UBC PARAMETRIC HUMAN PROJECT

Autodesk Research NSERC CRSNG FARO TECHNOLOGIES

ArtiSynth Tutorial

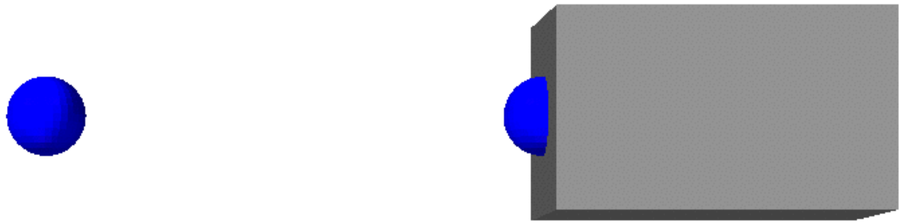
Model Interaction

Ian Stavness

Simple Muscle

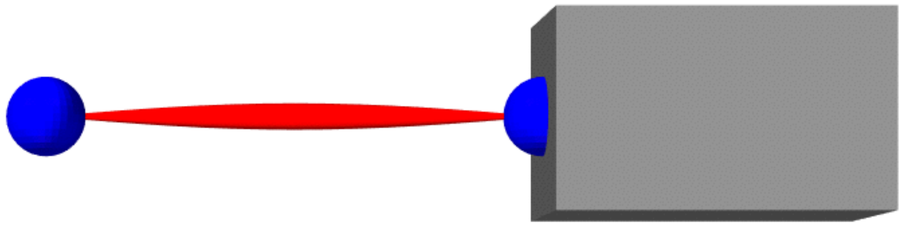
```
public class SimpleMuscle extends RootModel
{
   public void build (String[] args) throws IOException {

      mech = new MechModel ("mech");
      addModel (mech);
```

Simple Muscle



```
// create a fixed particle:
p1 = new Particle ("p1", /*mass=*/2, 0, 0, 0);
p1.setDynamic (false); // set to be fixed

// create box and set its pose:
box = RigidBody.createBox ("box", 0.5, 0.3, 0.3,
                                  /*density=*/20);
box.setPose (new RigidTransform3d (1.00, 0, 0));

// create marker point and connect it to the box:
FrameMarker mkr = new FrameMarker (-0.25, 0, 0);
mkr.setFrame (box);
```

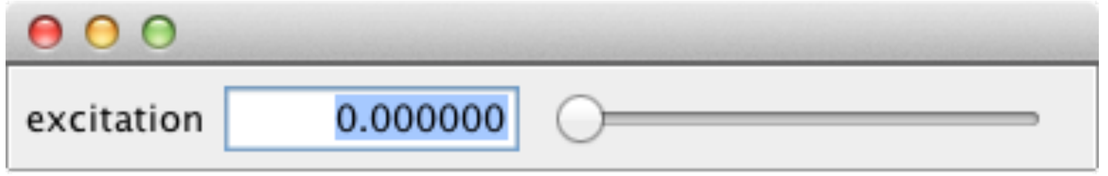
Simple Muscle



```
// create the muscle:
muscle = new Muscle ("mus", /*restLength=*/0);
muscle.setPoints (p1, mkr);
muscle.setMaterial (
    new SimpleAxialMuscle (
      /*stiffness=*/20, /*damping=*/10, /*maxf=*/10));

// add components to the mech model
mech.addParticle (p1);
mech.addRigidBody (box);
mech.addFrameMarker (mkr);
mech.addAxialSpring (muscle);
```

SimpleMuscle

Control Panel




```
public class SimpleMuscleWithPanel extends SimpleMuscle
{
   ControlPanel panel;

   public void build (String[] args) throws IOException {
     super.build (args);

      // add control panel to control muscle
      panel = new ControlPanel();
      panel.addWidget (muscle, "excitation");
      addControlPanel (panel);
   }
}
```

SimpleMuscleWithPanel

Controller

```
private class PointMover extends ControllerBase {

    Point myPnt;
    Point3d myPos0;

    public PointMover (Point pnt) {
       myPnt = pnt;
       myPos0 = new Point3d (pnt.getPosition());
    }

    public void apply (double t0, double t1) {
       // todo!!
    }
}
```

