



## Index

|                                                    |    |
|----------------------------------------------------|----|
| Philosophy of the Character Rigger.....            | 3  |
| Limitations.....                                   | 4  |
| 1. Scripts.....                                    | 5  |
| 2. Installation.....                               | 6  |
| 3. Description of the Rig.....                     | 6  |
| 3.1 Features.....                                  | 6  |
| 3.2 Additional Control Attributes.....             | 7  |
| 3.3 Inverse Kinematics and Forward Kinematics..... | 9  |
| 3.4 Pickwalking.....                               | 9  |
| 4. Functions and Settings.....                     | 10 |
| 4.1 Character.....                                 | 10 |
| 4.2 Define Character.....                          | 10 |
| 4.2.1 Spine.....                                   | 11 |
| 4.2.2 Head.....                                    | 12 |
| 4.2.3 Arms.....                                    | 13 |
| 4.2.4 Legs.....                                    | 15 |
| 4.2.5 Fingers.....                                 | 17 |
| 4.2.6 Toes.....                                    | 19 |
| 4.2.7 Main Ctrl.....                               | 20 |
| 4.3 Complete Rigging.....                          | 20 |
| 4.4 Add Custom Character Parts.....                | 21 |
| 4.5 Assign LowRes Geometry Names.....              | 22 |
| 5. Preparing for Rigging.....                      | 23 |
| 6. Tools.....                                      | 24 |
| 6.1 Pose Mirroring.....                            | 24 |
| 6.2 CharRig Store.....                             | 25 |
| 6.3 Switch Character Set.....                      | 26 |
| 6.4 Toggle Set Visibility/Display Status.....      | 26 |
| 7. Quick Start.....                                | 27 |
| 7.1 Rigging the Demo Scene Character.....          | 27 |
| 7.2 Rigging of a Custom Character.....             | 28 |
| 7.3 Skinning with an Established Rig.....          | 29 |
| 7.4 Adding Individual Joints and Controls.....     | 29 |

### DISCLAIMER

THE AUTHOR RESERVES THE RIGHT NOT TO BE RESPONSIBLE FOR THE TOPICALITY, CORRECTNESS, COMPLETENESS OR QUALITY OF THE SOFTWARE PROVIDED. THE SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS" AND THE AUTHOR EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, TO THE EXTENT PERMITTED BY LAW INCLUDING BUT NOT LIMITED TO WARRANTIES OF SATISFACTORY QUALITY, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE OR ANY MATERIALS.

THE AUTHOR HAS MADE EVERY EFFORT POSSIBLE TO ENSURE THAT THE SOFTWARE IS FREE OF ANY BUGS OR ERRORS, HOWEVER IN NO WAY IS THE SOFTWARE TO BE CONSIDERED ERROR OR BUG FREE. BY DOWNLOADING AND RUNNING THE SOFTWARE YOU ASSUME ALL RISK WITH RESPECT TO THE SOFTWARE PROGRAM AND ACKNOWLEDGE THAT THE SOFTWARE MAY HAVE DEFECTS OR DEFICIENCIES, AND THAT CERTAIN FEATURES AND FUNCTIONS IN SUCH SOFTWARE MAY NOT BE IN SUBSEQUENT RELEASES OR BE AVAILABLE THROUGHOUT ALL TIMES.

LIABILITY CLAIMS REGARDING LOSSES, DEMANDS OR DAMAGES OF ANY KIND WHATSOEVER WITH RESPECT TO ANY INFORMATION AND/OR SERVICES PROVIDED BY THE AUTHOR INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL LOSS OR DAMAGES, COMPENSATORY DAMAGES OR LOSS OF PROFITS OR DATA CAUSED BY THE USE OF ANY INFORMATION OR SOFTWARE PROVIDED, INCLUDING ANY KIND OF INFORMATION WHICH IS INCOMPLETE OR INCORRECT, WILL THEREFORE BE REJECTED. ALL DATA IS NOT-BINDING AND WITHOUT OBLIGATION.

PARTS OF THE SCRIPTS OR THE COMPLETE SOFTWARE INCLUDING ALL DOCUMENTATION AND INFORMATION MIGHT BE EXTENDED, CHANGED OR PARTLY OR COMPLETELY DELETED BY THE AUTHOR WITHOUT SEPARATE ANNOUNCEMENT.

## Philosophy of the Character Rigger

Every tool handling the automated rigging process of a digital character is based on an individual philosophy or at least on the needs of the person creating that tool. In a very few cases the basic idea is to create a tool that suits just about any situation for anyone using it, if this is possible at all. Even if the most common production requirement is to create a biped rig for characters that are mostly alike in their basic structure, there are often exceptions regarding refinements or specialties for anatomy, equipment, influence of environment or production conditions, which can't be handled by a global solution.

A more common case is that aside from these all-rounder-tools rigging scripts are being written for the individual needs of the artist creating and using them to optimize the personal workflow and the task of rigging itself. Therefore the number of available tools is increasing but they do not always fit the personal expectations resulting in the search for alternatives or even writing own scripts.

The *Character Rigger* has evolved this way. Many scripts available for free have common and practical rigging techniques implemented but lacking full integration of known and applied methods in any possible aspect. For example it is very common to lock and hide all unnecessary attributes of the animation controls in the channel box for a cleaner rig, also avoiding that these attributes can be animated by accident, making it easier for an animator to focus on the artistic animation process. Unfortunately this concept is mostly only applied on direct animation controls leaving other nodes in the hierarchy still accessible and animatable. Though it is in the hand of the animator to leave these attributes alone, it would be more logical to include all nodes in the concept.

It is also imperative for a rig to be logical and visually clean avoiding the search for hidden animation controls slowing down the animation process. This is often amplified by the fact that the creation of the rig, as well as the rig itself is not well documented.

The goal of the Character Rigger is to establish a most complete and productive rig, consisting of common rigging methods and concepts tied together for easy animation and handling

1. Clean and structured hierarchy
2. Clean naming of all nodes; importing of other characters does not result in double names or renaming
3. Organization with character sets, sets and layer
4. Iconic representation of the animation controls and widely spread use of custom attributes
5. Only animatable attributes on all nodes are listed in the channel box
6. Handling of the rig with best possible flexibility
7. Clean setup though leaving numerous animation possibilities
8. Extended toolset for pickwalking, pose mirroring, storing and loading of poses/animations and joint orientation
9. Integration of additional joints and controls in the character structure, such as layer and character sets after creation of the rig
10. Limited number of constraints for a fast rig
11. Documentation of all specialties and facts of the rig and the toolset

The *Character Rigger* is aiming towards the main goal for an automated setup of digital characters, namely establishing an efficient and flexible rig for animation. Through this workflow it is not the intention to generate a guide-rig which just has to be modified by the user to fit the character-skin. These guides are often limited in their flexibility and do not suit the need of any situation. It is also the question if the manual creation of joints with the use of efficient tools needs more time than the fitting of a guide-rig lacking flexibility by having limitations.

Though it is possible that the Character Rigger is not suited for everyone whose goal it is to create a rig with the least amount of clicks and work it is meant for those to gain a rig, flexible and string in function, with a little amount of work.

At last it should be noted that with whatever automated rigging tool the user tries to reach the production goal, individual advantages or disadvantages of the different scripts should be weighted against each other. This is only possible if the individual specialties are transparent to the user, giving an insight of what is happening in the rig or during the creation of it. This is the idea of this documentation.

## Limitations

1. The *Character Rigger* does not create a guide-rig. These guides are often used to create a default skeleton for the user which must then be fit to the existing character, making the manual creation of the joint structure unnecessary.  
This hierarchical joint structure including the correct joint orientation must be created and given by the user. The *Character Rigger* will then work with these given joints.
2. The size of the automatically created controls is not interactively controllable through the interface of the *Character Rigger*. The workflow assumes that the scale factor for the control is set by the user before creating this part of the rig for testing purposes. If the size of the control is unsatisfactory the undo function must be used.  
The scale of the controls depends on the individual size of the character making it mostly unnecessary to adjust the given default scale factors.
3. The *Character Rigger* is designed to work on two-legged humanoid characters. Further applications are currently not planned.

