

Planar Mechanics

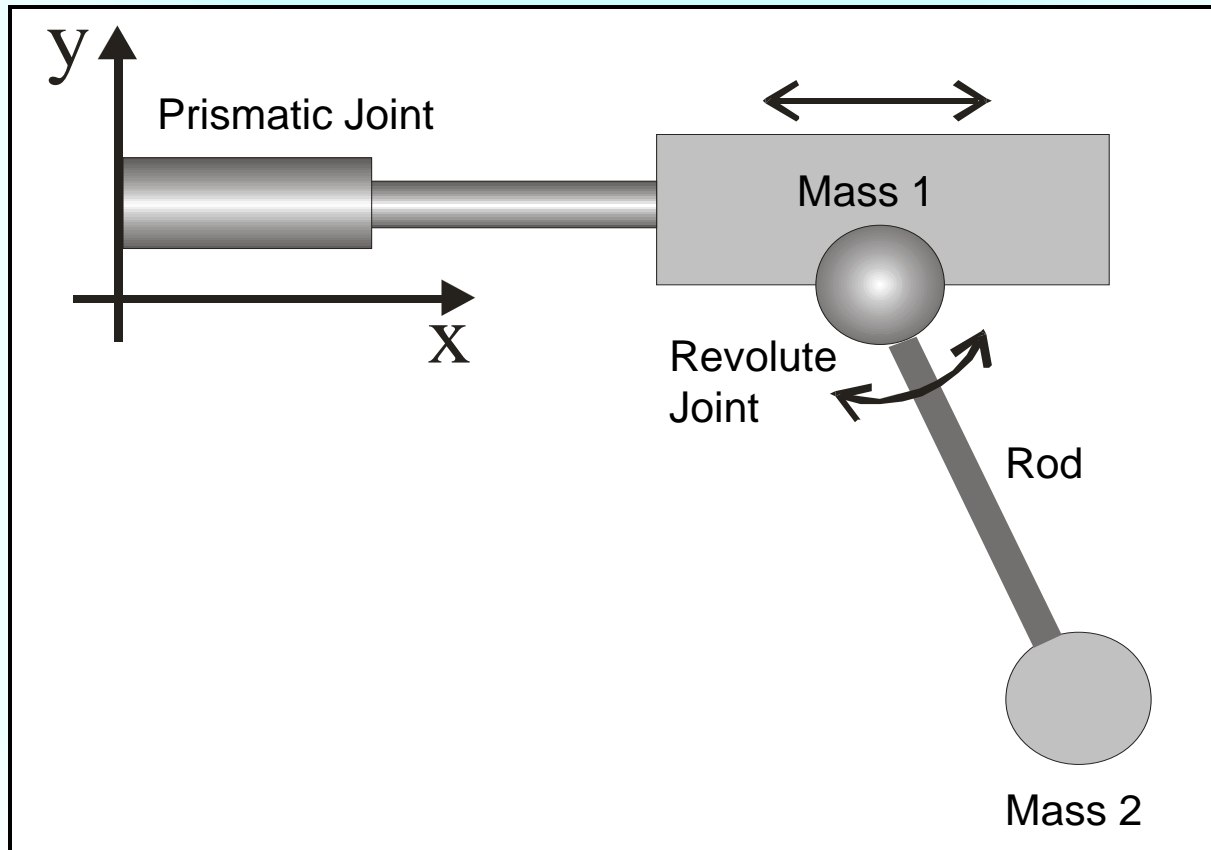
- We shall now look at a first application of multi-bond graphs: *planar mechanics*.
- We shall notice that a mechanical model composed of multi-bond graphs grows quickly in size and becomes poorly readable.
- For this reason, it is important to wrap multi-bond graph models of mechanical components in a framework that is more suitable to a modular description of mechanical systems.

Table of Contents

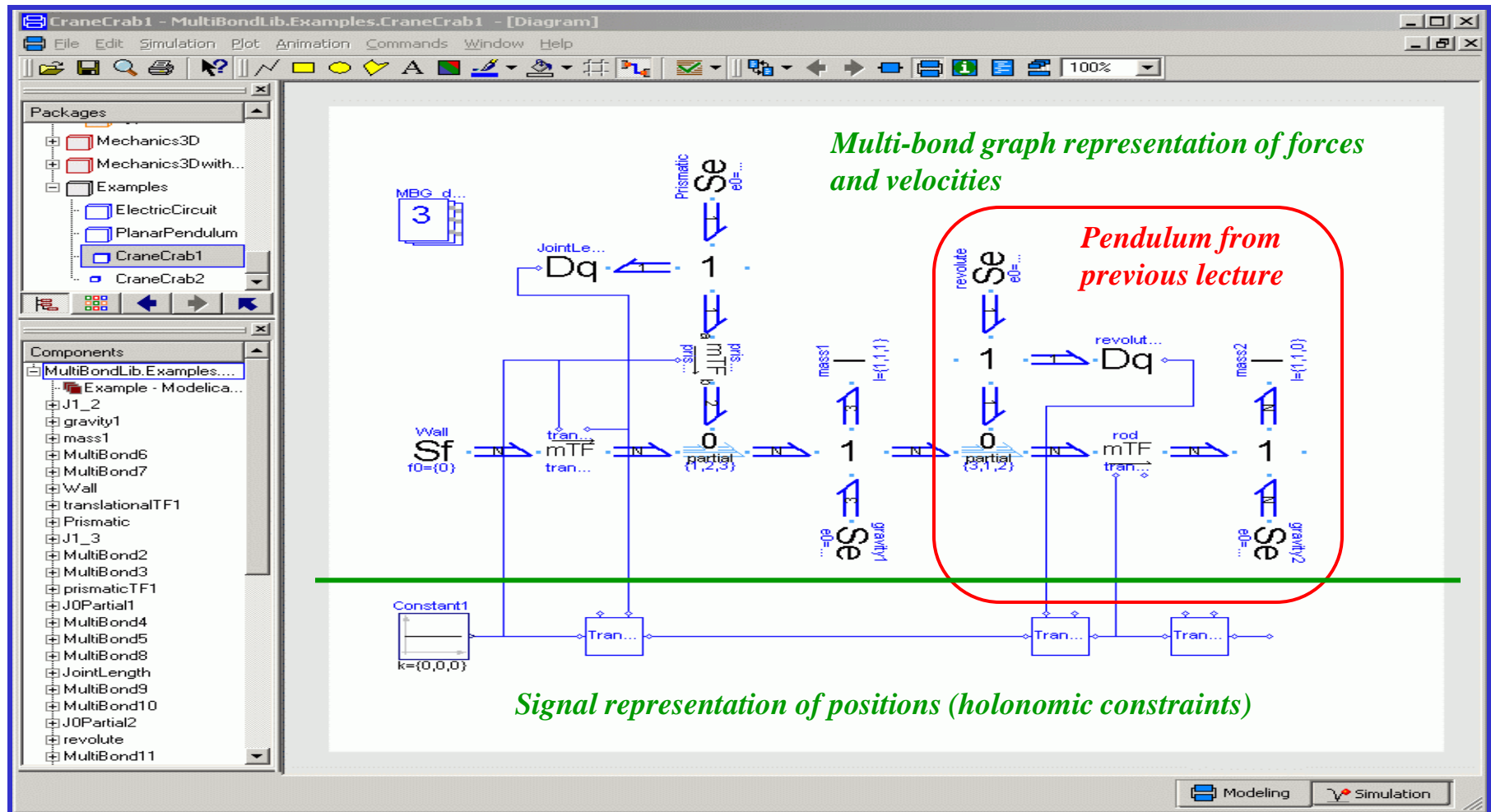
- Crane crab
- Mechanical connectors
- Revolute joints
- Rationale for multi-bond graphs
- Animation
- Wrapper models
- Position translation model
- Planar world model

