Escola Universitària d'Enginyeria Tècnica
de Telecomunicació La Salle

Trabajo de Fin de Grado (TFG)
Grado en Animación

Planificación de un Rig Modular

Alumno
Enrique Velasco Mairal

Profesor Ponente
Ignasi Duelo Fandos

Índice

Resumen

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 óptima de llevar a cabo el desarrollo de los personajes de los proyectos de 4.º.

Veremos que hay muchas maneras de proceder y que ninguna es menos apta que otra. Para saber qué método escoger, hay que entender qué técnicas existen, cuáles serán las acciones del personaje en escena y cuáles sus limitaciones, y cuánto 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.

Agradecimientos

Me gustaría dar las gracias a una serie de personas sin las cuales este trabajo no habría sido posible, ya sea por su guía, apoyo o experiencias.

Sergio Graña, Ona Molas, Enrique Alberola, Judit Navarro y Sancho Albano han estado, día a día, cuidando todos los aspectos previos y posteriores al rig para que los personajes pudieran ir a través del pipeline. Agradecer a las compañeras de clase y de proyectos, a los mentores y tutores.

Por otro lado, a los compañeros de trabajo: Felix Balbas, Vincenzo Leombruno, Alessandro Boschian, Albert Vivó, Roure Ossó, Xavier Roca Crespo y Sara Saez Hoces, por la guía y por la ayuda a la hora de resolver situaciones de los rigs y hacerme entender el conjunto del proceso.

También agradecer a mentores externos: Iker J. de los Mozos por sus experiencias explicadas en entrevistas y las largas conversaciones; y a las mentoras de Ilion, Silvia Montes e Isabel Bértolo, por aportar sus conocimientos sobre cómo funciona el pipeline de groom y de simulación de ropa.

A nivel de código, me gustaría agradecer a David González Cuéllar y a Vasil Shotarov por las bases que han prestado en los sistemas de guardado y lectura de los pesos de los deformadores y por el sistema de guardado y carga de las formas de los controles.

Acrónimos

ACES — Academy Color Encoding System

API — Application Programming Interface

FK — Forward Kinematics (cinemática directa)

FPS — Fotogramas por segundo

GPU — Graphics Processor Unit

IDE — Integrated Development Environment

IK — Inverse Kinematics (cinemática inversa)

PyCharm — IDE de programación basado en el lenguaje Python

Python — Lenguaje de programación de alto nivel

TFG — Trabajo de Fin de Grado (Treball Final de Grau)

USD — Universal Scene Description

UV — Espacio 2D de coordenadas para texturas; U y V definen las dos componentes

Introducción

Este trabajo se enfoca al desarrollo de la producción técnica de los proyectos de 4.º, específicamente: Last in Battle, Path of Sand y World Of Lost Things.

Se contemplan los apartados de: rig, groom, pipeline y herramientas. En cada uno se explica el proceso que se ha seguido, su definición y las dificultades encontradas. El objetivo no es únicamente explicar el proceso, sino que se pretende ampliar las partes más teóricas de los campos mencionados.

Como material resultante hay 6 rigs, de los cuales 5 son bípedos y uno cuadrúpedo, junto con scripts y utilidades recogidas bajo el LS_MENU dentro de Maya.

También se expone un proyecto sobre rigs preparados para recoger datos de mocap, hecho conjuntamente con la Cardiff School of Art & Design.

Todos los conceptos redactados en el documento están referidos y extraídos del software de 3D Autodesk Maya 2018.5.

La reel irá enfocada a la parte de rigging, que es a la que va más dirigido el trabajo y a la que se le ha dedicado más tiempo.

Concepto general

Antes de atacar el área de rigging, hay que ver qué es, de dónde viene y cómo ha evolucionado.

Definición

El rigging es la parte de la producción del 3D y los VFX que se encarga de crear las estructuras lógicas de deformación del personaje. Mediante los recursos del programa, es capaz de proporcionarle al personaje la habilidad de moverse “anatómicamente”.

No hace falta que sea anatómicamente perfecto, ni que respete todas las normas de la anatomía del personaje. Se busca más bien que el personaje tenga appeal. Si para hacerlo hay que romper las normas de la mecánica de la anatomía, se puede tomar la licencia artística de hacerlo.

Un personaje que funcione con una estructura anatómica correcta tendrá unas deformaciones más cuidadas que uno que no. Entran más variables, no solo las pertinentes al rig, para que el personaje funcione correctamente. Entran en juego variables que no son controlables en el departamento, como el diseño, la topología, el groom y la animación.

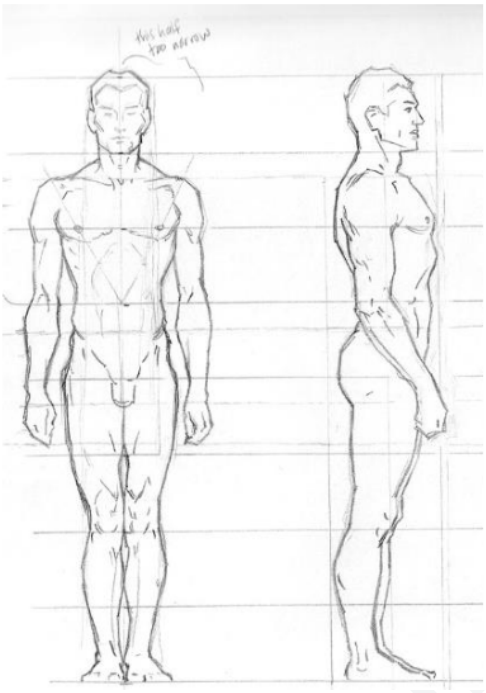

Ilustración 1: Esquema de anatomía corporal de Andrew Loomis

Objetivo

Se puede definir que el objetivo principal del rig es dotar al animador de controles, atributos y herramientas para poder dar expresividad al personaje.

Hay que tener en cuenta que, muy probablemente, la geometría esculpida por el animador en el viewport no será la que suba a render, sino que puede pasar por sistemas de músculos, telas, pelos y retoques.

En los personajes cartoon hay que procurar que todas las deformaciones sean limpias y redondas, y que las arrugas sirvan para marcar las líneas de expresión.

Una de las premisas importantes es que el rig debe poder ir a 25 fps. Esto es problemático ya que, por mucho que se optimicen los personajes, todo depende de:

- Cantidad de personajes en escena
- Funcionalidades de display del viewport
- Subdivisión de los objetos
- Visibilidad de los objetos
- Deformaciones activadas
- Evaluación del programa (DG, Paralelo, Serie)
- Evaluación de la GPU (override)

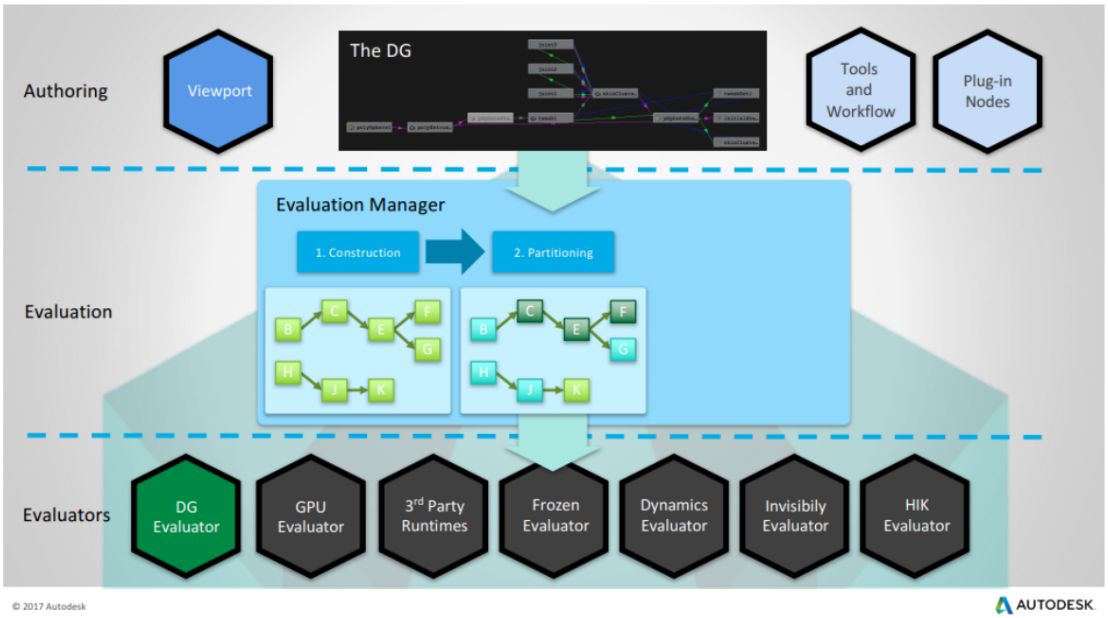

Ilustración 2: Esquema del orden de evaluación de los nodos dentro de Maya

Orígenes

El concepto de rigging tardó en surgir en la producción, aunque estuvo presente desde sus primeros inicios, incluso antes de empezar la producción 3D.

Se puede considerar que el rigging empieza en las primeras animaciones en stop-motion, cuando se vio la necesidad de poder articular los personajes para optimizar la producción y, así, no tener que hacer cada figura para cada fotograma.

Poco a poco fue evolucionando según las necesidades de cada una de las producciones, pasando del stop-motion a la articulación de robots y naves en las primeras pruebas de la producción 3D.

Ilustración 3: Armadura del animatrónico de King Kong hecha por Ray Harryhausen (1933)

Tipos y Técnicas

Según estética

Enfocándonos solo en el ámbito del 3D, se pueden extraer dos tipologías diferentes.

Realista: Toda la expresividad corporal y facial se aproxima más a la realidad. Las ropas y los cabellos se dejan para entornos de simulación. Desde la parte de rigging, se proporcionan sistemas para falsear las deformaciones musculares.

Cartoon: Las expresiones son más extremas; se aplican más drásticamente sistemas que den al animador la posibilidad de dibujar la forma que quiera con el personaje. Los conceptos de squash y stretch están más presentes.

Ilustración 4: A la izquierda está el wireframe de Gollum de "El señor de los anillos" desarrollado por Weta Digital, en referencia a la visualización de una geometría realista. A la derecha está el Capitán Charles T. Baker de "Planet 51" desarrollado por Ilion Animation Studios, que muestra una de las deformaciones máximas del personaje.

Según relevancia

Se pueden encontrar:

Personajes

Principales (Hero/A). Los que aparecen más tiempo en pantalla y los que tienen más relevancia a nivel narrativo. Son los que tienen sistemas de deformación más avanzados y personalizados.

Secundarios (B). Aparecen con menor frecuencia pero interactúan con los principales. Tienen un set de deformaciones completas pero no tan extremas como los A.

Terciarios. Aparecen de fondo o para rellenar escena. Suelen ser personajes genéricos con variaciones de ropas y peinados.

Props

Estáticos. Se basan en sistemas de pivotaje, ya que servirán para que los personajes los puedan coger o interactuar con ellos.

Animados. Tienen sistemas de deformación mecánica simple.

Vehículos. Tienen sistemas de interacción con el terreno, además de sistemas avanzados de deformación mecánica.

Simulación dirigida. Son sistemas donde el animador tiene la posibilidad de controlar el camino genérico de cada deformación pero que, por encima, tienen sistemas dinámicos. A los sistemas dinámicos se les aplican campos de deformación o expresión, ya que facilita la animación y la visualización final.

Según cámara

Finalmente, también se pueden separar por su proximidad a cámara: dependiendo de la distancia de cámara a personaje, se diseñará un sistema más complejo o más simple.

Procedimiento

Para la creación de assets riggeados hay varias maneras de proceder. Dependiendo de cada producción se escoge el o los métodos más óptimos para contar la historia. Pueden ser rigs basados en sistemas de:

- Varias geometrías donde están modeladas las deformaciones a las que puede llegar la geometría
- skinClusters pesados con diversas capas de deformación donde los joints tienen el control
- Atributos que controlen varias deformaciones, sean del tipo que sean
- Manivelas limitadas que controlen deformaciones
- Deformadores de wire donde viven controles dinámicos
- Desplazamiento de geometrías por el movimiento de otras más simples (ShrinkWrap, WrapReversed)

Aparte de los mencionados, hay otros que también generan deformaciones interesantes en los personajes. No existe una única manera genérica de construir un rig, sino que hay que adaptarlo a las necesidades de cada producción.

Rigs basados en joints

Este tipo de rigs dan una gran libertad de movimiento en los controles, ya que funcionan a partir de las transformaciones de uno o varios controles que apuntan a sistemas lógicos que acaban en los joints de pesado.

Los sistemas basados en huesos se pueden clasificar en dos grandes categorías:

- Siguen al rig
- Relativos al mundo (Locales)

Sistemas relativos al mundo

Este tipo de sistemas son los más habituales en los rigs faciales. El sistema lógico y de pesado se calcula en el (0,0,0) y va conectado como capa de blendshape al rig del cuerpo.

Sirven para poder dividir el cálculo de las deformaciones en varios bloques y poder tener una capa para cada sistema de cálculo.

Blendshapes

Los sistemas basados en shapes son un tipo de sistemas relativos al mundo, donde se modelan las formas y se conectan. Son muy útiles para los sistemas faciales.

Conceptos previos

Antes de iniciar cualquier rig hay que tener en consideración cuáles son los departamentos previos que intervienen en la creación del asset y cuáles son los posteriores. Es importante saber de dónde nos viene la data y hacia dónde irá la nuestra.

Actualmente nos podemos encontrar con los siguientes departamentos previos:

Modelado. Da el input de la geometría del personaje. Iterarán muchas veces sobre el personaje. En algunas producciones también proveen las shapes faciales de deformación.

Texturas. Se dedicará a hacer las texturas junto con el desplegado de las UV.

Groom. Proveerá las piezas del cuero cabelludo. En el caso de que el cabello vaya previamente animado, las curvas de simulación.

Y como departamentos posteriores:

Animación. Utiliza los controles del rig para dar expresividad al personaje.

Simulación. Utiliza las geometrías del cuero cabelludo animadas y las curvas que estaban extraídas del groom. También usa la animación de la ropa para tener ya una base.

Shot Sculpting. Con el archivo alembic (.abc) extraído de la animación, corrige penetraciones.

Hay que asegurar que toda la información que llega esté en buen estado y que el resultado que se entregue también lo esté.

Las consideraciones que hay que hacer son las siguientes:

- La geometría de modelado debe tener una topología fluida, donde los loops estén repartidos de forma homogénea por la superficie del modelo y sin que se desvíen.
- La topología debe estar recta: cuanto más recta y relajada esté la topología, mejores deformaciones tendrá.
- En los puntos de rotación (codos, rodillas, ingle y hombro) debe haber más densidad de topología para así acompañar la pérdida de la misma en las deformaciones.
- Si las UV no están en el primer cuadrante, habrá que hacer un set propio para rig, aunque sea en automático, para que los sistemas de seguimiento como los folículos puedan funcionar sin errores.
- Las geometrías del cuero cabelludo, preferiblemente, deben tener una topología parecida a la geometría a la que va pegada.
- Las curvas de groom deberán estar separadas por descripciones y ordenadas en el outliner por grupos.
- Tener en el rig un grupo donde solo esté la geometría limpia para cache.

