

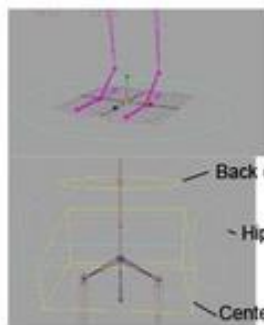
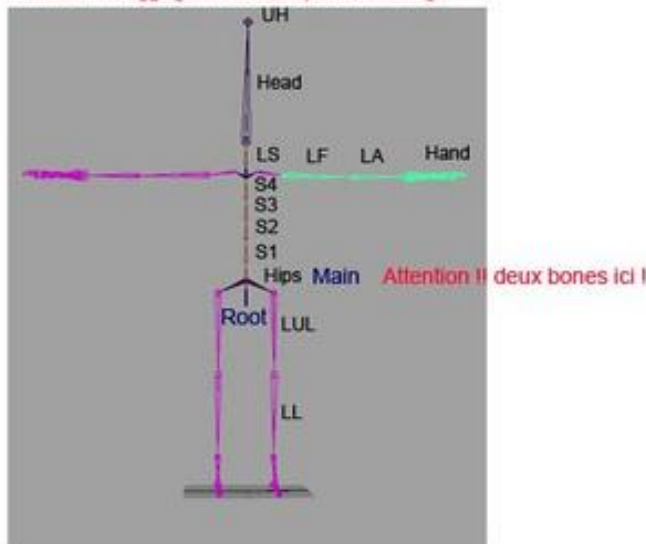
# Faire un Rigging complet

Par Ed (eleona25@yahoo.fr)

Dans la construction du squelette, pour que l'animation fonctionne, il faut faire très attention à la construction au niveau de l'hips.

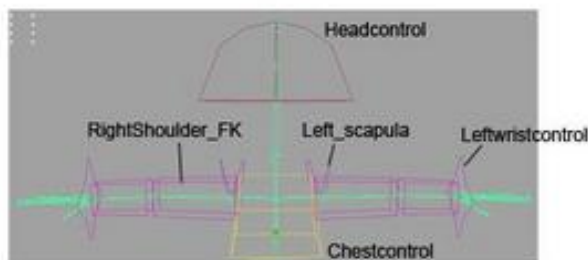
Rigging :: voici le squelette du départ. Les bones son nommés comme dans le topic bones.

Attention !! le rigging doit être fait après le skinning !!



On commence par créer le **global control** à la base des pieds en n'oubliant pas de faire un freeze transformation

On crée ensuite les 3 contrôleurs suivants. On snappe le **Back\_control** sur le **spine2** (ou quelque part au milieu du ventre) et le **center\_o\_gravity** sur l'**hips**

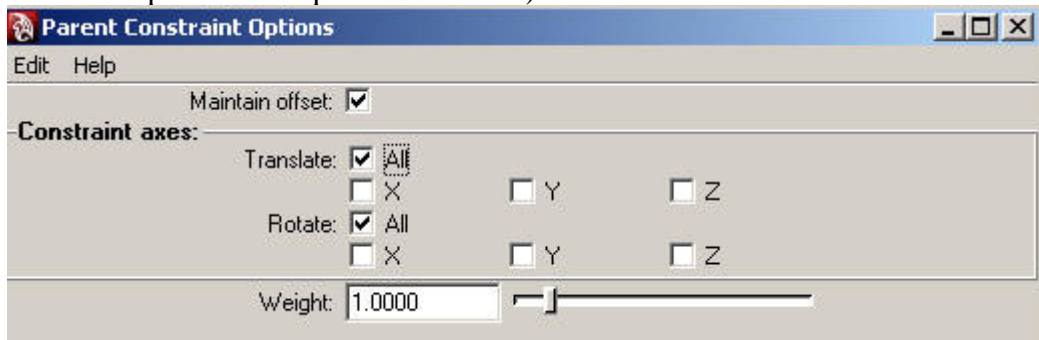


On crée ensuite les autres contrôleurs comme ci-dessus

On crée d'abord le joint **Root** à la base de la colonne vertébrale, un peu en dessous du bassin même, ensuite en remontant, on crée le **main** pile dans le creux du bassin, puis l'**hips**, a peu près au même endroit. On valide en appuyant sur entrée. On recrée une nouvelle chaîne de bones à partir du **spine 1**(S1 sur le dessin) et on crée tous les autres bones normalement. On parente ensuite le **S1** au **Main** pour relier les deux chaînes de bones et on obtient ensuite la colonne vertébrale.

On parente ensuite l'**hips** vers le **global control** en maintenant **P**.

Ensuite on parente chaque courbe au bone correspondant, avec l'aide des contraintes. Pour le **center\_o\_gravity**, on sélectionne la courbe, puis le bone et dans **constrain =>Parent** (on ouvre les options en cliquant sur le carré)

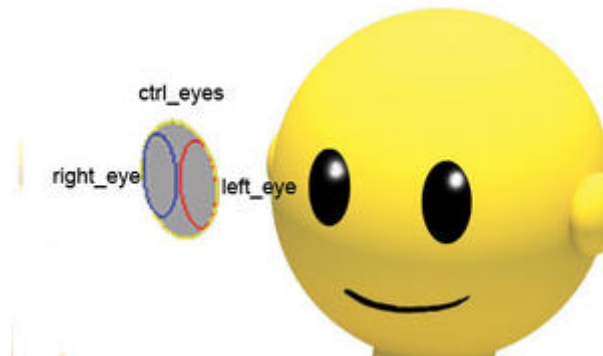


On fait pareil pour l'**hipsway**, le **backcontrol** et le **chestcontrol**. Pour ce qui est des bras, on ne parente que en rotation.

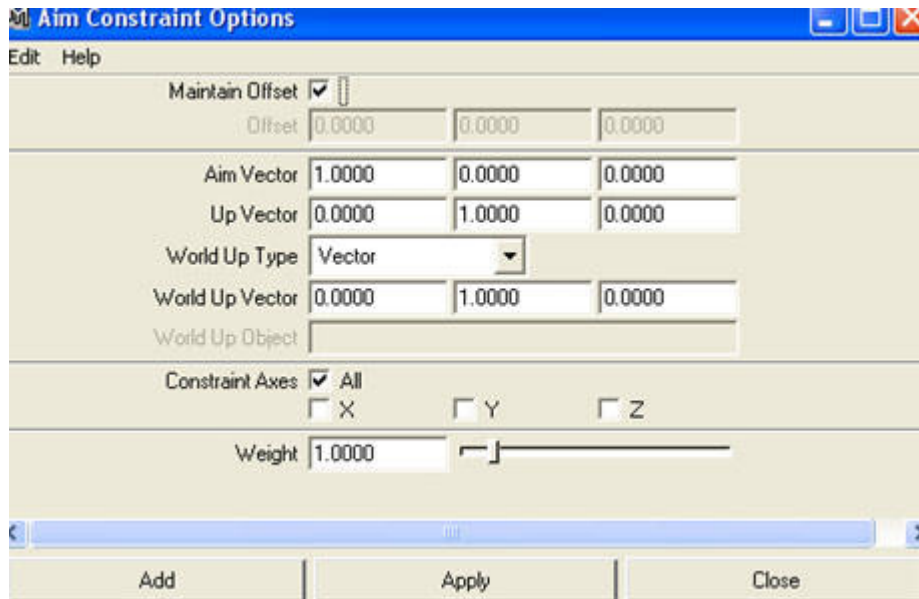
Ensuite, il faut parenter les courbes entre elles. On parente le **headcontrol** au **chestcontrol**, l'avant bras au bras (ou **rightshoulder\_FK**), lui-même est parenté au **chestcontrol**. Le **chestcontrol** est ensuite parenté au **backcontrol** qui lui-même est parenté au **center\_o\_gravity**. L'**hipsway** est parenté au **center\_o\_gravity** également.

