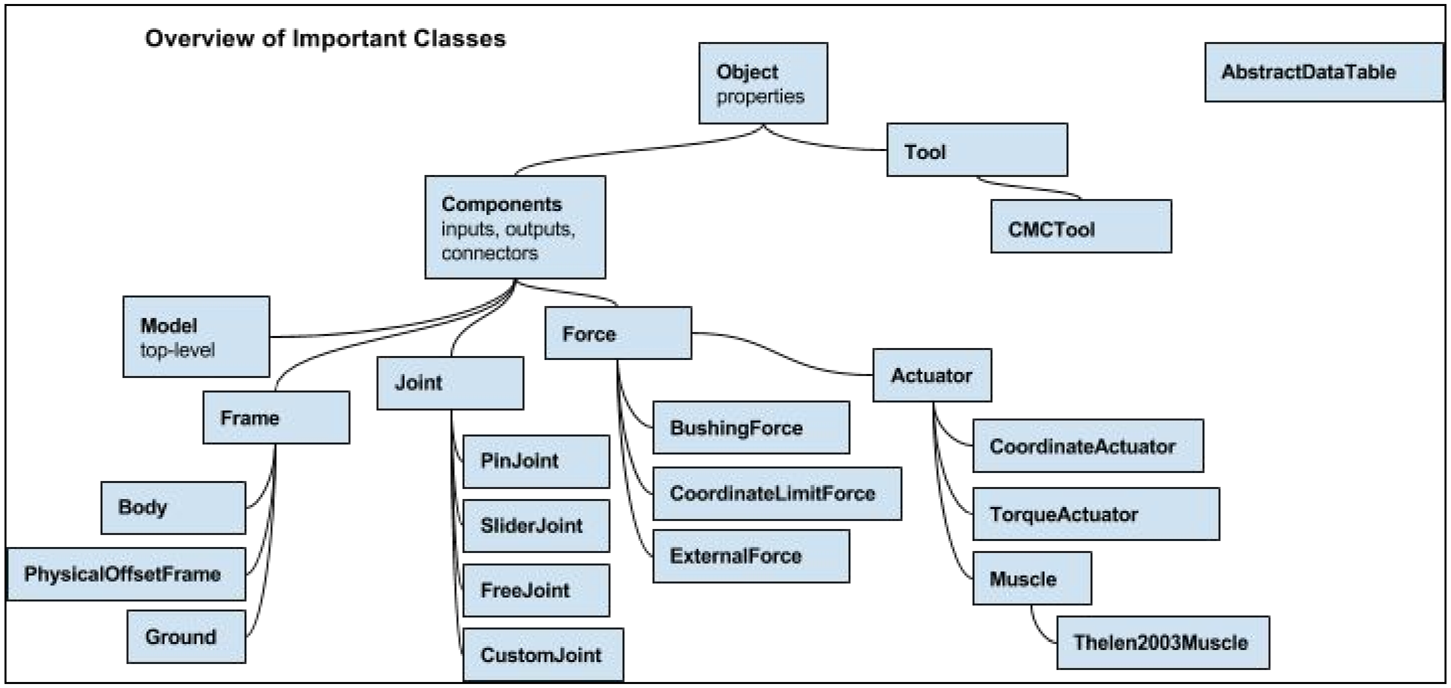
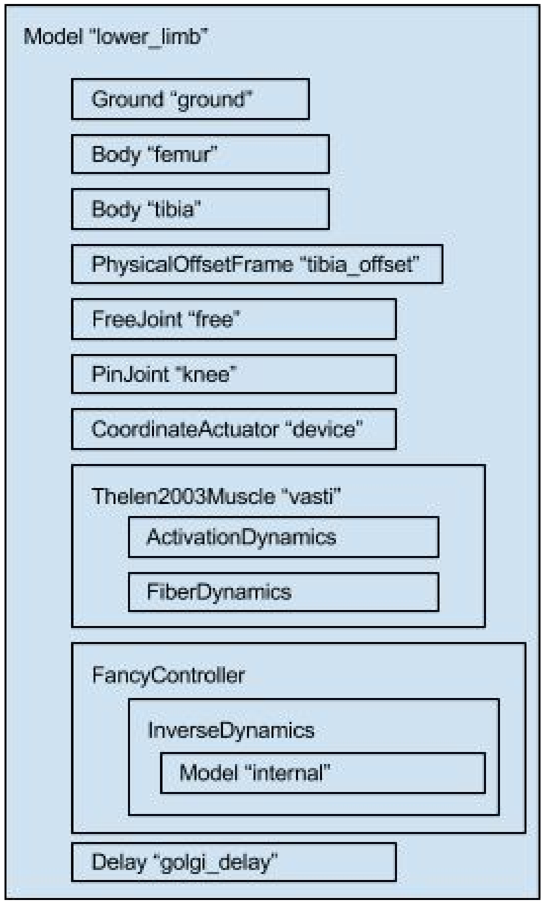
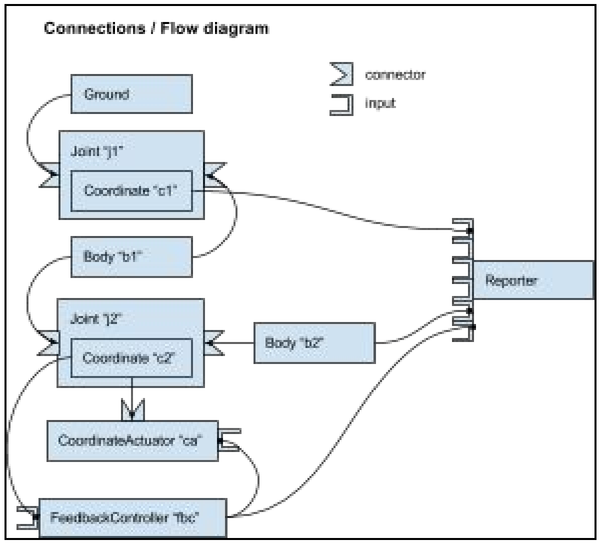
Components have 4 configurable aspects

