3D Mechanics

- We shall now look at a second application of multi-bond graphs: ***3D mechanics***.
- 3D mechanical models look superficially just like planar mechanical models. There are additional types of joints, but other than that, there seem to be few surprises.
- Yet, the seemingly similar appearance is deceiving. There are a substantial number of complications that the modeler has to cope with when dealing with 3D mechanics. These are the subject of this lecture.

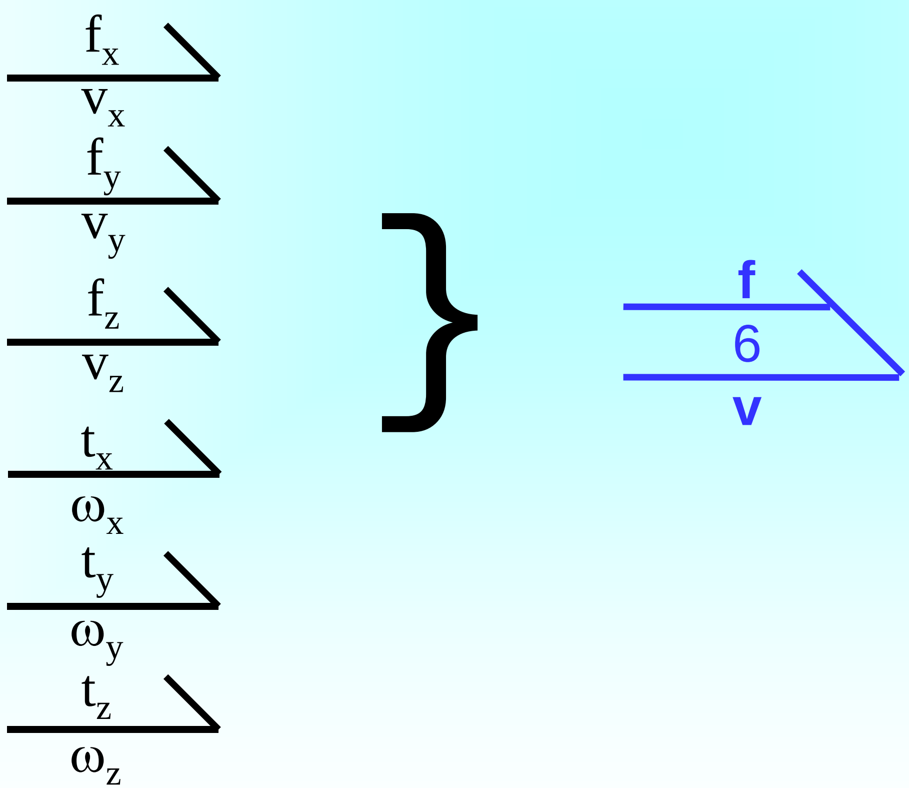
Table of Contents

Degrees of Freedom

- 1D mechanical systems exhibit exactly one degree of freedom (either translational or rotational).
- 2D mechanical systems have three degrees of freedom. They can translate along two axes, and they can rotate around an axis that is perpendicular to the plane spanned by the two translational axes.
- 3D mechanical systems allow six degrees of freedom. They can translate along three spatial axes, and they can rotate around each of those three axes as well.

3D Mechanical Multi-bonds

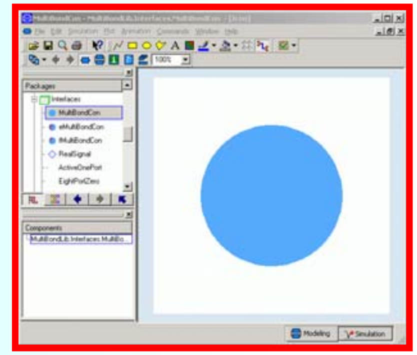
- Consequently, the 3D mechanical multi-bonds are expected to contain six parallel regular bonds, one for each degree of freedom:

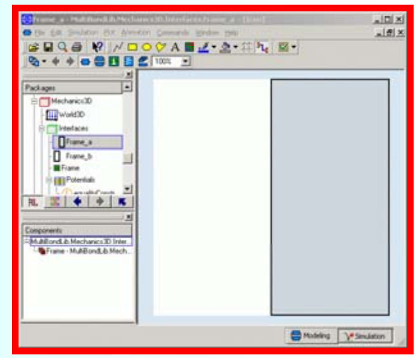


Composition of a multi-bond for 3D mechanics

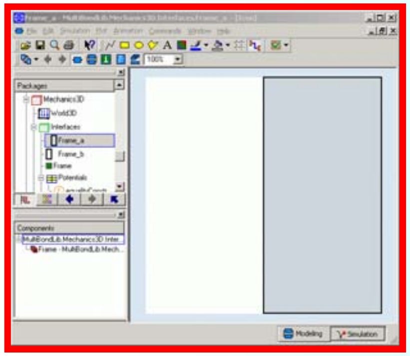
3D Mechanical Connectors

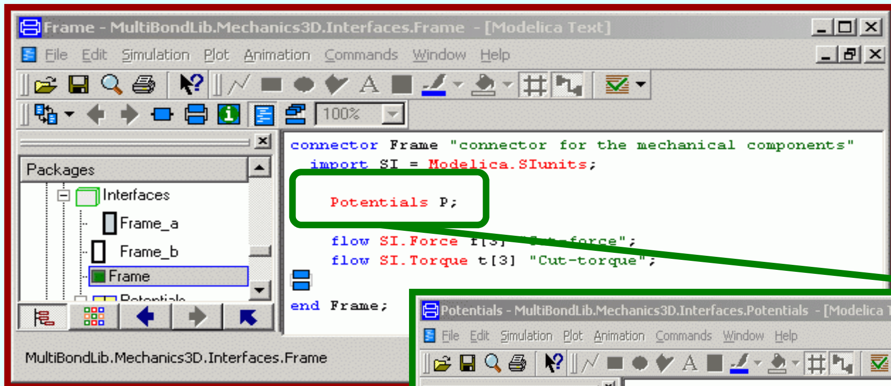
- The 3D mechanical multi-bond connectors should carry 13 variables, an effort vector, **e**, of length 6, a flow vector, **f**, also of length 6, plus the directional variable, *d*.
- The 3D mechanical multi-body connectors would need to carry 18 variables, namely 12 potential variables describing the 6 generalized positions and the 6 generalized velocities, and 6 flow variables describing the generalized forces.
- In reality, they carry 24 variables, as shown on the next slide.

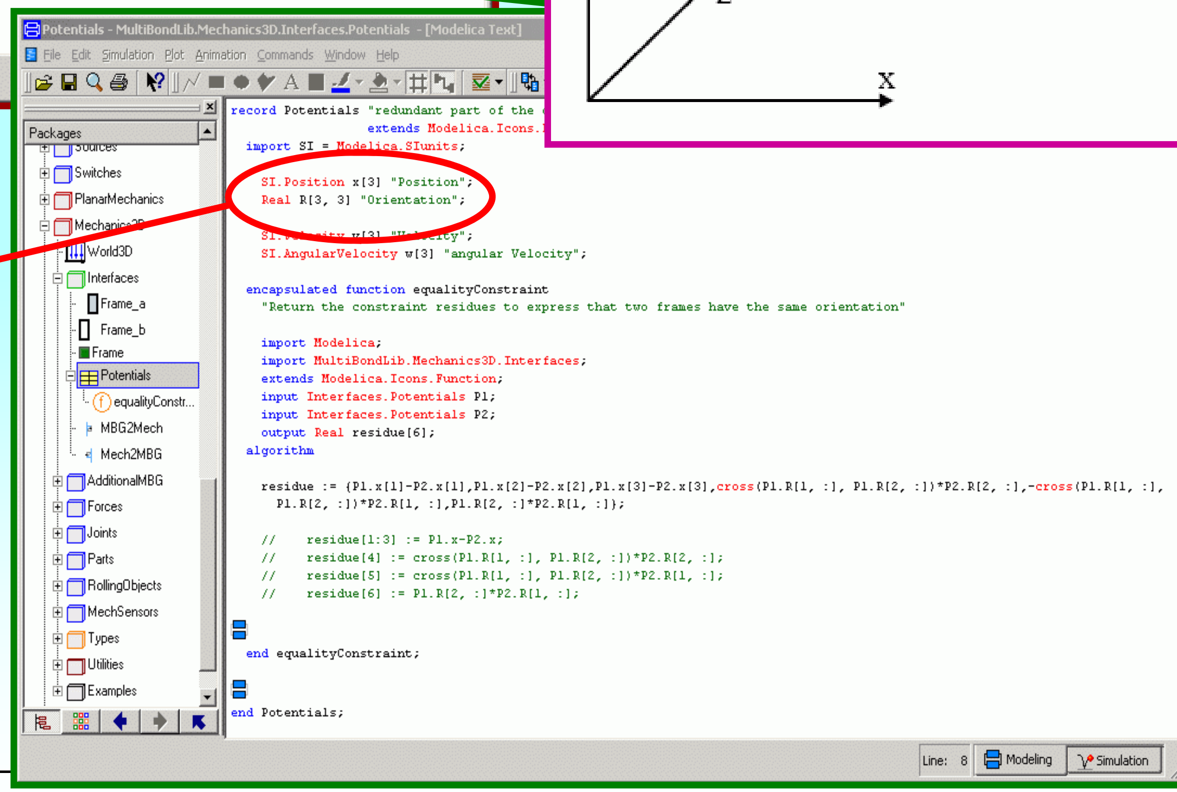


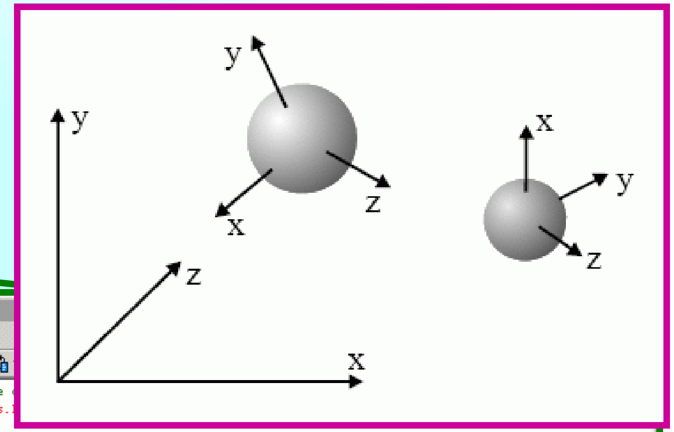


3D Mechanical Connectors II









The orientation matrix R rotates the axes of the world model into the axes accompanying the body.

$$\omega_{body} = R \cdot \omega_0$$

Orientation of the axes in the body coordinate system.

