La Préparation et Réalisation d'un Rig: Les Rigs du Court " L'Armoire"

Ana Maria Lacorazza

Master Arts et Technologies de l'Image Virtuelle

Année 2010/2011 - Soutenu en Septembre 2011

Sommaire

Introduction

Lorsque j'ai commencé à réfléchir sur un sujet pour écrire mon mémoire, le rigging m'est venu assez naturellement. C'est un sujet qui m'intéresse énormément, et auquel j'aimerais bien me consacrer un jour. Cependant, une fois le choix du sujet fait, il restait encore un grand nombre de questions à se poser. Je voulais écrire un mémoire sur le rigging... mais, comment aborder ce sujet?

Il y avait une chose dont j'étais sûre: je ne voulais pas écrire un document en décrivant pas à pas la création des rigs pour mon projet. Ceci n'aurais certainement pas eu d'intérêt pour moi, car je connaissais déjà bien la façon dont j'avais abordé la création de chacun des rigs. Et cela aurait été encore moins intéressant pour le lecteur: la plupart des techniques que j'ai adoptées lors de la création de ces personnages existent déjà ailleurs, et certainement expliquées d'une façon plus claire et mieux illustrées de ce que je pourrais faire sur ce texte.

La problématique que j'ai choisi alors, n'était pas "Comment créer un rig?", mais plutôt "Comment créer un BON rig?". Comment fallait-il aborder l'ensemble du processus de rigging pour qu'il soit le plus clair possible, et pour garantir de bons résultats en animation? Cette question me semblait d'importance capitale pour tous ceux qui créeraient un jour un rig.

Lors de mon année L3, j'ai dû rigger un personnage, une femme, pour mon court de fin de licence. Pour ce faire, j'ai fait un peu comme tout le monde: j'ai suivi un tutoriel. J'ai simplement regardé ce qui était fait sur la vidéo, et j'ai reproduit les mêmes gestes sur mon personnage. Le résultat était déplorable: non seulement j'étais incapable de faire tourner mon personnage, et devais donc tourner tout l'ensemble du décor pendant l'animation, mais aussi je n'étais absolument pas en mesure de réparer mon rig si jamais il y avait un problème. J'ai dû donc refaire mon rig médiocre à plusieurs reprises.

Evidemment, j'ai commencé mon année M1 en me disant que les riggers devaient être des masochistes, car cette partie de la réalisation d'un court-métrage était l'une des choses les plus pénibles qui existait dans l'ensemble des tâches à effectuer pour la réalisation d'un film. Heureusement, j'ai eu une deuxième chance pour apprendre à rigger, et cette fois-ci bien comme il faut (merci Cédric :)).

Lors de ce cours, je me suis rendu compte que le rigging n'étais pas du tout pénible. Il s'agissait juste d'une discipline dans laquelle il fallait être très rigoureux, et surtout, pour laquelle il fallait bien connaître ses outils.

C'est pour cela que ce mémoire, en lieu de faire une sorte de tutoriel, aborde les choses d'une façon un peu plus structurelle. Rassurez vous, je n'ai rien contre le fait de suivre des tutoriels. Le problème est de les suivre à la lettre, sans bien comprendre ce qui se passe derrière. Il n'y a jamais deux personnages identiques, et chacun à besoin d'avoir un rig qui soit créé spécialement pour lui. Et pour ce faire il est extrêmement important de ne pas mémoriser ce qui se fait dans un tutoriel où l'autre, mais d'entrevoir le fonctionnement de base de chacune des techniques exposées, pour après avoir la capacité de les adapter à n'importe quel personnage, et de les mélanger avec d'autres techniques pour créer le rig le plus approprié possible.

Dans les pages qui suivent, j'expose alors plusieurs techniques. Je développe les caractéristiques de base de chacune, en dirigeant le lecteur vers les sources où il pourra trouver chacune d'entre elles, tout cela dans le cadre de la création de mes propres rigs pour le court "L'Armoire". Mon but est de donner les bases nécessaires pour que chacun puisse facilement choisir les techniques qui lui conviennent le plus, suivant ses besoins et ses préférences. J'espère avoir réussi.

Connaissances nécessaires avant de commencer à rigger

Le but du jeu

Si on posait la question « Qu'est ce qu'un rig ? », la réponse serait très simple : Un rig est l'ensemble des dispositifs qui permettent d'animer un objet. Dans ce sens, si on apparentait l'animateur 3D à un marionnettiste, le rigger serait celui qui prépare la marionnette, attachant chacun des fils avec soin pour que celle-ci puisse être menée à la vie de la manière la plus naturelle possible.

Mais allons à présent un peu plus loin, et reformulons cette question comme suit : « Qu'est ce que c'est un BON rig ? ». La réponse est maintenant beaucoup plus complexe. Est-ce qu'un bon rig est celui qui aura la plus grande quantité de contrôleurs à disposition de l'animateur ? Ou bien celui qui automatise la plupart des tâches, avec un minimum de contrôle disponible, et de travail nécessaire de la part de l'animateur ? Est-ce que la réponse à cette question varie d'animateur en animateur, de modèle en modèle, et il n'y a pas, donc, de réponse concrète ?

Le segment qui suit, extrait du livre Animation Friendly Rigging de Jason Schleifer, nous donne une première idée des caractéristiques générales qui doit comporter un bon rig :

« Imaginez que vous conduisez une voiture, et à chaque fois que vous essayez de tourner à droite, l'essuie-glace se met en marche. Vous tournez le volant à gauche, et la voiture accélère. Quand vous appuyez sur l'accélérateur – le son de la radio monte. Si vous changez de station radio, la voiture commence à rouler lentement vers l'arrière. Bien sûr, à terme, vous arriveriez à votre destination. Mais votre vie serait certainement un cauchemar à cause de la frustration causée par ces dysfonctionnements répétés... En fait, vous ne voudriez probablement plus jamais conduire.

En revanche, un bon rig d'animation marchera exactement comme vous vous l'attendez. Tous les contrôleurs auront un sens, ils seront faciles à comprendre, et marcheront de manière cohérente. Un rig créé tenant compte des besoins de l'animateur sera un bonheur pour tous ceux qui s'en serviront, et permettra à l'animateur de se concentrer plus sur son travail, et moins sur la recherche d'un moyen pour arriver à s'imposer sur le sacré truc. » [1]

En plus d'être clair et fiable, un bon rig doit aussi être simple. De manière générale, il doit permettre à l'animateur de faire son travail de la manière la plus transparente possible : un rig ne doit jamais

être un obstacle pour la performance de l'animateur, il doit lui permettre de faire son travail sans sacrifier sa liberté créative.

Pour créer un rig doté de ces qualités, le rigger doit trouver l'équilibre entre [1]:

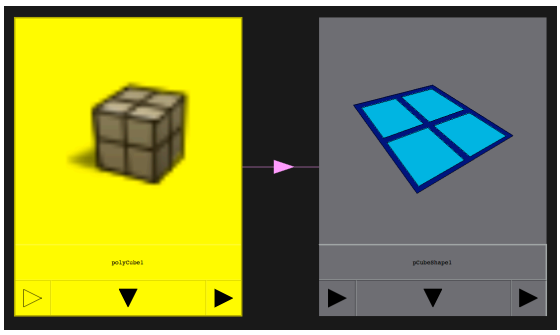
1. Les besoins en termes de la performance pour l'objet à rigger.
2. Les nécessités/envies de l'animateur qui travaillera avec cet objet.
3. Les exigences techniques pour l'animation de l'objet.

La création d'un rig nécessite, alors, beaucoup de préparation. Les storybards doivent être analysés pour découvrir le style de rig requis, et le type des mouvements faits par le personnage. Des références de personnages similaires, réels où imaginaires, doivent être consultées pour noter les types de déformations et comportements qu'elles exhibent. Enfin, et surtout, les besoins et préférences de l'animateur doivent être intégrés au mélange. On s'attardera sur ce processus de préparation un peu plus tard. Dans les pages qui suivent, nous examinerons quelques uns des concepts qui doivent être clairs avant de commencer n'importe quel rig.

Les Nodes

Afin de construire un rig stable et compréhensible, il est essentiel d'avoir une bonne connaissance de l'architecture de Maya, de la manière dont tous les objets sont représentés au sein du logiciel. Cette compréhension nous permettra d'avoir une vision claire de ce que nous pouvons faire, et de ce que ne doit pas être fait lors qu'on manipule des objets dans Maya. Nous serons ainsi capables d'entrevoir le fonctionnement de base de tous les éléments que nous créerons pour un rig donné.

Lorsqu'on ouvre Maya, on est confronté à son interface. Derrière cette interface, Maya fonctionne sur des commandes MEL, qui agissent sur le Dependency Graph, une collection de nodes qui décrivent tout ce qui se passe dans notre scène. Les nodes sont des petits modules qui contiennent des informations spécifiques, ou qui réalisent des opérations sur d'autres modules, qui possèdent des entrées et des sorties qui varient en nombre et en type de node en node. Les animations, la géométrie, les contraintes, les dynamiques... tout est présent dans le Dependency Graph sous forme de node, et les connexions entre les différents éléments décrivent la façon dont l'information parcoure la scène.



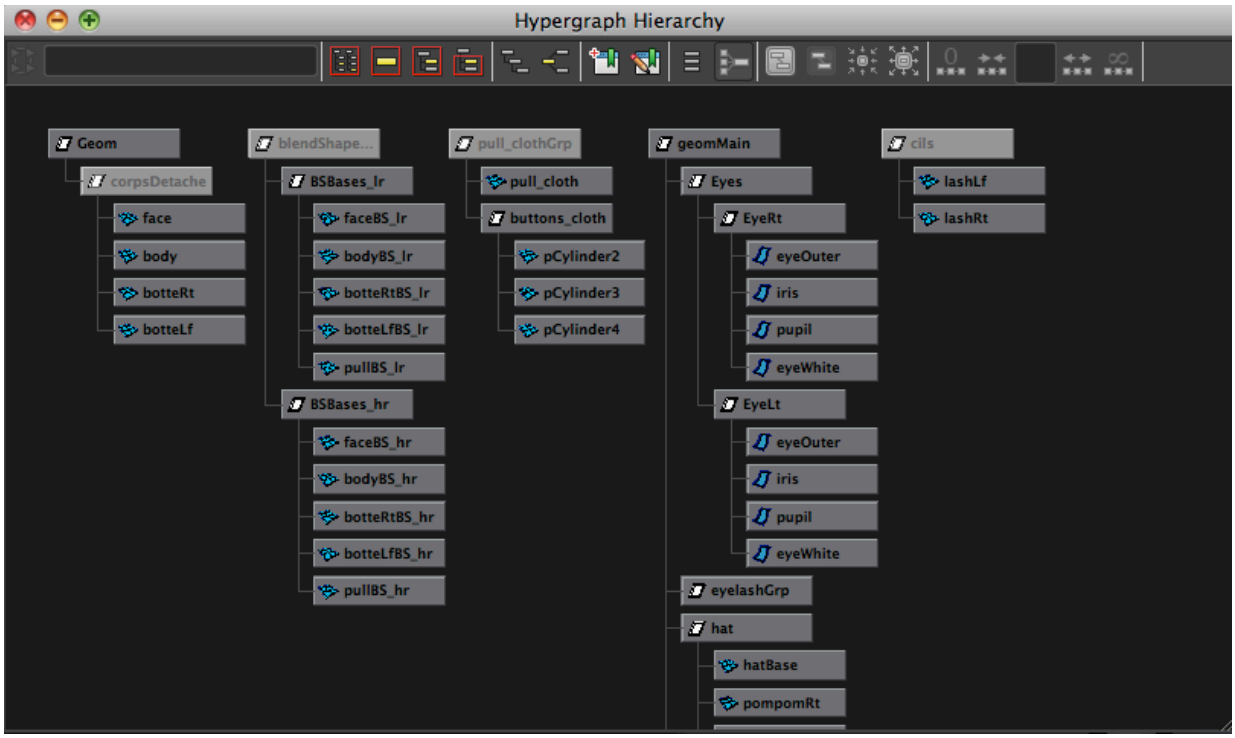
Les nodes associés à un cube dans Maya

Toutes les opérations dans Maya sont alors essentiellement des connexions entre ces modules. Par exemple, lors qu'on crée un simple cube dans une scène vide, deux nouveaux nodes apparaissent, contenant les informations du cube que l'on vient de créer .

De façon plus générale, tous les nodes ont des caractéristiques qui leur sont propres, appelées attributs [3]. Un node de type transform par exemple, contient toutes les informations relatives à la position d'un objet. Parmi ses attributs on retrouve, entre autres, l'attribut translateX, qui contient la valeur de la translation de l'objet dans l'axe X de la scène.

Il existe aussi des nodes conçus pour réaliser des opérations sur des données venant d'autres nodes. Pour donner un exemple simple, le node multiplyDivide nous permet de réaliser une multiplication ou une division avec des informations récupérées d'autres nodes. Celui ci est un node que l'on utilisera souvent dans le processus de rigging.

La totalité des nodes de la scène peut être vue dans l'Hypergraph. Cet outil nous permet de voir comment est organisée la hiérarchie de notre scène, et les différentes connexions entre nodes. On peut aussi voir le réseau de nodes entourant un objet donné dans l'Hypershade.



Un aperçu de l'Hypergraph.

Toutes les actions que nous effectuons à travers l'interface de Maya sont donc traduites par la création des nodes, ou la modification des nodes déjà présents dans le Dependency Graph. Là où tout cela devient intéressant, est lorsqu'on décide de créer des nodes "à la mano" pour modifier les objets présents dans la scène. Ces nodes peuvent être créés soit avec la commande MEL createNode, soit avec l'aide de l'Hypershade. Ensuite, pour qu'un node agisse sur un autre, il ne reste plus qu'à les connecter. Pour ce faire, on peut utiliser le Connection Editor.

Cet outil affiche, d'un côté, les paramètres de sortie d'un node, et de l'autre les entrées du node qui recevra la connexion. Pour connecter deux objets il suffit de sélectionner le premier objet, cliquer sur "Load Left", sélectionner l'objet auquel nous voulons le connecter, et cliquer sur "Load Right". Ensuite nous pourrons associer un attribut du côté gauche à un autre du côté droit pour symboliser la connexion. Evidemment, seulement les attributs du même type peuvent être connectés entre eux. Comme on verra plus tard, les associations faites de cette façon, par connexion directe, sont

permanentes, et affecteront les nodes en question jusqu'à ce que la connexion soit brisée, au moins d'utiliser un troisième node pour modifier cette relation, comme un node condition, par exemple.

Les nodes " dag"

Jusqu'à présent, nous avons vu que la structure de toute scène de Maya est essentiellement un réseau de nodes, et que nous pouvons créer des nodes nous mêmes pour réaliser des opérations sur les différents objets de notre scène. Dans l'état des choses, nous pourrions dire que nous pouvons connecter les nodes de façon entièrement arbitraire, pourvu que les attributs connectés soient du même type.

Cependant, ceci est uniquement vrai pour un des deux types de nodes de Maya. Dans Maya, nous avons deux types de nodes: les nodes de type DG, ou Directed Graph, et les nodes DAG, Directed Acyclic Graph.

La plupart de nodes dans Maya sont de type DAG. Ces nodes sont ceux qui sont visibles dans l'Hypergraph et l'Outliner. Avec eux, nous pouvons définir des relations hiérarchiques entre les différents objets. Lors qu'on travaille avec des nodes DAG il faut garder en tête que, comme leur nom l'indique, ces nodes sont acycliques, c'est à dire, qu'ils ne permettent pas de créer des relations dans lesquelles un objet est à la fois enfant et parent d'un autre.

Les nodes de type DG, en revanche, peuvent être connectés de façon arbitraire, et sont invisibles dans l'Outliner. Les keyframes et les shaders, par exemple, sont des nodes de type DG [4]. Bien que ces nodes permettent des connexions cycliques, les résultats de ce type d'opération peuvent être imprévisibles.

Etant donné que tout ce qu'on fait dans Maya peut être fait en manipulant des nodes, et que tout node dans Maya peut être créé et connecté à d'autres nodes avec quelques lignes de code, il s'avère que les nodes sont très utiles pour automatiser certaines taches du processus de rigging, ce qui est toujours encouragé.

Pour avoir une vision plus détaillé de la façon dont fonctionnent les nodes de Maya, je recommande le livre [3].

Les joints

Il est vrai, il est possible de créer un rig simple sans jamais poser un joint. Après tout, il existe des déformeurs, des contraintes, des expressions... Mais dans 99% des cas (à peu près), un rig comportera au moins une chaîne de joints. Et dans 98% des cas, ces chaînes de joints seront la base de tout le rig. Il est donc très important de comprendre comment fonctionnent les joints, car une chaîne de joints mal construite sera sans doute une source inépuisable de maux de tête et, selon le caractère de l'animateur, de souris cassées.

Dans Maya, un joint est un objet distinctif, doté d'une position, des rotations, et d'une orientation particulière, laquelle déterminera la façon dont les rotations seront effectuées. Cet objet est visible dans le viewport, mais il est invisible en rendu.

Les chaînes de joints sont des ensembles de ce type d'objets, liés entre eux de manière hiérarchique. Ces ensembles se comportent, à quelques différences près, comme toute autre hiérarchie dans Maya : les enfants héritent les rotations et translations effectuées sur les parents.

Un squelette est, à son tour, une collection de chaînes de joints, qui servira de structure de base pour le personnage animé.

Cinématique directe et cinématique inverse, « Fk » et « Ik »

On a vu dans la section précédente que les chaînes de joints se comportent comme toute autre hiérarchie dans Maya : Les transformations appliquées aux parents affectent la position des enfants. Ce mode de transformations est appelé cinématique directe, ou FK (pour «Forward Kinematics »).

Pour poser la main d'un personnage sur une table en transformant le squelette en FK, on devrait d'abord appliquer une rotation sur l'épaule, ensuite sur le coude, et enfin sur le poignet. Si on devait animer la main du personnage qui glisse sur la table, on devrait recommencer ce procédé, en espérant que le mouvement entre les deux poses semble naturel.

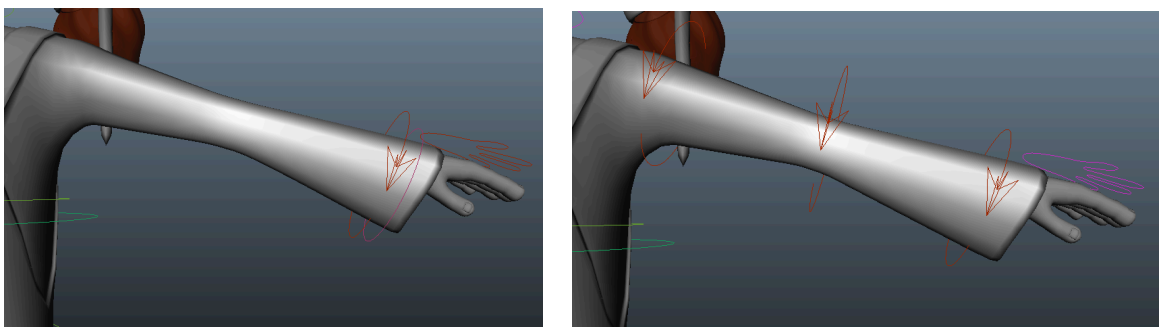
Heureusement, pour manipuler les chaînes de joints, il existe un deuxième mode de transformation, par cinématique inverse, ou IK (pour « Inverse Kinematics »). La cinématique inverse est le fait de modifier les rotations des joints intermédiaires d'une chaîne de joints en se basant sur la position

d'un contrôleur placé sur le dernier joint de celle-ci : c'est l'enfant qui contrôle les joints que le précèdent dans la hiérarchie, ses parents. On appelle ce type de déformation « goal directed motion». Dans Maya, ce mode de transformation est défini par un objet particulier, appelé IK Handle, qui est créé en sélectionnant le premier et dernier joint de la chaîne que l'on veut contrôler. La façon dont la chaîne sera déformée dépend du type de Handle utilisé. Il existe trois types de base d'IK Handle : Single Chain IK, Rotate Plane IK, et Spline IK (para mas info aide de maya). Dans les trois cas, l'IK Handle est représenté par une ligne droite allant du début à la fin de la chaîne.

