

Diva Tutorial

(Last edited 5/15/02)

TABLE OF CONTENTS

SECTION 1: OVERVIEW	2
DIVA DIRECTORY STRUCTURE.....	2
LOGGING IN	2
USER PREFERENCES	3
INTERFACE.....	3
NAVIGATION.....	3
HELP/TUTORIAL.....	4
SHORTCUTS.....	4
MARKER FILE.....	4
IMPORTING .C3D FILES	4
SELECTIONSETS.....	5
SECTION 2: CLEANING DATA	6
COMMANDS	6
SCRIPT EDITOR	6
DELETING REDUNDANT MARKERS.....	6
REMOVE ALL TRAJECTORIES OTHER THAN CORRECTLY LABELED MARKERS	6
REMOVE TAILS	7
REMOVE SPIKES.....	7
FILLING GAPS	8
SCRIPTS.....	8
BATCH PROCESSING.....	9
GAPS IN THE BEGINNING AND END OF YOUR CAPTURE SEQUENCE	9
FILTER	9
SECTION 3: CREATING SKELETONS	10
HUMAN SKELETON.....	10
NON-HUMAN SKELETON.....	10
SETTING DEGREES OF FREEDOM (DOF).....	12
ROTATING SKELETON OR MARKER DATA.....	13
CENTERING MARKER DATA ON THE SKELETON.....	13
ADJUSTING BONES TO MATCH MARKER DATA	14
MAKE BONES SYMMETRIC.....	15
CREATING CONSTRAINTS.....	15
CREATING SOLVERS.....	16
CONSTRAINT WEIGHTS.....	17
RUN THE SOLVER	18
SEED SCRIPT CREATION	18

This document is meant to be a BRIEF explanation of Diva and has been culled from experience and Diva's Online Help manual. For further information on any particular topic, see Diva's online help.

Section 1: Overview

Diva directory structure

Diva is located on the C drive of each of the machines under

C:\Program Files\House of Moves\Diva

In this folder, you will find the Diva executable, other default preference files and the following directories:

Docs – contains tutorial file DivaHelp.chm

Sample Files – contains the files used in the tutorial

Scripts – House of Moves created scripts that you can use

Logging in

You first logon to a machine with Diva using your Accad account, create a Diva folder in your H drive and copy the following files to that directory.

(Found in C:\Program Files\House of Moves\Diva

Default.ini (rename this file to something else. E.g Suba.ini)

HotKey.hot

HotKeys_pro.hot

MarkingMenus.mkm

MarkingMenus_Pro.mkm

SelectionSets.ssf

SelectionSets_Pro.ssf

CommandLog.log

On double-clicking on the Diva shortcut on the desktop (or the Diva.exe file in C:\Program Files\House of Moves\Diva), a window pops up asking for the User Profile. Click on the Change button and search for your .ini file in your H drive. Click the ok buttons to select the file and to open the Diva application.

User Preferences

You will now need to change the User Preferences under <Edit>/<Preferences>. On opening the Preferences window, under the Directories tab, make sure the Scripts directories points to a directory in your H drive where you will be saving your scripts and to the House of Moves Scripts directory. Change the Log file path and name, Marking menu file, Hot Key file, and Selection Set file to point to the corresponding file in your H drive. If you make any changes to any of these files, before logging out, go to <Edit>/<Preferences> and press the SAVE AS button for that file. By default, it should save all your preferences before closing, but I've noticed it doesn't always work.

Interface

The Diva interface is similar in look and feel as the Maya interface. If all the dialog boxes are closed, the Diva interface would consist of two components: the menu bar and the viewing windows. The top of the Diva interface contains the Menu bar consisting of File, Edit, View and Help. The main part of the Diva interface contains the viewing windows, which can contain a Perspective view, Orthographic views (Top, Front, Right), Graph window, and a Hypergraph window. These can be set by pressing Alt + RMB in any of the viewing windows and selecting Views.

Various Dialog boxes can be opened and placed in your Diva interface by selecting them under the View menu option. The common Dialog boxes are the Main and Manipulator toolbar, the Button Shelf, and the Playback Controls and Command Line. You will be using the others as you proceed through the Diva Tutorial.

