Escola Universitària d'Enginyeria Tècnica

de Telecomunicació La Salle

Treball Final de Grau

Grau en Animació

Planificació d'un Rig Modular

Alumne

Enrique Velasco Mairal

Professor Ponent

Ignasi Duelo Fandos

ACTA DE L'EXAMEN
DEL TREBALL FINAL DE GRAU

Reunit el Tribunal qualificador en el dia de la data, l'alumne

Enrique Velasco Mairal

va exposar el seu Treball de Final de Grau, el qual va tractar sobre el tema següent:

Planificació d'un Rig Modular

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Abstracte

Català:

La planificació i la realització d'un rig envolta molts processos que queden ocults i no es coneixen. L'objectiu d'aquest treball és fer una visualització de tots aquests passos, analitzar-los i aprendre'ls, trobant quina és l'opció més optima per a desenvolupar els personatges per a la realització dels projectes de 4rt.

Veurem que hi ha moltes maneres de procedir i que cap, és menys vàlida que una altre. Per tant, per saber com procedir, s'han de entendre quines maneres hi ha de fer, quines són les accions i limitacions de cada personatge, quina serà la seva actitud, de quant temps es disposa i quant de temps hi ha abans de que es comenci a fer el layout. Un cop tots aquests conceptes es tenen controlats es pot dissenyar un rig a nivell del personatge que el portarà.

Tot això s'explica en detall en el document que segueix.

Espanyol:

Planificar y realizar un *rig* conlleva una serie de procesos que quedan ocultos y no se conocen. El objetivo de este trabajo es visualizar, analizar y aprender todos estos pasos, encontrando la manera más optima de llevar a cabo el desarrollo de los personajes de proyectos de 4rto.

Veremos que hay muchas maneras de proceder y ninguna es menos apta que otra. Para saber que método escoger, se tienen que entender que técnicas existen, cuáles serán las acciones del personaje en escena y cuáles serán sus limitaciones y cuanto tiempo se dispone para desarrollar el *rig* antes de empezar el *layout*. Una vez todos estos conceptos están controlados, se puede diseñar el *rig* a medida del personaje.

Todo esto se cuenta en el documento que sigue.

Anglès:

There are so many hidden processes while planning and developing a rig that remain unknown. The propose of this project to put forward, analyze and learn these hidden steps in order to find the best guess to develop the end course project assets.

There are so many ways to proceed, and none of them is less valid than another. With the propose of getting the method that fits with each character, we must analyze which actions will have the asset in each shot, and which not, and how much time do we have before layout to build a rig. Once we have handled all the variables, the rig can be planned according to the character.

This document will explain all the process.

Agraïments

M’agradaria donar les gracies a un seguit de persones sense les quals aquest treball no hauria estat possible, ja sigui pel seu guiatge, suport o experiències.

En Sergio Graña, Ona Molas, Enrique Alberola, Judit Navarro i al Sancho Albano, han estat dia a dia cuidant tots els aspectes previs i posteriors al rig perquè els personatges puguin anar a través del pipe. Agrair a les companyes de classe i de projectes, als mentors i tutors.

D’altre banda als companys de feina: Felix Balbas, Vincenzo Leombruno, Alessandro Boschian, Albert Vivó, Roure Ossó, Xavier Roca Crespo i Sara Saez Hoces, pel guiatge i per l’ajuda a l’hora de resoldre situacions dels rigs i fer-me entendre el global del procés.

També agrair a mentors exteriors: IKer J. de los Mozos per les seves experiències explicades en entrevistes i les llargues converses. I a les mentores de Ilion, Silvia Montes i Isabel Bértolo, per aportar els seus coneixements sobre com funcionar el pipe de groom i de simulació de roba.

A nivell de codi, m’agradaria agrair a en David González Cuéllar i en Vasil Shotarov, per les bases que han prestat en els sistemes de salvaguarda i lectura dels pesos dels deformadors i pel sistema de guardat i carrega de les formes dels controls.

Acrònims

ACES: Academy Color Encoding System

API: Application Programming Interface

FK: Forward Kinematics

FPS: Fotogrames per segon

GPU: Graphics Processor Unit

IDE: Integrated Development Environment

IK: Inverse Kinematics

PYCharm: IDE de programació basat en el llenguatge de programació Python

Python: Llenguatge de programació d'alt nivell

TFG: Treball Final de Grau

USD: Universal Scene Description

UV: Espai 2D de coordenades per a textures, U i V defineixen les dues components

Índex

1 Introducció

Aquest treball va enfocat al desenvolupament de la producció tècnica dels projectes de 4rt, específicament: Last in Battle, Path of Sand i World Of Lost Things.

Es contemplaran els apartats de: ***rig***, ***groom***, ***pipeline*** i **eines**. Cada un dels apartats s'explicarà el procés que s'ha seguit, definició i dificultats trobades. L'objectiu no és únicament explicar el procés, sinó que es pretén expandir les parts més teòriques dels camps mencionats.

Com a material resultant són 6 *rigs* dels quals hi ha 5 bípedes i un quadrúpede, scripts i utilitats recollides sota de la LS_MENU dins del Maya.

També s'estendrà un projecte sobre *rigs* preparats per a recollir dades de mocap, fet conjuntament amb la universitat de **Cardiff School of Art & Design**.

Tots els conceptes redactats en el document estan referits i extrets al software de 3D Autodesk Maya 2018.5.

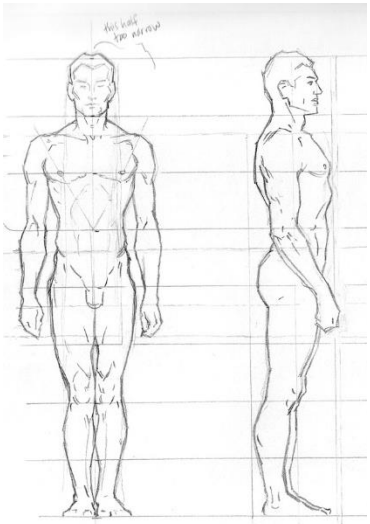
La reel anirà enfocada a la part de *rigging* que és la part a la que va més dirigida el treball i a la que se li ha dedicat més temps.

2 Concepte general

Abans de atacar l'àrea de Rigging, s'ha de veure que és, d'on ve i com ha evolucionat

2.1 Definició

El Rigging és la part de la producció del 3D i VFX que s'encarrega de **crear les estructures lògiques** de deformació del personatge. Mitjançant els recursos del programa, és capaç de proporcionar-li la habilitat de moure's "anatòmicament" al personatge.



Il·lustració 1: Esquema d'anatomia corporal d'Andrew Loomis

No fa falta que sigui anatòmicament perfecte, ni que respecti totes les normes de l'anatomia del personatge. Es busca més, que el personatge tingui *appel*. Si per fer-ho s'han de trencar amb les normes de la mecànica de la anatomia, es pot agafar la llicencia artística de fer-ho.

Un personatge que funcioni amb una estructura anatòmica correcta, tindrà unes deformacions més acurades que un que no. Entren més variables, que no solament les pertinents al *rig*, per a fer que el personatge funcioni correctament. Entren a valor variables que no son controlables al departament com el disseny, topologia, *groom* i animació.

2.2 Objectiu

Es pot definir que l'objectiu principal del *rig* és dotar al animador de controls, atributs i eines per a poder donar expressivitat al personatge.

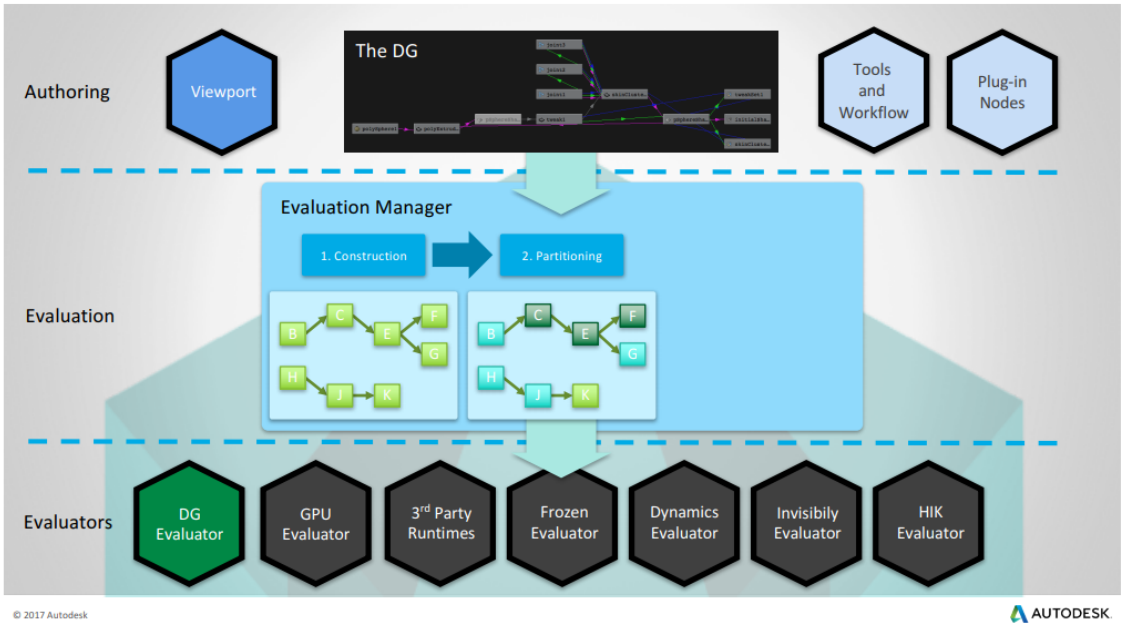
S'ha de tenir en compte que, molt provablement la geometria esculpida per l'animador en el *viewport*, no serà la que pujarà a render, sinó que pot passar per sistemes de músculs, teles, pels i retocs.

En els personatges *cartoon* s'ha de procurar que totes les deformacions siguin netes i rodones i que les arrugues serveixin per a marcar les línies d'expressió.

Una de les premisses importants, és que el rig ha de poder tenir la capacitat de poder anar a 25 *fps*. Això és problemàtic ja que per molt que s'optimitzin els personatges, tot depèn de:

- Quantitat de personatges en escena
- Funcionalitats de display del *viewport*
- Subdivisió dels objectes
- Visibilitat dels objectes
- Deformacions activades
- Avaluació del programa (DG, Paral·lel, Serial)

- Avaluació de la GPU (*override*)



Il·lustració 2: Esquema d'ordre d'avaluació dels nodes dins de Maya

2.3 Orígens

El concepte de *rigging* va trigar a sorgir en la producció, tot i què, va estar present des de els seus primers inicis, fins i tot abans de començar la producció 3D.

Es pot considerar que el *rigging* comença en les primeres animacions en *stop-motion*, quan es va veure la necessitat de poder articular els personatge, per optimitzar la producció i, així, no haver de fer cada figura per a cada fotograma.

Il·lustració 3: Armadura de l'aniamtrònic de King Kong feta per Ray Harryhausen (1933)

Poc a poc va anar evolucionant segons les necessitats de cada una de les produccions. Passant del *stop-motion*, a la articulació de robots i naus, en les primeres proves de la producció 3D.

2.4 Tipus i Tècniques

2.4.1 Segons Estètica

Nomes enfocant en el ambient del 3D, es poden extreure dues tipologies diferents.

Realista: Tota la expressivitat corporal i facial s'apropa més a la realitat. Les robes i cabells es deixen per ambients de simulació. Des de la part de *rigging*, es proporcionen sistemes per falsejar les deformacions musculars.

Cartoon: Les expressions són més extremes, s'apliquen més dràsticament sistemes que donin al animador la possibilitat de dibuixar la forma que es vulgui amb el personatge. Els conceptes de *squash* i *stretch* son més presents.

Il·lustració 4: A la dreta està el wireframe de Golum de *"El senyor dels anells"* desenvolupat per Weta Digital, fa referencia a la visualització d'una geometria realista. A la dreta està el Capità Charles T. Baker de *"Planet 51"* desenvolupat per Ilion Animation Studios, mostra quina és una de les deformacions màximes del personatge.

2.4.2 Segons rellevància

Es poden trobar:

Personatges

- **Principals** (*Hero*/A): Els que apareixen més temps a pantalla i els que tenen més rellevància a nivell narratiu. Són els que tenen uns sistemes de deformació més avançats i personalitzats.
- **Secundaris** (B): Són els que apareixen en menor freqüència però interactuen amb els principals. Tenen un set de deformacions completes però no tan extremes com els A.
- **Terciaris**: Apareixen en de fons o per omplir escena. Acostumen a ser personatges genèrics amb variacions de robes i pentinats.

Props

- **Estàtics**: Es basen en sistemes de pivotatge, ja que serviran per a que els personatges els puguin agafar o interactuar amb ells.
- **Animats**: Tenen sistemes de deformació mecànica simple.
- **Vehicles**: Tenen sistemes de interacció amb el terreny, apart de sistemes avançats de deformació mecànica.
- **Simulació dirigida**: Són sistemes on l'animador te la possibilitat de controlar el camí genèric de cada deformació però que a sobre tenen sistemes dinàmics. Als

sistemes dinàmics se'ls hi aplica camps de deformació o expressió ja que facilita la animació i la visualització final.

2.4.3 Segons càmera

Finalment, també es poden separar per la seva proximitat a càmera: depenent la distancia de càmera a personatge, es dissenyarà un sistema més complex o més simple.

2.4.4 Procediment

Per a la creació de *assets riggejats* hi ha varies maneres de procedir. Depenent de cada producció s'escull el o els mètodes més òptims per a assolir la historia. Poden ser *rigs* que estiguin basats en sistemes de:

- Varies geometries on estan modelades les deformacions on pot arribar la geometria
- *Skinclusters* pesats amb diverses capes de deformació on els *joints* són qui tenen el control
- Atributs que controlin varies deformacions siguin del tipus que siguin
- Manovelles limitades que controlin deformacions
- Deformadors de *wire* on viuen controls dinàmics
- Desplaçament de geometries pel moviment d'unes altres més simples (*ShrinkWrap*, *WrapReversed*)

Apart d'aquests mencionats hi ha d'altres que també generen deformacions interessants en els personatges. No existeix una única manera genèrica de construir un *rig*, sinó que s'ha d'adaptar a les necessitats de cada producció.

2.4.5 *Rigs* basats en *joints*

Aquest tipus de *rigs* donen una gran llibertat de moviment en els controls, ja que funcionen a partir de les transformacions d'un o varis controls que apunten a sistemes lògics que acaben en els *joints* de pesat.

Els sistemes basats en ossos es poden classificar en dues grans categories:

- Segueixen al *rig*
- Relatius al món (Locals)

2.4.6 Sistemes relatius al món

Aquest tipus de sistemes són els més usuals en els *rigs* facials. El sistema lògic i de pesat va calculat al (0,0,0) i va connectat com a capa de *blendshape* al *rig* del cos.

Serveixen per poder dividir el càlcul de les deformacions en diversos blocs i poder tenir una capa per a cada sistema de càlcul.

2.4.6.1 Blendshapes

Els sistemes basats en *shapes*, són un tipus de sistemes relatius al món, on es modelen les formes i es connecten. Són molt útils pels sistemes facials.

2.5 Conceptes previs

Abans d'iniciar qualsevol *rig* s'ha de tenir en consideració quins són els departaments previs que intervenen en la creació del *asset* i quins són els posteriors. Es important saber d'on ens ve la data i cap a on anirà la nostra.

Actualment ens podem trobar amb els següents departaments previs:

- **Modelat**: Dóna l'input de de la geometria del personatge. Iteraran moltes vegades sobre el personatge. En algunes produccions també proveeixen les *shapes* facials de deformació.
- **Textures**: Es dedicarà a fer les textures juntament amb el desplegat de les UV's.
- ***Groom***: Proveirà les peces del cuir cabellut. En el cas de que el cabell vagi prèviament animat, les corbes de simulació.

I com a departaments posteriors:

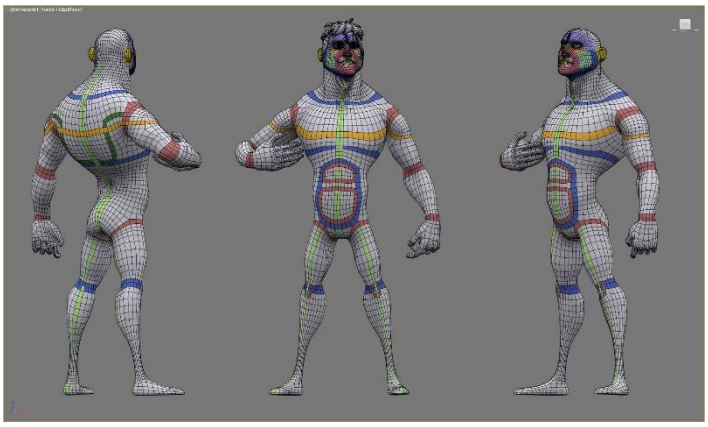
- **Animació**: Utilitza els controls del rig per a donar expressivitat al personatge.
- **Simulació**: Utilitza les geometries del cuir cabellut animades i les corbes que estaven extretes del *groom*. També usa l'animació de la roba per a tenir ja una base.
- ***Shot Sculpting***: Amb l'arxiu alembic (.abc) extret de l'animació, corregeix penetracions.

S'ha d'assegurar que tota la informació que arriba estigui en bon estat i que el resultat que s'entregui també ho sigui.

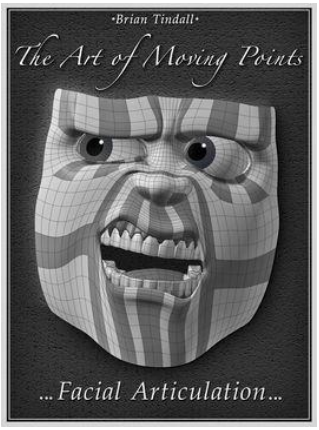
Com a consideracions que s'han de fer són les següents:

- La geometria de modelat ha de tenir una topologia fluida, on els *loops* estiguin repartits de forma homogènia per la superfície del model i sense que es desviïn.
- La topologia ha de estar recta, com més recta i relaxada estigui la topologia millor deformacions tindrà.
- En els punts de rotació(colzes, genolls, engonal i espatlla) hi ha d'haver més densitat de topologia per així acompanyar la pèrdua de la mateixa en les deformacions.
- Si les UV's no estan en el primer quadrant. S'haurà de fer un set propi per a *rig*, encara que sigui en automàtic, per a que els sistemes de seguiment com fol·licles puguin funcionar sense errors.
- Les geometries del cuir cabellut, preferiblement han de tenir una topologia semblant a la geometria on va enganxada.
- Les corbes de *groom* hauran de estar separades descripcions i ordenades en el *outliner* per grups.

- Tenir en el *rig* un grup on només hi hagi la geometria neta per a cache.



Il·lustració 6: Exemple de *flow* de *loops* de topologia d'un home cartoon, autor: Héctor Sanz DOrs



Il·lustració 5: Exemple de topologia facial, The Art of Moving Points

3 Creació del *rig* (Casos)

Hi ha molts factors que intervenen a l'hora de construir un *rig*. Es poden diferenciar en 4 àmbits:

- Tipus de programa a utilitzar
- Tipus d'animació
- Perfils d'animadors en el departament d'animació
- Tecnologia de la qual es disposa

El tipus de programa a utilitzar defineix quines eines es poden utilitzar. No tots els programes tenen les mateixes eines, cada un té les seves coses positives i les que no acaben de funcionar. També saber el programa a utilitzar dóna una gran possibilitat de *plug-ins* que podem incorporar en el *pipeline*.

És important saber si la producció aguanta el ús de *plug-ins*, ja que moltes vegades, el rendiment dels *plug-ins* millora els sistemes que proporciona el propi programa i a demés simplifica notablement de gruix de feina. Els *plug-ins* s'han de compilar per cada versió del programa i del sistema operatiu per tant fa difícil la seva implementació en produccions que tenen màquines diferents.

La manera d'animar juga un paper molt gran en el moment de preparar el personatge per a animació. No és el mateix un tipus d'animació realista, *cartoon*, semi-realista, amb arestes marcades o *stop-motion* 3D. Hi ha infinitat de maneres d'animar i s'ha de tenir clar quina s'usa en cada producció.

El nivell d'experiència dels animadors, és un dels altres punts que s'ha de considerar. No és el mateix tenir un equip on la majoria dels animadors tenen perfil de *Junnior* o *Mid*, que tenir un equip on en el que el gruix de l'animació recau en persones amb perfils *seniors* o superiors.

En un equip on la majoria sigui de perfil *senior* o superior es poden usar sistemes lògics de *joints*, *wires*…, més avançats i que portin les deformacions a un extrem on els personatges estan més vius i no tenen tantes limitacions. Això es pot fer ja que, una persona amb més experiència sabrà adaptar el seu estil d'animació amb el de la producció i es farà més difícil perdre l'aparença del personatge.

D'altra banda, amb un equip de perfils més *Junniors*, el *rig* demana ser limitat i que no deixi tanta llibertat de moviment en les deformacions, ja que la possibilitat de perdre l'actitud del personatge és més alta. Per tant les deformacions màximes estan preestablertes pel supervisor d'animació del projecte. Poden estar guardades en catàleg de poses o poden estar ja posades en les deformacions dels controls.

La tecnologia va avançant i any rere any es van implementant nous paradigmes que revolucionen la manera en la que es produeixen els *assets*. Actualment, s'està dirigint cap a la avaluació paral·lela amb *threads* i el càlcul de les deformacions per GPU. Hi ha diversos estudis d'animació i efectes que des de fa anys aposten cap aquesta via i han desenvolupat softwares i *plug-ins* propietaris. En el cas de "Pixar" tenen "Presto"

un sistema d'avaluació de les escenes i els personatges mitjançant el cache de les geometries i la implementació de sistemes “USD”

3.1 Anàlisi dels personatges

Prèviament a començar un *rig* s’ha d’observar com està construït el model, quina topologia té, quines són les seves limitacions i que és el que es vol en el *asset*.

També s’ha de tenir en compte quines són tots els extres que tindrà i els que no tindrà. Fer de més i invertir temps en sistemes que no s’usaran o innecessaris perquè treu temps de dedicar-lo als sistemes necessaris.

És molt important fer bé l’anàlisi, ja que a partir del mateix s’estructuraran les següents fases.

En el cas dels personatges de projectes podem analitzar les següents coses de cada un d’ells:

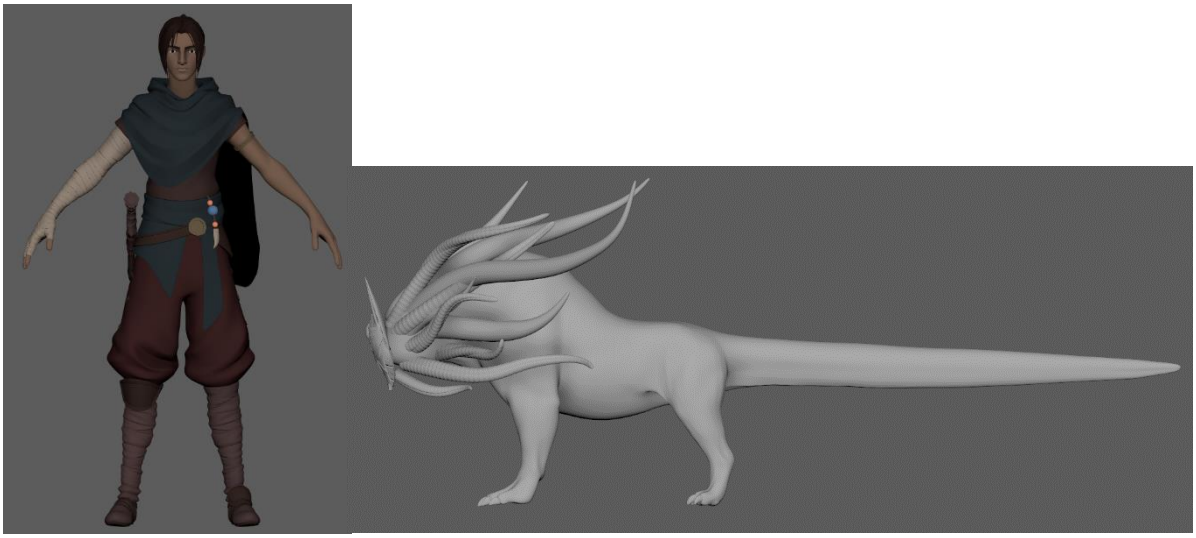
Ledyan (Bípede/LIB): El cos principalment és atlètic i necessita un *rig* per a que el *body mechanics* funcioni correctament. A nivell de expressivitat facial, en té poca, així que el facial pot ser més limitat. Com a extra té: el recipient de les fletxes, les fletxes, les proteccions, la ombrera i els braçalets. La roba anirà enganxada a sobre del cos amb *skinning*.

Yura (Bípede/LIB): De base es igual que el Ledyan, i d’expressivitat facial tindrà menys, degut al mocador que porta a la cara. Com a extres s’han de considerar les sabates, les proteccions de la cintura, els pits, les ombreres i els braçalets. La roba també anirà enganxada sobre del personatge.

Il·lustració 7: El personatge de Yura a l’esquerra i a la dreta el personatge de Ledyan. Els dos de “Last in Battle”

Ekko (Bípede/Path of Sand): Igual que els altres dos bípedes com a base. Aquest personatge té més expressivitat facial per tant necessitarà més cura en el facial. Com a extres té: el cinturo, la daga, el penjoll del cinturo, la tela que li penja del cinturó, la capa, el buff, un canvi de roba en shot, caputxa.

Monstre (Quadrúpede/ Path of Sand): En el shot surt caminant, per tant necessitarà un sistema de flexió en les cames darreres i en les frontals, un omòplat en la cama frontal i una clavícula en la cama posterior, sistema per la respiració, sistema per poder dibuixar la cua en una forma adient i sistemes de control dirigit en els tentacles i sistemes automàtics en els tentacles.

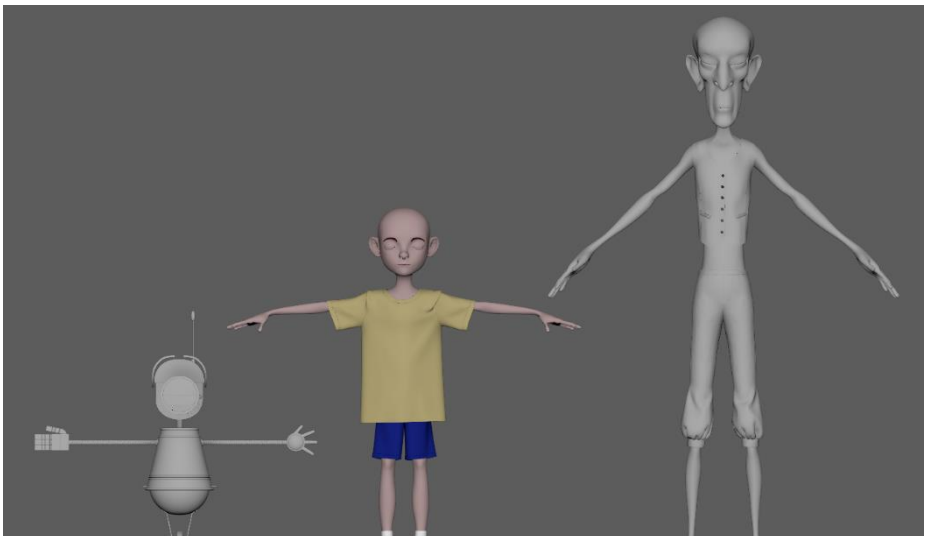


Il·lustració 8: Ekko a l'esquerra i monstre a la dreta. L'escala no es representativa entre els dos personatges. Personatges de "Path Of Sand"

Leo (Bípede/ WOLT): *Rig* corporal i facial bàsic. La roba anirà simulada per sobre de l'animació

Druidac (Bípede/ WOLT): *Rig* corporal i facial bàsic. La roba anirà simulada per sobre de l'animació

Nut (Robot/ WOTL): És un personatge que te certa rellevància en el projecte, al no ser un bípede, requerirà un *rig* personalitzat, on es pugui moure i que els braços articulin de manera elàstica. Molta expressivitat en les celles i que es puguin fer moltes formes diferents. Pupil·la retràctil.



Il·lustració 9: En ordre d'esquerra a dreta, Nut, Leo i Druidac. Personatges de World Of Lost Things

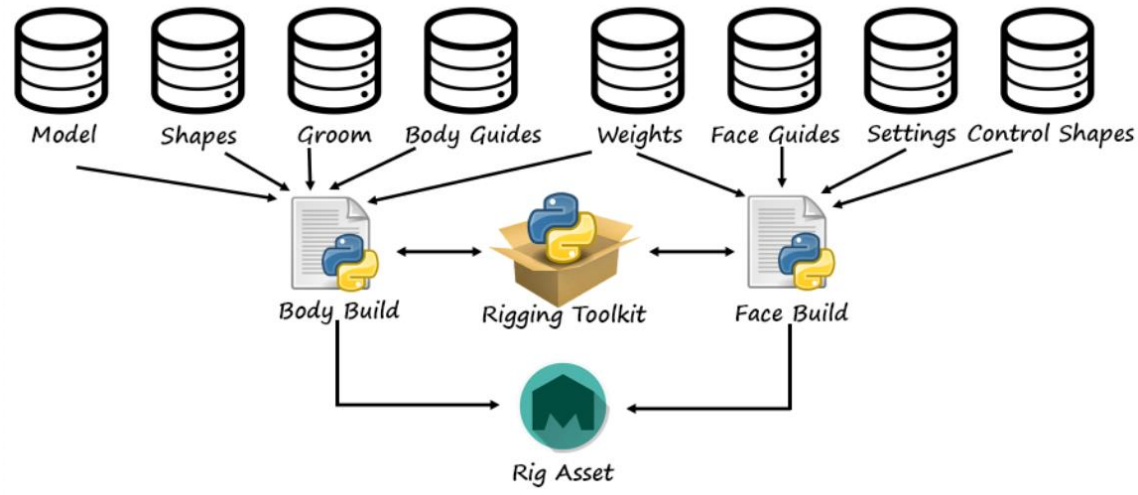
3.2 Plantejament

Per tal de poder controlar tots els canvis de versions de model, uv's i *groom* i per tal de poder començar a fer els *rigs* amb antelació, abans de tenir els models finals, el rig es planteja per fer per programació (Python). I com a sistema de control de versions: Git, especialment un repositori a GitHub

Així podem separar la creació del rig en les següents parts:

- **Model**: Carpeta amb un seguit d'arxius .ma que contenen les diverses versions del model
- ***Shapes***: Carpeta amb un seguit d'arxius .ma que contenen les diverses versions del model
- ***Groom***: Carpeta amb un seguit d'arxius .ma que contenen les scalps i les corbes
- Guies corporals: Carpeta amb un seguit d'arxius .ma on hi ha uns joints que defineixen la posició d'entrada perquè els mòduls sàpiguen on crear-se
- **Guies facials**: Carpeta amb un seguit d'arxius .ma on hi ha uns locators i unes nurbs que defineixen la posició dels sistemes lògics del facial
- **Pesos dels mapes**: Seguits de carpetes amb els diferents mapes de pesos guardats en binari
- **Settings**: Carpeta amb un seguit d'arxius .json amb propietats definides per a cada personatge
- Control Shapes: Carpeta amb la informació de construcció de les control shapes

Totes aquestes parts estan recollides dintre d'un arxiu python (.py) que serà el que construirà cada vegada que s'executi un nou *rig* amb els diferents inputs que hi hagi. Per accedir a les accions de Maya s'usa les llibreries "**maya.cmds"**, "**OpenMaya"**, "**OpenMayaAnim"** i per a la gestió d'arxius i altres utilitats s'utilitzen les llibreries de os, "shutil", "json", "cPickle", "StringIO", "time", "logging".



Il·lustració 10: Representació de l'estructura per a generar un *rig* segons el plantejament establert

3.2.1 Modularitat i proceduralitat

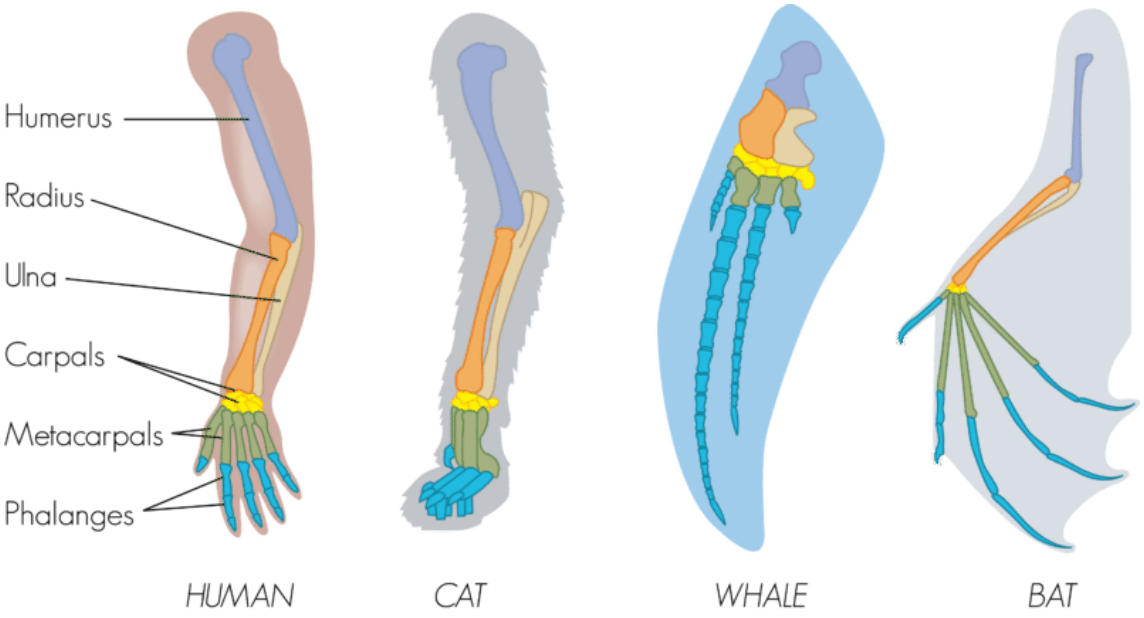
El *rig* **procedural** esdevé quan es fa, no només mitjançant codi, si no que també quan s'apliquen les regles de la programació orientada a objectes a la seva estructura.