Avec L'aide des IK Handles, on peut placer librement les bras du personnage, en plaçant les mains où pieds là où l'on veut, sans se soucier des rotations individuelles des genoux, coudes où épaules : c'est l'IK qui s'en charge. Bien sûr l'IK n'est pas utilisé que pour des extrémités, le Spline IK est un des outils les plus utilisés pour la création des colonnes des personnages bipèdes. Ce système est aussi utile dans toute situation où le mouvement d'une chaîne de joints est naturellement déterminé par la fin de la chaîne.

En revanche, les transformations en FK doivent être privilégiées si le fait d'avoir des rotations des joints précises est important, comme par exemple, pour le mouvement des bras lors d'un cycle de marche. Dans ce cas, le fait d'animer les bras en FK facilite la gestion des retards des différentes parties des bras lors de la marche.

Ainsi, il n'existe pas un mode d'évaluation des rotations des joints « passe-partout ». C'est pour cette raison que même les rigs de bipède les plus simples donnent à l'animateur le choix entre ces deux modes de transformation : le choix de travailler en IK ou en FK appartient à l'animateur, et peut changer selon les mouvements requis sur chaque scène.



Les modes IK et FK sur le bras de la fille.

Pour travailler avec n'importe lequel des deux modes, le squelette doit être construit correctement, de façon à ce que chaque chaîne de joints se déforme de façon prévisible. Or, qu'est ce qu'un squelette bien construit ? Dans les sections qui suivent, on verra ce que c'est un axe de rotation locale, un concept de base dont la maîtrise nous aidera à construire des rigs qui ne causeront pas une perte massive de cheveux. Mais avant de passer aux axes de rotation locaux, on fera un petit détour pour expliquer les différents repères qui existent dans Maya pour effectuer les transformations.

Les espaces « World », « Object » et « Local »

Comme on vient de voir, les joints dans Maya, en plus de posséder des rotations et une position, ont aussi une orientation. Celle-ci est définie par la position des axes des rotations locales.

Pour comprendre l'utilité de ces axes, il faut savoir que dans Maya, les transformations d'un objet peuvent être effectuées en trois repères différents : Le « world space », le « object space », et le « local space ».

Le « world space » est l'espace de la scène de Maya, le repère le plus général. Par défaut, un objet qui n'a pas de parents dans la scène, est déformé par rapport à cet espace : les valeurs de translation qui s'affichent sont calculées par rapport à l'origine de la scène.

Le « object space », comme son nom l'indique, est l'espace relatif à un objet lui-même, à ses axes « internes », en quelque sorte. L'origine (ou point « zéro ») de cet espace est le point de pivot de l'objet.

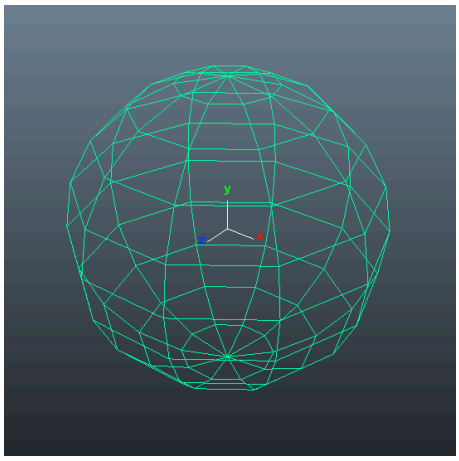
A la création, les axes de l'espace d'un objet quelconque, qu'on appellera « toto », sont alignés avec ceux de la scène, et les « world space » et « object space » de « toto » sont identiques. Si on effectue une rotation sur « Ballon », il emporte ses axes avec lui, déplaçant de cette manière, son « object space ». Chaque objet dans Maya possède ainsi son propre repère, qui change, par rapport à l'espace de la scène, à chaque fois qu'on effectue une rotation ou une translation sur lui. Les valeurs de translation d'un objet sont calculées par rapport à l'origine de la scène, et ses rotations peuvent être vues comme l'angle entre les axes du monde et les axes de l'objet lui même.
Si on parente « toto » sous un deuxième objet, « toto_parent », il sera maintenant dans un troisième espace, le « local space », où l'espace relatif à son parent. L'origine de cet espace se trouve au point

de pivot du parent. De cette manière, l'origine du « local space » de « toto » est le point de pivot de « toto_parent ». Notons que si un objet n'a aucun parent, son « parent » est la scène de Maya, et donc, les « world space » et « local space » sont identiques.

Avec une compréhension de ces espaces, passons enfin au cœur de la question, un des aspects les plus essentiels de la construction d'un BON squelette: les axes de rotation locaux.

Axes de rotation locaux

Dans Maya, la façon dont un objet agit quand on effectue sur lui des rotations en mode « local » où en « object space » est déterminée par l'orientation de ses axes de rotation locaux.



Les axes de rotation locaux d'une sphère.

La configuration de ces axes est particulièrement importante dans le cas des joints, car les rotations des joints sont par défaut calculées par rapport à ce repère. C'est pour cela que les joints sont les seuls objets dans Maya dont les axes de rotation locaux peuvent être orientés de façon arbitraire (the art of rigging).

Quand un joint est créé, les axes de rotation sont orientés automatiquement. Pour une chaîne de joints, par défaut, l'axe X de chaque joint sera orienté vers son enfant, l'axe Y s'approchant le plus possible de l'axe Y global. Si un joint ne possède pas d'enfants, ses axes de rotation seront alignés aux axes de la scène.

Cette configuration des axes n'est pas, dans la plupart des cas, la plus appropriée pour la création d'un squelette. Chaque chaîne de joints, selon sa fonction, a besoin d'une configuration particulière des axes. Les axes de rotation doivent être orientés d'une façon telle que les rotations soient cohérentes tout au long de la chaîne. Il est aussi important de s'assurer que les orientations des axes donnent lieu à des rotations qui correspondent à celles attendues de la part du modèle.

D'après Paul Thuriot, dans Hyper-Real : Body Setup :

« L'orientation des joints doit être ajustée d'une façon telle que, pour chaque joint, une rotation positive en Z entraine, dans la plupart des cas, un mouvement vers l'avant [dans le sens positif de l'axe Z], pour faciliter le décryptage des courbes d'animation dans le Graph Editor. De cette façon, on pourra lire le Graph Editor directement pour comprendre ce qui se passe dans l'animation, sans même avoir besoin de regarder la vue de la caméra » [2].

Prenons l'exemple des joints d'une jambe d'un bipède. Dans ce cas, les axes de rotation devraient être

orientés d'une telle façon que, en appliquant une rotation positive en Z, la jambe se déforme comme il est attendu de cette partie du corps, avec le talon qui s'approche de l'arrière de la cuisse.

L'importance du bon placement des axes de rotation locaux des joints est encore plus évidente dans le cas des doigts. Je vous laisse imaginer tout le travail supplémentaire qu'aurait à faire l'animateur pour former un poing avec une main formée des joints orientés n'importe comment. Alors que, avec une main où les axes de rotation des doigts ont été placés avec soin, il suffirait d'effectuer une simple rotation dans l'axe Z, ou autre, selon l'orientation choisie, sur tous les joints des doigts de manière simultanée.

Dans les modèles symétriques, il suffit d'ajuster les axes de rotation locaux d'un côté du modèle. Avec l'aide de l'outil Mirror Joints, l'option Mirror Behaviour cochée, le deuxième côté du squelette sera générée, et ses axes de seront alignés par Maya d'une façon telle que les rotations des contrôleurs des deux côtés du rig entraîneront une déformation symétrique du modèle.

Jusqu'à présent, j'ai privilégié le placement des axes comme le conseille Paut Thuriot, avec l'axe X qui pointe vers le joint suivant de la hiérarchie (ou comme axe de torsion ou twist comme il est souvent appelé dans la littérature), et l'axe Z comme axe principale de rotation [2]. Mais ce positionnement des axes dépend des préférences de chacun. On pourrait faire en sorte que ce soit Y l'axe de torsion, avec l'axe X comme axe de rotation principale, si on le voulait. Cela dit, ce choix doit rester cohérent tout au long du squelette, et il doit aussi être choisi en tenant en compte l'ordre de rotation que sera privilégié pour les joints.

L'Ordre de rotation

Comprendre le fonctionnement des ordres de rotation est l'une des choses les plus importantes avant de commencer à rigger. Malheureusement, c'est aussi un concept qu'on ne prend pas en compte, ou bien, qui est souvent mal compris.

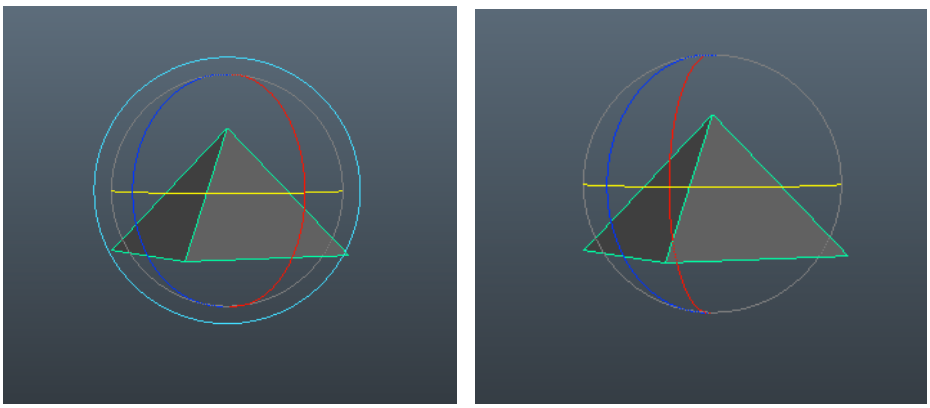
L'ordre de rotation s'applique à tout objet dans Maya, mais on l'abordera dans cette section, car il doit être décidé dès le début du processus de setup : le mouvement de chaque partie du corps doit être analysé, surtout pour les articulations qui auront des rotations dans plusieurs axes, comme pour les épaules et les poignées, et l'ordre de rotation des contrôleurs doit correspondre à celui du joint

qu'ils contrôlent. Mais, ça veut dire quoi « ordre de rotation » ? Pourquoi est-ce que ce concept est aussi important ?

Dans un environnement 3D, les rotations peuvent être évaluées de deux façons différentes : la méthode Euler, et la méthode par quaternions. Les quaternions sont des objets de quatre dimensions dérivés des nombres complexes. Dans le mode de rotation par quaternions, des courbes X, Y, et Z sont utilisés par Maya afin de trouver une quatrième courbe, W, qui servira pour trouver la rotation en quaternion d'un objet, comme il est indiqué dans l'aide de Maya.

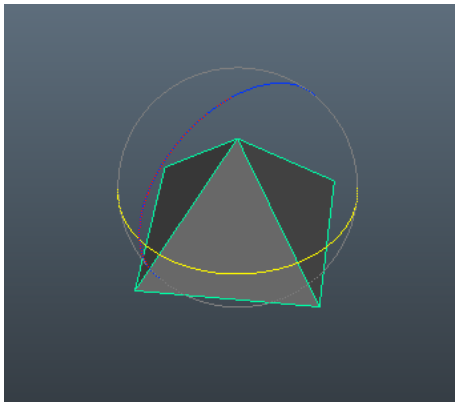
La méthode Euler, celle utilisé par défaut, l'orientation d'un objet est définie par trois valeurs de rotation, en X, Y, et Z, en plus d'un ordre de rotation. Cet ordre de rotation est celui qui définit la façon dont les rotations seront évaluées : si on applique une rotation à un objet dont l'ordre de rotation est XYZ, l'ordre de rotation par défaut, Maya calculera d'abord la rotation autour de l'axe X, ensuite celle autour de l'axe Y, et enfin la rotation correspondante à l'axe Z. L'ordre utilisé par Maya est déterminant pour savoir quelle sera l'orientation finale de l'objet.

En mode rotation, on se sert des trois anneaux colorés comme les axes X, Y, et Z pour effectuer des rotations dans les axes correspondants. Dans la plupart des cas, les rotations sont affichées en mode « Local » ou « World ». Le problème est que dans aucun de ces deux modes, on ne voit la façon réelle dont sont évaluées les rotations. On peut orienter tous les objets dans tous les sens, sans se douter des problèmes qui peuvent être cachés derrière ses axes qui se comportent aussi bien··· Quels problèmes ? Derrière cette conduite tellement sage des axes de rotation se cache la façon réelle de Maya de calculer les rotations en mode Euler, et avec elle, l'atroce Gimbal Lock ! Pour voir ce qui se passe vraiment pendant les rotations, il faut passer en mode « Gimbal », dans les options de l'outil de rotation.



Une même rotation vue en mode "World" et "Gimbal".

Ce qu'on dénomme Gimbal Lock, la peur de tous les animateurs, qui peut causer des interpolations insensées entre deux clés d'animation parfaitement correctes, est la situation dans laquelle deux axes de rotation se trouvent superposés. Avec un ordre de rotation de XYZ, l'axe Z « emporte » avec lui les axes Y et X, et, à son tour, l'axe Y emporte l'axe X avec lui. Les rotations faites autour de l'axe X n'affectent aucun autre axe. De cette façon, il suffit d'effectuer une rotation de 90° dans l'axe Y pour que les axes Y et Z soient alignés, car, par l'ordre de rotation désigné, l'axe Y emporte l'axe X avec lui :



Le Gimbal Lock!

La conséquence de cette situation est la perte d'un axe de rotation, d'un dégrée de contrôle. Dans les modes « World » et « Local », ceci ne se voit jamais. Maya modifie les rotations de tous les axes pour arriver au résultat voulu par l'utilisateur. Ceci n'est pas très grave, car en fin de comptes, on peut toujours aller sans problème d'une pose à l'autre, n'est-ce pas ? FAUX! Poser un personnage sans se soucier du Gimbal Lock peut engendrer des résultats pour le moins surprenants, comme une main qui fait trois fois le tour sur elle-même entre deux clés d'animation… Et je vous assure, ceci n'est pas très agréable à régler.

La bonne nouvelle, c'est qu'en choisissant des ordres de rotation adaptés pour les joints et les contrôleurs pour les différentes parties du rig, on peut éloigner la menace de l'ignoble Gimbal Lock. L'ordre de rotation choisi doit être tel que la rotation la plus importante, celle qui sera le plus utilisé pour un joint ou contrôleur particulier, soit la dernière à être évaluée. De cette façon elle ne sera pas affectée par les rotations des autres axes. En plus, comme ce dernier axe de rotation « emportera » avec lui les deux autres, on diminue ainsi le risque d'avoir des axes superposés.

Pour choisir l'ordre de rotation le plus adapté pour un objet, le mieux est de prendre le temps de tester des configurations différentes, en tenant en compte le comportement que l'on veut. Pour les joints, ceci dépendra fortement de la façon dont on aura orienté ses axes de rotation. Par exemple,

pour le joint de l'épaule d'un personnage, on privilégiera un ordre de rotation de XYZ. De cette façon, l'axe Z servira à lever et baisser le bras, l'axe Y sera celui qu'on utilisera pour bouger le bras vers l'avant et l'arrière, et l'axe X servira à orienter le bras autour du torse.

En tout cas, le plus important, c'est de toujours prendre le temps de choisir le bon ordre de rotation, car, comme le dit Jason Schleifer (et je le reprends en texte gras, comme il est écrit dans son livre) :

« Choisir le bon ordre de rotation pour vos contrôleurs est l'une des choses les plus importantes que vous pouvez faire pour votre rig. » [1]

Il conseille (fortement) aussi de favoriser un ordre de rotation de XZY ou ZXY pour tout contrôleur transformé en « world space », car de cette façon ces objets pourront être orientés dans le monde plus facilement. [1]

Considérations additionnelles

En prenant le temps de bien orienter les axes de rotation locaux, et en choisissant avec soin les ordres de rotation, on commence à avoir un squelette bien construit. Or, il y a encore quelques éléments à prendre en compte pour que le squelette soit vraiment « propre ».

Pour ces derniers détails, je laisse la parole à Paul Thuriot :

« Tous les joints doivent avoir des valeurs égales à zéro pour les translations en Y et Z, aussi bien que pour tous les axes de rotation. Les valeurs du Scale doivent être égales à 1. La valeur du Translate X correspondra à la longueur du l'os. Pendant le placement d'un joint, seulement les translations et les valeurs de jointOrientXYZ doivent être modifiées. Seulement le premier joint de chaque chaîne peut avoir des valeurs de translation différentes à celles décrites plus haut. » [2]

Il ajoute aussi que le dernier joint de chaque chaîne doit avoir une orientation de zéro, de cette façon son orientation sera égale à celle de son parent.

Enfin, tout joint doit être nommé pour qu'il soit facilement identifiable par la suite.
Pour plus de détails sur la construction d'un squelette, et sur comment obtenir un squelette « propre », je conseille vivement Hyper-Real Body Setup, de Paul Thuriot [2].

Contraintes et autres contrôles d'attributs

Dans un rig, le principe de base c'est de contrôler les attributs de certains objets à traves d'autres objets. Pour contrôler la rotation d'un joint, par exemple, on associe la rotation d'un contrôleur à la rotation de ce joint là, pour ne pas être obligés de sélectionner le joint à chaque fois qu'on voudra le transformer. Dans Maya, il existe plusieurs façons de lier les attributs des objets différents. Dans les lignes qui suivent, on fera un petit tour par ces différents modes con contrôle d'attributs.

Les connexions directes

La connexion directe, comme son nom l'indique, c'est le fait de lier directement deux attributs. Ceci est fait principalement en utilisant le Connection Editor. Dans cet éditeur, on peut associer directement deux attributs du même type, de deux objets quelconques (ou bien deux attributs d'un même objet). Par exemple, on peut connecter le rotateX d'une sphère à la translation en Y d'un cône. Avec cette configuration, la valeur de la translation en Y du cône correspondra exactement à celui de la rotation en X de la sphère.

Ce mode de contrôle est très rapide à évaluer, car il ne requiert pas de nodes intermédiaires. Il est à privilégier à chaque fois que l'on voudra faire correspondre exactement la valeur de l'attribut contrôlé à celle de l'attribut qui le contrôle, et ceci de manière permanente.

Les contraintes

Un autre moyen de contrôler les attributs est avec l'aide des contraintes. Il y a plusieurs différences entre cette méthode et celle que l'on vient de décrire. Dans un premier temps, à différence des connections directes, toutes les contraintes peuvent être activées où désactivées sans l'utilisation d'autres nodes, ce qui les rend très utiles dans les rigs qui disposent de plusieurs modes de contrôle. Aussi, avec les contraintes, on peut décider si les transformations des deux objets seront identiques, et dans ce cas, au moment de la création de la contrainte, les attributs concernés de l'objet contrôlé seront modifiés pour qu'ils correspondent exactement à ceux de l'objet contrôleur, ou bien si la correspondance entre les attributs des deux objets sera prise en compte à partir de leurs valeurs actuelles. Ce choix est fait à la création de la contrainte avec l'option « Maintain Offset ».

Un autre avantage de ce type de contrôle d'attributs est qu'un objet peut être contrôlé par plusieurs objets à la fois, et de la même façon qu'une contrainte peut être désactivée, on peut aussi changer la

valeur qui contrôle l'importance de l'influence de chacun des objets qui contrôlent un autre.

La façon dont un objet contrôlé répondra aux transformations de l'objet (ou les objets) qui le contrôlent dépend du type de contrainte appliquée. Dans tous les cas, les attributs contrôlés par une contrainte sont bloqués, et ne pourront plus être modifiés directement.

On verra à continuation les principaux types de contrainte disponibles dans Maya.

Le « Point Constraint »

Le « Point constraint » lie les valeurs de la translation d'un objet à celles d'un autre. Si l'objet contrôleur est déplacé, l'objet contrôlé suivra ses déplacements. Ce type de contrainte est souvent utilisé pour les contrôleurs liés aux IK Handles.