Orientation of the axes in the world coordinate system.

The Body-fixed Coordinate System

- In 3D mechanics, the inertial tensor depends on the orientation of the body relative to its coordinate system.
- Hence, if the ***world coordinate system*** is being used for formulating the d'Alembert principle for rotational motion, the inertial tensor must be constantly updated.
- Alternatively, we can formulate the d'Alembert principle in a ***body-fixed coordinate system***. In this way, the inertial tensor remains constant.
- However, we now must calculate the relative coordinate transformations across joints.
- We must also take into account the ***gyroscopic torques*** that result from formulating the d'Alembert principle in an accelerated frame.

The Body-fixed Coordinate System II

- In planar mechanics, this wasn't a problem yet. There is a single axis of rotation that is always perpendicular to the plane of translation.
- Consequently, the inertia remains constant, and we can (and have been) calculating all motions in the world coordinate system.
- This fact makes planar mechanics considerably simpler and more easy to understand than 3D mechanics.

The Orientation Matrix

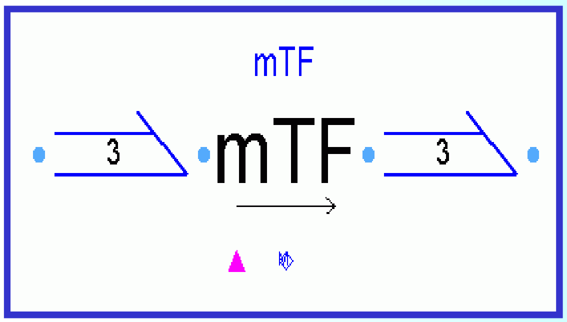
- The orientation matrix, ***R***, is a ***unitary matrix***.
- Hence:

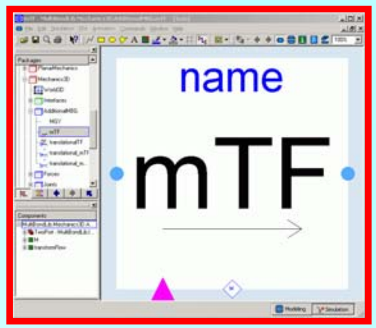
$$\boxed{\|\mathbf{R}\|_2 = 1} \qquad \boxed{\mathbf{R}^{-1} = \mathbf{R}^T}$$

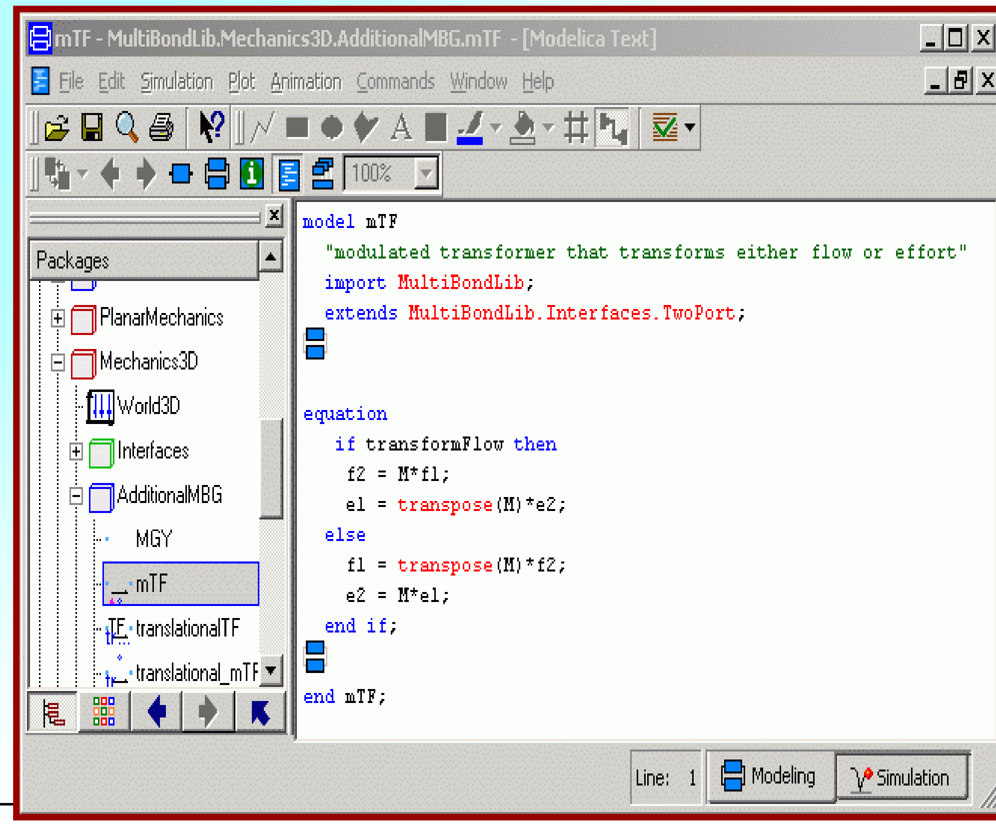
- Each row vector and each column vector of ***R*** is of length 1, hence there are 6 constraint equations connecting the 9 matrix elements.
- As expected, there are only 3 degrees of freedom, describing the relative rotation of one coordinate system to another.

Coordinate Transformations

- Coordinate transformations can be interpreted as an act of transformation in a bond graph sense:





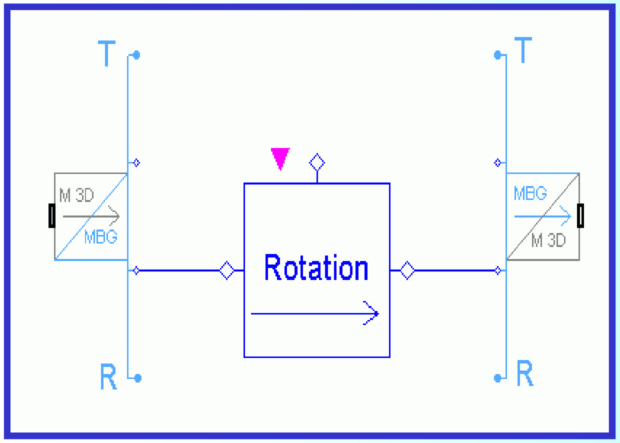


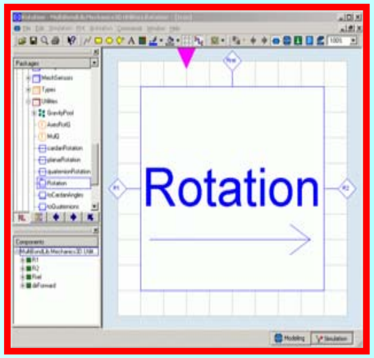
$$\omega_2 = R \cdot \omega_1$$

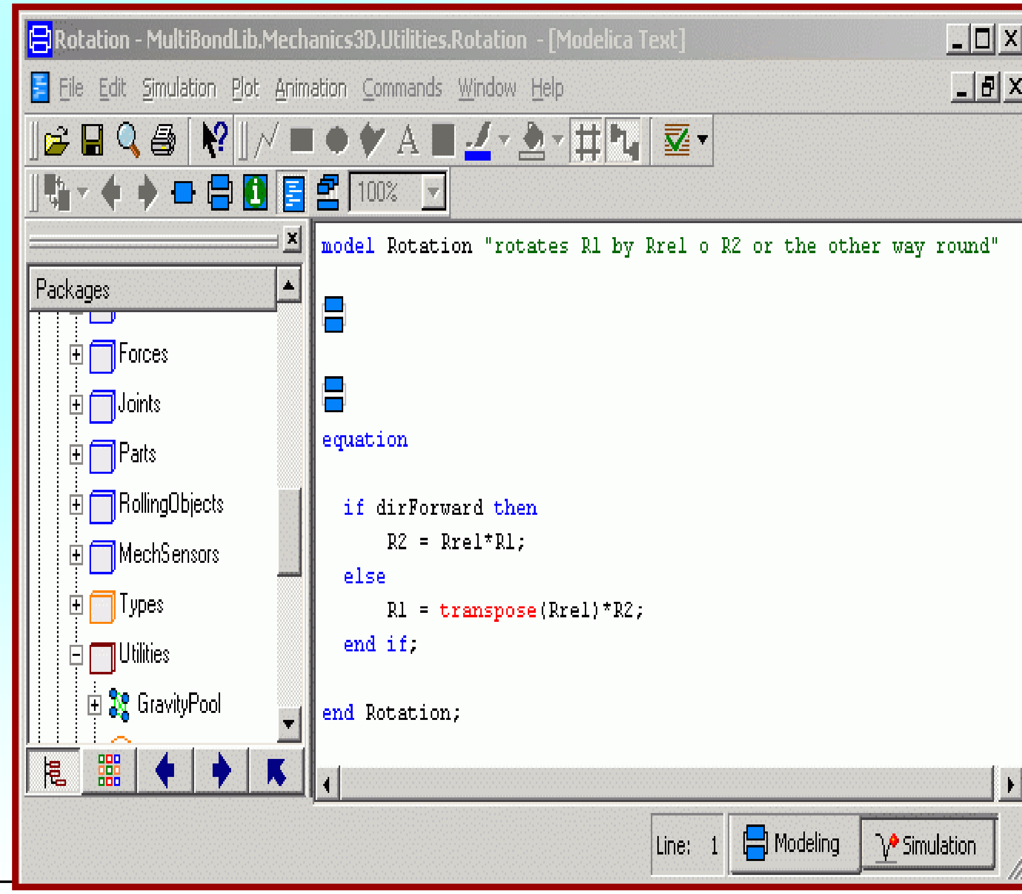
$$\Rightarrow \quad \tau_1 = R^T \cdot \tau_2$$

