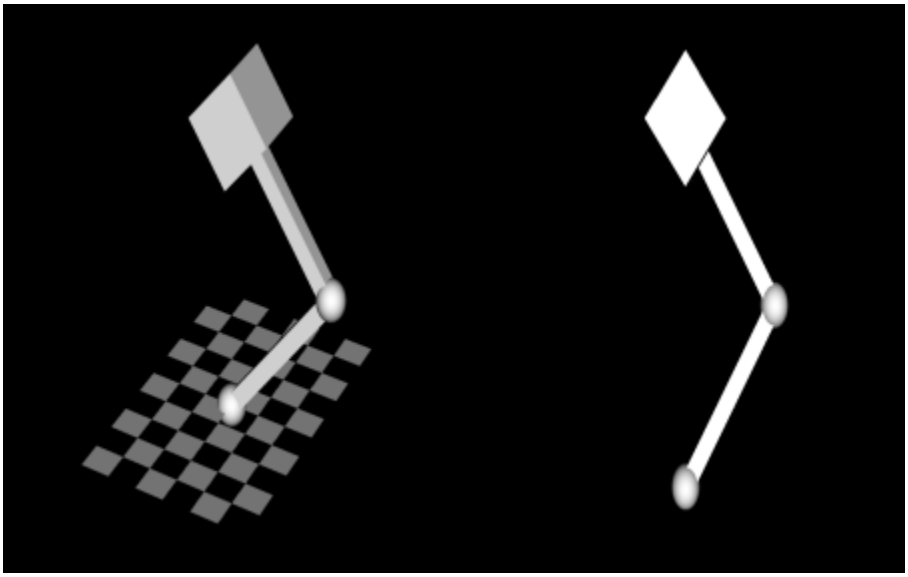
Creating an Actuator Part Two

The steps covered in part two are:

Using the ControllableSpring (toyLeg_example.cpp)

We can now use the ControllableSpring class in an example to see its effects. The *toyLeg_example.cpp* file we have provided implements a toy leg model that is driven by a PistonActuator (see toyLeg figure below).The model is built up in the sequence ground->linkage1->linkage2->block with pin joints between all the segments. The block is constrained to move only in the vertical direction. A PistonActuator called "piston" acts between the distal end of linkage1 and the center of the block. We will modify the main() routine to replace the piston actuator with a variable stiffness spring.



toyLeg example

Ready toyLeg_example.cpp to use the ControllableSpring

Open the *toyLeg_example.cpp* file, if you have not already done so. Add the ControllableSpring class to the included files as shown below.

```
#include "PistonActuator.h"
#include "ControllableSpring.h"
#include <OpenSim/OpenSim.h>

using namespace OpenSim;
using namespace SimTK;
```

Within Visual Studios, locate the Actuators_examples project. Right click it and select "Build" in order to rebuild *toyLeg_example.cpp* and force the first build of *ControllableSpring.h*. You will need to switch from "Debug" to either "Release" or "RelWithDebInfo" if you do not have debuggable OpenSim libraries with which to link.

Add a ControllableSpring to the model

Find the line after the piston is added to the model. At this location, create a ControllableSpring. Set it up to have the identical geometry as the piston, and add it to the model.

```
osimModel.addForce(piston);

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Add ControllableSpring between the first linkage and the second block
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ControllableSpring *spring = new ControllableSpring;
spring->setName("spring");
spring->setBodyA(block);
spring->setBodyB(&ground);
spring->setPointA(pointOnBodies);
spring->setPointB(pointOnBodies);
spring->setOptimalForce(2000.0);
spring->setPointsAreGlobal(false);
spring->setRestLength(0.8);
osimModel.addForce(spring);
```

Defining controls

We will use prescribed controller for the toy leg model, we give the piston actuator a constant function and the spring actuator a piece-wise linear function to simulate change in stiffness.

```
PrescribedController *legController = new PrescribedController();
legController->setActuators(osimModel.updActuators());

// specify some control nodes for spring stiffness control
double t[] = {0.0, 4.0, 7.0,  10.0, 15.0};
double x[] = {1.0, 1.0, 0.25,  0.25, 5.0};
// specify the control function for each actuator
legController->prescribeControlForActuator("piston", new Constant(0.982));
legController->prescribeControlForActuator("spring", new PiecewiseLinearFunction(5, t, x));

osimModel.addController(legController);
```

Save the resulting motion as a different file

Change the Save Results section in order to print the resulting toyLeg kinematics under a new file name.

```
// Save results
Storage statesDegrees(manager.getStateStorage());
osimModel.updSimbodyEngine().convertRadiansToDegrees(statesDegrees);

//statesDegrees.print("PistonActuatedLeg_states_degrees.mot");
statesDegrees.print("SpringActuatedLeg_states_degrees.mot");
```

Build and run the example

1. Build the **Actuator_examples** project again (see Section 4.3.3.1 above).
2. Then build the **INSTALL** project.
3. Make sure that the *<OpenSim2.0_intall_dir>/bin* directory appears at the front of your PATH. To check and/or set your PATH, go to Start -> System Properties (or System). Click on the Advanced tab and then select the Environment Variables button.
4. Navigate to the install directory and run the executable file, *toyLeg_example*. After running the executable, use the GUI to open the model *toyLeg.osim* and load the new motion file (*SpringActuatedLeg_states_degrees.mot*). Upon visualizing the motion, you should see the block oscillate at different magnitudes and frequencies as the spring stiffness is varied over time.

Next: Creating a Customized Muscle Model

Previous: Creating an Actuator Part One

Home: Scripting and Development | Developer's Guide | Adding New Functionality