HKU MoCap Studio Manuals v1.0

These manuals are meant to give an easy step-by-step guide for working with the Motion Capture system of the HKU. The manual can also be found online, together with all tools and example projects at the following page:
https://drive.google.com/folderview?id=0BzwxZounZTt9bjYxWmlnRGZYRE0#list

This document has manuals for working with the **Motive** control software, as well as using **Cinema4D**, **Blender** and the **Unity** game engine to use the captured motion data. The above folder also contains an example **Processing** project.

This document contains the following manuals

01 Preparing the Motion Capture system

This manual explains how to startup the motion capture system, calibrate the cameras and create a new session with rigid bodies and/or skeleton data.

02 Preparing Motion Capture Data

This manual explains how to record, export and/or stream the motion capture data so you can use it in your own project

03 Use Case: Rigging a 3D Model in Cinema4D

This manual shows how to use pre-recorded motion capture data in C4D to animate a 3D model.

04 Use Case: Visualising Trajectories in Cinema4D

In this manual we'll use recorded motion capture data to visualise movement through space.

05 Use Case: Visualising Real Time data in Unity3D

This manual explains how to use real time motion capture data in Unity.

06 Use Case: Getting Real Time data in Blender

This manual shows how use real time streaming data from Motive in the Blender Game engine.

01 Preparing the Motion Capture system

This chapters describes how to prepare the Motion Capture system so it provides accurate data for your project.

Check the Connections

Figure #1.1 shows a schematic setup of the MoCap system. 8 Infrared cameras (this may be more or less) are connected to a router, which connects to the main MoCap PC which runs the Motive control software. Motive can be used to record and export captured data, but clients can also connect to the Motive system via a network switch to receive data in real time.

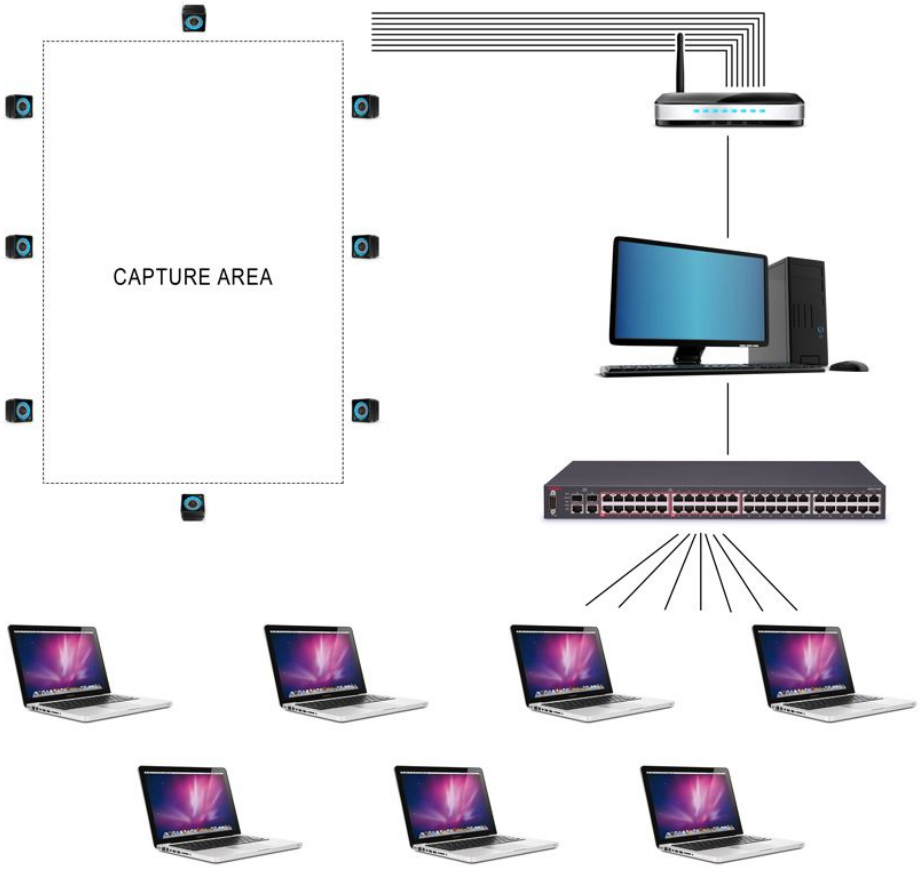

Image #1.1: System schema

Check if

- the router is turned ON (lights should be blinking on the router)
- all cameras are connected to the router (the rings on the cameras should light up)
- the MoCap PC is connected to the router

Calibrate the system

Step 1: Point the cameras

In the studio there are infrared cameras installed (image #1.2). Before you start, check if all the cameras point roughly towards the center of the area that you want to capture.

Image #1.2: IR camera

Image #1.3: The (Calibration) Wand

Image #1.4: Ground Plane Markers

Step 2: Clear the area

- Open the Motive software on the MoCap PC
- In Motive go to Layout → Calibrate (image #1.5)
- Make sure the Capture Area is cleared and empty and that there are as few as possible objects lying around. Especially objects with reflective surfaces like chairs can interfere with the MoCap system.
- If the camera previews at the bottom of the Motive screen show any white spots;
    - Try to figure out which objects in the room are causing this. If possible remove the objects or cover them with non-reflective material, like a jacket or tape.
    - If all physical objects are hidden and there are still white spots in the previews screens, hit the **Mask Visible** button in the **Calibration** panel (top-right). All white spots should turn red.

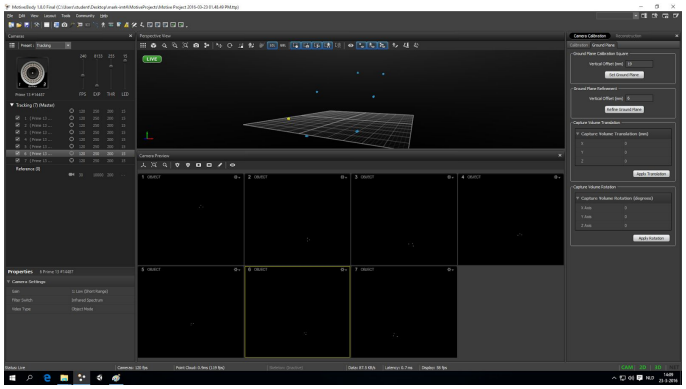

Image #1.5: Calibration view

Step 3: Wanding

Motive needs to calculate the positions and orientations of the cameras. To do this, it needs as much as possible calibration data to compare the cameras to each other. The process of creating calibration data is called *wanding*.

- Assemble the wand (image #1.3)
- Press the **Start Wanding** button in the **Calibration** panel
- Keep *wanding* until the calibration panel is green, and all camera previews are covered with as much color as possible (image #1.6)
    - Walk around the capture area waving the wand around
    - Try to cover the entire volumetric area; lows, highs, centers, corners, edges, etc.
- If you can't get any of the cameras to fill up, or you notice that the colors are showing up out of center, reset the calibration and start over from step 1;
    - you can hold the wand in the center of the Capture Area (at middle height) and use the white markers in the camera preview screens to check if the cameras are pointing in the right direction. The markers should show up in the middle of all the screens.
- If you have generated enough calibration data (image #1.6), click the **Calculate** button
- When the calculations are done and everything went well, motive should now show the cameras in the correct formation in the 3D view (image #1.7).
    - Don't worry if the cameras are not straight above the floor, we'll take care of that in the next step.