Coordinate Transformations II

- We must separately also transform the angular positions:





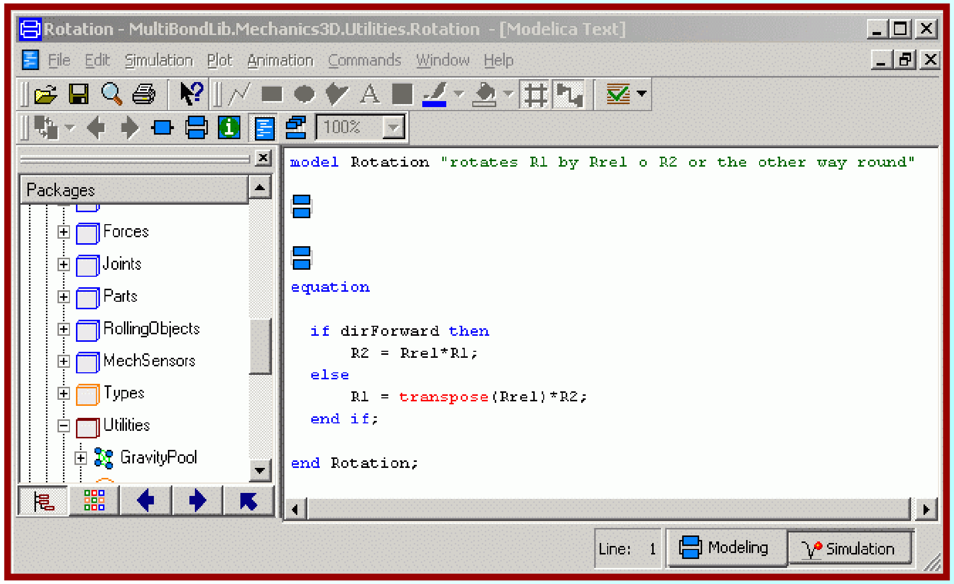


$$\omega_2 = R_{rel} \cdot \omega_1$$

$$\Rightarrow \quad \varphi_2 = R_{rel} \cdot \varphi_1$$

$$R_2 = R_{rel} \cdot R_1$$

Efficient Simulation Equations



- ***Dymola*** doesn't understand the concept of a ***unitary matrix***.
- Hence, if the computational causality requires an inversion of the ***R*** matrix, that is what ***Dymola*** will provide … in symbolic form.
- This leads to highly inefficient equations at run time.
- Thus, it is better to help ***Dymola*** by specifying the direction of computational flow explicitly.

$$R_2 = R_{rel} \cdot R_1$$

$$\Rightarrow \quad R_1 = R_{rel}^{-1} \cdot R_2 = R_{rel}^{T} \cdot R_2$$

Computation of Orientation Matrix

- One way to compute the orientation matrix, ***R***, in a non-redundant fashion is by means of ***Cardan angles***. These are the angles of rotation around the Carthesian coordinates: φ_x, φ_y, and φ_z.

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi_x) & sin(\varphi_x) \\ 0 & -sin(\varphi_x) & cos(\varphi_x) \end{pmatrix}$$

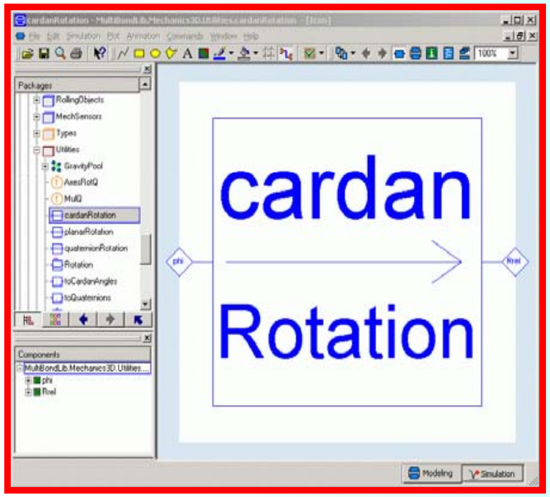
$$\mathbf{R}_y = \begin{pmatrix} cos(\varphi_y) & 0 & -sin(\varphi_y) \\ 0 & 1 & 0 \\ \sin(\varphi_y) & 0 & cos(\varphi_y) \end{pmatrix}$$

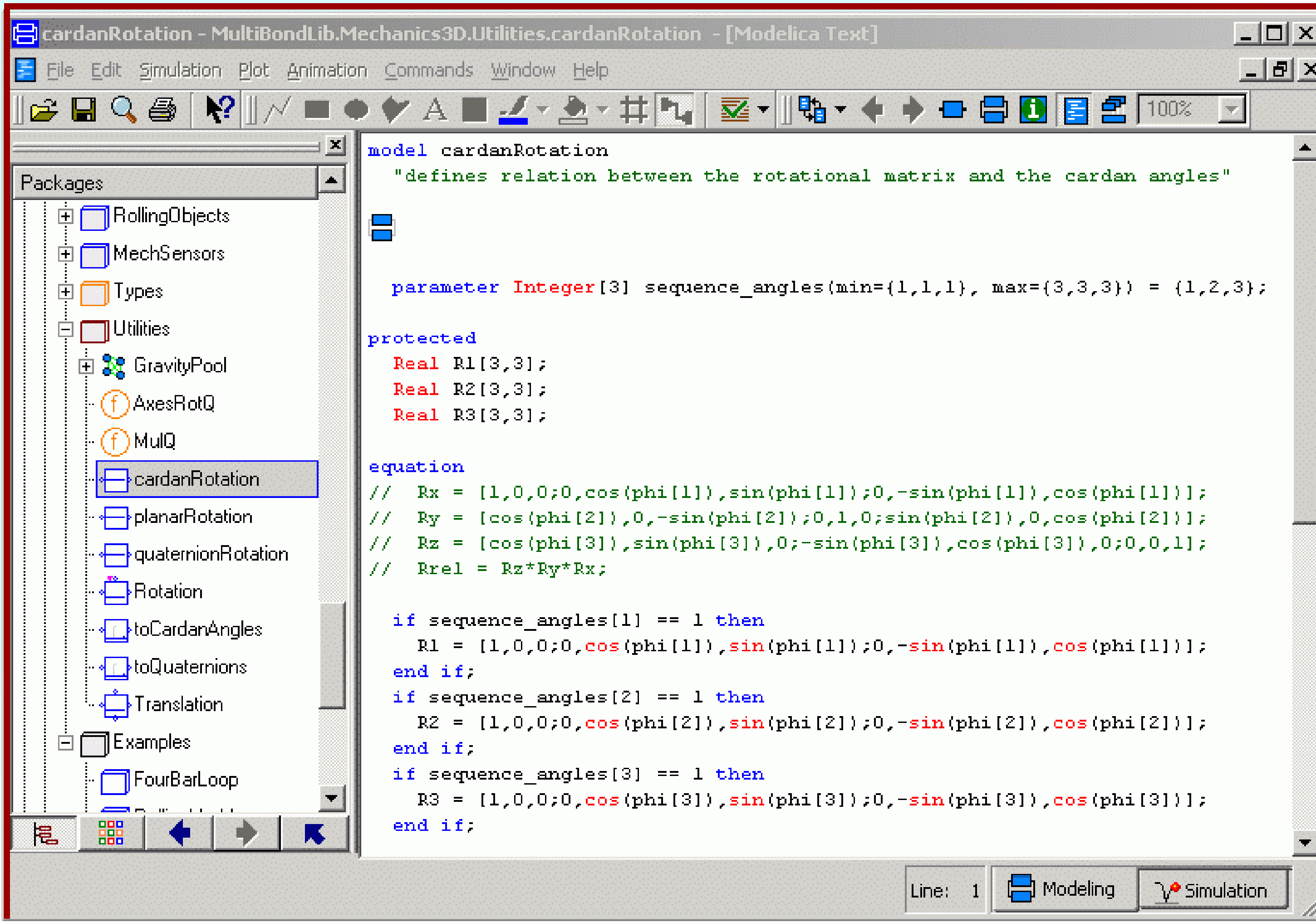
$$\mathbf{R}_z = \begin{pmatrix} cos(\varphi_z) & sin(\varphi_z) & 0 \\ -sin(\varphi_z) & cos(\varphi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\Rightarrow \boxed{R = R_z \cdot R_y \cdot R_x}$$

- Whereas ***R*** can always be computed out of φ_x, φ_y, and φ_z in a unique fashion, the opposite is unfortunately not true.
- If $\varphi_y = 90^o$, the other two axes are aligned, and φ_x and φ_z cannot be determined in a unique fashion.
- Hence ***Cardan angles*** are not always a good choice.

