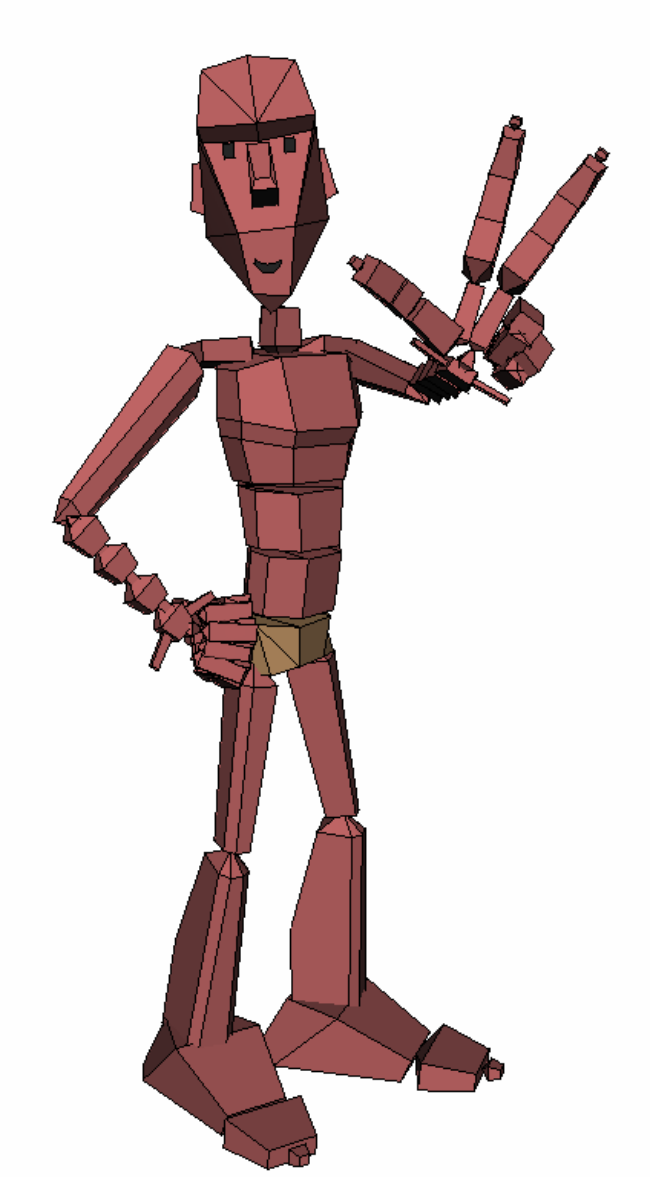
IK Joe v3

A FREE RIG FOR THE MAYA COMMUNITY



by Lluís "*Mr 2-many-bones*" Llobera
foreword by Javier "*Goosh*" Solsona
September 2003

TABLE OF CONTENTS

1. FOREWORD

The 3D community has been growing tremendously in the last few years. It wasn't long ago when most people kept very tight lids on their techniques, approaches and findings. Thanks to a few people and the easy access to so much information via the internet, this seems to be changing.

The fist version of IK-Joe was built by Daniel Martinez Lara for 3D Studio MAX. That inspired me to build a version of IK-Joe for Maya. At that point there were no free characters ready to animate available for such package.

IK-Joe proved to be a huge success. And so, things have evolved since then.

Lluis has done an amazing job in grabbing a core idea and taking it to the next level. IK-Joe v2 was a much cleaner, solid version of the original one. And now version 3 provides yet another approach on rigging a character.

With rigging101 we hope to shear with you and everybody in the 3D community a little bit of our thoughts, our techniques and approaches. This is just how we like to build things, by no means it's the only way to build them, it's merely, just one more way of doing things.

Happy rigging.

Javier "Goosh" Solsona

2. VERSIONS HISTORY

VERSION 1

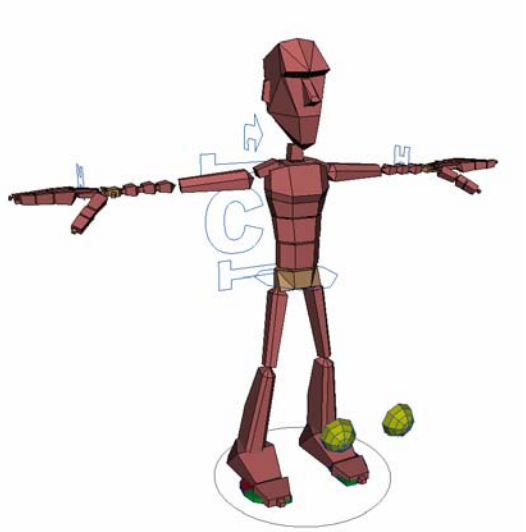
The first version of the IKJoe for Maya was done by Javier "*Goosh*" Solsona. It was created for Maya version 3.0.

The rig included an IK/FK spine, based in Jason Schiefler's method. It also used IK/FK controllers for the arms, IK controllers for the legs, and finger controllers.

It was a really excellent character setup, and I can clearly remember how it was an instant success the second it was released.

Its strong points were that it didn't have many controllers, and that each of the controllers was clearly recognisable from the distance. Also, that the rig was clearly *solid* and it was really easy to get a hold on to.

A very new thing in this setup was that some of the controllers were NURBS curves, while some were polygonal shapes that didn't get in the render.



The first version of the ikJoe rig was done by Javier "Goosh" Solsona

VERSION 2

Back then, I was a member of the *10SecondClub* (I still am, although not very active at the moment). There I found about Javier's work, and I was really impressed with it. I asked him if he would mind my doing a new version of the rig. He was very pleased with the idea, and very supportive also. It was the beginning of a good friendship.

Version 2 was done deleting the whole ikJoe v1 rig and starting it all over again. This is what I resolved to do to study Javier's rigging work in the first version. It was a great learning experience.

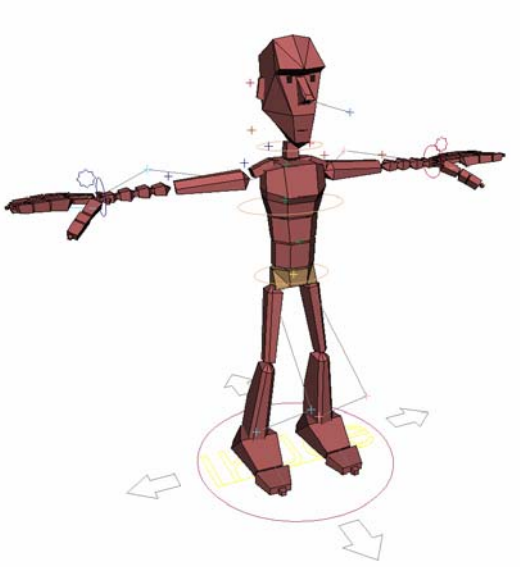
As I redid the whole rig, I canged a few things that I felt that worked better for me, and I also gave the character eyes and a mouth.

This version had an IK/FK spine, based in Jason Schiefler's. It also had direct/indirect neck controllers, IK legs with independent poleVectors, rotating clavicle controllers, IK/FK arms with a *stick* option for IK, normal and explicit finger controllers, and some basic facial controllers as well.

Looking back, I think that it was a good setup. However, it featured a lot of things that I avoid doing in my rigs at this moment, such as using handles for the controllers.

Modifying Daniel Lara's model, I created a femenine companion for the ikJoe. I called her ikJane, and as she had the same proportions of him, I could make the rig work for her. I remember having the idea of creating an ik-family… Who knows? It still sounds good, and it might be done in the future!

I did three upgrades to this rig. Instead of having rotating feet, in version 2.0 an extra controller was used to specify the rotating direction of each foot, and there were not clavicle controllers. The final version, 2.5, is seen in the image above.



The ikJoe v2 build had three upgrades : 2.0, 2.1 and 2.5

VERSION 3

One of the new things in Maya version 5.0 concerned the *stickiness* of the ikHandles. Up to that time, the *stickiness* parameter of the ikHandles had never been of much use, and since it was unabled by default very few people ever activated it. However, in Maya version 5.0 the ikHandles seemed to act weird. The rigged characters that worked well in version 4.5 did not in this new version. As it turned out, they needed their ikHandles' *stickiness* parameter activated.

If you ever find this problem, use this quick *MEL* shortcut to solve it :

```
for ($ELEMENT in `ls –type ikHandle`) eval ("setAttr " + $ELEMENT +
".stickiness 1")
```

However, since a lot of people did not know about this easy solution, and since my rigging had changed much since I did the ikJoe v2 rig, I thought that creating a version 3 could be a good solution.

Since July 2003, I had been working on my own auto-rigging script. I started writing it for a model that Ken Cougar asked me to rig. He is the one who modelled the Troll character, and I really like his models. But when I was around 80% finished, I had to start a new freelance job and the whole thing suddendly had to be paused.

However, in order to check if the script worked well enough, I decided to use what I had with the new ikJoe that I pretended to do. The result is the ikJoe v3 rig. Its features are described in this document.

I might do a new upgrade of version 3, but not for the moment. I would like to encourage everybody that has any suggestions or finds any flaws in the ikJoe v3 rig to e-mail me (*lluisllobera@hotmail.com*) so that I can work on it.

3. VERSION 3 IN DETAIL

In the following pages the different controllers of the ikJoe v3 rig are described in detail. Although it's not very complicated, I recommend reading this whole document through in order to learn how to use this setup.