L' « Orient Constraint »

Ce type de contrainte contrôle les valeurs de rotation d'un objet. Avec cette contrainte, toute rotation appliquée sur l'objet qui contrôle, est suivie par une rotation sur l'objet contrôlé, en « local space ». Ce type de contrainte peut être utilisé sur les contrôleurs de joins en mode FK, ou dans toute autre situation où l'on veut contrôler la rotation d'un objet.

Le « Parent Constraint »

Avec ce type de contrainte, on peut simuler un lien hiérarchique entre deux objets, sans qu'ils soient vraiment liés dans l'Outliner. Ce type de contrainte est très utile quand on veut un comportement de hiérarchie sans modifier la hiérarchie de la scène. En principe, un objet lié à un autre par un « Parent Constraint » se comporte exactement comme si l'objet contrôleur était son parent, sauf que le « Parent Constraint », comme toutes les autres contraintes, peut être désactivé à volonté. Un autre avantage de ce type de contrôle par rapport au parentage direct est qu'un objet peut être contrôlé par plusieurs objets à la fois. Le « ParentConstraint » est très utilisé dans le processus de rigging, aussi bien pour garder une scène propre, comme pour gérer les situations qui nécessitent un comportement similaire à celui engendré par un lien de parenté, mais que doit être désactivé selon les besoins de l'animation, comme par exemple, pour les interactions des personnages avec différents objets de la scène (props).

Le « Aim Constraint »

Le « Aim Constraint » contrôle les valeurs de rotation d'un objet, prises en compte autour de son axe locale, à la position d'un autre objet, de façon à ce que l'objet contrôle « pointe » vers l'objet qui le contrôle. Ce type de contrainte est souvent utilisé pour gérer le comportement des yeux des personnages en animation.

Le « Pole Vector Constraint »

Ce type de contrainte est destiné aux IK Handles de type « RP ». Avec cette contrainte, on peut designer un objet avec lequel on contrôlera l'orientation de l'IK, ou, plus précisément, l'orientation du plan sur lequel seront positionnés les joints contrôles par l'IK.

L'outil « Set Driven key »

Les « Driven Keys » sont encore un autre moyen de contrôler les attributs d'un objet avec les attributs d'un autre. La particularité de ce type de contrôle es que, pour les utiliser, il n'est pas nécessaire que les attributs à lier soient du même type : On peut associer tout et n'importe quoi, et de la façon que l'on veut. Avec les contraintes, on avait un peu plus de choix qu'avec les connections directes, mais la relation entre les attributs de deux objets liés restait directe : si l'attribut contrôleur augmente, l'attribut contrôlé le fera aussi. Avec l'aide des « Driven Keys », ou SDKs, comme ils sont souvent appelés, les relations entre les attributs de deux objets peuvent être beaucoup plus complexes.

Les SDKs sont des relations entre objets définies par des courbes d'animation. Partant de la valeur d'un attribut qu'on appelle « Driver », ou celui que contrôle, on lui associe une valeur de l'attribut contrôlé, ou « Driven », créant ensuite une clé d'animation, qui fixe ce comportement. On peut créer autant de clés que l'on veut. Les valeurs entre deux clés de ce type seront interpolées comme pour n'importe quelle courbe d'animation. Les courbes créées par un SDK, comme toute autre courbe d'animation dans Maya, sont visibles dans le graph editor. On peut ainsi modifier les valeurs et les positions des clés, aussi bien que la forma de la courbe.

Ce type de contrôle d'attribut est utile lors du rigging des mains ou des pieds. Dans ce deux cas, on crée souvent des attributs personnalisés, et les « Driven Keys » sont un très bon moyen de déterminer la façon dont ces parties du corps répondront aux modifications faites sur les nouveaux attributs.

Les Expressions

Avec l'usage des expressions, on peut lier deux attributs par une relation mathématique. Une expression peut être aussi simple que la ligne suivante (qui aurait, par ailleurs, le même comportement qu'une connexion directe) :

```
sphere.rotateX = cone.translateY
```

Ou bien être un mini programme, avec des commandes plus complexes.

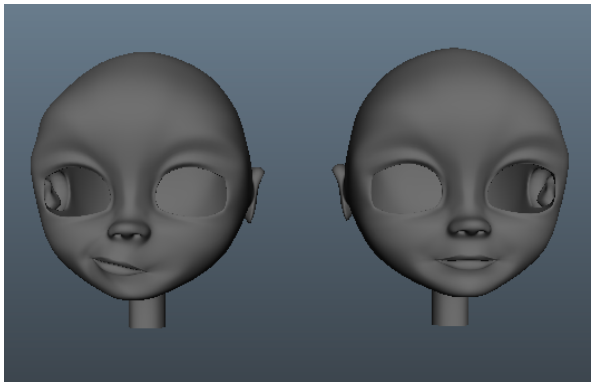
Les expressions peuvent être utilisées pour automatiser des comportements compliqués. Dans Animation Friendly Rigging, Jason Schleifer crée un système de squash and stretch avec l'aide des expressions [1].

Les « Deformers »

Dans Maya, il existe des outils appelés « Deformers », ou déformeurs, qui peuvent être utilisés pendant la modélisation, ou bien comme partie d'un rig, pour contrôler un objet lors de l'animation. On verra dans cette section certains des déformeurs les plus utiles lors du processus de rigging.

Le « Blend Shape Deformer »

Avec l'aide des Blend Shapes, on peut transformer un objet, appelé « base object » en autre objet, appelé « target object», pourvu qu'i ait le même nombre de vertex, organisés dans le même ordre, utilisant un slider pour contrôler la mesure dont la géométrie est déformée. Une fois le Blend Shape appliqué à un objet, Maya prendra en compte les différences de position, vertex par vertex, et appliquera progressivement les translations correspondantes pour transformer le « base object » de façon à ce qu'il corresponde au « target ».



"Target" et "Base" pour un des Blend Shapes de la fille.

Un même objet peut avoir plusieurs « targets ». Dans ce cas, la fenêtre qui affiche les Blend Shapes comportera un slider par « target ».

Les Blend Shapes sont typiquement utilisés pour créer les expressions du visage d'un personnage, ou pour corriger des problèmes ponctuels d'un skin.

Les « Cluster Deformers »

Les Clusters sont un type de déformeur souvent utilisé dans le processus de rigging. Ce déformeur en particulier permet d'avoir un contrôle sur les points d'un objet, comme par exemple, les vertex d'un polygone, ou bien les CVs d'une courbe.

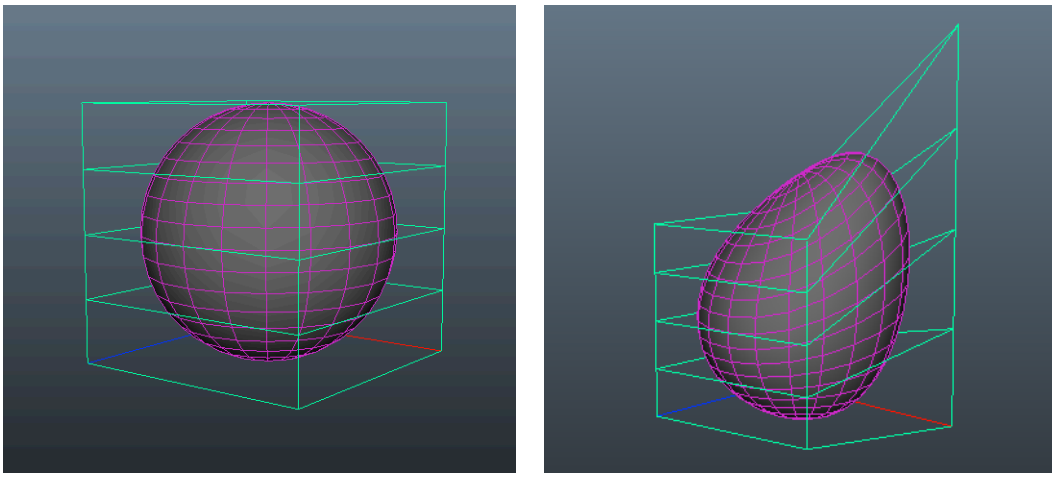
Ceci est particulièrement utile lors que des IK Handles du type Spline sont utilisés, car ils ne peuvent pas être modifiés directement, mais par intermédiaire des CVs de la courbe qui les contrôle. Or,

devoir passer en mode « Component » à chaque fois que l'on veut transformer un IK est loin d'être pratique.

On crée, alors, un Cluster par CV (ou par paire de CVs, selon le comportement voulu), et on associe ensuite un contrôleur plus accessible à chaque cluster pour pouvoir modifier facilement la forme de l'IK.

Les « Lattice Deformer » et « Wrap Deformer »

Un lattice est une « structure de points », une sorte de cage qui entoure une géométrie, et qui contrôlera sa forme. Quand les points du lattice sont déplacés, la forme de l'objet que le lattice déforme s'adapte à la nouvelle forme de celui ci:



Un exemple de "Lattice"

Les lattices peuvent être utilisés lors qu'on veut modifier la forme d'un objet de manière plus organique, et sont souvent associés aux « Non-Linear Sculpt Deformers », qui permettent de créer certaines déformations facilement.

Les Wrap Deformers peuvent être vus comme un type particulier de lattice, où, en lieu d'utiliser les points d'une « cage » pour déformer un objet, une surface, polygonale ou NURBS, est utilisé. Ce type de déformeur s'utilise beaucoup lors de la création des Blend Shapes des modèles symétriques.

Bonnes Pratiques

Après avoir vu quelques concepts de base nécessaires avant de commencer à rigger, passons à présent à quelques automatismes qui peuvent être d'une grande utilité pendant la création d'un rig.

Placement par « parent » ou « point constraint »

Un peu plus haut, on a vu que dans Maya, on a le choix entre plusieurs repères pour transformer les objets. Ceci est particulièrement important pour les contrôles qui gèrent les rotations de joints lors de l'animation en mode FK. Les contrôleurs doivent avoir la même orientation, et bien sûr le même ordre de rotation, que les joint qu'ils dirigent, de façon à avoir un comportement consistent.

De cette façon, en mode « Object », on pourra effectuer des rotations sur les contrôleurs de la même façon que si on le faisait sur le joint même. Si on ne fait pas attention à bien placer les contrôleurs par-dessus les joints, tous les efforts faits pendant le placement des joints n'auront servi à rien.

Il y a plusieurs façons de placer les contrôleurs exactement au même endroit que les objets qu'ils contrôlent. Ici, je vais vous présenter ma méthode préférée : le placement par moyen un « parent » ou « point constraint ».

Cette méthode consiste à créer une contrainte de type « parent » ou « point », de façon à ce que l'objet que l'on veut contrôler dirige le contrôleur. Dans les options de la création de la contrainte, il faut impérativement que l'option « maintain offset » soit décochée.

Une fois la contrainte appliquée, le contrôleur s'attachera à l'objet. Selon le type de contrainte utilisé, « point » ou « parent », le contrôleur aura non seulement acquis la position de l'objet, mais aussi ses rotations. Ensuite, on efface la contrainte créée, qui apparaît dans l'Outliner directement sous le contrôleur.

Une autre façon de reprendre les transformations de l'objet dans le contrôleur est de « parenter » le contrôleur directement sous l'objet qu'on veut contrôler, et remettre toutes ses transformations à zéro. Le contrôleur sera alors à l'origine de son « Local Space », le point de pivot de son parent. Ensuite il faudrait seulement le « déparenter ». Il passe maintenant au « World Space », où il aura exactement les transformations de son ancien parent.

J'aime particulièrement la méthode des contraintes car elle peut être facilement codée (et ce en deux lignes), ce qui représente un gain de temps, et aussi, comme pour toutes les actions répétitives, réduit la marge d'erreur dans ce genre de manipulations.

« Offset groups »

Une autre bonne habitude à prendre, c'est l'usage de « offset groups » pour tous les contrôleurs. Quand on parle de « offset group », on fait référence à un groupe qui contient un contrôleur, crée de façon à ce que le contrôleur garde toutes ses transformations à zéro, une fois placé sur l'objet qu'il contrôle. Ceci facilite le processus d'animation, car il est décidément plus facile de remettre un contrôleur à sa position original si elle est « 0, 0 ,0 » que « 25.78, 65.986, 89.6209 ».

Les plus sceptiques d'entre vous me diront : « Mais, on n'a pas besoin de groupes ! Il suffit de faire un Freeze Transformations sur chaque contrôleur et le tour est joué. Transformations à zéro ! ». Malheureusement, ceci est vrai. Trop vrai. La commande Freeze Transformations ne fait pas que remettre toutes les transformations à leurs valeurs par défaut. Elle a un méchant effet secondaire : lors qu'on « freeze » un objet, ses axes de rotation locaux sont alignés automatiquement à ceux de la scène.

Donc, si on avait placé le contrôleur soigneusement pour qu'il soit parfaitement aligné à l'objet qu'il contrôle, en faisant un Freeze Transformations tout ce travail est parti directement à la poubelle.

Par contre, si à la création d'un contrôleur on le groupe systématiquement, et on applique le « parent » ou « point constraint » de placement, comme dans la section précédente, sur le groupe en lieu que sur le contrôleur directement, celui-ci gardera ses transformations à zéro, puisque celui-ci n'a pas été déplace dans son « local space », l'espace du groupe . Tout le « bruit », dû au déplacement sera contenu dans les transformations du groupe, pas dans celles du contrôleur. En plus, ses axes resteront alignés à l'objet que l'on a choisi.

La création de ce genre de groupes à un deuxième avantage : la possibilité de continuer à animer un objet qui a sur lui une contrainte. Dans la section des contraintes on a vu que lorsqu'un objet est contrôlé par une contrainte, les attributs concernés sont bloqués, et ne peuvent plus être modifiés.

Avec les « offset groups », on crée une nouvelle couche de contrôle : en lieu d'appliquer les contraintes directement sur le contrôleur lui-même, on le fera sur son groupe. De cette façon, le

contrôleur héritera tous les caractéristiques de l'objet qui le contrôle, mais il restera disponible pour être animé indépendamment.

Dans le tutoriel de Gnomon, Creature Rigging : The puppet rig este mancito va encore plus loin : En lieu de créer un seul « offset group » par-dessus chaque contrôleur, il en crée deux, le deuxième étant dédié aux « Set Driven Keys ». De cette façon, chaque contrôleur peut non seulement avoir des contraintes sur lui et rester « animable », mais il peut aussi avoir des SDKs [6].

Ceci peut sembler un peut excessif, mais cette création régulière de deux « offset groups » par dessus chaque contrôleur peut être très utile sur des systèmes un peu complexes, comme on le verra plus tard sur les rigs du court « L'Armoire ».

Renommage correct

Un autre aspect important à garder en tête lors de la création d'un rig est de renommer systématiquement tout ce que l'on crée : joints, groupes, nodes, courbes, clusters···. Tout ! Le fait d'avoir des objets bien nommés dans un rig facilite le débugage, l'identification des différentes parties du rig, et la sélection par l'usage du langage MEL.

En plus de prendre le temps de tout renommer, il faut aussi être cohérent dans la façon de procéder, et garder une nomenclature cohérente tout au long du rig. Par exemple, si on décide que les éléments du côté gauche du rig porteront le suffixe « Lf » et on fait ainsi pour tous les contrôleurs et joints de la main, on ne doit pas soudainement commencer à nommer les joints de la jambe gauche en utilisant le suffixe « _L ». Une fois un système de noms choisi, il faut le garder jusqu'à la fin.

En parlant des suffixes, je dois dire qu'il s'agit encore d'un des reflexes qui peuvent aider à faire que le processus de rigging soit plus « friendly ». Même pour les rigs qui ne sont que moyennement complexes, on pourrait dire que l'usage de suffixes est obligé. On devrait automatiquement nommer les objets qui font partie de la géométrie en ajoutant le suffixe « _Geom », les IK Handles avec le suffixe « _IK », et les nodes multiplyDivide avec le suffixe « _md », pour donner quelques exemples. Evidemment, on peut choisir les suffixes que l'on veut, pourvu que les suffixes soient facilement identifiables, et restent cohérents sur tous les éléments qui comportent le rig.

Le grand nettoyage

Lors qu'on commence à travailler sur les derniers détails d'un rig, il faut penser à la façon dont la scène sera présentée, et à certains détails qui n'ont pas forcément un rapport direct avec le fonctionnement d'un rig, mais plutôt avec la façon dont il sera perçu par la personne qui s'en servira.

Dans un premier temps, avant de passer une scène à un animateur, tout ce qui ne lui servira pas doit être bloqué et caché. En particulier, tous les objets que peuvent causer la « casse » du rig, doivent être spécialement protégés. Selon Jason Schleifer, on doit considérer que l'animateur est comme un enfant, qui touchera à tout ce qu'il pourra, et, si un rig a des faiblesses, l'animateur ne tardera pas à les trouver. Il écrit

« Ne les laissez pas [les animateurs] toucher aux choses que vous ne voulez pas qu'ils touchent. S'ils peuvent saisir quelque chose, ils la déplaceront. Faites semblant d'être leur parent, et cachez tout ce qui peut leur faire du mal. » [1]

Jason Schleifer prend un soin particulier pour que son rig soit virtuellement « incassable ». Je conseille vivement la lecture de la dernière partie de ce livre si l'on veut se faire une idée claire de ce que c'est un rig « à l'épreuve des animateurs ».

Enfin, je voudrais m'attarder sur l'un des aspects les plus importants pour un rig : toute scène comportant un rig fini doit avoir un Outliner bien organisé. Mais, ça veut dire quoi, donc, un Outliner bien organisé ?

Pour commencer, l'Outliner doit comporter deux groupes : Un contenant le rig, et un autre pour la géométrie. Le groupe qui héberge la géométrie ne doit jamais être transformé, et, pour cette raison, toutes les transformations doivent être cachées et verrouillées.

Ceci est aussi vrai pour le groupe contenant le rig : dans aucun cas il ne doit être transformé directement, car on risquerait d'engendrer des déformations indésirables. Dans ce groupe on aura alors, plusieurs autres sous-groupes. On commence avec un groupe contenant tous les joints qui affectent le skin. Ce groupe aussi aura ses transformations bloquées, car ces joints seront transformés par l'intermédiaire des contrôleurs, et ses options de transformation ne doivent donc pas être disponibles pour l'animateur.

Ensuite, toute la hiérarchie de contrôleurs doit être placée dans un autre groupe, c'est-à-dire, tous les contrôleurs qui sont directement transformés par le contrôle global, celui qui sera utilisé pour placer le personnage dans la scène.

Mais, attention, dans ce groupe on placera seulement les contrôleurs qui n'héritent des transformations d'aucun autre contrôleur. Je m'explique : Si un contrôleur est dirigé par un autre par moyen d'une contrainte, expression ou autre, il ne doit pas être placé dans ce groupe car, encore, on aurait des problèmes de double transformation. Ce groupe doit pouvoir être transformé normalement, donc seulement son attribut Scale sera bloqué et caché.

En dernier lieu, on aura un groupe contenant tous les éléments du rig qui ne doivent pas être transformés, ou bien qui héritent déjà une transformation d'un autre élément du rig. On y placera par exemple les IK Handles, car normalement, ils sont dirigés par un contrôleur à travers un « point constraint ». Selon la complexité du rig, ce groupe contiendra beaucoup d'éléments. Il doit donc, à son tour, être bien organisé pour faciliter le débugage, du rig en cas de problème. En ce qui concerne ce dernier groupe, non seulement on bloquera toutes ses transformations, mais aussi on lui rajoutera le suffixe « _noTrans », pour qu'il soit bien clair qu'on ne doit, dans AUCUN cas, y toucher.

Bien sûr, il y a d'autres façons d'organiser une scène, et d'autres références d'organisation peuvent être trouvées dans [1], [2], et [7], par exemple.

Tous ces concepts deviendront plus clairs par la suite, lorsqu'on analysera en détail les rigs réalisés dans le cadre du court métrage « L'Armoire ».