Computation of Orientation Matrix II





Computation of Orientation Matrix III

- Any 3D rotation can be expressed as a ***planar rotation***, φ, perpendicular to a translational plane, $\boldsymbol{n}$.
- Given the rotation angle, φ, and the translational plane, $\boldsymbol{n}$, the orientation matrix can be computed as follows:

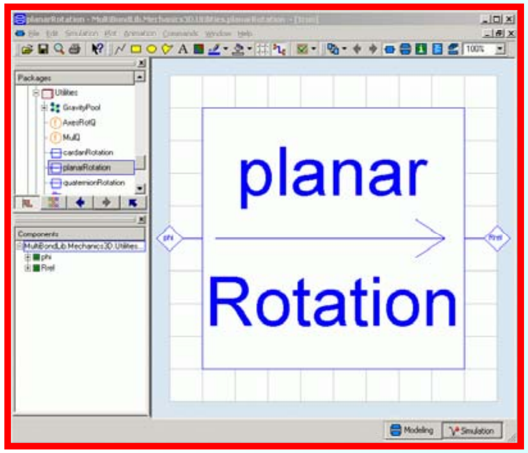
$$\boldsymbol{R} = \boldsymbol{n}\cdot\boldsymbol{n}^T + (\boldsymbol{I} - \boldsymbol{n}\cdot\boldsymbol{n}^T)\cos(\varphi) - \tilde{\boldsymbol{N}}\sin(\varphi)$$

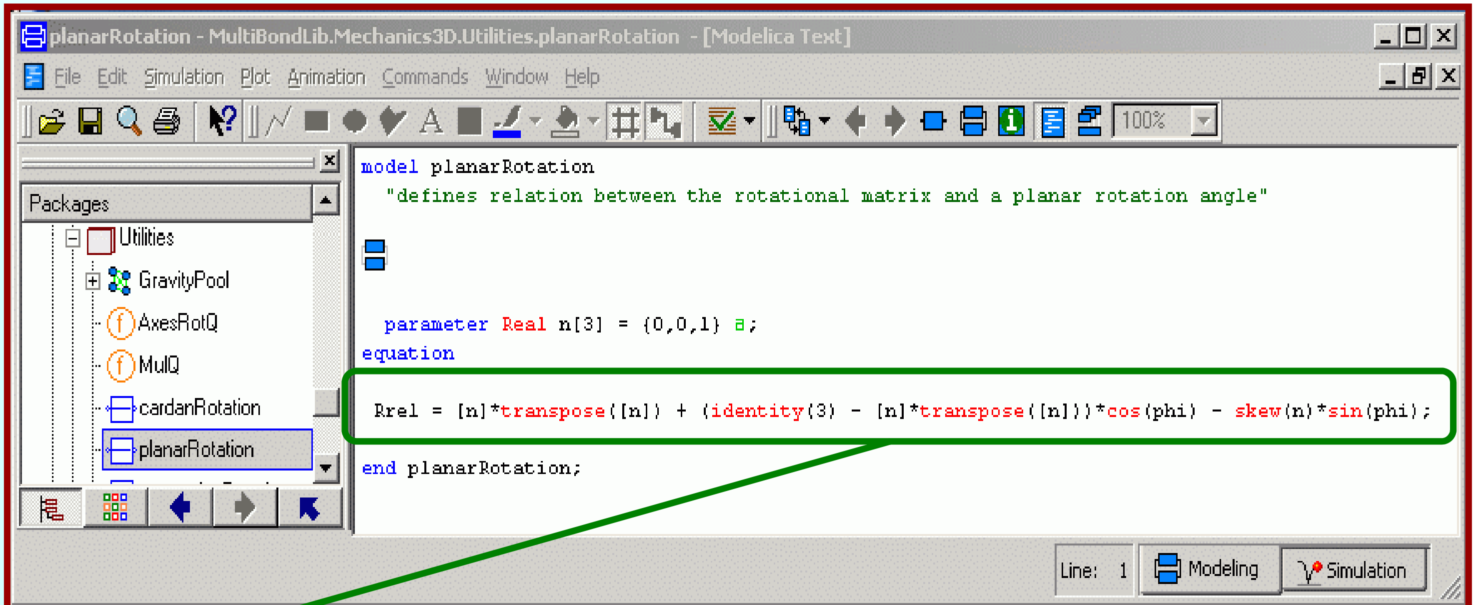
- where:

$$\tilde{\boldsymbol{N}} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

$$(\boldsymbol{a} \times \boldsymbol{b} = \tilde{\boldsymbol{A}} \cdot \boldsymbol{b})$$

Computation of Orientation Matrix IV





$$R = n \cdot n^T + (I - n \cdot n^T)\cos(\varphi) - \tilde{N}\sin(\varphi)$$

- Unfortunately, also the planar rotation method is not always invertible in a unique fashion. A null rotation does not have a well defined axis of rotation. Hence, this method should only be used if the axis of rotation is always known, as in a revolute joint.

Quaternions

- A redundant way of describing orientation that works in all situations is by means of ***quaternions***.
- Quaternions are a four-dimensional extension to complex numbers:

$$\boldsymbol{Q} = c + ui + vj + wk = c + \boldsymbol{u}$$

- Quaternions are characterized by the three imaginary components, i, j, and k that satisfy the following computational rules:

$$\begin{array}{lll} ij = k; & ji = -k; & i^2 = -1 \\ jk = i; & kj = -i; & j^2 = -1 \\ ki = j; & ik = -j; & k^2 = -1 \end{array}$$

Quaternions II

- The product of two quaternions can be written as:

$$\boldsymbol{Q}\boldsymbol{Q}' = (c + \boldsymbol{u})(c' + \boldsymbol{u}') = (cc' - \boldsymbol{u}\cdot\boldsymbol{u}') + (\boldsymbol{u} \times \boldsymbol{u}') + c\boldsymbol{u}' + c'\boldsymbol{u}$$

- The complement of a quaternion is being defined as:

$$\bar{\boldsymbol{Q}} = c + \bar{\boldsymbol{u}} = c - \boldsymbol{u}$$

- The norm of a quaternion is the product of the quaternion with its complement:

$$\boldsymbol{Q}\bar{\boldsymbol{Q}} = |\,\boldsymbol{Q}\,| = c^2 + |\boldsymbol{u}|^2$$

- A unit quaternion is a quaternion with norm 1:

$$|\,\boldsymbol{Q}\,| = c^2 + |\boldsymbol{u}|^2 = 1$$

Quaternions III

- In accordance with trigonometry:

$$\boxed{\cos(\varphi/2)^2 + \sin(\varphi/2)^2 = 1}$$

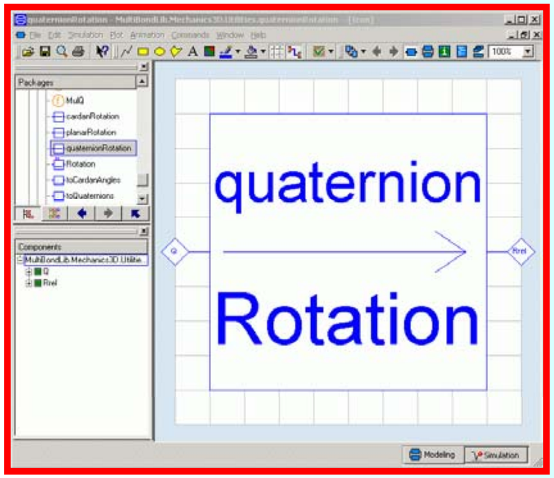
it is always possible to find an angle φ such that:

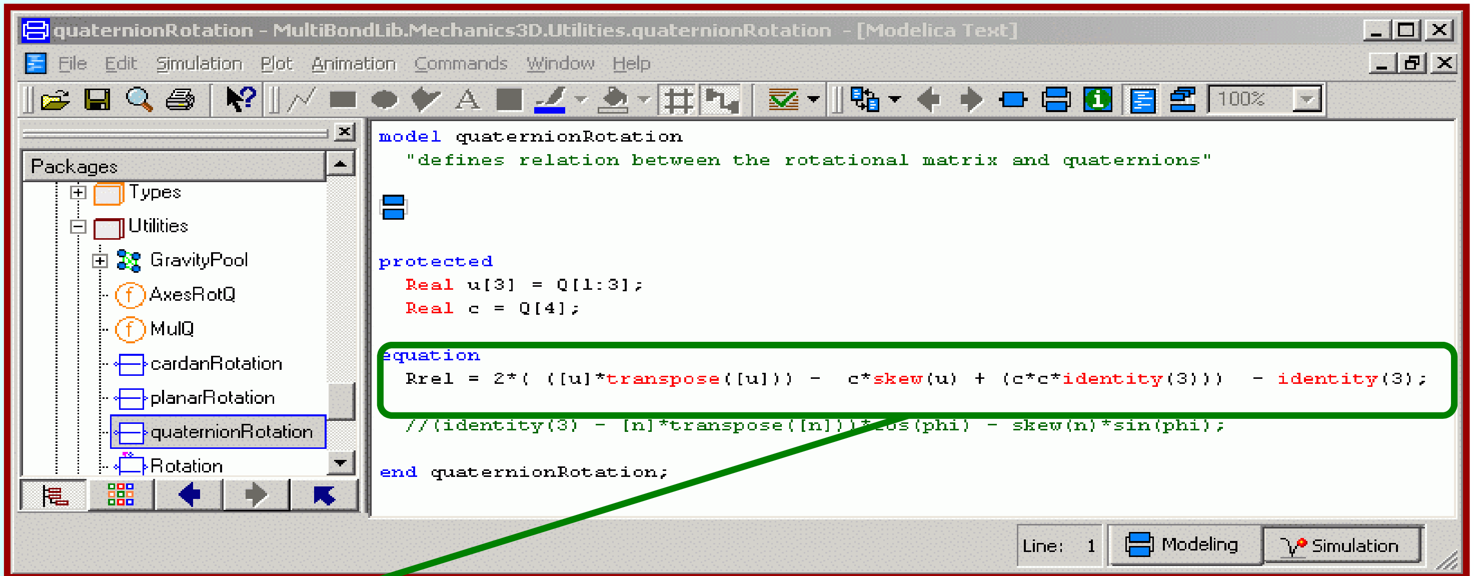
$$\boxed{c = \cos(\varphi/2); \quad |\boldsymbol{u}| = \sin(\varphi/2)}$$