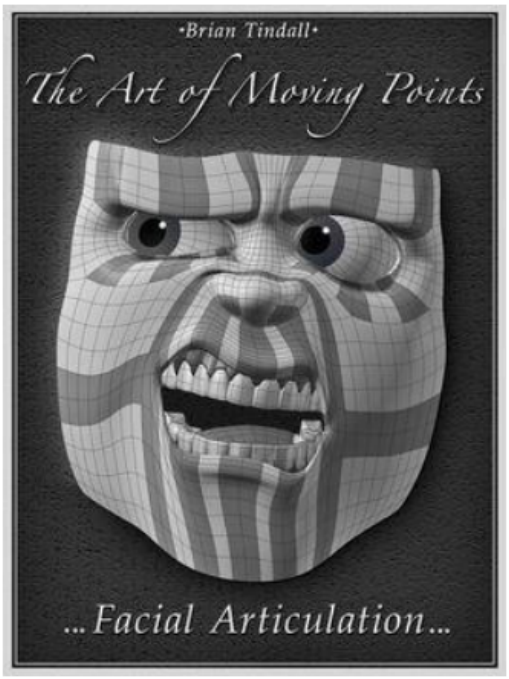

Ilustración 5: Ejemplo de topología facial, The Art of Moving Points

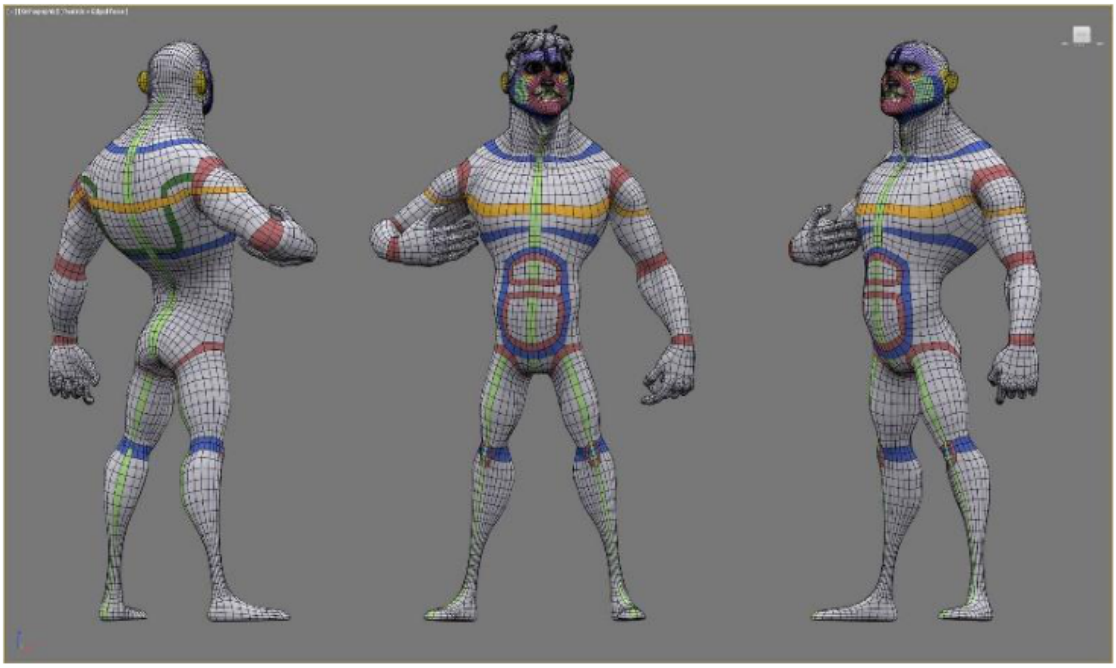

Ilustración 6: Ejemplo de flow de loops de topología de un hombre cartoon, autor: Héctor Sanz DOrs

Creación del rig (Casos)

Hay muchos factores que intervienen a la hora de construir un rig. Se pueden diferenciar en 4 ámbitos:

- Tipo de programa a utilizar
- Tipo de animación
- Perfiles de animadores en el departamento de animación

- Tecnología de la que se dispone

El tipo de programa a utilizar define qué herramientas se pueden usar. No todos los programas tienen las mismas herramientas; cada uno tiene sus puntos positivos y las cosas que no acaban de funcionar. Saber el programa a utilizar también da una gran posibilidad de plug-ins que podemos incorporar al pipeline.

Es importante saber si la producción aguanta el uso de plug-ins, ya que muchas veces el rendimiento de los plug-ins mejora los sistemas que proporciona el propio programa y, además, simplifica notablemente el grueso de trabajo. Los plug-ins se tienen que compilar para cada versión del programa y del sistema operativo, por lo que dificulta su implementación en producciones que tienen máquinas diferentes.

La manera de animar juega un papel muy grande en el momento de preparar el personaje para animación. No es lo mismo un tipo de animación realista, cartoon, semirrealista, con aristas marcadas o stop-motion 3D. Hay infinidad de maneras de animar y hay que tener claro cuál se usa en cada producción.

El nivel de experiencia de los animadores es otro de los puntos que hay que considerar. No es lo mismo tener un equipo donde la mayoría de los animadores tienen perfil Junior o Mid, que tener un equipo en el que el grueso de la animación recae en personas con perfiles senior o superiores.

En un equipo donde la mayoría sea de perfil senior o superior se pueden usar sistemas lógicos de joints, wires, etc., más avanzados y que lleven las deformaciones a un extremo donde los personajes estén más vivos y no tengan tantas limitaciones. Esto se puede hacer porque una persona con más experiencia sabrá adaptar su estilo de animación con el de la producción y se hará más difícil perder la apariencia del personaje.

Por otro lado, con un equipo de perfiles más junior, el rig pide ser limitado y no dejar tanta libertad de movimiento en las deformaciones, ya que la posibilidad de perder la actitud del personaje es más alta. Por tanto, las deformaciones máximas están preestablecidas por el supervisor de animación del proyecto. Pueden estar guardadas en un catálogo de poses o ya puestas en las deformaciones de los controles.

La tecnología va avanzando y, año tras año, se van implementando nuevos paradigmas que revolucionan la manera en la que se producen los assets. Actualmente se está dirigiendo hacia la evaluación paralela con threads y el cálculo de las deformaciones por GPU. Hay diversos estudios de animación y efectos que desde hace años apuestan por esta vía y han desarrollado softwares y plug-ins propietarios. En el caso de “Pixar” tienen “Presto”, un sistema de evaluación de las escenas y los personajes mediante el cache de las geometrías y la implementación de sistemas “USD”.

Análisis de los personajes

Previamente a empezar un rig hay que observar cómo está construido el modelo, qué topología tiene, cuáles son sus limitaciones y qué es lo que se quiere en el asset.

También hay que tener en cuenta cuáles son todos los extras que tendrá y los que no tendrá. Hacer de más e invertir tiempo en sistemas que no se usarán o innecesarios quita tiempo de dedicarlo a los sistemas necesarios.

Es muy importante hacer bien el análisis, ya que a partir de él se estructurarán las siguientes fases.

En el caso de los personajes de proyectos podemos analizar las siguientes cosas de cada uno de ellos:

Ledyan (Bípedo/LIB): El cuerpo es principalmente atlético y necesita un rig para que el body mechanics funcione correctamente. A nivel de expresividad facial tiene poca, así que el facial puede ser más limitado. Como extra tiene: el carcaj de las flechas, las flechas, las protecciones, la hombrera y los brazaletes. La ropa irá pegada encima del cuerpo con skinning.

Yura (Bípedo/LIB): De base es igual que Ledyan, y de expresividad facial tendrá menos, debido al pañuelo que lleva en la cara. Como extras hay que considerar los zapatos, las protecciones de la cintura, los pechos, las hombreras y los brazaletes. La ropa también irá pegada encima del personaje.

Ilustración 7: El personaje de Yura a la izquierda y a la derecha el personaje de Ledyan. Los dos de "Last in Battle"

Ekko (Bípedo/Path of Sand): Igual que los otros dos bípedos como base. Este personaje tiene más expresividad facial, por tanto necesitará más cuidado en el facial. Como extras tiene: el cinturón, la daga, el colgante del cinturón, la tela que le cuelga del cinturón, la capa, el buff, un cambio de ropa en shot y capucha.

Monstruo (Cuadrúpedo/Path of Sand): En el shot sale caminando, por tanto necesitará un sistema de flexión en las patas traseras y en las delanteras, un omóplato en la pata delantera y una clavícula en la pata posterior, sistema para la respiración, sistema para poder dibujar la cola en una forma adecuada y sistemas de control dirigido en los tentáculos y sistemas automáticos en los tentáculos.

Leo (Bípedo/WOLT): Rig corporal y facial básico. La ropa irá simulada por encima de la animación.

Druidac (Bípedo/WOLT): Rig corporal y facial básico. La ropa irá simulada por encima de la animación.

Nut (Robot/WOLT): Es un personaje que tiene cierta relevancia en el proyecto. Al no ser un bípedo, requerirá un rig personalizado, donde se pueda mover y que los brazos articulen de manera elástica.

Mucha expresividad en las cejas y que se puedan hacer muchas formas diferentes. Pupila retráctil.

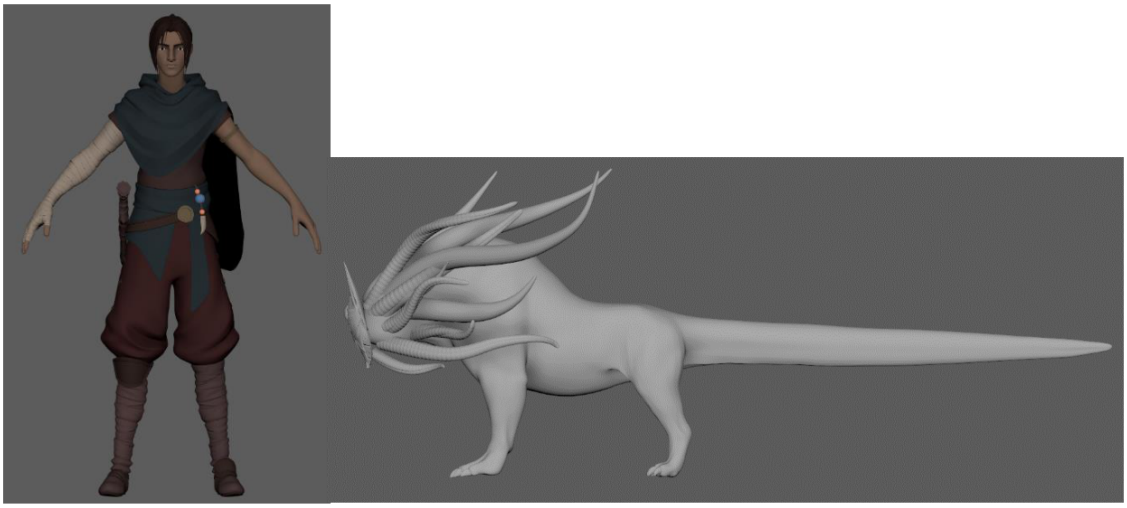

Ilustración 8: Ekko a la izquierda y el monstruo a la derecha. La escala no es representativa entre los dos personajes. Personajes de "Path Of Sand"

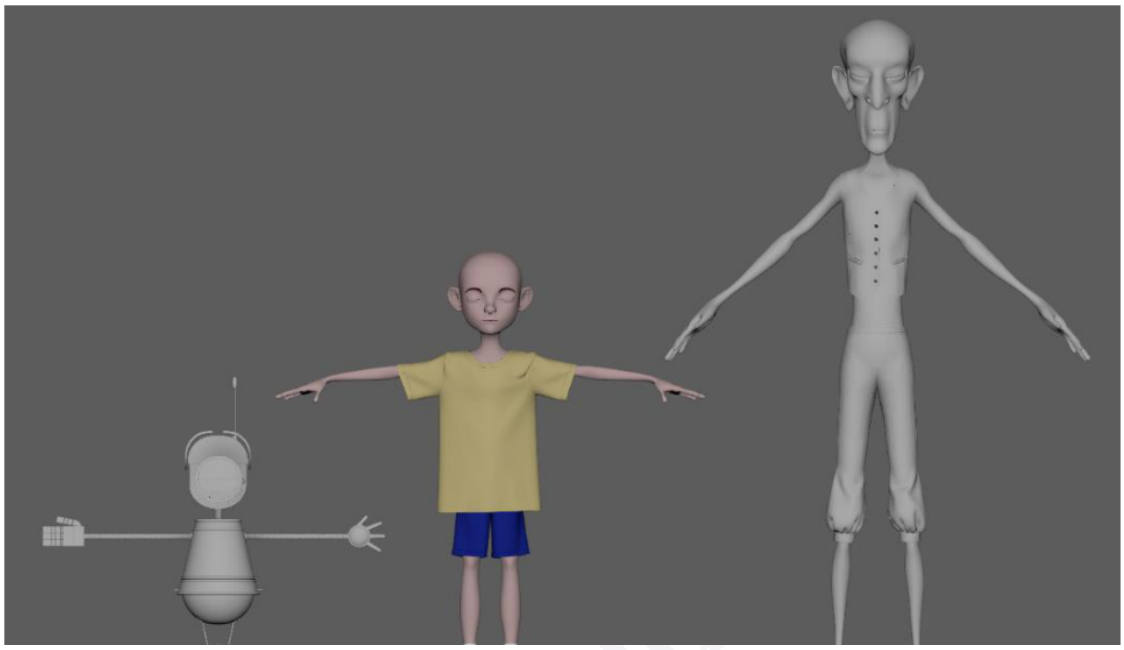

Ilustración 9: En orden de izquierda a derecha, Nut, Leo y Druidac. Personajes de World Of Lost Things

Planteamiento

Para poder controlar todos los cambios de versiones de modelo, UV y groom, y para poder empezar a hacer los rigs con antelación, antes de tener los modelos finales, el rig se plantea para hacerse por programación (Python). Y como sistema de control de versiones: Git, especialmente un repositorio en GitHub.

Así podemos separar la creación del rig en las siguientes partes:

Model. Carpeta con una serie de archivos .ma que contienen las diversas versiones del modelo
Shapes. Carpeta con una serie de archivos .ma que contienen las diversas versiones del modelo
Groom. Carpeta con una serie de archivos .ma que contienen los scalps y las curvas

Guías corporales. Carpeta con una serie de archivos .ma donde hay joints que definen la posición de entrada para que los módulos sepan dónde crearse

Guías faciales. Carpeta con una serie de archivos .ma donde hay locators y NURBS que definen la posición de los sistemas lógicos del facial

Pesos de los mapas. Series de carpetas con los diferentes mapas de pesos guardados en binario

Settings. Carpeta con una serie de archivos .json con propiedades definidas para cada personaje

Control Shapes. Carpeta con la información de construcción de las control shapes

Todas estas partes están recogidas dentro de un archivo Python (.py) que será el que construya cada vez que se ejecute un nuevo rig con los diferentes inputs que haya. Para acceder a las acciones de Maya se usan las librerías “maya.cmds”, “OpenMaya” y “OpenMayaAnim”, y para la gestión de archivos y otras utilidades se utilizan las librerías os, “shutil”, “json”, “cPickle”, “StringIO”, “time” y “logging”.

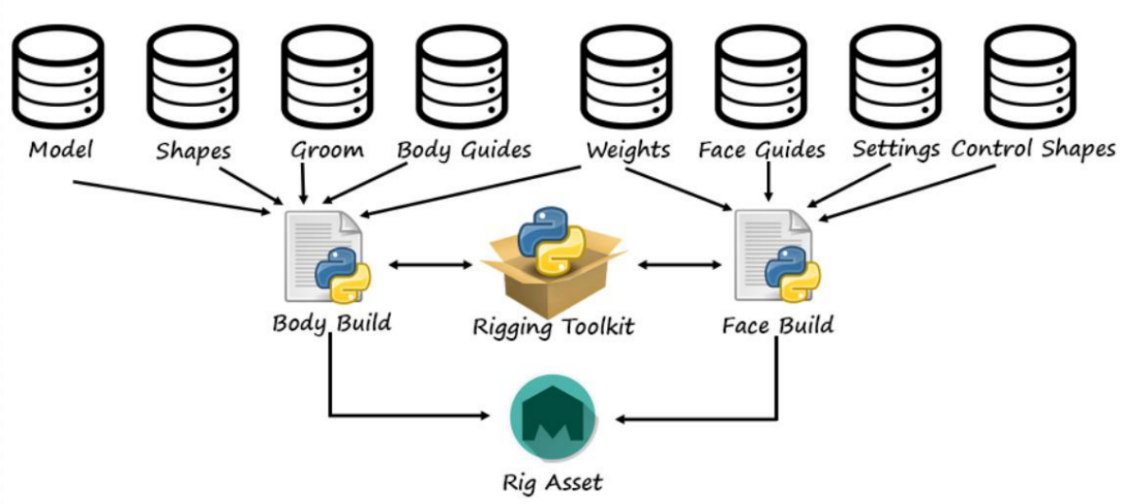

Ilustración 10: Representación de la estructura para generar un rig según el planteamiento establecido

Modularidad y proceduralidad

El rig procedural se da cuando se hace, no solo mediante código, sino también cuando se aplican las reglas de la programación orientada a objetos a su estructura.

