Introduction to OpenSim's Architecture and API

OpenSim Workshop @ SIMPAR | November 2010

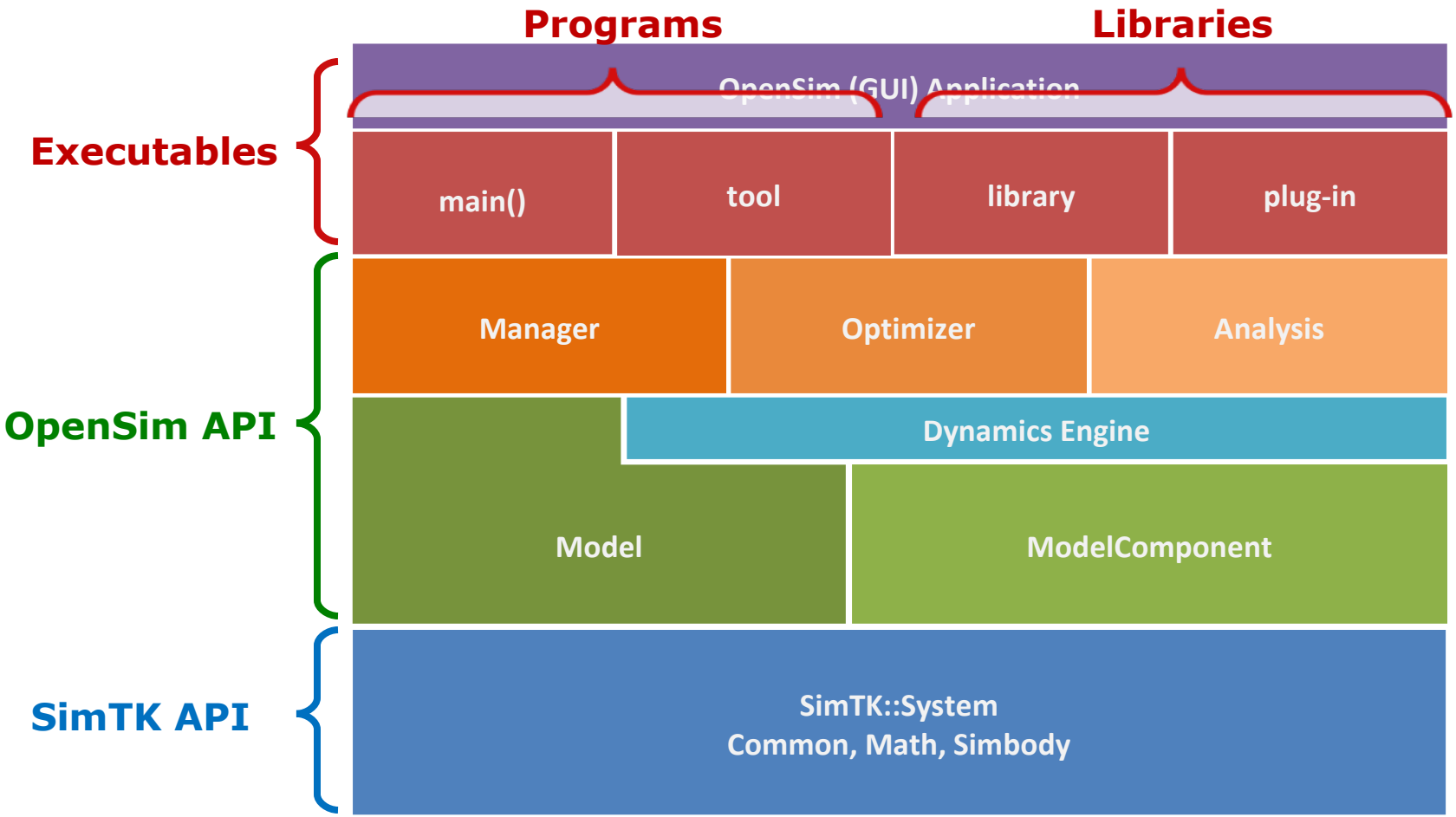
Purpose of the Workshop and the API:

Enable OpenSim users and developers to develop their own custom musculoskeletal models and simulations.