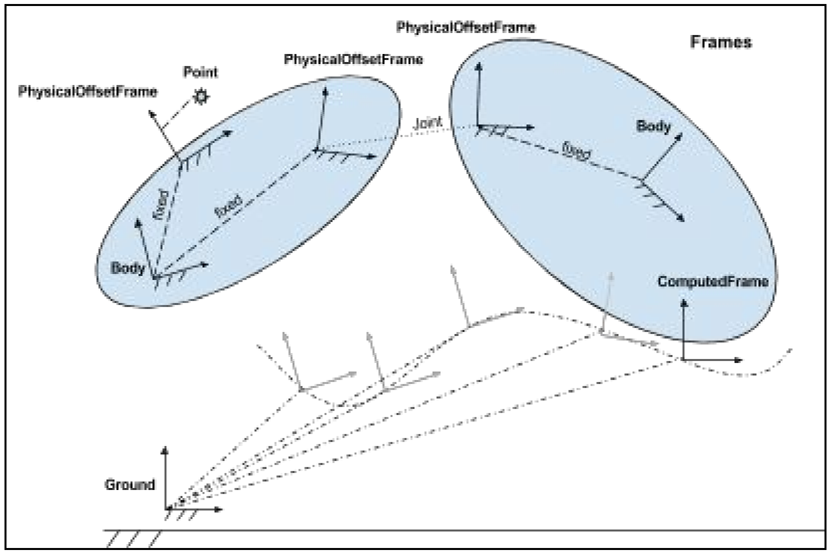
- **Properties**: numbers, flags, mass properties, material properties
- **Inputs**: numerical quantities that components need to perform calculations (Metabolic calculator needs muscle activation).
- **Outputs**: quantities/computations that a component can provide; can be fed to other components as inputs, and saved to a file.
- **Connectors**: other Components that a component depends on (Joints have Connectors to Bodies).



Composition (ownership)







Writing your own Component (extending OpenSim C++ API)

Properties blah blah

Outputs blah blah

Inputs blah blah

Connectors blah blah

Perform calculations at a given stage (e.g., Velocity)

Add Simbody resources to the underlying Simbody System (mobilized bodies, measures, cache variables, etc.)

This is called whenever properties have changed; if you have member variables that depend on properties, update those variables here.

```
class MyComponent : pubic Component {
OpenSim_DECLARE_CONCRETE_OBJECT(MyComponent, Component);
public:

    OpenSim_DECLARE_PROPERTY(...);
    OpenSim_DECLARE_OPTIONAL_PROPERTY(...);
    OpenSim_DECLARE_LIST_PROPERTY(...);
    OpenSim_DECLARE_LIST_PROPERTY_ATLEAST(...);
    OpenSim_DECLARE_LIST_PROPERTY_ATMOST(...);
    OpenSim_DECLARE_LIST_PROPERTY_SIZE(...);

    OpenSim_DECLARE_OUTPUT(...);

    OpenSim_DECLARE_INPUT(...);

    OpenSim_DECLARE_CONNECTOR(...);

protected:

    void extendRealizeVelocity(...) {
        Super::extendRealizeVelocity(s);
        // ** add your code here **
    }

    void extendAddToSystem(...) {
        Super::extendAddToSystem(system);
        m_cacheIndex = addCacheVariable(...);
    }

    void extendFinalizeFromProperties(...) {
    }
private:
    CacheVariableIndex m_cacheIndex;

};
```

XML format (Model files, setup files, etc.)

annotations....

```
<?xml version="1.0" encoding="UTF-8" ?>
<OpenSimDocument Version="30503">
    <Model name="default">
    <!--The model's ground reference frame.-->
        <Ground name="ground">
            <geometry>
                <FrameGeometry name="frame_geometry">
                </FrameGeometry>
            </geometry>
        </Ground>
        <FrameSet>
        </FrameSet>
        <BodySet>
        </BodySet>
        <ForceSet>
        </ForceSet>
    </Model>
</OpenSimDocument>
```

Starter code: pendulum

constructors that take a string are loading the object from an XML file

to add components to the model, create a new instance and call the appropriate `add()` method, which adopts the component.

causes, in order:
1. finalizeFromProperties
2. connectToModel
3. addToSystem

C++

```
using namespace OpenSim;
int main() {

Model model("empty_model.osim");

auto* b1 = new Body("b1", ...);

auto* j1 = new PinJoint(...);

auto* a1 = new CoordinateActuator
("coord0");
a1->setName("motor");

auto* c1 = new PrescribedController();
c1->addActuator(a1);
c1->prescribeControlForActuator("motor",
       new StepFunction(...));

model.addBody(b1);
model.addJoint(j1);
model.addForce(a1);
model.print("pendulum.osim");

State& state = model.initSystem();

Manager manager(model);
manager.integrate();
}
```

MATLAB/Python

```
import org.opensim.modeling.* % MATLAB
from opensim import * # python

model = Model("gait10dof18musc.osim")

b1 = Body("b1", ...)

j1 = PinJoint(...)


a1 = CoordinateActuator("coord0")
a1.setName("motor")

c1 = PrescribedController()
c1.addActuator(a1)
c1.prescribeControlForActuator("motor",
       StepFunction(...))

model.addBody(b1)
model.addJoint(j1)
model.addForce(a1)
model.printToXML("pendulum.osim");

state = model.initSystem()

manager = Manager(model)
manager.integrate()
```