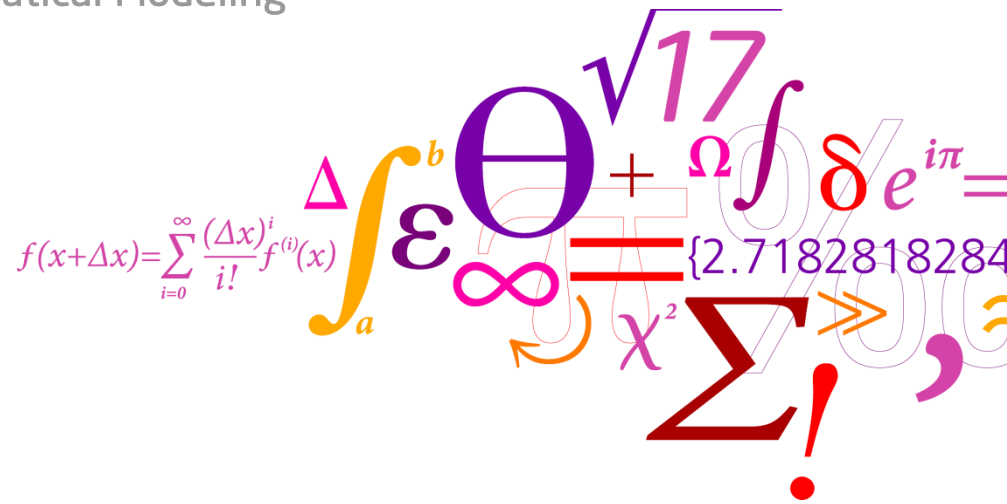
Software Engineering 2
Project presentation

Ekkart Kindler

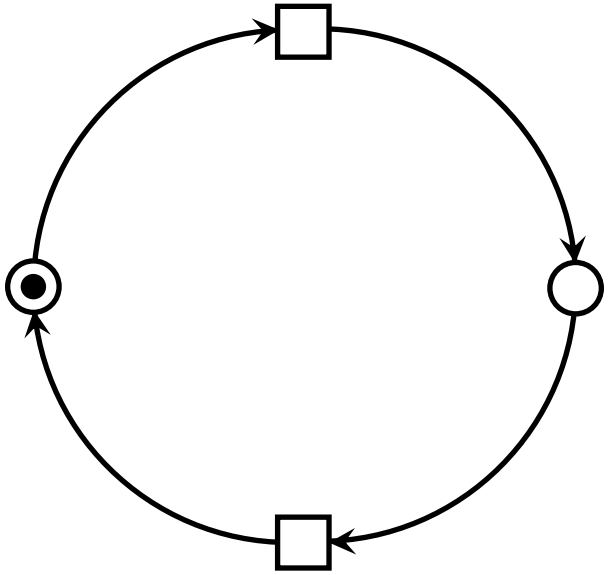
DTU Informatics
Department of Informatics and Mathematical Modeling



Project in a nutshell

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Petri net



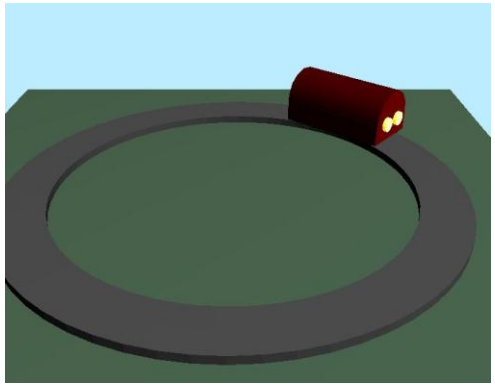
+

Animation info

shape: train
animation: move
geometry: track1
...

«3D models»

...



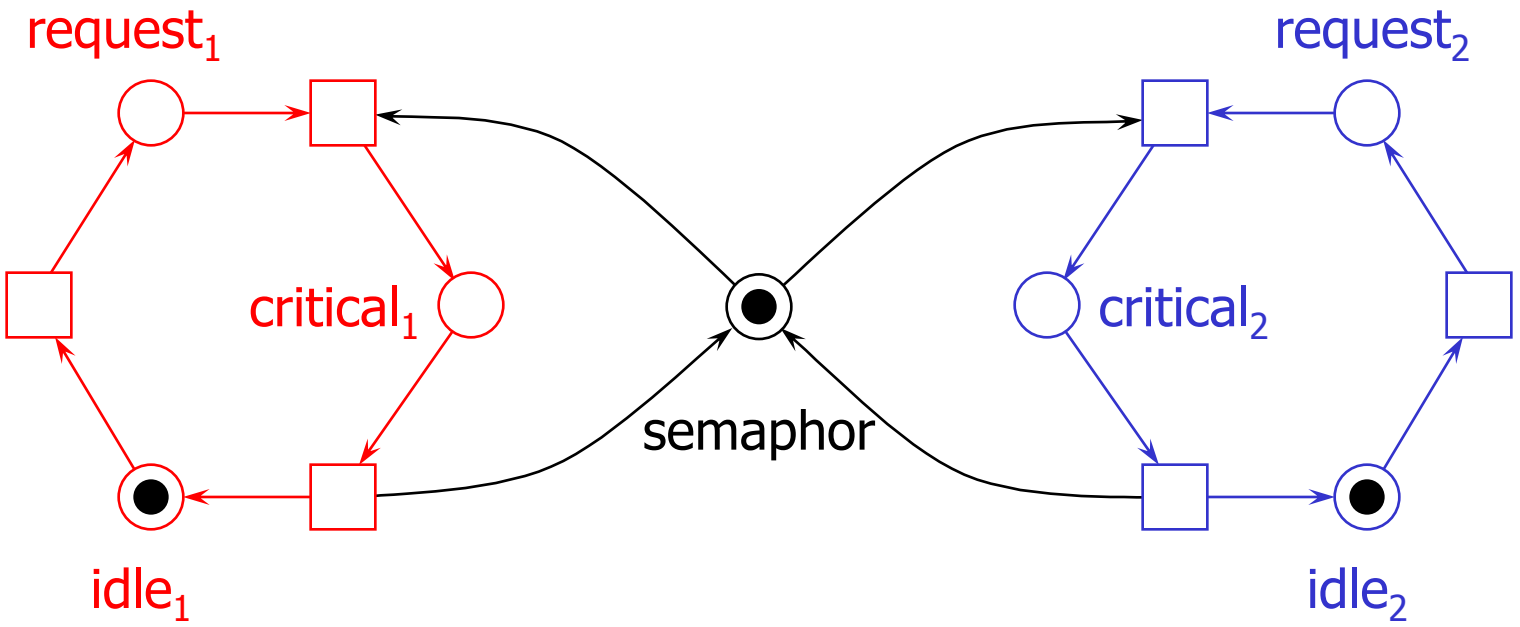
Simple interactive 3D animation of a system

Outline of talk

- What are Petri nets?
- What do we need to add for animating behaviour in 3D?
- Some more detailed concepts!
- More detailed requirements!