To do so one needs an understanding of:

1. How the OpenSim software is organized
2. What is the OpenSim Application Programming Interface (API)?

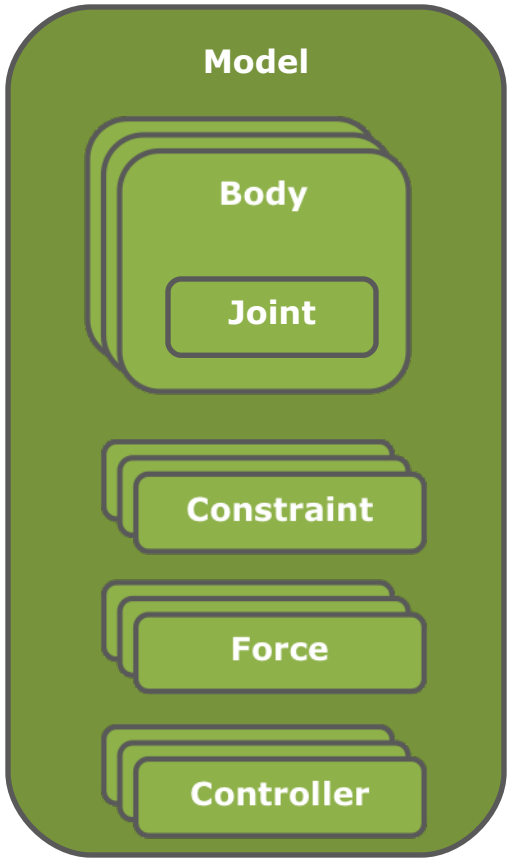
OpenSim Architecture: Interface Layers



OpenSim Model Class Structure:

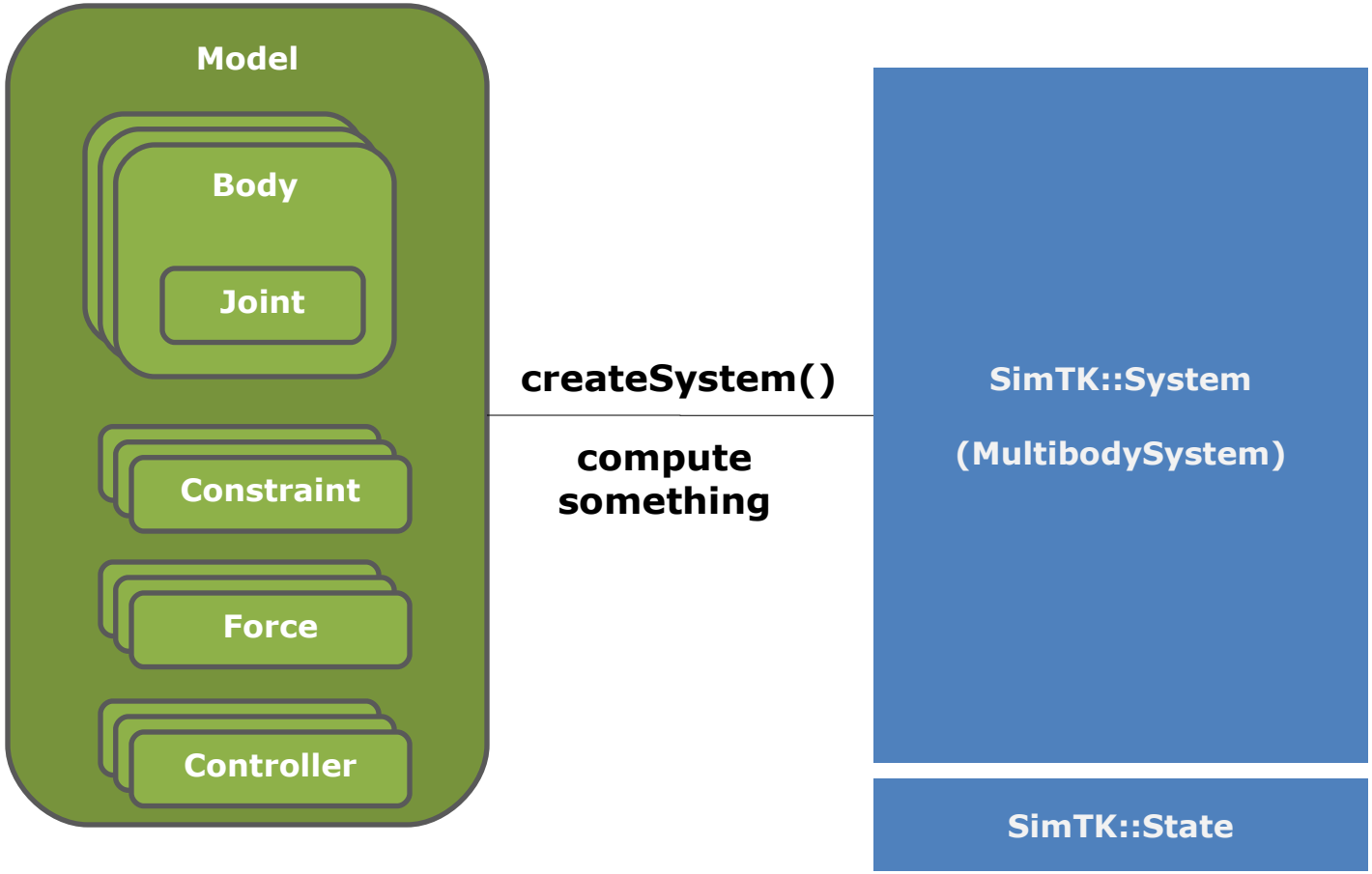
A model represents the physics of a mechanical system comprised of bodies and joints that are acted upon by forces to produce motion.

