1 Overview

This chapter describes how to import models from OpenSim into Artisynth.

1.1 OpenSimParser

To import an OpenSim model, we use the class OpenSimParser. The OpenSimParser requires 2 inputs: the OpenSim file, and the path where the OpenSim geometry is located. If no geometry path is specified, Artisynth will check for the folder "geometry1" relative to the OpenSimParser class path.

Listing 1 shows a complete example of importing an OpenSim model. The code is an amended version of the demo OpenSimSpine based on the OpenSim model by Anderson et al. (https://simtk.org/projects/spine_ribcage).

```
package artisynth.demos.test;

import java.io.*;
import artisynth.core.workspace.*;
import artisynth.core.mechmodels.*;
import artisynth.core.opensim.OpenSimParser;
import maspack.util.*;
import maspack.fileutil.*;
import maspack.fileutil.uri.URIx;

public class OpenSimSpineSimple extends RootModel {

   public static boolean omitFromMenu = false;
   private boolean useFrameSprings = true;

   String data_url =
      "https://www.artisynth.org/files/data/Female_Thoracolumbar_Spine_V1.zip";


   public void build (String[] args) throws IOException {

      MechModel mech = new MechModel ("mech");
      addModel (mech);

      String localPath = PathFinder.findSourceDir(OpenSimSpine.class);
      String dataPath = localPath+"/Female_Thoracolumbar_Spine_V1";
      if (!(new File(dataPath)).exists()) {
         System.out.println ("Downloading "+data_url+" ...");
         try {
            ZipUtility.unzip (new URIx(data_url), new File(localPath));
         }
         catch (Exception e) {
            e.printStackTrace();
            throw e;
         }
      }

      File osimFile =
         new File (dataPath+"/Female_Thoracolumbar_Spine_Model.osim");
      String geometryPath = dataPath + "/Geometry/";
      OpenSimParser parser = new OpenSimParser (osimFile);

      parser.setGeometryPath (new File(geometryPath));

      // create model
      parser.createModel (mech);
   }
}
```

Listing 1: Import of an OpenSim spine model into Artisynth

1.2 Accessing Components for Model Modification

Once we have imported our model, we can then make adjustments to the model. To do so, we need to access model components from the hierarchy. In Artisynth, when creating a MechModel, components are stored in lists specifying the component type. The default names of these lists can be found in MechModel. FrameMarkers in a MechModel are stored in a PointList<FrameMarker> called "frameMarkers", multiPointSprings are stored as a PointSpringList<MultiPointSpring> called "multiPointSprings", etc.

For example, in the demo MuscleArm, a MultiPoint muscle is created with 2 attachment points ("upperAttachment" and "lowerAttachment").

The hierarchy then looks as follows:

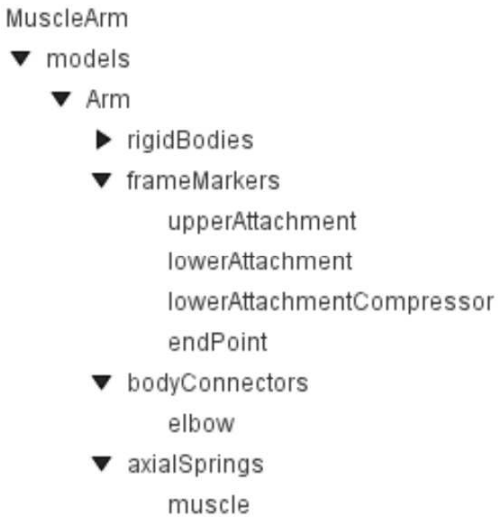

Figure 1: A typical Artisynth model hierarchy

> This demonstrates the typical Artisynth MechModel hierarchy: the frameMarkers are stored by type in the list "frameMarkers", regardless of which spring/muscle they belong to. This organization is in constrast to OpenSim, which stores the points (both the attachment points and path points of a muscle) in a list beneath that particular muscle in the hierarchy.