Remember that the ikJoe v3.0 rig is available for non-profitable tasks only, and it can be freely downloaded from www.rigging101.com

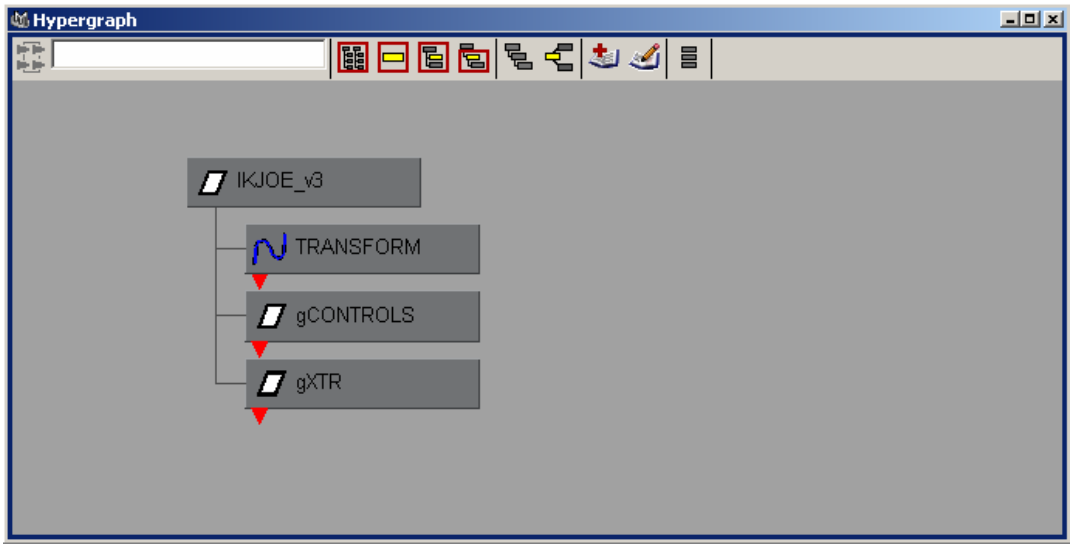
3.1. A FEW PARTICULARITIES

Before starting to explain how every single controller works, I think it would be wise to talk about a couple of things that are quite important. I'd also like to stress that this is the kind of rig that I would like to animate, and it's built on my personal tastes and preferences. Some people might find some things missing, or perhaps they might have done some things differently. I am well aware of this, and I can only say that I have tried to make the rig as *general* as I can.

Nodes in the hypergraph

If you open the Hypergraph, at the top level of the *DAG* you'll see only one node: a group called *IKJOE_v3*. If you wish to export the character from an animated scene to another one, just select this top node and export it.

If you expand the group, you'll see it contains three nodes: a NURBS curve called *TRANSFORM*, and two groups called *gCONTROLS* and *gXTR*.



The TRANSFORM curve is a controller discussed in chapter 3.2.

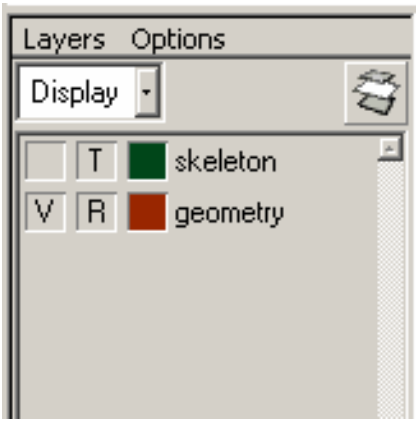
The gCONTROLS group contains all the controllers of the rig, properly parented and sub-parented as necessary.

The gXTR group contains a lot of extra nodes necessary for the rig, such as the skelegon, the inverse feet lock skeletons, the helper lines and the geometry that conforms the character.

Please be aware that neither the gCONTROLS nor the gXTR groups shold ever be manipulated for the sake of the rig's correct functioning.

Layers

There are two layers in the rig. The *skeleton* layer contains all the joints in the skeleton, and is by default hidden because seeing it is not necessary for the rig. The *geometry* layer contains the different surfaces that make out the ikJoe's body, and is put in reference mode so that the user cannot by mistake move, rotate or translate them.



FK spine

A thing that most people will miss from the previous ikJoe rigs is the IK spine. Of course, I have my reasons for having omitted this feature.

In my experience, I have found that IK spines are visually impressive, but that's it. When you start playing with an IK spine you have a lot of fun, because the character seems very much *fluid*, and besides it's very quick to create a pose for a character.

On the other hand, IK spines are not so great when animating. For one thing, there are certain poses that you can't really get with an IK spine. And, worst of all, it's very hard to create an *overlapping* sensation with it.

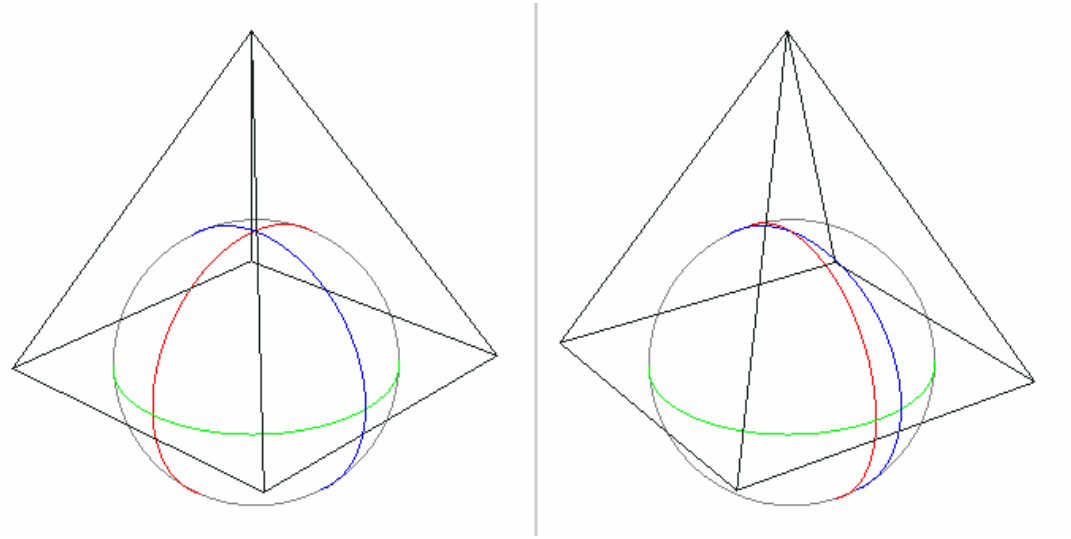
There's something about the IK spine in the previous ikJoe rigs that I didn't like much either, and it's the fact that the shoulder area used to changed a little to adapt whenever the spine was being manipulated. After some investigation II found a way to solve this problem, but it didn't allow for an FK spine implementation on top of it, so I resolved to stick to what I liked.

FK spines *do* allow the user to create *overlapping* action, and they can get the character to any pose that an IK spine would. Of course, it takes more time to get the different poses, but in a way that's just a matter of practice.

Rotation

I like to animate in *gimbal* mode, where you can see the real disposition of the rotation axes of any node you select. Thus, you can change only *one* of the rotation axes and keyframe it, and be sure that Maya will not create any weird mess. In my experience, getting used to this makes animating the rotation axes a much easier task.

At the same time, if you use gimbal rotation you will sooner or later get into *gimbal lock*. This is the name that is given to the situation when two rotation axes are one on top of the other. The following examples illustrates it :



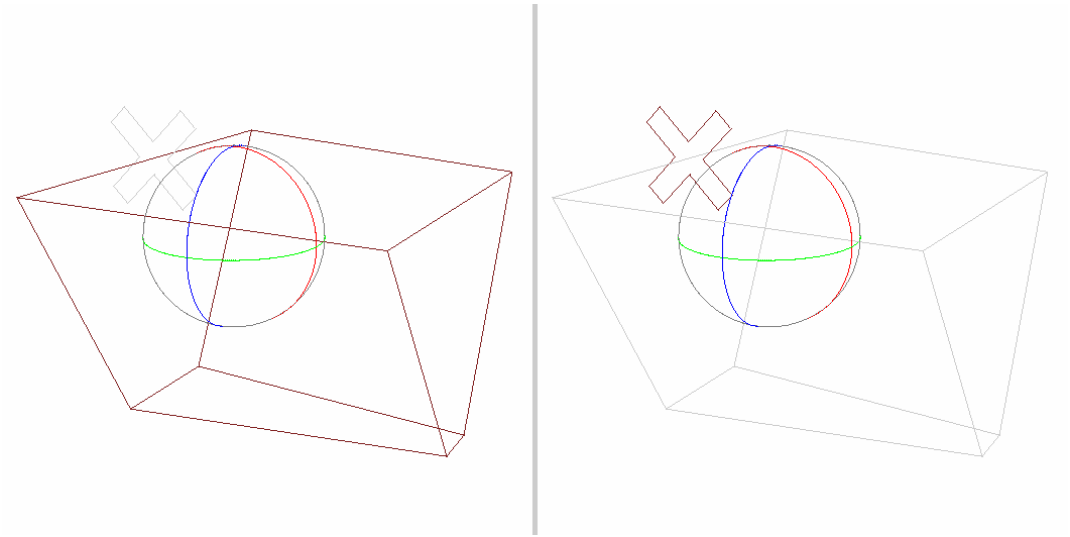
In the first image, we have just created a pyramid. We have selected it and are going to rotate it in *gimbal* mode. As you can clearly see, each of the three different axes describes a different rotating direction for the pyramid: we can identify an axis to either rotate it sideways (x), or front to back (z), or around (y).