Petri nets: Example

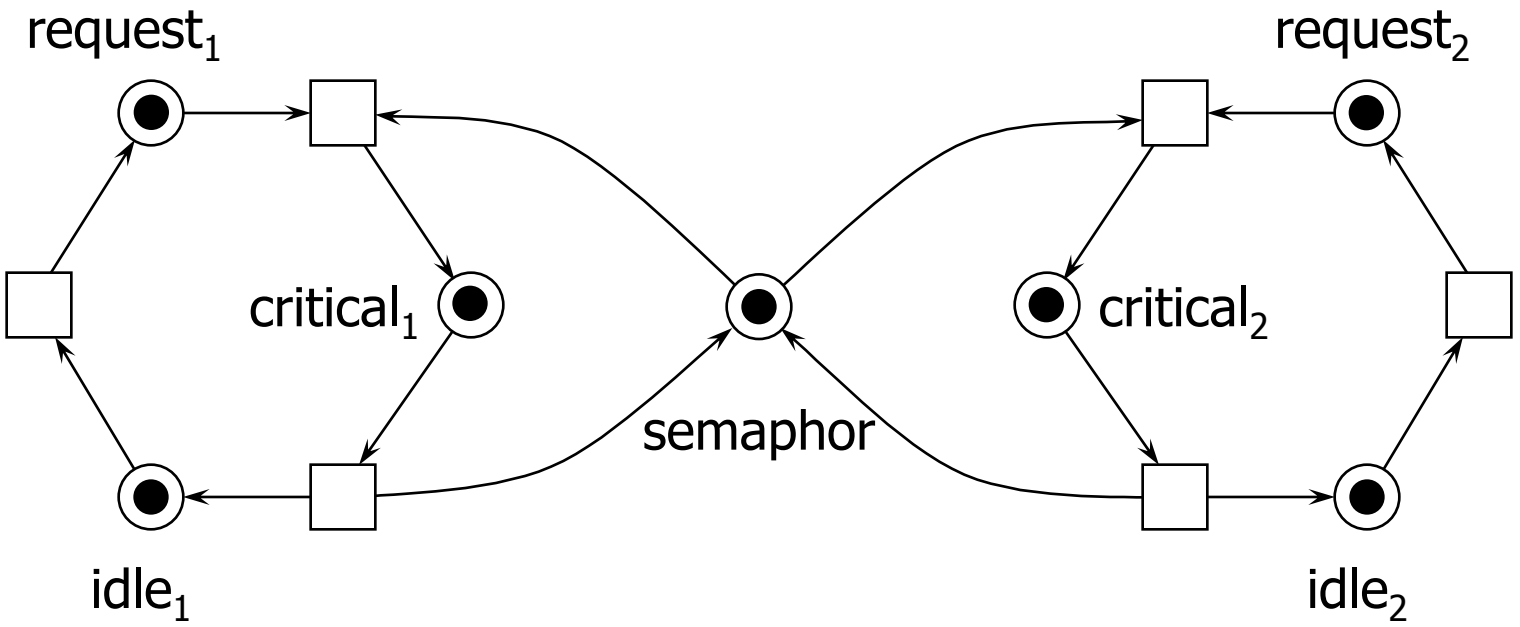
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Note that we consider a very simple version of Petri nets only: Place/Transition Nets (P/T nets).

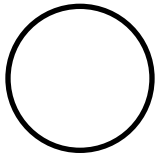
Petri nets: Example

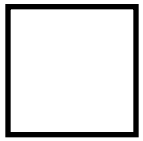
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Petri nets: Concepts

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Places: 

Transitions: 

Tokens: 

Arcs: 

Arcs run from a place to a transition or from a transition to a place only!

Marking: A distribution of tokens on the places

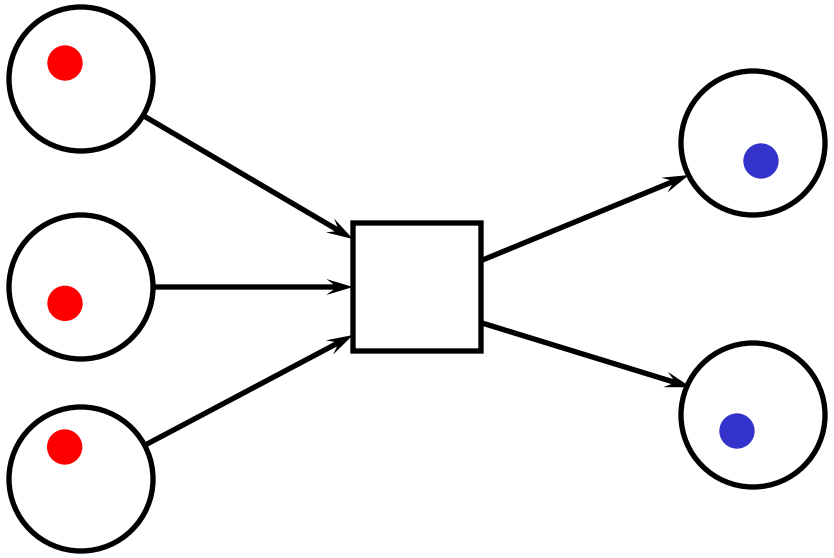
(there may be more than one token on a place)

Petri nets: Firing rule

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

before (must be there and are consumed)

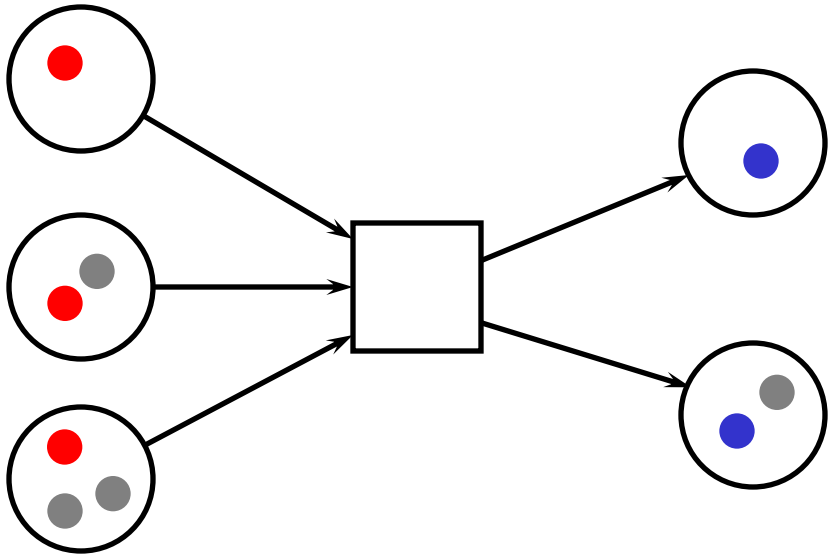
after (are produced)



Firing rule

before (must be there and are consumed)

after (are produced)