An OpenSim model is a bag of components corresponding to parts of the physical and subsequent computational system



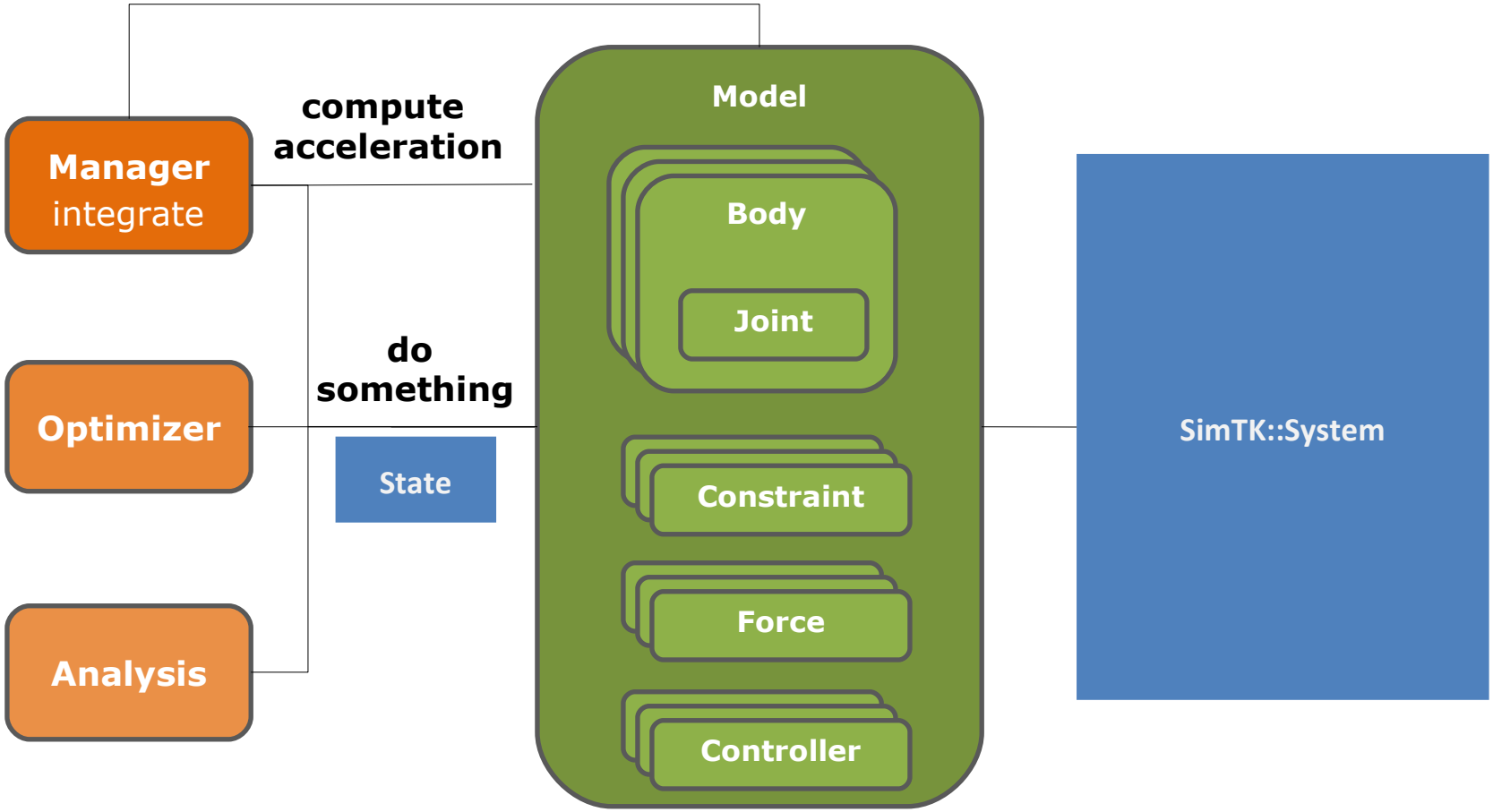
Organization vs. Computation

An OpenSim model and its components encapsulate properties of the physical system (mass, inertia, strength, etc …) and know how to add themselves to the underlying computational system (equations) to be solved.