In the second image, we have rotated the pyramid about 80 degrees around. The x and the z rotation axes are nearly one on top of the other, and therefore there is no axis that we can use to rotate the pyramid sideways if we want to. We have found *gimbal lock*.

There is a way to avoid the gimbal lock in this particular situation, and that implies changing the *rotation orders*. If we opened the Attribute Editor and changed the rotation order of the pyramid from its default value of *xyz* to *xzy*, we would have the three rotation axes apart, like we had at the beginning. This is because changing the rotation order means changing which rotation axes is evaluated first.

Unfortunately, rotating in gimbal mode means that sooner or later we will get into gimbal lock. If we rotated the new pyramid in the *z* axis, the *x* axis would sooner or later get on top of the *y* axis. The only way to avoid getting into gimbal lock is either to unable one of the nodes' rotation axes, or to use an extra rotation controller that can get us out of the way when we do get into that undesired situation.

This is the approach that I have followed in the ikJoe v3 rig. Most rotating controllers have an extra attribute called *rotateXTR*, which can be turned *on* or *off*. If activated, another rotating controller will appear that makes exactly the same function as the first controller, only this one won't have the rotation axes into gimbal lock and therefore can be of much help.



This image shows the rig's *COG* controller and the extra rotation controller for it, which I call the *xCOG* controller. As you can see, both controllers have the rotation pivot at exactly the same point, and they also have the rotation axes oriented in the same fashion. Therefore, if you ever get into gimbal lock with the *COG* controller, the *xCOG* will be of help: it will have the exactly the same effect as the *COG*, but its rotation axes will not be in gimbal lock.

Of course, this new rotation controller can also get into gimbal lock, so my advice is that you use the extra rotation controllers only to avoid a specific situation, and then reset their rotations to 0 as soon as you can and get back to working with the main controllers.

Switchers

Until now, I had not a clear idea of which was the best place to put the FK/IK switching attributes for the arms and the legs. I didn't like them in the controllers that were necessary to adjust the position or rotation of the limbs, since keyframing these controllers also meant keyframing the switching attribute, and this lead to having multiple innecessary keyframes in the Graph Editor.

I have resolved, in this version, to create the switchers as independent control nodes. They will become visible automatically whenever the appropiate control nodes do, and they are represented by little *pin*s. See below for more information.

Helper lines

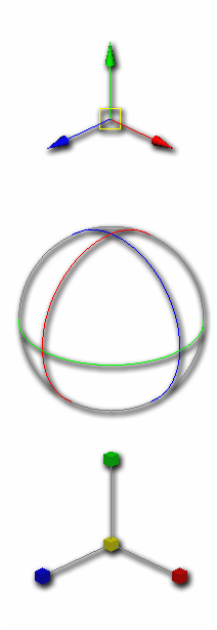
You'll notice there are certain templated *lines* that bind together some of the rig's controllers. They are just 1-degree NURBS curves, and I've put them there so that, hiding the geometry, the animator still can tell in what pose the character is.

Also, it might be interesting to note that all the geometry of the ikJoe is made of polygonal shapes, while all the controllers of the rig are made of NURBS curves. Thus, you could set up different panels and see just the controllers in one of them, and just the geometry in another.

Transformation pictograms

Each of the controllers described next are necessary for the rig. I have locked and hidden away the unnecessary channels; for example, the spine controllers cannot be moved and therefore their translation attributes are not to be seen in the Channel Box.

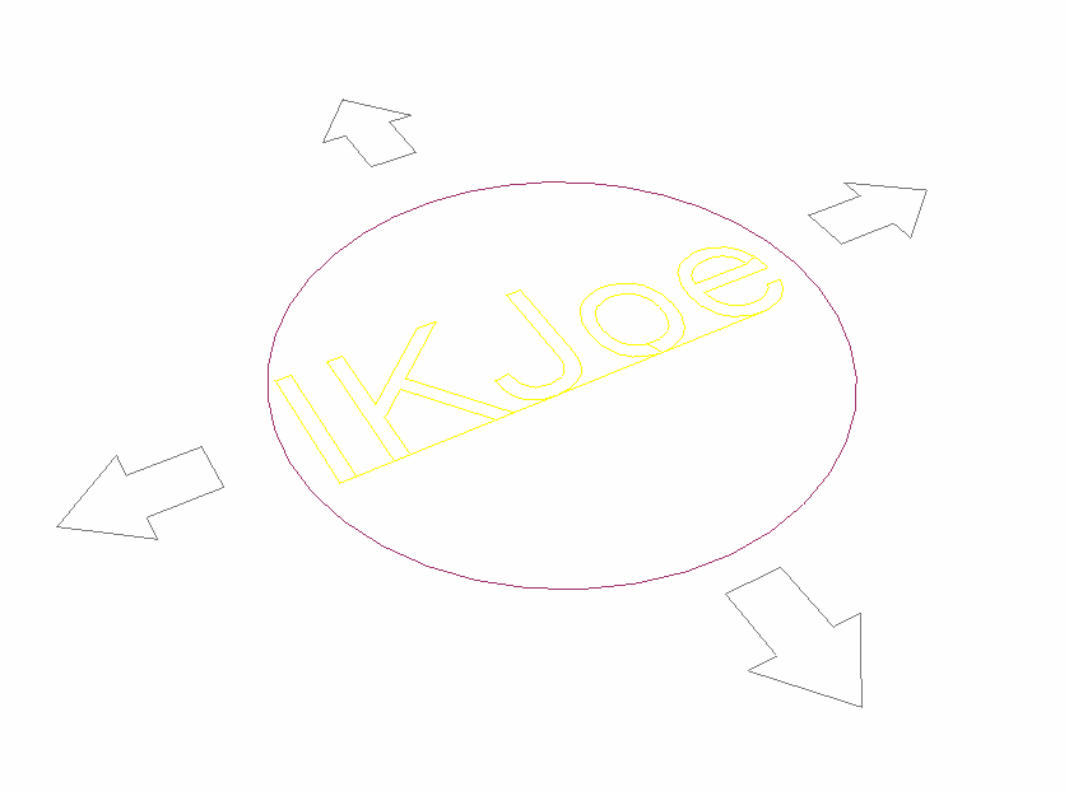
To make it even easier for the reader, I have added little pictures under each controller description. Those pictures indicate whether the controller can be transformed and, if so, with what tools. The pictures are as seen below.

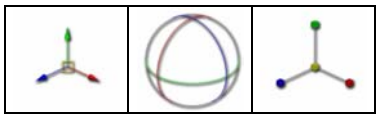


Bear in mind that, in addition to being transformed, some controllers might also have custom attributes. Look for them in the Channel Box.

3.2. TRANSFORM

The *TRANSFORM* node is a NURBS circular curve located at the character's feet. It can be moved, rotated or scaled, and doing so will also move, rotate or scale the whole rig without problems. I recommend using it only when you import the rig into a new scene, to quickly put is where you need.

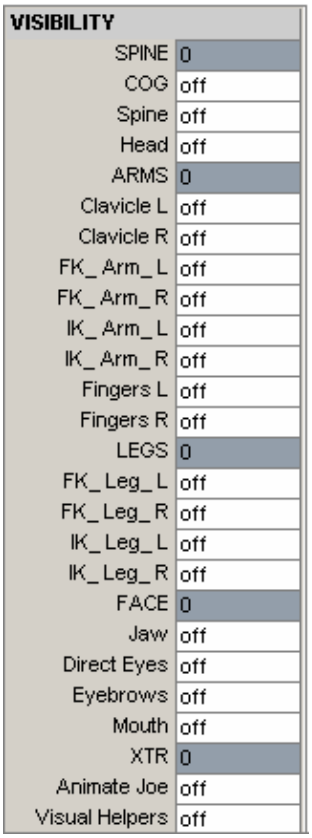




The *TRANSFORM* has some parented nodes, such as a control curve that spells out the character's name, called *VISIBILITY* (see below), and four templated direction arrows that are only there to make it visually interesting.

3.3. VISIBILITY

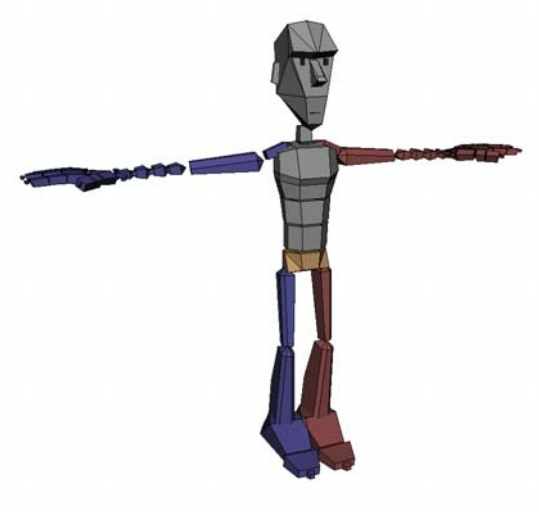
The *VISIBILITY* node is the NURBS curve that makes out the character's name, inside the *TRANSFORM* circle. It has a number of boolean attributes, that are used to toggle the visibility of each of the different rig's controllers.



