Lab 5 - Motion Capture - Animation
CMPUT 414

February 22, 2016

In this assignment you will need to create a tracking camera and animate it. This action will be automated through a script. For this assignment, you can follow the tutorial and complete it by using Blender. Submissions will also be accepted for Autodesk MotionBuilder.

1 Load a Motion Data file with Blender

For this part, you will need to download the motion data file located at `http://webdocs.cs.ualberta.ca/~salzvede/motion_sample/131_09_60fps.bvh`

After downloading the given MoCap file, you should import it into Blender. MoCap files can be imported through the following option:

File > Import > Motion Capture (.bvh)

After importing the skeleton, you need to scale it down. Use , approximately, the following scaling transformation: $S = (0.15, 0.15, 0.15)$.

It is also important to note that the MoCap file contains 1147 frames, recorded at 60 fps. For this reason, make sure that the parameters highlighted in figure 1 are correctly set. In case they are not, manually modify them.

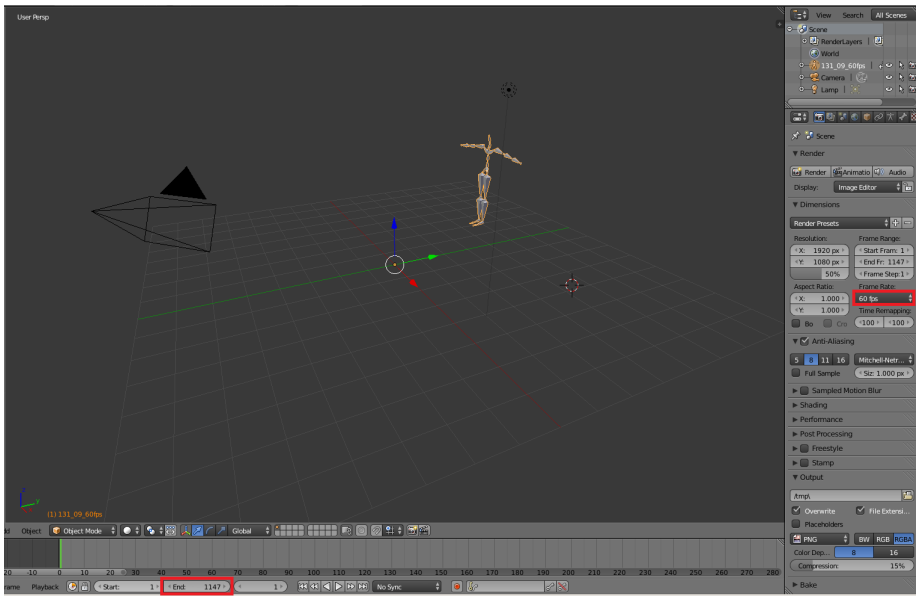

Figure 1: Blender initial setup

If the file has been correctly imported, you should now be able to play your motion sequence. Press **ALT + A** to try it.

2 Scripting in Blender with Python

This section describes most of the work to be done for this assignment. First, you will need to switch from the default screen layout to the scripting mode, as shown in figure 2.

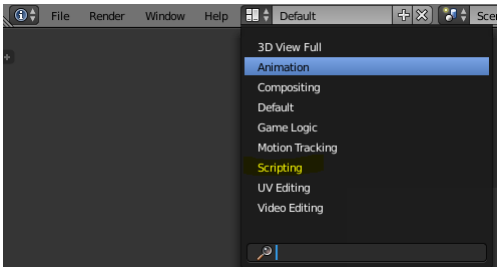

Figure 2: Scripting mode

The goal of your script is to create a camera that has the following properties:

- It tracks the skeleton's head during the whole sequence (skeleton has a joint called 'Head'. Track axis of the camera should be $-Z$, and its up axis should be Y. This way the head can be seen correctly;
- The distance between the camera and the head is always the same, more precisely, it is always 5 units;
- Camera has about the same Z coordinates as the head, at all frames (i.e. only X and Y coordinates are modified);
- The camera will orbit the head. It will give a **full rotation** around the head, as the animation plays. Camera starts rotating at frame 1, and only stops at frame 1147. This step may require some camera animation. Verify how to use keyframes, and remember that they will be automatically interpolated.

Your camera should be called 'rot_cam'. Every time your script is executed, it should delete any previous camera with the same name, in order to avoid duplicates. You might want to add this to a function

While there is not only a single way to implement this camera, the three first requirements can be relatively easily implemented with **constraints**, which link two objects. For instance, when tracking an object, the following constraint can be used

```
# ob is the object to have the constraint added to
# it has been previously created
    cns = ob.constraints.new('TRACK_TO')
    cns.name = name
    cns.target = target
    cns.track_axis = 'TRACK_NEGATIVE_Z'
    cns.up_axis = 'UP_Y'
    cns.owner_space = 'WORLD'
    cns.target_space = 'WORLD'
```

Other constraints can be found on the properties tab (illustrated in figure 3). It is important to note that while hovering over different buttons on the properties tab, Blender gives you tips on how a given action can also be used in its Python API. You may want to check if you can perform the same actions using Blender's graphical interface, before start coding it.

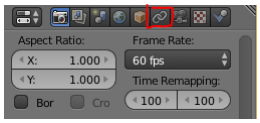

Figure 3: Constraints option

If you are in the correct track, you should visualize something similar to what is shown in figure 4.

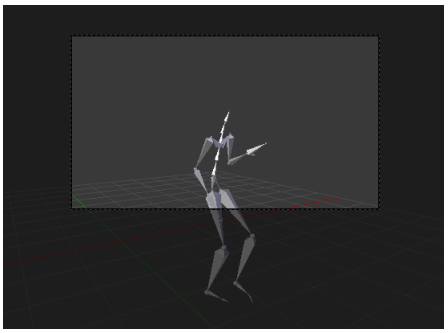

Figure 4: Sample view from 'rot_cam'

3 Recording your Results

Set the camera created on the previous section as the active view for the scene.

Under scene properties (same location where fps is modified), located **Output** settings. Set the output mode to **Ogg Theora BW** video.

After setting the output, go to

Render > OpenGL Render Animation

And verify if your animation has been correctly rendered. You should get a view of your camera. Make sure the whole sequence was rendered.

4 Deliverables

You should submit your python script, described in section 2, and the video recorded in section 3.

Observations

In case you are more comfortable with Autodesk MotionBuilder, you are allowed to use it and its Python API to complete this assignment.