A Crane Crab

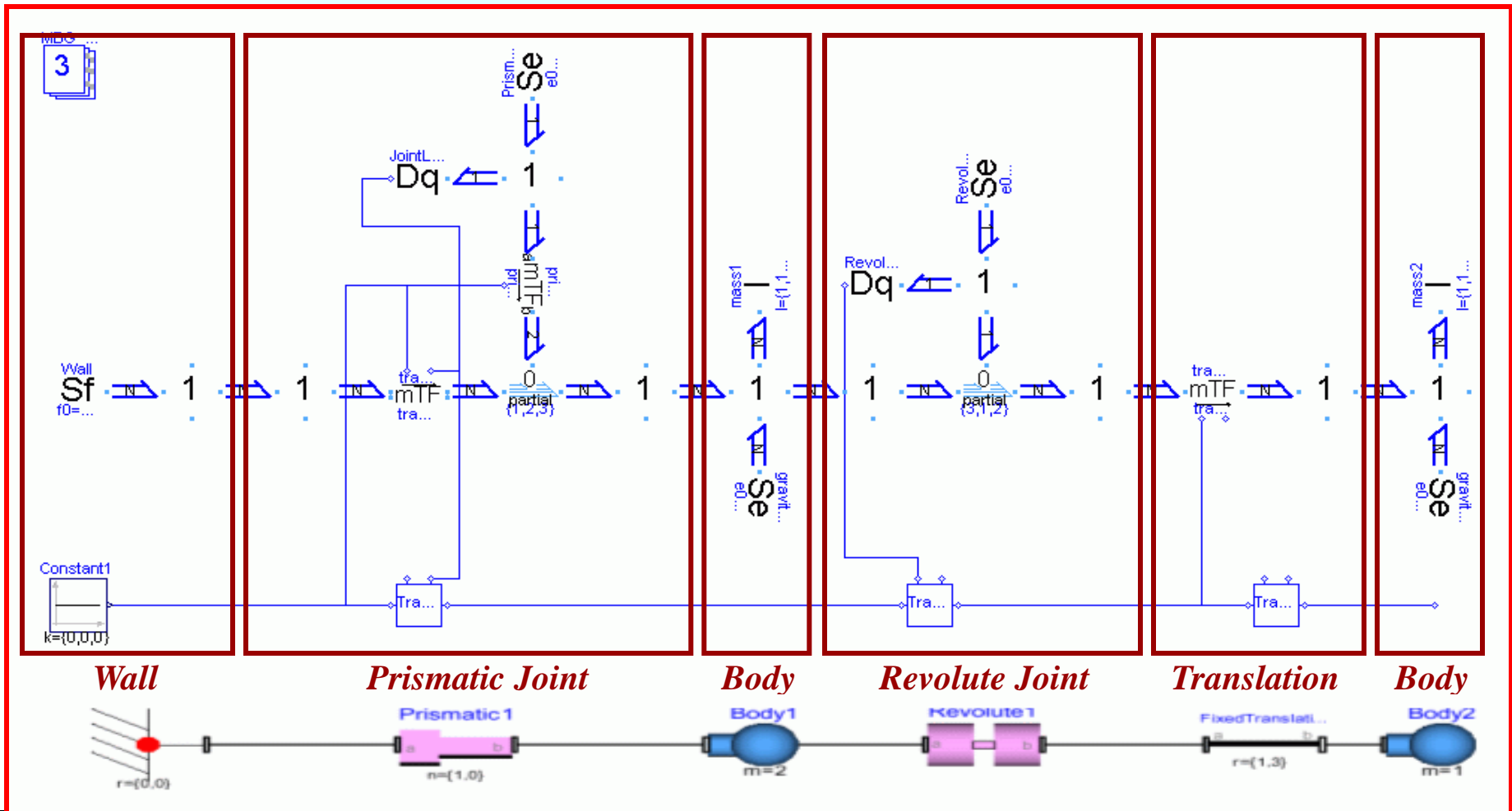
- Let us start by modeling the following crane crab:



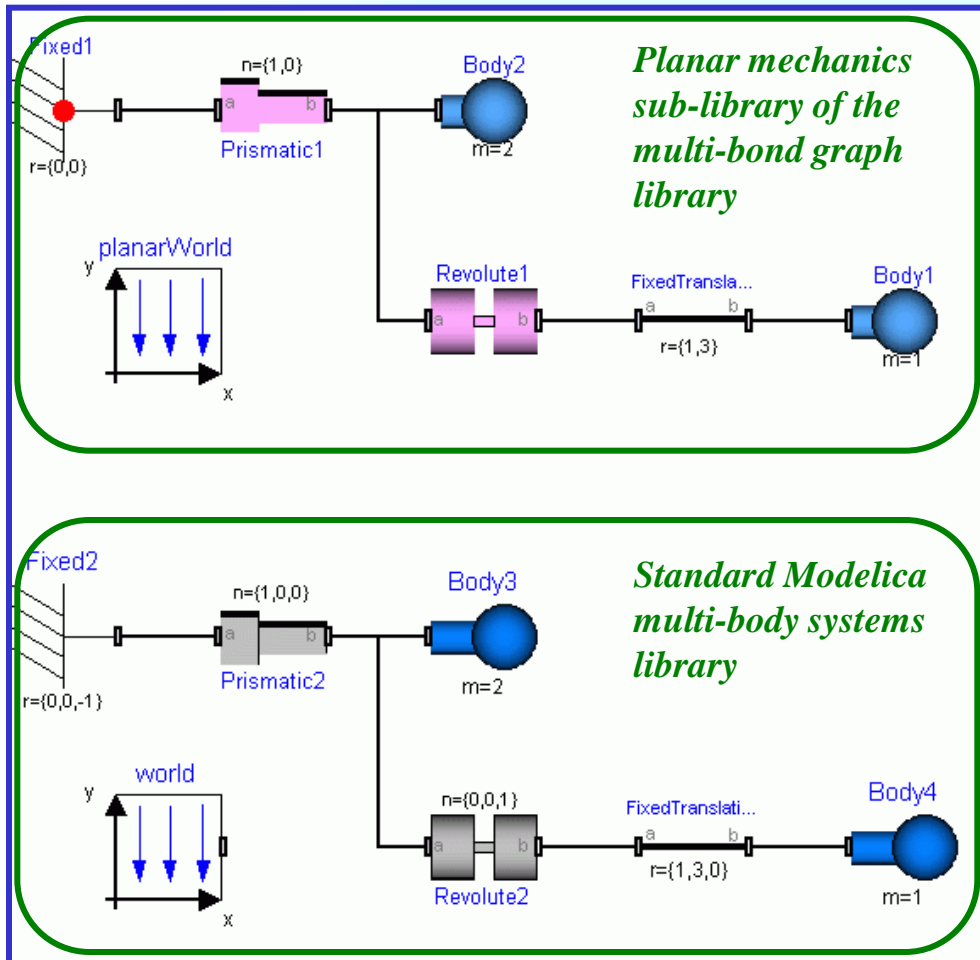
A Crane Crab II



A Crane Crab III



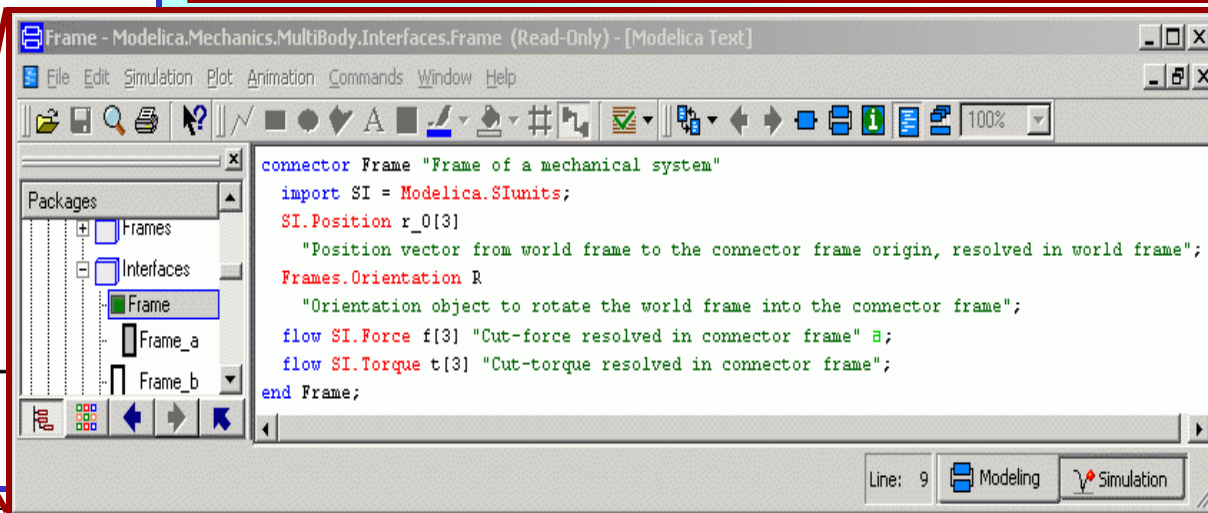
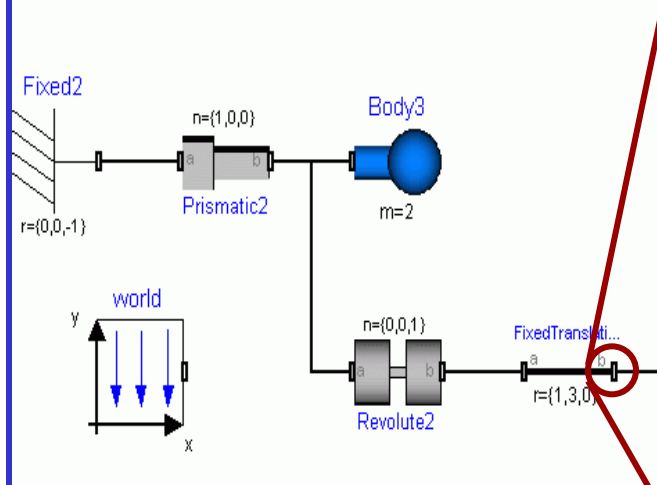
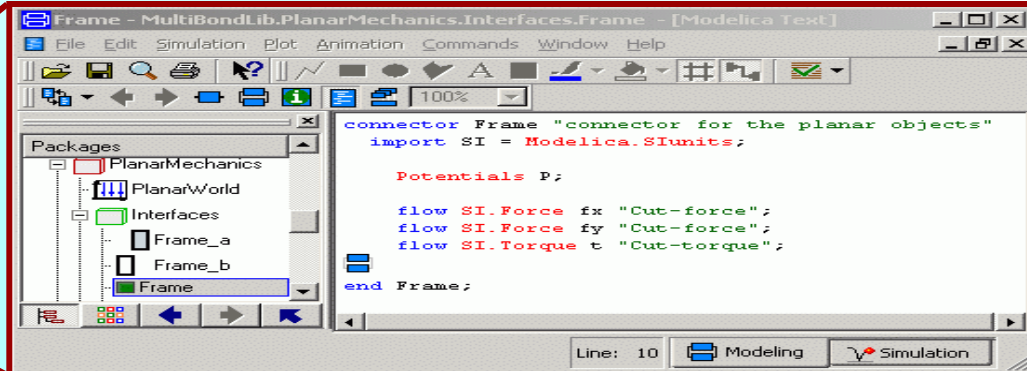
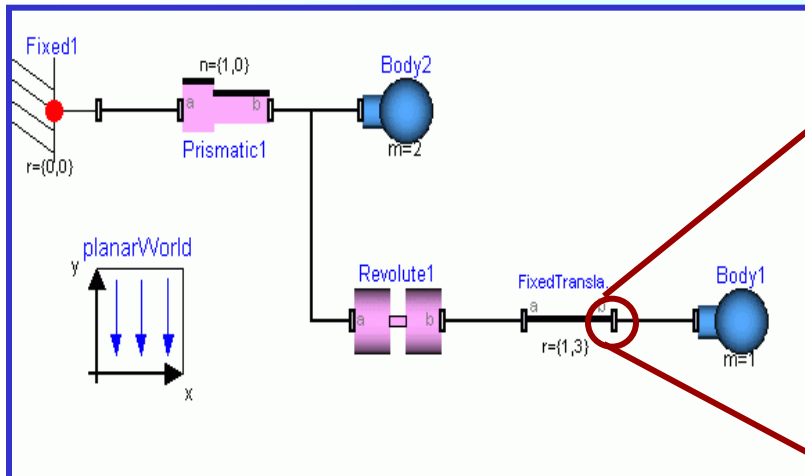
A Crane Crab IV



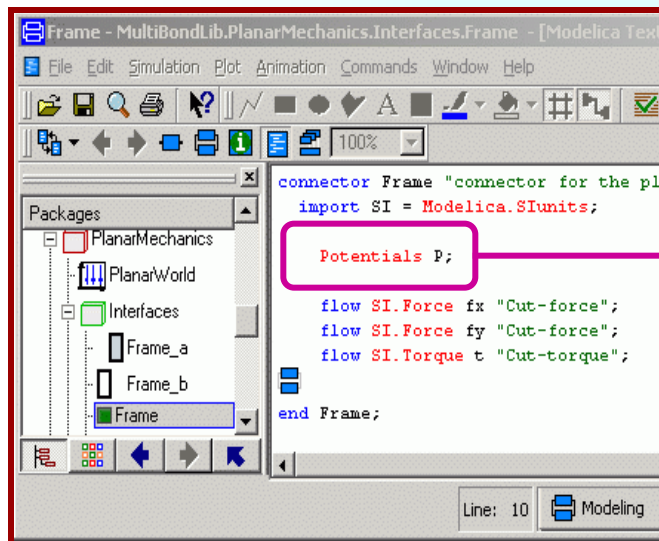
- The standard Modelica multi-body systems library is a general-purpose 3D mechanics library. No separate support for planar mechanics is currently being offered.
- The multi-bond graph library contains separate sub-libraries for planar mechanics and 3D mechanics, as well as for modeling hard collisions between mechanical bodies and for modeling gravitational pools (celestial mechanics).

Mechanical Connectors (Frames)