House of Moves has defined some default View window formats. In the Button shelf, select the Views tab. Eight views from View1 to View8 have been defined for our use.

Navigation

LMB - Rotate

RMB – Zoom

LMB + RMB - Translate (Note: if tracking has been set you will not be able to translate the view. You can fix this by pressing Alt + RMB in the viewing window and selecting “No Tracking”).

Alt + LMB - Select a marker by clicking on the marker or select an area by drawing a box in any of the viewing windows.

Ctrl + Alt + LMB – Make multiple selections

Ctrl + LMB - If a marker is selected and the position manipulator button has been pressed, you can change a marker position by pressing Ctrl + LMB and adjusting the X, Y, and Z manipulator handles.

Help/Tutorial

Go to <Help>/<Online Help> and select the Contents tab to see the Diva overview and Diva tutorials sections. Start with the Diva overview to get familiar with the interface. Missions 4 – 9 are the chapter you need to pay attention to in the Diva tutorials section.

Shortcuts

To get you started, if you are using the HumanRTKm.mkr marker set in Workstation, I have created some scripts and other files that will work with that marker set. Download findBadData.hsl, GenericCleaningScripts.hsl, GenericFillRigid.hsl, GenericRemoveUnlabelledTrajectories.hsl, removeTails.hsl, SelectionSetsGeneric.ssl, and HumanRTKm.mkr from my website at <http://www.accad.ohio-state.edu/~varadara/mocapclass.html> and click on the “My Diva Files” link.

Marker file

The .mkr file in Workstation uses a shortcut to draw the lines between markers. It automatically draws a line between markers of every rigid body. Diva expects you to explicitly define the lines between markers.

E.g.

RFHD, LFHD
LFHD, LBHD
etc.

The HumanRTKm.mkr file off my website has this added information defined after the rigid body definition in the [Autolabel] section.

Importing .c3d files

Before working on your .c3d files, I strongly suggest you make a copy of the .c3d files in a different directory and work on the copy. It's very easy to mess up your .c3d files in Diva with scripts.

To open a .c3d file select <File>/<Import> and choose your .c3d file. By default, when the file is loaded all the markers are selected, but if you have de-selected any markers by mistake, you can select them all by typing *select -all* in Diva's command line. With all the markers selected, select <File>/<Import> and choose the .mkr file.

SelectionSets

SelectionSets are a hierarchical way of grouping markers together based on rigid body definitions. A selection set can consist of markers, markers + other selection sets, other selection sets.

Ideally, the sets should be defined such that a single change in one of the lower sets should permeate to the upper level sets in the hierarchy. I strongly suggest you have a different selection set file for every marker set file.

Section 2: Cleaning Data

(Missions 4, 5, and 6 of the Diva Online Tutorial)

Commands

Commands can be entered in the command line dialog box and executed by pressing the enter key. The up arrow will move through previous commands. You can also previous commands by pressing the arrow next to the command line. Commands can also be executed in the script editor either as individual commands or as part of a script.

Script Editor

Select <View>/<Script Editor> to open the script editor dialog box. In the dialog box, open the GenericCleaningScript.hsl to get an idea of what a script looks like. To run a script, click on the Run Script button. To run a specific command in a script, highlight the command using the LMB and press the Execute selection button. (Moving the mouse over the buttons will pop up the button labels.)

Deleting redundant markers

To remove non-unique data, type and execute the following commands either in the script editor or in the command line.

```
select -all;  
deleteRedundant;
```

select -all; selects all the markers in the file.
deleteRedundant; is a Diva command that removes any non-unique marker data

Remove all trajectories other than correctly labeled markers

```
select;  
selectByType CharacterNode; // select the top level node  
selectSet -a Body; // selects all of the body markers  
select -invert; delete;
```

select; clears the selection i.e. deselects all selected items
selectByType CharacterNode; select the top level node of the marker set. You can see the node selected in the Hypergraph window.
selectSet -a Body; selects all the markers in the selection set called Body in addition to any markers already selected i.e. the CharacterNode in the previous step.

select -invert; delete; Inverts the selection. Selects everything that was not selected as a result of the previous step and de-selects everything that was selected in the previous step. Deletes those markers.