## 1. Scripts

### Character Rigger Scripts

- icCharRigSetup.mel installation of the shelf buttons
  - icCharRigStartUp.mel initializes important scripts upon maya startup
  - icCharRigUI.mel starts the Character Rigger script
  - icCharRigPickWalkToggle.mel toggles the hotkey assignment for pickwalking
  - icCharRigIkFkToggle.mel toggles between inverse and forward kinematic of the selection
  - icCharRigMirrorControls.mel copies character poses from one side to the other
  - icCharRigSwitchCharSet.mel switches to the character set of the current control
  - icCharRigStoreUI.mel starts the UI for loading/storing poses and animations
  - icCharRigToggleSetStatus.mel toggles the visibility and display status of controls and sets
  - icCharRigRecallWP.mel helper script for correct placement of the controls
  - icCharRigOrientJoint.mel script to correctly orient the local rotation axes of the joints
- 
- icCharRigElements.mel contains all names for automatic naming
  - icCharRigAddMsgAttr.mel
  - icCharRigBuildArm.mel
  - icCharRigBuildControlProcs.mel
  - icCharRigBuildFoot.mel
  - icCharRigBuildHand.mel
  - icCharRigBuildHead.mel
  - icCharRigBuildLeg.mel
  - icCharRigBuildMainControl.mel
  - icCharRigBuildSpine.mel
  - icCharRigBuildStretching.mel
  - icCharRigCollectCharacter.mel
  - icCharRigGetCharacterUI.mel
  - icCharRigHelpProcs.mel
  - icCharRigPickWalk.mel
  - icCharRigPickWalkSetup.mel
  - icCharRigReadShortName.mel
  - icCharRigStore.mel

### Icons



**CharRig UI**



**Reverse Character Pose**



**IkFk Toggle**



**PickWalk Toggle**



**Switch to Character Set**



**Toggle Set Vis.**



**Mirror Character Set**



**Switch to Subcharacter Set**



**Toggle Set Displ.**



**Mirror Control**



**CharRig Store**



**Orient Joint**



**Recall World Position**

## 2. Installation

1. Copy all Character Rigger scripts to the following directory:

(Win) **C:\Documents and Settings\*userName*\My Documents\maya\scripts**  
(Mac) **//User/Library/Preferences/Autodesk/Maya/scripts**

2. Copy all icons to the following directory:

(Win) **C:\Documents and Settings\*userName*\My Documents\maya\*x.x*\prefs\icons**  
(Mac) **//User/Library/Preferences/Autodesk/Maya/*x.x*\prefs/icons**

3. Start Maya

4. In Maya type in the following command into the command line:

**icCharRigSetup;** (enter with Ctrl + Return)

5. In the appearing dialogue type in the name of the shelf-tab, where the icons should appear.

6. Start the Character Rigger with the new icon in the shelf.

## 3. Description of the Rig

### 3.1 Features

#### 1. Complete rigging of a character

The character will automatically get fully rigged with the necessary animation controls to allow a streamlined, flexible and intuitive animation workflow. The rig contains the following features:

1. FK-control of the neck and head; either depending on the body movement or fully independently.
2. IK- and FK-animation of arms and legs.
3. Individual shoulder control with optional following to the movement of the arm.
4. Automatic setup of the lower arm with ulna and radius to allow advanced skin deformations.
5. Automatic setup of "twist joints" for the lower arm for better skin deformations, as an alternative to the ulna/radius setup. The same applies to the upper arms and the thighs.
6. Complete FK-control for the fingers through custom attributes. This also includes possible deformations of the palm, i.e. when making a fist.
7. IK- and FK-control for the upper body/back including a stretchy back. An optional shader can be generated to warn in case of extreme stretching.
8. Independent positioning of the knee from the foot to avoid joint flipping during animation.
9. "Reverse Foot" setup with custom attributes to allow flexible control over a variety of motions.
10. Complete FK-control for the toes through custom attributes.
11. Stretchable arms and legs.
12. Automatic breathing for the upper body, controllable in speed and depth.

#### 2. Character Sets

A character set will be automatically created, divided into sub-character sets for the main controls and the secondary controls, where each extremity of the character will be assigned a sub-character set.

#### 3. Organization in sets

All elements of the rig will be organized and color coded in sets. According to the general understanding the right side will be green and the left side red.

#### Note

The visibility of the set members, as well as the ability to select the nodes of a specific set is mostly dependent of the IK- or FK-mode this part of the character is currently in.

#### 4. Pickwalking

All controls will be connected to each other through custom pickwalking. More information about the use can be found in section 3.4.

#### 5. Organization of the nodes contained in a character

All elements of the rig are grouped under a group node named "character\_name". (More information can be found in section **4.3 Complete Rigging**.)

#### 6. Character Definition

A hidden locator with the name "characterDefinition\_LongName" is generated, which contains the character definition used in the UI during setup. A description of the locator can be found in section **4.1 Character**.

#### 7. Connection of all important nodes

Upon creation of the rig all important nodes, such as joints and controls, will get a custom message attribute assigned to them, to help finding certain nodes after completing the rig. All message attributes conclude in the group node "character\_LongName", making it possible to list all connected nodes in the hypergraph very easily. (A more specific description can be found in section **4.3 Complete Rigging**.)

#### 8. Clean channel box

Left in the channel box are only the attributes needed for animation. This applies not only for the attributes of the animation controls but all other nodes in the rig. This leaves attributes untouched that are not relevant for animation even during excessive pickwalking through the hierarchy. All hidden attributes are also locked.

## 3.2 Additional Control Attributes

The different controls of the rig are partly equipped with custom attributes to control various aspects of animation.

#### Head Control

- **Head Neck Follow (Range 0-1)**

Defines, how much the movement of the body translates through the neck and head. A zero value leaves the neck and head independent from the body.

#### Upper Body Control

- **Active**

Activates the automatic breathing for the upper body. It controls the *Node State* Attribute of the connected Expression.

- **Speed**

Controls the breathing speed.

- **Depth**

Controls the breathing depth.

## L/R Hand Control

- **Arm Wrist Rotate**

Drives the rotation of the hand around the lower arm axis to avoid gimbal locking. It also controls the radius/ulna rotation of the lower arm.

- **Lock Hand (Range 0-1)**

Allows for translation of the hand independent from the arm. Usually the hand follows the motion of the arm, but in the case of holding on to something the hand position must be controlled separately.

- **Auto Shoulder (Range 0-1)**

Defines the strength of how much the movement of the arm translates the shoulder. Aside from this automatism the shoulder can be individually controlled.

- **Elbow Follow (Range 0-1)**

Defines, how much the elbow control follows the arm to help the animation process. This makes it mostly unnecessary to always move the elbow when repositioning the hand control. (default 0.7)

- **Stretch (Range 0-1)**

Enables the arm stretching.

## L/R Finger Control

- Contains the custom attributes to control the individual finger joints.

## L/R Foot Control

- **IK Blend**

This attribute cannot be adjusted and is just for visual reference if the arm is in IK- or FK-mode.

- **Heel Pivot Up**

Raises and lowers the foot around a pivot at the heel.

- **Heel Pivot Twist**

Turns the foot left and right around a pivot at the heel.

- **Heel Lift**

Lifts the heel by pivoting the foot on the ball joint.

- **Ball Twist**

Turns the foot left and right around the ball joint.

- **Tip Toe**

Raises and lowers the foot around the toe tip.

- **Toe Twist**

Turns the foot left and right around the toe tip.

- **Side Roll**

Rotates the foot around the long axis to allow a side roll around the ball joint.

- **Toe Wiggle**

Rotates the toe tip up and down.

- **Stretch (Range 0-1)**

Enables the leg stretching.

## L/R Toe Control

- Contains the custom attributes to control the individual toe joints.

## 3.3 Inverse Kinematics and Forward Kinematics

The arms and legs of the character can be controlled through either IK or FK. To switch mode with the *IK/FK Toggle* shelf button just select a control or a joint of the appropriate arm or leg.

When in IK-mode only the IK controls are visible and selectable. This is controlled by the status of the individual layers. When in FK-mode only the FK controls and joints are visible and selectable.