Controller

```
public void apply (double t0, double t1) {

   double ang = Math.PI*t1/2;

   Point3d pos = new Point3d (myPos0);
   pos.x += 0.5*Math.sin (ang);
   pos.z += 0.5*(1-Math.cos (ang));

   myPnt.setTargetPosition (pos);

}
```

Simple Muscle With Controller

```
public class
SimpleMuscleWithController extends SimpleMuscleWithPanel
{
   public void build (String[] args) throws IOException {

      super.build (args);

      addController (new PointMover (p1));

      mech.setBounds (-1, 0, -1, 1, 0, 1);
   }
}
```

SimpleMuscleWithController

Properties

```
public static PropertyList myProps =
    new PropertyList (
        <ThisClass>.class,
        <SuperClass>.class);
```

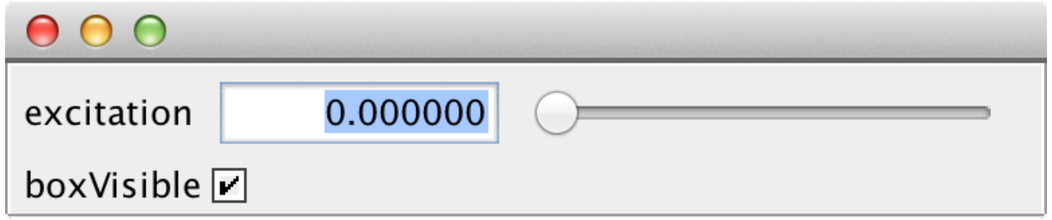
Properties

```
public static PropertyList myProps =
    new PropertyList (
        SimpleMuscleWithProperties.class,
        SimpleMuscleWithPanel.class);

static {
    myProps.add ("boxVisible", "box is visible", false);
}

public PropertyList getAllPropertyInfo() {
    return myProps;
}
```

Properties

```
public boolean getBoxVisible() {
    return box.getRenderProps().isVisible();
}

public void setBoxVisible (boolean visible) {
    RenderProps.setVisible (box, visible);
}
```

Properties

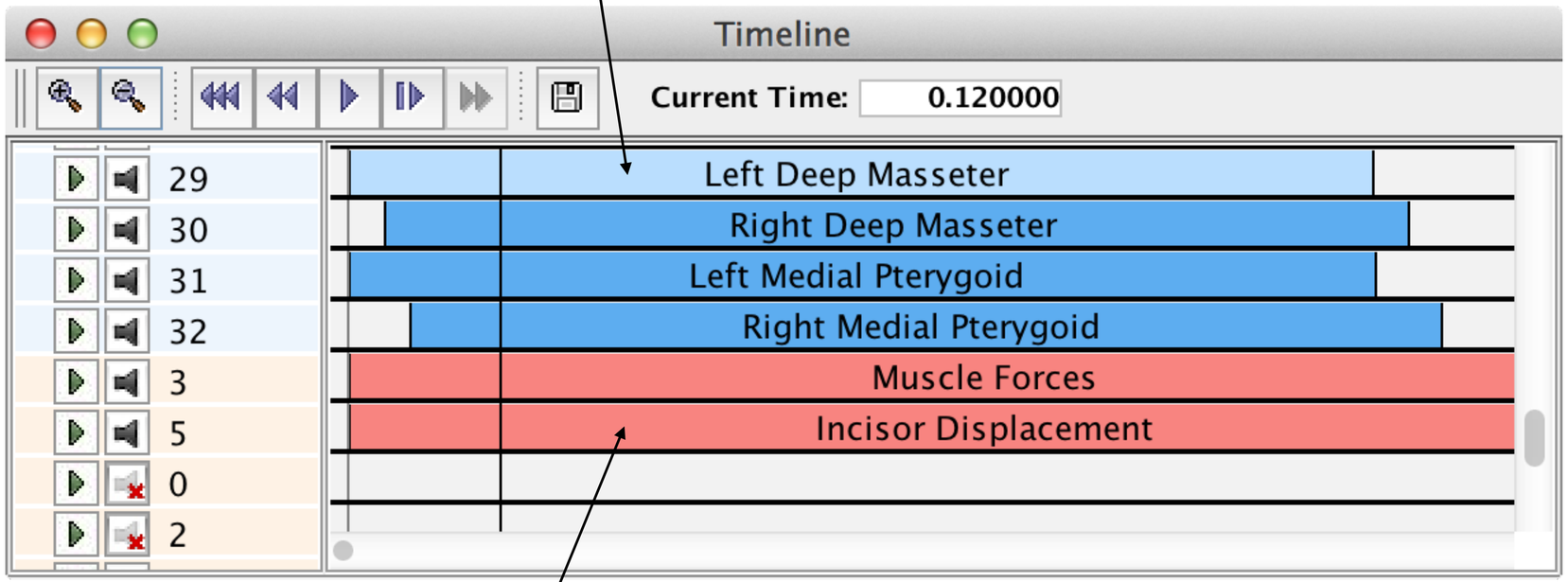



```
public class
SimpleMuscleWithProperties extends SimpleMuscleWithPanel {

   public void build (String[] args) throws IOException {

      super.build (args);

      panel.addWidget (this, "boxVisible");
      panel.pack();
   }

}
```