Debe satisfacer las premisas de:

Abstracción. Extraer las necesidades comunes de los objetos. Separar el comportamiento de su implementación.

Encapsulación. Ocultación de información. El animador desconoce la implementación interna.

Herencia. Relación entre los diferentes objetos de rig. Módulos que van desde la generalización hasta la especialización.

Polimorfismo. Un mismo objeto de rig se comporta de manera diferente en función de la operación que se le aplica.

Se puede establecer que un sistema de extremidad es igual ya sea una pierna o un brazo. Así pues, con un objeto de rig de extremidad se pueden satisfacer las necesidades de brazos y de piernas. También ocurre con las otras partes del cuerpo. Solo es necesario implementar un sistema de dedo del cual dependan los otros.

Esto no solo ocurre con los humanos. El concepto de la anatomía comparada describe que, entre todos los seres vivos, hay un patrón similar en cuanto a estructuras. Por tanto, podemos reaprovechar los módulos creados entre diversos personajes del mismo tipo.

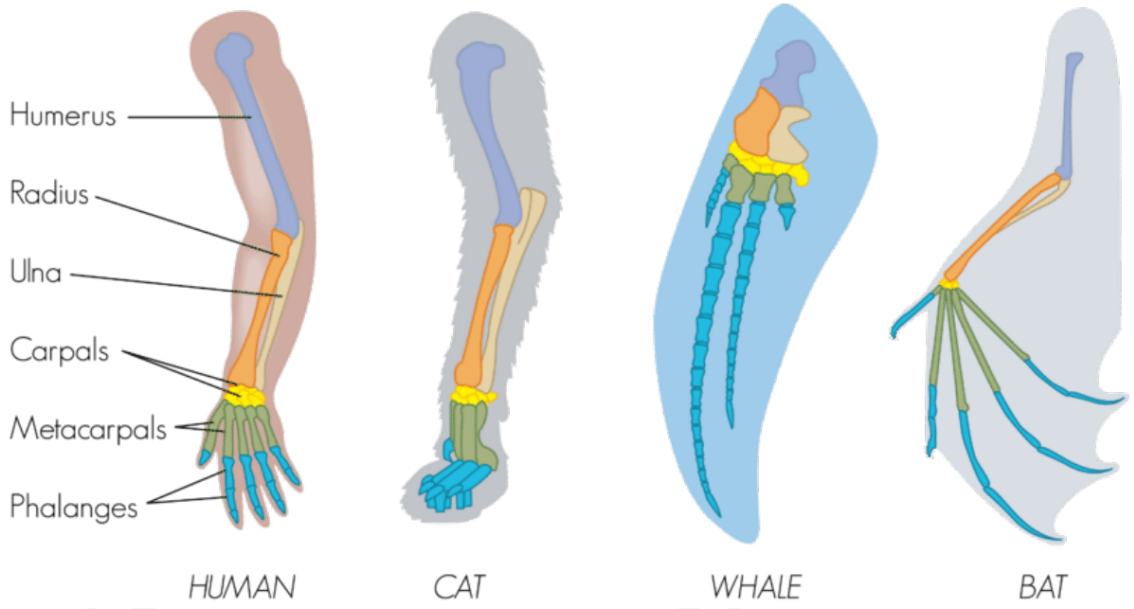

Ilustración 11: Comparación de las extremidades de diversos animales según la teoría de la anatomía comparada.

Sistemas corporales básicos

Los sistemas de los bípedos tendrán las siguientes características:

El setup de un bípedo será similar al de la Mary Project, un personaje modelado y texturado por José Manuel García Alvarez y riggeado por Antonio Méndez Lora. Es un personaje que todos los compañeros y compañeras tienen por la mano y no les costará usar.

Para el rig corporal general hay sistemas de IK y FK para las extremidades. El sistema de IK tiene también un sistema de stretch, para llevar las deformaciones más lejos.

En el sistema de las extremidades tienen un control extra que permite hacer que sea más larga o más corta la parte superior o inferior de la extremidad, manteniendo o no el volumen. También permite arquearla.

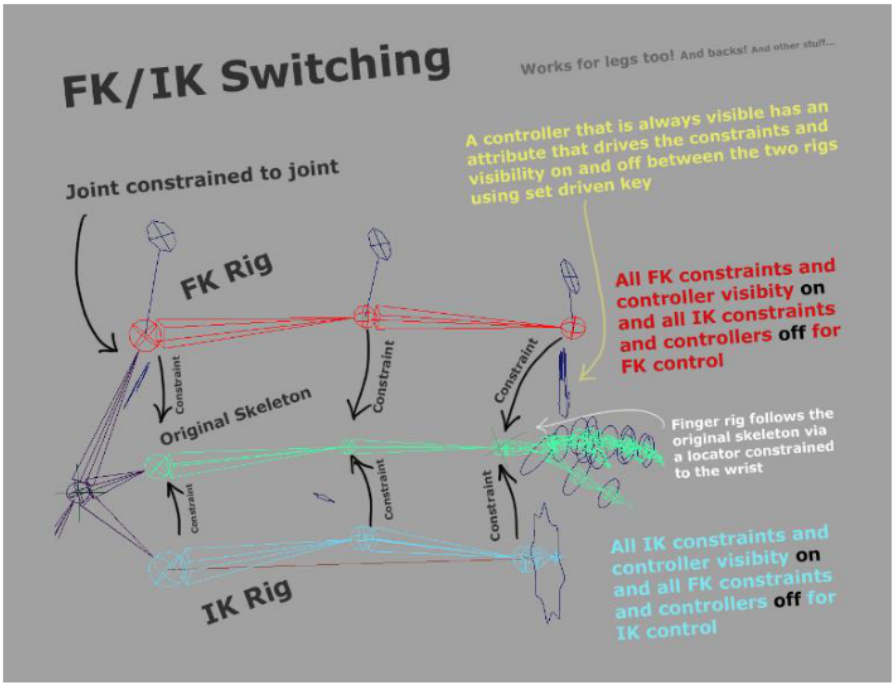

Ilustración 12: Esquema de cambio de IK a FK establecido por Intro Rigging [1]

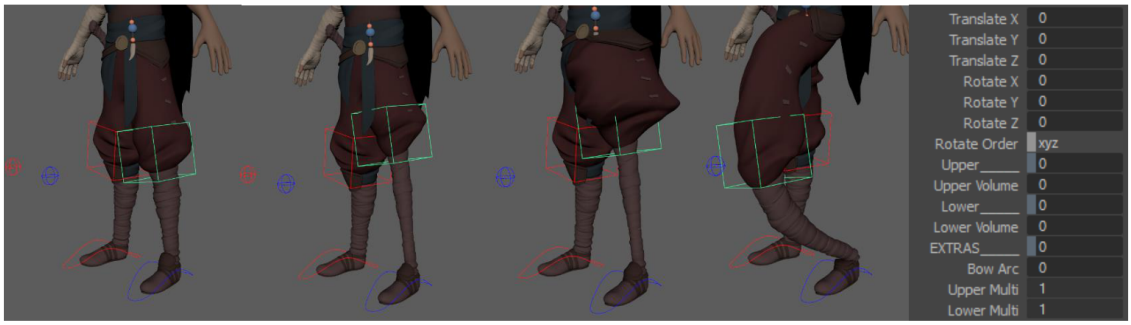

Ilustración 13: Ejemplo de extremidad en IK, donde hay un control que mueve la pierna con el IK, un pole vector que define la dirección que mira la pierna y un control que permite darle un offset a la rodilla.

Para la espalda hay un sistema de IK y FK, pero este es diferente ya que los controladores del FK están posicionados después del IK, como se puede ver en la siguiente imagen. También tiene unos parámetros de squash y stretch con valores mínimos y máximos, para que le dé más dinamismo a la animación.

El sistema de las clavículas se lleva los brazos. Para los brazos en FK tiene una opción para que los brazos sigan las rotaciones de la clavícula o no, así se le da otro efecto a este sistema.

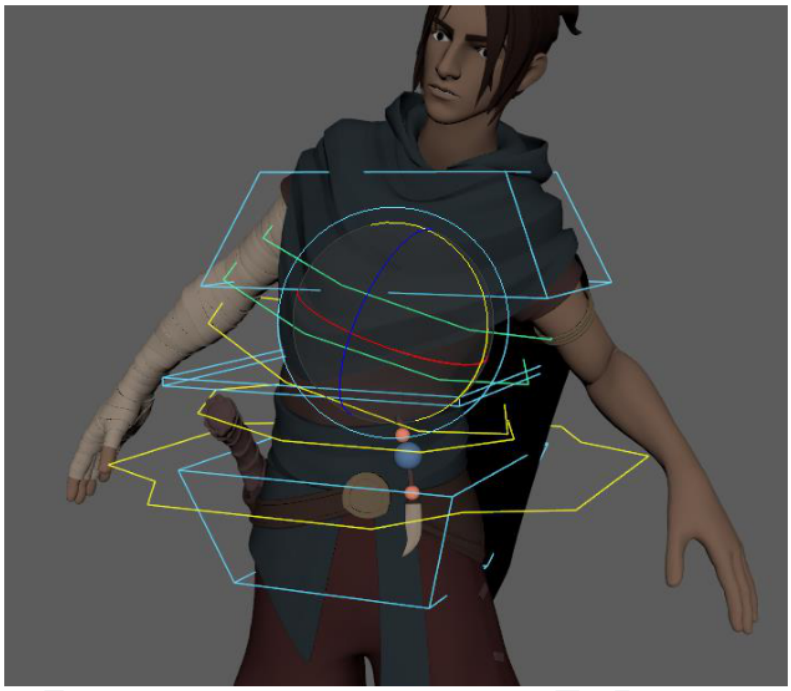

Ilustración 14: Ejemplo de espalda en el sistema IK/FK vivo

Ilustración 15: Ejemplo de funcionamiento de la clavícula con el sistema de seguimiento del brazo en FK

Los dedos de las manos tienen sistemas de control, tanto en IK como en FK. Cuando está en FK tiene atributos de spread, fist y twist.

Para el pie en IK hay un sistema de IK inverso (reverse foot) que da las siguientes opciones 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. Todos estos sistemas están formados por conexiones directas a grupos de offset sobre el IK de la pierna.

El sistema de la cabeza tiene un control FK desde la base del cuello y un control IK desde la punta. También tiene una opción para que la cabeza no siga las rotaciones del pecho.

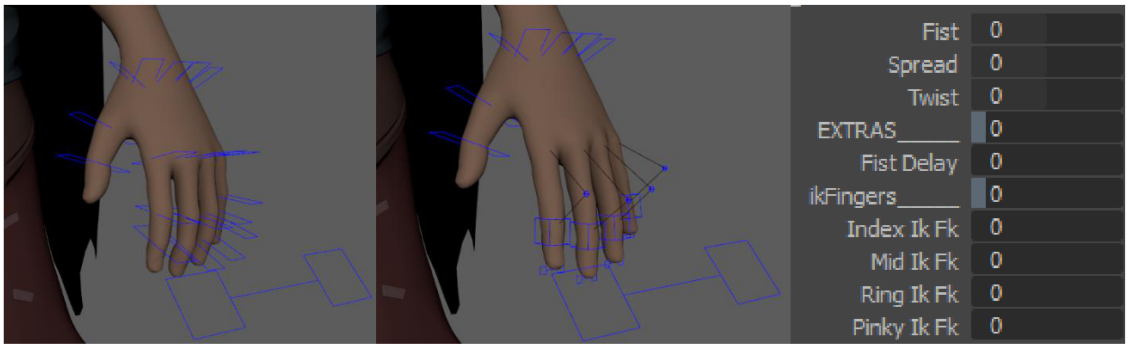

Ilustración 16: Sistema de control de los dedos de las manos, con los atributos presentes en el control genérico. En el centro se puede observar el sistema de IK en los dedos.

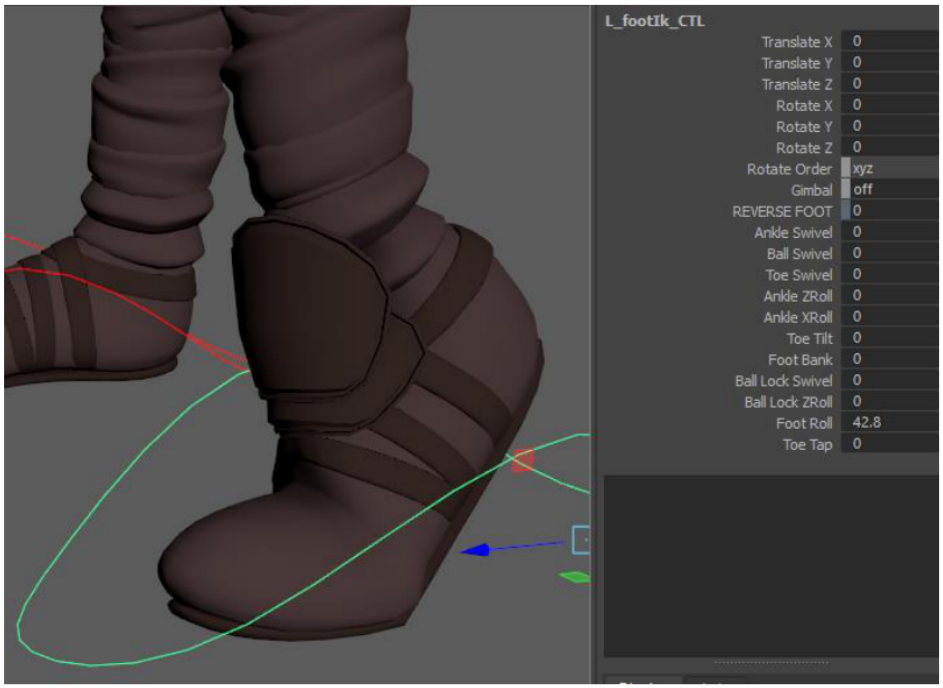

Ilustración 17: Sistema de pie inverso; a la izquierda está colocado el roll del pie, a la derecha están presentes los atributos específicos del control.

En los ojos hay un sistema para controlar el aim de los mismos, separado por derecho e izquierdo y uno global que se lleva los dos. También tienen control de las dimensiones de las pupilas y el iris.