The following arm controls and joints can be selected when in the appropriate mode:

### IK Mode

Elbow Control  
Hand Control

### FK Mode

Shoulder Joint  
Up Arm Joint  
Low Arm Joint

The following leg controls and joints can be selected when in the appropriate mode:

### IK Mode

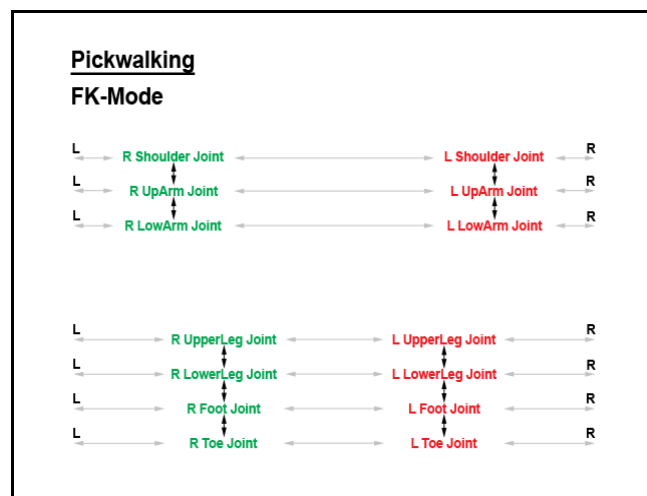
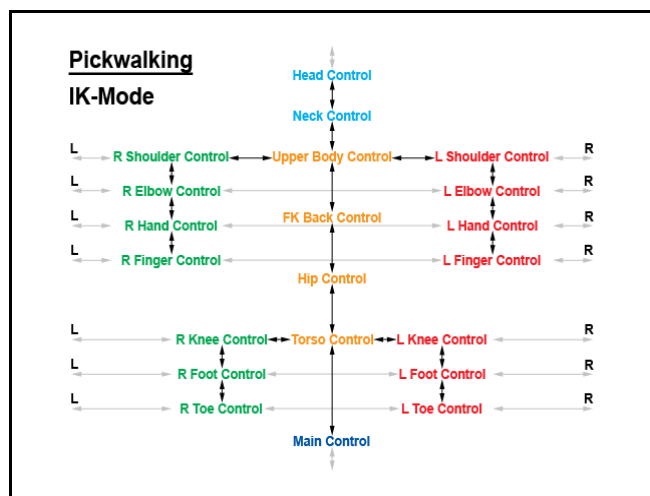
Foot Control

### FK Mode

Upper Leg Joint  
Lower Leg Joint  
Foot Joint  
Toe Joint

## 3.4 Pickwalking

The rig and the controls are equipped with a customized pickwalking to allow fast switching between them. All included nodes contain a custom attribute to connect to the neighboring nodes.



When building the rig via *Complete Rigging/Build Char.* the default pickwalking will be switched over to the *Character Rigger* pickwalking. The additional script *icCharRigPickWalkToggle.mel* allows for convenient switching between the default pickwalking and the *Character Rigger* pickwalking. The current status can be displayed in the *Command Feedback* line of Maya at the time of toggling.

### Note

It is advised to keep a copy of the current hotkey settings before building the rig or using the toggle hotkey function as the current settings will be overwritten in the preferences files.

## 4. Functions and Settings

### 4.1 Character

For building the rig it is important to name the character. For naming the *Short Name* is substantial, but the *Long Name* can be left out.

#### Long Name

Is just used to name the node under which the entire character is grouped. If this field is left blank the character node will get the *Short Name*.

| Character          |                                      |                                              |
|--------------------|--------------------------------------|----------------------------------------------|
| Long Name          | <input type="text" value="JohnDoe"/> | <input type="button" value="Put Character"/> |
| Short Name (4 max) | <input type="text" value="JOHN"/>    | <input type="button" value="Get Character"/> |

#### Short Name

This field can contain up to four characters to uniquely identify the nodes during the creation of the rig and the workflow afterwards. This avoids double names or renaming of nodes when importing other characters into the scene.

#### Note

The functions *Put Character* and *Get Character* are just for manually saving and loading different character definitions without having to build a complete rig.

When creating a rig with *Build Character* in the section *Complete Rigging* all the information contained in the *Character Rigger* will be automatically saved in a locator. This information can be later read back into the *Character Rigger* with *Get Character* to allow for checking upon the creation values.

#### Put Character

Saves all settings of the *Character Rigger* to a hidden locator as user attributes. The locator is named "characterDefinition\_*characterName*" and should not be renamed. To identify the character definition at a later point it is necessary to have the name of the character begin with an underscore and which should be the only underscore in the name of the locator.

The user attributes are locked and should not be modified.

When a character definition with the same name is already existing in the scene it will get overwritten.

#### Get Character

Searches for a locator that contains the user attribute *character\_icCRconnection*. If the name has been modified (see *Put Character*), the character definition cannot be found.

All characters found in the current scene will be listed and loaded into the *Character Rigger*.

### 4.2 Define Character

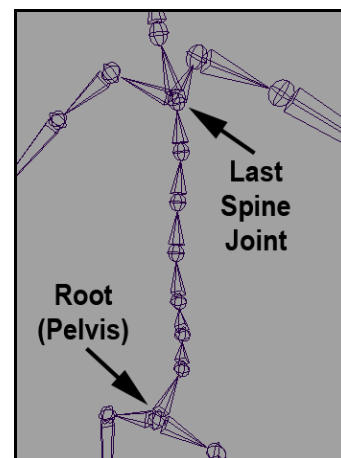
#### Create *bodypart* Rig

Builds the specified part of the character with an automated rigging. It's used mainly for testing the size of the controls or this part of the character. All temporary rigs created through this section can be undone by the regular undo function in Maya.

## 4.2.1 Spine

The back should be at least built by 4 to 5 joints to allow smooth deformations of the spine. The back is build with Spline-IK to allow such flexible deformations. On top of this the spine will be controlled by FK to make the back bend through the Spline-IK. The *FK Back Controller* controls the bending, the *Upper Body* and *Hip Controls* adjust the motions in the shoulder and hip area to enhance the flexibility. The entire upper body translation and rotation is controlled via the *Torso Control*.

An optional shader can be generated when the back is build. It shows a warning color on the connected low resolution geometry when the back has been stretched too far. In a natural spine this stretching occurs in small amounts with different types of movements and thus the stretching has been implemented into the rig. The shader can be adjusted to show the warning color beginning at a certain degree of stretching.



### Assign Character Joints

#### Root (Pelvis)

The joint representing the pelvis and acting as the parent joint for the entire hierarchy.

#### Last Spine Joint

The last joint of the spine followed directly by the neck joint.

#### FK Back Joints

The number of automatically created joints for controlling the FK-back bending efficiently. Three joints should be sufficient in most cases as the deformation is applied through the given spine joints and the controlling spline-IK.

#### Upper Body Control Scale / FK Back Control Scale / Hip Control Scale / Torso Control Scale

Individual adjustable scale for the controls of the shoulder / the FK-back / the pelvis / the entire upper body.

The starting scale of the controls depends on the overall size of the character.

#### Upper Body Control Position

Places the *Upper Body Control* either at pelvis or shoulder depth to respect different default poses of the character.

#### Create Stretch Warning

Activates the generation of the *Stretch Shader* for the low resolution geometry.

#### Stretch Warning Start (%)

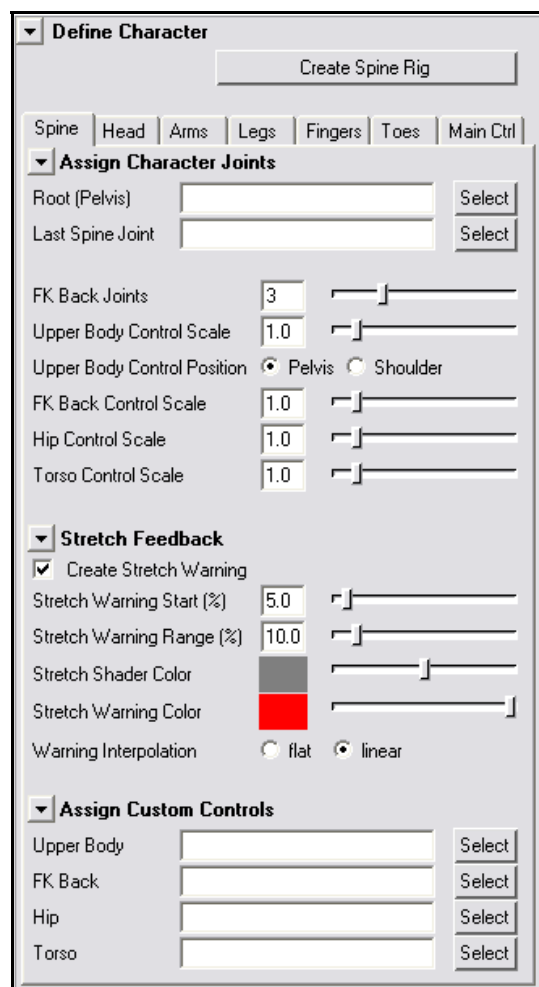
Defines the percentage of allowed stretching before the shader starts to show the warning color.