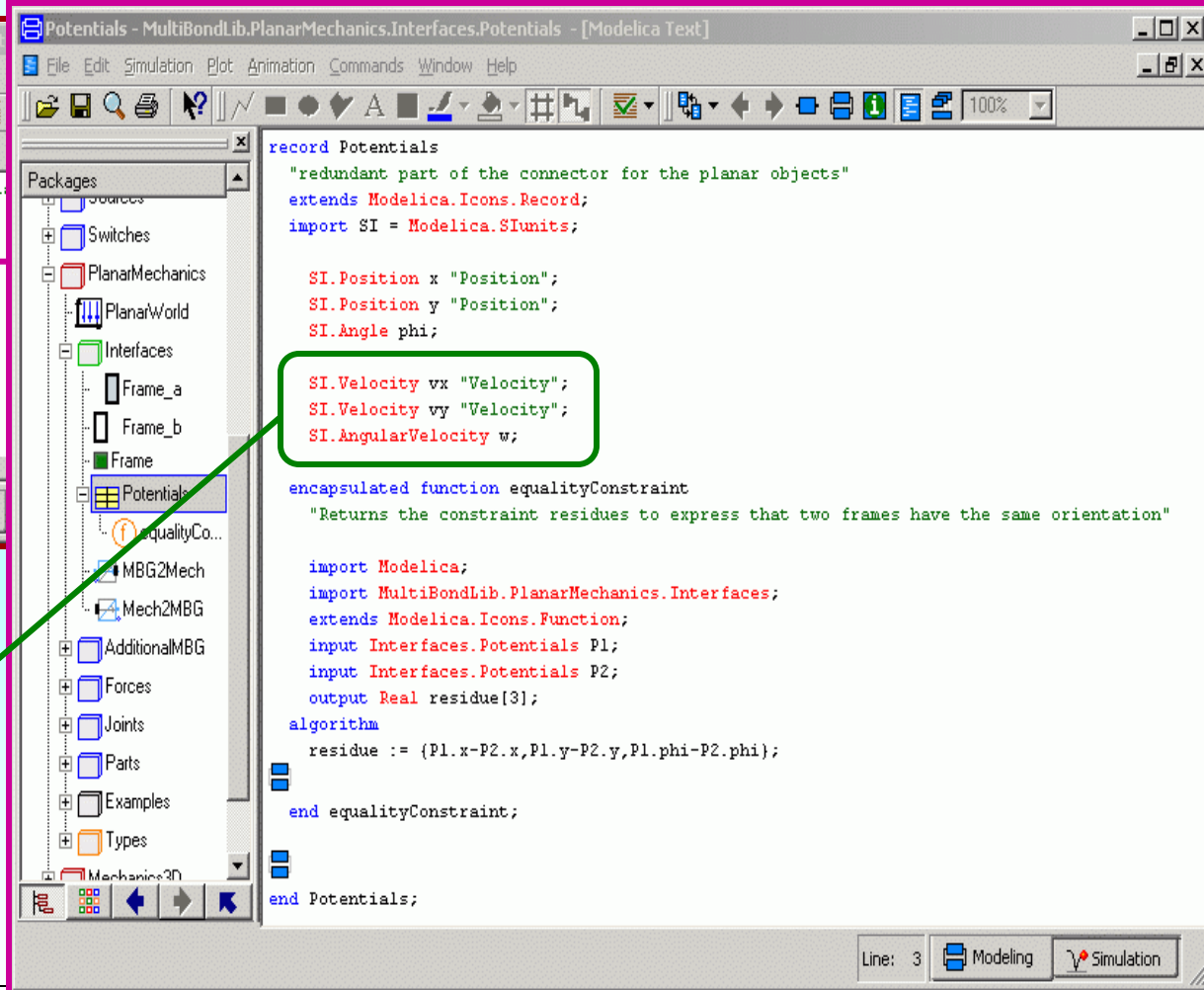
Although the connectors look the same, they are not compatible.



Mechanical Connectors (Frames) II



*Redundant connections
needed because of the
bond graph approach.*



*The component models of the standard
multi-body systems library and the
wrapped multi-bond graph library
cannot be mixed.*

Revolute Joints

The diagram illustrates a mechanical system with two revolute joints. The top part shows a system with a fixed point (Fixed1) connected to a prismatic joint (Prismatic1) with $n=(1,0)$, which is connected to a revolute joint (Revolute1) with $m=2$. The bottom part shows a similar system with a fixed point (Fixed2) connected to a prismatic joint (Prismatic2) with $n=(1,0,0)$, which is connected to a revolute joint (Revolute2) with $n=(0,0,1)$. Both systems are shown in a 2D coordinate system (x, y) with a planarWorld or world frame.

The top Modelica window shows the internal structure of the Revolute joint, including the definition of the revolute joint and the associated equations. The bottom Modelica window shows the internal structure of the Revolute joint, including the definition of the revolute joint and the associated equations.

Revolute - MultiBondLib.PlanarMechanics.Joints.Revolute - [Modelica Text]

```
equation
  defineBranch(frame_a.P, frame_b.P);
  phi = Dq_phi.q[1]-phi_offset_rad;
  w = J1_1.MultiBondConsl.f[1];
  z = der(w);
end Revolute;
```

Revolute - Modelica.Mechanics.MultiBody.Joints.Internal.Revolute (Read-Only) - [Modelica Text]

```
equation
  assert(cardinality(frame_a) > 0,
    "Connector frame_a of revolute joint is not connected");
  assert(cardinality(frame_b) > 0,
    "Connector frame_b of revolute joint is not connected");

  if not planarCutJoint then
    defineBranch(frame_a.R, frame_b.R);

    angle = Cv.from_deg(phi_offset) + phi;
    w = der(phi);
    a = der(w);

    // relationships between quantities of frame_a and of frame_b
    R_rel = Frames.planarRotation(e, angle, der(angle));
    frame_b.r_0 = frame_a.r_0;

    if rooted(frame_a.R) then
```

Revolute Joints II

- Using the multi-bond graph library, almost the entire model of the revolute joint has been coded graphically. There are only very few equations to be coded in the equation window. (There is still quite a bit of code there, because the object is being animated, and *Dymola* doesn't offer graphical support yet for coding animation models.)
- Using the multi-body library of the standard *Modelica* library, the entire revolute joint had to be coded by means of equations, leading to a fairly large equation model that is difficult to understand and even harder to maintain.

Rationale for Multi-bond Graphs

- It is important to keep the distance between the lowermost graphical layer and the equation layer small, such that as few equations as possible need to be maintained in alphanumerical form.
- *Bond graphs and multi-bond graphs provide the most primitive graphical interface that is still fully object-oriented.* Hence, when using bond graphs, the distance between the lowermost graphical layer (the bond graph layer) and the equation layer is minimized.