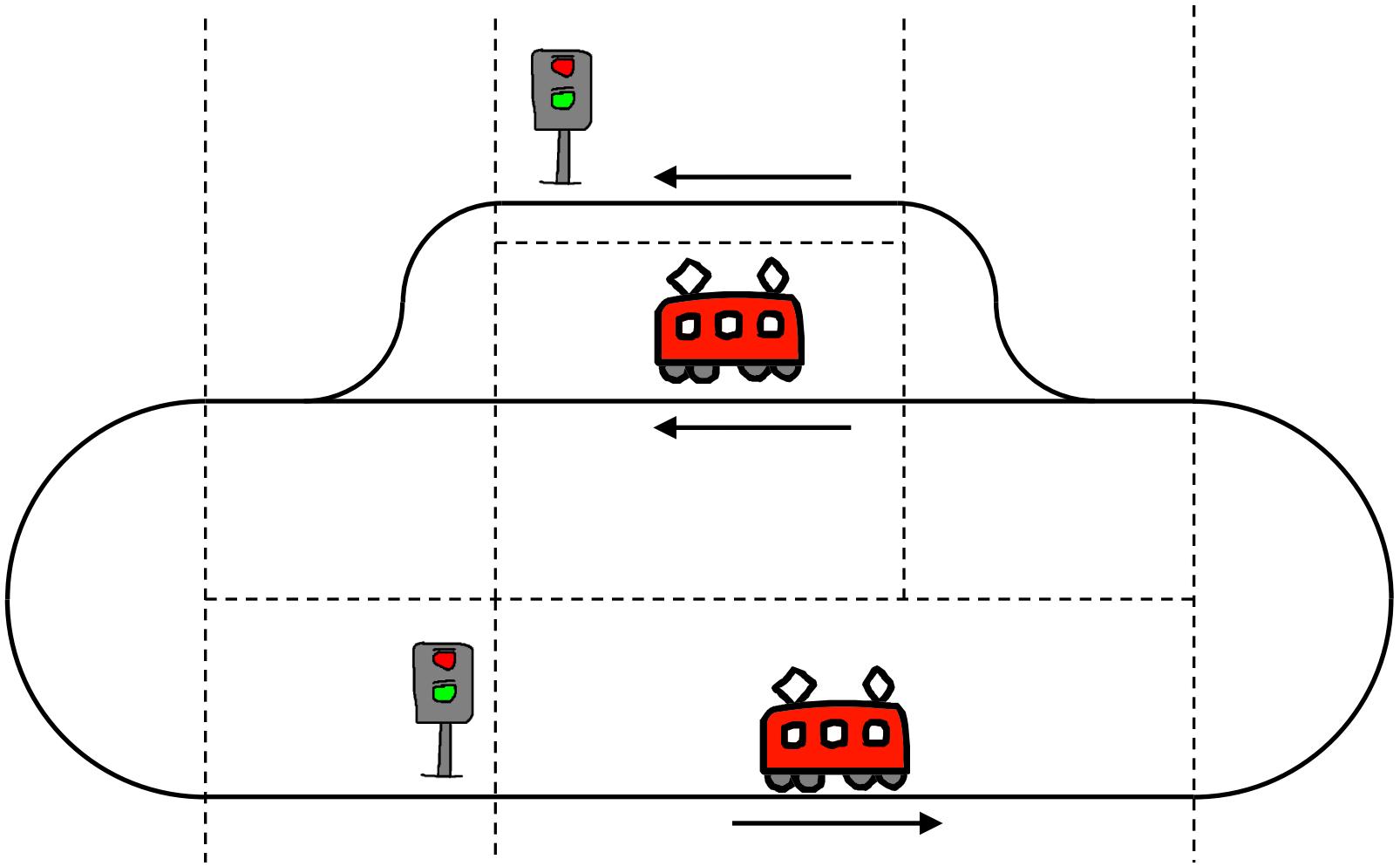


Other tokens might be there (do not change)

Different transitions might be able to fire; the choice is non-deterministic.