#### Stretch Warning Range (%)

Defines the range of stretching in percent for the shader to increase it's intensity from 0% to 100%.

#### Stretch Shader Color

Defines the color of the shader in it's unstretched state. The grey value is equal to the default lambert shader in Maya.



## Stretch Warning Color

Defines the color in the shaders incandescence channel if the stretch value is 100% and above.

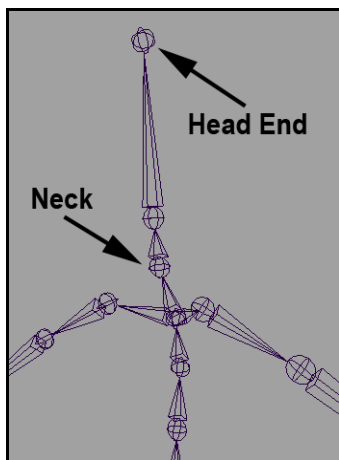
## Warning Interpolation

Sets the type of interpolation the warning color goes through from 0% to 100%. *Flat* sets the set driven key curve to ease out/ease in.

## Assign Custom Controls

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.

### 4.2.2 Head



The head has two controls assigned to guide the FK motion of the neck and head. Both controls are independent from each other to allow as much freedom as possible during animation. As a default setting both neck and head are independent from the movement of the body. Though it is possible to let the movement of the body influence the neck and head through a custom attribute on the *Head Control*.

## Assign Character Joints

### Neck

The neck joint of the character.

### Head End

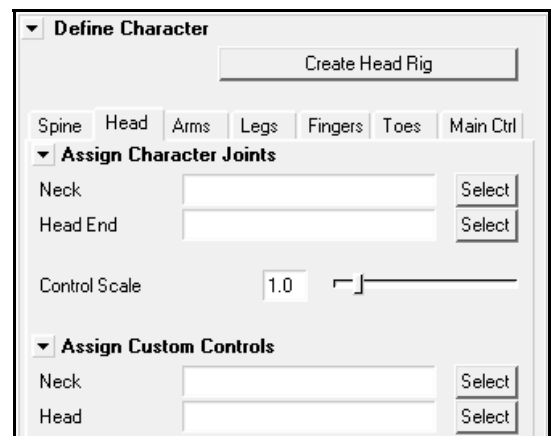
The last joint at the top of the head.

## Control Scale

Individual adjustable scale for the *Neck-* und *Head Controls*. The starting scale of the controls depends on the overall size of the character.

## Assign Custom Controls

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.



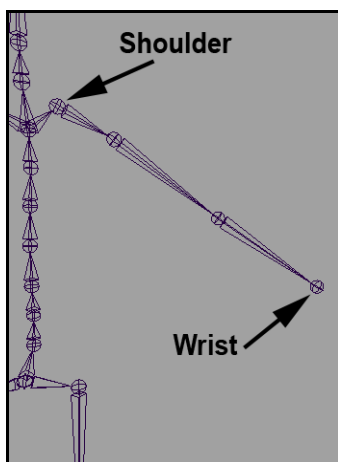
## 4.2.3 Arms

The final arm rig is based on two joint hierarchies which allow both IK- and FK-animation. The rig also supports an automatic shoulder, following the movement of the arm. The degree of transmitted motion to the shoulder is adjustable but deactivated by default.

To yield a better deformation of the lower arm two additional joints are generated (*lowArmTwist1/2*) which are placed along the given lower arm joint. Turning the hand will result in a distributed rotation along the lower arm.

During the creation of the rig a radius and ulna joint are being added to the given joint structure to allow a more realistic rotation of the lower arm. The setup is based on real anatomy and allows the application of a muscle driven deformation. These joints are not defined as skinning joints by default.

### Assign Character Joints



#### L/R Shoulder

The first joint of the arm branching directly from the back joint followed by the upper arm joint.

#### L/R Wrist

The wrist joint. This joint will also get applied to the field *Wrist* in the section *Assign Character Joints* of the *Fingers*.

#### Radius/Ulna Position Scale

Sets the relative position between the radius and ulna joint of the lower arm. The

starting position depends on the overall size of the character and is sufficient in most cases.

#### Control Scale

Individual adjustable scale for the controls of the arms. The starting scale of the controls depends on the overall size of the character.

#### Limit FK Joint Rotation

When animating the arms through FK a limited degree of freedom can be useful. Especially when rotating the shoulder or the lower arm. If set to *limit* certain axes of the target joints are locked. Unfortunately this leads to an error message in Maya when switching from IK to FK, though it is only a warning message not influencing the animation process. Because of this behavior the limitation is not activated by default.

#### Stretchable Arms

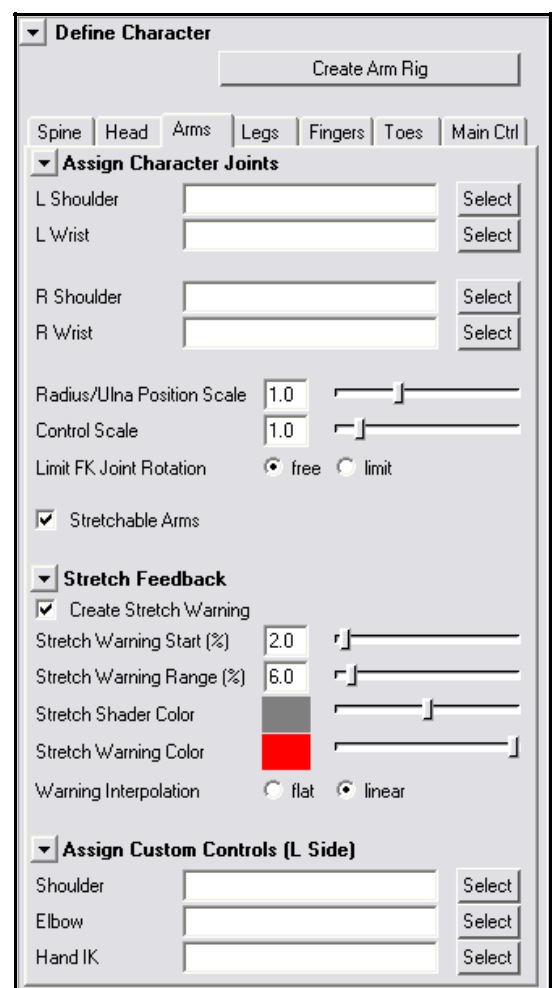
Implements the possibility to make the arms stretchable. If deactivated no stretching will be possible.

#### Create Stretch Warning

Activates the generation of the *Stretch Shader* for the low resolution geometry.

#### Stretch Warning Start (%)

Defines the percentage of allowed stretching before the shader starts to show the warning color.



### **Stretch Warning Range (%)**

Defines the range of stretching in percent for the shader to increase it's intensity from 0% to 100%.

### **Stretch Shader Color**

Defines the color of the shader in it's unstretched state. The grey value is equal to the default lambert shader in Maya.

### **Stretch Warning Color**

Defines the color which the shader is assigned in it's incandescence channel if the stretch value is 100% and above.

### **Warning Interpolation**

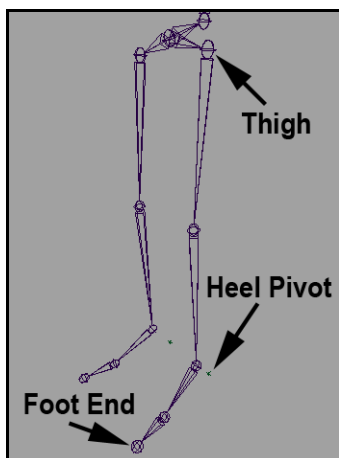
Sets the type of interpolation the warning color goes through from 0% to 100%. *Flat* sets the set driven key curve to ease out/ease in.

## **Assign Custom Controls**

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.

The custom controls are only requested for the left side of the character (as seen from the characters point of view). In the case of creating two arms the controls will be automatically duplicated and mirrored to the opposite side.