As you can see in the image above, the channels in the *VISIBILITY* node are separated in sections. I believe that the sections called *spine*, *arms*, *legs* and *face*, and the attribute names that they contain, speak for themselves.

The *xtr* section, however, is worth a quick explanation. It contains two boolean attributes: one called *animateJoe*, and another one called *visualHelpers*.

AnimateJoe



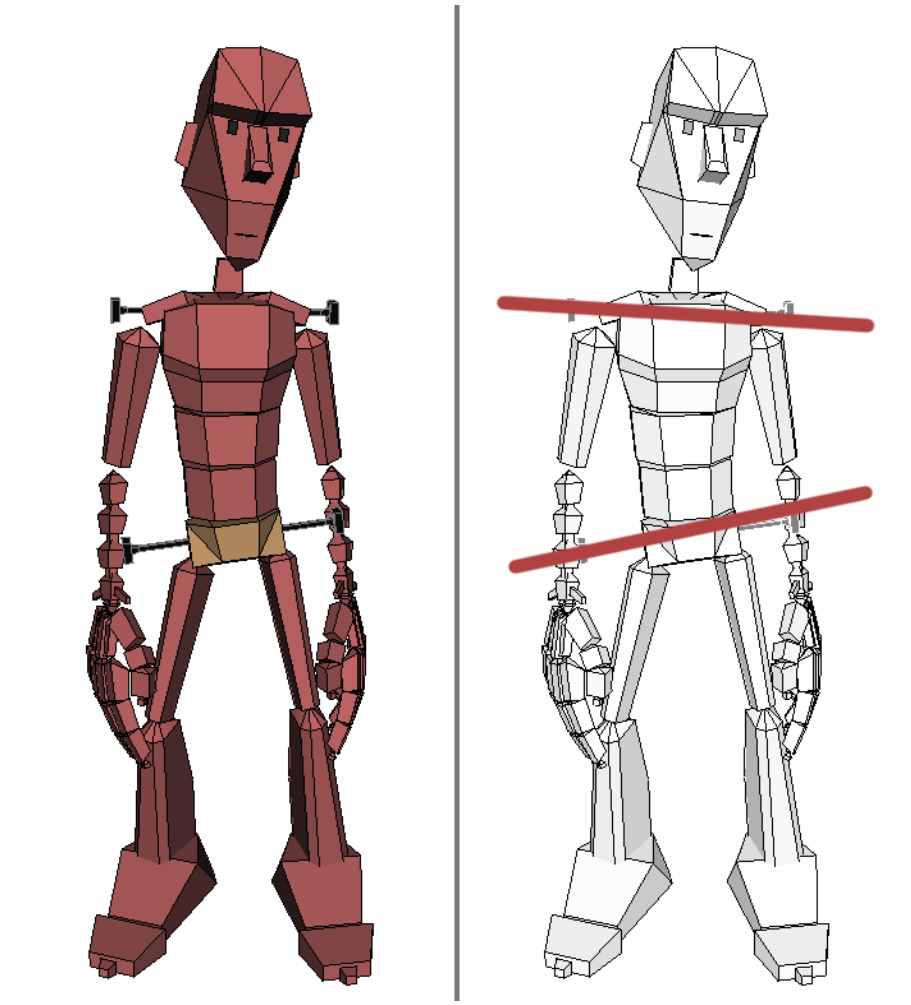
This attribute, if enabled, changes the colors of the geometry of the ikJoe. The left-side surfaces of his body become red, the surfaces at his rigt become blue, and the torso and face become grey. This is meant to help the animators when they are working with the character.

Suppose you are creating a walk cycle for the ikJoe and you run a playblast on an orthographic view. When you check it out, you notice that one of the legs sinks in the floor… Instead of playing back the scene from within Maya, you can quickly know which foot must be adjusted if you observe its color.

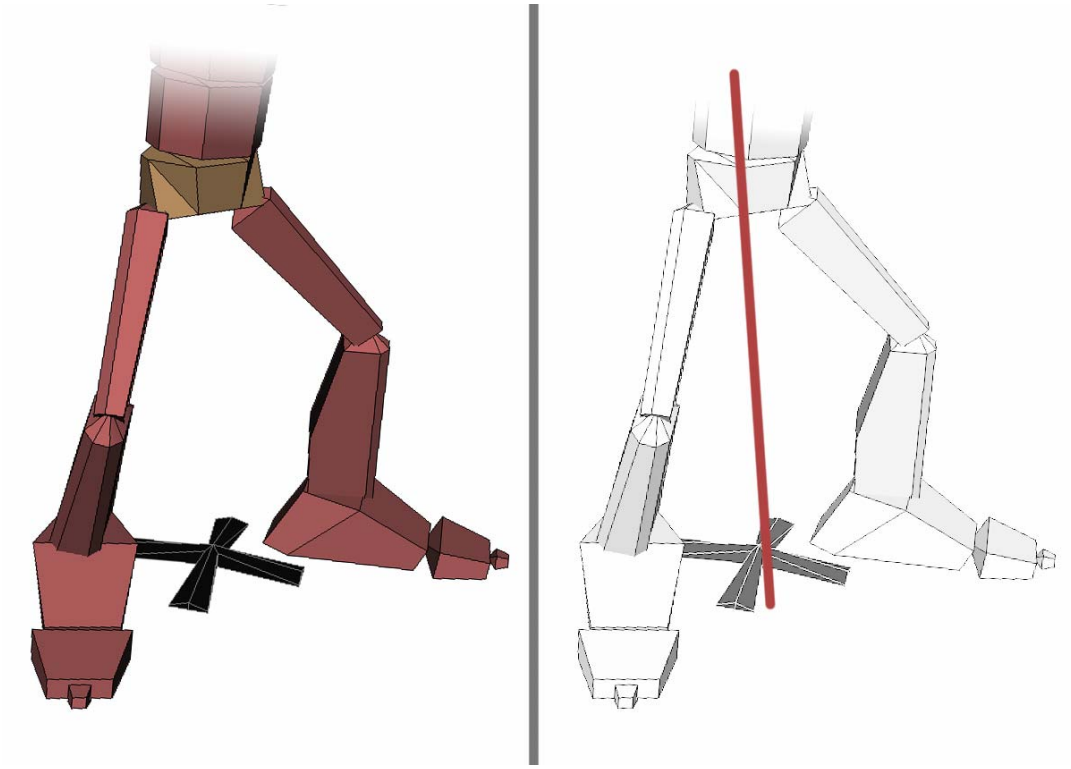
As a side note, please observe that the rest of controllers of ikJoe also follow a certain coloration. Again, the left-side controllers are all red, and the right-side controllers are all blue. I believe that knowing this can be really helpful.

VisualHelpers

This is a new thing to the rig. A new concept. I hope you'll find it useful. When *visualHelpers* is enabled, three polygonal shapes are made visible. Those polygonal shapes do not gen in the render, as they have their render flags modified, and are meant to help the animator.



The first two of such visual helpers are seen above. They are two polygonal prisms that should help the animator visualize the *line of balance* of the character's body. Remember that those two lines should have opposite directions most of the time.



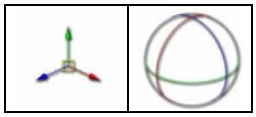
The other visual helper represents the *weight balance* of the body. It is a black little cross that is always exactly at the middle point between the two feet, above the ground level. It can be very useful for the animator if it is used to have an idea of where to place the character's *COG*.

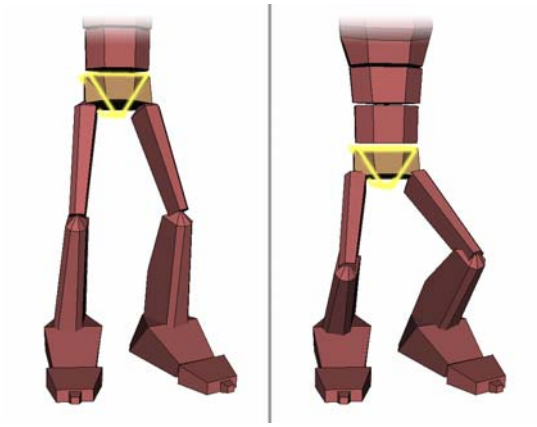
3.4. COG, SPINE AND HEAD

The first section of the *VISIBILITY* node contains three attributes: *COG*, *spine* and *head*. Those are the three elements that are used to change the character's torso position and orientation.

COG

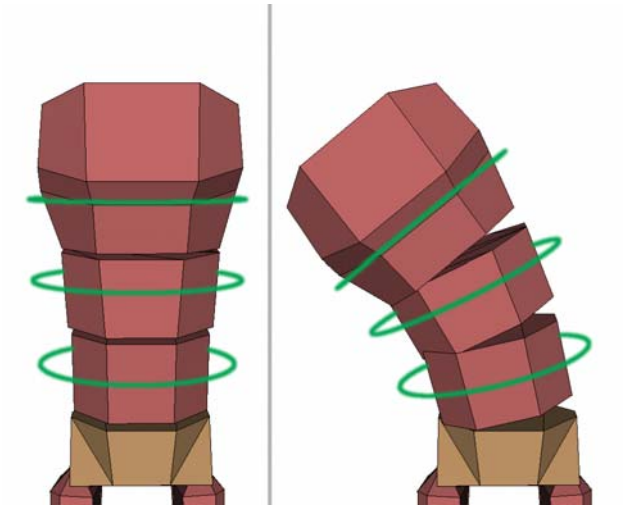
The *COG*, also called *root* by some people, can be moved and rotated to change the position of the whole character's torso. Its pivot point is exactly at the character's hips. Remember Walt Disney : "*all action always starts at one's hips*".