Remove Tails

The keys just before and after a gap are usually dirty (incorrect) data. This data is referred to as tails. To remove such data:

```
selectProperty Translation;  
select -all;  
findTail 2;  
cutKeys;
```

selectProperty Translation; selects the Translation channels. You can see this in the Channel editor.

select -all; selects all the markers

findTail 2; selects keys in the tails of all the markers. 2 is a parameter that can be changed to get varying results. As the number increases, the command becomes more selective.

cutKeys; delete the keys on all three channels of the selected frames. Even if the previous command selected keys on only one channel say translation x in frame 34 of marker LFHD, *cutKeys* will delete the translation x, y, and z channels of LFHD in frame 34. *cutKeys* cannot be applied selectively to a particular channel, but will work on all three channels.

Remove Spikes

A spike is where for a few frames there's a sudden jump in the trajectory before returning to a smooth trajectory.

```
selectProperty Translation;  
findBadData -select 2 2 -all 2 0.15 0;  
filter -selected 0.2 45;
```

```
selectProperty Translation;  
findBadData -select 2 2 -all 2 0.15 0;  
filter -selected 0.2 45;
```

findBadData -select 2 2 -all 2 0.15 0; runs a filter on all the markers on all the translation channels, compares the original data to the filtered data and based on a threshold value decides which keys are spikes and selects them.

filter -selected 0.2 45; the filter command runs on the selected keys only (does not run on all three channels like *cutKeys* does). There are two parameters for the filter command the cut-off value and threshold value. Based on these two values, the filter can range from

a very mild filter to a very strong filter. In this case, the filter is a strong filter. Produces almost linear data.

Filling Gaps

Gaps can be filled either linearly, rigidly, or using copyPattern. I'll be covering the two more common ones linear fill and rigid fill here. For more information on copyPattern, see the tutorial.

Linear fill

```
selectProperty Translation;  
select -all;  
fillGaps -maxGapWidth 20 -all;
```

fillGaps -maxGapWidth 20 -all; fills all the gaps in all the selected markers with a gap size less than 20 frames linearly.

Rigid fill

```
selectProperty Translation;  
selectSet head;  
findGap -select 0 0;  
fillGaps -rigid -all;
```

selectSet head; selects the markers in the selection set head.

findGap -select 0 0; finds all gaps in the selected markers and selects those gaps. Gaps show up highlighted in pink in the graph window.

fillGaps -rigid -all; fills all selected gaps in the selection set rigidly i.e. based on the other markers in that set.

You would repeat the last three commands for all the first-level selection sets in the marker file. (i.e. You would run this on left_upperarm and left_lowerarm, but not on LeftArm.)

Scripts

You can start by using my GenericCleaningScript to create your own scripts. The GenericCleaningScript was written with the HumanRTKm marker file in mind. You will need to modify the script for other marker sets. The main difference will be in the RigidFill part of the script. Note that a script can call other scripts. All commands and scripts will be highlighted in blue. If on trying to execute a script it doesn't work and it is not highlighted in blue, either the filename is wrong or in your <Edit>/<Preferences> you have not included the directory containing the scripts.

Batch Processing

Once you've written and tested a script on a .c3d file, you can run the same script on all the files with the same marker set as a batch process. Select <File>/<Batch Process>. A dialog window opens. You will need to import all the .c3d files you wish to run the script on, select the script, output directory, output type and filename, and export frame rate. Press the convert files button when everything has been selected.

Gaps in the beginning and end of your capture sequence

Once you've run your scripts on a .c3d file, you'll need to fill in any gaps at the beginning and end of the motion sequence. You can do this by creating a virtual marker in the first or last frame of the sequence respectively. Move the playbar to frame 1, select the marker you wish to create data for. Press the position manipulator button in the Manipulator toolbar. You should now see the X, Y, and Z axes for that marker. You can manipulate the position by using Ctrl + LMB. Adjust the marker position to what appears visually correct. Run the fill rigid commands again for the selection set that the marker is a part of.

Filter

The final step is to play the motion sequence and look for any abnormalities in the data. Any jumping markers, noisy markers, etc. You can then go in by hand and select the marker and sequence of frame and either cutKeys and fill in gaps or filter the selected data. At this stage it's probably easiest to select the data for a marker in the graph window.

