QFit Guide

# Introduction

This is a guide to the “QFit” scripts which fit a skeleton to a markerset. These scripts are tied a very specific combination of skeleton and markerset. Each script handles one such combination. As of the initial version of this document there are two such scripts: one for fitting a standard QTM animation skeleton to an animation markerset and one for fitting an Unreal Metahuman skeleton to a standard animation markerset.

These are Maya python scripts, one main script file for each application. There are also helper script files, notably “qscipy.py” which provides a rotation class similar to one in the scipy module for python. Scipy is a very large, memory heavy module that can overwhelm memory for 32bit applications. It is to avoid this problem that qscipy was created.

The approach for each script is to have a list of joint names for the skeleton. A “rule” (python function) is associated with each segment. The job of each rule is to figure out the correct orientation of the joint based on whatever other joints and markers it wants to use to determine the result. Each rule can assume that its parent joint has already been positioned so it can use that information for its own calculations. Helper functions for calculating a full transform based on just two vectors – a “forward” vector and an “up” vector for the joint - are provided so that each rule can simply concentrate on constructing two vectors. In practice, the forward and up vectors are locally aligned with the cardinal axes of each joint. This is different for each skeleton based on its construction. In the animation skeleton the rules construct the local Z and Y axes for Up and Forward (respectively). In the Metahuman skeleton the rules construct the local X and Y axes.

# Usage

It's handy to have the invocation of the script contained in an icon on the Maya shelf. These are the commands used for testing the two scripts “Anim Fit” and “MH Fit”:

Graphical user interface

Description automatically generated

The “Command” code for the “Anim Fit” icon is:

import importlib

import qtm\_connect\_maya.QFitAnimationSkeleton

importlib.reload(qtm\_connect\_maya.QFitAnimationSkeleton)

qtm\_connect\_maya.QFitAnimationSkeleton.FitSkeleton()

The “Command” code for the “MH Fit” icon is:

import importlib

import qtm\_connect\_maya.QFitMetahumanSkeleton

importlib.reload(qtm\_connect\_maya.QFitMetahumanSkeleton)

qtm\_connect\_maya.QFitMetahumanSkeleton.FitSkeleton()

The use of the importlib is handy in the case you are editing the scripts. “reload” is the only way to update the script to reload it with your changes.

Simply highlight the above command text and drag it onto the Maya shelf to create the icon. Then edit the “Icon Label” field under the “Shelves” tab to set the name of the icon.

# Maya Configuration Requirements

These scripts assume that Maya has its units set to centimeters and that the world up axis is set to Z (to match QTM).

# Using QFit Metahuman Skeleton

This section has two parts, the first is a quick start guide to demonstrate the process of setting up a metahuman from scratch; the second is a collection of notes on controlling how the fit happens.

## QFit Metahuman Quick Start

This guide will show how to set up a Metahuman Skeleton from the very beginning. Here is a fresh metahuman character called “Vincent”:

A picture containing sky, outdoor, ground, person

Description automatically generated

This is imported into Maya (see the Unreal guides for using Bridge to send a metahuman to Maya). This example is done using Maya2023 and Unreal 5.1.

A person wearing a garment

Description automatically generated with low confidence

Turn on the joints display. Delete the “root\_drv” skeleton to which the main character skeleton is constrained. This also deletes all the constraints. Also delete the head skeleton and the facial controls, these aren’t needed. You should have something like this when done:

A picture containing green, board, carrying, plastic

Description automatically generated

Now we’ll add all the custom solver attributes to the skeleton by loading an existing solver XML file. A sample one is provided in the “example” folder of the “qtm\_connect\_maya” folder. First select the root joint of skeleton:

A person wearing a garment

Description automatically generated with medium confidence

Now click on the “Import Skeleton Definition” icon A picture containing text, device, meter, gauge

Description automatically generated. Select the MH.xml example file. This will apply the skeleton definition found in the MH.xml file and change the namespace of the metahuman skeleton to match, making it ready for export to a new xml file for solving in QTM.

A picture containing graphical user interface

Description automatically generated

Try selecting various joints and looking at the Extra Attributes section to verify that the attributes have been added. Here is what the “thigh\_l” joint should look like:

Graphical user interface

Description automatically generated

You will see that markers have been added to the scene under “MH:Markers”, delete this.

