lo sviluppo e l'implementazione degli strumenti verrà descritto nelle sezioni a seguire.

Organizzazione del progetto basato sul linking Uno dei fattori che maggiormente ha determinato la necessità di creare l'Add-On ReveRig riguarda la struttura organizzativa dei file che è stata definita per la produzione della serie animata. Come già riportato nella sottosezione 8.3.4, per la produzione di *Reverie Dawnfall* si è deciso di utilizzare uno strumento fornito da Blender stesso, ovvero il *linking*.

Tale tecnica consiste nel suddividere in distinti file `.blend` le diverse componenti che devono essere utilizzate in modo ricorrente all'interno della serie animata, come ad esempio gli ambienti e

i personaggi. In tale modo si introduce un grosso vantaggio in termini di facilità di gestione dei file di progetto, in quanto questi risulteranno più leggeri e facilmente lavorabili anche con hardware non specializzato. Inoltre, permette di creare un'organizzazione molto ordinata dei file di progetto, migliorandone l'usabilità. Tuttavia, si introducono delle limitazioni non sottovalutabili, in quanto, con questo meccanismo, si perde la possibilità di agire sulle deformazioni degli oggetti se non attraverso degli appositi oggetti chiamati *proxy*.

Ciò significa che, allo stato attuale, non ci è concesso di accedere in maniera diretta e completa alla mesh dei personaggi che vogliamo animare. Le blendshape generate precedentemente, ovvero le 52 relative alle espressioni facciali e le 6 relative alla posizione della lingua, risultano inaccessibili e pertanto inutilizzabili. Si viene a creare, quindi, un blocco al proseguimento del lavoro in quanto non è possibile trasferire i dati acquisiti, con tecnologia ARKit per il volto e con l'utilizzo di Papagayo-NG per la lingua, sul modello del personaggio che vogliamo animare.

Risulta fondamentale creare un sistema che funga da tramite tra dati acquisiti e blendshape sulle quali lavorare. Come vedremo nella sezione 11.3 la soluzione è stata trovata nella creazione di un apposito rig facciale.

Unire i due workflow di animazione Come visto nei capitoli 9 e 10, il processo di lavoro relativo all'animazione delle espressioni facciali e quello relativo all'animazione della lingua possono essere compiuti in modo separato, offrendo alla produzione maggiore spazio di manovra nella gestione delle risorse disponibili. Tuttavia, tra essi intercorre una relazione dovuta all'influenza che la blendshape jawOpen ha sulla determinazione dell'animazione finale della lingua. Diventa necessario, all'interno di Blender, legare questi due processi fino ad ora distinti.

Realizzare un prodotto di animazione stilizzato Ormai risulta chiaro che l'obiettivo finale è quello di realizzare un prodotto di animazione con "stile Pixar". Tuttavia, gli strumenti fino ad ora individuati, consentono solo in parte di raggiungere questo obiettivo.

Il sistema di motion capture adottato consente di catturare le espressioni facciali da un volto reale ma non fornisce dei metodi semplici e diretti per trasformare tali dati in espressioni più stilizzate. Ciò comporta la necessità di dover manipolare manualmente i dati ottenuti in modo da enfatizzare determinate deformazioni. Tuttavia, anche se all'interno di Blender ci sono strumenti ed interfacce dedicate alla gestione di keyframe e delle blendshape, il lavoro di manipolazione dei dati raccolti risulta un processo lungo e caratterizzato da scarsa intuitività. Questo richiede di dover agire in maniera diretta sulle curve di interpolazione o di animare manualmente le blendshape, processo che all'interno di una produzione seriale come quella di *Reverie Dawnfall* richiederebbe troppo tempo, gravando negativamente sul budget a disposizione.

Diventa necessario definire un'interfaccia che consenta di lavorare, in modo pratico e veloce, sui dati raccolti così da ottenere come risultato l'animazione voluta. La soluzione a questo problema si è trovata individuando diversi strumenti da utilizzare contemporaneamente. In particolare, l'animazione desiderata può essere ottenuta combinando da un lato la manipolazione dei dati acquisiti con i sistemi di mocap facciale attraverso l'utilizzo dell'add-on ReveRig; dall'altro l'utilizzo del rig facciale del personaggio che si sta animando creato precedentemente come descritto in [33].

Importare i dati per l'animazione della lingua Come visto nel capitolo 10, i dati relativi all'animazione della lingua vengono esportati da Papagayo-NG nel formato dat. Diventa necessario creare uno strumento interno a Blender che ci consenta di importare tale file e che sia in grado di leggere ed elaborare come desiderato i dati in esso racchiusi. In particolare, i dati importati devono consentire di poter animare le blendshape relative alla posizione della lingua. Per compiere queste operazioni sono state create apposite funzioni all'interno dell'add-on ReveRig.

Unificare la pipeline di lavoro Oltre alla necessità di unire i workflow relativi all'animazione delle espressioni facciali e della lingua, risulta utile racchiudere all'interno di un unico add-on tutti i principali strumenti che devono essere utilizzati all'interno di Blender per effettuare l'animazione desiderata.

ReveRig è stato pensato per soddisfare tale requisito. Al suo interno compaiono strumenti che consentono di agire in modo fine sulle singole blendshape e di gestire nuovi parametri appositamente creati per consentire all'animatore di lavorare con maggiore semplicità ed intuitività.

Alcune delle esigenze sopra riportate, sebbene siano state individuate in modo specifico per la produzione di *Reverie Dawnfall*, in realtà sono necessità che caratterizzano l'intero settore dell'animazione facciale computerizzata. Per questo motivo, si è deciso di procedere allo sviluppo di ReveRig avendo, da un lato come obiettivo principale quello di creare uno strumento appositamente pensato per essere inserito all'interno della produzione di *Reverie Dawnfall* e quindi porre rimedio ai limiti e alle esigenze sopra riportate; ma dall'altro lato, durante l'intero processo di sviluppo, non è mancata occasione di rivolgere lo sguardo anche al mercato globale nel settore dell'animazione facciale, soprattutto in riferimento alla community di Blender. È per questo motivo che ReveRig, con alcune necessarie semplificazioni rispetto a quanto verrà descritto nelle sezioni 11.3, 11.4 e 11.5, può rivelarsi (ed in parte si sta già rivelando) uno strumento di grande importanza, oltre che a Robin Studio, anche a tutti gli artisti digitali che utilizzano Blender come programma per l'animazione. I primi risultati di una distribuzione internazionale dell'Add-On ReveRig sono riportati nella sezione 11.6.

Siccome si è deciso che ReveRig debba comprendere, quanto più possibile, tutti i passaggi necessari per portare a termine l'animazione facciale, ne risulta che il numero di azioni che possono essere svolte tramite l'add-on è elevato. Questo potrebbe causare un conseguente aumento della complessità e una diminuzione drastica dell'usabilità di tali strumenti.

Per queste ragioni, oltre ad organizzare il codice sia a livello di impaginazione sia basandosi sul principio della singola responsabilità, come riportato nel capitolo 8.3.5, si è deciso di suddividere ReveRig in tre macro-sezioni separate (o pannelli), ciascuna delle quali racchiude le azioni principali relative a una o più fasi all'interno del processo di animazione (vedi figura 11.1). Tale suddivisione ha lo scopo di aumentare l'usabilità dell'add-on e di consentire all'animatore di gestire in modo più comodo ed intuitivo il suo lavoro.

ReveRig comprende: una prima sezione relativa alla creazione del rig facciale e alla sua unione con le geometrie che compongono il volto (vedi sezione 11.3); una seconda sezione relativa alla fase di retargeting perciò fornisce gli strumenti utili per trasferire i dati importati, relativi

all'animazione, sul modello da animare (vedi sezione 11.4); una terza ed ultima sezione che racchiude una serie di strumenti utili, come bottoni e slider, per consentire una manipolazione fine dell'animazione (vedi sezione 11.5).

Di seguito andremo a descrivere nel dettaglio come ciascuno di questi pennelli è stato sviluppato, come si compone la sua interfaccia utente (UI) e come l'animatore può sfruttare al meglio gli strumenti previsti per realizzare l'animazione facciale. Siccome ReveRig è pensato anche in un'ottica internazionale, si è deciso di sviluppare l'intero codice e l'UI facendo ampio utilizzo della lingua inglese. Per mantenere maggiore coerenza tra i contenuti di questo documento e le funzionalità presenti nell'add-on, di seguito alcuni termini verranno proposti nella stessa forma con la quale sono presenti nel codice sorgente di ReveRig.

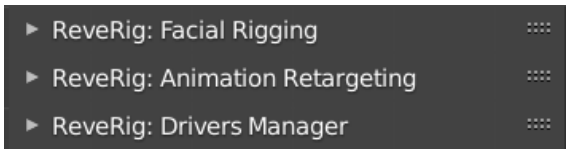


Figura 11.1: I tre pannelli principali di ReveRig

11.2 Organizzazione dei file e prerequisiti

Come detto nella sezione 11.1, siccome si è deciso di organizzare i file di progetto sfruttando la possibilità, che offre Blender, di poter effettuare il *Link* di un file `.blend` da una libreria esterna, è necessario creare un sistema che possa essere utilizzato come *oggetto proxy* all'interno della scena sulla quale si sta lavorando.

Per comprendere meglio l'intero meccanismo, si riporta di seguito l'organizzazione dei file adottata per la produzione della seconda versione del trailer di *Reverie Dawnfall*.

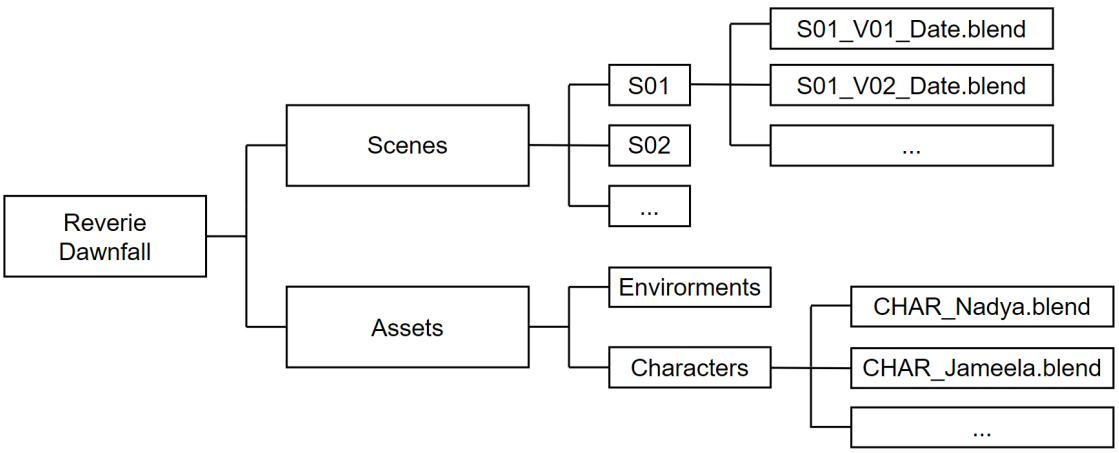


Figura 11.2: Schema semplificato dell'organizzazione dei file di progetto

I file di progetto si dividono in due principali categorie, gli asset e le scene, dalle quali sono state create due cartelle corrispondenti. All'interno della cartella `Assets` si trovano tutti quei file

di progetto che contengono elementi ricorrenti all'interno della produzione della serie animata. In particolare sono state create due cartelle apposite per contenere rispettivamente: i personaggi (denominata `Characters`), e gli ambienti (denominata `Enviroments`).

Nella cartella relativa alle scene (`Scenes`) vengono inseriti tutti i file di progetto relativi alla realizzazione di un'unica scena o di un'unica inquadratura (es `S01_V01_Date.blend`). Ciò significa che il vero e proprio lavoro di animazione verrà svolto unicamente all'interno di questi file. I file di progetto relativi ai personaggi (es `CHAR_Nadya.blend`) conterranno unicamente ciò che riguarda il personaggio stesso, ovvero il modello 3D, il rig corrispondente e altri oggetti necessari. In questo file il modello del personaggio dovrà essere mantenuto in quella che Blender chiama *Rest Position*, ossia la posa base; tipicamente si adotta una *T-pose* o una *A-pose*. È necessario inserire all'interno di questi file tutti gli strumenti che consentiranno di poter animare correttamente il personaggio una volta che questo sarà stato linkato all'interno di una scena.

Bisogna ricordare che tutte le modifiche apportate al file di origine si ripercuoteranno su tutti i progetti nei quali il personaggio è stato linkato. Mentre le modifiche apportate sui file di progetto locali, ovvero le scene, non andranno a modificare in alcun modo il file di origine. Ciò consente di poter linkare, all'interno di una moltitudine di scene, lo stesso file origine e di lavorare in modo del tutto indipendente tra una scena e l'altra. Diventa quindi possibile animare come desiderato il personaggio in questione.

Il primo pannello dell'add-on ReveRig è stato sviluppato appositamente per creare, per tutti i personaggi in modo semplice e veloce, un apposito rig facciale da poter utilizzare come oggetto proxy.

Data l'organizzazione dei file di progetto vista sopra e riportata in figura 11.2, il primo step necessario, ovvero la creazione del rig facciale, sarà effettuato per ciascun personaggio presente nella cartella `Characters` per il quale è prevista l'animazione tramite mocap facciale con tecnologia ARKit.

Di seguito si vanno a definire alcuni prerequisiti che il modello del personaggio su cui si vuole lavorare dovrà rispettare; nelle sezioni 11.3, 11.4 e 11.5 si descrive come le diverse parti dell'add-on siano state strutturate e sviluppate, andando ad individuare le componenti più significative del codice realizzato.

11.2.1 Prerequisiti

Il primo pannello di ReveRig (vedi sezione 11.3) comprende due fasi fondamentali per il corretto utilizzo dell'add-on. La prima riguarda la creazione automatica del rig facciale, la seconda comprende una sorta di operazione di *skinning*, in particolare si vanno ad associare alle diverse componenti del rig facciale precedentemente creato, le blendshape corrispondenti tramite un sofisticato utilizzo dei *driver*. Se per la creazione del rig facciale non vi è un'interazione diretta con il modello del personaggio, e quindi è possibile creare la struttura del rig in qualsiasi condizione; diverso è per la seconda fase. Siccome in questa si va a creare una relazione stretta tra rig e modello del personaggio, è fondamentale che quest'ultimo sia provvisto di tutti i requisiti necessari. La mancanza totale o parziale di tali prerequisiti comporta: nei migliori dei casi un funzionamento parziale dell'add-on e nel peggiore dei casi una sua totale inutilità.

Siccome il legame tra rig facciale e le deformazioni del volto è realizzato grazie all'utilizzo di driver per andare a guidare le diverse blendshape, il prerequisito principale riguarda proprio la presenza o meno di tutte le blendshape necessarie. Per il completo funzionamento di ReveRig

il personaggio dovrebbe essere dotato di tutte le blendshape viste nei capitoli 9 e 10, ovvero le cinquantadue blendshape previste dall'utilizzo dei sistemi di mocap con tecnologia ARKit e le sei blendshape per l'animazione della lingua. Queste risultano fondamentali per poter animare in modo completo il volto del personaggio. Tuttavia, per alcuni personaggi, ad esempio quelli secondari o le comparse, per questioni di budget non si ha la possibilità di gestire anche l'animazione della lingua, limitandosi quindi alle espressioni facciali. Questo, sebbene ponga dei limiti alle potenzialità di ReveRig, non inficia il corretto funzionamento delle altre componenti. In pratica, ReveRig è stato progettato per funzionare anche laddove le blendshape presenti siano in numero limitato; questo però comporta un risultato di minore qualità, dovuto proprio all'impossibilità di poter agire su determinate deformazioni.

ReveRig non possiede una conoscenza diretta del tipo di deformazione che una blendshape comporta, perciò per funzionare basa la relazione tra rig e blendshape unicamente sulla nomenclatura utilizzata. Diventa, quindi, fondamentale assegnare a ciascuna blendshape il nome corretto, in modo che ReveRig possa creare le giuste relazioni con le varie componenti del rig. Per le cinquantadue blendshape individuate da ARKit la nomenclatura che deve essere adottata è la stessa riportata nel capitolo 9.1, mentre per quanto riguarda le blendshape della lingua si deve rispettare quanto riportato nel capitolo 10.3.

Verificare la corretta corrispondenza tra nome della blendshape ed effettiva deformazione apportata è compito dell'animatore in quanto ReveRig non ha possibilità di effettuare controlli dei questo tipo. Tuttavia, se si utilizza Faceit per la creazione delle blendshape del volto (come visto nel capitolo 9.2.1), si ottiene automaticamente la nomenclatura corretta mentre per le sei blendshape relative alla lingua deve essere gestita manualmente.

Oltre alle blendshape sopra indicate, ReveRig prevede la possibilità di gestire quattro ulteriori blendshape relative alle azioni di estensione e compressione delle pupille. Sebbene il loro utilizzo possa consentire un risultato visivo migliore, rimangono degli elementi opzionali e quindi la loro mancanza non andrà ad inficiare sull'utilizzo dell'add-on. La tabella C.1 nell'appendice C riporta la corretta nomenclatura per tutte le 62 blendshape che ReveRig è in grado di gestire.

Dopo essersi assicurati di aver creato e nominato correttamente tutte le blendshape necessarie, un altro requisito fondamentale riguarda la presenza, o meglio la non presenza, di *driver* legati alle blendshape, sia per il campo `value`, sia per il campo `slider_max`. Questo perché, la seconda fase di questo primo pannello si occupa di creare i driver necessari per ciascuna blendshape, e la presenza di driver pregressi potrebbe generare malfunzionamenti nell'utilizzo dell'add-on. Perciò, laddove fossero già presenti dei driver, sarà necessario andare ad eliminarli.

Rimane comunque possibile creare ulteriori blendshape specifiche per ciascun personaggio. Infatti, ReveRig si limita a gestire al massimo le sessantadue blendshape sopra indicate, tutto ciò che viene aggiunto non avrà influenza sul funzionamento dell'add-on. Ovviamente, data la natura *open source* di ReveRig, per l'utente resta possibile apportare eventuali modifiche in modo tale da poter aggiungere ulteriori controlli al rig facciale e quindi gestire un numero maggiore di blendshape.

Nella sezione 11.3 si va a definire meglio il funzionamento della prima parte dell'Add-On.

11.3 Pannello 1 - Facial Rigging

Come già introdotto nella sezione 11.2, la prima sezione di ReveRig è a sua volta suddivisa in due fasi principali. La prima fase consiste nella creazione del rig facciale che sarà poi utilizzato per controllare le differenti blendshape. La seconda fase prevede di legare il rig appena creato con le blendshape corrispondenti di ogni geometria necessaria del volto.

L'obiettivo del rig che si andrà a creare è quello di fornire un'interfaccia semplice ed intuitiva all'animatore, in modo che si possa agire sul rig per controllare le diverse blendshape. Perciò, il rig creato non agirà in maniera diretta sui vertici della mesh del personaggio per creare le espressioni facciali ma fungerà da intermediario tra l'animatore e le blendshape, in particolare avrà lo scopo di essere un supporto sul quale sarà possibile applicare le animazioni di faccia e lingua precedentemente catturate.

In base a quanto riportato nel capitolo 4, l'interfaccia che si va a creare con il rig è di tipo *viewport 2D*, ovvero tutti i controller saranno disposti sullo stesso piano e saranno circoscritti da un contenitore rettangolare. Come vedremo nella sezione 11.5, ReveRig presenta anche un'interfaccia *window-based* utile per un controllo più fine dell'animazione.

Siccome l'obiettivo è quello di guidare i valori delle diverse blendshape, è risultato automatico pensare al rig facciale come una collezione di differenti *slider*, ciascuno legato alla blendshape corrispondente. In questo modo si ha una relazione immediata ed intuitiva tra traslazione dello slider del rig su un determinato asse e corrispondente cambiamento del valore di influenza della blendshape.

Per la creazione dell'interfaccia finale del rig di ReveRig sono statue effettuate diverse ricerche tra vari "competitor" del settore che hanno realizzato rig di tipo viewport 2D (es. Polywink, MocapX, Kinetic Motive e Dynamixyz). Tali ricerche sono state fondamentali per definire le forme, i colori ma soprattutto la posizione dei vari slider che comporranno il rig facciale.

11.3.1 Fase 1 - Creazione del rig facciale

Le componenti del codice che gestiscono questa prima fase si possono individuare in due classi principali. Un operatore che racchiude tutti i metodi e le funzioni necessarie per creare il rig facciale e una classe pannello con il compito di creare l'interfaccia grafica con la quale l'utente potrà interagire; quest'ultima verrà approfondita a pagina 121 in quanto comune anche con la fase di creazione dei driver; tuttavia si riporta in figura 11.3 la parte di UI inizialmente visibile all'utente.

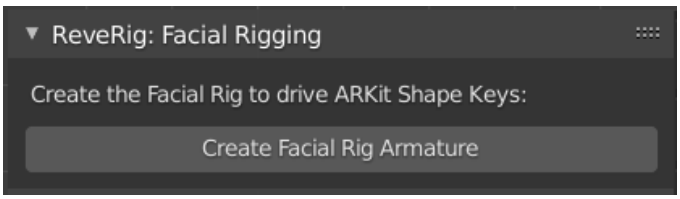


Figura 11.3: UI per la prima fase del pannello *Facial Rigging*

Classe FR_OP_ShapekeySliderCreator

La struttura dell'operatore `FR_OP_ShapekeySliderCreator` può essere riassunta come segue:

```
#OPERATOR CREATE ARMATURE FOR FACIAL RIG
class FR_OP_ShapekeySliderCreator(bpy.types.Operator):
  """Create the ARKit Shape Keys Slider Drivers Armature"""
  bl_idname = "object.arkssda"
  bl_label = "Create Facial Rig Armature"
  bl_options = {'REGISTER', 'UNDO'}
  # Method - Create Slider Armature
  def createSlider(self):
    #creazione e posizionamento degli slider...
  # Method - Create Text for Tongue slider, for better user interface
  def createText(self):
    #creazione e posizionamento del testo...
  # Method - Create Bones Custom Shape
  def createShapes(self):
    #definizione custom shapes...
  def execute(self, context):
    print("Creating the Slider Armature...")
    self.createShapes() #define bones custom shape
    self.createSlider() #create armature
    self.createText() #Text for Tongue Slider
    return {'FINISHED'}
```

La parte iniziale ha come scopo quello di definire alcune importanti caratteristiche dell'operatore. Nella terza riga, tra le tre virgolette, compare la scritta che verrà utilizzata come *tooltip* all'interno di Blender per gli oggetti del menu e i bottoni. Il tooltip ha lo scopo di fornire, su richiesta dell'utente, maggiori informazioni senza sovraccaricare eccessivamente l'UI. In questo modo è possibile mantenere un'interfaccia pulita senza però rinunciare alla completezza delle indicazioni fondamentali. Nella quarta riga viene specificato il `bl_idname` ovvero l'ID che sarà utilizzato per identificare univocamente l'operatore in modo da poterlo legare ai bottoni o alle componenti del menu che devono fare riferimento a questo operatore. Con `bl_label` invece si definisce il nome che verrà mostrato nell'interfaccia, infine `bl_options` abilita la funzione di *undo* sull'operatore.

Il compito dell'operatore è quello di eseguire tutto ciò che è contenuto nel metodo `execute`. All'interno di `execute`, quindi, vengono effettuate tutte le operazioni necessarie per portare a termine il compito[1]. Per questo operatore sono stati pensati tre metodi con il compito di creare varie componenti del rig facciale e sono eseguiti secondo il seguente ordine: `createShapes`, `createSlider`, `createText`.

[1]Quando il codice si fa più complesso è vantaggioso suddividere i diversi compiti in specifiche funzioni o metodi. In questo caso, dovuto alla necessità di dover mantenere maggior coerenza tra i dati utilizzati, si è deciso di optare per la definizione di metodi interni alla classe. Per le altre funzionalità dell'add-on, come vedremo meglio in seguito, si è deciso di creare delle funzioni esterne alla classe in modo da mantenere una migliore organizzazione del codice.

Metodo createShapes

Il primo metodo chiamato ha lo scopo di definire quelle che in Blender vengono chiamate *custom shapes*. Il rig che viene generato in questa prima fase dell'utilizzo di ReveRig, altro non è che un oggetto definito in Blender come di tipo *ARMATURE*, ovvero una struttura gerarchica composta da componenti articolate dette ossa, *bone* in Blender. Il principio è identico a quello che si cela dietro la creazione dello scheletro per animare il corpo di un personaggio. Tuttavia, la creazione di strutture complesse può limitare notevolmente l'usabilità del sistema, in quanto di default le ossa che compongono l'armatura saranno tutte quante simili tra loro.

Per facilitare le operazioni di animazione, Blender consente di assegnare a ciascun osso una custom shape ovvero una forma personalizzata, in modo che risulti più semplice distinguere i diversi giunti. Siccome il rig facciale che si vuole creare conterrà centinaia di ossa, che andranno a creare gli slider necessari per controllare le sessantadue blendshape previste, diventa necessario definire delle forme specifiche in base alle diverse caratteristiche che si vogliono evidenziare. Inoltre, visto che l'obiettivo è creare degli slider, le custom shape serviranno soprattutto per trasformare semplici ossa in sistemi composti da: un binario, o un'area di movimento, e un cursore. Nel proseguimento del documento il termine cursore verrà spesso sostituito con il più generico termine slider.

Sebbene il rig sarà composto da slider, vi è la necessità di crearne di diverse tipologie, ognuna realizzata in base a specifiche caratteristiche. Per la creazione del rig facciale sono state individuate sei differenti custom shape che devono essere realizzate (il risultato finale viene mostrato in figura 11.4b a pagina 120).

La prima, contenuta nella variabile `sliderObj`, è relativa alla forma del cursore stesso, questa sarà unica e uguale per tutti gli slider ed è definita dalla forma di un quadrato.

La seconda forma definita (`baseObj`) riguarda il binario di scorrimento per tutti gli slider monodimensionali con range compreso tra [0,1], dentro questa categoria rientrano la maggior parte degli slider che compongono il rig facciale, la forma prevista è un rettangolo in grado di contenere il cursore sia in larghezza sia nel compimento del suo percorso.

Per il controllo del movimento degli occhi, invece di andare ad utilizzare quattro differenti slider per ciascun occhio, si è deciso di creare un unico slider in grado di muoversi su due assi contemporaneamente, in modo da definire la posizione destra/sinistra e su/giù. Tale forma, salvata nella variabile `baseObj_Eye`, sarà rappresentata da un grosso quadrato in grado di contenere completamente tutti i possibili movimenti dello slider, ovvero [-1,1] su entrambi gli assi.

Una situazione simile si verifica per la gestione della mascella ma a differenza di quanto visto per gli occhi, in questo caso il cursore avrà un range [-1,1] sull'asse orizzontale e [0,1] su quello verticale; questo determina la forma rettangolare dell'area di azione del cursore, tale forma è memorizzata nella variabile `baseObj_Jaw`.

La penultima forma prevista è relativa al controllo delle pupille (`baseObj_Pupil`), siccome l'azione prevista può essere di compressione e di estensione si è deciso di creare un unico slider per pupilla con range [-1,1].

Infine, l'ultima custom shape prevista, nominata `boxObj`, riguarda l'osso che verrà utilizzato per creare il riquadro dentro il quale verrà costruito il rig; in questo modo è possibile dare un perimetro certo al rig facciale, rendendo l'interfaccia più comprensibile.

Grazie alla creazione delle custom shape descritte sopra, sarà possibile nella fase di creazione dell'armatura, legare le ossa alle rispettive forme, migliorando notevolmente l'interfaccia utente.

Metodo createSlider

Dopo aver definito le custom shape necessarie, ora è possibile creare effettivamente gli slider del rig. Tale compito è demandato al metodo `createSlider`. Dato l'elevato numero di ossa da dover gestire tale metodo risulta uno dei più lunghi in termini di righe di codice. Infatti, all'interno di `createSlider` si vanno a generare tutte le ossa necessarie per la creazione degli slider e per ciascuno di essi si va a definirne la corretta posizione e orientamento in modo da creare l'interfaccia desiderata. Il processo che ha portato alla definizione dell'interfaccia attuale è stato lungo e complesso, caratterizzato da un approccio di tipo *trial&error*. Molti sono stati i tentativi ma alla fine si è riusciti a creare un'interfaccia pulita, semplice e di facile comprensione (si veda la figura 11.4 a pagina 120). La struttura di questo metodo può essere suddivisa in diverse parti.

Nella prima parte viene creato l'oggetto armatura e vengono definite tutte le variabili che torneranno utili successivamente, come ad esempio dei semplici contatori. All'armatura viene legato il primo osso denominato `Box_Parent` al quale viene associata la custom shape `boxObj`. Sempre in questa prima parte vengono creati quattro nuovi *Bone Group* e per ciascuno di essi ne viene definito il colore. Infatti, oltre alla possibilità di creare delle custom shape, in Blender è possibile assegnare a ciascun osso uno specifico colore andandolo ad inserire all'interno del Bone Group relativo. L'utilizzo di differenti colori migliora notevolmente la facilità di utilizzo di un determinato rig. In questo caso particolare sono stati creati quattro gruppi di ossa così definiti:

- `Group_Slider` conterrà tutte le ossa relative ai cursori e il colore definito è il giallo (`THEME09`), in modo che risultino facilmente visibili dall'utente in quanto sono l'unica componente che deve essere manipolata per controllare le blendshape.
- `Group_Base_Right` definito dal colore rosso (`THEME01`) racchiude tutte le ossa relative al binario dello slider; all'interno del codice tali ossa verranno indicate come *base* o come *holder*, che sono legate a quelle blendshape che vanno ad agire unicamente sul lato destro del volto.
- `Group_Base_Left` definito dal colore blu (`THEME04`), comprende tutte le ossa base legate a blendshape che influenzano unicamente il lato sinistro del volto.
- `Group_Base_Neutral` racchiude tutte le ossa base relative a blendshape che non prevedono la deformazione solo da uno specifico lato del volto, come ad esempio l'osso `Base_brown-InnerUp` o tutte quelle legate al movimento della lingua. Per questo gruppo viene utilizzato il colore di default.

Nella seconda parte del metodo `createSlider` vengono create tutte le ossa utili alla composizione del rig facciale. L'elenco completo delle ossa create può essere visionato nell'appendice C nel codice C.1 . Per ciascuna blendshape che si ha intenzione di controllare è necessario creare almeno due differenti ossa imparentate tra loro; un osso *base* nominato con la convenzione `Base_blendshapeName` e un osso *slider* ad esso imparentato nominato secondo la convenzione `Slider_blendshapeName`. In questo modo è possibile definire la traslazione, sui diversi assi, dell'osso slider rispetto alla posizione dell'osso base corrispondente.

Tale processo di creazione è stato implementato andando prima a definire una lista (`list_Bones`) contenente tutti i nomi delle sessantadue blendshape previste da ReveRig e, come vedremo, alcune aggiuntive, dopo di che tramite un ciclo iterativo attraverso tale lista (`for elem in self.list_Bones`) si è andati a creare e organizzare in modo corretto le ossa necessarie. Come prima cosa le ossa vengono create entrando in *Edit Mode*, questo consente di dare una prima organizzazione ai vari giunti e di impostarne la parentela. Tuttavia, sarà necessaria una fase successiva in *Pose Mode* per posizionare nel modo definitivo gli slider.

Per facilitare la comprensione del codice e per organizzare meglio la distribuzione delle ossa nell'UI, si è deciso di dividere in macrogruppi la lista sopra definita. Il primo relativo alle sopracciglia, il secondo relativo ad occhi e pupille, un terzo riguardante guance e naso, il quarto riguardante le blendshape della bocca, un quinto per le azioni della mascella ed infine l'ultimo gruppo relativo alle blendshape della lingua. Questo ha permesso di lavorare secondo un'organizzazione a "blocchi" consentendo di aumentare la comprensibilità del codice. Tale organizzazione in blocchi può essere in parte notata anche nel risultato finale dell'interfaccia (vedi figura 11.4 a pagina 120).

