Simulated Railroad Framework, http://simulrr.sourceforge.net
Synopsis: 000_Synopsis

This file valid for step 0033.10
Issue Date: 2017-03-17

SRR Objects for Tracks and Turnouts, Replicators

1 Synopsis

The "Train Manager" extension of the SMUOS Framework provides the central services that are necessary for the implementation of the SRR Framework.

However, there are decentral facilities needed, too. Those decentral facilities are provided by SRR Objects (which are MIDAS objects of the SRR Framework).

The present paper describes the SRR Objects for tracks and turnouts – the static elements that are necessary to guide the vehicles and trains.

The paper 402_Trains describes the SRR Objects for the rail vehicles and trains, themselves (*not yet implemented*).

2 Purpose

The purpose of SRR Objects for tracks and turnouts is to enable modelers to build bound models of tracks and turnouts very easily. These SRR Objects provide the connection to the SRR Objects of rail vehicles and trains, hence they guarantee that every SrrTrains vehicle can run on every SrrTrains layout.

Most parts of the mathematics for movement of trains are hidden within the SRR Objects, so the modelers can concentrate on the visual/acoustic effects.

The SRR Objects for tracks and turnouts are based on the results of the "Rollercoaster" project from year 2008. They are intended to be used
- in bound/intrinsic models in static modules
- in bound/intrinsic models in dynamic modules (not yet tested)

2.1 Results of the "Rollercoaster" Project

- the track layout will be modeled by double-nodes and edges
- each axle of the trains will be moved independently over the mesh of nodes and edges, based on "deltaEss" events
- each axle 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/some of their axles
- each edge will have geometric properties that allow to calculate the position and orientation of each axle, based on the axle's properties "ess", "parentEdge", "isAtoB" and "inverse", where
  - "ess" is the current position of the axle, relative to the parent edge
  - "parentEdge" is the current parent edge of the axle
  - "isAtoB": if "isAtoB"=true, then "deltaEss" will be added to "ess", otherwise it will be subtracted
  - "inverse": indicates, whether the axle is inserted "in backward direction" into the vehicle. This is important, if the rotation of the axle is visible or if the axle is not symmetric.
- turnouts are modelled by switch objects (exactly MoosSwitchB objects) that are the children of nodes. The "actualState" of the switch will be taken as input for the routing of axles.

3 External View

Basically, a track layout is modeled by edges and nodes, like shown in figure 1.

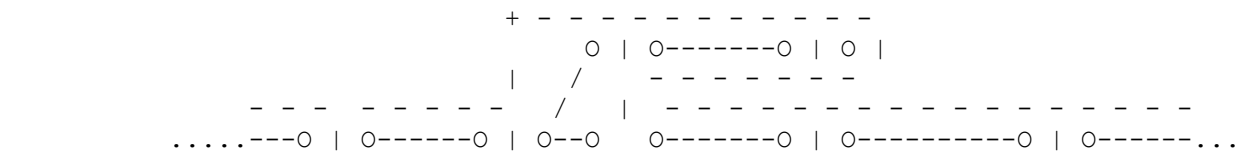

Figure 1 Modeling of track layouts with edges and nodes

Following rules apply:

1) Each edge is terminated by two nodes, the "A" node and the "B" node
2) Each node has exactly one neighbour node
3) Each node has zero or more edges attached
4) If a node has attached more than one edge, then the node contains a switch

Though it is possible to model any track layout with these rules and with the elements "node", "edge" and "switch", the elements are nevertheless combined to more specific elements (shown with the dashed lines in figure 1):

- track section (one edge and two nodes)
- 2-way turnout (two edges and three nodes)
- bumper (one node) (*not yet implemented*)

Additionally, it has been decided to outsource the geometric properties of each track edge to an external element (the "track geometry node"), to enable users of the SRR Framework to define their own track geometries.

3.1 The "Class Diagram" of SRR Objects for Tracks and Turnouts

In figure 2, each box depicts an SRR Object, with the exception of the TGN (track geometry node). The TGN is just a simple X3D prototype that provides the interface miGeo at its external interface.