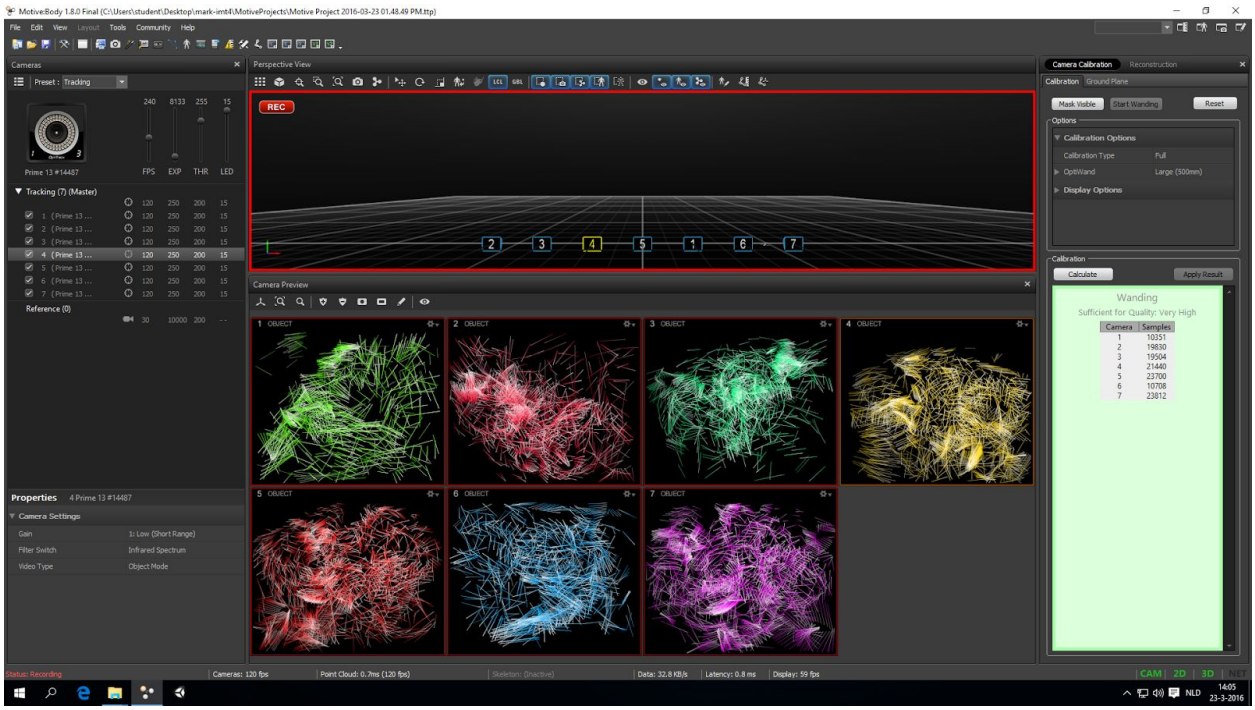

Image #1.6: calibration screen during the 'wanding' process

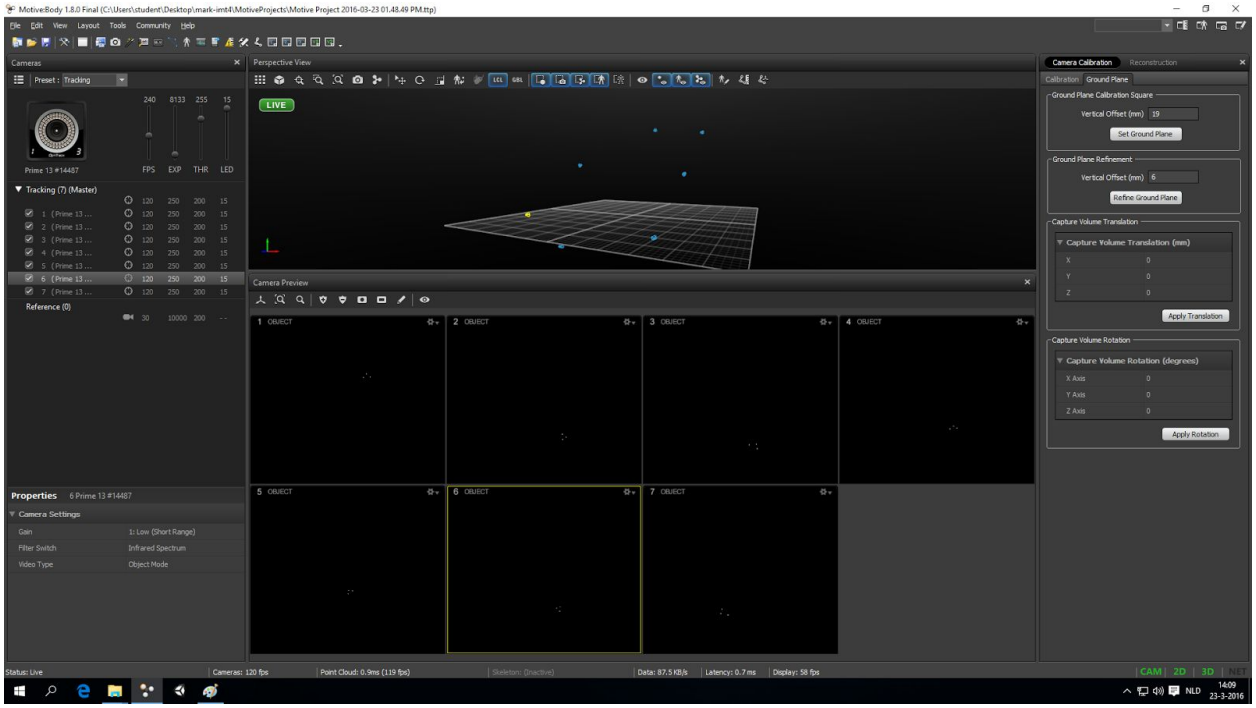

Image #1.7: wanding done, ground plane incorrect

Step 4: Set the ground plane

Right now, Motive knows how the cameras are positioned and oriented *relative to each other*, but not yet relative to the rest of the world. The 3D view probably shows the cameras in the right formation, but tilted in relation to the floor (image #1.7). To fix this we have to set the *ground plane*.

- Take the **Ground Plane Markers** (image #1.4) and place them on the floor in the middle of the capture area (check if the markers show up in the camera preview screens)
- Make sure the long side points forward (this will become the z-axis of the 3D scene) and the short side points sideways (the x-axis)
- Hit the **Set Ground Plane** button in the **Ground Plane Panel** (top-right, image #1.7)
- The cameras should now show up straight above the floor in the 3D view

Step 5: Test

The MoCap system is now calibrated and ready to be used. You can grab some markers (or use the wand) and move them around in the Capture Area. The markers should show up correctly in the 3D view in Motive. You might need to move/rotate/zoom the 3D view to see them.

For more detailed information on calibration, go to the optitrack Quick Start guide at: http://wiki.optitrack.com/index.php?title=Quick_Start_Guide:_Getting_Started#Camera_Calibration

Create Rigid Bodies

Motive can get the *position* of individual markers, but not the orientation (rotation). To calculate the orientation of an object, Motive needs at least 3 markers to be attached to the object in a fixed position.

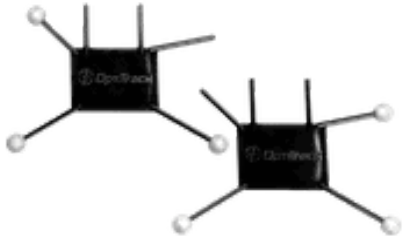

Image #1.8: marker-set (for hands)

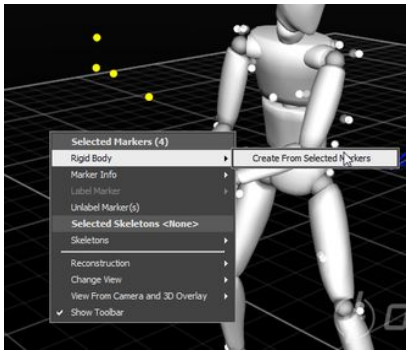

Image #1.9: Select markers

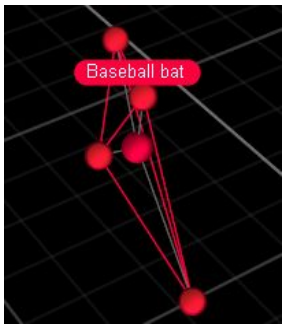

Image #1.10: RigidBody

- Fix at least 3 markers (see for example image #1.8) to the object you want to track (this can be any solid physical object; a box, a whiteboard wiper, your hand, etc.)
    - Avoid using objects with reflective surfaces
    - Make sure the markers are positioned in a non-symmetrical formation
- Place the object inside the Capture Area
- In Motive go to **Layout → Create**
- Find the right markers in the 3D view (you might need to move/rotate/zoom)
- Select the markers in the 3D view (image #1.9)
- In the 3D view go to **Right Mouse Button → Rigid Body → Create From Selected Markers**
- In the 3D view the Rigid Body should show up as colored connected markers with a name (image #1.10)
- You can give the rigid body an appropriate name in the properties panel on the left (image #1.11)
- You can create as many rigid bodies as you need

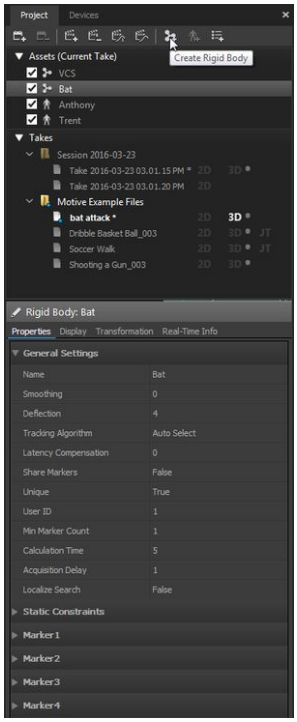

Image #1.11: Rigid Body Properties

Create a Skeleton

For tracking the movements of a human body you'll need create a skeleton in Motive and dress your actor up in a mocap suit.