## 4.2.4 Legs



The foot rig is based in the reverse-foot-setup and equipped with a set of custom attributes to allow flexible and easy accessible control and animation.

### Assign Character Joints

#### L/R Thigh

Defines the thigh joint of the character.

#### L/R Foot End

The last joint at the tip of the foot directly after the ball joint. In case of individual toes these joints are branching off the *Ball-Joint*. The given joint here is then separate from these toe joints.

#### L/R Heel Pivot

To allow a higher degree of freedom an additional pivot point at the heel is necessary, as i.e. a walk cycle benefits from it. Since every character has a different physique the location of this pivot point cannot be calculated and thus gets marked by a locator.

#### Control Scale

Individual adjustable scale for the controls of the legs. The starting scale of the controls depends on the overall size of the character.

#### Limit FK Joint Rotation

When animating the arms through FK a limited degree of freedom can be useful. Especially when rotating the lower leg. If set to *limit* certain axes of the target joint are locked. Unfortunately this leads to an error message in Maya when switching from IK to FK, though it is only a warning message not influencing the animation process. Because of this behavior the limitation is not activated by default.

#### Stretchable Legs

Implements the possibility to make the legs stretchable. If deactivated no stretching will be possible.

#### Create Stretch Warning

Activates the generation of the *Stretch Shader* for the low resolution geometry.

#### Stretch Warning Start (%)

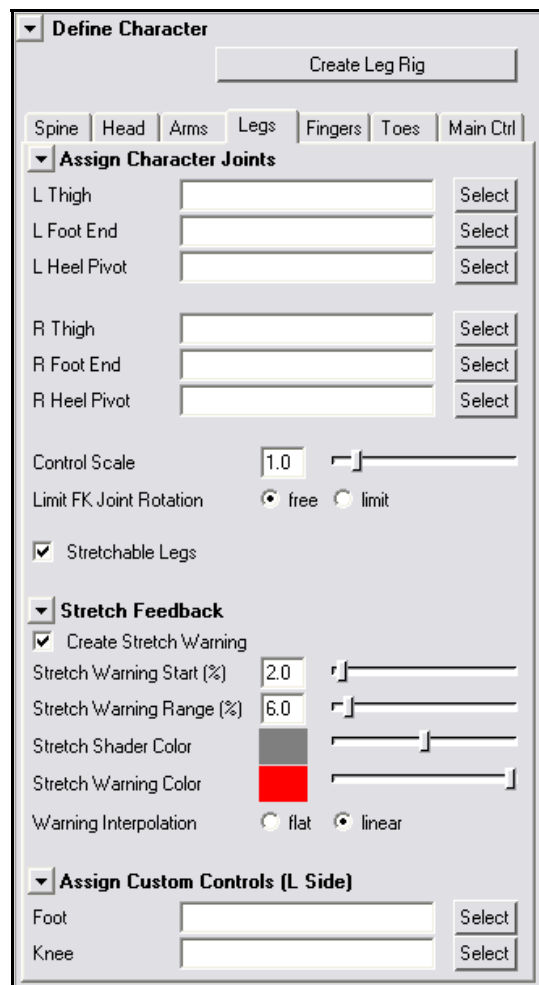
Defines the percentage of allowed stretching before the shader starts to show the warning color.

#### Stretch Warning Range (%)

Defines the range of stretching in percent for the shader to increase it's intensity from 0% to 100%.

#### Stretch Shader Color

Defines the color of the shader in it's unstretched state. The grey value is equal to the default lambert shader in Maya.



### **Stretch Warning Color**

Defines the color which the shader is assigned in it's incandescence channel if the stretch value is 100% and above.

### **Warning Interpolation**

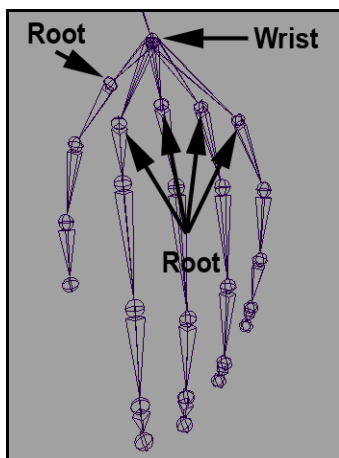
Sets the type of interpolation the warning color goes through from 0% to 100%. *Flat* sets the set driven key curve to ease out/ease in.

## **Assign Custom Controls**

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.

The custom controls are only requested for the left side of the character (as seen from the characters point of view). In the case of creating two legs the controls will be automatically duplicated and mirrored to the opposite side.

## 4.2.5 Fingers



The fingers are controlled via custom attributes and set driven keys through a unified control. All joints are animatable for best flexibility.

### Assign Character Joints

#### L/R Wrist

The joint of the wrist. If the joints for the arms have been already assigned the wrist joints are already automatically entered in these fields.

#### L/R Thumb/Index/Middle/Ring/Pinky Root

The first joints of the respective fingers. The root joint, or metacarpal joint (metacarpus) allows the deformation of the palm, i.e. when making a fist. If the character has less than five fingers the respective fields remain empty.

#### Hand Control Scale (World Units)

Individual adjustable scale for the controls of the fingers. The scale of the controls is defined in *World Units* in Maya and thus, in contrast to the controls of other body parts, not dependent on the overall size of the character.

### Define Fingers

#### Thumb Joints

Defines how many joints the thumb has.

#### Finger Joints

Defines how many joints the fingers have.

#### SetDrivenKey Interpolation

Sets the type of interpolation blending between the minimum and maximum angles of the joints. *Flat* sets the set driven key curve to ease out/ease in.

#### Thumb Base: Left/Right

The max/min degree of freedom for the left and right motion of the first thumb joint. Since this joint usually has a higher range of movement in comparison to the other fingers it's degree of freedom gets listed separately.

#### Thumb Base: Roll

The rotation of the first thumb joint around it's long axis for an extended range of motion.

#### Root: Up/Down

The max/min degree of freedom for the up and down motion of the metacarpal joint. This can be necessary for making a fist or holding something.

#### Root: Left/Right

The max/min degree of freedom for the left and right motion of the metacarpal joint. This motion is not necessarily useful and only implemented for better flexibility in certain cases.

#### Base: Up/Down

The max/min degree of freedom for the up and down motion of the first finger joint.

**Define Character**

Create Hand Rig

Spine | Head | Arms | Legs | **Fingers** | Toes | Main Ctrl

**Assign Character Joints**

|               |                      |        |
|---------------|----------------------|--------|
| L Wrist       | <input type="text"/> | Select |
| L Thumb Root  | <input type="text"/> | Select |
| L Index Root  | <input type="text"/> | Select |
| L Middle Root | <input type="text"/> | Select |
| L Ring Root   | <input type="text"/> | Select |
| L Pinky Root  | <input type="text"/> | Select |
| R Wrist       | <input type="text"/> | Select |
| R Thumb Root  | <input type="text"/> | Select |
| R Index Root  | <input type="text"/> | Select |
| R Middle Root | <input type="text"/> | Select |
| R Ring Root   | <input type="text"/> | Select |
| R Pinky Root  | <input type="text"/> | Select |

Hand Control Scale (World Units)

**Define Fingers**

Thumb Joints  2  3

Finger Joints  3  4

SetDrivenKey Interpolation  flat  linear

|                        |                                    |                                    |
|------------------------|------------------------------------|------------------------------------|
| Thumb Base: Left/Right | <input type="text" value="60.0"/>  | <input type="text" value="-60.0"/> |
| Thumb Base: Roll       | <input type="text" value="40.0"/>  | <input type="text" value="-40.0"/> |
| Root: Up/Down          | <input type="text" value="20.0"/>  | <input type="text" value="-20.0"/> |
| Root: Left/Right       | <input type="text" value="10.0"/>  | <input type="text" value="-10.0"/> |
| Base: Up/Down          | <input type="text" value="110.0"/> | <input type="text" value="-80.0"/> |
| Base: Left/Right       | <input type="text" value="30.0"/>  | <input type="text" value="-30.0"/> |
| Base: Roll             | <input type="text" value="20.0"/>  | <input type="text" value="-20.0"/> |
| Middle: Up/Down        | <input type="text" value="120.0"/> | <input type="text" value="-30.0"/> |
| Tip: Up/Down           | <input type="text" value="90.0"/>  | <input type="text" value="-30.0"/> |

**Assign Custom Controls (L Side)**

Finger Control

**Base: Left/Right**

The max/min degree of freedom for the left and right motion of the first finger joint.

