Università degli Studi di Roma Tor Vergata

Macroarea di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in
Scienze e Tecnologie per i Media

TESI DI LAUREA

Rigging: Come nasce il movimento di un Character 3D

Candidato:
Giulia Isaia

Relatore:
Carmine Di Fiore

Correlatore:
Andrea Felice

Anno Accademico 2017/2018

Rigging:

Come nasce il movimento di un Character 3D

Indice:

1. Introduzione:

Cinema, tv, videogiochi. Al giorno d'oggi è sempre più comune imbattersi in un prodotto di animazione digitale 3D e si finisce con il dare per scontato il movimento che compie un personaggio in Computer Grafica interpretando il suo ruolo. Se, però, ci si interroga su come nasce un personaggio e come faccia ad interpretare la sua parte e sembrare vivo empatizzando con il pubblico e ingannando l'occhio umano quasi fosse vero, si scopre un intero mondo "magico" fatto di una lunga catena produttiva che mette all'opera un gran numero di artisti e tecnici del settore.

All'interno di questa grande catena vi è un particolare anello di giunzione che permette a un personaggio 3D di prendere vita e muoversi: il rigging. Si tratta di un insieme di tecniche atte a definire comandi e controlli per il movimento del character.

Lo sviluppo di tecniche e algoritmi per la creazione di prodotti animati in Computer Grafica ha portato a grandi risultati e la velocità con cui questo campo si evolve e si aggiorna sta portando l'industria cinematografica e videoludica a livelli sempre più alti.

Nel campo della cinematografia, oggi si sta sviluppando con sempre maggior cura la tecnica del "motion capture", ossia della cattura del movimento di un attore tramite tanti piccoli sensori attaccati ad una tuta per il corpo o facciali.

Di particolare interesse è sempre stato lo studio del movimento umano e animale da trasporre poi su animazioni computerizzate. Si tratta di movimenti non facili da riprodurre tramite tecniche classiche frame by frame, perciò lo studio della motion capture sta dando una svolta in tal senso permettendo la riproduzione sempre più fedele di movimenti anche molto complessi.

Il mocap è molto utilizzato anche nei videogiochi, permettendo così un'animazione molto più fluida dei personaggi dando al giocatore un'esperienza sempre più vicina al reale.

Seguendo questo tipo di ragionamento, si sta cercando di andare oltre il limite tra reale e immaginario sviluppando sempre più sofisticate tecniche di realtà virtuale, tramite cui lo spettatore (o giocatore) può immergersi completamente nell'esperienza visiva.

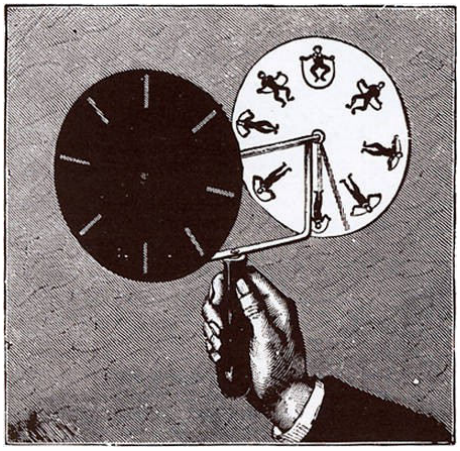
2. Il cinema d'animazione e la Computer Grafica:

2.1 Dall'animazione tradizionale all'animazione di modelli 3D

La cinematografia, nella sua storia, ha attraversato diverse fasi e periodi, che l'hanno portata dai primi rudimentali "esperimenti" dei fratelli Lumière ai moderni film digitali, ricchi di effetti speciali realizzati principalmente con la Computer Grafica.

Il cinema d'animazione è un importante capitolo della storia del cinema, che deve la sua nascita agli esperimenti ottici fatti da scienziati ed inventori che durante il diciannovesimo secolo hanno intrigato il pubblico con alcuni dispositivi che potevano prendere una sequenza di disegni e dare l'impressione che si muovessero. La prima di queste particolari invenzioni è da attribuire al Dr. Joseph Antoine Plateau e il Dr. Simon Ritter von Stampfer. Il loro dispositivo consisteva in due dischi rigidi montati su una singola asta. Le immagini erano attaccate cronologicamente nel disco interno che veniva fatto ruotare, così che un osservatore vedeva l'immagine in movimento guardando attraverso delle piccole fessure sul bordo esterno del secondo disco.

Questo sistema venne sviluppato creando lo zootropio.



La nascita ufficiale dell'animazione si deve attribuire a Émile Reynaud, l'inventore del Théâtre optique, una complessa macchina che proiettava su un telo figure disegnate su un rullo di carta, grazie a un gioco di specchi.

Con la successiva invenzione del proiettore e della macchina da presa, il cinema d'animazione dovette reinventarsi. Dapprima le animazioni riguardarono il movimento di oggetti inanimati, come i film di trucchi di Georges Méliès, poi utilizzando il procedimento fotogramma per fotogramma (frame by frame) si approdò ad una forma più moderna di cinema d'animazione.

Si diffondono inizialmente i primi cartoons e poi veri e propri lungometraggi animati.

2.2 Sviluppo delle tecniche di animazione:

Dall'avvento del proiettore e della macchina da presa, l'animazione è stata contrassegnata dalla tecnica "fotogramma per fotogramma", che consiste nel far scorrere un fotogramma alla volta davanti ad una macchina da presa posta in posizione verticale.

Di seguito alcune delle tecniche utilizzate nell'animazione:

- *I fogli di celluloide (cel animation):*

  Si tratta della tecnica più tradizionale, in cui si disegnano i singoli movimenti (frammentati nelle loro fasi) su dei sottili fogli di celluloide. Vengono poi fatti scorrere davanti ad una macchina da presa, dando così l'illusione di un movimento fluido e continuato. Nel caso di fotogrammi colorati, su un lato del foglio si disegna il contorno della figura, sul lato opposto si stende il colore. I movimenti principali (key-frame o fotogrammi chiave) vengono disegnati dal key animator, i movimenti intermedi vengono realizzati dall'intercalatore mentre il colore è posto infine dall'inchiostratore.

  Questa tecnica è quella maggiormente conosciuta ed usata, specialmente nel campo dell'animazione tradizionale.

- *L'animazione stop motion:*

  Questa tecnica significa letteralmente “fermare il movimento”: si fotografa l'oggetto da animare realizzato materialmente in plastilina o carta o argilla, gli si fa compiere

un leggero movimento e si esegue un altro scatto. La sequenza di scatti proiettata velocemente dà l'idea di un movimento continuo.

Grandi amanti di questo tipo di animazione sono il regista *Tim Burton*, autore di tre lungometraggi animati in stop motion, Nightmare Before Christmas (1993), La sposa cadavere (2005) e Frankenweenie (2012), e *Nick Park*, ideatore dei personaggi di Wallace & Gromit, protagonisti di diversi corti e lungometraggi.

– *L'animazione di silhouette:*

Le silhouette usate nel cinema d'animazione sono figure nere riprese di profilo, ritagliate solitamente su un foglio metallico, che a differenza della carta è più robusto e quindi più difficile da rovinare, oppure su cartoncino. Queste silhouette non sono figurine uniche, ma sono composte da articolazioni mobili e sostituibili per rendere più agevole il lavoro di animazione. Le silhouette vengono poste su un piano di lavoro orizzontale e retroilluminato per conferire massima opacità al nero e luminosità ai fondali colorati, vengono poi riprese dall'alto. Questa tecnica fu utilizzata largamente da Lotte Reiniger e Noburo Ofuji, caduta poi in disuso.

– *Animazione al computer (Computer Animation):*

L'animazione digitale è l'insieme delle tecnologie informatiche (software e hardware) applicate al campo dell'animazione.

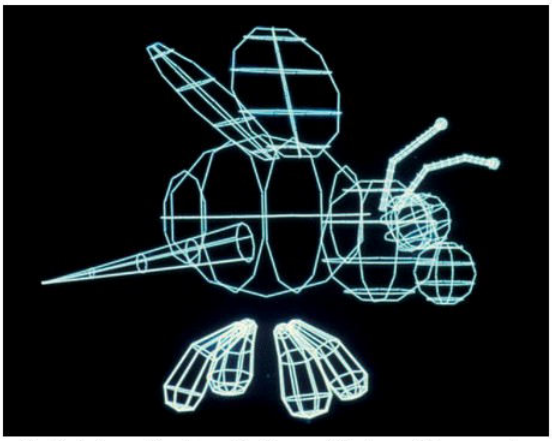
Essa cominciò a diffondersi negli anni sessanta. Dopo sperimentazioni accademiche e prove di concetto, le prime applicazioni commerciali furono per grafiche pubblicitarie, effetti speciali per film e sigle di programmi televisivi. Uno dei primi effetti dell'informatica applicata al campo dell'animazione fu il rinnovamento della classica animazione bidimensionale disegnata.

Con il successivo sviluppo della grafica tridimensionale, ci fu un ulteriore espansione delle possibilità dell'animazione.

Edwin Catmill, informatico e imprenditore statunitense, fu uno dei precursori che diede maggior contributo alle ricerche nell'ambito della grafica computerizzata. Nel 1973 riuscì a creare una versione digitale e animata della sua mano in 3D.

Nel 1979 iniziò a lavorare alla LucasFilm, cooperando con George Lucas nella divisione per gli effetti digitali. Successivamente, con l'arrivo di John Lasseter come animatore e con la collaborazione di Alvy Ray Smith, un altro pionere della Computer Grafica, diedero vita al primo cortometraggio interamente realizzato in 3D: “The adventure of Andrè and Wally B.”

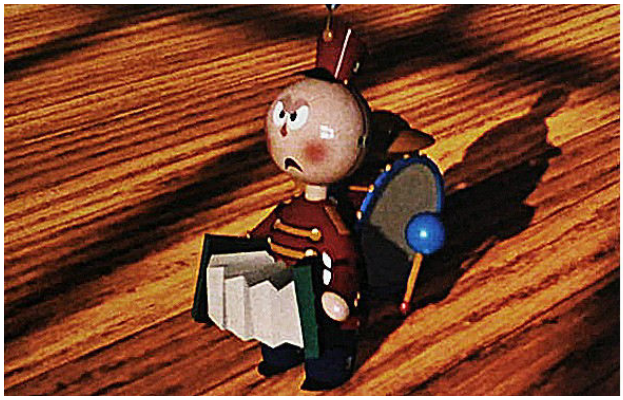
John Lasseter's design sketch of the character Wally B., from the short film *The Adventures of André & Wally B. Copyright © 1984 Pixar*

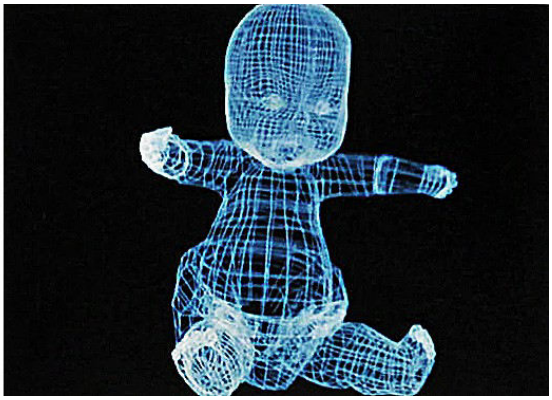


The “wireframe,” or the underlying architecture of the computer model, of the character Wally B. *Copyright © Pixar*

Presentato al SIGGRAPH (conferenza sulla grafica computerizzata) del 1984 il corto ebbe un grandissimo successo. In seguito alla fama ottenuta, la Computer Division della Lucas Film fu acquistata da Steve Jobs, diventando la Pixar.

Nel 1988, un altro ambizioso progetto Pixar, vinse un premio all'Academy Award: Tin Toy. Questo cortometraggio era incentrato sull'incontro tra un bambino di pochi mesi e un piccolo giocattolo meccanico. Questo progetto diede alla luce il primo modello 3D di un bambino umano.





Tin Toy è stato il precursone di “Toy Story - Il mondo dei giocattoli” (1995) il primo lungometraggio d'animazione realizzato interamente in computer animation tridimensionale.

Fu una rivoluzione nel campo dell'animazione che ha visto la nascita di un nuovo genere di film con un ventaglio di pubblico molto più vasto di quanto l'animazione tradizionale avesse in precedenza.

Con il passare degli anni, le tecniche e lo studio della Computer Grafica hanno portato a uno sviluppo sempre maggiore anche nel settore videoludico e televisivo.

3. Preparazione del modello:

3.1 Pipeline di produzione di un character 3D:

Le forme tridimensionali nascono in realtà come concept 2D. Si parte da un'idea, la si disegna su carta e dopo l'aggiunta di più o meno dettagli, si realizza il modello 3D.

Con la fase di concept si intende la bozza concettuale dell'idea in fase embrionale. E' la fase in cui si definiscono degli elementi fondamentali di un progetto; si parte dalla realizzazione di alcuni sketch di prova, che vengono poi analizzati e raffinati fino ad arrivare ad un disegno più o meno particolareggiato del modello che si vuole realizzare.

Dopo l'approvazione del concept, si passa alla sua realizzazione in 3D ossia il processo atto a definire una forma tridimensionale in uno spazio virtuale.

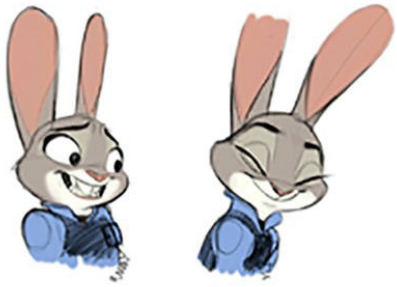
Inizialmente si crea una mesh di base del modello utilizzando poligoni semplici con pochissimi vertici e si procede con estrusioni atte solo a definirne un contorno. La mesh base viene in seguito dettagliata e definita con l'aggiunta di vertici e edge (lati) fino a creare il modello definitivo su cui andare ad applicare materiali e texture.

Prima di passare all'animazione vera e propria, è necessario creare una "imbracatura" digitale che permette il movimento del modello, ossia uno scheletro e un sistema di controlli atti a modificare le trasformazioni e le deformazioni del modello stesso. Questa fase viene chiamata "**rigging**".

Infine il modello preparato e riggato può essere animato.

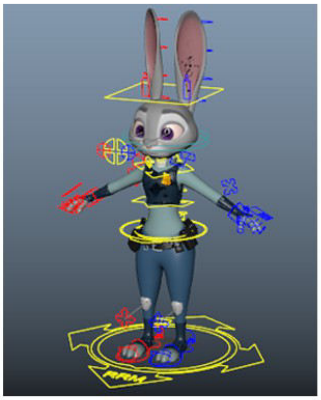
PIPELINE Character Animation

1. Concept

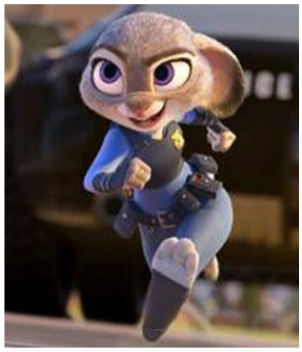


2. 3D Model & Texture

3. Rigging



4. Animation

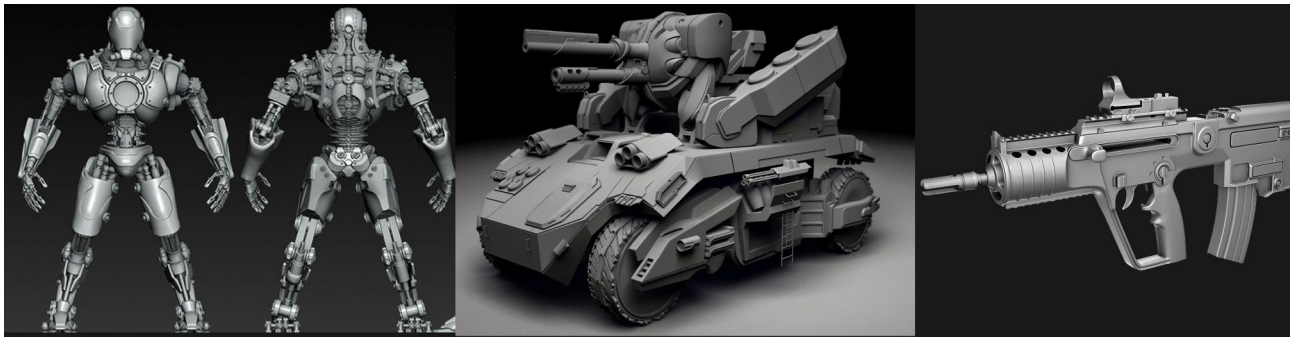


3.2 Specifiche del modello: elementi e tipologie

Un modello 3D è una rappresentazione virtuale di un corpo fisico.

I modelli 3D si differenziano a seconda dell'utilizzo e del tipo di modellazione in:

- **Hard Surface models:** si tratta di oggetti artificiali che hanno una superficie rigida, piatta e non presentano grandi rotondità. Sono oggetti che non subiscono particolari deformazioni morbide in fase di animazione. Alcuni esempi sono gli oggetti di uso comune, quelli meccanici o i robot.



- **Organic Models:** si tratta di oggetti organici più curvi e arrotondati, dalle forme sinuose. Questi oggetti subiscono deformazioni morbide in fase di animazione. Ne sono un esempio i corpi umani e animali.

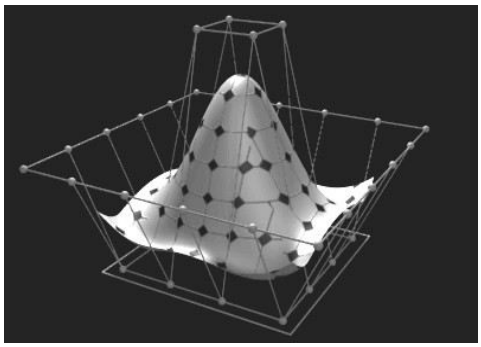
La maggior parte dei modelli tridimensionali può essere divisa in due grandi categorie:

3.2.1 Superfici NURBS e curve:

Le Nurbs, acronimo di "Non Uniform Rational B-Spline", sono particolari funzioni matematiche riconducibili a figure, linee e superfici, definite da un algoritmo che rappresenta oggi il più efficace ed avanzato grado evolutivo della modellazione tridimensionale.