- This enables us to encode the orientation of a coordinate system as a quaternion, whereby the axis of rotation is encoded as $\boldsymbol{u}$, where $[u,v,w]^T$ is being interpreted as a vector pointing in the direction of the axis of rotation. The fourth quantity, c, of the quaternion encodes the angle of rotation, φ.
- Then:

$$\boxed{\boldsymbol{R} = 2\boldsymbol{u}\cdot\boldsymbol{u}^T + 2c\tilde{\boldsymbol{U}} + 2c^2\boldsymbol{I} - \boldsymbol{I}}$$

Computation of Orientation Matrix V



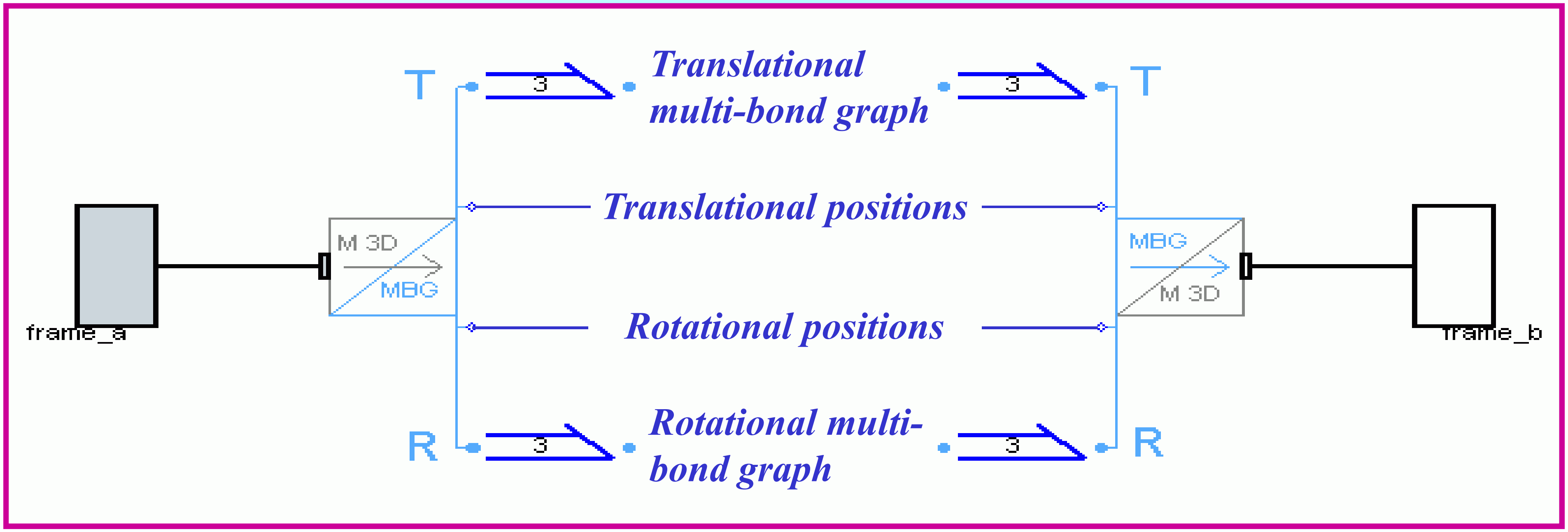


$$R = 2u \cdot u^T + 2c\tilde{U} + 2c^2 I - I$$

- The multi-bond graph library uses all three representations. It uses the ***planar rotation*** method inside revolute joints, and either ***Cardan angles*** or ***quaternions*** (user's choice) within more general joints, such as the spherical joints.

The Wrapper Models

- In the multi-bond graph library, the equations of motion are formulated in the world coordinate system for translational motions, and in a body-fixed coordinate system for rotational motions.
- For this reason, the bond graphs for translational and rotational motions are kept separate from each other, and the 3D mechanics multi-bonds have therefore still a cardinality of 3.



Equations of Motion in Body System

- Let us formulate the ***equations of motion*** in a ***body-fixed coordinate system***:

$$\boxed{\tau_0 = J_0 \cdot \dot{\omega}_0}$$

$$\Rightarrow \boxed{\tau_0 = \frac{d}{dt}\left(R^T J_{body} \omega_{body}\right)}$$

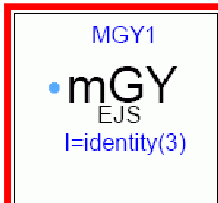
$$\Rightarrow \boxed{\tau_0 = R^T J_{body} \dot{\omega}_{body} + \dot{R}^T J_{body} \omega_{body}}$$

$$\Rightarrow \boxed{R^T \tau_{body} = R^T J_{body} z_{body} + R^T \tilde{\Omega}_{body} J_{body} \omega_{body}}$$

$$\Rightarrow \boxed{\tau_{body} = J_{body} z_{body} + \underbrace{\omega_{body} \times J_{body} \omega_{body}}_{\text{Gyroscopic torque}}}$$

The Eulerian Junction Structure

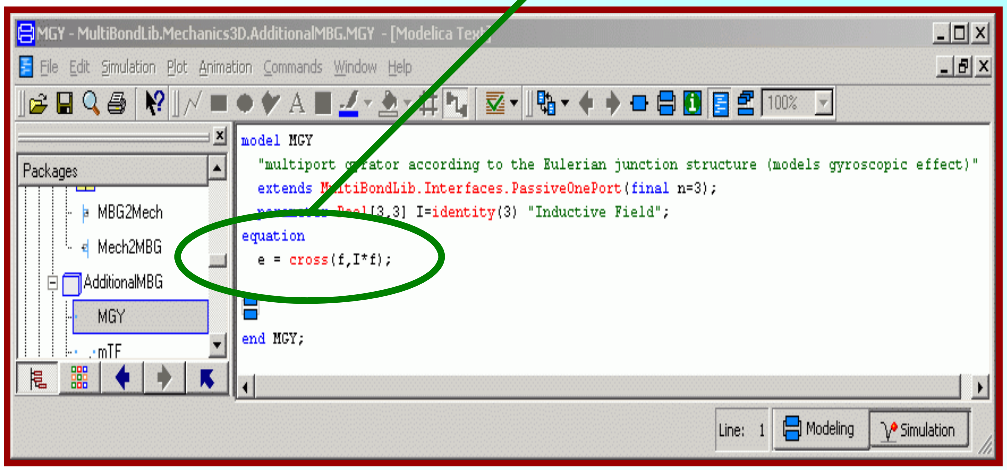
- The *gyroscopic torque* can be formulated, in terms of bond graphs, as a so-called ***Eulerian Junction Structure (EJS)***:



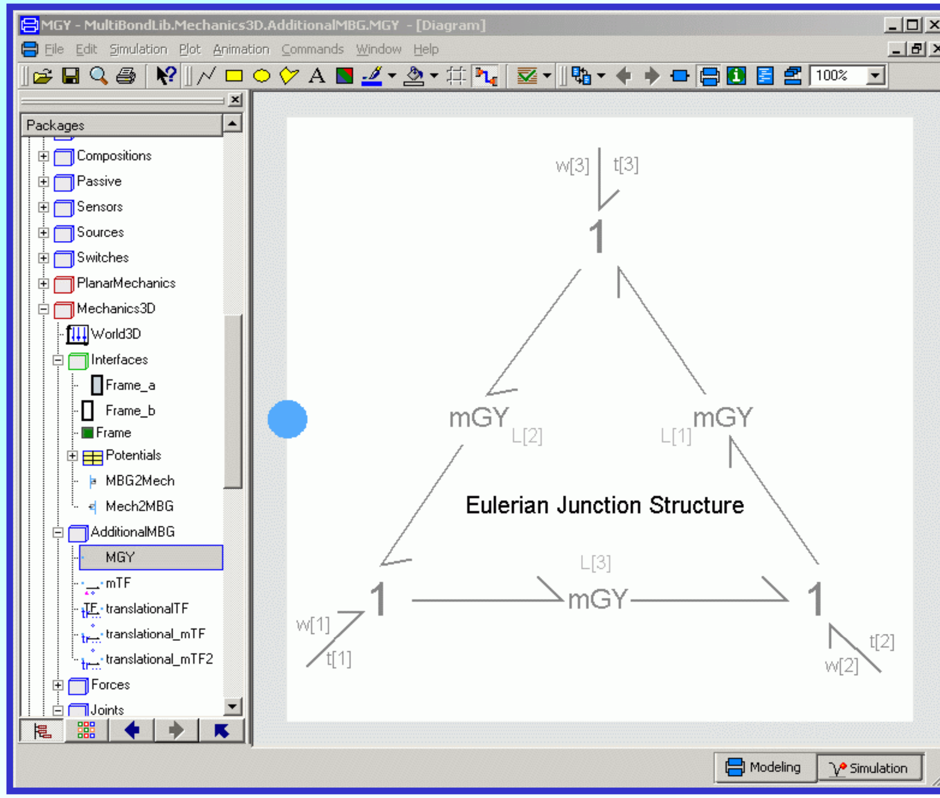
Name:	MGY
Location:	MultiBondLib.Mechanics3D.AdditionalMBG
Parameters:	Inertia tensor **J**.
Equations:	

