Creating a Customized Actuator

In this exercise, we will create a specific type of actuator that implements a spring with controllable stiffness. The source code and associated files for this example come with the OpenSim 3.0 distribution under the directory:

C:\Program Files\OpenSim 3.0\sdk\APIExamples\CustomActuatorExample

When defining a new actuator, you can either start from scratch by deriving from the base class, CustomActuator, or if your actuator builds on an existing class, you can derive from that class. In this example we will implement a controllable stiffness spring by deriving from the PistonActuator class. The topics covered in this section include:

Actuator Overview

An actuator is a Force who's force (or forces or moments) applied to the model is a function of one or more controls. These could be torques applied between two bodies along a common axis, forces applied between two points defined on two separate bodies, or some combination of forces and moments applied according to their relative location and orientation (governed by the state of the model) and an external control signal that could be the voltage to a torque motor or the tension itself between two points. The key function of any actuator class is to calculate and apply forces and moments to its associated bodies based on its control value and the state of the model.

The PistonActuator class

In this exercise we wish to create a spring with controllable stiffness that acts between two points located on different bodies. Instead of building this actuator from the abstract Actuator class, we will instead derive our new class from the pre-existing PistonActuator class. This class is similar to the PointToPointActuator class defined within OpenSim. However, to serve as an example of how we design our actuator classes we have implemented and included the renamed version, PistonActuator, within the source material of this example. The figure below illustrates the PistonActuator class. This actuator applies a force between two points fixed on two bodies. These bodies do not need to be consecutive bodies in a kinematic chain. This class calculates the magnitude of its force as the product (optimalForce x control value) and uses the convention that a positive force magnitude acts to increase the distance between points P_A and P_B.

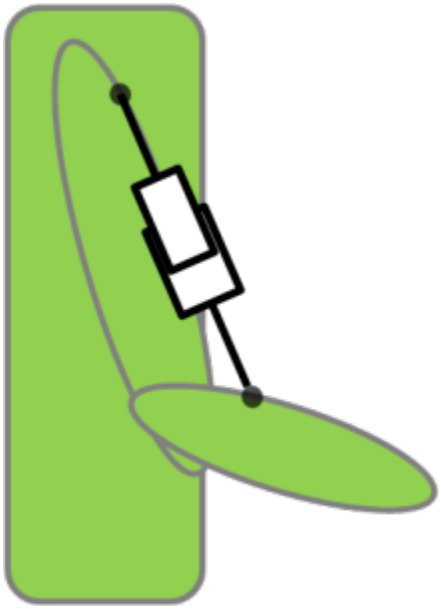

Illustration of the PistonActuator class

The ControllableSpring class

The figure below illustrates the ControllableSpring class that we will define. Just like PistonActuator, ControllableSpring will act between two points fixed on two different bodies. However, the force magnitude will not simply be calculated as the product of the optimal force and the control value. Instead, this product will represent the spring stiffness: **k = (optimalForce x control value)**. We will also have to define a rest length at which the spring produces no force. The force magnitude will then be calculated as **F = k**(restLength – currentLength).*

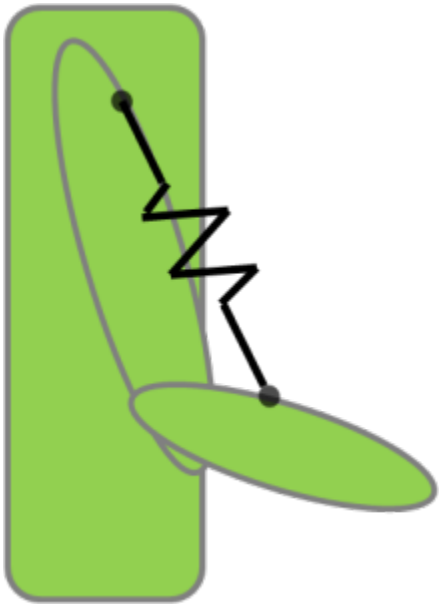

Illustration of the ControllableSpring class to be implemented

Next: Creating an Actuator Part One

Previous: Creating an Optimization

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