The next step is determined by the name of the markerset you’re using. For this example, we’ll use the Eli markerset from the QFL Skeleton Solver example dataset. Open QTM and load the “EliROM.qtm” file and make sure QTM is on the first frame of data.

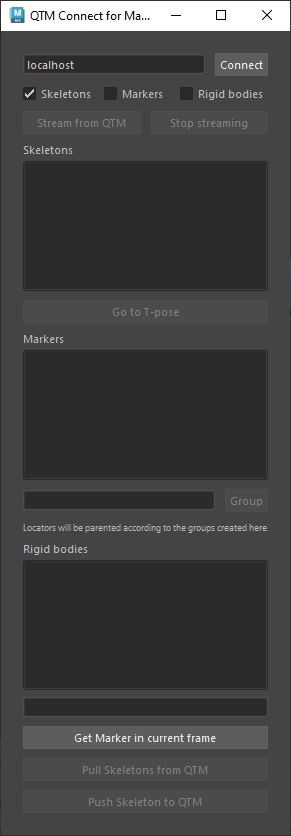
In Maya, open the Namespace Editor and change the “MH” namespace to “Eli”: A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

Now open the “Connect To QTM” icon, Graphical user interface, application, Teams

Description automatically generated, which brings up the “QTM Connect for Maya” dialog:



Select the “Get Marker in current frame” button. This imports the current frame of marker data from QTM. In this case, since QTM is on frame one of the file, the T-Pose for Eli:

A person in a garment

Description automatically generated with low confidence

Now select the root of the skeleton and click on the MH Fit icon to invoke the QFitMetahumanSkeleton script. You should see the skeleton pop into place, fitted to the marker cloud.

A picture containing green, sky, ground

Description automatically generated

Now select the pelvis segment and export the solver definition. Import this definition into QTM. You should see this in QTM:

A picture containing text, outdoor, colorful, outdoor object

Description automatically generated

Now “Play with realtime output” in QTM and go back to the Unreal scene with the metahuman. Set up the Livelink connection and open up the animation blueprint for the character.

Graphical user interface, application

Description automatically generated

Then in the scene hit play and you should see your character moving with the data in QTM:

Radar chart

Description automatically generated A picture containing sky, outdoor, beach

Description automatically generated

## QFit Metahuman Usage Notes

There are some additional controls built in to the QFitMetahumanSkeleton script to allow the user to modify/adjust how the skeleton is fit within the marker cloud.

After the first application of the script, certain joints have extra “QFit” attributes added to them to act as flags or values for the rules function of that joint. Joint Orient values for all segments are set to zero, this is current requirement of the solver tools which don’t allow for non-zero joint orient values, so this is enforced in the script. Also, all joints have the “Segment Scale Compensate” flags turned off to allow for setting a global scale in the pelvis joint. A scale value is not currently automatically calculated as it is best to allow the user to set this manually without having an automatic override. See the pelvis example below for how to adjust the global scale value. The “Preferred Angle” values are also set to zero. The solver exporter uses these values to set the “Default Transform” information in the XML file. QTM will transmit this over the RT connection. Sometimes you will want these set to, for example, a valid T-Pose for use with HIK in Maya or Motionbuilder. However, the Qualisys LiveLink plugin for Unreal requires that these values be set to zero.

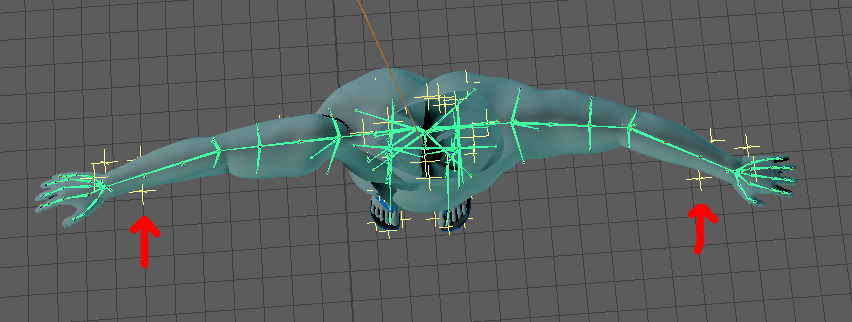
As an example of a QFit attribute, the pelvis joint has “Forward Offset” and “Vertical Offset” values that adjust the translational location of the pelvis (note: the pelvis is the only joint for which both translation and orientation are set, all other joints affect rotation only). The location (and orientation) of the pelvis is determined by the 4 waist markers. The two rear markers set the base location, the front markers are used to set the forward direction of the hips. The default values for Forward and Vertical are 10 and 5, respectively (these scripts assume the units are set to centimeters). Increasing the vertical offset will push the skeleton closer to the floor. To use the value, change it then invoke the script again to have it applied.

