Extended Petri Net 3D Simulator Handbook

Group A

Introduction

This document consists of a handbook and a how to use manual for the Extended Petri Net 3D Simulator designed as a project for Software Engineering 2.

Installation

The following section describes the components required in order for the software to work.

Prerequisites

The product consists of a series of Eclipse plugins and in order to be able to execute it, installation of the following is required:

- Java JDK 6[1]
- Eclipse Kepler[2]

Moreover, in order to be able to run the Petri Net Editor, the ePNK should be installed using the instructions found here: http://www2.imm.dtu.dk/~ekki/projects/ePNK/ for the Kepler version of Eclipse.

Product installation

For the current version of the product, there is a set of plug-in projects that need to be downloaded locally and imported into the Eclipse Development Environment. After all the projects have been imported, they can be run in the Eclipse Runtime Workbench by selecting them and clicking on the *Run* command.

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[2] http://eclipse.org/downloads/

Tutorial

This tutorial is intended as a how-to for the end user of out Petri Net 3D Simulator. Technical details of it can be found in the attached System Specification document.

The Petri Net 3D Simulator is composed by four editors and a configuration: Petri net editor, Geometry editor, Appearance Editor and the Configuration.

The first one is intended for modeling the Petri net that the user wants to model. The second one lets the user define where the elements of the Petri net model will be shown in the 3D simulation. The appearance editor is used to link an appearance to every element of the Petri net model. Finally, the configuration links the three previous models and allows the user to launch a simulation.

Description of the executables and examples

Together with this document, two files are attached:

- dk-dtu-se2-petri-net-simulator.zip
- dk-dtu-se2-example.zip

The first one contains the Eclipse projects that are required to run the plugins that are part of the software, and that have to be imported to Eclipse as existing projects:

1. Launch Eclipse Kepler standard Edition with the following extensions installed:
    a. EMF
    b. GMP (GMF)
    c. Xtext SDK
2. File > Import...
3. General > Existing Projects into Workspace
4. Browse into the location of dk-dtu-se2-development.zip and import all of them.

Moreover, given the large size of the 3D library that has been used, it is not packed with the rest of the software, but it is required for running it. In order to do it, the user should check it out from the following Subversion repository and import it as an Eclipse project.

https://svn.imm.dtu.dk/se2/svn/e13-groupA/src/jMonkeyEclipse

The projects would have appeared into the workspace, and the code can be checked. In order to run the examples, the following steps have to be followed:

1. Click Run in Eclipse. A new Eclipse Runtime will be launched.

2. File > Import...
3. General > Existing Projects into Workspace
4. Browse into the location of dk-dtu-se2-example.zip and import it.

Petri Net editor

Steps to create a Petri net from the Petri net editor

In order to create a Petri net document, we need to do the following steps:

- First, create a new empty project and add a new file to this project
- The file must be a PNML Document, as shown below:

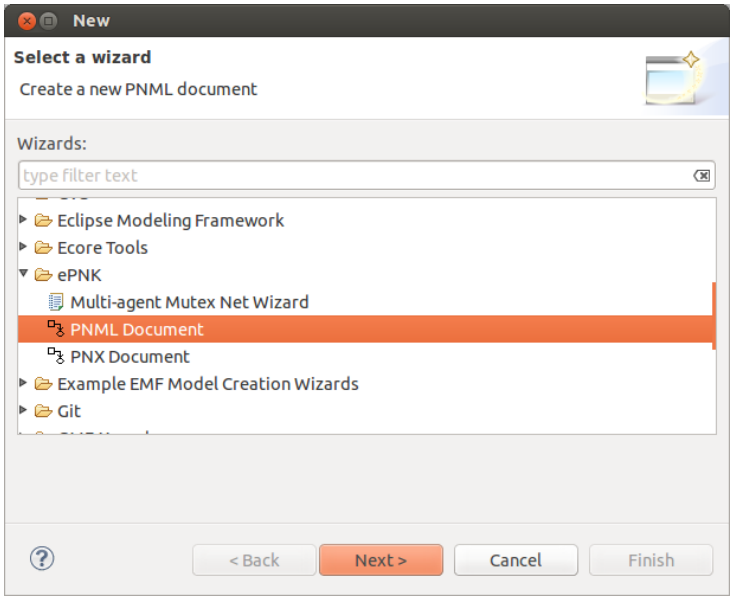

Figure 1: Creating a PNML document

- Select the folder you want to put the document into and name the file as you want to. (As shown below)

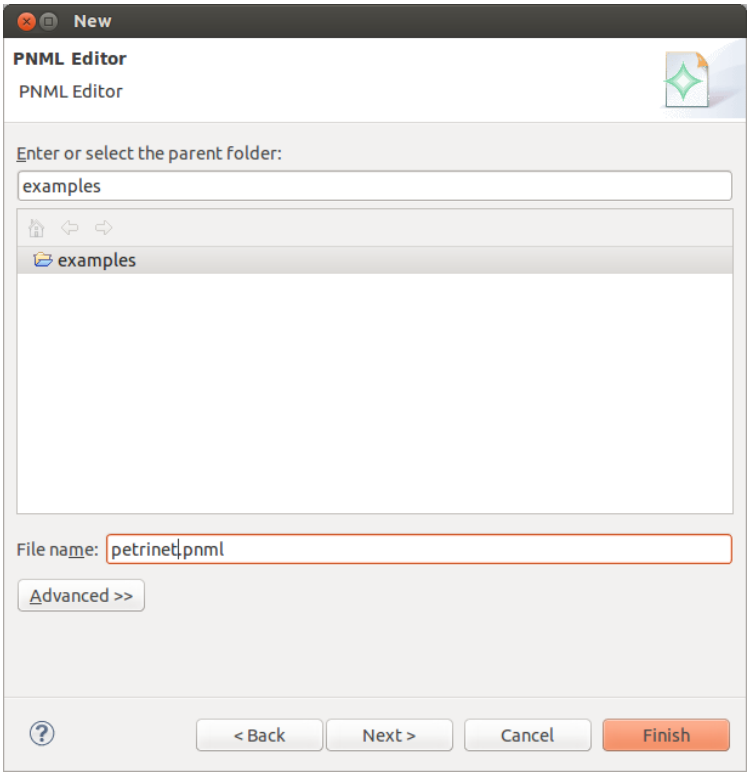

Figure 2: Creation of a PNML editor file

- Use a Petri Net Doc in UTF-8 encoding for this document. (As shown below)

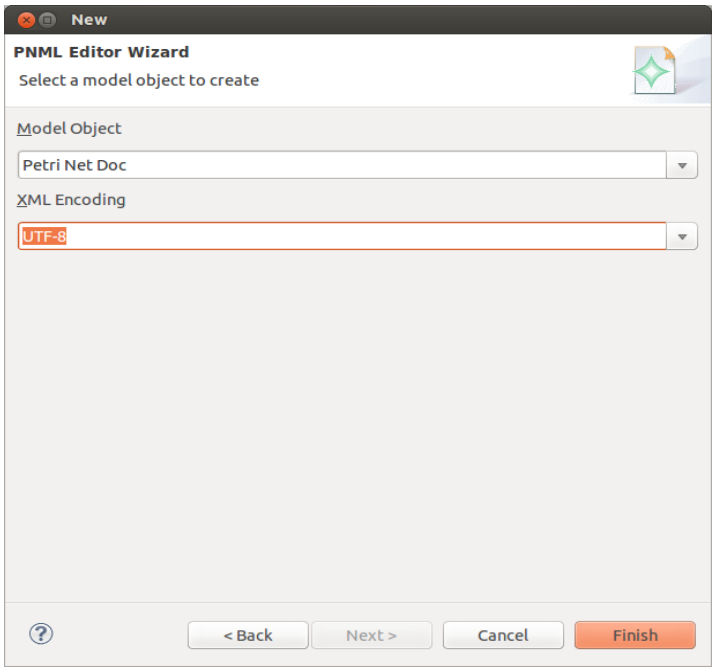

Figure 3: Selection of the Model Object

- From the Root element which is the Petri Net Doc, right click and create a New Child from our Petri net type. The Petri net type we have implemented is named: "ExtendedPetriNet", as shown below