Basically, the SRR Framework does not contain a TGN, but it expects the user to provide a TGN, which specializes the abstract miGeo interface to some concrete track geometry. However, the SRR Framework is accompanied by an "example track geometry", which provides such a node (besides some example bound models of tracks and turnouts).

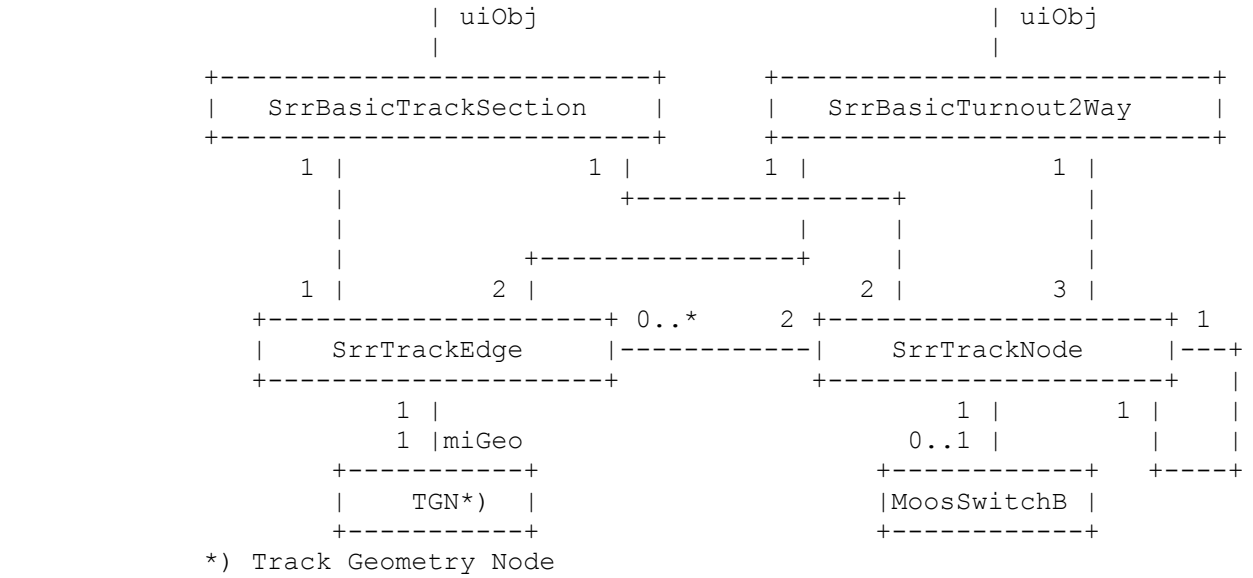

Figure 2 Class diagram of tracks and turnouts

3.2 SrrBasicTrackSection

SrrBasicTrackSection is a "no-state" SRR Object, which needs the "Train Manager" extension of the SMUOS Framework.

SrrBasicTrackSection can only be used in bound or intrinsic models of track sections. A track section is an element of the track layout that consists of exactly one track edge (with a contained TGN) and of exactly two track nodes.

The interface uiObj provides following fields:

Standard Fields

Please refer to chapter 5 of the paper 013_ModelsAndObjects for a description of fields that must be supported by any MIDAS Object.

"trackNodeA" (SFNode)

The model author provides an instance of an SrrTrackNode object here. The model author must set the "objId", "neighbourName" and "gauge" fields of that SRR Object, the other fields are set by SrrBasicTrackSection.

"trackNodeB" (SFNode)

The model author provides an instance of an SrrTrackNode object here. The model author must set the "objId", "neighbourName" and "gauge" fields of that SRR Object, the other fields are set by SrrBasicTrackSection.

"trackEdge" (SFNode)

The model author provides an instance of an SrrTrackEdge object here. The model author must set the "objId", "trackGeometry"*) and "exitViewpoint"**) fields of that SRR Object, the other fields are set by SrrBasicTrackSection.

*) "trackGeometry" must refer to a TGN. The TGN of the example track geometry is described in chapter 3.4.
Chapter 3.5 explains, how to implement your own TGN.
**) "exitViewpoint" refers to the viewpoint, which is bound, when the avatar of the user leaves a rail vehicle, which is located on this track edge

3.3 SrrBasicTurnout2Way

SrrBasicTurnout2Way is a "no-state" SRR Object, which needs the "Train Manager" extension of the SMUOS Framework.

SrrBasicTurnout2Way can only be used in bound or intrinsic models of 2-way turnouts. A 2-way turnout is an element of the track layout, which consists of exactly two track edges (with contained TGNs) and of exactly three track nodes.

The state of the switch is modeled by a MoosSwitchB object.

The interface uiObj provides following fields:

Standard Fields

Please refer to chapter 5 of the paper 013_ModelsAndObjects for a description of fields that must be supported by any MIDAS Object.

"trackNodeA" (SFNode)

The model author provides an instance of an SrrTrackNode object here. The model author must set the "objId", "neighbourName", "gauge" and "turnoutSwitch"*) fields of that SRR Object, the other fields are set by SrrBasicTurnout2Way.

*) "turnoutSwitch" must contain the reference to an MoosSwitchB object.
MoosSwitchB is described in 360_NwaySwitch.

"trackNodeB0" (SFNode)

The model author provides an instance of an SrrTrackNode object here. The model author must set the "objId", "neighbourName" and "gauge" fields of that SRR Object, the other fields are set by SrrBasicTurnout2Way.

"trackNodeB1" (SFNode)

The model author provides an instance of an SrrTrackNode object here. The model author must set the "objId", "neighbourName" and "gauge" fields of that SRR Object, the other fields are set by SrrBasicTurnout2Way.

"trackEdge0" (SFNode)

The model author provides an instance of an SrrTrackEdge object here. The model author must set the "objId", "trackGeometry"**) and "exitViewpoint"**) fields of that SRR Object, the other fields are set by SrrBasicTurnout2Way.

"trackEdge1" (SFNode)

The model author provides an instance of an SrrTrackEdge object here. The model author must set the "objId", "trackGeometry"**) and "exitViewpoint"**) fields of that SRR Object, the other fields are set by SrrBasicTurnout2Way.

**) see the foot notes in chapter 3.2.

3.4 The TGN of the Example Track Geometry (SrrTrackGeometryABI)

SrrTrackGeometryABI is an X3D prototype that provides two logical interfaces at its external interface
- the interface "miGeo", which is used by the SRR Framework (and which must be identical at all kinds of TGNs)
- some contributions to the interface "uiObj", which is used by the author of bound/intrinsic track and turnout models

The interface "miGeo" is described in chapter 3.5.

The "ABI Track Geometry" defines each track edge by three geometric locations:
- the starting point "A"
- the end point "B"
- and an intermediate point "I"

During initialization, the TGN solves an equation system to calculate the center and the radius of a circle, which intersects all three points.

Additionally, the length of the section of the circle is calculated and stored for later use.

The cross slope of the track section is defined by two unit vectors,
- one at point "A"
- and one at point "B".

Those unit vectors point "up" (the direction, where a steam locomotive would blow its smoke to).

Following fields must be set by the user (model author), before the track edge is initialized:

- vectorA (SFVec3f): coordinates of point "A", relative to the parent module
- vectorB (SFVec3f): coordinates of point "B", relative to the parent module
- vectorI (SFVec3f): coordinates of point "I", relative to the parent module
- normalA (SFVec3f): unit vector, pointing "up" at point "A"
- normalB (SFVec3f): unit vector, pointing "up" at point "B"
- trackElementLength (SFFloat): an initial estimate for the length of a track element. This value influences the number of coordinates, which will be calculated during initialization. This value is adapted in a way that the whole track section is filled by an integer number of track elements.

At the end of the initialization, SrrTrackGeometryABI will output following events, which should be used as basis for the visual representation of the track section:

- trackCoordinates (MFVec3f): an array of "length" / "trackElementLength" + 1 coordinates of points, which identify the axis of the track
- alongVectors (MFVec3f): an array of "length" / "trackElementLength" + 1 unit vectors, which point "along the track", "from A to B"
- normalVectors (MFVec3f): an array of "length" / "trackElementLength" + 1 unit vectors, which point "up" at every border of two track elements

3.5 How to implement your own TGN

3.5.1 How Vehicles Interact with Tracks and Turnouts

The interaction between vehicles and tracks is based on the SRR object SrrAxle.

Each vehicle/train is responsible to initialize all of its SrrAxle objects and to send to them a deltaEss event every frame.

The value Ds (deltaEss) is calculated by the current velocity of the train and the frame time (time delta between last frame and current frame) and gives the value, which is to be added to or subtracted from the ess property of the axle.

After having received the deltaEss event, SrrAxle will update all its properties that describe the position of the axle within the track layout.

For this purpose, SrrAxle needs the "length" property of the TGN contained in the parent edge.

After having updated the own properties, SrrAxle will call the TGN (it will send a "calculateAxleTransformation" event to the TGN).

This will trigger the TGN to calculate and update the "transformation"."transformation" property of the SrrAxle.

The vehicle will listen to the update of the contents of "transformation" and will display the axle and the vehicle at the correct position and orientation.

3.5.2 The Minimum Interface miGeo

Hence the track geometry node has to provide following services
- initialization
- positioning of axles

For this purpose, following fields are used
- "objId" (SFString)
- "modParam" (SFNode)
- "initialized" (SFNode)
- "length" (SFFloat)
- "calculateAxleTransformation" (SFNode)

3.5.2.1 Initialization

As common at the uiObj interface, the TGN will be initialized by the surrounding node, i.e SrrTrackEdge. SrrTrackEdge will set the "objId" property of the TGN and afterwards it will report the module parameters "modParam" to start the initialization.

Latest now the track geometry will calculate and provide its "length" property.

After successful initialization, it will report it's specific geometric parameters (depending on the concrete track geometry) directly to the model.

It will return a reference to itself in the "initialized" field. In case of failure it may return null.

3.5.2.2 Positioning of Axles

After an axle has updated it's position (by the deltaEss event), its properties
- "parentEdge" (pointer to the parent edge SrrTrackEdge object)
- "ess" (distance of the axle from the "A" node in meters)
- "isAtoB" (true, when the "forward" direction of the train points from "A" to "B" at this axle)

will be up to date.

In order to generate a real position on the screen, SrrAxle will send a "calculateAxleTransformation" event to the TGN of the parent edge. This is an SFNode event that points to the axle itself.

Now the track geometry node is able to use the properties of the axle
- SrrAxle.ess
- SrrAxle.isAtoB
- SrrAxle.inverse
- SrrAxle.parentVehicle.inverse

to calculate and set the property
- SrrAxle.transformation.transformation.

This will display the axle and hence the vehicle at the right position of the track layout.

3.6 Replicators

The SRR Object SrrReplicator is used to define a position within the track layout, where vehicles can be setup on the tracks.

Furthermore, the SRR Object presents a list of (filtered) vehicle type ids to the user, so that he can select a vehicle, which will then be setup.

It can be used
- in bound/intrinsic models in a static module
- in bound/intrinsic models in a dynamic module (not yet tested)

SrrReplicator provides following fields at the "uiObj" interface:

Standard Fields

Please refer to chapter 5 of the paper 013_ModelsAndObjects for a description of fields that must be supported by any MIDAS Object.

"uocs" (MFString)

During initialization, this parameter indicates the UOCs that will be considered by the replicator.

"categories" (MFString)

When vehicle types are registered (via the uiControl interface), then the user (frame author) provides a list of categories for each vehicle type.

The SRR Controller (Train Manager) uses the "categories" field of a replicator to filter all registered vehicle types for this replicator.

parentEdgeExtObjId (SFString), ess (SFFloat), isAtoB (SFBool)

These parameters have to be set to specify the position and direction within the track layout, relative to which the vehicles will be setup.

registeredVehicleTypeIds (MFString)

This field outputs the filtered list of registered vehicle types.

createVehicle (SFString) (*not implemented yet*)

The user (model author) takes one value of the list "registeredVehicleTypeIds" and inputs it at the field "createVehicle", in order to create a 1-vehicle-train with this vehicle type at this replicator.

progressIndicator (SFFloat) (*not implemented yet*)

The replicator outputs several values at this field to indicate the progress of the creation process of the 1-vehicle-train.

Output of values "0.0" and "1.0" is guaranteed, intermediate values may be output additionally.

4 Internal View

tbd.

5 Additional Info

none