Image #1.12: Mocap suit / T-pose

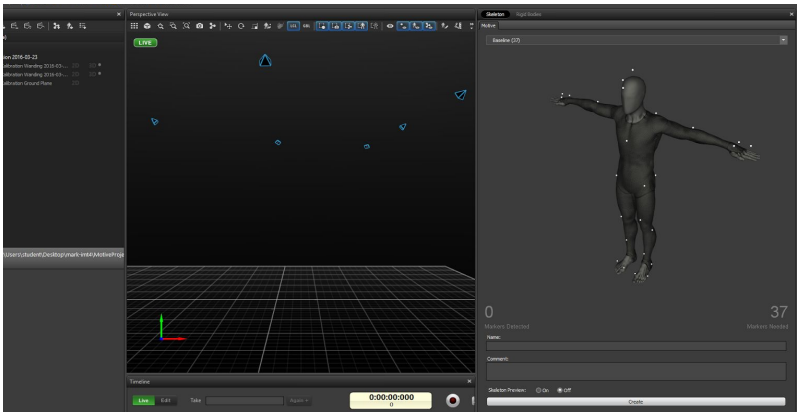

Image #1.13: Skeleton Template / T-pose

Step 1: Pick a skeleton template

Motive provides a couple of *skeleton templates* that work well for full-body tracking.

- In Motive go to **Layout → Create**
- Choose a skeleton template from the skeleton panel (on the right, image #1.12)
- The skeleton panel shows;
    - how many markers you need (bottom right)
    - where each marker should go (you can rotate and zoom the image)
    - How many markers the system can currently see (bottom left)

Step 2: Suit up!

- Put your actor in the velcro mocap suit and remove any markers that might still be attached to it (image #1.12)
- Attach the markers according to the Motive template. Work your way from top to bottom, or the other way around, to not miss any markers.
- If your actor is standing inside the Capture Area, motive shows how many markers it can currently find at the bottom left of the skeleton template panel (image #1.13).
    - Other objects or people might be blocking markers, so clear the area and let the actor move around to check your markers

Step 3: Create, Name, Test

- When all markers are placed on the suit and Motive can find all of them, press the **Create** button at the bottom of the skeleton panel (image #1.13).
- A body should show up in the 3D View (image #1.14)
- Give the created skeleton an appropriate name

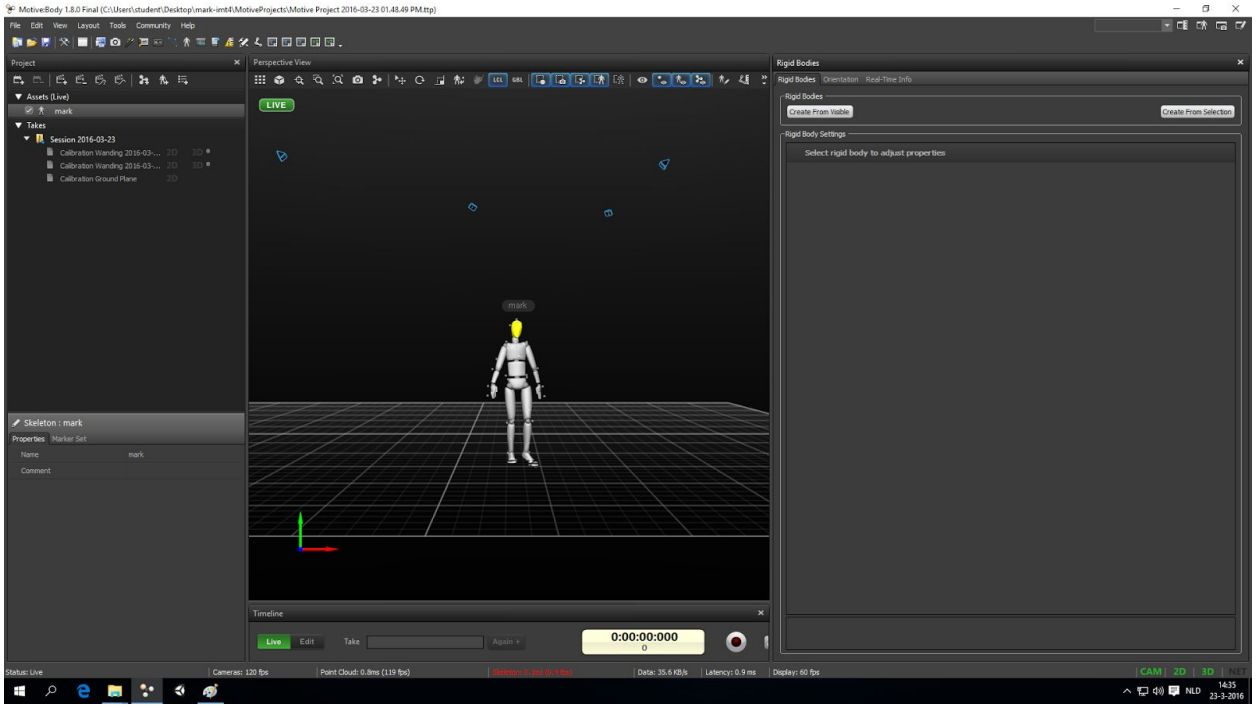

Image #1.14: Skeleton, created & named

The optitrack quick start guide has more in depth information about how to place markers (1) and how to create Rigid Bodies and Skeletons (2):

1. http://wiki.optitrack.com/index.php?title=Quick_Start_Guide:_Getting_Started#Marker_Up
2. http://wiki.optitrack.com/index.php?title=Quick_Start_Guide:_Getting_Started#Define_Skeletons_and_Rigid_Bodies

02 Preparing Motion Capture Data

This chapter assumes you have a working and calibrated motion capture system and you have created all the rigid bodies and/or skeletons that you need (see the *Preparing the Motion Capture system* manual for instructions).

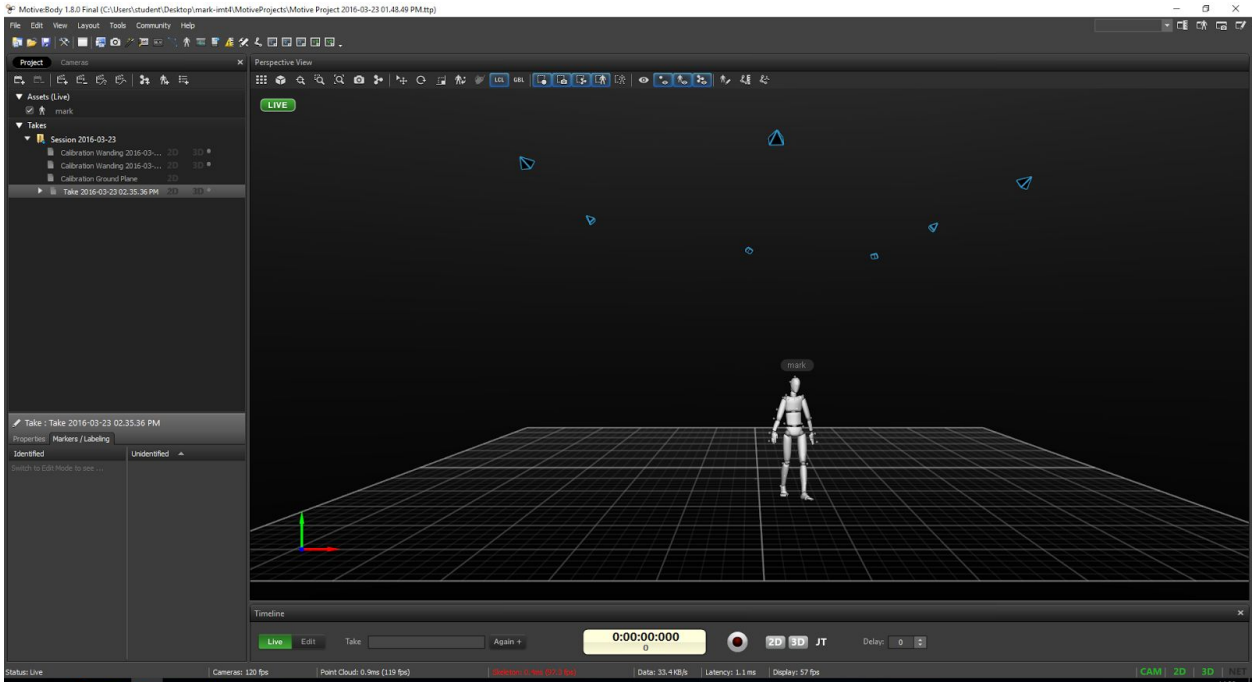

Image #2.1: Capture Layout

Record tracking data

- In Motive, got to **Layout** → **Capture** (image #2.1)
- Check if everything looks good and your skeleton and/or rigid bodies are showing up correctly
- Practice all movements and check if everything looks good in the 3D view
- Any time if a skeleton in the 3D view looks strange, your actor can do a T-pose (image #1.12 and #1.13) to help Motive find the position of the skeleton
- In some cases certain markers might not be visible to enough cameras for Motive to track the motion. Try doing the movements in a different position or change your camera setup
    - Note that if you change the position of the cameras -even a little bit- you have to recalibrate the system
- To start recording, press the record button at the bottom of the screen (image #2.1). The timer starts running
- To stop recording, press the record button again. A new *take* shows up on the left (image #2.1)

- Double-click on a take to play it back and check if everything was recorded properly.