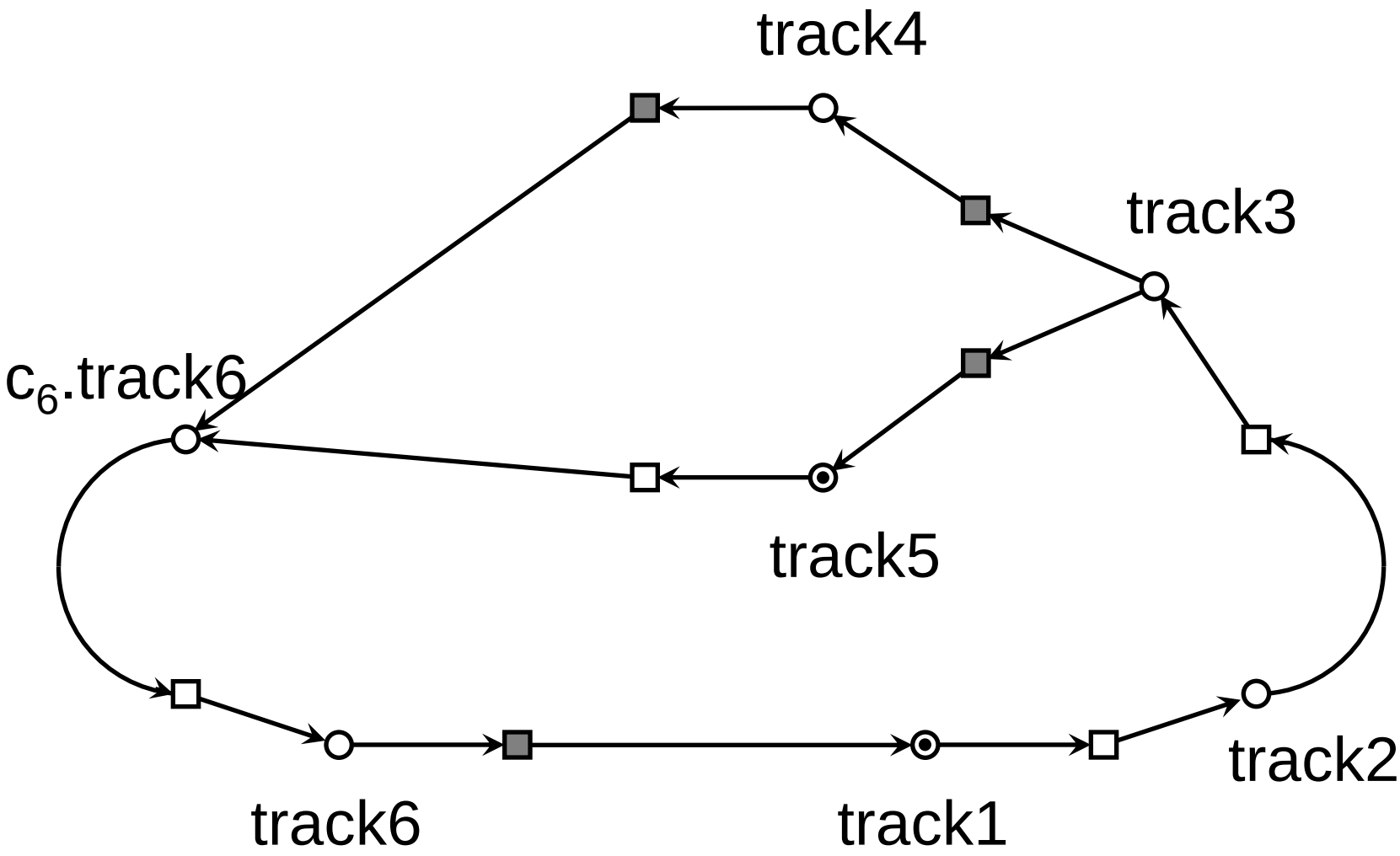
Example: Toy train

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



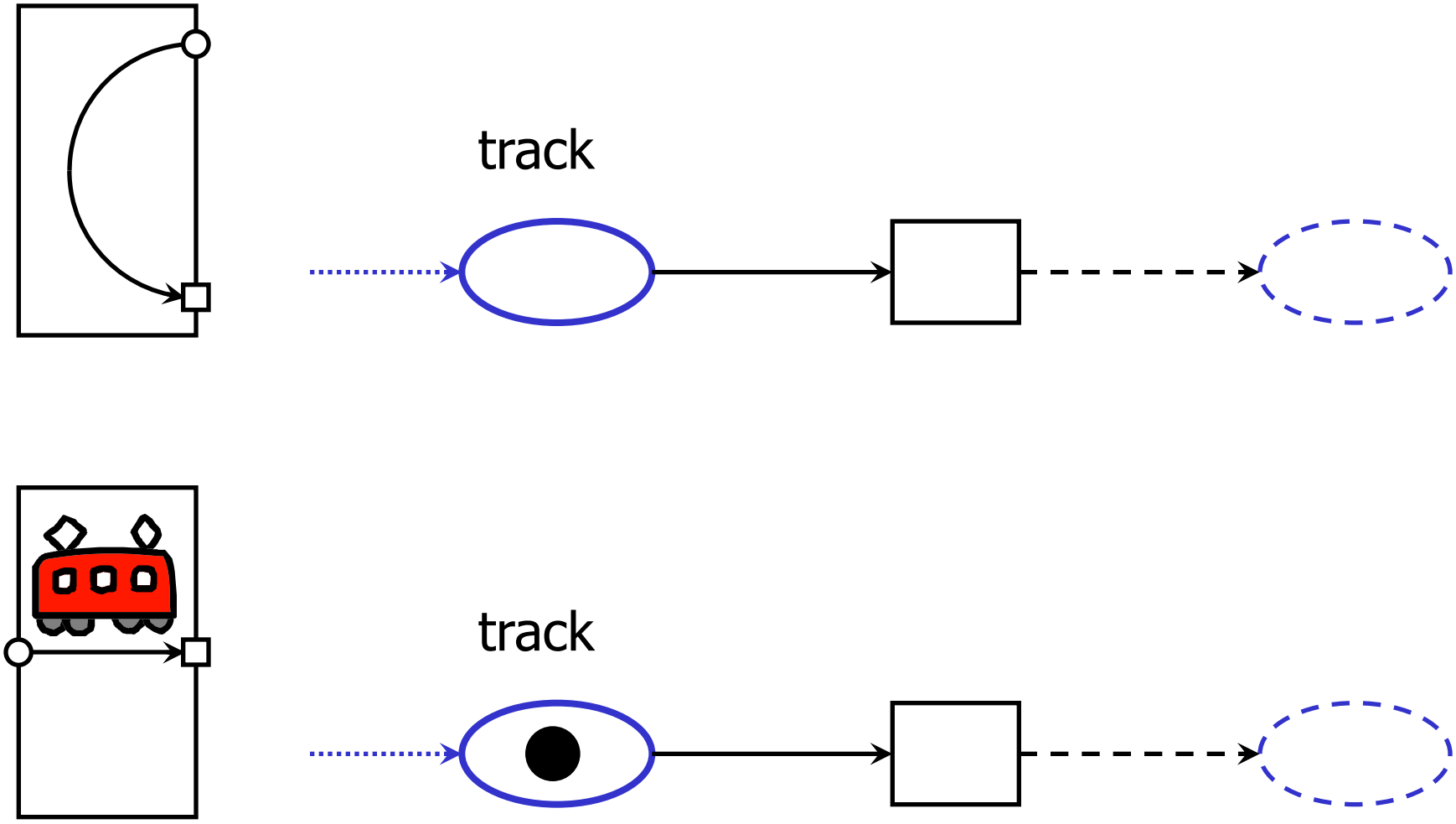
Toy train: Simple net

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



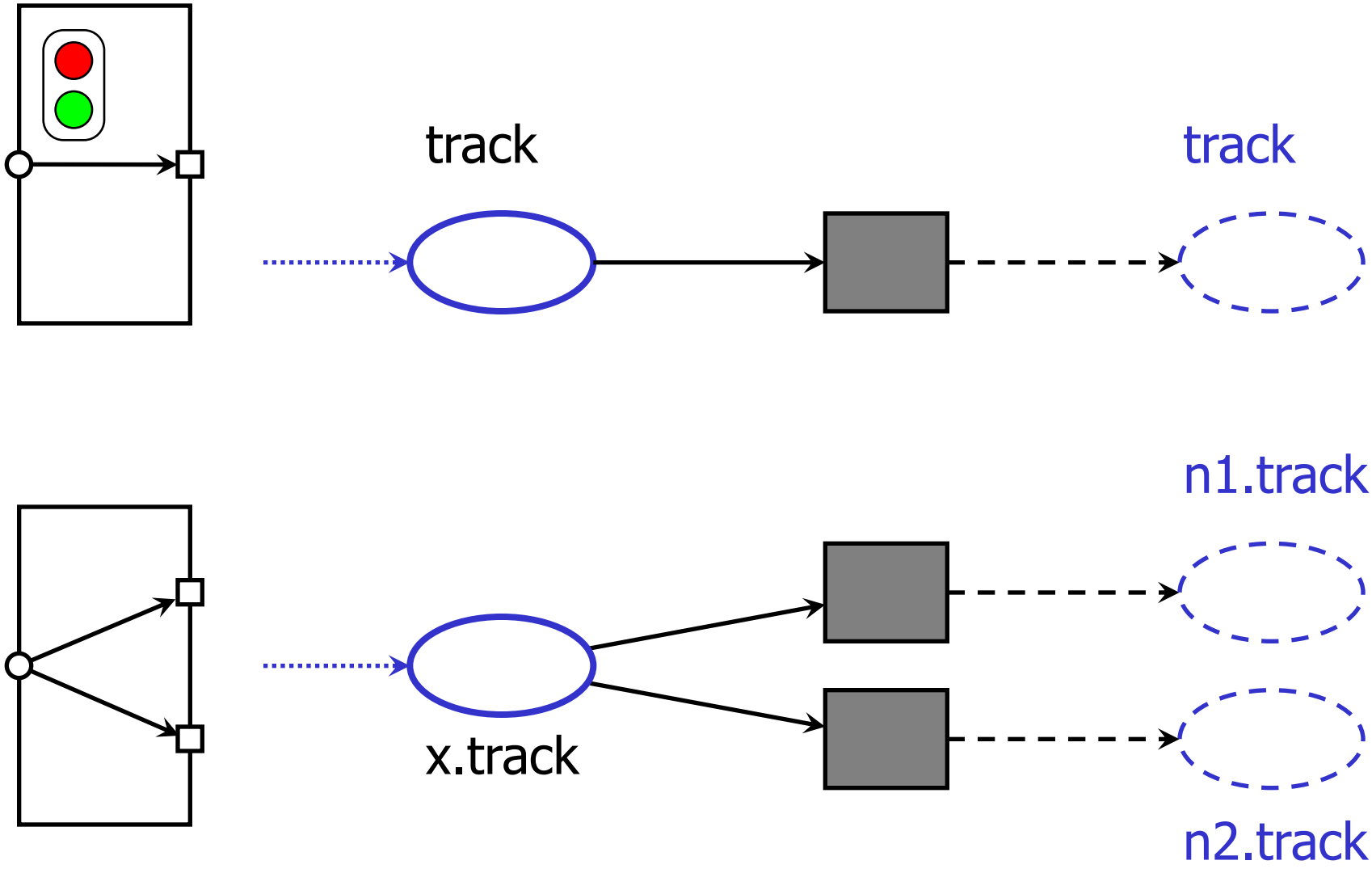
Models for components

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



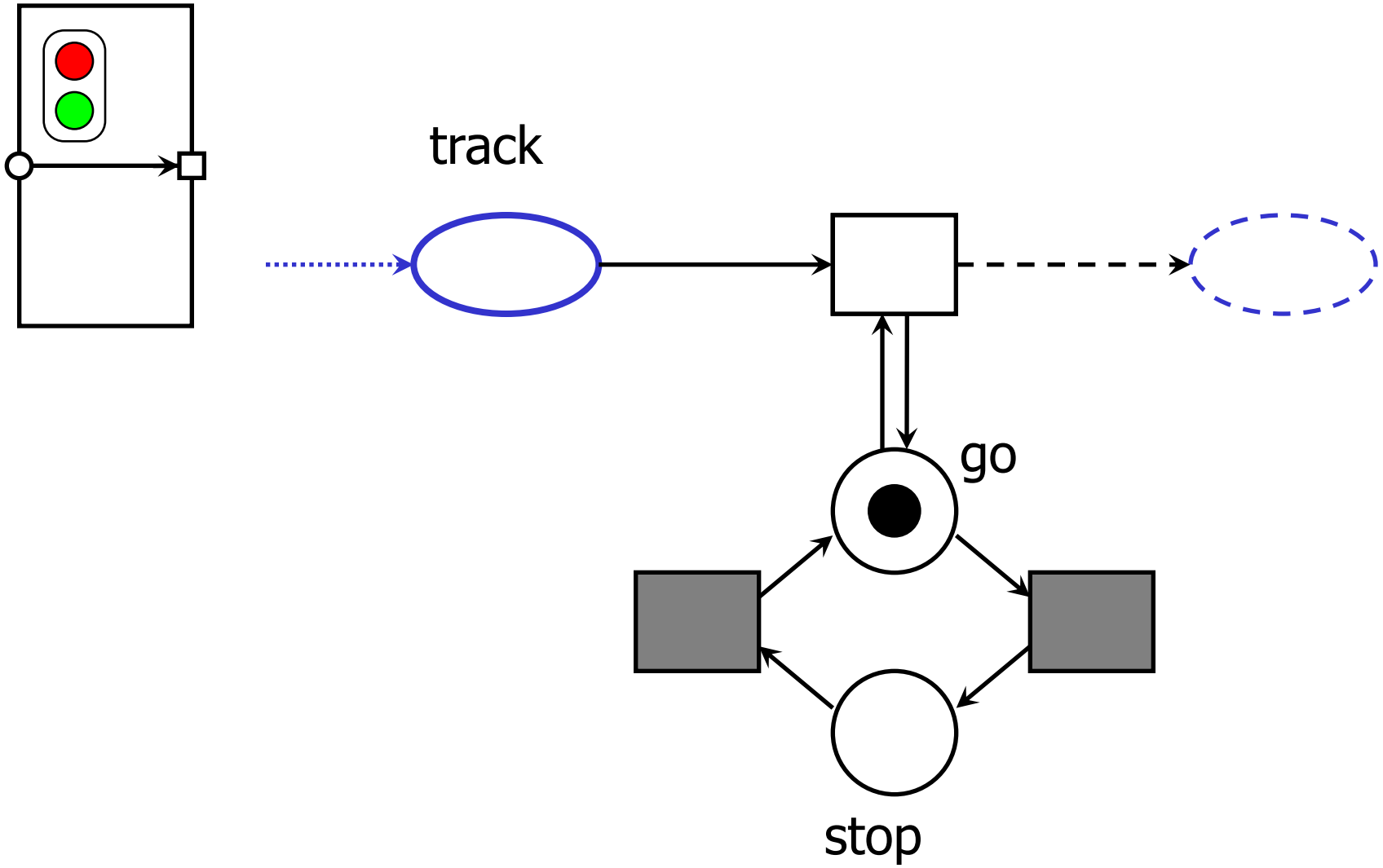
Models for components