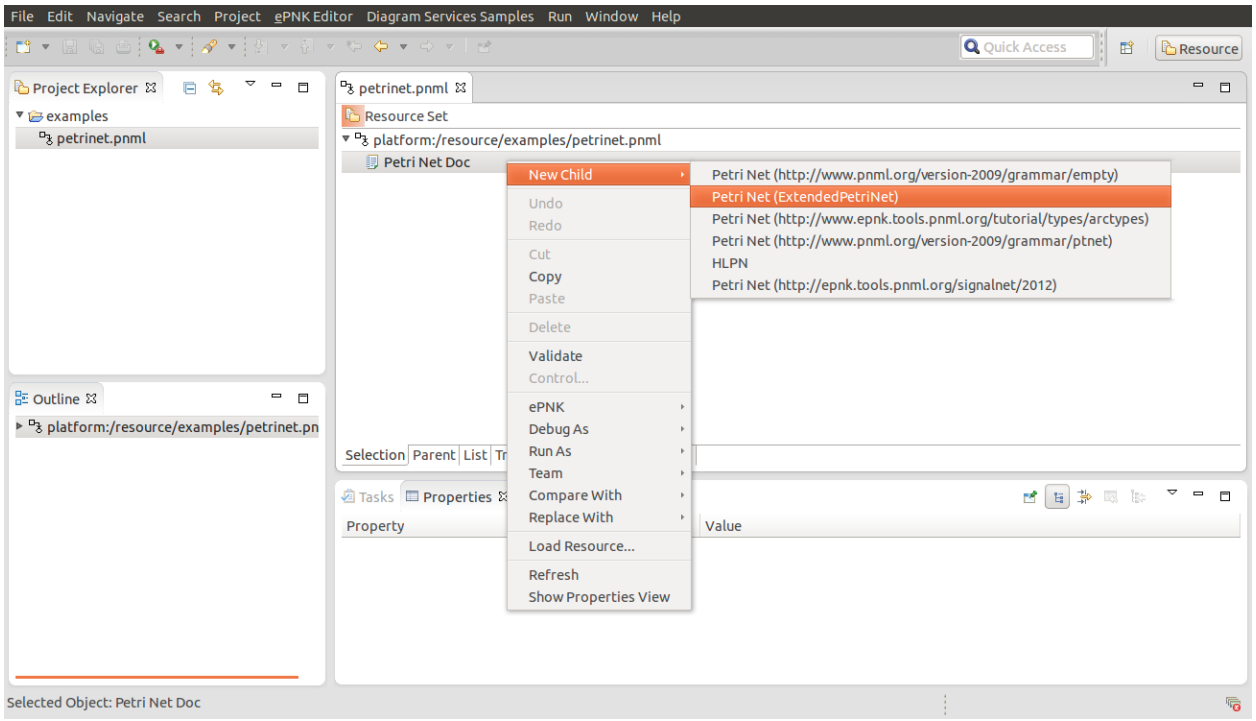

Figure 4: Creation of a Petri Net

- Select the new element created named "Petri Net: ExtendedPetriNet", right click on it and add a new Page. You will then be able to add children to this Page, corresponding to the Petri net elements you want. (Shown below)

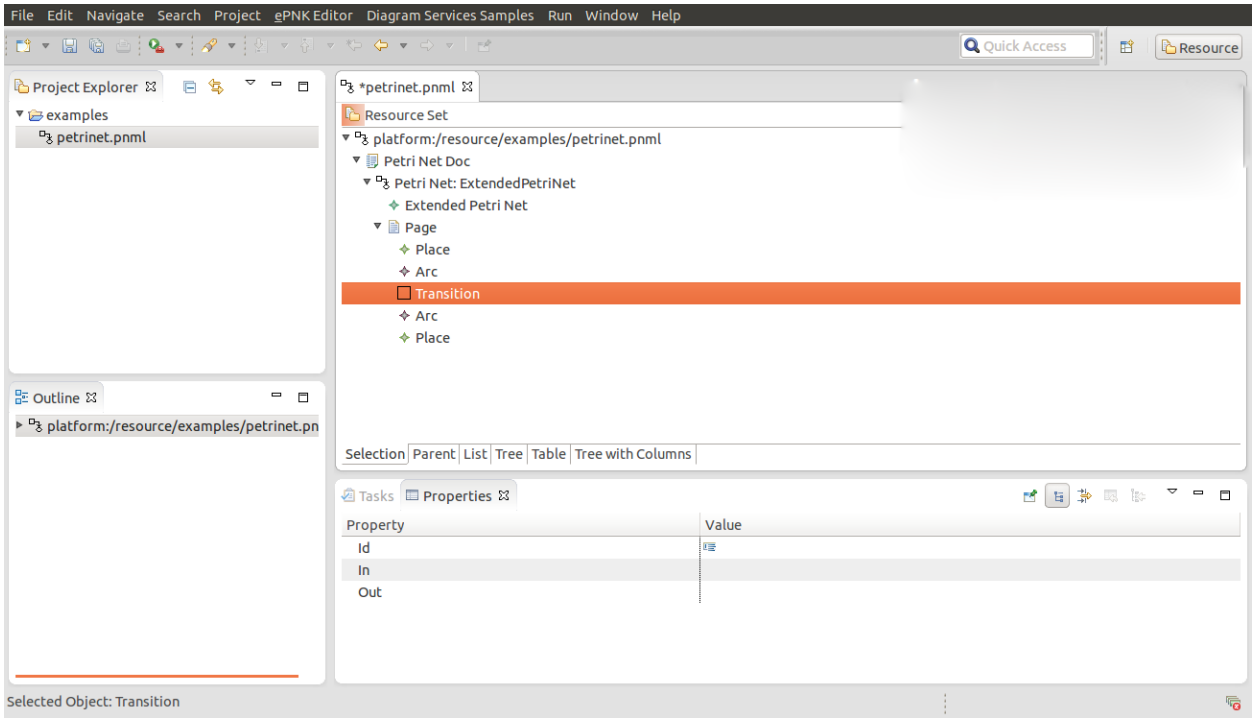

Figure 5: Example of a Petri Net

Petri Net Graphical Editor

There is the possibility of editing a Petri net with a graphical editor instead of a tree editor. To enter in the graphical mode, simply double click the “Page” line in the tree editor. This will open up the new editor as below:

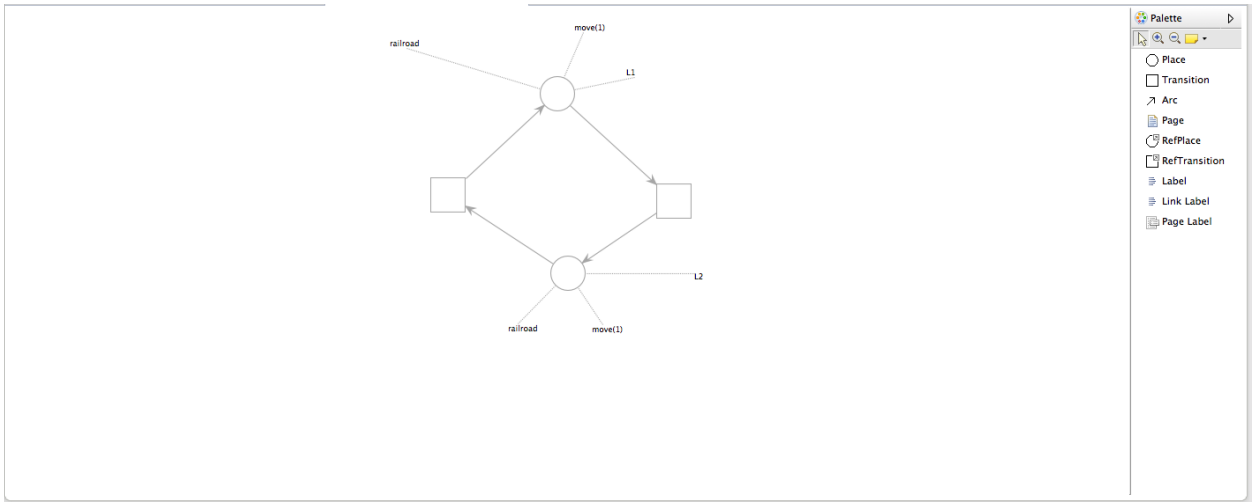

Figure 6: Example of a Petri net designed with the graphical editor

All the petri net objects are available to the user on the right side panel, and once an item has been clicked on, a click in the canvas creates the specified object.

To edit the properties of the items created, simply click on an object to open its properties view shown as below:

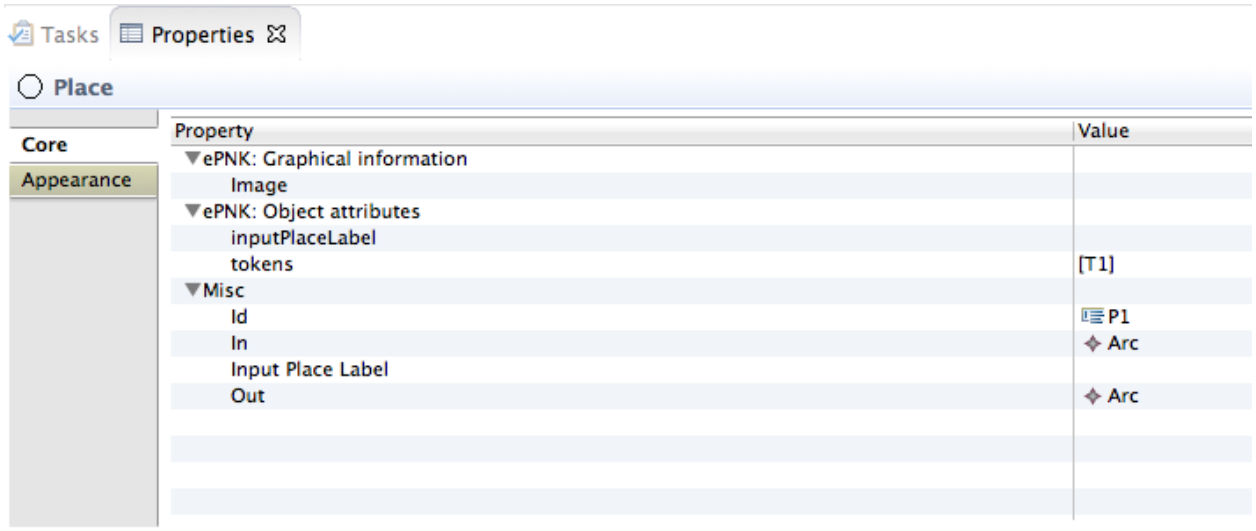

Figure 7: The properties of a Place item in the graphical editor

This graphical editor presents an alternative to the tree editor commonly used.