Les rigs du court « L'Armoire »

« L'Armoire », court métrage réalisé avec Marjorie Vallinas comme projet de fin d'études, est l'histoire d'une petite fille qui a un invité très particulier qui loge dans son grenier. Sur ce projet, je me suis occupé principalement du développement des personnages. J'ai me suis chargée en particulier de réaliser un rig pour chacun d'entre eux.

Dans ce chapitre, on procédera à l'analyse des rigs créés pour les deux personnages du court « L'Armoire ». Pour ce faire, on traitera séparément les différentes parties du corps : le torse, la tête, les bras et les mains, et, enfin, les jambes et les pieds. Les rigs du visage auront droit à leur propre section un peu plus tard.

On commencera par la création du squelette, divisée elle aussi en sections pour attaquer chaque partie du corps à son tour. Ensuite, nous passerons à la description des rigs de chacune des parties du corps de chacun des personnages. Nous examinerons ensemble les rigs réalisés sous la lumière des principes expliqués plus haut. Mais avant la création de tout rig il faut passer par une étape très importante: la préparation.

Dans les lignes qui suivent, je vous présenterai l'histoire et les personnages du court, pour ensuite analyser son storyboard, afin de trouver le type de rig le plus adapté aux besoins de l'animation, avant de procéder à la création des squelettes et des rigs.

Présentation

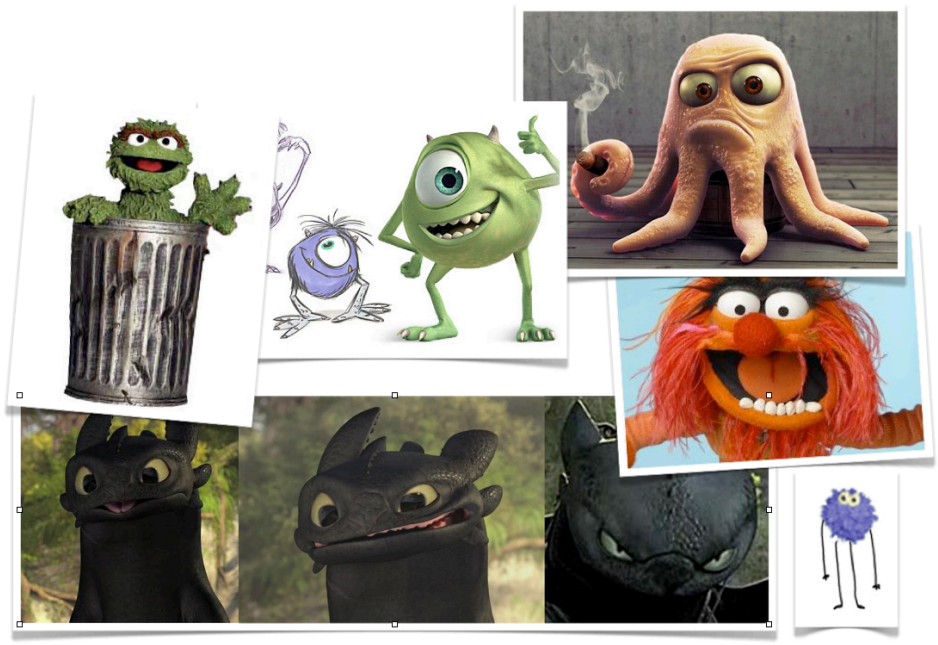
Dans « L'Armoire » on découvre Clémentine, une petite fille d'environ 9 ans, qui habite dans une maison de la banlieue d'une grande ville. Dans son grenier, au fond d'une vielle armoire, se cache un ami très spécial : un petit monstre poilu et un peu grognon. Cette petite créature demande à Clémentine de lui apporter des objets qu'elle cherche ensuite dans la rue. Mais sans un vrai moyen de communiquer, et malgré les efforts de Clémentine, cette situation donne lieu à plusieurs échecs, avant qu'elle ne réussisse enfin à trouver l'objet tant convoité par son ami violet.

Personnages

Les personnages impliqués dans ce court sont la petite fille, Clémentine, le monstre et ses 3 amis.
Pour la création de la fille, l'idée était de trouver un personnage à la fois attachant et un peu maladroit.

Les références pour la création de la petite fille.

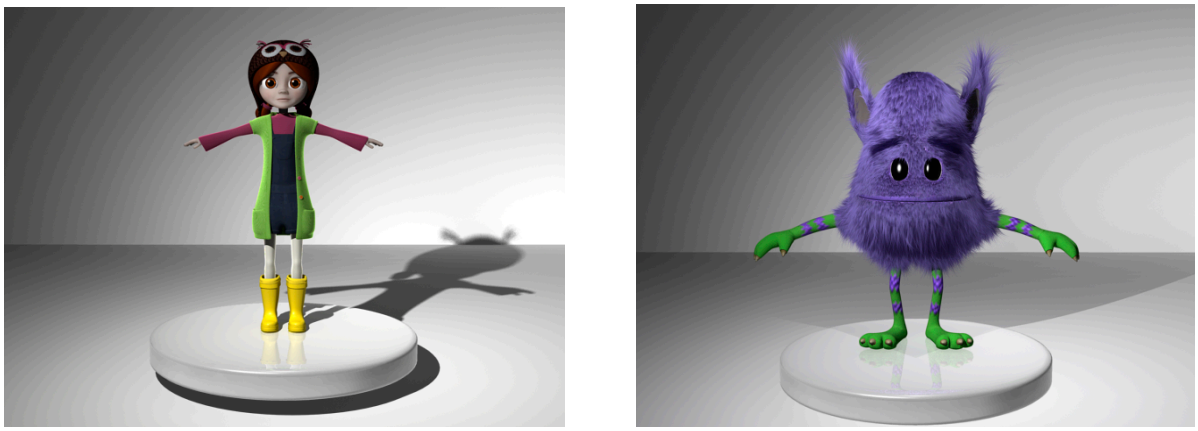
Pour le monstre, nous voulions un personnage assez mignon, mais qui permettrait aussi de montrer un aspect un peu grognon. Pour ce faire, nous avons cherché beaucoup d'images de monstres d'histoires pour enfants, en particulier des images des Muppets.



Les références pour la création du monstre.

Nous voulions un personnage assez expressif, da façon à ce que ses émotions soient facilement identifiables.

En nous inspirant de ces exemples, et après beaucoup de discussion, nous sommes arrivées aux modèles définitifs de nos personnages :



Les modèles définitifs de la fille et le monstre.

Les autres monstres.

Les amis du monstre seraient des variations en forme et en couleur du monstre original:

Une fois le modelling des personnages fini, il était temps de passer à la création de rigs.

La préparation

Comme on a vu dans le chapitre précédent, avant de commencer un rig, il faut passer par une étape de préparation assez importante. Les « edge loops » du modèle doivent être pris en considération, pour commencer à se faire une idée de comment les personnages pourront être déformés.

Dans les mots de Paul Thuriot, avant de commencer un rig il faut se poser les questions suivantes :

« A quoi ressemble le personnage ? Qu'est ce qu'il doit faire dans la production ? Est-ce que divers systèmes de rig différents seront nécessaires ? Tout ce que le personnage doit faire de spéciale sur un plan spécifique ou qui peut potentiellement causer des problèmes ou qui requiert un traitement spéciale nécessite une attention particulière. Quels sont les préférences de l'animateur pour ce rig ? » [2]

Jason Schleifer, dans [1] est un peu plus spécifique. D'après lui on doit regarder chaque plan du story en se demandant si les bras du personnage auront besoin de s'étirer, la façon dont les jambes bougeront, et quels déformations seront nécessaires pour le visage, par exemple. Il touche aussi un point très important : le temps que l'on a disponible pour développer le rig.

En tant que rigger, on voudra toujours rajouter les contrôles le plus sophistiqués, même s'ils ne sont pas nécessaires, ça fait partie du métier. Mais ceci est rarement faisable. On doit donc, commencer par ce qui est strictement nécessaire, et rajouter les contrôles les plus complexes, ou les moins nécessaires seulement après avoir réussi à avoir un rig fonctionnel adapté pour la production.

On doit aussi prendre en considération les vœux de l'animateur. Un rigger doit toujours être à l'écoute de l'animateur. Après tout, c'est lui qui devra travailler avec le rig, et sa performance dépendra énormément de la facilité qu'il aura pour comprendre et manipuler les contrôleurs qui sont à sa disposition. Un rig difficile à comprendre aura comme seul résultat un animateur mécontent, et probablement une animation pas terrible.

Egalement, il faut garder en tête les objectifs artistiques de l'animation, et penser à résoudre les problèmes de la façon la plus logique et simple. La création d'un rig ne doit pas être vue comme l'opportunité de montrer tout notre talent en tant que riggers, mais comme l'opportunité de créer une marionnette efficace et agréable à manœuvrer, qui aidera à l'animateur à accomplir tous ce qui est requis par le directeur, sans sacrifier sa performance et sa liberté créative.

Avant donc, de commencer les rigs pour les personnages du court « L'Armoire », il a fallu passer par une analyse détaillé du storyboard.

Après cette analyse, on avait déjà une petite idée des actions qui devaient réaliser les personnages. Tous les deux devaient être capables de marcher et de ramasser des objets, et ils devaient aussi avoir un rig assez complet du visage, car, étant donné que les personnages ne parlent pas dans le court, toute la communication devait être faite à niveau des expressions faciales.

En ce qui concerne le monstre, je voulais avoir la possibilité de déformer ses bras comme dans certains dessins animés, un type de rig qui se dénomme souvent « bendy arm ». Aussi, je voulais que pour la plupart son torse se comporte comme un seul bloc, ne se déformant que dans le cas de l'ouverture de la bouche. Enfin, je voulais pouvoir orienter ses oreilles des différentes façons, pour que cette partie du corps puisse accentuer l'expression générale du personnage.

Les autres monstres auraient un rig similaire à celui du monstre principal. Il était donc important de penser à une façon de réutiliser tous les éléments du rig du premier monstre.

Gardant toutes ces idées dans l'esprit, passons au design et à la construction des squelettes et des rigs.

Design du rig

Après avoir défini les aspects les plus généraux qui doivent comporter les rigs de ce court, en ce qui concerne la partie fonctionnelle, passons à présent à la partie formelle : Comment doivent être les contrôleurs du rig ?

Suivant les idées exprimées par Jason Schleifer, les contrôleurs d'un rig doivent remplir les conditions suivantes :

• Les contrôleurs doivent être simples et uniques

Le rôle de chacun des contrôleurs doit être immédiatement compréhensible, et leur fonction doit être simple, limité uniquement aux interactions nécessaires. Le nombre total de contrôleurs présents dans le rig doit être limité, et deux contrôleurs séparés ne doivent jamais servir pour la même manipulation. Comme règle générale, l'animateur doit avoir suffisamment de contrôle à sa disposition sans être submergé par la quantité d'options disponibles.

• La fonction de chaque contrôleur doit être clairement indiquée par sa forme[1].

Il rajoute également que tous les contrôleurs doivent être facilement lisibles seulement par leur forme. Par exemple, un contrôleur qui gère une translation doit avoir une forme de flèche, indiquant les directions possibles du mouvement. Ceci peut, cependant, devenir un peu chaotique en termes de la simplicité visuelle de la scène.

En plus d'être lisibles, les contrôleurs doivent aussi être faciles à sélectionner : le fait de devoir changer de vue de caméra pour pouvoir sélectionner un contrôleur devient vite fastidieux.

• L'ordre de rotation des contrôleurs doit être juste.

Comme on a vu dans le chapitre précédent, les ordres de rotation doivent être choisis avec soin, pour qu'ils soient les plus appropriés non seulement pour le bon fonctionnement de chaque contrôleur particulier, mais aussi pour les préférences de l'animateur.

• Les contrôleurs doivent avoir des noms adéquats.

Admettons que vous avez réussi à finir un rig sans avoir rien renommé. Peut être que vous avez eu la chance de pouvoir travailler comme ça sans perdre la tête. Cela dit, ce ne sera certainement pas le cas pour la personne qui travaillera avec votre rig : imaginez l'enfer pour un animateur qui, voulant trouver le contrôleur qui gère les rotations du petit orteil du pied gauche, va dans l'Outliner, uniquement pour découvrir une liste d'objets du genre nurbsCurve564, nurbsCurve1456, nurbsCurve32... Je vous assure que vous aurez de ses nouvelles assez rapidement. Prenez donc, le temps de tout renommer, même si vous ne le faites que pour le bienêtre mental des gens qui hériteront votre rig.

• Chaque contrôleur doit avoir ses channels inutiles cachés

En plus des avantages que l'on a mentionnés plus haut, cette pratique assure que seulement les channels nécessaires auront des keyframes, ce qui simplifie énormément la lecture du Graph Editor, et allège la scène.

Pour les rigs du court « L'Armoire », j'ai décidé d'utiliser surtout des contrôleurs en forme de cercle. Pour moi, il est plus important d'avoir une scène visuellement propre que d'avoir des contrôleurs « parlants ».

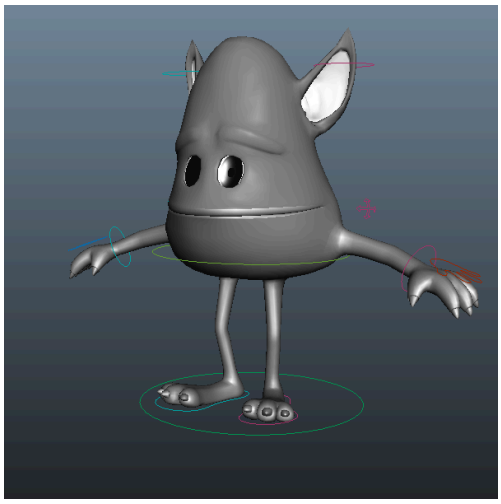
Dans les cas où il y a eu une ambiguïté pour comprendre la fonction d'un contrôleur, comme dans le cas des mains de la fille par exemple, où j'ai privilégié l'option d'avoir deux contrôleurs, un pour les rotations et un autre pour les translations, j'ai choisi deux cercles différents, l'un d'entre eux comportant une petite flèche, qui indiquait que le contrôleur servait à effectuer des rotations.

J'ai aussi privilégié ce type de courbe pour les contrôleurs en FK, pour garder une cohérence entre tous les contrôleurs qui serviraient uniquement pour des rotations.

Apperçu du rig de la fille.

Avant de passer à la création des rigs eux-mêmes, je voudrais m'attarder sur une technique que j'utilise pour chacun de mes rigs. Je conseille vivement, d'ailleurs, l'adoption cette façon de procéder. Le principe de cette technique, est de ne pas créer, de

façon générale, aucun contrôleur pour agir directement sur le squelette qui sera skinné. A la création du rig de chaque partie du corps, on duplique la chaîne de joints concernée, on renomme cette nouvelle chaîne de façon à ce que son nom indique clairement sa fonction, on crée un « parent constraint » allant de la nouvelle chaîne de joints à la chaîne originale, et on crée tous les contrôleurs, IKs, etc., sur cette nouvelle chaîne de joints.



Apperçu du rig du monstre.

Ce mode de procéder a plusieurs avantages. Tout d'abord, il nous permet de garder notre squelette dirigeant le skin libre de tout contrôleur. De cette façon, on peut skinner le personnage directement après la création du squelette, sans nous soucier des possibles modifications que l'on voudra faire sur le système de contrôle. Pendant tout le processus de rigging, le squelette du skin restera intact.

Admettons par exemple que l'on a créé un IK Handle pour contrôler un bras. Une fois notre rig fini, bien rangé, beau et prêt à être envoyé à l'animateur, on se rend compte que l'IK Handle a un bug : le twist n'est pas bien réglé, ou bien le contrôleur n'a pas été correctement mis en place, ou un système de Squash and Stretch donne des résultats étranges… Tous les utilisateurs de Maya savent que les sources de bugs incompréhensibles de ce logiciel sont inexhaustibles. Je vous invite alors à choisir votre bug préféré, un qui vous oblige à récréer tout le système d'IK. On sait bien que le fait de supprimer l'IK tout simplement laisse des traces sur les channels des joints du bras. Il faudrait donc, ne pas seulement supprimer l'IK Handle, mais aussi les joints qu'il contrôle, pour les créer à nouveau, de façon à ce qu'ils aient tous les channels comme avant la création de l'IK. Je vous laisse imaginer les implications que ceci aurait si ces joints là étaient les mêmes joints contrôlant le skin… Mais, en choisissant l'option de séparer le squelette de contrôle de celui à skinner, tout ce qu'on aurait à faire ce serait de créer une nouvelle copie du bras du squelette d'origine, et recréer l'IK dessus.

Un autre avantage est la possibilité d'avoir une diversité de types de contrôle pour chaque partie du corps. Il suffit d'avoir une copie de la partie du corps en question par mode de contrôle, et un système simple pour switcher d'un mode à l'autre. Un exemple de ceci est le changement entre modes IK et FK, que l'on verra dans une prochaîne section. Il est clair que pour créer ce type de système il n'est pas indispensable d'avoir une chaîne de joints par mode, mais c'est une façon plus propre de procéder. En plus, cette méthode nous permet d'avoir encore d'autres types de contrôle,

comme un bras « Bendy », par exemple. Avec ce type de setup, il n'y a pas de limites au nombre de types de contrôle que l'on peut créer pour une partie du corps quelconque.

Tout ceci dit, je ne conseille pas cette création de répliques pour toutes les parties du corps. Je pense que pour les chaînes de joints qui auront un mode de contrôle assez simple, uniquement par FK, par exemple, ne requièrent pas ce type de setup. Ici, les risques de « casser » la chaîne de joints skinnée sont infimes, et ne justifient pas la complexité additionnelle que représente la création systématique des squelettes séparés, contrôle/skin. On verra un exemple plus clair de ceci lors de la création des rigs des mains des personnages du court « L'Armoire ». Pour voir un autre exemple de l'utilisation de ce type de système dans un rig, je conseille [2], où des copies de chaînes de joints riggées sont utilisées pour contrôler le squelette principale dans le cas des bras et des jambes.

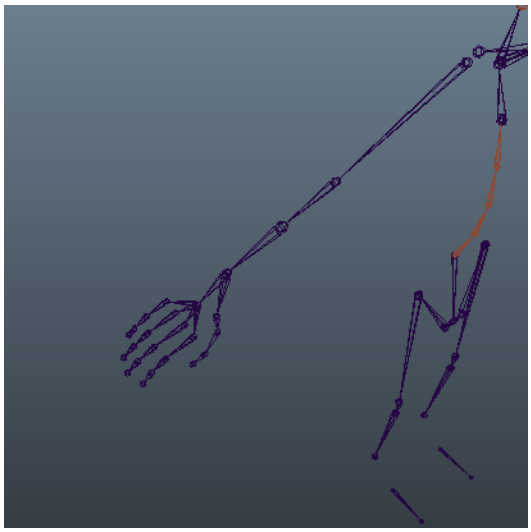
En ce qui nous concerne, on passera à la réalisation des rigs suivant l'analyse que l'on a faite plus haut, et en utilisant le type de contrôleurs que l'on a défini. D'abord, on décrira le rig correspondant au corps de chacun des personnages, suivi d'une présentation des rigs utilisés pour les visages des personnages.

La création du squelette

Plus haut, on a vu les aspects les plus importants qu'il fallait tenir en compte lors de la création d'un squelette, comme les axes de rotation locaux, aussi bien que les caractéristiques générales que devait comporter une chaîne de joints finie. Or, ceci ne nous dit rien concernant la façon dont les joints doivent être placés. Par où faut-il commencer lors du placement des joints ?

Comme pour la plupart des choix faits lors du processus de rigging, il faut d'abord commencer par analyser un bon nombre de références. Dans le cas d'un personnage bipède, l'option la plus naturelle est de s'inspirer sur l'anatomie humaine pour créer le squelette. Je dis "inspirer" parce que le but n'est pas de récréer un squelette humain. Ceci serait excessivement compliqué, et pas forcement très utile. L'idée c'est plutôt de s'inspirer de la façon dont le corps se déforme grâce aux os qui le soutiennent, et reproduire cet effet avec un système plus facile à gérer.