**Base: Roll**

The rotation of the first finger joint around it's long axis for an extended range of motion.

**Middle: Up/Down**

The max/min degree of freedom for the up and down motion of the second finger joint.

**Tip: Up/Down**

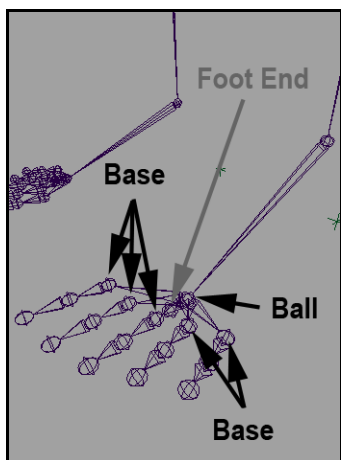
The max/min degree of freedom for the up and down motion of the third finger joint.

**Assign Custom Control (L Side)**

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.

The custom control is only requested for the left side of the character (as seen from the characters point of view). In the case of creating two hands the control will be automatically duplicated and mirrored to the opposite side.

## 4.2.6 Toes



The toes are controlled via custom attributes and set driven keys through a unified control. All joints are animatable for best flexibility.

### Assign Character Joints

#### L/R Ball Joint

The joint of the ball.

#### L/R Thumb-/Index-/Middle-/Ring-/Pinky-Base

The first joints of the respective toes. The base joint, or metatarsal joint allows the deformation of the main foot area. If the character has less than five toes the respective

fields remain empty.

#### Toe Control Scale (World Units)

Individual adjustable scale for the controls of the toes. The scale of the controls is defined in *World Units* in Maya and thus, in contrast to the controls of other body parts, not dependent on the overall size of the character.

## Define Toes

#### Toe Joints

Defines how many joints the toes have.

#### SetDrivenKey Interpolation

Sets the type of interpolation blending between the minimum and maximum angles of the joints. *Flat* sets the set driven key curve to ease out/ease in.

#### Base: Up/Down

The max/min degree of freedom for the up and down motion of the metatarsal joint.

#### Base: Left/Right

The max/min degree of freedom for the left and right motion of the metatarsal joint.

#### Middle: Up/Down

The max/min degree of freedom for the up and down motion of the first toe joint.

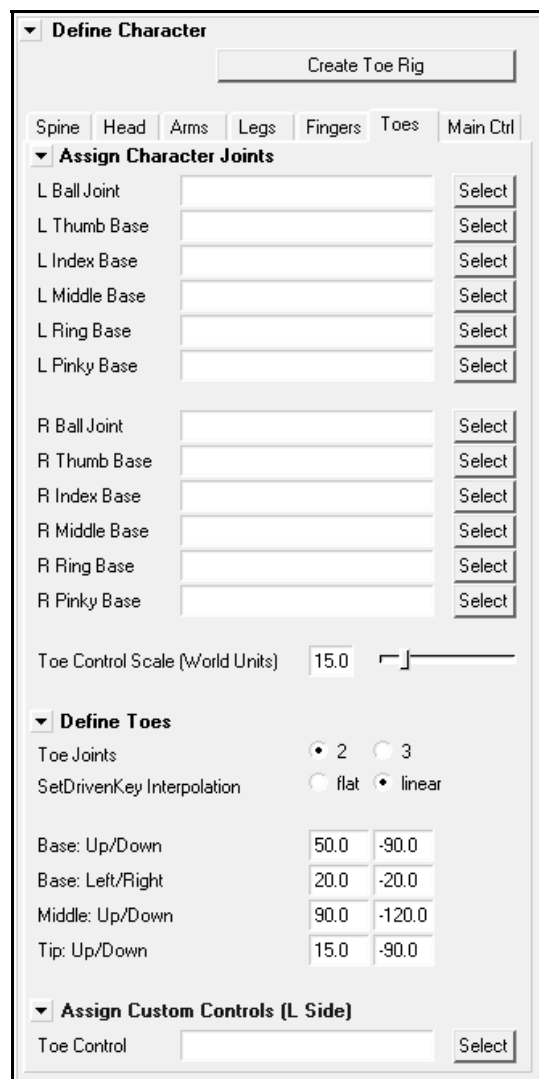
#### Tip: Up/Down

The max/min degree of freedom for the up and down motion of the second toe joint.

#### Assign Custom Control (L Side)

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.

The custom control is only requested for the left side of the character (as seen from the characters point of view). In the case of creating two feet the control will be automatically duplicated and mirrored to the opposite side.



## 4.2.7 Main Ctrl

This control allows the overall placement of the character. All joints and controls are being transformed as well when placing this control.

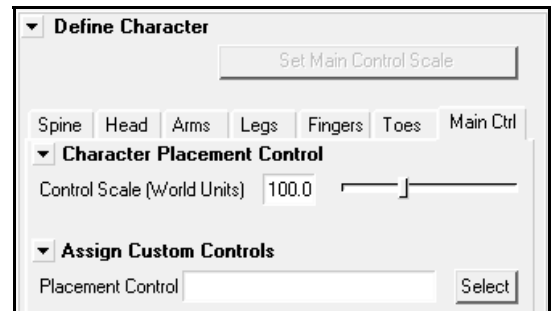
### Character Placement Control

#### Control Scale (World Units)

Individual adjustable scale for the *Main Control*. The scale of the controls is defined in *World Units* in Maya and thus, in contrast to the controls of other body parts, not dependent on the overall size of the character.

### Assign Custom Controls

Instead of having the controls automatically generated, custom objects can be used to control the rig. It is imperative to have these controls correctly aligned.



## 4.3 Complete Rigging

After entering all joints and controls in section *Define Character* the rig can be completed in one step. This is different to creating the rig in successive steps through **Create bodypart Rig** in section *Define Character*. Here not only all the different parts of the character get established by a single function but they also get connected to each other with the following features:

#### 1. Pickwalking

All controls get linked to each other through a specialized pickwalking. More information about the pickwalking can be found in section **3.4 Pickwalking**.

#### 2. Organization of the different nodes of the character

All elements of the rig are grouped under a node called "character\_name", where *name* is equivalent to, if given, to *Long Name* otherwise it will be the given *Short Name*.

All controls for the IK- and FK-behavior of the rig are grouped under the node "animationCtrls\_ShortName". They are zeroed out to regain the original pose by setting back all attributes in the channel box.

The group node "dontTouchJoints\_ShortName" consists of the complete joint hierarchy of the character. These joints are the given joints at the time of creating the rig as well as the automatically created joints, i.e. for the radius and ulna. This group also contains the low resolution geometry, parented to the respective joints.

The group node "dontTouchShortName" contains additional nodes created through the establishing of the rig. These nodes are not for the general usage of the rig and therefore not visible.

#### 3. Character Definition

An invisible locator by the name "characterDefinition\_LongName" is created carrying the character definition and settings of the *Character Rigger*. A detailed description of this locator can be found in section **4.1 Character**.

## 4. Interconnection of all important nodes

Upon creation of the rig all important nodes, such as joints and controls, will get a custom message attribute assigned to them, to help finding certain nodes after completing the rig. All message attributes conclude in the group node "character\_LongName", making it possible to list all connected nodes in the hypergraph very easily. There are eight different types of nodes:

1. skinJoint
2. IKhandle
3. IKcontrol
4. FKcontrol
5. controlJoint
6. FKjoint
7. character
8. customControl (additional controls, which control joints implemented after rig creation)

## Establish Connections

### Nodes

With this option selected all animation nodes will get connected to the group node "character\_LongName" with their custom message attribute. This setting is usually sufficient enough to help finding all animation relevant nodes of the rig.

### Nodes and Channels

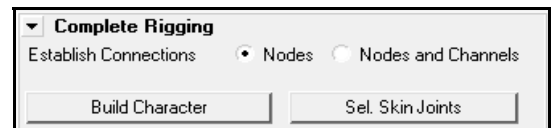
This option connects the custom message attribute of all animation nodes as well as all animatable channels of these. All joints get their rotation channels included, the spine joints also their translation channels and all other nodes their attributes from the channel box.

### Build Character

This function executes the complete rigging of the character.

### Sel. Skin Joints

Selects all joints of the character necessary for skinning. Usually these are the given joints at the point of rig creation as well as automatically created joints, i.e. the joints for radius and ulna.