Voici les contrôleurs pour les yeux. On les crée avec des courbes et surtout, il ne faut pas oublier de faire un **freeze transformation**.

Ne pas oublier de parenter les yeux au bones correspondant à l'intérieur de la tête ou sinon, les parenter au bone de la tête. Ici, j'ai parenté les yeux à leurs bones respectifs.



Sélectionner ensuite le **right\_eye** sur le contrôleur, puis le bone correspondant à l'oeil droit et appliquer une **constrain => aim**. On fait pareil avec l'oeil gauche. Puis parenter les deux contrôleurs au **ctrl\_eyes**.



Ainsi si on déplace le contrôleur des yeux, ceux-ci suivent. On patente ensuite le contrôleur des yeux au **center\_o\_gravity**. Pour que le **ctrl\_eyes** reste toujours face aux yeux du perso on lui applique une **constrain =>aim**. On sélectionne le bone des yeux puis le **ctrl\_eyes** pour appliquer la contrainte.

## Le rig du bras

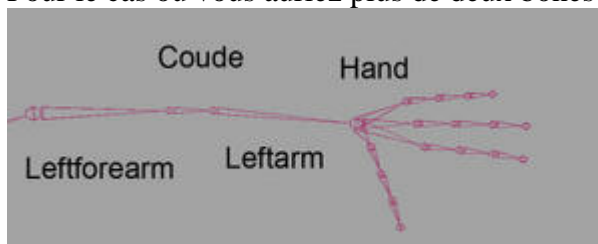
### 1) Le bras est simplement composé de deux bones.

L'Ik fonctionne bien quand on relie deux bones seulement. On crée donc un **skeleton =>IkHandle tool** du **LF (leftforearm)** au **LA (LeftArm)**. On crée ensuite la courbe de la main pour contrôler tout ça ou **leftWristControl**.

On applique ensuite une **constrain => parent** sur les translate du **leftWristcontrol** à l'IK. Puis une **constrain => parent** du **leftWristcontrol** au **LA** (pour la rotation du poignet) et enfin une autre contrainte parent du **leftwristcontrol** au bone **Hand**, juste en *rotate* (pour que les bras suivent le corps si on tire dessus) ou a la fois sur les rotations et les translations (pour que les mains s'étirent)

### 2) Le bras est composé de 3 bones

Pour le cas où vous auriez plus de deux bones seulement dans votre bras, comme ici :

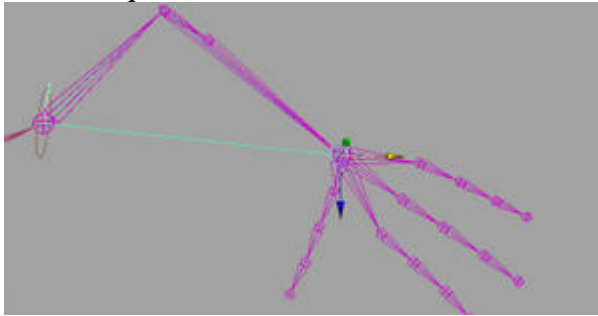


Il faut dupliquer le bras original. Dans l'outliner, on peut voir cette copie qui vient d'être créée. On supprime le hand qui lui est rattaché. On renomme les bones dupliqués. Ici on va l'appeler **controlArm1** et **controlArm2**.

(il faut qu'il n'y en ai que 2 pour que l'IK fonctionne) Donc on snappe le **controlArm2** sur le **Hand**.

Ensuite, on crée un ikHandle du **controlArm1** au **controlArm2**. Ensuite, on selectionne le controlArm1 puis le bone du bras : **left\_forearm** et on lui applique une contrainte => **constrain =>orient**. On coche **maintain Offset** et constraint Axes. On fait pareil avec le **controlArm2** sur le coude et le **controlArm2** sur le **leftArm**. Maintenant, si on déplace l'ik, le bras suit.

Voilà ce qu'on obtient :

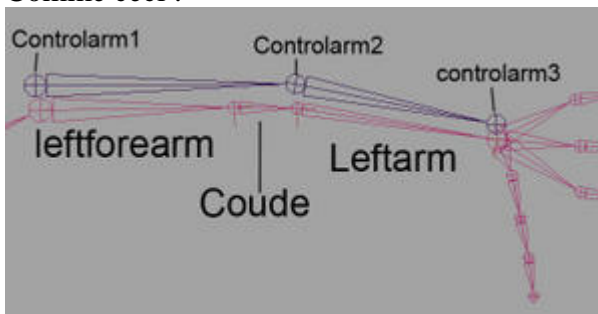


### 3) Le bras est composé de 3 bones et on veut que chaque bone se plie

On crée un Ik Handle en Scsolver du **leftforearm** au **coude**, puis du **coude** au **leftarm** et enfin du **leftarm** a l'**hand**.

On crée ensuite des bones en vue de dessus juste derrière le bras que l'on nomme en partant de l'épaule **controlArm1**, **controlArm2** et **controlArm3**.

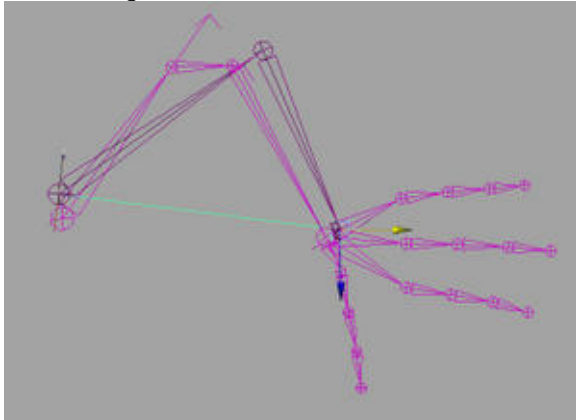
Comme ceci :



On parente ensuite les ik aux nouveaux bones leur correspondant. Et les deux IKs des bones du milieu du bras, on les parente tous les deux au **controlArm2**.

On crée ensuite un IK handle RP solver du **controlArm1** au **controlArm3**. Et si on bouge cet Ik, tout le bras suit.