Lors de la définition d'un squelette, il est conseillé de tester plusieurs configurations différentes: différents placements de joints, chaînes de joints avec plus ou moins de membres, etc. A chaque fois que l'on construit un squelette, il faudrait skinner le personnage pour s'assurer que les transformations données par cette configuration particulière correspondent à ce que l'on veut. Ceci est particulièrement important pour la colonne, pour trouver un nombre de joints optimale, qui permettra un bon niveau d'articulation sans avoir des joints superflus.



Ce joint au milieu de l'avant bras est un exemple d'un joint d'aide au skinning.

Les autres parties du squelette sont en générale plus simples à construire. Cela dit, dans certains cas, en plus des joints de base, il faut aussi rajouter des joints d'aide au skinning. Ces joints peuvent ne pas être requis directement par les directives de l'animation, mais ils sont essentiels pour gérer les déformations de certains modèles, notamment les plus volumineux, ou ceux qui devront avoir des transformations complexes. Un exemple de ce type de joints est le joint placé au milieu de l'avant bras.

Ce joint n'a aucune justification anatomique si l'on se base uniquement sur la forme du squelette humain. Par contre, si l'on pense au fonctionnement de celui ci, cette configuration aide à simuler la torsion

présente dans l'avant bras lorsque l'on tourne la main.

A continuation on regardera la construction de chacune de ces parties du squelette un peu plus en détail, mais avant de commencer à construire le squelette, je voudrais vous parler d'un concept qui peut être très utile lors de la création de certains rigs, les hiérarchies cassées.

Les Broken Hierarchies ou « hiérarchies cassées »

Suivant le mode de procéder le plus "classique" pour la construction d'un squelette, celui-ci est une chaîne de joints unique, sans rupture, avec le root comme parent. Cette méthode n'a pas vraiment d'inconvénients, mais elle reste très limitée: les relations entre les différentes parties du corps étant définies par parentage directe, de façon à ce que tout joint, hormis le root, est placé sous l'autre dans la hiérarchie de Maya, le contrôle qui ont les parents sur les enfants ne peut jamais être désactivé. Cependant la possibilité de désactiver cette relation hiérarchique peut être très utile si l'on veut avoir des comportements un peu plus intéressants, comme par exemple la possibilité de dissocier le mouvement de la tête de celui des épaules.

Une Broken Hierarchy, comme son nom l'indique, est une hiérarchie composée de plusieurs parties indépendantes, liées entre elles par des contraintes qui contrôlent leur position et orientation. Comme on a vu plus haut, ces contraintes peuvent avoir plusieurs parents, et aussi être désactivées à tout moment, ce qui lnous ouvre un monde de possibilités.

Pour clarifier ce concept, prenons un exemple concret: les bras. Dans un squelette traditionnel le joint des épaules serait directement parenté sous celui de la clavicule. Avec cette configuration, si jamais il était nécessaire que les rotations du reste du corps n'affectent pas le comportement du bras, un squelette comme celui ci serait très loin de nous donner les bons résultats.

Alors, pour avoir plus de possibilités avec le même comportement de base, on fait appel aux broken hierarchies. Pour un setup de bras en broken hierarchy nous ne parenterions pas le bras sous la clavicule directement. Il serait dessiné comme une chaîne de joints indépendante, liée à la clavicule par deux parent constraints: un dirigeant uniquement les translations et un autre uniquement pour les rotations. La raison pour laquelle on utilise deux contraintes séparées est que, lors du travail avec des broken hierarchies, on ne désactive pas l'influence d'une chaîne sur une autre en termes de la translation, mais uniquement en rotation. La première contrainte restera alors intacte, et tous les changements de mode se réaliseront à niveau de la deuxième.

L'exemple que nous venons de voir peut être généralisé à d'autres parties du corps. On peut, aussi, vouloir garder les joints entre une partie du corps et l'autre, pour plus de visibilité et continuer à avoir une broken hierarchy. Par exemple, nous pouvons construire une broken hierarchy à niveau de la main, en dupliquant le bras entier, qui est dans ce cas la chaîne de joints qui contient le poignet, le parent de la main, et en effaçant tous les autres joints de cette copie. On construira alors la chaîne de la main partant de cette copie du poignet. Nous appliquerions alors les contraintes entre le bout du bras et le début de la main, deux joints identiques superposés.

Cette technique peut être appliquée pour n'importe quelle chaîne de joints, notamment les bras, les mains, la tête et les jambes. Et même si un squelette n'a pas été initialement conçu avec des broken hierarchies, celles ci peuvent être construites à tout moment, car elles ne changent pas ni l'emplacement, ni les rotations du joints.

Il existe plusieurs façons de procéder pour la création d'une broken hierarchy. Pour en voir un autre exemple, je vous conseille [2].

Gardant ces notions en tête, passons à la construction du squelette.

La colonne

Pour construire la colonne, on doit non seulement penser aux déformations voulues, comme on a vu plus haut, mais aussi à la façon dans laquelle on a besoin de manipuler le personnage. La colonne héberge le joint appelé Root ou centre de gravité. Ce joint est le parent de toute la hiérarchie du squelette, et il est aussi le joint qui sera liée au contrôleur global, celui qui servira à placer le personnage entier n'importe où dans la scène.

Pour choisir l'emplacement du Root, Eric Allen et Kelly L. Murdock, dans le livre Body Language : Advanced Character Rigging [8] présentent deux façons différentes de procéder : le Body Root et le Free Root. Le Body Root est le type le plus simple de Root. Ici, le Root est le parent direct des joints de la colonne et des hanches.

Il s'agit d'un système stable, facile à contrôler et à construire, mais il ne permet pas d'avoir un mouvement indépendant des hanches, ce qui le rend inutile pour la plupart des rigs.

Pour construire un squelette avec un Free Root, on a plusieurs possibilités. Une d'entre elles, présentée dans ce même ouvrage, est de construire cette partie du squelette d'une façon à ce que que le root ne soit pas placé à la base de la colonne, mais un joint plus haut.

Ce type de placement des joints permet un mouvement indépendant des hanches, mais rend la zone lombaire du personnage immobile, et ne sert pas à la création d'une colonne avec un IK Handle [8].

Il existe une troisième option, similaire à celle exposée par Paul Thuriot dans [2]. Cette configuration comporte trois joints superposés, qui servent comme base de la colonne, parent des hanches, et Root, respectivement. Pour créer ce type de colonne, il suffit de commencer par une colonne comme celle faite pour un système Body Root. Le Root est ensuite dupliqué deux fois, et tous ses enfants supprimés. De cette façon on aura trois joins identiques, exactement au même endroit. Ces joints (comme tout élément du rig !!) doivent être renommés pour que leur fonction soit claire. Chacun est libre de renommer les joints de son rig comme il veut, mais pour cet exemple admettons que les joints ont des noms RootJnt, HipJnt, et SpineBaseJnt. Pour terminer de construire la colonne il suffit de parenter les joints des hanches sous le joint nommé HipJnt, les joints de la colonne au SpineBaseJnt, et tant HipJnt et SpineBaseJnt sous RootJnt. De cette façon on aura les bénéfices du système Free Root sans en avoir ses limitations.

Le cou et la tête

La partie du squelette correspondant au cou et à la tête dépend fortement de l'anatomie du personnage. Pour la plupart des personnages bipèdes une simple chaîne de trois joints, commençant à la base de la colonne, et finissant au début du "crâne", suffit pour avoir les déformations requises. Ici encore il faudrait tester plusieurs configurations, en particulier pour trouver l'emplacement du joint de la tête, le pivot autour duquel se feront toutes les rotations.

Pour les personnages plus stylisés, un cou plus long peut être requis. Dans ce cas, le mieux et encore de faire plusieurs tests afin de trouver la définition la plus adapté à chaque personnage.

Partant du joint de la base de la tête on terminera cette partie du corps en rajoutant un dernier joint, situé à peu près au centre du front du personnage. Ce joint ci servira plus tard pour skinner les cheveux du personnage, par exemple.

Les bras et les jambes

En ce qui concerne les bras et les jambes il y a principalement trois façons de procéder. Nous nous limiterons à la description des chaînes de joints correspondant au bras, le cas des jambes étant très similaire en construction à celles-ci.

La façon la plus simple de créer un bras est de poser trois joints: un pour l'épaule, un autre pour le coude et un autre pour le poignet. Or, un bras réel possède deux os à niveau de l'avant bras, le radius et l'ulna, qui sont responsables pour la torsion qui se génère dans cette partie du bras lors que le poignet est tourné. Tel qu'il est, ce bras construit avec trois joints ne servira pas à simuler une déformation de ce type. Toutefois, comme on verra plus tard, qu'il existe un moyen pour simuler les déformations de l'avant bras, même sur un bras simple comme celui-ci.

Une deuxième façon de créer le bras, est de commencer par placer les trois joints principaux, comme décrit tout à l'heure, et en rajouter un quatrième, placé au milieu entre le joint du coude et celui du poignet. Le placement de ce joint additionnel est une des façons les plus courantes pour gérer les déformations de l'avant bras, mais il introduit une certaine complexité lors qu'on travaille avec des IK Handles. On peut aussi généraliser cette méthode en plaçant plusieurs joints à niveau de l'avant bras, pour arriver à simuler des déformations de plus en plus fines.

Une autre configuration possible est de créer un bras avec deux joints pour définir le coude. Ce type de bras (ou de jambe) peut être très utile lors de la création des personnages qui ont des bras très volumineux, car il permet une répartition plus efficace des pois à niveau du coude. Un exemple de ce type de bras peut être trouvé sur le tutoriel [7].

Quelle qu'elle soit la configuration choisie, il faut tester l'emplacement des joints avant de créer le squelette définitif. Aussi, il faut bien choisir les ordres de rotation, en particulier pour les joints des épaules, car ils ont trois degrés de rotation, et sont donc très sensibles au risque de Gimbal Lock.

Les mains et les pieds

Pour les mains, nous allons commencer avec un joint identique à celui du poignet. La meilleure façon de l'obtenir est de dupliquer le bras entier, et effacer tous les joints, sauf celui du poignet. On ne créera pas de contraintes pour l'instant pour le lier au reste du bras, on s'occupera du comportement de la chaîne lors de l'étape de rigging.

Pour créer la main il y a des milliers de configurations, selon le type d'action qui devra faire le personnage, et la flexibilité requise à niveau de la main. Pour les rigs qui nous concernent, nous allons dessiner des mains assez simples, mais qui permettront tous les mouvements dont nous aurons besoin: attraper des objets, faire des gestes, etc. Dans tous les cas, il est très important de placer les joints bien au milieu de la géométrie.

Pour ce faire, Aaron Holly, de Fahrenheit Digital, a une bonne astuce: pour créer une chaîne de joints bien centrée, il se sert des vertex qui définissent la géométrie, notamment des vertex autour d'un edge loop. Pour créer une chaîne de joints, il répète le processus suivant pour chacun des joints à placer: Il sélectionne les vertex de l'edge loop au centre duquel il voudrait placer son joint, pour ensuite en créer un cluster. Une fois tous les clusters construits, il dessine une courbe linéaire, de dégrée 1, représentant le chemin qui suivra la chaîne de joints, avec les clusters comme guide.

Une fois la courbe créée, il s'en sert pour placer les joins sur chacun de points de contrôle de la courbe. Quand la chaîne de joints est construite, il ne reste plus qu'à effacer les clusters et la courbe, et orienter les joints [7]. La courbe résultante sera non seulement au centre de la géométrie, mais elle aura aussi les conditions de transformation que nous avons établi dans la première partie: les rotations seront toutes à zéro, aussi bien que les translations, sauf pour la translation en X, dont la valeur représente la longueur de l'os.

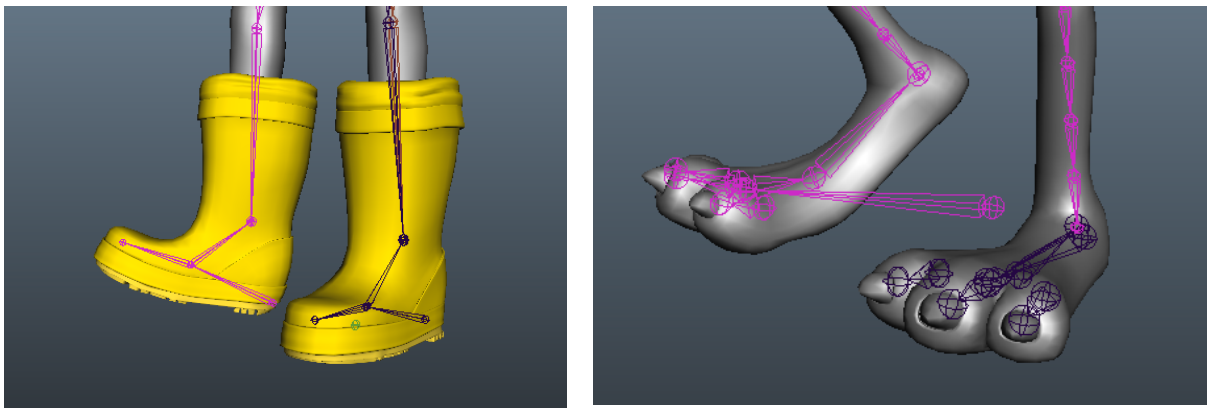
Une fois les joints placés, il faut penser à bien les orienter, et ce, pour tous les doigts, en particulier pour les pouces. Ceci dépend fortement de la géométrie du personnage, et pour cette raison, je ne rentrerai pas dans le détail de l'orientation des joints. En général, il suffit de suivre les directives indiquées dans la section consacrée aux joints.

Pour créer les pieds il y a encore énormément de possibilités: selon le modèle, ou bien le niveau de détail que l'on voudra pour les déformations suivant ces consignes, on choisira soit un système de

base avec un joint pour la cheville, construit à partir du joint du bas de la jambe comme nous avons fait pour le cas du poignet, un autre placé sur le point où naissent les orteils et un dernier allant jusqu'au bout des orteils.

Ce type de construction sera suffisante pour la plupart de personnages, en particulier ceux qui portent des chaussures. Par contre, si le personnage a des gros orteils, est pieds nus, ou bien a besoin d'avoir des déformations plus précises à niveau des orteils, d'autres configurations peuvent être envisagées.

Pour la fille, c'est le premier système qui a été choisi, car elle porte de grosses bottes en caoutchouc. En revanche, pour le monstre, un système légèrement plus complexe a été privilégié, pour permettre éventuellement le mouvement des orteils.



Deux types de setup des pieds.

Il existe plusieurs ouvrages qui présentent différentes possibilités pour la construction d'un squelette. Le livre Hyper-Real Body Setup de Paut Thuriot [2], par exemple, est une référence que je considère indispensable, puisqu'il qui comporte aussi des outils d'aide à la construction d'un squelette propre. Cela dit, j'insiste sur le fait que chaque modèle a des besoins particuliers, et chaque configuration doit être testée. Une fois le bon squelette trouvé, il ne reste plus qu'à le reconstruire tenant en compte les exigences présentées plus haut.

Les squelettes des personnages du court " L'Armoire"

Pour le torse du monstre, j'ai testé plusieurs configurations. Vu la façon dont il a été conçu, il n'avait pas vraiment besoin d'avoir une colonne proprement dite. Comme on a vu dans la partie du décryptage de l'animatique, l'idée c'était que le torse du monstre bouge comme un seul morceau, en un seul bloc.

Les squelettes de base des personnages du court.

J'avais plusieurs solutions différentes en tête pour arriver à cet effet, que l'on verra au moment du rigging. Pour l'instant alors, on se contentera de placer un unique joint, qui aura la fonction de root, comme on verra un peu plus tard, dans la section consacrée à la construction des squelettes. Pour la fille, le squelette suit plus de près les lignes d'un squelette bipède cassique, avec une colonne comme celle décrite dans [2].

A continuation, nous passerons à la construction des divers systèmes de contrôle conformant le rig du corps. Pour ce faire, nous procéderons d'une manière similaire à celle que nous avons adoptée pour le squelette, traitant chaque partie du corps à la fois.

Les Rigs

Le torse

Le but de cette section est de décrire la façon dont la colonne de la fille a été construite. Comme pour toutes les autres parties du corps, celle ci a été déterminé par les conditions requises pour son mouvement dans le court. Pour procéder à construire la colonne de n'importe quel personnage bipède il faut bien se préparer, prendre le temps de choisir le type de contrôle le plus adapté, puisque l'attitude, l'expression du personnage est en très grande partie définie par la forme de cette partie du corps.

Avant de commencer, il faut penser d'abord à la manière dont le personnage devra se déformer. S'agit il d'un personnage de type cartoon ? Ou d'un personnage plutôt réaliste ? Doit-il se déformer de façon extrême ? Tout ceci est très important pour savoir le type de colonne le plus juste pour le rig.

Lorsque l'on avance sur la création du rig, toutes les parties doivent travailler ensemble de façon harmonieuse, mais pendant cette étape, l'idéal est de se concentrer uniquement sur le mouvement du torse. Dans les mots d'Eric Allen et Kelly L. Murdock,

« Pendant que vous visualisez le torse, oubliez la tête, les bras, et les jambes ; pensez seulement aux hanches et à la colonne »[8].

Dans cette partie encore, l'analyse de références se révèle particulièrement utile pour définir les exigences qui doivent être relevées par le rig de la colonne. De forme générale, le torse d'un personnage bipède devra au moins remplir les conditions suivantes [1] :

1. Le rig du torse doit permettre la rotation des hanches et des épaules
2. La rotation doit aussi être possible dans tous les axes, de façon à ce que le personnage puisse se pencher vers l'avant, vers les côtés, et se tordre sur lui-même.
3. Les épaules et les hanches doivent pouvoir être manipulés de façon indépendante.

Selon le type de personnage, le torse doit aussi pouvoir s'étirer. Ce type de déformation est couramment appelé Squash and Stretch. Même s'il s'agit d'un personnage réaliste, le Squash and Stretch doit être inclus au moins comme une option. Ce type de déformation peut être utilisé sur

n'importe quel personnage, et le choix de s'en servir ou pas appartient à l'animateur. Plus tard, on rentrera plus dans le détail du fonctionnement du Squash and Stretch appliqué sur un torse.

Gardant cet ensemble d'exigences en tête, on peut passer à la sélection de type de contrôle le plus approprié parmi les différentes options disponibles. Il existe deux types principaux de rig de colonne : le premier est basé sur un IK Handle de type Spline. Le deuxième, un peu plus sophistiqué, appelé Ribbon Spine consiste en un certain nombre de joints attachés à une courbe.

La colonne contrôle par un Spline IK Handle est assez simple à construire, et le processus de constructions peut être trouvé dans plusieurs tutoriels, comme par exemple [1], [6], [7], et [8]. Ici je ne cherche pas à expliquer comment construire une colonne pas à pas, mais plutôt à développer les caractéristiques principales de ce type de système. En cas de doute, donc, je vous conseille de vous diriger vers l'une des sources nommées plus haut.

Comme on a vu dans un chapitre précédent, le Spline IK Handle est un IK qui est contrôlé par une courbe NURBS. On a vu aussi que ce type d'IK ne peut pas être manipulé directement, mais qu'il peut être déformé par moyen des CVs, les points de contrôle, de la courbe qui le dirige. Pour faciliter l'interaction avec ce type de système, il faut associer les points de contrôle de la courbe à des Clusters.

Dans le cas de la colonne, le plus courant est de créer l'IK Handle avec une courbe de dégrée 4, avec 5 points de contrôle. Partant de ces CV's, on créera 3 clusters pour contrôler la colonne : Un pour les deux CV's de la base de la colonne, un autre pour le CV du milieu de la colonne, et un troisième pour les deux CVs qui restent, ceux du haut de la courbe.