Below we can see the default hierarchy created from our OpenSimSpineSimple example above: The hierarchy then looks as follows. All attachment points for the muscle "Ps_L1_VB_r" are stored in a folder called "Ps_L1_VB_r_path", adjavent to the muscle model itself.

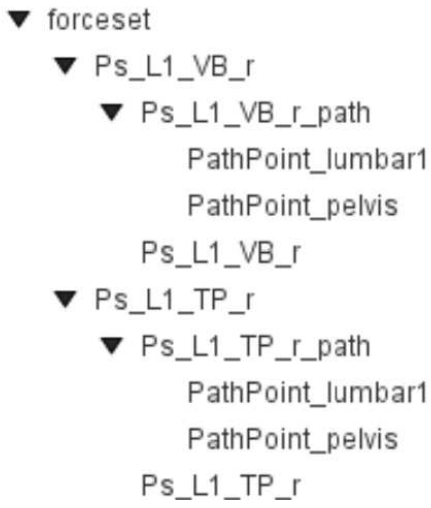

Figure 2: A typical OpenSim model hierarchy

Knowledge of the hierarchy is imperative for accessing components in the model for modification.

1.2.1 Modifying Model Components

Let's take a case in which we wish to modify the location of an attachment point of a muscle in the demo. Let's change the location of "PathPoint_pelvis" of the muscle "Ps_L1_VB_r".

To change its location, we first need to access that specific frameMarker. **Usually**, we could just access all of the frameMarkers attached to a rigid body, and then get the frameMarker by its name, as follows:

```
ComponentList<RigidBody> bodies =
(ComponentList<RigidBody>)mech.get("bodyset");
RigidBody pelvis = bodies.get ("pelvis");
for (FrameMarker mk :  pelvis.getFrameMarkers ()) {
 if (mk.getName ().contains ("PathPoint_pelvis")) {
    mk.setLocation (new Point3d(-0.023708, -0.0544284, 0.0756066));
 }
}
```

The method (`getFrameMarkers()`) works even with the OpenSim hierarchy, where the markers are contained within the muscle folders. **However**, as we can see in 2 in the OpenSimSpine model, several muscles have an attachment point of the same name of "PathPoint_pelvis". Therefore, our method above would modify all of those pelvis attachment points. Instead, to modify just the pelvis attachment point of our particular muscle "Ps_L1_VB_r", we first need to retrieve the muscle by name, then get the folder "Ps_L1_VB_r_path", and *then* retrieve the point. To do this, we can use the following code:

```
RenderableComponentList<ModelComponent> forceset =
(RenderableComponentList<ModelComponent>)mech.get ("forceset");
for (ModelComponent mc : forceset) {
  if (mc instanceof RenderableComponentList) {
     if (mc.getName().contains ("Ps_L1_VB_r")) {
        ComponentList<ModelComponent> rl = (RenderableComponentList)mc;
        RenderableComponentList<ModelComponent> pathPointList = ( ↩
            RenderableComponentList<ModelComponent>)rl.get (0);
        for (ModelComponent c : pathPointList) {
           System.out.println(c.getName());
           if (c.getName().contains ("PathPoint_pelvis")) {
              System.out.println("found pelvis marker");
              FrameMarker pelvisPoint = (FrameMarker)c;
              pelvisPoint.setLocation (new Point3d( -0.023708, -0.0559504, 0.083552));
           }
        }
     }
   }
 }
```

1.2.2 FrameMarkers vs JointBasedMovingMarkers

Muscle points imported from OpenSim can either be `PathPoints` or `MovingPathPoints`. PathPoints are equivalent to Artisynth `FrameMarkers`; these points keep a constant position relative to the body they are attached to, specified by their `Location`. OpenSim `MovingPathPoints`, which are called `JointBasedMovingMarkers` in Artisynth, have their position defined as a function of joint coordinates. The OpenSimParser in Arisynth automatically imports these two different types of points as their Artisynth equivalent. Note that one cannot simply ovverride the location of a MovingPathPoint with `setLocation()`, since the function of the JointBasedMovingMarker will override this as soon as the model loads. Instead, to set a rigid location for a JointBasedMovingMarker, it must first be converted into a FrameMarker.