Export tracking data

Motive can export the capture tracking data in various formats. Have a look at the Cinema4D use-case chapters of this manual to see which format you might want to use.

For working with skeletons and animating 3D models, you'll most likely want to use the BVH format. See the *Use Case: Rigging a 3D Model in Cinema4D* chapter for more information.

For working with point clouds and trajectories, you might want to use the FBX format. See the *Use Case: Visualising Trajectories in Cinema4D* chapter for more information.

- Go to File → Export Tracking Data
- Choose the right format
- Export

Stream tracking data

Motive can both stream the live situation and pre-recorded sessions.

To start streaming from Motive

- Open the streaming panel via **View → Data Streaming**
- Make sure **Broadcast Frame Data** is enabled
- Check if **Local Interface** is set to **loopback** (or otherwise a 192.168.1.* address)
- Make sure the following are enabled:
    - Stream Rigid bodies
    - Stream Skeletons
    - Skeleton As Rigid Bodies
- Under **Advanced Network Settings** check if
    - **Multicast Interface IP address** is set to **239.255.42.99**
    - **Port** is set to 1511

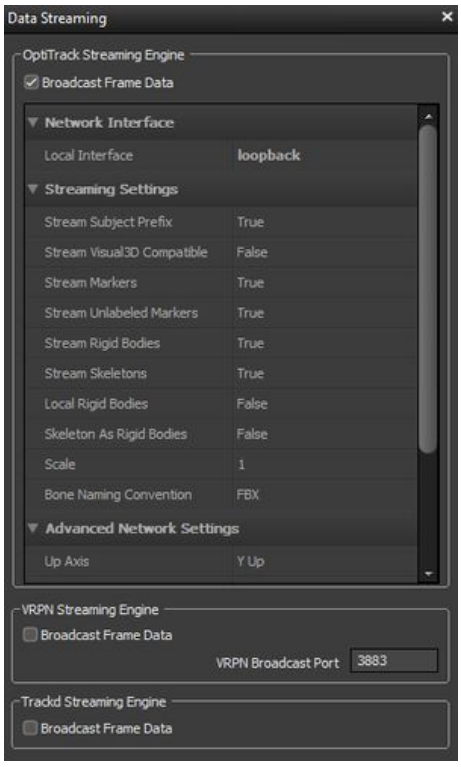

Image #2.2: Data Streaming Panel

03 Use Case: Rigging a 3D Model in Cinema4D

This manual assumes you have recorded MoCap data in BVH format. For example data and the a c4d project file with the results of this manual, download the file **MoCap-cinema4d.zip** from the following page:
https://drive.google.com/folderview?id=0BzwxZounZTt9bjYxWmlnRGZYRE0#list

Importing BVH

- In Cinema4D go to File → Open
- Find and open the BVH file you want to use
- There will be another popup window asking for the scale, you can leave the value at 1 cm at press Ok
- A Null object with joints should appear in the scene (image #3.1)

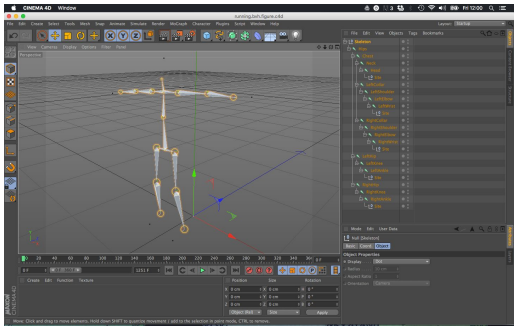
c4d-02.02-animation

image #3.1: imported skeleton

- This skeleton is animated using the MoCap data; scroll through the timeline to watch it move (image #3.2)

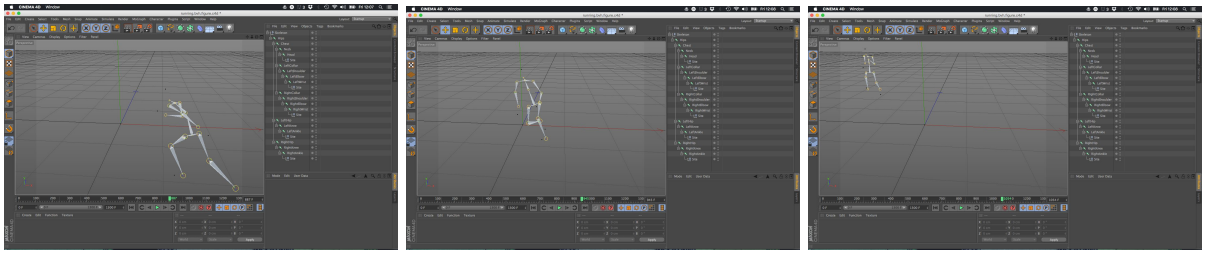

image #3.2: animated skeleton

- You might have to scale down the timeline to see the entire animation (image #3.2)

- These joints can be used to animate your 3D model, on their own they’re not visible in the render (press CMD+R and you'll see the picture will be all black).
- Next up we'll create a simple 3D model and rig it with the skeleton we've just imported

Rigging you 3D Model

- For this example we'll create a simple cinema4d 'Figure' model, but you're free to use any model of your own. If you want to add the imported skeleton to an existing C4D scene, you can simply copy-paste the Null object containing the joints into your existing project.
- Move the timeline back to the position where your skeleton is in the calibration position (image #3.3)

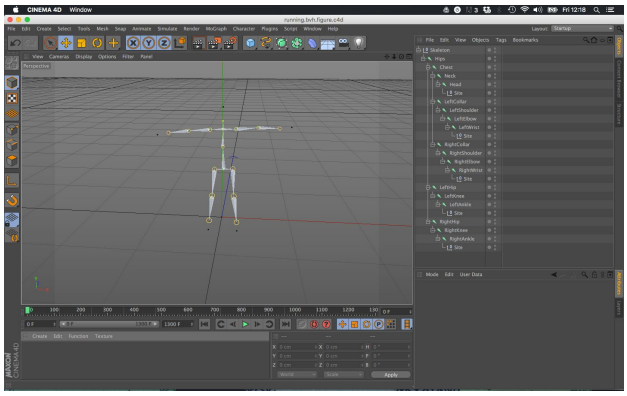

image #3.3: T-pose calibration position

- Create a 'Figure' model by going to Create → Object → Figure (or hold your left mouse button down on the cube icon and select the Figure button, image #3.4)

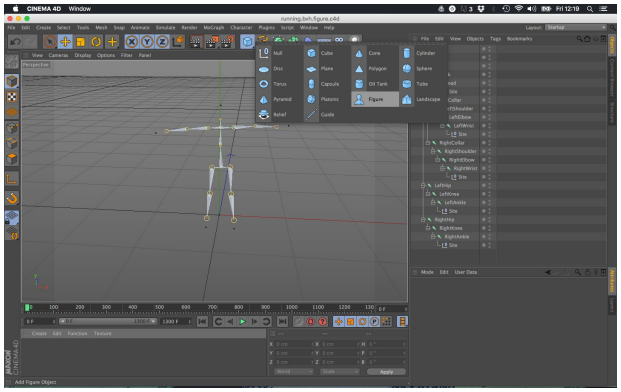

image #3.4: creating a figure model

- Scale and position the new figure mesh so it fits with the skeleton as good as possible (image #3.5)

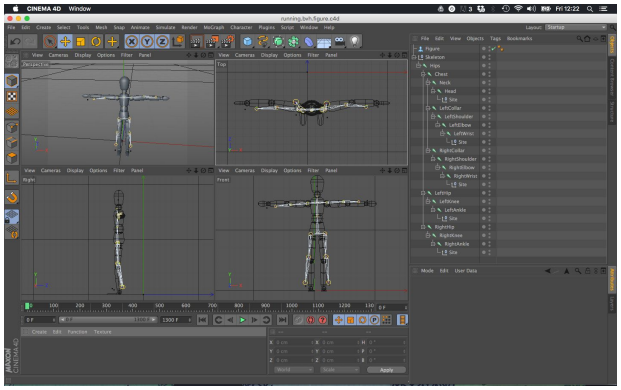

image #3.5: matching skeleton to model

- For this introduction we'll do it slightly quick-n-dirty. For proper rigging techniques checkout the youtube link below
- Before we can *bind* our model to the skeleton, we have to turn our *Figure* object into an editable *Polygon object*.
    - When you unfold the figure object in the object manager (select the figure and in the object manager click View → Folding → Unfold Selected) you'll see that the object is subdivided into several different parts, like Thighs, Shins, Upper Body, Feet, Hands, etc. (see image #3.6)