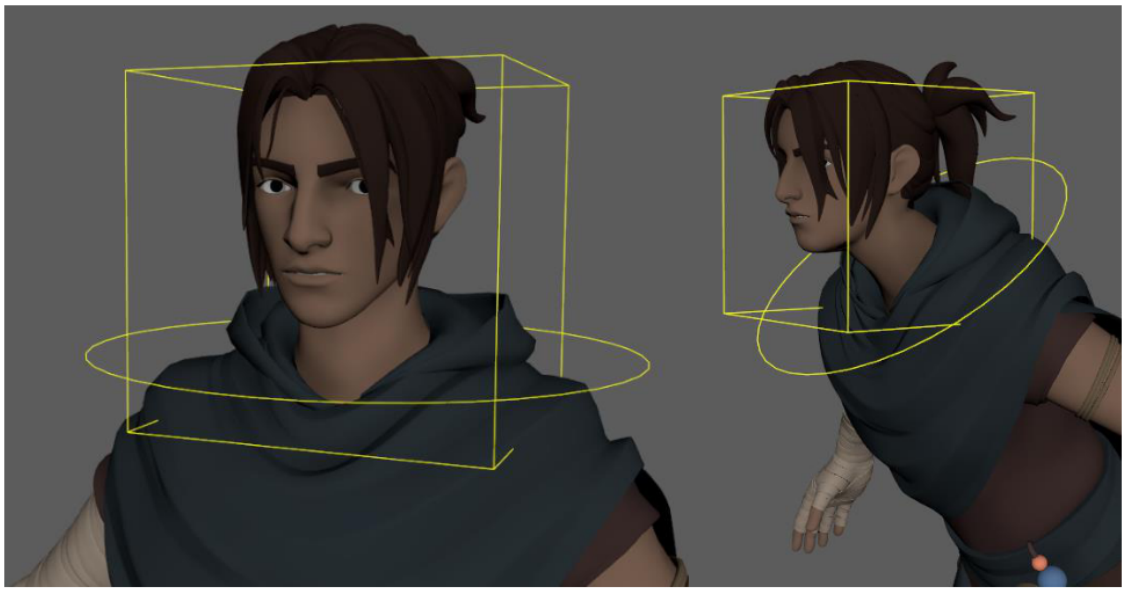

Ilustración 18: Sistema de control de la cabeza y del cuello. A la derecha se ve cómo está actuando el sistema de no seguimiento de la cabeza a las rotaciones del pecho.

Ilustración 19: Sistema de control de los ojos. Se puede ver qué control se tiene con los sistemas del iris y de la pupila.

Sistemas faciales básicos

El rig facial es relativo al mundo, por tanto va conectado a las deformaciones del cuerpo mediante una blendshape. Todos los controladores faciales están en espejo.

El orden en el que se exponen las capas no es el orden en el que están implementadas en el rig; solo son agrupaciones de ideas lógicas.

Capa 1

Sistema de huesos para la cara básica: parte inferior de la mandíbula, parte superior de la mandíbula y nariz. La mandíbula superior nunca va por debajo de la inferior, lo que hace que la boca se cierre y

se empuje con la colisión.

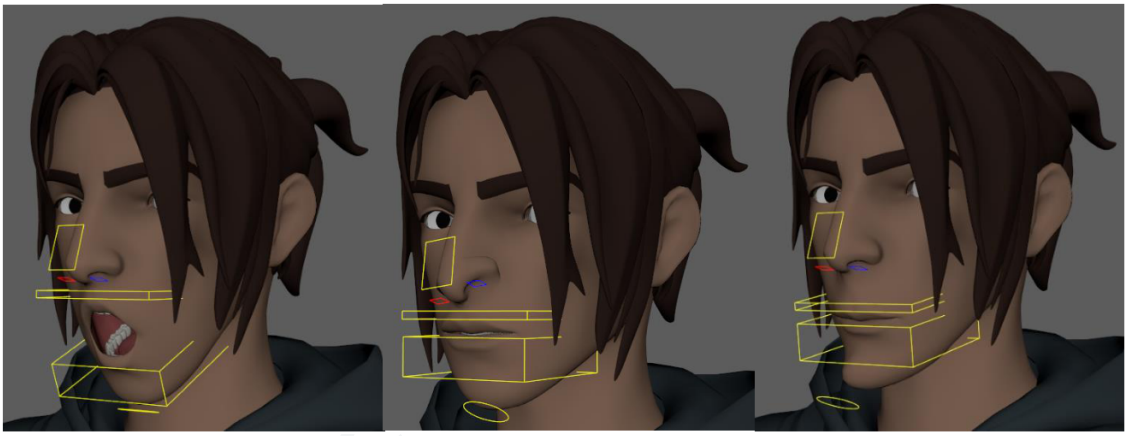

Ilustración 20: Sistema de control de la mandíbula y la nariz

Capa 2

Sistema de huesos para la parte superior de la mandíbula, los huecos de los ojos y las orejas. Para poder moverlos enteros y posicionarlos.

Ilustración 21: Sistema de control para ojos y orejas.

Capa 3

Shape correctiva de la apertura de la mandíbula para ayudar a conseguir la compresión de las mejillas al abrir la boca y tener la forma de O. Se hará con un joint y será escalado.

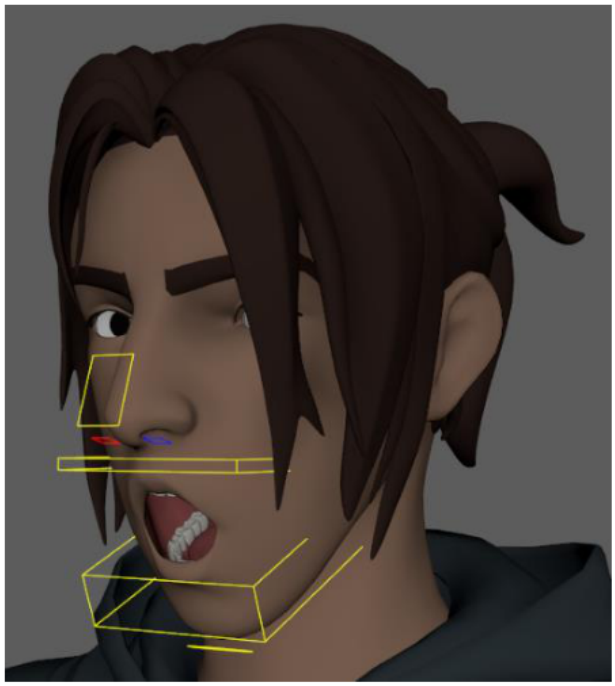

Ilustración 22: Sistema de compresión de los labios al abrir la boca

Capa 4

Sistema de joints de los párpados, rivet y blendshape entre los edges superiores e inferiores del párpado. Se puede definir la altura del parpadeo, o bien si se quiere hacer por separado.

Ilustración 23: Sistema de control de los párpados. Se pueden ver los atributos.

Capa 5

Sistema de retoque fino de las shapes. Controles de retoque para la zona de los labios, surco nasolabial y cejas.

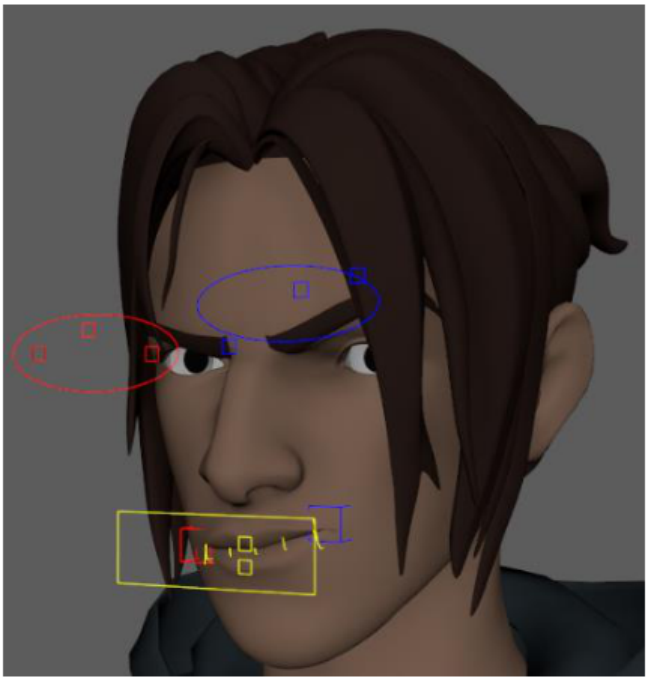

Ilustración 24: Sistema de retoque de las cejas y los labios

Capa 6

Sistema de blendshapes (los modelos básicos y simétricos de las shapes son de Enrique Alberola; mi papel fue hacer el diseño de las shapes, cómo conectarlas y pequeños retoques y repasos de las shapes):

Boca

Tendrá un sistema de 8 blendshapes; se modelan las formas de comisura estirada, comisura aplastada, comisura subida, comisura bajada y sus combinaciones. A la hora de modelar las shapes de la boca, hay que tener en cuenta que los labios deben deslizar por encima de la mandíbula que forma la zona de los dientes. Es importante que el labio superior y el inferior no se separen.

Comisura estirada. Es importante que la deformación sea paralela a los labios base, ya que con la combinación de las otras generarán otras formas. Principalmente se mueve la comisura de los labios y se lleva un poco del medio. Se hunde la zona nasolabial de manera horizontal y estira muy suavemente la zona de la fosa nasal.

Comisura aplastada. Es la inversa de la comisura estirada; se trata de hacer que las comisuras de los labios estén a punto de tocarse. La parte central de los labios se comprime (debe deslizar por la superficie de la mandíbula).

Comisura subida. Se levanta la comisura de los labios, acompañando levemente la nariz y la zona nasolabial. La mejilla acompaña levemente a la deformación; debe conseguir llegar a la forma de U.

Comisura bajada. Es la contraria a la anterior; debe bajar la comisura de los labios y estirar los loops de la zona nasolabial. La mejilla se estira un poco.

Ilustración 25: Shapes en orden de izquierda a derecha: comisura estirada, comisura aplastada, comisura subida y comisura bajada

La unión de las 4 deformaciones anteriores se interpola mediante un controlador que se mueve en traslación y, x. Aunque se interpolen las deformaciones, las shapes que quedan resultantes en los extremos [(10,10), (-10,10), (-10,-10), (10,-10)] no acaban de tener una deformación adecuada en cuanto a las mejillas y la armonía entre ellas. Para arreglar las deformaciones resultantes, hay que generar shapes de corrección que forzarán la forma que determinemos. Y sus combinaciones: para generar las shapes, hay que partir de la unión de:

- Estirada + Arriba
- Estirada + Abajo
- Comprimida + Arriba
- Comprimida + Abajo

El objetivo de las correctivas es marcar la zona de las mejillas. Son shapes clave para marcar la actitud del personaje.

Ilustración 26: Shapes de corrección, en orden de izquierda a derecha: estirada arriba, estirada abajo, comprimida abajo y comprimida arriba

Mejilla

Tiene 4 shapes (mejilla inflada, mejilla aspirada, mejilla subida y mejilla deslizada)

Mejilla inflada. Las formas esféricas funcionan bien. Para hacerla, va bien poner una esfera como guía de ayuda. También suele funcionar sacar la forma con un deformador de “softMod”.

Mejilla aspirada. Hay que tener en cuenta que bajo los ojos está el hueso cigomático, por tanto la mejilla hay que colocarla un poco por debajo.

Mejilla subida. Tiene que hacer la compresión de la mejilla con el ojo.

Mejilla deslizada (4). Se trata del deslizamiento, en traslación y, x, de la mejilla por encima de la cara.

Ilustración 27: Shapes de las mejillas, en orden de izquierda a derecha y de arriba a abajo: inflada, aspirada, deslizada subida, deslizada bajada, deslizada adelante y deslizada atrás.

Shapes de los labios

(Roll positivo, roll negativo, beso) Sirven para poder hacer las deformaciones fonéticas y para dar intención a los movimientos de los labios. Estarán controladas por atributos limitados de 0 a 10.

Roll positivo. Se trata de la rotación hacia fuera de los labios, tanto inferior como superior; deben verse parte de los dientes.

Roll negativo. Igual que el anterior, pero en este caso hacia dentro. Sirve para estrechar los labios y hacer fonemas bilabiales (p, b).

Beso. Similar a la de boca estrecha, esta hace de beso, es decir: aparte de estrechar las comisuras y los labios, saca más volumen en los labios y comprime las mejillas.

Ilustración 28: Shapes específicas de los labios, en orden de izquierda a derecha: roll positivo, roll negativo, beso

Cejas

Hay 8 shapes (subidas, bajadas, juntas, separadas, giradas en positivo, giradas en negativo, adelante y atrás). Es importante que deslicen por la superficie ficticia del cráneo del personaje.

Subidas. Las cejas deben subir vertical y simétricamente hasta conseguir igualar la topología. Las cejas deben quedar un poco curvadas.

Bajadas. Las cejas bajan hasta la altura de la mitad de los ojos; hay que vigilar que no colapsen con las pestañas, en el caso de que las tenga largas. No se debe marcar el entrecejo.

Marcaje del entrecejo. Se debe marcar el entrecejo, hacer que colapse la ceja.

Separadas. Las cejas deben deslizar por encima del cráneo hacia fuera.

Juntas. Las cejas deben deslizar por encima del cráneo hacia dentro.

Giradas en positivo. Deben hacer un seno en positivo.

Giradas en negativo. Deben hacer un seno en negativo.

Adelante. Las cejas deben salir hacia fuera; sirve para corregir otras deformaciones.

Atrás. Las cejas deben ir hacia dentro; sirve para corregir otras deformaciones.

Ilustración 29: Shapes de las cejas, en orden de izquierda a derecha y de arriba a abajo: subidas, bajadas, adelante, atrás, separadas, juntas, marcaje del entrecejo, giradas en positivo, giradas en negativo.

Nariz

Tiene 3 shapes (sneer, flare, sniff)

Sneer. Subida de las fosas nasales

Flare. Inflado de las fosas nasales

Sniff. Aspiración de las fosas nasales

Ilustración 30: Shapes de la nariz, en orden de izquierda a derecha: sneer, flare, sniff

Ejecución

Corporal

El primer paso es establecer las guías corporales. Son una serie de joints, locators y NURBS que establecen la información de posición y orientaciones para que los diversos módulos (clases) tengan los datos para operar.

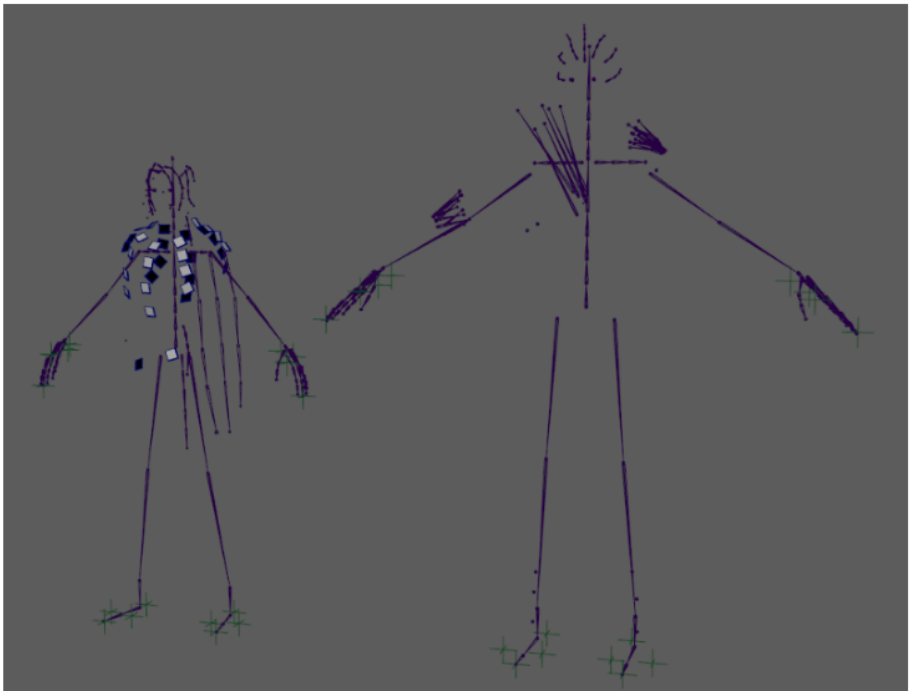

Ilustración 31: Ejemplo de guías corporales. Las de Ekko a la izquierda y las de Ledyan a la derecha

Estos datos son las bases para la creación de todos los sistemas corporales, tanto genéricos como específicos.

El segundo paso es diseñar la estructura del archivo de Python que será el responsable de crear todos los sistemas usando el “riggingToolkit” diseñado. Este es un ejemplo del inicio de la construcción, donde se carga el modelo y las guías, se crea la estructura básica del outliner y se crea

el módulo de la espalda.