Voilà ce que l'on obtient :



Si vous avez quelques soucis lors du parentage de vos bones et qu'il vous met un message d'erreur comme quoi il y a déjà une connection ou bien que c'est déjà un enfant du parent, c'est parce que tout à l'heure, on a connecté les contrôleurs aux bones. Il faudra briser les connections pour pouvoir attacher les autres bones qui guideront le bras.

Ensuite, il faudra biensûr reparerer les contrôleurs aux bones rajoutés pour les deux derniers exemples. Donc on reselectionne le **LeftShoulderIk** puis le **controlarm1** et on lui ajoute de nouveau une **constrain => parent** que sur les rotates. On contraint ensuite le **leftElbonwIk** au **controlArm2**.

Attention : Il faut sélectionner le controlarm1 et le leftShoulderFK (le controleur) et leur appliquer une **contrainte => parent** afin que les bones rajoutés suivent toujours le bras.

**Attention !! pour le contrôleur de la main, le *leftWristControl* on lui ajoute une **constrain=> parent** sur les translates mais avec L'Ik. Puis on fait une **constrain=> parent** du *leftwristcontrol* au *leftarm* (pour la rotation du poignet) puis du *leftwristcontrol* au bone hand juste en *rotate*.**

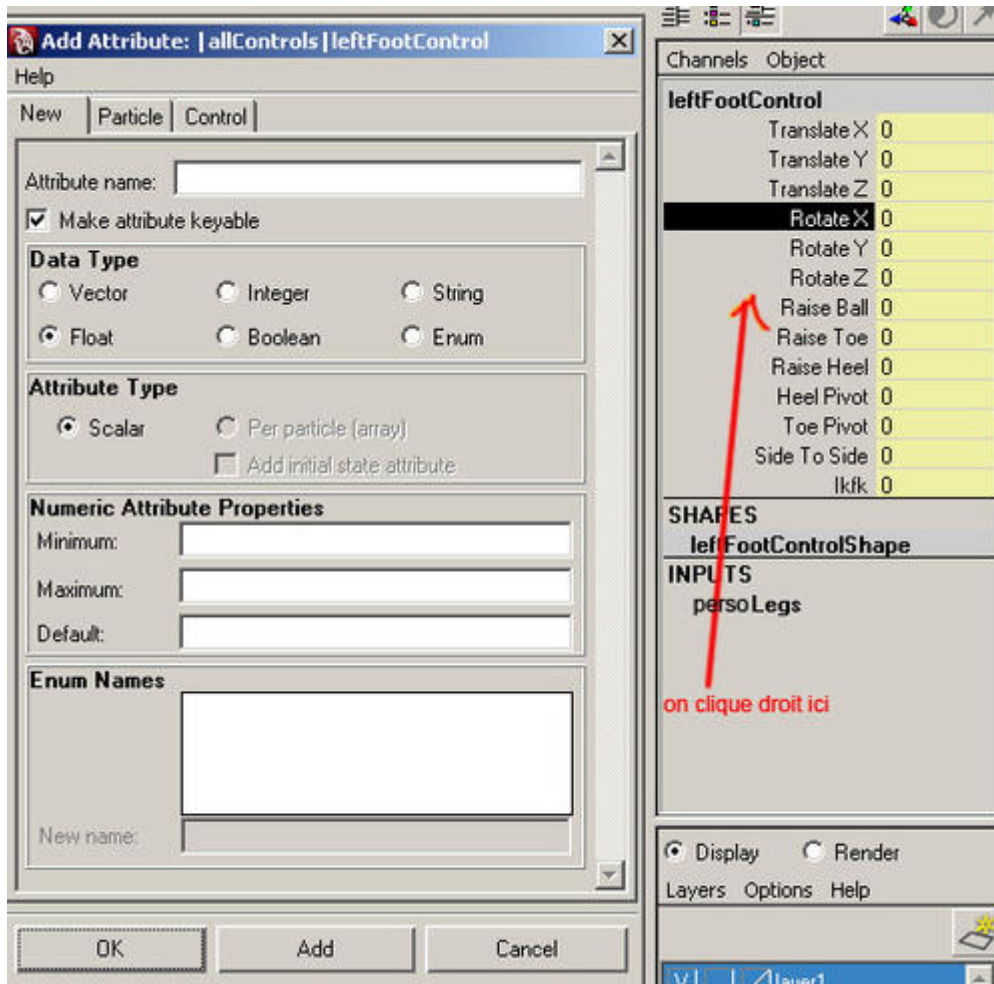
**Par contre, si on paramètre le leftwristcontrol de cette manière, on ne peut plus utiliser des contrôleurs des bras. Personnellement, je préfère pouvoir intervenir directement sur les bras que de tout bouger avec la main. Alors je rig le bras comme je l'ai expliqué avant cet encadré. Et pour le leftwristcontrol, je le parente simplement dans les rotates avec le bone Hand en oubliant pas ensuite de parenter le leftwristcontrol au bone hand pour qu'il suive la main quand on agit sur un autre controleur du bras.**

## L'épaule

On selectionne le **leftShoulder** et le **leftScapula** et on leur applique une **constrain =>Orient**. Puis on selectionne le **leftShoulder** et le **leftScapula** et on leur applique une **constrain=> point**.