Nous prenons deux CVs à la fois pour la base et le haut de la colonne parce que de cette façon on aura la possibilité d'effectuer des rotations sur ces deux parties du torse. Une rotation effectuée sur un seul CV n'a aucun effet, mais elle l'aura sur un groupe de deux CVs ou plus. Une fois les Clusters créés (et renommés !), on peut déplacer leur point de pivot, si besoin. Encore, sur cette partie il s'agit surtout de tester le fonctionnement de la colonne avec des différentes positions des pivots des Clusters. Une fois les points de pivot placés, il ne reste plus qu'à créer les courbes de contrôle qui guideront les transformations des clusters.

Ensuite, afin d'avoir un contrôle plus précis sur les déformations de la colonne, et pour avoir un comportement plus réaliste, il faudrait régler les paramètres du twist. Ces paramètres contrôlent la

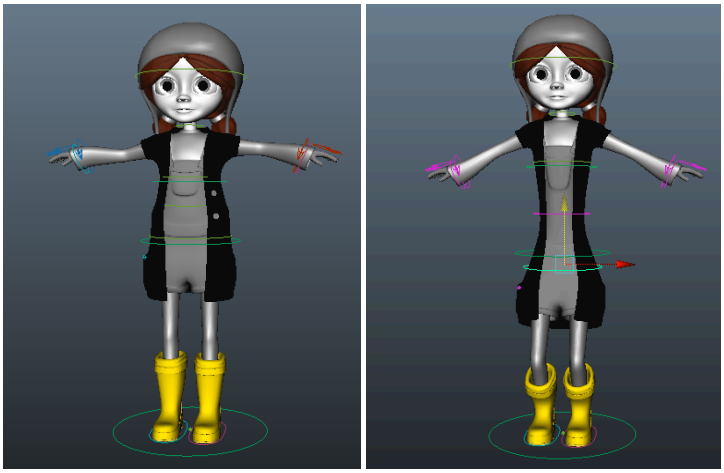
façon dont les joints intermédiaires de la chaîne s'orienteront sur leurs axes de torsion lors que les joints du début et de la fin de la chaîne sont transformés. Cet étape est donc déterminante pour avoir une torsion correcte au long de la colonne.

C'est justement avec le twist que l'on rencontre des problèmes lors de l'utilisation de ce type de colonne. Lors qu'un personnage riggé de cette façon est déformé de façon moyennement extrême, en tournant son torse près de 180° , les joints intermédiaires de la chaîne n'auront plus des orientations cohérentes, ce qui engendre des déformations pas tout à fait correctes.

Un autre type de rig de colonne est donc requis pour certains personnages. Ce type de rig, développé par Aaron Holly chez Fahrenheit Digital, est le Ribbon Spine. Cette colonne se base sur une série de joints attachés à une géometrie. Pour voir la construction de ce type de rig je vous recommande [7].

Squash and Stretch

Après avoir choisi un type de colonne, il faut à présent décider de quelle manière on va tacler le problème du Squash and Stretch. Il y a des tonnes de solutions pour ce problème, surtout pour les colonnes contrôles par un Spline IK. Cela dit, le principe reste toujours le même : modifier la longueur des os suivant le changement de la distance entre les contrôleurs qui les gèrent.



Système de Squash and Stretch sur la colonne de la fille.

De façon générale, pour créer un système de S&S on prend d'abord en compte la distance « de repos » entre les contrôleurs. Cette distance correspond, dans le cas d'un Spline IK, à la longueur de la courbe qui le contrôle. Ensuite, on crée un système pour pouvoir mesurer, à tout moment, la différence entre cette distance et la distance actuelle entre les contrôleurs. Ceci peut être fait par exemple, avec l'aide d'un Distance Tool. Si la distance actuelle est plus grande que la distance « de repos », les os doivent s'étirer de façon proportionnelle à la différence entre les deux distances.

Cet étirement des os peut être fait par moyen du Scale des joints, mais certains auteurs [1]

déconseillent cette méthode, privilégiant plutôt les translations pour arriver au même effet.

On peut aussi créer l'effet inverse, en raccourcissant les os lors que la distance actuelle entre les contrôleurs est plus petite que la distance originale. Cet effet est très utile pour simuler la compression de la colonne.

Finalement, pour que l'effet du Squash and Stretch, soit réussi, il faut aussi modifier le volume des joints de façon inversement proportionnelle à l'étirement ou rétrécissement des os.

Il existe beaucoup de tutoriels qui couvrent ce sujet, et qui parviennent à créer des systèmes de Squash and Stretch des formes très différentes. On peut les créer à l'aide d'expressions, de nodes, et même de Set Driven Keys et des déformeurs. Personnellement, la méthode que je recommande le plus est celle présentée par Jason Schleifer [1], non seulement grâce à sa façon d'expliquer, qui est très claire, mais aussi parce que le Squash and Stretch qu'il dévoile donne des très bons résultats en animation.

La colonne de la fille

Ayant vu ces deux types de colonne, et les grandes lignes du Squash and Stretch, passons aux systèmes que j'ai sélectionnés pour le rig de la fille. Au début, j'ai été attirée par l'idée de faire une colonne Ribbon pour ce personnage, afin d'éviter les limitations inhérentes à l'utilisation du Spline IK. Cependant dans cette animation je savais bien que n'aurais jamais besoin de déformer le personnage assez pour arriver à rencontrer des problèmes avec un Spline. Pour ce rig, j'ai donc privilégié le premier type de colonne. Avec celui-ci, je pouvais amplement remplir toutes les conditions de mouvement requises pour l'animation.

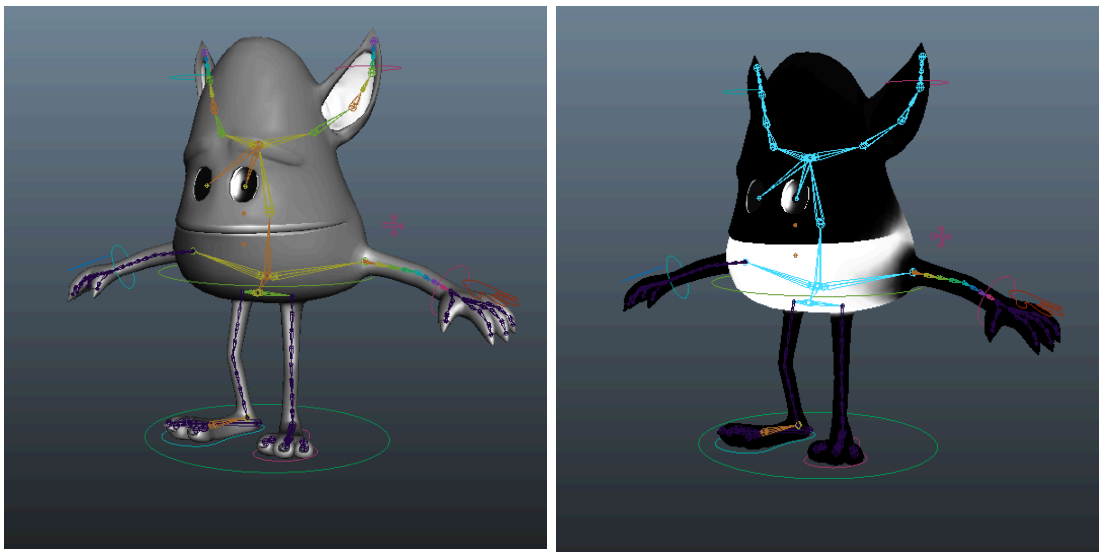
Le torse du monstre

Comme mentionné un peu plus haut, pour le monstre, à niveau de la "colonne" l'idée c'était d'avoir un rig qui nous permettait de traiter le torse comme un seul bloc, avec peut être quelques déformations subtiles à niveau du haut de la tête.

Il n'était pas nécessaire donc, d'avoir une colonne sophistiqué. En revanche, il fallait trouver la bonne configuration de contrôleurs pour arriver au résultat voulu.

Pour le torse du monstre, j'avais principalement deux voies possibles: soit je déformais le torse avec un lattice, et de cette façon je pourrait avoir des déformations plus fines par dessus si jamais j'en avais besoin, soit je travaillais avec un skin centré sur un seul joint, celui du root. Dans ce dernier cas, les déformations additionnelles pouvaient être rajoutées avec des Blend Shapes.

Après avoir testé les deux configurations, j'ai finalement tranché pour la deuxième, qui était plus légère en termes des calculs, et qui avait aussi un moindre risque de générer des problèmes de compatibilité avec les rigs des autres parties du corps. J'ai donc laissé ce joint unique pour la "colonne", et j'ai skinné le monstre en assignant un poids de "1" à tous les vertex du corps pour le joint du root.



Le root du monstre et la répartition de son poids dans le skin.

La tête et le cou

Une fois la colonne décryptée, passons à la description du rig de la tête et le cou. Le rig de ta tête peut être aussi simple qu'une chaîne de trois joints contrôlé en FK, ou bien aussi avancé qu'un rig similaire à celui de la colonne, avec du Squash and Stretch. L'important, c'est de penser aux besoins du personnage, pour ne pas se retrouver avec un rig qui soit trop simple pour les besoins de l'animation, ou bien trop développé pour les actions requises.

Avant de commencer à rigger cette partie du corps, pensons aux caractéristiques générales que ce rig doit avoir. Comme le rappelle Jason Schleifer dans Animation Friendly Rigging [1] :

« C'est la relation entre la tête et les épaules qui définit l'attitude d'un personnage »

La tête doit, donc, en premier lieu, pouvoir être orienté de manière indépendante par rapport aux épaules. La plupart du temps, la tête doit suivre les rotations des épaules, mais si besoin, l'animateur doit pouvoir changer «d'espace », de façon à ce que la tête reste fixe lorsque le torse est tourné. Ceci minimisera la quantité de « contre-animation » requise pour certaines séquences, pour les cycles de marche, par exemple.

En outre, la tête doit idéalement pouvoir être déplacée. Ceci peut ne pas correspondre à l'anatomie du corps humain, mais c'est encore un type de contrôle qui donnera plus d'opportunité aux animateurs pour s'exprimer librement. En plus d'avoir la possibilité de déplacer la tête, il serait utile de pouvoir encore changer «d'espace » pour les translations de la tête. Mais, c'est quoi cette histoire de « espaces » ? Avant de continuer avec le rig de la tête faisons un petit détour pour découvrir se qui se cache derrière cette question.

Le changement d'espace

De la même façon d'un squelette, un rig est principalement hiérarchique. Ceci veut dire que la plupart éléments dans un rig sont construits pour être enfants d'autres éléments. Or, comme on a vu dans le chapitre sur les bases du rigging, lors qu'un objet est enfant d'un autre, celui ci se comporte comme si l'origine de son espace se trouvait au point de pivot de son parent. On a vu aussi que, si le parent est transformé en rotation ou translation, l'enfant se déplacera avec lui. Cependant, de fois il est avantageux de changer le parent d'un objet, ou bien de transformer un objet comme s'il n'avait

pas de parent, ou encore de le faire suivre une partie du corps différente à celle qu'elle suit d'habitude. C'est là où entre en jeu le « changement d'espace ».

Pour rendre ce concept plus clair, passons à un exemple. Les contrôleurs des mains, lors qu'un bras est en mode IK sont souvent parentés sous les contrôleurs du haut du torse. De cette façon, lors que le torse est tourné, les bras le suivent.

Ceci pet être pratique, sauf, par exemple, si le personnage pousse un objet. Riggé de cette façon, si nous déplaçons notre personnage vers l'avant, voulant garder ses mains sur l'objet qu'il pousse, il faudrait d'abord faire avancer son torse, et ensuite faire reculer les contrôleurs de ses mains la même distance.

Comme ceci n'est pas très pratique, on pourrait décider de ne pas parenter les contrôleurs des mains du tout, et permettre qu'ils soient déplacés uniquement de façon indépendante. Ceci est fait sur plusieurs rigs, mais c'est une solution qui reste limitée, car à chaque fois que l'on voudra bouger l'ensemble du torse, bras compris, on sera obligé de tout animer séparément.

Et si, par exemple, le personnage tenait sa tête, pendant qu'il regarde un véhicule passer (la voiture de ses rêves, peut être, ou le bus qu'il vient de louper), par exemple ? Il serait vraiment pratique que sa main suive sa tête pendant le mouvement, n'est ce pas ? En lieu de devoir replacer sa main sur sa tête à chaque frame··· Que fait-on dans ce cas là ? On parente la main à la tête ? Cela risquerait sans doute d'avoir des effets assez particuliers.

La solution optimale serait alors, une méthode qui nous laisse possibilité de changer le parent des contrôleurs des mains à n'importe quel moment. Ce type de rig, où l'on peut changer les parents de certains éléments est appelé un rig avec changement d'espaces. On peut se servir de ce type de système sur les mains, les pieds, la tête···

La façon dont ce type de setup marche suit la même idée derrière la Broken Hierarchy : En lieu de parenter les contrôleurs directement, on utilisera des « Parent Contraints » avec plusieurs parents, et un mécanisme pour changer de parent à volonté. Pour retourner à l'exemple de la main, en appliquant ce système on appliquerait un « Parent Constraint » sur les contrôleurs des mains, avec le contrôleur du torse, le contrôleur des hanches, le contrôleur global, et le contrôleur de la tête comme parents. Ensuite il ne resterait plus qu'à trouver un moyen de faire que les options soient disponibles à l'animateur.

Bien sûr, la contrainte ne se créerait pas directement sur les contrôleurs, la méthode est un peu plus complexe que ce que l'on vient de voir, mais l'idée du fonctionnement reste la même.

Le rig de la tête

Ayant vu ce qui veut dire « se transformer dans un autre espace », retournons à ce qui nous intéresse à présent : le rig de la tête et le cou.

Rappelons-nous des conditions que l'on a trouvées pour le rig de la tête :
La tête doit pouvoir tourner de façon indépendante à la rotation des épaules
La tête doit pouvoir être déplacée, et son mouvement doit pouvoir être fait dans au moins deux espaces : l'espace du torse, et l'espace global

Voilà nos conditions pour le rig de la tête. Mais, entretemps, quelle est le rôle du cou ? De façon générale, l'animateur ne veut pas devoir se soucier du cou. Il voudra plutôt pouvoir créer les poses nécessaires pour son animation en plaçant la tête là ou il faut ; le cou doit la suivre. Cela dit, ça dépend aussi du personnage. Pour les personnages qui on un cou très long, il conviendrait d'avoir un peu plus de contrôle sur les déformations de cette partie du corps. Cela dit, le rôle du cou reste le même : assurer une liaison entre les épaules et la tête, et ce de la manière la plus naturelle possible.

Comme pour la colonne, pour rigger le cou et la tête il y a plusieurs possibilités. Dans les lignes qui suivent on verra les deux principales : une avec un type de cou assez simple, et une autre où le cou est riggé comme une mini colonne.

Pour la première technique, le rig est bien élémentaire. On garde le cou tel qu'on l'a construit dans la section du setup. Dans cette configuration on aura deux possibilités : soit on garde un seul contrôleur, pour les rotations de la tête, soit on en a deux : un pour la tête et un autre pour le cou. Dans les deux cas, tout ce qui reste à faire pour avoir le rig de la tête est connecter les courbes de contrôle aux joints correspondants. Pour lier le contrôleur de la tête au torse on peut utiliser un « Parent Constraint » pour que l'on puisse disposer d'une rotation indépendante de la tête.

Ce système est raisonnablement facile à créer, mais il nous prive de la possibilité de déplacer la tête. Avec ce type de rig, on ne peut qu'appliquer des rotations sur la tête, les translations ne sont pas

possibles. Ceci peut être acceptable sur des personnages secondaires, ou qui auront des animations limitées. Cependant, pour les personnages principaux, cette configuration est franchement trop restreinte pour être accepté.

C'est ici que le deuxième type de cou entre en jeu. Pour pouvoir créer ce rig de tête, il faudra légèrement modifier le squelette que l'on a crée. Ici, l'idée est d'avoir un cou qui soit une chaîne de joints séparée. Pour ce faire, on duplique toute la chaîne de joints formant le cou et la tête, et on supprime tous les joints de la copie, sauf celui qui correspond au joint du haut du cou. Ensuite, on parente les joints de la tête sous ce nouveau joint. De cette façon, la tête et le cou seront des chaînes différentes.

A continuation, on procède comme pour la création de la colonne de type Spline : on crée un Spline IK Handle sur la chaîne du cou, et on crée les Clusters et les courbes de contrôle comme on a fait pour le torse.

Pour assurer des translations, il ne resterait plus qu'à appliquer un système de Squash and Stretch sur l'IK du cou.

Il est vrai que ce type de rig prend beaucoup plus de temps à faire, mais les résultats le valent. Cette dernière méthode peut être trouvée dans [1].

Le cou et la tête de la fille

Pour la fille, c'est ce dernier système que j'ai choisi. Je voulais avoir le plus de liberté possible pour animer cette partie du corps, étant donné que des nombreux plans de l'animation étaient des gros plans de la tête, où avoir une expression claire, et surtout bien lisible, était d'importance capitale.

Les bras

Les bras sont une partie énormément expressive du corps humain, leur position étant de grande importance pour définir le caractère d'un personnage. Pour rigger cette partie du corps il faut alors prendre sont temps et, comme toujours, faire des tests pour trouver les bons ingrédients pour faire le type de bras le plus approprié pour les besoins de l'animation. Considérons ensuite les conditions que doivent remplies par un bras typique [1]:

1. La capacité de faire des gestes avec les bras est indispensable pour un personnage. La plupart de personnages bougent leurs bras pendant qu'ils parlent, par exemple. Pour certains mouvements, le plus pratique c'est d'animer en FK, comme pour les cycles de marche, ou bien pour les séquences où il est important d'avoir un mouvement des bras en arcs. Cependant, la plupart du temps le plus facile c'est d'animer en mode IK. L'idéal est ainsi, d'avoir les deux types de contrôle disponibles, et une manière de passer d'un mode à l'autre.
2. Dans certains cas, on voudra que les bras et les mains bougent avec le reste du corps. Mais il y aura des moments où il faudra que les mains du personnage restent fixes, soit sur un objet, ou bien sur une autre partie du corps. En FK c'est le même cas : de fois on voudra que les bras suivent les rotations du corps, mais d'autres où l'on voudra que leurs rotations soient indépendantes, qu'elles puissent être isolées. Suivant ceci, il nous faut alors n système que permette le changement d'espace tant en mode IK, comme en FK.
3. Dans tous les cas, un système de torsion de l'avant bras doit être inclus.

Dans cette section, on travaillera comme décrit un peu plus haut: les différents rigs ne seront pas créés sur le squelette principale, mais sur des sous chaînes copiées de celui ci. On appellera ce type d'approche "Setup à triple bras". Toutefois, la plupart des principes exposés peuvent être appliqués à un système différent, où les contrôles sont créés directement sur le squelette principale.

Tout d'abord, on va s'intéresser à la construction d'un bras en IK, avec toutes les caractéristiques qu'il doit comporter, pour ensuite passer au bras en mode FK. On traitera ces deux bras séparément dans un premier temps. On finira le rig du bras en intégrant ces deux systèmes en les utilisant pour contrôler notre squelette principale.

Bras Ik

On commencera alors par la construction d'un rig très simple, conformé par une chaîne de trois joints, un pour l'épaule, un autre pour le coude et un dernier pour le poignet, le tout contrôlée par un IK Handle. Celui-ci doit être ensuite contrôlé par une courbe simple pour faciliter sa manipulation. Mais avant de nous lancer dans la création de l'IK il faut noter que notre squelette, tel qu'il a été conçu plus tôt, comporte un bras à 4 joints. Les manipulations qui suivent ne pourront pas être directement appliqués sur ce bras là. Cependant, on verra vers la fin de cette section comment contrôler un bras de plus de 3 joints avec un système comme celui que nous nous apprêtons à construire.

On commence donc, par la création d'un IK Handle sur une chaîne de joints formée par l'épaule, le coude, et le poignet de notre squelette. Cette chaîne de trois joints est obtenue à partir d'une copie du bras du squelette d'origine, sur laquelle les joint superflus ont été supprimés. On renomme cette copie en lui donnant un suffixe qui indique sa fonction. Par exemple, dans le cas du bras gauche, on pourrait renommer cette chaîne "armIKGuide_Lf".Il est très important que les joints que l'on garde pour ce bras en IK soient des copies conformes des joints correspondants dans le squelette principale, pour assurer le bon fonctionnement du bras plus tard lorsqu'on connectera toutes les différentes parties.