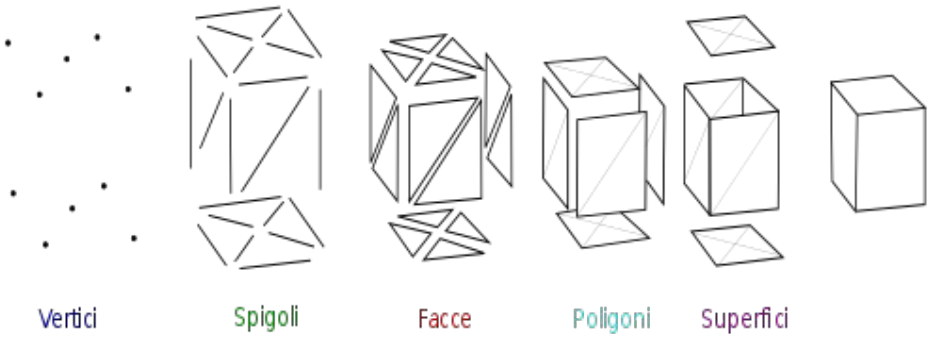
La geometria NURBS presenta alcune proprietà che la rendono adatta per la modellazione:

- si possono rappresentare sia oggetti geometrici di base come linee, cerchi, ellissi, sfere e tori, sia geometrie dalla forma libera (free-form);
- le quantità di informazioni richieste per rappresentare una geometria mediante NURBS è inferiore alle informazioni necessarie per rappresentare la stessa geometria mediante maglie poliginali.



3.2.2 Mesh Poligonale:

Una mesh è un reticolo che definisce un oggetto nello spazio. Questo reticolo è composto fondamentalmente da tre elementi: vertici, spigoli, e facce. Differentemente da un oggetto solido reale, non presenta una massa ed è quindi una sorta di volume vuoto, privo di spessore, le cui facce sono piani superficiali.



I vertici (“vertex” in inglese) sono dei punti dello spazio dotati di coordinate x, y, z che ne determinano la posizione e costituiscono la base per definire gli spigoli (“edges”), ovvero i segmenti che congiungono due vertici nello spazio. A loro volta, la connessione e la chiusura tra gli spigoli, genera le facce (“face”) o poligoni della mesh.

Il numero di poligoni che una mesh presenta va a determinare il livello di dettaglio e accuratezza di tutto il modello. Si fanno quindi due particolari distinzioni:

- **Modelli Low poly:** possiedono un basso numero di poligoni, o in altre parole, un basso numero di vertici, spigoli e facce. Il modello è privo di dettaglio; gli spigoli appaiono come irregolari e affilati.

  Vantaggi:

  – Risparmio di memoria: l'avere una bassa conta poligonale permette di non sovraccaricare la macchina a livello computazionale.

  – Risparmio di spazio su disco: grazie al basso numero di poligoni, il file su cui si lavora occupa meno spazio sull'hard disk (pesa meno).

  Svantaggi:

  – Mancanza di dettaglio: il basso numero di poligoni non permette di avere un alto dettaglio, tanto che alcuni spigoli non voluti potrebbero rimanere visibili.

  Applicazioni:

  La modellazione low poly trova maggior impiego nello sviluppo dei videogiochi, in quanto la velocità di calcolo gioca un ruolo fondamentale per permettere una visione fluida delle animazioni di gioco, calcolate in real time.

  I modelli Low Poly possono anche essere utilizzati in cinematografia, con particolari scelte stilistiche.

- **Modelli High poly:** possiedono un elevato numero di poligoni. Questi modelli presentano maggior dettaglio, non hanno evidenti spigoli su superfici lisce e risultano più fotorealistici.

  Vantaggi:

  – Livello di dettaglio: si possono aggiungere elementi andando a scolpire minime parti per conferire maggior realismo.

  – Animazioni realistiche: grazie all'alto numero di vertici disponibili, le deformazioni della mesh saranno più fluide garantendo così un'animazione più vicina alla realtà.

  Svantaggi:

  – Sfruttamento di memoria: a causa dell'elevato numero di poligoni da dover calcolare, le prestazioni della macchina risultano più basse.

  – Sfruttamento di spazio su disco: il file del modello con alta densità poligonale occupa molto spazio nell'hard disk.

  Applicazioni:

  I modelli con alta conta poligonale sono utilizzati maggiormente nei prodotti cinematografici, in cui il fotorealismo gioca un ruolo chiave.

3.3 Topologia:

Durante la creazione del modello tridimensionale, va prestata particolare attenzione alla sua "topologia" .

Nel campo della Computer Grafica, con il termine "topologia" ci si riferisce generalmente alla distribuzione e organizzazione dei poligoni che definiscono la superficie di una mesh.

Una corretta topologia prevede quindi che i poligoni vengano studiati, disegnati e inseriti al fine di ottimizzare la rappresentazione del volume che si va a ricostruire, usando così il minor numero di poligoni per definire al meglio l'oggetto.

Ovviamente la topologia varia notevolmente da oggetto a oggetto: l'andamento dei poligoni che avrà un viso sarà totalmente differente da quello di un oggetto di design o ancora, di una autovettura.

Tutto ciò permette di rappresentare senza imperfezioni oggetti statici che non subiranno deformazioni fisiche, e soprattutto personaggi, creature, oggetti che andranno animati, e quindi di conseguenza deformati.

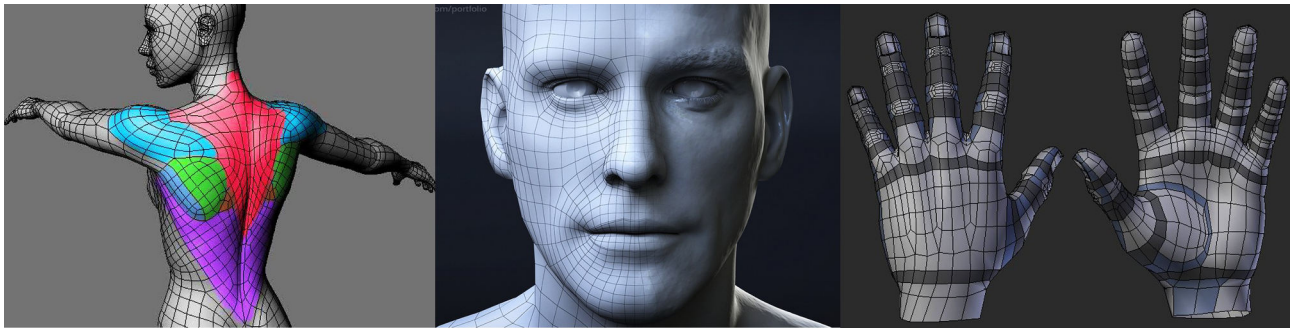
Regole di una corretta topologia:

- L'andamento dei poligoni lungo la superficie dovrà seguire e tener conto non solo dei volumi esterni, ma anche dell'anatomia e morfologia interna dell'oggetto che si intende virtualizzare (animali, creature, umani);

- Ogni Edgeloop poligonale (ossia un insieme ad "anello" di lati contigui) deve essere posizionato con una ragione d'essere, ossia deve aiutare a definire la forma ma anche la funzione della parte di oggetto che si sta modellando.

– I poligoni devono essere posizionati ad una distanza uniforme gli uni dagli altri (Edge flow), con diversa densità a seconda di quanto deve deformarsi la zona di interesse. In questo modo verrà mantenuto un equilibrio visivo che, oltre ad avere una ovvia miglioria estetica, aiuterà la rappresentazione dei cambiamenti di curve e piani del nostro oggetto.

– L'uso principale di "Quad", ossia di poligoni a quattro lati, permette una distribuzione più pulita della geometria e consente di effettuare la funzione di "smooth mesh", che consiste nel suddividere i quadrilateri passando per i punti medi del loro perimetro, consentendo maggior "morbidezza" della mesh. Nel campo dei videogiochi vengono usati i Tris (poligoni a tre lati), in quanto consentono velocità di calcolo in real time. Mentre sono assolutamente da evitare gli Ngons (poligoni con più di quattro lati) poichè creano problemi a livello di render per la possibile sovrapposizione di vertici e edge.



Una corretta topologia è necessaria per poter gestire più facilmente le deformazioni del modello nella successiva fase di rigging.

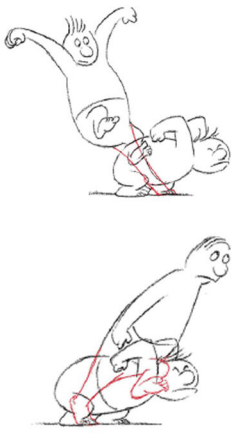
4. Approfondimento della fase di Rigging di un Character 3D:

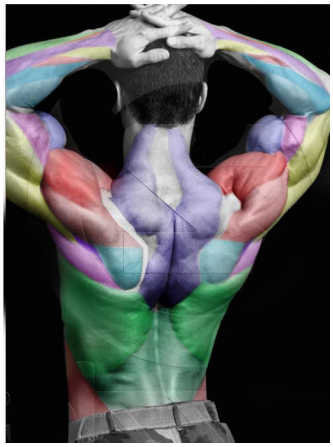
Nella pipeline di produzione di un modello 3D animabile, il rigging si colloca dopo la creazione del modello perché, prima di poterlo animare, è necessario creare l'intera ossatura interna che andrà ad influenzare le deformazioni della mesh tramite l'uso di controlli esterni atti a modificare le trasformazioni di traslazione, rotazione e scala dei vertici della mesh stessa.

Il rig di un personaggio 3D deve essere ben studiato a seconda dei movimenti e delle deformazioni che esso dovrà fare nelle animazioni successive. Per un character di fattezze umane o animali lo studio si basa principalmente sulla comprensione dell'anatomia del personaggio, con particolare attenzione alle deformazioni muscolari che esso dovrà compiere nei suoi movimenti. Inoltre bisogna considerare se il character dovrà essere sottoposto a deformazioni di tipo "cartoon" che prevedono particolari "squash" e "stretch" della mesh; o se si tratta di deformazioni realistiche.

I rig di oggetti meccanici invece prevedono uno studio approfondito di tutto il sistema meccanico, col fine ultimo di automatizzarne il più possibile il funzionamento.

Un buon rig deve essere pensato per facilitare al meglio il lavoro dell'animatore.





4.1 Software dedicato – Autodesk Maya 3D:

Autodesk Maya è un software di computer grafica 3D, apprezzato in tutto il mondo per l'alta qualità degli strumenti di modellazione, animazione e rendering. E' uno dei software più usati per la realizzazione di film e videogiochi in CGI.

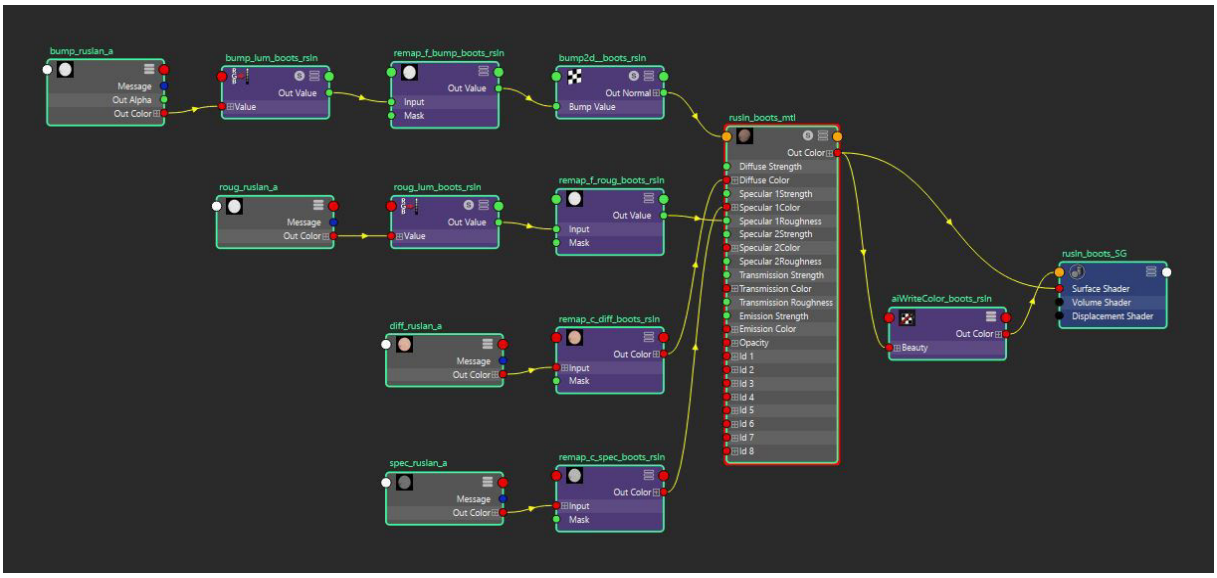
La sua ricchissima interfaccia grafica è completamente personalizzabile e offre grande libertà di creazione di tool tramite l'uso di uno script editor che supporta linguaggi ad oggetti quali MEL e Python. Il suo grande punto di forza è la struttura a nodi.

Ogni oggetto che viene creato in Maya è formato da diversi "nodi":

- un nodo di "creation" che si occupa delle opzioni di creazione dell'oggetto;
- un nodo di "transform" che gestisce le traslazioni, le rotazioni e la scala dell'oggetto;
- un nodo di "shape" che gestisce la descrizione matematica dei punti di controllo dell'oggetto.

Maya permette di manipolare questi nodi, modificandoli ad hoc e aggiungendone di nuovi, creando sistemi anche complessi.

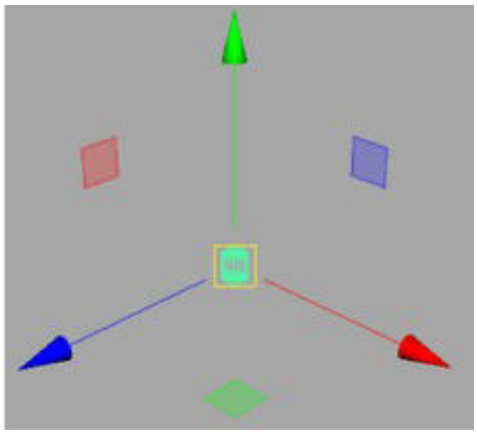
Offre quindi grandi possibilità per quanto riguarda sia una programmazione ad "oggetti" che una programmazione a "nodi".



4.2 Elementi base del rig:

4.2.1 Spazio di riferimento 3D in Maya:

Lo spazio tridimensionale nel quale si lavora è un'estensione del piano bidimensionale in cui un punto nello spazio viene definito tramite tre coordinate (x,y,z). Ogni punto, quindi, definisce una posizione nello spazio dove “x” rappresenta l'asse orizzontale, “y” l'asse verticale e la “z” ne rappresenta la profondità.



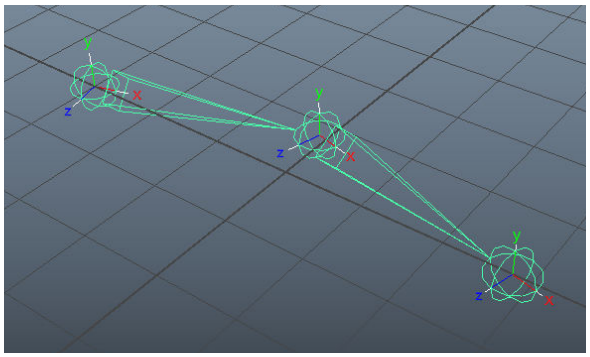
4.2.2 Joint – ossatura:

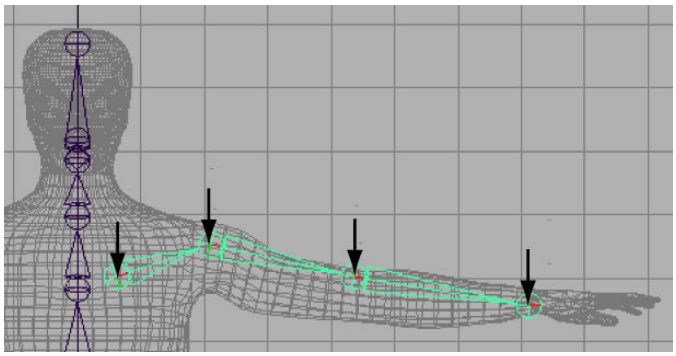
Con il termine "joint" nel rigging si intende un centro di trasformazioni fittizio per i vertici di una geometria, ossia un centro relativo di trasformazioni quali la traslazione, la rotazione e la scala.

Può essere visto come una connessione tra parti di una mesh, una parte movibile che costituisce lo scheletro di un modello 3D.

Ogni joint possiede le proprie trasformazioni nello spazio (x,y,z) e il suo compito è quindi quello di definire la libertà di movimento del modello. I joint di solito seguono una

struttura gerarchica a partire da un joint "padre" a un joint "figlio", fino a creare vere e proprie catene di joint.