El tercer paso es definir los espacios que tendrá cada control, o serie de controles. Existen los espacios dinámicos, como el del brazo o la cabeza, y los de seguimiento como la mano o el pie.

```
def build(character_name, body=False, falece=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()
```

Ilustración 32: Ejemplo de inicio de archivo de construcción del asset

```
    ##################################
    ##################################
    ##################################
    # 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"
```

Ilustración 33: Ejemplo de código de la definición de los espacios de seguimiento de los controladores

El tercer paso es ejecutar el archivo de construcción y adaptar las dimensiones de los controladores que salen por defecto. Pasar de los controladores de la izquierda a los de la derecha.

El cuarto paso pasa a ser el skinning de las geometrías, que es cuando se define qué zona de influencia tiene cada hueso. Para hacerlo, se usan meshes proxies y herramientas como el “ngSkinTools” y el “brSmoothWeights”. Para agilizar la lectura y el guardado de los mapas, se guardan en archivos binarios creados con “cPickle”.

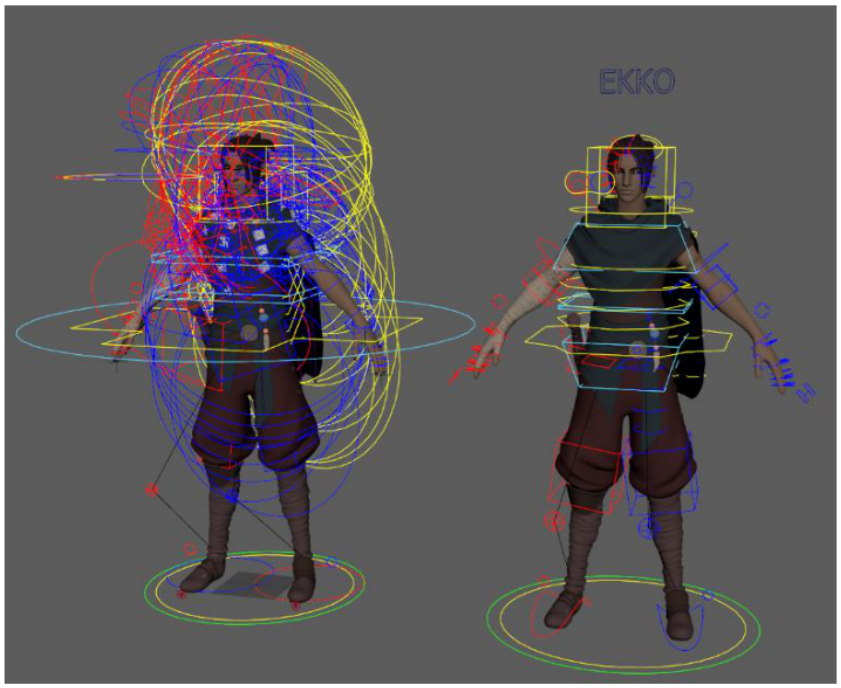

Ilustración 34: Ejemplo de guías del personaje; a la izquierda las guías por defecto, a la derecha las guías colocadas

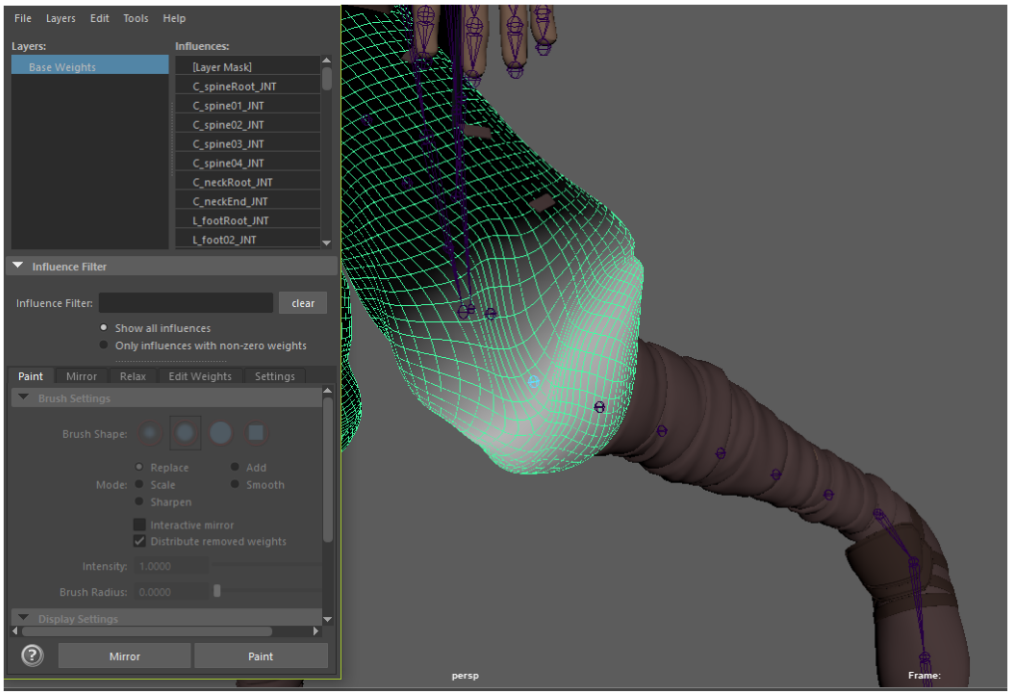

Ilustración 35: Ejemplo de pintar pesos con ngSkinTools, donde hay una capa de pesos.

El quinto paso es cargar los settings y los pesos que se han establecido. Por ejemplo, los de fist, spread y twist de los dedos de la mano.

El sexto paso, para acabar la construcción del cuerpo, es bloquear y colocar ciertos atributos a medida del supervisor de animación. Así, se hace más difícil que el animador rompa el rig accediendo a opciones que no están visibles de primeras.

```
# 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)
```

Ilustración 36: Ejemplo de carga de formas de los controles y de los mapas de pesos

```
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)
```

Ilustración 37: Ejemplo de preparación del rig para animación

Facial

La construcción del rig facial es similar a la construcción del corporal.

El primer paso también es definir las guías de dónde estarán posicionados los sistemas. Los locators marcan posiciones y orientaciones, y las superficies marcan espacios de proyección de los sistemas

faciales. La superior marca cuál es el rango de movimiento de las cejas y la inferior hace lo mismo con los labios. De esta manera se mantienen más las formas del personaje.

El segundo paso es generar el archivo de construcción del facial. La diferencia con el corporal es que se carga el rig corporal y se crean capas de deformación.

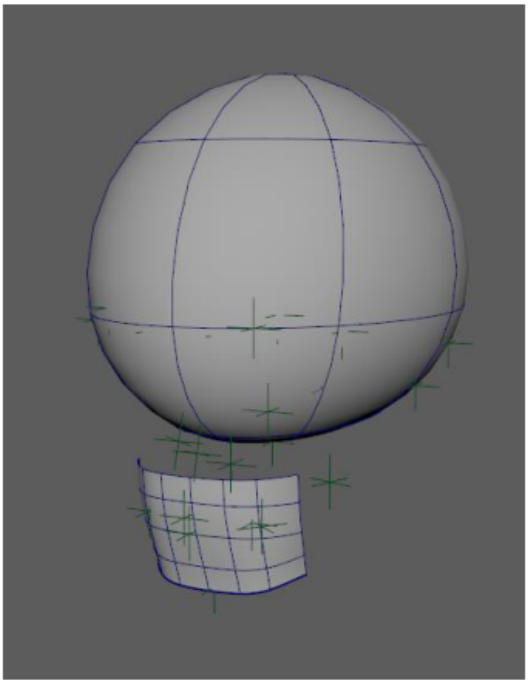

Ilustración 38: Ejemplo de guías faciales, Ekko

```
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)
```

Ilustración 39: Ejemplo de la creación de las capas de deformación para el facial

Como tercer paso, se cargan las shapes, los pesos y los settings de deformación igual que en el sistema corporal.

Comprobaciones

Aunque el rig se haga de manera procedural, hay que comprobar cada vez si las deformaciones de la nueva construcción funcionan con las antiguas animaciones. Por tanto, es importante tener un sistema como el “Studio Library” o exportaciones e importaciones de “Atom”, con animaciones guardadas, para corroborar que la nueva versión es compatible con las anteriores.

Extras

Aparte de los sistemas genéricos para los personajes, se han creado sistemas específicos para cada uno de ellos.

Ledyan

Se le han creado sistemas de control FK para la hombrera, para las hojas de la hombrera, para el carcaj, para sus flechas y para el cabello.

Las espinilleras y las cintas tienen un sistema de un joint que descansa sobre una NURBS surface que tiene los mismos pesos que la geometría de debajo.

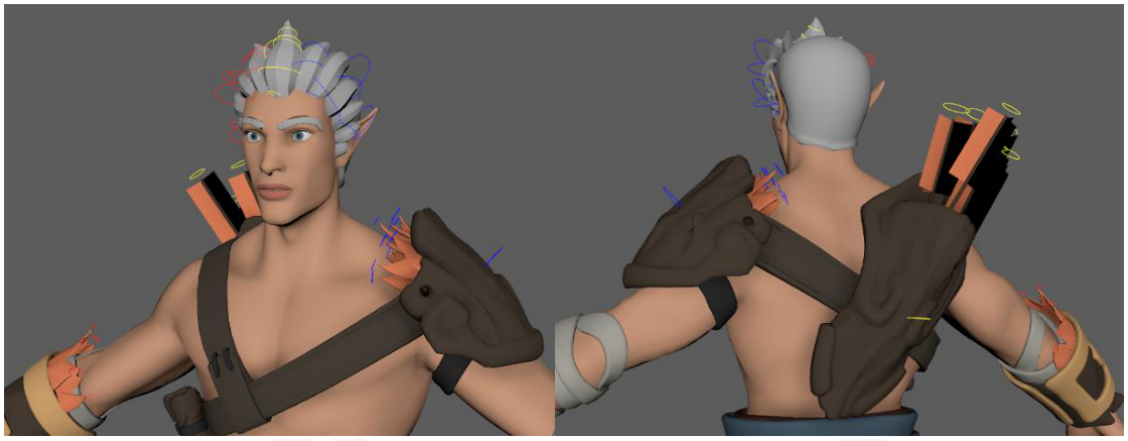

Ilustración 40: Ejemplo de los sistemas FK en Ledyan

Ilustración 41: Ejemplo de sistemas de joint sobre una NURBS.

Yura

Tiene sistemas de FK para las hombreras, los pechos y los mechones del cabello. Para el moño tiene un sistema similar al de la espalda: un IK con squash y stretch para darle movimientos secundarios.

Ilustración 42: Sistemas de FK y de spline para Yura

Ekko

Todas las cosas que le cuelgan a este personaje — amuleto del cinturón, tela del cinturón, espada, cabellos — son cadenas FK pegadas a diversas partes del cuerpo.

La capucha está hecha con el mismo sistema de IK con tangencias y squash que el moño de Yura.

Para el buff, la hebilla y el punto donde se engancha la daga, hay un sistema de una NURBS pesada con las mismas deformaciones que la geometría de debajo, que mueve, en una segunda capa, la geometría que controla.

Ekko tiene la posibilidad de poner y quitar la capucha. La capucha puesta tiene una serie de sistemas FK para que se pueda animar el comportamiento dinámico sin tener que simularla. Algunos controles siguen diversos huesos del cuerpo (cuello (4), cabeza (2)).

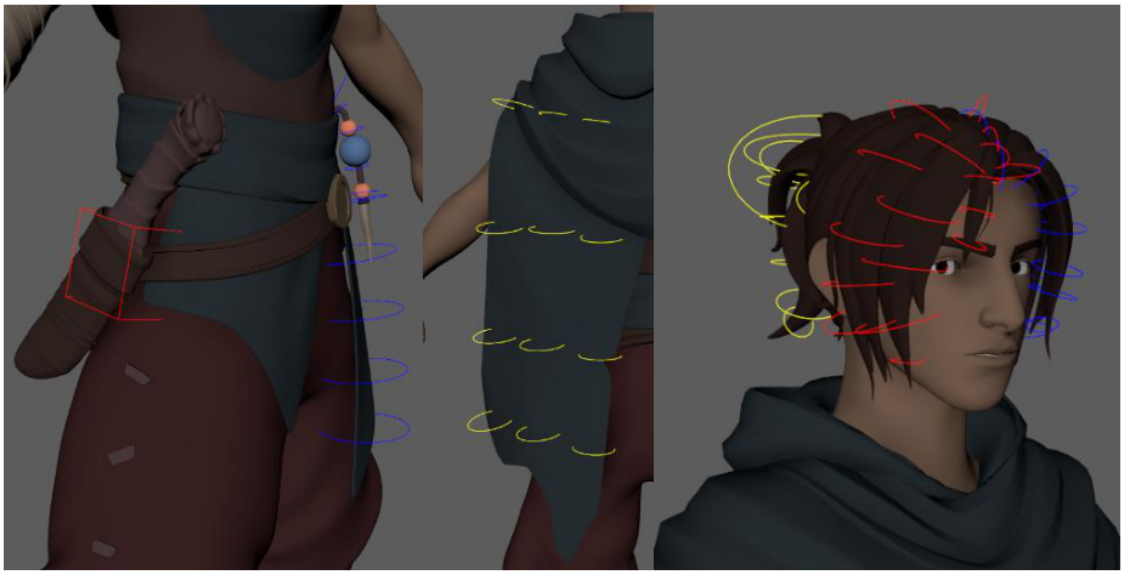

Ilustración 43: Sistemas de FK en Ekko

Ilustración 44: Sistema de IK de la capucha de Ekko

Ilustración 45: Sistemas de joint sobre una NURBS en Ekko

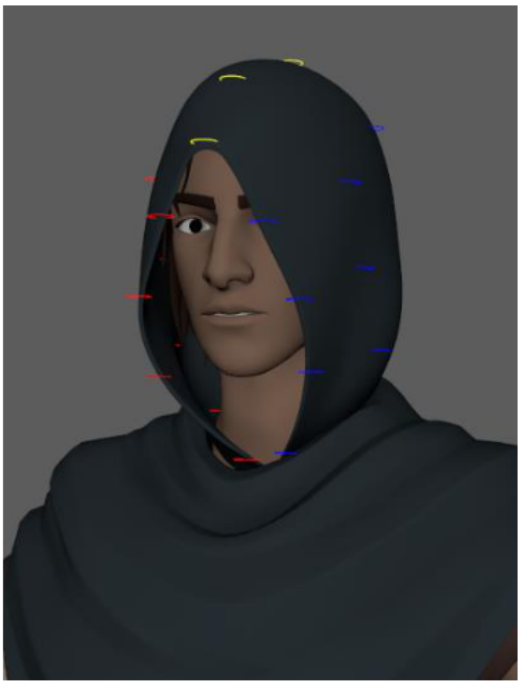

Ilustración 46: Sistemas de FK para la capucha puesta de Ekko

Monstruo

El monstruo de “Path of Sand” no es un bípedo sino un cuadrúpedo y necesita un sistema de deformación diferente.

Para las patas tiene un sistema IK que permite articular la pata y otro sistema IK que permite rotar el talón hacia adelante y hacia atrás. El sistema de la clavícula trasera es un sistema de clavícula normal de bípedo. Pero para la clavícula delantera tiene un sistema reverso de la clavícula. Esto permite sacar el omóplato del cuadrúpedo.

Para la espalda tiene un sistema de IK con un control de tangencia para el pecho y uno para la cintura. La espalda tiene la posibilidad de alargarse el porcentaje que se escoja. También hay una serie de 9 controles para mover la barriga y las costillas del monstruo.

La cola del monstruo tiene un sistema de IK y uno de FK que vive encima del de IK. También tiene posibilidad de estirarse y contraerse. Los controladores del IK están emparentados entre sí para así definir mejor la forma de la trayectoria que seguirá la cola.