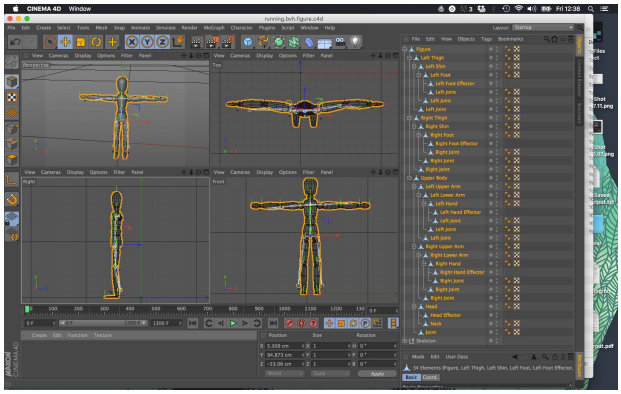

image #3.6: Model parts

    - We'll need to connect these parts together into a single mesh
        - Select all parts of the figure by clicking on the very first object with the right mouse button and clicking *Select Children*
        - Next merge them all together by going to Mesh → Conversion → Connect Objects + Delete
        - You should now have a single polygon object without any child-objects (image #3.7)

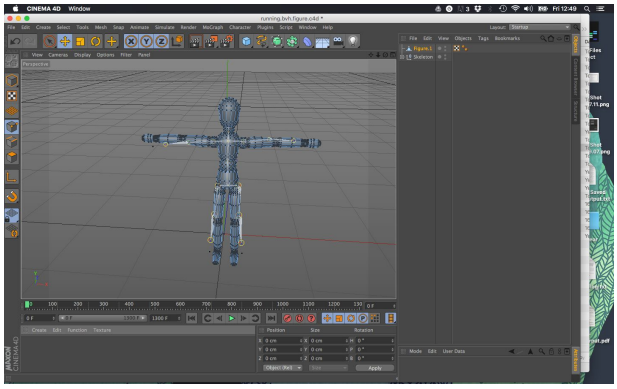

image #3.7: single mesh model

- Bind 3d model to your skeleton
  - Select all of the skeleton by selecting the top 'skeleton' object in the object manager and going to Edit → Select Children
  - Also select the 3d model by holding the CMD key and clicking on the figure object in the object manager
  - Make sure all of the 3d model's points (vertices) are selected by going to Tools → Modes → Points and in the 3D view pressing CMD-A. Your model should light up with orange dots (figure #3.8)

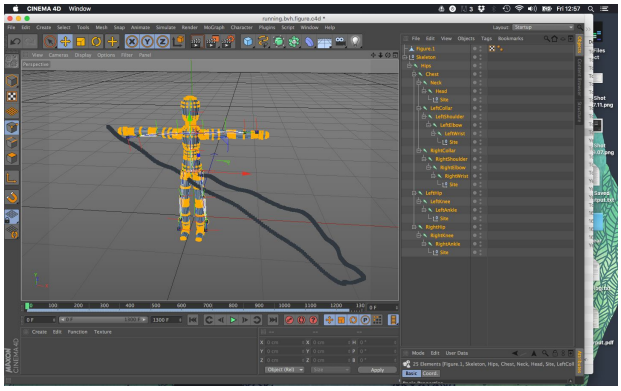

image #3.8: select all points of the 3d model

  - With all of your model and skeleton selected, now click Character → Commands → Bind
  - Nothing much seems to change, but your figure object should now have a *Weight Expression* tag and *Skin* child object (figure #3.9)

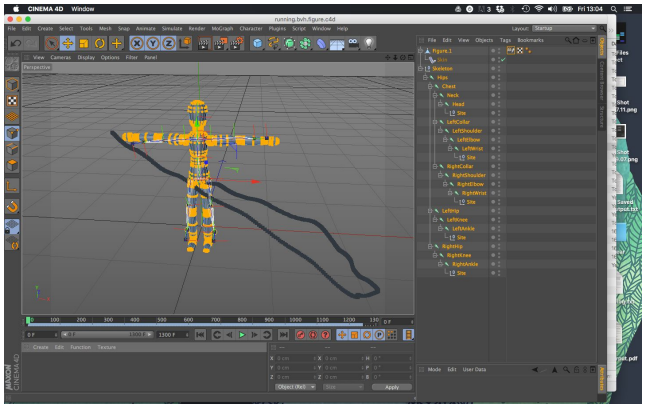

image #3.9: model bound to animated skeleton

- Go back to model mode (Tools → Modes → Model) and DE-select everything by clicking somewhere outside of any of the objects. If you scroll through the timeline, now the 3D model should move along with the skeleton (figure #3.10)

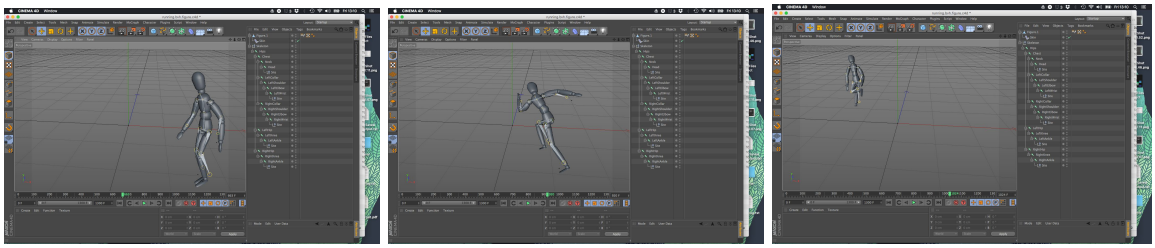

image #3.10: animated 3d model

For more information have a look at the following youtube tutorial:
https://www.youtube.com/watch?v=wHMyJ5w_KEA

04 Use Case: Visualising Trajectories in Cinema4D

This manual assumes you have recorded MoCap data in FBX format. For example data and c4d project files you can download the file **MoCap-cinema4d.zip** from the following page: https://drive.google.com/folderview?id=0BzwxZounZTt9bjYxWmlnRGZYRE0#list

Import FBX

- In Cinema4D go to File → Open
- Find and open the FBX file you want to use
- There will be another popup window; make sure that basic > animation is checked and press Ok
- In the next popup select which take you want to import (if there is only one take in the file, you can simply press OK)
- This should import a whole bunch of animated Null objects into your scene (image #4.1)

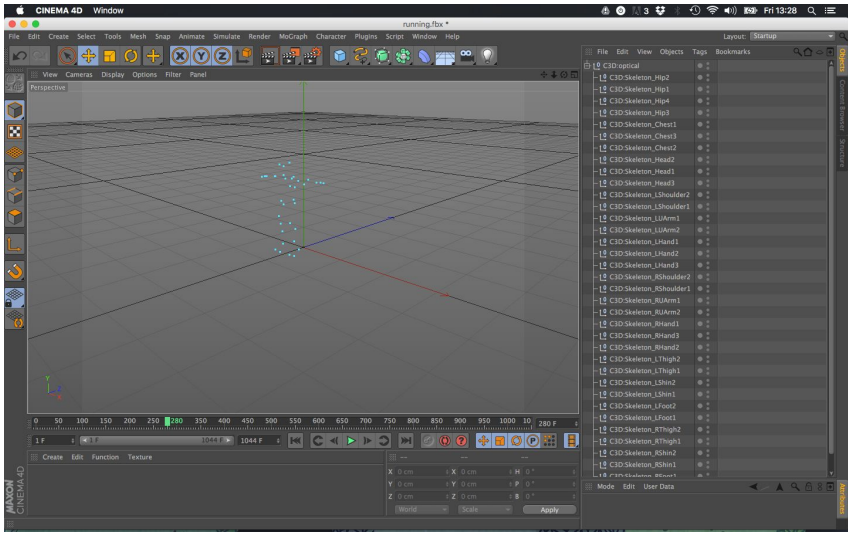

Image #4.1: imported Null objects

Trace Object Paths

- Create Tracer Object (MoGraph → Tracer)
- Add all imported Nulls that you want to trace to the *Trace Link* property in the Tracer's Object tab by clicking on the circle with the arrow and selecting each Null in the object manager one-by-one (image #4.2)

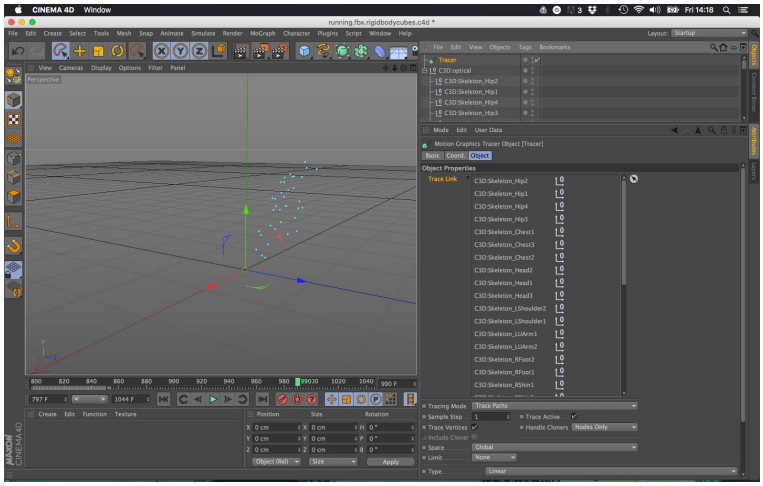

Image #4.2: add Nulls to the Trace Link

- Now, when you scroll through the timeline, you see the paths that are being traced show up as black lines (image #4.3)