Sempre all'interno del ciclo `for`, si va ad attribuire per ciascun osso la relativa custom shape e si va ad inserire l'osso all'interno del *Bone Group* corretto. Di seguito viene riportato un esempio di codice utilizzato.

```
slidered.custom_shape=bpy.data.objects[self.sliderObj.name] #assing cs
slidered.bone_group= group_S #add to Slider Bone Group
```

Inoltre, è fondamentale definire per le ossa di tipo slider i vincoli necessari. In generale tali vincoli sono relativi alla possibilità di traslazione sui diversi assi. Ciò che si fa è inserire un *constraint* di tipo `LIMIT_LOCATION`. In questo modo è possibile definire su quali assi il cursore ha possibilità di muoversi e definirne il range di azione. La maggior parte degli slider può muoversi su un unico asse con un range [0,1] in modo da rispettare il range di default utilizzato nella creazione delle blendshape. Tuttavia, come visto sopra, alcuni slider devono poter avere la possibilità di muoversi contemporaneamente su due assi. Il codice seguente mostra un esempio relativo all'inserimento dei vincoli per gli slider "classici" ovvero con movimento su un solo asse e con range [0,1].

```
#Insert Constraint
pMid = bpy.context.object.pose.bones[Slider_name]
cns1 = pMid.constraints.new('LIMIT_LOCATION')
cns1.name = 'LocLimit'
cns1.owner_space = 'LOCAL'
cns1.use_min_y = True
cns1.min_y = 0.0
cns1.use_max_y = True
cns1.max_y = 1
cns1.use_transform_limit = True
```

Dopo averne definito forma, colore ed eventuali limiti è possibile procedere con il posizionamento finale delle diverse ossa in *Pose Mode* in modo da comporre l'interfaccia definitiva. Come detto sopra, il lavoro relativo al corretto posizionamento delle ossa ha richiesto diversi tentativi e diversi giorni di lavoro, andando di volta in volta a rendere l'UI sempre più pulita e dal design più comprensibile. Per posizionare le ossa è sufficiente agire sull'osso *base* (nominato in questa parte del codice come `holder`) in quanto lo slider corrispondente, essendo imparentato ad esso,

ne seguirà i movimenti. Di seguito si riporta una parte di codice relativa al posizionamento finale delle ossa relative allo slider per il controllo della blendshape `browOuterUpRight`.

```
if elem == 'browOuterUpRight':
  # Set rotation mode to Euler XYZ, easier to understand
  # than default quaternions
  holder.rotation_mode = 'XYZ'
  # select axis in ['X','Y','Z']  <--bone local
  axis = 'Z'
  angle = 20
  holder.rotation_euler.rotate_axis(axis, math.radians(angle))
  holder.location = (0,-0.2,0)
```

Il metodo `createSlider` termina andando a impostare la posizione corrente dell'armatura, ovvero quella definita muovendo le ossa in *Pose Mode*, come *Rest Position*. Senza questo passaggio, quando un personaggio viene linkato all'interno di una scena, il rig facciale avrà l'interfaccia provvisoria creata nel momento dell'aggiunta delle ossa in *Edit Mode*, e non quella definitiva.

```
bpy.ops.pose.armature_apply(selected=False) #Set Rest Position
```

Come accennato precedentemente, le ossa che si andranno a creare saranno in numero maggiore rispetto a quanto necessario per le sessantadue blendshape. Questo è dovuto al fatto che bisogna conciliare da un lato la miglior interfaccia possibile e dall'altro le funzionalità e i controlli disponibili. Come vedremo meglio dopo, ad ogni osso è stata aggiunta una nuova proprietà denominata `amp_value`, tale valore verrà utilizzato in fase di animazione per gestire l'ampiezza delle varie blendshape. Questo è stato introdotto per soddisfare la necessità, vista a pagina 108, di dover creare animazioni più stilizzate e quindi con espressioni facciali più enfatizzate .

Ovviamente l'`amp_value` di uno specifico osso andrà ad influenzare la blendshape corrispondente; tuttavia, per la necessità di mantenere un UI pulita, alcune blendshape vengono controllate da un unico osso, come ad esempio accade per gli occhi e per la mascella. Questo non consentirebbe di gestire l'ampiezza delle blendshape in modo indipendente ma sarebbero tutte guidate da un unico `amp_value`.

Per risolvere tale limite si è deciso di creare almeno un osso slider per ogni blendshape seguendo la convenzione `Slider_blendshapeName`, dopo di che sono state create nuove ossa appositamente pensate per essere utilizzate come interfaccia. Tali ossa sono `eyeLookAllRight` e `eyeLookAllLeft` che servono per controllare le blendshape `eyeLookDown`, `eyeLookUp`, `eyeLookOut` e `eyeLookIn` sia Right che Left. Un altro osso appositamente creato per questioni di interfaccia è `jawAll` che consente di controllare contemporaneamente `jawRight`, `jawLeft` e `jawOpen`. Inoltre, si è deciso di creare una copia aggiuntiva delle ossa `eyeLookAllRight` e `eyeLookAllLeft` nominandole con la convenzione `Slider_Eyes_boneName` in modo tale da consentire un controllo maggiore sugli occhi. In particolare, questo consente di gestire le palpebre e l'occhio vero e proprio in maniera separata, aumentando così le possibili espressioni che un animatore è in grado di creare.

L'ultima parte del metodo `createSlider`, si occupa proprio di nascondere dall'interfaccia finale le ossa relative a quelle blendshape che verranno poi controllate dalle apposite ossa viste sopra. In questo modo si otterrà l'interfaccia desiderata senza dover rinunciare alla possibilità di un controllo fine di ciascuna blendshape.

Come si può notare dalla figura 11.4 a pagina 120, si è deciso di organizzare gli slider seguendo a grandi linee le caratteristiche principali del volto. Posizionando i vari slider in

maniera simmetrica rispetto all'asse verticale, andando a orientare e posizionare i vari slider secondo l'azione che la blendshape corrispondente comporta. In questo modo si ha un UI più intuitiva e la funzione di ogni slider può essere semplicemente dedotta osservandone posizione e orientamento. In questo modo è possibile limitare al minimo l'utilizzo di testo all'interno dell'interfaccia, mantenendo così tutto il più pulito possibile. L'utilizzo delle diverse forme e dei diversi colori introduce un ulteriore miglioramento alla facilità d'uso generale del rig. Anche con questa organizzazione è possibile che la funzione di alcuni slider non sia di immediata comprensione, tuttavia è sufficiente un limitato periodo di utilizzo del rig per poter prendere maggiore confidenza con lo strumento.

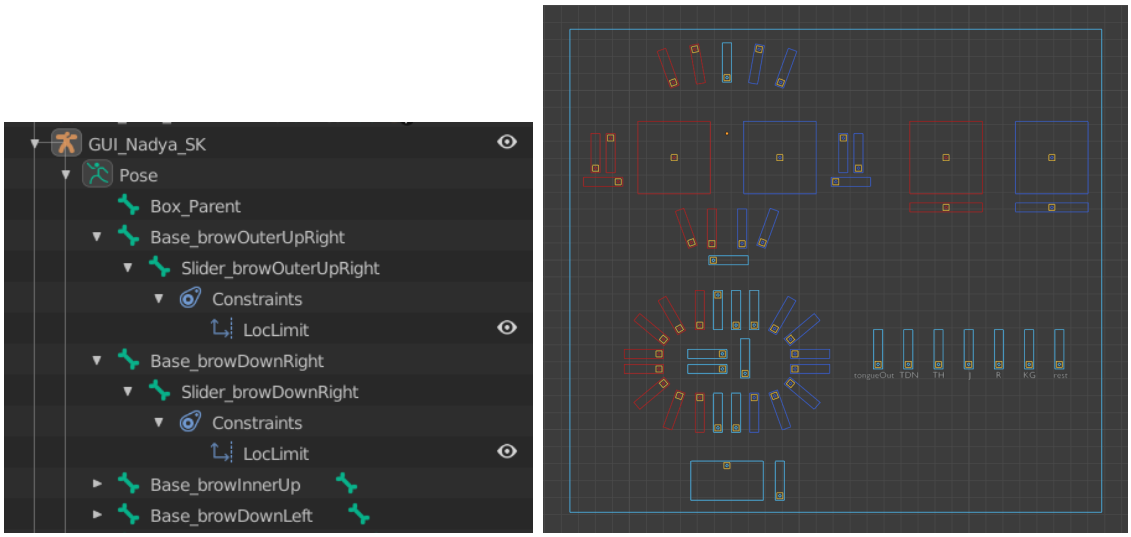
Metodo createText

Sebbene si sia scelto di sfruttare la posizione, orientamento, colore e forma degli slider per guidare l'utente all'utilizzo del rig, per alcuni slider risulta necessario creare un riferimento testuale. In particolare per i sette slider relativi all'animazione della lingua, in quanto diventa difficile trovare un'alternativa unicamente visiva che sia in grado di definire in modo chiaro la funzione di questi slider. Per questi si è deciso, quindi, di creare una sorta di sezione a parte, dove vengono rappresentati in modo del tutto impersonale. Per garantire l'usabilità dell'UI, a questi slider è stato aggiunto un riferimento testuale, ovvero per ogni slider si è andati a riportare nelle immediate vicinanze il nome della blendshape di cui ha il controllo (`tongueOut`, `TDN`, `TH`, `J`, `R`, `KG`, `rest`). In questo modo diventa facile per l'utente comprenderne il funzionamento e l'interfaccia rimane sostanzialmente invariata. Con questo metodo terminano le principali operazioni svolte dall'operatore `FR_OP_ShapekeySliderCreator`.

Terminate le operazioni svolte dall'operatore `FR_OP_ShapekeySliderCreator` si viene a creare il rig facciale utilizzato da ReveRig. L'oggetto relativo al rig sarà un oggetto di tipo armatura avente di default nome `GUI_Char_SK`, dove `SK` serve a dare maggior riferimento al fatto che tale rig ha il compito di guidare delle shape key. In questo modo diventa possibile adattare il nome del rig a quello del personaggio a cui si andrà a legare mantenendo però la stessa convenzione. Ad esempio, per il personaggio di Nadya di *Reverie Dawnfall* si potrà rinominare il rig come `GUI_Nadya_SK`.

Nella figura 11.4a viene mostrato parte dell'outliner relativo all'armatura del rig facciale. Come si può notare si ha una struttura parzialmente gerarchica in quanto per ogni blendshape si ha un osso base e un osso slider ad esso imparentato. Vedremo meglio nel seguito che la convenzione che si è scelta per andare a nominare le differenti ossa (`Base_bendshapeName` e `Slider_blendshapeName`) risulta di fondamentale importanza per i passaggi successivi. Infatti, in questo modo diventa possibile discriminare facilmente tra ossa base e slider ed è possibile mantenere un costante riferimento alla blendshape corrispondente.

Tutto ciò che è stato descritto sopra rimane compito dell'operatore `FR_OP_ShapekeySlider-Creator`, ciò significa che l'utente che utilizzerà ReveRig tramite interfaccia non avrà diretta conoscenza su quanto compiuto ma all'animatore sarà sufficiente premere un bottone sull'interfaccia e attendere che il rig facciale compaia a schermo. Il processo è completamente automatizzato, in questo modo si riduce estremamente il lavoro che dovrà essere svolto dall'animatore andando a limitare significativamente il peso sul budget disponibile.



(a) Outline del rig facciale di ReveRig

(b) Interfaccia del rig facciale di ReveRig

Figura 11.4: Rig facciale di ReveRig.

Dopo aver creato il rig facciale è possibile passare alla seconda fase del pannello *Facial Rigging* dell'interfaccia di ReveRig, ovvero alla creazione del legame tra rig e blendshape tramite la generazione di driver.

11.3.2 Fase 2 - Creazione dei driver

Il compito di questa seconda fase consiste nell'andare a indicare le mesh delle varie componenti del volto interessate e per ciascuna di esse creare tutti i driver necessari, in modo da poter utilizzare il rig precedentemente creato per controllare le diverse deformazioni di volto, occhi, lingua ecc. Come per la creazione del rig, anche in questo caso le operazioni sono demandate a specifiche componenti del codice in modo da rispettare il più possibile il principio della singola responsabilità. A comporre tale fase entrano in gioco tre classi e due funzioni:

- La classe `MyProperties` che è di tipo *PropertyGroup* ed è utilizzata per creare nuove proprietà all'interno di Blender.
- Il pannello `CSD_PT_CreateArmaturePanel` che racchiude tutta l'interfaccia per la prima sezione di ReveRig e quindi anche il bottone dell'operatore per la creazione del rig facciale.
- L'operatore `FR_OP_ShapekeyDriverCreator` dentro il quale viene iniziata l'esecuzione del codice per la vera e propria creazione dei driver.
- Una funzione `createDrivers` che viene chiamata all'interno dell'operatore indicato sopra.
- Una funzione `createDriversFunction` che viene chiamata all'interno di `createDrivers` ed è stata introdotta per mantenere il codice più ordinato e pulito.

Di seguito si va a definire più nel dettaglio come ciascuna di queste componenti è stata realizzata, descrivendo come sono composte e di che cosa si occupano.

Classe MyProperties

Per il corretto svolgimento di questa seconda fase è divenuto necessario creare nuove proprietà all'interno di Blender, in modo tale da poter avere degli strumenti che consentano di registrare più oggetti contemporaneamente. Questa classe verrà utilizzata anche nella fase di retargeting (vedi sezione 11.4) ma per il momento ci si concentrerà su quanto d'interesse per questa fase. Le righe di codice a seguire mostrano come tale classe è definita e composta, e come sia stata registrata in modo tale che le nuove proprietà siano effettivamente create ed utilizzabili all'interno di Blender.

```
### MY PROPERTIES  (useful to bind drivers correctly)
class MyProperties(bpy.types.PropertyGroup):
  face_target : bpy.props.PointerProperty(name='Face',type=bpy.types.Object)
  eyes_target : bpy.props.PointerProperty(name='Eyes',type=bpy.types.Object)
  tongue_target : bpy.props.PointerProperty(name='Tongue',type=bpy.types.
    Object)
  upper_teeth : bpy.props.PointerProperty(name='Upper Teeth',type=bpy.types.
    Object)
  lower_teeth : bpy.props.PointerProperty(name='Lower Teeth',type=bpy.types.
    Object)
#Some code...
def register():
  bpy.types.Scene.my_settings = bpy.props.PointerProperty(type=MyProperties)
def unregister():
  del bpy.types.Scene.my_settings
```

Sono state create cinque nuove proprietà di tipo puntatore con lo scopo di essere utilizzate, per l'appunto, come puntatore verso contenuti di Blender di tipo oggetto. Da come è possibile evincere dai nomi utilizzati, tali puntatori verranno usati per dare dei riferimenti a ReveRig rispetto alle mesh con le quali si deve creare il legame con il rig facciale tramite creazione di driver. Basandosi su quanto visto per Faceit e su quanto si è potuto conoscere attraverso l'esperienza personale, si è deciso di creare cinque puntatori per le principali geometrie del volto ovvero: faccia, occhi, lingua, denti superiori e denti inferiori; ad eccezione di faccia e occhi, le restanti geometrie sono facoltative.

Per registrare correttamente tali proprietà, nella funzione `register` è stato necessario creare una nuova proprietà chiamata `my_settings` legata al contesto della scena (`bpy.types.Scene.my_settings`), anche essa di tipo puntatore, utilizzata per fare riferimento alla classe `MyProperties`. In questo modo le diverse proprietà diventano raggiungibili all'interno del codice ad esempio con l'indicazione `context.scene.my_settings.face_target`. Nella funzione `unregister` si procede con l'eliminazione delle proprietà create in modo tale che, qualora l'add-on fosse disabilitato, tali proprietà non siano più presenti all'interno di Blender.

Classe CSD_PT_CreateArmaturePanel

All'interno di tale classe viene racchiuso tutto il codice utile per creare correttamente l'interfaccia relativa al primo pannello di ReveRig. La struttura interna può essere semplificata come segue.

```
### Layout Panel - Create Facial Rig and create Shape Key Drivers, all based
    on ARKit shape keys
class CSD_PT_CreateArmaturePanel(bpy.types.Panel):
  bl_space_type = 'VIEW_3D'
```

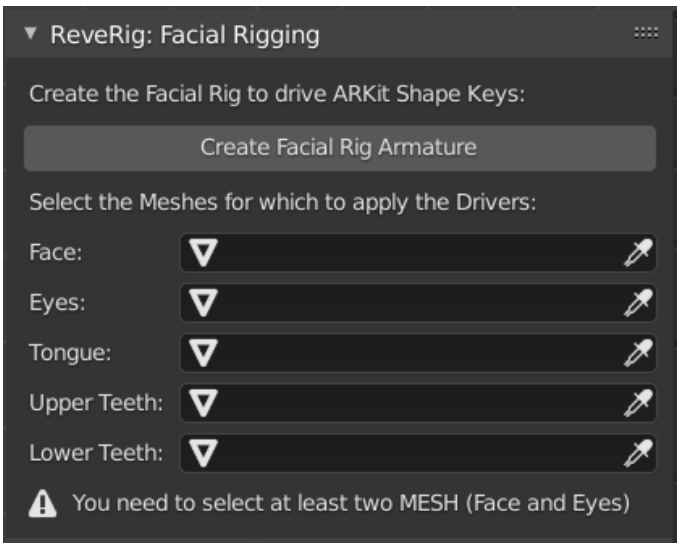
```
  bl_region_type = 'UI'
  bl_category = "REVERIG"
  bl_options = {"DEFAULT_CLOSED"}
  bl_label = "ReveRig: Facial Rigging"
  bl_idname = "CREATEDRIVERS_PT_layout"

  def draw(self, context):
    layout.label(text="Create the Facial Rig to drive ARKit Shape Keys:")
    row = layout.row()
    row.scale_y = 1.2
    row.operator("object.arkssda") #Button to create the amrature of Facial
    Rig
    #Can see this part only if I have selected an ARMATURE
    if obj_active:
      if obj_active.type =='ARMATURE':
        layout.label(text="Select the Meshes for which to apply the Drivers:
    ")
        #Le seguenti 2 linee di codice sono state ripetute per le 5
    proprietà. Sostituire "property_name" con: "face_target", "eyes_target",
    "tongue_target", "upper_teeth" e "lower_teeth".
        row = layout.row()
        row.prop_search(my_tool, "property_name", context.scene, "objects",
    icon="OUTLINER_OB_MESH") #Search Bar with picker

        #Perform checks on the selected items
        #All items should be MESH and at least two items should be selected
    (face and eyes)
        if procedi_1==False: layout.label(text="You need to select at least
    two MESH (Face and Eyes)", icon='ERROR')
        if procedi_2==False: layout.label(text="You need to select only MESH
     objects", icon='ERROR')
        elif procedi_1==True and procedi_2==True:
          layout.label(text="Press Button to create Shape Keys Drivers:")
          #...
          row.operator("object.arkscd",icon='DRIVER') #Button to create
    shape key drivers
```

Nella prima parte, come visto per l'operatore `FR_OP_ShapekeySliderCreator`, si vanno a definire alcune caratteriste fondamentali per il pannello. Il campo `bl_idname` serve ad indicare un ID univoco per il pannello, il campo `bl_label` viene usato per indicare il testo, relativo al nome del pannello, che verrà mostrato nell'interfaccia. Con `bl_space_type` si va ad indicare dove il pannello dovrà comparire all'interno di Blender, in questo caso si è deciso di inserirlo nella *View 3D*, specificando con `bl_region_type` l'esatta zona in cui dovrà comparire. Fondamentale nel caso di ReveRig specificare per ogni pannello il campo `bl_category`, in quanto andare ad indicare per ciascun pannello dell'add-on la stessa categoria, in questo caso chiamata come l'add-on stesso, ovvero `REVERIG`, consente di creare un unico gruppo dentro il quale inserire l'intera interfaccia dell'add-on. Altrimenti i tre diversi pannelli creati per ReveRig sarebbero potuti essere separati tra loro, andando così ad inficiare notevolmente sull'usabilità complessiva dell'interfaccia.

Dopo la fase iniziale di definizione della classe, inizia il metodo `draw`. All'interno di questo metodo si andrà a comporre la vera e propria interfaccia del pannello *Facial Rigging*. Dopo una necessaria fase di inizializzazione di diverse variabili, si è andati a creare il bottone relativo

all'operatore `FR_OP_ShapekeySliderCreator`, come si può notare si è deciso di renderlo sempre visibile in quanto la creazione del rig facciale non necessita di particolari condizioni da dover rispettare e, in questo modo, risulta più facile per l'animatore capire come procedere.

Diverso invece è stato deciso per la fase di creazione dei driver. Infatti, in questo caso sono stati inseriti dei controlli che consentono di poter visualizzare l'interfaccia completa, e quindi di proseguire con l'operazione di creazione dei driver, solo nel caso in cui i vincoli siano rispettati. In particolare, sono stati creati due livelli differenti di visualizzazione, il primo riguarda la possibilità di poter accedere all'inserimento delle diverse mesh del volto, il secondo riguarda la possibilità di poter raggiungere il bottone legato all'operatore `FR_OP_ShapekeyDriverCreator` che consente di creare effettivamente i driver. Il primo vincolo è più blando in quanto è sufficiente che l'oggetto attivo sia di tipo `ARMATURE`, questo porta automaticamente l'animatore a selezionare il rig facciale precedentemente creato. Nel caso sia rispettato questo primo vincolo allora l'interfaccia mostrerà cinque barre di ricerca corrispondenti alle cinque proprietà create precedentemente. Per agevolare l'operazione di selezione degli oggetti, l'interfaccia fornisce per ciascuna barra di ricerca anche uno strumento chiamato *picker* che consente di indicare la mesh semplicemente andando a selezionarla dall'outliner o dalla view 3D. La selezione delle mesh corrette spetta all'utente, infatti ReveRig non è in grado di discriminare in anticipo se la mesh selezionata sia effettivamente quella corretta. Tuttavia, l'operazione di selezione è guidata dall'interfaccia tramite alcune specifiche indicazioni proposte sotto forma di testo. Inizialmente, quando nessuna mesh è stata ancora impostata, l'UI si presenta come rappresentato in figura 11.5a.



(a) Messaggio che invita l'utente a riempire i campi di selezione.

(b) Messaggio che invita l'utente a selezionare solo oggetti di tipo MESH.

Figura 11.5: UI pannello *Facial Rigging* in presenza di errori

Dove, al posto del bottone che consente di procedere con l'operazione di creazione dei driver, viene mostrato un messaggio dove viene riportato che è necessario indicare almeno due oggetti di tipo `MESH` relativi al campo *Face* ed *Eyes*. L'indicazione di tali mesh è obbligatoria in quanto

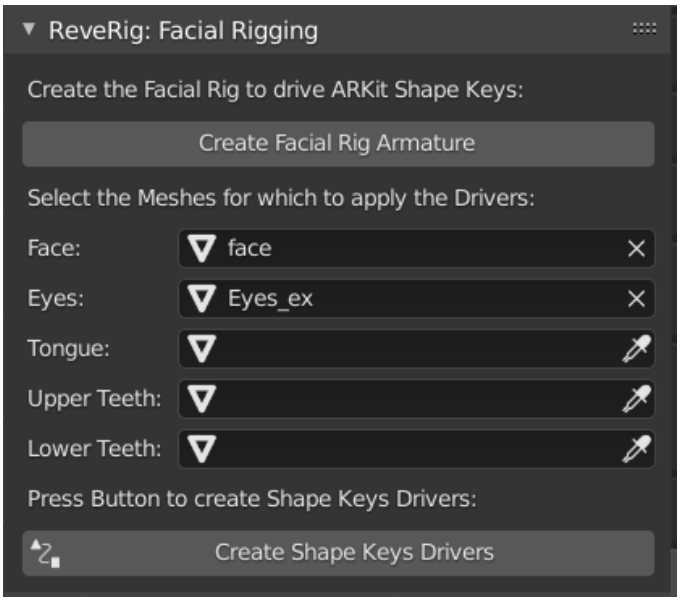


Figura 11.6: UI completa del pannello *Facial Rigging*

altrimenti l'utilizzo di ReveRig sarebbe di quasi nulla utilità. È fondamentale indicare almeno la geometria del volto, alla quale sono legate le cinquantadue blendshape individuate da ARKit, e la geometria degli occhi perché, come indicato precedentemente, la gestione di palpebre ed occhi avviene con componenti differenti del rig facciale. La possibilità di indicare le geometrie relative a lingua, denti superiori e inferiori rimane facoltativa in quanto non tutti i personaggi sono dotati di oggetti separati per ogni componente della faccia. Grazie a questa interfaccia risulta di immediata comprensione la necessità di dover indicare le mesh necessarie. Un altro importante vincolo è che ReveRig è in grado di lavorare solo con le blendshape e quindi con oggetti di tipo `MESH`, per questo motivo, qualora uno o più oggetti selezionati e riportati all'interno delle relative barre di ricerca fossero di differente natura verrà mostrato il messaggio di avvertenza presentato in figura 11.5b.

Nel momento in cui tutti i vincoli sopra citati siano rispettati, allora l'interfaccia mostrerà il bottone che consente di proseguire con la creazione dei driver tramite la chiamata all'operatore `FR_OP_ShapekeyDriverCreator`. La figura 11.6 riporta l'UI che consente di proseguire con la creazione del legame tra rig e mesh.

Classe FR_OP_ShapekeyDriverCreator

L'operatore che si occupa di eseguire il codice utile per la creazione dei driver è così composto.

```
### OPERATOR CREATE DRIVERS ############################
class FR_OP_ShapekeyDriverCreator(bpy.types.Operator):
  """Create the Drivers for the Value and Slider Max properties of the ARKit
     Shape Keys"""
  bl_idname = "object.arkscd"
  bl_label = "Create Shape Keys Drivers"
  bl_options = {'REGISTER', 'UNDO'}

```

```
  def execute(self, context):
    createDrivers(context)
    return {'FINISHED'}
```

A differenza di quanto fatto per l'operatore `FR_OP_ShapekeySliderCreator` in questo caso si è deciso di creare delle funzioni esterne alla classe in modo tale da consentire una migliore organizzazione del codice. In particolare, dopo una prima fase di definizione dell'operatore, del tutto simile a quanto visto precedentemente, il metodo `execute` avrà l'unico compito di chiamare la funzione `createDrivers`.

Funzione createDrivers e createDriversFunction

Queste due funzioni sono il vero cuore dentro le quali avviene l'effettiva creazione dei driver. La struttura interna della funzione `createDrivers` può essere così riassunta:

```
### Method - Create Shape Keys Drivers
def createDrivers(context):
  armature_target = bpy.context.active_object
  scene= bpy.context.scene

  dTarget_Face = scene.my_settings.face_target
  dTarget_Eyes = scene.my_settings.eyes_target
  dTarget_Tongue = scene.my_settings.tongue_target
  dTarget_UpperTeeth = scene.my_settings.upper_teeth
  dTarget_LowerTeeth = scene.my_settings.lower_teeth
  #inizializzazione liste...
  #(list_sk[...], list_pupil[...], list_tongue[...], list_jaw[...], list_eye
    [...], list_eye_y[...], list_negative[...])
  #Face
  createDriversFunction(armature_target,dTarget_Face,list_sk,list_tongue,
    list_jaw,list_eye,list_eye_y,list_negative)
  #Eyes
  for block in dTarget_Eyes.data.shape_keys.key_blocks:
    if block.name in list_sk:
      #creazione driver per gli occhi...
  #Tongue
  if dTarget_Tongue:
    createDriversFunction(armature_target,dTarget_Tongue,list_sk,list_tongue
    ,list_jaw,list_eye,list_eye_y,list_negative)
  #UpperTeeth
  if dTarget_UpperTeeth:
    createDriversFunction(armature_target,dTarget_UpperTeeth,list_sk,
    list_tongue,list_jaw,list_eye,list_eye_y,list_negative)
  #LowerTeeth
  if dTarget_LowerTeeth:
    createDriversFunction(armature_target,dTarget_LowerTeeth,list_sk,
    list_tongue,list_jaw,list_eye,list_eye_y,list_negative)
```

Vi è una prima parte dove si inizializzano tutte le variabili necessarie e dove si definiscono una serie di liste, contenenti i nomi delle blendshape alle quali dovranno essere associati i driver, molto utili per gestire in maniera più pratica il codice successivo. Nella seconda parte si crea, per ciascuna delle cinque potenziali mesh inserite tramite UI, l'effettivo legame con il rig facciale. Per mantenere il codice più pulito e leggibile si è deciso di creare un'ulteriore funzione,

chiamata `createDriversFunction`, con il compito di racchiudere tutto il codice necessario per la creazione dei driver. Tuttavia, siccome il processo di creazione dei driver per la mesh relativa agli occhi necessita di esigenze specifiche, si è deciso di separare questa parte di codice rispetto a quello realizzato per le altre mesh. Quindi, per le mesh di faccia, lingua, denti superiori e inferiori si procederà con la chiamata alla funzione `createDriversFunction` passandogli tutti gli elementi necessari per il corretto funzionamento, mentre per quanto riguarda gli occhi si andrà a sviluppare il codice direttamente dentro `createDrivers`. Le necessità peculiari della mesh relativa agli occhi sono dovute principalmente a come si è deciso di costruire il rig facciale. In particolare, per controllare le azioni degli occhi sono stati creati quattro slider appositi, due sono per il controllo delle pupille mentre gli altri due altro non sono che dei "doppioni" di quelli utilizzati per gestire anche le altre mesh. Questi doppioni sono stati nominati secondo la convenzione `Slider_Eyes_boneName`, in questo modo è possibile discriminarli in maniera facile dagli altri. Ad eccezione di queste peculiarità, il codice di base che è stato sviluppato per creare i driver in modo corretto è del tutto simile.

Di seguito si va ad identificare i punti principali del codice che sono necessari per portare a termine questo compito, in particolare si fa riferimento a quanto realizzato per la funzione `createDriversFunction`.

I driver che si ha necessità di creare sono due per ciascuna delle blendshape che si vogliono controllare tramite rig. Il primo driver serve per legare l'azione di un osso slider con il campo *Value* della blendshape corrispondente, il secondo serve per legare sempre l'osso di tipo slider al campo *Range Max* della blendshape.

Per quanto riguarda il campo `value` di una blendshape, ciò che si va a fare è creare un driver di tipologia `SCRIPTED`, che consente quindi di creare delle vere e proprie espressioni matematiche per gestire in modo corretto il comportamento del driver stesso. Per realizzare l'espressione da noi voluta è necessario creare due variabili, una nominata `var_slider` e la seconda nominata `var_amp`. La prima è di tipo `TRANSFORMS` il che consente di legare ad essa informazioni relative alla traslazione di un determinato osso. Come si farebbe andando a creare il driver tramite interfaccia, anche con il codice è necessario indicare per `var_slider` diversi parametri. Prima di tutto è necessario indicare l'oggetto a cui fare riferimento, o meglio l'ID di tale oggetto; nel nostro caso l'oggetto a cui si farà sempre riferimento è l'oggetto di tipo armature relativo al rig facciale, ovvero `GUI_Char_SK`. Dopo aver definito come oggetto target il rig facciale è necessario indicare l'osso specifico a cui si deve fare riferimento. Da questo punto sono necessari da fare alcuni controlli in base alla tipologia di slider e alla nomenclatura utilizzata; tuttavia, per maggior chiarezza del discorso, di seguito definiamo il processo che porta alla creazione dei driver per la maggior parte delle blendshape, una discussione relativa alle singole necessità verrà fatta al termine del discorso. Nella maggior parte dei casi l'osso di riferimento è indicato secondo la seguente convenzione: `Slider_blendshapeName`; ciò consente di scorrere facilmente lo stack delle blendshape di una determinata mesh e di anteporre semplicemente la stringa '`Slider_`' al nome della blendshape per poter ottenere l'osso corrispondente. Dopo aver definito l'osso target è necessario individuare il tipo di trasformazione che `var_slider` dovrà registrare; per la maggior parte delle blendshape tale trasformazione sarà di tipo `LOC_Y` ovvero si andranno a registrare le informazioni relative alla traslazione sull'asse y. Inoltre, è necessario indicare lo spazio di riferimento, nel nostro caso sarà per tutte le blendshape `LOCAL_SPACE`. Termina così la definizione della prima variabile.

Differente è il processo che porta alla creazione della seconda variabile, `var_amp`. Questa, a differenza della prima, sarà di tipo `SINGLE_PROP`, il che necessita di specificare l'oggetto di riferimento e l'`RNA` della proprietà che si vuole utilizzare. Nel nostro caso, per tutte le blendshape l'oggetto di riferimento sarà sempre l'armatura `GUI_Char_SK`, come nel caso precedente, mentre la proprietà sarà `amp_value`, una proprietà di tipo floating point appositamente creata per l'add-on ReveRig. Il percorso (`data_path`) relativo a tale proprietà sarà indicato tramite la riga di codice `pose.bones["'+bone_name+'"].amp_value`, dove `bone_name` altro non è che la variabile contenente il nome dell'osso slider basato sulla convenzione vista precedentemente.

Dopo aver creato le due variabili è possibile definire l'espressione che va a caratterizzare il comportamento del driver. Tale espressione, per la maggior parte delle blendshape, sarà definita come `var_slider.name +'*'+ var_amp.name`, ovvero il campo *Value* di una blendshape sarà guidato dal prodotto tra il valore della traslazione dello slider e il valore della proprietà `amp_value` dello stesso osso. Ricordiamo che il range di traslazione per ciascuno slider è stato impostato nella fase di creazioni del rig e tipicamente è limitato tra [0,1], esattamente come il campo `value` delle blendshape. La moltiplicazione con `amp_value` garantisce all'animatore la possibilità di gestire in modo facile l'amplificazione, o la riduzione, dell'effetto di una determinata blendshape. Tale sistema è stato aggiunto per consentire di realizzare espressioni facciali stilizzate in modo semplice e intuitivo.