Section 3: Creating Skeletons

(Missions 8 and 9 of the Diva Online Tutorial)

Human skeleton

You will need the Attributes dialog <View>/<Attributes>, Channels dialog <View>/<Channels>, and Skeleton editor <View>/<Skeleton Editor>.

Create a human skeleton by selecting Press the Create New Skeleton button. Name the skeleton Skeleton_1. Determine what options you wish to set for your skeleton like the number of spine and neck segments, etc. Click the Ok button.

Non-human skeleton

There are two ways to start creating a non-human skeleton. The first is to create a Human skeleton, remove any bones you don't want and then insert the rest of the bones. If you do follow this method, name the skeleton as in the previous step Skeleton_1. It will keep everything consistent at the end of the tutorial when we create the Skeleton seed script to automate creating skeletons. When removing bones, you can remove from anywhere in the hierarchy as long as you have the box Re-parent children selected.

BUG REPORT: Diva 1.7 has a problem with removing bones too fast. Pause about a second between pressing the Remove button otherwise Diva will crash. Problem is to be fixed in Diva 2.0.

The second method is to start from scratch. I'll be describing the second method since you have more control over creating the bones hierarchy.

In the command line, type

```
create CharacterNode Skeleton_1;  
create BoneNode root;  
select root Skeleton_1;  
parent;
```

create CharacterNode Skeleton_1; every skeleton you create should have a character node as the topmost node.

create BoneNode root; the skeleton itself will consist of bone nodes. We're creating the root of the skeleton

select root Skeleton_1; select the root and Skeleton_1 nodes in that order

parent; parent the root node to the Skeleton_1 node

If you want to change how the hierarchy looks in the hypergraph (i.e. placement of the nodes in the graph), select a node using the Alt +LMB, and move the node in the hypergraph using the Ctrl+LMB.

Now, select the root node. Open the skeleton editor and click the setRoot button. You should see the root node in the skeleton editor.

We are now going to add the other bones in the hierarchy.

BUG REPORT: There are some problems with defining the local axis and offset length of the bones in the skeleton editor (Diva 1.7). To be fixed in Diva 2.0.

To get around the bug, create the skeleton hierarchy in the skeleton editor, and change the length of the bones and the local axes in the Attributes dialog box for each bone.

Example:

In the command line, type

```
cameraView -zUp on;
```

The above command is only to make sure everyone's seeing the same thing. Once you've understood how to create skeletons, you don't have to keep the z axis up.

To create three bones off the root, select the root node in the skeleton editor and press the insert after button. The default name of the bone is NewBone_#, where # is a number that increases sequentially with each bone you create. You can change the name of the bone in the Name textbox found at the top of the Attributes dialog box. Change the name of this bone to lowerBack. Insert a second bone under the lowerBack and name it upperBack. Under the root, add two more two-bone chains, lhipjoint → lfemur, and rhipjoint → rfemur. When you are done, the hierarchy in the skeleton editor should look as follows:

```
Root → lowerBack → upperBack
      → lhipjoint → lfemur
      → rhipjoint → rfemur
```

In the perspective view, you will see just a dot representing all the bones you just created. This is because we have not added any offset lengths, or local axes values to the bones.

Now, I want the three bones off the root representing the lowerBack, lhipjoint and rhipjoint. First, we'll change the lengths of those three bones. To change the length of a bone, you adjust the Tx value of it's child. So, to change the length of the lowerBack bone, we change the Tx value of upperBack to say 100. Make sure you hit the enter key after changing the Tx value in the Channel dialog box. Similarly, change the length of the lhipjoint bone and rhipjoint bone to 100, by changing the Tx values for the lfemur and rfemur respectively.

You should see in the perspective view what looks like one bone of length 100. This is because all three bones are laying one on top of the other. We will now change their

rotations. You change the rotation of a bone by changing its local axis values in the Attributes dialog box. DO NOT adjust the Rotate X, Y, and Z in the Channels editor. Once a default skeleton structure is created, you will adjust the Rotate values in the Channels editor to fit the skeleton to your marker data. We're going to keep the lowerBack as is and rotate the other two bones. Select the lhipjoint. Change the Local Axis Z value to -135. You should see the skeleton lying in the XY plane. Similarly, change the rhipjoint Local Axis Z value to 135. You now should see a three bone pinwheel lying in the XY plane. Remember that the Z axis is up, which is why rotating about the z-axis results in the skeleton lying in the XY plane.