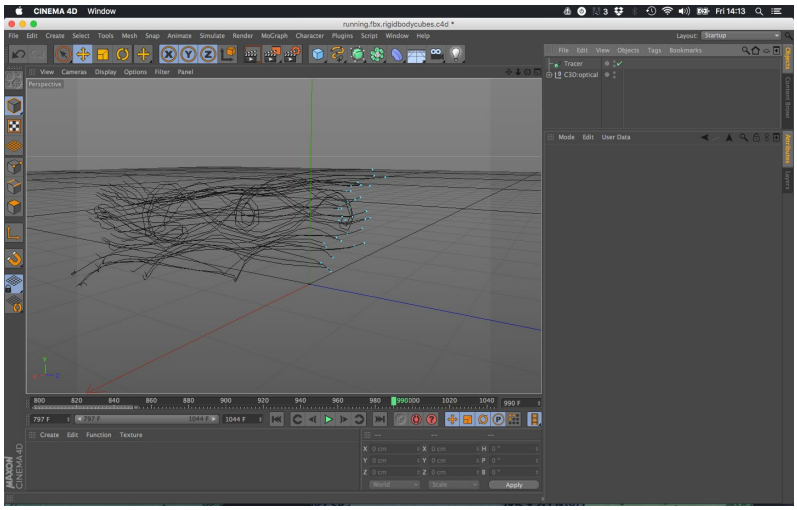

Image #4.3: traced path

- These paths aren't ready to be rendered yet (if you press CMD+R, you'll see you still get an empty black image)
- Create a Sweep NURBS object (Create → NURBS → Sweep NURBS) and a circle spline (Create → Spline → Circle)
- Make the circle very small (radius of about 0.5cm) and make the circle object a child of the Sweep NURBS object
- Make the Tracer object the second child of the Sweep NURBS object. Now the traced paths should appear as 3d meshes (image #4.4 and #4.5).

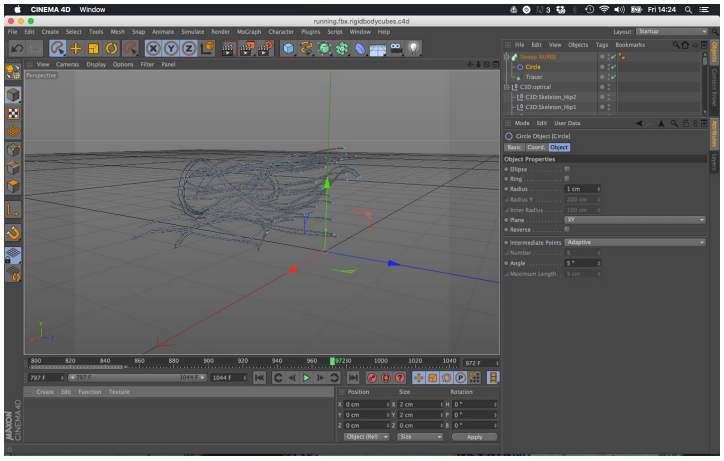

Image #4.4: 3d meshed of traced paths

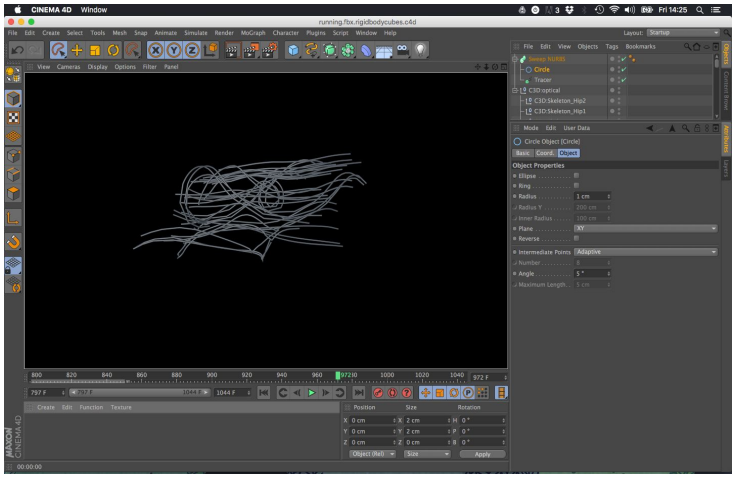

Image #4.5: rendered traces

For more fun stuff you can do with a tracer have a look at:
http://greyscalegorilla.com/tutorials/how-to-use-the-tracer-object-in-cinema-4d/

05 Use Case: Visualising real-time MoCap data in Unity3D

This manual assumes you have a calibrated the MoCap system (see the *Preparing the Motion Capture system* chapter) and that Motive is streaming real time data over the network (see the *Preparing Motion Capture Data* chapter).

Before you start, go to the following page and download the files listed below:
https://drive.google.com/folderview?id=0BzwxZounZTt9bjYxWmlnRGZYRE0#list

- **MoCap-forwarder-1.0.app.zip** (for OSX; translates Motive's NatNet data into OSC)
- **MoCap-OSC.unitypackage** (package with unity assets that receive the OSC data)

Connect your computer to the MoCap Network

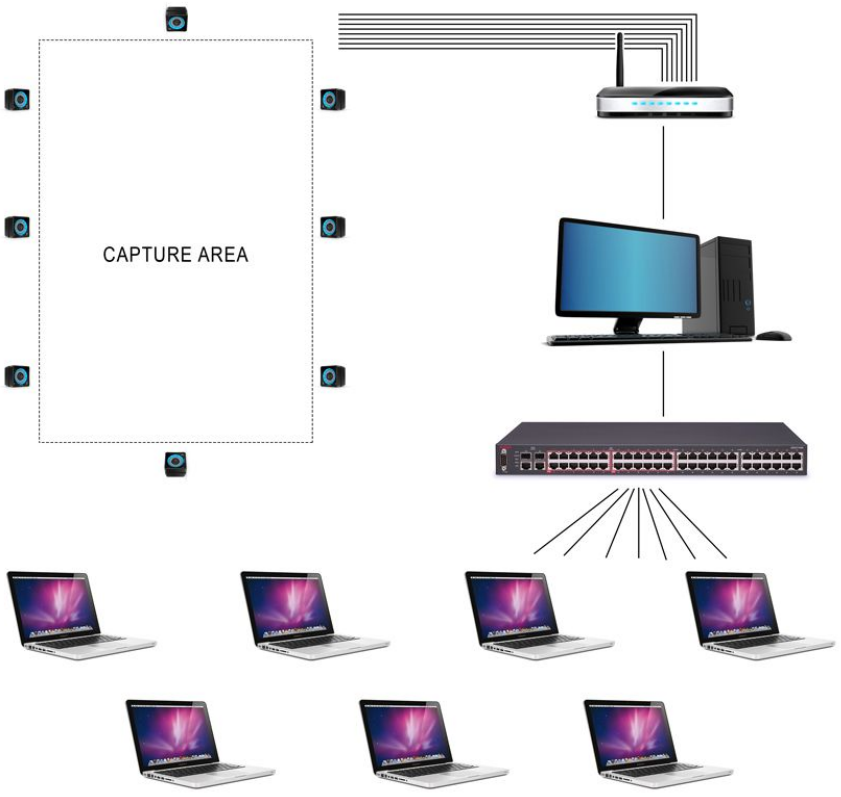

Image #5.1: System schema

- Make sure the MoCap PC is connected to the network switch
- Connect your own computer (which will be running Unity) to the network switch

- In the network settings of your operating system configure your computer with a static/manual IP address, something like 192.168.1.X where X can by anything between 10 and 200, just make sure the number isn't already taken by somebody else on the MoCap network.
- **Disconnect your computer from any other networks (like WIFI)**

Run the PyMoCap Osc Forwarder

The Motive data has to be translated to OSC messages. For this you have to run a separate *forwarder* application.