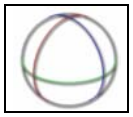




Spine

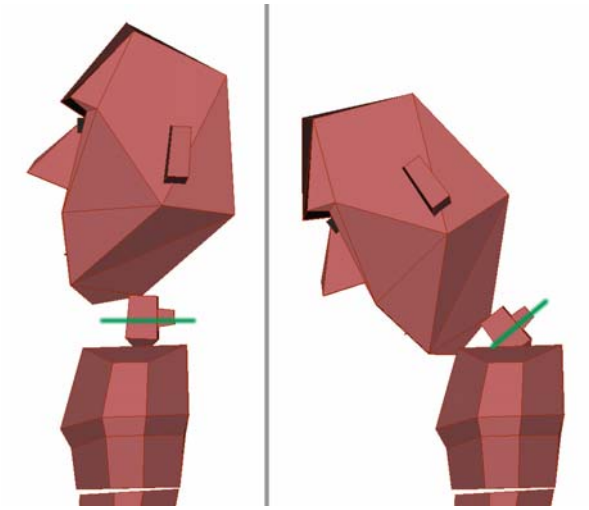
The spine is, as I already explained a little above, is FK. To be more precise, ikJoe's is a 3-segmented FK spine, which means that there are three control curves that can be used to rotate the character's torso.

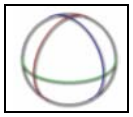




Neck

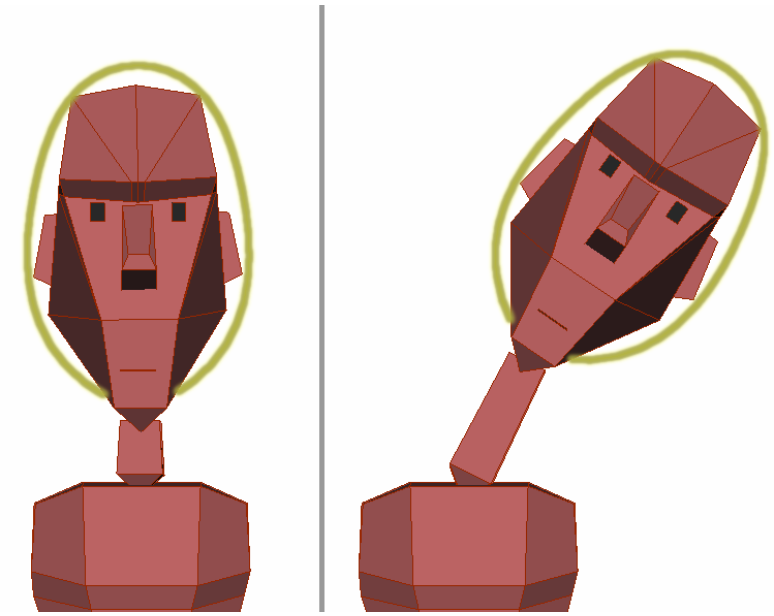
The neck controller can be rotated in any axis, just like the spine controllers.

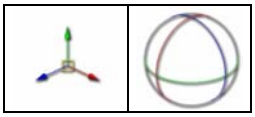




Head

The head controller can be either moved or rotated around. Most of the time rotating it will be the thing to do, but the animator has also the option of moving it around to get interesting effects. For example, if this controller is moved far away from the character's torso, the neck will become *elastic*.

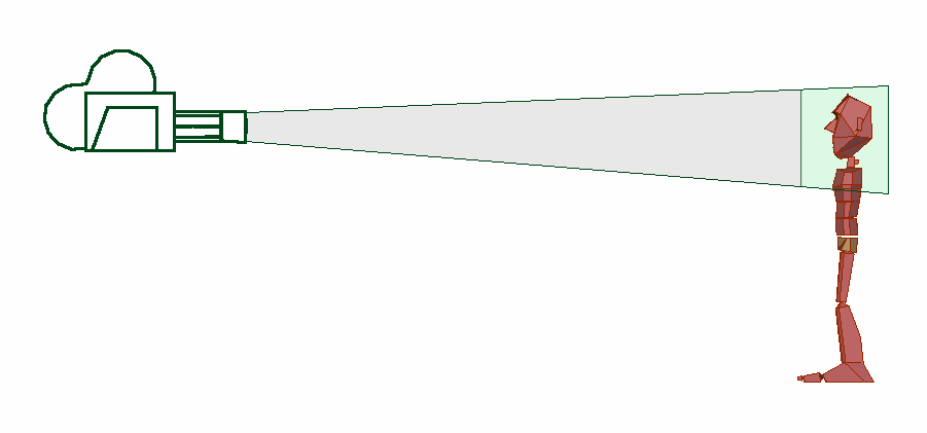




3.5. FACE

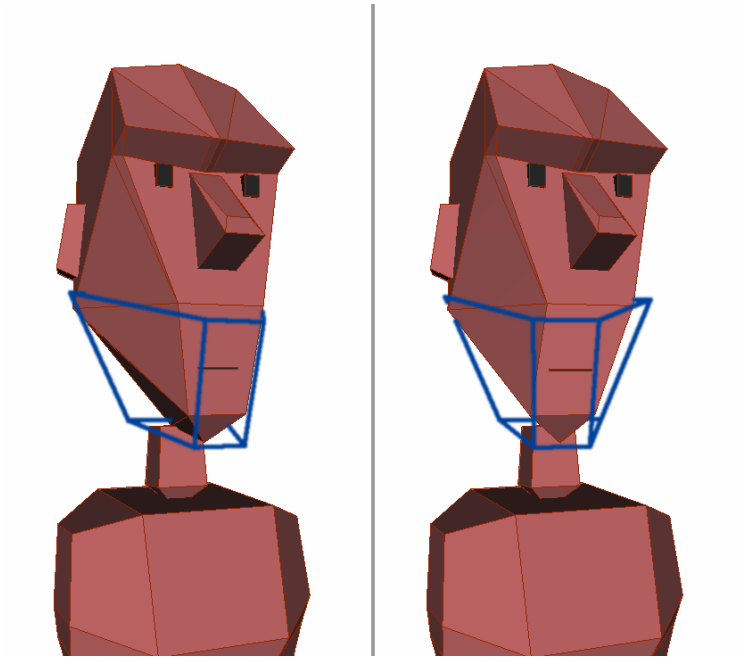
The facial controllers intend to be intuitive and easy to use for the animators. There are four different kind of facial controllers : the *jaw* controller, the *eyes* controller, the *eyebrows* controller and the *mouth* controller.

Note that there is a hidden locked camera parented to the ikJoe's head joint. This camera, called *facialCamera*, is therefore always looking straight at the character's face, no matter what pose he is in, and should be ideal when animating the facial expressions. Its near / far clip planes are adjusted so that no objects get in the way.



Jaw

The *jaw* controller can be used to slightly change the character's face shape. It is recommended rotating it up and down when the character is talking, to give a more dynamic feeling to his jaw.

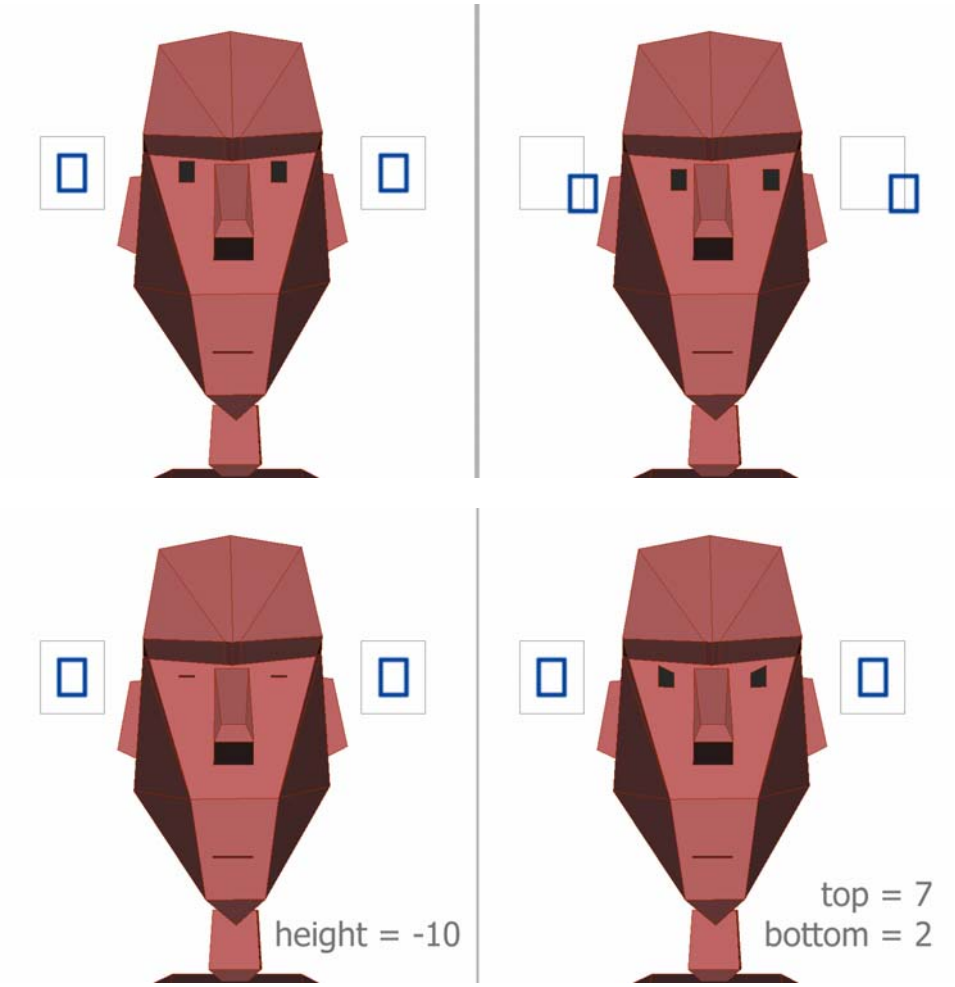


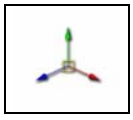
Eyes

When the eye controllers are enabled, four NURBS curves appear in the panel. The greyed-out ones are not the controllers themselves, but are just there to show the *limits* of the proper controllers.