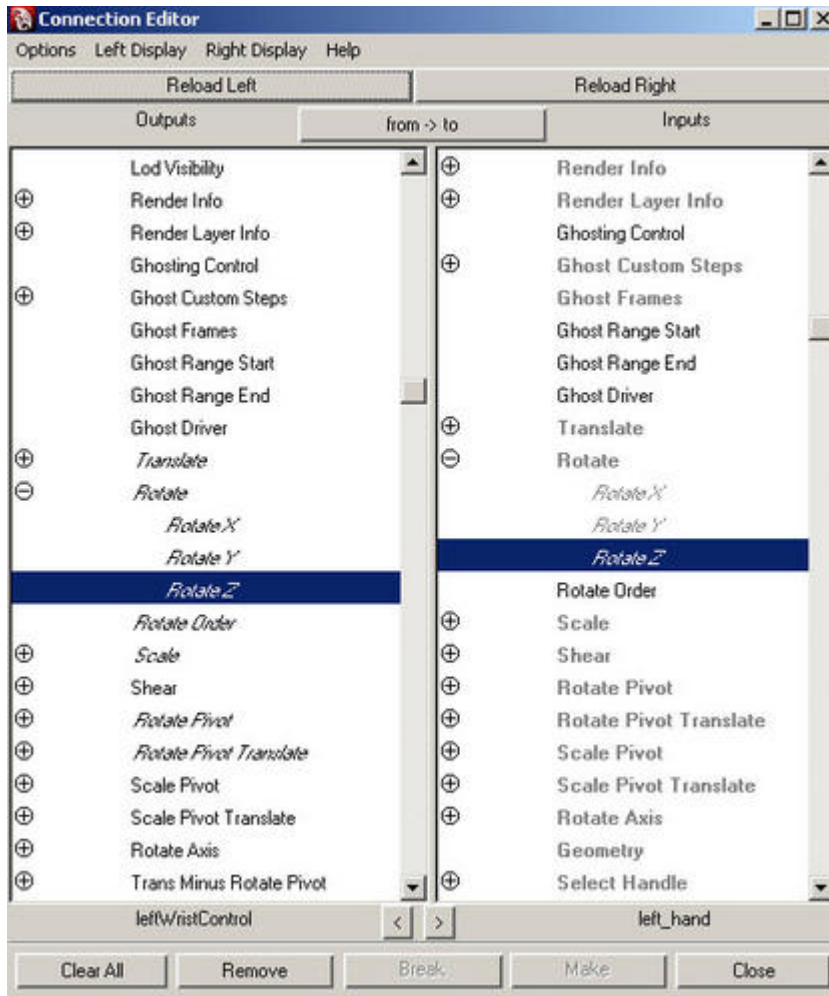
## Le rig de la main

Tout se joue sur le leftWristControl. Nous allons tout d'abord rajouter tous les attributs dont nous aurons besoin pour l'animation. Pour cela, on clique droit dans la fenêtre des attributs et on **add attributes**.



On ajoute donc : **Thumb** (qui nous servira de séparation avec le reste, on le locke pour qu'il apparaisse en grisé), **Thumb Roll** (min -10 max10), **Thumb Root Bend** (idem), **Thumb End Bend** (min0 max10), **IndexThruPinky** (en grisé), **Index** (min-10 max10), **Mid** (idem), **Pinky** (idem), **Move fingers** (grisé), **Spread** (min-10 max10), **Fist** (min0 max10).

Pour connecter les attributs avec les bones, il suffit de faire un clic droit dans la **channelBox** et d'aller sur connection editor (mais cette technique n'est pas encore très au point. Rien ne vaut les set driven keys ):



Détail de chaque attribut :

**RotateX** : on connecte a la rotate X du bone hand.

**RotateY** : " y "

**RotateZ**: " Z "

**Thumb Roll** : On le connecte à la rotation Z du premier bone du pouce.

**Thumb Root Bend** : On le connecte à la rotation X du premier bone du pouce, ainsi qu'à la rotation X du deuxième bone.

**Thumb End Bend** : On le connecte à la rotation X du dernier bone du pouce.

On fait pareil pour les autres doigts en pensant bien à connecter chaque phalange au contrôleur.

**Spread** : il sert à écarter ou à resserrer tous les doigts de la main. On peut le paramétrer en set driven Key ou bien jouer sur les rotations Z de tous les bones de base des doigts.

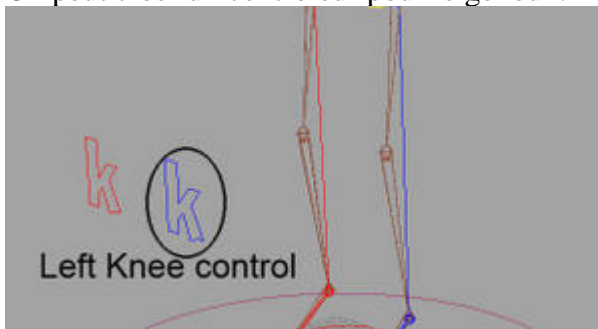
**Fist** : permet de fermer le poing complet. On le paramètre comme le spread mais de façon à ce que le poing se ferme.

## Le rig de la jambe

### 1) La jambe est composée de deux bones

On fonctionne de la même manière que pour le bras. On crée ensuite l'ikHandle du **LUL (LeftUpLeg)** au **LL(leftLeg)**

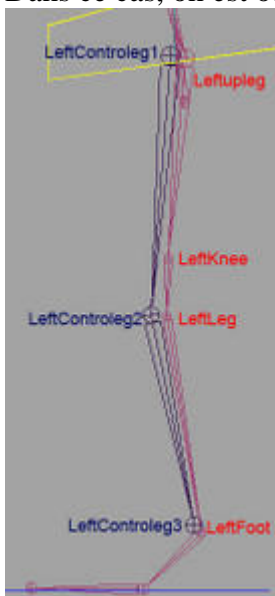
On peut créer un contrôleur pour le genou.



Pour cela, on utilise le **constrain => pole vector**. On sélectionne le **leftKneeControl**, puis l'ik de la jambe et on lui applique la contrainte précédemment citée. On fait ensuite un **freeze transformation** sur le **leftKneeControl**.

### 2) La jambe est composée de plus deux bones

Dans ce cas, on est obligés de rajouter des bones comme pour le bras.



Comme pour le bras, on va créer des IK Handle en Scsolver, du **leftUpLeg** au **leftKnee**, puis du **leftKnee** au **leftLeg**, puis du **leftLeg** au **leftFoot**.

On parente ensuite les Ik au nouveaux bones créés en pensant bien à parenter les deux IK du

genoux sur le **leftcontrolleg2**.

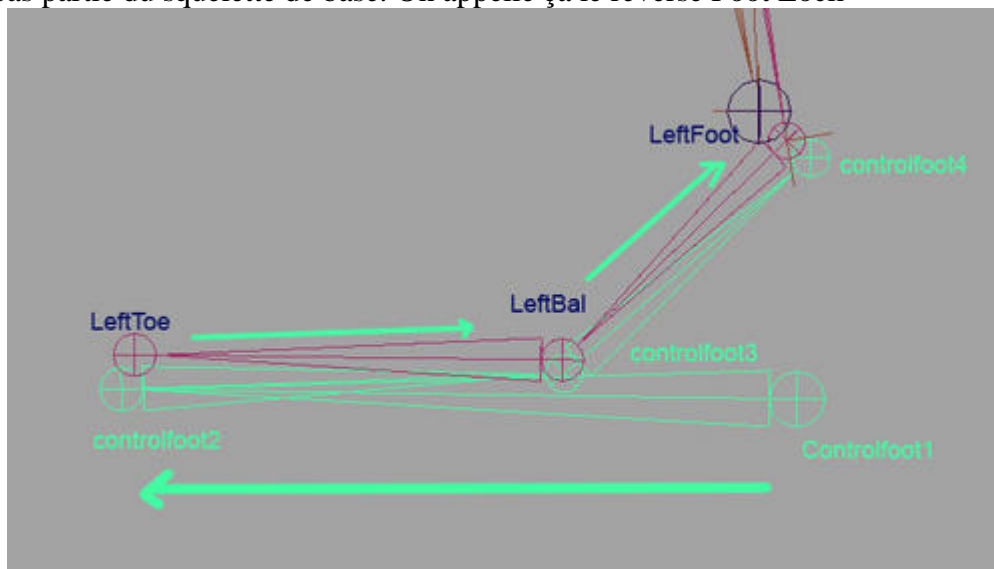
On crée ensuite un IK Handle RP scolver du **leftcontrolleg1** au **leftcontrolleg3**.