- Unzip the downloaded **MoCap-forwarder-1.0.app.zip** file
- Run the application (Image #5.2, currently only works on OSX)

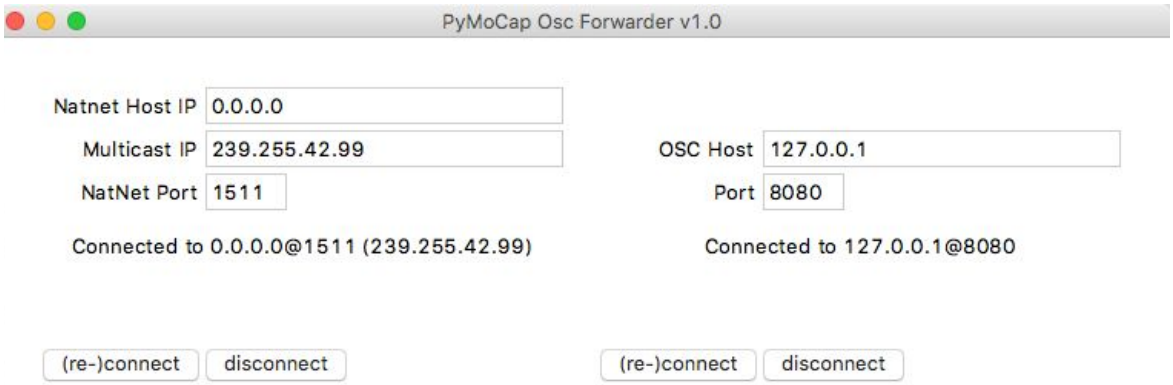

Image #5.2 MoCap Bridge

Create your Unity application

- Open your Unity project or start a new one
- While your unity project is loaded, open the **MoCap-OSC.unitypackage**
- Make sure all assets are selected and click "Import" (image #5.3)
- Drag the **MoCap-OSC** prefab from the **MoCap-OSC** asset folder into your scene

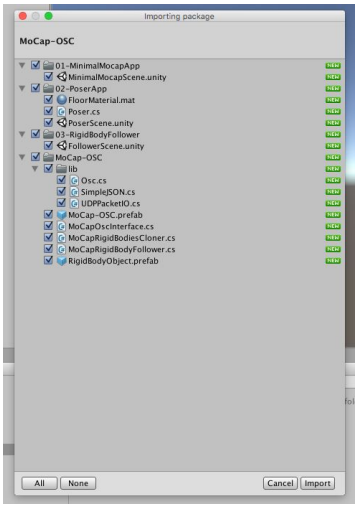

Image #5.3: Import assets

Press play to run the game. If the MoCap system and the forwarder are running properly, you should see a cube for every rigid body.

Have a look at the 3 example scenes from the MoCap-OSC package for more functionality.

06 Use Case: Getting real-time MoCap data in Blender

This manual assumes you have a calibrated the MoCap system (see the *Preparing the Motion Capture system* chapter) and that Motive is streaming real time data over the network (see the *Preparing Motion Capture Data* chapter).

Before you start, go to the following page and download the files listed below:
https://drive.google.com/folderview?id=0BzwxZounZTt9bjYxWmlnRGZYRE0#list

- **blender-pymocap-addon-v1.0.0.zip**
- **blender-mocap-example.zip**

Connect your computer to the MoCap Network

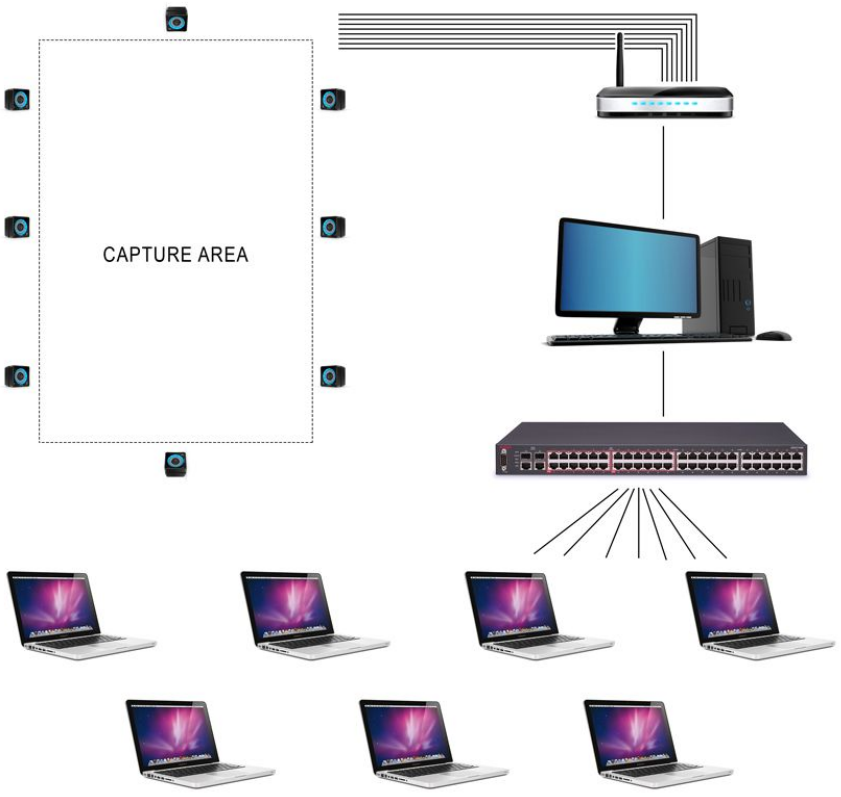

Image #6.1: System schema

- Make sure the MoCap PC is connected to the network switch
- Connect your own computer (which will be running Unity) to the network switch (image #6.1)

- In the network settings of your operating system configure your computer with a static/manual IP address, something like 192.168.1.X where X can by anything between 10 and 200, just make sure the number isn't already taken by somebody else on the MoCap network.
- **Disconnect your computer from any other networks (like WIFI)**

Install Blender addon

- Open blender
- Open User Preferences (File → User Preferences)
- Go to the Add-ons tab and click **Install from File…** at the bottom (image #6.2)
- In the file dialog find the file **blender-pymocap-addon-v1.0.0.zip** and install it
- Search for the addon by typing **mocap** in the search field (top left, image #6.2)
- Enable it by checking the checkbox on the right of the **System: PyMoCap** addon

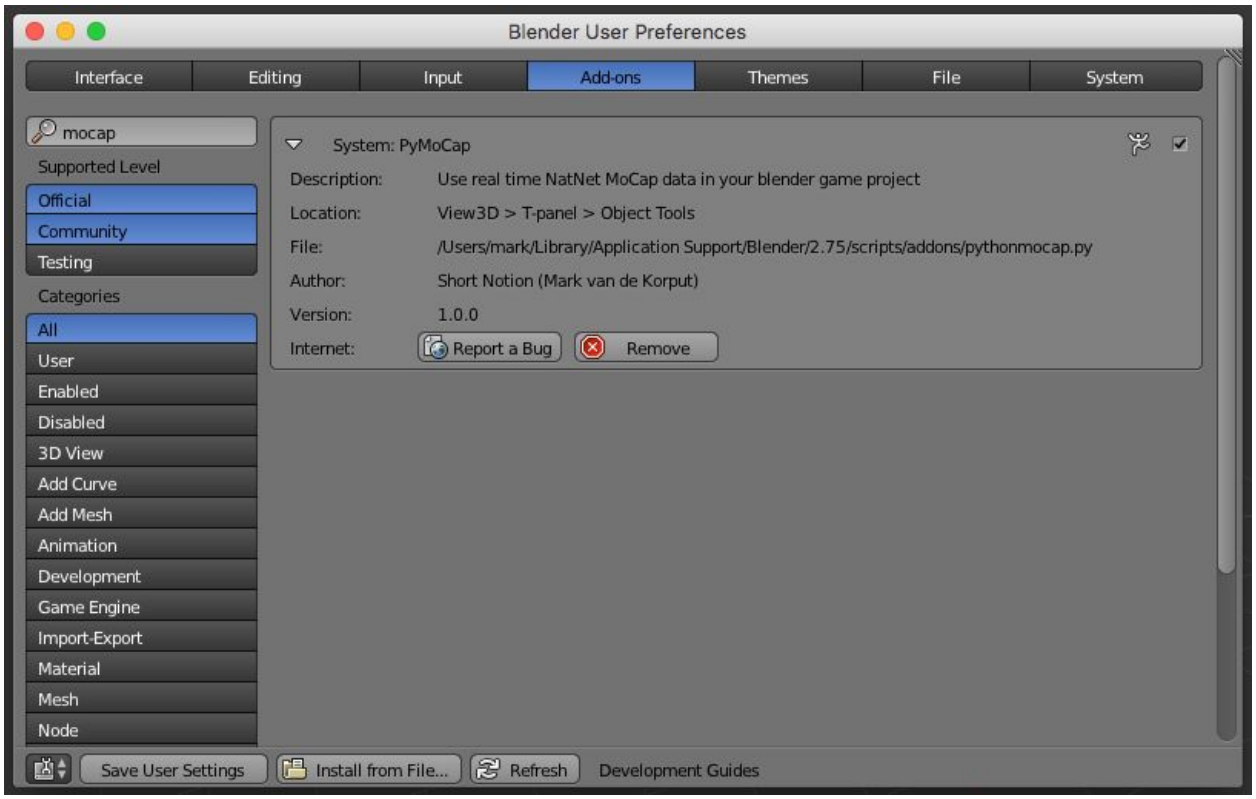

Image #6.2: installing and enabling the PyMoCap blender addon

- Finally click **Save User Settings** (at the bottom, image#62.2), otherwise you have to go back and enable the addon every time blender restarts