Come accennato precedentemente, per alcune blendshape si ha necessità di apportare alcune modifiche al processo sopra descritto. In particolare, questo è necessario per tutte le blendshape legate a slider particolari come quelli per il controllo di occhi, pupille e mascella. In questi casi un unico slider controlla più blendshape e il range di movimento può variare tra [-1,1] e su due possibili assi. Da questo deriva la necessità di definire in modo differente alcuni parametri per alcune blendshape, ad esempio sostituendo `LOC_Y` con `LOC_X` e inserendo un segno negativo prima dell'espressione del driver. L'obiettivo finale, infatti, è quello di controllare il campo `value` delle blendshape e tale campo di default varia tra [0,1]; pertanto il driver così creato deve essere in grado di agire su di esso nel modo corretto. Per ovvie questioni di organizzazione del documento non è possibile riportare per intero tutte le possibili varianti realizzate.

Il range di default della proprietà `value` di una blendshape varia tra [0,1]. Tuttavia, Blender offre la possibilità di modificare tali valori in modo da soddisfare specifiche esigenze. È quindi possibile agire sui campi *Range Min* e *Max* per variare il campo di azione di una determinata blendshape e quindi andando potenzialmente ad ampliare la deformazione apportata. Tuttavia, visto che per la produzione di *Reverie Dawnfall* si è deciso di utilizzare per i file un sistema basato sul *Link*, la manipolazione diretta di questi campi diventa inaccessibile. Siccome è tra le nostre necessità quella di garantire la possibilità di creare animazioni stilizzate, diventa molto utile poter ampliare il range di azione di una blendshape per ottenere deformazioni più accentuate. Infatti, la sola introduzione della proprietà `amp_value` non garantisce la possibilità di esagerare a piacimento una determinata espressione, in quanto questa sarà comunque limitata dal range di azione della proprietà `value`. Siccome `amp_value` altro non è che un fattore moltiplicativo aggiunto all'animazione di una blendshape, diviene logico utilizzare lo stesso valore di ampiezza per controllare il range di `value`. Ovvero si può utilizzare `amp_value` per guidare tramite driver il valore della proprietà `slider_max` (*Range Max*) di una blendshape. Come vedremo nella sezione 11.5, il campo `amp_value` per ciascuna blendshape sarà presente all'interno dell'UI di ReveRig, in questo modo le deformazioni massime di una determinata espressione saranno sempre sotto il controllo dell'animatore.

La creazione del driver per la proprietà `slider_max` è simile a quanto visto precedentemente. Il driver è anch'esso di tipo `SCRIPTED` e necessita la creazione di un'unica variabile, chiamata anche in questo caso `var_amp`. Tale variabile è di tipo `SINGLE_PROP`, ha come oggetto target l'armatura `GUI_Char_SK` e per ciascuna blendshape è necessario indicare il `data_path` corretto che è identico a quello usato nella variabile `var_amp` del driver precedente. In questo caso l'espressione è molto semplice e contiene unicamente le informazioni relative ad `amp_value`.

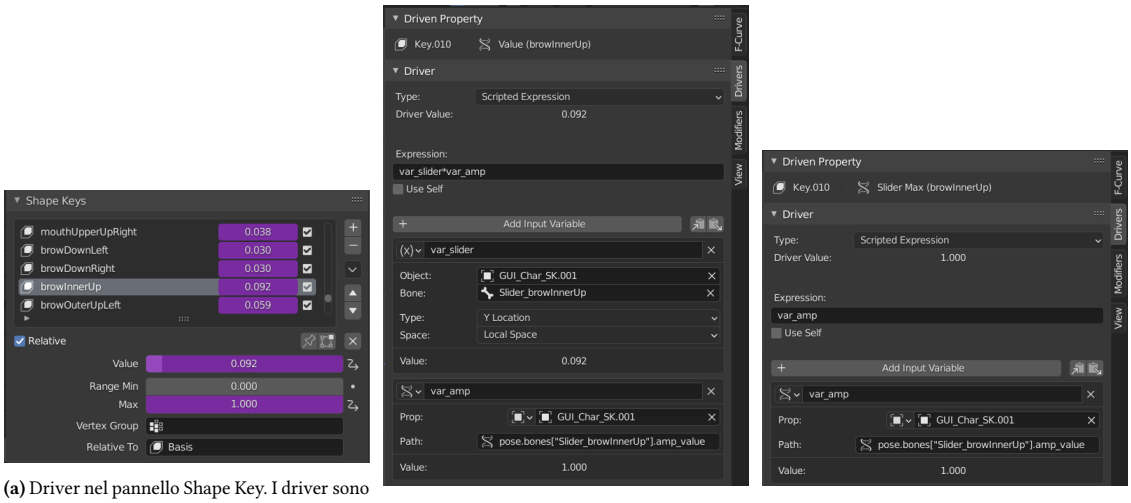
Tale procedimento viene effettuato per tutte le blendshape di tutte le mesh precedentemente individuate. Per ottimizzare tale processo si è deciso di effettuare un ciclo `for` per scorrere tutte le blendshape presenti all'interno dello stack di una determinata mesh. Per ciascuna di esse se ne estrae il nome che viene utilizzato all'interno del codice per effettuare i controlli e per trovare in modo semplice l'osso corrispondente. I controlli principali vengono effettuati grazie all'ausilio delle liste precedentemente definite, che consentono di discriminare in modo comodo le blendshape con differenti necessità. Di seguito si riporta la parte iniziale della funzione `createDriversFunction`.

```
### Functions - Create Shape Keys Drivers
def createDriversFunction(armature_target,mesh_target,list_sk,list_tongue,
    list_jaw,list_eye,list_eye_y,list_negative):

  for block in mesh_target.data.shape_keys.key_blocks:
    if block.name in list_sk+list_tongue:
      #Resto del codice...
```

La variabile `armature_target` contiene un riferimento all'armatura del rig facciale, che nel momento di creazione dei driver sarà l'oggetto attivo della scena. La variabile `mesh_target`, invece indica la mesh per la quale dovranno essere creati i driver, ricordiamo che le mesh possono essere relative al volto, lingua, denti superiori e inferiori mentre quella relativa agli occhi viene gestita direttamente nelle funzione `createDrivers`.

Dopo aver eseguito la creazione dei driver è possibile verificarne il corretto esito controllando le diverse blendshape delle mesh individuate. Il campo *Value* e il campo *Range Max* devono comparire come colorati con un color viola, come è possibile vedere nella figura 11.7a. Per effettuare un controllo più approfondito è possibile recarsi nell'interfaccia relativa ai driver (shortcut *Shift+F6*), selezionare la mesh interessata e nel pannello a sinistra selezionare uno dei driver che vengono mostrati. Per i driver relativi al campo *Value* deve essere possibile ottenere una schermata simile a quella rappresentata in figura 11.7b. Mentre per quanto riguarda i driver relativi al campo *Slider Max* si deve ottenere quanto indicato nella figura 11.7c.

Con la corretta creazione dei driver termina il ruolo svolto dal primo pannello di ReveRig (nominato *Facial Rigging*), diventa ora possibile spostarsi sui file di progetto relativi alle scene, dentro i quali è stato linkato il personaggio per il quale è stato creato il rig facciale, e procedere con la vera e propria animazione facciale.



(a) Driver nel pannello Shape Key. I driver sono evidenziati con il colore viola

(b) Driver per il campo *Value*.

(c) Driver per il campo *Slider Max*.

Figura 11.7: Esempio di driver creati con ReveRig.

11.4 Pannello 2 - Animation Retargeting

La seconda sezione dell'interfaccia di ReveRig, chiamata *Animation Retargeting*, racchiude tutti gli strumenti utili per effettuare l'operazione di *retargeting* sia dell'animazione del volto sia dell'animazione della lingua. Come abbiamo visto nella sezione 11.1, il lavoro svolto per la creazione dei dati relativi all'animazione del volto e il lavoro svolto per l'animazione della lingua sono stati fino ad ora due processi pressoché distinti e indipendenti tra loro. Tuttavia, per una migliore gestione del processo di animazione, è necessario che queste due componenti distinte vengano unite, in modo tale da poter proseguire con un processo di lavoro più lineare. ReveRig offre la possibilità di unire sotto un unico strumento la capacità di gestire sia l'animazione della lingua che delle espressioni facciali, agevolando notevolmente il lavoro dell'animatore.

Le operazioni di retargeting vengono presentate in un pannello apposito dell'UI perché sono considerate un passaggio opzionale. Ovvero per poter utilizzare ReveRig nella fase di animazione non è necessario che l'animazione che si vuole realizzare sia stata catturata con strumenti di mocap o di altro tipo. ReveRig offre la possibilità di poter agire tramite apposito rig facciale e apposita interfaccia per animare manualmente il volto, basandosi sulla tecnica dell'animazione con keyframe vista nel capitolo 5.1.1. Tuttavia, per i motivi ampiamente descritti nel capitolo 7.3, per la realizzazione della serie animata *Reverie Dawnfall* si è deciso di utilizzare sistemi che permettono di semi-automatizzare il processo di animazione. Da questo deriva la necessità di offrire degli strumenti che consentano di trasferire in modo semplice e veloce i dati precedentemente creati, sul personaggio che si vuole animare. Siccome per gestire il progetto si è deciso di organizzare i file andando ad utilizzare la tecnica di linking vista nella sezione 11.3, l'operazione di trasferimento dell'animazione non può più essere effettuata direttamente sulle blendshape delle mesh del volto, ma sarà necessario applicare i dati al rig facciale precedentemente creato, e appositamente pensato per agevolare il processo di retargeting.

Il pannello nominato *Animation Retargeting* di ReveRig, comprende a sua volta due principali fasi: la prima per il trasferimento dell'animazione delle espressioni facciali basate sul sistema ARKit, la seconda per il trasferimento dell'animazione della lingua basata sul file `.dat` creato

con Papagayo-NG. Per migliorare l'usabilità dell'add-on si è deciso di separare queste due fasi tramite l'utilizzo di un sistema gerarchico di pannelli; infatti, il trasferimento dell'animazione della lingua viene effettuato tramite un pannello figlio del pannello relativo alla prima fase. Nella figura 11.8 è possibile vedere un esempio completo dell'UI.

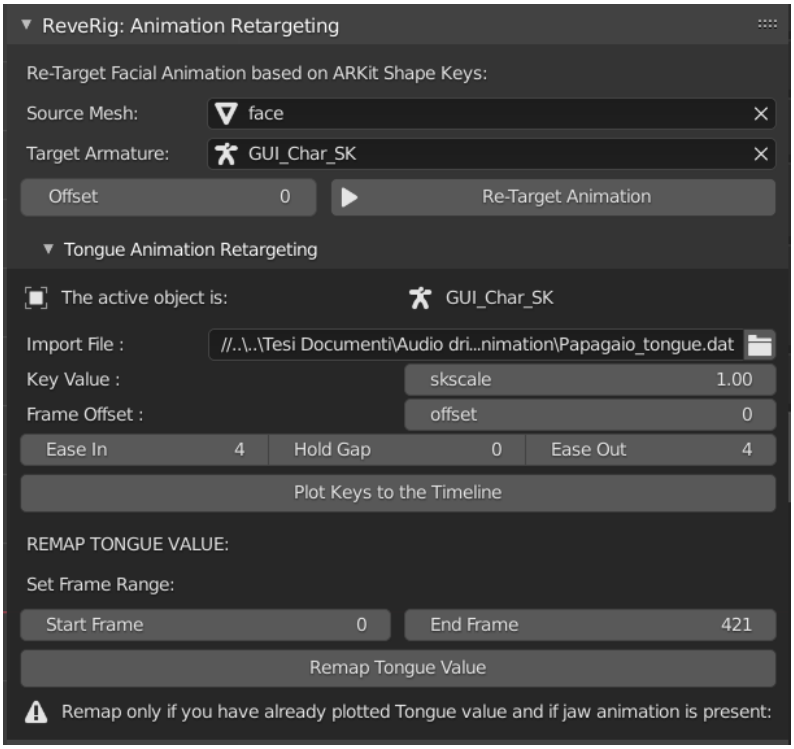


Figura 11.8: UI competa del pannello *Animation Retargeting*.

11.4.1 Fase 1 - Retargeting dei dati acquisiti con il mocap

La prima fase di questo secondo pannello di ReveRig è concepita per consentire di trasferire in modo semplice e intuitivo i dati relativi all'animazione delle cinquantadue blendshape definite da ARKit, dall'oggetto importato come file fbx, creato con gli appositi strumenti visti nel capitolo 9.1, al rig facciale precedentemente creato. Infatti, per effettuare le operazioni di cattura delle espressioni facciali sono state individuate due applicazioni principali: Face Capture di Rokoko e FaceCap di Bannaflak. Entrambe consentono di creare ed esportare l'animazione catturata tramite un file di tipo fbx. Sebbene vi siano alcune piccole differenze, entrambi i file prodotti propongono le informazioni secondo una struttura del tutto simile (vedi capitolo 9.1). Ovvero, il file FBX generato contiene un modello 3D rappresentante una faccia, tale modello è dotato di tutte le cinquantadue blendshape previste dalla tecnologia ARKit. Per ciascuna delle blendshape del modello 3D importato viene importata anche l'animazione corrispondente. Questa è basata dal susseguirsi dei dati registrati precedentemente, tali dati infatti vengono trasformati in keyframe e utilizzati per definire l'animazione risultante. Per controllare il numero e la frequenza dei keyframe che compongono l'animazione per il modello importato, è sufficiente selezionare il modello in questione e andare sul *Graph Editor*. All'interno di questa finestra è possibile

vedere tutte le curve di interpolazione (IPO) relative alla proprietà *Value* di ciascuna blendshape. Nella prima immagine della figura 11.9 è possibile vedere un esempio di curva IPO relativa alla blendshape `brownInnerUp` e nella seconda immagine in figura 11.9 è possibile vedere l'insieme totale di tutte le curve che vanno a definire l'intera animazione catturata tramite i sistemi di mocap visti prima.

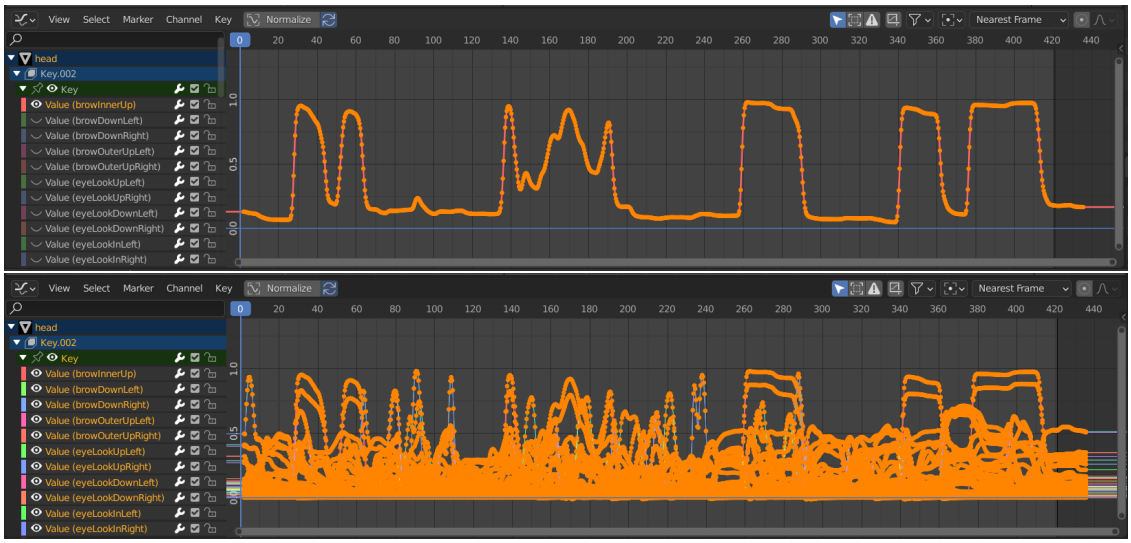


Figura 11.9: Esempio di curve di interpolazione per l'animazione delle blendshape

Come si può intuire il numero di curve e di keyframe da dover trasferire sul rig facciale è molto elevato. All'interno di Blender le curve di interpolazione relative all'animazione di una qualche proprietà vengono chiamate *F-curve*, questo termine verrà molto utilizzato all'interno del codice dell'add-on.

A comporre questa prima fase entrano in gioco tre classi e due funzioni:

- La classe `MyProperties`, già introdotta a pagina 121, alla quale sono state aggiunte due proprietà appositamente pensate per agevolare il retargeting.
- Il pannello `CA_PT_CopyFCurvesAnimation` che racchiude tutta l'interfaccia utile per consentire all'animatore di portare a termine questa fase.
- L'operatore `ANIM_OP_CopyFCurvesAnimation` che ha il compito di eseguire il codice che consentirà di trasferire l'animazione da sorgente a target.
- La funzione `insertKeyFrames` che ha il compito di creare un primo keyframe per ciascuna componente del rig facciale.
- La funzione `copyFCurvesAnimation` che racchiude il codice che effettivamente permette di copiare l'animazione dalla sorgente al target.

Di seguito definiamo più nel dettaglio come ciascuna di queste componenti è stata realizzata, descrivendo come sono composte e di che cosa si occupano.

Classe MyProperties

Per sapere come tale classe è stata creata, definita e registrata, il lettore può fare riferimento a quanto scritto a pagina 121. Infatti, la classe `MyProperties` è esattamente quella descritta

precedentemente con la semplice aggiunta di due ulteriori proprietà anch'esse di tipo puntatore verso elementi di tipo oggetto. Queste sono state introdotte per agevolare le operazioni di definizione di *oggetto sorgente* e *oggetto target* per effettuare il trasferimento dei dati. Le proprietà create sono così definite:

```
copyFc_target : bpy.props.PointerProperty(name='Target Armature',type=bpy.
    types.Object)
copyFc_source : bpy.props.PointerProperty(name='Source Mesh',type=bpy.types.
    Object)
```

In questo modo le due proprietà diventano raggiungibili all'interno del codice ad esempio con l'indicazione `context.scene.my_settings.copyFc_source`. In particolare, la proprietà della scena nominata `copyFc_target` viene utilizzata per registrare l'oggetto che verrà utilizzato come target mentre `copyFc_source` serve a registrare, come è possibile intuire, l'oggetto sorgente.

Classe CA_PT_CopyFCurvesAnimation

All'interno di tale classe viene racchiuso tutto il codice utile per creare correttamente l'interfaccia relativa al secondo pannello di ReveRig. Come già anticipato, la seconda sezione di ReveRig è composta da una struttura gerarchica di pannelli. La classe `CA_PT_CopyFCurvesAnimation` copre il ruolo di pannello principale e di pannello padre rispetto a quello che verrà definito successivamente quando si parlerà di animazione della lingua. Ciò significa che l'UI creata da `CA_PT_CopyFCurvesAnimation` sarà la prima componente visibile dell'interfaccia. Tale struttura è utile per mantenere l'UI più pulita ed organizzata ma anche per suggerire all'utente di eseguire le diverse fasi secondo l'ordine previsto, evitando in questo modo possibili errori. La struttura interna di questa classe può essere semplificata come segue.

```
### Layout Panel - Retarget Facial Animation from MESH with ARKit ShapeKeys
    (face) to ARMATURE target (facial rig with drivers)
class CA_PT_CopyFCurvesAnimation(bpy.types.Panel):
  bl_space_type = 'VIEW_3D'
  bl_region_type = 'UI'
  bl_category = "REVERIG"
  bl_options = {"DEFAULT_CLOSED"}
  bl_label = "ReveRig: Animation Retargeting"
  bl_idname = "COPY_FC_ANIMATION_PT_layout"

  def draw(self, context):
    #...
    layout.label(text="Re-Target Facial Animation based on ARKit Shape Keys:
    ")
    # Le 2 righe di codice a seguire sono ripetute due volte andando a
    sostituire a "property_name": "copyFc_source" e "copyFc_target".
    row = layout.row()
    row.prop_search(my_tool, "property_name", context.scene, "objects", icon
    ="OUTLINER_OB_MESH") #Search Bar with picker

    #Perform checks on the selected items
    #Source need to be a MESH and target need to be an ARMATURE
    if my_tool.copyFc_source and my_tool.copyFc_target:
      if my_tool.copyFc_source.type=='MESH' and my_tool.copyFc_target.type==
    'ARMATURE':
```

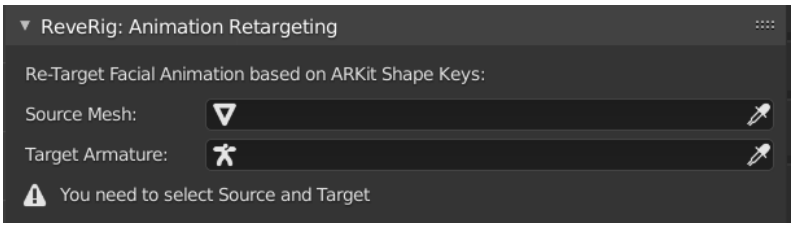
```
            #...
            col.prop(scene, "frame_current",text='Offset')
            #...
            col.operator("animation.copyfcanim", icon='PLAY')
        else: # Messaggio di errore 1.
    else: # Messaggio di errore 2.
```

Nella prima parte, come sempre, si definiscono alcune informazioni fondamentali relative alla classe stessa. I campi previsti sono identici a quelli visti per la classe `CSD_PT_CreateArmature-Panel`. Si può notare come anche per questo pannello, si sia indicata la categoria "`REVERIG`", questo consente di creare tutti i pannelli dell'interfaccia all'interno di una stessa scheda di visualizzazione.

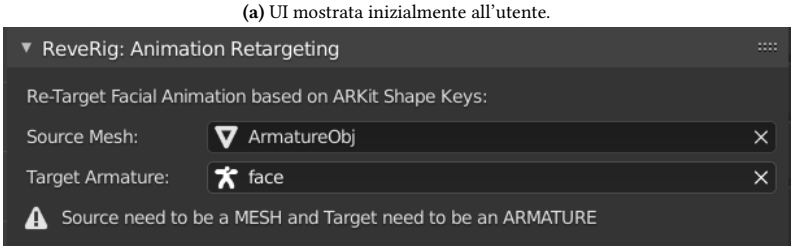
Il metodo `draw` racchiude tutto il codice utile per gestire correttamente l'interfaccia. In questo caso, qualsiasi sia la condizione di partenza, l'UI mostra sempre due barre di ricerca, dotate di strumento picker, relative alla registrazione dell'oggetto sorgente e dell'oggetto target per il trasferimento dei dati. Tali dati verranno memorizzati all'interno delle rispettive proprietà sopra definite. Sebbene tali strumenti di ricerca siano sempre accessibili dall'utente, il bottone che consente l'effettiva operazione di retargeting viene nascosto finché non saranno rispettati alcuni vincoli fondamentali per il corretto funzionamento. Tali vincoli impongono all'utente di indicare correttamente entrambi i campi relativi a sorgente e target. In particolare, i controlli effettuati verificano che l'oggetto definito come sorgente sia di tipo `MESH` mentre l'oggetto target sia di tipo `ARMATURE`. Questo perché, come abbiamo visto nel capitolo 9.1, i dati dell'animazione a noi utili sono contenuti all'interno della mesh *face* per Rokoko e *head* per FaceCap. Mentre come target è necessario impostare il rig facciale creato tramite il pannello *Facial Rigging* dell'add-on, ovvero l'oggetto `GUI_Char_SK`.

Grazie all'utilizzo di appositi messaggi di testo è possibile guidare l'utente verso la corretta selezione di sorgente e target. Nella figura 11.10 vengono mostrati i due possibili messaggi di avvertenza: in figura 11.10a nel caso non siano stati riempiti entrambi i campi delle barre di ricerca, in figura 11.10b nel caso in cui non siano superati i controlli relativi alla tipologia dell'oggetto.

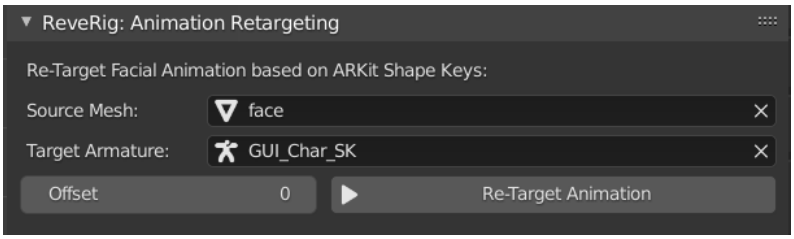
Nella figura 11.10c, invece, è possibile vedere l'UI competa relativa alla fase di retargeting dei dati acquisiti con i sistemi di mocap previsti. Come si può notare al posto del messaggio di avvertenza vengono mostrati due nuovi strumenti. Il primo, denominato *Offset*, serve per impostare il primo frame dal quale i dati copiati dalla sorgente dovranno essere inseriti. In particolare, offset è direttamente legato alla proprietà di Blender chiamata `frame_current` che indica il fotogramma corrente sulla quale è posizionato il cursore della *Timeline*. Quindi, agire con lo strumento offset o tramite cursore della timeline risulta identico. Tuttavia, lo strumento interno a ReveRig è stato appositamente concepito per agevolare e velocizzare tale operazione. Accanto allo strumento nominato offset viene presentato il bottone che consente di chiamare l'operatore `ANIM_OP_CopyFCurvesAnimation` che si occuperà di eseguire il codice per effettuare l'operazione di trasferimento dell'animazione.



(a) UI mostrata inizialmente all'utente.



(b) Indicazione scorretta di *source* e *target*.



(c) UI che consente di proseguire con il retargeting.

Figura 11.10: Tre stati mostrati dall'UI

Classe ANIM_OP_CopyFCurvesAnimation

Questo operatore si occupa di eseguire, tramite apposite funzioni, il codice che effettivamente consentirà di trasferire correttamente i dati da sorgente a target. L'operatore è così formato:

```
### OPERATOR COPY/RETARGET FACIAL ANIMATION ######
class ANIM_OP_CopyFCurvesAnimation(bpy.types.Operator):
  """Copy the FCurves Animation from Source to Target at current frame. It
    may take some time"""
  bl_idname = "animation.copyfcanim"
  bl_label = "Re-Target Animation"
  bl_options = {'REGISTER', 'UNDO'}

  def execute(self, context):
    start= time.time()
    print("Copying animation...")
    print("It may take some time")
    insertKeyFrames(context)
    copyFCurvesAnimation(context)
    print("It took this time to copy animation:",time.time()-start,"\n")
    return {'FINISHED'}
```

Come si può notare, il metodo `execute` si limita principalmente a chiamare le due funzioni: `insertKeyFrames` e `copyFCurvesAnimation`. Tuttavia, siccome l'operazione di trasferimento

dei dati potrebbe richiedere una quantità di tempo variabile (da alcuni secondi a qualche minuto), strettamente dipendente dal numero di keyframe che compongono l'animazione totale e dalle prestazioni dell'hardware su cui si lavora, si è deciso di fornire all'utente, tramite la console di sistema, un messaggio che indichi se l'operazione sia ancora in corso o se abbia terminato correttamente. Inoltre, al termine dell'operazione viene fornito un messaggio che indica il tempo impiegato per portare a termine il compito.

Funzione insertKeyFrames

L'operazione di retargeting viene effettuata copiando i keyframe di tutte le F-curve delle blendshape del modello 3D creato tramite gli strumenti di mocap, e incollandoli nelle F-curve delle ossa slider che compongono il rig facciale. Tuttavia, tali ossa non sono dotate di F-curve finché non viene inserito loro un keyframe per la proprietà corrispondente, tipicamente la traslazione sull'asse y. Le ossa che compongono il rig facciale di ReveRig sono quindi sprovviste delle F-curve necessarie. Diventa necessario inserire per ciascun osso un keyframe iniziale. Per soddisfare tale esigenza è stata creata la funzione `insertKeyFrames`; di seguito è possibile vederne la struttura.

```
### Method - Insert Keyframes for Retarget Animation
def insertKeyFrames(context):
  target_armature = bpy.context.scene.my_settings.copyFc_target
  FRAME_CURRENT = bpy.context.scene.frame_current
  list_x= ["Slider_eyeLookAllRight","Slider_eyeLookAllLeft","
    Slider_Eyes_eyeLookAllRight","Slider_Eyes_eyeLookAllLeft","Slider_jawAll"
    ]
  for bone in target_armature.pose.bones:
    if 'Slider_' in bone.name:  #only Slider_ bones
      if bone.name in list_x:
        bone.location[0] = 0 #set value to 0
        bone.keyframe_insert(data_path='location',index=0, frame=
    FRAME_CURRENT) #index=0 insert only X
        bone.location[1] = 0 #set value to 0
        bone.keyframe_insert(data_path='location',index=1, frame=
    FRAME_CURRENT) #index=1 insert only Y
```

Nella prima parte vengono inizializzate le variabili, come `target_armature` che fa riferimento al rig facciale `GUI_Char_SK`, `FRAME_CURRENT` che memorizza il frame corrente, definito precedentemente grazie allo strumento offset, e `list_x` che contiene il nome delle ossa slider del rig facciale che hanno possibilità di muoversi su due assi. Tramite un ciclo `for` che scorre tutte le ossa dell'armatura si va a lavorare unicamente su quelle relative agli slider. Dopo di che, per ogni osso individuato, si va a definirne la traslazione sugli assi impostando come valore di default zero. Infine, si va ad inserire un keyframe, nel frame corrente, relativo alla proprietà `location` rispetto all'asse individuato. Terminato il ciclo `for` si ottiene che, per ciascuna delle ossa individuate, è stata creata la F-curve per la proprietà di location, nella maggior parte dei casi solo per l'asse y, mentre per le ossa contenute in `list_x` anche per l'asse x. Diventa ora possibile procedere con il trasferimento dei dati grazie alla funzione `copyFCurvesAnimation`.

Funzione copyFCurvesAnimation

Tale funzione è quella che racchiude il codice che permette effettivamente di copiare l'animazione voluta ed è quella che richiede più tempo per essere portata a termine. L'organizzazione generale del codice può essere riassunta come segue:

```
### Method - Copy F-Curves from source to target for Retarget Animation
def copyFCurvesAnimation(context):
  target_armature = bpy.context.scene.my_settings.copyFc_target
  source_animation = bpy.context.scene.my_settings.copyFc_source
  FRAME_CURRENT = bpy.context.scene.frame_current #Useful for copying the
    curve where user want
  #...
  min_frame=100000 #arbitrarily high number
  for fcurve_sk in act_sk.fcurves:
    sk_name = fcurve_sk.data_path.split('"')[1]
    #Find the least value of the first frame to copy to the correct location
    if fcurve_sk.keyframe_points[0].co[0]<min_frame:
      min_frame = fcurve_sk.keyframe_points[0].co[0]
    #Store the f-curves of some shape keys to use them later
    #Ad esempio per la shape key jawLeft:
    if sk_name=='jawLeft':
      JawLeft= fcurve_sk.keyframe_points
    #...
  for fcurve_sk in act_sk.fcurves:
    sk_name = fcurve_sk.data_path.split('"')[1]
    for fc in act_obj.fcurves:
      #...
      bone_name = fc.data_path.split('"')[1].split('Slider_')[1]
      #...
      if sk_name==bone_name:
        for point in fcurve_sk.keyframe_points:
          fc.keyframe_points.insert(FRAME_CURRENT-min_frame+point.co[0],
    point.co[1])
```

Dopo una prima fase in cui si vanno ad inizializzare le diverse variabili, si effettua un ciclo `for` per scorrere i dati relativi a tutte le F-curve della mesh registrata precedentemente come sorgente. All'interno di questo primo ciclo `for` si va a cercare tra tutti i keyframe di tutte le F-curve il frame con valore minore e lo si memorizza nella variabile `min_frame`. Questo dato è fondamentale per consentire un corretto allineamento dei keyframe quando questi verranno copiati sul rig facciale. Inoltre, per alcune F-curve relative a determinate blendshape, si memorizza in apposite variabili il dizionario contenente i dati relativi ai keyframe di una determinata F-curve. Tale dizionario è indicato come `keyframe_points[n].co[m]`, dove con `n` si indica l'indice del keyframe all'interno del dizionario, mentre con `co[m]`, dove `m` assume valore 0 o 1, è possibile accedere ai dati specifici di un determinato keyframe. In particolare, `co[0]` fornisce il valore del frame dove è inserito il keyframe, e `co[1]` fornisce il valore della proprietà per quel keyframe.

Terminato il primo ciclo `for`, viene effettuato un secondo ciclo `for` sempre per scorrere tutte le F-curve della mesh sorgente. All'interno di questo ciclo si crea un ulteriore ciclo annidato che permette di scorrere tra le F-curve dell'oggetto target. Si vanno effettivamente a copiare tutti i keyframe delle varie F-curve che compongono l'animazione della mesh sorgente, sul rig facciale indicato come target. Il codice mostrato sopra riporta in modo semplificato quanto effettuato per la maggior parte degli slider del rig facciale. Tuttavia, come si è già potuto constatare, alcune

ossa hanno esigenze particolari e questo comporta una necessaria complicazione del codice che, per ragioni di leggibilità, non può essere riportato per interezza all'interno di questo documento. Le esigenze particolari riguardano principalmente quegli slider che hanno range di azione [-1,1] su uno o più assi. Questo comporta che al movimento dello slider su uno stesso asse siano legate due blendshape differenti.

