Matlab Resources for the Advanced User Workshop

Scripting Matlab is a common theme for many of the workshop projects, so we have compiled a set of helpful resources and links.

- **Useful Links**
- Example Matlab Code and Utilities
  - Matlab Utilities
  - Example Code
- Troubleshooting Tips
  - Case 1: I'm getting an error that a function/method I want to use doesn't exist, but I know that it does!
  - Case 2: The thing (class) I want to create has a name like `Array_<double>`.
- Adding Geometry Paths to Matlab

Useful Links

The OpenSim documentation includes several resources for scripting. The following pages are helpful resources:

- Common Scripting Commands (a list and description of the most commonly used scripting commands)
- Scripting Versions of OpenSim C++ API Calls (information to help you use what are called "templatized" classes in OpenSim through the scripting interface)
- Scripting with Matlab (set-up instructions, examples and utilities, and other Matlab-specific scripting information)
- Converting C3D Files (a guide to working with C3D files through scripting)

Example Matlab Code and Utilities

Matlab Utilities

The OpenSim 4.0 resources directory includes a folder of Matlab Utilities that can be used for common tasks.

Once you setup your Matlab environment, add the OpenSim Utilities folder to your Matlab path. The Utilities folder is located in your OpenSim resources directory. (This is typically a path like `/Documents/OpenSim/4.0/Code/Matlab/Utilities`).

For example, one of the utilities (osimVec3FromArray) helps you to convert between an OpenSim Vec3() and a Matlab Array:

```matlab
>>> import org.opensim.modeling.*
>>> vec3 = osimVec3FromArray([1 2 3]); % Converts a Matlab Array to a OpenSim Vec3
>>> matlabArray = osimVec3ToArray(vec3); % Converts an OpenSim Vec3 to a Matlab Array
```

Two additional functions that may be helpful to many are:

- osimTable2Struct() and
- osimTableFromStruct()

These utilities convert between OpenSim `TimeSeriesTables` (a new storage format in 4.0 for time series data) and Matlab Structs. Be aware that any metadata in your Table will be lost when you convert to a Matlab Struct.

Example Code

Often the best way of figuring out how to code is to use examples from already written scripts or functions. Many example scripts and functions can be found in the OpenSim 4.0 resources directory (Typically `/Documents/OpenSim/4.0/Code/Matlab`). You can also download some additional example scripts that use the current API:

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plotMuscleFLCurves.m</td>
<td>Function to compute and plot the active and passive force-length muscle curves</td>
</tr>
<tr>
<td>createActuatorsFile.m</td>
<td>Function to generate a generic OpenSim Actuator File from a Model</td>
</tr>
<tr>
<td>build_and_simulate_simple_arm.m</td>
<td>Demonstrates building and simulating a simple arm model</td>
</tr>
<tr>
<td>pendulum_marker_positions.m</td>
<td>Build, simulate, and generate outputs for a double-pendulum model. Writes the results to .sto and .trc files.</td>
</tr>
</tbody>
</table>

Troubleshooting Tips
In addition to looking at example code, another great way to learn more about scripting in Matlab is to review the OpenSim Doxygen. The doxygen documents all the functionality of the OpenSim API. Guide to Using Doxygen.

The OpenSim API is written in the C++ programming language and not all of the functionality of C++ (as it is documented in the doxygen) translates perfectly to Matlab. There are two common cases where you may run into confusion or errors.

Case 1: I'm getting an error that a function/method I want to use doesn't exist, but I know that it does!

If you are calling a method or function (e.g., getting or setting properties) that you are pretty sure should work, but you are getting an error that the method doesn't exist, this may mean that you need to do something called downcasting. In the C++ programming language, programmers use a concept called "inheritance" to build up complexity without re-writing the same code multiple times. For example, in OpenSim Thelen2003Muscle and MillardMuscle both rely on code in the common parent Muscle class that they "inherit" from.

If you have a handle to a base class object (e.g., Muscle) you may need to downcast the object to one of its derived (or concrete) classes, like the Thelen2003Muscle, in order to gain access to properties and methods specific to the concrete class.

In this code example, we do get a reference to a muscle in the model and return the class name and concrete class name.

```matlab
>>> model = Model(path2model)
>>> muscle = model.getMuscles().get(0);
>>> muscle.getClassName()
muscle
>>> muscle.getConcreteClassName()
Thelen2003Muscle
```

Then to get a version of the object that you can call the functions of interest on, you use safeDownCast()

```matlab
>>> muscle = model.getMuscles().get(0);    #the object you get here is of base class Muscle
>>> thelenMuscle = Thelen2003Muscle.safeDownCast(muscle)  # to use a method specific to Thelen2003Muscle you need to safeDownCast
>>> timeConstant = thelenMuscle.getActivationTimeConstant()
```

Case 2: The thing (class) I want to create has a name like Array_<double>.

As you look through the OpenSim doxygen or if you look at OpenSim API examples, in C++, you may see objects/classes with names like Array_<double> or Vec<3>. These are called templatized classes; this is another functionality that exists in C++ to help programmers simplify and reuse common code. But this doesn't exist in Matlab. So we have created scripting versions of the most commonly used templatized classes. You can find a list of all of these on the page Scripting Versions of OpenSim C++ API Calls. If you see a class name with angle brackets (< >), you can look it up on this page to find the class name to use in Matlab.

Adding Geometry Paths to Matlab

OpenSim verifies that Model's Geometry mesh files exists when you load a model and whenever you subsequently operate on it. If you have a Geometry file defined in your model, and the Geometry path is not set, you will get a warning for each piece of Geometry that can't be found, as shown in the image below.
These warning can get cumbersome, especially if you are doing batch processing or iterative analysis. To stop the warnings, you have to set the Geometry folder path of OpenSim:ModelVisualizer() in Matlab. Copy and paste the below code and update the path to your local OpenSim Geometry directory.

```
>>> path='/Users/username/Applications/OpenSim 4.0/OpenSim 4.0.app/Contents/Resources/OpenSim/Geometry';
>>> ModelVisualizer.addDirToGeometrySearchPaths(path);
```

Alternatively, a ’workaround’ that if you are not visualizing the model is to remove the geometry from the model. This should only be used if you are doing many model instantiations or initSystem() calls and don’t need to visualize the model.

## Outputs

OpenSim 4.0 uses component outputs and reporters to collect variables of interest and print them to file. To display the output names for a component, use the method getOutputNames();

```
>>> muscle.getOutputNames();
```