El monstruo tiene una serie de 21 tentáculos en la cabeza; animarlos totalmente a mano es bastante laborioso. Por tanto, tienen un sistema propio de control. Por defecto hay controles para mover cada tentáculo individualmente en su totalidad. Se pueden extraer los controles que definen la trayectoria principal del tentáculo. Bajo el sistema de trayectoria del IK tiene dos curvas dinámicas: una tiene aplicado un campo de turbulencia de baja frecuencia que le da la forma general, y la otra un campo de turbulencia de alta frecuencia que le da el micro detalle al tentáculo. Para que funcione la evaluación de los nodos, debe estar establecida en Dependency Graph.

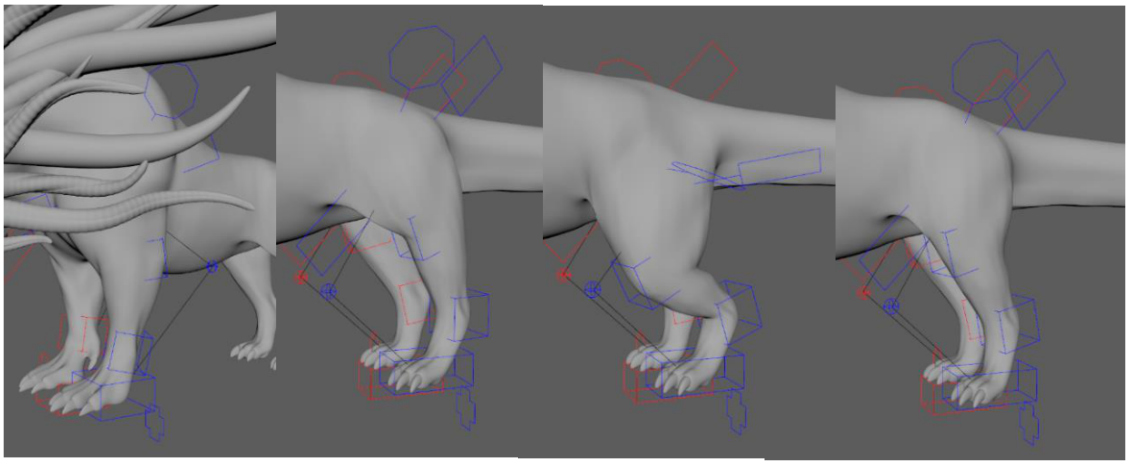

Ilustración 47: Sistema de IK de las patas del Monstruo

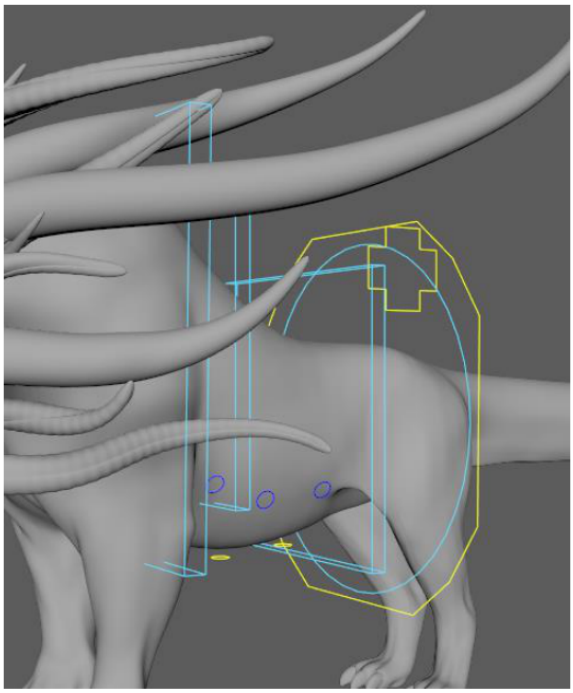

Ilustración 48: Sistema de la espalda del monstruo

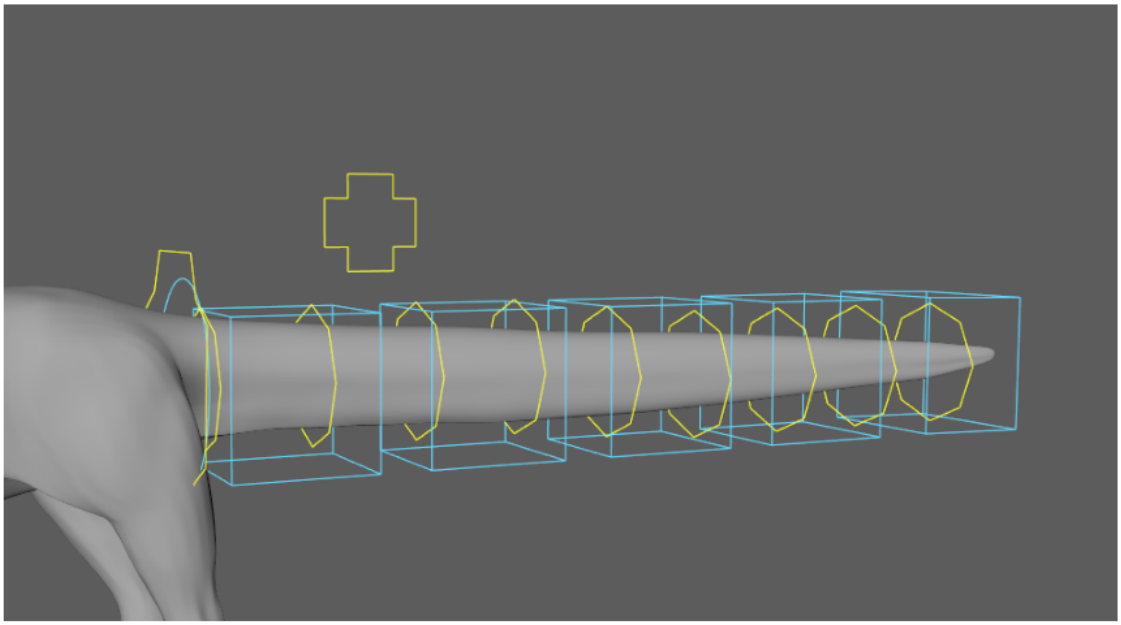

Ilustración 49: Sistema de IK de la cola

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.

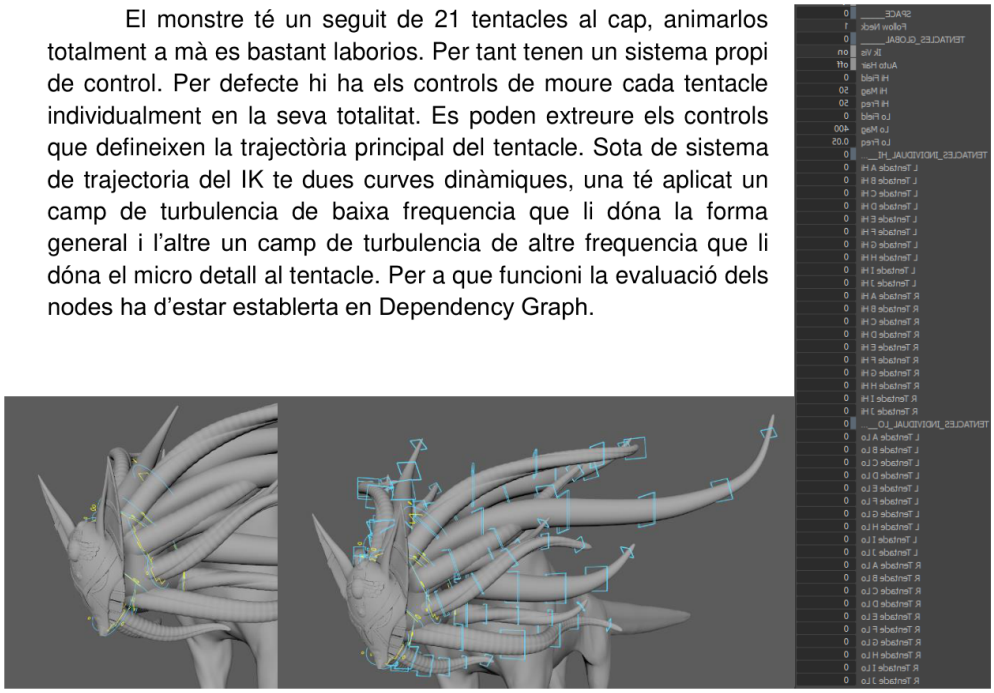

Ilustración 50: Sistema dinámico dirigido de los tentáculos

Nut

Es el robot de "World Of Lost Things". Es un personaje hecho a medida ya que no es un bípedo y necesita unas prestaciones concretas para animación.

En los brazos tiene un sistema de bend bones, para darle ondas al personaje.

Tanto en las cejas como en la antena tiene un sistema de IK, con dos controles principales y dos de tangencia, que le da la posibilidad de estirar y hacer formas orgánicas a objetos estáticos y mecánicos.

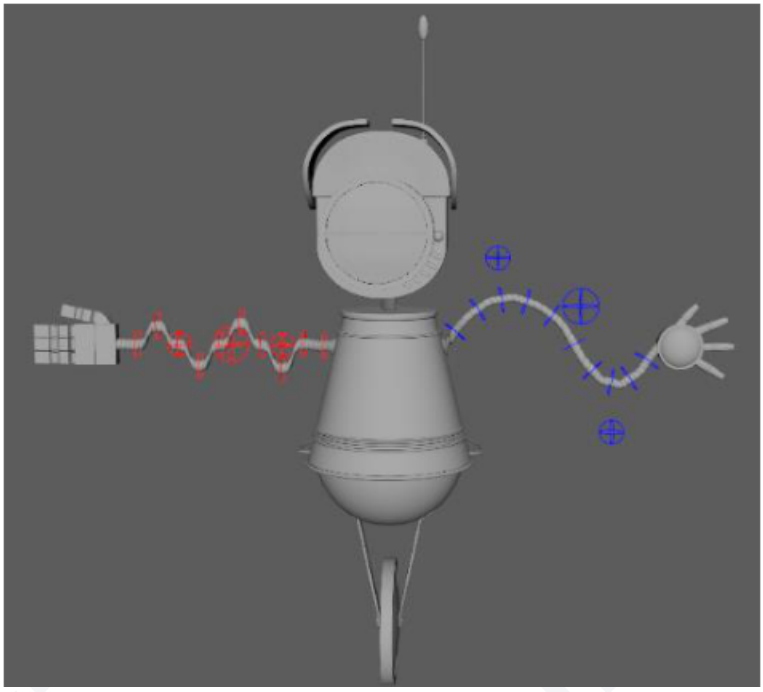

Ilustración 51: Sistema de huesos de doblamiento (bend bones) en los brazos de Nut

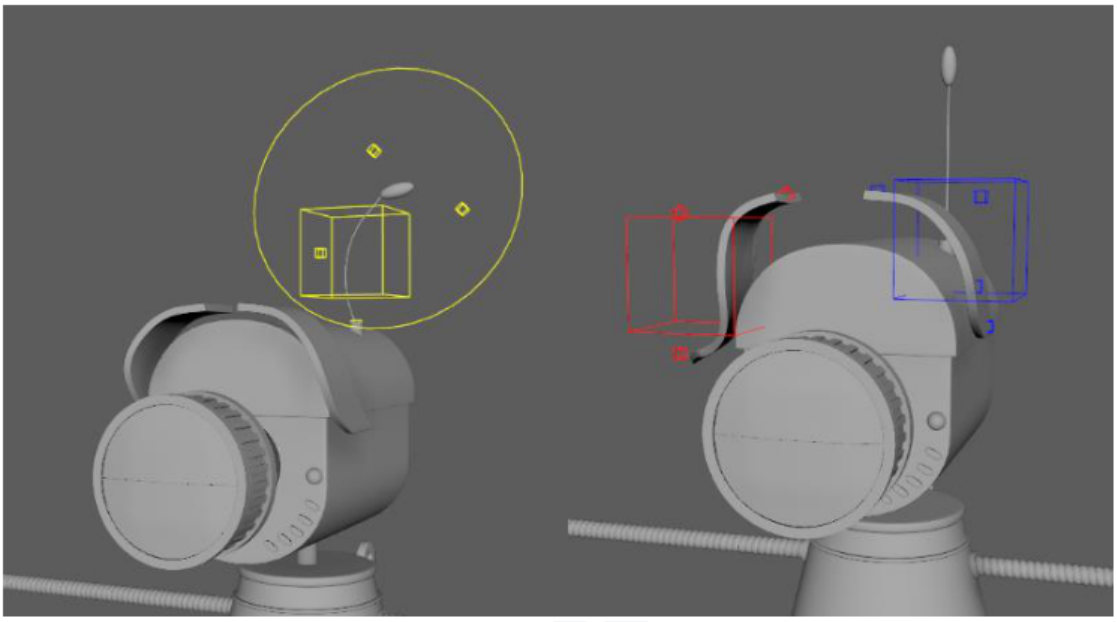

Ilustración 52: Sistema de control IK para Nut

Human IK

Para el proyecto de “La Universidad de Cardiff” se necesitaban personajes que pudieran funcionar con el sistema de mocap. Unos debían llevar animación por encima y los otros solo los datos del mocap. El sistema también es procedural.

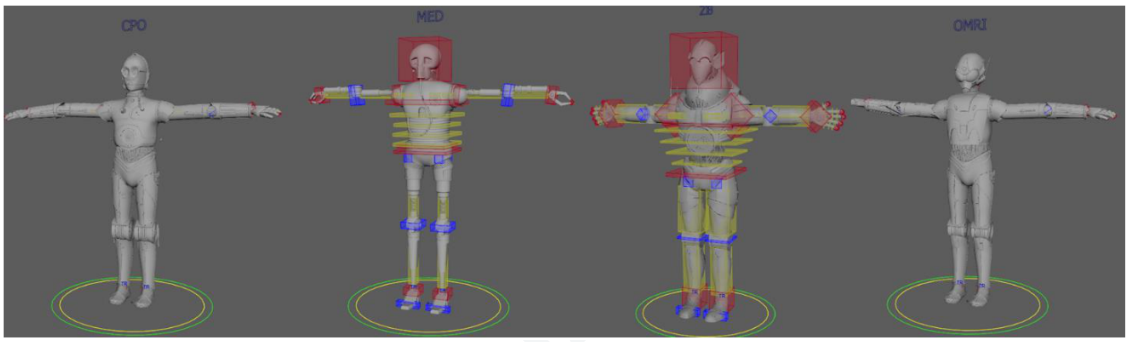

Ilustración 53: Personajes Human IK

Estáticos

También se necesitaban dos personajes que fueran estáticos pero a los que se pudiera animar su posición, rotación y escala.

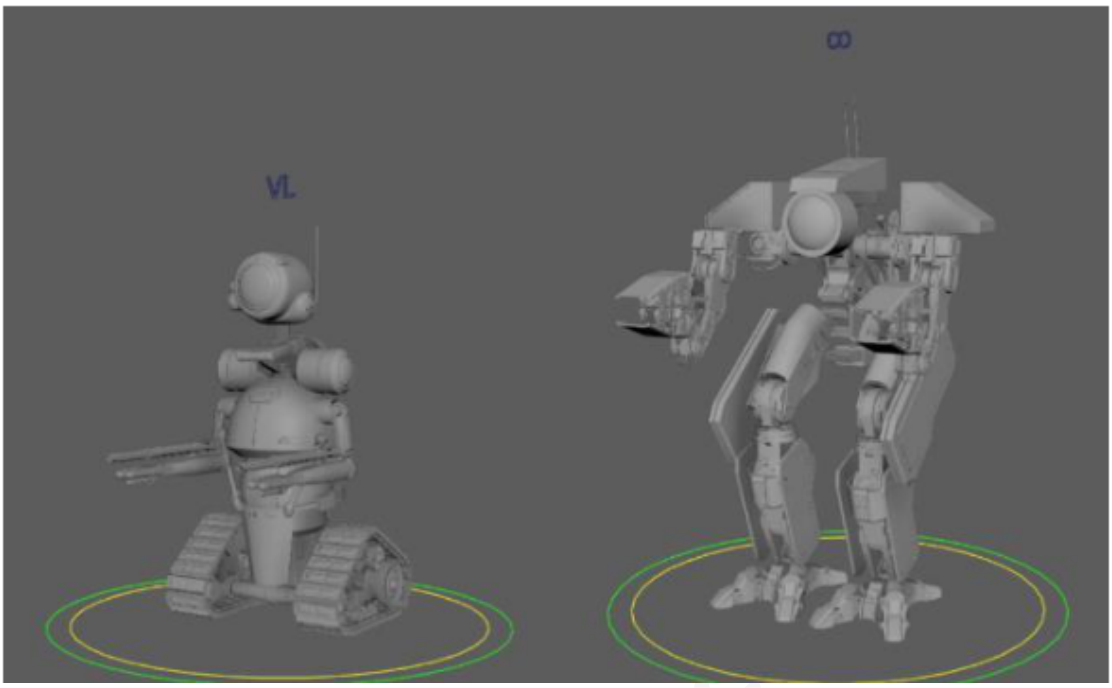

Ilustración 54: Personajes con controles generales

Pipeline

El pipeline es la parte de la producción que se encarga de la gestión del intercambio de datos entre las partes de la producción. Una correcta gestión de versiones es la clave para que una producción vaya correctamente y de manera ordenada.