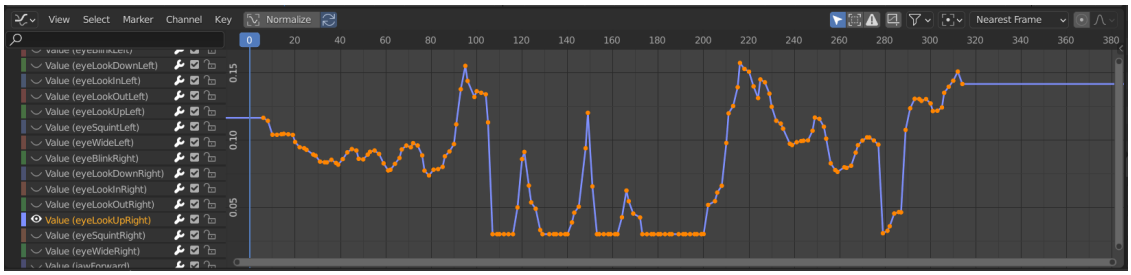
Prendiamo ad esempio le due blendshape `eyeLookDownRight` e `eyeLookUpRigh`. Queste, per la mesh sorgente, corrisponderanno a due F-curve separate, entrambe con keyframe aventi valore nel range [0,1]. Mentre, nel rig facciale creato con ReveRig corrispondono ad un'unica F-curve, ovvero la curva di interpolazione dell'osso nominato `eyeLookAllRight` relativa ai valori di traslazione rispetto l'asse y. È necessario trovare un sistema che consenta di unire i dati provenienti da due F-curve all'interno di un'unica F-curve senza perdere dati importanti. Si può notare che il range di azione dello slider sull'asse y è [-1,1], ovvero il doppio di quanto previsto per una singola blendshape. Quindi siamo sicuri che c'è spazio di "manovra" a sufficienza per i dati di entrambe le F-curve della mesh sorgente. Siccome metà dell'azione dello slider è in un range negativo, diventa necessario "ribaltare" i valori dei keyframe di una delle due blendshape e sottrarli all'altra. Nel nostro esempio si procede andando a sottrarre, per ogni frame previsto dalle F-curve, il valore del keyframe della F-curve corrispondente a `eyeLookDownRigh` al valore del keyframe della F-curve corrispondente alla blendshape `eyeLookUpRigh`. Di seguito si riporta la parte di codice che effettua tale operazione.

```
for idx in enumerate(EyeUp_R):
    fc.keyframe_points.insert(FRAME_CURRENT-min_frame+EyeUp_R[idx[0]].co[0],
     EyeUp_R[idx[0]].co[1]-EyeDown_R[idx[0]].co[1])
```

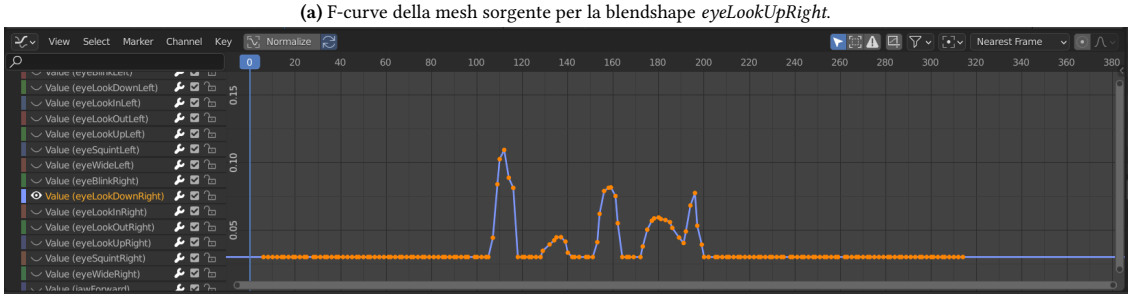
Dove `EyeUp_R` e `EyeDown_R` sono variabili che contengono l'intero dizionario dei keyframe delle rispettive F-curve, mentre `idx` è la variabile contatore che indica l'indice del keyframe a cui fare riferimento all'interno del dizionario `EyeUp_R`. Nella figura 11.11 vengono mostrate in 11.11a e 11.11b le curve di interpolazione delle due blendshape della mesh sorgente, mentre in 11.11c viene mostrata la F-curve creata per il rig facciale di ReveRig.

Al termine della funzione `copyFCurvesAnimation` tutti i dati relativi all'animazione delle espressioni facciali sono stati copiati dalla mesh sorgente e incollati sul rig facciale. Grazie allo strumento offset visto quando si parlava dell'interfaccia, è possibile inserire i dati relativi all'animazione facciale a partire dal frame che si desidera, evitando in questo modo di sovrapporre i keyframe appena copiati con dei keyframe già esistenti. Come accennato precedentemente, l'intera operazione di retargeting richiede in generale più tempo rispetto a tutte le altre operazioni previste da ReveRig. Il tempo necessario a portare a termine l'operazione dipende fortemente dal numero di keyframe che compongono l'animazione. Più la sequenza da copiare è lunga e più i keyframe sono vicini tra loro, più sarà il tempo che l'add-on impiegherà per portare a termine l'operazione. Molto dipende anche dalle prestazioni raggiungibili dall'hardware in utilizzo. In generale è bene considerare che tale operazione necessita da una decina di secondi fino a qualche minuto circa per terminare. Tuttavia, nei casi più estremi si potrebbe arrivare a dover attendere tre o quattro minuti. Considerando che l'intero processo è quasi del tutto automatizzato, ad eccezione della fase iniziale in cui l'utente deve indicare sorgente e target, il vantaggio complessivo nell'utilizzare ReveRig è molto elevato.

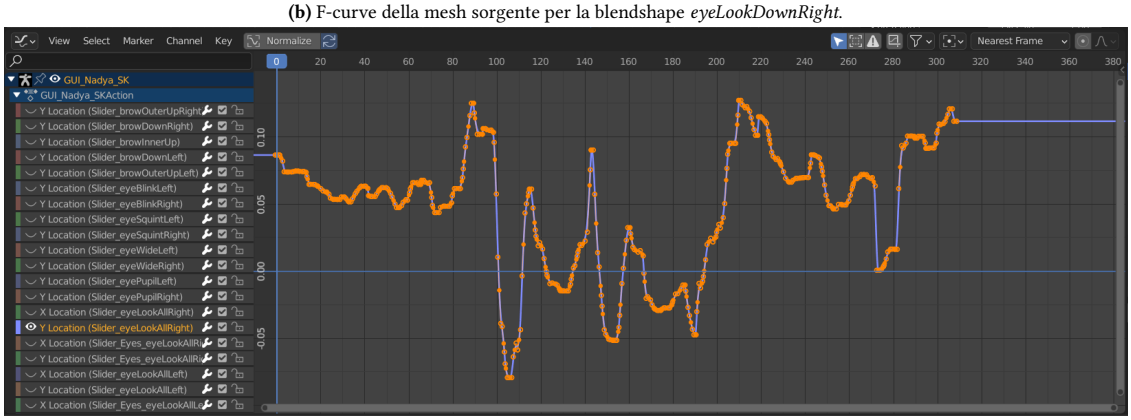
Dopo aver effettuato il retargeting per l'animazione catturata tramite i sistemi di motion capture facciale, è possibile procedere con il retargeting dell'animazione relativa alla lingua.



(a) F-curve della mesh sorgente per la blendshape *eyeLookUpRight*.



(b) F-curve della mesh sorgente per la blendshape *eyeLookDownRight*.



(c) F-curve creata da ReveRig. Valori positivi per *eyeLookUpRight*, valori negativi per *eyeLookDownRight*.

Figura 11.11: Confronto tra F-curve della mesh sorgente e quelle create da ReveRig

11.4.2 Fase 2 - Retargeting dell'animazione della lingua

Terminato il retargeting delle espressioni facciali, diventa possibile proseguire con la seconda fase prevista nel pannello *Animation Retargeting*. Le operazioni per il retargeting dell'animazione della lingua vengono proposte all'interno di un sotto-pannello apposito dal titolo *Import Tongue Animation*; tale pannello è imparentato alla classe `CA_PT_CopyFCurvesAnimation` che ne è il padre. Questo sotto-pannello si divide a sua volta in due step principali che, per ottenere il risultato desiderato, devono essere eseguiti in ordine. Il primo step riguarda l'importazione del file `.dat` generato con Papagayo-NG (vedi capitolo 10) e la creazione dei keyframe corrispondenti ai dati interni al file importato. Il secondo step consente di ridefinire i valori dei diversi keyframe appena creati, in modo che questi si adattino all'animazione relativa all'apertura della mascella.

Di seguito si va a descrive più approfonditamente ciascuno dei due step.

Step 1 - Importazione dati e creazione keyframe

Questo primo step racchiude tutte quelle operazioni che consentono di importare correttamente il file contenente i dati relativi all'animazione della lingua e di leggere tale file per creare, per ciascuna delle sei blendshape previste, i keyframe necessari. Ovviamente, esattamente come visto nella nella sezione 11.4.1, i dati non vengono trasferiti per animare direttamente le blendshape ma vengono applicati al rig facciale appositamente creato e dotato di tutto il necessario per consentire la corretta esecuzione di questa fase. Sarà tramite i vari slider del rig che sarà poi gestita in modo fine l'animazione completa di volto e lingua.

A comporre questo primo step sono i seguenti elementi:

- La classe di tipo pannello `CA_PT_ImportTongueAnimation`, comune anche al secondo step, che racchiude tutto il codice necessario alla creazione e al corretto funzionamento dell'UI.
- L'operatore `ANIM_OP_ImportPlotLipsyncer` che ha il compito di eseguire il codice per la corretta importazione dei dati e della creazione dei keyframe corrispondenti.
- La funzione `lipsyncer` che si occupa di aprire e leggere i dati contenuti nel file importato.
- La funzione `createTongueAnimation` che si occupa effettivamente di inserire in modo corretto i keyframe per l'animazione della lingua.

Il codice relativo a questo primo step è stato sviluppato basandosi principalmente su quanto realizzato per l'add-on *LipSync Importer & Blinker*[2] che, come già introdotto nel capitolo 10, è stato appositamente creato per importare in modo corretto i dati provenienti da Papagayo e non solo. Data la necessità di unire tutti gli strumenti utili per l'animazione facciale all'interno di un unico sistema, si è deciso di non adottare l'add-on *LipSync Importer & Blinker*, ma di inserire alcuni suoi componenti all'interno di *ReveRig*, in modo tale da avere tutto il necessario all'interno di un'unica interfaccia, semplificando così il lavoro dell'animatore. Da *LipSync Importer & Blinker*, siccome racchiude diverse possibili operazioni e strumenti, sono state prese solo le componenti utili per il nostro scopo e dopo di che sono state modificate e adattate per integrarsi completamente con il restante codice creato per ReveRig. Sebbene tale operazione sia ampiamente consentita dalla licenza *GPL* con la quale l'add-on è distribuito, per garantire massima trasparenza e rispetto verso il lavoro altrui, all'interno del codice di ReveRig viene indicato ogni elemento che è stato realizzato a partire dall'add-on *LipSync Importer & Blinker*.

Classe CA_PT_ImportTongueAnimation È la classe di tipo pannello che racchiude tutto il codice utile per creare e gestire l'UI relativa all'intera fase di retargeting dell'animazione della lingua. Di seguito andremo a descrivere come la classe è definita e quanto è utile per questo primo step. La restante parte verrà descritta successivamente. La struttura generale del pannello può essere così riassunta:

```
1 class CA_PT_ImportTongueAnimation(bpy.types.Panel):
2   bl_space_type = 'VIEW_3D'
3   bl_region_type = 'UI'
4   bl_category = "REVERIG"
5   bl_options = {"DEFAULT_CLOSED"}
```

[2] Il codice sorgente dell'add-on *LipSync Importer & Blinker* può essere scaricato al seguente indirizzo: https://developer.blender.org/T24080.

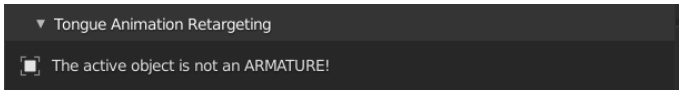
```
  bl_label = "Tongue Animation Retargeting"
  bl_parent_id= "COPY_FC_ANIMATION_PT_layout"

  newType= bpy.types.Scene
  newType.fpath = bpy.props.StringProperty(name="Import File ", description=
    "Select your voice file (.dat)", subtype="FILE_PATH")
  newType.skscale = bpy.props.FloatProperty(description="Smoothing shape key
     values", min=0.1, max=1.0, default=1)

  newType.easeIn = bpy.props.IntProperty(description="Smoothing In curve",
    min=1, default=4)
  newType.easeOut = bpy.props.IntProperty(description="Smoothing Out curve",
     min=1, default=4)
  newType.holdGap = bpy.props.IntProperty(description="Holding for slow keys
    ", min=0, default=0)

  def draw(self, context):
    #inizializzazione variabili...
    #Check if selected object is an ARMATURE
    if obj != None:
      if obj.type == "ARMATURE":
        split.label(text="The active object is: ", icon="OBJECT_DATA")
        split.label(text=obj.name, icon="OUTLINER_OB_ARMATURE")
        col.prop(context.scene, "fpath")
        #Riga sottostante ripetuta sostituendo property_name con: skscale,
    easeIn, easeOut e holdGap.
        split.prop(context.scene, "property_name")
        if scn.fpath!='':
          col1.operator('import.tonguels', text='Plot Keys to the Timeline')
     #Button Plot tongue animation
        else: #Messaggio di errore.
```

La prima parte consiste nella consueta definizione di alcune informazioni fondamentali per la corretta caratterizzazione della classe. A differire da quanto visto fino ad ora, si ha una sostituzione del campo `bl_idname` con `bl_parent_id`. Questo perché, come già anticipato, l'interfaccia per la creazione dell'animazione della lingua è inserita all'interno di un sotto-pannello. Con `bl_parent_id` è possibile indicare l'ID di quello che sarà il pannello padre, nel nostro caso `COPY_FC_ANIMATION_PT_layout`. Dopo di che, prima del metodo `draw`, si vanno a creare delle nuove proprietà per la scena, che torneranno molto utili per portare a termine correttamente l'operazione di retargeting, fornendo degli strumenti utili da poter inserire nell'UI. Tali proprietà sono:

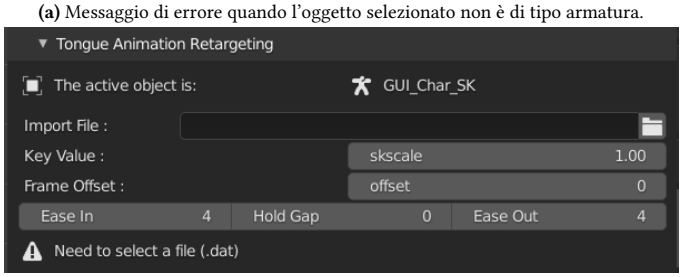
- La stringa `fpath`, di sottotipo `FILE_PATH`, che viene utilizzata per registrare il percorso relativo al file `.dat` che deve essere importato.
- Il floating point `skscale`, utilizzato per definire il valore che dovrà essere assegnato a ciascun keyframe; impostato di default a 1.
- Tre interi `easeIn`, `easeOut` e `holdGap` che servono ad indicare il range di influenza che i vari keyframe dovranno avere e per definirne la curva di accelerazione e decelerazione. Di default si ha `easeIn` e `easeOut` con valore 4 frame e `holdGap` con valore 0 frame.

Per guidare l'utente durante la fase di retargeting, è stata sviluppata un'interfaccia dinamica che si adatta al contesto e che permette di effettuare dei controlli preventivi prima di consentire

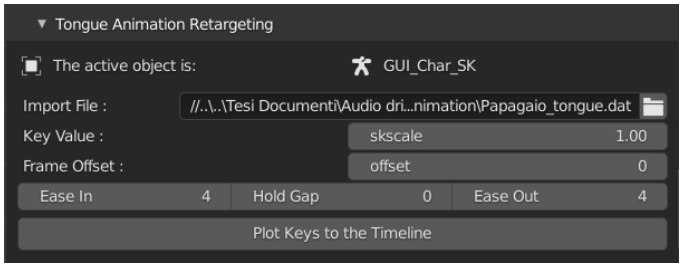
all'utente di compiere determinate azioni. In particolare, fintanto che l'utente non seleziona un oggetto di tipo armatura, viene mostrato a schermo unicamente un messaggio di avvertenza che impedisce all'utente di proseguire con l'operazione (vedi figura 11.12a). Tale messaggio porta automaticamente l'animatore a selezionare il rig facciale `GUI_Char_SK` creato precedentemente. Se la prima condizione è rispettata, allora l'UI mostrata sarà come quella in figura 11.12b.



(a) Messaggio di errore quando l'oggetto selezionato non è di tipo armatura.



(b) UI per importare l'animazione della lingua.



(c) UI completa che consente di proseguire con il retargeting.

Figura 11.12: Tre stati principali dell'UI relativa alla seconda fase del pannello *Animation Retargeting*

All'utente vengono mostrate alcune informazioni e alcuni strumenti utili per poter eseguire nel modo corretto l'operazione di retargeting dell'animazione della lingua. Prima di tutto viene data un'indicazione testuale e grafica sull'oggetto attivo in quel momento, che identifica anche l'oggetto target per l'operazione di trasferimento dati. In questo modo l'utente potrà avere sempre sotto controllo l'oggetto per la quale l'operazione verrà svolta. Dopo di che l'UI riporta sei strumenti, cinque dei quali sono relativi alle proprietà precedentemente descritte. In aggiunta a queste viene riproposto lo strumento, anche in questo caso nominato offset, che consente di manipolare il valore del frame corrente. Tutti questi campi sono più o meno liberamente gestibili dall'animatore, in modo da poter garantire ampio margine d'azione e consentire all'utente di identificare i valori più consoni per una determinata situazione. L'utilizzo effettivo di tali dati verrà approfondito successivamente. Come riportato in figura 11.12b, quando il campo *Import File* è vuoto, viene mostrato un messaggio di avvertenza che invita l'utente a selezionare un file di tipo `.dat`. La correttezza del file inserito spetta all'utente stesso in quanto ReveRig non è in grado di effettuare alcun tipo di controllo in modo preventivo. Una volta che è stato inserito in modo corretto il percorso del file che si vuole utilizzare, viene mostrato all'utente, al posto del messaggio di avvertenza, il bottone relativo alla chiamata all'operatore che deve eseguire il codice per importare e leggere i dati del file e creare i corrispettivi keyframe (si veda la figura 11.12c).

Classe ANIM_OP_ImportPlotLipsyncer L'operatore che ha il compito di eseguire il codice che importa e legge il file dat indicato dall'utente, e che utilizza i dati estratti dal file per creare i corretti keyframe è così composto:

```
### OPERATOR LIPSYNC IMPORTER AND PLOTTER #########
class ANIM_OP_ImportPlotLipsyncer(bpy.types.Operator):
```

```
  bl_idname = 'import.tonguels'
  bl_label = 'Plot to the Timeline the imported data'
  bl_description = 'Plots the imported voice file keys to timeline of the
    specific Slider Driver'
  bl_options = {'REGISTER', 'UNDO'}

  def execute(self, context):
    global lastPhoneme
    lastPhoneme="nothing" #initialize
    lipsyncer(context)
    return {'FINISHED'}
```

All'interno del metodo `execute`, prima di effettuare la chiamata alla funzione `lipsyncer`, viene inizializzata la variabile `lastPhoneme` che sarà fondamentale per gestire nel modo corretto la creazione dei vari keyframe.

Funzione lipsyncer Questa funzione ha il compito estrarre ed elaborare i dati contenuti all'interno del file indicato precedentemente e memorizzato grazie all'ausilio della proprietà `fpath`. Per dare al lettore un miglior riferimento a quanto si è andati a fare, un esempio di file dat generato tramite il lavoro svolto sul programma Papagayo-NG può essere visionato nell'appendice B di questo documento. All'interno della funzione `lipsyncer` si va, prima ad aprire il file e a ignorarne la prima riga riportante informazioni a noi non utili. Dopo di che, tramite un ciclo `for`, si leggono tutte le righe che compongono il documento. Da ogni riga si estrae il valore del frame e il nome del fonema che, grazie alla convenzione utilizzata fino ad ora, corrisponde al nome della blendshape che dovrà essere gestita dal rig facciale; in particolare, l'osso di riferimento sarà nominato `Slider_blendshapeName`. All'interno del primo ciclo che scorre le diverse righe del file, viene creato un ulteriore ciclo che permette di scorrere tutte le ossa slider del rig facciale per identificare quelle corrispondenti al fonema corrente. Individuato l'osso, viene effettuata la chiamata alla funzione `createTongueAnimation(bone.name, frame)`, dove `bone.name` contiene il nome dell'osso in questione e `frame` contiene il valore del fotogramma per il quale dovrà essere inserito il keyframe.

```
def lipsyncer(context):
  #inizializzazione...
  f=open(bpy.path.abspath(scn.fpath)) # importing file
  f.readline() # reading the 1st line that we don't need ('MohoSwitch1')
  for line in f:
    # removing new lines
    lsta = re.split("\n+", line)
    # building a list of frames & shapes indexes
    lst = re.split(":? ", lsta[0])# making a list of a frame & number
    frame = int(lst[0])

    for bone in obj.pose.bones:
      if 'Slider_' in bone.name:
        bone_name= bone.name.split('Slider_')[1]
        if lst[1] == bone_name:
          createTongueAnimation(bone.name, frame)
```

Funzione createTongueAnimation(phoneme, frame) All'interno di questa funzione si vanno a creare i vari keyframe necessari per l'animazione della lingua. In questa fase si utilizzano tutte quelle informazioni inserite precedentemente dall'utente, grazie all'ausilio delle proprietà appositamente create. Questa funzione viene chiamata per ogni riga letta dal file importato, e durante la chiamata vengono passate le informazioni rispetto a quale osso debba essere inserita l'animazione, ovvero il fonema previsto, e per quale frame. Per ogni fonema pronunciato ad un determinato frame, possono essere creati teoricamente fino ad un massimo di quattro keyframe in base a quanto indicato dall'utente.

Per tutti fonemi presenti all'interno del file, si andranno ad utilizzare gli stessi valori delle proprietà inserite precedentemente dall'utente, ovvero:

- La variabile `offst` è utilizzata per indicare una potenziale traslazione di tutti i keyframe generati rispetto a quanto indicato nel file di origine. Corrisponde al valore del frame corrente ed è fondamentale per inserire l'animazione dove è necessario.
- La variabile `skVlu` contiene il valore indicato nella proprietà `skscale`, e viene utilizzata per indicare il valore che dovrà essere attribuito ad ogni keyframe relativo al fonema.
- La variabile `frmIn` corrisponde al valore della proprietà `esaeIn`, e serve ad indicare in quale frame dovrà essere inserito il keyframe di valore 0 per creare una fase di accelerazione.
- La variabile `frmOut`, avente il valore indicato in `easeOut`, ha la stessa funzione di quella precedente ma per andare a creare la fase di decelerazione.
- La variabile `hldIn`, corrispondente a `holdGap`, viene utilizzata per indicare per quanti frame debba essere mantenuto il valore `skVlu` per il fonema corrispondente.

Da questo ne deriva che l'inserimento corretto dei keyframe viene così effettuato:

```
# inserting the In key only when phonem change
if lastPhoneme!=phoneme:
  obj.pose.bones[phoneme].location[1]=0.0
  obj.pose.bones[phoneme].keyframe_insert(data_path="location",index=1,
    frame=offst+frame-frmIn)

obj.pose.bones[phoneme].location[1]=skVlu
obj.pose.bones[phoneme].keyframe_insert(data_path="location",index=1, frame=
    offst+frame)

obj.pose.bones[phoneme].location[1]=skVlu
obj.pose.bones[phoneme].keyframe_insert(data_path="location",index=1, frame=
    offst+frame+hldIn)

obj.pose.bones[phoneme].location[1]=0.0
obj.pose.bones[phoneme].keyframe_insert(data_path="location",index=1, frame=
    offst+frame+hldIn+frmOut)

lastPhoneme = phoneme
```

Come si può notare, il keyframe relativo alla fase di attacco verrà creato solamente se il fonema corrente è diverso da quello che lo ha preceduto. In questo modo si evita di creare un'animazione a "singhiozzi". Tale operazione viene effettuata per ogni riga letta dal file contenete i dati dell'animazione della lingua.

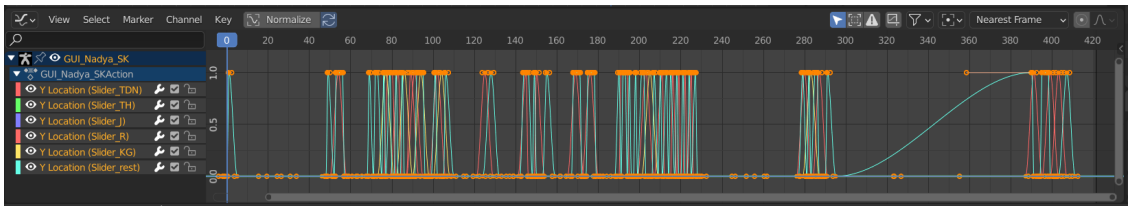


Figura 11.13: Esempio di F-curve ottenute al termine del primo step

Terminata l'operazione di creazione di tutti i keyframe necessari per realizzare l'animazione della lingua, ciò che si ottiene è simile a quanto raffigurato in figura 11.13. Come è possibile vedere, per ogni fonema presente nell'animazione, si crea un keyframe con valore costante pari a quanto indicato dall'utente nel campo `skscale`, di default pari a 1. Questo è un grosso limite in quanto si va a creare un'animazione meccanica e di scarsa naturalezza. È necessario gestire i diversi keyframe in modo che questi seguano un'animazione più realistica e maggiormente integrata al resto dell'animazione facciale in particolare con l'apertura della bocca. Come è già stato anticipato, la soluzione si è trovata legando l'intensità di ogni movimento della lingua con la curva di interpolazione relativa alla blendshape `jawOpen`. Da questa necessità nasce il secondo step previsto da ReveRig, che consente di mappare in modo corretto i keyframe dell'animazione della lingua.

Step 2 - Ridefinire l'animazione della lingua

Se nel primo step si è andati a creare, in base ai dati contenuti nel file importato, i vari keyframe che compongono le diverse F-curve per i diversi fonemi, in realtà ci si è praticamente limitati ad indicarne solo la corretta posizione. In questo secondo step si va, quindi, a ridefinire per ciascun keyframe, il valore che deve assumere per integrarsi al meglio con l'animazione generale del personaggio. Tale step viene presentato per ultimo in quanto deve essere effettuato dopo che è già stato effettuato il retargeting sia delle espressioni facciali che dei movimenti della lingua, in quanto tale processo si basa sui dati precedentemente creati. Rimane, quindi, l'ultimo step da eseguire all'interno del pannello *Animation Retargeting* di ReveRig. Tale step è composto delle seguenti componenti:

- Il pannello `CA_PT_ImportTongueAnimation`, già introdotto a pagina 139, che racchiude il codice per la gestione corretta dell'UI.
- L'operatore `ANIM_OP_RemapTongue`, che si occupa di eseguire il codice necessario per questa operazione.
- La funzione `remapTongueValue` che racchiude il codice che effettivamente permette di portare a termine l'operazione.

Classe CA_PT_ImportTongueAnimation Per maggiori informazioni relative su come questa classe è stata gestita, il lettore può fare riferimento a quanto riportato a pagina 139. Di seguito si va a descrivere solo la parte di codice appositamente creata per questa fase.

```
#ReMap Tongue Value
row = layout.row()
layout.label(text="REMAP TONGUE VALUE:")

```

```
layout.label(text="Set Frame Range:")
row = layout.row()
row.prop(scene, "frame_start") #set start frames range
row.prop(scene, "frame_end") #set end frames range

#RemapOperator button
row = layout.row()
row.scale_y = 1.2
row.operator("animation.remaptongue")
layout.label(text="Remap only if you have already plotted Tongue value and
    if jaw animation is present:", icon="ERROR")
```

L'interfaccia creata risulta piuttosto semplice, oltre a un titolo iniziale, utile per creare separazione rispetto allo step precedente, vengono presentati due strumenti: *Start Frame* ed *End Frame*. Tali campi vengono utilizzati per consentire all'utente di definire il range di fotogrammi su cui andrà effettuata l'operazione. In questo modo è possibile agire solo su una parte dell'animazione lasciando inalterate quelle circostanti. Tale meccanismo è pensato principalmente per quei casi in cui l'animazione della lingua va a coprire solo parte del range definito dall'animazione facciale completa. Questi input fanno riferimento diretto rispettivamente alle proprietà di Blender `frame_start` e `frame_end`, accessibili anche dalla finestra della timeline e consentono di avere un riferimento grafico all'interno dello spazio di lavoro. Dopo di che viene presentato il bottone che consente di chiamare l'operatore `ANIM_OP_RemapTongue` che va ad eseguire il codice. Infine, per dare all'utente maggiori informazioni, è stato inserito un messaggio di avvertimento che indica all'animatore la necessità di eseguire questo step solo dopo aver prima eseguito i precedenti. Nella figura 11.14 viene mostrata l'UI relativa a questo secondo step.

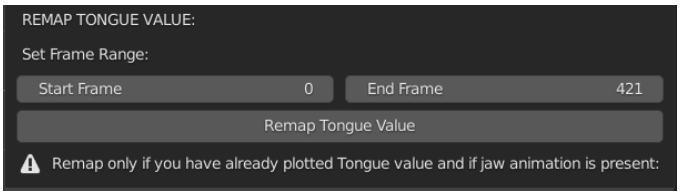


Figura 11.14: UI del secondo step della seconda fase del pannello *Animation Retargeting*

Classe ANIM_OP_RemapTongue L'operatore che viene chiamato dall'utente nel momento in cui viene premuto il bottone corrispondente. È così definito:

```
### OPERATOR CREATE THE REMAP
class ANIM_OP_RemapTongue(bpy.types.Operator):
  """Remap Shape Keys's values  of the Tongue based on the values  of jawOpen
    """
  bl_idname = "animation.remaptongue"
  bl_label = "Remap Tongue Value"
  bl_options = {'REGISTER', 'UNDO'}

  def execute(self, context):
    remapTongueValue(context)
    return {'FINISHED'}
```

Principalmente si limita ad effettuare la chiamata alla funzione `remapTongueValue` all'interno del metodo `execute`.

Funzione remapTongueValue Dopo una necessaria prima parte dove si vanno ad inizializzare le diverse variabili utili, il codice che compone questa funzione può essere suddiviso in due fasi.

Nella prima fase si effettua un ciclo che scorre tutte le F-curve di tutte le ossa del rig facciale, per individuare quella relativa all'osso che consente di controllare la blendshape `jawOpen`, ovvero la F-curve dell'osso `Slider_jawAll` per la traslazione sull'asse x. Dopo di che si vanno a valutare i valori della F-curve per tutti i frame compresi nel range definito dall'utente. Per ciascun frame viene analizzato il valore corrente e questo valore viene mappato secondo una curva discreta e non lineare definita come segue:

```
val = fc.evaluate(fr) #local variable to write values  for each frame
#check on limits
if fc.evaluate(fr)>max_range:
  val=max_range
elif fc.evaluate(fr)<min_range:
  val=min_range
#reparametrization of values  according to a discrete non-linear curve
elif fc.evaluate(fr)>=min_range and fc.evaluate(fr)<0.1:
  val=0.2
elif fc.evaluate(fr)>=0.1 and fc.evaluate(fr)<0.2:
  val = 0.3
elif fc.evaluate(fr)>= 0.2 and fc.evaluate(fr)<0.3:
  val = 0.4
elif fc.evaluate(fr)>=0.3 and fc.evaluate(fr)<0.4:
  val= 0.5
elif fc.evaluate(fr)>=0.4 and fc.evaluate(fr)<0.5:
  val = 0.55
elif fc.evaluate(fr)>= 0.5 and fc.evaluate(fr)<0.6:
  val= 0.6
elif fc.evaluate(fr)>=0.6 and fc.evaluate(fr)<0.7:
  val=0.65
#from 0.7 to max_range use linear parametrization
sample_dic[fr] = val
```

I valori ottenuti dopo la conversione vengono salvati all'interno di un dizionario nominato `sample_dic`, dove si ha una corrispondenza frame-valore. Tale dizionario altro non è che una copia manipolata dei dati della F-curve indicata sopra.