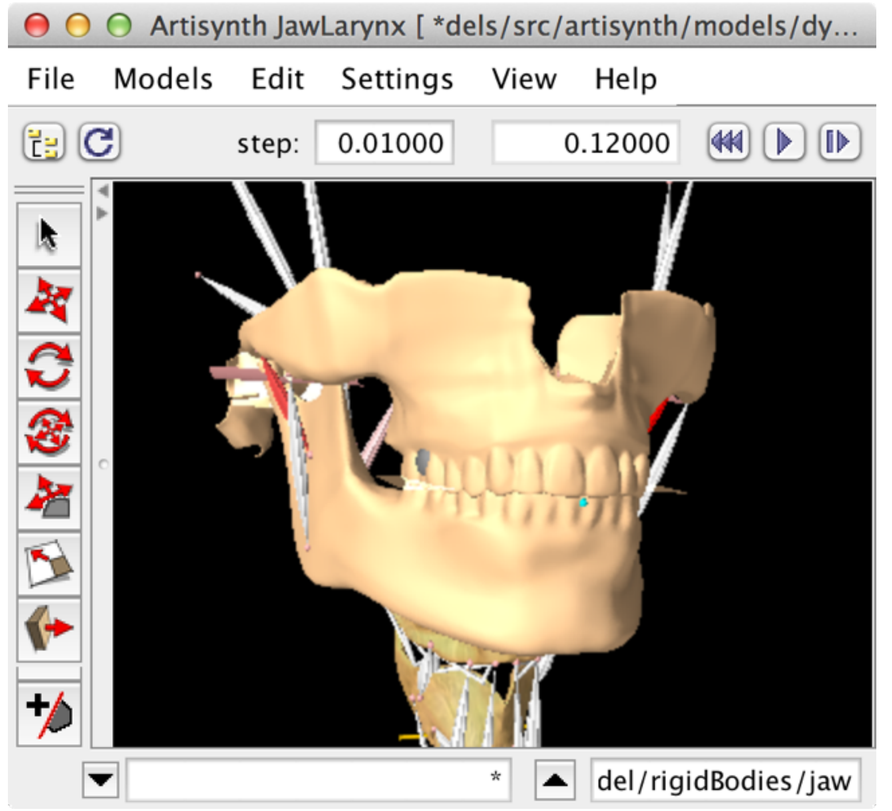
SimpleMuscleWithProperties

Probes

Input Probe



Output Probe



Input Probes

```
NumericInputProbe inputProbe =
    new NumericInputProbe (
        <Component>,
        "<ComponentPath>:<PropertyName>",
        "<DataFileName>");
```

Input Probes

```
public void addInputProbe() throws IOException {

    NumericInputProbe p1probe =
        new NumericInputProbe (
            <Component>,
            "<ComponentPath>:<PropertyName>",
            "<DataFileName>");
```

Input Probes

```
public void addInputProbe() throws IOException {

    NumericInputProbe p1probe =
        new NumericInputProbe (
            mech,
            "<ComponentPath>:<PropertyName>",
            "<DataFileName>");
```

Input Probes

```
public void addInputProbe() throws IOException {

    NumericInputProbe p1probe =
        new NumericInputProbe (
            mech,
            "particles/p1:targetPosition",
            "<DataFileName>");
```

Input Probes

```
public void addInputProbe() throws IOException {

    NumericInputProbe p1probe =
        new NumericInputProbe (
            mech,
            "particles/p1:targetPosition",
            ArtisynthPath.getSrcRelativePath (this,
                "simpleMuscleP1Pos.txt"));
```

Input Probes

```
public void addInputProbe() throws IOException {

    NumericInputProbe p1probe =
        new NumericInputProbe (
            mech,
            "particles/p1:targetPosition",
            ArtisynthPath.getSrcRelativePath (this,
                "simpleMuscleP1Pos.txt"));

    p1probe.setName ("Particle Position”);
    p1probe.setStopTime (5);
    addInputProbe (p1probe);
}
```

Output Probes

```
NumericOutputProbe outProbe =
    new NumericOutputProbe (
        <Component>,
        "<ComponentPath>:<PropertyName>",
        "<DataFileName>",
        updateInterval);
```

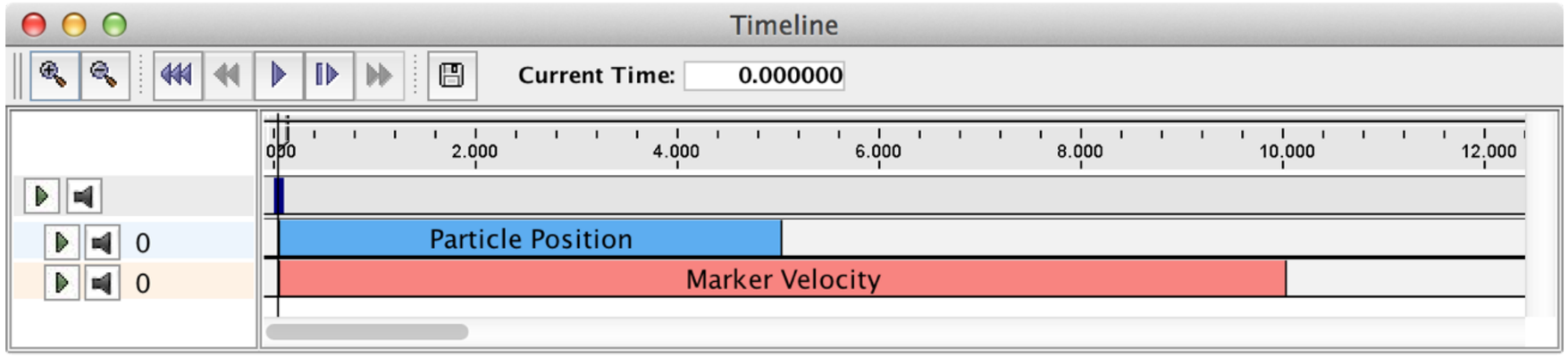
Output Probe

```
public void addOutputProbe() throws IOException {
    NumericOutputProbe mkrProbe =
        new NumericOutputProbe (
            mech,
            "frameMarkers/0:velocity",
            ArtisynthPath.getSrcRelativePath (this,
                  "simpleMuscleMkrVel.txt"),
             0.01);
```

Output Probe

```
public void addOutputProbe() throws IOException {
    NumericOutputProbe mkrProbe =
        new NumericOutputProbe (
            mech,
            "frameMarkers/0:velocity",
            ArtisynthPath.getSrcRelativePath (this,
                "simpleMuscleMkrVel.txt"),
            0.01);

    mkrProbe.setName ("Marker Velocity");
    mkrProbe.setDefaultDisplayRange (-4, 4);
    mkrProbe.setStopTime (10);
    addOutputProbe (mkrProbe);
}
```