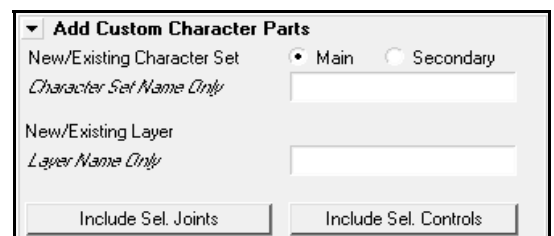


## 4.4 Add Custom Character Parts

Since not any character consists of the same features, thus making it impossible for an automated rig to obey any specialties, this section gives the option to include individual extensions to the rig. This means that not only additional nodes are named correctly but they are also added to the structure of layers and character sets.

### New/Existing Character Set

Selected nodes can be added to a new or existing character set containing main or secondary animation controls. The set *Main* contains the primary animation elements, such as arms and legs, the set *Secondary* only the secondary elements, such as fingers and toes. The textfield is for setting the name of the existing or new character set. It is not important to include the name of the character (*Short Name*) as it will be automatically added. If the set name is separated from the *Short Name* by an underscore, the underscore has to be added manually. An empty field ignores the linking to a character set.



### New/Existing Set

Selected nodes can be added to a new or existing set. The textfield is for setting the name of the existing or new set. It is not important to include the name of the character (*Short Name*) as it will be

automatically added. If the set name is separated from the *Short Name* by an underscore, the underscore has to be added manually. An empty field ignores the linking to a set.

### Include Sel. Joints

Adds the selected joints to the existing character rig. All node names will be extended by the *Short Name* of the character. All nodes will also get a custom message attribute applied which connects them to the group node like any other node in the rig. This connection is depended on the setting **Nodes** or **Nodes and Channels** in section **4.3 Complete Rigging / Establish Connections**, connecting either only the nodes or also all animatable channels from the channel box.

The process of including selected joints to the rig can also add these nodes to a character set and/or to a layer by choice.

### Include Sel. Controls

Adds the selected controls to the existing character rig. All node names will be extended by the *Short Name* of the character. All nodes will also get a custom message attribute applied which connects them to the group node like any other node in the rig. This connection is depended on the setting **Nodes** or **Nodes and Channels** in section **4.3 Complete Rigging / Establish Connections**, connecting either only the nodes or also all animatable channels from the channel box.

The process of including selected controls to the rig can also add these nodes to a character set and/or to a layer by choice.

### Note

By adding new joints/controls to the rig the existing hierarchy is not being adjusted and must be modified manually. The same applies to the visibility of attributes in the channel box of the main nodes and their input/output nodes.

## 4.5 Assign LowRes Geometry Names

During animation it is very common to use a low resolution geometry of the character as a visual guide for volume and motion. This is realized through separating the different parts of a character into individual pieces and linking them to the respective joints usually done by simple parenting. To identify the different character geometry parts it is important to name them correctly which can easily be done with the *Character Rigger*.

All character joints will be listed in this section with their names they have gotten during the creation of the rig. For the partly automated naming of the geometry pieces it is only necessary to select the appropriate part and the corresponding name in the list. The selected geometry will get the name of the joints plus the extension "\_Geo\_". After naming all the low resolution geometry the parenting can be done with either *Attach Selected* or *Attach All*.

### Root (Pelvis)

The root joint of the entire hierarchy must be entered here to list all joints of the character. If a root joint has been already assigned under *Define Character / Spine / Root (Pelvis)* the joint will also automatically appear in this field.

### Refresh

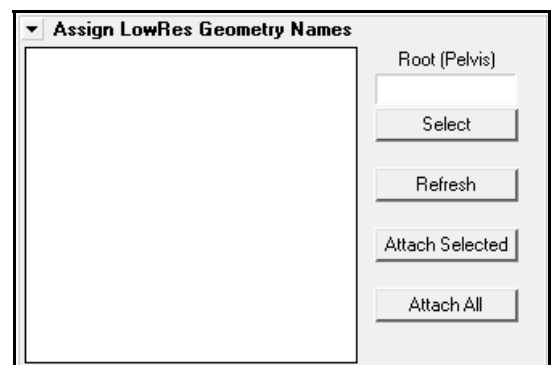
Refreshes the list if the name of the joints has been modified after opening this section. When opening this section the list is automatically refreshed.

### Attach Selected

Only the selected parts of the character will be parented to the corresponding joints.

### Attach All

All parts of the character with the correct name will be parented to the corresponding joints.



## Note

If the stretch shader has been created during the setup of the spine indicating a stretching of the back, it will be assigned to all parts of the back geometry.

## 5. Preparing for Rigging

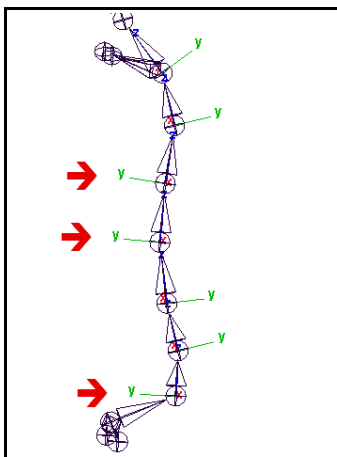
### Joints

Before putting the *Character Rigger* into action a complete skeleton must be set up. The hierarchy has to contain a certain number of joints for most parts of the body:

- Head: 3 Joints
- Arms: 4 Joints
- Legs: 5 Joints
- Fingers: 4-5 Joints
- Toes: 3-4 Joints

The back can contain an undefined number of joints. Only the amount of joints defines the degree of bending.

### Joint Orientation



A very important factor in preparing the character for rigging is the joint orientation. Depending on the creation and positioning of the joints the orientation can not be aligned correctly. It is also likely to happen that the orientation for neighboring joints is opposite to each other resulting in unaligned rotations.

To check the joint orientation it is helpful to make the *Local Rotation Axis* visible. For a correct joint orientation it is advisable to use the script *icJointOrient.mel* as a helper to orient the *Local Rotation Axis* of a single joint or a joint selection.

For a successful, logical and continuous rig regarding the rotation of joints it is advised to use the following joint orientation:

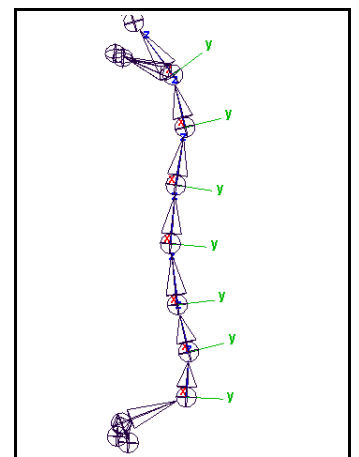
- Long axis of the joints: Z
- Up-axis of the joints: Y

Through this setup it can be granted that an up/down movement of the joints is always around the X-axis, a sideways rotation is always around the Y-axis and the rotation around the length is always around the Z-axis.

The automatically created controls of the rig also follow this logic of keeping orientation continuity.

### Joint Orientation of fingers and toes

Having a geometric model with already bend fingers or toes can make it hard to establish correct joint orientation through automated processes. Often the correction has to be done manually with making the *Local Rotation Axis* visible and adjusting it in *Component Type-mode* through the *Selection Mask*.



### Important

The joints of legs and arms have to be layed out in a two-dimensional plane. Especially the joints of the knees and elbows are only allowed to be bend in one direction. To make sure that this is correct it is best to create the leg joints in the side view and the arms in the top view before rotating them around the hip joint/shoulder joint into the final position.

## 6. Tools

### 6.1 Pose Mirroring

The shelf contains two icons for creating mirrored poses during animation. The first tool mirrors the entire character set of the selected control to the other side, i.e. if the hand control is selected the entire arm will be mirrored. The other tool just mirrors the selected control, i.e. only the elbow control.

**Reverse Pose** completely reverses the current pose in both directions, i.e. to help creating walkcycles. In addition to the regular *Pose Mirroring* this function works on both sides of the character as well as all center controls of the body and the head.

#### Note

*Pose Mirroring* also switches the IK/FK-State of the affected controls to correctly mirror the pose but only if the character is not currently in a transition between IK and FK. Automatic IK/FK switching of the *Pose Mirroring* can only happen if the controls or joints are in either full IK or FK mode.

#### Important

It is recommended to not undo the Pose Mirroring, especially when mirroring an arm or a hand. At times it can happen that this will cause a disconnection of the hand controlled by the arm movement which cannot be fixed later.