On peut biensûr également créer un pole vector pour le genoux, de la même manière que pour le cas présenté en 1.

Attention !! pour que les bones rajoutés suivent les bones du squelette quand on va les déplacer, il faut sélectionner le bone de la jambe **LeftUpLeg** par exemple puis le **LeftControlLeg** et leur appliquer une **constrain => parent**.

### Le reverse Foot Lock

Pour animer le pied correctement, on rajoute quelques bones supplémentaires, qui ne feront pas partie du squelette de base. On appelle ça le reverse Foot Lock



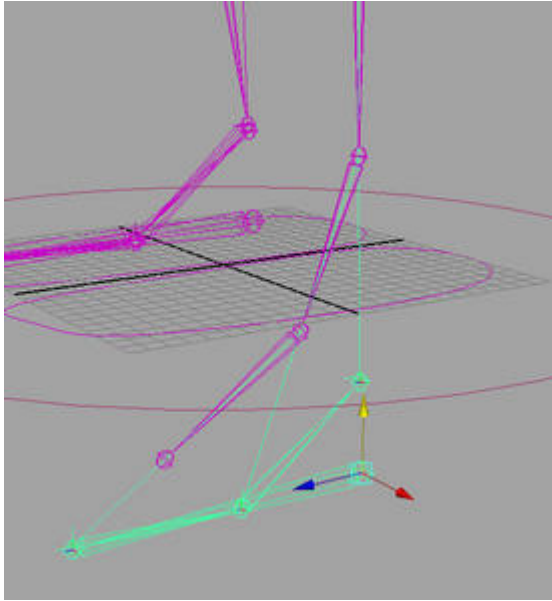
On snape bien les bones sur les bones du pied.

On crée ensuite un Ik Handle Ssolver du **leftFoot** au **LeftBall** et du **left Ball** au **leftToe**.

On parente ensuite l'ik de la jambe (que l'on a créé précédement) au **controlFoot4**.

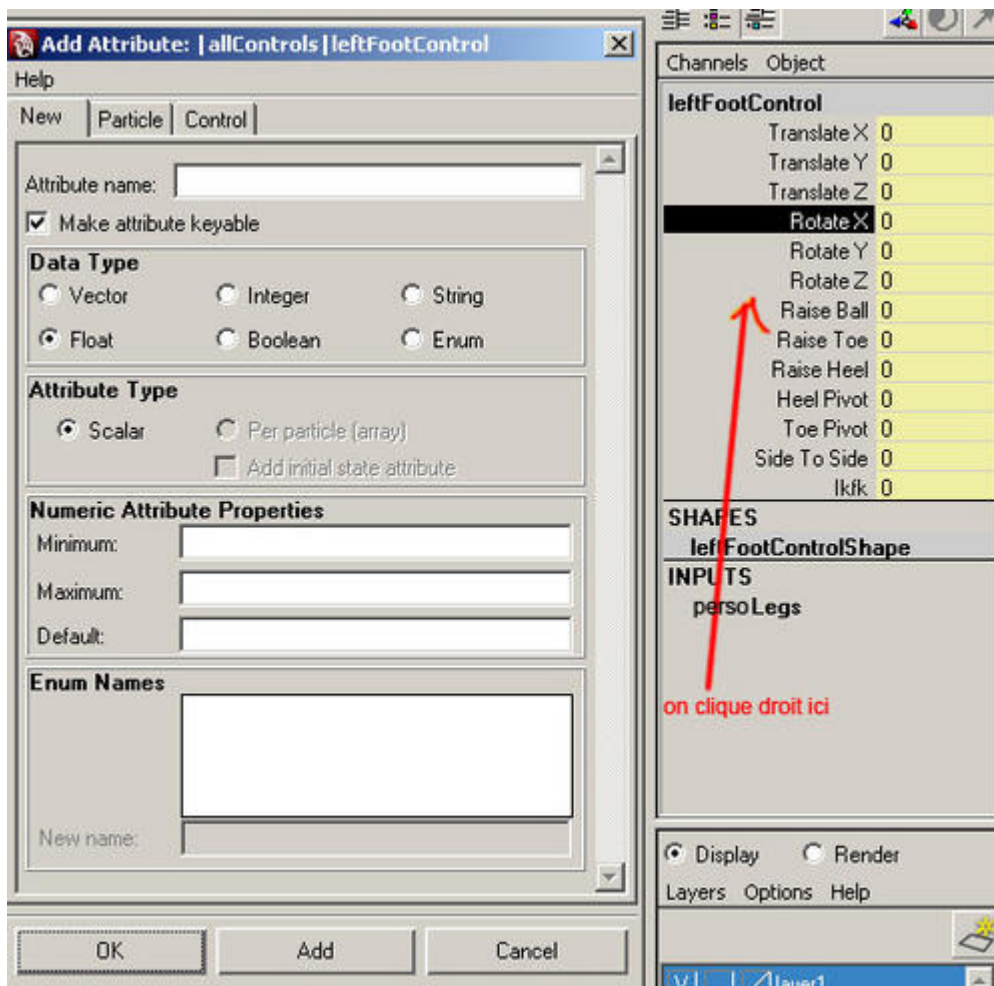
On parente ensuite l'ik du **leftBall** au **controlFoot3** et pour finir, on parente l'ik du **leftToe** au **controlfoot2**.

Ainsi, si on bouge le controlFoot1 le pied et la jambe suivent.



On parente ensuite les bones rajoutés au leftFootControl qui est la courbe au sol de contrôle du pied.

On peut ensuite lui ajouter des attributs pour l'animation du pied et de la jambe. Pour ajouter un attribut d'animation, on clique droit dans le panneau de gauche et on fait **add attribute**.



On peut ajouter donc des nouveaux attributs. Ici on va rajouter : **Raise Ball**. Le minimum de cet attribut est 0, il n'y a pas de maximum et la valeur de départ est 0. On rajoute aussi **Raise Toe** (même paramètres que précédemment), **Raise Hell** (on lui entre une valeur maximum), **Heel Pivot** (ni max ni min), **Toes Pivot** (idem), **Side to side**.

On peut ensuite paramétrer les mouvements prédéfinis et arrêtés avec des setdrivenkeys et pour les autres, il s'agit de connecter les attributs aux bones ou ik correspondants.

#### Définition des nouveaux attributs :

**Raise Ball** : Il s'agit du lever de talon. La valeur ne peut être que positive. La connection se fait sur le **controlfoot3** en *rotate Z*.

**Raise Toe** : il s'agit du pied qui se lève mais en gardant toutefois le bout du pied au sol. La valeur ne peut être que positive elle aussi. La connection se fait au niveau du bone **controlfoot2** *rotation Z*, celui qui correspond au Toe du vrai squelette. Ainsi, il entraîne également le reste du pied.