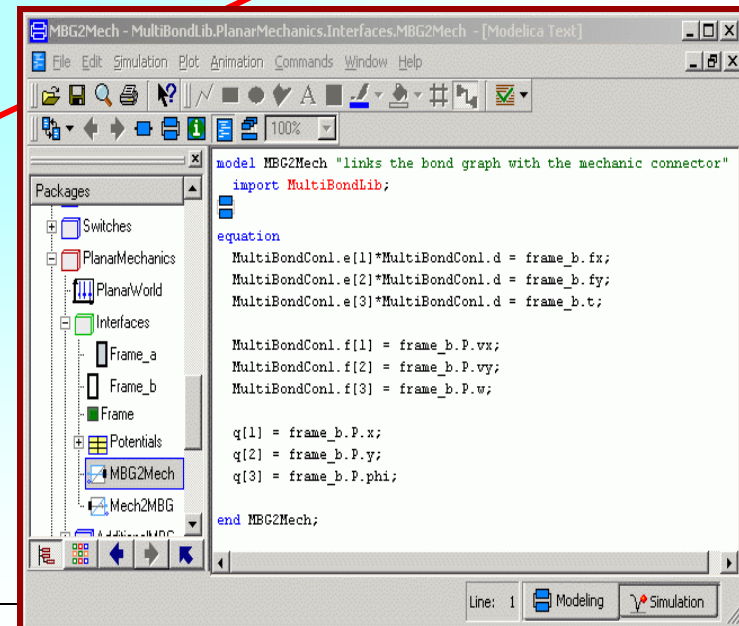
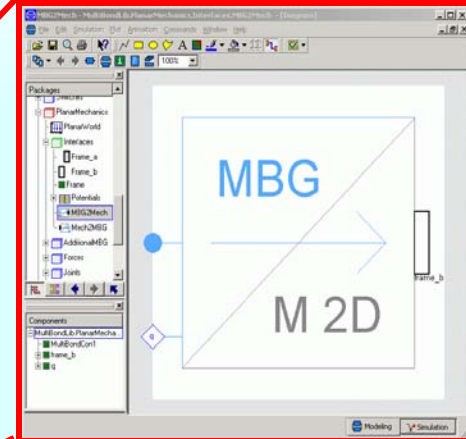
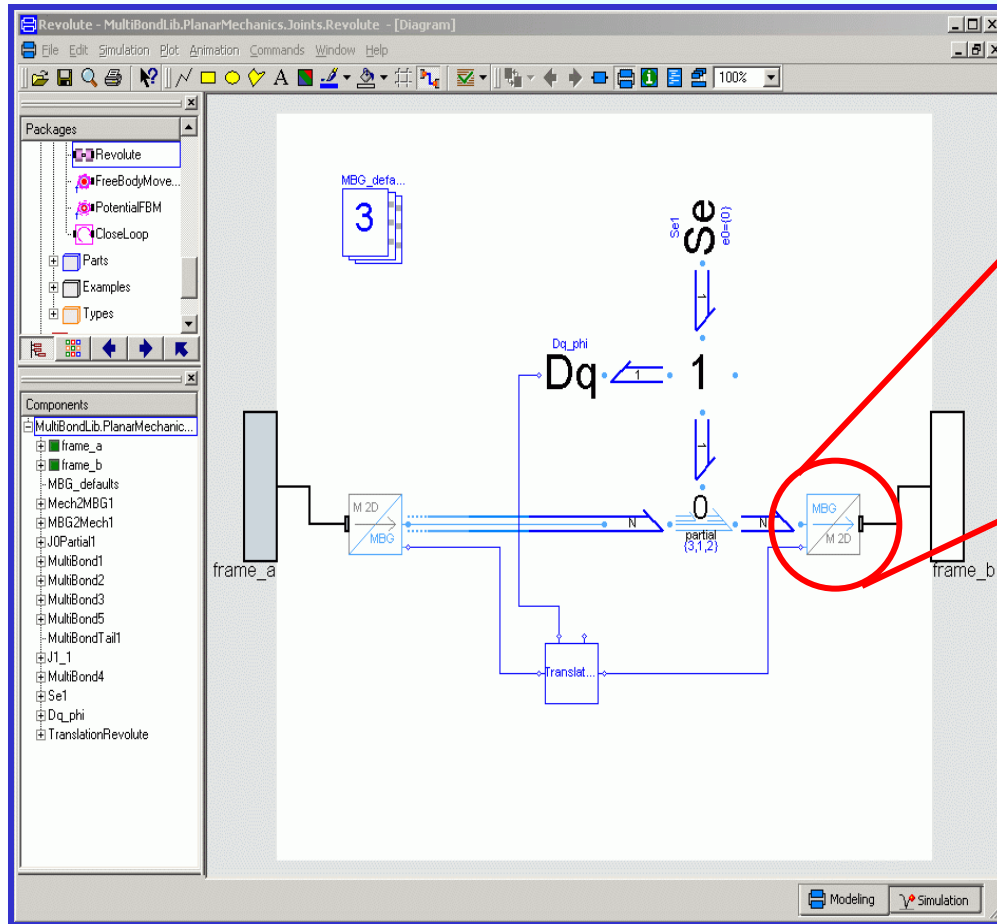
Rationale for Multi-bond Graphs II

- However, this does not imply that bond graphs offer an optimal user interface. For mechanical systems, this is certainly not the case.
- *Wrapping bond graphs* enables the modeler to map *any* graphical object-oriented modeling paradigm onto a lower-level bond graph layer that simplifies the maintenance of the resulting application libraries.

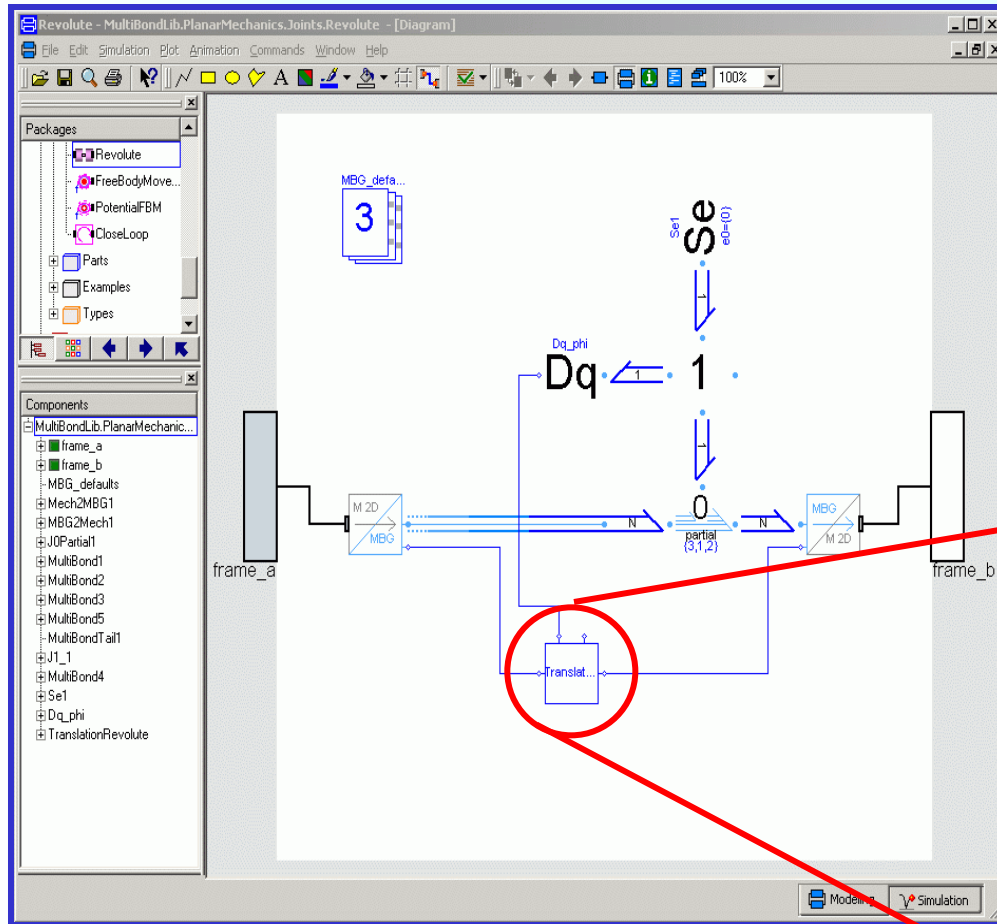
Animation

- In *Dymola*, mechanical models can be *automatically animated*. The end user of the models doesn't need to be concerned about this facet of modeling.
- However, individual bonds cannot be animated. The animation must take place at a higher conceptual level, namely that of multi-body system components, such as masses and joints.
- For this reason, a wrapping of multi-bond graphs is *necessary* if the resulting models are to be animated.

The Wrapper Models



The Position Translation Model

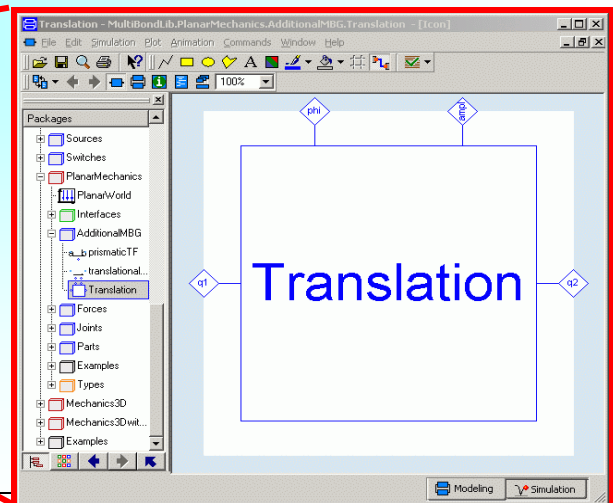


The code defines the `Translation` model, which translates the position. It includes parameters for `d[2]` and `phi`, and equations for the position components `q2[1]`, `q2[2]`, and `q2[3]`.