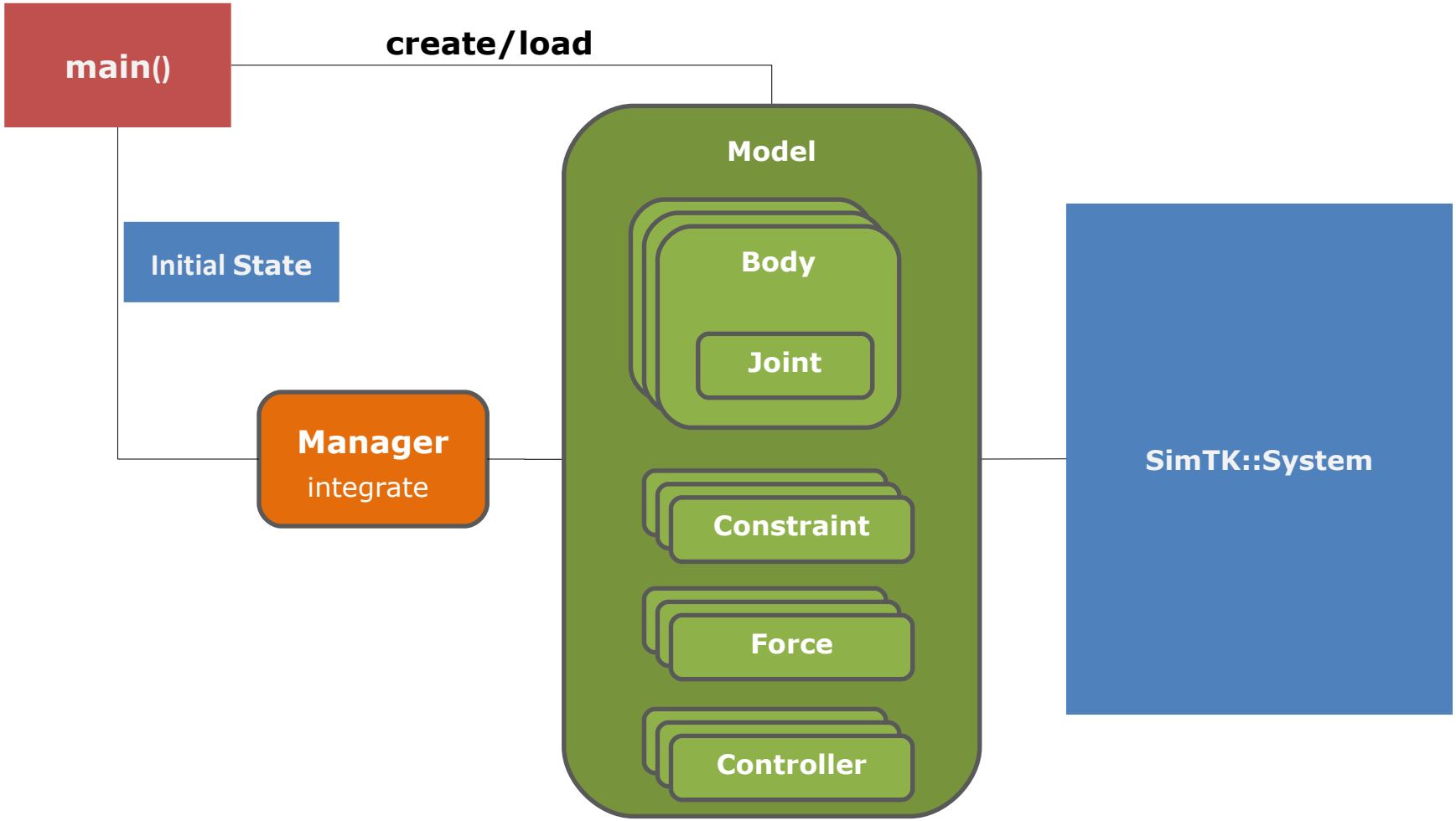
Objects that Make a Model "Do Something"

A model is called by higher level objects to perform (calculate something) and provide the model with a state.