Receiving live NatNet data

- Create an empty object (Add → Empty → Plain Axes) and give the object a nice name. Let's call it mocap.
- With the mocap object selected, open the **object panel** (image #6.3)

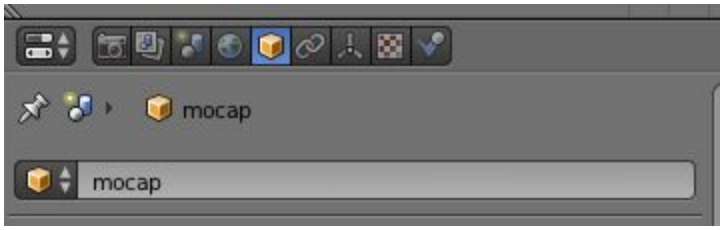

Image #6.3: object panel

- Enable and unfold the PyMoCap section at the bottom (image #6.4)
- Check the Natnet Receiver option (image #6.4) this enables to module that receives the live streaming data from Motive
- Click the Configure button at the bottom of the PyMoCap section (unless it already says **Game Logic: OK**, image #6.4). This creates an *Always sensor* and a *Python module controller* for the object in the *Game Logic* editor

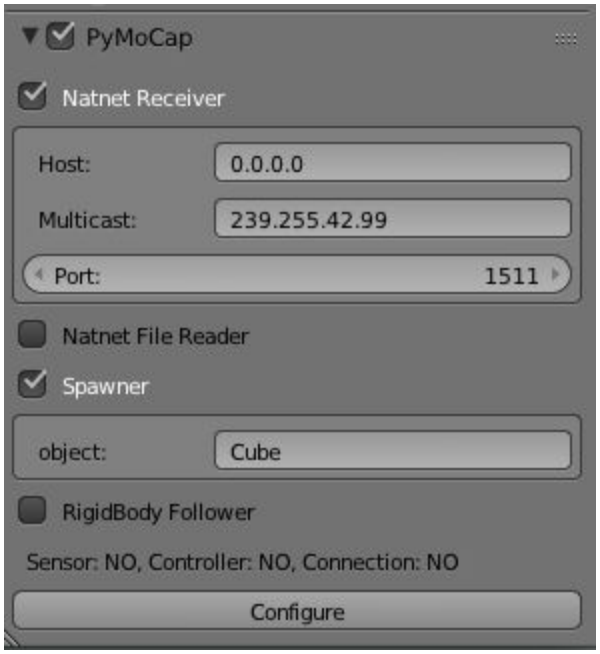

Image #6.4: PyMoCap section

Your project is now configured to receive the live MoCap data. Next, it's time to do something with that data.

Spawning Rigid Body objects

To quickly visualise all rigid bodies in the mocap system, the PyMoCap addon has a convenient *Spawner* module. Before we can use it though, we need to create an object to be spawned for the rigid bodies. We'll be creating a simple cube for this tutorial. The catch is that the blender game engine requires objects to be on a 'hidden' layer, otherwise we can't spawn new instances.

- Select any other layer than the layer with your game scene (image #6.5)
- With this new layer selected, create your object
- Take note of the name of your new object, **Cube** in our case (image #6.6)
- Go back to your original layer

Image #6.5: Select a hidden layer

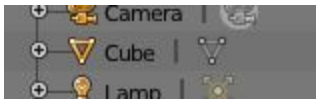

Image 6.6: Object Name

Now that we have a (hidden) template object, we'll configure the Spawner module to create one of these objects for every rigid body in the MoCap system.

- Go back to the **PyMoCap** section of your mocap object (image #6.4).
- Check the **Spawner** option (image #6.4)
- In the object field, enter the name of the hidden object we created earlier. **Cube** in our case (image #6.4)
- If the game logic for this object wasn't created yet, click the **Configure** button (image #6.4)

That's almost it. Now it's time to test. Before you can run the game, you have to set the *render engine* to **Blender Game** (image #6.7). Then start the game by going to Game → Start Game Engine.

Image #6.7: Render Engine

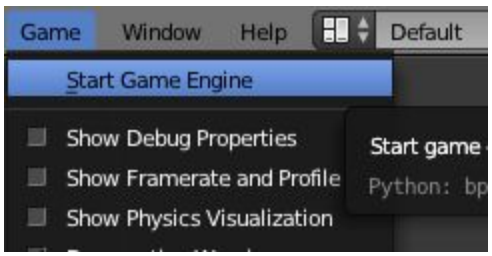

Image #6.8: Start Game Engine

Additional Notes

The object with the *PyMoCap Natnet Receiver* module enabled doesn't have to be the same object as the one with the *Spawner* module. The data from the *Natnet Receiver* will be available to all objects in the game. This does mean that only one object should be running the Natnet Receiver.

You can create multiple *Spawner* objects. The rigid body objects are positioned relative to the Spawner object, so you can use the position of the objects to change the global position of the rigid bodies (image #6.9)

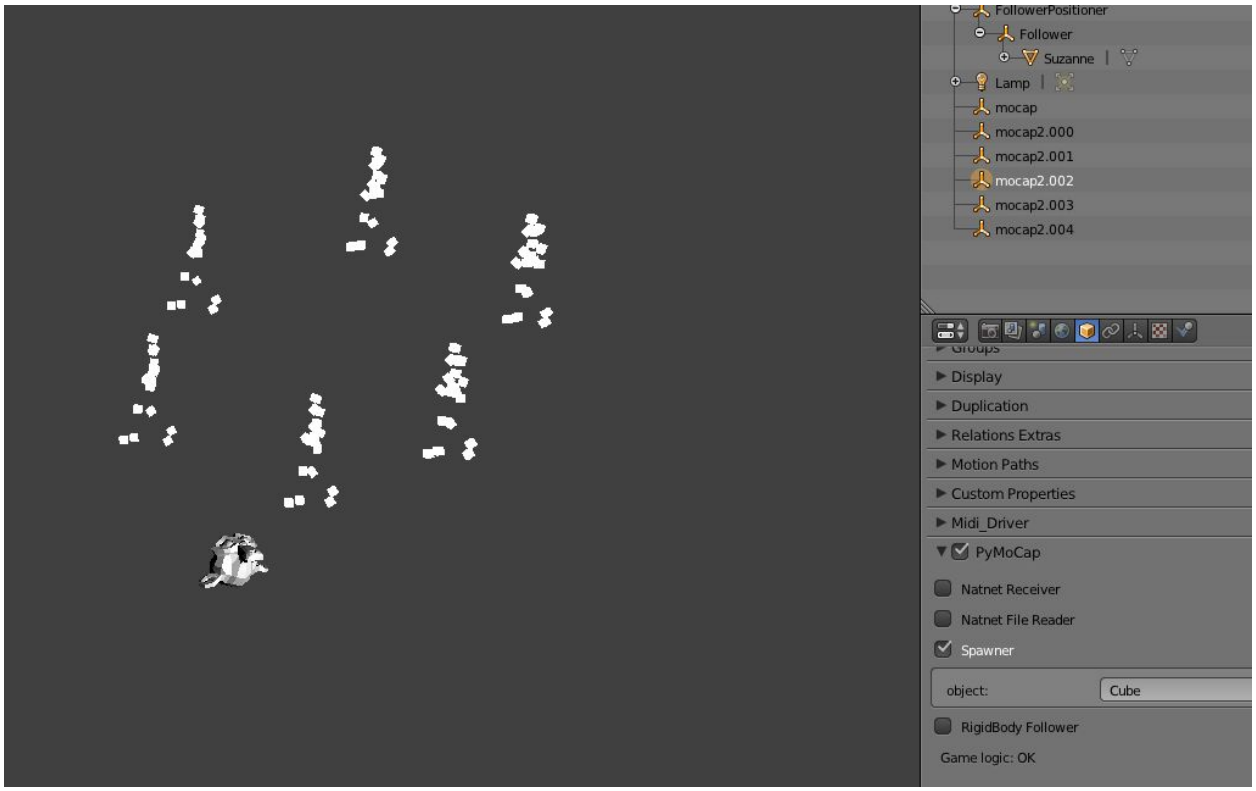

Image #6.9: Example project - multiple spawner objects and a follower monkey head

Unzip the downloaded **blender-mocap-example.zip** file and open the mocap.blend example project for a demo of the **RigidBody Follower** module and how to use MoCap data from a pre-recorded file (image #6.9)