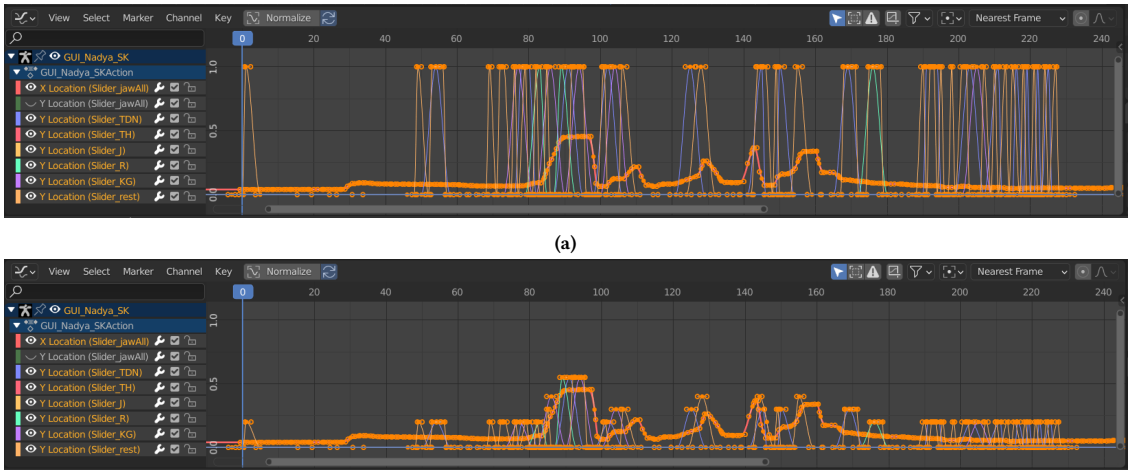
I dati utilizzati per effettuare la conversione sono stati definiti tramite conoscenze empiriche acquisite grazie a un lungo processo di *trial&error*. Pertanto, seppur i valori per la conversione siano discreti e non lineari, risultano adatti al compito che devono svolgere, considerando anche che l'animazione della lingua risulta secondaria rispetto a quella delle espressioni facciali e quindi necessita di minor precisione. Dopo aver creato il dizionario contenente i valori che definiscono la nuova curva di interpolazione relativa all'animazione dell'apertura della mascella, è possibile proseguire con la seconda fase.

Nella seconda fase si ridefiniscono i valori dei keyframe basandosi sui valori corrispondenti contenuti nel `sample_dic` appena creato. In pratica, vengono individuate le F-curve relative all'animazione delle sei blendshape della lingua (`tongueOut` è esclusa), e per ciascuna di esse, per ciascun keyframe, si va ad aggiornare il valore corrente, facendo il prodotto tra valore corrente

del keyframe (di default pari a 1) e il valore presente nel `sample_dic` relativo allo stesso frame. Dopo di che si va ad inserire il keyframe avente nuovo valore. Di seguito viene mostrato il codice che permette si svolgere questo compito.

```
#list of the path for the Tongue ShapeKey fcurves
list_fcpath = ['pose.bones["Slider_TDN"].location', 'pose.bones["Slider_TH
    "].location','pose.bones["Slider_J"].location', 'pose.bones["Slider_R"].
    location','pose.bones["Slider_KG"].location', 'pose.bones["Slider_rest"].
    location']
#scroll through all the fcurves of the selected mesh
for fc in act_obj.fcurves:
  percorso = fc.data_path
  if fc.data_path in list_fcpath and fc.array_index==1:
    keyframePoints = fc.keyframe_points #User-editable keyframes
    sk_K = percorso.split('"')[1] #Key/name of the shapeKey Bone "Slider_SK"
    sk_K = str(sk_K)
    for keyframe in keyframePoints:
      sk_V = keyframe.co[1] #shapeKey Value at keyframe.co[0]
      sk_JO = sample_dic[keyframe.co[0]] #JawOpen sk Value at keyframe.co[0]
      #set the new value for sk_K as the product of sk_K and sk_JO
      obj.pose.bones[sk_K].location[1]= sk_V*sk_JO
      #Update Keyframe
      obj.pose.bones[sk_K].keyframe_insert(data_path='location',index=1,
    frame=keyframe.co[0])
```

Al termine dell'operazione ciò che si ottiene sono delle F-curve, relative all'animazione della lingua, che seguono l'andamento della F-curve per l'animazione della blendshape `jawOpen`. Nella figura 11.15 è possibile vedere un esempio di ciò che si ha prima (11.15a) e dopo (11.15b) l'operazione di ridefinizione dei valori.



(a)

(b)

Figura 11.15: Confronto tra prima e dopo l'operazione di rimappatura

Con questa operazione termina l'intero processo di retargeting dell'animazione. Effettuando tutti i passaggi indicati nella sezione 11.4 si andrà a trasferire l'intera animazione facciale sul rig creato da ReveRig. Idealmente il processo di animazione facciale terminerebbe qui. Nella realtà,

però, è quasi sempre necessario del lavoro da parte dell'animatore per rendere l'animazione appena trasferita in un prodotto finito. Infatti, difficilmente i dati generati tramite mocap e tramite Papagayo-NG saranno in grado di fornire immediatamente l'animazione desiderata. È compito dell'animatore manipolare nel modo corretto i dati catturati per ricreare esattamente le espressioni facciali volute dal regista. Tale processo di rifinitura può essere molto lungo e faticoso, per questo motivo si è deciso di fornire, tramite ReveRig, tutti gli strumenti necessari per agevolare il lavoro. Oltre a creare il rig facciale descritto precedentemente, l'add-on fornisce all'utente un'apposita UI contenente gli strumenti principali per gestire in modo fine, semplice e intuitivo l'animazione facciale.

Il lavoro dell'animatore si sposta dal secondo pannello dell'interfaccia al terzo ed ultimo pannello di ReveRig, chiamato *Drivers Manager*.

11.5 Pannello 3 - Drivers Manager

La terza e ultima grande sezione di ReveRig è stata pensata e sviluppata per offrire all'utente tutti quegli strumenti utili per gestire in modo semplice e intuitivo il controllo sull'animazione facciale. Con la creazione del rig facciale si è andati a fornire all'animatore un'interfaccia di tipo *viewport 2D*, questa tipologia di UI, nel campo dell'animazione facciale, è molto importante in quanto consente all'animatore di poter lavorare direttamente nello spazio di lavoro nel quale si trova il modello del personaggio. Inoltre, grazie alla struttura e all'organizzazione che si è deciso di dare al rig, l'animatore ha a disposizione uno strumento con una relazione praticamente diretta tra componenti manipolabili del rig facciale, ovvero gli slider, e deformazione apportata al modello 3D. Tuttavia, sebbene l'utilizzo del rig sia più immediato, per mantenere un'UI viewprot 2D più pulita ed ordinata possibile si è deciso di non andare ad inserire il controllo completo di tutte le proprietà utili, ma solo di quella proprietà che corrisponde alla variabile `var_slider` utilizzata nella creazione dei driver. Ciò significa che si deve fornire all'utente, tramite un'ulteriore interfaccia, la possibilità di agire anche sulle altre proprietà, in particolare su `amp_value` (nominata `var_amp` nella creazione dei driver - vedi pagina 125).

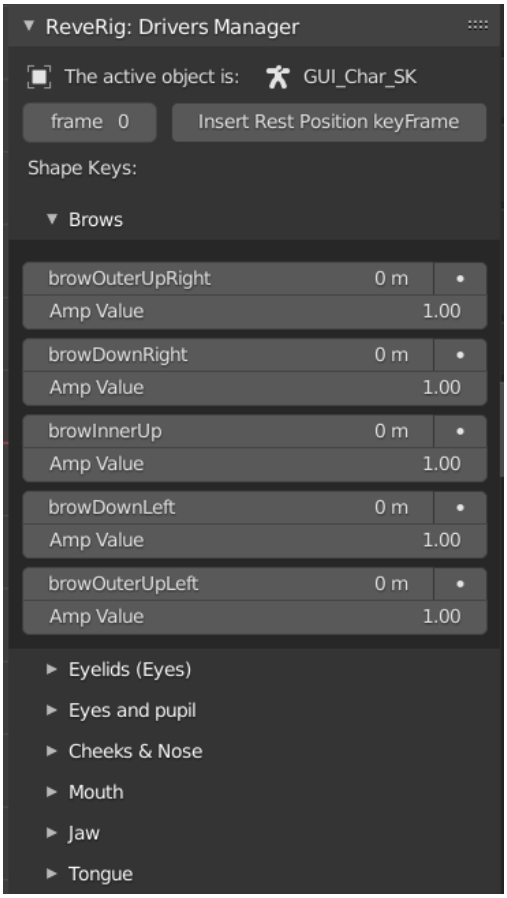


Figura 11.16: UI pannello Drivers Manager

Da queste esigenze nasce la necessità di sviluppare e creare la terza parte di ReveRig che, secondo quanto definito nel capitolo 4, consiste in una UI di tipo *window-based*. In generale questa terza parte

altro non è che un pannello di interfaccia, contenente strumenti utili per controllare in modo fine i parametri che consentono di realizzare l'animazione facciale. Tali strumenti sono proposti sotto forma di slider e di bottoni. In particolare, si avranno due slider, uno che consente di controllare lo stesso parametro controllabile tramite rig, solitamente la traslazione di un osso di tipo slider lungo l'asse y nel range di azione [0,1]. Il secondo strumento di tipo slider serve per consentire all'animatore di manipolare la proprietà `amp_value` relativa a ciascuna blendshape. I bottoni, invece, vengono proposti per consentire all'animatore di inserire in modo veloce ed intuitivo i keyframe necessari per ciascuna blendshape. Come si può vedere nella figura 11.16, per mantenere maggior ordine all'interno dell'interfaccia, si è deciso di suddividere in diversi sotto pannelli i comandi relativi a differenti aree del volto.

Sempre nella figura 11.16 si può notare che l'interfaccia è divisa in due macro-sezioni. La prima, che occupa la parte alta dell'UI, presenta strumenti che consentono di apportare modifiche a tutte le blendshape contemporaneamente. La seconda, invece, racchiude i controlli per una gestione fine e precisa di ogni singola blendshape in modo indipendente.

Di seguito si va a descrivere come entrambe queste sezioni siano state organizzate e create.

11.5.1 Sezione 1 - Rest Position

La prima sezione è stata pensata per consentire all'animatore di controllare tramite pochi strumenti, il comportamento di tutte le sessantadue blendshape previste da ReveRig. In particolare, è stata pensata per ricreare la *Rest Position* per l'armatura del rig facciale. Impostare la posizione di riposo per tutte le ossa del rig significa attribuire per ciascuna blendshape un valore pari a zero, tale comando consente di reimpostare facilmente l'espressione facciale di base. È uno strumento molto potente in quanto risulta molto frequente la necessità di dover tornare ai valori di partenza.

Senza tale strumento, ogni volta che dovesse essere necessario reimpostare l'espressione base, l'animatore si troverebbe costretto a dover agire su tutti gli slider legati alle sessantadue blendshape. Come è possibile immaginare, questo richiederebbe un lavoro lungo e ripetitivo. Si è deciso pertanto di realizzare uno strumento che consentisse di automatizzare completamente il processo.

A comporre questa prima sezione del pannello *Driver Manager* sono i seguenti elementi:

- Le due classi di tipo pannello `DMANAGER_PT_ui` e `DMANAGER_PT_Parent` che racchiudono il codice utile a realizzare l'interfaccia desiderata. Come vedremo queste classi vengono utilizzate per sviluppare anche la seconda sezione.
- L'operatore `ANIM_OP_InsertRestPoseKeyFrame` che ha il compito di eseguire il codice per portare a termine il compito individuato.
- La funzione `insertKeyFrames_ActiveObj` che racchiude il codice che effettivamente permette di effettuare l'operazione.

Classe DMANAGER_PT_ui e DMANAGER_PT_Parent

Queste due classi di tipo pannello sono utilizzate per gestire nel modo desiderato l'interfaccia utente. Siccome si è deciso di organizzare il pannello *Driver Manager* tramite l'utilizzo di diversi sotto pannelli, è risultato necessario strutturare il codice in modo da renderlo facilmente gestibile, sia in fase di sviluppo sia nelle future fasi di manutenzione. Per questo motivo si è deciso di creare la classe `DMANAGER_PT_ui` contenente le informazioni principali e di base utili a definire

le caratteristiche del pannello, dopo di che le classi relative al pannello padre e ai sotto pannelli andranno ad estendere questa classe. In questo modo, eventuali modifiche successive, potranno essere gestite unicamente all'interno di questa classe senza dover modificare decine di righe sparse in tutto il codice. La classe `DMANAGER_PT_ui` è così composta:

```
### Global definition
class DMANAGER_PT_ui(bpy.types.Panel):
  bl_space_type = 'VIEW_3D'
  bl_region_type = 'UI'
  bl_category = "REVERIG"
  bl_options = {"DEFAULT_CLOSED"}
```

La classe `DMANAGER_PT_Parent`, da come si può evincere dal nome, è la classe pannello che ha funzione di genitore all'interno del sistema gerarchico che si va a creare. È proprio all'interno di questa classe che si è andati a gestire l'UI relativa alla creazione dei keyframe per la *Rest Position*. La struttura interna della classe può essere riassunta come segue:

```
### Layout Panel Parent - Insert Rest Position Keyframes
class DMANAGER_PT_Parent(DMANAGER_PT_ui):
  bl_label = "ReveRig: Drivers Manager"
  bl_idname = "MANEGER_PARENT_PT_layout"

  def draw(self, context):
    #inizializzazione...
    #If selected an armature I can set a key frame for Rest Position,
    #all the sliders return to the value 0
    if obj_active:
      if obj_active.type =='ARMATURE':
        split.label(text="The active object is: ", icon="OBJECT_DATA")
        split.label(text=obj_active.name, icon="OUTLINER_OB_ARMATURE")
        #...
        col.prop(scene, "frame_current", text ='frame')
        #...
        col.operator("animation.insertrpkf")
      else:
        col = layout.column()
        col.label(text="The active object is not an ARMATURE!", icon="
    OBJECT_DATA")
    else:
      layout.label(text="No object is selected", icon="OBJECT_DATA")
    layout.label(text="Shape Keys:")
```

Come si può notare, in questo caso invece di ereditare dalla classe `bpy.types.Panel` si eredita dalla classe precedentemente creata `DMANAGER_PT_ui`. La struttura interna è piuttosto semplice, nella prima parte si vanno a definire quelle informazioni (`bl_idname` e `bl_label`) che non si è potuti inserire all'interno della classe `DMANAGER_PT_ui` in quanto non informazioni globali ma specifiche del pannello padre. Dopo di che, nel metodo `draw` si va a realizzare l'interfaccia relativa unicamente a questa prima sezione. Dopo una serie di controlli che permettono di verificare se l'oggetto selezionato sia di tipo armatura, in modo da guidare l'utente a selezionare il rig facciale creato precedentemente, se questi vengono superati, l'interfaccia mostrerà all'utente due strumenti (vedi figura 11.17). Il primo, collegato alla proprietà `frame_current`, serve a definire il fotogramma per il quale creare i keyframe relativi alla posizione di riposo. Il secondo è il bottone che consente di chiamare l'operatore `ANIM_OP_InsertRestPoseKeyFrame` che si

occupa di andare effettivamente a creare i keyframe necessari. Infine, viene presentato un testo che ha lo scopo di separare questa prima sezione dalla gestione singola delle blendshape.

Classe ANIM_OP_InsertRestPoseKeyFrame

L'operatore è così definito:

```
### OPERATOR SET A KEYFRAME FOR REST POSITION ########
class ANIM_OP_InsertRestPoseKeyFrame(bpy.types.Operator):
  """Insert a Rest Position KeyFrame at current frame"""
  bl_idname = "animation.insertrpkf"
  bl_label = "Insert Rest Position keyFrame"
  bl_options = {'REGISTER', 'UNDO'}

  def execute(self, context):
    insertKeyFrames_ActiveObj(context)
    return {'FINISHED'}
```

Principalmente si limita di effettuare la chiamata alla funzione `insertKeyFrames_ActiveObj` all'interno del metodo `execute`.

Funzione insertKeyFrames_ActiveObj

Questa funzione racchiude il codice che consente di creare correttamente i vari keyframe necessari per impostare l'espressione base per il frame indicato dall'animatore. Il codice che compone questa funzione è sostanzialmente identico a quanto visto a pagina 135 per la funzione `insertKeyFrames`, l'unica differenza sostanziale è dovuta al modo con il quale si va ad indicare l'oggetto su cui effettuare l'operazione. Siccome per questa fase si è deciso di non complicare l'UI andando ad utilizzare strumenti di selezione, l'oggetto su cui effettuare l'operazione è l'oggetto che risulta attivo nel momento in cui viene chiamato l'operatore. Dati i controlli effettuati precedentemente si è sicuri che l'oggetto attivo sarà di tipo armatura ma l'effettiva correttezza della selezione spetta all'animatore. Per consentire all'utente di avere sempre sotto controllo la situazione, come si può vedere nella figura 11.17, si è deciso di inserire nell'interfaccia un'indicazione sull'oggetto attivo in quell'istante.

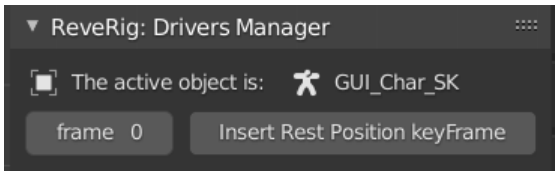


Figura 11.17: UI prima parte del pannello *Drivers Manager*

11.5.2 Sezione 2 - Gestione delle singole Shape Key

Se nella prima parte sono stati presentati strumenti che consentono di agire in modo automatico su tutte le blendshape, passando sempre attraverso il rig facciale, in questa seconda sezione si forniscono all'utente una serie di strumenti che consentono di agire manualmente sulle deformazioni apportate da ogni singola blendshape prevista da ReveRig. È tramite questa interfaccia che si va ad effettuare l'animazione facciale vera e propria, o per lo meno dove si

modifica in modo fine e a livello manuale l'animazione realizzata tramite i sistemi di cattura visti nei capitoli 9 e 10.

A differenza delle funzionalità viste fino ad ora, dove l'obiettivo principale era quello di automatizzare il più possibile determinati processi, come la creazione del rig facciale e le fasi di retargeting, in quest'ultima parte di ReveRig l'obiettivo è di agevolare l'animatore nel realizzare l'animazione facciale di miglior qualità possibile. Realizzare un'animazione facciale credibile è un processo incredibilmente complesso e, sebbene siamo riusciti ad individuare tecniche e tecnologie in grado di velocizzare e aiutare notevolmente tale lavorazione, risulta indispensabile una fase più o meno lunga di lavoro manuale basato sull'utilizzo di keyframe. Questa è una realtà che accomuna sia le produzioni indipendenti sia le grandi produzioni dell'industria cinematografica. Pertanto, risulta fondamentale offrire all'animatore tutti gli strumenti necessari per fare in modo che tale processo risulti il più veloce ed intuitivo possibile. Proprio con questo obiettivo in mente è stata realizzata quest'ultima sezione di ReveRig. In questa fase l'obiettivo non è più tanto la velocità quanto la precisione e la qualità del risultato finale.

Il pannello *Driver Manager* offre all'utente tutti i principali strumenti necessari per gestire in modo fine l'animazione vera e propria. Come anticipato precedentemente, dato che si vuole offrire la possibilità di agire su tutte le sessantadue blendshape, è risultato necessario organizzare l'UI in modo tale da garantire maggior usabilità. In particolare, si è deciso di suddividere in sette gruppi i controlli relativi alle diverse blendshape; tali gruppi sono basati sulla nomenclatura utilizzata per le diverse blendshape che a sua volta si basa sulle principali aree di interesse della faccia umana: sopracciglia (borws), occhi (eyes), palpebre (eyelids), guance e naso (cheeks&nose), bocca (mouth), mascella (jaw) e lingua (tongue). Per ciascuno dei gruppi indicati è stato creato un sotto pannello apposito, tale da contenere tutti gli strumenti utili per il controllo delle blendshape appartenenti a quel gruppo.

A comporre quest'ultima parte di ReveRig ci sono le seguenti componenti:

- Le due classi di tipo pannello `DMANAGER_PT_ui` e `DMANAGER_PT_Parent` descritte in modo approfondito a pagina 149 e pertanto di seguito non trattate.
- Le classi relative ai sette sotto-pannelli sopra citati: `DMANAGER_PT_Brows`, `DMANAGER_PT_Eyes_Eyelids`, `DMANAGER_PT_Eyes_Eyes`, `DMANAGER_PT_CheekNose`, `DMANAGER_PT_Mouth`, `DMANAGER_PT_Jaw`, `DMANAGER_PT_Tongue`.
- L'operatore `ANIM_OP_InsertSingleKeyFrame` che racchiude il codice utile per inserire i keyframe necessari.

Classi sotto-pannello

Come visto precedentemente, si è deciso di organizzare l'interfaccia tramite l'utilizzo di sette sotto pannelli, ciò ha comportato la realizzazione di sette classi separate. Ciascuna di queste classi fornisce l'interfaccia per la manipolazione di un gruppo definito di blendshape. È utile ricordare che la manipolazione non avviene direttamente su valori delle blendshape ma si va ad agire sulle diverse proprietà delle ossa che compongono il rig facciale precedentemente creato. La struttura interna di queste classi è simile per tutte ma in alcune di esse è stato necessario apportare delle modifiche in modo che meglio si adattassero alle esigenze particolari di determinate ossa del rig facciale. Tali peculiarità sono già state introdotte nelle sezioni precedenti, principalmente sono dovute alla struttura del rig facciale, ovvero al fatto che un unico osso sia in grado di gestire fino a quattro blendshape differenti. Nonostante queste differenze strutturali interne alle classi, le funzionalità messe a disposizione dell'utente finale sono pressoché identiche. Per questo motivo

non si va ad analizzare ogni singola classe ma si definisce il funzionamento generale facendo riferimento una volta ad una classe e una volta ad un'altra. La struttura di base può essere così riassunta:

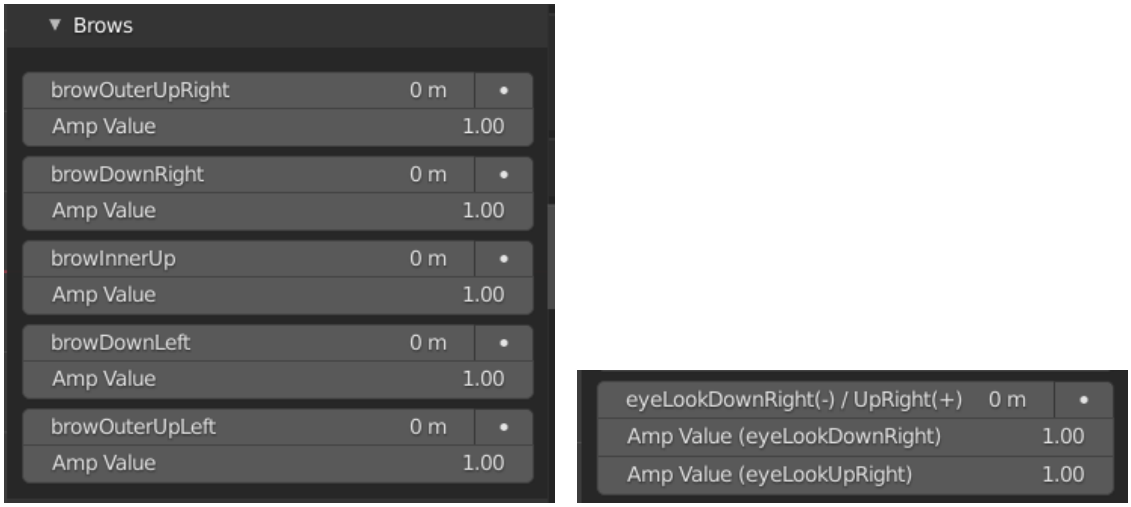
```
### Brows Sub-Panel
class DMANAGER_PT_Brows(DMANAGER_PT_ui):
  bl_parent_id = "MANEGER_PARENT_PT_layout"
  bl_label = "Brows"
  def draw(self, context):
    #inizailizzazione...
    if obj:
      if obj.type =='ARMATURE':
        for bone in obj.data.bones:
          if 'Slider_brow' in bone.name:
            bone_name = bone.name.split("Slider_")[1]
            col_1 = layout.column(align=True)
            split = col_1.split(factor=0.9, align=True)
            col = split.column(align=True)
            col.prop(obj.pose.bones[bone.name], "location",index=1, text=
    bone_name)
            col = split.column(align=True)
            col.operator("animation.insertsingkf", icon="DOT", text="").
    bone_property = str(bone.name)+'-Y'
            col_1.prop(obj.pose.bones[bone.name], "amp_value")
```

Dopo i primi controlli attuati per verificare che l'oggetto attivo sia un'armatura, si esegue un ciclo `for` per scorrere tutte le ossa del rig facciale. In ogni sotto pannello si va a lavorare su un gruppo differente di ossa. Ogni osso compare al più in un solo sotto pannello, questo serve per evitare confusione e possibili errori da parte dell'animatore. Nel codice sopra presentato, le ossa su cui si va a lavorare sono quelle aventi all'interno del nome la stringa '`Slider_borw`' ovvero quelle che controllano le blendshape relative alla deformazione delle sopracciglia. Per ciascun osso selezionato, si vanno a creare tre strumenti.

- Uno slider che consente di manipolare la traslazione dell'osso rispetto ad un determinato asse, tipicamente l'asse y (`index=1`); il nome utilizzato per tale slider è quello della blendshape corrispondente in modo da rendere tutto più intuitivo e facilmente comprensibile.
- Un bottone legato all'operatore `ANIM_OP_InsertSingleKeyFrame` che consente di inserire in modo semplice un keyframe alla proprietà gestita dallo slider visto sopra. Per tale bottone si è deciso di non utilizzare del testo ma di inserire un simbolo in grado di evocare nell'utente l'immagine di un keyframe.
- Un ultimo slider, nominato *Amp Val*, per controllare il valore della proprietà `amp_value` presente per ciascun osso.

Per ogni osso, questi tre strumenti vengono raggruppati insieme in modo da aumentare l'usabilità dell'UI; nella figura 11.18a viene mostrato il sotto pannello relativo alle sopracciglia. Per le ossa del rig facciale che consentono di gestire più blendshape contemporaneamente è stato necessario apportare alcune modifiche, sia alla struttura delle classi sia all'interfaccia finale. In particolare, ciò è dovuto al fatto che su uno stesso asse di traslazione, l'osso sia in grado di controllare due blendshape distinte ma con comportamento opposto. In questo gruppo rientrano le ossa per il controllo del movimento degli occhi e della mascella. Nella figura 11.18b è possibile vedere un esempio di interfaccia realizzata per questo tipo di ossa. Nell'esempio riportato si può notare

come il controllo del primo slider consente di manipolare la blendshape `eyeLookDownRight` per valori negativi [-1,0] e la blendshape `eyeLookUpRight` per valori positivi [0,1]. Tale divisione dello slider è basata su quanto realizzato per il rig facciale in modo tale che l'utente abbia una continuità di azione e il tutto risulti il più naturale possibile. Dopo di che ci sono due slider differenti per controllare la proprietà `amp_value` relativa a ciascuna blendshape. Per maggiore comprensione si è inserito tra due paretesi il nome relativo alla blendshape sulla quale lo slider ha effetto.



(a) Sotto-pannello per il controllo delle sopracciglia.

(b) Parte di UI per il controllo del movimento degli occhi.

Figura 11.18: Organizzazione dell'UI per la gestione manuale dell'animazione

Parte del codice relativo alla creazione dell'UI rappresentata in figura 11.18b è il seguente:

```
if obj:
  if obj.type =='ARMATURE':
    for bone in obj.data.bones:
      if bone.name=="Slider_eyeLookAllRight":
        #...
        col.prop(obj.pose.bones[bone.name], "location",index=1,text='eyeLook
    '+'DownRight(-) / UpRight(+)')
        col.operator("animation.insertsingkf", icon="DOT", text="").
    bone_property = str(bone.name)+'-Y'
        col_1.prop(obj.pose.bones['Slider_eyeLookDownRight'], "amp_value",
    text='Amp Value (eyeLookDownRight)')
        col_1.prop(obj.pose.bones['Slider_eyeLookUpRight'], "amp_value",
    text='Amp Value (eyeLookUpRight)')
```

In figura 11.19 viene mostrato come l'utilizzo dello slider *Amp Val* può influenzare l'intera espressione facciale. In particolare nella figura vengono mostrate le diverse espressioni ottenibili modificando unicamente l'`amp_value` della blendshape jawOpen.

Classe ANIM_OP_InsertSingleKeyFrame

A comporre questa parte dell'add-on ReveRig entra in gioco un solo operatore. Questo operatore è stato pensato per agevolare il processo di animazione facciale, offrendo all'utente

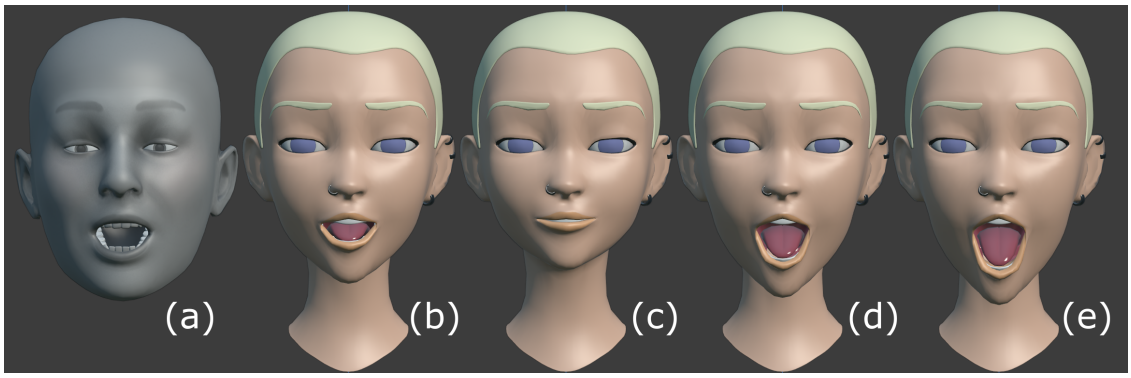


Figura 11.19: Esempio di utilizzo della proprietà amp_value. In figura (a) si ha l'animazione catturata con Face Cap, in figura (b) l'Amp Value della blendhsape jawOpen impostato a 1 (default), (c) a 0, (d) a 2 ed infine (e) a 2.5.

finale uno strumento che consenta di inserire nuovi keyframe o aggiornare quelli già presenti direttamente accanto allo slider su cui si va ad agire per impostare il valore. Questo consente all'animatore di gestire l'intera animazione senza dover necessariamente entrare in Pose Mode per controllare gli slider del rig facciale; ne consegue una significativa diminuzione delle tempistiche di lavoro. La struttura dell'operatore è la seguente:

```
### OPERATOR INSERT OR UPDATE A SINGLE KEYFRAME ########
class ANIM_OP_InsertSingleKeyFrame(bpy.types.Operator):
  """Insert or Replace Single Keyframe"""
  bl_idname = "animation.insertsingkf"
  bl_label = "Insert/Replace Single KeyFrame"
  bl_options = {'REGISTER', 'UNDO'}

  bone_property: bpy.props.StringProperty(name ='text',default = '')
  def execute(self, context):
    obj_act = bpy.context.active_object
    FRAME_CURRENT = bpy.context.scene.frame_current

    bone_name = self.bone_property.split('-')[0]
    property_name = self.bone_property.split('-')[1]

    if property_name=='Y':
      obj_act.pose.bones[bone_name].keyframe_insert(data_path='location',
    index=1, frame=FRAME_CURRENT)
    elif property_name=='X':
      obj_act.pose.bones[bone_name].keyframe_insert(data_path='location',
    index=0, frame=FRAME_CURRENT)
    return {'FINISHED'}
```

Come si può notare, il codice che è stato realizzato per portare a termine il compito, in questo caso è direttamente eseguito all'interno del metodo `exectue` senza dover passare attraverso una funzione esterna. Prima di poter creare il keyframe è necessario definire su quale fotogramma inserirlo, ovvero il frame corrente, per quale oggetto e per quale proprietà. L'oggetto viene individuato come l'oggetto attivo al momento dell'operazione, ovvero il rig facciale, mentre determinare su quale osso e per quale proprietà creare il keyframe necessita di un passaggio in più.

All'interno dell'operatore viene creata una nuova proprietà denominata `bone_property` che avrà lo scopo di contenere il nome dell'osso di riferimento e dell'asse sul quale agire. Infatti, se analizziamo il codice a pagina 153 si noterà che la chiamata all'operatore è realizzata in questo modo:

```
col.operator("animation.insertsingkf", icon="DOT", text="").bone_property =
    str(bone.name)+'-Y'
```

Oltre a effettuare la chiamata all'operatore si va a passare la stringa relativa alla proprietà `bone_property` composta dal nome dell'osso e dall'asse di traslazione per il quale inserire il keyframe. In questo modo, all'interno del metodo `execute` dell'operatore, diventa immediato discriminare le diverse condizioni. Il keyframe viene quindi inserito unicamente per l'asse di traslazione indicato e non per tutti e tre gli assi, come si avverrebbe andando ad inserire il keyframe tramite l'utilizzo dei comandi da tastiera, ovvero premendo il tasto `I` quando il cursore del mouse si trova sulla proprietà desiderata. L'inserimento dei keyframe solo per le proprietà necessarie risulta un grande vantaggio in quanto riduce significativamente le F-curve realizzate e riduce quindi la complessità di gestione dell'intera animazione facciale.