Une fois cette chaîne de joints crée et renommée, on lui crée un IK Handle de type "Rotate Plane", que l'on contrôle par moyen d'un « point constraint » dirigé par une courbe NURBS.

Étant donné que le bras doit pouvoir être orienté sur plusieurs plans, il nous faut un IK Handle de type RP. Or, pour orienter ce type d'IK, il est nécessaire d'avoir un Pole Vector, où une contrainte venant d'un objet quelconque, comme un Locator ou une courbe NURBS, qui déterminera l'orientation du plan sur lequel s'oriente la chaîne de joints contrôlée par l'IK.

Le « Pole Vector »

La création d'un Pole Vector de base est très simple. Il suffit de sélectionner l'objet vers lequel on veut orienter l'IK, suivi de l'IK Handle, et aller sur Constraint > Pole Vector. Le problème avec ce mode de création est que, si l'objet que l'on a choisi pour définir le Pole Vector n'est pas placé d'une façon précise, cette manipulation introduira des valeurs de rotation dans les channels des joints du bras. Sachant que depuis le début, lors de la construction du squelette on s'est battu pour que ces

channels restent propres, avec des jolis zéros partout, ceci n'est pas acceptable ! Heureusement pour nous, il existe une petite astuce pour éviter que ceci n'arrive.

Cette technique consiste à bien placer notre contrôleur avant de créer la contrainte. Ce « placement précis » fait en sorte que le nouveau plan d'orientation de l'IK Handle, déterminé par le Pole Vector, coïncide avec le plan sur lequel l'IK a été crée. Pour bien placer l'élément que l'on utilisera comme Pole Vector on commence par créer un « point constraint » sur lui, dirigé par les joints de l'épaule, du coude et du poignet. Grâce à cette contrainte, notre élément sera placé directement sur le plan sur lequel se situent ces trois joints. Ensuite, on crée une deuxième contrainte, cette fois de type aim, de façon à ce que l'objet qui nous servira comme Pole Vector pointe vers le joint correspondant au coude. Pour terminer, il faut seulement effacer les deux contraintes, et placer l'objet derrière le bras, prenant bien soin à le déplacer en mode « objet », et uniquement dans l'axe X. Une fois cet objet placé, nous pouvons créer le Pole Vector sans changer les rotations des joints du bras [6].

Une fois le Pole Vector crée, on doit choisir la façon dont on contrôlera l'orientation du coude : soit on la contrôle par moyen de l'objet qui a servi à la création de la contrainte, soit on cache cet objet, et on contrôle l'orientation du coude grâce à un attribut additionnel créé sur le contrôleur de l'IK. Personnellement, je préfère la deuxième option. Procédant de cette façon, non seulement on réduira le nombre totale de contrôleurs présent sur notre rig, simplifiant ainsi son utilisation, mais aussi on facilitera le travail de l'animateur, puisque de cette manière, il pourra modifier toutes les caractéristiques du bras sans devoir manipuler plusieurs contrôleurs.

Le premier type de contrôle est très simple à mettre en place. En fait, il ne faut presque rien de plus. Cependant, si l'on veut contrôler l'orientation du bras par moyen de l'objet utilisé pour la création du Pole Vector, il faut, comme pour tout autre contrôleur, que celui-ci ait un group d'offset au dessus de lui dans la hiérarchie, afin de garder les transformations du contrôleur à zéro. Ensuite, on peut (et je le conseille, d'ailleurs) créer un « Parent Constraint » afin que le contrôle du Pole Vector suive le contrôleur de la main. Autrement, le coude pointera constamment au même point, même si le contrôleur de la main est déplacé. Ceci qui risque d'être gênant pour l'animateur, qui sera obligé de déplacer le contrôleur du Pole Vector systématiquement pour répondre à chaque mouvement de l'IK. Cette contrainte, comme toutes les contraintes appliquées sur des contrôleurs, doit être appliquée sur un node au dessus du contrôleur lui-même, pour pouvoir continuer à manipuler celui ci malgré la contrainte.

Pour le deuxième type de contrôle, on commence par créer le « Parent Constraint » sur l'objet utilisé

pour créer le Pole Vector. Même si normalement on ne le manipulera directement, je conseille que ceci soit fait sur un groupe au dessus du contrôleur. De cette façon, si on change d'avis sur le mode de contrôle que l'on veut pour l'orientation du bras, l'option d'utiliser ce contrôleur reste disponible.

Ensuite, on doit cacher ce contrôleur, il ne nous servira plus. Pour manipuler le Pole Vector on créera un nouvel attribut, appelé IkTwist, sur la courbe NURBS qui contrôle l'IK. On associe ensuite ce nouvel attribut à l'attribut du même nom qui se trouve sur l'IK lui-même. Ceci peut être fait par moyen d'une expression, de Set Driven Keys, ou bien d'une connexion directe entre nodes.

Ayant intégré le contrôle du Pole Vector, notre bras simple est prêt. Maintenant nous devons associer ce rig à un bras identique à celui présent sur notre squelette de base. Pour ce faire, on crée une deuxième copie du bras, et on la renomme. Personnellement je choisirai un nom dans les lignes de 'ArmIK', pour le différencier du bras FK que l'on créera plus tard.

Afin que l'IK que nous avons construit sur le bras de trois joints dirige cette nouvelle chaîne, nous devons créer quelques contraintes entre les deux. Pour commencer, chacun des joints de la chaîne "guide" doit agir sûr le joint que lui correspond sur la deuxième chaîne. En l'occurrence, cette deuxième chaîne à quatre joints, donc il ne nous reste qu'un joint à contrôler, mais cette méthode peut être adaptée à des chaînes à n'importe quel nombre de joints.

Dans le cas qui nous concerne, le quatrième joint est pile au milieu des joints correspondant au coude et au poignet. On le contrôlera alors, avec un « Point Constraint » dirigé par les joints correspondant à ceux ci sur la chaîne "Guide", en mettant une valeur de 0.5 sur le poids correspondant à chacun des joints. La raison pour laquelle on utilise un « Point Constraint » en lieu d'un « Parent Constraint » est que, pour ce joint ci, on aura besoin de garder les channels correspondant aux rotations libres, car ce sera à partir des rotations de ce joint que l'on créera la torsion de l'avant bras plus tard.

Comme je l'ai dit plus tôt, cette méthode peut être appliquée sur des bras avec plus de joints, ou bien avec des joints placés différemment . Il suffit de changer la valeur de chacune des contraintes en conséquence, mais le principe est toujours le même. C'est d'ailleurs cette même méthode qui a été utilisée pour le rigging du bras du monstre.

A présent, notre bras simple est prêt. On pourrait ensuite le modifier pour qu'il ait un système de Squash and Stretch, un Smooth IK, ou bien un mode qui permette de fixer le coude dans une position

définie pour que tout mouvement se fasse uniquement à niveau de l'avant bras, mais on gardera ce setup pour le moment. Tous les autres modes de contrôle peuvent être incorporés une fois le bras "de base" terminé.

Le bras Fk

Pour le bras FK le premier pas est, comme pour le cas du bras IK, de créer une copie du bras du squelette original et la renommer. Dans le cas du FK on n'aura pas besoin d'une chaîne de contrôle intermédiaire, on travaillera directement sur cette copie.

On commence par créer trois contrôleurs, un pour l'épaule, un autre pour le coude, et un dernier pour le poignet, placés sur les joints qu'ils doivent contrôler, avec un ou deux groupes par dessus pour garder les transformations à zéro. Il est important que ces contrôleurs soient facilement différenciables des contrôleurs utilisés en mode IK, pour éviter toute confusion. Si possible, il est préférable de choisir des courbes qui indiquent que ces contrôleurs serviront à effectuer des rotations.

On commence alors par l'essentiel: les contrôleurs FK doivent guider les rotations des joints correspondants. On crée alors un « Orient Constraint » entre chacun des contrôleurs et son joint associé, sauf pour celui du poignet, que l'on traitera un peu plus tard. A présent, les joints suivront les rotations des contrôleurs. Cependant, si nous appliquons une rotation sur le contrôleur de l'épaule, par exemple, les joints du coude et du poignet se déplaceront, laissant leurs contrôleurs derrière. Il nous faut donc, quelques contraintes pour nous assurer que les contrôleurs suivront le mouvement des joints quand ceux ci se déplaceront: des « Point Constraints » allant de chacun des joints jusqu'à leur contrôleur associé. De cette façon, notre problème de contrôleurs égarés est résolu.

Ensuite, nos recréerons la hiérarchie des joints contrôlés sur les contrôleurs eux mêmes. C'est à dire, on parentera le contrôleur du poignet sus celui du coude, et celui du poignet sous celui du coude. Pour ce faire on a deux possibilités: Soit on met les contrôleurs directement l'un sous l'autre dans l'Outiner, soit on crée des « Parent Constraints ». Normalement la deuxième option est préférable, car, comme on a vu plus haut, les contraintes peuvent être activées et désactivées à volonté, ce qui peut être pratique si l'on veut avoir plusieurs comportements différents avec un seul rig. En créant un attribut pour contrôler le parent constraint qui lie chaque contrôleur à son parent, on peut ainsi avoir

une sorte de “changement de space” en mode FK.

Le setup du bras FK est maintenant terminé. Il ne nous reste plus qu’à rassembler les deux bras que l’on à construit sur un seul système.

Le bras Ik/Fk

A présent, nous avons deux bras riggés avec des systèmes différents, un en IK et un autre en FK créés sur des copies exactes du bras du squelette original. Tout ce qu’il nous reste à faire c’est connecter ces deux chaînes au bras principale, et créer un node qui nous permette de passer facilement d’un mode à l’autre.

On commencera par la création de « Parent Constraints » allant des bras IK et FK vers le bras principale, celui qui sera skinné. A la création de chacune des contraintes, on peut procéder dans n’importe quel ordre, soit en sélectionnant d’abord un joint du bras IK et ensuite le joint correspondant sur le bras FK, ou l’inverse, mais il est important d’utiliser toujours le même ordre, et aussi de prendre note de l’ordre choisi, car ceci facilitera la connexion des attributs plus tard. Ensuite on doit choisir sur quel élément on placera l’attribut qui servira à passer d’un mode à l’autre. On peut créer l’attribut sur le contrôleur de la main (mais dans ce cas il faudrait penser à créer l’attribut sur les contrôleurs IK et FK de la main, pour que l’attribut soit disponible dans les deux modes), ou bien sur un élément séparé placé ailleurs dans la scène, ou sur un groupe vide... Ceci dépend surtout des préférences de l’animateur.

Personnellement je préfère placer tous les switches de ma scène sur un objet indépendant du reste du rig, comme un groupe vide ou bien un locator. De cette façon, si jamais on a besoin de changer l’emplacement d’un switch particuler, on n’a qu’à le récupérer de ce node là. Une fois l’attribut crée, il faut le connecter aux « Parent Constraints » des joints que l’on a créé sur le bras original. Pour ce faire on peut travailler avec des nodes, des expressions, ou bien des SDKs. Quelque soit la méthode choisie, avant de terminer, il faut aussi penser à contrôler la visibilité des contrôleurs selon le mode actif: lors qu’on est en mode FK, on n’a pas besoin de voir les contrôles IK, et vice-versa.

Comme je l’ai mentionné un peu plus haut, on peut à continuation rajouter les contrôles plus sophistiqués, selon les besoins de l’animation. On peut commencer, par exemple, par rajouter un système de Squash and Stretch. Celui ci serait créé suivant les mêmes principes que pour le Stretch de la colonne, avec la particularité que, pour les bras et les jambes, on ne voudra pas généralement

que les joints se raccourcissent. On limiterait donc les transformations pour le cas où la distance du contrôleur de la main à la base du bras est plus longue que la somme des longueurs des joints qui le forment.

On peut aussi rajouter des caractéristiques au rig du bras on créant un système qui permette de “bloquer” le coude d’un personnage suivant la position d’un locator. Ceci peut être utile lorsqu’un personnage doit par exemple, faire des gestes avec ses mains tout en gardant ses coudes sur une table. Un bon exemple de comment construire un rig qui permette ce type de comportement peur être trouvé en [1].

Le bras que l’on vient de créer remplit la première des conditions pour du rig que nous avions imposées au début de cette section: il permet d’animer en FK ou en IK, et il permet de passer facilement d’un mode à l’autre. Passons alors à la deuxième condition: nous devons alors modifier le rig du bras que nos avons créé afin qu’il soit possible de changer d’espace de transformation.

Changement d'espace

Pour créer un système de changement d’espace pour les bras, nous procéderons de façon similaire à comme on a fait pour la tête. Pour le cas des contrôleurs de mains en mode IK, nous devons au minimum avoir le choix de les déplacer dans l’espace des épaules, l’espace du monde, et l’espace du corps. Dans ce dernier mode, les mains suivront le mouvement du contrôleur global, alors que dans l’espace du monde les contrôleurs des mains seront indépendants de tout autre contrôleur du rig.

Comme on a vu pour l’exemple de la tête, ce type de comportement peut être généré en utilisant un « Parent Constraint » avec plusieurs parents, et un attribut pour que l’animateur puisse accéder facilement aux différents modes de transformation.

Avec l’ajout du système de changement d’espace, le bras que l’on a construit n’est pas loin de remplir toutes les conditions que nous avions déterminées au début de cette section : on a un bras qui peut être transformé en IK ou FK, et ce en différents espaces.

A présent il nous reste un dernier aspect du rig du bras à aborder, la rotation de l’avant-bras. Mais, avant de nous lancer, créons des connexions entre nos contrôleurs des mains et le joint du poignet.

Tenant en compte la façon dont on a construit le squelette, les mains sont une chaîne de joints indépendante commençant par un joint qui coïncide avec celui qui se trouve à la fin du bras. Tout d'abord, nos devons nous assurer que les mains suivent les translations des bras. Pour ce faire, il suffit de créer un « Point Constraint » allant des contrôleurs des mains jusqu'au début des mains, et ce pour chacun des modes de transformation, IK et FK. Bien sûr, cette contrainte aussi doit être modifiée selon le mode choisi. Ensuite, il faut créer aussi une contrainte pour gérer les rotations des mains, cette fois de type Orient, liée elle aussi au mode choisi. Le système qu'on construira à continuation, reprend les rotations du joint du poignet pour simuler la torsion de l'avant bras.

La déformation de l'avant bras

Pour attaquer cette problématique, nous avons essentiellement trois solutions. La première, la plus répandue, est le fait de rajouter un joint additionnel à la création du squelette, placé au milieu entre le joint du poignet et celui qui correspond au coude. C'est d'ailleurs la solution que l'on a implicitement décidé d'adopter lors de la création du squelette il y a quelques pages. On peut aussi généraliser cette méthode en créant plusieurs joints additionnels entre le joint du coude et du poignet pour avoir une répartition de plus en plus fine des déformations. Au moment du skinning, ces joints seront utilisé pour distribuer les rotations du poignet au long du bras.

Pour ce faire, il faut associer la rotation dans l'axe de torsion (l'axe x, par défaut) du contrôleur de la main à celles des ou du joint de l'avant bras, pour chacun des bras de notre rig (IK, FK, Bendy...), de façon à ce que ces dernières héritent une petite partie de la torsion du poignet. On peut par exemple, faire ceci avec une expression : si IKHandCtrl_Lf est le contrôleur correspondant de la main gauche, et forearmJnt_Lf est le joint de l'avant bras correspondant, pourrait créer l'expression suivante, par exemple:

```
forearmJnt_Lf.rotateX = 0.5*IKHandCtrl_Lf.rotateX
```

On peut aussi faire plus sophistiqué, en créant un attribut sur le contrôleur de la main qui contienne le coefficient qui modifiera la rotation de l'avant bras. De cette façon, la magnitude de la déformation pourra être modifiée à tout moment et de manière transparente, simplement en modifiant un attribut.

La deuxième solution fait appel aux clusters. De cette façon, on peut avoir la torsion de l'avant bras même en ayant un bras sans joints additionnels. On associe un cluster aux vertex de l'avant bras de notre modèle, et on fait en sorte que ce cluster hérite, comme on a fait pour la méthode précédente, une partie des rotations du poignet. Encore une fois ceci peut être fait de plusieurs manières.

La dernière option est un mélange des deux précédentes: on garde le ou les joints additionnels, en rajoutant une déformation plus fine par dessus en se servant d'un ou plusieurs clusters.

La clavicule

Ayant taclé le problème des bras du personnage, passons à présent à la clavicule et aux épaules. Quel type de mouvement doit-elle accomplir cette partie du corps ?

Au moins, nous devons avoir assez de contrôle pour pouvoir lever les épaules. Cette capacité donne déjà beaucoup de nouvelles nuances aux gestes du personnage. Et, normalement, c'est tout ce que nos épaules peuvent faire : se lever et descendre, et aller vers l'avant ou l'arrière pour accentuer la courbature de la colonne. Pour ce faire, il suffirait d'ajouter un contrôleur pour effectuer des rotations sur les clavicules, afin de faire monter ou descendre les épaules.

Mais, pour quoi ne pas aller un peu plus loin ? On peut, en effet, manipuler les épaules autrement. Pour quoi ne pas donner la possibilité à l'animateur de déplacer les épaules directement ? De cette façon là, en lieu d'être soumis aux possibilités limitées du contrôle de la clavicule par FK, l'animateur aura entière liberté pour poser les épaules du personnage.

Cet effet eut être accompli grâce au placement d'un IK Handle de type Single Chain allant de la clavicule jusqu'au joint de l'épaule. Evidemment, une fois l'IK créé il faut créer aussi un contrôleur pour le diriger.

Les jambes et les pieds

Lors de la construction des jambes et des pieds nous devons garder dans l'esprit que, comme l'indique Jason Schleifer, les jambes ne sont pas uniquement un moyen de locomotion, mais qu'elles transmettent un sentiment de force et d'intention [1]. Pour ceci, une grande partie de notre attention doit être dirigée à la création d'un rig versatile et facile à utiliser pour contrôler les différents mouvements nécessaires pour animer une marche, comme le foot roll, la suite des mouvements d'un pied à chaque pas, à partir du moment que le talon touche le sol jusqu'à ce que les orteils le quittent.

En ce qui concerne la jambe en soi, on verra que la façon de procéder est très proche à celle qu'on a vu pour le bras, sauf pour les personnages qui devront réaliser des actions extrêmes, pour lesquels un système particulier, souvent dénommé No Flip Pole Vector, devra être adopté afin d'éviter les rotations subites de la jambe.

Le rig des jambes doit alors remplir les conditions suivantes:

1. Comme pour les bras, les jambes doivent comporter un système de cinématique inverse. Et même si ce type de contrôle est celui qui sera le plus souvent privilégié, un mode FK doit aussi être conçu, aussi bien qu'un système qui permette de passer d'un mode à l'autre.
2. Afin de permettre à l'animateur de donner l'impression du poids d'un personnage, un bon système de foot roll et foot pivot doit être inclus.
3. Selon le type de mouvement réalisé par le personnage, un No Flip Pole Vector peut être requis.

Commençons alors par créer les trois chaînes de joints, comme on a fait pour le bras. Bien que la structure des os de la jambe soit similaire à celle des bras, la rotation à niveau de la partie basse de la jambe reste très légère, et souvent on peut se passer d'un système comme celui créé pour l'avant bras. Cependant, ceci dépend du modèle et des mouvements requis de sa part. En ce qui nous concerne, nous procèderons à rigger la jambe avec une chaîne de trois joints, comme celle construite lors de la création du squelette.

La création du système IK/FK sur la jambe est alors beaucoup plus simple que pour le cas du bras, car nous n'avons pas besoin d'une chaîne de joints "guide" simplifiée pour la création de l'IK. Le setup terminé de la jambe serait alors comme suit:

Toutes les manipulations étant analogiques à celles que nous avons faites pour le cas des bras, nous ne nous attarderons pas sur les détails de la création de ce système.

Par contre, ce qui mérite une attention particulière dans le cas des jambes est le Pole Vector. Pour la plupart des personnages, un Pole Vector créé de façon similaire à celui que nous avons créé pour l'IK des bras suffit largement. Cependant, certains personnages doivent réaliser des actions qui requièrent que les pieds soient levés au dessus du niveau des genoux, ce qui peut engendrer des déformations extrêmes à niveau de la jambe. Ceci arrive lors que le genou croise le plan sur lequel se situe l'élément utilisé comme Pole Vector.

Pour ce genre de situations, on nécessite un No Flip Pole Vector, un Pole Vector à l'épreuve des rotations abruptes quand les jambes sont déformées de façon extrême. L'idée derrière ce type de setup, que l'on peut trouver dans [7], est de créer un Pole Vector qui ne se trouve pas sur le même plan que la jambe, mais sur un plan parallèle situé à côté de celui ci.

Ayant construit une jambe qui remplit nos conditions de mouvement, passons à présent à la création du rig des pieds.

Les pieds

Pour créer un setup pour les pieds qui nous permette de manipuler notre personnage facilement lors de la création d'un cycle de marche, entre autres, il nous faut un système de Reverse Foot Lock, qui nous donne la possibilité manipuler les joints des pieds de manière inverse, de façon à ce que le talon puisse être levée partant du point de pivot des orteils, aussi bien que de faire pivoter le pied sur le talon.

Pour ce faire, il existe beaucoup de manières différentes de procéder. La première, est de créer deux IK Handles de type SC sur les joints des pieds: le premier allant du joint de la cheville jusqu'au ball, la demi-pointe du pied, et le deuxième allant du ball jusqu'au bout des orteils. Ensuite nous devons créer une chaîne de joints auxilière, construite dans le sens inverse à celle des pieds: d'abord un joint pour le talon, créé là où devra pivoter le pied, un autre placé sur le joint correspondant aux orteils, un autre sur le ball, pour finir sur le joint de la cheville.

Chacun des IK Handles créés sur les pieds seront parentés sous les joints de ces contrôleurs correspondants, qui seront, à leur tour, dirigés par des courbes de contrôle [8]. Même si avec cette

configuration nous pouvons déjà poser les pieds de manière appropriée pour un foot roll, une de nos conditions d'animation est que ce contrôle soit transparent et facilement accessible. Nous devons alors créer un attribut sur chacun des contrôleurs des pieds pour diriger le foot roll, allant de -10 à 10, par exemple. Ensuite il suffit de associer cet attribut aux rotations des contrôleurs du pied, grâce à des Set Driven Keys, de façon à ce que le pied repose sur le talon quand l'attribut foot roll soit égale à -10, et sur les orteils quand il est égale à 10. Pour voir la création de ce type de contrôle plus en détail, je vous conseille [8].

Une autre possibilité pour créer le reverse foot nous vient du tutoriel The Puppet Rig de Gnomon. Dans ce tutoriel, le reverse foot est créé exclusivement avec des IK Handles et des contraintes, sans faire appel à une chaîne de joints additionnelle[6]. Je trouve la simplicité de cette méthode très attirante, mais il faut savoir que le comportement généré par ce type de rig peut être un peu limité, et n'est donc, pas adapté aux personnages qui ont des besoins spéciaux, comme ceux qui ont des pieds qui doivent se déformer de façon complexe.

La dernière technique que je voudrais présenter se trouve dans le tutoriel de Fahrenheit: Rigging for Feature Animation. Dans ce tutoriel deux rigs différents sont proposés: un pour des personnages avec des orteils qui doivent être riggés de façon indépendante, et un autre pour des personnages avec des pieds plus simples [7].

Les jambes et pieds des personnages du court

Pour le court "L'Armoire", c'est ce dernier type de reverse foot que j'ai décidé d'adopter. J'ai me suis inspiré de la version plus simple pour les pieds de la fille, et de la deuxième méthode pour les pieds du monstre, comme on a vu dans la section de la création du squelette.

Le Rigging du Visage

Dans cette section, je vous présenterai les rigs de visage faits pour les personnages du court « L'Armoire ».

Un parcours complet du rigging du visage nécessiterait au moins un autre mémoire. Ici donc, je vous présenterai simplement les grandes lignes des contrôles que j'ai inclus sur mes modèles.

La mâchoire

Une des premières décisions que l'on doit prendre avant de rigger le visage, concerne la façon dont on va gérer l'ouverture de la bouche. De façon générale, le rigging de la mâchoire peut être faite comme pour n'importe quelle expression, en utilisant des Blend Shapes. Toutefois, cette approche à une limitation principale: Les Blend Shapes sont interpolés de façon linéaire.

En utilisant deux formes de base pour le visage, une avec la bouche fermée, et une autre avec la bouche ouverte, les vertex du modèle iront en ligne droite d'une position à l'autre. Ceci peut être géré en utilisant des formes intermédiaires pour "forcer" en quelque sorte une interpolation plus arrondie, mais en général, au moins pour les personnages qui ont une grande bouche, il est préférable de rigger la mâchoire en utilisant des joints.

Pour ce faire, nous devons placer deux joints additionnels sur notre squelette: un premier, attaché à la base de la tête, qui servira comme pivot au joint de la mâchoire, et un deuxième pour la mâchoire elle même. Si ce joint est créé après avoir fait un skinning sur le corps, celui ci doit être modifié pour que les déformations du visage soient correctes. Dans Stop Staring, Jason Osipa va encore plus loin, en créant des sous joints, qui suivent les mouvements du joint principale de la mâchoire dans des différents dégrées, en fonction à leur distance à celui-ci [9].

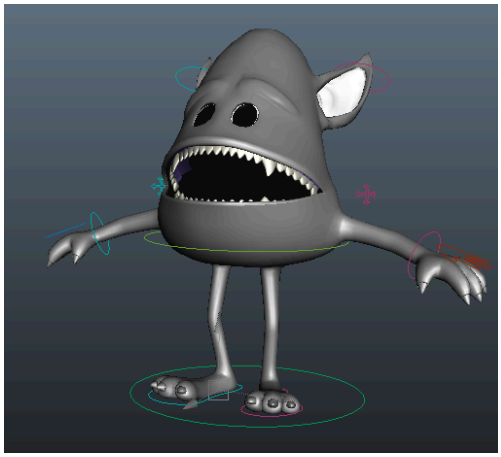
Avec un joint pour la mâchoire, nous pouvons être sûrs que les vertex suivront un chemin arrondi lors que le personnage ouvre et ferme la bouche.

Sur les personnages du court "L'Armoire" j'ai décidé d'adopter cette deuxième technique, sans toute fois avoir eu recours aux sous joints de la mâchoire.

Dans le cas du monstre, j'ai légèrement modifié cette stratégie, pour faire en sorte que lorsque la

bouche du monstre s'ouvrait, ce soit le haut de sa tête qui se déplace, et non pas le bas de celle ci.

La mâchoire de la fille.



La mâchoire du monstre.

Blend Shapes vs. joints

Un des grands débats qui existe dans le monde du rigging tourne autour de l'utilisation des Blend Shapes, en lieu d'un rig basé sur des joints, ou vice-versa. Les adeptes des Blend Shapes, d'un côté, soutiennent que cette forme d'approcher la création des expressions est suffisamment simple à mettre en place, et donne des meilleurs résultats avec un minimum d'effort de la part de l'animateur.

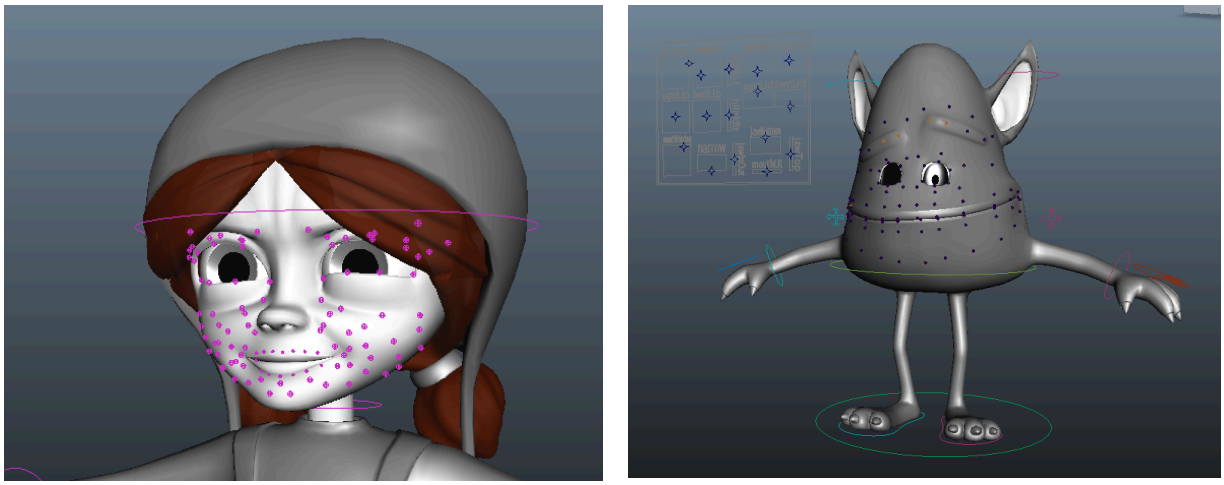
A cela, les adeptes de rigging basé sur des joints répondent que le nombre d'expressions possibles avec des Blend Shapes est limité, donne des résultats trop prévisibles en animation, et prive l'animateur d'une grande partie de sa liberté créative.

Personnellement, je trouve que les deux arguments sont pertinents. Toutefois, la qualité des résultats obtenus avec les Blend Shapes dépend largement de la façon dont ceux ci seront conçus. La construction des Blend Shapes en se basent sur des formes appelées des "unités d'expressions", formes simples qui peuvent être combinées pour former des expressions plus complexes donne beaucoup plus de possibilités d'expression que les Blend Shapes de base, pour lesquelles on créerait une forme par expression, du genre "Heureux", "Triste", etc. [4] [9]. De l'autre côté, je trouve la gestion des expressions avec des joints assez pénible, et, en tant qu'animatrice, je n'aimerais pas travailler avec ce type de rig, même s'il me donne plus de liberté lors de la création des expressions.

Heureusement, il existe un moyen d'avoir les deux choses en même temps: Avoir un rigging à partir des Blend Shapes, créés en suivant des "unités d'expression", qui peuvent être modifiées grâce à une

deuxième couche de contrôle venant d'un rig fait partir des joints. Cette méthode est exposée dans Stop Staring, de Jason Osipa, un livre que je conseille vivement à ceux qui s'intéressent au rigging du visage.

C'est cette option que j'ai choisi pour mes personnages.



"Blend Shapes" avec des contrôleurs locaux.

Les yeux

Les yeux sont, pour moi, l'exemple parfait d'un rig qui, à la base, peut être fait de façon très simple, mais qui peut être modifié pour générer des comportements assez complexes.

Pour clarifier cette idée, voyons ce que comporte un rig simple pour les yeux. A la base, ce que nous voulons des yeux est de pouvoir les orienter vers un point précis dans l'espace. Pour ce faire, il y a plusieurs solutions. Par exemple, nous pouvons créer des joints pour les yeux, parentés directement au joint de la base de la tête. Ensuite, nous pourrions parenter la géométrie des yeux sous ces deux joints (ou trois, ou quatre, selon le personnage :)).

Cependant, ceci n'est pas une très bonne pratique. En parentant la géométrie sous les joints des yeux nous aurions de la géométrie dans le groupe du squelette, ce qui n'est pas très propre. En plus, si on avait besoin de sélectionner la géométrie des joints dans l'Outliner, il faudrait parcourir toute la hiérarchie du squelette pour les trouver. Pour éviter ceci, il est conseillé soit d'utiliser un « Parent constraint », soit de skinner la géométrie de chacun des yeux sous son joint correspondant.

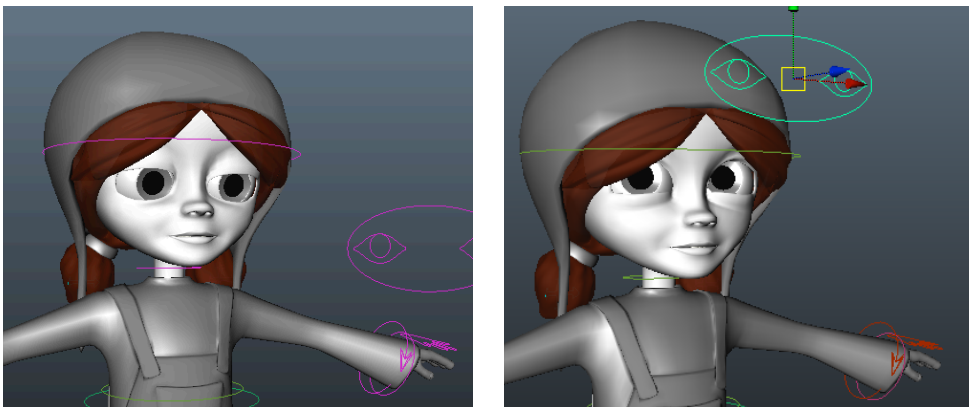
Ensuite, nous n'aurions qu'à créer un contrôleur pour les yeux, un objet qui déterminera la direction du regard, et créer un « aim constraint » du contrôleur sur les joints des yeux, et le tour est joué.

Cependant, ce système est beaucoup trop limité. Et si jamais on voulait que les yeux regardent dans des directions différentes? Pour ce faire alors, on créera un contrôleur par œil, et on créera un contrôleur général sous lequel chacun des contrôleurs indépendants sera parenté. De cette façon, si on voulait contrôler les deux yeux en même temps, on manipulerait le contrôleur général, mais on garderait la possibilité de changer la direction de chacun des joints de manière indépendante.

Et si on voulait que les yeux regardent toujours devant? Simple, on parente le contrôleur des yeux sous celui de la tête. Et si parfois on voulait que le mouvement des yeux ne suive pas la tête? Simple, on ferait comme pour les broken hierarchies, et en lieu de parenter le contrôleur des yeux on l'associerait au contrôleur de la tête avec une contrainte, que l'on pourrait désactiver si besoin. Et si on voulait que les paupières se déforment suivant la direction du regard? Et qu'elles s'ouvrent ou se ferment pour garder toujours la même position relative à la pupille (ce qui se passe dans la vraie vie)?

Les possibilités sont inépuisables.... on peut passer d'un rig très simple, à un très sophistiqué. Pour voir un exemple d'un rig des yeux avancé, je vous conseille encore [9].

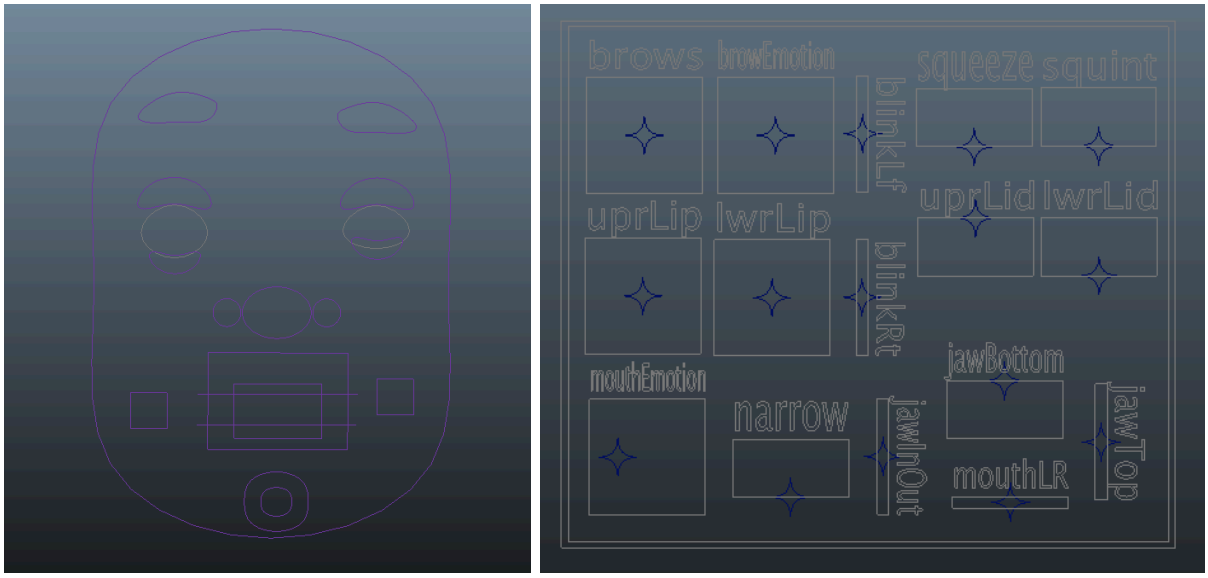
Pour les personnages du court j'ai adopté un système assez complexe pour les yeux, avec tracking des pupilles et déformation des paupières selon la position des yeux.



Le "tracking" des yeux.

Interface Faciale

Indépendament de la méthode choisie, il est essentiel d'avoir une interface qui facilite l'animation du visage. Cette interface est un ensemble de contrôleurs liés à chacun des Blend Shapes disponibles. On a plusieurs options pour créer cette interface, soit avec une représentation du visage lui même, soit avec des sliders pour contrôler chaque partie du visage.



Deux exemples d'interface faciale.

Conclusion

J'espère qu'après la lecture de ce mémoire vous ayez une vue d'ensemble assez claire sur tout le processus de la création d'un rig, de la préparation jusqu'à l'organisation de la scène. Bien sûr, je ne prétends pas aborder tous les sujets qui entourent cette problématique, mais je vous encourage à aller un peu plus loin, en regardant les sources sur lesquelles je me suis basée pour la rédaction de ce texte.

Personnellement, l'écriture de ce mémoire m'a aidé à comprendre les raisons qui motivent chacune des manipulations faites pendant la création d'un rig. J'ai eu l'occasion de chercher les motivations qui se cachaient derrière certaines techniques que j'avais adoptées en tant qu'automatismes, et ceci a rendu le processus de rigging plus transparent et agréable pour moi.

À présent, le plus important à retenir, c'est le rôle que jouent la rigueur et la bonne compréhension des techniques à utiliser, même dans la création des rigs les plus simples. Prenez bien note de la façon dont vous avez construit chacune des parties de votre rig, pour vous assurer que l'ensemble aura un comportement correcte, et pour régler les éventuels problèmes. Testez tout ce que vous faites, et n'ayez pas peur d'adopter des nouvelles techniques, et d'aller plus loin que ce qui vous est proposé dans les livres ou les tutoriels. Les meilleurs rigs naissent de l'exploration et du mariage de plusieurs techniques existantes, lancez-vous donc, et bon rigging!

Bibliographie

[1] SCHLEIFER, Jason. Animator Friendy Rigging. Maya MasterClass Seminar. Siggraph, 2006.

[2] THURIOT, Paul. Hyper-Real Body Setup. Maya MasterClass Seminar. Siggraph, 2004.

[3] GOULD, David. Complete Maya Programming: An Extensive Guide to MEL and C++ API (The Morgan Kaufmann Series in Computer Graphics) Morgan Kaufmann, 2003.

[4] RITCHIE, Kiaran, Jake CALLERY, and Karim Biri. The Art of Rigging. N.p.: CGTOOLKIT Press, 2007.

[7] ALLEN, Eric, and Kelly L. MURDOCK. Body Language: Advanced 3D Character Rigging. 5th ed. Indianapolis, Indiana: Wiley, 2008.

[8] OSIPA, Jason. Stop Staring: Facial Modeling and Animation Done Right. 3rd ed. Indianapolis, Indiana: Sybex, 2010.

Tutoriels:

[5] Character Rigging: The Puppet Rig. Maya Rigging Techniques with Carlo Sansonetti. The Gnomon Workshop, 2007.f

[6] Rigging for Feature Animation. Fahrenheit Digital, 2006.