Ha de satisfer les premisses de:

- **Abstracció**: Extreure les necessitats comunes dels objectes. Separar el comportament de la seva implementació.
- **Encapsulació**: Ocultació d'informació. L'animador desconeix la implementació interna.
- **Herència**: Relació entre els diferents objectes de *rig*. Mòduls que van des de la generalització fins l'especificació.
- **Polimorfisme**: Un mateix objecte de *rig* es comporta de manera diferent en funció de l'operació que se li aplica.

Es pot establir que un sistema d'extremitat és igual ja sigui una cama o un braç. Així doncs, amb un objecte de *rig* d'extremitat es pot satisfer les necessitats de braços i de cames. També passa amb les altres parts del cos. Només es necessari implementar un sistema de dit del qual depenguin els altres.

Això no només passa amb els humans. El concepte de la **anatomia comparativa**, descriu que entre tots els éssers vius hi ha un patró similar en quant a estructures. Per tant, podem reaprofitar els mòduls creats entre diversos personatges del mateix tipus.



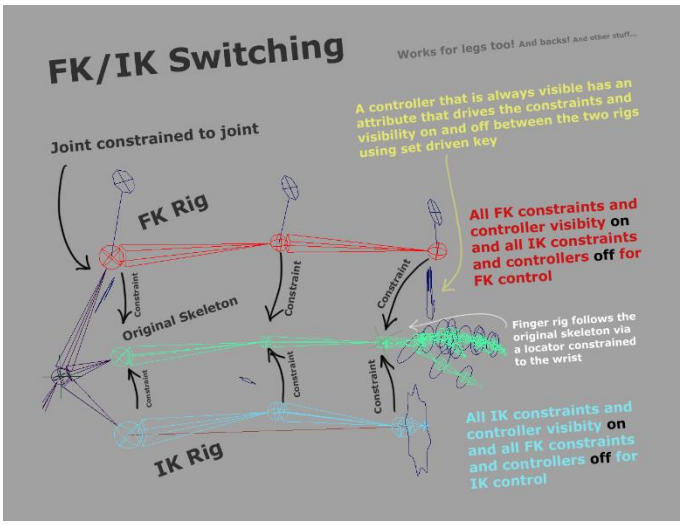
Il·lustració 11: Comparació de les extremitats de diversos animals segons la teoria de la anatomia comparativa.

3.2.2 Sistemes corporals bàsics

El sistemes dels bípedes tindran les següents característiques:

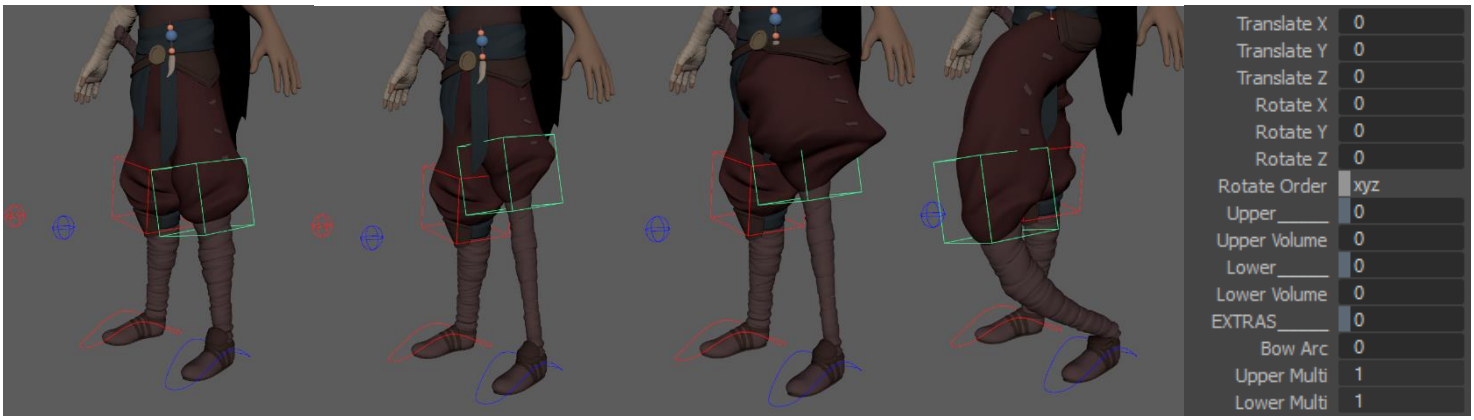
El *setup* d'un bípede serà similar al de la Mary Project un personatge modelat i texturat per José Manuel García Alvarez i *riggejat* per Antonio Méndez Lora. És un personatge que tots els companys i companyes tenen per la mà i no els hi costarà d'utilitzar.

Per al *rig* corporal general hi ha sistemes de IK i FK per a les extremitats. El sistema de IK té també un sistema de *stretch*, per a portar les deformacions més lluny.



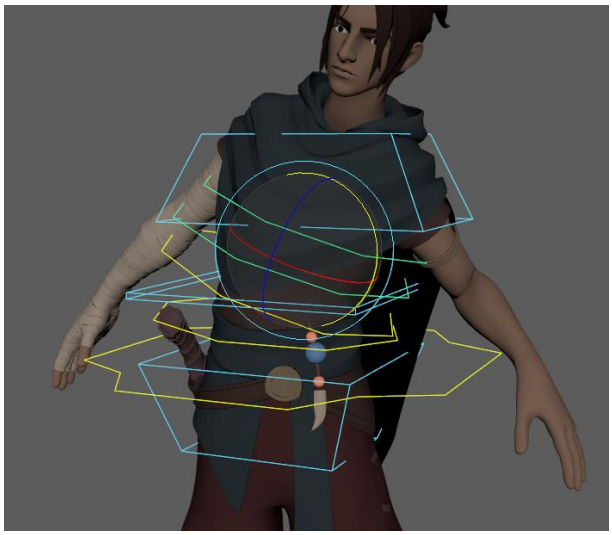
Il·lustració 12: Esquema de canvi de IK a FK establerta per Intro Rigging [1]

En el sistema del les extremitats tenen un control extra que permet fer que sigui mes llarga o més curta la part superior o inferior de l'extremitat, mantenint o no el volum. També permet arquejar-la.



Il·lustració 13: Exemple de extremitat en IK, on hi ha un control que mou la cama amb el IK, un *pole vector* que defineix la direcció que mira la cama i un control que permet donar-li un *offset* al genoll.

Per l'esquena hi ha un sistema de IK i FK, però, aquest es diferent ja que els controladors del FK estan posicionats després del IK, com es pot veure en la següent imatge. També te uns paràmetres d'*squash* i *stretch* amb valors mínims i màxims, per a que li doni més dinamisme a l'animació.

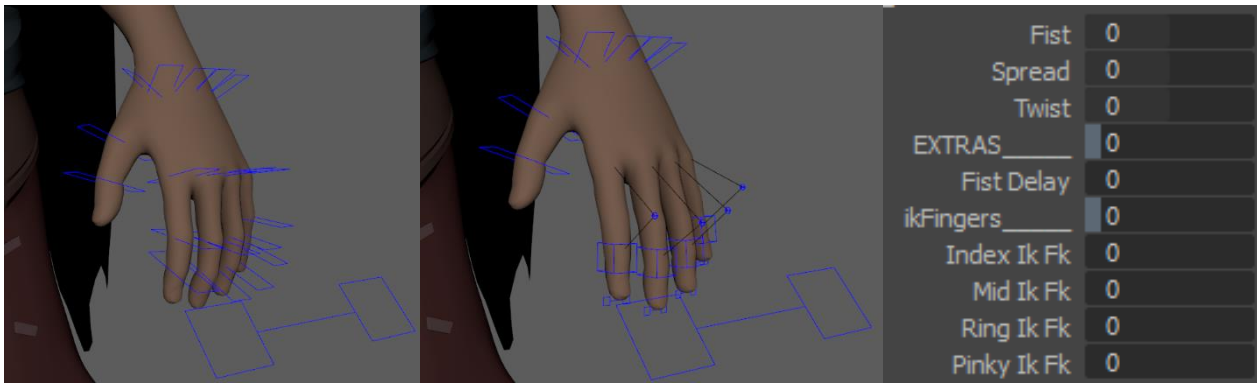


Il·lustració 14: Exemple de esquena en el sistema IK/FK viu

El sistema de les clavícules s'emporta els braços. Per als braços en FK té una opció per a que els braços segueixin les rotacions de la clavícula o no, així se li dóna un altre efecte a aquest sistema.

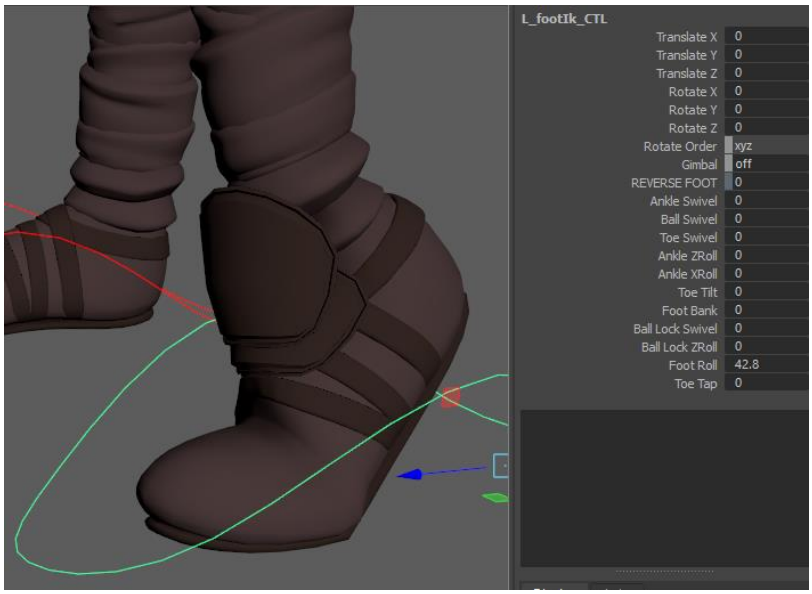
Il·lustració 15: Exemple de funcionament de la clavícula amb el sistema de seguiment del braç en FK

Els dits de les mans tenen sistemes de control, tant en IK com en FK. Quan està en FK té atributs de *spread*, *fist* i *twist*.



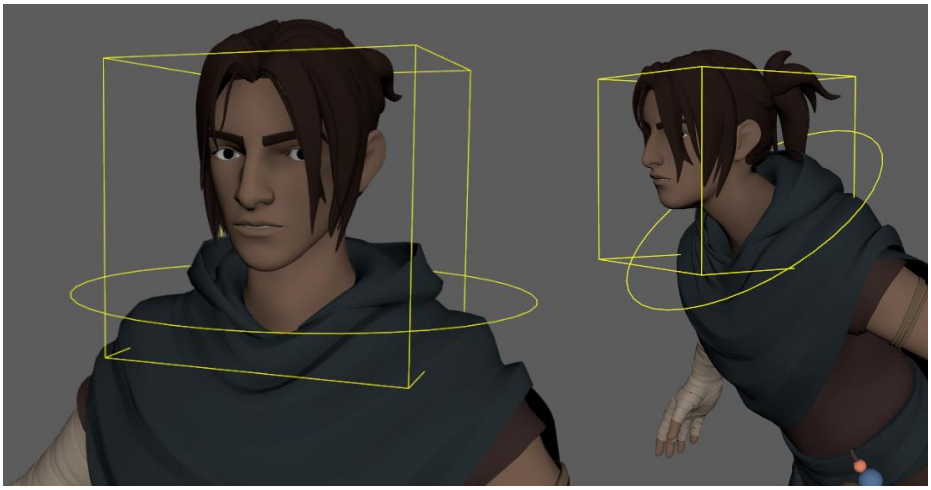
Il·lustració 16: Sistema de de control dels dits de les mans, amb els atributs presents en el control genèric. Al centre es pot observar el sistema de IK en els dits.

Per al peu en IK hi ha un sistema revers de IK que dóna les següents opcions al animador: *Ankle Swivel, Ball Swivel, Toe Swivel, Ankle ZRoll, Ankle XRoll, Toe Tilt, Foot Bank, Ball Lock Swivel, Ball Lock ZRoll, Foot Roll, Toe Tap*. Tots aquests sistemes estan formats per connexions directes a grups de offset sobre el IK de la cama.



Il·lustració 17: Sistema de peu invers, a la esquerra està col·locat el roll del peu, a la dreta estan presents els atributs específics del control

El sistema del cap té un control FK des de la base del coll i un control IK des de la punta. També te una opció per a que el cap no segueixi les rotacions del pit.



Il·lustració 18: Sistema de control del cap i del coll. A la dreta es veu com esta actuant el sistema de no seguiment del cap a les rotacions del pit.

En els ulls hi ha un sistema per a controlar el *aim* dels mateixos, separat per dret i esquerra i un global que s'emporta els dos. També tenen control de les dimensions de les pupil·les i el iris.

Il·lustració 19: Sistema de control dels ulls. Es pot veure quin és el control que es té amb els sistemes del iris i de la pupil·la

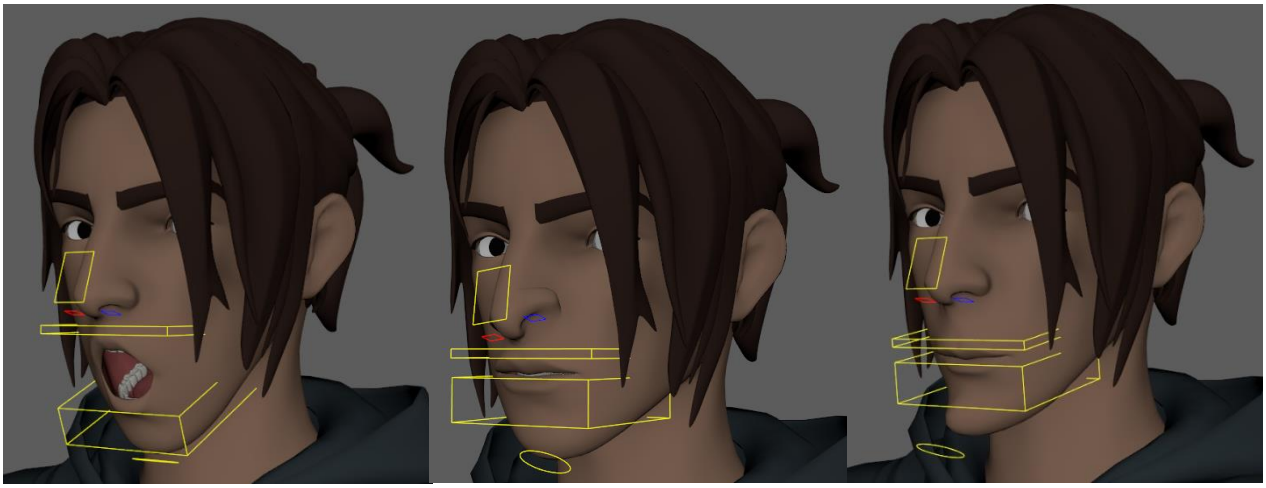
3.2.3 Sistemes facials bàsics

El *rig* facial, és relatiu al món, per tant, va connectat a les deformacions del cos mitjançant una *blendshape*. Tots els controladors facials estan en mirall.

El ordre en el que s'exposen les capes no és en l'ordre que hi són implementades en el *rig*, només són agrupacions d'idees lògiques.

3.2.3.1 Capa 1

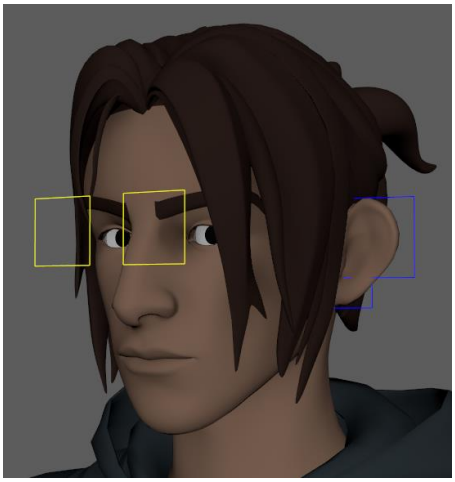
Sistema d'ossos per a la cara bàsica: part inferior de la mandíbula, part superior de la mandíbula i nas. La mandíbula superior mai va per sota de la mandíbula inferior, això fa que el la boca es tanqui i s'empenyi amb la col·lisió.