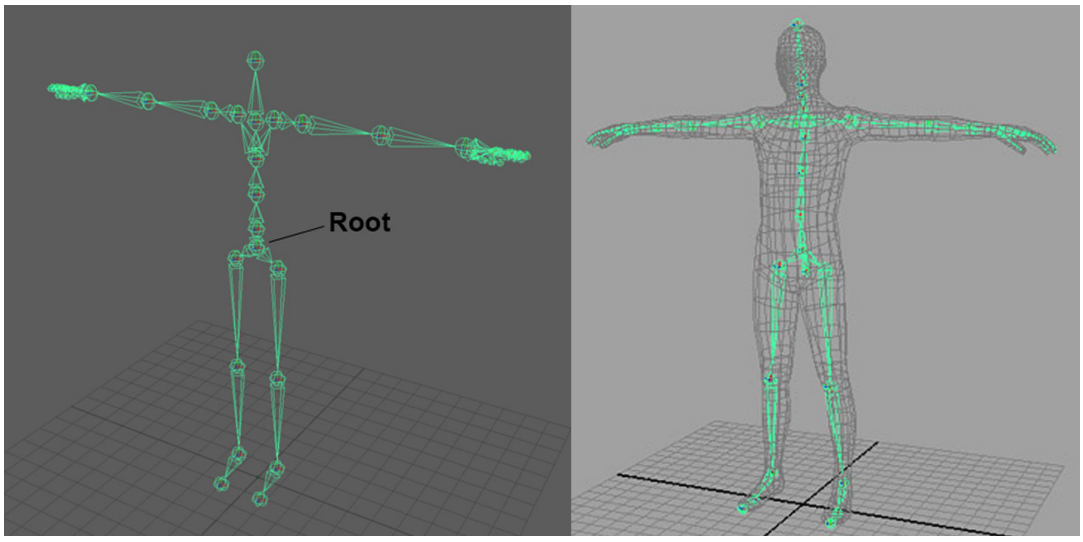




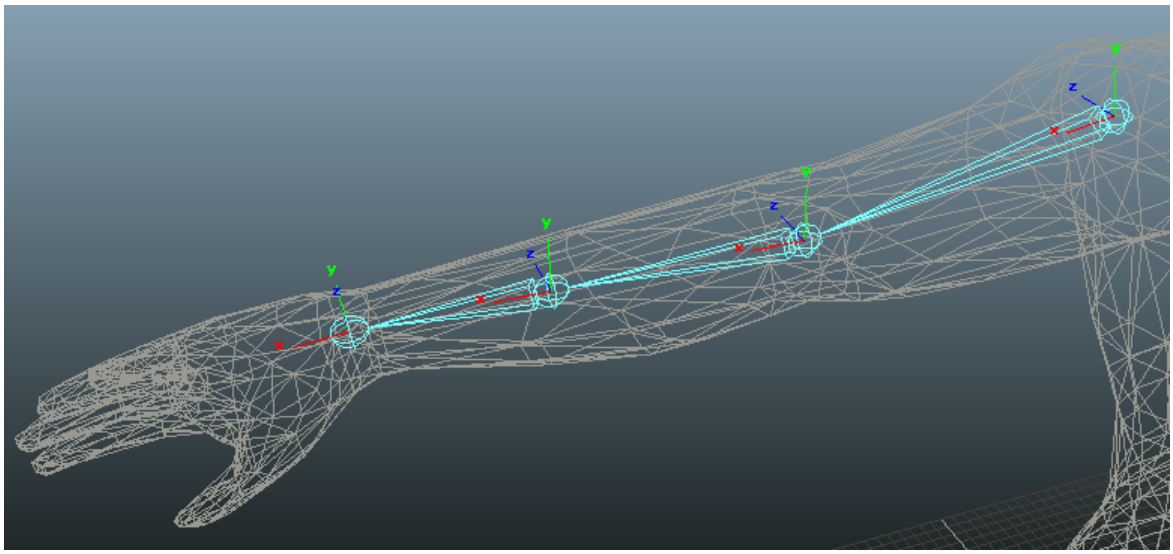
La gerarchia fa si che le trasformazioni del "padre" vadano ad influenzare le trasformazioni dei joint "figli", a cascata.

La prima cosa da fare al fine di creare un buon rig di un personaggio, è proprio quella di creare i joint del suo scheletro. Essi vanno posizionati più o meno correttamente seguendo la struttura ossea del personaggio, tenendo conto delle giunture, delle parti mobili e delle parti deformabili. Bisogna inoltre tenere conto della gerarchia delle ossa di un personaggio e fare in modo che il joint "root" sia a capo di tutte le catene in senso logico, perché esso influenzerà le trasformazioni di tutti i suoi "figli".

Questo significa che, nel caso di un personaggio umanoide, la "root" (radice) principale di tutta l'ossatura sarà rappresentata dal joint posizionato al posto dell'osso sacro (pelvis - bacino). Questo joint sarà il punto di partenza delle catene che andranno a formare le gambe e di quella che formerà la spina dorsale fino alla testa, da cui partiranno le catene di joint delle braccia. Ogni joint, quindi, va posizionato in prossimità di una giuntura o articolazione.



Ogni joint rappresenta un centro relativo di trasformazioni, quindi, ognuno di essi possiede il proprio "pivot" (ossia il proprio centro) che ne determina l'orientamento. E' importante assicurarsi che l'orientamento di ogni joint sia coerente con l'andamento della catena di cui fa parte. Un corretto orientamento degli assi permette una corretta rotazione delle giunture.

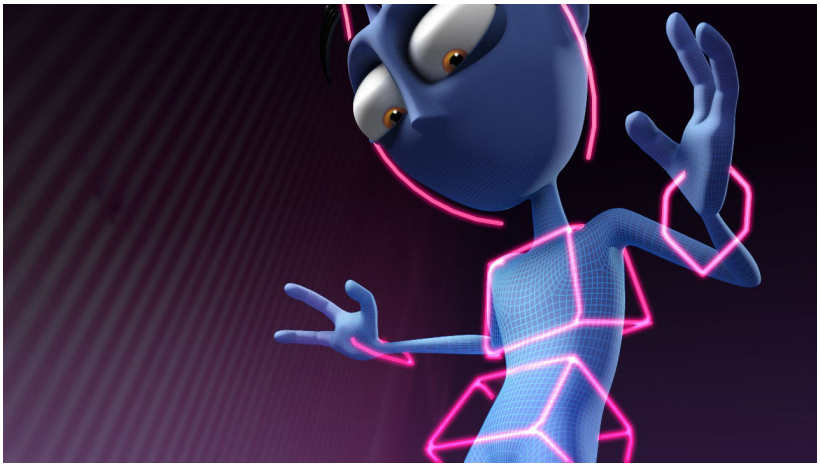


Ogni joint, come detto, possiede un proprio orientamento ma dopo la creazione di una catena gerarchica è possibile che questo orentamento differisca per ognuno di essi. Ci si deve quindi accertare che gli assi di rotazione siano tutti uguali e conformi, ovvero, a partire dal primo joint "padre" della catena, il suo asse primario dovrà puntare al joint successivo e così tutti gli altri assi primari dei suoi figli, a cascata.

Un altro importante fattore da tener conto è il numero di joint da usare in una catena. Un joint va ad influenzare la geometria su cui viene applicato in base a un "peso" che varia da 0 a 1, perciò avere un gran numero di joint significa avere un maggior controllo sulla deformazione della mesh. Purtroppo però, questo va a pesare anche a livello computazionale, abbassando notevolmente le prestazioni del computer nella fase di calcolo. Questo renderebbe inutilizzabile il file e, soprattutto, renderebbe impossibile animare il modello.

Bisogna perciò tener conto di tutti questi fattori per ottenere un buon rig, che sia preciso nelle deformazioni ma anche "leggero" a livello computazionale.

4.2.3 Controlli:



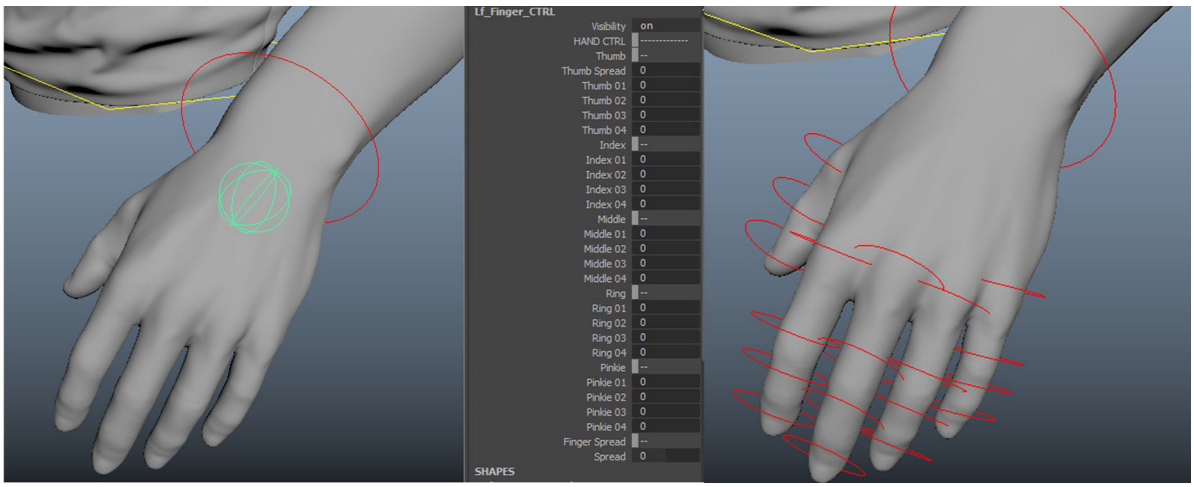
Una volta ottenuto un buono scheletro di joint, è necessario creare i controlli che costituiscono la parte visibile esterna del rig finale e che l'animatore userà per modificare le trasformazioni della mesh nel tempo.

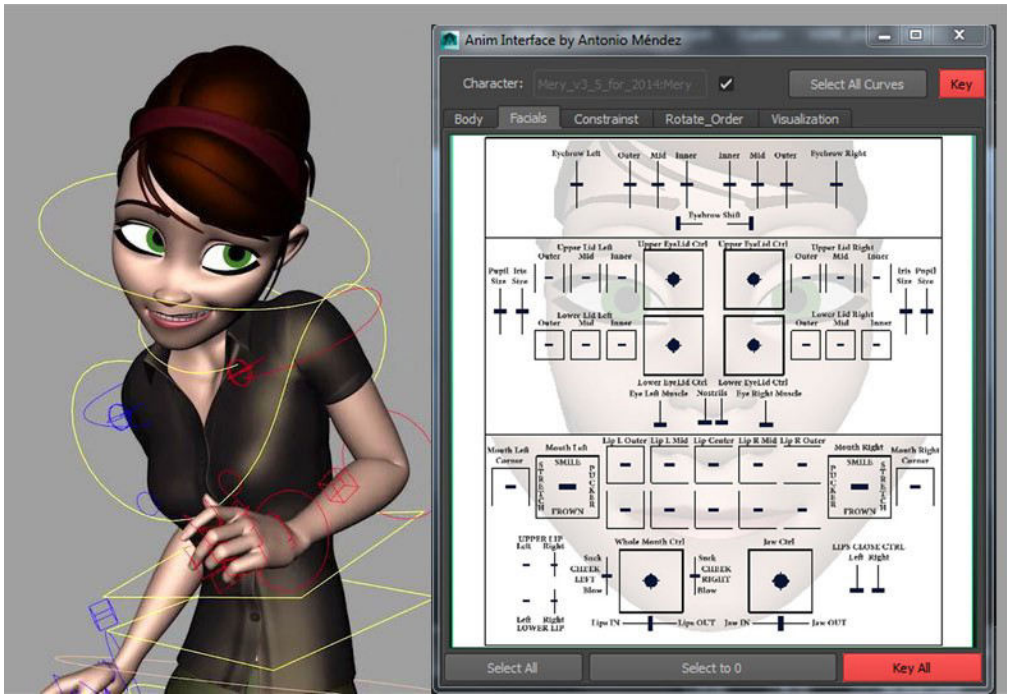
Si tratta, per la maggior parte, di curve NurbS posizionate in prossimità dei joint. Esse devono avere lo stesso orientamento dei joint che dovranno controllare in modo da evitare errori consistenti nelle deformazioni, dati da un disallineamento dei pivot e causando, quindi, un disallineamento delle rotazioni con una possibile perdita di grado di libertà corrispondente ad un asse bloccato (gimbal lock – blocco cardanico).

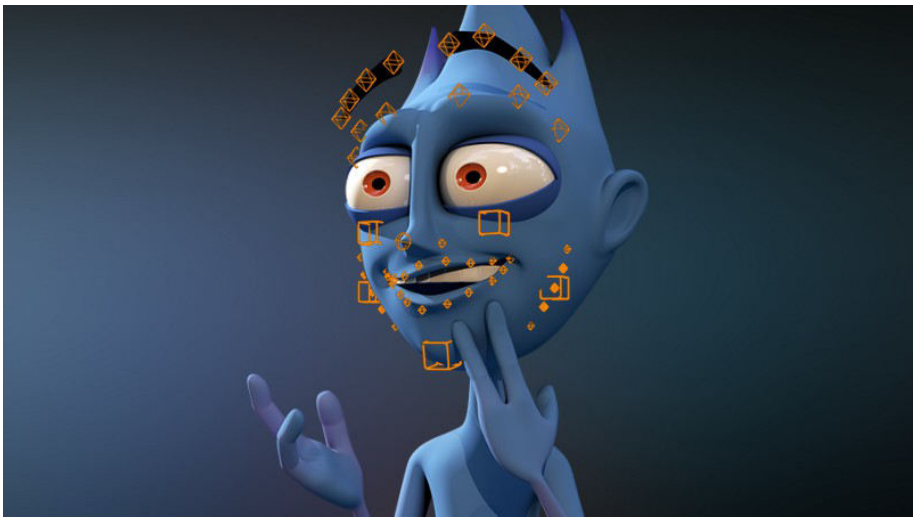
Tramite i controlli si possono gestire tantissimi parametri anche grazie l'aggiunta di particolari attributi che facilitano alcune deformazioni e automatizzano alcuni sistemi. Ad esempio, si può creare un unico controllo che gestisca la deformazione di tutte le dita di una mano, anche singolarmente, senza il bisogno di creare tanti controlli per ogni falange. Oppure si può creare un controllo ad hoc che gestisca un sistema meccanico automatico. Oppure, ancora, si possono creare interi pannelli di controlli atti a gestire "a distanza" alcune deformazioni. Questi pannelli vengono maggiormente usati per il rig facciale, creando delle piccole interfacce.

I controlli devono essere di facile comprensione e facilmente cliccabili e selezionabili da parte dell'animatore.

Inoltre, è necessario rispettare la stessa gerarchia delle ossa.


Lf_Finger_CTRL
Visibility on
HAND CTRL
Thumb
Thumb Spread
Thumb 01
Thumb 02
Thumb 03
Thumb 04
Index
Index 01
Index 02
Index 03
Index 04
Middle
Middle 01
Middle 02
Middle 03
Middle 04
Ring
Ring 01
Ring 02
Ring 03
Ring 04
Pinkie
Pinkie 01
Pinkie 02
Pinkie 03
Pinkie 04
Finger Spread
Spread
SHAPES


Anim Interface by Antonio Méndez
Character:
Select All Curves
Key
Body
Facials
Constrainst
Rotate_Order
Visualization
Select All
Select to 0
Key All



4.2.4 Constraint:

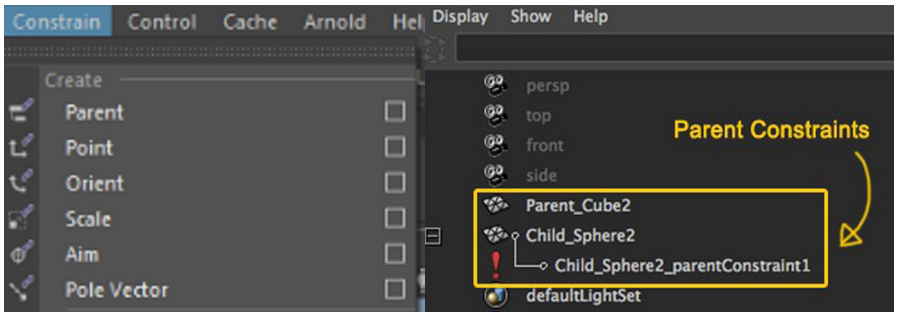
I constraint sono particolari tools che permettono di "costringere" la posizione, la rotazione e la scala di un oggetto "padre" ad un altro oggetto "figlio" (o target).

La costrizione che si genera tra i due oggetti permette di controllare il target tramite la modifica delle trasformazioni dell'oggetto "padre". In altre parole, il "padre" costringe il "figlio".

Gli oggetti del rig su cui lavorano i constraint sono i controlli, che svolgono il ruolo di oggetto "padre", e i joint, che svolgono il ruolo di oggetto "figlio". Così facendo si crea un sistema per cui è possibile muovere, ruotare o scalare un joint tramite l'uso dei controlli appositamente creati e settati.

In Maya, i principali tipi di constraint usati sono:

- *Constraint Point:* permette all'oggetto "padre" di costringere l'oggetto "figlio" solo per quanto riguarda le sue traslazioni, lasciandolo libero sulle rotazioni e sulla scala.
- *Constraint Orient:* permette all'oggetto "padre" di costringere l'oggetto "figlio" solo sui parametri di rotazione, lasciando libere le traslazioni e la scala.
- *Constraint Parent*: agisce sia sui parametri di traslazione che sui parametri di rotazione.



- *Constraint Scale*: agisce solo sui parametri di scala.
- Constraint Aim: questo tipo particolare di constraint va ad influenzare l'orientamento di un oggetto affinchè punti sempre in una direzione ben specifica: costringe il pivot dell'oggetto "figlio" ad orientarsi sempre nella direzione dell'oggetto "padre". Viene usato spesso per controllare il movimento delle pupille degli occhi in un personaggio.

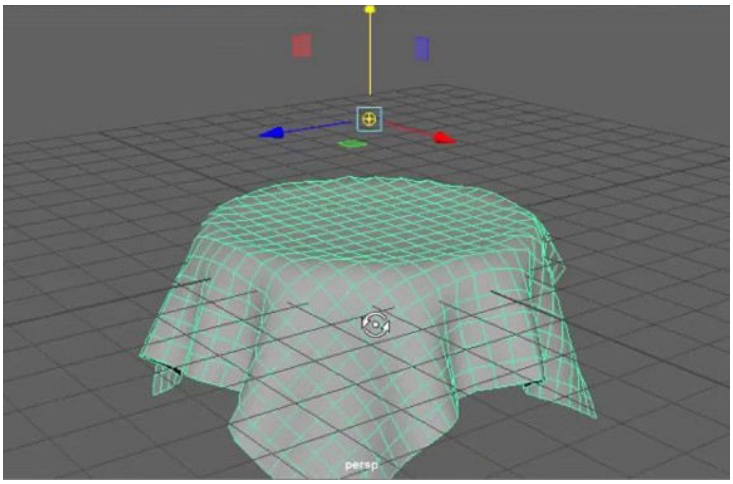
4.3 Cinematica diretta, Cinematica inversa e Dinamica

Nei software di computer grafica, quali Maya, è possibile studiare e strutturare la scena da animare seguendo le regole della fisica. Se si prendono in considerazione solo le posizioni e le velocità degli oggetti sulla scena si parla di “Cinematica”; se si considerano anche le forze che agiscono sugli oggetti si parla di “Dinamica”.