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



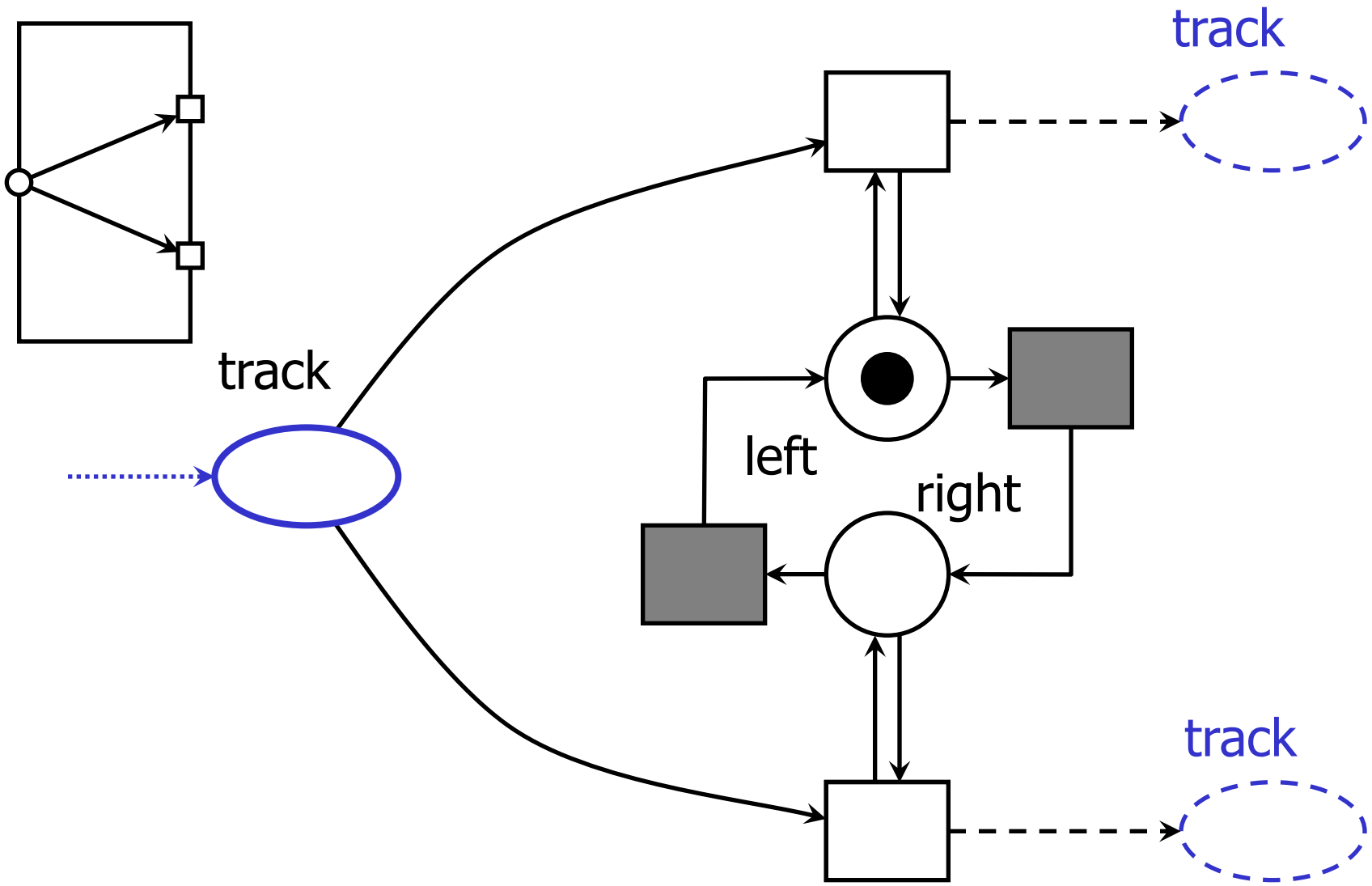
Signal: Detailed model

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Switch: Detailed model

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Outline of talk

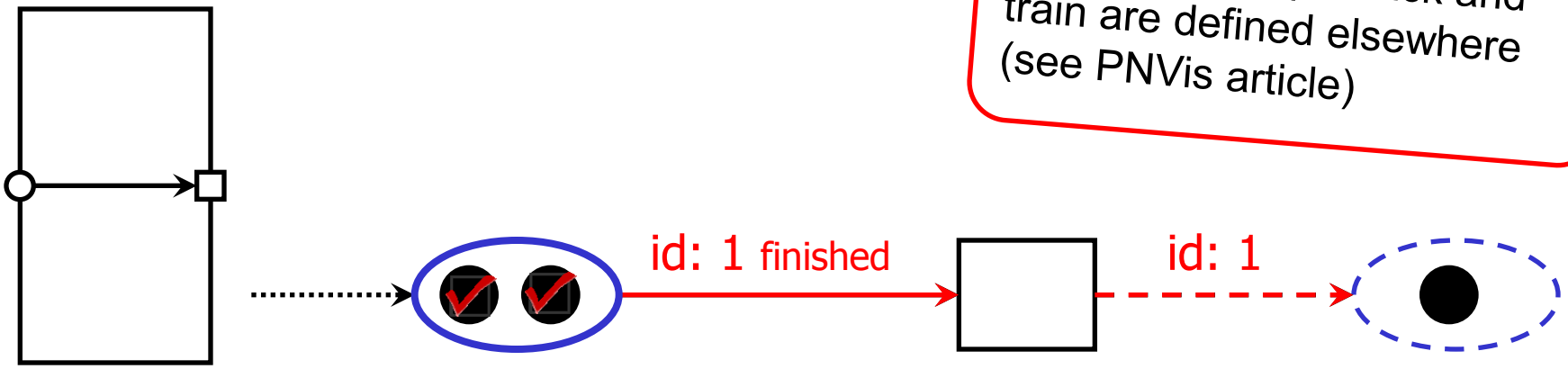
- What are Petri nets?
- What do we need to add for animating behaviour in 3D?
- Some more detailed concepts!
- More detailed requirements!

Animation info