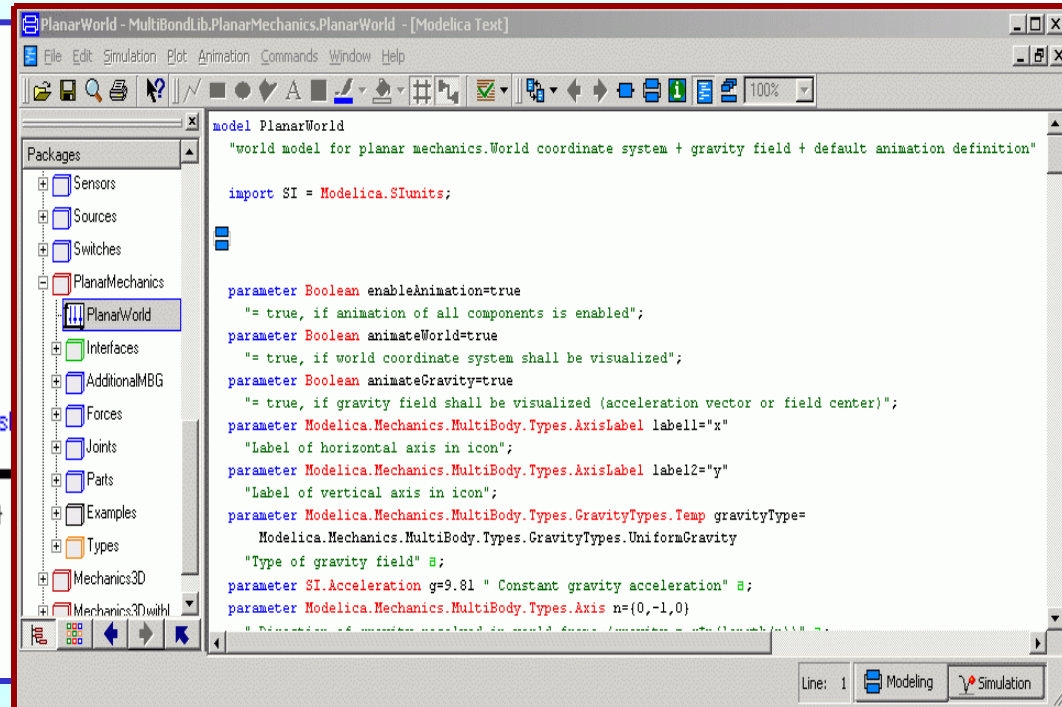
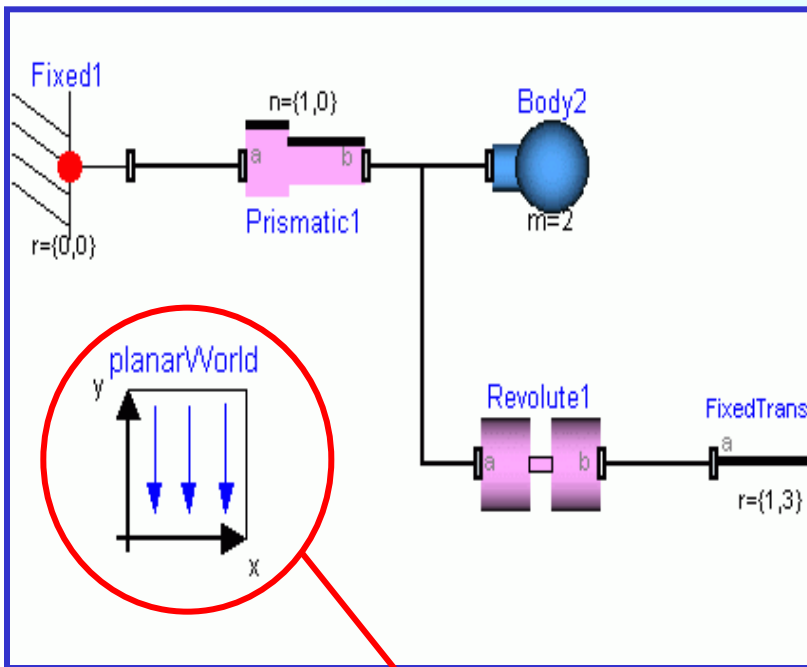
```

model Translation "translates the position"
  parameter Real d[2] = {1,0} ;
equation
  if cardinality(ampl) == 0 then
    ampl = 1;
  end if;
  if cardinality(phi) == 0 then
    phi = 0;
  end if;
  q2[1] = q1[1] + (cos(q1[3])*d[1] + sin(q1[3])*d[2])*ampl;
  q2[2] = q1[2] + (-sin(q1[3])*d[1] + cos(q1[3])*d[2])*ampl;
  q2[3] = q1[3] + phi;
end Translation;

```



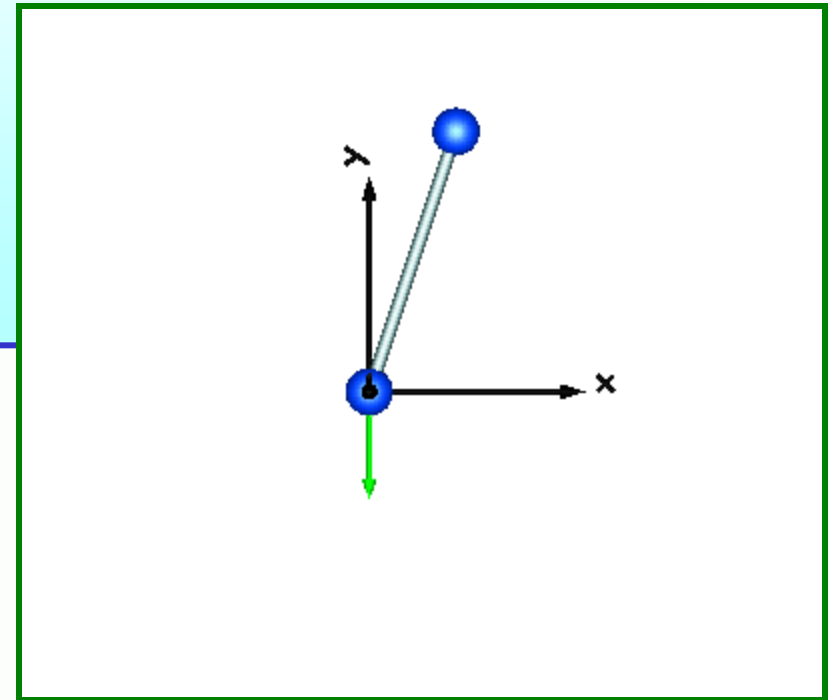
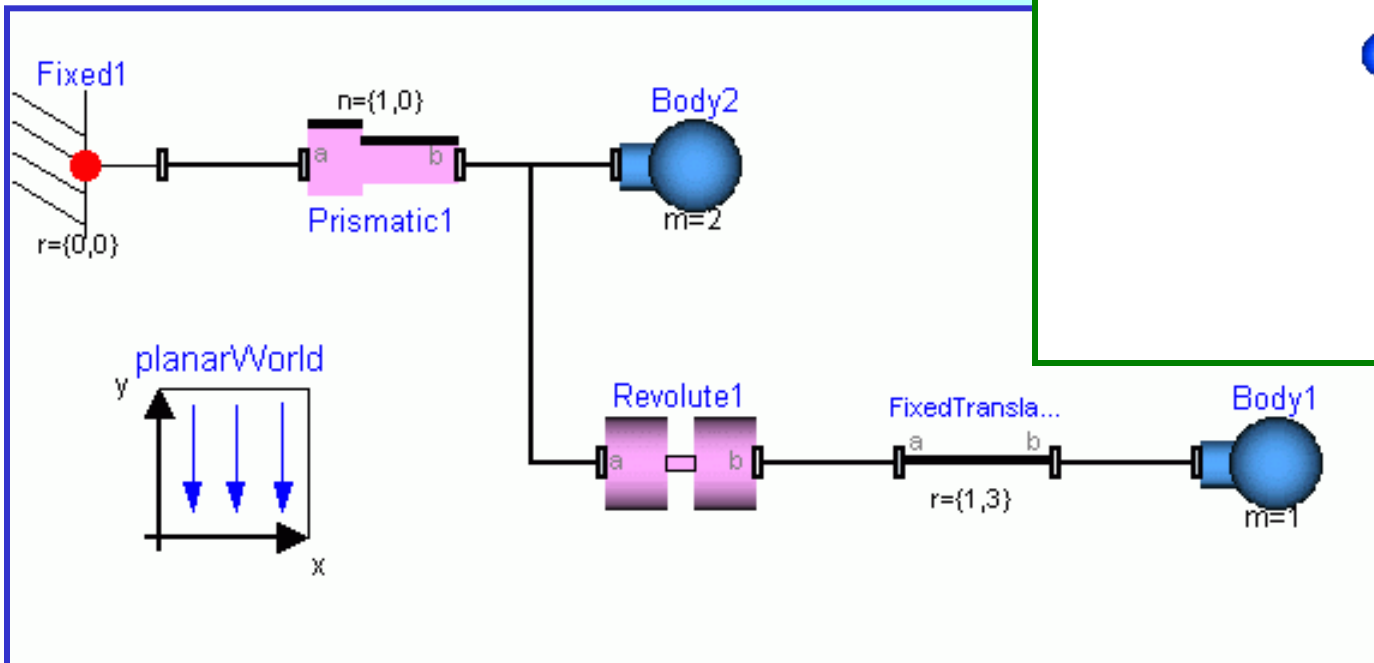
The Planar World Model