Per quanto riguarda la “dinamica” degli oggetti, esistono tools dedicati che permettono di lavorare su molti parametri al fine di simulare l'azione di una forza su un oggetto il più realisticamente possibile.
Si possono, ad esempio, modificare i parametri della forza di gravità a seconda della simulazione voluta, o della velocità del vento da applicare su una scena, o, addirittura, è possibile simulare collisioni tra oggetti.

Il rig in dinamica (Dynamic System) è utilizzato maggiormente per quanto riguarda i capelli, i vestiti, i tessuti o parti di creature come code, pinne e tentacoli.

Questa tecnica offre meno controllo “manuale” da parte dell'animatore, in quanto sarà sufficiente impostare le pose dei keyframe (fotogrammi chiave) e tutte le altre posizioni interpolanti verranno gestite e calcolate automaticamente dal software, con pochissima possibilità di modifica.
Questo può essere un vantaggio, se il rig è studiato alla perfezione e se i calcoli della dinamica sono settati correttamente ma la grande quantità di calcoli necessari rischia di sovraccaricare la macchina rallentandone le prestazioni.

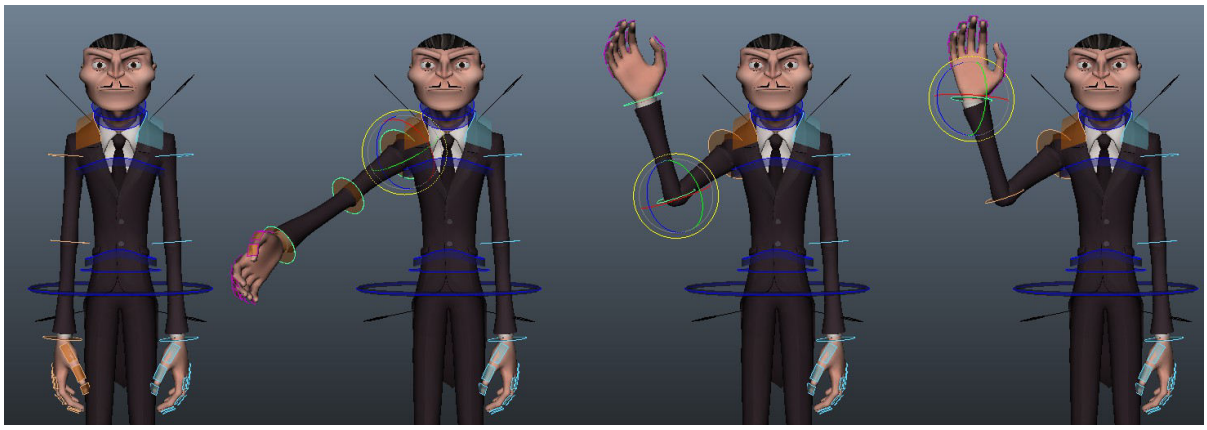


Esistono poi due differenti tipi di “cinematica” in cui strutturare il rig.

- cinematica diretta (Forward Kinematics – FK)
- cinematica inversa (Inverse Kinematics – IK)

I due tipi di cinematica differiscono nel modo in cui si va ad agire sull'oggetto.

4.3.1 Cinematica diretta – FK

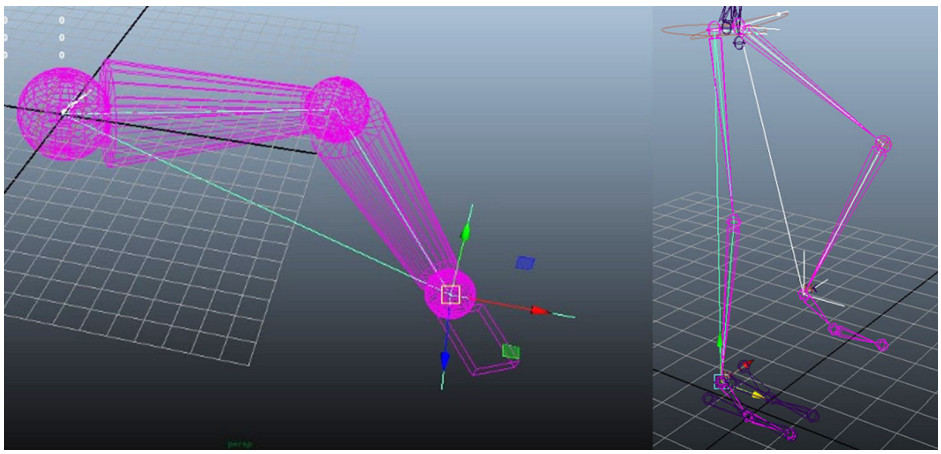


Nella cinematica diretta il rig di un personaggio o oggetto 3D è basato su sequenze di rotazioni applicate ai punti di snodo.

Una volta posizionati e orientati adeguatamente i joint e i rispettivi controlli, si va ad applicare tra essi un Constraint Orient in modo da poter gestire le rotazioni dei joint tramite le rotazioni dei controlli.
La gerarchia tra joint e controlli fa si che le rotazioni degli oggetti "padre" si ripercuotano sulle rotazioni degli oggetti "figli", perciò è necessario partire dalla rotazione della radice (root) della catena al fine di controllare le rotazioni che si ripercuotono sui joint successivi.
Questa tecnica è detta, appunto, Forward Kinematics perché si procede "in avanti" nella gerarchia delle catene di joint.

Tramite il rig in FK si ha un enorme controllo sulla geometria ed è possibile mettere in posa il soggetto con particolare precisione; lo svantaggio sta nel fatto che è necessario muovere singolarmente le parti dell'oggetto, quindi, se esso è formato da un gran numero di articolazioni e snodi, sarà necessario gestire le rotazioni di ognuno di essi per giungere alla posa finale desiderata.

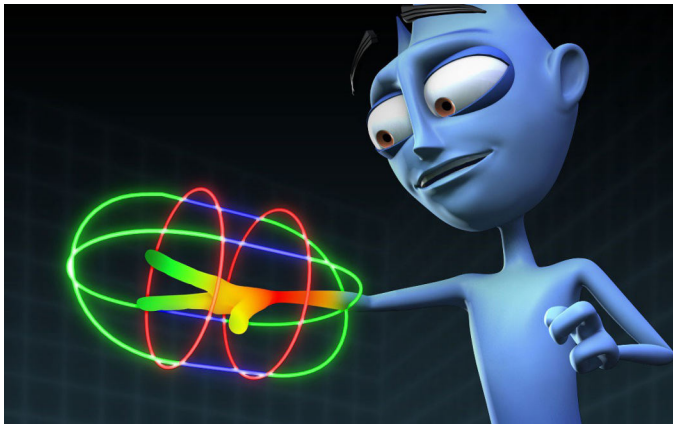
4.3.2 Cinematica inversa – IK



Nella cinematica inversa si “inverte” la direzione della manipolazione a catena. Ossia, è possibile determinare i parametri di posizione degli oggetti in base alla manipolazione delle sole estremità. Questo significa che invece di lavorare dalla radice (root) e proseguire da “padre” a “figlio” lungo le catene, si procede partendo dall'ultimo joint della catena mentre le posizioni dei joint intermedi vengono calcolate automaticamente.

Ad esempio, nella manipolazione di un braccio anziché dover gestire le rotazioni della clavicola, della spalla, del gomito e, infine, del polso, è sufficiente gestire la posizione del polso.
Il grande vantaggio di questo processo sta proprio nel fatto che diminuisce il numero di controlli da dover manipolare al fine di ottenere una posa; lo svantaggio, invece, risiede nella difficoltà di calcolare le posizioni intermedie, in quanto non ne esiste una univoca. Per ovviare a questo tipo di problema, si ricorre a limitazioni e vincoli delle posizioni e rotazioni dei joint.

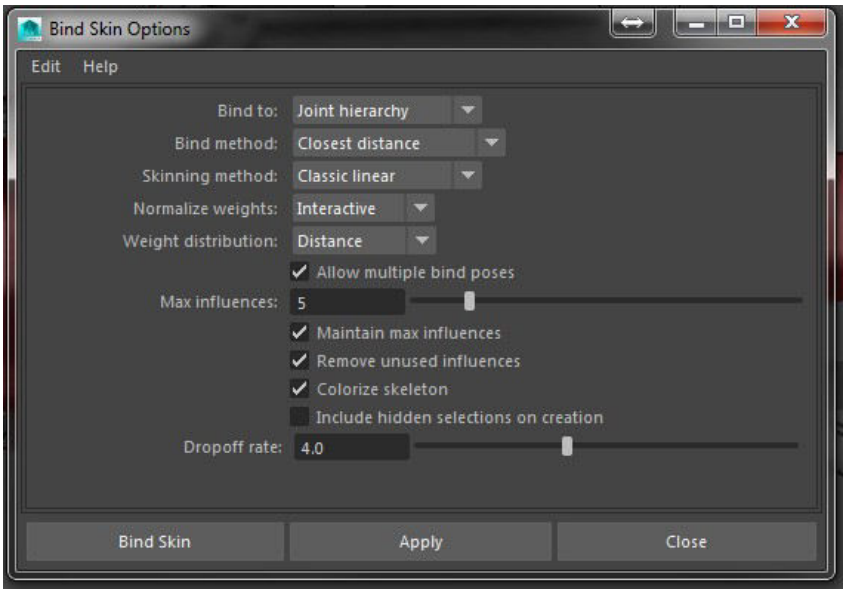
4.4 Bind Smooth Skin



Una volta creato il rig base di un character, costituito dallo scheletro di joint e dai controlli posizionati correttamente e settati in modo da manipolare le trasformazioni dello scheletro secondo i metodi FK o IK, si giunge al punto in cui è necessario far si che queste manipolazioni vadano ad intaccare la geometria del modello.
Si deve, in altre parole, “attaccare” la mesh allo scheletro.

Su Maya questo è possibile utilizzando l'apposito tool di “Bind Skin” (letteralmente “legare/congiungere la pelle”).

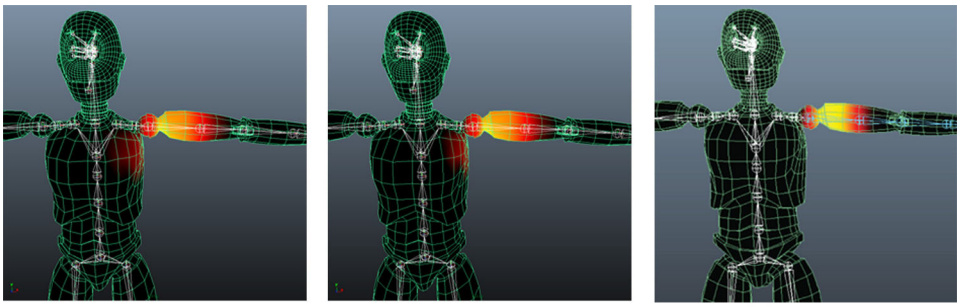
Si deve semplicemente selezionare la mesh (o parti di essa) e i joint dello scheletro da voler legare e poi si applica il tool adeguatamente settato.



Le principali opzioni sono:

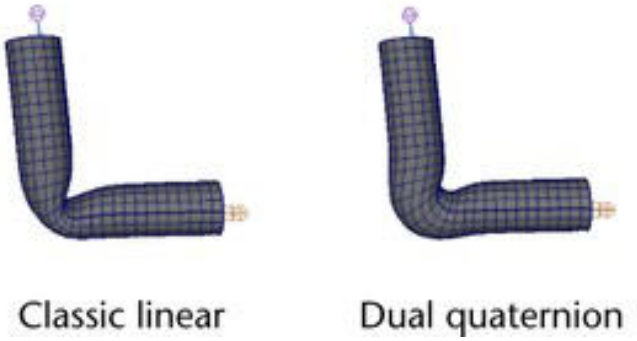
- **Bind to:** specifica se è necessario legare l'intero scheletro a partire dalla root (Joint Hierarchy), o solo parte di esso selezionando i joint di interesse (Selected Joint).

- **Bind Method:** specifica *come* i joint andranno ad influenzare i punti della geometria vicini ad essi. Esistono diversi algoritmi, tra cui:
  - *Closest in Hierarchy*: l'influenza dei joint si basa sulla gerarchia dello scheletro. Questo metodo evita che ci siano influenze non desiderate da parte di joint distanti nella catena;
  - *Closest Distance:* l'influenza dei joint si basa solo sulla prossimità dei punti della mesh. In questo caso Maya ignora la gerarchia dello scheletro perciò possono crearsi delle influenze inappropriate da parte di joint situati in altri punti;
  - *Geodesic Voxel:* usa una rappresentazione della mesh tramite voxel (unità di misura del volume, rappresenta il più comune pixel 2D riportato in 3D) che aiuta il calcolo del peso delle influenze tramite la "distanza geodetica" (curva più breve che congiunge due punti nello spazio) tra i voxel sullo scheletro e la mesh; il risultato di questo calcolo viene poi applicato allo "skinning method".



Closest Distance **Closest Hierarchy** **Geodesic Voxel**

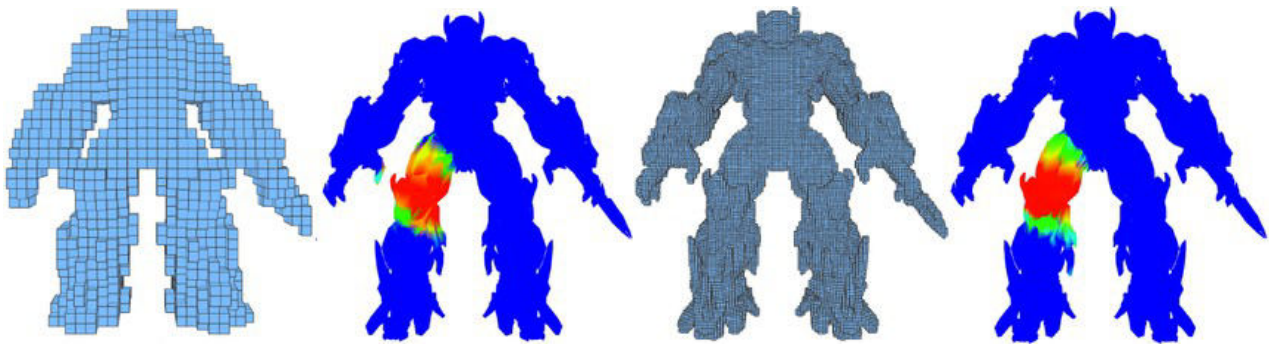
- **Skinning Method:** specifica l'algoritmo con cui la mesh segue l'andamento dei joint. Maya offre due metodi principali di smooth skin:
  - *Classic Linear*: è il metodo base, lineare, che consente il verificarsi di alcuni effetti di riduzione del volume della mesh;
  - *Dual Quaternion:* questo metodo consente di preservare il volume della mesh durante le deformazioni. E' progettato per ridurre o eliminare il problema della perdita di volume così da ottenere una deformazione più realistica dei punti critici. Ad esempio, quando il braccio si piega, nei punti di deformazione del gomito può venirsi a creare l'indesiderato "effetto cannuccia" con una perdita di volume e un effetto alquanto sgradevole: il Dual Quaternion riduce enormemente questo difetto.



- **Normalize Weights:** consente di impostare il modo in cui si desidera normalizzare le pesature. Solitamente viene usato il metodo "interactive" con cui è possibile aggiungere o togliere influenza dai joint tramite appositi tool di "paint skin weights".

- **Max Influences:** specifica il numero di joint che possono influenzare ogni punto della geometria.

E' poi possibile settare altri parametri che consentono di colorare lo scheletro, rimuovere le influenze non necessarie e scegliere la risoluzione della "voxelizzazione", ossia l'atto di convertire la geometria in un insieme di voxel che approssima la mesh. Quest'ultima opzione è disponibile solo se si usa il metodo "geodesic voxel" in quanto aumenta la precisione del calcolo dei voxel stessi.



4.4.1 Come settare le opzioni – alcuni esempi:

Il settaggio del tool dipende dal character che si va ad ottimizzare, si deve tenere conto di cosa il modello rappresenta, di quali movimenti andrà ad effettuare e di che tipo di materiale è composto: la deformazione di un oggetto rigido di metallo o legno è diversa da un oggetto di tessuto o di gomma o, ancora, le deformazioni muscolari umane sono diverse da quelle animali.

Nel caso di un personaggio umanoide o animale, che dovrà poi essere sottoposto ad animazioni di tipo realistico, si preferisce la combinazione del Bind Method “Geodesic Voxel”, unito allo Skinning “Dual quaternion”. Questi due metodi insieme conferiscono una buona deformazione nei punti critici, morbidezza e un errore relativamente basso nella distribuzione dei pesi. Il grande svantaggio risiede nella mole di calcoli che la macchina deve effettuare, con conseguente perdita di prestazioni.

Se, invece, si sta lavorando con oggetti di tipo meccanico, quali robot o autoveicoli, si preferisce utilizzare il Bind Method “Closest Distance” unito allo Skinning “Classic linear” con influenza massima a “1”, questo fa si che i pesi dei joint si distribuiscano in maniera netta e lineare, evitando deformazioni morbide nelle giunture di tipo meccanico. Questa combinazione è molto leggera a livello computazionale.