Existen herramientas que controlan toda esta información, por ejemplo: “Ftrack” y “Shotgun”.

Por otro lado, también hay que controlar la gestión de las escenas; esto se puede hacer con diversos sistemas y programas. Como ensambladores hay: “Katana”, “Clarisse” y “Houdini”, programas capaces de gestionar grandes cantidades de geometría. También está el sistema de USD, desarrollado por “Pixar”, que ayuda a la gestión entre todos los programas, e “Hydra” para visualizar un viewport con los objetos.

Para los proyectos se decidió no usar ninguno de estos sistemas porque no los dominábamos y porque no había soporte. Esto complicó toda la gestión del proyecto, ya que toda la gestión de los assets se hizo manualmente, donde el error humano está más presente.

Como trabajo práctico de pipeline, desarrollé un sistema para que una serie de herramientas (tanto propias como ya existentes) estuvieran en todos los ordenadores solo con colocarlas en una carpeta del servidor. Este sistema se denominó “LS_mayaToolkit”. Para que funcione, hay que colocar el siguiente script en un archivo userSetup.py en la ruta de scripts de 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')
```

Tabla 1: Archivo userSetup.py

Allí se encontrará una recopilación de herramientas y scripts útiles para la producción.

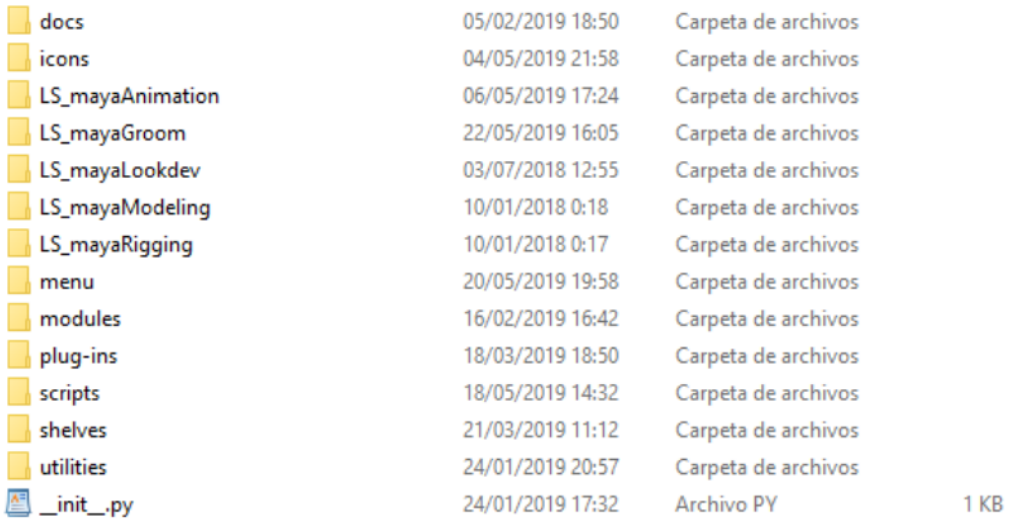

Ilustración 55: Estructura de carpetas de las utilidades en el servidor

Herramientas visibles

LS_Menu: Es la visualización de una serie de herramientas en un menú en Maya.

LS_mayaAnimation: contiene herramientas para ayudar a la animación

- aTools
- AnimFix
- Character importer

LS_mayaGroom: utilidades para la gestión del groom

- Import
- Export

LS_mayaLookdev:

- Transformaciones de luces
- Cambiar la exposición de las luces seleccionadas
- Duplicar objetos con historial
- Cambiar los perfiles de lectura de color cuando se usa OCIO ACES 1.0.3

LS_mayaModeling

- absSymMesh: Herramienta para operar entre geometrías, comprobar la simetría y arreglarla si es necesario.
- Braid Creation: Herramienta para crear trenzas
- goZ: Herramienta que intercambia la escena de Maya a zBrush y de zBrush a Maya.

Herramientas no visibles

Son las herramientas no visibles que solo funcionan por comando.

Módulos

brSmoothWeights. desarrollado por “brave rabbit”, su función es suavizar los pesos de la geometría con un algoritmo más avanzado que el que lleva Maya por defecto

Plug-ins

AnimSchoolPicker. Diseño y uso de una GUI para hacer sets de selección de controles; se puede diseñar a gusto. Sirve para optimizar la selección de controles. Creado por Anim School.

dcSkin. Desarrollado por David Cuellar, herramienta para guardar y cargar pesos en archivos binarios.

extractDeltas. Desarrollado por “brave rabbit”, sirve para extraer la variación entre una shape correctiva y la misma con los deformadores aplicados. Como dice su nombre, extrae las deltas de la mesh.

jQuadCloth. Desarrollado por Jacopo Ortolani. Retopologiza geometrías de muchos polígonos a unas donde todo está cuadrangulado. Funciona especialmente bien con la ropa.

ngSkinTools. Desarrollado por Viktoras Makauskas. Sirve para operar con los mapas de pesos. Da la posibilidad de tener capas de pesos aditivas. Internamente hace todos los cálculos para que estén normalizados.

smoothSkinClusterWeight. Desarrollado por “brave rabbit”, es el predecesor del módulo brSmoothWeights; no tiene tantas utilidades y funciona más lento.

transferSkinCluster. Desarrollado por “brave rabbit”. Sirve para copiar el skinning entre geometrías.

Scripts

advancedSkeleton5. Auto-rig para la creación de estructuras para controlar la geometría.

cometScripts. Conjunto de herramientas desarrolladas por Michael Comet, donde hay todo tipo de herramientas. La más conocida es el Comet Renamer.

cvshapeinverter. Desarrollado por Chad Vernon, sirve para extraer la forma inversa de una deformación y la mantiene dinámicamente.

MG-PickerStudio. Picker similar al AnimSchoolPicker, desarrollado por Miguel Gao.

mGear. Framework de rigging desarrollado por Miquel Campos, da una gran flexibilidad para la creación de estructuras de rig.

ml_tools. Una serie de herramientas y utilidades para animación y rigging, desarrolladas por Morgan Loomis.

studioLibrary. Herramienta creada por el usuario de GitHub "krathjen", para guardar poses y animaciones y así poder pasarlas entre archivos.

Zen. Conjunto de herramientas desarrolladas por David Belais, donde hay una serie de todo tipo de herramientas extra para Maya.

zooTools. Una serie de herramientas desarrolladas por Andrew Silke.

cosmos. Herramienta que gestiona el acceso a las ventanas de Maya. Desarrollada por Martin Gunnarsson.

sortCircleTool. Herramienta para hacer círculos sobre la serie de edges seleccionados.

bosSmear. Herramienta para deformar la geometría según el shot de cámara.

bSkinSaver. Herramienta desarrollada por Thomas Bittner, para guardar los pesos de los nodos skinCluster.

changeColors. Script para cambiar los colores de los controladores en masa.

compactRenamer. Script de Erik Lehmann para renombrar geometrías.

delete turtle. Script para eliminar el nodo turtle de la escena.

DPK_reorderAttrs. Herramienta desarrollada por Daniel Pook-Kolb para reordenar los atributos del channel box.

Dupes. Script para comprobar si hay objetos duplicados en la escena. Desarrollado por Jorn-Harald Paulsen.

mtAlignTool. Herramienta para alinear objetos en el viewport.

tweenMachine. Herramienta para interpolar keyframes, desarrollada por Justin S. Barrett.

ACES

El Academy Color Encoding System (ACES) se está estableciendo como el estándar de gestión de color en toda la producción de cine y TV.

Se establece como un proceso de gestión del color lineal y sin pérdidas. De esta manera, la cadena de color queda inalterada desde la grabación de la pieza audiovisual hasta su visualización, pasando por la creación del 3D.

Ilustración 56: Estructura de la gestión de los datos de ACES a lo largo de la producción

Para usar la configuración de ACES dentro de Maya, primero hay que descargar el repositorio de OCIO de Sony (https://github.com/imageworks/OpenColorIOConfigs/tree/master/aces_1.0.3). Una vez descargado, hay que activar la siguiente configuración.

Y para cambiar la configuración del espacio de color de todos los archivos que se cargan, hay que cambiarlos a utilidades (https://github.com/enriquevelmai/utils/blob/master/toAces.py)

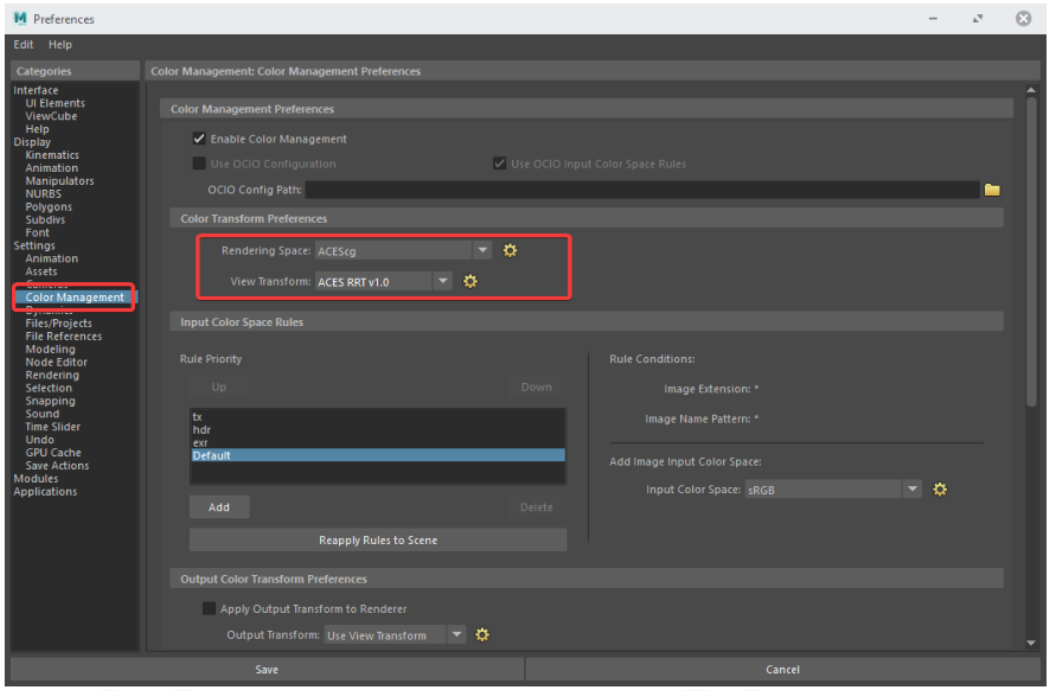

Ilustración 57: Configuración de la gestión de color dentro de Maya con los perfiles de ACES

Script “current”

Como no se usó un gestor de versiones, se creó un script “current” que leía las últimas versiones y creaba una carpeta con ellas.

Groom

La gestión del groom en el proyecto se convirtió en un quebradero de cabeza, ya que todos teníamos muy poca experiencia gestionando el cabello en el pipeline.

Se decidió usar el xGen de Maya para el cabello. Después de investigar a fondo cómo funciona el xGen internamente, se derivó nuestra propia manera de gestionar el cabello.

El xGen funciona teniendo una geometría base a la que se le aplican una serie de colecciones y descripciones.

Las colecciones son la agrupación de muchas descripciones sobre una o varias geometrías. En las descripciones es donde se peina el cabello a partir de una serie de guías, mapas y expresiones. Los mapas que se pintan están en formato PTX.

La información del xGen se guarda en un nodo verde dedicado en una escena de Maya, que es un plug-in que lee información de dos sitios: todo lo relacionado con las colecciones, descripciones y expresiones se guarda en un archivo .xgen con los datos guardados; la información de los mapas se guarda en una jerarquía de carpetas dentro de la carpeta de xGen del proyecto (o el sitio donde se defina), donde hay carpetas para cada colección, descripción y modificador.

Estructura del archivo .xgen

Dentro del archivo xgen los datos se estructuran de la siguiente manera

Globals

La “palette” es la base, lo que equivale a la colección.

Definición de las variables globales del xgen, mapeo de dónde está la base del proyecto.

```
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
```

Ilustración 58: Opciones globales de la colección

Por descripción

Datos generales de la descripción; hay uno para cada descripción

```
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/
```

Ilustración 59: Opciones generales de la descripción

Visualización de las guías 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
```

Ilustración 60: Opciones de visualización de las primitivas en viewport

Visualización de las guías en render; se puede definir el tipo de render que se quiere, qué scalp sigue y si tiene motion blur o no. Se pueden hacer overrides para cada una de las variables. Se puede ver que el motor de render que usa el xgen es “RenderMan”, ya que es 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
```

Ilustración 61: Opciones de visualización de las primitivas en render

También se pueden definir los datos generales de las guías de la descripción.