- Geometric information
  - arrangement of tracks (geometry)
- Physical appearance
  - appearance of objects (3D model/shape)
  - appearance of tracks (mostly texture)
- Animations
  - “Behaviour” of a token while on a place

Animation information

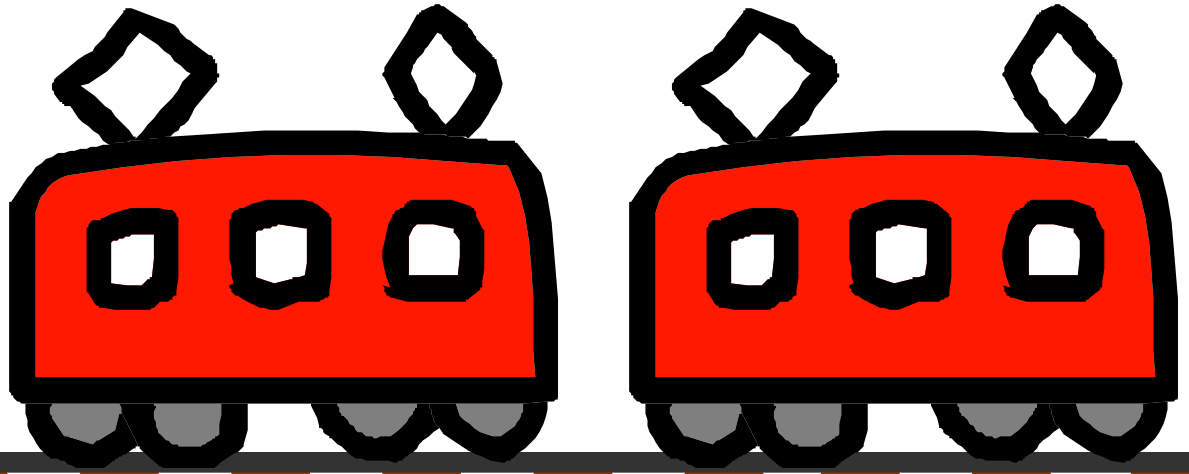
The actual shape track and train are defined elsewhere (see PNVis article)




```
geometry:
...

  line:
    id = track1
    x1, y1, x2, y2
    shape: track
...
```