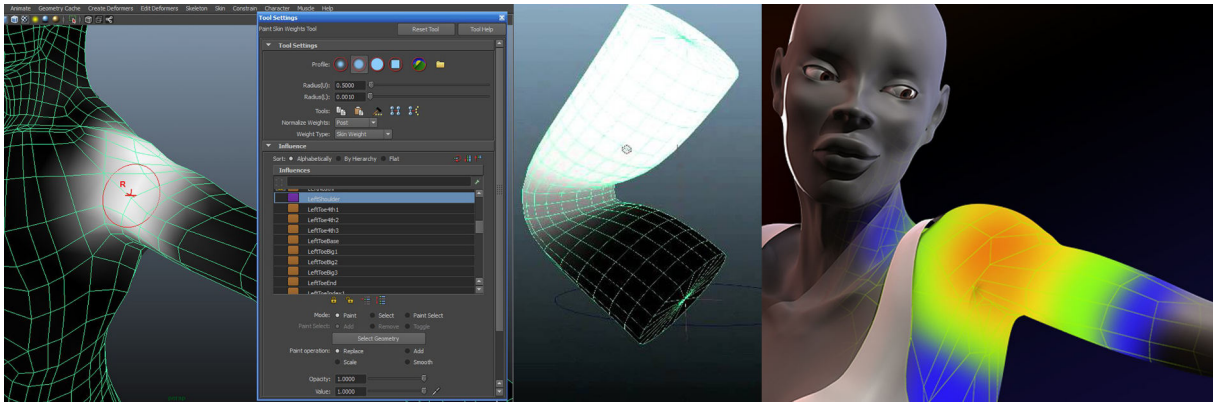
4.4.2 Paint Skin Weights – Modificare le pesature

Dopo aver applicato il “bind skin” alla mesh, gli oggetti deformabili vengono chiamati “smooth skin object” e i punti di questi oggetti sono definiti “smooth point”.

Per ognuno di questi “smooth point” Maya assegna un peso (tra 0 e 1) per ogni joint che influenza il punto della mesh. Modificare i valori del peso per ogni “smooth point” permette di controllare meglio la deformazione dello “smooth skin object”.

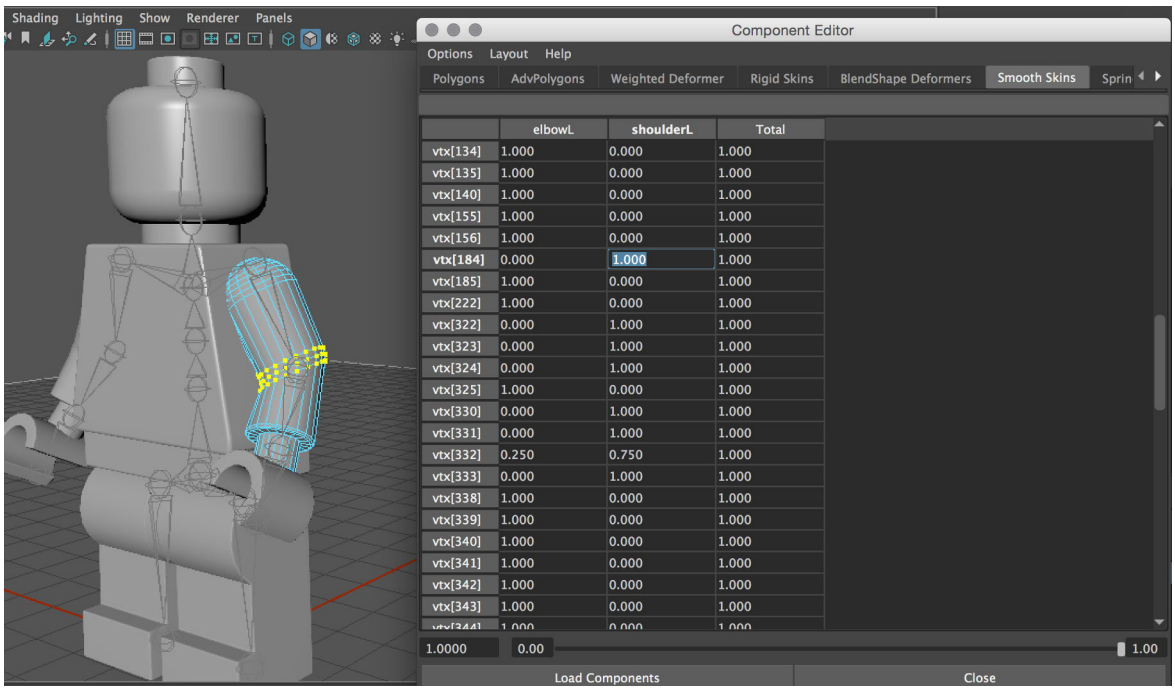
Di default, i joint più vicini agli smooth point hanno influenza maggiore rispetto ai joint più lontani.
Quali altri joint hanno più o meno influenza dipende da come vengono settati il bind method e lo skinning method.
Successivamente, si può modificare il risultato del bind skin utilizzando l'apposito strumento pennello “paint skin weights” che permette di andare a colorare le zone di interesse.

Si è detto che ogni joint va ad influenzare i punti della mesh con un peso tra 0 e 1.
Questo sta a significare che quando l'influenza è settata a 0, il joint non darà alcun contributo alla deformazione di quei punti; viceversa, se l'influenza è impostata a “1” il joint darà contributo massimo ai punti.



Il tool “paint skin weigths” permette di visualizzare le pesature colorando di *bianco* le zone con influenza massima (1) e di *nero* le zone con influenza minima (0). Le scale di grigio andranno a descrivere le influenze intermedie. Questi valori possono essere modificati andando ad agire sui punti tramite un pennello e aggiungendo o togliendo bianco ai singoli smooth point. La mesh si adatterà al nuovo tipo di deformazione.
E' anche possibile abilitare un color ramp per avere una visualizzazione a colori che permette di individuare a colpo d'occhio le zone “calde” ad alta influenza (rappresentate in rosso) e le zone “fredde” a bassa influenza (rappresentate in blu).

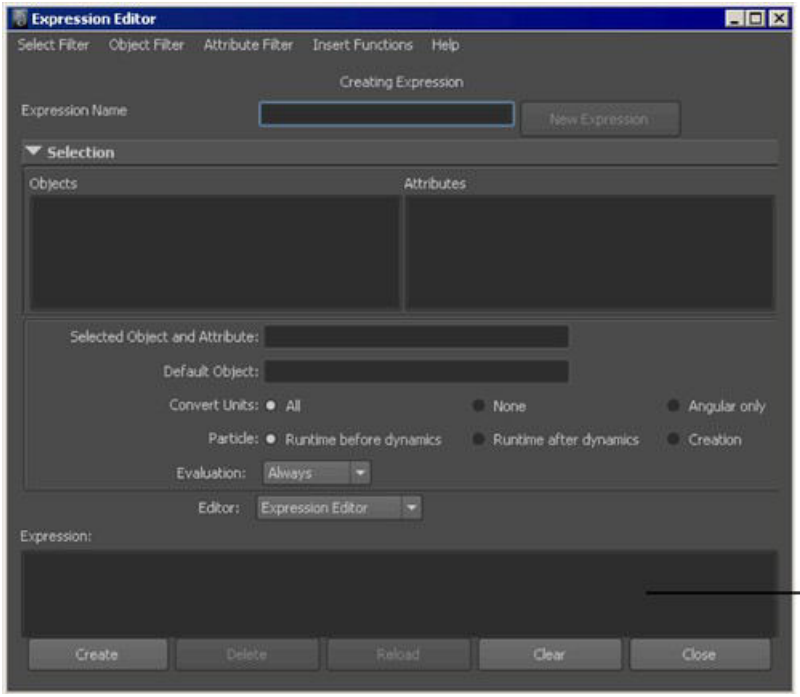
Il “paint skin weights” non è l'unico tool che permette la modifica delle pesature, ma ne esistono di più precisi. Ad esempio lo strumento “hammer skin weights” permette di uniformare la pesatura selezionando gruppi di punti sulla mesh, oppure il “component editor” che, tramite una tabulazione dei vertici, permette di andare ad aggiungere o togliere influenza da parte dei singoli joint modificando i valori di smooth associati ai punti.



5. Programmare per il Rigging - Esempi

Maya offre varie vie per implementare e automatizzare funzioni, tra cui: Node Editor, Script Editor (MEL o Python), Expression Editor.

5.1 Expression Editor:



Questo tool permette di passare istruzioni a Maya e di collegare alcune espressioni matematiche ad oggetti della scena. Queste espressioni sono composte da equazioni matematiche e/o da dichiarazioni condizionali. L'idea è quella di gestire attributi che si vogliono cambiare randomicamente o incrementare nel tempo.

L'espressione andrà a sincronizzare specifici attributi di un oggetto a quelli di un altro, in modo tale che al variare dei primi cambino gli altri.
Ad esempio, si potrebbe sincronizzare la rotazione della ruota di una moto alla traslazione del telaio tramite la semplice espressione:

rotazione = (traslazione / (6.283 * raggio)) * 360;

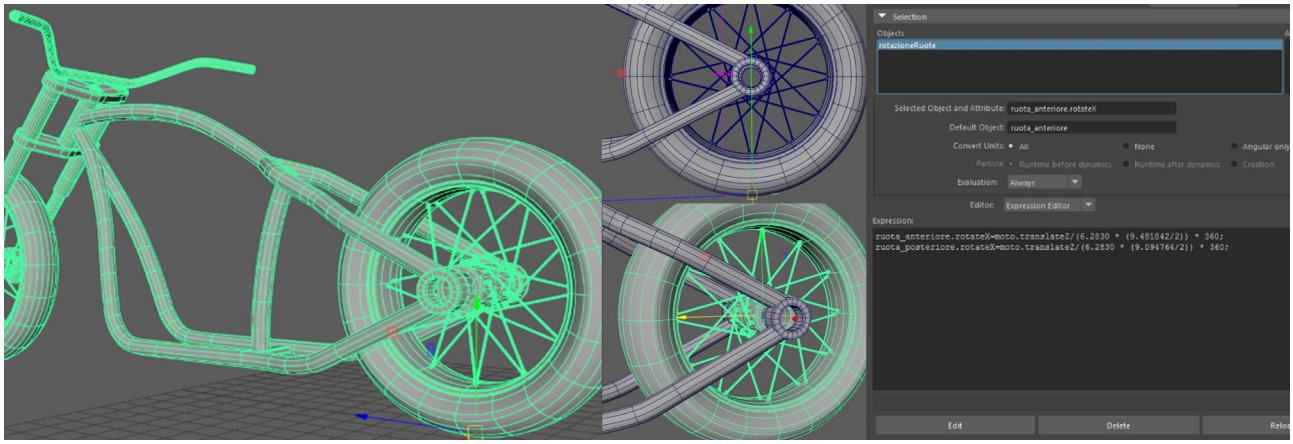
in modo che la ruota giri automaticamente quando il telaio viene spostato.

In questo processo è necessario l'ausilio di un tool di misurazione per poter ricavare il raggio della ruota.

Nell'expression editor andrà settata l'espressione:

ruota.rotateX=(telaio.translateZ/(6.2830*raggio)) * 360;

Particolare attenzione va prestata ai pivot degli oggetti, che devono essere allineati e conformi nei punti che faranno da perno, giuntura o centro di rotazione, a seconda dei casi.

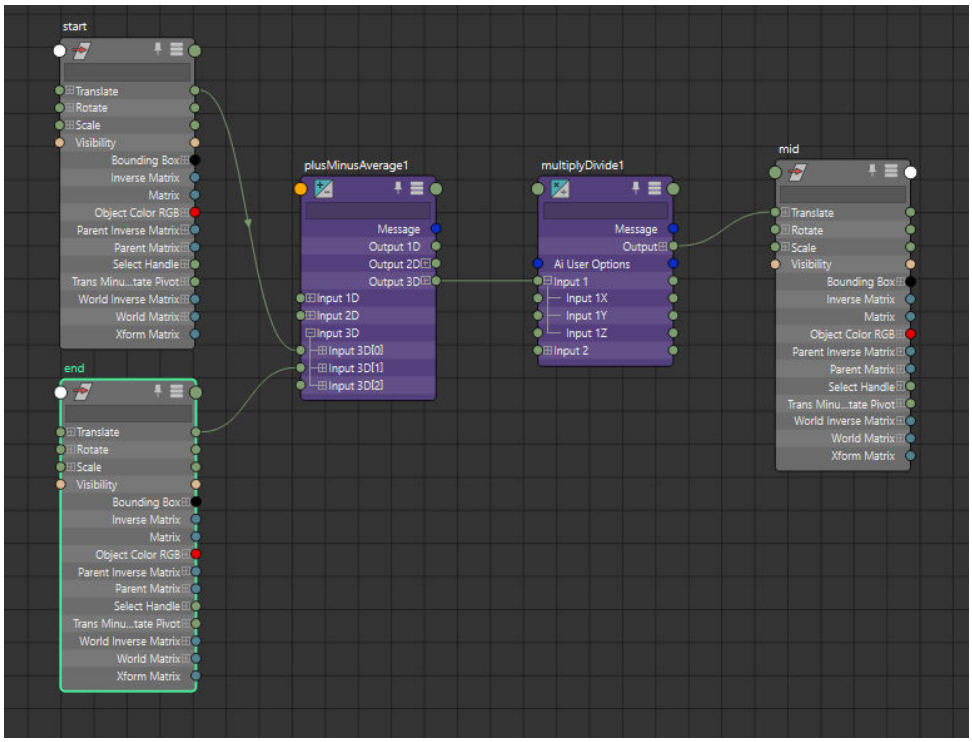


5.2 Node Editor:

Rappresenta uno schema modificabile del grafico delle dipendenze, visualizzando i nodi e le connessioni tra i loro attributi.

Permette di visualizzare, modificare e creare nuove connessioni nodali.

Lo script nodale di seguito, ad esempio, mostra come calcolare il punto medio tra due punti: l'oggetto target si posizionerà sempre a metà tra le posizioni di altri due oggetti.



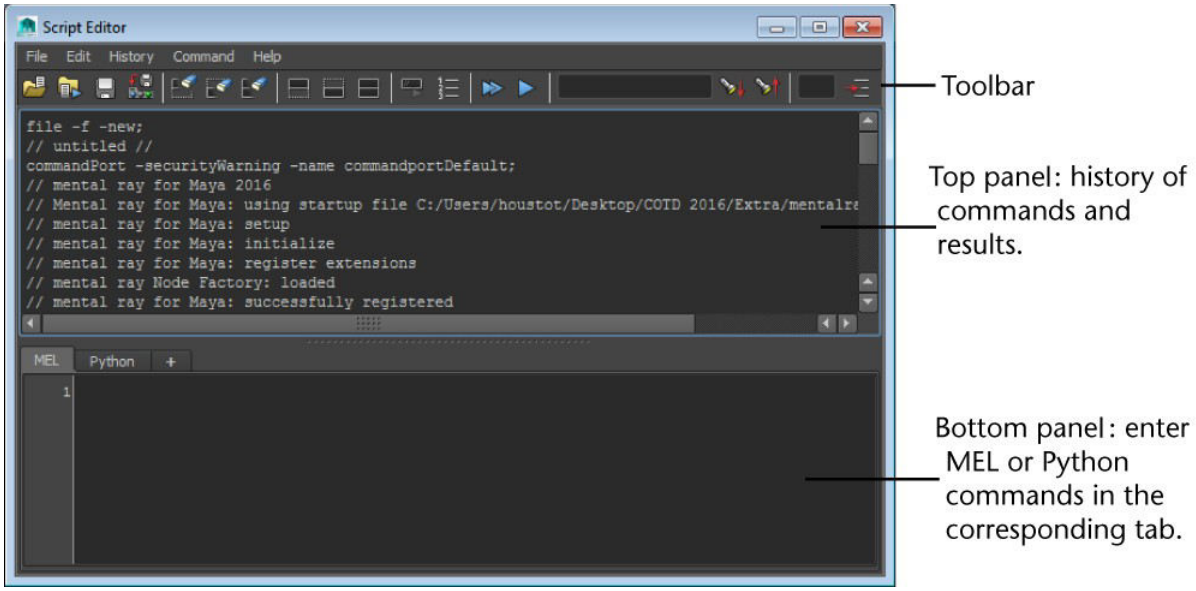
Il nodo “**plusMinusAvarage**” prende in input i valori di traslazione del punto iniziale e finale e ne restituisce in output la somma.

In seguito, il nodo “**multiplyDivide**” prende come input la somma, la divide per due e restituisce il valore medio.

L'output così ottenuto farà da input per la traslazione del target.

Esistono tantissimi nodi all'interno di Maya che permettono connessioni di ogni tipo.

5.3 Script Editor:



Lo script editor permette di creare schede sia in MEL (Maya Embedded Language) che in Python, compatibili con il relativo linguaggio di programmazione usato.

MEL è un linguaggio di programmazione orientato agli oggetti “che sta nel cuore di Maya”, ossia tutto Maya è scritto in MEL ed è quindi la via più facile da utilizzare per modificare a piacimento l'interfaccia del programma ed implementare nuovi tool.
La conoscenza di MEL implica una conoscenza profonda di Maya.

Il grande vantaggio di MEL risiede nel fatto che ogni azione che si compie in Maya viene registrata e scritta nel Top Panel, così da poterla reperire rapidamente e riutilizzare nel pannello di scripting facilitando e velocizzando la scrittura dei programmi.

MEL permette quindi di bypassare l'interfaccia grafica del programma creando direttamente via script procedure per modellare, animare e renderizzare.

Python è un linguaggio di programmazione ad alto livello utilizzabile anche al di fuori di Maya, su altri programmi o su altri editor. I suoi punti di forza sono la semplicità di sintassi, flessibilità e dinamicità. È stato progettato in modo da essere facilmente leggibile e ha

pochi costrutti sintattici rispetto ad altri linguaggi strutturati.
Per esempio, ha solo un costrutto condizionale "**if...else**" e due forme cicliche: "**for**", che scorre gli elementi di una lista o su un iteratore; e "**while**", che scorre fin tanto che l'espressione booleana indicata risulti vera.

Un aspetto particolare di Python è il metodo che usa per delimitare i blocchi di programma: indentazione.
Solitamente, i blocchi vengono separati tramite l'uso di parentesi {} o parole chiave; Python invece usa un suo sistema di indentazione per nidificare i blocchi.
Questo metodo si rivela molto vantaggioso, perché conciso e aumenta la leggibilità del codice.