Probes



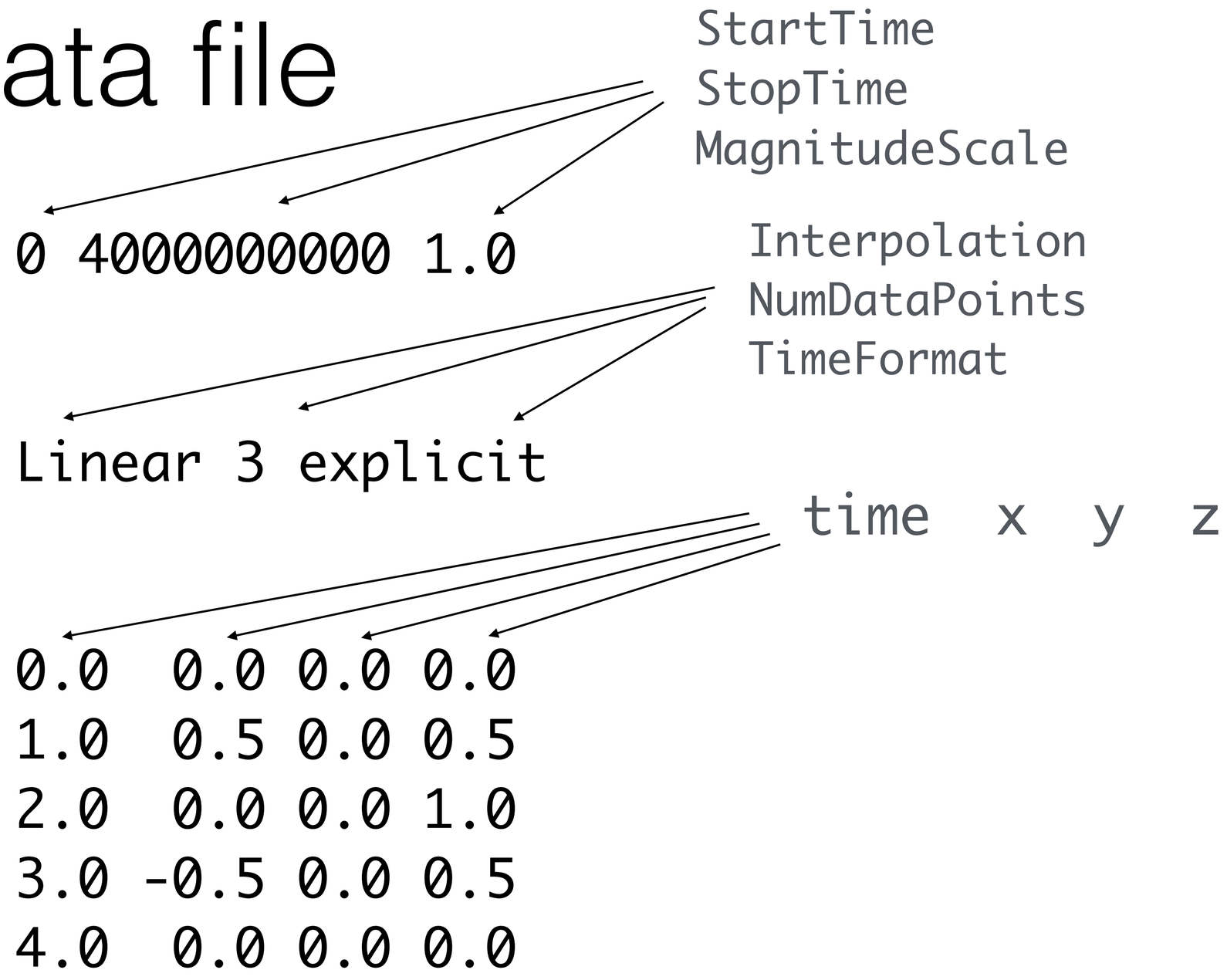

```
public class SimpleMuscleWithProbes extends SimpleMuscleWithPanel
{
  public void build (String[] args) throws IOException {

      super.build (args);

      addInputProbe ();
      addOutputProbe ();

  }
}
```

SimpleMuscleWithProperties

Data file header

```
StartTime       StopTime        MagnitudeScale
Interpolation   NumDataPoints   TimeFormat

Interpolation = {Step, Linear, Cubic}

TimeFormat = {explicit, <number>}
 – if explicit, then first column is time
 – else <number> is time step
```

Data file


StartTime
StopTime
MagnitudeScale
0 4000000000 1.0
Interpolation
NumDataPoints
TimeFormat
Linear 3 explicit
time x y z
0.0 0.0 0.0 0.0
1.0 0.5 0.0 0.5
2.0 0.0 0.0 1.0
3.0 -0.5 0.0 0.5
4.0 0.0 0.0 0.0

Creating data in code

```
p1probe.addData (
    new double[] {
        0.0,  0.0, 0.0, 0.0,
        1.0,  0.5, 0.0, 0.5,
        2.0,  0.0, 0.0, 1.0,
        3.0, -0.5, 0.0, 0.5,
        4.0,  0.0, 0.0, 0.0 },
        NumericInputProbe.EXPLICIT_TIME);
```

ArtiSynth Tutorial

Model Interaction