Animation Labels

The animation labels that can be defined in the Petri Net Editor have a specific format which needs to be followed in order to work.

Animation	Syntax	Description
Move (speed)	move(1.0)	The tokens inside a place move with speed 1.0 (it has to be a double).
Appear (geometry, appareance)	appear(TL, red)	Changes the appearance of the element TL with the one defined in “red”.

Geometry editor

Geometry editor is used to define geometries of the user defined Petri net. The places within petri net have geometries which define each places graphical shape in simulation. The shapes can be either lines or coordinate locations, which are called as input points. These shapes are created and edited in the geometry editor.

To use geometry editor in Eclipse, first thing needed to do is to create a geometry and geometry diagram. To create them, a new General Project should be created by **General > Project** which is found in **File > New > Project**, if it hasn't been made already. Give the project some name and click finish.

Right click the project folder **New > Other**. Find Geometry diagram from the list, the search functionality might help, choose it and click next. Refer to Figure 8. Name it, click next, name the geometry and then click finish. On the project explorer leaf two new files have now been generated, geometry and geometry diagram. The geometry diagram is Graphical geometry editor and the geometry is the model, presented in a tree view, that geometry diagram edits. Editing geometry within the tree view of the geometry will not do correct changes on the graphical view of the geometry. If the geometry diagram is not open, open it now.

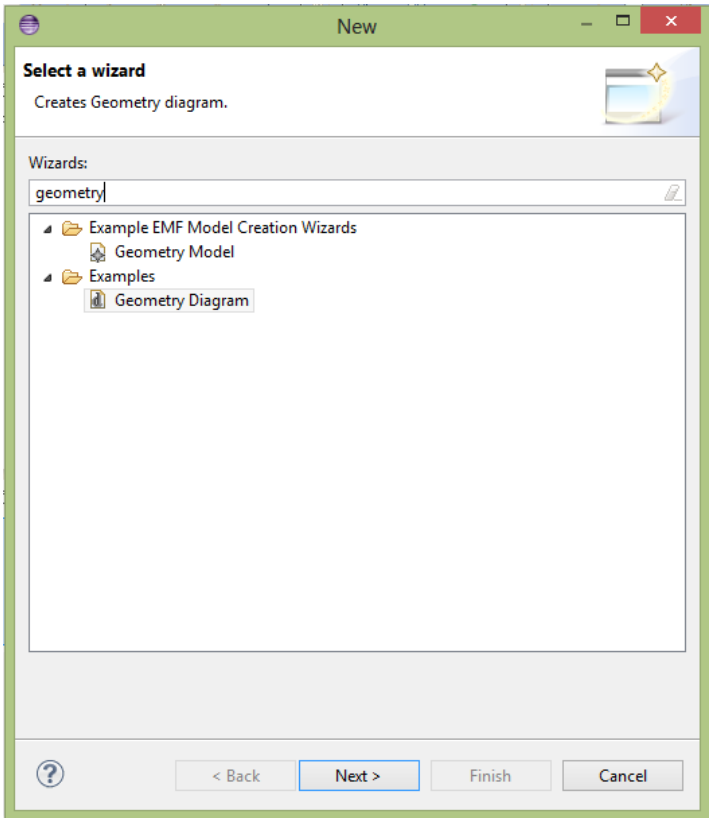

Figure 8: Create new Geometry Diagram

Figure 9 presents empty graphical geometry editor view. Right hand side of the view has tool

palette, which has all the available tools to be used. Rest of the view is taken by the canvas. On the geometry palette clicking Input point, or connector allows user to create different points on the canvas. These points can be also created by hovering mouse on canvas and a selector tool appears after a while. Input point and connector are described there by the symbols seen in geometry palette. Line tool allows drawing connections between connectors with each other. Line can also be created by hovering mouse over a connector, which presents arrows. By clicking and dragging the arrow the new line can be thus created. Bend points are generated by clicking a point within a line and then by dragging said point.

Figure 9: Geometry graphical editor - geometry palette on the right hand and canvas on the left side

Picture Figure 10 illustrates one example created with graphical geometry editor. Input points are represented by circles and connectors are represented by squares. A geometry object line is presented by a line in the canvas. The bend points of a line can be seen as filled squares. However, the bend points are only visible when the line is selected. The labels, appearance labels and token appearance labels are visible in the canvas next to their graphical visualization on the canvas. The labels have prescript in the same order, called Label, Visual and Token. The labels are used to connect geometry objects to petri net and to their 3D visualization.

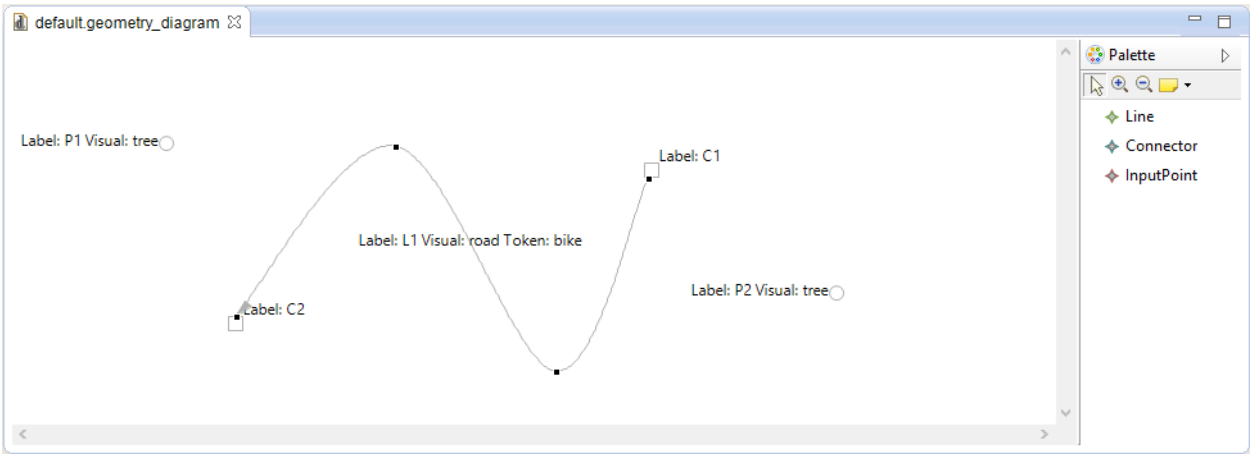

Figure 10: Geometry graphical editor with a geometry diagram example

By clicking some geometry object, a property view of said object opens. If this is not visible right click on a geometry object within the canvas and click Show Properties View to open property view.
The properties view of input point as seen in Figure 11 has following properties available for editing: Appearance Label, Label, XLocation and YLocation. Appearance Label denotes the appearance, such as a tree, a traffic light, a planet, etc, of the input point within the 3d simulator. Appearance label has to have some labeling in it. Label is the name used in referring the input point within the software. All the geometry objects must have different names so that references don't lead to multiple objects. XLocation and Ylocation describe coordinate location of the input point. However, editing them will not change the location input point in the canvas.

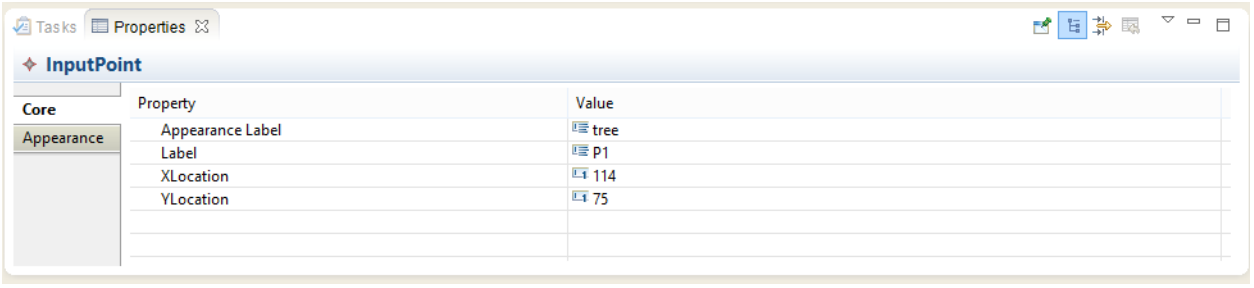

Figure 11: Properties view of an Input Point

The properties view of Connector as seen in Figure 12 has following properties available for editing: In, Label, Out, XLocation and YLocation. In refers to the lines that go to the connector. And Out refers to the lines that go from the connector. Label is the name used in referring the Connector within the software. All the geometry objects must have different names so that references don't lead to multiple objects (in particular, **geometry labels have to be unique** for each place**!**). XLocation and Ylocation describe coordinate location of the connector. However, editing them will not change the location connector in the canvas.