Per poter utilizzare Python in Maya, è necessario richiamare l'apposita libreria dei comandi MEL tramite la dicitura:

"from maya import cmds"

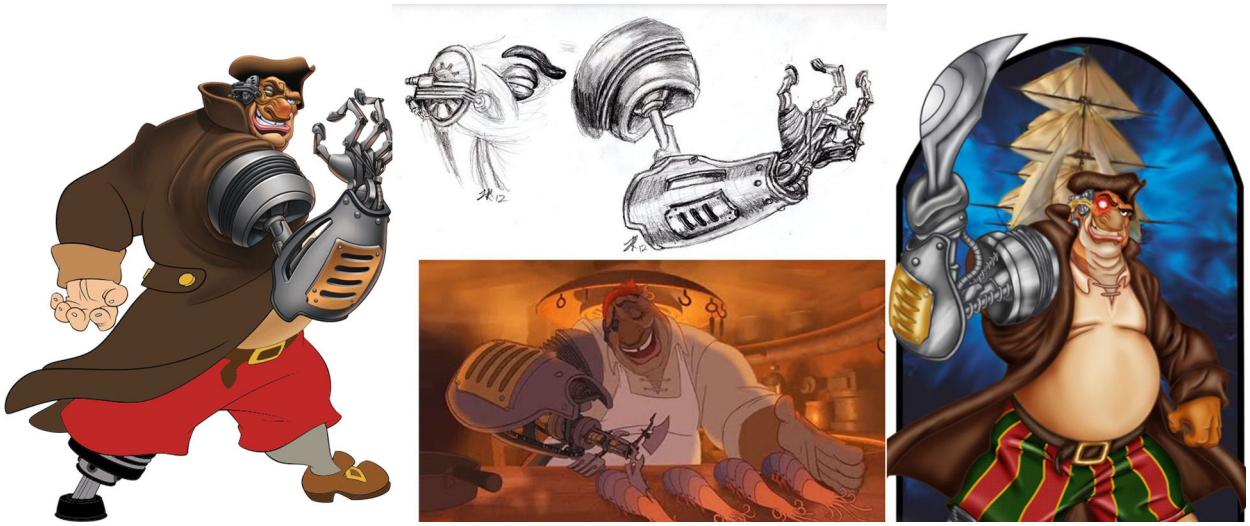
Permettendo di utilizzare i comandi MEL, tradotti nel linguaggio semplificato di Python.

6. Rigging – Applicazioni: Progetto Personale

6.1 Concept: Long John Silver – Il Pianeta del Tesoro (Disney)

Il soggetto, Long John Silver, è il rifacimento 3D del personaggio piratesco del famoso film Disney “Il Pianeta del Tesoro” (2002).
Si tratta di un pirata con alcune parti del corpo meccaniche che lo rendono un cyborg, ossia un personaggio metà umano e metà macchina.

Nello specifico il suo braccio meccanico è pensato per potersi trasformare in oggetti multiuso a seconda delle situazioni. Nel famoso cartone animato, John Silver, cuoco di bordo, utilizza il suo braccio per cucinare trasformandolo in utensili da cucina quali coltello, forbici o posate; inoltre, nelle situazioni di pericolo, può trasformarlo in un'arma a sua scelta tra cannone o sciabola.



La scelta del soggetto è dovuta alle sue particolari caratteristiche anatomiche che permettono uno studio diversificato del rig a seconda della parte del corpo presa in esame.
Il braccio e la gamba robotici permettono applicazioni di tipo meccanico, con automatizzazioni di sistema.
La grande pancia rotonda permette lo studio di deformazioni morbide e sinuose, senza perdita di volume.
La corporatura robusta e muscolosa permette lo studio di deformazioni muscolari.

Partendo dall'idea del film di animazione, si è studiato il modello 3D.

6.2 Preparazione del Modello:

Modello poligonale high poly con 2.475.789 poligoni.
Corpo: Organic Model
Braccio robotico: Hard Surface Model

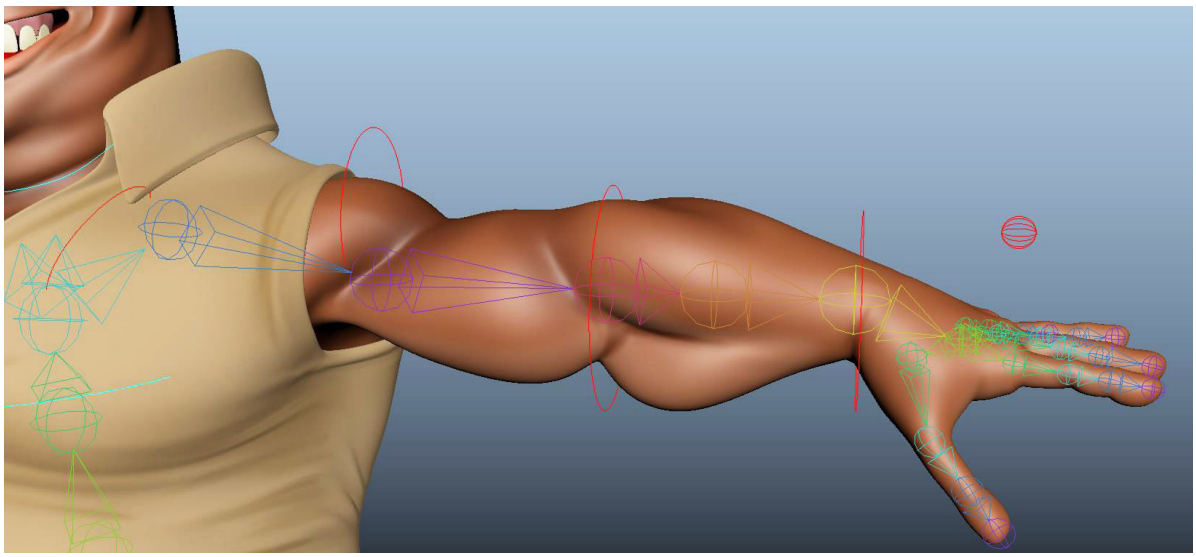
L'alto numero di poligoni conferisce al modello grande dettaglio e risulta particolareggiato in ogni sua parte, mantenendo comunque uno stile cartoon.

La topologia del corpo segue correttamente l'anatomia umana e le linee muscolari con elevata densità nei punti di maggiore deformazione, permettendo un calcolo più preciso delle pesature in fase di rig.

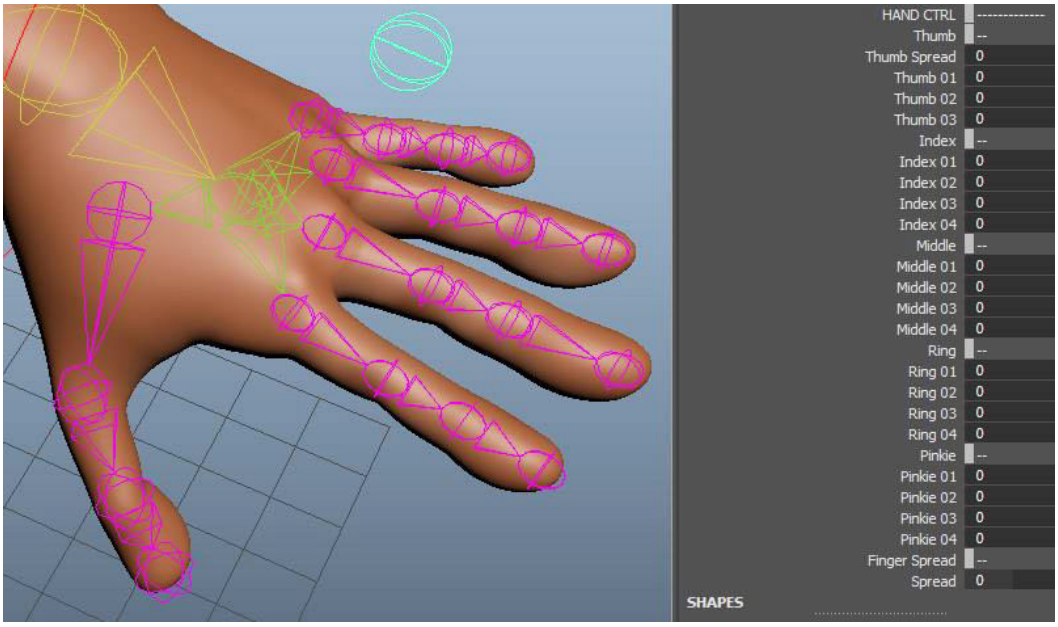
6.3 Rigging del personaggio:

Per creare lo scheletro del modello, i joint sono stati posizionati seguendo la normale anatomia umana con attenzione alle particolari proporzioni delle articolazioni. Il personaggio, infatti, ha il bacino molto basso, le gambe corte, la pancia prominente e il braccio umano lungo e muscoloso.
A capo della gerarchia dei joint è il “pelvis”, punto di congiunzione tra le catene delle gambe e della colonna vertebrale, baricentro del personaggio e radice (root) di tutto lo scheletro.

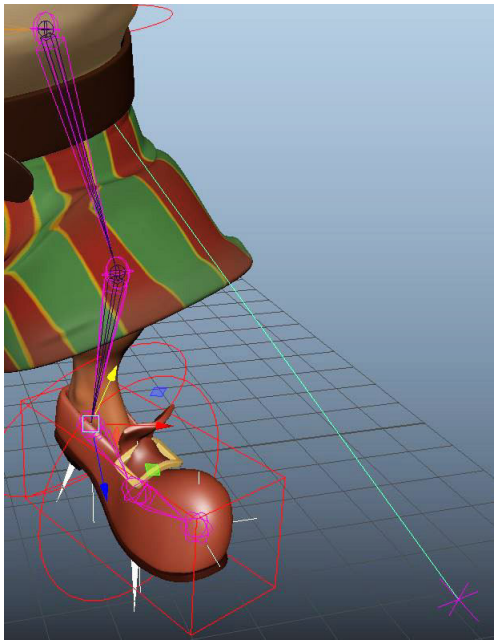
Per quanto riguarda il rig del braccio umano, si è seguito un approccio Forward Kinematics (FK) così da poter gestire meglio le rotazioni delle singole articolazioni a partire dalla clavicola fino ad arrivare al polso.



Per la mano si è creato un unico controllo a cui sono stati aggiunti gli attributi di rotazione di ogni falange permettendo la gestione delle dita singolarmente.

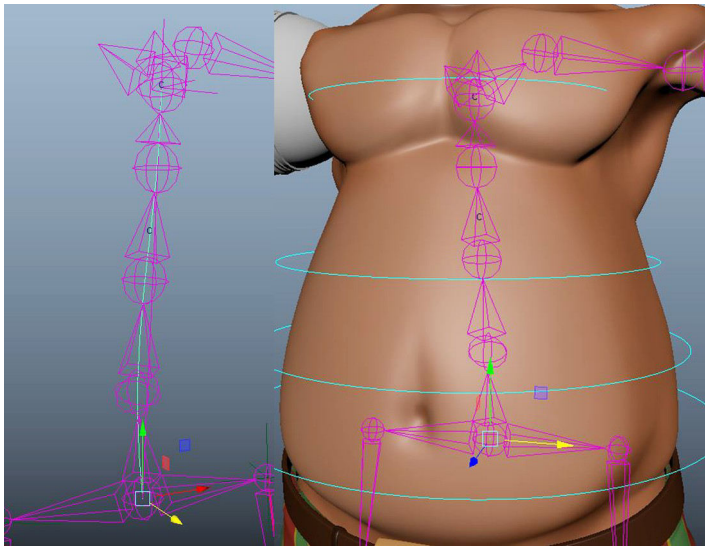


Per le gambe si è seguito un approccio Inverse Kinematics (IK) così che con un unico controllo sul piede si può gestire la posizione di tutta la gamba e, con l'aggiunta di alcuni attributi, la rotazione del tallone e della punta del piede.

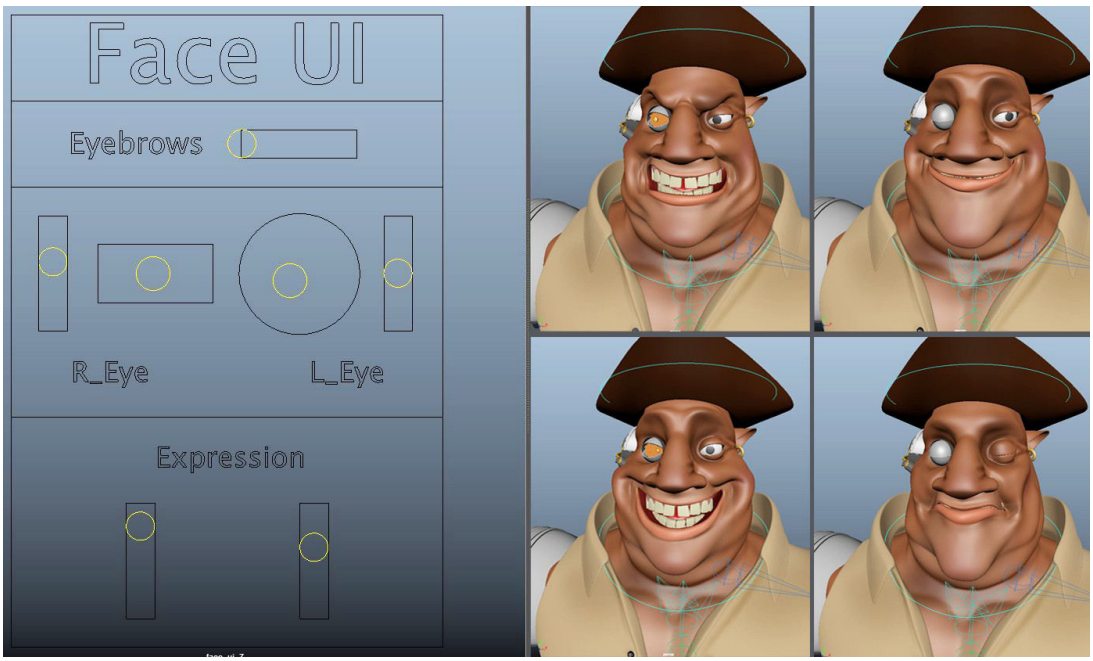


Per la spina dorsale si è fatto uso di un particolare tool del rig in IK, chiamato "IK Spline Handle", tramite cui viene creata una curva Nurbs morbida passante tra i joint della catena che saranno legati ad essa e alle sue trasformazioni.
Questo consente di gestire i joint tramite la curva e permette di avere deformazioni sinuose ad “S”.

La schiena così riggata risulta molto flessibile. Inoltre, la pancia prominente del personaggio non sarà soggetta a deformazioni pesanti, ma seguirà l'andamento della spline senza perdita di volume.

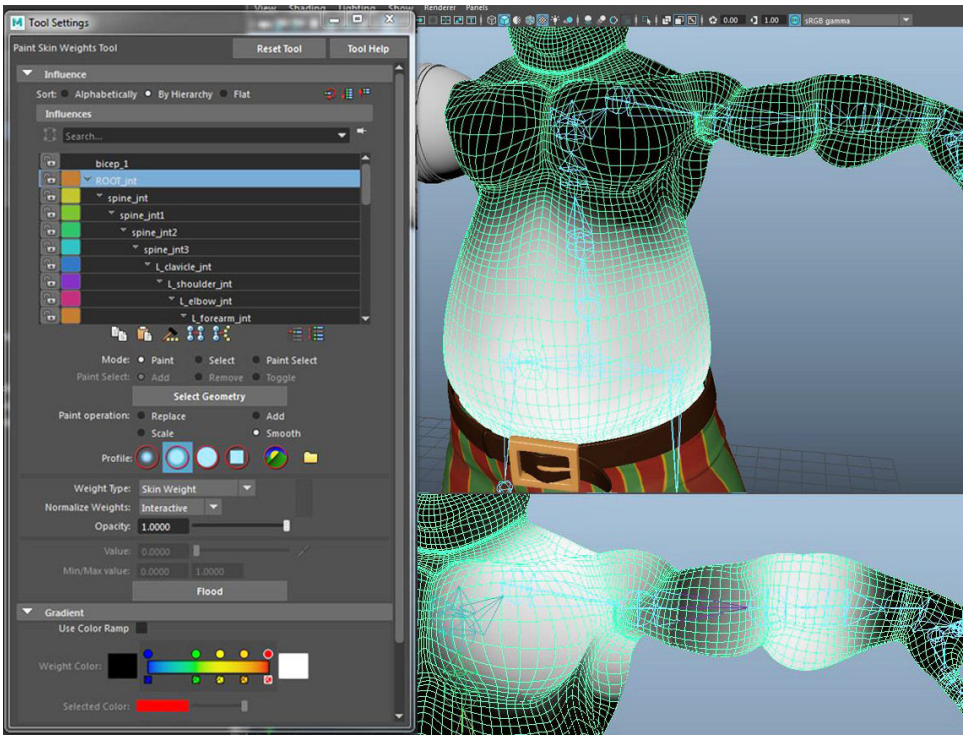


Le espressioni facciali sono invece gestite da un pannello esterno appositamente settato. Si è creato un sistema di AimConstraint per la pupilla dell'occhio umano, mentre per quello meccanico è stato creato un sistema tramite Node Editor per permettere alla pupilla di traslare e alle palpebre meccaniche di chiudersi: particolare attenzione va prestata ai pivot degli oggetti, che dovranno avere lo stesso centro per consentire una giusta chiusura delle palpebre.
Il cambio di espressione viene gestito da appositi controlli collegati ai blendshape della mesh: si tratta di particolari varianti del modello del viso, che mantengono la stessa topologia del viso di partenza.



Dopo aver impostato correttamente tutto lo scheletro, si è scelto il giusto settaggio del Bind Skin.
Partendo dalla root dello scheletro e procedendo in gerarchia (Joint Hierarchy) si è utilizzato il metodo Geodesic Voxel unito allo skinning Dual Quaternion, con una risoluzione a 512 e influenza massima a 4 joint.