Every planar mechanical model must invoke the “planarWorld” model.

The primary purpose of the world model is to set up the animation.

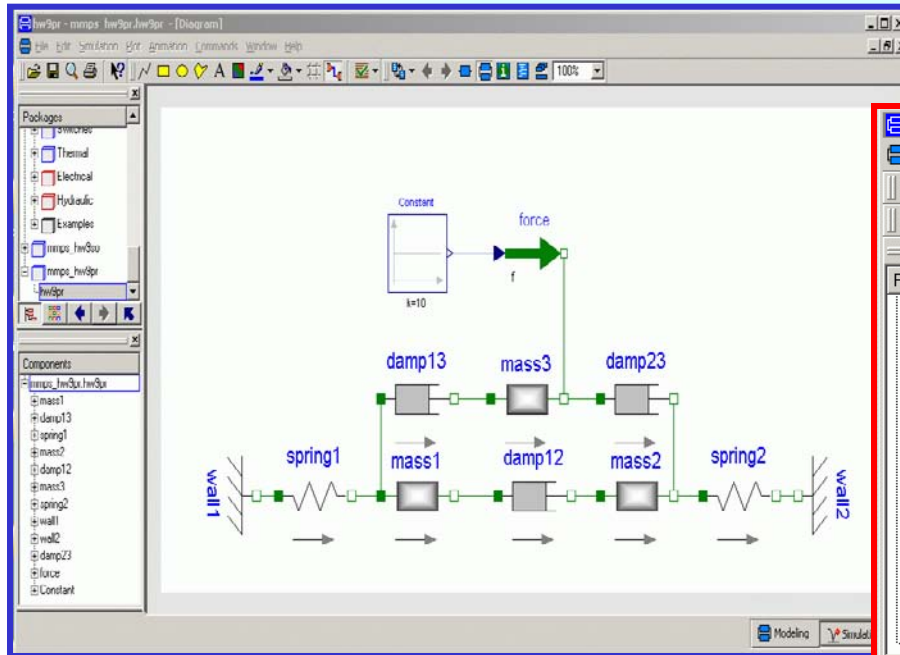
Crane Crab Simulation Results



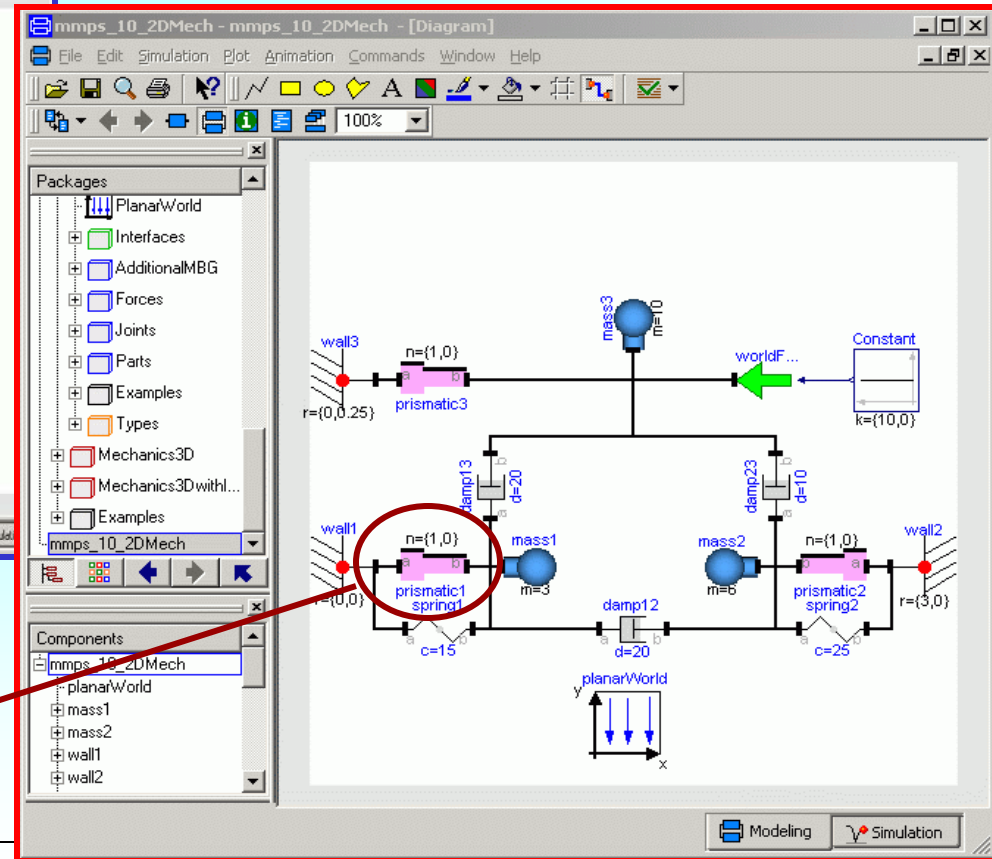
2D Simulation of 1D Models

- It is of course always possible to make use of the planar library also for the simulation of 1D models.
- Let us investigate, what the overhead of such an approach would be.
- To this end, we shall simulate the sliding mass model now using the planar mechanics library.

2D Simulation of 1D Models II



Prismatic joints need to accompany the masses in order to limit their degrees of freedom to one.



Translation Logs

Wrapped 1D mechanical bond graph model

```

Messages - Dymola
Syntax Error  Translation  Dialog Error  Simulation

Translation of mmps\_hw9sol1.mechanical.Translation.Examples.SlidingMasses:
DAE having 685 scalar unknowns and 685 scalar equations.

STATISTICS

Original Model
Number of components: 105
Variables: 706
Constants: 0
Parameters: 59 (59 scalars)
Unknowns: 647 (685 scalars)
Differentiated variables: 6 scalars
Equations: 491
Nontrivial: 250

Translated Model
Constants: 185 scalars
Free parameters: 40 scalars
Parameter depending: 19 scalars
Inputs: 0
Outputs: 0
Continuous time states: 6 scalars
Time-varying variables: 36 scalars
Alias variables: 464 scalars
Assumed default initial conditions: 3
LogDefaultInitialConditions=true; gives more information
Number of mixed real/discrete systems of equations: 0
Sizes of linear systems of equations: {}
Sizes after manipulation of the linear systems: {}
Sizes of nonlinear systems of equations: {}
Sizes after manipulation of the nonlinear systems: {}
Number of numerical Jacobians: 0

Finished
// experiment StopTime=10
Finished
  
```

Wrapped 2D mechanical bond graph model

```

Messages - Dymola
Syntax Error  Translation  Dialog Error  Simulation

Translation of mmps\_10\_2DMech:
DAE having 3705 scalar unknowns and 3705 scalar equations.

STATISTICS

Original Model
Number of components: 268
Variables: 2470
Constants: 0
Parameters: 623 (715 scalars)
Unknowns: 1847 (3705 scalars)
Differentiated variables: 18 scalars
Equations: 1486
Nontrivial: 984

Translated Model
Constants: 1722 scalars
Free parameters: 83 scalars
Parameter depending: 577 scalars
Inputs: 0
Outputs: 0
Continuous time states: 6 scalars
Time-varying variables: 101 scalars
Alias variables: 2045 scalars
Number of mixed real/discrete systems of equations: 0
Sizes of linear systems of equations: {10, 10, 10}
Sizes after manipulation of the linear systems: {0, 2, 0}
Sizes of nonlinear systems of equations: {}
Sizes after manipulation of the nonlinear systems: {}
Number of numerical Jacobians: 0

Finished
// experiment StopTime=10
Finished
  
```

Simulation Logs

```
Messages - Dymola
Syntax Error Translation Dialog Error Simulation
Log-file of program ./dymosim
(generated: Tue Dec 12 18:45:32 2006)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "SlidingMasses.mat" creating (simulation result file)

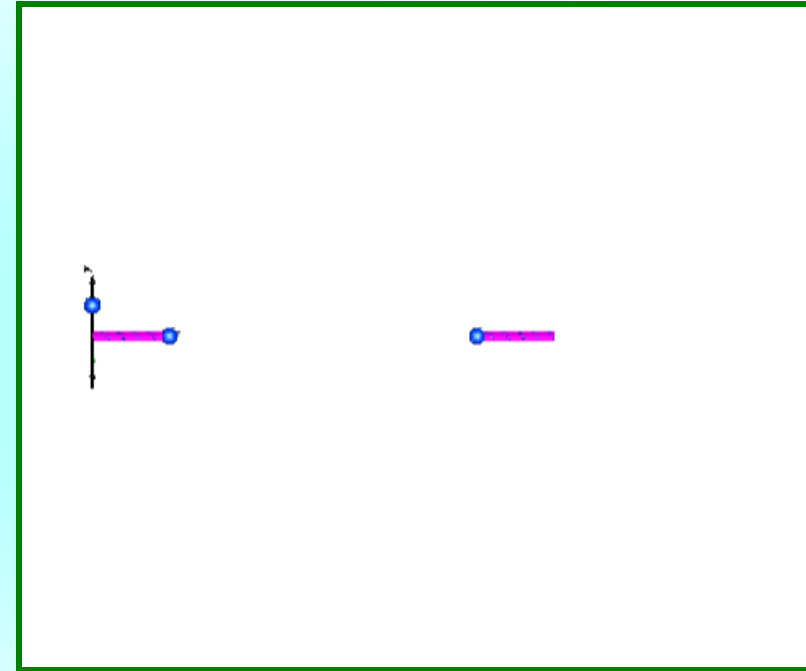
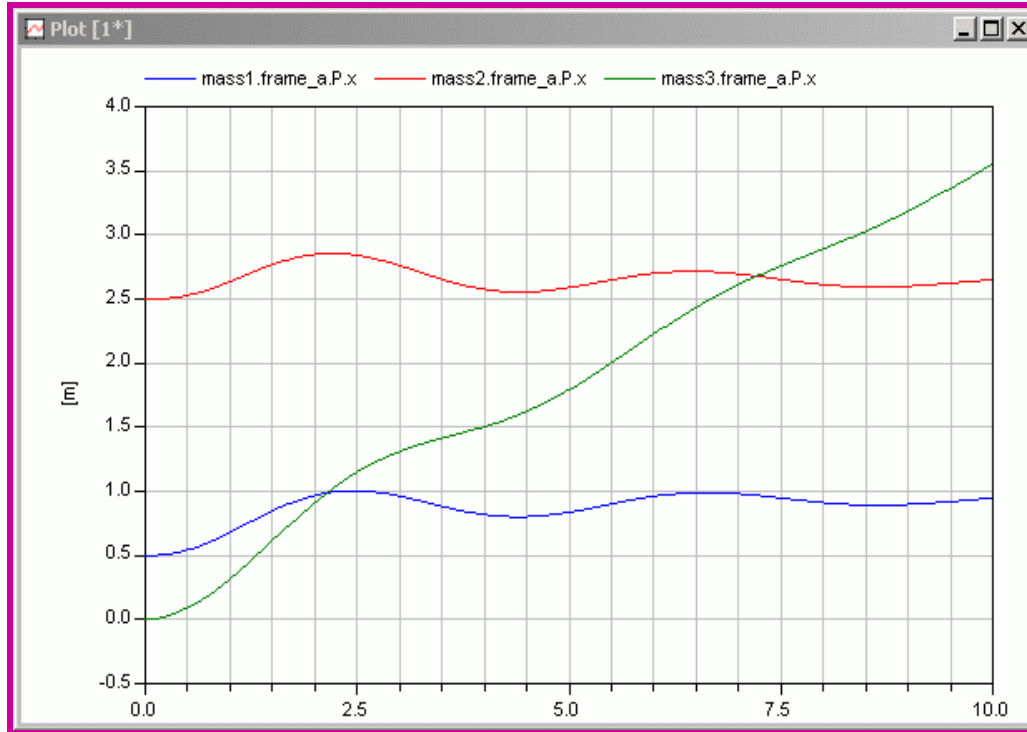
Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 10
CPU-time for integration      : 0.01 seconds
CPU-time for one GRID interval: 0.02 milli-seconds
Number of result points      : 501
Number of GRID points       : 501
Number of (successful) steps : 75
Number of F-evaluations      : 237
Number of Jacobian-evaluations: 15
Number of (model) time events: 0
Number of (U) time events    : 0
Number of state events       : 0
Number of step events        : 0
Minimum integration stepsize : 2e-005
Maximum integration stepsize : 0.253
Maximum integration order    : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```

```
Messages - Dymola
Syntax Error Translation Dialog Error Simulation
Log-file of program ./dymosim
(generated: Thu Dec 14 21:56:27 2006)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "mmps_10_2DMech.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 10
CPU-time for integration      : 0.02 seconds
CPU-time for one GRID interval: 0.04 milli-seconds
Number of result points      : 501
Number of GRID points       : 501
Number of (successful) steps : 76
Number of F-evaluations      : 239
Number of H-evaluations      : 576
Number of Jacobian-evaluations: 15
Number of (model) time events: 0
Number of (U) time events    : 0
Number of state events       : 0
Number of step events        : 0
Minimum integration stepsize : 2e-005
Maximum integration stepsize : 0.253
Maximum integration order    : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```

Simulation Results



References I

- Zimmer, D. (2006), *A Modelica Library for MultiBond Graphs and its Application in 3D-Mechanics*, MS Thesis, Dept. of Computer Science, ETH Zurich.
- Zimmer, D. and F.E. Cellier (2006), “The Modelica Multi-bond Graph Library,” *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.2, pp. 559-568.

References II

- Cellier, F.E. and D. Zimmer (2006), “Wrapping Multi-bond Graphs: A Structured Approach to Modeling Complex Multi-body Dynamics,” *Proc. 20th European Conference on Modeling and Simulation*, Bonn, Germany, pp. 7-13.