## 6.2 CharRig Store

With this tool it is possible to store and load entire poses and animations. The pose and animation data gets stored in a text file in the projects folder to which the custom folder *CharLib* can be added for organization purposes. The file extensions of the saved files represent the containing data; *.crp*-files contain poses, *.cra*-files contain animations.

The tool respects any attributes of the character, IK and FK switching and any animation-necessary information. This means that not only keyframes will be saved but also relevant information about tangent angle and -weighting.

If one or more controls are selected upon tool invocation all textfields regarding the character name will be automatically filled with the necessary information.

### Long Name

Is just used to name the node under which the entire character is grouped.

### Short Name

This field can contain up to four characters to uniquely identify the nodes during storing and loading of poses and animations.

### Namespace

Shows the namespace of the selected character.

### Get Character

Searches for locators that contain the user attribute *character\_icCRconnection*. If the name has been modified (see *Put Character*), the character definition cannot be found.

All characters found in the current scene will be listed and can be loaded individually.

### Store/Read Type

Defines if only the *Pose* is stored from the current position or the entire *Animation* of the character.

### Character Set

Defines if the current process of storing/loading a pose or animation works on the entire character with all character sets and their subcharacter sets (*All*) or only the character set of the selected control (*Current*).

### Current Auto-Prefix

This option allows to use the name of the current character set as a prefix for the created file if a pose or animation is stored on the current character set only. The feature is implemented to help naming and identifying stored poses and animations. It works only on *Character Set: Current*.

### Frame Range

Defines whether the entire frame range of the scene will be evaluated during the store or only the current portion of the time range set by the *Playback Range*.

### Read Main Control

For reading back in a pose or animation this option toggles whether the stored information will be applied to the *Main Control* or not. This allows for applying a pose or animation at a different position in space.

### Insert Time

When loading back in a pose or animation this toggle inserts it at either the current time position (*Current*) or the original time (*Original*) stored in the file.

The screenshot shows a software dialog box with a light gray background and a dark border. It is titled 'Character' at the top left. Below the title are three text input fields: 'Long Name', 'Short Name (4 max)', and 'Namespace'. A 'Get Character' button is located below these fields. The second section is titled 'Store Library'. It contains a 'Library Path' label, a 'Store/Read Type' section with radio buttons for 'Pose' (selected) and 'Animation', a 'Character Set' section with radio buttons for 'All' (selected) and 'Current', a 'Current Auto-Prefix' checkbox (checked), a 'Frame Range' section with radio buttons for 'All' (selected) and 'Current', a 'Read Main Control' checkbox (checked), an 'Insert Time' section with radio buttons for 'Current' (selected) and 'Original', and a 'Pose/Clip Name' text input field. At the bottom, there is an 'Insert Time Base' section with radio buttons for '0' (selected) and '1', and two buttons labeled 'Store' and 'Read'.

## Pose/Clip Name

The name of the pose or animation. If the options *Current* and *Current Auto-Prefix* are active, the name given here will be added after the name of the character set of the current selection.

The button on the right side of the name field (not shown in the illustration) pastes the selected item from the list into the name field. It allows for easier overwriting of existing files or for renaming purposes of file versions.

## Insert Time Base

On reading back in animations keyframes will be set at their original position. *Insert Time Base* defines whether the the start frame the original time range was based on is frame 0 or frame 1.

## 6.3 Switch Character Set

For easy switching between the different character sets two icons are placed in the shelf during setup.

### Switch to Character Set

Selects the character set **Main\_characterName**, independently from the currently selected control. This primary character set contains all subcharacter sets for the entire rig.

Using this function with one of the finger controls selected, the character set **Secondary\_characterName** will be activated.

### Switch to Subcharacter Set

This function sets the subcharacter set of the currently selected control as active.

If the function *Switch Character Set* is executed without a control selected the current character set will be cleared.

## 6.4 Toggle Set Visibility/Display Status

For easy switching between of the visibility or the display status (none/reference) two icons are placed in the shelf during setup.

### Toggle Set Visibility

Toggles the visibility of the selected control and the other set members of the same set or the selected set. In case the desired control is not visible the corresponding set has to be selected.

### Toggle Set Display Status

Toggles the display status between none and reference of the selected control and the other set members of the same set or the selected set. In case the desired control is not visible the corresponding set has to be selected.

## 7. Quick Start

### 7.1 Rigging the Demo Scene Character

1. Open the demo scene file **character.mb**.

The scene file already contains a prepared character, consisting of a polygonal geometry for the entire body (objekt *hiRes*), the individual parts of the geometry for linking to the joints for animation (group *loRes*), the complete joint hierarchy, the necessary locators and the locator *characterDefinition\_JohnDoe*. All polygonal objects are hidden.

2. Initialize the *Character Rigger* with the shelf icon (more about the installation can be found in section **2. Installation**).
3. **Character / Get Character** opens the *Select Character-Window*. Here the character definition "John Doe" must be loaded. All fields of the *Character Rigger* will be loaded with the appropriate joints of the character in the scene.
4. **Complete Rigging / Build Character** starts the automated rigging process for the entire character.
5. With **Attach All** in section **Assign LowRes Geometry Names** all parts of the character geometry can be linked to the respective joints.  
The character is now ready for animation.
6. The shelf button **Pick Walk Toggle** switches between the default pickwalking and the *Character Rigger Pickwalking*.

#### Note

It is advised to keep a copy of the current hotkey settings before building the rig or using the toggle hotkey function as the current settings will be overwritten in the preferences files.

## 7.2 Rigging of a Custom Character

1. Create a complete joint hierarchy for the character and place the joints at the correct position for animation (more information about the number of allowed joints can be found in section **5. Preparing for Rigging**).
2. Adjust the joint orientation with the **Joint Orientation Tool** from the shelf (more information about the joint orientation can be found in section **5. Preparing for Rigging**).
3. Create two locators for each heel of the character, acting as a pivot point for these parts of the feet.
4. Initialize the *Character Rigger* with the shelf icon (more about the installation can be found in section **2. Installation**).
5. A name consisting of up to four characters must be entered in **Character / Short Name**. A full name (**Long Name**) is optional and can be left empty.
6. In section **Define Character** the individual joints must be entered into the respective fields. The joint is selected first and can then be added to the field with **Select**.  
**Create ... Rig** allows for testing this part of the rig. An interactive adjustment of the parameters in the *Character Rigger* window with direct feedback on the rig is not available.  
The temporarily created rig can be set back with the *undo*-function; only the created rig of the spine cannot be undone. In this case the scene file must be saved prior creation.  
The size of the controls are mainly dependent on the characters dimensions and can be left at their defaults in most cases.
7. **Complete Rigging / Build Character** starts the automated rigging process for the entire character.
8. In section **Assign LowRes Geometry Names** the individual parts of the character geometry can be named. Select the appropriate geometry. While selecting the corresponding joint in the list in this section of the *Character Rigger* the renaming will be instantly executed.
9. **Attach All** links all renamed parts of the geometry to the respective joints.  
The character is now ready for animation.
10. The shelf button **Pick Walk Toggle** switches between the default pickwalking and the *Character Rigger Pickwalking*.

### Note

It is advised to keep a copy of the current hotkey settings before building the rig or using the toggle hotkey function as the current settings will be overwritten in the preferences files.

## 7.3 Skinning with an Established Rig

1. **Complete Rigging / Sel. Skin Joints** automatically selects all joints of the character relevant for skinning.
2. Add-select the characters skin with the *Shift*-key.
3. Apply a regular skinning with *Skin / Bind Skin / Smooth Bind* with the option *Selected Joints* selected.

## 7.4 Adding Individual Joints and Controls

(more information about this procedure can be found in section **4.4 Add Custom Character Parts**)

1. The nodes to be added (joints and controls) have to be cleaned up in such a way, so that only attributes that are relevant for animation are visible in the channel box. All other attributes have to be locked and hidden.
2. The nodes have to be added to the existing hierarchy of the character via regular parenting.
3. In section **Add Custom Character Parts** the **Main** oder **Secondary** character set has to be defined as the target character set.
4. The name of the new or existing set can be entered in the fields **Character Set Name Only** and **Set Name Only**. An empty field ignores the creation of a set.
5. **Include Sel. Joints** or **Include Sel. Controls** adds the selection to the existing structure of the character rig.