Grazie alla buona topologia del modello, il calcolo dei voxel e della distanza geodetica tra essi e i joint risulta essere quasi ottimale, quindi questa combinazione di opzioni permette alle influenze di distribuirsi in maniera ideale con un errore relativamente basso. È stato necessario andare ad agire sulle pesature tramite lo strumento pennello per correggere qualche errore nella distribuzione dei pesi, soprattutto sulla pancia dato che in quella zona il modello presenta una topologia con meno poligoni.

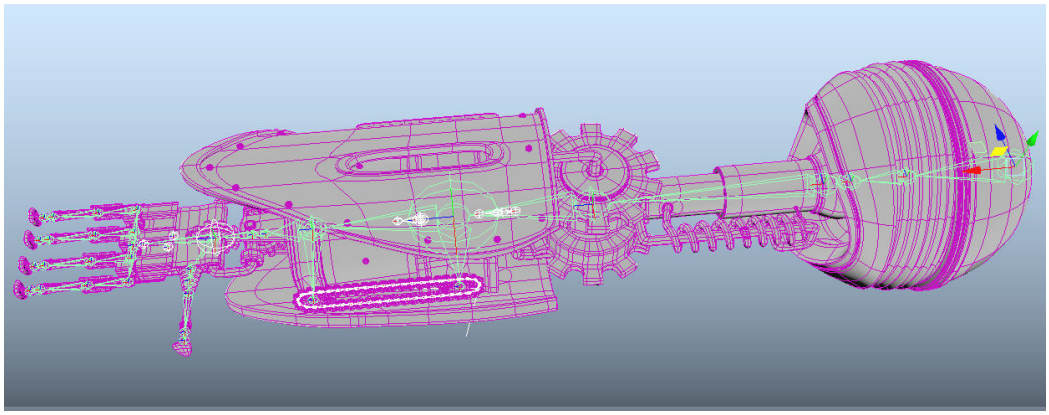


Infine, per il rig del braccio meccanico è necessario fare un discorso a parte.

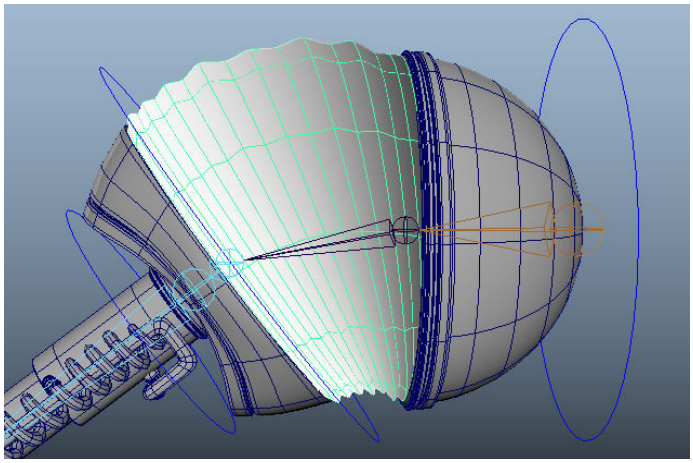
Per prima cosa è stata creata la catena di joint base, partendo dalla stessa anatomia del braccio umano:

Clavicle – Shoulder – Elbow – Forearm – Wrist - Hand - Fingers

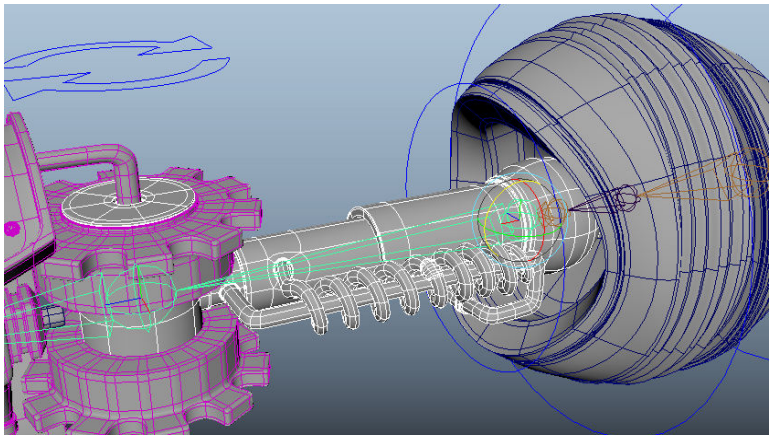
Dopodiché sono state necessarie delle catene aggiuntive per poterne meccanizzare il funzionamento.



La molla che sostituisce la spalla deve simulare le deformazioni di un oggetto di gomma, quindi sono stati inseriti due joint che influenzano l'oggetto al 50% ognuno, così da avere un allungamento e una deformazione morbidi.

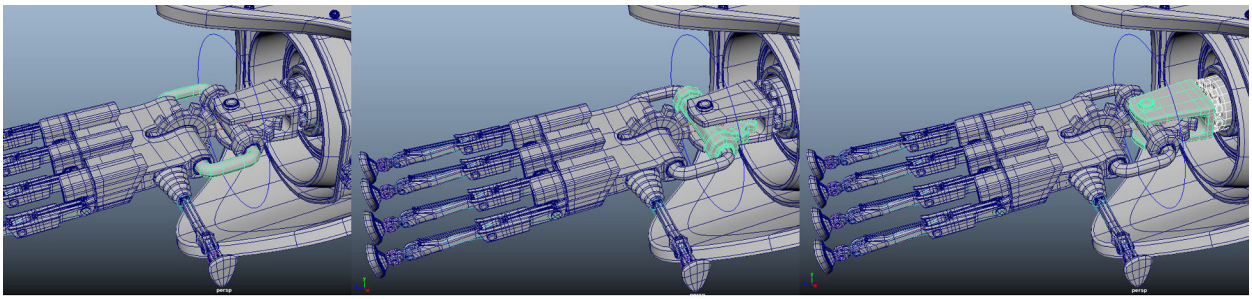


Il bicipite meccanico ha come centro di rotazione il perno interno alla spalla, ed essendo un unico pezzo metallico non subisce alcuna deformazione, quindi è sufficiente l'impiego di un solo joint con pesatura lineare settata a 1, così che avrà la massima influenza sull'oggetto, rendendolo rigido.



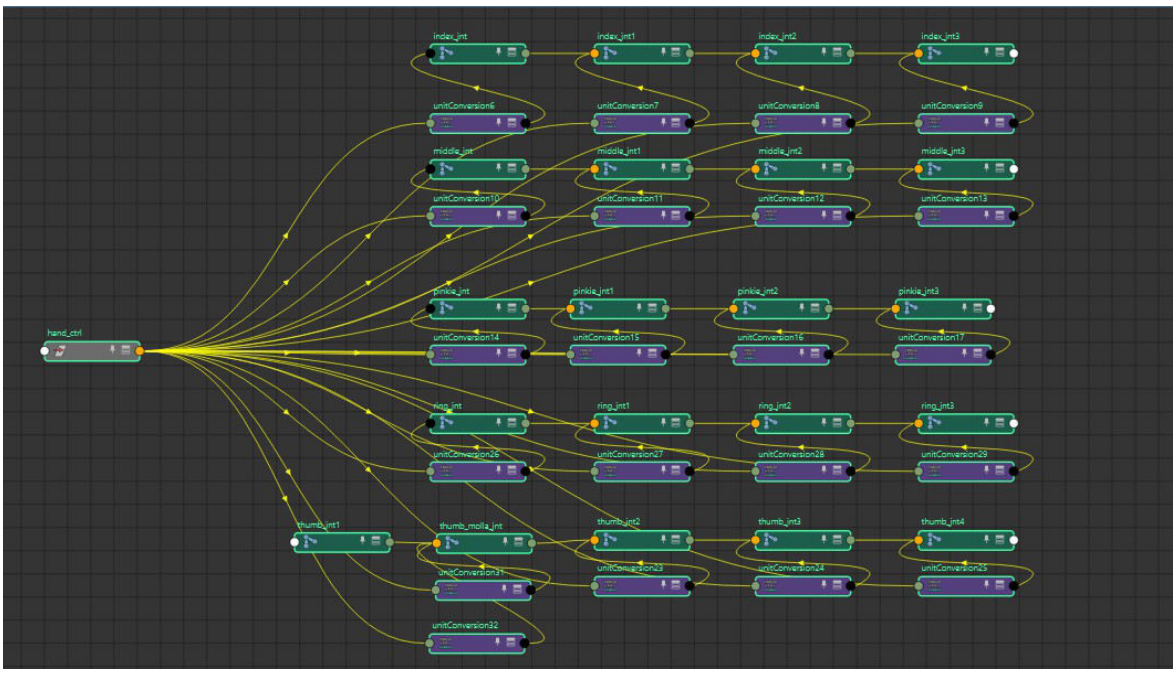
Lo stesso ragionamento e settaggio si applica anche all'ingranaggio che sostituisce il gomito, alla grata metallica attorno all'avambraccio che nasconde gli ulteriori ingranaggi di trasformazione del braccio e alle singole parti rigide di tutto il modello. E' necessario fare particolare attenzione al corretto posizionamento dei pivot, affinchè le rotazioni non generino compenetrazioni indesiderate.

I tre assi di rotazione del polso sono ripartiti su tre oggetti differenti, studiati per essere tre perni rigidi.
Ognuno di essi ha il proprio joint a cui sono stati vincolati gli attributi non necessari, lasciando libero solo l'attributo di rotazione corrispondente. Dopodiché è stato settato appositamente un unico controllo per il polso che gestisce le tre rotazioni tramite un Constraint Orient applicato separatamente su ognuna di esse.



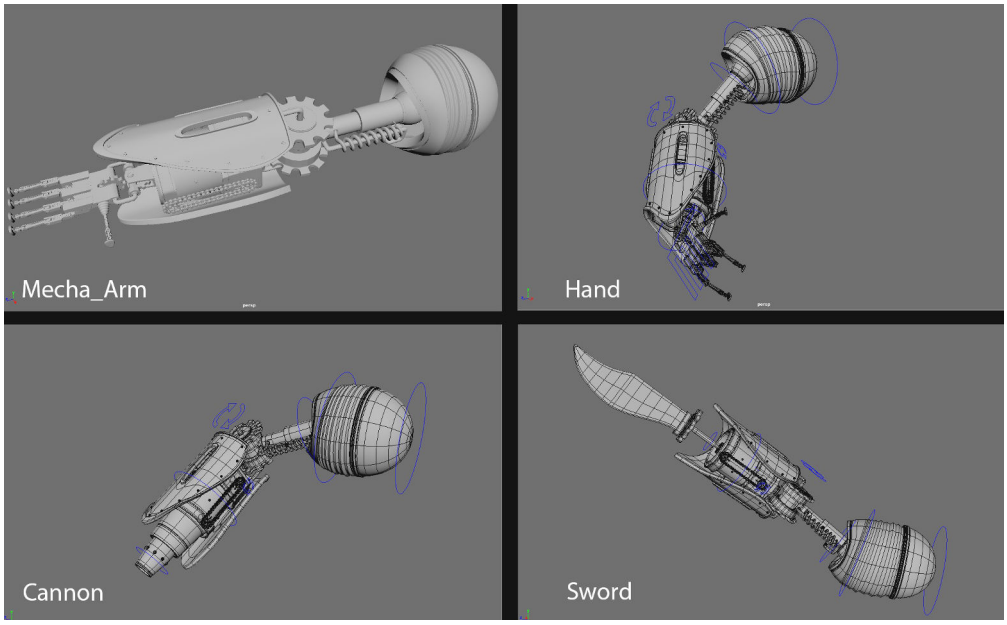
Rotazione Asse X **Rotazione Asse Y** **Rotazione Asse Z**

Un singolo controllo della mano permette il movimento di tutte le dita singolarmente. E' stato settato tramite Node Editor, aggiungendo attributi al controllo e collegandoli alle opportune rotazioni di ogni joint delle dita. Anche in questo caso si tratta di deformazioni rigide per cui è stato necessario un Bind Skin lineare.



6.3.1 Sistema di trasformazione del braccio meccanico:

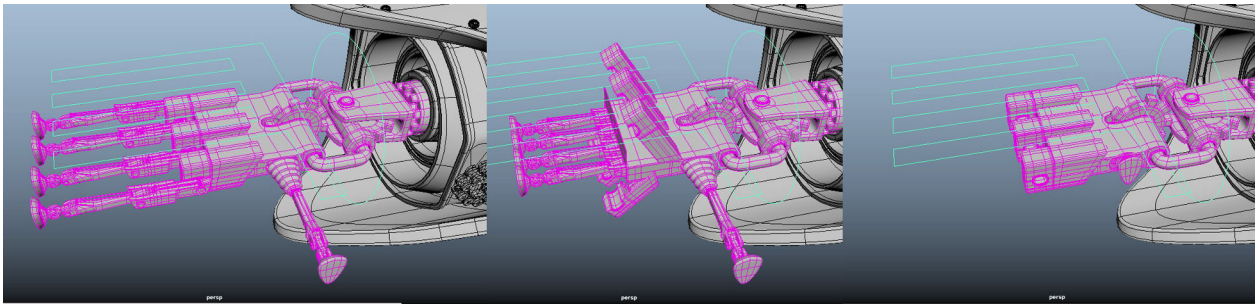
Si è studiato un sistema per far si che il braccio si possa trasformare in tre differenti forme: mano, cannone, sciabola.



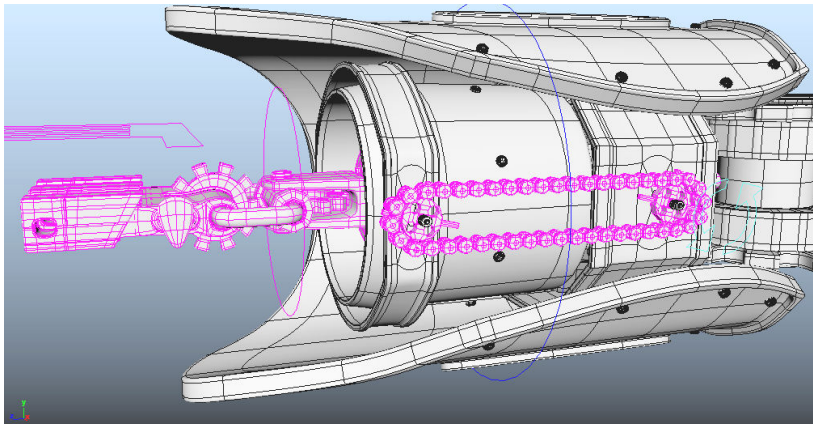
Inizialmente, il cannone fa parte di quello che è l'insieme di ingranaggi dell'avambraccio, mentre la sciabola è nascosta al suo interno.

Come prima cosa è necessario rendere logico l'inserimento della mano all'interno dell'avambraccio, con conseguente fuoriuscita del cannone, o della sciabola.

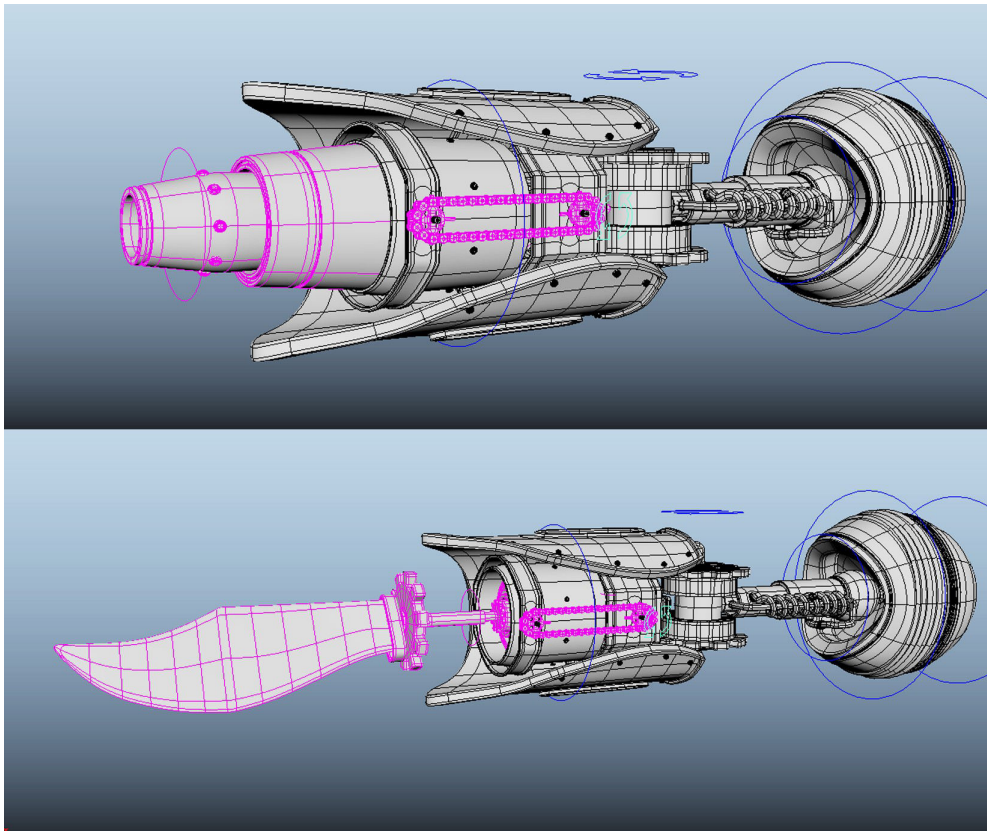
Il controllo del polso, tramite l'attributo aggiuntivo “Hand System” settato con il Node Editor, gestisce la rotazione dell'ingranaggio della mano e fa sì che le dita si ritraggano all'interno del palmo.
A questo punto, è possibile far scorrere l'intero sistema della mano all'interno della grata dell'avambraccio, facendo uscire il cannone o la spada.