shape: train
animation: move
geometry: track1



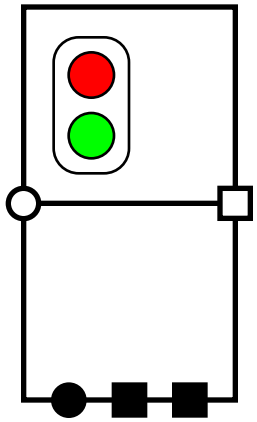
Outline of talk

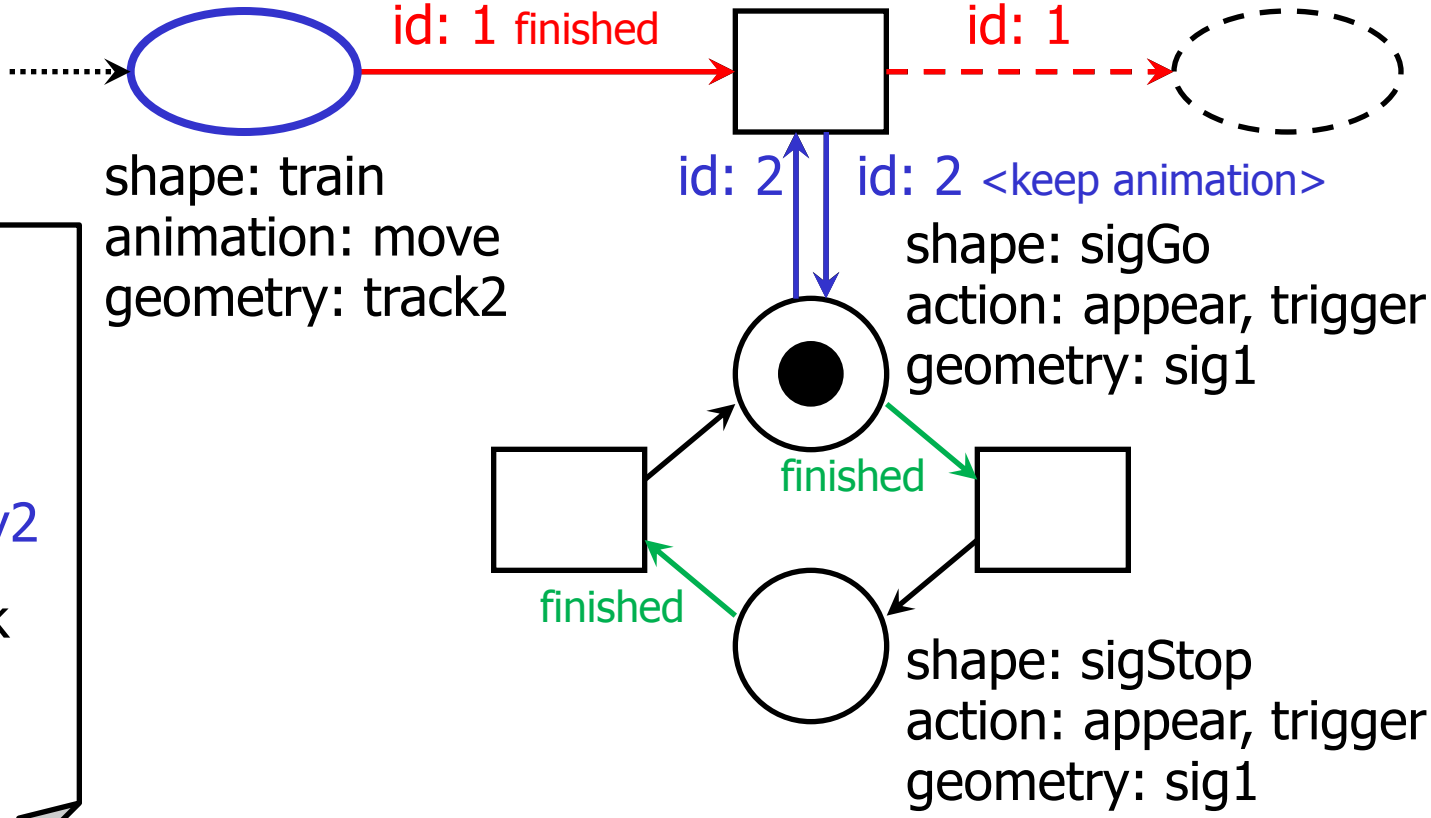
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- What are Petri nets?
- What do we need to add for animating behaviour in 3D?
- Some more detailed concepts!
- More detailed requirements!

Animation information

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler





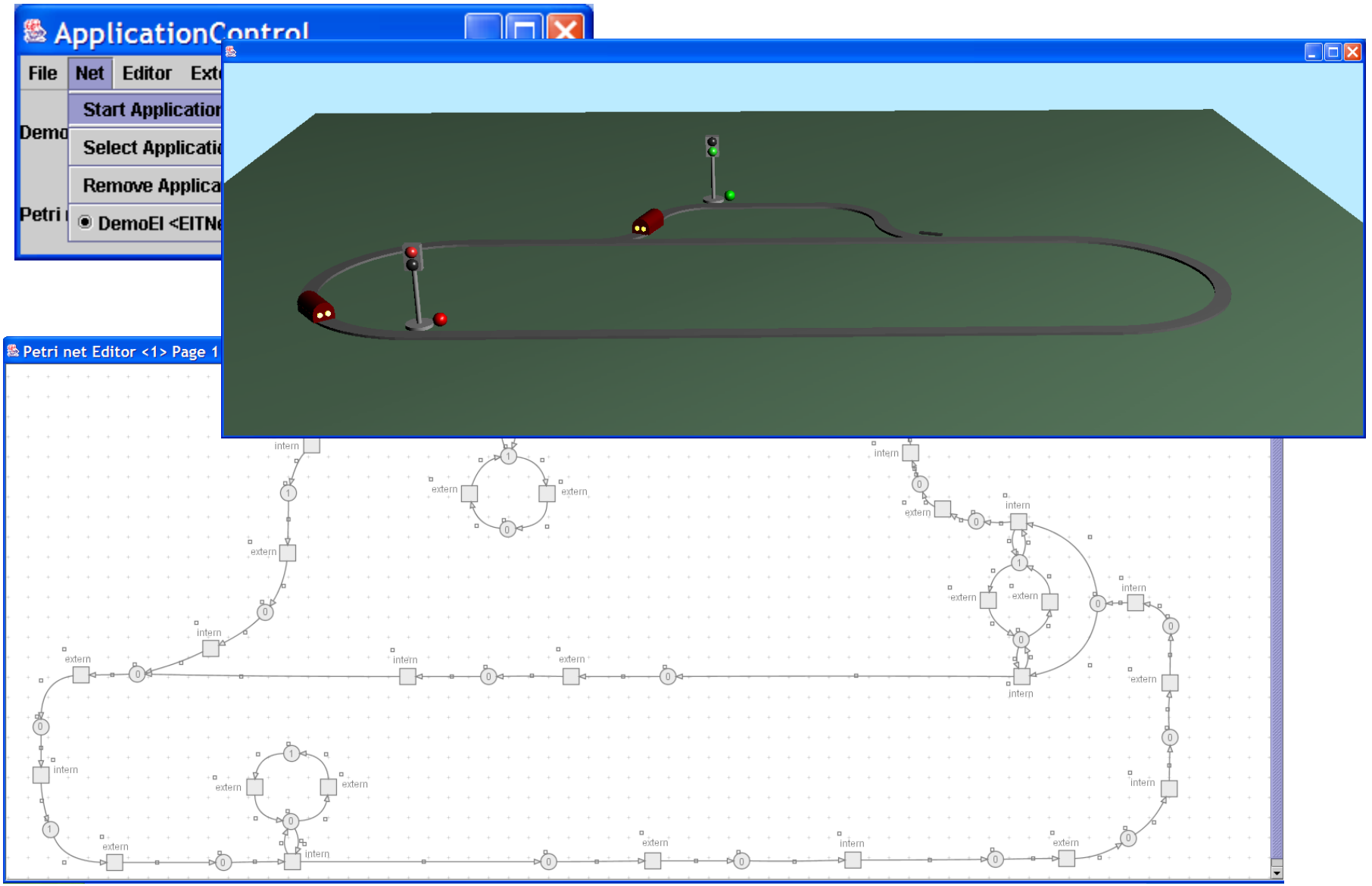

```
geometry:
...
  line:
    id = track2
    x1, y1, x2, y2

    shape: track

  point
    id = sig1
    x3, y3
```

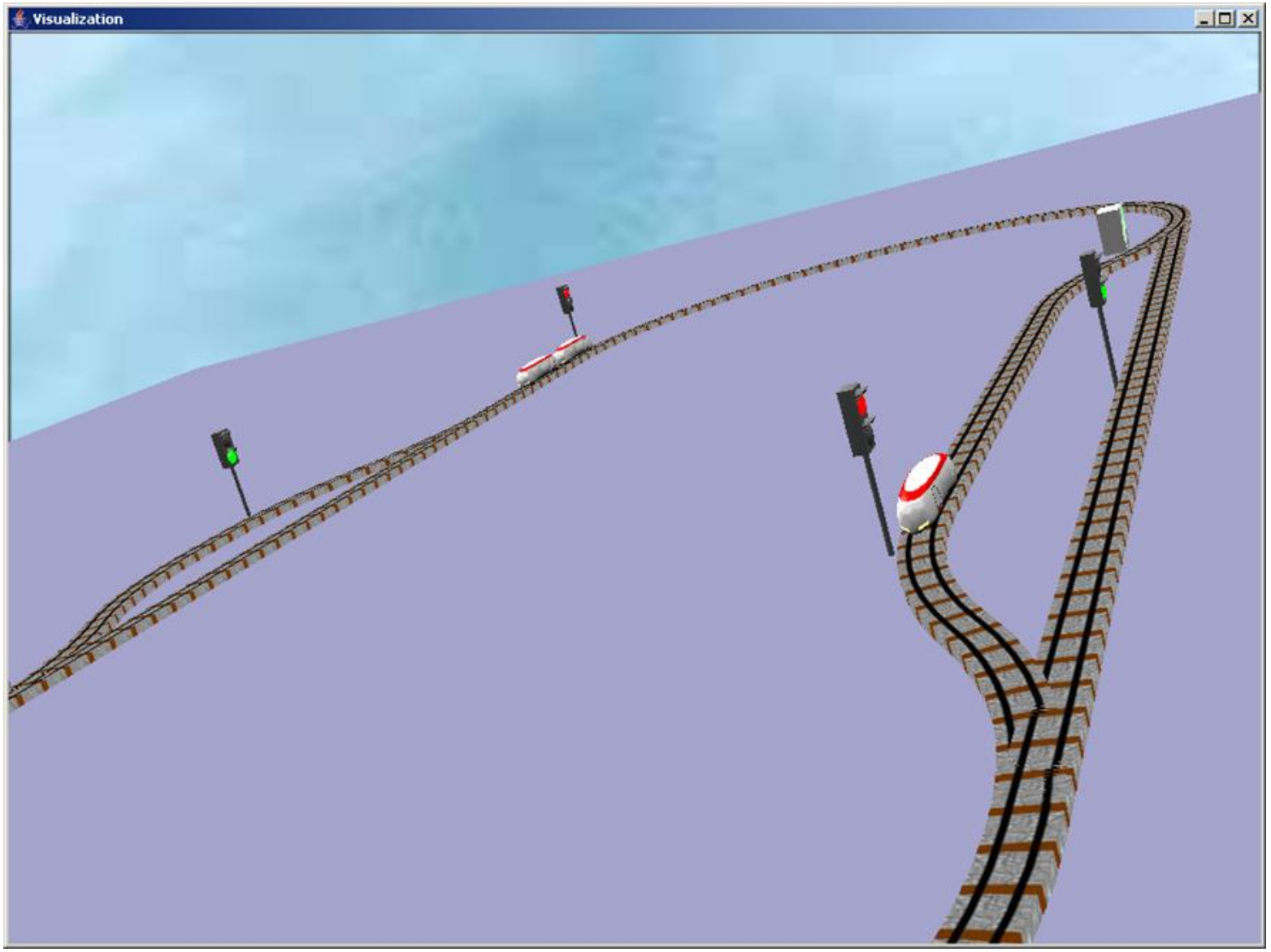
Screenshot

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



More fancy version

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Which shape?

- How do the following shapes look like?
  - train
  - sigGo
  - sigStop

  } Shapes corresponding to tokens: dynamic shapes

  - track

  } Shapes corresponding to geometry objects: static shapes

- The appearance of each shape is defined in a separate models file
  - a reference to a VRML model for a dynamic shape (you are free to use other models)
  - a profile and a texture for a static shape

Objective

- Simple extension for equipping a Petri net model with a 3D-visualization
- Cheap way of showing a customer what a system modelled as a Petri net would really do – for validation purposes

Task

Re-implementation of PNVis based on a new model-based Petri net tool (the ePNK) and with more modern development technologies (EMF)

Outline of talk

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- What are Petri nets?
- What do we need to add for animating behaviour in 3D?
- Some more detailed concepts!
- More detailed requirements! (will be continued)

Sub tasks

- **Extended Petri net type** for the ePNK that covers the extensions that are needed for the animations
- **Graphical editor for geometries** (points and lines the ”Petri net animations” refers to)
- **Editor for defining the appearance** of objects and tracks (refering to external 3D-models and textures)
- **Simulator** for the extended Petri net type that interacts with the 3D animation engine
- **3D animation engine** that interacts with the Petri net simulation (and with the end user)
- **GUI** for starting and **controlling 3D animations** from a simple **configuration file**

Further information

- SE2 project page:
  http://www2.imm.dtu.dk/courses/02162/e13/project/

- Ekkart Kindler and Csaba Páles: 3D-Visualization of Petri Net Models. In: J. Cortadella and W. Reisig (eds.): ICATPN 2004, LNCS 3099, pp. 464–473, Springer 2004:
  http://www2.imm.dtu.dk/courses/02162/e13/project/PDF/PNVis-PN04.pdf

- ePNK home page:
  http://www2.imm.dtu.dk/~eki/projects/ePNK/