The controllers are, then, the two black squares in the middle. They can be moved around to change the position of the ikJoe's eyes. Note that they also have a number of extra attributes, such as *top*, *bottom*, *width* and *height*. The first two of such attributes can be used to slightly change the eye's shape. The other two offer a way to change the eye's proportions.

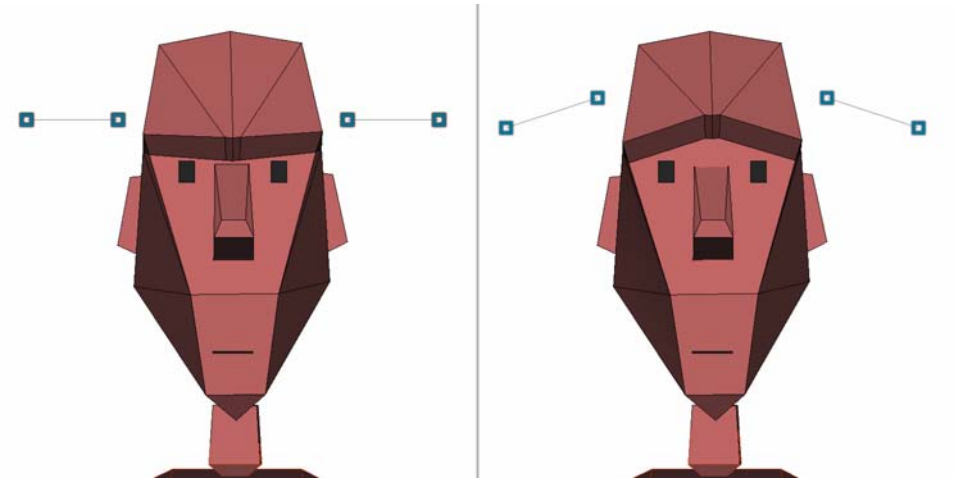
To make the character blink, it is recommended to set the *height* attribute to a value of -10.

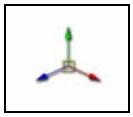




Eyebrows

Each eyebrow can be controlled with two square curves. Note that these curves can only be moved up and down along the *y* axis.

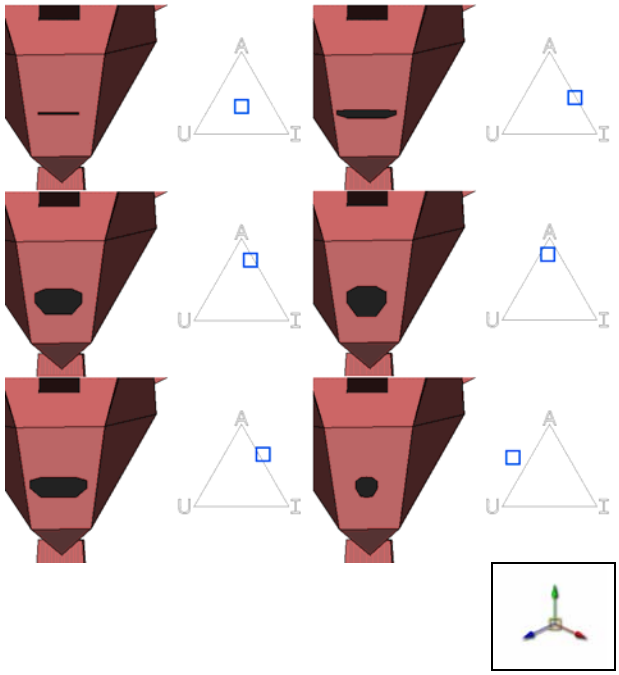




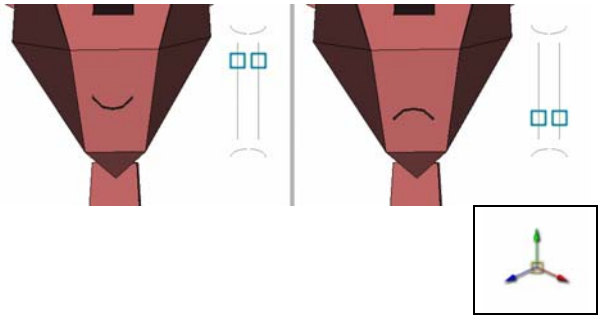
Mouth

The mouth has three different controllers, all found by the character's left cheek.

The first of such controllers is the *mouthShape*, which consists of a square that can be moved inside of a triangular shape. In the corners of this triangle there are three different vowel sounds : *A* (as in c*a*r), *I* (as in f*ee*l) and *U* (as in m*o*ve). Moving the controller to one of the triangle corners will make the mouth change its shape to match the sound. The rest of vowel sounds can be found inside the triangle.



The other two controllers are the *smile* controllers. They can only me moved up and down along the *y* axis, and in doing so the corners of ikJoe's mouth will bend accordingly.

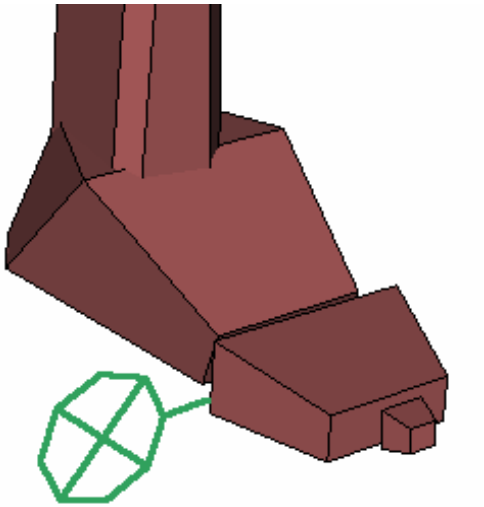


Needless to say, the mouth controllers can be mixed to create convincing shapes. Remember to avoid symmetry to make them even more interesting.

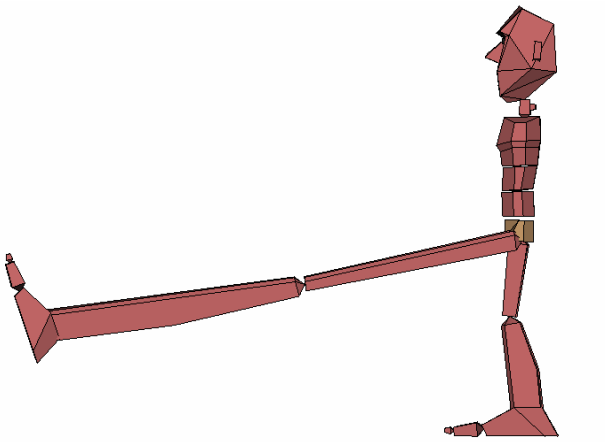
3.5. LEGS

This rig's legs can be animated either in IK or FK, depending on what the animator needs. Of course, most of the time the legs will need to be in IK mode, for they are nearly always in contact with the floor. However, it is also true that in some situations (for example, when the character needs to swim) FK will be more appropriate.

Whether you make the FK controllers visible, or the IK controllers, another one will show up. It's a green controller that will also be by the toe's joint position. It is the *switch* controller. As I explained earlier, this controller has the FK_IK attribute which will allow the animator to smoothly change the leg's behaviour.