Quest'ultima funzione completa le possibili operazioni che possono essere eseguite tramite il pannello *Drivers Manager* di ReveRig. Conclude così la descrizione delle principali funzionalità introdotte con ReveRig e i principali vantaggi apportati nella realizzazione di animazioni facciali tramite Blender.

Nella sezione a seguire verrà trattata la distribuzione sul mercato internazionale dell'Add-On ReveRig; verranno riportate le motivazioni che hanno spinto in questa direzione e i primi risultati ottenuti.

11.6 ReveRig: distribuzione internazionale

All'interno di questa sezione andremo a descrivere più nel dettaglio le motivazioni che hanno portato alla pubblicazione online dell'Add-On ReveRig, i primi risultati ottenuti e le possibili prospettive future che tale operazione ha creato.

Come indicato a più riprese all'interno di tutto il documento di tesi, sebbene l'obiettivo principale sia quello di fornire a Robin Studio S.r.l.s. un workflow produttivo che agevoli il processo di animazione facciale riducendo le tempistiche e le risorse impiegate, si è sempre mantenuto uno sguardo rivolto verso il mercato internazionale cercando di captare esigenze e bisogni nascenti.

Grazie alle ricerche approfondite necessarie per offrire la miglior soluzione allo studio Robin, si è avuta la possibilità di approfondire le dinamiche interne al settore dell'animazione facciale indipendente, in particolar modo relativo all'utilizzo del sistema di motion capture markerless ARKit e del software 3D Blender. É risultato immediatamente evidente che la maggior parte delle richieste e le necessità dello studio Robin per la realizzazione di *Reverie Dawnfall* fossero del tutto simili a quanto richiesto dal mercato indipendente. Da questa evidenza nasce l'idea di progettare e sviluppare l'Add-On ReveRig mantenendo un approccio più generico e trasversale senza però perdere il focus sull'obiettivo principale del progetto di tesi.

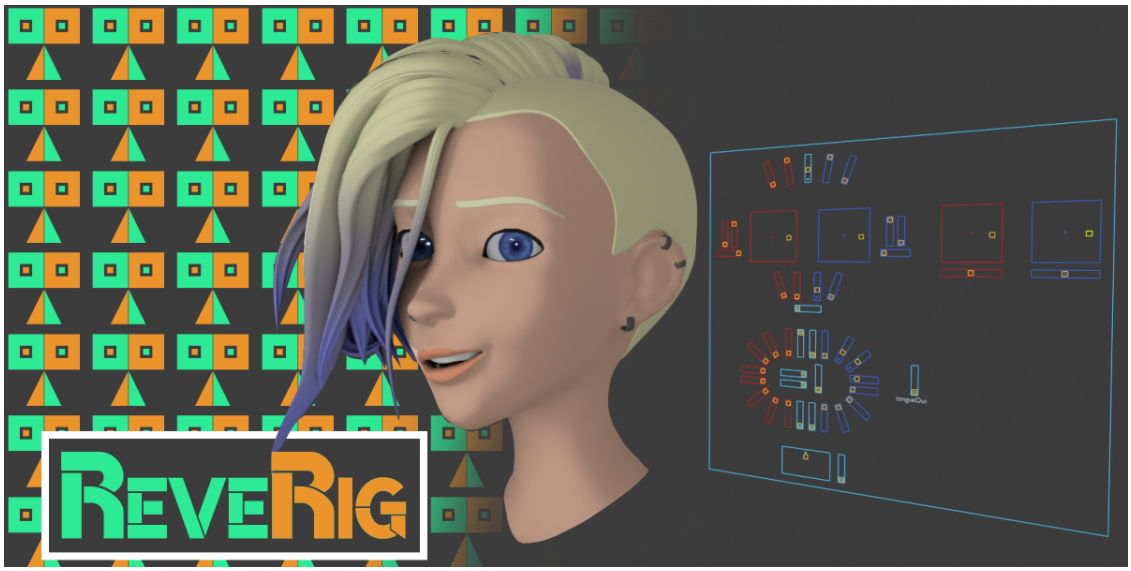


Figura 11.20: ReveRig - identità visiva

Il risultato finale consiste in due versioni differenti di ReveRig. Una prima versione, ampiamente descritta nelle sezioni precedenti di questo capitolo, pensata per soddisfare in modo completo e preciso le necessità di Robin Studio. Questa racchiude al suo interno più funzionalità, come ad esempio tutta la fase dell'animazione della lingua, ed ha una complessità di utilizzo più elevata. La seconda versione, basata sulla prima, invece è pensata per essere distribuita sul mercato internazionale e quindi ha lo scopo di soddisfare un'audience più ampia. Per questo si è deciso di escludere alcune funzionalità inserite per lo studio Robin e quindi di semplificare in parte l'esperienza utente complessiva.

Le principali differenze tra le due versioni di ReveRig riguardano sostanzialmente la fase di animazione della lingua e quindi tutte le funzionalità ad essa correlate. Siccome all'interno del workflow realizzato per *Reverie Dawnfall* l'animazione della lingua viene effettuata per la maggior parte tramite il programma Papagayo-NG (vedi capitolo 10), questo introduce un livello di complessità che difficilmente gli utenti di Blender utilizzerebbero all'interno dei loro progetti. Per questo motivo, nella versione destinata alla community di Blender, si è deciso di non inserire queste funzionalità, riducendo in questo modo anche le possibili problematiche lato cliente. Un'altra variazione introdotta nella versione destinata alla distribuzione consiste nel fatto che in quest'ultima, nella fase di creazione dei driver, non è più obbligatorio dover indicare la mesh per gli occhi ma rimane una componente facoltativa. Questa modifica è stata introdotta in quanto la diversità di progetti e la diversità di personaggi che gli utenti vorranno animare sarà molto vasta, pertanto offrire un maggior grado di libertà consente di includere anche scenari non previsti nella realizzazione di *Reverie Dawnfall*.

La rapida diffusione dagli strumenti di motion capture facciale promossi da Apple ha generato un crescente interesse verso la realizzazione di animazioni facciali anche in contesti completamente indipendenti (piccoli studio di animazione, liberi professionisti e artisti indipendenti).

Unendo a questa tendenza, anche il fatto che Blender sta diventando sempre più popolare e sempre più apprezzato sia in contesti completamente indipendenti che in contesti più professionali, emerge che il settore indipendente dell'animazione facciale può offrire un ottimo potenziale di crescita.

Dalle valutazioni sopra riportate, si è deciso di pubblicare online la nuova versione di ReveRig e di renderla disponibile potenzialmente a tutta la community di Blender, con l'intento quindi di offrire un servizio utile e di valore.

La distribuzione online di ReveRig ad oggi (23/06/2021) può contare due canali principali: uno spazio dedicato sulla piattaforma di distribuzione Blender Market e un canale YouTube ufficiale[3].

Dopo aver preparato tutto il materiale necessario per l'avvio dell'attività, in data 13/04/2021 è stata aperta la pagina ufficiale di ReveRig sulla piattaforma di distribuzione Blender Market. Blender Market è il più grande mercato indipendente online dedicato alla community di Blender. Creato nel 2014 da CG Cookie, ha uno stretto legame con la Blender Foundation alla quale recapita parte degli incassi. Nella figura 11.21 vengono riportate le statistiche di Blender Market per il mese di giugno 2021. Come si può notare l'intero mercato è molto attivo e offre un enorme potenziale per i creators. Oltre al potenziale raggiungimento di una vasta audience, si è scelto di distribuire ReveRig tramite Blender Market in quanto questo offre ai propri creator una serie di strumenti molto utili per gestire ed organizzare la distribuzione stessa. Inoltre, si prende carico di tutta la parte di gestione degli ordini e di invio del prodotto, riducendo significativamente il lavoro necessario da parte del venditore. Offre poi, una serie di strumenti per mettersi in contatto con clienti acquisiti e potenziali e per effettuare il customer service in modo pratico.

Per ReveRig, su Blender Market, sono state create diverse sezioni in modo da andare incontro alle esigenze degli utenti. Una pagina di presentazione del prodotto, con diretto riferimento al canale YouTube; una pagina dedicata alla documentazione; una pagina contenente le principali FAQ ed infine una pagina per le eventuali recensioni da parte dei clienti.

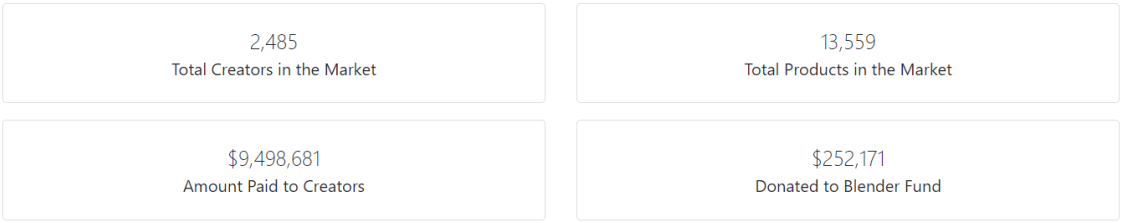


Figura 11.21: Statistiche di Blender Market (1/06/2021 - 23/06/2021)

In aggiunta al profilo di distribuzione su Blender Market, si è deciso di aprire e gestire il canale YouTube ufficiale di ReveRig. Le motivazioni che ci hanno spinto a creare il canale YouTube sono principalmente due. Innanzitutto, si è reso necessario fornire ai clienti una serie di strumenti per comprendere in modo semplice ed immediato il funzionamento di ReveRig; siccome la documentazione scritta, seppur completa e precisa, può risultare non di facile comprensione per tutto il pubblico, allora si è deciso di realizzare dei video tutorial in modo che gli utenti potessero

[3]Pagina di ReveRig su Blender Market. URL: https://blendermarket.com/products/reverig. Canale YouTube ufficiale di ReveRig. URL: https://www.youtube.com/channel/UCfuZ6N12J65fiP3M51WJPgQ

Figura 11.22: Identità visiva di ReveRig su YouTube.

seguire passo per passo quanto mostrato. Da qui è divenuto naturale aprire un canale YouTube per pubblicare i video tutorial. Oltre ad una funzione di supporto al cliente, il canale YouTube è stato pensato anche come punto di ritrovo e di riferimento per i clienti già acquisiti e i potenziali clienti. Infatti grazie a YouTube è possibile raggiungere nuova audience, creare un'identità del brand (vedi figura 11.22) e generare brand awareness. Inoltre risulta uno strumento molto utile per convertire nuovi potenziali clienti e dirigerli verso la pagina Blender Market.

11.6.1 Primi Risultati

Di seguito si presentano i primi risultati ottenuti dalla distribuzione online di ReveRig. Come vedremo, sebbene ReveRig sia sul mercato da poco più di due mesi (13/04/2021 - 23/06/2021), ha ottenuto dei risultati molto incoraggianti.

Distribuzione su Blender Market Come indicato in precedenza, ReveRig è stato reso ufficialmente disponibile online il 13 aprile 2021. Ad oggi (23/06/2021) ReveRig ha totalizzato un totale di 44 vendite e 3992 visualizzazioni del prodotto. Il dato sicuramente più interessante riguarda il numero di vendite giornaliere e il numero di vendite cumulate realizzate nel periodo analizzato. Come è possibile vedere nel grafico in figura 11.23a, i primi acquisti vengono effettuati già dal giorno successivo alla pubblicazione online. Dall'analisi del grafico in figura 11.23a è possibile vedere l'andamento giornaliero delle vendite ma è complicato identificare una tendenza generale. Per questo motivo è stato realizzato il grafico in figura 11.23b che mostra l'andamento cumulativo delle vendite totali. Sebbene il periodo di analisi è breve e i dati a disposizione non sono molti, grazie al grafico 11.23b è possibile affermare con discreta sicurezza che l'andamento delle vendite segue una crescita pressoché lineare.

Nel grafico in figura 11.23c vengono mostrate le visualizzazioni giornaliere delle pagine relative a ReveRig su Blender Market. Ad una prima vista l'andamento mostrato è piuttosto inusuale. In questo grafico si possono distinguere tre componenti principali: una fase iniziale di rapida crescita e successiva decrescita; un picco di visualizzazioni in data 24 maggio 2021 e infine una tendenza lineare e piuttosto costante che occupa la maggior parte del grafico. Il picco iniziale è dovuto all'organizzazione interna a Blender Market, infatti la piattaforma ha deciso di fornire maggiore visibilità ai prodotti pubblicati più recentemente, questo spiega come mai dopo i primi giorni le visualizzazioni abbiano iniziato a calare per poi stabilizzarsi sulle 50 giornaliere. Il secondo e più evidente picco (data 24 maggio 2021) è dovuto a cause ancora non ben definite, come vedremo quando analizzeremo i dati di YouTube, tale picco si è trasferito anche sulla piattaforma di condivisione video. Sebbene non si abbiano fonti certe, la spiegazione più logica prevede che con molta probabilità il prodotto ReveRig sia stato inserito per un certo lasso di tempo nella prima pagina del sito. Esclusi i due picchi si può notare come l'andamento generale sia pressoché lineare e globalmente costante nel tempo.

Analizzando tutti i dati si può notare che l'interesse del pubblico verso ReveRig è piuttosto costante e pertanto è plausibile pensare che, senza attivare nessuna campagna di promozione, l'andamento generale rimanga costante nel breve e medio periodo. Tuttavia risulta evidente anche che al momento, sebbene i dati siano incoraggianti, non si è verificata ancora una vera e propria esplosione di interesse verso ReveRig e questo ci può far capire che potrebbe essere necessario creare maggior brand awereness e raggiungere un pubblico più ampio.

Il canale YouTube Il canale YouTube di ReveRig conta in data 24/06/2021 un totale di cinque video caricati. I video, sebbene abbiano tutti la funzione di tutorial, possono essere divisi in tre categorie principali. Il primo video è dedicato alle istruzioni da seguire per l'installazione dell'add-on; poi vi sono i due video con durata maggiore che hanno lo scopo di spiegare e mostrare agli utenti come ReveRig può essere utilizzato in combinazione con FaceCap e Face Capture (fase di retargeting); infine vi sono due video di durata minore che si concentrano maggiormente sulla fase di rigging e di creazione dei driver.

Il canale conta 55 iscritti, un totale di circa 3500 visualizzazioni e 24700 impression (vedi grafico in figura 11.24a). Analizzando più nel dettaglio i dati però si nota che il video *ReveRig w/ FaceCap: How to Retarget Facial Mocap Animation* conta da solo più di 2500 visualizzazioni. Il maggior traffico generato da questo video è dovuto al fatto che all'interno della pagina Blender Market di presentazione di ReveRig c'è una call to action mirata a questo tutorial. Analizzando il grafico in figura 11.24b è possibile notare come gli utenti unici registrati siano poco meno di duemila, ciò significa che più utenti hanno visualizzato il video più volte. Un altro dato interessante che si può cogliere è che la curva, seppur leggera, ha una tendenza crescente; ciò significa che l'interesse verso ReveRig sta aumentando nel tempo. Inoltre, si può notare che in data 24 maggio 2021 si è verificato un picco di 99 visitatori; tale dato ci conferma che il picco visto in figura 11.23c (di più di 400 visualizzazioni) sia dovuto a meccaniche interne a Blender Market in quanto i dati evidenziano che il traffico si è originato su Blender Market e poi in parte si è spostato su YouTube e non viceversa.

Accoglienza da parte degli utenti Sebbene i dati mostrati sopra siano importanti per individuare quali sono le tendenze generali relative a ReveRig, questi non riescono a catturare completamente quella che è stata l'accoglienza da parte degli utenti. Nonostante il periodo di analisi sia breve e i dati raccolti siano piuttosto limitati, è possibile affermare che la community di

Blender abbia accolto con entusiasmo l'add-on ReveRig. Su un totale di 44 clienti, non sono mai stati riscontrati problemi ne tanto meno è mai stato richiesto il rimborso. Questo dimostra che ReveRig funziona in contesti anche molto diversi tra loro ed effettivamente offre un servizio utile. Inoltre, non si sono mai registrati commenti o recensioni negative sul prodotto in nessuna delle due piattaforme utilizzate; anzi, su Blender Market ReveRig ha ricevuto una recensione da cinque stelle su cinque, e analizzando il canale YouTube è possibile notare che tutti i video caricati contano il 100% di like rispetto ai dislike, e che i commenti pubblicati sono per la maggior parte di apprezzamento o di richiesta di maggiori informazioni. Inoltre, è già presente su YouTube un video generato da utenti (UGC) dove viene consigliato l'utilizzo di ReveRig per il processo di animazione facciale. Questo dimostra che ReveRig effettivamente porta a termine ciò che si propone di fare e che le funzionalità, l'interfaccia e l'esperienza utente complessiva sono state progettate e sviluppate in modo accorto ed efficace.

Oltre ai commenti e alle recensioni di pubblico dominio, molti sono stati i messaggi ricevuti da clienti e da potenziali clienti. Grazie a questo dialogo diretto con gli utenti è stato possibile raccogliere maggiori informazioni relative alle esigenze più richieste così da indirizzare in modo più preciso gli sforzi per i futuri sviluppi di ReveRig.

Grazie a questa raccolta di informazioni è stato possibile riscontrare che in molti richiedevano la possibilità di tradurre automaticamente la nomenclatura delle blendshape create con FaceCap nella nomenclatura classica prevista da ARKit, ReveRig, Faceit e Face Capture. Si è deciso pertanto di aggiornare l'add-on ReveRig aggiungendo questa nuova funzionalità direttamente all'interno del processo di retargeting. In questo modo è stato possibile mantenere la stessa interfaccia utente senza aggiungere maggiore complessità di utilizzo. Tale modifica del codice sorgente è stata poi inserita anche all'interno della versione di ReveRig destinata allo studio Robin.

Questo può considerarsi solo il primo di molti aggiornamenti futuri. Infatti, in una situazione di avanzamento tecnologico repentino come quella che stiamo vivendo oggigiorno, è fondamentale ascoltare il mercato e rispondere alle nuove esigenze.

Sviluppi futuri Se fino ad oggi la versione di ReveRig destinata a Robin Studio e la versione destinata al mercato sono pressoché uguali e strettamente legate, è possibile che negli sviluppi futuri la versione destinata alla distribuzione veda l'introduzione di funzionalità non necessarie all'obiettivo dello studio Robin e viceversa.

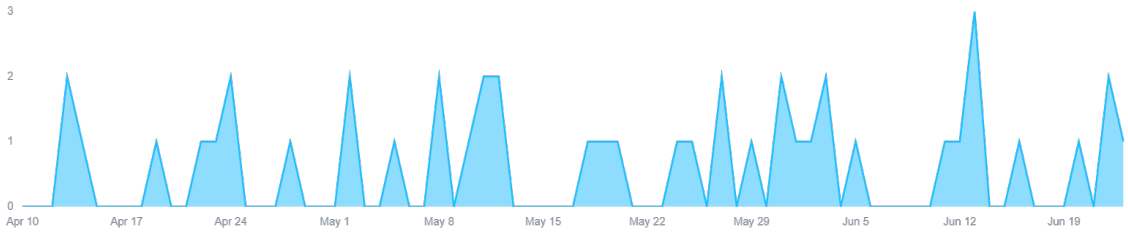
In particolare è possibile registrare un aumento di interesse verso la creazione di Avatar virtuali animati in tempo reale, chiamati anche Youtuber virtuali o VTuber (vedi figura 11.25). Pertanto per i futuri sviluppi di ReveRig potrebbe essere interessante introdurre delle funzionalità che vadano a consentire agli utenti di animare in tempo reale il volto dei propri personaggi, effettuando il trasferimento dei dati tramite OSC o altri sistemi, consentendo contemporaneamente di mantenere la funzionalità di gestione dell'espressività del personaggio.

Oltre alla possibilità di introdurre nuove funzionalità ed effettuare modifiche al codice, si stanno aprendo interessanti opportunità relative a collaborazioni con realtà più importanti. In particolare è nata da poco una collaborazione con gli sviluppatori dell'add-on di Blender *Human Generator*[4]. Anche questo è uno strumento nato di recente ed ha riscontrato immediatamente un enorme successo; al momento conta più di 3000 vendite ed è in maniera quasi costante presente nella home page di Blender Market. Gli sviluppatori mi hanno contattato per iniziare

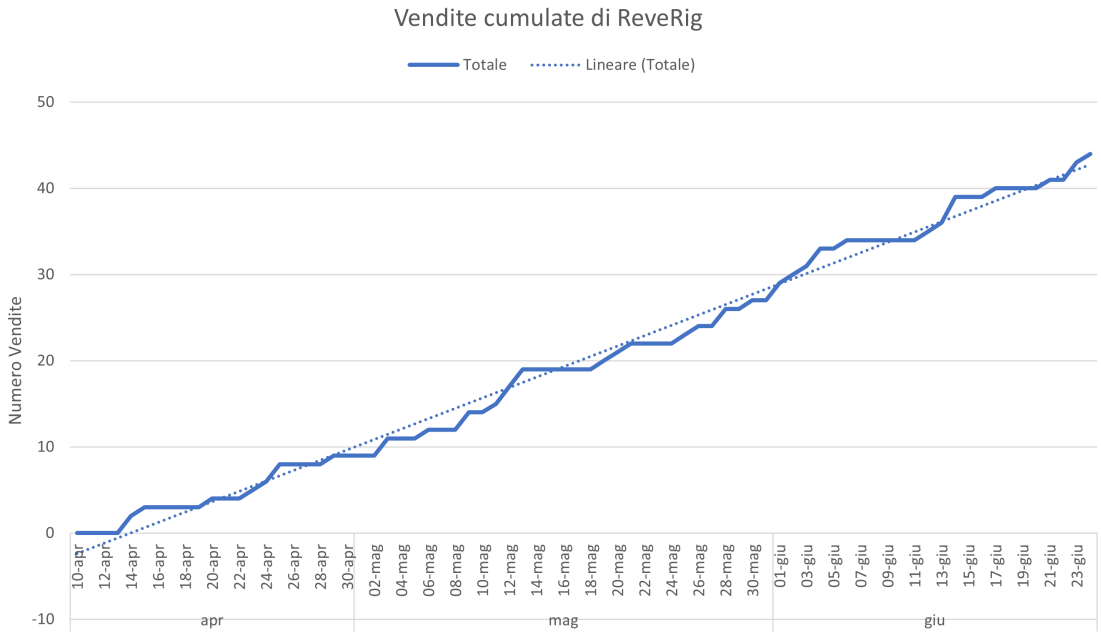
[4]Sito ufficiale di Human Generator. URL: https://www.humgen3d.com/.

una collaborazione nella quale io offro consulenza relativamente alle creazione e gestione delle blendshape definite da ARKit in modo che loro possano implementarle all'interno dei loro personaggi. In cambio loro garantiscono di rendere Human Generator compatibile con ReveRig e di consigliarlo alla propria utenza per chi avesse necessità di realizzare animazioni facciali più complesse. Al momento tale collaborazione è agli stadi iniziali in quanto gli sviluppatori sono al lavoro per cercare di creare in modo automatico le blendshape necessarie. Nel momento in cui tale funzionalità verrà rilasciata sul mercato, potrebbe rivelarsi un'enorme opportunità per la crescita di ReveRig all'interno della community di Blender.

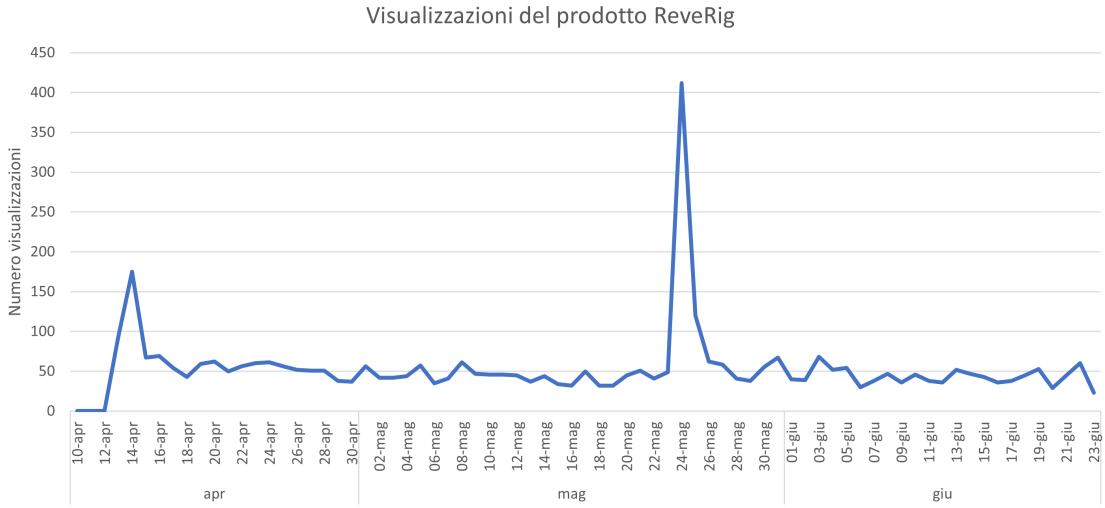
La scelta di distribuire online ReveRig ha portato significativi vantaggi, ha consentito di introdurre una nuova funzionalità all'interno dell'add-on, di entrare in relazione più stretta con le esigenze degli utenti ed infine ha aperto nuove prospettive per i futuri sviluppi dell'add-on. La possibilità di aver realizzato uno strumento utile sia per Robin Studio che potenzialmente per l'intera community di Blender è motivo di orgoglio ed è dimostrazione della qualità del lavoro svolto all'interno di questa tesi.



(a) Vendite giornaliere di ReveRig (periodo 10/04/2021 - 23/06/2021)



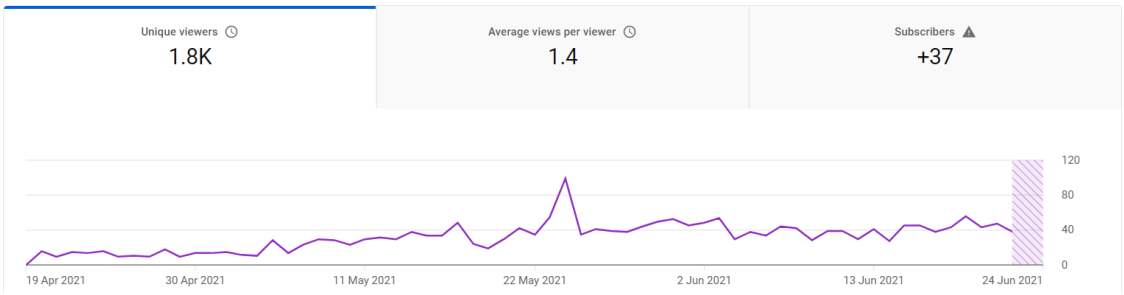
(b) Vendite cumulate di ReveRig (periodo 10/04/2021 - 23/06/2021)



(c) Visualizzazioni del prodotto ReveRig su Blender Market (periodo 10/04/2021 - 23/06/2021)

Figura 11.23: Statistiche del prodotto ReveRig su Blender Market

(a) Statistiche canale YouTube di ReveRig (periodo 25/03/2021 - 22/06/2021)



(b) Statistiche del video *ReveRig w/ FaceCap: How to Retarget Facial Mocap Animation* (periodo 20/04/2021 - 24/06/2021)

Figura 11.24: Statistiche del canale YouTube di ReveRig

Figura 11.25: Ricerche via Google dell'argomento YouTuber virtuale (periodo 1/01/2017 - 24/06/2021). Fonte Google Trends (`https://trends.google.it/trends/`)

Capitolo 12

Implementazione del nuovo workflow

All'interno di questo capitolo tratteremo i primi risultati ottenuti nel processo di animazione facciale per la serie animata *Reverie Dawnfall* grazie all'introduzione del nuovo workflow, composto dalle metodologie, dalle tecnologie e dagli strumenti analizzati all'interno di questo documento di tesi.

Prima di procedere è importante fare una precisazione. Il lavoro di tesi è cominciato ed è portato avanti durante la pandemia da SARS-CoV-2; questo ha comportato che l'intero lavoro sia stato svolto in modalità da remoto. Tale condizione ha introdotto dei limiti in merito all'applicazione di quanto presentato all'interno del documento di tesi nell'effettiva realizzazione della serie animata *Reverie Dawnfall*. Pertanto, i risultati riportati all'interno di questo capitolo si basano su uno scenario rivisitato, tuttavia è stato possibile utilizzare il materiale a disposizione per effettuare le ricerche e i test necessari per fornire dei risultati chiari all'interno della tesi. Per massima trasparenza, lo scenario "completo" prevedeva l'introduzione del nuovo workflow all'interno della produzione di una scena vera e propria della serie animata; andando ad integrare completamente il processo di animazione facciale con il processo di animazione globale del personaggio. Come sopra accennato, tale soluzione non è stata fattibile, pertanto, i test si sono svolti basandosi unicamente sul processo di animazione facciale, utilizzando i dati a disposizione per ipotizzare che l'integrazione con il processo di cattura del moto tramite tuta inerziale sia del tutto efficace.

12.1 Il processo di lavoro in step

Sebbene tutte le fasi del nuovo workflow siano state presentate nei capitoli precedenti del documento, risulta utile fornire una versione semplificate dei vari step previsti e seguiti nella creazione delle animazioni facciali per la serie *Reverie Dawnfall*.

1. **Condizione di partenza**
   Per prima cosa è necessario avere a disposizione i modelli 3D dei personaggi completi di tutte le mesh necessarie. Nel caso sia necessario effettuare delle modifiche alle geometrie relative al volto, occhi o bocca, è bene apportarle prima di procedere con gli step successivi.

Inoltre, tutti i personaggi di interesse devono essere stati precedentemente riggati come descritto in [33].

2. **Creazione delle blendshape**
   Ipotizzando di avere il modello completo del personaggio ma non dotato delle blendshape a noi utili, allora è necessario creare tutte le shape key indicate nell'appendice C. Le 52 blendshape definite da ARKit possono essere generate utilizzando l'add-on Faceit seguendo i passaggi riportati nella sezione 9.2.1. Per quanto riguarda invece le 6 blendshape necessarie per l'animazione della lingua e le 4 blendshape per l'animazione delle pupille, è necessario procedere manualmente per ogni personaggio. Una volta che il personaggio che vogliamo animare è completo di tutte le blendshape richieste, allora è possibile procedere con le operazioni di animazione facciale vere e proprie.

3. **Acquisizione dei dati**
   Per la serie animata *Reverie Dawnfall* si è deciso di utilizzare le tecniche di motion capture markerless per velocizzare e semplificare notevolmente il processo di animazione. Come visto nel capitolo 9 la scelta ricade principalmente sull'utilizzo dell'applicazione FaceCap di Bannaflak o dell'app Face Capture di Rokoko; entrambe valide per il nostro obiettivo. É ora arrivato il momento di catturare l'animazione del volto con gli strumenti di Mocap facciale e i dispositivi Apple compatibili. Per i test sono stati utilizzati due iPhoneX e sono state testate entrambe le applicazioni in modo da poter individuare quella più adatta al progetto. Dopo aver catturato l'animazione è necessario esportarla come file FBX in modo da poterlo importare all'interno di Blender. Per maggiori informazioni fare riferimento al capitolo 9.

   In contemporanea alla cattura del moto è consigliato registrare anche la traccia audio relativa al dialogo recitato. Sebbene l'audio finale può essere realizzato in fase di doppiaggio, è fondamentale avere la traccia originale come riferimento durante il processo di animazione.

   Come precisato ad inizio capitolo, per questo lavoro di tesi i dati relativi all'animazione facciale sono stati catturati in sessioni dedicate. Tuttavia, rimane possibile effettuare l'acquisizione dei dati relativi all'animazione del corpo (tramite tuta inerziale) e del volto (tramite mocap markerless) in un'unica sessione. In questo modo è possibile risparmiare tempo ed inoltre si avrà maggior corrispondenza tra tutte le azioni del personaggio, rendendo l'animazione più organica. Per fare questo potrebbe essere necessario l'utilizzo di head mounted camera o altri sistemi per fissare il dispositivo Apple in modo da riprendere costantemente il volto dell'attore senza ingombrarne i movimenti. All'interno di questo lavoro di tesi non è stato possibile effettuare test più approfonditi in merito, pertanto, questo potrebbe essere un campo di interesse per future ricerche.

4. **Creazione del rig facciale**
   Basandosi sull'organizzazione dei file di progetto vista nella sezione 11.2, questo step riguarda l'utilizzo del primo pannello di ReveRig (*Facial Rigging*) per la creazione del rig facciale e dei driver necessari per controllare l'animazione delle blendshape del personaggio in questione. Tale passaggio va effettuato all'interno del file di progetto del personaggio (cartella Assets).