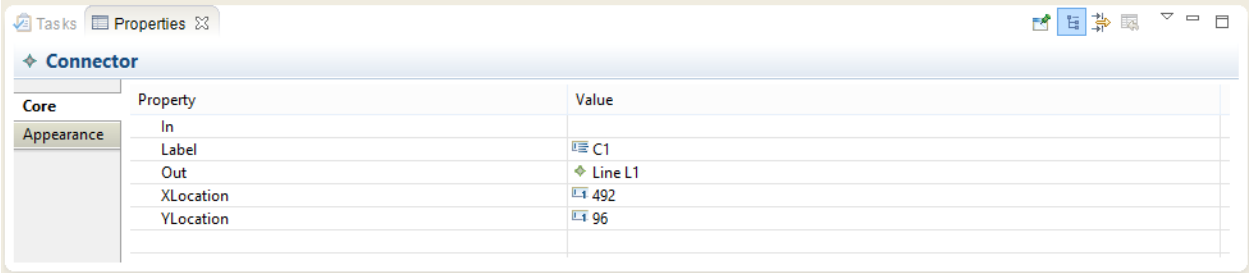

Figure 12: Properties view of a Connector

The properties view of Line as seen in Figure 13 has following properties available for editing: Appearance Label, Begin, End, and Label. Appearance Label denotes to the appearance, such as a cable, a track, empty space, etc, of the line within 3d simulator. The appearance will extrapolated over the curve of the line, thus creating continuous shape. Begin and End denotes the connector of the line, which is the starting and ending point respectively of the parametric curve that is line geometry. Label is the name used in referring the line within the software. All the geometry objects must have different names so that references don't lead to multiple objects.

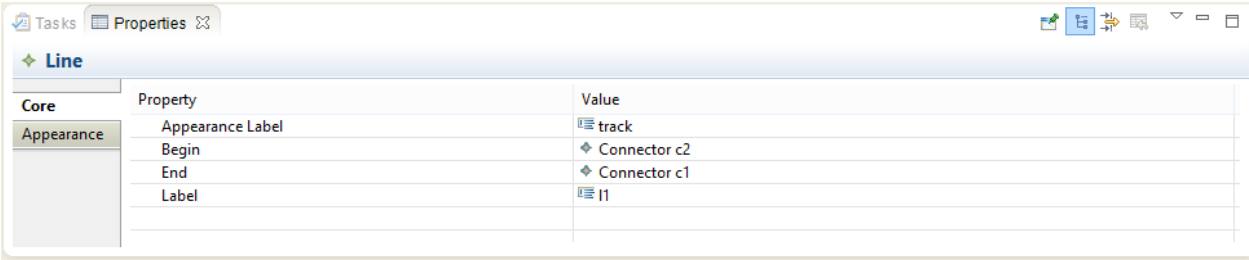

Figure 13: Properties view of a Line

To validate that the geometry doesn't have any duplicate Labels, the geometry should be opened in the tree view. Right click on the tree view window and choose the Validate option. If there are duplicate labels, an error message will pop up. The tree view of the geometry used as an example in this part of handbook can be seen in Figure 14.

Figure 14: Tree view of geometry.

Appearance editor

Author: Morten

In the Geometry editor the user has specified an appearance label for each object. To specify the appearance of the geometry objects in the 3D visualization we have to create a new Appearance model, where the user specifies the model and texture for each appearance label. The necessary steps are described below:

1. Select the project created before in the Project Explorer and right click on it to select "**New > Other**".
2. In the new Wizard window type the name Appearance and select *Appearance Model* and then click "**Next**".

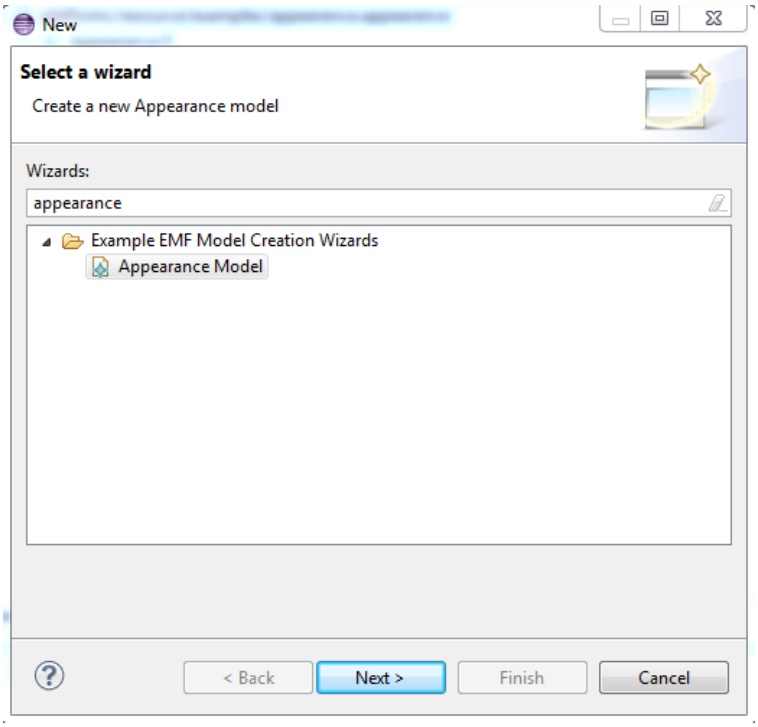

Figure 15: Create new Appearance model file

3. Once again make sure that the previously created project is selected as the parent folder (if it is not, manually select it from the list of projects). Give a proper name to the Appearance Model file (keeping the *.appearance* extension) and then click "**Finish**".
4. Open the newly created appearance file, click on the small arrow to the left of the file and select the "Appearance" node. **Right click** on the node and select "**New Child > AObject**".

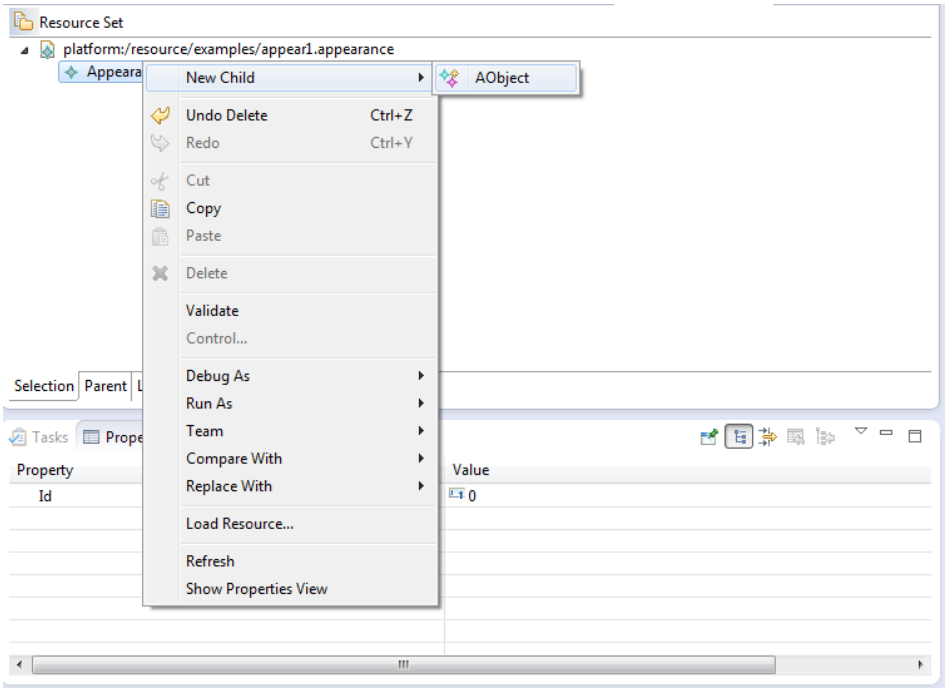

Figure 16: Create a new AObject.

5. Once you have created a new AObject, select it and open the Properties view. Name the label of the AObject accordingly to the appearance label of the Geometry object(s), to create/edit the appearance of the object(s) with this label.