Il·lustració 20: Sistema de control de lla mandíbula i el nas

3.2.3.2 Capa 2

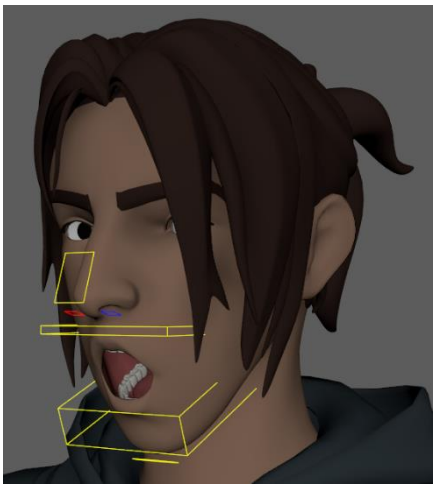
Sistema d'ossos per la part superior de la mandíbula, els forats dels ulls, orelles. Per a poder moure'ls sencers i posicionar-los.



Il·lustració 21: Sistema de control per a ulls i orelles.

3.2.3.3 Capa 3

Shape correctiva de la obertura de la mandíbula per ajudar aconseguir la compressió de les galtes al obrir la boca i tenir la forma de O. Serà feta amb un *joint* i serà escalat.



Il·lustració 22: Sistema de compressió dels llavis al obrir la boca

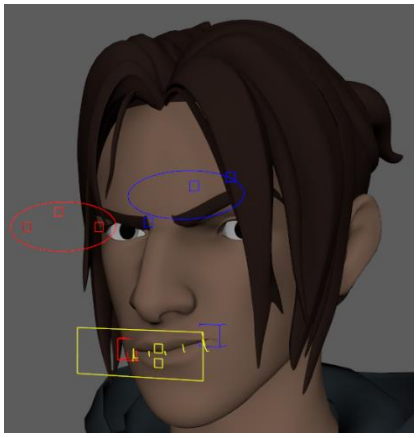
3.2.3.4 Capa 4

Sistema de *joints* de les parpelles, *rivet* i *blendshape* entre els *edges* superiors i inferior de les parpella. Es pot definir l'altura del parpalleig, o bé si es vol fer per separat.

Il·lustració 23: Sistema de control de les parpelles. Es poden veure els atributs

3.2.3.5 Capa 5

Sistema de retoc fi de les *shapes*. Controls de retoc per a zona dels llavis, nas labial i celles



Il·lustració 24: Sistema de retoc de les celles i els llavis

3.2.3.6 Capa 6

Sistema de *blendshapes* de (Els models basics i simètrics de les *shapes* són de l'Enrique Alberola), el meu paper va ser fer el disseny de les *shape*, com connectar-les i petits retocs i repassos de les *shapes*):

Boca: Tindrà un sistema de 8 *blendshapes*, són modelades les formes de comissura estirada, comissura aixafada, comissura pujada, comissura baixada i les seves combinacions. A l'hora de modelar les *shapes* de la boca, s'ha de tenir en compte que els llavis han de lliscar per sobre de la mandíbula que forma la zona de les dents. Es important que el llavi superior i l'inferior no es separin.

- ***Comissura estirada***: És important que la deformació sigui paral·lela alls llavis base, ja que amb la combinació de les altres generaran altres formes. Principalment es mou la comissura dels llavis i s'emporta una mica del mig. S'enfonsa la zona nas-labial de manera horitzontal i estira molt suaument la zona de la fosa nasal.

- ***Comissura aixafada***: És la inversa que la de la comissura estirada, es tracta de fer que les comissures dels llavis estiguin a punt de tocar-se. La part central dels llavis es comprimeix (ha de lliscar per la superfície de la mandíbula)

- ***Comissura pujada***: S'aixeca la comissura dels llavis, acompanya lleument el nas i la zona nas labial. La galta acompanya lleument a la deformació, ha de aconseguir arribar a la forma de U.

Il·lustració 25: Shapes en ordre d'esquerra a dreta: comissura estirada, comissura aixafada, comissura pujada i comissura baixada

- ***Comissura baixada***: És la contraria a l'anterior, ha de baixar la comissura dels llavis i estirar els *loops* de la zona nas-labial. La galta s'estira una mica.
- La unió de les 4 deformacions anteriors, s'interpola mitjançant un controlador que es mou en translació y i x. Tot i que s'interpolin les deformacions, les *shapes* que queden resultants en els extrems [(10,10), (-10,10), (-10,-10), (10,-10)] no acaben de tenir una deformació adient pel que fa a les galtes i l'harmonia entre elles. Per arreglar les deformacions resultants, s'han de generar *shapes* de correcció que forçaran la forma que determinem. I les seves combinacions: Per a generar les *shapes*, s'ha de partir de la unió de:
    - ***Estirada + Amunt***
    - ***Estirada + Avall***

- ***Comprimida + Amunt***
- ***Comprimida + Avall***

L'objectiu de les correctives és marcar la zona de les galtes. Són *shapes* claus per a marcar l'actitud del personatge.

Il·lustració 26: Shapes de correcció, en ordre d'esquerra a dreta: estirada amunt, estirada avall, comprimida avall i comprimida amunt

Galta: Té 4 *shapes* (galta inflada, galta aspirada, galta pujada i galta lliscada)

- ***Galta inflada***: Les formes esfèriques funcionen bé. Per a fer-la va bé posar una esfera com a guia ajuda. També acostuma a funcionar treure la forma amb un deformador de "*softMod*"
- ***Galta aspirada***: S'ha de tenir en conta que sota dels ulls esta l'os zigomàtic, per tant la galta s'ha de col·locar una mica per sota
- ***Galta pujada***: Ha de fer la compressió de la galta amb l'ull.
- ***Galta lliscada (4)***: Tracta del lliscament en translació y i x de la galta per sobre de la cara.

Il·lustració 27: Shapes de les galtes, en ordre de esquerra a dreta i de a dalt a baix: inflada, aspirada, lliscada pujada, lliscada baixada, lliscada endavant i lliscada endarrere.

***Shapes* dels llavis**: (Roll positiu, roll negatiu, petó) Serveixen per poder fer les deformacions fonètiques i per donar intenció als moviments dels llavis. Estaran controlades per atributs limitats de 0 a 10.

- ***Roll positiu***: Tracta de la rotació cap a fora dels llavis, tant inferior com superior, s'han de veure una part de les dents.
- ***Roll negatiu***: Igual que l'anterior, aquest cas es cap a dins. Serveix per estrènyer llavis i fer fonemes bilabials (p, b)
- ***Petó***: Similar a la de boca estreta, aquesta fa de petó, és a dir, apart d'estrènyer les comissures i els llavis, treu més volum en els llavis i comprimeix les galtes.

Il·lustració 28: Shapes especifiques dels llavis, en ordre d'esquerra a dreta: roll positiu, roll negatiu petó

Celles: Hi ha 8 *shapes* (pujades, baixades, juntes, separades, girades en positiu, girades en negatiu, endavant i enrere). És important que llisquin per la superfície fictícia del crani del personatge.

- ***Pujades***: Les celles han de pujar verticalment i simètricament fins a aconseguir igualar la topologia. Les celles han de quedar una mica corbades.
- ***Baixades***: Les celles baixen fins l'altura de la meitat dels ulls, s'ha de vigilar que no col·lapsi amb les pestanyes, en el cas que les tingui llargues. No s'ha de marcar l'entrecella.
- ***Marcatge de l'entrecella***: S'ha de marcar l'entrecella, fer que col·lapsi la cella
- ***Separades***: Les celles han de lliscar per sobre del crani cap a fora.
- ***Juntes***: Les celles han de lliscar per sobre del crani cap a dins
- ***Girades en positiu***: Han de fer un sinus en positiu
- ***Girades en negatiu***: Han de fer un sinus en negatiu

Il·lustració 29: Shapes de les celles, en ordre d'esquerra a dreta i d'amunt a avall: pujades, baixades, endavant, endarrere, separades, juntes, marcatge de l'entrecella, girades en positiu, girades en negatiu.

- ***Endavant***: Les celles han de sortir cap a enfora, serveix per a corregir altres deformacions
- ***Enrere***: Les celles han de sortir cap a dins, serveix per a corregir altres deformacions

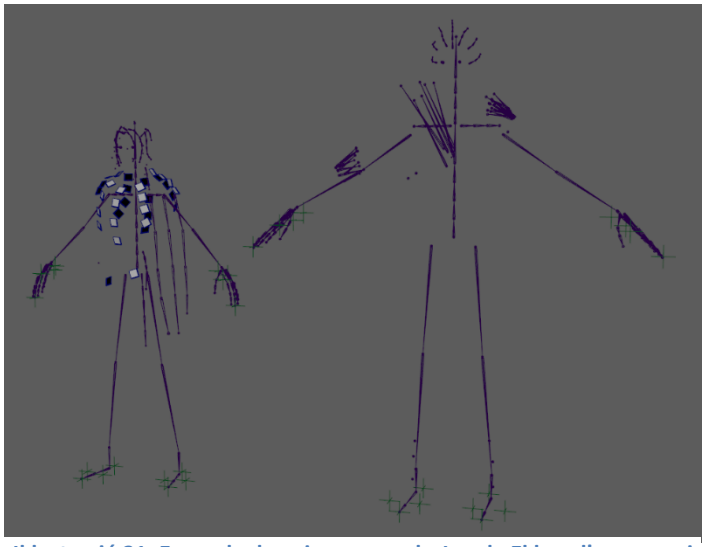
Nas: Té 3 shapes (sneer, flare, sniff)
- ***Sneer***: Pujada de les foses nassals
- ***Flare***: Inflat de les fosses nassals
- ***Sniff***: Aspiració de les fosses nassals

Il·lustració 30: Shapes del nas, en ordre d'esquerra a dreta: sneer, flare, sniff

3.3 Execució

3.3.1 Corporal

El primer pas és establir les guies corporals. Són un seguit de *joints*, *locators* i *nurbs*, que estableixen la informació de posició i orientacions per a que els diversos mòduls (classes) tinguin les dades per operar.



Il·lustració 31: Exemple de guies corporals. Les de Ekko a l'esquerra i les de Ledyan a la dreta

Aquestes dades són les bases per a la creació de tots els sistemes corporals, tant genèrics com específics.

El segon pas és dissenyar l'estructura del arxiu de Python que serà el responsable de crear tots els sistemes usant el "**riggingToolkit"** dissenyat. Aquest és un exemple de l'inici de la construcció, on es carrega el model, les guies, es crea l'estructura bàsica del *outliner* i es crea el mòdul de l'esquena.

```
def build(character_name, body=False, faleco=False, publish=False):
    # create new file
    mc.file(new=True, f=True)
    for plugin in ['matrixNodes.mll', 'MayaMuscle.mll']:
        if not mc.pluginInfo(plugin, q=True, l=True):
            mc.loadPlugin(plugin)

    # create base rig modules
    arm = asset.Asset(character_name)
    char = character.Character(character_name=character_name, asset=arm, extra_scale=20)

    # load models and guides (body)
    char.load_model()
    char.load_guides(body=body)

    # create base structure
    char.prepare(body=body)
    # place geometry group inside the geo transform
    mc.parent(character_extras.character_geometries(character_name=character_name), char.geometry_grp)

    ################################
    # create spine
    first_joint = 'C_spineRoot_JNT'
    childs = mc.listRelatives(first_joint, allDescendents=True)
    childs.reverse()
    spine = spineCmds.Spine(name='spine',
                            start_joint=first_joint,
                            mid_joints=childs[:-1],
                            end_joint=childs[-1],
                            character=char,
                            hook_to=char.masterWalk.transform)
    spine.create()
```

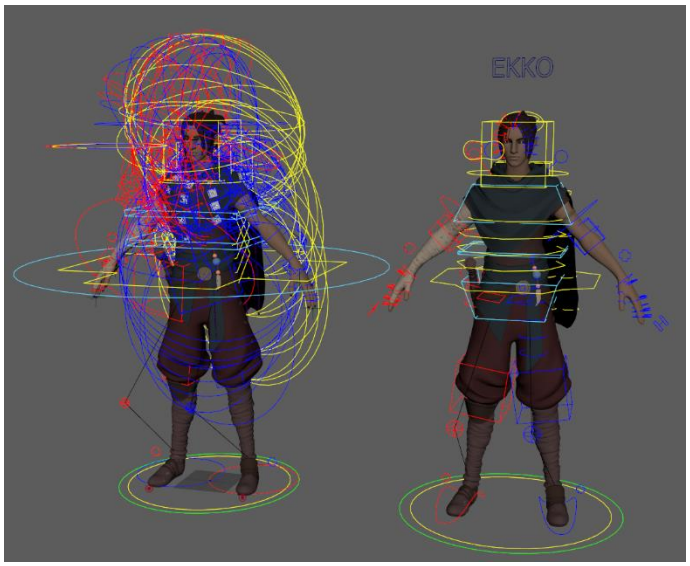
Il·lustració 32: Exemple d'inici de fitxer de construcció del asset

El tercer pas es definir els espais que tindrà cada control, o seguit de controls. Existeixen els espais dinàmics, com el del braç o el cap i els de seguiment com la mà o el peu.

```
    ###################################
    ###################################
    ###################################
    # Define space switches
    for index, arm in enumerate(arms):
        spaceSwitchesCmds.space_switches(node=arm.pv_control.transform,
                                         space_dict={
                                             'hand': arm.ik_control.transform,
                                             "clavice": clavices[index].end_joint,
                                             "world": char.masterWalk.transform,
                                             "pelvis": spine.start_joint
                                         },
                                         default='clavice'
                                         )
        spaceSwitchesCmds.space_switches(node=arm.ik_control.transform,
                                         space_dict={"clavice": clavices[index].end_joint,
                                                     "world": char.masterWalk.transform,
                                                     "pelvis": spine.start_joint,
                                                     "chest": spine.end_joint,
                                                     "head": head.start_joint,
                                                     },
                                         default="world"
                                         )
        spaceSwitchesCmds.orient_switches(node=arm.fk_start_control.transform,
                                          space_dict={"clavice": clavices[index].end_joint,
                                                      "world": char.masterWalk.transform
                                                      },
                                          default="clavice"
                                          )
```

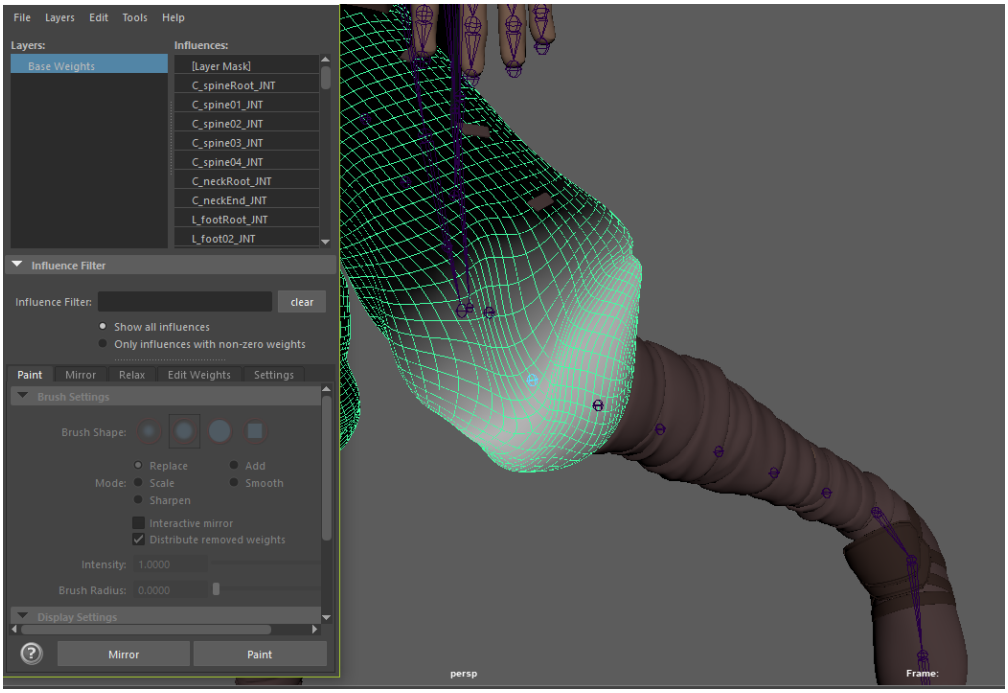
Il·lustració 33: Exemple de codi de la definició dels espais de seguiment dels controladors

El tercer pas és executar el arxiu de construcció i adaptar les dimensions dels controladors que surten per defecte. Passar dels controladors de l'esquerra als de la dreta.



Il·lustració 34: Exemple de guies del personatge, a l'esquerra les guies de defecte, a la dreta les guies col·locades

El quart pas passa a ser el *skinning* de les geometries, que és quan es defineix quina zona d'influència té cada os. Per a fer-ho, s'usen *meshes* proxies i eines com el "ngSkinTools" i el "brSmoothWeights". Per agilitzar la lectura i el salvatge dels mapes es guarden en arxius binaris creats amb "cPickle".



Il·lustració 35: Exemple de pintar pesos amb el ngSkinTools, on hi ha una capa de pesos.

El cinquè pas, és carregar els *settings* i els pesos que s'han establerts. Per exemple els de *fist*, *spread* i *twist* dels dits de la mà.

```
# set freeze joint for skinning
freeze_jnt = mc.createNode('joint', n='C_bodyFreeze_JNT')
mc.parent(freeze_jnt, char.body_skel)
char.skinning_objects.append(freeze_jnt)

# label joints
joints.label_joints()

# load saved curves
curves.load_all_curves(char)

# load settings
settings.load_all_settings(char)

# load skinCluster.
# load last available skin
skinCluster.load_david_cuellar(character=char)
```

Il·lustració 36: Exemple de càrrega de formes dels controls i dels mapes de pesos

El sisè pas per acabar la construcció del cos, és bloquejar i col·locar certs atributs a mesura del supervisor d'animació. Així, es fa més difícil que el animador trenqui el *rig* accedint a opcions que no estan visibles de primera forma.

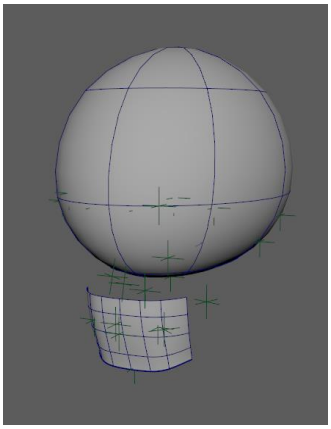
```
if publish:
    # lock everything for the animator
    [mc.setAttr(o + '.ihi', 0) for o in mc.ls('*Shape*', '*_REV*', '*_MDN*', '*Constraint*', '*_SKN*', 'bindPose*',
                                              '*IF*', '*MDL*', '*_PMA*', '*RMV*')]

    mc.hide(mc.ls(type='ikHandle'))
    mc.hide(mc.ls('*_LOC'))

    # set geometry grp to reference mode
    mc.setAttr(char.geometry_grp + '.overrideEnabled', 1)
    mc.setAttr(char.geometry_grp + '.overrideDisplayType', 2)

    # set skeleton grp to reference mode
    mc.setAttr(char.body_skel + '.overrideEnabled', 1)
    mc.setAttr(char.body_skel + '.overrideDisplayType', 2)
    mc.hide(char.body_skel)

    mc.hide(char.body_rig)

    mc.select(d=True)

    mc.setAttr('L_armExtra_CTL.IkFk', 1)
    mc.setAttr('R_armExtra_CTL.IkFk', 1)
    mc.setAttr('L_legExtra_CTL.IkFk', 0)
    mc.setAttr('R_legExtra_CTL.IkFk', 0)

    mc.setAttr('L_legExtra_CTL.stretch', 1)
    mc.setAttr('R_legExtra_CTL.stretch', 1)
    mc.setAttr('L_armExtra_CTL.stretch', 1)
    mc.setAttr('R_armExtra_CTL.stretch', 1)
    if mc.objExists('working'):
        mc.hide('working')

    # cleanup things
    cleanup.delete_unknown_nodes()
    try:
        cleanup.removed_unsused_influences([char.geometry_grp])
    except:
        mc.warning('Cannot remove influences')
    # cleanup.delete_unused_nodes()

    # # add character name
    curve_name = textToCurves.text_to_curve(text=character_name.upper(), curve_name='C_characterName_CRV', size=100)
    mtx_bb = mc.exactWorldBoundingBox(char.geometry_grp, calculateExactly=True)
    mc.xform(curve_name, t=(0, mtx_bb[-2] * 1.1, 0), ws=True)
    mc.makeIdentity(curve_name, apply=True, t=True)
    mc.parent(mc.listRelatives(curve_name, ad=True, s=True), char.global_ctl.transform, r=True, s=True)
    mc.delete(curve_name)
```

Il·lustració 37: Exemple de preparació del rig per a animació

3.3.2 Facial

La construcció del *rig* facial, és similar a la construcció de la corporal.

El primer pas també és definir les guies d'on estaran posicionats els sistemes. Els *locators* marquen posicions i orientacions, i les superfícies marquen espais de projecció dels sistemes facials. La superior marca quin es el rang de moviment de les celles i la inferior fa el mateix amb els llavis. D'aquesta manera es mantenen més les formes del personatge.



Il·lustració 38: Exemple de guies facials, Ekko

El segon pas és generar l'arxiu de construcció del facial. La diferencia amb el corporal és que es carrega el rig corporal i es creen capes de deformació.

```
def build(character_name, body=False, face=False, publish=False):
    # create new file
    mc.file(new=True, f=True)
    for plugin in ['matrixNodes.mll', 'MayaMuscle.mll']:
        if not mc.pluginInfo(plugin, q=True, l=True):
            mc.loadPlugin(plugin)

    # create base rig modules
    a = asset.Asset(character_name)
    char = character.Character(character_name=character_name, asset=a, extra_scale=20, face=True)

    # load models and guides (body)
    char.load_body_rig()
    char.load_guides(face=True)

    # create base structure
    char.prepare(body=body, face=True)
    main_blendshape = blendShape.BlendShapeCommand(name='faceDeformation',
                                                   base=character_extras.character_body_geo(character_name)
                                                   )
    tune_blendshape = blendShape.BlendShapeCommand(name='tune',
                                                   base=character_extras.character_face_geo(character_name)
                                                   )
    tune_mesh = tune_blendshape.duplicate_and_add(name='C_tune_GEO')
    main_blendshape.add_shape(tune_mesh.split('.')[1])
    mc.parent(tune_mesh.split('.')[1], char.face_geo)
    sticky_blendshape = blendShape.BlendShapeCommand(name='sticky', base=tune_mesh.split('.')[1])
    sticky_mesh = sticky_blendshape.duplicate_and_add(name='C_sticky_GEO')
    local_blendshape = blendShape.BlendShapeCommand(name='local', base=sticky_mesh.split('.')[1])
    local_mesh = local_blendshape.duplicate_and_add(name='C_local_GEO')
    jaw_layer = char.face_layer(geometry=local_mesh.split('.')[1], name='jawNoseEars')
    sliding_layer = char.face_layer(geometry=local_mesh.split('.')[1], name='sliding')
    face_shapes_layer = char.face_layer(geometry=local_mesh.split('.')[1], name='faceShapes')
    eyelids_layer = char.face_layer(geometry=local_mesh.split('.')[1], name='eyelids')
    move_layer = char.face_layer(geometry=local_mesh.split('.')[1], name='move')

    char.load_blendshapes()
    face_shapes_blendshape = blendShape.BlendShapeCommand(name='faceShapes',
                                                          base=face_shapes_layer.split('.')[1]
                                                          )
    character_extras.wrap_stuff(character_name=character_name)

    # extra layers
    character_extras.extra_layers(character_name=character_name, char=char)

    ##################################
```

Il·lustració 39: Exemple de la creació de les capes de deformació per al facial

Com a tercer pas, es carreguen les *shapes*, els pesos i els *settings* de deformació igual que en el sistema corporal.

3.4 Comprovacions

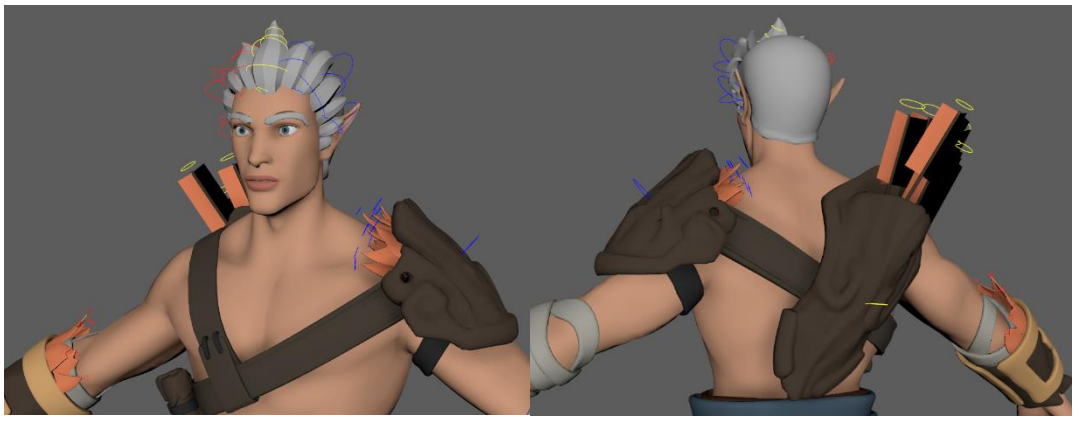
Tot i que el *rig* es faci de manera procedural, s'ha de comprovar cada vegada si les deformacions de la nova construcció funcionen amb les antigues animacions. Per tant és important tenir un sistema com el "Studio Library" o exportacions i importacions d'"Atom", amb animacions guardades, per a corroborar que la nova versió es compatible amb les anteriors.

3.5 Extres

Apart dels sistemes genèrics per als personatges s'han creat sistemes específics per a cada un d'ells.

3.5.1 Ledyan

Se li han creat sistemes de control FK per a la ombrera, per a les fulles de la ombrera, pel carcaix, per les fletxes del mateix i pel cabell.



Il·lustració 40: Exemple dels sistemes FK en Ledyan

Les canyelleres i les cintes tenen un sistema d'un *joint* que descansa sobre una *nurbs surface* que té els mateixos pesos que la geometria de sota.

Il·lustració 41: Exemple de sistemes de de *joint* en una nurbs.

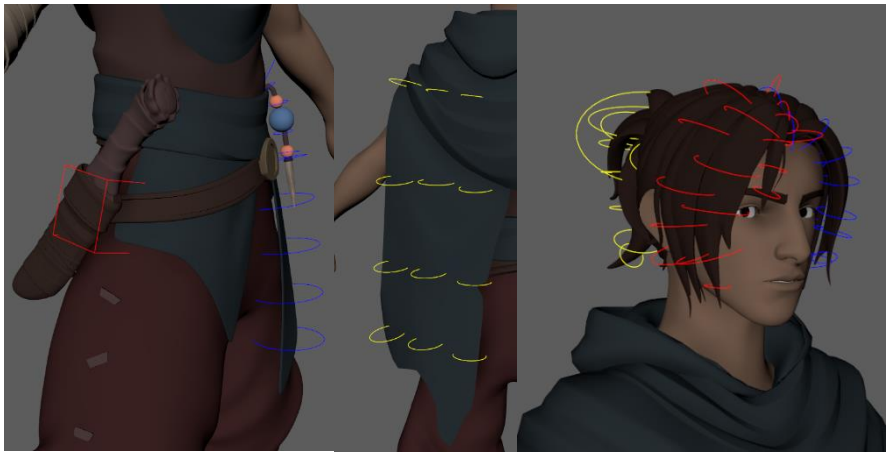
3.5.2 Yura

Té sistemes de FK per les ombreres, els pits i els flocs del cabell. Pel monyo té un sistema similar al de l'esquena, té un IK amb *squash* i *stretch* per a donar-li moviments secundaris.

Il·lustració 42: Sistemes de FK i de splne per a Yura

3.5.3 Ekko

Totes les coses que li penjen a aquest personatge, amulet del cinturó, tela del cinturó, espasa, cabells són cadenes FK enganxades a diverses parts del cos.



Il·lustració 43: Sistemes de FK en Ekko

La caputxa esta feta amb el mateix sitema de IK amb tangencies i squash que el monyo de Yura.

Il·lustració 44: Sistema de IK de la caputxa d'Ekko

Il·lustració 45: Sistemes de joint a una nurbs en Ekko

Pel buff, per la sivella i pel punt on s'enganxa la daga, hi ha un sistema de una *nurbs* pesada amb les mateixes deformacions que la geometria de sota i mou en una segona capa la geometria que controla.

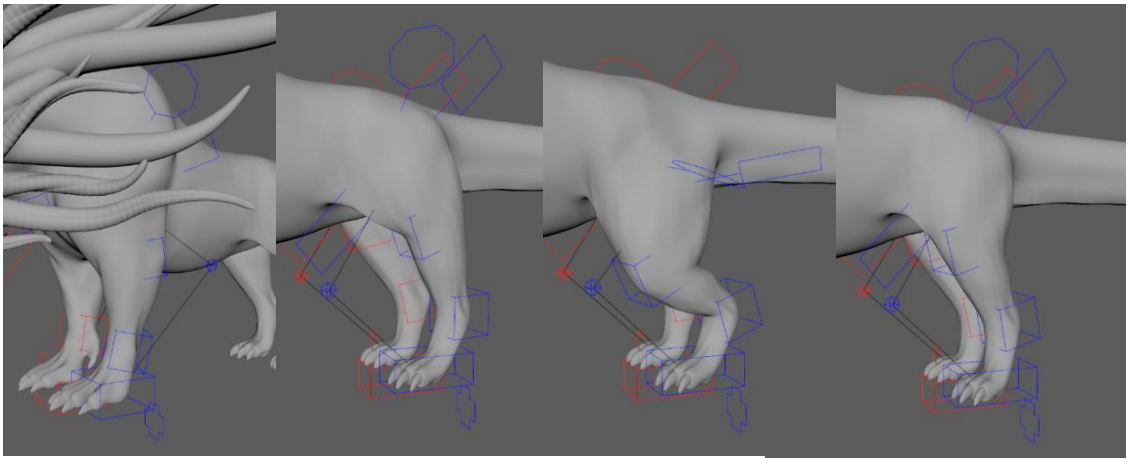
Ekko, té la possibilitat de posar i treure la caputxa. La caputxa posada te un seguit de sistemes FK per a que es pugui animar el comportament dinàmic, sense haver-la de simular. Alguns controls segueixen diversos ossos del cos (coll (4), cap(2))

Il·lustració 46: Sistemes de FK per a lla caputxa posada de Ekko

3.5.4 Monstre

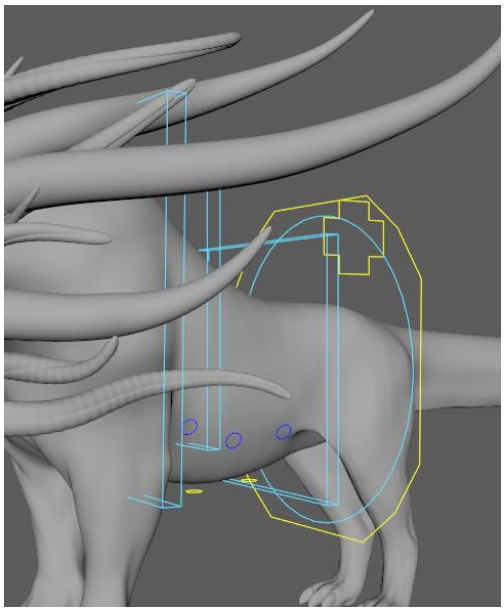
El monstre de "Path of Sand", no es un bípede sinó que, és un quadrúpede i necessita un sistema de deformació diferent.

Per les cames té un sistema IK que permet articular la cama i un altre sistema IK que permet rotar el taló cap endavant i cap a darrera. El sistema de la clavícula darrera es un sistema de clavícula normal de bípede. Però per la clavícula davantera té un sistema revers de la clavícula. Això permet treure el omòplat del quadrúpede



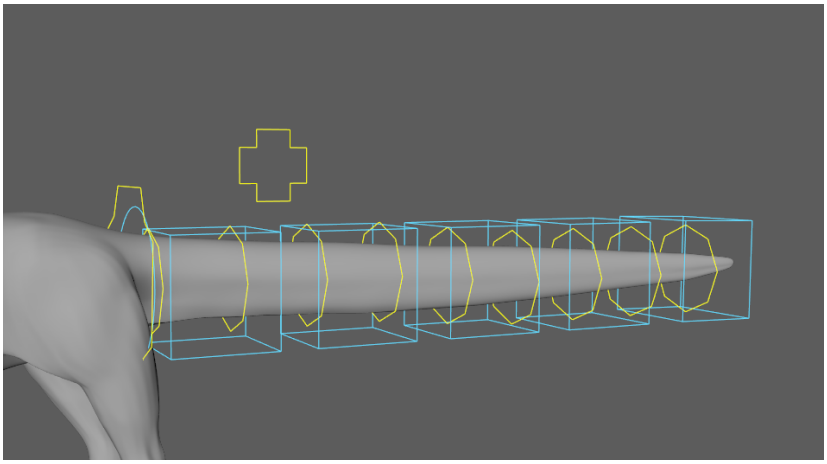
Il·lustració 47: Sistema de IK de les cames del Monstre

Per la esquena té un sistema de IK amb un control de tangència pel pit i un per la cintura. L'esquena té possibilitat d'allargar-se el percentatge que s'esculli. També hi ha un seguit de 9 controls, per a moure la panxa i les costelles del monstre.



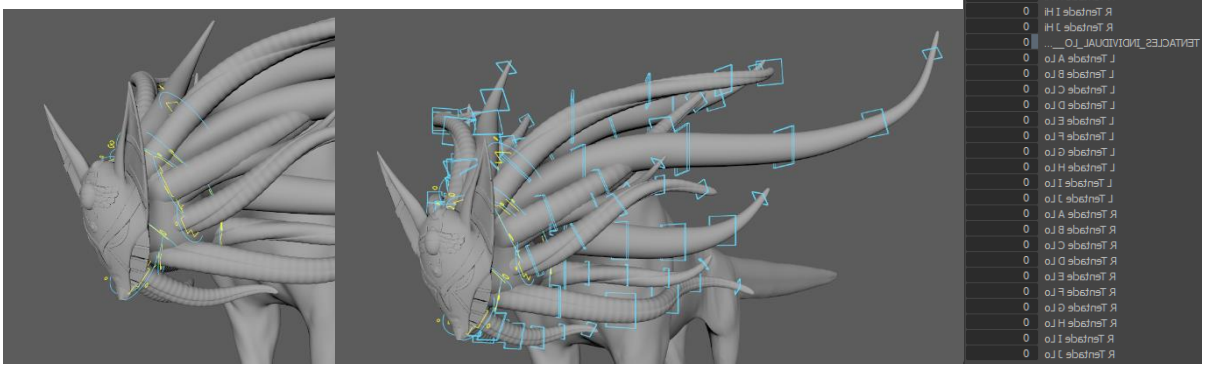
Il·lustració 48: Sistema de la esquena del monstre

La cua del monstre té un sistema de IK i un de FK que viu a sobre del de IK. També te possibilitat de estirar-se i contreure's. Els controladors del IK estan emparentats entre si per així definir millor la forma de la trajectòria que seguirà la cua.



Il·lustració 49: Sistema de IK de la cua

El monstre té un seguit de 21 tentacles al cap, animarlos totalment a mà es bastant laborios. Per tant tenen un sistema propi de control. Per defecte hi ha els controls de moure cada tentacle individualment en la seva totalitat. Es poden extreure els controls que defineixen la trajectòria principal del tentacle. Sota de sistema de trajectoria del IK te dues curves dinàmiques, una té aplicat un camp de turbulencia de baixa frequencia que li dóna la forma general i l'altre un camp de turbulencia de altre frequencia que li dóna el micro detall al tentacle. Per a que funcioni la evaluació dels nodes ha d'estar establerta en Dependency Graph.

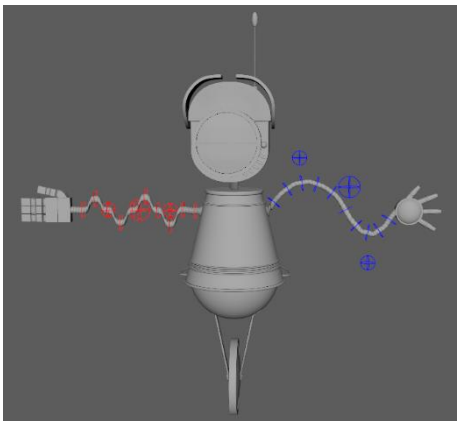


Il·lustració 50: Sistema dinàmic dirigit dels tentacles

3.5.5 Nut

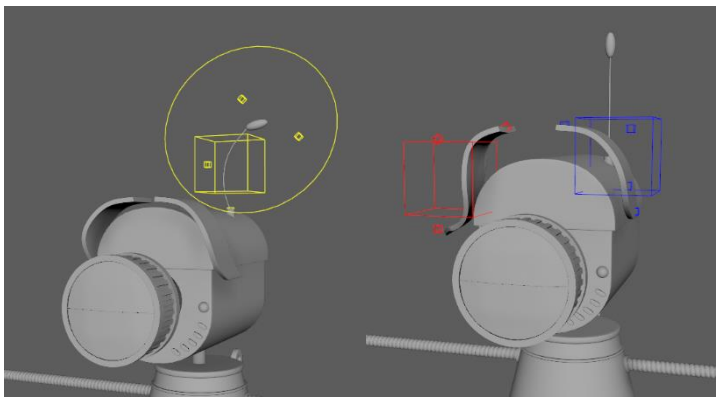
És el robot de "World Of Lost Things". Es un personatge fet a mesura ja que no és un bípede i necessita unes prestacions concretes per a animació.

En els braços te sistema de *bend bones*, per a donar-li ones al personatge.



Il·lustració 51: Sistema de ossos de doblegament als braços d'en Nut

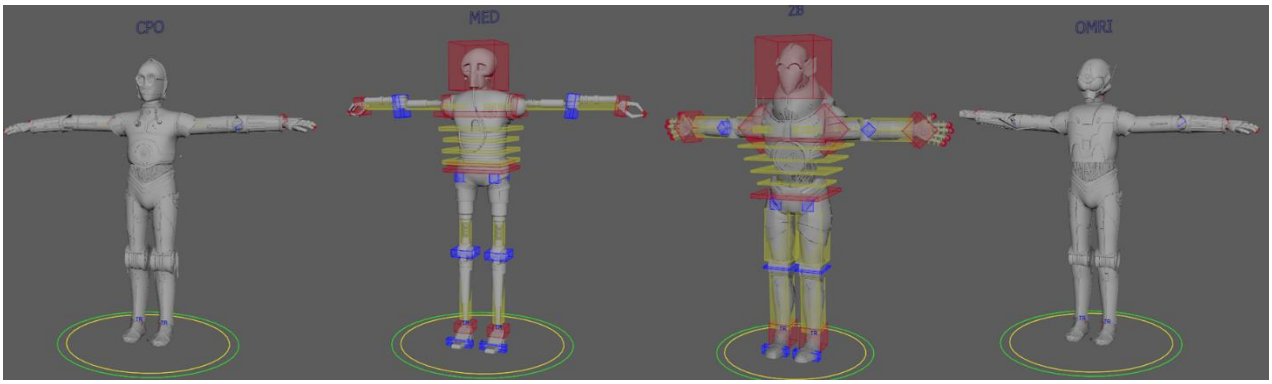
Tant en les celles com en l'antena té un sistema de IK, amb dos controls principals i dos de tangència que li dóna la possibilitat de estirar i fer formes orgàniques a objectes estàtics i mecànics.



Il·lustració 52: Sistema de control IK per a en Nut

3.5.6 Human IK

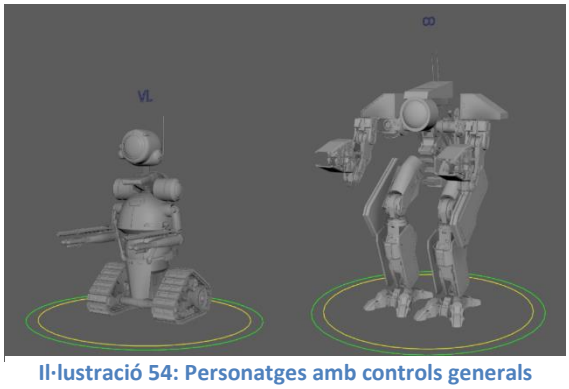
Pel projecte de "La Universitat de Cardiff" es necessitaven personatges que poguessin funcionar amb el sistema de mocap. Uns havien de portar animació a sobre i els altres només les dades del mocap. El sistema també és procedural.



Il·lustració 53: Personatges Human IK

3.5.7 Estàtics

També es necessitaven dos personatges que fossin estàtics però que es pogués animar la seva posició rotació i escala.



Il·lustració 54: Personatges amb controls generals

4 Pipeline

El *pipeline* és la part de la producció que s'encarrega de la gestió del intercanvi de les dades, entre les parts de la producció. La correcte gestió de les versions es la clau per que una producció funcioni correctament i de manera ordenada.

Hi ha eines que controlen totes aquestes informacions, per exemple: "Ftrack" i "Shotgun".

D'altre banda, també s'ha de controlar la gestió de les escenes, es pot fer amb varis sistemes i programes. Com a assemblers hi ha: "Katana", "Clarise" i "Houdini", són programes que son capaços de gestionar gran quantitat de geometria. També hi ha el sistema USD, desenvolupat per "Pixar", que ajuda a la gestió entre tots els programes, i per visualitzar amb "Hydra" un viewport amb els objectes.

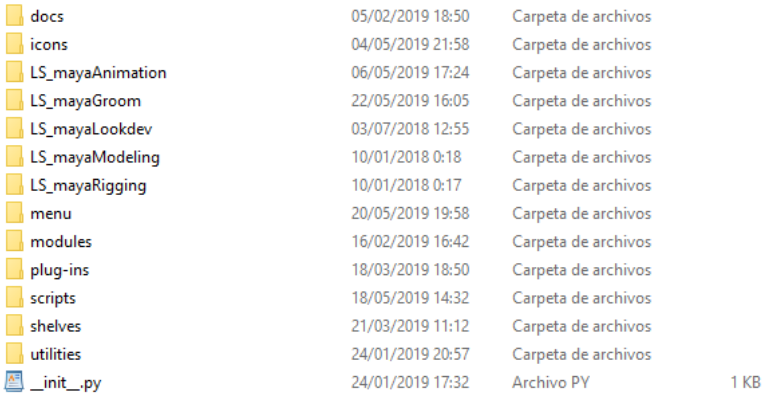
Per a projectes es va decidir no usar cap d'aquests sistemes perquè no els dominàvem i per que no hi havia suport. Això ha complicat tota la gestió del projecte, ja que tota la gestió del *asset* es feia de manera manual, on l'error humà es més present.

Com a treball pràctic de *pipeline* vaig desenvolupar un sistema d'eines (pròpies i ja creades) estiguessin a tots els ordinadors, només posant-les en una carpeta del server. Aquest sistema es va anomenar "LS_mayaToolkit". Per que funcioni es necessita posar el següent script en un fitxer userSetup.py en la ruta de scripts del Maya.

```
import os
import pymel.core as pm

LS_PATH = 'T:\\maya'
if LS_PATH in os.sys.path:
        os.sys.path.remove(LS_PATH)
os.sys.path.insert(0, LS_PATH)

pm.evalDeferred('import LS_mayaToolset')
```

Taula 1: Fitxer userSetup.py

Allà es troba a un recull d'eines i scripts útils per a producció.



Il·lustració 55: Estructura de carpetes d'utilitats en el servidor

4.1 Tools visibles

LS_Menu: És la visualització de un seguit d'eines en un menú a Maya.

LS_mayaAnimation: conté eines d'ajuda a l'animació

- aTools
- AnimFix
- Importador de personatges

LS_mayaGroom: utilitats per a la gestió del *groom*

- Importació
- Exportació

LS_mayaLookdev:

- Transformacions de llums
- Canvi de les exposicions de les llums seleccionades
- Duplicat d'objectes amb historia
- Canvi de perfils de lectura de color quan s'usa el OCIO ACES 1.0.3

LS_mayaModeling

- absSymMesh: Eina per operar entre geometries, revisar la simetria i posar-la bé si escau.
- Braid Creation: Eina per a crear trenes
- goZ: Eina que intercanvia la escena de Maya a zBrush i de zBrush a Maya.

4.2 Tools no visibles

Són les eines no visibles i que nomes funcionen per comanda.

4.2.1 Mòduls

- brSmoothWeights: desenvolupat per "brave ravit", té la funció de suavitzar els pesos de la geometria amb un algorisme més avançat que el que esta integrat en Maya

4.2.2 Plug-ins

- AnimSchoolPicker: Disseny i us de una GUI per a fer sets de selecció de controls, es pot dissenyar al gust. Serveix per optimitzar la selecció de controls. Creada per Anim School.
- dcSkin: Desembolupat per David Cuellar, eina per a salvar i carregar els pesos en fitxers binaris.
- extractDeltas: Desenvolupada per "brave rabbit", serveix per a extreure la variació que hi ha entre una *shape* correctiva i la mateixa amb els deformadors aplicats. Com diu el seu nom extreu els deltes de la *mesh*.

- jQuadCloth: Desenvolupat per Jacopo Ortolani. Fa retopologies de geometries amb molts polígons per a unes on esta tot quadrangulart. Funciona especialment amb robes.
- ngSkinTools: Desenvolupada per VIKtoras Makauskas. Serveix per a operar amb mapes de pesos. Dóna la possibilitat de tenir Capes additius de pesos. Internament fa tots els càlculs per que estiguin normalitzats.
- smoothSkinClusterWeight: Desenvolupada per "brave rabbit", es el predecessor del mòdul de brSmoothWeights, no té tantes utilitats i funciona més lentament
- transferSkinCluster: Desenvolupat per "brave rabbit". Serveix per a copiar *skinning* entre geometries.

4.2.3 Scripts

- advancedSkeleton5: Auto *rig* per a la creació de estructures per a controlar la geometria.
- cometScripts: Toolkit desenvolupat per Michael Comet, conté eines de tot tipus. La més coneguda es el Comet Renamer
- cvshapeinverter: Desenvolupat per Chad Vernon, serveix per a treure la *shape* inversa a una deformació, la manté de manera dinàmica.
- MG-PickerStudio: *Picker* similar al AnimShoolPicker, desenvolupat per Miguel Gao.
- mGear: Rigging framework desenvolupat per Miquel Campos, dóna una gran flexibilitzar per a crear estructures de *rig*,
- ml_tools: Seguit d'eines i utilitats per a animació i *rigging*, desenvolupat per Morgan Loomis
- studioLibrary: Eina generada per l'usuari de github "krathjen", per a guardar poses i animacions i així poder transferir-les entre arxius.
- Zen: Toolkit desenvolupat per David Belais, conte un seguit de eines de tot tipus extres a Maya
- zooTools: Seguit d'eines desenvolupades per Andrew Silke
- cosmos: Eina que gestiona l'accés a les finestres de Maya. Desenvolupat per Martin Gunnarsson.
- sortCircleTool: Eina per a fer cercles en el seguit de *edges* seleccionats.
- bosSmear: Eina per a fer deformacions de la geometria segons el tir de càmera.
- bSkinSaver:Eina desenvolupada per Thomas Bittner, per a guardar els pesos dels nodes *d'skinCluster*.
- changeColors: Script per a canviar els colors dels controladors en grup
- compactRenamer: Script de ErIK Lehmann per a reanomenar geometries
- delete turtle: Script per borar el node de *turtle* de la escena
- DPK_reorderAttrs: Eina desenvolupada per Daniel Pook-Kolb per a canviar els atributs en el *channel box*.
- Dupes: Script per a revisar si hi ha objectes repetits en escena. Desenvolupat per Jorn-Harald Paulsen.
- mtAlignTool: Eina per alinear objectes en el viewport
- tweenMachine: Eina per interpolar keyframes, desenvolupada per Justin S Barrett

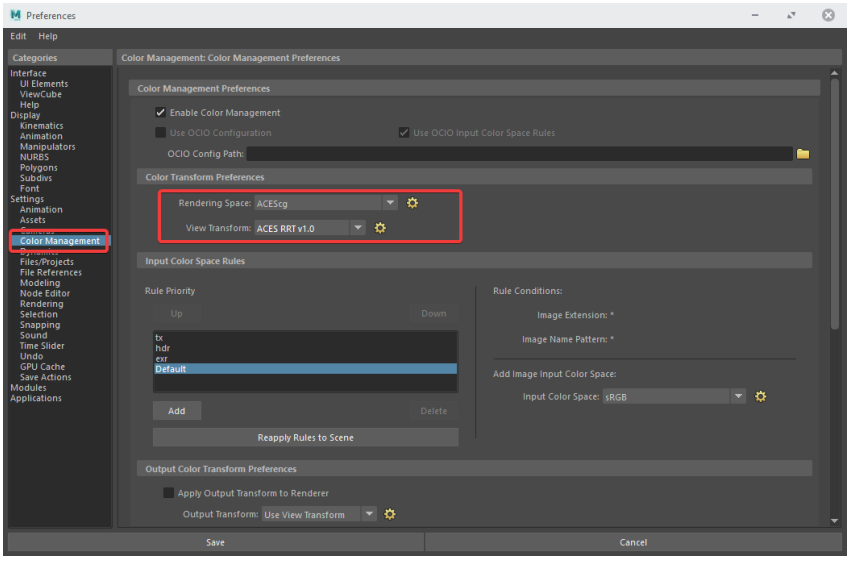
4.3 ACES

El Academy Color Encoding System (ACES) s'està establint com l'estàndard de gestió del color a través de la producció de de cine i tele.

Esta establert com a procés de gestió de color lineal i sense pèrdues. Així, la cadena de color es manté inalterada des de la gravació de la peça audiovisual fins a la visualització, passant per la creació del 3D.

Il·lustració 56: Estructura de gestió de les dades ACES a través de la producció

Per a usar la configuració del ACES dintre de Maya primer s'ha de baixar el repositori de OCIO de Sony (https://github.com/imageworks/OpenColorIO-Configs/tree/master/aces_1.0.3). Un cop descarregat s'ha d'activar la següent configuració.



Il·lustració 57: Configuració de la gestió del color dins de Maya amb els perfils ACES

I canviar la configuració de l'espai de color de tots els fitxes que estan carregats s'han de canviar a *utilities* (https://github.com/enriquevelmai/utils/blob/master/toAces.py)

4.4 Script de "current"

Com que no s'usava cap gestor de versions es va crear un script de corrent, que el que feia era llegir les ultimes versions i crear una carpeta amb elles.

5 Groom

Gestionar el groom en el projecte, es va convertir en un mal de cap, ja que tots teníem molt poca experiència en la gestió del cabell en el *pipeline*.

Es va decidir usar xGen de Maya per a fer el cabell, després de fer una recerca a fons de com funciona internament el xGen, es va extreure una manera pròpia de gestionar el cabell.

El xGen funciona tenint una geometria base en la que se li apliquen un seguit de col·leccions i descripcions.

Les col·leccions son l'agrupació de moltes descripcions en una o varies geometries. En les descripcions és on es pentina el cabell des d'un seguit de guies, mapes i expressions. Els mapes que es pinten són en format PTX.

La informació del xGen es guarda en un node propi de color verd en una escena de Maya que es un *plug-in* que llegeix la informació de dos llocs: tot el que esta relacionat amb les col·leccions, descripcions i expressions es guarda en un arxiu .xgen amb les dades guardades; la informació dels mapes es guarda en una jerarquia de carpetes dins de la carpeta xGen del projecte (o bé del lloc on es defineix) on hi ha carpetes per a cada col·lecció, descripció i modificador.

5.1 Estructura del arxiu .xgen

Dintre del arxiu de xgen les dades s'estructuren de la següent manera

5.1.1 Globals

La "*palette*" es la base, el que ve a ser la col·lecció.

Definició de les variables globals del xgen, mapeig de on esta la base del projecte.

```
Palette
    name            LB_Yura_default
    parent          groom
    xgDataPath      ${PROJECT}xgen/collections/LB_Yura_default
    xgProjectPath       //nassus/Project/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Grooming/
    xgDogTag
    endAttrs
```

Il·lustració 58: Opcions globals de la col·lecció

5.1.2 Per descripció

Dades de generals de descripció, hi ha una per cada descripció

```
Description
    name            Fringe
    flipNormals     false
    strayPercentage     0.0
    lodFlag         false
    averageWidth        1.0
    pixelCullSize       0.0
    pixelFadeSize       20.0
    cullFade        0.1
    minDensity      0.01
    cullWidthRatio      0.01
    maxWidthRatio       20.0
    groom
    descriptionId       1
    xgDataPath      ${PROJECT}xgen/collections/LB_Yura_default/
    xgProjectPath       //nassus/Project/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Grooming/
```

Il·lustració 59: Opcions generals de la descripció

Visualització del les guies en el *viewport*

```
GeometryRenderer
    percent         25.0
    startPercent        0.0
    inCameraOnly        false
    inCameraMargin      0.0
    outputInstances     true
    useWidthRamp        true
    geometryType        0
    shapeType       0
    convertSelected     false
    combineMesh     false
    createStripJoints   false
    stripJointPlacementType 0
    jointNumOnStrip     3
    createGuideJoints   false
    guideJointPlacementType 0
    jointNumOnGuide     3
    meshOrientation     0
    insertWidthSpan     false
    widthSpanNum        1
    curvature       0.500000
    uvInTiles       true
    uvLayoutType        0
    uvTileSeparation    0.000000
    endAttrs
```

Il·lustració 60: Opcions de visualització de les primitiva en viewport

Visualització de les guies a render, es pot definir el tipus de render que es vol, quina *scalp* segueix i si te *motion blur* o no. Es poden fer *overrides* de cada una de les variables. Es pot observar que el motor de render que usa el xgen es “RenderMan”, ja que és el motor de render de Pixar.

```
RendermanRenderer
    percent         100.0
    startPercent        0.0
    inCameraOnly        false
    inCameraMargin      0.0
    length_XP       true
    width_XP        true
    T_XP            false
    stray_XP        false
    id_XP           false
    descid_XP       false
    ri_XP           true
    rf_XP           true
    u_XS            true
    v_XS            true
    faceid_XS       true
    geomid_XS       false
    geomName_XS     true
    P_XS            true
    Pref_XS         false
    Pg_XS           false
    Prefg_XS        false
    N_XS            true
    Ng_XS           false
    Nref_XS         false
    Nrefg_XS        false
    dPdu_XS         true
    dPduref_XS      false
    dPdug_XS        false
    dPdurefg_XS     false
    dPdv_XS         true
    dPdvref_XS      false
    dPdvg_XS        false
    dPdvrefg_XS     false
    renderer        None
    renderMethod        2
    draMode         0
    primitiveBound      1.0
    custom__arnold_rendermode   0
    custom__arnold_curveMode    0
    custom__arnold_minPixelWidth    0.0
    custom__arnold_motion_blur  0
    custom__arnold_motion_blur_mode 1
    custom__arnold_motion_blur_steps    2
    custom__arnold_motion_blur_factor   0.5
    custom__arnold_useAuxRenderPatch    0
    custom__arnold_auxRenderPatch   0
    custom__arnold_multithreading   1
    endAttrs
```

Il·lustració 61: Opcions de visualització de les primitives en render

També es poden definir les dades generals de les guies de la descripció.

```
SplinePrimitive
    _patchNames
    length          $a=1.0000;#0.05,5.0\n$a
    width           rand(0.01,0.03,163)
    depth           $a=1.0;#0.05,5.0\n$a
    offU            $a=0.0000;#-2.0,2.0\n$a
    offV            $a=0.0000;#-2.0,2.0\n$a
    offN            $a=0.0000;#-180.0,180.0\n$a
    aboutN          $a=0.0000;#-180.0,180.0\n$a
    regionMap       ${DESC}/RegionFringe
    regionMask      1
    iMethod         1
    useCache        false
    liveMode        true
    _wireNames
    cacheFileName       ${DESC}/guides.abc
    attrCVCount     3
    bendParam[0]        $a=0.5000;#0.0,1.0\n$a
    bendU[0]        $a=0.0000;#-2.0,2.0\n$a
    bendV[0]        $a=0.0000;#-2.0,2.0\n$a
    fxCVCount       35
    uniformCVs      true
    taper           $a=0.0000;#-1.0,1.0\n$a
    taperStart      $a=0.0000;#0.0,1.0\n$a
    displayWidth        true
    faceCamera      true
    tubeShade       false
    tubes
    guideSpacing        1.0
    guideMask       1.0
    cutParam        1.0
    texelsPerUnit       10.0
    CVFrequency     1.0
    widthRamp       rampUI(0.0,0.605263157895,1:0.152597402597,0.75,2:0.808441558442,0.697368421053,1:1.0,0.460526315789,1)
    endAttrs
```

Il·lustració 62: Opcions generals de les guies en viewport

Tot seguit estan les dades dels modificadors, es carreguen com a paquets de FX (establerts per la API).

```
ClumpingFXModule
    active          true
    mask            $a=map('${DESC}/paintmaps/FringeMask2');#3dpaint,256.0\n$a\n
    name            Clumping1
    cvAttr          false
    mapInitialized      True
    pointDir        ${DESC}/${FXMODULE}/Points/
    mapDir          ${DESC}/${FXMODULE}/Maps/
    clump           1
    clumpScale      rampUI(0.0132450331126,0.539473684211,3:0.331125827815,0.447368421053,1:0.634868421053,0.276315789474,1:1.0,0.0,1)
    clumpVolumize       false
    clumpVariance       0.0
    cut         0.0
    copy            0.0
    copyScale       rampUI(0.0,0.0,3)
    copyVariance        0.0
    curl            0.0
    curlScale       rampUI(0.0,0.5,3)
    offset          0.0
    offsetScale     rampUI(0.0,0.0,3:0.5,1.0,3:1.0,0.0,3)
    flatness        0.0
    flatnessScale       rampUI(0.0,0.0,3)
    frame           0.0
    noise           0.0
    noiseScale      rampUI(0.0,0.0,3)
    noiseFrequency      0.0
    noiseCorrelation    0.0
    exportCurves        false
    exportDir       curves/
    exportFaces
    texelsPerUnit       10.0
    radiusVariance      0.5
    ptDensity       1.0
    ptMask          1.0
    ptLength        1.0
    colorPreview        false
    useControlMaps      1
    controlMask     1
    controlMapDir       ${DESC}/RegionFringe
    endAttrs
```

Il·lustració 63: Opcions del modificador de clump

Altres exemples de dades que es guarden d'altres tipus de modificadors.

```
NoiseFXModule
    active          false
    mask            1.0
    name            Noise3
    frequency       1.0
    magnitude       $min=0.0100;#0.00,0.20 \n$max=0.2000;#0.20,10.00\n$seed=4;#0,10\n$mag=rand($min,$max,$seed);
    magnitudeScale      rampUI(0.0,0.0,1:0.370860927152,0.421052631579,1:1.0,0.75,1)
    correlation     0.0
    preserveLength      0.0
    mode            0
    bakeDir         ${DESC}/${FXMODULE}/
    endAttrs

CutFXModule
    active          true
    mask            1.0
    name            Cut1
    amount          rand(0.0,0.2)
    rebuildType     1
    endAttrs
```

Il·lustració 64: Opcions dels modificadors de Noise i Cut

L'espai de *random generator* s'encarrega de gestionar les propietats de densitat, offset, entre altres genèriques.

```
RandomGenerator
    displacement        $a=0.0000;#-1.0,1.0\n$a
    vectorDisplacement  0
    bump            $a=0.0000;#-1.0,1.0\n$a
    offset          $a=0.0000;#-1.0,1.0\n$a
    cullFlag        false
    cullBackface        false
    cullFrustrum        false
    cullAngleBF     0.0
    cullAngleF      0.0
    cullExpr        $a=0.0000;#0.0,1.0\n$a
    density         250.0
    mask            $a=map('${DESC}/paintmaps/DensityFringe');#3dpaint,256.0\n$a\n
    dcFlag          false
    scFlag          true
    usePoints       false
    pointDir        ${DESC}/Points/
    ptLength        1.0
    endAttrs
```

Il·lustració 65: Opcions generals de la descripció

Propietats de visualització de les guies en el *viewport*.

```
GLRenderer
    percent         100.0
    startPercent        0.0
    inCameraOnly        true
    inCameraMargin      0.0
    patchNames      false
    faceIds         false
    primIDs         false
    primIDsAt       1.0
    vertices        false
    poly            false
    culled          false
    unitCube        false
    color           map('${DESC}/Clumping4/Maps/')
    guideColor      map('${DESC}/RegionFringe')\n#$a=[1.0,0.4313725,0.0];#color\n#$a
    TEXCOORD3       [ $cWidth, 0, 0 ]  # red channel reserved by XGen
    TEXCOORD4
    TEXCOORD5
    TEXCOORD6
    TEXCOORD7
    splineSegments      2
    primNumLimit        100000000
    endAttrs
```

Il·lustració 66: Opcions de visualització de les guies en viewport

L'apartat de *map textures* diu d'on es llegeixen part dels mapes dels modificadors. També indica quines són les parts que estan actives del xgen per la descripció en qüestió i quin és el mètode de render que s'usa en el *viewport*.

```
MapTextures
    SplinePrimitive regionMap   P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yur
    Clumping1   mask    P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Groomi
    Clumping1   controlMapDir   P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yur
    Clumping2   controlMask P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Gr
    Clumping2   mask    P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Groomi
    Clumping3   controlMask P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Gr
    Clumping3   mask    P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Groomi
    Clumping4   controlMask P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Gr
    Clumping4   mask    P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Groomi
    RandomGenerator mask    P:/VFX_Project_02/Last_in_Battle/PROJECT/01_Assets/01_Characters/Yura/06_Grooming/Lb_Yura_Gr
    endAttrs

    Active  SplinePrimitive
    Active  RandomGenerator
    Active  RendermanRenderer
    Preview GLRenderer
```

Il·lustració 67: Rutes de lectura dels mapes pintats

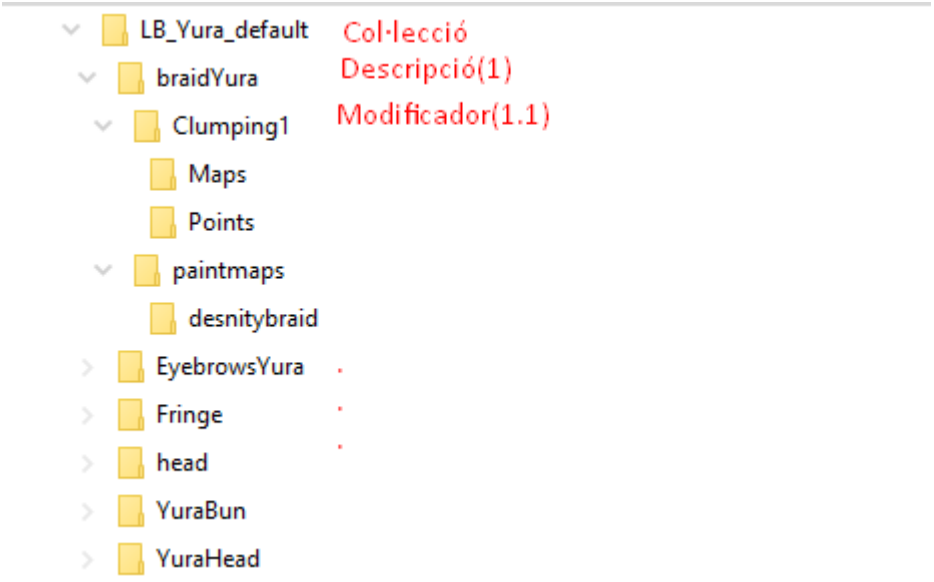
5.1.3 D'ancoratge

Un cop s'han definit totes les dades de visualització i modificació de cada una de les descripcions. Es defineixen totes les dades relacionades amb les *scalps*, on es veuen plasmades les posicions de cada una de les guies, els *patches* on s'enganxen, els cv que tenen i moltes més. A continuació hi ha un abstrac d'aquesta part del arxiu, ja que és la part de major extensió del arxiu.

```
Patches Fringe  75
Patch   Subd
    name        C_Scalp
    faceIds     478  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 3
    culledPrims 1   267 3  13 71 139
    animCurves  0
Guides  Spline  75
    id          153
    loc         2.4972599744796753e-01 4.3792399764060974e-01 255
    blend           0.0000000000000000e+00
    interp          2.4695467396369639e+00:2.4695367396369639e+00:1.7584112027669696e-01:1.6502745752980459e+00:8.951390
    CVs         10
    1.1121873270923022e+00 2.3192748608178423e-01 -4.6769724034053983e-01
    2.4943217153189980e+00 1.5723913981566215e-01 -6.0031637909566926e-01
    3.8385200089239566e+00 -1.1658772585250568e-02 -5.9032405953457645e-01
    5.0798875946303816e+00 -3.4336530525622094e-01 -8.8084518318328853e-01
    6.2716621619440565e+00 -8.1470959240584917e-01 -1.4184699771725648e+00
    7.3988718233863109e+00 -1.2610266666336747e+00 -2.0203560342579467e+00
    8.5462482870015268e+00 -1.7006292268026326e+00 -2.5085058680913224e+00
    9.6599110699878832e+00 -2.1136057934431891e+00 -3.1745458208112067e+00
    1.0304415305916406e+01 -2.4258583848547848e+00 -4.1200042724164705e+00
```

Il·lustració 68: Propietats d'ancoratge entre les guies i la geometria

5.2 Estructura de salvaguarda dels mapes

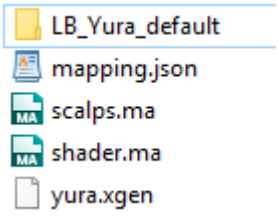


Il·lustració 69: Estructura de carpetes pel salvatge dels mapes

5.3 Concepte de packaging

Com es veu hi ha moltes variables que intervenen en la gestió del cabell, per tant es defineix el concepte de *packaging* del cabell. Es tracta de tenir un recipient que contingui totes les dades per a poder exportar i remuntar el sistema de deformació del cabell.

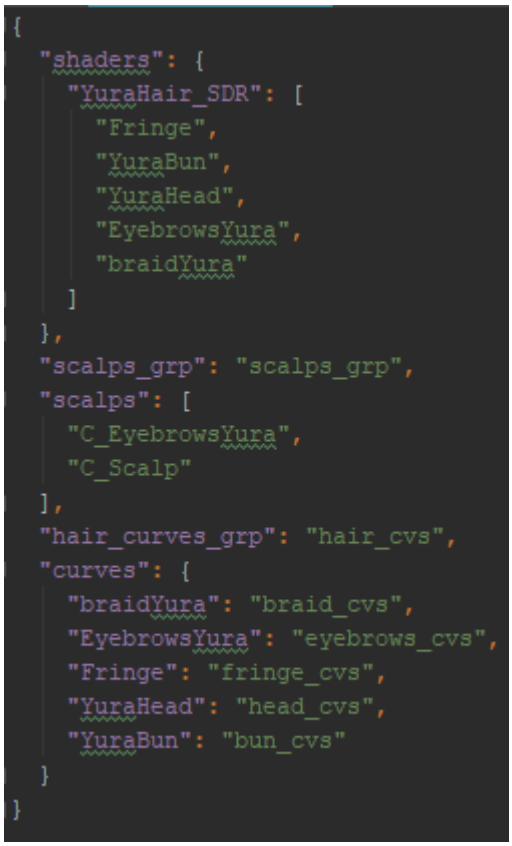
Per a poder gestionar el xGen, definim tenir els següents elements:



Il·lustració 70: Estructura pròpia de paquet de groom

La carpeta conte tota la estructura de mapes PTX dels modificadors i les descripcions.

El arxiu de mapping.json conté informació útil per a remuntar el *groom*.



Il·lustració 71: Estructura JSON pròpia de guardat de informació

L'arxiu de scalps.ma conte les *scapls* des d'on creix el cabell, s'ha d'exportar mantenint la historia de les geometries ja que tenen connectats mapes i arxius files per a que es pugui remuntar sense pèrdues.

L'arxiu shader.ma conté el material del cabell.

El fitxer .xgen és l'exportació del arxiu esmentat anteriorment.

5.4 Animació contra Simulació

Hi ha dues maneres de gestionar que el cabell es mogui. La primera és simulant-lo, un cop esta feta la animació, fer que el cabell es mogui seguint unes dinàmiques físiques i uns camps de força. La segona es animant una *mesh* proxy on s'enganxen unes corbes i aquestes s'usen per a moure el *groom*. És important exportar un arxiu .abc amb les guies especifiques per a dada descripció i per a cada *shot*.

5.5 Gestió de les dades

Per a gestionar aquestes dades sense caure en errors humans, ja que hi ha molts passos, definim un script per a que s'encarregui d'aquestes coses .

Les següents llibreries donen peu al us de les comandes de xGen:

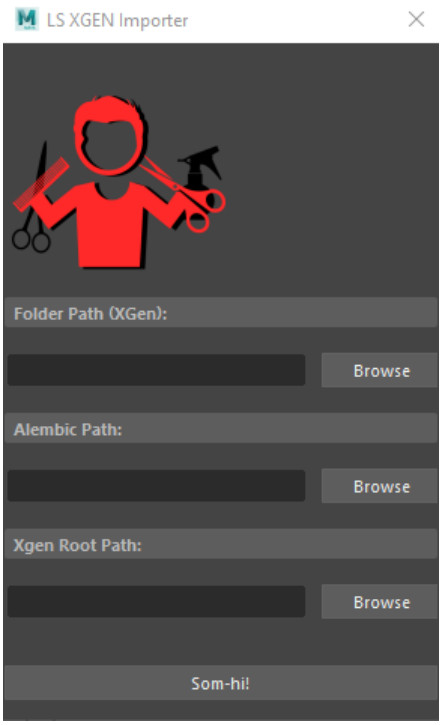
```
import xgenm as xg
import xgenm.XgExternalAPI as xge
import xgenm.xgGlobal as xgg
```

Taula 2: Importació de les llibreries d'xGen

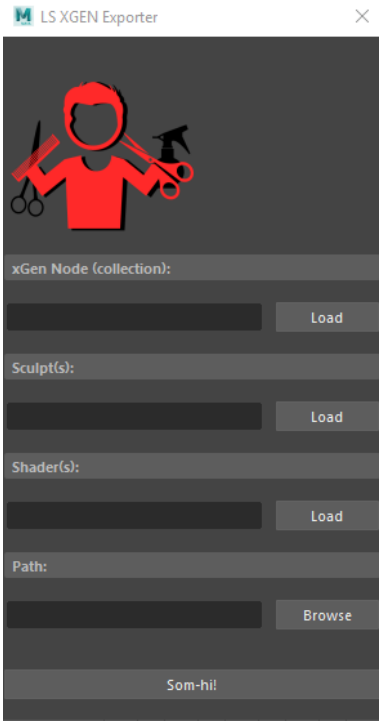
Aquestes eines es dediquen a importar i a exportar de manera genèrica el *groom* genèric.

El exportador necessita definir el node de xGen, seleccionar les *scalps*, el *shader* i el *path* on s'exportarà el *package* definit abans.

El importador necessita el *path* del *package*, el alembic amb la animació del *shot* i el *path* del xGen en cas de que hagi canviat.

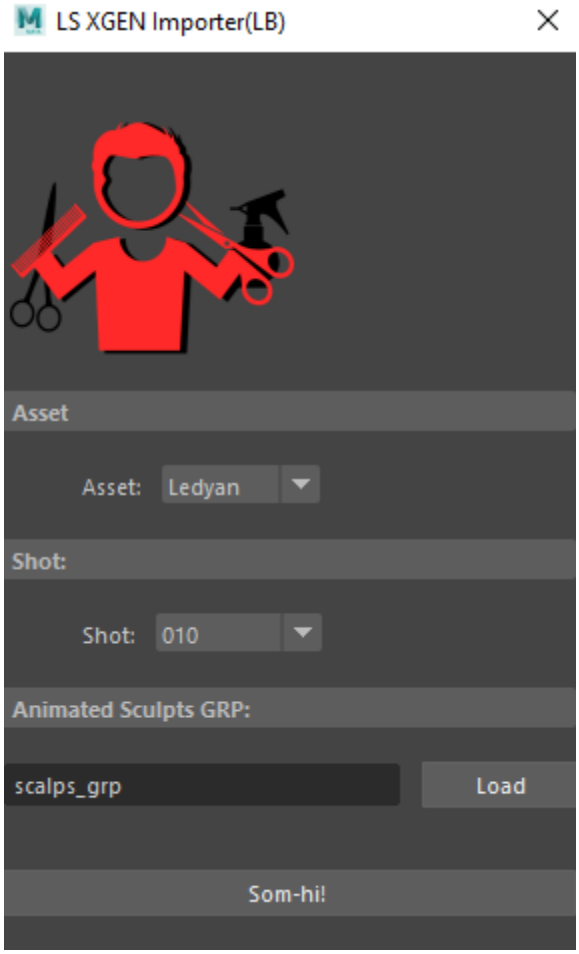


Il·lustració 73: Eina per importar el xGen



Il·lustració 72:Eina per exportar el xGen

A més a més de les tools de abans hi ha una tool especifica per a Last in Battle, basada en la seva propia gestió d'arxius. Aquí només fa falta definir el asset, el shot i el grup d'scalps.



Il·lustració 74: Eina per importar el xGen, adaptada per Last in Battle

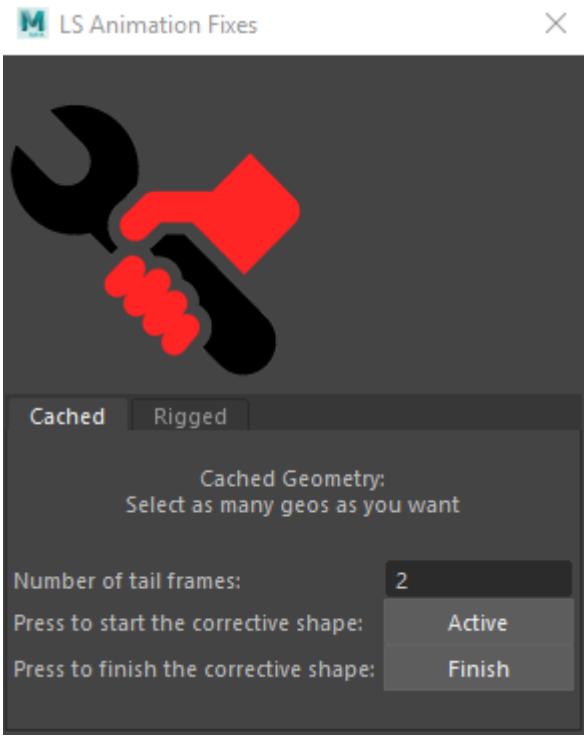
6 Fixing

Un cop l'animació està feta, pot ser que hi hagi penetracions de la geometria o no s'aconsegueix la deformació desitjada, s'ha de passar a la part de *shot sculpting*.

On s'agafa el cache de l'animació i amb la següent eina, es modela la deformació desitjada.

Mètode d'ús:

1. Posicionar-se en el *frame* on es vol modelar la correcció
2. Obrir l'eina
3. Seleccionar les geometries a les que es vol modelar la correcció
4. Definir les cues de *frames* que tindrà la correcció (una *blendshape* amb 3 keys, dos de 0 i un de 1)
5. Aplicar el botó de *Active* i modelar-les
6. Aplicar el botó de *Finish*, s'esborren les geometries



Il·lustració 75: Eina per a fer *shot sculpting*

7 Conclusions

La realització d'aquest projecte ha servit per poder tenir una visió global i completa del desenvolupament d'un projecte des d'inici a fi. Passant per cada un dels àmbits he pogut veure quines són les complicacions i els processos a seguir en cada fase. Ha costat molt saber quan tancar un procés, és a dir, quan establir que el model, textures, *rig*, animació, ... ja era suficient i s'havia de deixar fluir *l'asset* pel *pipe*. Al treballar en grup ens hem hagut d'organitzar, ja que no és el mateix fer un projecte íntegrament sol. Les escenes havien de tenir un nom concret, el ordre en *l'outliner* havia d'estar molt cuidat i els materials publicats amb els mínims errors possibles. Al ser la primera vegada que ens enfrontàvem, com universitat i com a grup, a tota la producció ha estat un repte la manera en la que havíem de procedir. Per exemple, com gestionar el cabell. Com no el simulàvem sinó que l'animàvem, va costar trobar la manera de juntar l'animació de les guies amb les guies del *groom*. Un cop ja funcionava en local ens vam adonar que en granja no funcionava, ja que el "Deadline" no canviava les rutes d'unitat de xarxa configurada a una lletra per la ruta sencera en els arxius de xGen. Un altra dificultat va ser com s'ajuntava els materials amb la geometria animada. Es van plantejar les opcions de: el rig porta els materials, es crea un codi que importi i configuri els materias un cop l'animació estava computada o un sistema de l'Alex Leon que consistia en armar una escena amb els materials aplicats i després que quedes com a referencia en l'escena d'il·luminació.

Crear un sistema totalment procedural i orientat a objectes és més tediós del que sembla. S'ha d'estructurar i dissenyar prèviament bastat, fent suposicions dels obstacles que aniran apareixent pel camí. També he estat pendent dels nous mètodes que es van publicant i les eines que van sortit, per així poder-les adaptar ràpidament al sistema creat. Personalment, se m'ha fet molt dur crear i mantenir 13 personatges amb tots els fixes de *rig* i els canvis de geometria. Ara veient-ho amb perspectiva veig que la opció de triar fer els *rigs* procedurals ha esta una bona decisió, ja que em veig incapaç de haver-ho fet de la manera "artesanal", a demés de la implementació d'eines. Un dels problemes principals que va haver-hi en els *rigs* va ser el *flipping* del advanced twist dels sistemes d'IK spline, despès de dies de baralles vaig portar el problema a la feina i em van aconsellar que uses el IK spline per calcular la posició i que d'allà crees una corba d'offset a la del IK per marcar la posició i usar una projecció d'aquesta en la corba d'offset, usant aquesta segona corba com a up vector d'un sistema *d'aim*, on el os apunta al següent. Aquesta solució va arreglar els *flippings*, ells mòduls d'extremitats, coll, esquena i tentacles porten implementats aquests sistemes. Les dobles transformacions van causar també mal de caps a l'hora d'escalar el model, crear grups que no interessin les transformacions en cada mòdul de *rig* va ser la solució a aquest problema.

De tots els personatges el que va ser un repte major per la complexitat dels sistemes va ser el monstre, degut als seus tentacles. Tot i que els sistemes funcionen correctament, el *rig* va molt lent quan els automatismes s'activen. M'agradaria trobar una solució més endavant per arreglar aquest punt en contra del *rig*.

Els sistemes de projecció de nodes de transformació sobre d'una superfície ha sigut un dels punts més claus de tot el projecte, ja que ha donat molt de joc als sistemes facials de celles, llavis i parpelles.

El pintatge de pesos en *dual quaterion* en algunes parts ha estalviat el modelat de moltes formes correctives. L'ús de la nova eina de brSmoothSkinWeights ha estat també un dels majors descobriments de l'any, ha agilitzat molt la feina.

Al ser una persona sola, que ha desenvolupat el sistema i l'ha estat mantenint, el temps de producció ha estat limitat i molts sistemes que estaven plantejats com ha extres no s'han pogut realitzar. Sort de les companyes que han estat allà ajudant i fent que les coses funcionin des de els departaments anteriors de *groom* i modelat. I l'ajuda que va haver en començar a modelar les *shapes* facials. Hauria ajudat també tenir algú present al dia a dia que donés un cop de mà a la programació de les eines ja que moltes coses se m'escapen i tenir algú amb experiència hauria estat molt productiu i hauria après altres maneres de procedir, que no només la meva.

Actualment el codi del Rigging Toolkit està molt brut a nivell de codi. M'agradaria donar-li una volta al sistema i abstraure'l encara més, per a deixar només les classes més genèriques i limitar el rig a un sistema de spline i un de projecció sobre de una superfície i que a partir d'aquests tots s'heretin. També canviar l'ús de maya.cmds i usar PyMel ja que esta totalment orientat a objectes. Apart estaria molt bé dotar-li d'una interfície gràfica per fer-lo més usable.

M'ha agradat molt el desenvolupament d'eines per a *pipeline* , ja que era una cosa que no havia fet mai i ha sortit força bé

8 Referencies

[1] http://introtorigging.blogspot.com

9 Bibliografia i Webgrafia

ROB O'NEILL. *Digital Character Development: Theory and Practic.* Second Edition. A K Peters/CRC Press. 27 de octubre de 2015. ISBN: 1482250772

W. ELLENBERGER. *An Atlas of Animal Anatomy for Artists (Dover Anatomy for Artists).* Dover Publications Inc.; Edición: New impression. 1 de diciembre de 1966. ISBN: 0486200825

ELIOT GOLDFINGER. *Animal Anatomy for Artists: The Elements of Form*. OUP USA. 11 de marzo de 2004. ISBN: 0195142144

ALBERTO LOLLI. *Struttura uomo. Manuale di anatomia artística*. Colla Editore; Edición: 2. 19 de abril de 2018. ISBN: 8894272214

WILLIAM C VAUGHAN. *The Pushing Points Topology Workbook: Volume 01*. CreateSpace Independent Publishing Platform. 5 de abril de 2018. ISBN: 1987728610

TINA O'HAILEY. *Rig it Right! Maya Animation Rigging Concepts, 2nd Edition*. Routledge; Edición: 2. 27 de julio de 2018. ISBN: 9781138303164

JASON OSIPA. *Stop Staring: Facial Modeling and Animation Done Right*. John Wiley & Sons Ltd; Edición: 3rd. 8 de octubre de 2010. ISBN: 0470609907

GOPINATH JAGANMOHAN, VENKATESHWARAN LOGANATHAN. *PySide GUI Application Development - Second Edition*. Packt Publishing; Edición: 2nd Revised Edition. 28 de enero de 2016. ISBN: 178528245X

ADRIAN HERBEZ. *Maya Programming with Python Cookbook*. Packt Publishing. 29 de julio de 2016. ISBN: 1785283987

ADAM MECHTLEY, RYAN TROWBRIDGE. *Maya Python for Games and Film: A Complete Reference for Maya Python and the Maya Python API.* CRC Press; Edición: 1. 1 de noviembre de 2011. ISBN: 0123785782

ROBERT GALANAKIS. *Practical Maya Programming with Python*. Packt Publishing. 25 de julio de 2014. ISBN: 1849694729

MARK SUMMERFIELD. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. Prentice Hall; Edición: 01. 4 de septiembre de 2015. ISBN: 0134393333

KIARAN RITCHIE, JAKE CALLERY, KARIM BIR. *The Art of Rigging. Volume I-II-III*.

XGen Python API. Disponible en: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-33ECC43B-5CF6-4BE1-8EAE-8C6C0D698020-htm.html

Python API 2.0: Disponible en: http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__files_GUID_B1CD0989_EB49_46BE_934F_F23BFB869142_htm

PyMEL for Maya. Disponible en: http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__PyMel_index_html

Maya Commands. Disponible en: http://help.autodesk.com/view/MAYAUL/2019/ENU/?guid=__CommandsPython_index_html

Josh Sobel Face Rigging. Disponible en: https://vimeo.com/ondemand/sobelfacerig/

Cult of Rig. Disponibble en: http://www.cultofrig.com/

Skinning Techniques. Disponible en: http://www.3dfiggins.com/writeups/paintingWeights/

Blendshape Techniques. Disponible en: http://petershipkov.com/tutorials/blendShapes/blendShapes.htm

Bind Pose. Disponible en https://bindpose.com/

*Doug Schiebe*r. Disponible en: https://www.dougschieber.com/new-index

MICHAEL TODD – xGen. Disponible en: https://www.mtodd.work/xgen-product-design

Alan Mckay Potcast: Disponible en https://www.allanmckay.com/1/

Beginners guide to: XGen pipeline for beginners. Disponible en: https://www.mikecauchiart.com/single-post/2017/08/29/Beginners-guide-to-XGen-pipeline-for-beginners