5. **Retargeting dell'animazione facciale**
   Ora è possibile spostarsi sul file di progetto relativo alla scena da realizzare (cartella Scenes);

tramite link bisogna importare tutti gli asset necessari (personaggi e ambientazione) e poi importare il file fbx relativo all'animazione facciale precedentemente catturata; quindi procedere con l'animazione facciale vera e propria.

Grazie all'utilizzo delle funzionalità presenti nel secondo pannello di ReveRig (*Animation Retargeting*) è possibile trasferire l'animazione dall'oggetto fbx importato sul nostro personaggio in modo semplice e veloce.

6. **Controllo fine dell'animazione**
   Effettuata l'operazione di retargeting, che può richiedere al massimo qualche minuto, è possibile utilizzare il rig facciale e l'interfaccia *window-based* del pannello *Drivers Manager* offerti da ReveRig in combinazione con il rig facciale di tipo *viewport 3D*, precedentemente creato assieme allo scheletro del corpo, per controllare in modo fine l'animazione, correggendo eventuali imprecisioni dovute al processo di cattura del volto e gestendo nel modo desiderato tutte le espressioni facciali così da ottenere l'animazione desiderata.

7. **Animazione della lingua**
   Una volta che siamo soddisfatti dell'animazione realizzata per volto, occhi e bocca è possibile procedere con l'animazione della lingua. Per prima cosa è necessario spostarsi sul programma Papagayo-NG e seguire i passaggi descritti nella sezione 10.4 per generare il file .dat contenente tutti i dati necessari. Dopo di che è possibile tornare su Blender e utilizzare le funzionalità presenti nel pannello *Animation Retargeting* di ReveRig per importare i dati relativi all'animazione della lingua e rimapparli seguendo i valori dell'apertura della mascella (per maggiori informazioni vedi la sezione 11.4.2). Nel caso fosse necessario è possibile tornare allo step 6 per aggiustare l'animazione finale.

Terminate queste operazioni si conclude il processo di animazione facciale per i personaggi della serie animata *Reverie Dawnfall*.

12.2 Risultati ottenuti

Di seguito descriviamo ed analizziamo i principali risultati ottenuti mettendo a confronto quanto realizzato seguendo il workflow proposto con questo lavoro di tesi con quanto si sarebbe ottenuto utilizzando il vecchio procedimento. Le metriche principali alle quali si farà riferimento riguardano la qualità del prodotto finale ottenuto e le risorse, principalmente in termini di tempo, richieste per portare a termine il compito. Siccome il settore dell'animazione è molto complesso e i progetti realizzati possono variare molto tra di loro, i dati riportati hanno una natura più qualitativa che non quantitativa in quanto risulta estremamente complicato ricreare le stesse condizioni di partenza per ogni test effettuato.

12.2.1 Creazione delle blendshape

Il workflow di animazione facciale progettato prevede l'utilizzo di 62 blendshape principali (vedi appendice C). Come già discusso nei capitoli 3.2 e 9.2, il processo di creazione delle blendshape è tendenzialmente molto lungo e in produzioni di alta qualità può richiedere più di un anno di lavoro svolto da professionisti specializzati [13]. Sebbene non sia questa la situazione di partenza, dal documento della collega Abate [33] emerge che il processo di creazione delle

52 blendshape definite da ARKit per il personaggio di Nadya abbia richiesto moltissimo lavoro manuale e un lungo processo di trial&error. Nonostante tutto il lavoro svolto, il risultato finale presentava l'insorgenza di diverse interferenze e di una mancata corrispondenza con l'animazione catturata [33].

Con il nuovo workflow proposto, grazie all'introduzione dell'add-on Faceit, è stato possibile creare tutte le 52 blendshape definite da ARKit in tempi ridotti, circa due ore di lavoro per le 52 shape key del volto e un'altra ora circa per la creazione delle 6 blendshape della lingua e delle 4 blendshape per le pupille.

Con l'utilizzo di Faceit è stato quindi possibile ridurre significativamente le tempistiche necessarie per la creazione della maggior parte delle blendshape richieste e ottenere al contempo un risultato privo di interferenze visibilmente fastidiose. Questo ha permesso di avere il tempo per creare nuove blendshape per l'animazione di lingua e pupille prima non previste, migliorando in questo modo l'espressività complessiva del personaggio. In figura 12.1 è possibile mettere a confronto il risultato ottenuto prima e dopo l'introduzione del workflow proposto con questo lavoro di tesi. Sebbene l'immagine sia statica, grazie al confronto diretto con l'animazione catturata con Face Cap (figura 12.1a)), è possibile notare la grande differenza ottenuta utilizzando i due diversi processi di lavoro. La presenza di interferenze tra le diverse blendshape e la mancata corrispondenza con l'animazione catturata è molto marcata nel risultato ottenuto con il vecchio processo di creazione delle blendshape (figura 12.1b)); al contempo si può notare che con il nuovo processo di lavoro proposto, il risultato ottenuto (figura 12.1c)) sia più organico e più fedele ai dati catturati.

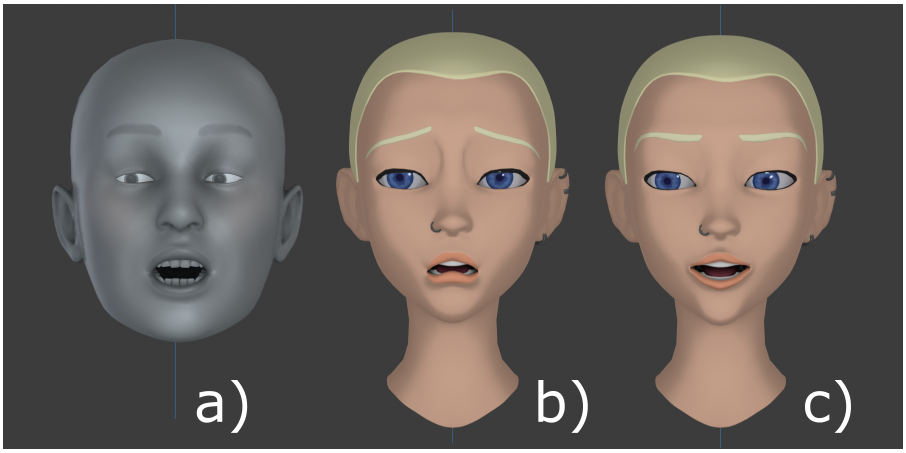


Figura 12.1: Confronto tra i risultati ottenuti partendo dalla stessa animazione catturata con Face Cap a), con la creazione manuale delle blendshape b) e con l'utilizzo di Faceit c).

Non solo l'introduzione di Faceit ha ridotto significativamente il tempo necessario per la generazione delle blendshape, ma la miglior qualità di ciascuna delle blendshape create garantisce una riduzione importante del lavoro svolto durante l'animazione vera e propria in quanto saranno richiesti meno aggiustamenti e correzioni.

Per quanto riguarda il personaggio di Jameela non è possibile effettuare un confronto diretto in quanto nel lavoro dei miei colleghi, Jameela non era dotata di tutte le 52 blendshape di ARKit ma solo di un set di cinque shape key relative ad espressioni facciali standard. Tale scelta era stata presa per questioni di tempo [33].

Tale situazione avvalora la tesi che l'introduzione dell'add-on Faceit all'interno del processo di animazione facciale abbia introdotto enormi vantaggi, sia per la qualità dell'animazione ottenuta che per l'importante risparmio di tempo e quindi di risorse. Inoltre, come vedremo meglio dopo, l'introduzione dell'animazione di lingua e pupille consente un'espressività maggiore al personaggio e quindi maggiore qualità al risultato finale.

12.2.2 Processo di cattura dell'animazione facciale

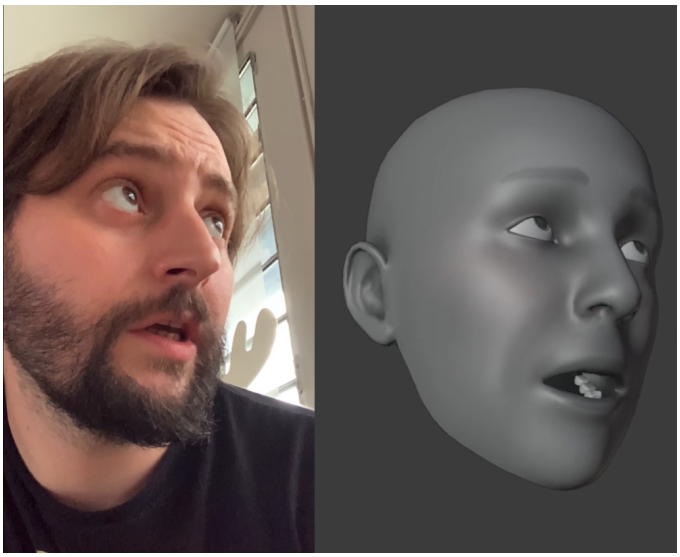


Figura 12.2: Cattura dei dati utilizzando Face Cap.

Come introdotto ad inizio capitolo, la fase di cattura dei dati relativi all'animazione facciale è quella che maggiormente ha subito limitazioni dovute alle restrizioni dettate durante la pandemia da SARS-CoV-2. Per quanto riguarda i dati utilizzati per i test descritti all'interno di questo capitolo, tutte le registrazioni e le take di mocap facciale sono state realizzate da Robin Studio dopo di che sono state consegnate a me per le fasi successive.

Siccome i dati catturati non rispecchiano fedelmente quanto fatto dai miei colleghi negli anni precedenti e contestualmente si è verificato l'avanzamento tecnologico sia dei dispositivi utilizzati sia dei sistemi di cattura del moto, diventa difficile effettuare un paragone diretto tra quanto realizzato prima e dopo l'introduzione del processo di lavoro descritto in questa tesi. Tuttavia rimane possibile fare alcune considerazioni.

Entrambe le applicazioni utilizzate, Face Cap e Face Capture (vedi capitolo 9), risultano valide per il nostro scopo. Dai test effettuati risulta che Face Cap sia in parte più precisa nella cattura dell'animazione ma tali piccole differenze potrebbero essere dovute al contesto circostante (illuminazione, attore, inquadratura e complessità dell'animazione). Questo apre la possibilità di effettuare studi più approfonditi nelle ricerche future.

Entrambe le applicazioni utilizzate, però, possono introdurre errori o imprecisioni nei dati catturati, pertanto difficilmente l'animazione acquisita potrà essere applicata direttamente sul personaggio ma sarà richiesto un lavoro di controllo e gestione finale dell'animazione.

12.2.3 Retargeting dei dati dell'animazione facciale

Relativamente alla corrispondenza tra l'animazione catturata con i sistemi di motion capture markerless e quella trasferita sul personaggio, ne abbiamo già discusso nella sottosezione 12.2.2 e verrà approfondita nella sottosezione 12.2.4. Di seguito ci limiteremo a descrivere la fase di retargeting evidenziando le principali differenze tra il "vecchio" metodo e il nuovo workflow che prevede l'utilizzo dell'add-on *ReveRig*.

Le motivazioni che hanno portato alla nascita dell'Add-On ReveRig sono ampiamente descritte nel capitolo 11. Tuttavia, è bene ricordare che una delle principali funzionalità introdotte grazie a ReveRig è proprio la possibilità di effettuare il retargeting dell'animazione facciale in modo semplice e veloce. Come descritto dettagliatamente nel capitolo 11.4 l'operazione, grazie all'add-on, è completamente automatizzata e il suo completamento può richiedere da qualche secondo fino a un paio di minuti in base al numero di keyframe da trasferire e all'hardware utilizzato. Inoltre, all'interno del processo di lavoro, è stata introdotta la possibilità di gestire anche l'animazione della lingua, offrendo così all'animatore maggiore spazio di manovra.

ReveRig, grazie alla creazione del rig facciale, ha introdotto la possibilità di trasferire i dati dell'animazione sul rig e non direttamente sulle blendshape del personaggio. Questo, come discusso nel capitolo 11, consente da un lato di trasferire l'animazione sul proprio personaggio anche laddove quest'ultimo sia stato inserito nella scena tramite Link (condizione fondamentale per la produzione di *Reverie Dawnfall*); dall'altro lato offre anche la possibilità di utilizzare un'interfaccia intuitiva per gestire e controllare il risultato finale dell'animazione. Ricordiamo che l'introduzione del rig facciale è nata principalmente dall'esigenza da parte di Robin Studio di poter applicare l'animazione facciale anche a personaggi inseriti nella scena tramite Link.

Un'altra funzionalità, aggiunta dopo aver iniziato a distribuire online l'add-on (vedi capitolo 11.6), riguarda la possibilità di tradurre in modo automatico e istantaneo i nomi delle blendshape del modello creato con Face Cap sul rig del nostro personaggio. Prima di questa funzionalità, come vedremo meglio in seguito, era necessario rinominare manualmente tutti i nomi delle 52 blendshape presenti nel modello di Face Cap. Tale operazione doveva essere ripetuta per ogni take catturata, rendendo l'intero processo lungo e macchinoso.

Test ed analisi dei dati

Il confronto con il "vecchio" metodo adottato non può essere diretto in quanto per la realizzazione del trailer, non essendo le animazioni facciali preponderanti, i colleghi non hanno sviluppato un vero e proprio workflow, inoltre per risolvere il limite dettato dall'utilizzo di una gestione dei file basata sul linking, avevano trovato un sistema rudimentale per trasferire l'animazione senza però poterla modificare. Pertanto, il confronto avverrà maggiormente basandosi su quello che è il processo di lavoro "standard" per l'animazione facciale in Blender. Ovvero, per il processo di confronto si ipotizza di non utilizzare un organizzazione basata sull'utilizzo dei link, quindi l'animazione può essere automaticamente trasferita sulle blendshape del personaggio. In questo modo è possibile confrontare le tempistiche richieste dai diversi processi analizzati, riferendosi unicamente alla fase di retargeting e a quelle strettamente collegate ad essa. Inoltre, siccome la possibilità di animare lingua e pupille è stata introdotta con ReveRig e non era prevista precedentemente, i test effettuati si baseranno unicamente sulle 52 blednshape definite da ARKit, non tenendo quindi in considerazione tutto ciò che riguarda l'animazione di lingua e pupille.

Di seguito si riportano le informazioni principali che caratterizzano i test effettuati:

- Utilizzo di un personaggio 3D dotato di una mesh per il volto (denti, lingua, ciglia e sopracciglia incluse) e di una mesh per gli occhi. La prima dotata delle 52 shape key definite da ARKit, la seconda dotata delle sole 8 shape key definite da ARKit utili al movimento degli occhi (vedi appendice C);
- Utilizzo di una take di mocap facciale catturata con l'applicazione Face Cap di durata 18 secondi, con un totale di 434 frame per ciascuna delle 52 blendshape (22568 keyframe totali). Su Blender si è utilizzata una timeline di lavoro a 24 fps;
- Tutti i test sono stati effettuati sul medesimo PC, in questo modo si è cercato di eliminare dalle variabili, per quanto possibile, la qualità dell'hardware utilizzato;
- Per tutti e tre i processi di lavoro descritti di seguito, sono stati effettuati almeno cinque test separati per ciascuna delle fasi previste e cinque test consecutivi per l'intero processo. I dati presentati in tabella 12.1 riportano la media approssimata delle tempistiche registrate.

Per confrontare i dati sono stati presi in considerazione tre processi di lavoro differenti. Due processi di lavoro utilizzati come base per il confronto; il primo ipotizzando l'utilizzo dei dati catturati con l'applicazione Face Cap e il secondo ipotizzando l'utilizzo di dati catturati con l'applicazione Face Capture di Rokoko. Infine, il processo di lavoro relativo all'utilizzo dell'add-on ReveRig, introdotto grazie a questo lavoro di tesi. Di seguito se ne descrive brevemente la struttura.

1. Processo di lavoro, basato sull'applicazione Face Cap di Bannaflak, utilizzato come confronto:

    - Fase 1: ridenominazione delle blendshape presenti nel modello generato da Face Cap. Le blendshape sono in totale 52 ma solo 36 di esse necessitano di essere rinominate. (es. con Face Cap si ha: eyeLookIn_L e deve essere rinominato in: eyeLookInLeft);
    - Fase 2: nel caso in cui le blendshape del nostro personaggio fossero prive di keyframe, è necessario inserirne almeno uno per ciascuna shape key manualmente.
    - Fase 3: trasferimento dell'animazione dal modello catturato con l'applicazione Face Cap sulle blendshape del nostro personaggio (operazione copia/incolla).

2. Il processo di lavoro, basato sull'applicazione Face Capture di Rokoko, utilizzato come confronto è composto unicamente dalla Fase 2 e dalle Fase 3 del processo sopra descritto, in quanto il modello generato presenta già le blendshape nominate secondo quanto previsto da ARKit.

3. Processo di lavoro basato sull'utilizzo dell'add-on ReveRig. Per i test si è utilizzata l'ultima versione dell'Add-On, quindi dotata della funzionalità per rinominare automaticamente le blendshape create dall'applicazione Face Cap direttamente all'interno della fase di retargeting:

    - Fase 1: selezione dell'oggetto (mesh) da utilizzare come sorgente e dell'oggetto (armatura) da utilizzare come target;
    - Fase 2: avvio della fase di retargeting cliccando sull'apposito bottone offerto dall'interfaccia del pannello *Animation Retargeting* di ReveRig.

Tabella 12.1: Tempistiche medie relative ad una sola operazione di retargeting, ottenute per ciascun processo.

Fase	Processo 1 (Face Cap)	Processo 2 (Face Capture)	Processo 3 (ReveRig)
Fase 1	∼1’ 48”	/	∼4”
Fase 2	∼38”	∼38”	∼18”
Fase 3	∼14”	∼14”	/
Totale	∼2’ 40”	∼52”	∼22”

Dai dati riportati in tabella 12.1, risulta subito evidente l’impatto che la fase di ridenominazione di tutte le blendshape ha sull’intero processo di retargeting basato sull’applicazione Face Cap. Infatti, il processo che prevede l’utilizzo di Face Cap richiede molto più tempo rispetto a quanto previsto dall’utilizzo di ReveRig e dal processo basato sull’applicazione Face Capture di Rokoko. Inoltre, si può notare come l’operazione di trasferimento dei dati relativi all’animazione facciale richieda tempi differenti tra i vecchi processi (Fase 3) e quello basato su ReveRig (Fase 2). In particolare, per la take di mocap utilizzata come base per i test, si può notare che ReveRig impiega più secondi per portare a termine l’operazione (18 secondi totali con 4 secondi di differenza con gli altri due processi testati). Da questi dati emerge che l’algoritmo implementato all’interno di ReveRig, sebbene offra ottime prestazioni, potrebbe non essere quello ottimale. Pertanto questo campo potrebbe essere oggetto di future ricerche e di possibili sviluppi, in modo da ridurre maggiormente il tempo richiesto per l’operazione di retargeting.

Detto questo, dai dati relativi ad un unico trasferimento dell’animazione diventa difficile comprendere la portata più ad ampio raggio. Bisogna precisare che da ora in poi verranno fatte ipotesi, assunzioni e approssimazioni che potrebbero influire in modo incisivo sui dati conclusivi. Sebbene tutte le ipotesi proposte siano dettate da basi motivate e da studi approfonditi, potrebbero comunque essere soggette ad errore. Pertanto si sconsiglia al lettore di interpretare i dati proposti come definitivi ed immutabili. Lo scopo perseguito, invece, è quello di fornire un’indicazione generale dei risultati ottenibili, per la sola fase di retargeting, con l’utilizzo di ReveRig all’interno dalla produzione della serie animata *Reveire Dawnfall*.

Analizzando la sceneggiatura dell’episodio pilota della serie animata (S01 E00), possono essere estratti i seguenti dati:

- Durata prevista: 20 minuti
- Numero di scene: 12
- Numero di personaggi principali: 8
- Numero di personaggi secondari o comparse: almeno 6
- Numero totale di personaggi per i quali si prevede l’applicazione del nuovo processo di animazione facciale proposto: 14 (o più)
- Numero stimato totale di take di dati acquisiti con sistemi di motion capture: 165
- Numero di volte per la quale è previsto inserire almeno un keyframe iniziale per ogni shape key delle mesh del personaggio: 48

Utilizzando i dati presentati in tabella 12.1 come tempistiche per una singola take di dati si ottiene quanto riportato in tabella 12.2. Dai dati emerge che il rapporto tra il tempo necessario per terminare le operazioni di retargeting con l’ausilio di ReveRig e quello necessario utilizzando un approccio standard basato sull’utilizzo dell’applicazione Face Cap è di circa 1:6 in favore di ReveRig. Inoltre, è possibile notare che i dati ottenuti dai test effettuati con il processo basato

sull'utilizzo dell'applicazione Face Capture sono molto simili a quelli ottenuti con ReveRig, pertanto, nel proseguimento faremo unicamente riferimento ai dati relativi al processo 1 (Face Cap) e al processo 3 (ReveRig).

Tabella 12.2: Tempistiche, relative alle operazioni di retargeting, richieste per la produzione dell'episodio pilota di *Reverie Dawnfall*

Fase	Processo 1 (Face Cap)	Processo 2 (Face Capture)	Processo 3 (ReveRig)
Fase 1	4h 57'	/	11'
Fase 2	30' 24"	30' 24"	49' 30"
Fase 3	38' 30"	38' 30"	/
Totale	6h 5' 54"	1h 8' 54"	1h 0' 30"

I dati riportati nella tabella 12.2 fanno riferimento al tempo stimato per la fase di retargeting per il solo episodio pilota della serie animata. Tuttavia, è possibile spingerci oltre e tentare di dare una stima di quanto si potrebbe risparmiare con l'utilizzo di ReveRig in confronto con un processo "standard" basato sull'utilizzo di Face Cap. Come prima, anche in questo caso le ipotesi e le approssimazioni effettuate sono molte e decisamente influenti sul risultato finale, tuttavia si ritiene siano utili per poter fornire una prospettiva generale sull'intera realizzazione della serie animata. Si è comunque consapevoli che le ipotesi effettuate potrebbero rivelarsi imprecise e che le condizioni di partenza potrebbero essere soggette a modifiche nel futuro.

Ipotizzando che i dati raccolti dall'analisi della sceneggiatura dell'episodio pilota della serie, siano applicabili anche a tutti i successivi episodi, allora è possibile fare delle assunzioni su quanto l'utilizzo dell'operazione di retargeting offerta da ReveRig potrebbe avere ripercussioni sull'intera produzione di *Reverie Dawnfall*.

I dati mostrati nella tabella 12.3 sono ottenuti partendo dalle seguenti ipotesi e condizioni di partenza:

- La serie animata *Reverie Dawnfall* è composta da 3 stagioni da 10 episodi ciascuna (30 episodi totali);
- Ogni episodio ha la durata di 20 minuti circa (10 ore di prodotto totali);
- Si ipotizza di poter applicare a ciascuno dei 30 episodi della serie le informazioni estratte dall'episodio pilota (vedi pagina 172).
- Come confronto per il processo che prevede l'utilizzo di ReveRig, si ipotizza che tutte le take di animazione facciale siano catturate con l'applicazione Face Cap (l'opzione che in tabella 12.2 ha comportato la tempistica più alta).
- Si ipotizza che al processo di animazione facciale ci lavori un unico animatore con 8 ore lavorative giornaliere, 5 giorni a settimana.

Tabella 12.3: Tempistiche, relative alle operazioni di retargeting, previste per l'intera produzione di *Reverie Dawnfall*

Unità di misura	Processo 1(Face Cap)	Processo 3 (ReveRig)
Ore totali	182h 57'	30h 15'
Giorni lavorativi	22d 6h 57'	3d 6h 15'
Settimane lavorative	~4sett 3d	~4d

Con tutti i limiti sopra specificati, il risultato ottenuto e presentato in tabella 12.3, mostra che grazie a ReveRig, per l'intera produzione della serie animata *Reverie Dawnfall*, sia possibile

risparmiare una significativa quantità di lavoro, circa un mese di lavoro svolto da un singolo animatore; riducendo in modo evidente il lavoro ripetitivo richiesto dalle operazioni di retargeting in favore del più semplice utilizzo dell'interfaccia di ReveRig.

Per contestualizzare meglio il dato ottenuto, è bene notare che i calcoli sono stati effettuati basandosi sull'intera produzione della serie animata; ovvero 3 stagioni da 10 episodi di 20 minuti ciascuno, per un totale di circa dieci ore di prodotto. Da queste informazioni è comprensibile ipotizzare che l'intero processo di produzione possa richiedere almeno un anno di lavoro per stagione. All'interno di questo contesto, una differenza di circa un mese di lavoro, unicamente per portare a termine tutte le operazioni di retargeting, rimane comunque un dato che incide in modo significativo sulle risorse impiegate per la produzione della serie animata.

Nel caso, invece, all'interno della produzione si facesse unicamente utilizzo del sistema di mocap facciale di Rokoko, allora il vantaggio in termini di tempistiche offerto da ReveRig per le operazioni di retargeting, potrebbe considerarsi non significativo (circa un giorno di differenza). Tuttavia ReveRig, a parità di tempo impiegato, consente di ridurre significativamente le azioni richieste all'animatore, delegando la maggior parte dei compiti all'add-on; eliminando quasi del tutto quelle operazioni ripetitive che possono causare frustrazione all'utente.

12.2.4 Controllo e qualità complessiva dell'animazione facciale

Come discusso nella sottosezione 12.2.1 e come anticipato in figura 12.1, già con la sola introduzione all'interno del nuovo workflow dell'add-on Faceit si è riusciti, oltre a ridurre significativamente i tempi richiesti per creare le blendshape, a garantire una qualità di partenza decisamente migliore rispetto a quanto fatto precedentemente.

Tuttavia, difficilmente ci si può limitare unicamente a trasferire i dati catturati sul nostro personaggio, perché l'animazione catturata con i sistemi di mocap facciale non sempre corrisponde a quanto desiderato e a volte può essere soggetta ad errori o ad imprecisioni. Diventa quindi indispensabile avere una serie di strumenti per poter aggiustare e correggere l'animazione in modo da ottenere quanto desiderato.

Inoltre, i dati catturati si basano su un volto umano "reale", con tutto ciò che questo comporta, ovvero proporzioni "normali" e movimenti facciali "naturali". Tuttavia, avendo noi come obiettivo quello di realizzare un prodotto di animazione stilizzato, applicare direttamente questi dati sui nostri personaggi senza apportare alcuna modifica potrebbe creare uno strano effetto e più precisamente potrebbe far cadere il prodotto all'interno dell'Uncanny Valley (vedi capitolo 2.2). Per evitare questa situazione, come descritto ampiamente nel capitolo 11.5, all'interno dell'add-on è stata introdotta un interfaccia apposita per poter controllare in modo semplice ed intuitivo l'ampiezza di ciascuna blendshape. In questo modo diventa possibile enfatizzare i movimenti e le espressioni facciali del personaggio ottenendo un risultato più adatto ad un prodotto 3D stilizzato.

La gestione di ogni blendshape spetta all'animatore, e sebbene si possa fare riferimento a quanto descritto nel capitolo 5.1.1, diventa difficile definire un approccio standard da seguire; sarà l'animatore, in collaborazione con chi di dovere (tipicamente il regista), a definire di volta in volta come gestire la singola animazione. Tale operazione è fortemente soggetta a scelte stilistiche ed artistiche. Tuttavia, tramite quanto realizzato con i nostri test e dall'apprezzamento ottenuto dagli utenti che utilizzano ReveRig, si può concludere che la funzionalità di amplificazione delle espressioni facciali introdotta grazie a ReveRig è un ottimo valore aggiunto al processo di

animazione e garantisce all'animatore maggiore spazio di manovra aprendo a nuove soluzioni artistiche.

La possibilità di poter gestire anche l'animazione della lingua e delle pupille ha introdotto un nuovo livello di espressività, aumentando la qualità complessiva del prodotto.

Per quanto riguarda l'animazione della lingua, introdotta grazie al nuovo workflow, è necessario fare una precisazione. L'impatto che il movimento della lingua ha sul risultato finale può variare molto da una scena e l'altra e tra un'inquadratura e l'altra. Come discusso all'interno del capitolo 10, la realizzazione dell'animazione della lingua ha maggiormente senso in inquadrature strette sul volto del personaggio e dove per il personaggio stesso è previsto un dialogo importante. Infatti, in situazioni in cui il personaggio si trova distante dalla camera o dove il dialogo risulta limitato, allora procedere con l'animazione della lingua potrebbe rivelarsi un investimento poco saggio delle risorse a disposizione. Tuttavia, quando presente, il movimento della lingua anche se non immediatamente percepito dall'utente, aiuta a dare maggiore organicità all'animazione rendendo la scena più credibile e godibile dal pubblico.

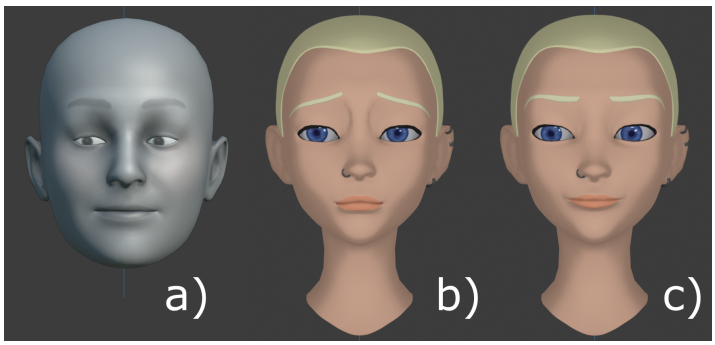
Per ovvie ragioni in figura 12.3 ci si limita a riportare alcuni fotogrammi delle sequenze realizzate in modo da fornire un'indicazione visiva di quanto ottenuto. Si è consapevoli che l'unico modo per apprezzare completamente quanto realizzato è analizzando le clip video prodotte. Data l'enorme varietà e l'imprevedibilità delle animazioni facciali che possono essere realizzate, il processo che porterà al risultato finale potrà essere più o meno lungo a seconda delle necessità. In figura 12.3 vengono proposte tre differenti situazioni: in figura 12.3a il risultato ottenuto unicamente grazie all'utilizzo di Faceit può rivelarsi sufficiente; in figura 12.3b invece si è deciso di procedere modificando l'ampiezza di alcune blendshape in modo da dare maggior espressività, inoltre la situazione richiedeva l'introduzione dell'animazione della lingua; infine in figura 12.3c viene mostrato l'ottenimento di un risultato per niente soddisfacente dopo l'operazione di retargeting, pertanto si è proceduti a gestire l'animazione grazie agli strumenti offerti da ReveRig e al rig facciale di tipo Bone-Based.

Complessivamente l'introduzione, grazie a ReveRig, di diversi tipi di interfaccia, il rig facciale e l'interfaccia window-based, consente all'animatore di gestire in modo pratico e veloce tutte le 62 blendshape del personaggio previste per l'animazione facciale. In questo modo si eliminano o si riducono significativamente le volte in cui sarà necessario agire sui singoli keyframe all'interno del Graph Editor, offrendo così un'esperienza di utilizzo migliore.

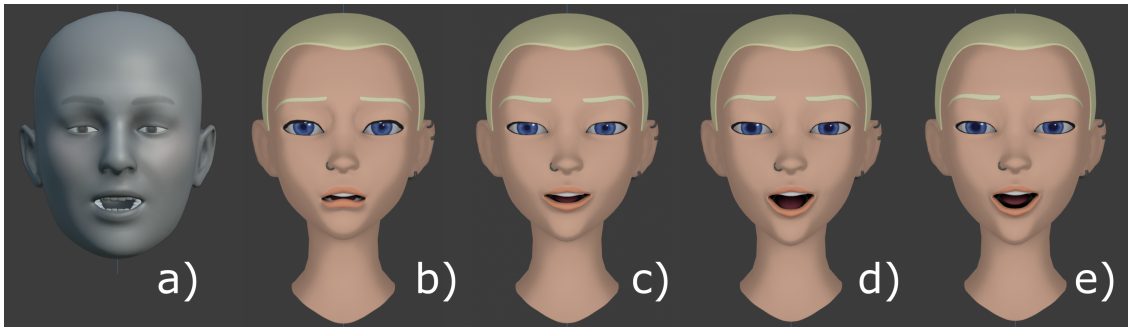
A volte potrebbe essere necessario o comunque potrebbe portare ad un risultato migliore, in base alle esigenze di produzione ed artistiche, integrare all'utilizzo di ReveRig anche il rig facciale di tipo viewport 3D creato insieme all'armatura del corpo del personaggio, come indicato in [33] (vedi figura 12.3c). L'integrazione tra utilizzo di interfacce *viewport 2D* e *window-based* per gestire le 62 blendshape e l'utilizzo di un interfaccia *vieport 3D* per gestire direttamente i vertici della mesh tramite tecniche Bone-Based e Free Form Deformation può consentire all'animatore di ottenere un risultato di ottima qualità e di soddisfare le richieste artistiche del prodotto.

