UD1436 – Assignment: Scripting for Animation

Submission: zip-file uploaded to ItsLearning

Deadline: 2016-03-13

1 Overview

Character rigging and animation are key techniques in the entertainment industry. Motion capture (mocap) allows us to obtain real movements from, for example, human characters. These movements can then be transferred to existing 3D models to provide a more rapid or even real time result, dramatically reducing the cost of animation based on keyframes.

For this assignment, you learn how to transfer an animation from one skeleton to another which has different orientation axes in its joints. This task is particularly useful as new tools for Mocap are available every year but sometimes they do not follow the same conventions used in the Digital Content Creation (DCCs) tools like Maya. This assignment aims to deepen your knowledge of 3D animation and also using scripting for animation.

2 Description

The assignment is split into several independent tasks that can be handled based on what has been covered in the course and the previous scripting plugin course.

2.1 Animation Transfer

You will be provided with an FBX animation created with the software iPi Soft, which can be imported into the Maya scene as a skeleton with all the keyframes.

You will also have your own skeleton, with different proportions, naming convention and orientation axis in each joint. If you do not have a skeleton, you can simply create one from Maya Skeleton->HumanIK menu.

From now on, we will refer to the skeleton of the mocap as the **source** data and to your own skeleton as the **target**.

With both skeletons loaded in the scene (**source** and **target**), a script should be written to transform and transfer the keyframes of the **source** data to the **target** for each keyframe. This is the key part of the assignment and this part should be working before moving to the UI design and coding of the UI. The following subsections describe in detail the steps needed in the coding process.

2.1.1 Skeleton joints matching

The two skeletons can have different number of joints. Therefore, if your skeleton (**target**) is missing any joint that is in the **source** data, you have to edit and insert those joints by hand in the **target** before proceeding. This step would not be needed if you use the Maya skeleton creation tool. If your skeleton has more joints than the **source** data, then only those bones that are present in both skeletons should be matched and the rest should be ignored. If you use Maya's skeleton, this will be the case (more bones in the **target** than in the **source**)

2.1.2 Matching of Skeleton (for a better result)

Align the joints (with rotations and overall skeleton scaling) of your skeleton to match the initial T-pose of the mocap data as much as possible without altering the proportions of your skeleton. This task will be done by hand.

2.1.3 Animation Transformation

It is not uncommon to have access to rigged characters that are using completely different joint orientations. If this wasn't the case the animation transfer would be a trivial copy of the keyframes. To account for the difference in join orientations, the transforms need to be recalculated from the basis of the **source** to the basis of the **target**.

Write a function that for each existing key-frame in the scene, and **for each join that is in both skeletons** transforms and transfer the rotation from the **source** to the **target**. For the hip joint (root), the **position** has to be transferred from the **source** to the **target** as well, because the rest of the joints are relative to this position and will be in the right place automatically.

Detailed steps in the process:

- Populate lists with the names of the joints in the skeletons, one for the **source** and one with the counterparts from the **target**.
- For each keyframe, in each joint, isolate the difference in rotation between the current keyframe and the translation for the **bindpose** (from **source** skeleton). When isolated, transform the difference from the coordinate system of the **source** to the coordinate system of the **target** and transfer to **target** skeleton.

Isolating the difference in translation requires knowledge of all joint rotations starting from the root, this makes a recursive approach suitable. You can use the recursion example in the slides as a template for this task. A non-recursive solution is possible as well, but it is recommended to use the recursive approach. An explanation of the math required for the change of basis will be available on ItsLearning.

2.1 GUI Design

After finishing the two PyMEL functions from the previous section, design a Qt User Interface similar to Figure 1, and modify the template provided for the lab to implement the GUI behavior.

- When the root joint is put into the text field ”Root”, and the return key is pressed, the whole DAG of joints should be loaded into the list. This applies for both skeletons.

- The buttons ”up”, ”delete” and ”down” are used to re-order the joints in case they do not match, or additional joints are found in the target skeleton and should be deleted from the list. The mapping of which joint corresponds to which joint is determined implicitly by the order in which they are listed.

- Once the joints are correctly mapped, and for every joint in the mocap data there is a joint in the target, pressing the button ”Transfer animation” should transform and apply all the attributes to the target for every keyframe defined in the original animation.

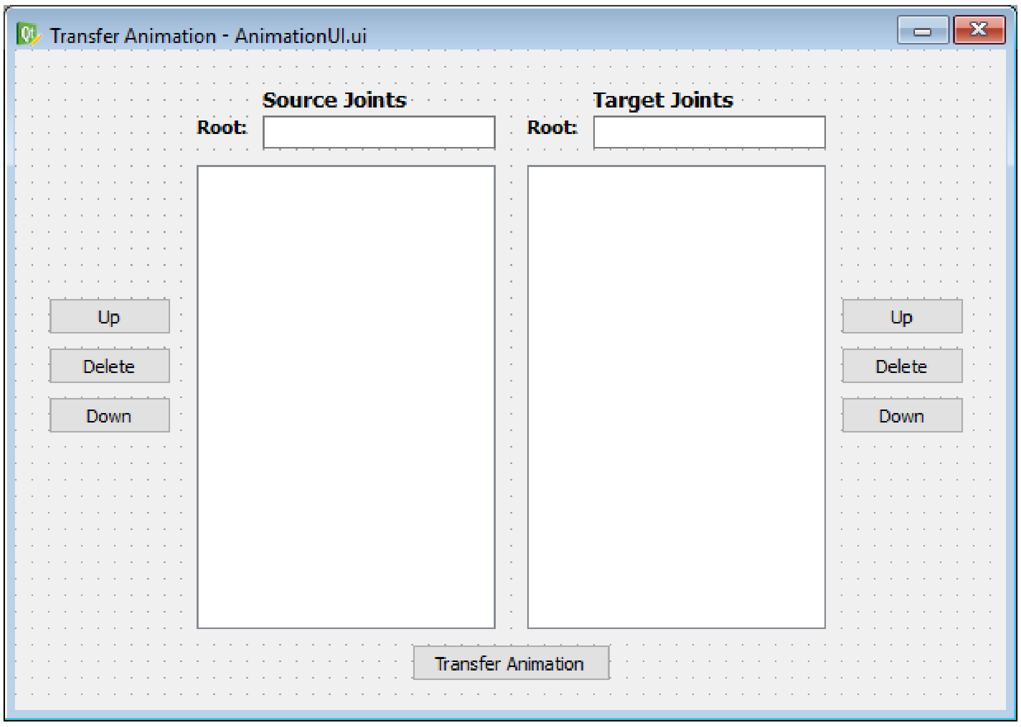

Figure 1