

Concept Paper – Train Manager Module (TMM)

Unfinished Draft (step 0021c)

1 List of Contents and Figures

Table of Contents

1 List of Contents and Figures.....	1
2 Already Published in Concept Paper – Basismodul.....	1
3 Introduction.....	3
3.1 Results of the "Rollercoaster"-Project (2008).....	3
3.2 Modeling the Track Layout.....	3
3.3 Class Diagram of Tracks and Turnouts.....	5
3.4 Vehicles.....	5
3.4.1 Class Diagram for Vehicles.....	6
3.4.2 Specific Vehicle Prototypes.....	6
3.4.2.1 SrrWagon2axA.....	7
3.5 Trains.....	7
4 The Example Track Geometry.....	8
4.1 SrrTrackGeometryABI.....	9
4.2 SrrTrackSectionA.....	10
4.3 SrrTurnout(Left/Right)A.....	10
5 Appendix.....	10
5.1 The Concrete SRR Objects of SRR v0.01 – Train Manager Module.....	10
5.1.1 SrrBasicTrackSection – Basic Track Section.....	10
5.1.2 SrrSwitchB – N-way Switch.....	10
5.1.3 SrrBasicTurnout2Way – Basic 2-Way Turnout.....	10
5.1.4 SrrTransformationA (experimental!).....	11
5.1.5 SrrAxle (experimental!).....	11
5.1.6 SrrVehicleDriveBasic (experimental!).....	11
5.1.7 SrrWagon2axA (experimental!).....	11

Table of Figures

Figure 1: Modeling of the track layout with edges and nodes.....	4
Figure 2: Class diagram of tracks and turnouts.....	5
Figure 3: Class Diagram for Vehicles.....	6
Figure 4: Transformations of a 2-axle wagon.....	7
Figure 5: Modules, Trains, Train Parts and Vehicles.....	8
Figure 6: "ABI" example track geometry.....	9
Figure 7: 30° turnout geometry.....	10

2 Already Published in Concept Paper – Basismodul

- Overall Concept and Architecture

- Glossary
- SRR v0.01
- Master Concepts
- SRR Framework - Overview

3 Introduction

The SRR Framework comprises following components

- SRR Controller
- Module Coordinator
- SRR Objects

The SRR Controller and the module coordinator are composed by modules

- a base module
- a train manager module
- maybe some additional modules in future

SRR objects that are not related to tracks and trains, can be operated without a train manager module in the module coordinator.

Module coordinators without TMM don't need a TMM in the SRR Controller.

On the other hand, the SRR objects which are described in the present paper, need a TMM in the module coordinator and hence a TMM in the SRR Controller.

3.1 Results of the "Rollercoaster"-Project (2008)

- the track layout will be modeled by double-nodes and edges
- every axis of the trains will be moved independently over the mesh of nodes and edges, based on the *deltaEss* event.
- each axis will have a *transform* property, that is updated every frame
- the position and orientation of the vehicles will be derived from the *transform* properties of all their axes.
- every edge will have geometric properties that allow to calculate the position and orientation of each axis, based on the axis-properties *ess*, *parentEdge*, *isAtoB* and *inverse*.
 - *ess* is the current position of the axis in meters, relative to the parent edge
 - *parentEdge* is the current parent edge of the axis
 - *isAtoB*: if *isAtoB*=true, *deltaEss* will be added to *ess*, otherwise it will be subtracted
 - *inverse*: indicates, whether the axis is inserted "in backward direction" into the vehicle/train. This is important, when the rotation of the axis is visible or when the axis is not symmetric.
- turnouts are modelled by *switch* objects (exactly *SrrSwitchB* objects), that are the children of nodes. The *actualState* of the switch will be taken as input for the routing of axes.

3.2 Modeling the Track Layout

As shown in figure Figure 1, the track layout will be modeled by edges and nodes.

- Each edge will be terminated by two nodes, the "*A*" node and the "*B*" node
- Each node has exactly one neighbour node
- Each node has zero or more edges attached.
- If a node has attached more than one edge, then the node contains a switch

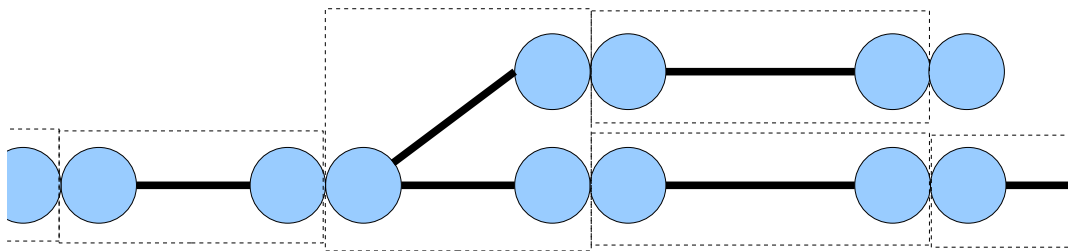


Figure 1: Modeling of the track layout with edges and nodes

Edges and nodes are modeled by the SRR objects *SrrTrackNode* and *SrrTrackEdge*.

Edges and nodes are organized by *track sections* and *turnouts* (see the dashed boxes in Figure 1). Track sections and turnouts will be basically realized by the prototypes *SrrBasicTrackSection* and *SrrBasicTurnout2Way*.

An *example track geometry* is published parallel to the SRR framework. This example track geometry comprises typical track sections and turnouts that include the graphical representation and can be instantiated easily by the user (i.e. they are static models).

3.3 Class Diagram of Tracks and Turnouts

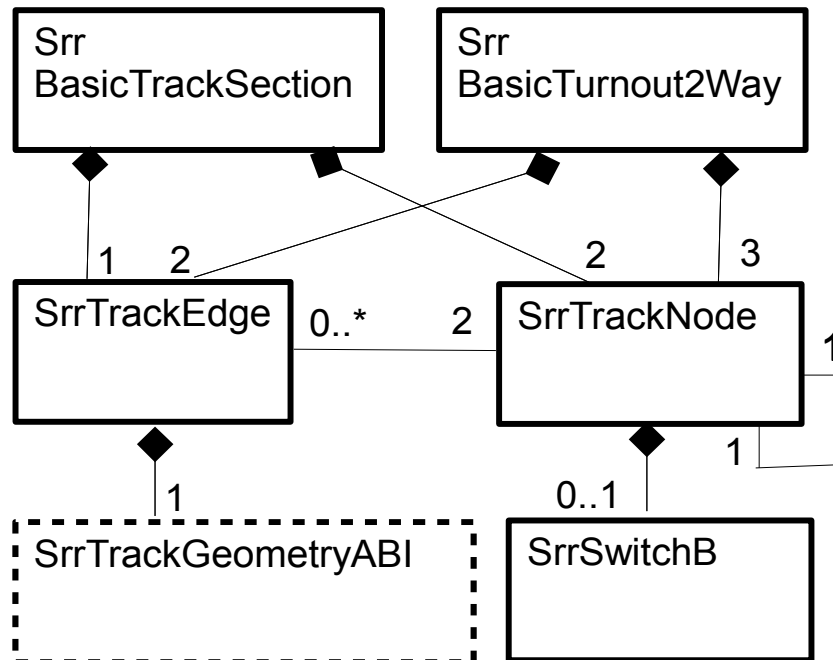


Figure 2: Class diagram of tracks and turnouts

Figure 2 shows the SRR objects that are used to create a track layout. The box around SrrTrackGeometryABI is shown in dashed lines, because this prototype is not a native SRR object but it is a helper provided by the *example track geometry*.

The user (module author) does not need to know all the details of this diagram. He just uses the prototypes provided by the *example track geometry*. Those prototypes will use these native SRR prototypes to create tracks and turnouts.

Remark: 3-Way Turnouts are not yet realized.

3.4 Vehicles

Vehicles are instrumented by the following SRR objects

- Axles
- Drives
- Cabs

Even a waggon can contain a "drive" (the "basic vehicle drive"), which will be used to "doze the vehicle manually".

3.4.1 Class Diagram for Vehicles

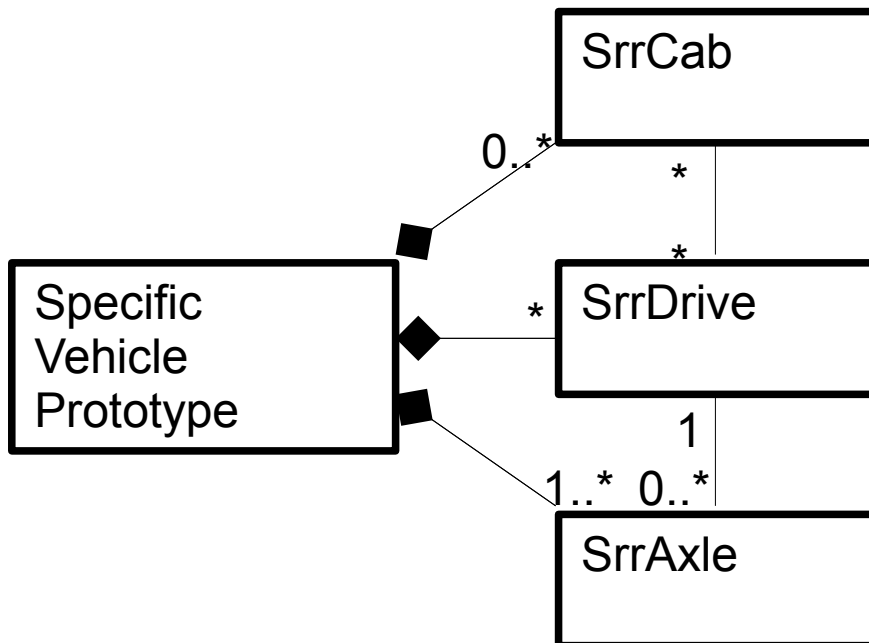


Figure 3: Class Diagram for Vehicles

Figure 3 displays the relations among all SRR objects, which are necessary to model a vehicle

3.4.2 Specific Vehicle Prototypes

Specific vehicle prototypes are some means to orchestrate the native SRR objects SrrVehicle, SrrVehicleDriveBasic and SrrAxle (will be more soon).

The specific vehicle prototypes make it easier for the model author to instrument a specific model. One criteria to distinguish the different specific vehicle prototypes, is the axle arrangement and another one is the fact of being a locomotive or a wagon, e.g.

Currently there are following specific vehicle prototypes:

- SrrWagon2axA

Specific vehicle prototypes provide functionality in three different situations

- standalone vehicle – user has opened the file of the model directly
- static model – a module has initialized a vehicle model by modParam, but it has not been set up on the tracks. Module is responsible for the placement of the model.
- Dynamic model – a layout has created the vehicle object and set it up on the tracks with the help of the mechanisms of the train manager – the vehicle is part of a train

3.4.2.1 SrrWagon2axA

SrrWagon2axA is a wagon with two axles, that calculates the transformation of the wagon from the two transformations of the axles.

The model takes all three transformations to display the graphic representation of the axles and the wagon.

Following geometric relations are valid:

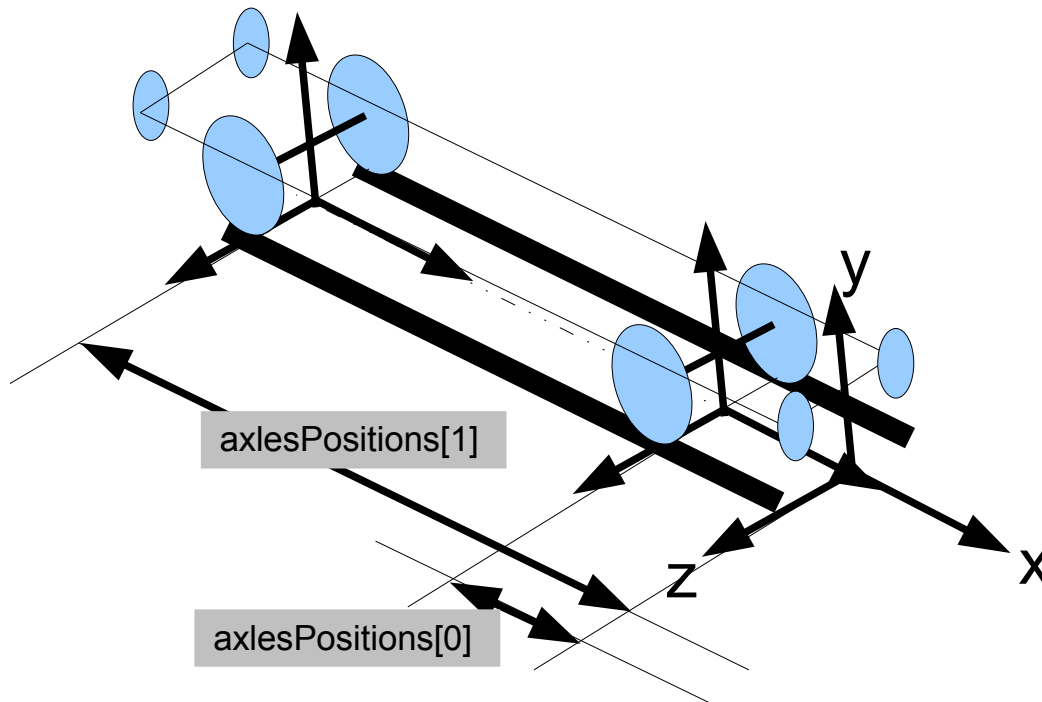


Figure 4: Transformations of a 2-axle wagon

The values *axesPositions[0]* and *axesPositions[1]* (and the value *length* – length over buffers) have to be given by the model author, the transformations are calculated by the SRR objects.

If the wagon is used in standalone mode or is used as a static model, the plane of the front buffers is located above the origin (x is "forward", y is "up" and z is "right"), if the model has been set up on the tracks, the transformations will result from the properties (*ess*, *parentEdge*) of the axles.

The model author must forward the transformations to <Transform> nodes to display the wagon and the axles at the right position and orientation, relative to the parent modules coordinate system.

3.5 Trains

Trains are built by vehicles.

Every vehicle is assigned to a train any time (except in standalone mode and except as a static model).

Each *train* and each *vehicle* are assigned to a parent module. If a train moves over the border of two modules, some of the vehicles of the train can already be assigned to the other module. In this case, the train consists of two *train parts*.

The SrrTrains object and the SrrTrainPart object are invisible, neither the module author nor the model author need to care about them, they are automatically created and deleted, when.

- A one-vehicle train is created
- A train is deleted (not yet implemented)
- Two trains are coupled together (not yet implemented)
- A train is splitted into two trains (not yet implemented)
- A vehicle is derailed (not yet implemented)
- A train moves from module X to module Y (not yet implemented)

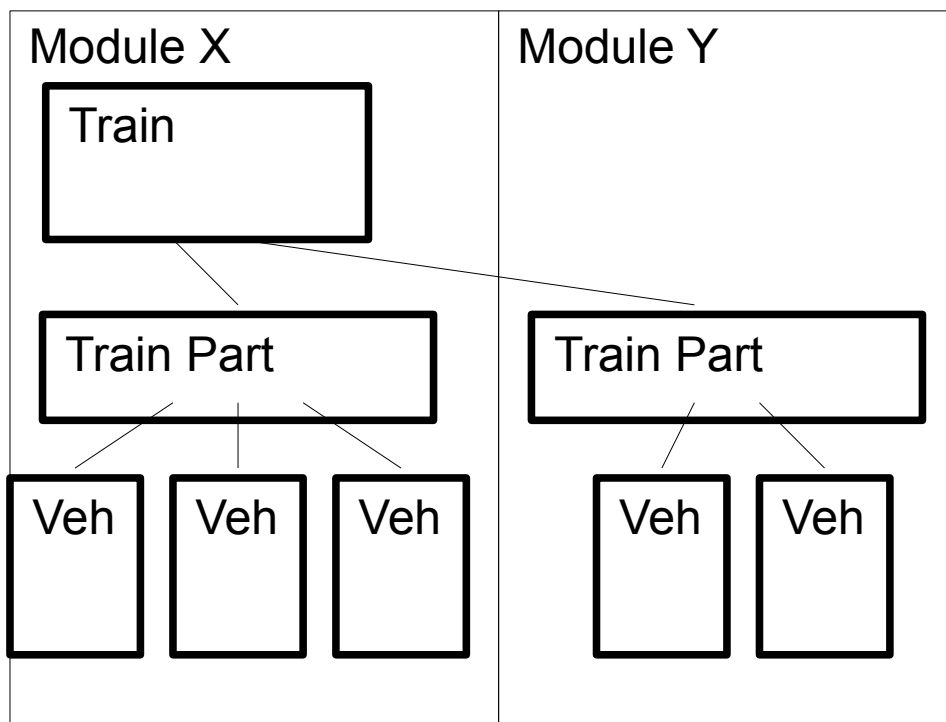


Figure 5: Modules, Trains, Train Parts and Vehicles

4 The Example Track Geometry

Following the findings of the *Rollercoaster Project*, the envisioned track geometry for SRR v0.01 will be buildt by segments of sections of circles and by straight track sections.

There were several remarks, that it should be possible to model *transition bends*.

To solve this conflict, it was decided to define an abstract interface that must be wrapped by a *concrete track geometry* in order to provide a concrete interface for the user (module author).

SRR is published with a *concrete example track geometry*, that realizes the „ABI“ approach. The „ABI“ track geometry *can be replaced* by another track geometry any time.

Additionally some *static models* are published, that realize a track section and a 30° left turnout and a 30° right turnout.

4.1 SrrTrackGeometryABI

Each track edge (SrrTrackEdge) comes with geometric properties, that allow to

- calculate a graphic representation of the edge during initialization
- calculate a transformation for each axis, the transformation consists of 4 vectors
 - translation
 - forward
 - up
 - right

Those geometric properties are encapsulated in an own prototype, the „track geometry“ prototype. In the case of the example track geometry, it is the SrrTrackGeometryABI prototype, that defines a track section by 5 vectors,

- vectorA.....position vector of the starting point
- vectorB.....position vector of the end point
- vectorI.....position vector of an intermediate points
- normalA.....unit vector to define the cross slope at point A
- normalB.....unit vector to define the cross slope at point B

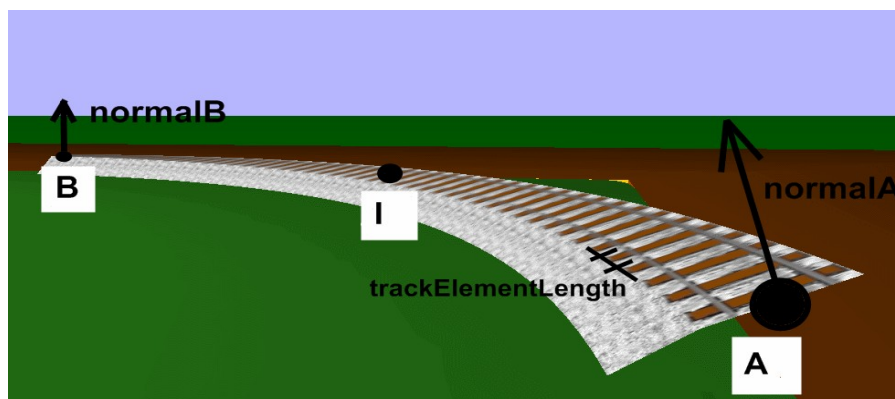


Figure 6: "ABI" example track geometry

4.2 SrrTrackSectionA

This prototype realizes an example track section, based on SrrBasicTrackSection and on SrrTrackGeometryABI.

It contains a <TriangleStripSet> that displays the graphical representation of the track section.

4.3 SrrTurnout(Left/Right)A

These prototypes realize example 30° turnouts, based on SrrBasicTurnout2Way and on SrrTrackGeometryABI.

It contains two <TriangleStripSet> nodes that display the graphical representation of the turnout.

The 5 vectors, which are input to two SrrTrackGeometryABI nodes, are calculated from 3 input parameters, A, B0 and R. (starting point and end point of the straight branch and radius of the curved branch), according to following figure:

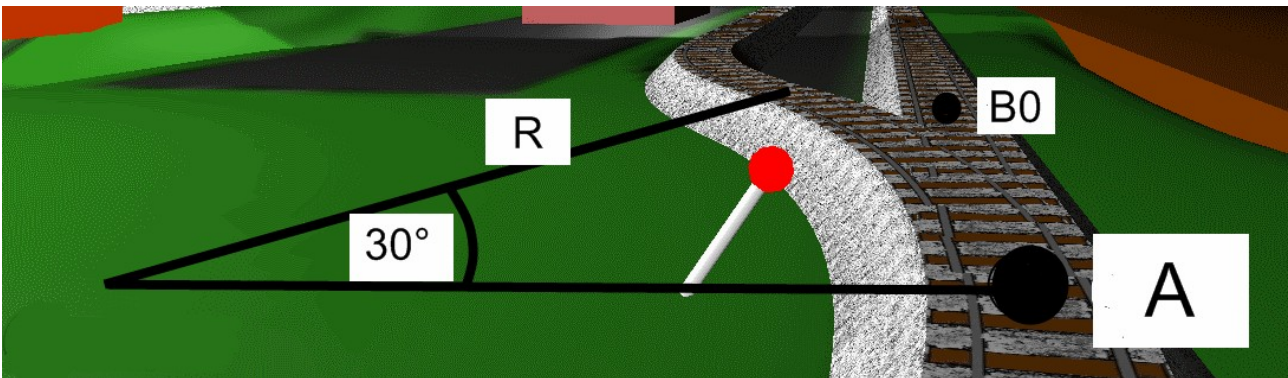


Figure 7: 30° turnout geometry

5 Appendix

5.1 The Concrete SRR Objects of SRR v0.01 – Train Manager Module

5.1.1 SrrBasicTrackSection – Basic Track Section

This SRR object orchestrates two SrrTrackNode nodes and one SrrTrackEdge node. It provides the basis for every track section, regardless which track geometry is used.

5.1.2 SrrSwitchB – N-way Switch

This SRR object provides the basis for an n-way switch (to be used in 2-way turnouts and 3-way turnouts).

5.1.3 SrrBasicTurnout2Way – Basic 2-Way Turnout

This SRR object orchestrates three SrrTrackNode nodes and two SrrTrackEdge nodes. It provides the basis for every 2-way turnout, regardless which track geometry is used.

5.1.4 SrrTransformationA (experimental!)

This prototype (which is no SRR object) provides the basis, to transport the transformation of an axle from the "track geometry" node to the user of the SrrAxle node (the vehicle). The transformation is calculated for each axle at each frame.

5.1.5 SrrAxle (experimental!)

This SRR object provides the basis to model any axle of any vehicle. It receives the *deltaEss* event from the vehicle/train and therefore moves the axle and with the axle moves the vehicle.

5.1.6 SrrVehicleDriveBasic (experimental!)

Each vehicle contains an arbitrary number of drive, but the SrrVehicle prototype adds at least this drive (SrrVehicleDriveBasic), so that even a wagon without drives will have this drive – to be used to doze a vehicle manually.

5.1.7 SrrWagon2axA (experimental!)

This SRR object orchestrates 2 SrrAxle objects and an SrrVehicle object. It calculates the wagon's transformation from the transformations of the two axles.