To finish this example, we'll create the left leg of a human skeleton. If you closed your skeleton editor, open the editor again, select the root node in the hypergraph, and press the Set Root button in the Skeleton editor. You should again see the bone hierarchy in the skeleton editor. In the skeleton editor, select the lfemur bone. Insert a four-bone chain under the lfemur and call them ltibia, lfoot, ltoes, and ltoes_end.

Change the lengths of the following bones. Remember you should be changing the Tx value of the bone's child.

```
lfemur 200
ltibia 180
lfoot 80
ltoes 20
```

Now adjust the Local Axes of the bones as follows:

```
lfemur Z 45
ltibia do nothing
lfoot Y -75
ltoes Y -15
```

You will note that the bone is rotated with respect to its parent's position. Since the ltibia continues in the same line as the lfemur we don't adjust the local axes of the ltibia. It remains 0 0 0.

Setting degrees of freedom (DOF)

Open the Attribute dialog by selecting <View>/<Attributes>. The following degrees of freedom can be used for a human skeleton. In general, the root node should have all six degrees of freedom and the other joints will just have rotational degrees of freedom depending on how the bones rotate. The more degrees defined the harder it will be to solve the skeleton positions for the entire motion sequence.

- Root: All translation and rotation DOF on
 - The root is the only bone that will have translation as well as rotation DOF.

- lhipjoint and rhipjoint: All DOF off
 - These bones are generally dummies; therefore, turning off all DOF will prevent any rotations from being calculated.
- lfemur and rfemur: All rotation DOF on
- ltibia and rtibia: Only Y Rotation on
- lfoot and rfoot: All rotation DOF on
- ltoes and rtoes: Only Y Rotation on
- upperBack, lowerBack, thorax, head and neck bones: All Rotation DOF on
 - Also, for the head and neck bones only, you should also set the Spine_Bone attribute ON. The other three bones should NOT have the Spine_Bone attribute on.
- lcollar and rcollar: Ry and Rz on.
- lhumeral and rhumerus: All rotation DOF on
- lwrist and rwrist: Rx and Rz on
- lhand and rhand: Ry and Rz on

Rotating skeleton or marker data

If your skeleton is lying down and the marker data is standing upright, under the Main tab of the Buttons Dialog, click the Z-up button. This will make the skeleton stand up (Z is now up) and the marker data will be lying down.

- Select the topmost node of your marker data hierarchy. Any commands now executed will work on all the markers in the hierarchy.
- Type `setProperty Rotation -c -90 180 0;` in the command line.
- This will rotate the marker data so that it's standing upright. If you find that your marker data is facing forward and the skeleton is facing backwards or vice versa, type `setProperty Rotation -c -90 0 0;` in the command line.

Centering marker data on the skeleton

- `select LFWT LBWT RFWT RBWT;`

- Click the Position Manipulator then click the 3D Manipulator Follow to follow the four main waist markers. The difference between 3D Manipulator Follow and 3D Manipulator Follow Selection Changes buttons, is that the latter will move the manipulator as you change your selection. In this case, we want the entire marker setup to move based on the Position Manipulator centered at the waist.
- Again, select the topmost node of your marker data.
- Click to depress the Offset Edit tool button in the Manipulator toolbar and translate your marker data until the skeleton/root is slightly below and centered between the waist markers. The Offset Edit button does not create keys automatically on the 6 channels (Tx, Ty, Tz, Rx, Ry, Rz) unlike the Key Insertion Tool. If you don't know which button is the Offset Edit tool button, move your mouse over the buttons in the toolbar and a description of the button will pop up.
- After completing this step, switch back to the Key Insertion Tool.

Adjusting bones to match marker data

We will be adjusting the bones length as well as rotation values to match the marker data. We want the lengths to be fixed but the rotations should be removable if we want to get the skeleton back to its base pose again. To do this, we create keys on the rotation channels before adjusting the rotations of the bones.

```
selectProperty Rotation;
selectByType BoneNode;
```

Verify you are at the beginning of the file (frame 1), and create a key for all of the selected bones by typing the following into the Command Line and hitting ENTER:

```
createKey;
```

This will set a key only on the rotation values of all the bones and will allow us to animate the rotations of the bones, placing them accurately within the marker field, while changing the constant translation value of each bone.