**Raise Hell** : Il s'agit du pied qui se lève mais dans l'autre sens cette fois-ci, en maintenant le talon au sol. La valeur de base est 0 et c'est la valeur minimum. La connection se fait sur le **controlfoot1** en *rotation Z* également. Il entraîne ainsi le reste du pied.

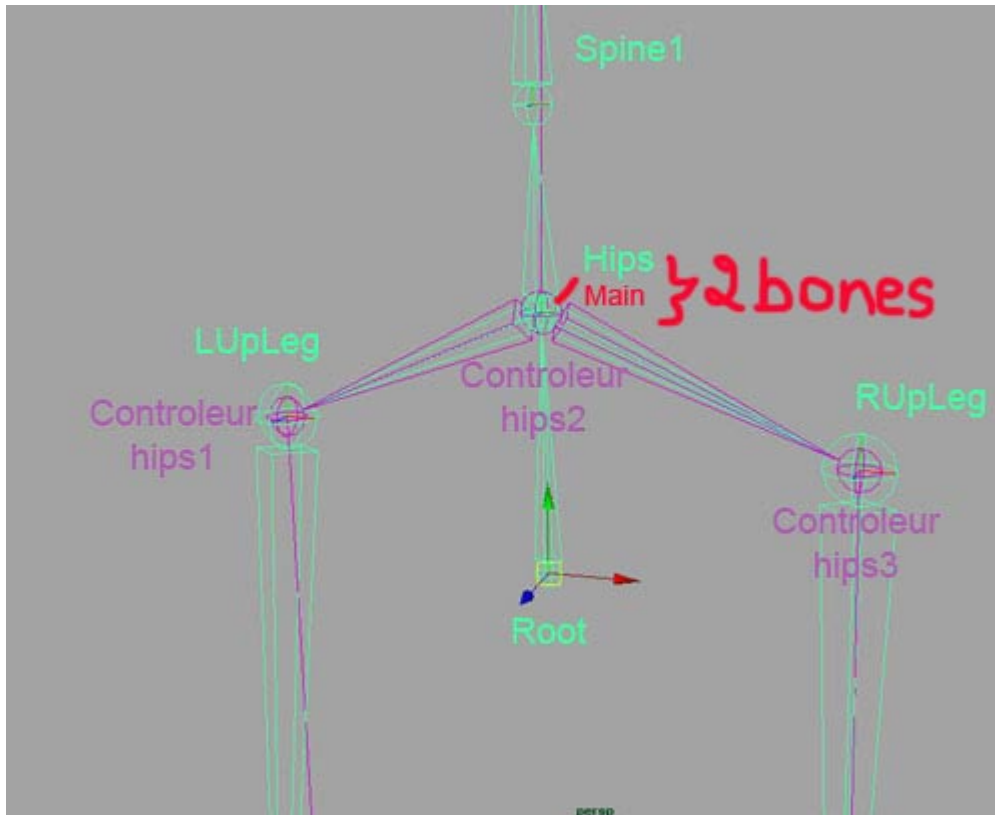
**Heel Pivot** : Il correspond au pivotement du pied, de gauche à droite, quand le talon est toujours au sol. La connection se fait sur le **controlfoot1** en *rotation Y*.

**Toe Pivot** : Il s'agit du pivot du pied en gardant les doigts de pied au sol. La connection se fait sur le **controlfoot2**, en *rotation Y*.

**Side to side** : Il s'agit de l'inclinaison de la cheville avec le pied toujours maintenu au sol. La connection se fait au **controlfoot2** en *rotation X*.

## Le rig de la colonne vertébrale

Pour la colonne vertébrale, on utilise une **ik Handle Spline Solver**.



Ensuite, on crée donc l'ik spline du **main** au dernier spine avant le cou. Ici l'ik spline s'arrête pile à la jonction des deux bras (ici en l'occurrence au **spine5**). On la nomme Ik\_colone\_vertébrale.

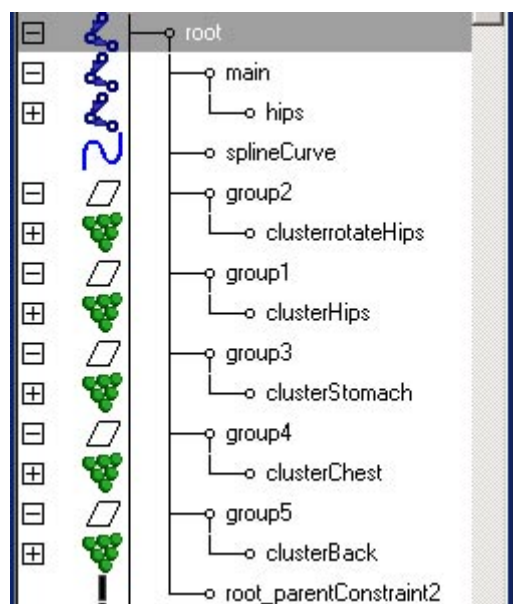
On va ensuite créer des clusters sur l'ik spline, pour permettre de l'animer avec les contrôleurs.

Pour cela, on sélectionne la courbe, on sélectionne le vertex qui est au niveau de l'hips et on va dans *create Deformers* et on crée un **cluster**. Dans la création des cluster, il faut cocher **relative** et dans advanced, **déformation order => Parallel**.

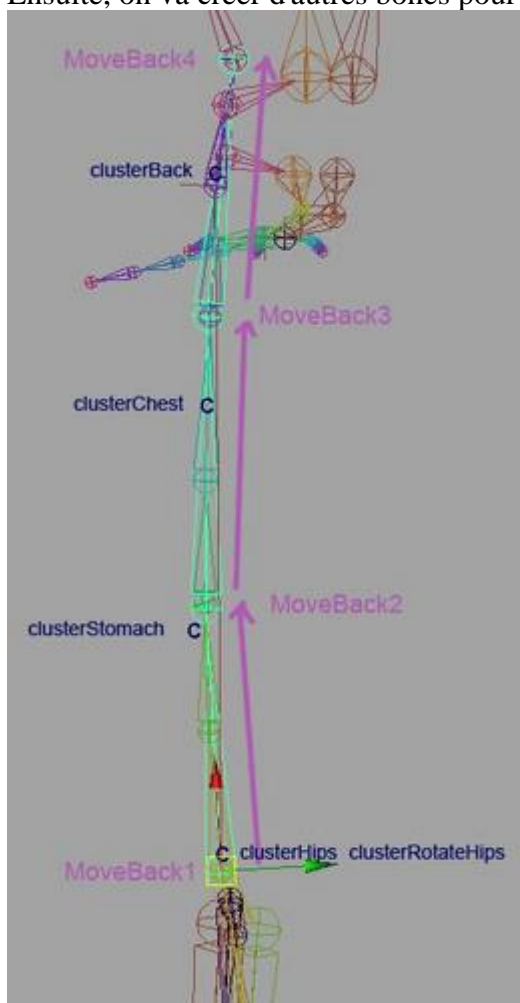
Ce cluster on le renomme **clusterHips**, puis un au niveau du *back\_control* que l'on va appeler **clusterStomach**, un autre au niveau du *chest\_control* que l'on va appeler **clusterChest** et un autre sur le dernier point de la curve que l'on appelle **clusterBack** et le dernier on rajoute un point sur la courbe vers le hips et on lui colle un cluster qu'on appelle **clusterRotateHips**.