Ovviamente, la qualità finale, come visto nel capitolo 6, è correlata quasi direttamente con la quantità di risorse disponibili in termini di tempo, budget e competenze. Tipicamente ad un aumentare di queste ultime si è in grado di ottenere un prodotto di maggiore qualità.

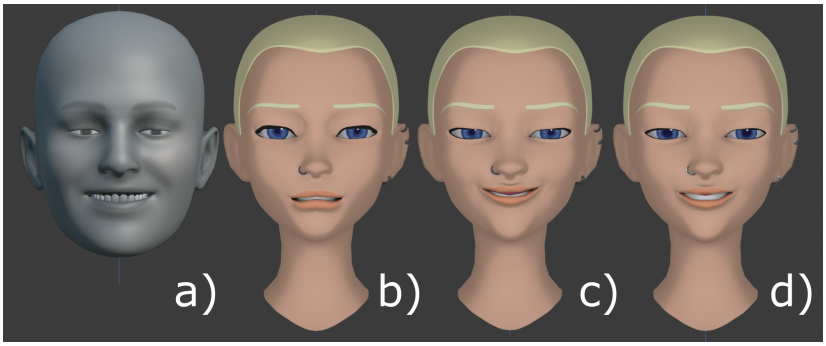
Grazie all'introduzione del nuovo workflow è stato possibile ridurre significativamente il tempo necessario per compiere le diverse fasi precedenti, offrendo quindi la possibilità di dedicare maggior risorse al controllo finale dell'animazione.



(a) Confronto tra a) animazione catturata con Face Cap, b) risultato ottenuto con il vecchio workflow e c) risultato ottenuto unicamente grazie all'utilizzo di Faceit.



(b) Confronto tra a) animazione catturata con Face Cap, b) risultato ottenuto con il vecchio workflow, c) risultato ottenuto unicamente grazie all'utilizzo di Faceit, d) risultato ottenuto enfatizzando alcune blendshape ed e) introduzione dell'animazione della lingua.



(c) Confronto tra a) animazione catturata con Face Cap, b) risultato ottenuto con il vecchio workflow, c) risultato ottenuto unicamente grazie all'utilizzo di Faceit (non soddisfacente) e d) risultato ottenuto utilizzando ReveRig e il rig facciale di tipo Bone-Based.

Figura 12.3: Esempi di confronto tra risultati ottenuti con il vecchio metodo di lavoro e quelli ottenuti nelle diverse fasi del nuovo workflow.

Capitolo 13

Conclusioni

Il presente studio si è posto come obiettivo quello di individuare tecniche e strumenti già esistenti e di creare strumenti appositi da poter integrare all'interno dalla già avviata produzione indipendente della serie animata *Reverie Dawnfall* per la realizzazione delle animazioni facciali, mantenendo al contempo uno sguardo costante al mercato indipendente internazionale.

Suddividendo il lavoro di animazione facciale in tre fasi principali: manipolazione della geometria, acquisizione dei dati ed infine applicazione dei dati; è stato possibile definire un nuovo processo di lavoro in grado di ridurre significativamente le risorse richieste, soprattutto in termini di tempo, e contemporaneamente di garantire una maggiore qualità del prodotto finale.

Il processo di lavoro proposto si basa principalmente sull'utilizzo di blendshape per deformare la geometria, sull'utilizzo della tecnologia di motion capture facciale markerless chiamata ARKit e offerta da Apple per l'acquisizione dei dati ed infine prevede l'introduzione dell'Add-On *ReveRig* per gestire la fase di applicazione dei dati e la gestione complessiva del processo di animazione.

All'interno del nuovo processo di lavoro, inoltre, è stato trovato spazio per inserire la possibilità di animare in modo semplice e veloce anche il movimento della lingua, dando così maggior libertà espressiva agli animatori.

Con l'obiettivo di soddisfare i bisogni presentati da Robin Studio, è stato ideato e sviluppato l'Add-On di Blender *ReveRig*; presto lo strumento ReveRig è diventato centrale all'interno di questo progetto. Grazie alla decisione iniziale di mantenere uno sguardo rivolto verso l'intero settore dell'animazione facciale indipendente, è stato possibile riconoscere le potenzialità di ReveRig e cogliere l'opportunità di una distribuzione a livello internazionale e potenzialmente globale. Dai risultati mostrati nei capitoli 11.6 e 12 emerge che ReveRig è in grado di soddisfare sia le esigenze richieste da Robin Studio che i bisogni presenti all'interno della community di Blender, dando in questo modo un valore aggiunto al lavoro svolto.

Sebbene, molte delle tecniche studiate e descritte nella parte I del documento ad oggi sono difficilmente implementabili all'interno di una produzione indipendente in modo sostenibile, la ricerca e l'innovazione tecnologica avanzano molto velocemente e molte tecnologie oggi irraggiungibili un domani potrebbero essere accessibili anche per il mercato consumer.

Il lavoro dettagliato presentato nella parte I del documento, oltre ad essere stato un elemento fondamentale per prendere decisioni più consapevoli all'interno di questo lavoro di tesi, potrà

essere utilizzato come base e come linea guida per tutti gli studi futuri relativi all'animazione facciale in contesti indipendenti. Potrà essere fonte di importanti spunti e idee e potrà essere utilizzato come strumento di confronto con le tecnologie che il futuro ci presenterà.

Discutendo quelli che possono essere gli sviluppi futuri a partire da questo lavoro di tesi è bene distinguere il lavoro che potrà essere svolto per la serie animata *Reverie Dawnfall* e quello per l'Add-On *ReveRig*.

Il processo di animazione introdotto grazie a questo lavoro di tesi, ha portato significativi vantaggi sia in termini di qualità del risultato finale che in termini di risparmio delle risorse impiegate. Tuttavia, il rapido avanzamento tecnologico apre a nuove possibilità che è bene inseguire. Ricerche future potrebbero concentrarsi sulla fase di cattura dell'animazione, effettuando test più approfonditi sul processo da implementare nella fase di produzione a massimo regime della serie animata *Reverie Dawnfall*; valutare la possibilità di utilizzare delle Head Mounted Camera e studiare le condizioni necessarie per ottenere la massima qualità dalle registrazioni. Un altro elemento non approfondito all'interno di questo lavoro di tesi riguarda la possibilità di aggiungere degli elementi per rappresentare le linee del volto durante l'animazione. Infatti, si ritiene che aggiungere le rughe al volto dei personaggi possa aumentare in modo significativo il potenziale espressivo del prodotto.

Per quanto riguarda gli sviluppi futuri di ReveRig, il crescente interesse verso gli Avatar Virtuali può essere una grande opportunità per l'add-on. L'introduzione, all'interno di ReveRig, di funzionalità per gestire in tempo reale l'animazione facciale potrebbe essere un'ottima soluzione per intercettare nuovo pubblico e per consolidare la presenza di ReveRig nel mercato dell'animazione facciale su Blender. Inoltre, sebbene le prestazioni offerte siano già soddisfacenti, future ricerche possono essere indirizzate con lo scopo di ottimizzare l'algoritmo che si occupa di effettuare la fase di retargeting permettendo quindi di ridurre maggiormente le risorse richieste.

La scelta di distribuire online ReveRig ha portato significativi vantaggi, ha consentito di introdurre una nuova funzionalità all'interno dell'Add-On, ha permesso di entrare in relazione più stretta con le esigenze degli utenti ed infine ha aperto nuove prospettive per i futuri sviluppi di ReveRig.

Grazie a questo lavoro di tesi ho avuto la possibilità di realizzare uno strumento utile sia per Robin Studio che potenzialmente per l'intera community di Blender. Tale trasversalità di utilizzo dell'Add-On ReveRig è motivo di estremo orgoglio e indicatore della qualità del lavoro svolto.

Appendice A

Nuovo set di fonemi/visemi

```
# This file contains a mapping between the CMU phoneme set to the phoneme set you use for animating. By default, the animation phoneme set is the Preston Blair phoneme set, as found in the book: "Cartoon Animation", by Preston Blair, page 186.
# The phonemeset is defined in the two tables below. The first table contains the basic list of phonemes to use in your animation. The second table contains a mapping from the CMU phonemes to the phonemes in the first table.
# Tongue position phonemes set: TDN TH J R KG rest

phoneme_set = [
    'TDN', #-> T D N S Z J L SH CH DH ZH
    'TH', #-> TH
    'J', #->JH
    'R', #->R ER
    'KG',  # this covers -> K G Q NG
    'rest' # not really a phoneme - this is used in-between phrases when the tongue is at rest covers->AA AE AH AO AW AY EH EY IH IY OW OY UH UW B M P HH F V W Y E21
]
# Phoneme conversion dictionary: CMU on the left to Tongue Position on the right
phoneme_conversion = {
    'AA0': 'rest',  # odd     AA D
    'AA1': 'rest',
    'AA2': 'rest',
    'AE0': 'rest',  # at   AE T
    'AE1': 'rest',
    'AE2': 'rest',
    'AH0': 'rest',  # hut  HH AH T
    'AH1': 'rest',
    'AH2': 'rest',
    'AO0': 'rest',  # ought AO T
    'AO1': 'rest',
    'AO2': 'rest',
    'AW0': 'rest',  # cow   K AW
    'AW1': 'rest',
    'AW2': 'rest',
```

```
    'AY0': 'rest',  # hide HH AY D
    'AY1': 'rest',
    'AY2': 'rest',
    'B': 'rest',  # be    B IY
    'CH': 'TDN',  # cheese   CH IY Z
    'D': 'TDN',  # dee   D IY
    'DH': 'TDN',  # thee DH IY
    'EH0': 'rest',  # Ed    EH D
    'EH1': 'rest',
    'EH2': 'rest',
    'ER0': 'R',  # hurt  HH ER T
    'ER1': 'R',
    'ER2': 'R',
    'EY0': 'rest',  # ate   EY T
    'EY1': 'rest',
    'EY2': 'rest',
    'F': 'rest',  # fee    F IY
    'G': 'KG',  # green G R IY N
    'HH': 'rest',  # he   HH IY
    'IH0': 'rest',  # it   IH T
    'IH1': 'rest',
    'IH2': 'rest',
    'IY0': 'rest',  # eat   IY T
    'IY1': 'rest',
    'IY2': 'rest',
    'JH': 'J',  # gee  JH IY
    'K': 'KG',  # key   K IY
    'L': 'TDN',  # lee L IY
    'M': 'rest',  # me    M IY
    'N': 'TDN',  # knee  N IY
    'NG': 'KG',  # ping P IH NG
    'OW0': 'rest',  # oat   OW T
    'OW1': 'rest',
    'OW2': 'rest',
    'OY0': 'rest',  # toy  T OY
    'OY1': 'rest',
    'OY2': 'rest',
    'P': 'rest',  # pee   P IY
    'R': 'R',  # read  R IY D
    'S': 'TDN',  # sea   S IY
    'SH': 'TDN',  # she  SH IY
    'T': 'TDN',  # tea   T IY
    'TH': 'TH',  # theta    TH EY T AH
    'UH0': 'rest',  # hood  HH UH D
    'UH1': 'rest',
    'UH2': 'rest',
    'UW0': 'rest',  # two   T UW
    'UW1': 'rest',
    'UW2': 'rest',
    'V': 'rest',  # vee    V IY
    'W': 'rest',  # we W IY
    'Y': 'rest',  # yield Y IY L D
    'Z': 'TDN',  # zee   Z IY
    'ZH': 'TDN',  # seizure  S IY ZH ER
```

```
    # The following phonemes are not part of the CMU phoneme set, but are
    meant to fix bugs in the CMU dictionary
    'E21': 'rest'  # E21 is used in ENGINEER
}
```

Codice A.1: File `phonemes_tongue_position.py` per la mappatura tra il set di fonemi definito nel CMUdict e il set di fonemi `tongue_position` realizzato appositamente per il lavoro di tesi.

```
# This file contains a the list of available phoneme sets.
# Each element should have corresponding .py file with the "phoneme_" prefix
# Example:
#     for 'preston_blair' there is a 'phoneme_preston_blair.py' file

phoneme_sets = [
    'preston_blair',
    'fleming_dobbs',
    'tongue_position'
]
```

Codice A.2: File `phonemes.py` contenente la lista dei set di fonemi disponibili.

Appendice B

Esempio file animazione lingua

```
MohoSwitch1
1 rest
49 rest
50 TDN
52 rest
53 rest
54 TDN
56 rest
69 rest
70 TDN
72 rest
73 rest
74 TDN
75 rest
76 rest
77 KG
78 TDN
79 rest
79 rest
80 TDN
81 rest
82 rest
83 R
85 rest
85 rest
86 KG
88 rest
...
```

Appendice C

Blendshape e ossa previste da ReveRig

Riferimenti a p. 112 e 116.

Tabella C.1: Le 62 blendshape previste da ReveRig

Area del volto	Nomi delle blendshape
Occhi (14)	eyeBlinkLeft, eyeLookDownLeft, eyeLookInLeft, eyeLookOutLeft, eyeLookUpLeft, eyeSquintLeft, eyeWideLeft, eyeBlinkRight, eyeLookDownRight, eyeLookInRight, eyeLookOutRight, eyeLookUpRight, eyeSquintRight, eyeWideRight
Pupille (4) (Facoltative)	eyePupilDilateRight, eyePupilContractRight, eyePupilDilateLeft, eyePupilContractLeft
Mascella (4)	jawForward, jawLeft, jawRight, jawOpen
Bocca (23)	mouthFunnel, mouthPucker, mouthLeft, mouthRight, mouthRollUpper, mouthRollLower, mouthShrugUpper, mouthShrugLower, mouthClose, mouthSmileLeft, mouthSmileRight, mouthFrownLeft, mouthFrownRight, mouthDimpleLeft, mouthDimpleRight, mouthUpperUpLeft, mouthUpperUpRight, mouthLowerDownLeft, mouthLowerDownRight, mouthPressLeft, mouthPressRight, mouthStretchLeft, mouthStretchRight
Sopracciglia (5)	browOuterUpRight, browDownRight, browInnerUp, browDownLeft, browOuterUpLeft
Guance (3)	cheekPuff, cheekSquintLeft, cheekSquintRight
Naso (2)	noseSneerLeft, noseSneerRight
Lingua (7)	tongueOut, TDN, TH, J, R, KG, rest

```
#Lista contenente tutti i nomi delle ossa create per il rig facciale di
    ReveRig. ALl'interno della lista list_Bones i nomi riportati sono
    principalmente quelli delle blendshape di riferimento; a questi si
    aggiungono i nomi: "eyeLookAllRight", "eyeLookAllLeft" e "jawAll" non
    corrispondenti direttamente ad alcuna blendshape ma utilizzati per creare
     un'interfaccia più pulita e aumentarne l'usabilità.
list_Bones= [
    "browOuterUpRight","browDownRight","browInnerUp","browDownLeft","
    browOuterUpLeft","eyeLookUpRight","eyeLookDownRight","eyeLookInRight","
    eyeLookOutRight","eyeLookUpLeft","eyeLookDownLeft","eyeLookInLeft","
    eyeLookOutLeft","eyeBlinkLeft","eyeBlinkRight","eyeSquintLeft","
    eyeSquintRight","eyeWideLeft","eyeWideRight","eyePupilLeft","
    eyePupilRight","eyeLookAllRight","eyeLookAllLeft","cheekSquintRight","
    noseSneerRight","noseSneerLeft","cheekSquintLeft","cheekPuff","mouthLeft"
    ,"mouthRight","mouthRollUpper","mouthRollLower","mouthShrugUpper","
    mouthShrugLower","mouthSmileLeft","mouthSmileRight","mouthFrownLeft","
    mouthFrownRight","mouthDimpleLeft","mouthDimpleRight","mouthUpperUpLeft",
    "mouthUpperUpRight","mouthLowerDownLeft","mouthLowerDownRight","
    mouthPressLeft","mouthPressRight","mouthStretchLeft","mouthStretchRight",
    "mouthFunnel","mouthPucker","mouthClose","jawRight","jawLeft","jawOpen","
    jawForward","jawAll","tongueOut","TDN","TH","J","R","KG","rest"
]
#Ciascun nome interno alla lista list_Bones può essere elaborato come un
    singolo elemento (elem). Per la creazione del rig facciale per ciascun
    elem si sono create due ossa: "Base_"+elem e "Slider_"+elem. (es
    Base_browInnerUp e Slider_browInnerUp). Per gli elementi contenenti la
    stringa "eyeLook" si sono create ulteriori due ossa nominate come segue:
    "Base_Eyes_"+elem e "Slider_Eyes_"+elem (es Base_Eyes_eyeLookUpRight e
    Slider_Eyes_eyeLookUpRight).
```

Codice C.1: Elenco completo delle ossa create da ReveRig

Bibliografia

Bibliografia classica

[1] Frederick I Parke. «Computer generated animation of faces». In: *Proceedings of the ACM annual conference-Volume 1*. 1972, pp. 451–457 (cit. alle pp. 9, 34).

[2] Frederic I. Parke e Keith Waters. *Computer Facial Animation*. Second. 888 Worcester Street, Suite 230 Wellesley, MA 02482: AK Peters Ltd, 2008. ISBN: 1568814488 (cit. alle pp. 16, 27, 30, 37, 38, 49, 50).

[3] P. L. Williams et al. *Gray's anatomy*. 37th. Edinburgh: Churchill Livingstone: BJS (British Journal of Surgery), 1989 (cit. a p. 16).

[4] Daniel C. Krawczyk. «Chapter 12 - Social Cognition: Reasoning With Others». In: *Reasoning*. A cura di Daniel C. Krawczyk. Academic Press, 2018, pp. 283–311. ISBN: 978-0-12-809285-9. DOI: https://doi.org/10.1016/B978-0-12-809285-9.00012-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128092859000120 (cit. a p. 22).

[5] Masahiro Mori. «The Uncanny Valley». In: *Energy* 7.4 (1970), pp. 33–35 (cit. a p. 22).

[8] O. Johnson F. Thomas. *Disney Animation: The Illusion of Life*. New York, NY, USA: Abbeville Press, 1981 (cit. a p. 28).

[9] John Lasseter. «Principles of Traditional Animation Applied to 3D Computer Animation». In: SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 35–44. ISBN: 0897912276. DOI: 10.1145/37401.37407. URL: https://doi.org/10.1145/37401.37407 (cit. a p. 28).

[10] Michele Cannata. «Studio del Workflow per la creazione di Personaggi 3D per la Produzione Indipendente di un Prodotto di Animazione». Tesi di laurea mag. Politecnico di Torino, apr. 2018. URL: http://webthesis.biblio.polito.it/id/eprint/7589 (cit. alle pp. 30, 66).

[11] Noor Mat Noor et al. «Review on 3D Facial Animation Techniques». In: *International Journal of Engineering and Technology* 7 (dic. 2018), pp. 181–187. DOI: 10.14419/ijet.v7i4.44.26980 (cit. a p. 32).

[12] Verónica Orvalho et al. «A Facial Rigging Survey». In: *Eurographics 2012 - State of the Art Reports*. A cura di Marie-Paule Cani e Fabio Ganovelli. The Eurographics Association, 2012. DOI: 10.2312/conf/EG2012/stars/183-204 (cit. alle pp. 33, 37, 41, 42).

[13] J. P. Lewis et al. «Practice and Theory of Blendshape Facial Models». In: *Eurographics 2014 - State of the Art Reports*. A cura di Sylvain Lefebvre e Michela Spagnuolo. The Eurographics Association, 2014. DOI: 10.2312/egst.20141042 (cit. alle pp. 34, 35, 43, 58, 87, 167).

[14] B Raitt. «The making of Gollum». In: *Presentation at U. Southern California Institute for Creative Technologies's Frontiers of Facial Animation Workshop, August*. 2004 (cit. a p. 35).

[15] J.P. Lewis et al. «A User Interface Technique for Controlling Blendshape Interference». In: *Data-Driven 3D Facial Animation*. A cura di Zhigang Deng e Ulrich Neumann. London: Springer London, 2008, pp. 132–144. ISBN: 978-1-84628-907-1. DOI: 10.1007/978-1-84628-907-1_7. URL: https://doi.org/10.1007/978-1-84628-907-1_7 (cit. alle pp. 35, 36).

[16] Zhigang Deng e Junyong Noh. «Computer Facial Animation: A Survey». In: *Data-Driven 3D Facial Animation*. A cura di Zhigang Deng e Ulrich Neumann. London: Springer London, 2008, pp. 1–28. ISBN: 978-1-84628-907-1. DOI: 10.1007/978-1-84628-907-1_1. URL: https://doi.org/10.1007/978-1-84628-907-1_1 (cit. alle pp. 38, 40).

[17] S.M. Platt. «A structural model of the human face». Tesi di dott. Philadelphia, PA, USA: University of Pennsylvania, 1985 (cit. a p. 38).

[18] Keith Waters. «A Muscle Model for Animation Three-Dimensional Facial Expression». In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 17–24. ISBN: 0897912276. DOI: 10.1145/37401.37405. URL: https://doi.org/10.1145/37401.37405 (cit. a p. 38).

[19] Demetri Terzopoulos e Keith Waters. «Physically-based facial modelling, analysis, and animation». In: *The Journal of Visualization and Computer Animation* 1.2 (1990), pp. 73–80. DOI: https://doi.org/10.1002/vis.4340010208. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/vis.4340010208. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/vis.4340010208 (cit. a p. 38).

[20] Jun-yong Noh. «A Survey of Facial Modeling and Animation Techniques». In: 2001 (cit. a p. 39).

[21] Prem Kalra e Nadia Magnenat-Thalmann. «Modeling of vascular expressions in facial animation». In: *Proceedings of Computer Animation'94*. IEEE. 1994, pp. 50–58 (cit. a p. 39).

[22] «4 - Performance and Motion Capture». In: *The VES Handbook of Visual Effects*. A cura di Jeffrey A. Okun e Susan Zwerman. Boston: Focal Press, 2010, pp. 335–386. ISBN: 978-0-240-81242-7. DOI: https://doi.org/10.1016/B978-0-240-81242-7.00004-1. URL: https://www.sciencedirect.com/science/article/pii/B9780240812427000041 (cit. alle pp. 41, 51–53, 55, 57, 58).

[27] Tero Karras et al. «Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion». In: *ACM Trans. Graph.* 36.4 (lug. 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073658. URL: https://doi.org/10.1145/3072959.3073658 (cit. alle pp. 49–51).

[28] Roy Paul Madsen. *Animated film: concepts, methods, uses*. Interland Pub, 1969 (cit. a p. 49).

[30] Michael M. Cohen e Dominic W. Massaro. «Modeling Coarticulation in Synthetic Visual Speech». In: *Models and Techniques in Computer Animation*. A cura di Nadia Magnenat Thalmann e Daniel Thalmann. Tokyo: Springer Japan, 1993, pp. 139–156. ISBN: 978-4-431-66911-1 (cit. a p. 50).

[31] Daniel Cudeiro et al. «Capture, Learning, and Synthesis of 3D Speaking Styles». In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10101–10111 (cit. a p. 50).

[32] Melissa Coarezza. «Studio del Workflow per l'uso di Motion Capture nella Produzione Indipendente di un Prodotto di Animazione». Tesi di laurea mag. Politecnico di Torino, dic. 2017. URL: http://webthesis.biblio.polito.it/id/eprint/6618 (cit. alle pp. 52, 66).

[33] Stefania Abate. «Rigging di un Personaggio Orientato al Motion Capture per un Prodotto di Animazione Indipendente». Tesi di laurea mag. Politecnico di Torino, dic. 2019. URL: http://webthesis.biblio.polito.it/id/eprint/13104 (cit. alle pp. 52, 66, 87, 89, 108, 166–168, 175).

[35] Timothy F Cootes, Gareth J Edwards e Christopher J Taylor. «Active appearance models». In: *European conference on computer vision*. Springer. 1998, pp. 484–498 (cit. a p. 56).

[36] Pedro F. Felzenszwalb e Daniel P. Huttenlocher. «Pictorial Structures for Object Recognition». In: *International Journal of Computer Vision* 61 (2003), pp. 55–79 (cit. a p. 56).

[37] Sofien Bouaziz. «Realtime Face Tracking and Animation». Tesi di dott. École Polytechnique Fédérale de Lausanne (EPFL), 2015 (cit. alle pp. 58, 81).

[44] Ilaria Ginepro. «Modellazione di ambienti 3D per la realizzazione di un prodotto indipendente di animazione: Reverie Dawnfall». Tesi di laurea mag. Politecnico di Torino, dic. 2019. URL: http://webthesis.biblio.polito.it/id/eprint/13118 (cit. a p. 66).

[45] Giacomo Davide Massimo Balma e Andrea Lorusso. «Illuminazione e shading procedurale non fotorealistico in un prodotto di animazione indipendente». Tesi di laurea mag. Politecnico di Torino, dic. 2019. URL: http://webthesis.biblio.polito.it/id/eprint/13106 (cit. a p. 66).

[46] Nicola Figari Barberis. «Cloth and hair simulation workflow analysis for an independent animation product». Tesi di laurea mag. Politecnico di Torino, 2020. URL: http://webthesis.biblio.polito.it/id/eprint/14418 (cit. a p. 66).

[66] Johan Verwey e Edwin Blake. «The Influence of Lip Animation on the Perception of Speech in Virtual Environments». In: gen. 2005 (cit. a p. 98).

Sitografia

[6] Norri Kageki. *An Uncanny Mind: Masahiro Mori on the Uncanny Valley and Beyond. An interview with the Japanese professor who came up with the uncanny valley of robotics.* Ultimo accesso il 22/02/2021. IEEE Spectrum. 12 Giu. 2012. URL: https://spectrum.ieee.org/automaton/robotics/humanoids/an-uncanny-mind-masahiro-mori-on-the-uncanny-valley (cit. a p. 22).

[7] Paul Ekman Group LLC, cur. *Facial Action Coding System.* Ultimo accesso il 22/02/2021. 2021. URL: https://www.paulekman.com/facial-action-coding-system/ (cit. a p. 27).

[23] Victor Navone. *Facial animation for feature animated films. Animating stylized facial expressions with Victor Navpne.* The Gnomon Workshop. 2020. URL: https://www.thegnomonworkshop.com/tutorials/facial-animation-for-feature-animated-films (cit. a p. 44).

[24] Visual Effects Society VES, cur. *7TH ANNUAL VES AWARDS.* Ultimo accesso il 12/02/2021. 2009. URL: https://www.visualeffectssociety.com/portfolio-items/2008-7th-annual-ves-awards/?portfolioCats=29 (cit. a p. 44).

[25] Pixar Wiki, cur. *Victor Navone.* Ultimo accesso il 12/02/2021. 2020. URL: https://pixar.fandom.com/wiki/Victor_Navone (cit. a p. 44).

[26] International Phonetic Association, cur. *International Phonetic Association.* Ultimo accesso il 12/02/2021. URL: https://www.internationalphoneticassociation.org/ (cit. a p. 49).

[29] Pier Marco Bertinetto. *fonetica.* A cura di Enciclopedia dell'Italiano. Ultimo accesso il 22/02/2021. Treccani, 2010. URL: https://www.treccani.it/enciclopedia/fonetica_(Enciclopedia-dell'Italiano)/ (cit. a p. 50).

[34] MOVA®CONTOUR, cur. *Technology : CONTOUR Reality Capture - Mova.* Ultimo accesso il 12/02/2021. URL: http://www.mova.com/technology.php (cit. a p. 55).

[38] Chris Baraniuk. *Faceshift: Apple buys Star Wars motion-capture company.* A cura di BBC. Ultimo accesso il 12/02/2021. 25 Nov. 2015. URL: https://www.bbc.com/news/technology-34920548 (cit. alle pp. 58, 81).

[39] Ingrid Lunden e Natasha Lomas. *Apple Has Acquired Faceshift, Maker Of Motion Capture Tech Used In Star Wars*. A cura di TechCrunch. Ultimo accesso il 12/02/2021. 25 Nov. 2015. URL: https://techcrunch.com/2015/11/24/apple-faceshift/?guccounter=1 (cit. alle pp. 58, 81).

[40] Box Office Mojo by IMDbPro, cur. *Toy Story 4 (2019)*. Ultimo accesso il 12/02/2021. 2020. URL: https://www.boxofficemojo.com/release/rl3798500865/ (cit. a p. 60).

[41] ANSA, cur. *Gatta Cenerentola, è miracolo napoletano*. Ultimo accesso il 12/02/2021. 5 Set. 2017. URL: https://www.ansa.it/sito/notizie/cultura/cinema/2017/09/05/gatta-cenerentola-e-miracolo-napoletano_5b08a253-1e3b-4cc2-95a6-e3f44c55de59.html (cit. a p. 60).

[42] MAD Entertainment, cur. *Gatta Cenerentola*. Ultimo accesso il 12/02/2021. URL: https://www.madentertainment.it/gatta-cenerentola/ (cit. a p. 62).

[43] Robin Studio S.r.l.s., cur. *PressbookReverieIta*. Ultimo accesso il 12/02/2021. URL: http://robin.studio/reverie/ReveriePressbook.pdf (cit. a p. 65).

[47] *GNU General Public License*. Ultimo accesso il 22/02/2021. Free Software Foundation, 2020. URL: https://www.gnu.org/licenses/gpl-3.0.html (cit. a p. 72).

[48] Blender, cur. *Blender - About*. Ultimo accesso il 12/02/2021. URL: https://www.blender.org/about/ (cit. a p. 72).

[49] Blender, cur. *Blender - History*. Ultimo accesso il 12/02/2021. URL: https://www.blender.org/foundation/history/ (cit. a p. 72).

[50] Blender Foundation, cur. *Blender 2.91.0 Python API - Quickstart*. Ultimo accesso il 12/02/2021. 2020. URL: https://docs.blender.org/api/current/info_quickstart.html (cit. a p. 74).

[51] Blender Foundation, cur. *Blender 2.91.0 Python API - Overview*. Ultimo accesso il 12/02/2021. 2020. URL: https://docs.blender.org/api/current/info_overview.html (cit. a p. 74).

[52] Blender Foundation, cur. *Blender 2.91 Manual - Shape Keys*. Ultimo accesso il 12/02/2021. 2020. URL: https://docs.blender.org/manual/en/latest/animation/shape_keys/introduction.html (cit. a p. 75).

[53] Blender Foundation, cur. *Blender 2.91 Manual - Drivers*. Ultimo accesso il 12/02/2021. 2020. URL: https://docs.blender.org/manual/en/latest/animation/drivers/introduction.html (cit. a p. 76).

[54] Blender Foundation, cur. *Blender 2.91 Manual - Link & Append*. Ultimo accesso il 12/02/2021. 2020. URL: https://docs.blender.org/manual/en/latest/files/linked_libraries/link_append.html (cit. a p. 76).

[55] MOHO, cur. *Papagayo - Lip Synching Support for Moho and Anime Studio*. Ultimo accesso il 12/02/2021. URL: https://www.mohoanimation.com/papagayo.shtml (cit. a p. 79).

[56] MOREVNA Project, cur. *Papagayo-NG*. Ultimo accesso il 12/02/2021. URL: https://morevnaproject.org/papagayo-ng/ (cit. a p. 80).

[57] *Augmented Reality ARKit*. Ultimo accesso il 12/02/2021. Apple Inc., 2021. URL: https://developer.apple.com/augmented-reality/arkit/ (cit. a p. 82).

[58] *Class ARFaceAnchor*. Ultimo accesso il 12/02/2021. Apple Inc., 2021. URL: https://developer.apple.com/documentation/arkit/arfaceanchor (cit. a p. 82).

[59] *Structure ARFaceAnchor.BlendShapeLocation*. Ultimo accesso il 12/02/2021. Apple Inc., 2021. URL: https://developer.apple.com/documentation/arkit/arfaceanchor/blendshapelocation (cit. alle pp. 82, 93).

[60] *Face Capture*. Ultimo accesso il 12/02/2021. Rokoko, 2020. URL: https://www.rokoko.com/products/face-capture (cit. a p. 85).

[61] *Face Cap. Documentation*. Ultimo accesso il 12/02/2021. Bannaflak, 2020. URL: https://www.bannaflak.com/face-cap/documentation.html (cit. a p. 86).

[62] *Animation For IPhone X*. Ultimo accesso il 12/02/2021. Poliwink, 2021. URL: https://www.polywink.com/15-78-facial-animation-for-iphone-x.html#/35-option-fbx (cit. a p. 88).

[63] *Faceit - Iphonex Shape Keys And Performance Capture*. Ultimo accesso il 12/02/2021. BlenderMarket, 2021. URL: https://blendermarket.com/products/faceit (cit. a p. 88).

[64] *Faceit Documentation*. Ultimo accesso il 12/02/2021. 2021. URL: https://faceit-doc.readthedocs.io/en/latest/ (cit. alle pp. 88, 91).

[65] *The CMU Pronouncing Dictionary*. Ultimo accesso il 12/02/2021. Carneige Mellon University (CMU). URL: http://www.speech.cs.cmu.edu/cgi-bin/cmudict (cit. a p. 98).