```
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
```

Ilustración 62: Opciones generales de las guías en viewport

A continuación están los datos de los modificadores; se cargan como paquetes de FX (establecido por 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
```

Ilustración 63: Opciones del modificador de clump

Otros ejemplos de datos que guardan otros tipos de modificadores.

```
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
```

Ilustración 64: Opciones de los modificadores Noise y Cut

El espacio del generador aleatorio gestiona la densidad, el offset y otras propiedades genéricas.

```
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
```

Ilustración 65: Opciones generales de la descripción

Propiedades de visualización de las guías 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
```

Ilustración 66: Opciones de visualización de las guías en viewport

El apartado de texturas de mapas dice de dónde se lee parte de los mapas de los modificadores. También indica qué partes del xgen están activas para la descripción en cuestión y qué método de render se 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
```

Ilustración 67: Rutas de lectura de los mapas pintados

Anclaje

Una vez definidos todos los datos de visualización y modificación de cada una de las descripciones, se definen todos los datos relacionados con los scalps, donde se reflejan las posiciones de cada una de las guías, los patches a los que se enganchan, los CVs que tienen y muchos más. A continuación hay un extracto de esta parte del archivo, ya que es la parte más grande del archivo.

```
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
```

Ilustración 68: Propiedades de anclaje entre las guías y la geometría

Estructura de guardado de los mapas

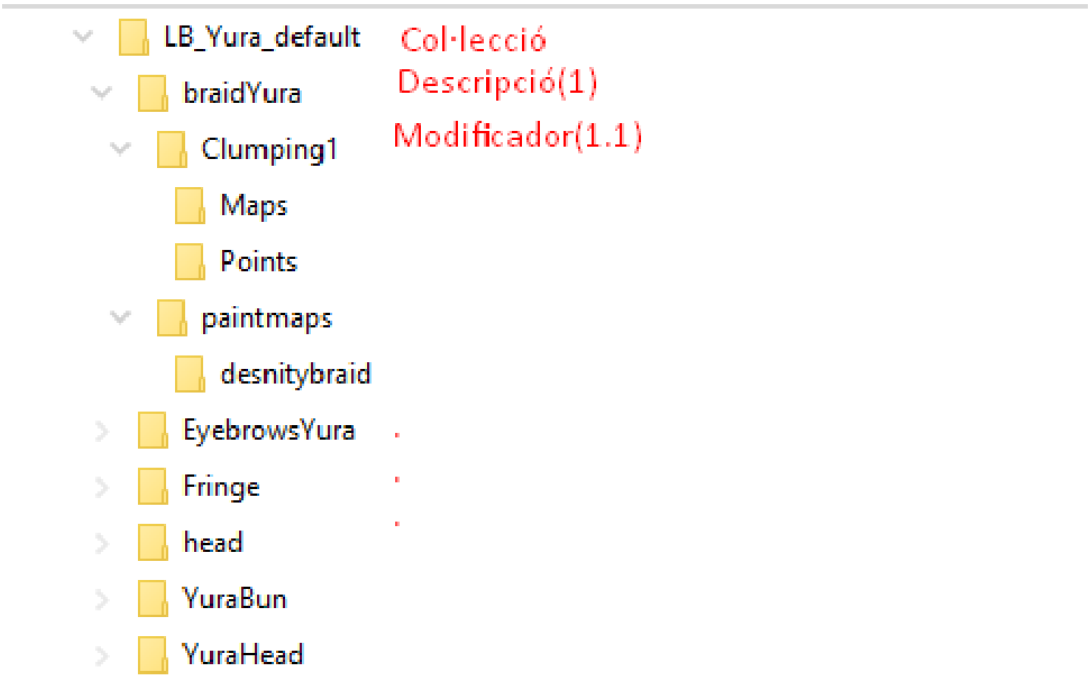

Ilustración 69: Estructura de carpetas para guardar los mapas

Concepto de empaquetado

Como se puede ver, hay muchas variables que intervienen en la gestión del cabello, por tanto se define el concepto de empaquetado del cabello. Se trata de tener un contenedor que guarde todos los datos necesarios para exportar y volver a montar el sistema de deformación del cabello.

Para poder gestionar el xGen, definimos tener los siguientes elementos:

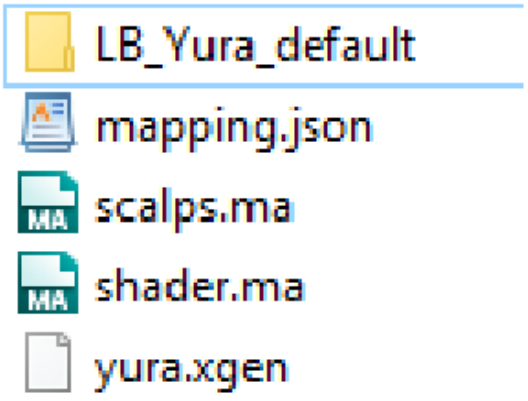

Ilustración 70: Estructura propia de paquete de groom

La carpeta contiene toda la estructura de mapas PTX de los modificadores y descripciones.

El archivo mapping.json contiene información útil para volver a montar el groom.

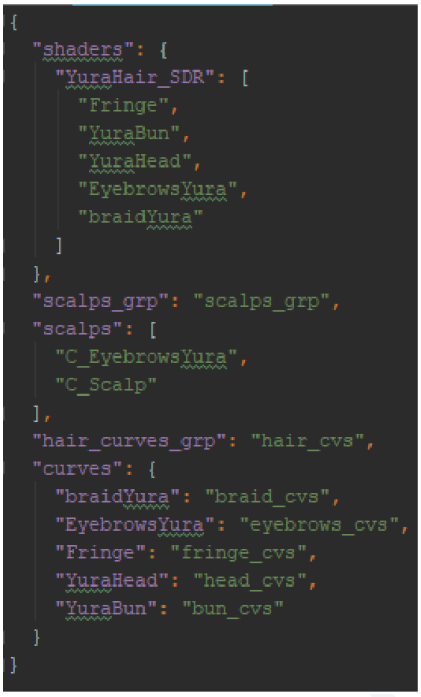

Ilustración 71: Estructura JSON propia de guardado de información

El archivo scalps.ma contiene los scalps de donde crece el cabello; hay que exportarlo manteniendo el historial de las geometrías, ya que tienen mapas y file nodes conectados para que se pueda volver a montar sin pérdidas.

El archivo shader.ma contiene el material del cabello.

El archivo .xgen es la exportación del archivo mencionado anteriormente.

Animación vs Simulación

Hay dos maneras de gestionar que el cabello se mueva. La primera es simulándolo: una vez hecha la animación, hacer que el cabello se mueva siguiendo dinámicas físicas y campos de fuerza. La segunda es animar una proxy mesh sobre la que se enganchan curvas, y estas se usan para mover el groom. Es importante exportar un archivo .abc con las guías específicas para cada descripción y para cada shot.

Gestión de los datos

Para gestionar estos datos sin caer en el error humano — ya que hay muchos pasos — definimos un script para gestionar estas cosas.

Las siguientes librerías permiten el uso de los comandos de xGen:

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

Tabla 2: Importación de las librerías de xGen

Estas herramientas se dedican a importar y exportar el groom genérico de manera genérica.

El exportador necesita definir el nodo de xGen, seleccionar los scalps, el shader y la ruta donde se exportará el paquete definido anteriormente.

El importador necesita la ruta del paquete, el alembic con la animación del shot y la ruta del xGen por si ha cambiado.

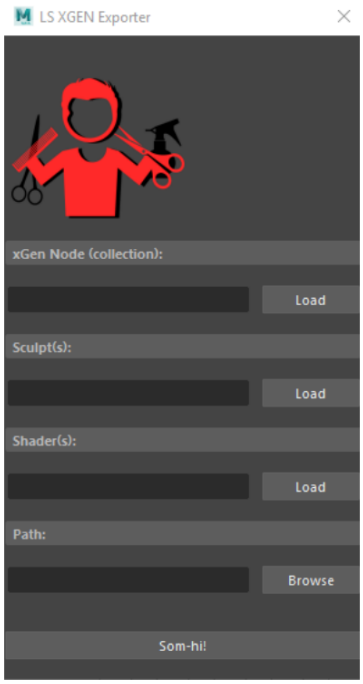

Ilustración 72: Herramienta para exportar el xGen

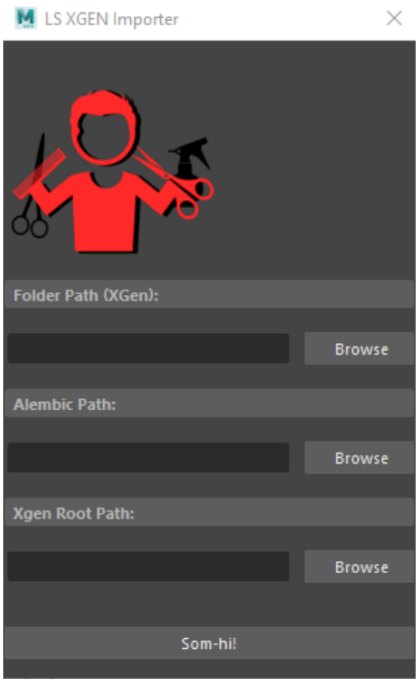

Ilustración 73: Herramienta para importar el xGen

Además de las herramientas anteriores, hay una herramienta específica para Last in Battle, basada en su propia gestión de archivos. Aquí solo hace falta definir el asset, el shot y el grupo de scalps.

Ilustración 74: Herramienta para importar el xGen, adaptada para Last in Battle

Fixing

Una vez hecha la animación, puede haber penetraciones de la geometría o que no se consiga la deformación deseada, y hay que pasar a la parte de shot sculpting.

Allí se coge el cache de la animación y, con la siguiente herramienta, se modela la deformación deseada.

Método de uso:

1. Posicionarse en el frame donde se quiere modelar la corrección
2. Abrir la herramienta
3. Seleccionar las geometrías sobre las que se quiere modelar la corrección
4. Definir los frames de cola que tendrá la corrección (una blendshape con 3 keys, dos de 0 y una de 1)
5. Pulsar el botón Active y modelarlas
6. Pulsar el botón Finish; las geometrías se eliminan

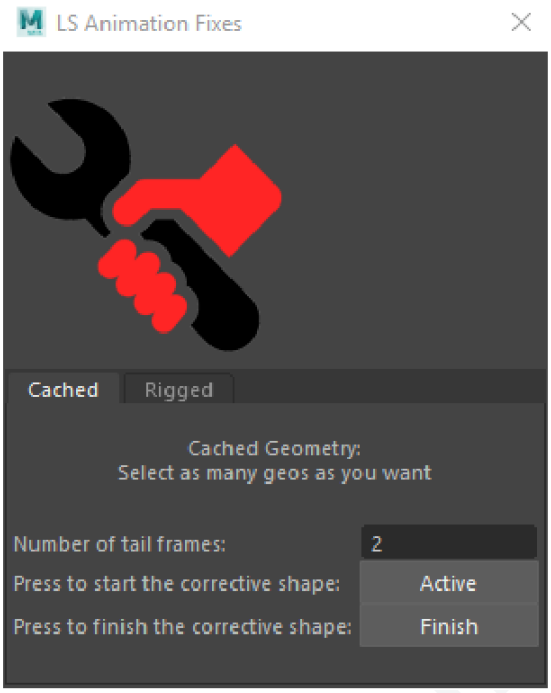

Ilustración 75: Herramienta para hacer el shot sculpting

Conclusiones

Llevar a cabo este proyecto ha servido para coger una visión global y completa de lo que es el desarrollo de un proyecto de inicio a fin. Pasando por cada uno de los apartados, he podido ver cuáles son las complicaciones y los procesos a seguir en cada fase. Costó mucho saber cuándo cerrar un proceso, es decir, cuándo establecer que el modelo, las texturas, el rig, la animación, etc., ya era suficiente y había que dejar fluir el asset por el pipe. Trabajando en grupo, nos tuvimos que organizar, ya que no es lo mismo hacer un proyecto totalmente solo. Las escenas debían tener un nombre concreto, el orden en el outliner había que llevarlo muy cuidado y los materiales publicarlos con los menos errores posibles. Al ser la primera vez que nos enfrentábamos — como universidad y como grupo — a una producción completa, la manera en la que teníamos que proceder fue un reto. Por ejemplo, cómo gestionar el cabello. Como no lo simulábamos sino que lo animábamos, costó encontrar una manera de unir la animación de las guías con las guías del groom. Una vez funcionaba localmente, nos dimos cuenta de que no funcionaba en la granja, ya que “Deadline” no cambiaba las rutas de la unidad de red mapeada configuradas como letra a la ruta completa en los archivos de xGen. Otra dificultad fue cómo se unían los materiales con la geometría animada. Las opciones que se contemplaron fueron: que el rig lleve los materiales; que se escriba un trozo de código que importe y configure los materiales una vez calculada la animación; o un sistema de Alex Leon que consiste en montar una escena con los materiales aplicados y después dejarla como referencia en la escena de lighting.

Crear un sistema totalmente procedural y orientado a objetos es más tedioso de lo que parece. Hay que estructurarlo y diseñarlo bastante de antemano, haciendo suposiciones de los obstáculos que aparecerán por el camino. También he estado atento a los nuevos métodos que se van publicando y a las herramientas que salen, para poder adaptarlas rápidamente al sistema creado. Personalmente, crear y mantener 13 personajes con todos los arreglos de rig y cambios de geometría se me hizo muy duro. Ahora, viéndolo en perspectiva, veo que escoger hacer los rigs procedurales fue una buena

decisión, ya que me veo incapaz de haberlo hecho de la manera “artesanal”, además de la implementación de las herramientas. Uno de los principales problemas en los rigs fue el flipping del advanced twist en los sistemas de IK spline; después de días peleándome, llevé el problema al trabajo y me aconsejaron usar el IK spline para calcular la posición y, a partir de ahí, crear una curva de offset a la del IK para marcar la posición, y usar una proyección de esta sobre la curva de offset, usando esta segunda curva como up vector de un sistema de aim, donde el hueso apunta al siguiente. Esta solución arregló los flippings; los módulos de extremidad, cuello, espalda y tentáculo tienen estos sistemas implementados. Las dobles transformaciones también dieron quebraderos de cabeza al escalar el modelo; crear grupos que no quieren las transformaciones en cada módulo de rig fue la solución a este problema.

De todos los personajes, el que fue un reto mayor por la complejidad de los sistemas fue el monstruo, por sus tentáculos. Aunque los sistemas funcionan correctamente, el rig va muy lento cuando se activan los automatismos. Me gustaría encontrar una solución más adelante para arreglar este punto en contra del rig.

Los sistemas para proyectar nodos de transform sobre una superficie fueron uno de los puntos más clave de todo el proyecto, ya que dieron mucho juego a los sistemas faciales de cejas, labios y párpados.

Pintar pesos en dual quaternion en algunas partes ha ahorrado modelar muchas shapes correctivas. El uso de la nueva herramienta brSmoothSkinWeights fue también uno de los grandes descubrimientos del año; ha agilizado mucho el trabajo.

Al ser una persona la que desarrolló el sistema y lo ha ido manteniendo, el tiempo de producción fue limitado, y muchos sistemas que estaban planeados como extras no se pudieron llevar a cabo. Menos mal que estaban los compañeros y compañeras que ayudaban y hacían funcionar las cosas desde los departamentos previos de groom y modelado. Y por la ayuda que hubo a la hora de empezar a modelar las shapes faciales. También habría ayudado tener a alguien presente en el día a día que pudiera echar una mano con la programación de las herramientas, ya que muchas cosas me sobrepasan, y tener a alguien con experiencia habría sido muy productivo; también habría aprendido otras maneras de proceder, no solo la mía.

Actualmente el código del Rigging Toolkit está muy desordenado a nivel de código. Me gustaría dar otro repaso al sistema y abstraerlo aún más, para dejar solo las clases más genéricas y limitar el rig a un sistema de spline y un sistema de proyección sobre superficie, con todo heredando de estos. También, cambiar el uso de maya.cmds por PyMel, ya que está totalmente orientado a objetos. Aparte de eso, sería genial darle una interfaz gráfica para hacerlo más usable.

Disfruté mucho desarrollando herramientas para pipeline, ya que era algo que nunca había hecho y salió bastante bien.

Referencias

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

Bibliografía y Webgrafía

ROB O'NEILL. Digital Character Development: Theory and Practice. Second Edition. A K Peters/CRC Press. 27 October 2015. ISBN: 1482250772

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

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

ALBERTO LOLLI. Struttura uomo. Manuale di anatomia artistica. Colla Editore; 2nd ed. 19 April 2018. ISBN: 8894272214

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

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

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

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

ADRIAN HERBEZ. Maya Programming with Python Cookbook. Packt Publishing. 29 July 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; 1st ed. 1 November 2011. ISBN: 0123785782

ROBERT GALANAKIS. Practical Maya Programming with Python. Packt Publishing. 25 July 2014. ISBN: 1849694729

MARK SUMMERFIELD. Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Prentice Hall; 1st ed. 4 September 2015. ISBN: 0134393333

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

XGen Python API. Available at: 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. Available at: http://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=__files_GUID_B1CD0989_EB49_46BE_934F_F23BFB869142_htm

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

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

Josh Sobel Face Rigging. Available at: https://vimeo.com/ondemand/sobelfacerig/

Cult of Rig. Available at: http://www.cultofrig.com/

Skinning Techniques. Available at: http://www.3dfiggins.com/writeups/paintingWeights/

Blendshape Techniques. Available at: http://petershipkov.com/tutorials/blendShapes/blendShapes.htm

Bind Pose. Available at: https://bindpose.com/

Doug Schieber. Available at: https://www.dougschieber.com/new-index

MICHAEL TODD – xGen. Available at: https://www.mtodd.work/xgen-product-design

Allan McKay Podcast. Available at: https://www.allanmckay.com/1/

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