Attention, il est important de bien respecter l'ordre de création des clusters pour ne pas avoir d'ennuis par la suite.

Dans l'outliner, les clusters sont représentés sous forme de grappes qui ressemblent à des grappes de raisin. Votre hiérarchie doit se décomposer comme ceci :



Ensuite, on va créer d'autres bones pour la colonne vertébrale. Comme ceci :



On crée donc 4 nouveaux bones que l'on nomme **MoveBack**.

On va donc commencer par connecter le **center\_o\_gravity**. Celui là, entraîne le root. On sélectionne donc le **center\_o\_gravity**, puis le **root** et on leur applique une **constraint => parent** dans les *translates* et les *rotates*.

On sélectionne ensuite le **root** puis le **moveBack1** et on leur ajoute une **constraint => parent** sur les *translates* et les *rotates*.

Ensuite nous avons l'**hipsway** qui est la courbe en forme de noeud papillon. Pour obtenir une bonne animation, nous allons procéder comme pour le pied. et rajouter quelques bones supplémentaires pour le bassin. Ici sur l'image, les bones rajoutés sont ceux en rose sur l'image où l'hips est de face.

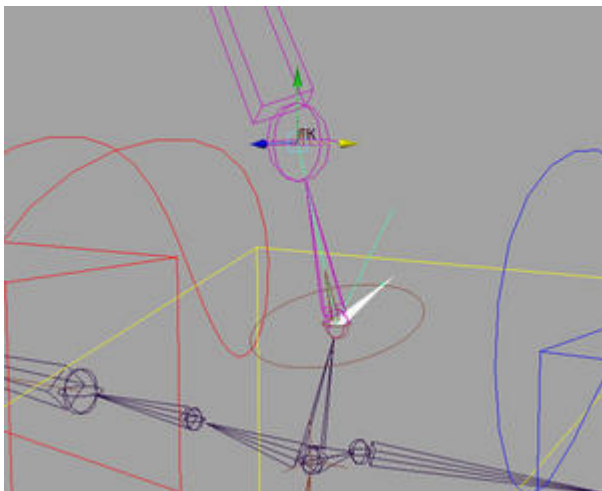
On sélectionne L'**hipsway** et le **clusterHips** et on leur applique une **constraint => parent** sur les *translates* et les *rotates*. On sélectionne ensuite le **clusterHips** et le **controleurHips2** et on leur applique également une **constraint => parent** sur les *translates* et les *rotates*. On sélectionne de nouveau le **controleurHips2**, puis l'**hips** et on leur applique une **constraint => parent** sur les *translates* et les *rotates*. On re sélectionne le **controleurHips2** et cette fois le **clusterRotateHips** et on leur applique une **constraint => orient** en *cochant bien tous les axes*.

Passons au **backcontrol**. On sélectionne le **backControl** puis le **MoveBack2** et on leur applique une **constraint => parent** sur les *translates* et les *rotates*. On sélectionne ensuite le **MoveBack2** puis le **ClusterStomach** et on leur applique aussi une **constraint => parent** sur les *translates* et les *rotates*.

On sélectionne ensuite le **chestcontrol** et le **moveBack3** et on leur applique une **constraint => parent** dans les *translates* et les *rotates* et on fait de même avec le **moveBack3** et le **clusterChest**.

On sélectionne ensuite le **MoveBack4** et le **clusterBack** et on leur applique également une **constraint=> parent**.

On crée ensuite un deuxième ik Handle RP scolver de la base du cou à la tête :



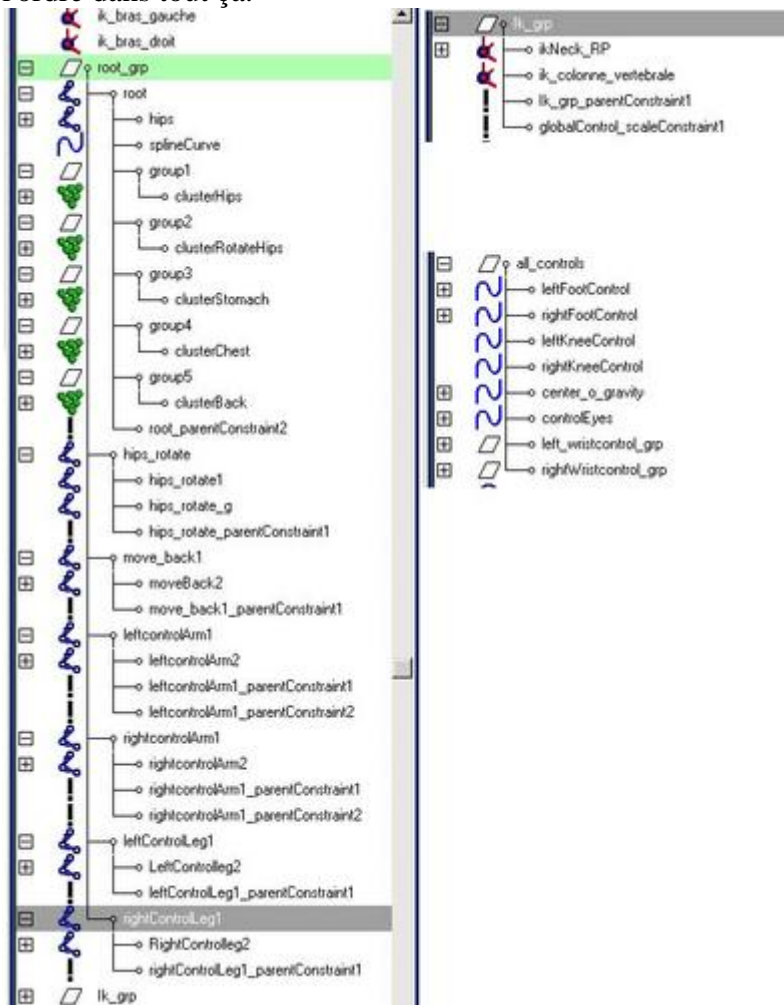
et pour terminer, on connecter le bone de la tête au **headControl**.

## La Tête

On connecte le **head\_control** au bone **Head** avec une **contrainte => orient**, Puis on connecte le **headControl** à l'Ik du cou avec une **constraint => point**.