$$\mathbf{e} = \mathbf{f} \times \mathbf{J} \cdot \mathbf{f}$$

External description



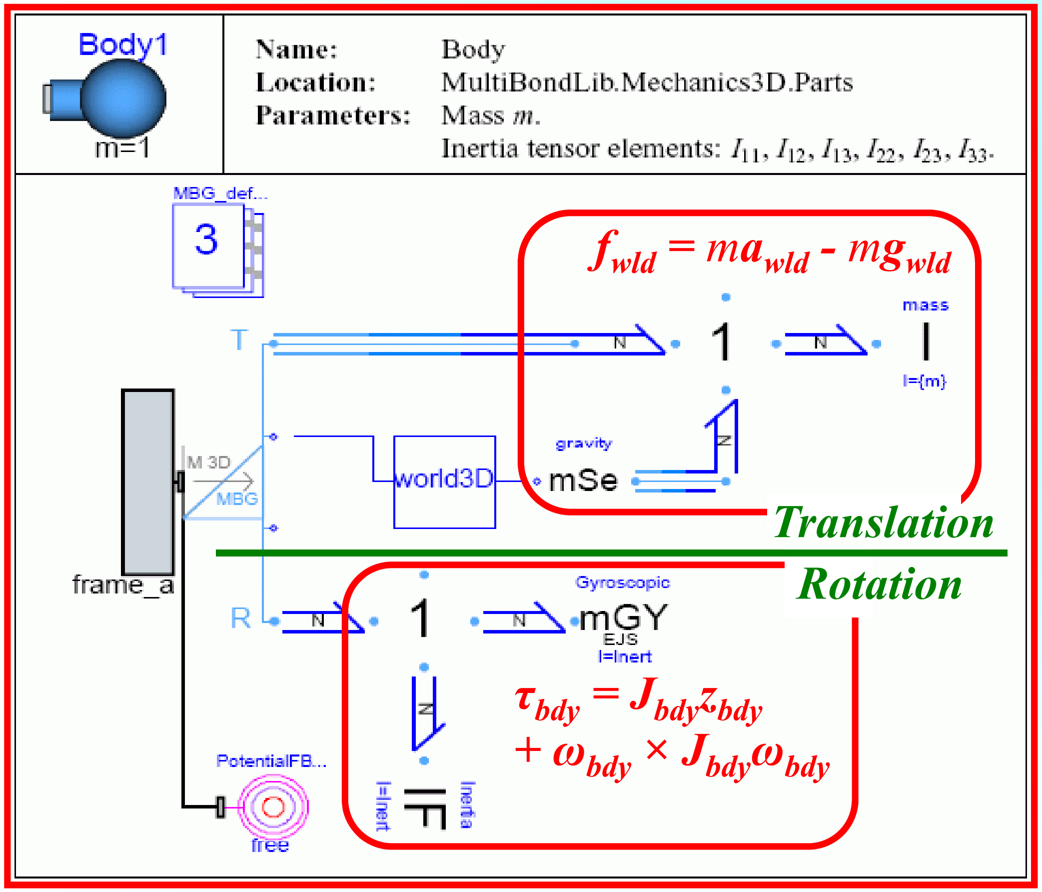
Multi-bond graph implementation



Bond graph description

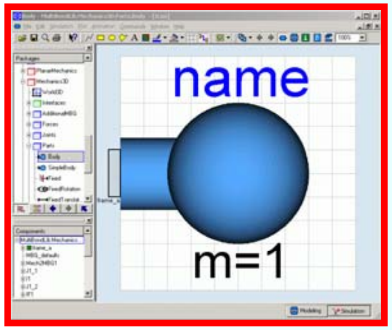
The Model of a Body

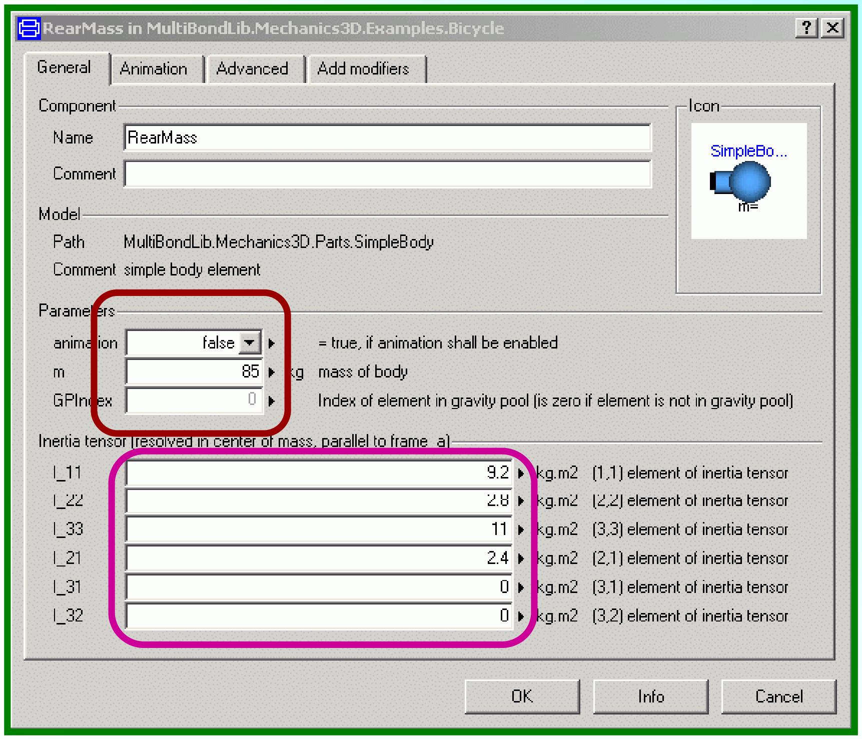
- We are now ready to model a general body using the multi-bond graph library:

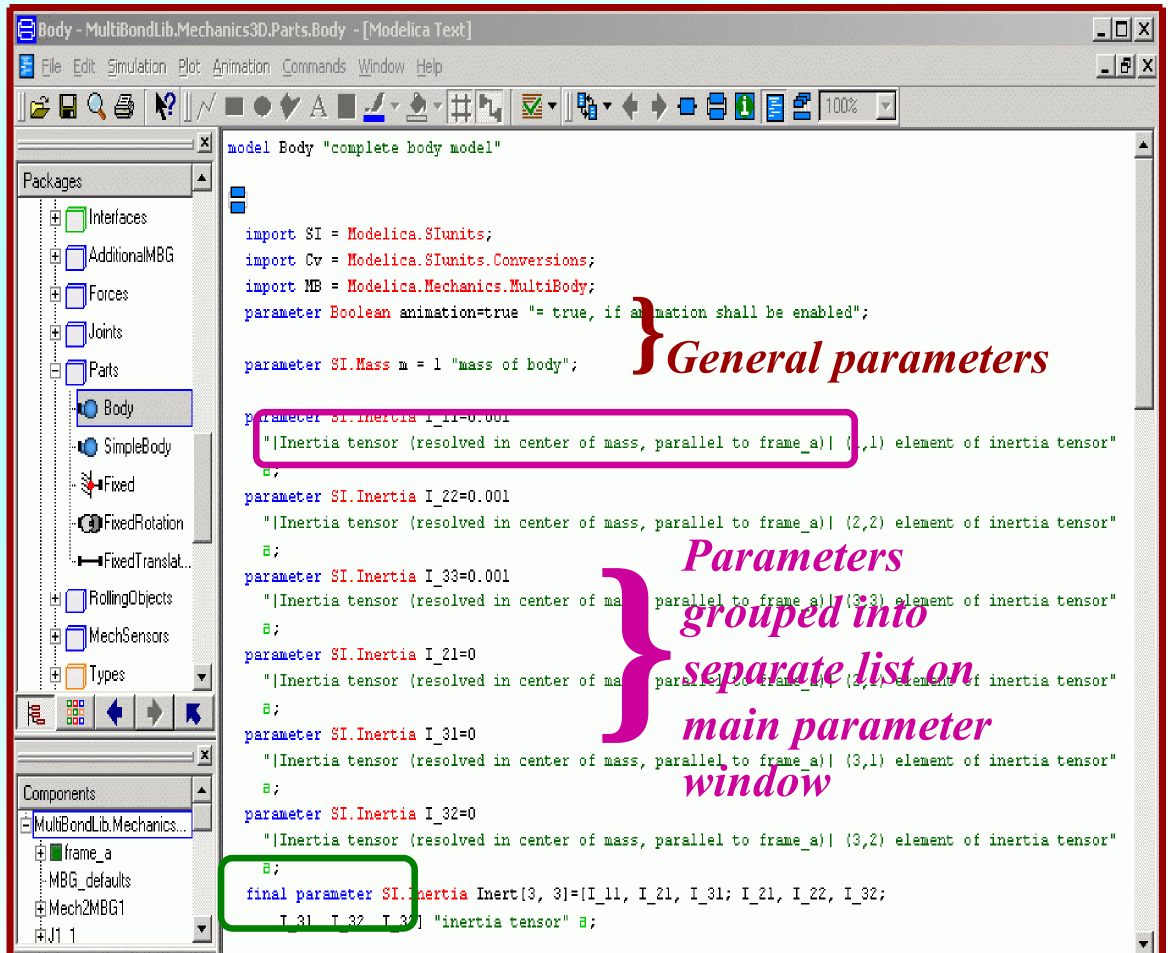


- The translational equations of motion are formulated in world coordinates.
- The rotational equations of motion are formulated in body-fixed coordinates.
- The gravitational pool is computed by the world model of wrapped 3D mechanics.