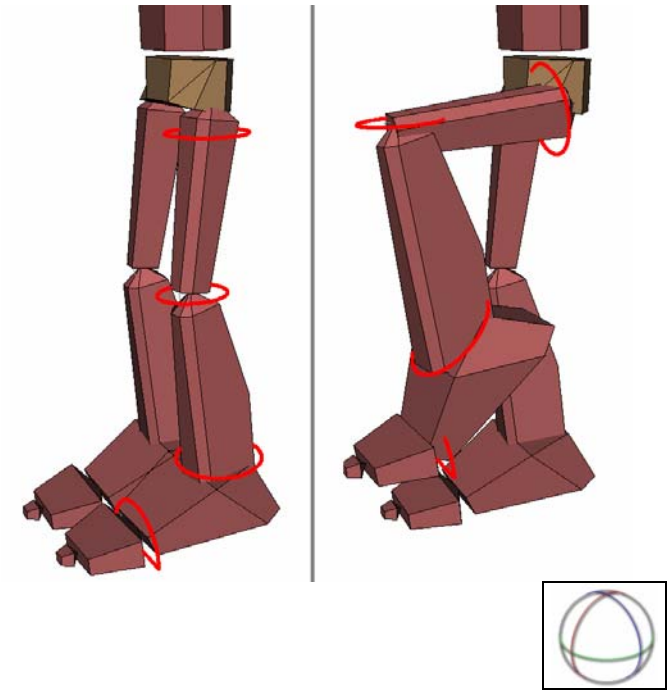


Note that this controller has also two extra attributes, called *elasticFactorFK* and *elasticFactorIK*. If you wish to make the legs *elastic*, you will have to animate those attributes. The first one, *elasticFactorFK*, will make the legs longer automatically when set above a value of 1. The second, *elasticFactorIK*, will allow the leg to elongate when the IK foot controller is at enough distance from the pelvis.



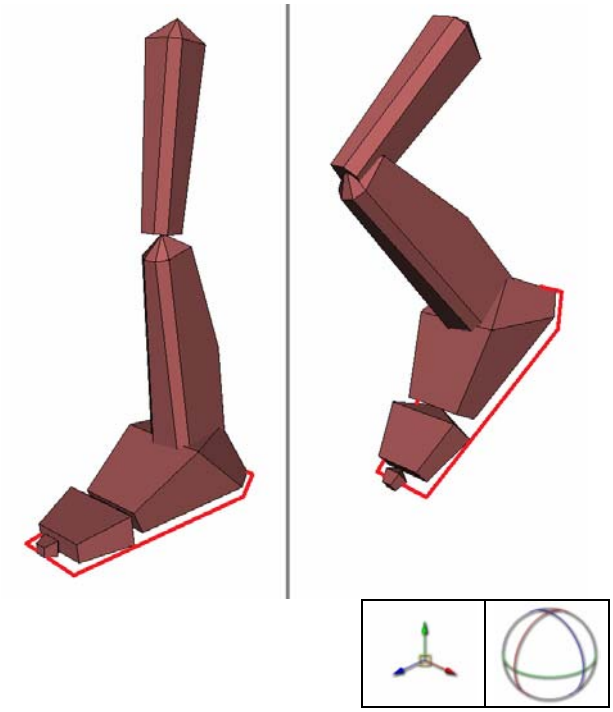
FK legs

The controllers for the FK legs can only be rotated. There's an individual controller in each point where the leg bends.

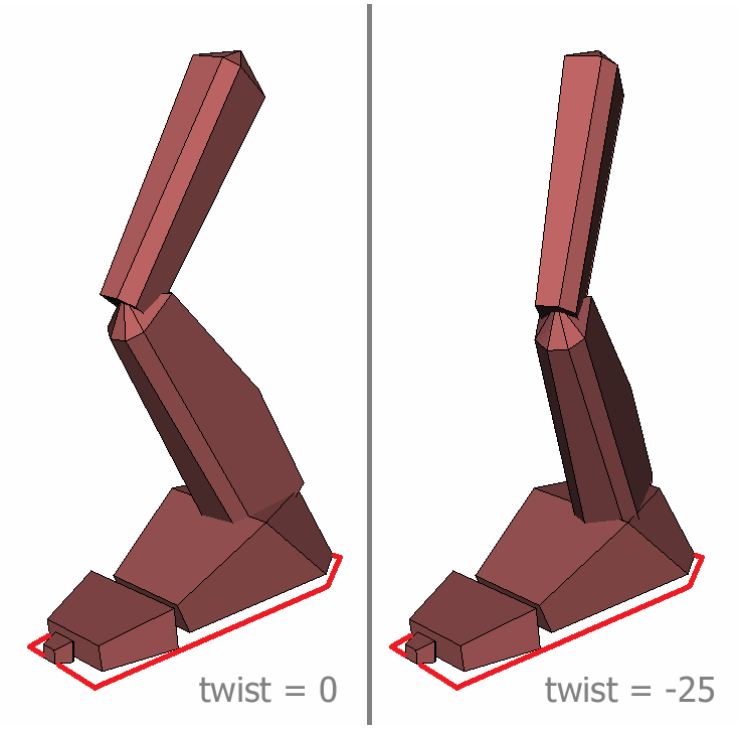


IK legs

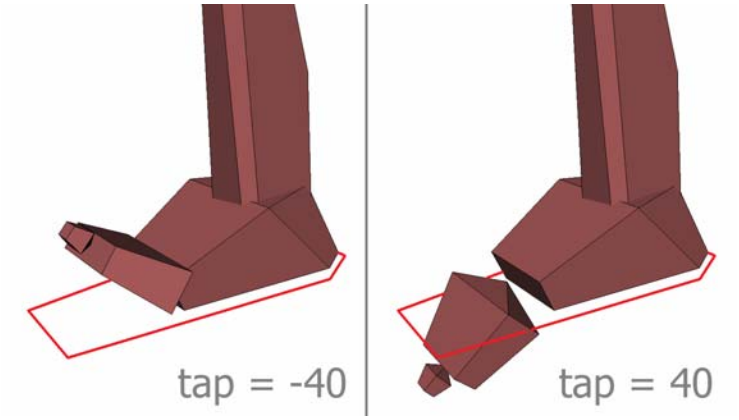
There's only one IK controller for each leg. It can be either moved or rotated, and it also has got a few extra attributes, described in the next pages.



The *twist* attribute is connected to the same attribute of the ikHandle of the leg. Therefore, modifying this attribute will make the knee rotate in one direction or the other. This control substitutes the *poleVector* that the ikJoe v2 rig used to have.

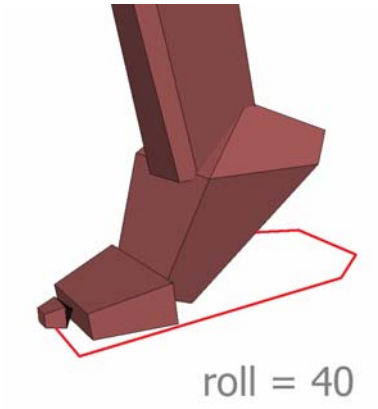


The *tap* attribute rotates the toes of the foot.

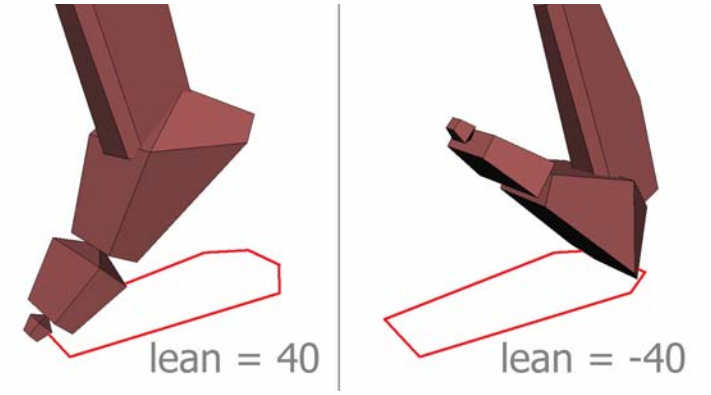


The rest of attributes are not to be used frequently for the animator. This is why they are in a separate section of the Channel Box, labelled *XTR*.

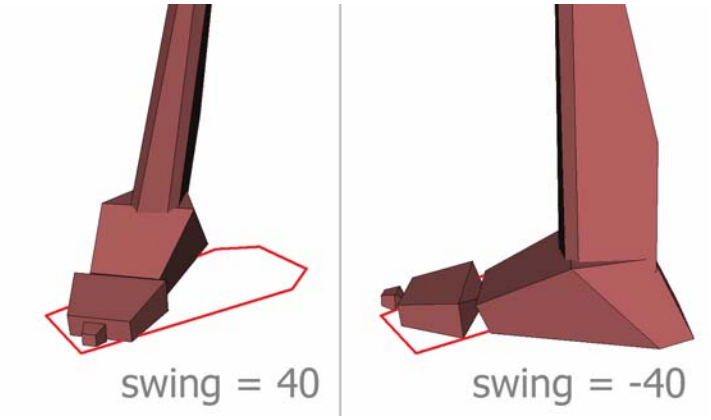
The *roll* attribute makes the character stand on his toes. Note that It cannot be set to a negative value.



The *lean* attribute can be set above zero or below zero. When above zero, the foot will stand on its fingertips. When it's below, it will stand on its heel.