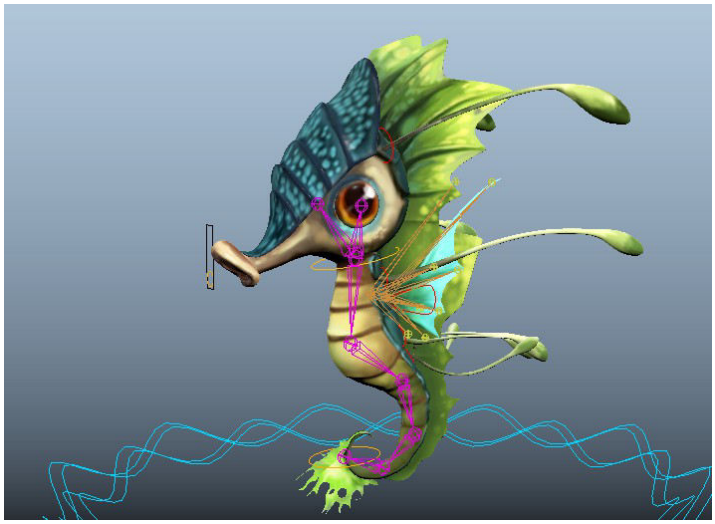
Un unico controllo, “Weapon System”, posto in prossimità della catenella dell'avambraccio, gestisce l'intero sistema di trasformazione tramite Node Editor:



Modificando il valore del Weapon System, sarà possibile far ruotare la catenella innescando il meccanismo di trasformazione. La rotazione della catena e degli ingranaggi ad essa connessi influisce sulla traslazione delle parti del cannone o della sciabola: ruotando la catena in senso antiorario, il blocco della mano entra all'interno dell'avambraccio e contemporaneamente le tre parti di cui è composto il cannone fuoriescono incastrandosi l'una con l'altra; invece, ruotando la catena in senso orario, mentre il blocco della mano rientra, si ha la fuoriuscita della sciabola.
E' poi possibile gestire alcuni parametri di rotazione sia della sciabola che del cannone tramite appositi controlli.



6.4 Rig in dinamica di una creatura marina:



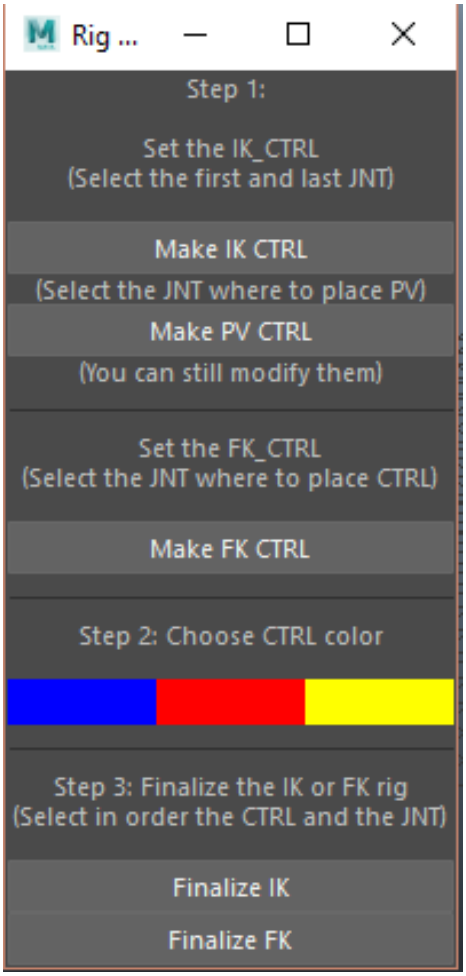
Si prende ora in esame una creatura simile ad un drago marino.
Il rig di questo personaggio è studiato quasi interamente in dinamica, ossia le deformazioni del suo corpo fluttuante sono rese automatiche tramite una simulazione di forza di gravità.

Il corpo del drago marino ha un'unica catena di joint che influenza la mesh tramite l'algoritmo Geodesic Voxel con una distribuzione morbida dei pesi; si tratta di un rig in IK, così che tramite lo spostamento e allungamento della coda, è possibile lo spostamento e l'allungamento di tutto il corpo.

Quando viene applicato il rig in dinamica, tramite appositi tool di “nCloth” o “nHair”, si genera un Nucleo all'interno della scena dentro il quale è possibile modificare una moltitudine di parametri, tra cui: gravità, densità dell'aria, velocità e direzione del vento e collisioni.

Le piccole antenne che circondano il suo corpo e la cresta hanno un settaggio tale da renderle fluttuanti come se immerse nell'acqua e il calcolo delle loro deformazioni è influenzato dallo spostamento dell'intero corpo.
Per le ali del drago, si sono create due apposite catene di joint e la rotazione della root va ad influenzare il loro battito. La dinamica applicata ai tessuti delle ali rende sinuoso l'effetto.

6.5 Script in Python: Autorig



Per risparmiare tempo e fatica durante la creazione e il settaggio del rig base è di grande utilità la creazione di tool di Autorig atti ad automatizzare procedure sempre uguali.
È quindi possibile rendere automatico il processo di creazione dei controlli in FK e IK.

Tramite uno script in Python si è creata una semplice interfaccia utente contenente alcune opzioni di base quali la scelta del controllo da voler utilizzare, la scelta del colore da volergli attribuire e il tipo di rig da voler implementare (FK o IK).

Per prima cosa è necessario selezionare i joint che dovranno poi essere gestiti dai controlli tramite Constraint. Dopodiché, cliccando sul relativo pulsante, una semplice procedura "*crea controllo*", contenente la stringa di creazione di una curva Nurbs, andrà a creare il controllo nella posizione dei joint selezionati avendo anche cura di allineare i pivot.
Quando vengono creati i controlli è ancora possibile modificarli nella forma e nella dimensione, assicurandosi di mantenerne inalterati gli attributi.

Una volta ottenuti tutti i controlli desiderati, è necessario selezionarli insieme ai relativi joint e cliccare sul pulsante "*Finalizza FK*" (o IK): ad ogni controllo andrà applicato un ConstraintOrient nel caso del rig in FK o un IK_Handle con relativo ConstraintPoint nel caso di rig in IK.
È inoltre possibile scegliere il colore dei controlli tra quelli standardizzati, ossia il rosso per i controlli di sinistra, il blu per quelli di destra e il giallo per i controlli globali e centrali.

Questa piccola interfaccia utente permette quindi di velocizzare e semplificare tutto il processo di creazione del rig base.

7. Conclusioni:

Un Character 3D nasce e prende vita tramite un susseguirsi di passaggi che vanno a formare la pipeline di produzione.

La fase di Rigging è, forse, la parte più tecnica dell'intera pipeline e rappresenta l'anello della catena tra la parte di creazione statica (modellazione e texturing) e la parte di animazione. E' quindi un passaggio fondamentale per far s' che un personaggio prenda vita e possa interpretare al meglio il suo ruolo all'interno del prodotto finale, che sia esso un film, una serie tv, un videogioco o uno spot pubblicitario.

Il Rigging fa anche parte di quello che si può definire reparto di "ricerca e sviluppo", perché si occupa di risolvere problemi tecnici tramite la creazione di tool appositi e tramite lo studio sempre più approfondito di particolari sistemi di automatizzazione, atti a velocizzare le successive fasi di produzione. Inoltre, la ricerca nel campo sta portando avanti tecniche all'avanguardia per il real time, consentendo così di implementare sistemi di motion capture istantanei.

Bibliografia:

The Art of Walt Disney – From Mickey Mouse to the Magic Kingdoms and Beyond – Christopher Finch – Abrams, New York (2011)

The Art of Rigging – Vol. II – Kiaran Ritchie, Oleg Alexander, Karim Biri - CGToolKit (2006)

Maya Python for Games and Film – Adam Mechtley, Ryan Trowbridge – Morgan Kauffman - Taylor&Frencis Group (2011)

Rigging for Games – Eyal Assaf - Taylor&Frencis Group (2016)

Fondamenti di Computer Grafica – Primo Zingaretti – Pitagora (2004)

Sitografia:

http://www.cinemecum.it

https://www.thinkscan.it/images/tutorials/curve-superfici-nurbs.pdf

http://www.ilasmagazine.com/2016/06/29/topology/

https://www.pluralsight.com/blog/film-games/rigging-guideline-artist-whats-important-good-rig

https://www.pluralsight.com/blog/film-games/5-tips-character-rigging

https://knowledge.autodesk.com/

https://www.python.it/

http://forums.cgsociety.org/

Ringraziamenti:

Ringrazio il mio relatore Prof. Carmine Di Fiore per avermi seguito nella stesura della tesi e per la curiosità dimostrata nel mio progetto; ringrazio sentitamente il mio correlatore Prof. Andrea Felice per avermi supportata anche durante il tirocinio formativo che mi ha poi permesso di vincere il Bando Torno Subito della Regione Lazio e poter così frequentare il Master in Computer Grafica di Big Rock a Treviso, senza di lui tutto questo non sarebbe stato possibile.

Un grazie sentito va inoltre a tutti i miei amici, in particolar modo a Gian Marco Tavella che ha realizzato il bellissimo modello di John Silver; Andrea Guerreri per il consulto tecnico; Sandro Farina per il consulto psicologico; Alessio Tacchi per l'infinita pazienza nell'avermi accanto ogni giorno e, ultima ma non meno importante, la mia amica e compagna di avventure e disavventure Giuseppina Cataldo.

Ringrazio inoltre tutta la mia famiglia per avermi supportato psicologicamente ed economicamente in questo lungo e tortuoso percorso.

Grazie a Big Rock per i preziosi insegnamenti e approfondimenti della materia.

Allegato: Script di Autorig

```
#Simple_Autorig

#My autorig launchs a simple User Interface that allows to set the type
#of controls that you want to use.
#After that you can set the rig type in FK or IK and choose color of the controls.


from maya import cmds


#Creating the UI:
def gui():
    NewWindow = 'Auto_Rig'
    if cmds.window(NewWindow, exists=True):
        cmds.deleteUI (NewWindow)
    myWindow = cmds.window(NewWindow, title = "Rig Maker" , widthHeight = (195,400), rtf =
True)
    main_layout = cmds.columnLayout( adjustableColumn=True )


    #Step 1: Choose the ctrl for IK rig
    cmds.text(label='Step 1: ')
    cmds.text(label='')
    cmds.text(label='Set the IK_CTRL')
    cmds.text(label='(Select the first and last JNT)')
    cmds.text(label='')
    cmds.button('ik_ctrl_btn', label="Make IK CTRL" , w=195, command ='Make_IK_ctrl()')
    cmds.text(label='(Select the JNT where to place PV)')
    cmds.button('pv_ctrl_btn', label="Make PV CTRL" , w=195, command ='Make_PV_ctrl()')
    cmds.text(label='(You can still modify them)')
    cmds.separator('IK_sep', w=195, h=20)

    #Step 2: Choose the ctrl for the FK rig
    cmds.text(label='Set the FK_CTRL')
    cmds.text(label='(Select the JNT where to place CTRL)')
    cmds.text(label='')
    cmds.button('fk_ctrl_btn', label="Make FK CTRL" , w=195, command='Make_FK_ctrl()')
    cmds.separator('FK_sep', w=195, h=20)

    #Step 3: Choosing color for the ctrl
    cmds.text(label='Step 2: Choose CTRL color')
    cmds.text(label='')
    cmds.gridLayout(nr=1, nc=3, cellWidthHeight=(65,20))
    cmds.iconTextButton('Blue_Btn', bgc=(0,0,1), command='B_color_ctrl()')
    cmds.iconTextButton('Red_Btn', bgc=(1,0,0), command='R_color_ctrl()')
```

```
    cmds.iconTextButton('Yellow_Btn', bgc=(1,1,0),command='Y_color_ctrl()')
    cmds.setParent(main_layout)
    cmds.separator('color_sep', w=195, h=20)

    #Step 4: Finalizing the IK or FK rig
    cmds.text(label='Step 3: Finalize the IK or FK rig')
    cmds.text(label='(Select in order the CTRL and the JNT)')
    cmds.text(label='')
    cmds.button('IK_btn', label="Finalize IK" , w=195,command='finalize_IK()')
    cmds.button('FK_btn', label="Finalize FK" , w=195, command='finalize_FK()')

    cmds.showWindow()


gui()


def Make_FK_ctrl():
    #declaration of variables
    jnt_list = 0
    name_grp = "_GRP"

    #Get selected joints
    jnt_list = cmds.ls("*_JNT", sl=True, type='joint')

    #check if in the list there is at least a joint
    if jnt_list == 0:
        print "No joint selected"
    else:
        #For each joint
        for obj in jnt_list:
                name_ctrl = obj.replace("_JNT", "_CTRL")
                #Create a nurbs circle control shape for the joint
                circle_icon = cmds.circle( nr=(1, 0, 0), c=(0, 0, 0), n=name_ctrl, r=2)
            #Create the offset group
            cmds.group(name_ctrl, n= name_ctrl+name_grp)
                #Parent the group to the joint
                tempConst = cmds.parentConstraint(obj,name_ctrl+name_grp,mo=False)
                cmds.delete(tempConst)


def Make_IK_ctrl():
    #declaration of variables
    jnt_list = 0
    name_grp = "_GRP"

    #Get selected joints
```

```
    jnt_list = cmds.ls("*_JNT", sl=True, type='joint')

    #check if in the list there is at least a joint
    if jnt_list == 0:
        print "No joint selected"
    else:
        last_jnt = jnt_list[-1]
        name_ctrl = last_jnt.replace("_JNT", "_IK_CTRL")
        #Create a nurbs square control shape for the joint
        square_icon = cmds.curve(d=1, p=[(4,4,0),(-4,4,0),(-4,-4,0),(4,-4,0),(4,4,0)], k=[0,1,2,3,4],
n=name_ctrl)
        #Create the offset group
        cmds.group(name_ctrl, n= name_ctrl+name_grp)
        #Parent the group to the joint
        tempConst = cmds.parentConstraint(last_jnt,name_ctrl+name_grp,mo=False)
        cmds.delete(tempConst)


def Make_PV_ctrl():

    #declaration of variables
    jnt_list = 0
    name_grp = "_GRP"

    #Get selected joints
    jnt_list = cmds.ls("*_JNT", sl=True, type='joint')
    #check if in the list there is at least a joint
    if jnt_list == 0:
        print "No joint selected"
    else:
        PV_jnt = jnt_list[0]
        name_ctrl = PV_jnt.replace("_JNT", "_PV_CTRL")
        #Create a locator for the joint
        cmds.spaceLocator( p=(0, 0, 0), n=name_ctrl)
        #Create the offset group
        cmds.group(name_ctrl, n= name_ctrl+name_grp)
        #Parent the group to the joint
        tempConst = cmds.parentConstraint(PV_jnt, name_ctrl+name_grp,mo=False)
        cmds.delete(tempConst)


def B_color_ctrl():

    ctrl_list = 0
    #Get selected CTRL
    ctrl_list = cmds.ls("*_CTRL", sl=True)
    #check if in the list there is at least a ctrl
```

```
    if ctrl_list == 0:
        print "No CTRL selected"
    else:
        #For each ctrl
        for obj in ctrl_list:
            cmds.color(ctrl_list, rgb=(0,0,1))

def R_color_ctrl():

    ctrl_list = 0
    #Get selected CTRL
    ctrl_list = cmds.ls("*_CTRL", sl=True)
    #check if in the list there is at least a ctrl
    if ctrl_list == 0:
        print "No CTRL selected"
    else:
        #For each ctrl
        for obj in ctrl_list:
            cmds.color(ctrl_list, rgb=(1,0,0))


def Y_color_ctrl():

    ctrl_list = 0
    #Get selected CTRL
    ctrl_list = cmds.ls("*_CTRL", sl=True)
    #check if in the list there is at least a ctrl
    if ctrl_list == 0:
        print "No CTRL selected"
    else:
        #For each ctrl
        for obj in ctrl_list:
            cmds.color(ctrl_list, rgb=(1,1,0))


def finalize_FK():

    #Select CTRL and JOINT to Constraint

    #Get selected CTRL
    ctrl_list = cmds.ls("*_CTRL", sl=True)
    #Get selected joints
    jnt_list = cmds.ls("*_JNT", sl=True, type='joint')
    #check if in the list there is at least a ctrl
    if ctrl_list == 0 and jnt_list == 0:
        print "No CTRL or JNT selected"
    else:
```

```
        n = 0
        #Do Constraint Orient for each ctrl to each joint
        for obj in ctrl_list:
            cmds.orientConstraint(obj,jnt_list[n], mo=True)
            cmds.setAttr(obj+'.tx', lock=True, keyable=False)
            cmds.setAttr(obj+'.ty', lock=True, keyable=False)
            cmds.setAttr(obj+'.tz', lock=True, keyable=False)
            cmds.setAttr(obj+'.sx', lock=True, keyable=False)
            cmds.setAttr(obj+'.sy', lock=True, keyable=False)
            cmds.setAttr(obj+'.sz', lock=True, keyable=False)
            n = n+1

    #parent the ctrl by jnt hierarchy
    n=0
    max = len(ctrl_list) -1
    while(n < max):
        ctrl_grp = cmds.listRelatives(ctrl_list[n+1], p=True)
        cmds.parent(ctrl_grp[0], ctrl_list[n])
        n=n+1

def finalize_IK():

    #Get selected joints
    jnt_list = cmds.ls("*_JNT", sl=True, type='joint')
    name_handle = jnt_list[-1].replace("_JNT", "")
    cmds.ikHandle(sj = jnt_list[0], ee=jnt_list[-1], n =name_handle+"_IK_handle")
    cmds.select(clear=True)

    cmds.pointConstraint("*_IK_CTRL",name_handle+"_IK_handle", mo=True)
    cmds.poleVectorConstraint("*_PV_CTRL",name_handle+"_IK_handle" )

    #lock the attributes ok IK_CTRL and PV_CTRL
    for obj in ['.rx', '.ry', '.rz', '.sx' ,'.sy', '.sz']:
        cmds.setAttr("*_IK_CTRL"+ obj, lock=True, keyable=False)
        cmds.setAttr("*_PV_CTRL"+ obj, lock=True, keyable=False)

    #put in order the outliner
    cmds.parent("*_PV_CTRL_GRP","*_IK_CTRL" )
    extra = cmds.group(em=True, n="EXTRA_GRP")
    cmds.parent(name_handle+"_IK_handle",extra)
    cmds.setAttr(extra+'.v', 0)

#end
```