The Model of a Body II





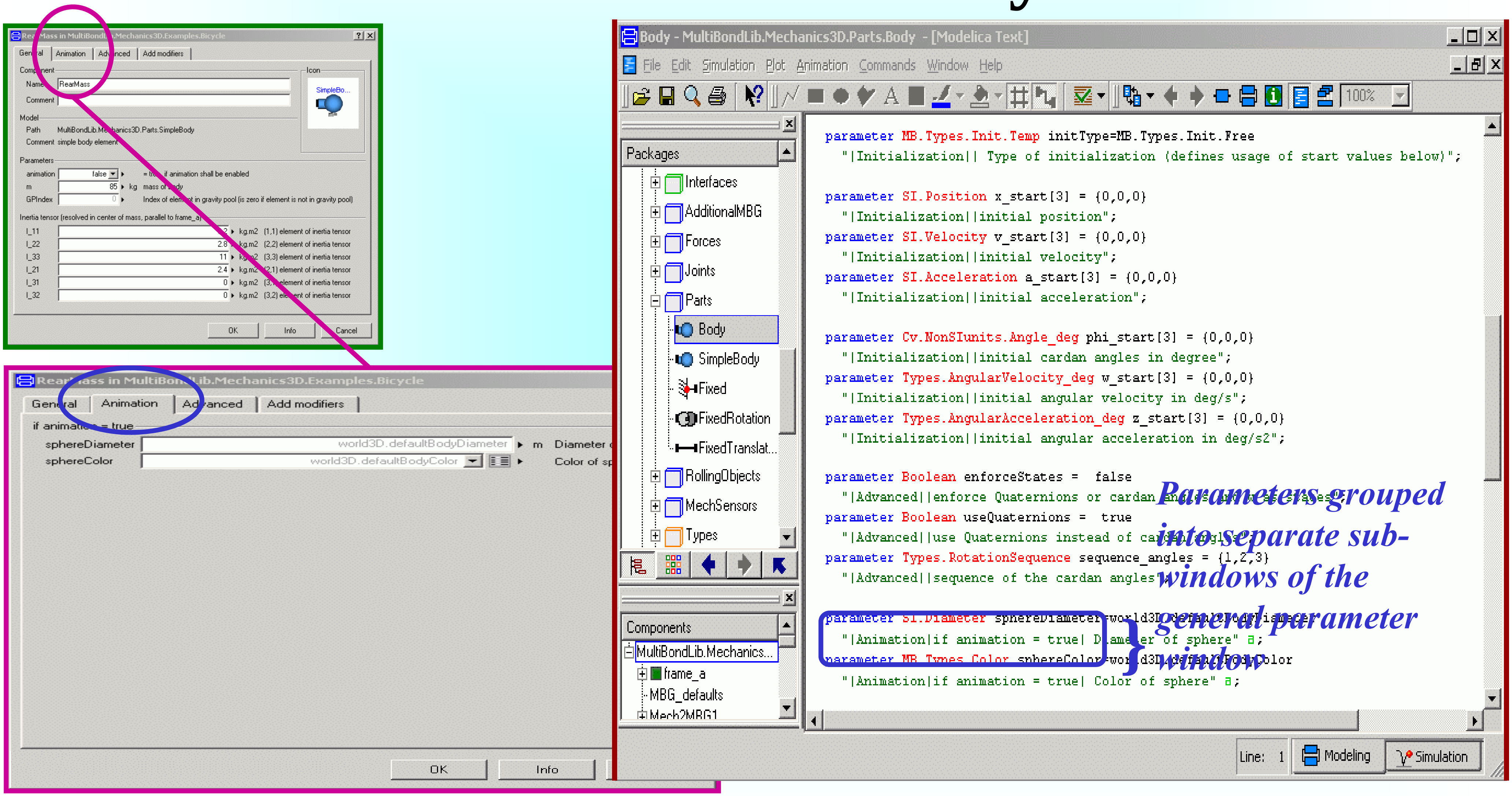


General parameters

Parameters grouped into separate list on main parameter window

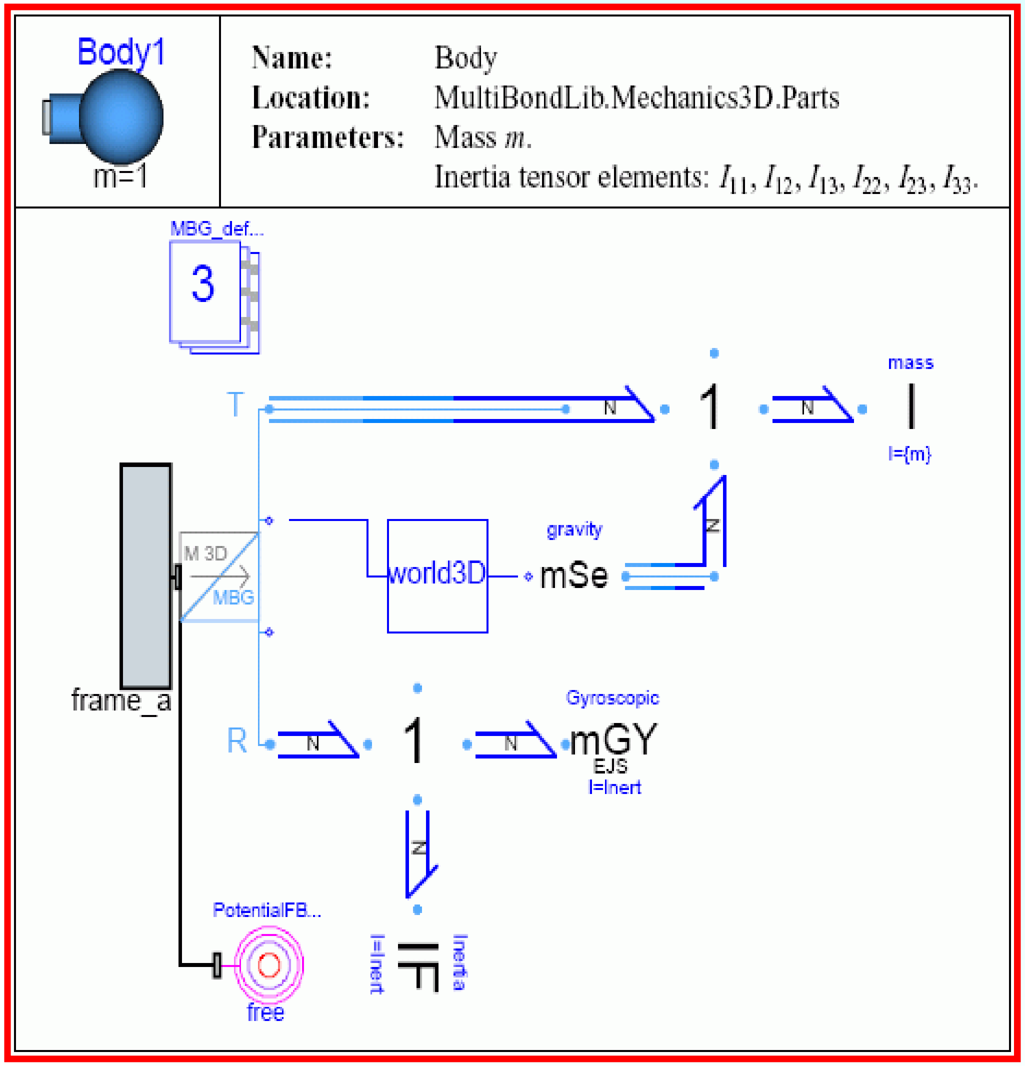
Final parameters are computed parameters. They are not listed.

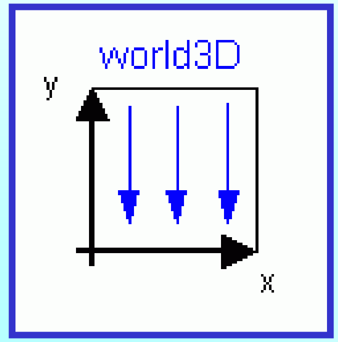
The Model of a Body III



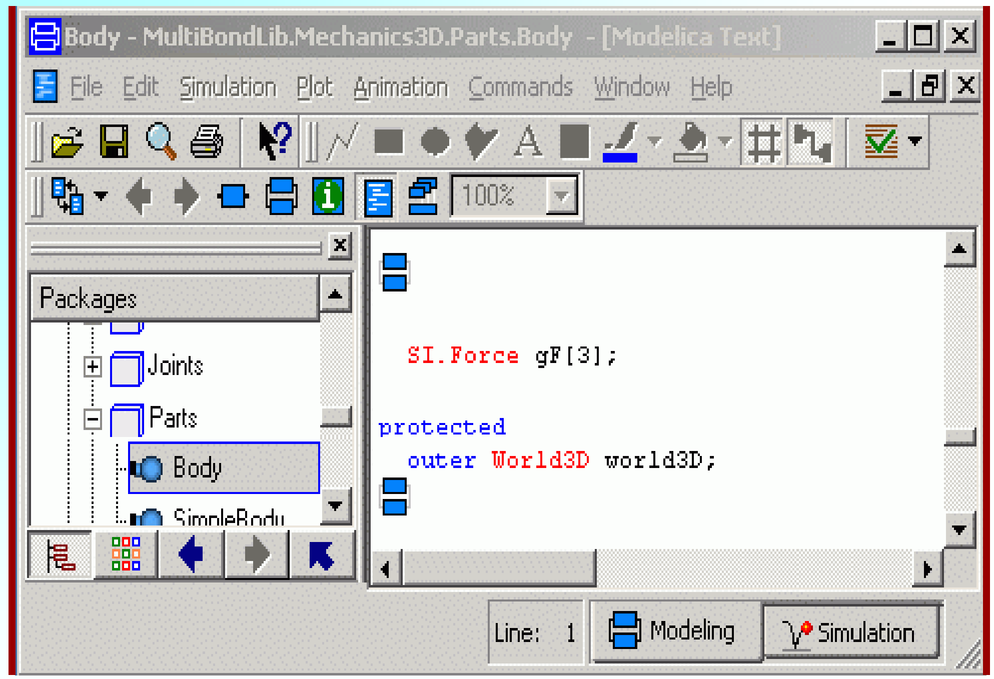
Parameters grouped into separate sub-windows of the general parameter window