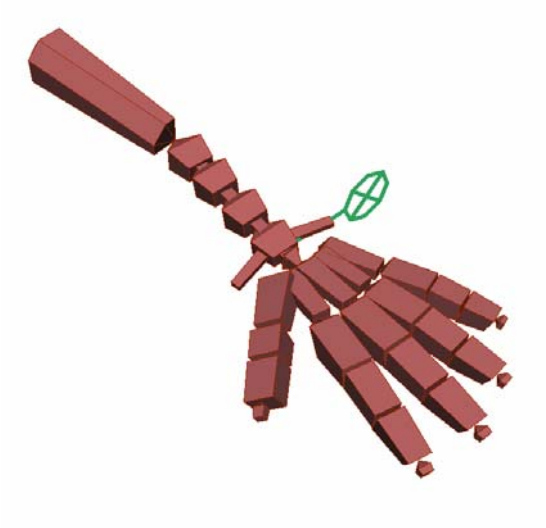
The *swing* attribute makes the foot rotate on its toes, from side to side. It might be handy if you ever need the character to step on a cigarette butt…



3.6. ARMS

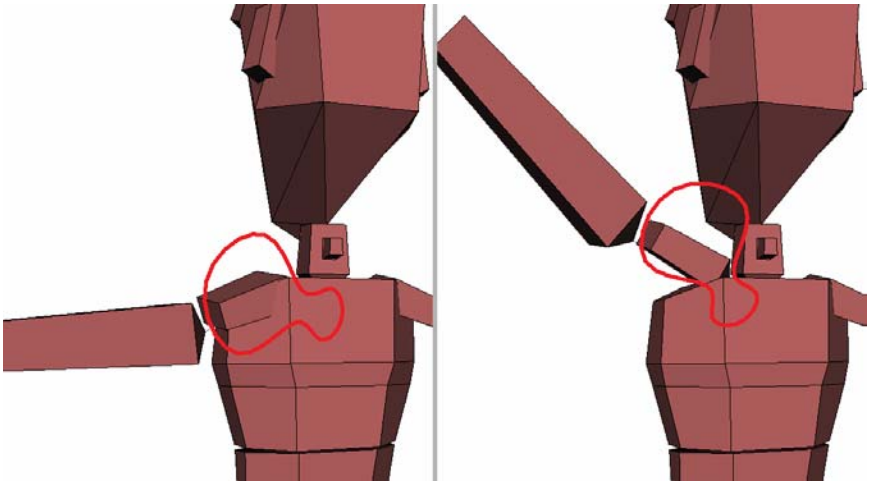
The arms are rigged in a fashion similar to the leg's. They also can be animated in FK or IK. Of course, most of the time you will want FK, but IK is nice when you want the character to put his hand against something, like a wall or a table.

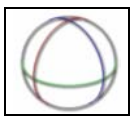
Again, to alternate between FK and IK there's a green *switch* controller, which can always be found at the character's wrist. And this controller also has the attributes necessary to make the arms *elastic*.



Clavicles

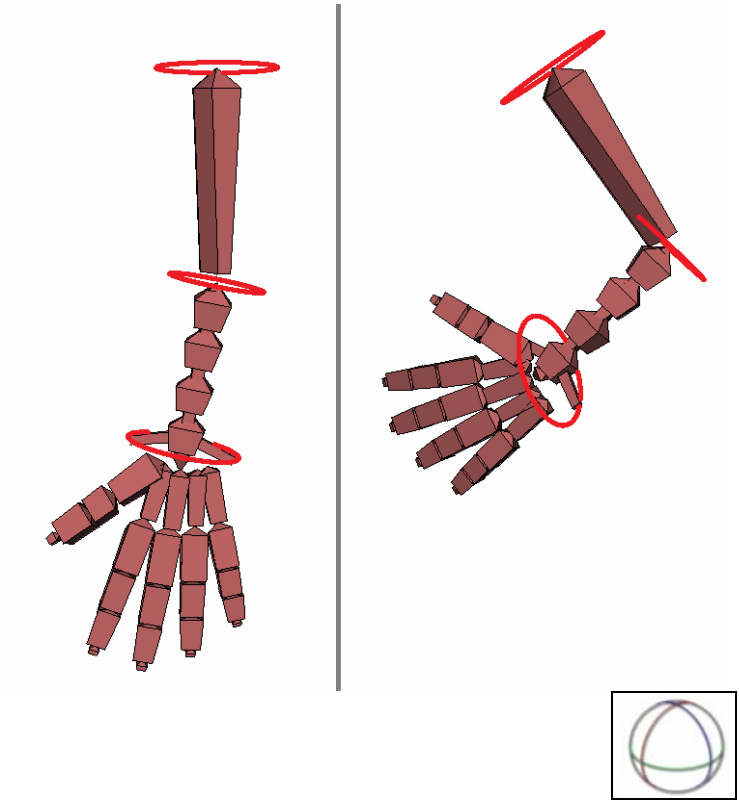
There are two rotating clavicle controllers, one for the left and one for the right one. This is a very important controller, often overlooked in other rigs. Have you tried raising your arm *without* moving your clavicle?





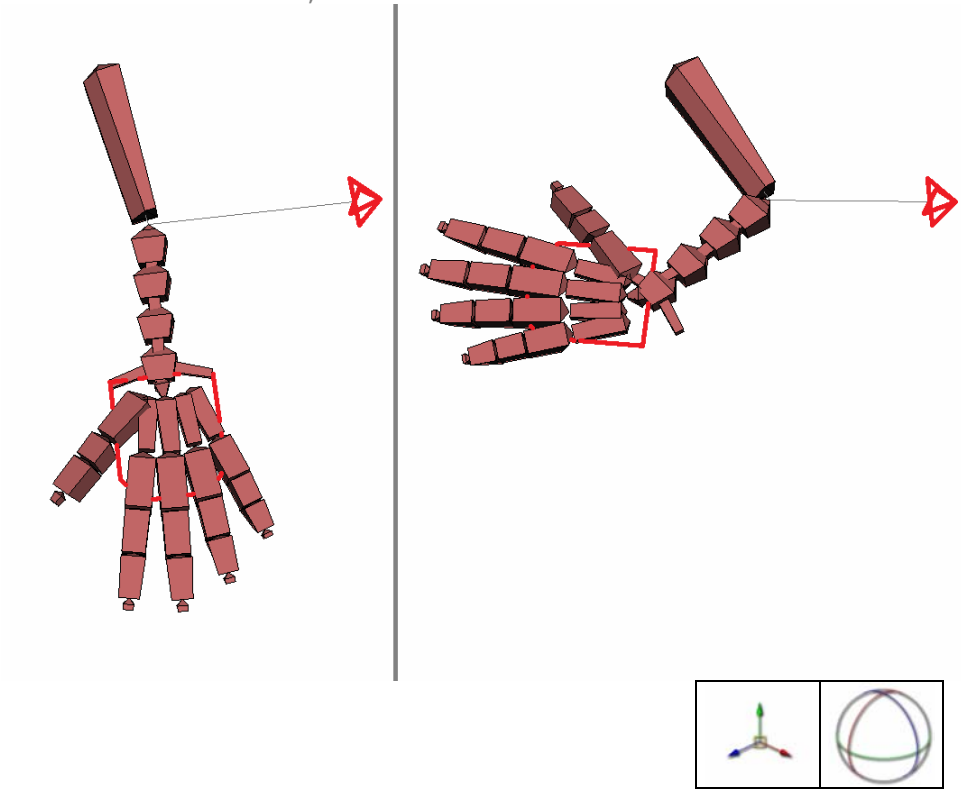
FK arms

By default, ikJoe's arms are FK. This means that there is a controller for each *joint* of the arm: one for the shoulders, one for the elbow and one for the wrist.



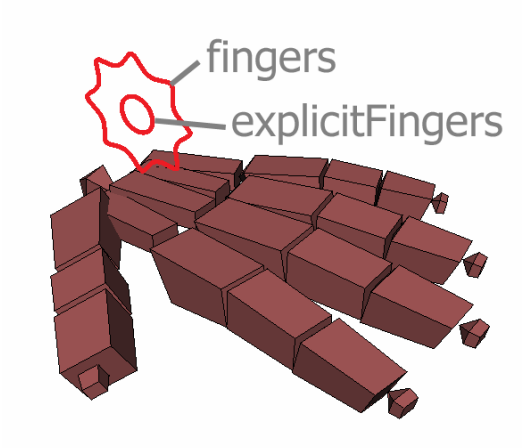
IK arms

The IK arms are made of two different controllers. One is the *wrist* controller, which indicates the exact position and rotation of the arm when in IK mode. The other one is the *elbow* controllers, which indicates the orientation of the elbow.



Fingers

The fingers are manipulated with two different controllers: the *fingers* controller and the *explicitFingers* controller. The first one will allow the animator to quickly change the rotation of each finger of the hand. The second one, which can only be made visible through the first, can be used to fine-tune the rotation of each joint until the exact wanted hand position is assumed.



3.8. THE RIG IN NUMBERS

For the curious, here's a table with some factual numbers from the ikJoe v3 rig.

178	meshes
0	NURBS surfaces
109	NURBS curves
208	joints
10	IK Handles
166	point constraints
99	orient constraints
2	scale constraints
134	clusters
42	multiplyDivides
4	blendColors
4	clamps
2	conditions
12	reverses
4	setRanges
12	distance tools
145	animation curves
139	unit conversions

4. CREDITS

The ikJoe model was made by Daniel Lara (www.pepeland.com)

The ikJoe rig v3.0 can be freely downloaded from www.rigging101.com

Special thanks to

My good friend Javier "*Goosh*" Solsona, for giving me support and ideas at all times, and for ikJoe v1… Way to go, J !!

Jason Schiefler for his awesome tutorials on rigging.

Michael Ferraro for making me think of elastic extremities and a new mouth system.

Kyle Balda for his fantastic animation master class, filled with rigging concepts.

Arslan Elver for a little idea that led to giving eyebrows to the ikJoe.

The *10 Second Club*… just because ;-)

Contact

You can e-mail me at lluisllobera@hotmail.com

For samples of my work and some free MEL scripts, visit www.lluisllobera.com