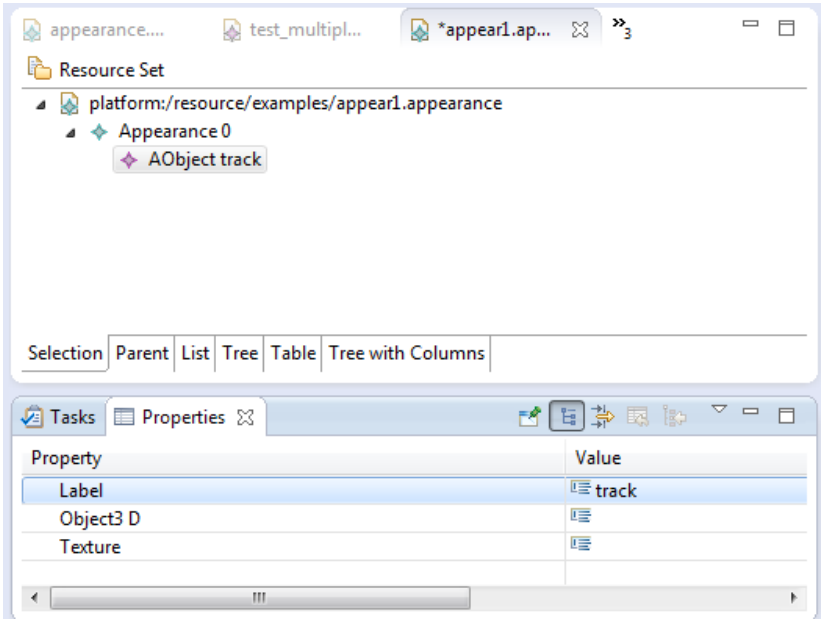

Figure 17: Change the label of the AObject to the same as the appearance label of the geometry object(s) in order to change the appearance accordingly.

6. **IMPORTANT**
   Due to limitations of the jMonkey Engine models and textures have to be placed inside

the “assets” folder in the “jMonkeyEclipse” folder of your workspace in order for jMonkey to recognize them. Example:
Textures **have** to be placed in:
C:\Users\user\Documents\my_workspace\jMonkeyEclipse\assets\Textures
Models **have** to be placed in:
C:\Users\user\Documents\my_workspace\jMonkeyEclipse\assets\Models
Remember to Refresh your workspace in Eclipse, when you have added new files.

7. To add a model to the AObject click on the “…” button in the Object3D property in the Properties view. A file dialog will open, locate your model file inside the “assets/Models” folder and press Open.

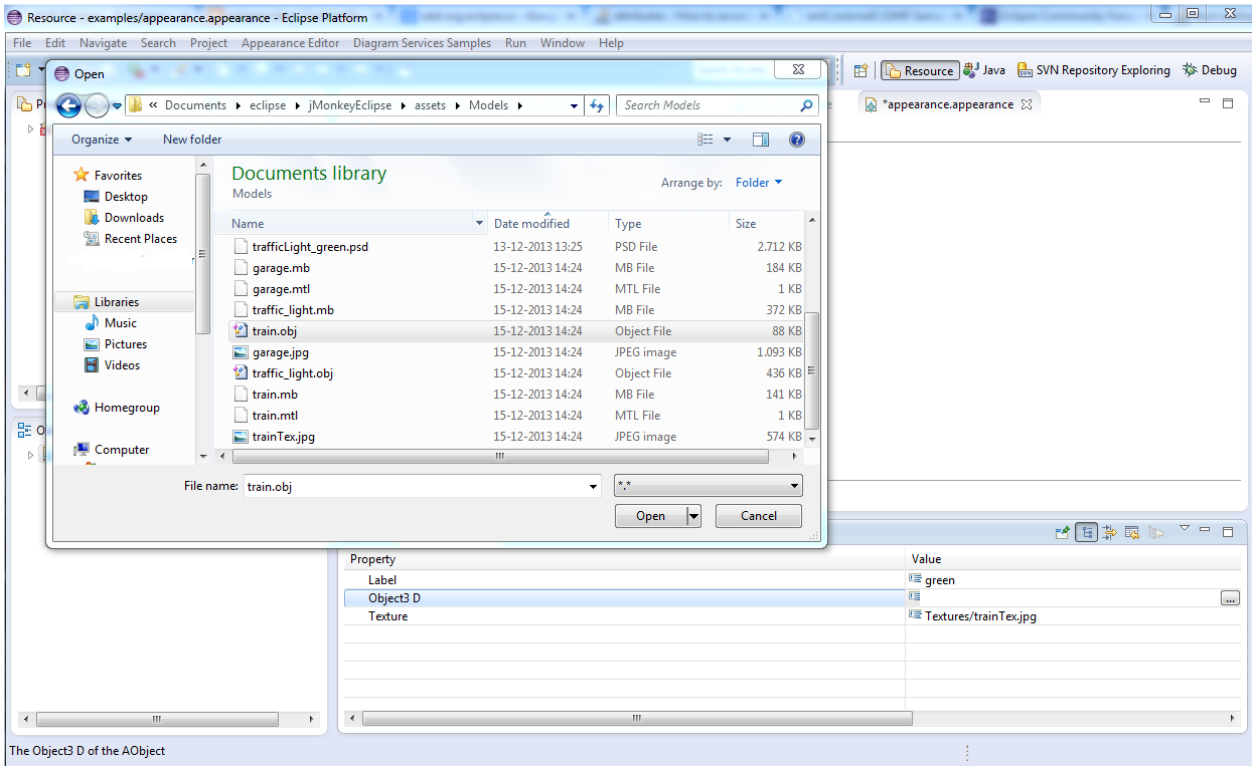

Figure 18: Add your own 3D objects.

8. To add a model to the AObject click on the “…” button in the Texture property in the Properties view. A file dialog will open, locate your model file inside the “assets/Textures” folder and press Open.
9. Repeat the steps 4 through 8 for all the appearance labels until you have an AObject for each appearance label.

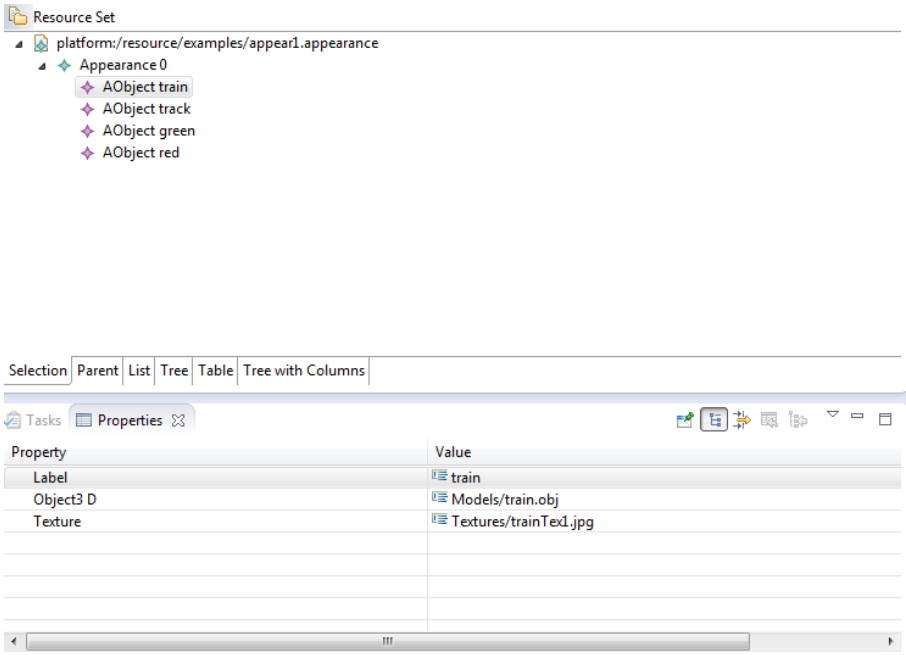

Figure 19

Configuration editor

Author: Monica

Once all the models, the *Petri net* model, the *Geometry* model and the *Appearance* model have been configured, they need to be connected in the Configuration editor. The necessary steps are described below:

1. Select the project created before in the Project Explorer and right click on it to select “New > Other”.

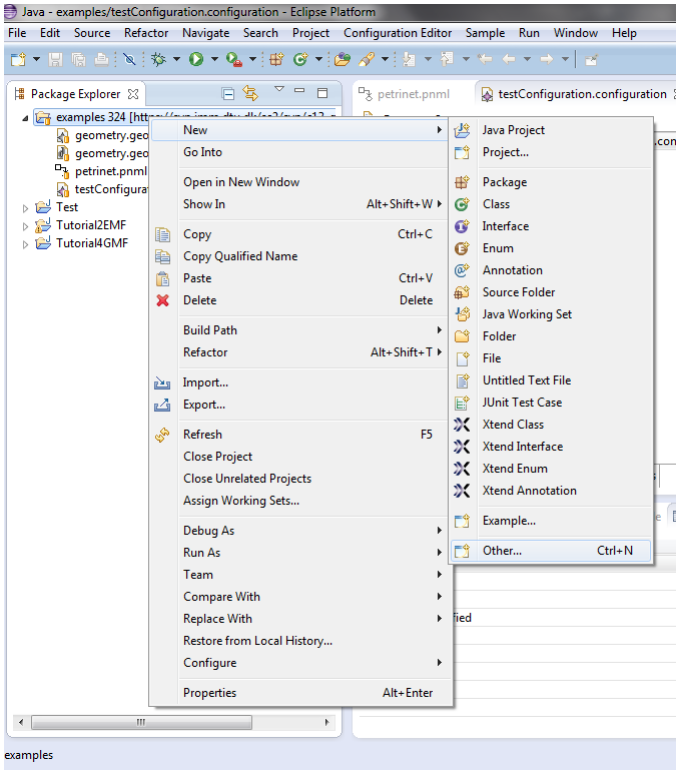

Figure 20: Create new file

2. In the new Wizard window type the name Configuration, select *Configuration Model* and then click “Next”.

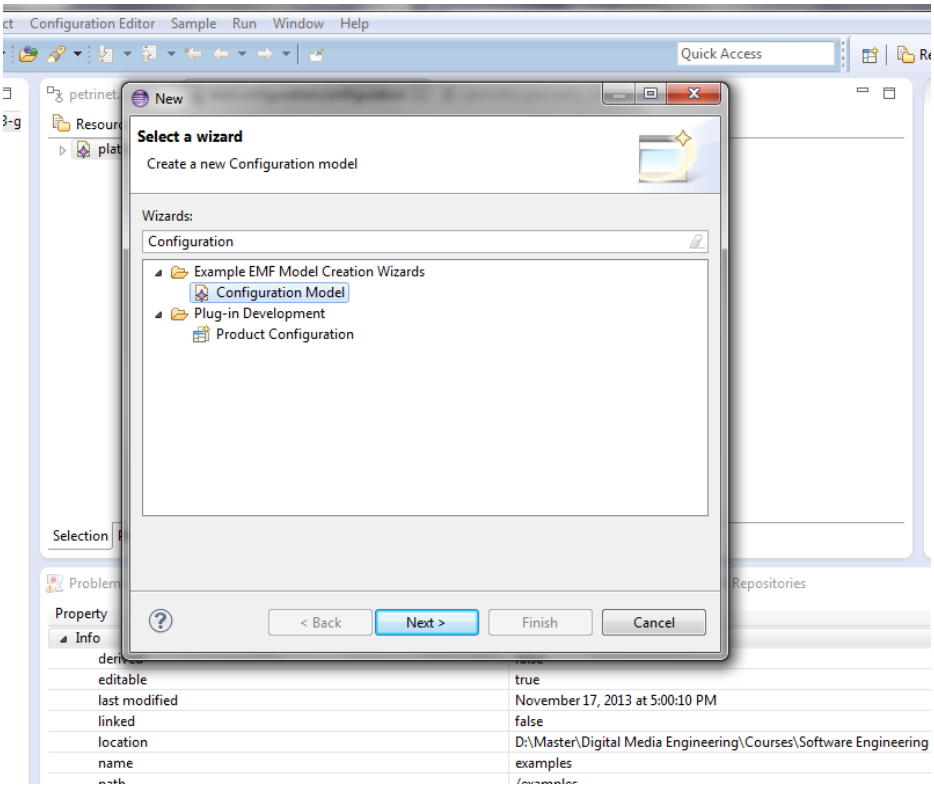

Figure 21: Choose Configuration Model type of file

3. Once again make sure that the previously created project is selected as the parent folder (if it is not, manually select it from the list of projects). Give a proper name to the Configuration Model file (keeping the *.configurator* extension) and then click “Finish”.

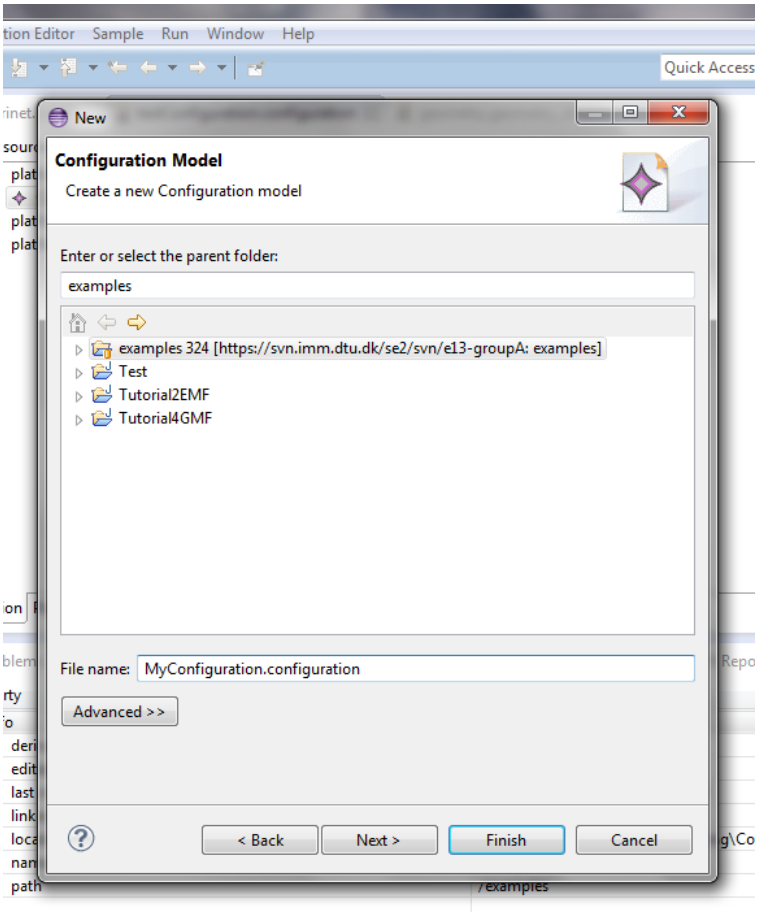

Figure 22: Rename the configuration file

4. Open the newly created file, click on the small arrow to the left of the file and select the “Configuration” node. Right click on the node and select “Load Resource…”.

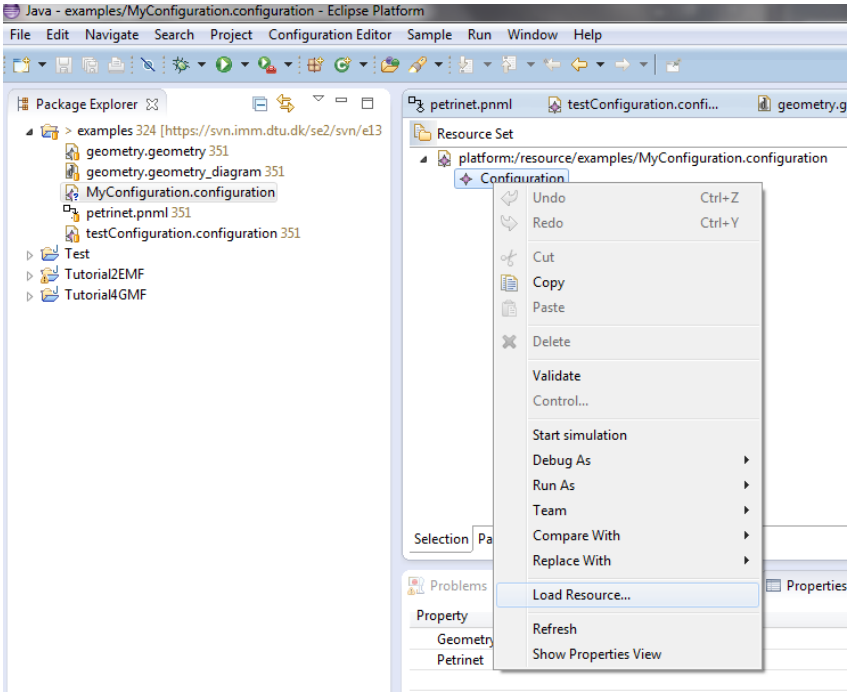

Figure 23: Load resources

5. A new window will appear where you can select the files that need to be connected. Click “Browse Workspace…”, look for your project and select the *.pnml* file previously created. Click OK twice to confirm your choice.

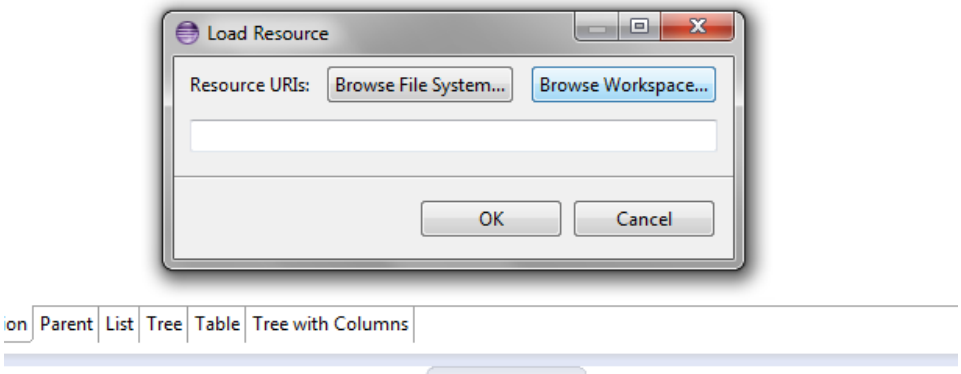

Figure 24: Browse workspace for resources

6. Repeat the previous two steps to also add the *.geometry* and *.appearance* files.

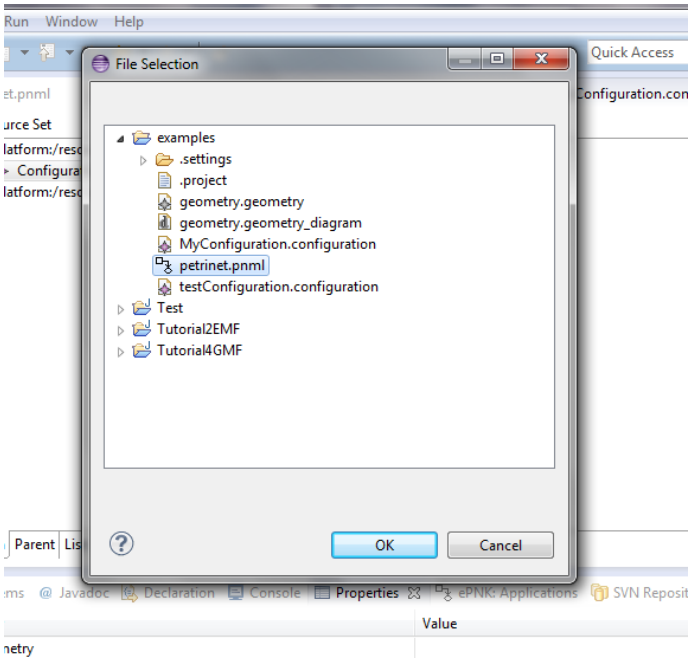

Figure 25: Select Petri net model

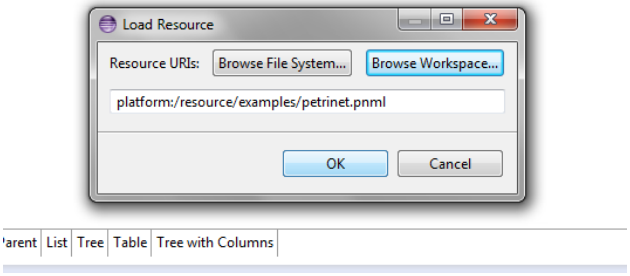

Figure 26: Load Petri net file

7. Once the three files are loaded as resources, they need to be attached to the Configuration. To do this, you need to open the Properties View of the Configuration object and edit its three properties by selecting the model corresponding to each property (in our case, *Geometry* for the geometry property and *Petri Net Doc* for the Petrinet property). If the Properties View is not open you can find it in the top Eclipse menu “Window > Show view > Other... ”. In the Show View window you type Properties as search filter, select the Properties view and click OK.

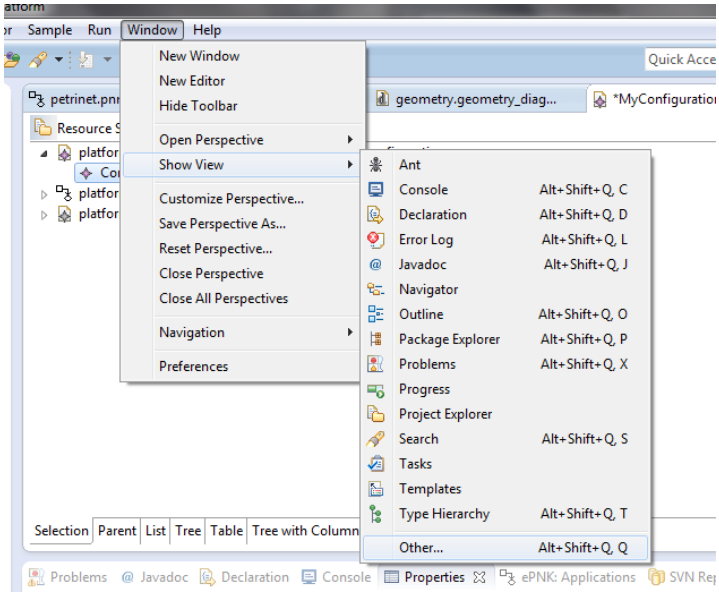

Figure 27: Open view

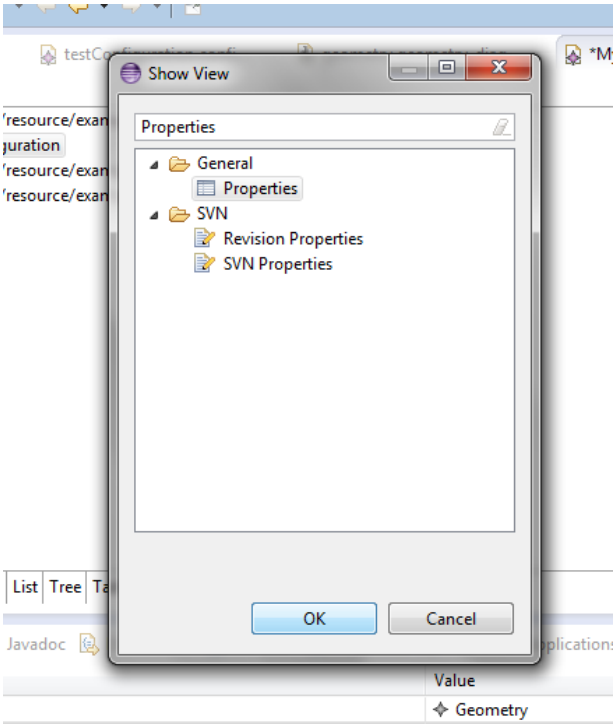

Figure 28: Select Properties view

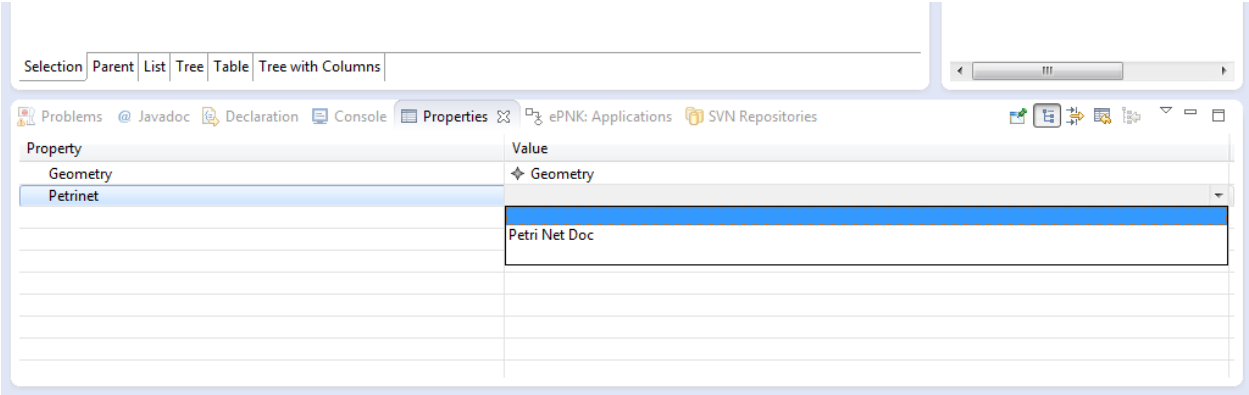

Figure 29: Attach resources to configuration

8. The Configuration is now set up so in order to start the simulation you must select the Configuration node and click “Start simulation”.

Simulator

The simulator is launched by right clicking on the *configuration* and selecting *run simulation*, as illustrated in figure 30.

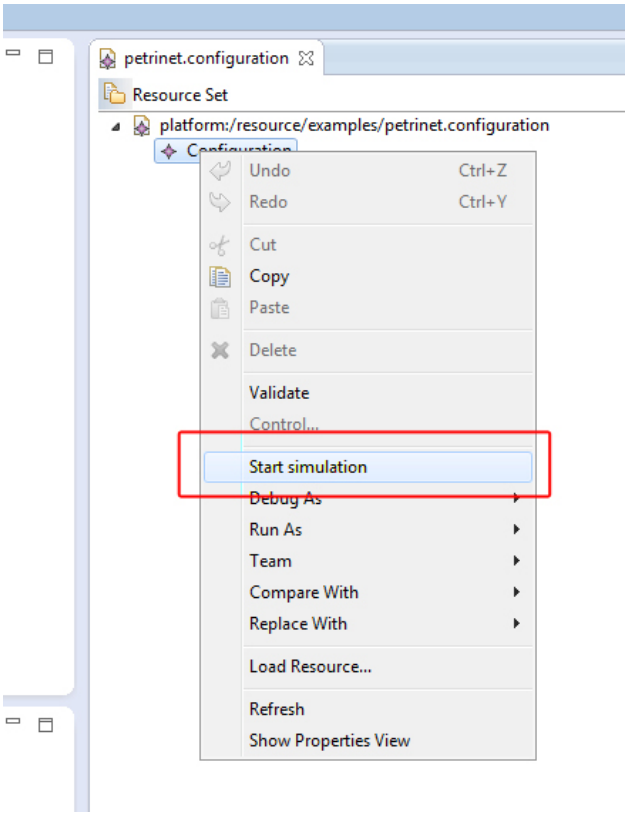

Figure 30: Start simulation

The simulator is paused when it is launched, so nothing will be moving. To start the simulation, use the mouse to press the *start button* on the screen. Notice that the button on the screen will change into a *pause button*. This means that the simulation will be paused, if this button is pressed again. Next to the *play/pause button* is a *reset button*, which will reset the simulation to the initial configuration. See figure 31.

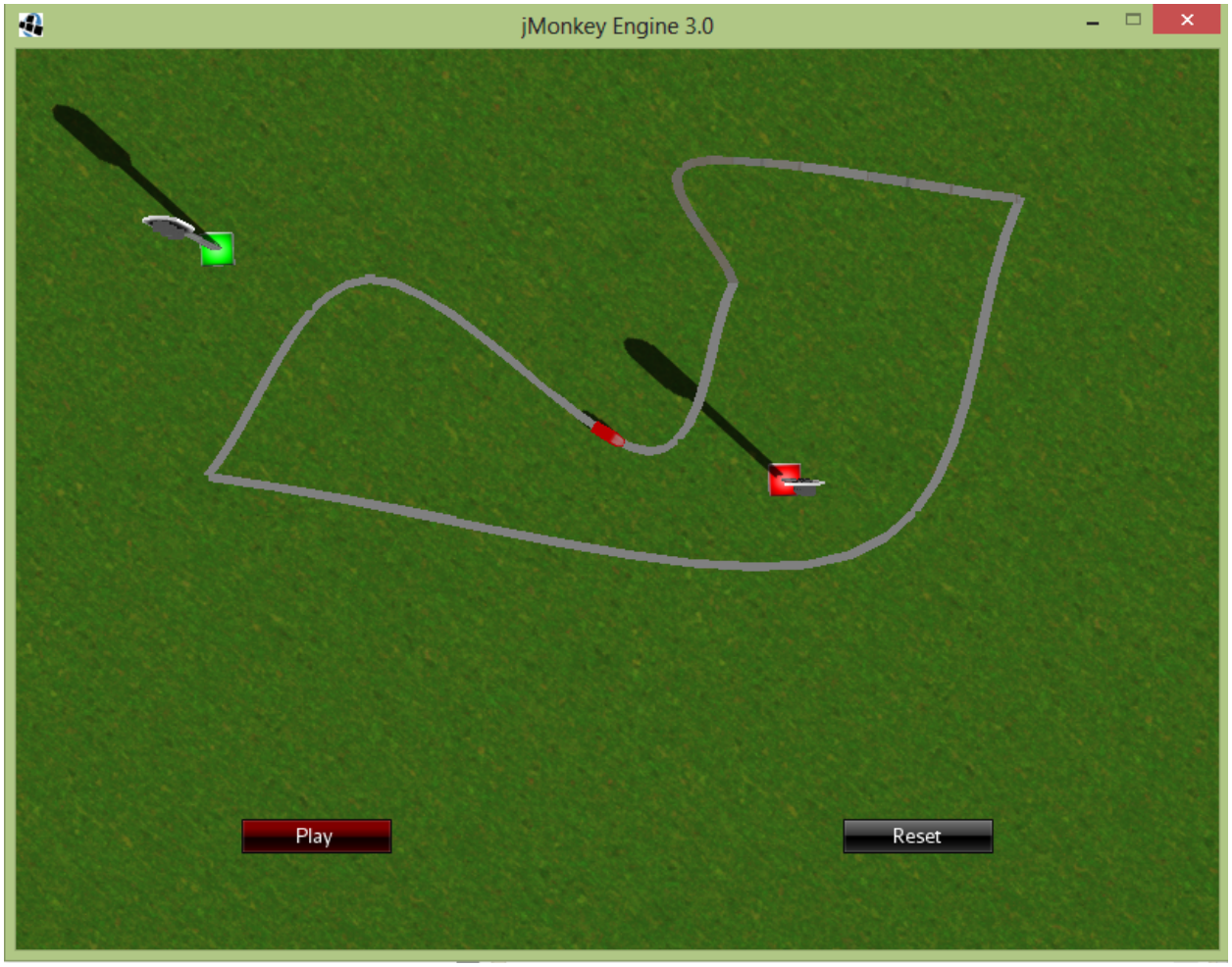

Figure 31: Simulation, default camera position.

To control the camera, press and hold the right mouse button. The camera will pan and tilt with the mouse movement. To translate the camera forward, backwards and sideways use the keyboard keys ‘w’, ‘s’, ‘a’ and ‘d’. When the right mouse button is not pressed, the left mouse button can be used to press the objects that represent input points. To exit the simulator, press ‘esc’. See figure 32.

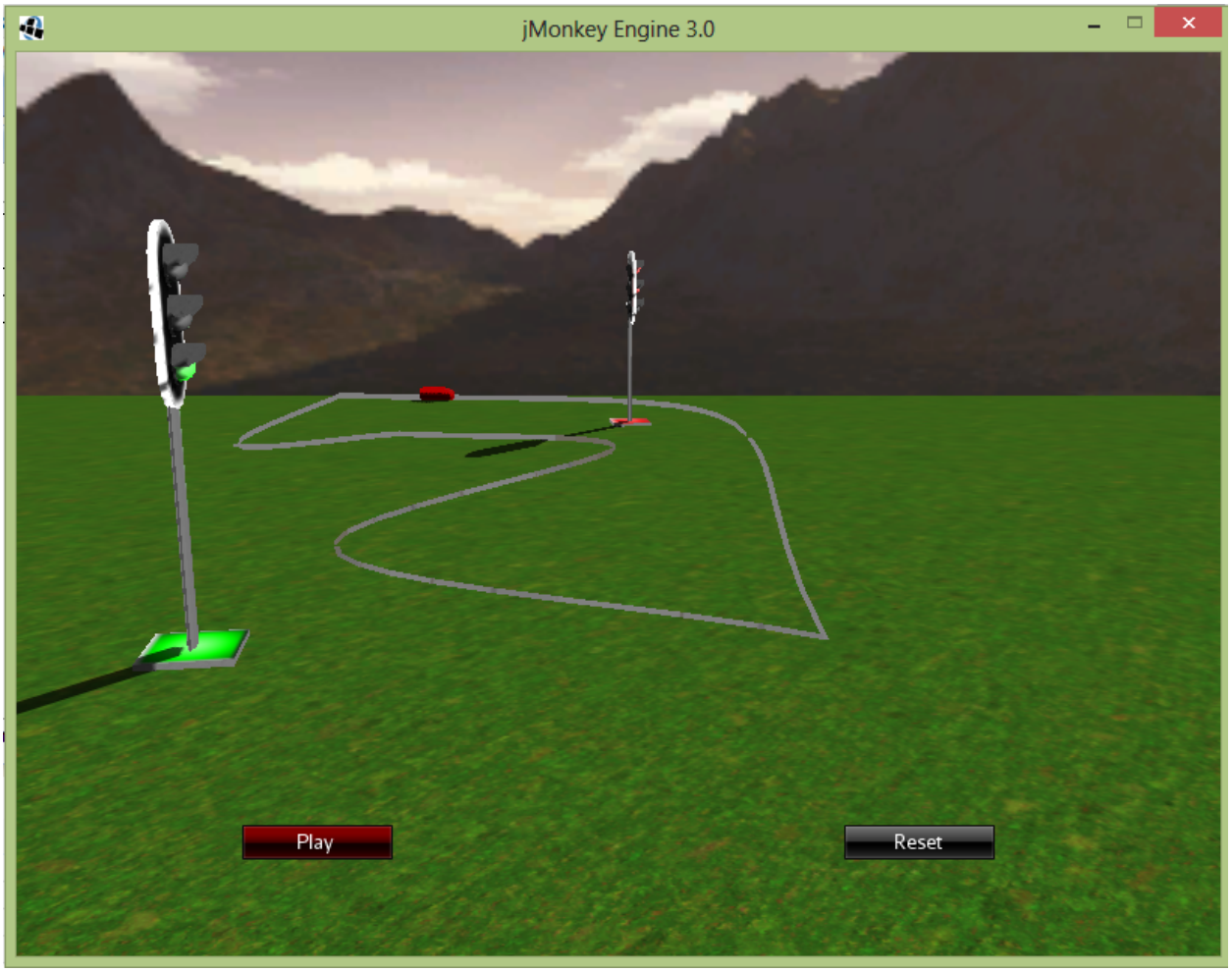

Figure 32: Simulator, camera position set by user.