Here is a vertical offset of 12, applied to the root:

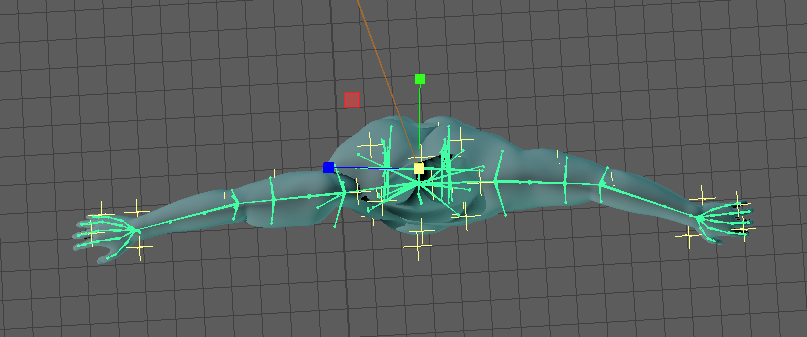
Graphical user interface

Description automatically generated with medium confidence

It is useful to use the vertical offset in conjunction with modifying the global scale of the pelvis to fit the scale of the skeleton to match the markers. A recommended way of setting the scale is to look at the locations of the hand joints relative to the wrist markers from above:

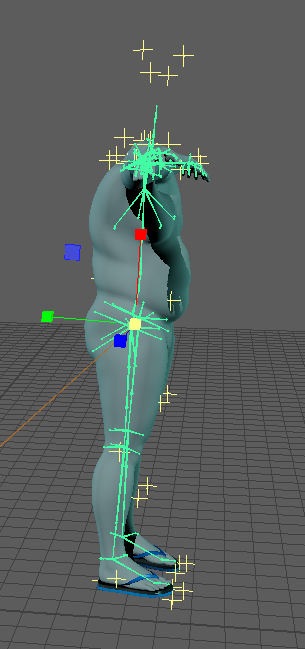


In the above image you can see that the wrist markers are not in alignment with the hand joints. This implies that the skeleton scale is too large for the arm span of the subject. Set the scale factor for all three axes of the pelvis to 0.9 and reapply the script to get this:

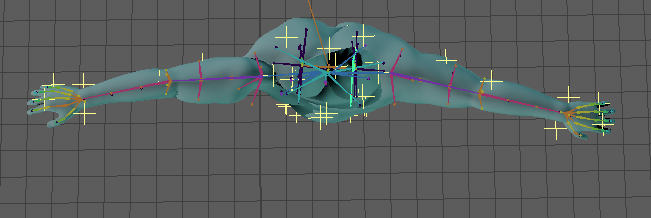


Now you can see that the hand joints are much more closely aligned with the wrist markers. This will be even more important to get right when using finger markers in the claw markerset (which this script supports) because getting fingers to solve correctly required a very close match of the finger joints to the finger markers.

In addition, if you view the whole skeleton from the side you will see that our new vertical offset is just right for the new scale:



Other QFit flags are added to the upperarm, hand, thigh and foot joints. For the upperarms and thighs there is an option for keeping the limb straight. For example, here is the upper left arm with the flag turned on:



The arm is straight, it is aligned with the mid of the wrist markers (just like the lower arm so they end up in the same alignment). With the flag off, as in the right arm, the location of the joint of the lower arm is determined by the ElbowOut marker.

Likewise, for the thigh joints, the direction of the segment is normally determined by the location of the knee markers. With the flag turned on, the upper and lower legs are aligned to “point” at a location midway between the ankle and heel markers but on the line from the heel to toe.

The hands have a “Palm Down” option which forces the palms to be perfectly flat with the floor, otherwise they’re aligned with the hand markers. Keeping the arms straight with the palms down is useful for use in conjunction with HIK in Maya or Motionbuilder. But for Unreal its best to have a fit that most closely follows the markers.

The feet have a “Keep Flat” option which assumes that the feet start in a flat global orientation. This orientation is saved at the beginning of the script so it can be re-instated after the rest of the body has been positioned. When not on, the foot is oriented to point at a location between the two foot markers.