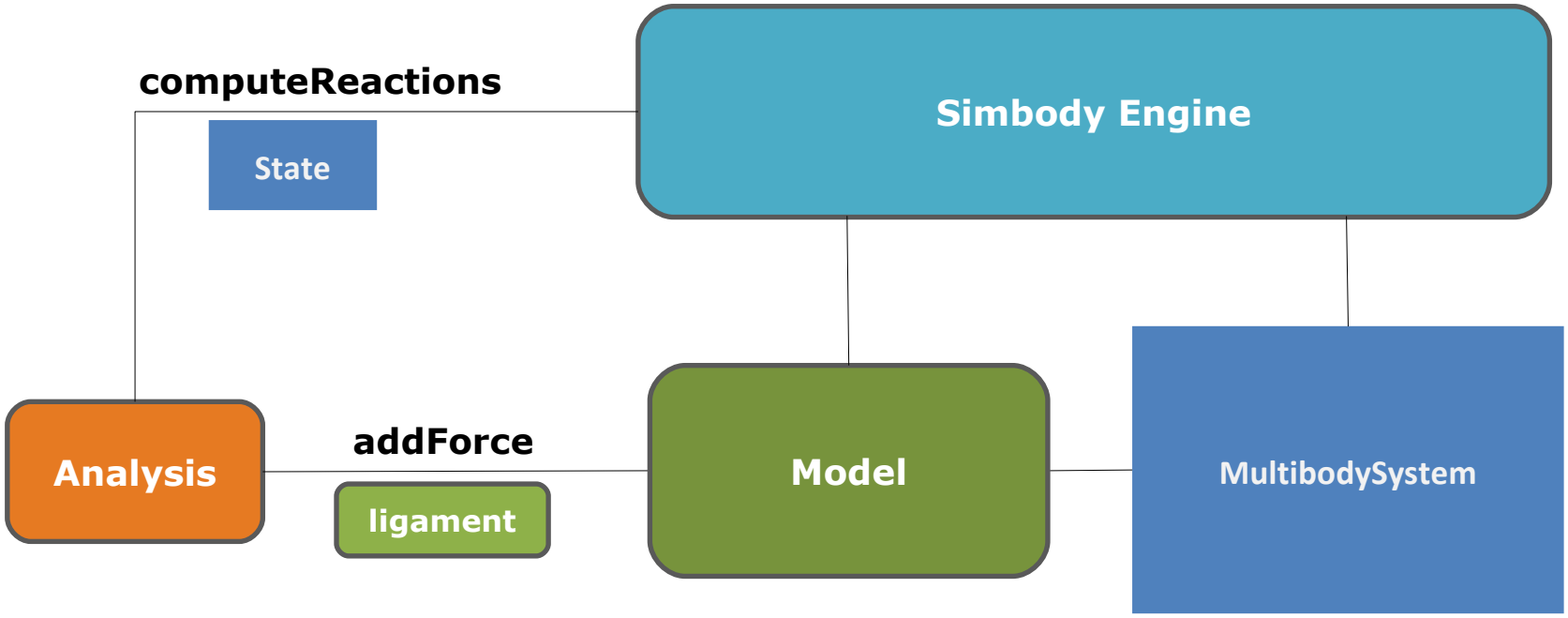
Your Program Creates & Commands Objects

You can define a main program to create a model and its manager and then execute a simulation.