Hot Tip: You can tell if you have accidentally inserted any keys in the translation channels because the color will change from white to yellow in the Channel dialog box.

To see the bone axes while you're adjusting bone rotations and lengths, type

```
selectByType BoneNode;
setProperty Show_Axis on;
```

Click the 3D Manipulator Follow Selection Changes and switch to the Rotation Manipulator

Select the Offset Edit Tool

Select the lfemur bone node

To adjust a bone's length, you need to change the Translation X value of its child. To change a bone's rotation, you change the Rotation X, Y, and Z values of the bone itself. For example, to change the bone length of the lhipjoint, you change the Translation X value of the lfemur bone. To rotate the lhipjoint, you need to select the lhipjoint and using the Ctrl + LMB on the rotation manipulator adjust the rotation of the bone.

In general, select a bone, adjust the translation if necessary (i.e. changing the length of the previous bone), and then adjust the rotation. The bone should be positioned as if it were the real bone inside the body. So, for example, the lfemur bone would be slightly interior of the knee marker with the bottom part of the bone in line with the side knee marker.

Make bones symmetric

If you want the bones to be symmetric between the left and right legs or left and right arms for example,

Change the Tx value of right femur to match left femur OR

Select the lfemur then the rfemur, paste the following into the Command Line and hit the enter Key or into the Script Editor and press Run Script.

```
paste_boneOffsets_toPrimary_autoHierarchy;
```

This is assuming that you have already set the bone lengths for all the bones in the left leg.

Once you've adjusted all the bones in the skeleton, make sure first that there are no keys in the Translation channels of any of the bones. You can see if there are any, by selecting each bone and seeing if the Translation channels in the channel dialog box are yellow.

Creating Constraints

From this point on, we will be recording the commands that we execute so that the script created can be used to create skeletons for other motion capture takes in the same folder.

Open the script editor.

Press the record button

Open the constraints dialog.

Set the following constraints.

Root: LFWT, LMWT, LBWT, RFWT, RMWT, RBWT

i.e. select the Root bone on the left hand side of the window, then select the above markers on the right hand side of the window. You should see commands being recorded in the script editor.

Femur: LKNE/RKNE, LKNI/RKNI, LHIPJOINT/RHIPJOINT

Tibia: LSHN/RSHN, LANK/RANK

Foot: LHEL/RHEL, LMT5/RMT5, LMT1/RMT1

Toe: LTOE/RTOE

Thorax: C7, T10, CLAV, STRN

Head: LFHD, LBHD, RFHD, RBHD

Collar: LSHO/RSHO

Humerus: LUPA/RUPA, LELB/RELB, LIEL/RIEL

Wrist: LWRI/RWRI, LWRE/RWRE

Hand: LFIN/RFIN

In general, you define constraints with markers at one end of the bone, and not on both sides. For example, you would not use both the LKNE and LANK markers for the tibia bone.

Turn off the record button

Save the script, by pressing the Save or Save as button in the Script editor.

Creating Solvers

Open the Script Editor, clear the window by selecting the New button and select Record actions.

Open the <View>/<Solvers> dialog box

Click the New button

Type in your solver name, in this case left_arm_solver

Click OK

In the Solver dialog box, select the Bones button. This changes the left window to show the Bone chain. In the Bone Chain window, scroll down until you see the left arm bones (lcollar, lhumerus, lwrist, lhand) and click on the circles next to these bone names.

To the right is the Constraints window (if it is labeled Solvers, click the Constraints button above the window, so it reads Constraints)

You should now see the constraints associated with the bones that are members of that solver.

The solvers you will need to create for the HumanRTKm.mkr set are:

left_arm_solver: lcollar, lhumerus, lwrist, lhand

right_arm_solver: rcollar, rhumerus, rwrist, rhand

left_leg_solver: lfemur, ltibia, lfoot, ltoes

right_leg_solver: rfemur, rtibia, rfoot, rtoes

waist_solver: root

chest_solver: lowerBack, upperBack, thorax

head_solver: neck, head

Turn off the record button in the Script editor and save the script.

Constraint weights

As a general rule, you will want to weight constraints heavier for parts of the skeleton that are usually more important. For instance, shoulders, wrists and ankles are good candidates for a greater weight.

- Open the Solvers dialog.
- Select left_leg_solver from the pull-down menu
- Under Select click the Constraints button
- In the Channels dialog, select the left knee constraint (LKNE_DirCnstr)
- Set its weight value at 0.75 by entering it into the Weight field

Enter the weights for the other constraints in the left_arm_solver in a similar manner. Then select the other solvers one by one, click the Constraints buttons under Select and set their weights.

LSHO, RSHO constraints weight: 5

RUPA, LUPA constraints weight: 0.25

LKNE, RKNE, LKNI, RKNI, LELB, RELB, LIEL, RIEL constraints weight: 0.75

LSHN, RSHN constraints weight: 0.25

LANK, RANK, LWRI, RWRI constraints weight: 3

LHEL, RHEL constraints weight: 0.5

LMT1, RMT1, LMT5, RMT5, LWRE, RWRE constraints weight: 2

LTOE, RTOE, LFIN, RFIN constraints weight: 0.5

Run the solver

You will now run the solvers over the motion sequence. In the solvers dialog box, in the right hand side window if you see constraints listed, press the Constraints button just above that window, it will toggle to Solvers. Make sure all the solvers you created have been selected in the right hand side window. Press the All Frames button. This will run the solvers over all the frames.

On playing the motion sequence, you should see the skeleton moving with the marker data. If there are any problems, you can adjust the constraint weights and run the solvers again over all the frames. Note: Before adjusting weights, make sure you didn't make any mistakes in either defining the constraints between the bones and the markers, creating the solvers, or setting the constraint weights especially if you're using the HumanRTKm.mkr set or something very similar.

Seed Script creation

We have already created scripts for a number of the steps involved including creating constraints and solvers. We are going to combine all the scripts to create a seed script that can be run on other c3d files in the same session to create skeletons.

First, we still haven't created a script for creating the skeleton.

To create the skeleton-creation script, follow these steps:

- Select the root bone
- Open <View>/<Skeleton> Editor
- In the Skeleton Editor dialog, click the Set Root button

- Click the Create Script button
- In the pop-up, confirm that the script will be sent to the Script Editor. You could also create an individual file
- Click the OK button to create your script

In line 13 of this script, you will notice that the character node created for the skeleton is called `Character_#`.

- Using the Script Editor pop-up menu (RMB on the script editor window), replace all instances of this in the script with `Skeleton_1` or whatever name you used for the `CharacterNode` when you created your skeleton.
- Save this script as `Skeleton Creation` (as backup)

Now additional bone settings, such as bone geometry and twist, will be queried from the scene and pasted into the seed script. The script that will retrieve this information is called `getBoneNode_Properties`. It will retrieve the required information and read it into the Command Log.

In the command line, execute the following

```
getBoneNode_Properties;
```

From the Command Log, copy the information read in from the script and paste it into the seed script, immediately below the skeleton setup portion

***Hot tip:** This script sets a standard for bone properties and checks all bones against this standard. Only the properties that differ from the standard are read into the Command Log.*

Paste the contents of the Constraints script you created earlier to the end of the seed script. Save the seed script (use a different name than `Skeleton Creation`).

Now that the constraints are created, their offsets and weights must be adjusted so when the file is solved the bone-to-marker relationships are accurate. To get these values follow these steps:

- To query the existing scene for the offsets and weights and read them into the Command Log, execute the following in the command line.
`getConstraint_Properties;`
- Copy the offsets from the Command Log and paste them at the end of your seed script
- Save the seed script.

Paste the contents of the Solvers script you created earlier to the end of the seed script. Save the seed script.

Now, only two things remain; solving the skeleton and adjusting the camera view:

```
solve;
```

Add the following line only if your mocap data is coming in lying down with respect to the skeleton created.

```
cameraView -zUp on;
```

At this point, it would also be helpful to include a header. The header should contain information such as project, performer, and character as well as date created, if applicable. Use `//` to enter this information as comments.

- Save your script

To test your seed script, import another c3d file. Run your seed script. You should see a solved skeleton.