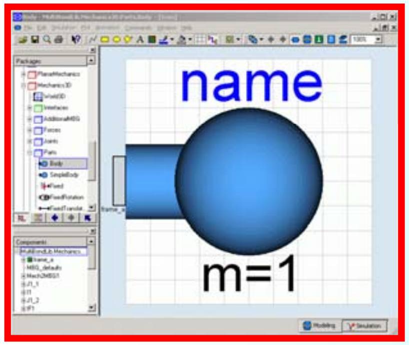
The Model of a Body IV





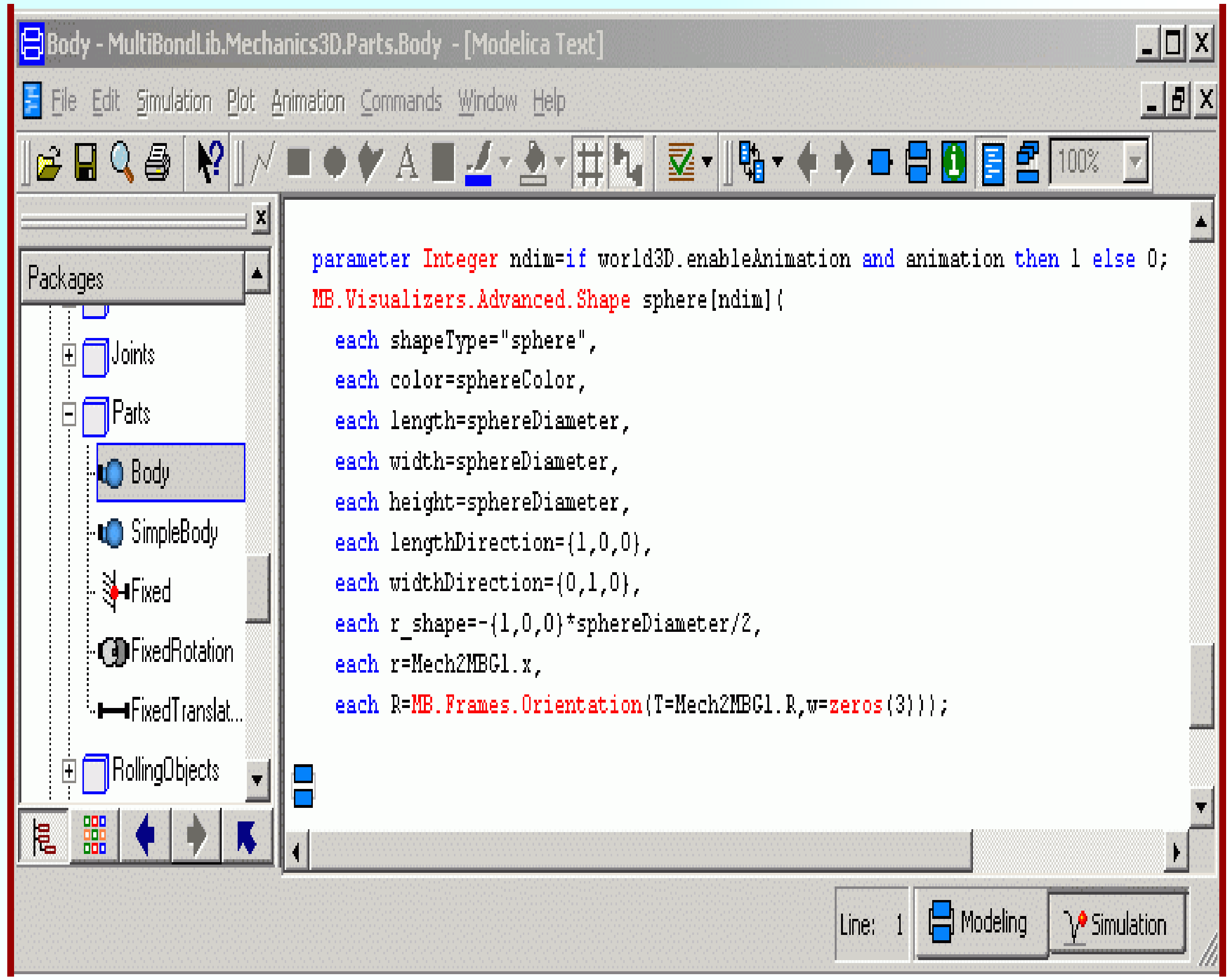
Every 3D mechanical wrapped multi-bond graph model must invoke the ***world3D*** model that must be declared in each wrapped multi-bond graph component model as an outer model.



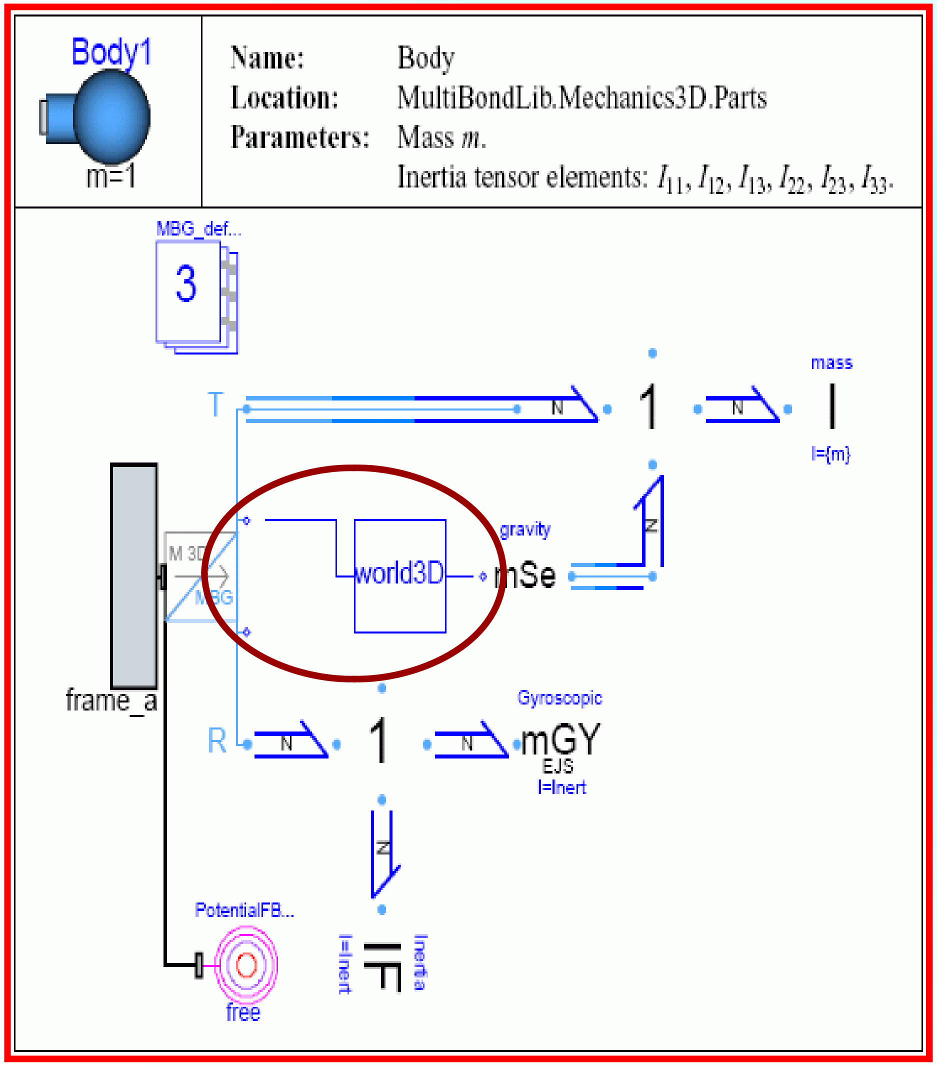


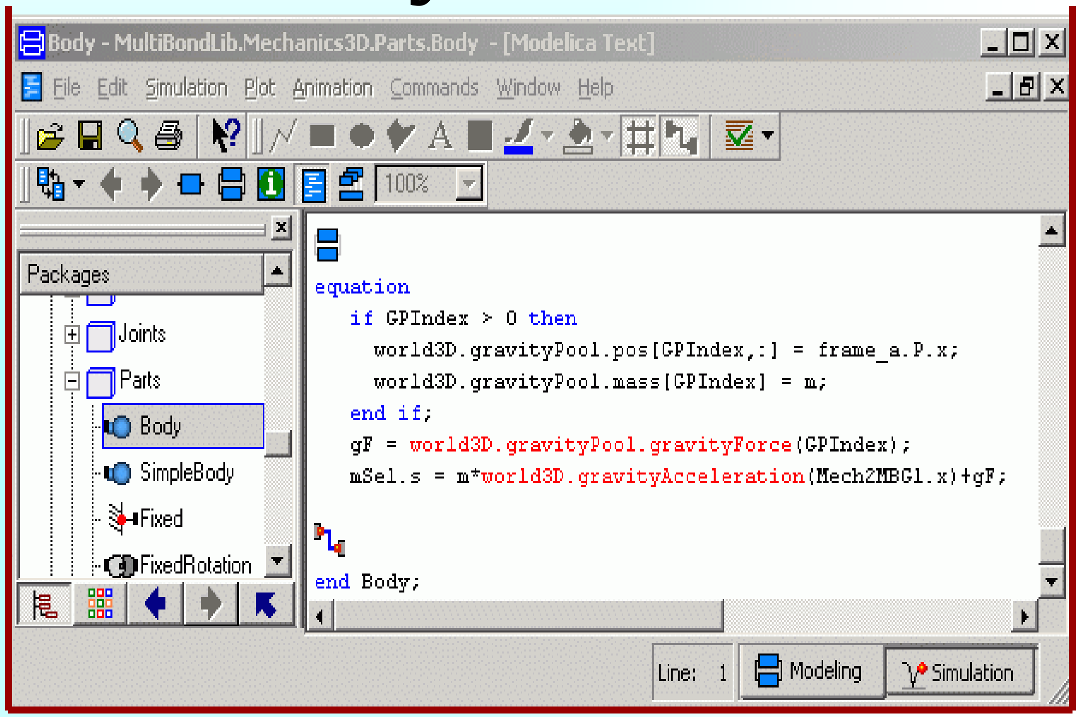
The Model of a Body V

- The shapes and sizes of bodies can be declared for the purpose of animation.
- This feature is borrowed from the multi-body systems sub-library of the standard ***Modelica*** mechanics library.
- You find documentation there for predefined shapes under the sub-heading ***Visualizers***, and more generally under the sub-sub-entry of ***Advanced → Shape***.



The Model of a Body VI





- The center of the gravitational pool is specified in the equation window. The corresponding graphical element is only a drawing. The user is reminded of this fact, by not connecting the "connections" all the way to the connectors. The user then knows that "something is fishy."

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.