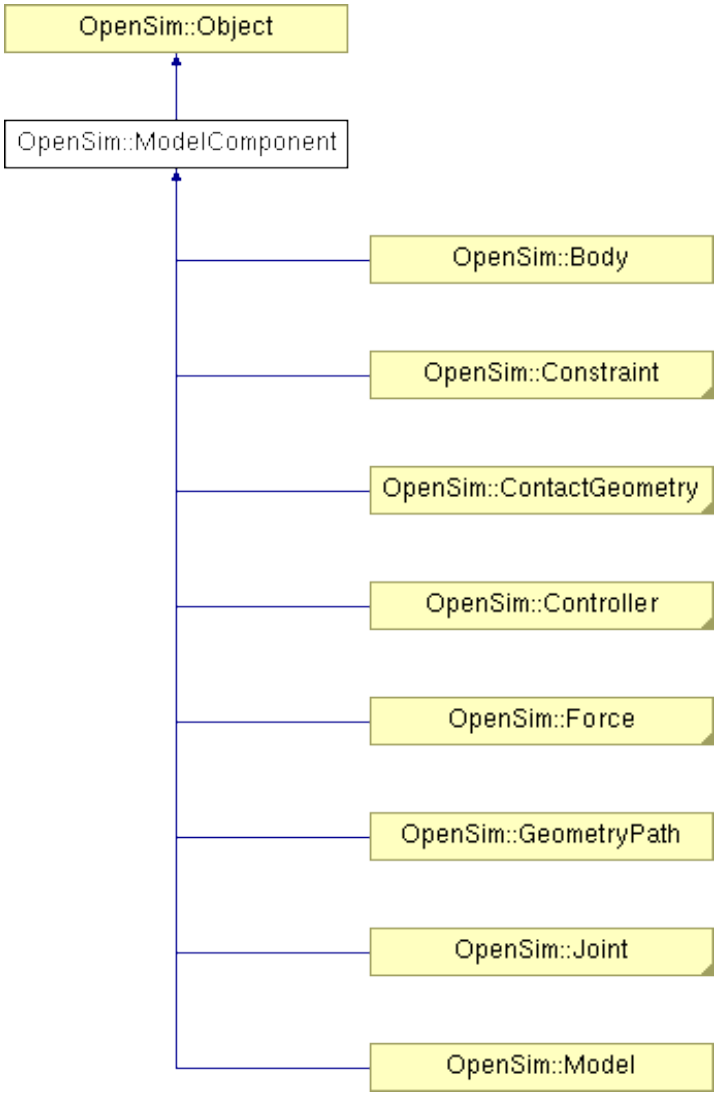


The SimbodyEngine (DynamicsEngine)

The SimbodyEngine is a convenience interface to the computational multibody dynamics system (Simbody) specified by the OpenSim Model.



OpenSim API Defined by Indidividual Classes



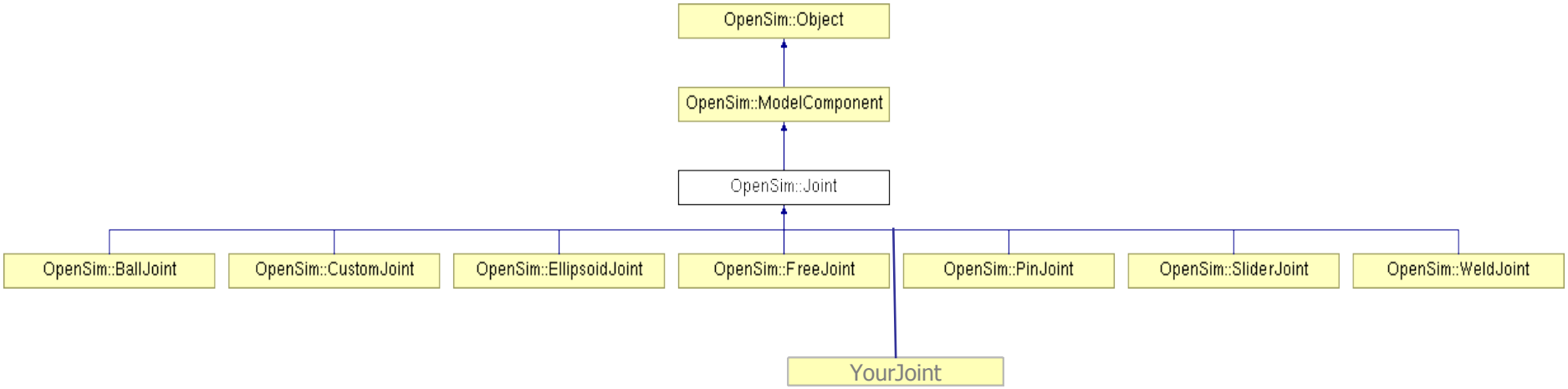
The classes that implement model components in OpenSim all derive from a single base ModelComponent class: implements **`createSystem`**()

ModelComponent is an OpenSim::Object, which supports reading and writing object attributes to/from XML file.

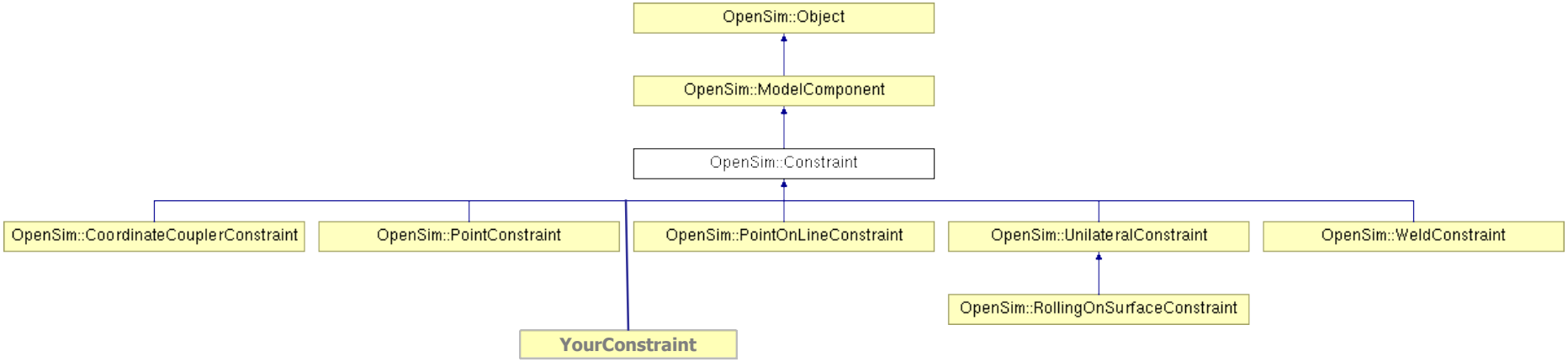
Object Hierarchy and Class Definitions available as Doxygen documentation:

<OpenSim2.0_Install>\sdk\doc\html\index.html

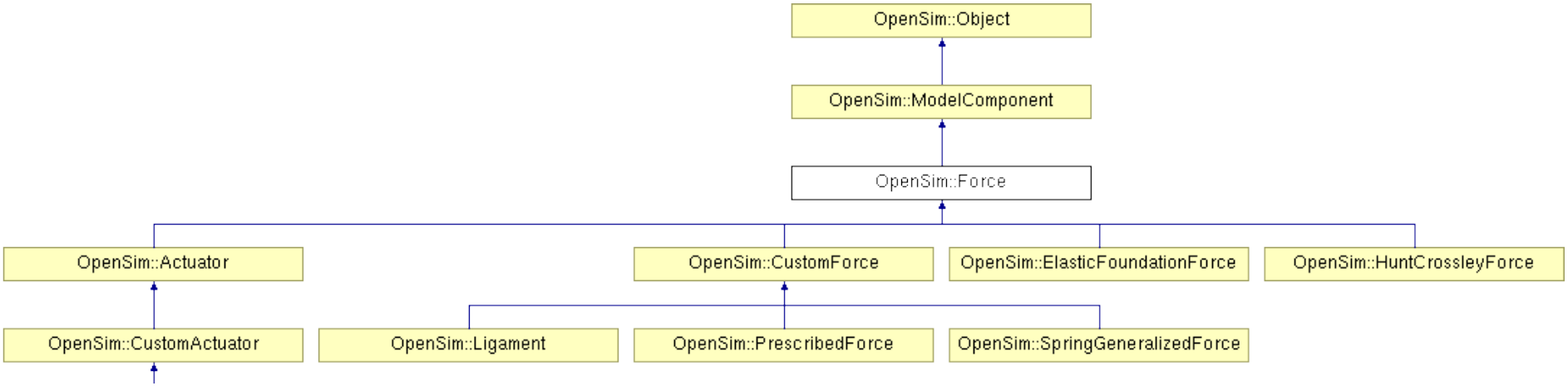
Specific Joint Behavior by Subclasses



Specific Constraint Behavior by Subclasses



Specific Force Behavior by Subclasses



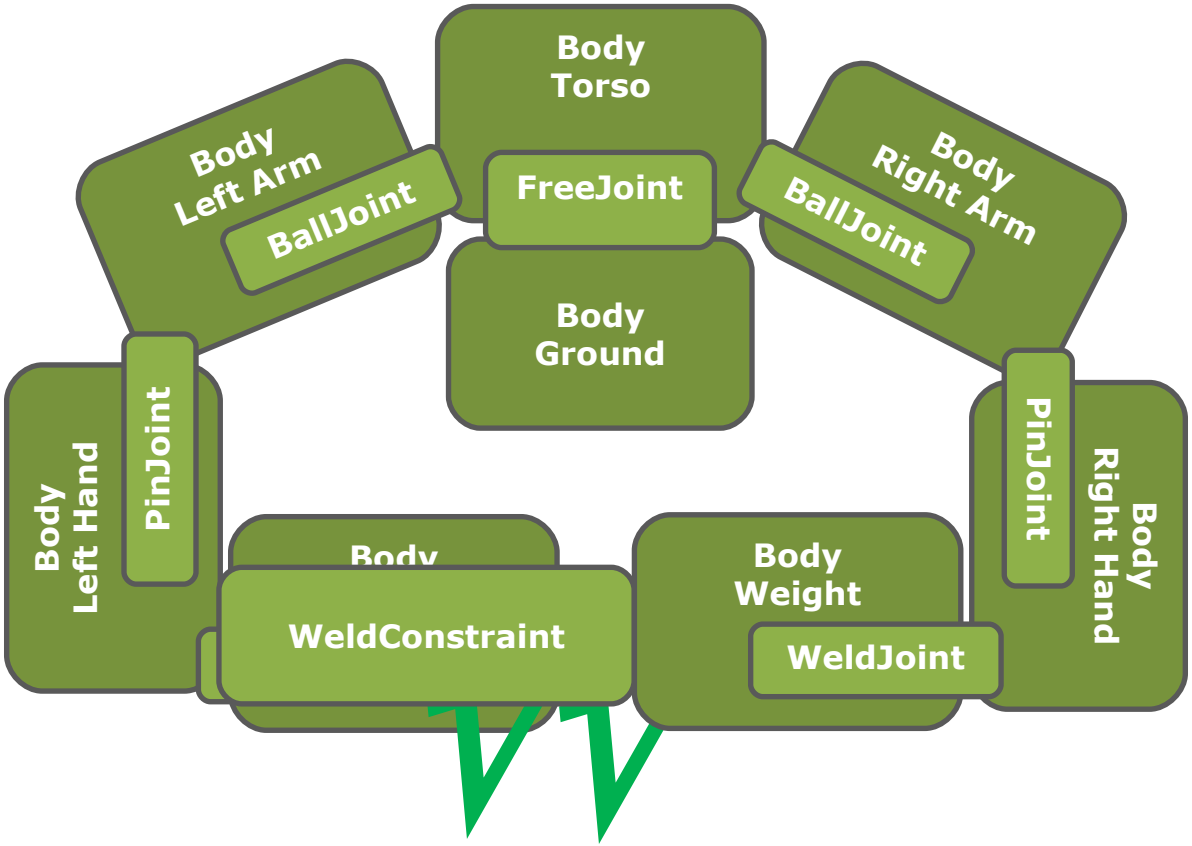
A force can be dependent on and add state values to the system state.

An actuator is a force that is also dependent on control(s).

Controllers provide control values to actuators.