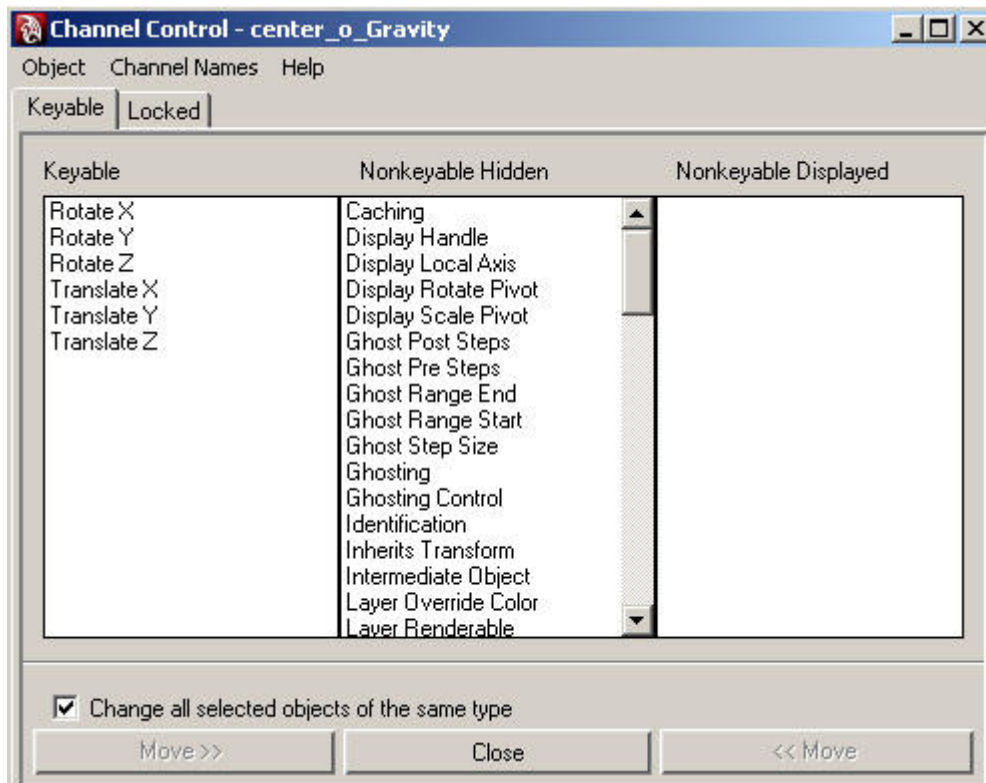
## L'outliner

Il est temps de faire un peu de ménage dans l'outliner et de faire les dernières connections. Voilà à quoi doit ressembler votre outliner. N'hésitez pas à créer des groupes et à remettre de l'ordre dans tout ça.



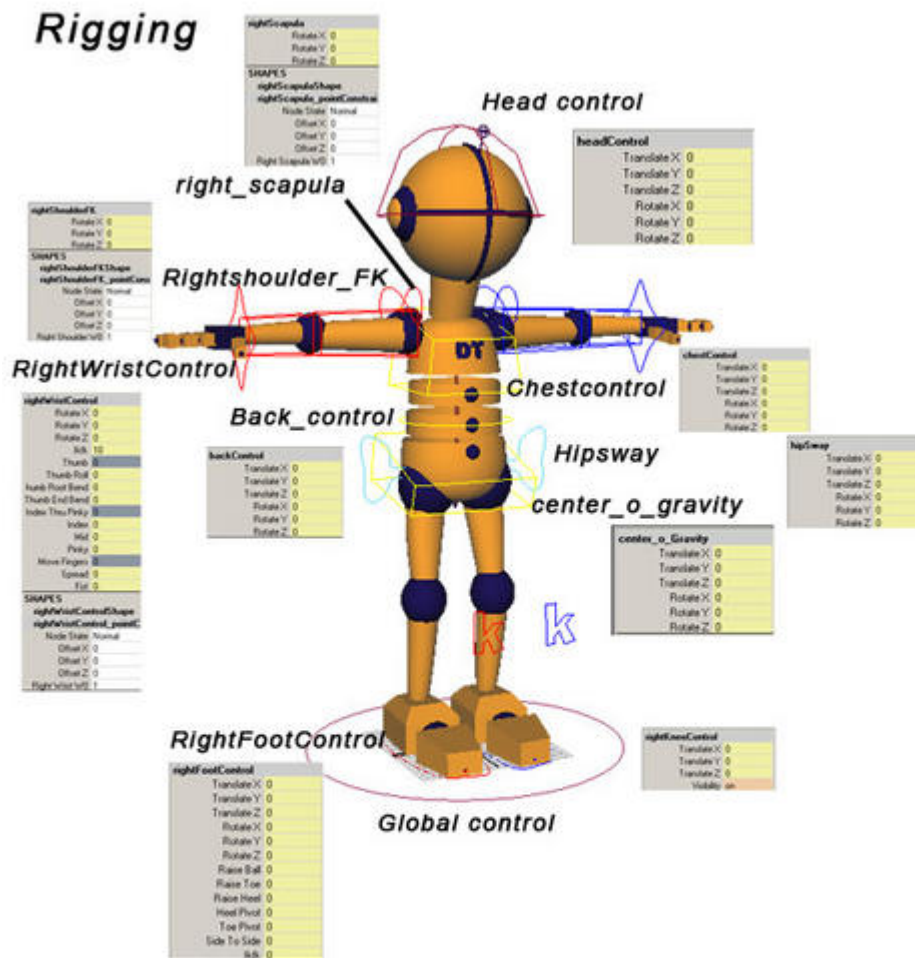
Pour les connections de dernière minute, on connecte le groupe des Iks au **globalControl** avec une **constrain=>scale** et on applique une **constrain=> parent** du **global\_control** au groupe d'iks. On fait de même avec le groupe de **root\_grp** et **all\_controls**.

Pour finir, on masque les trucs qui servent à rien sur les contrôleurs. On va dans windows => general editor => channel control.



pour le center\_o\_gravity par exemple, on ne laisse visible et Keyable que les rotate et les translate. et on Lock le scale et la visibility.

On fait de même pour tous les contrôleurs, jusqu'à obtenir la config suivante :



*Le rightshoulder\_fk est parenté au right\_scapula*

*La tête et les deux épaules sont parentées au chestcontrol*

*Le chestcontrol est parenté au back\_control qui lui même est parenté au center\_o\_gravity. Le hipsway est aussi parenté au center\_o\_gravity.*

Je précise que le personnage utilisé dans cet exemple, est la propriété de digital Tutors.

## Skinning

Si après le rigging, vous rencontrez des problèmes de skinning, ou que votre skinning à craqué pour x, ou y raison, sachez que vous pouvez toujours le modifier. Moi par exemple, en construisant mon perso, j'ai rajouté des bones après l'avoir skinné et du coup, ils n'avaient plus d'influences sur mon skinning. Pour que ces nouveaux bones soient pris en compte, il suffit juste d'aller dans l'**outliner**, dans **display**, décocher : **DAG Object Only** et vous allez sélectionner votre **bind pose** et vous la supprimez. Vous pouvez ensuite détacher votre partie à reskinner et la réattacher.

Je voudrais remercier Erik Svensson, de CG Talk, qui m'a bien aidée sur ce rig.