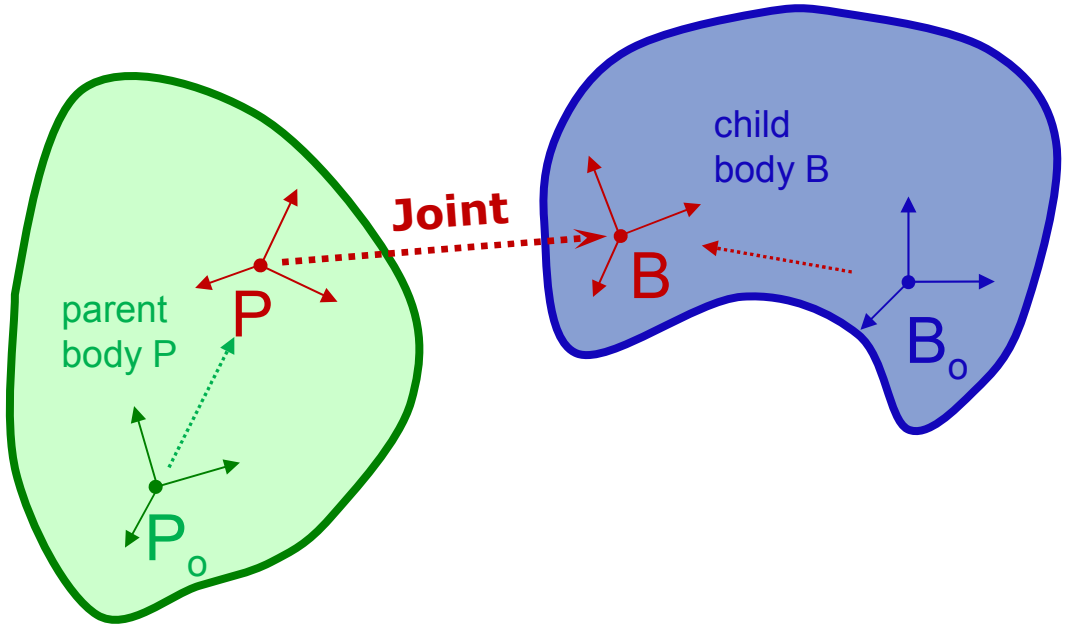
Tree Topology of Multibody Models

Each body is connected by ONE joint to create a chain or open tree structure.



Constraint is required to form a closed loop

Joint Reference Frames



B specified by joint `location` and `orientation`

P specified by joint `locationInParent` and `orientationInParent`

Joint coordinates specify the kinematics of B relative to P

Model components in the upcoming exercise: