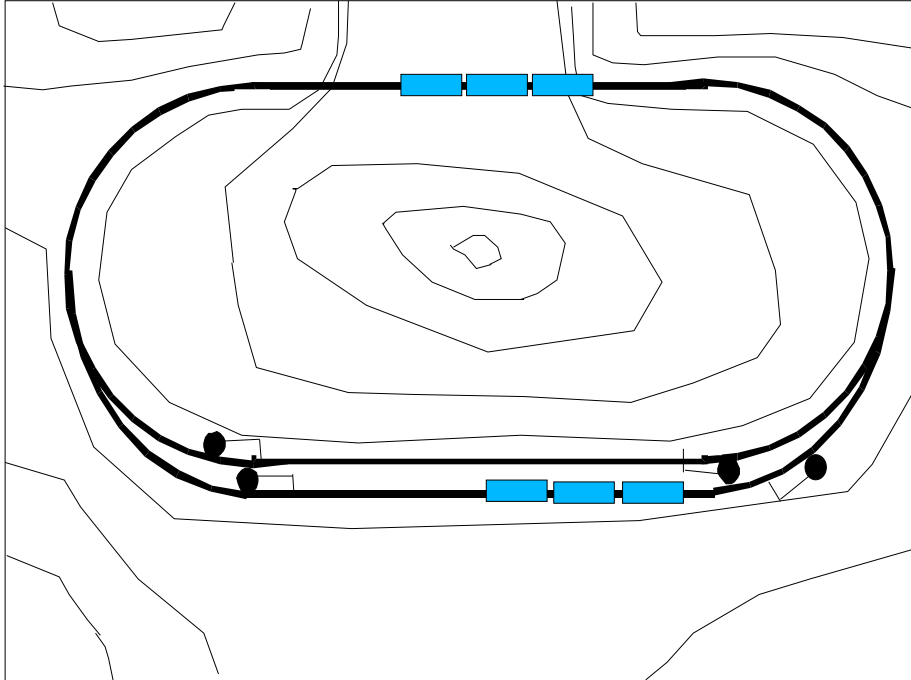


ANL



SIMUL-RR: VIRTUAL MODEL RAILROAD ON X3D BASIS

How To Build My Own SIMUL-RR

Tracks and Turnouts

1 Introduction

It's one of the main goals of SIMUL-RR to provide railroad kinematics for the user, who wants to create his own virtual model railroad based on X3D/VRML.

The rollercoaster project done in 2008 indicated the approach:

- 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*.
- turnouts are modelled by *switch* objects, that are the children of nodes. The *actualState* of the switch will be taken as input for the routing of axes.

2 Basic Model for 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

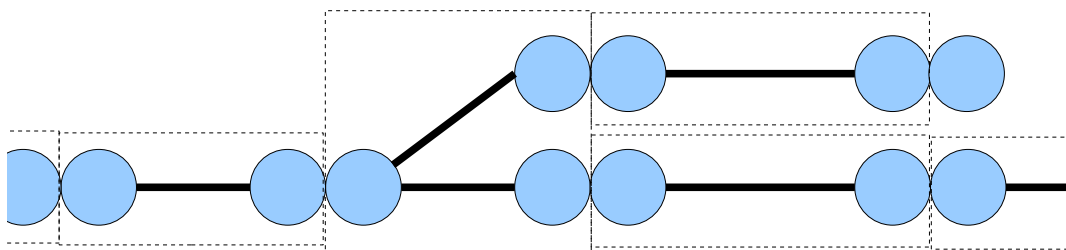


Figure 1: Modeling of the track layout with edges and nodes

Edges and nodes are modeled by the SIMUL-RR objects *SrrTrackNode* and *SrrTrackEdge*.

Edges and nodes are organized by *track sections* and *turnouts* (see the dashed boxes in Figure 1).

SIMUL-RR comes with an *example track geometry*, i.e typical track sections and turnouts that

June 2009

"Howto"-Document "Tracks and Turnouts" Version v0.01 – Step 0012

include the graphical representation and can be created easily by the user (module author).

3 Class Diagram of Tracks and Turnouts

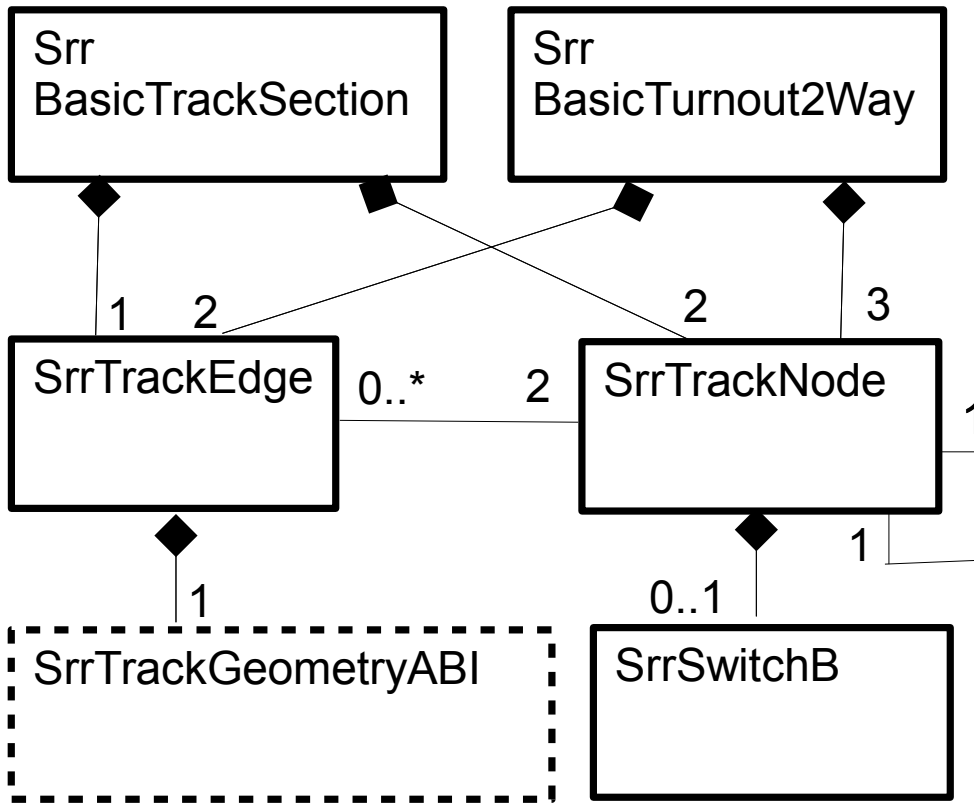


Figure 2: Class diagram of tracks and turnouts

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* (see chapter 5). Those prototypes will use these native SIMUL-RR prototypes to create tracks and turnouts.

SrrBasicTrackSection gets the parameters

- objId
- trackEdge (reference to an uninitialized SrrTrackEdge node)
- trackNodeA (reference to an uninitialized SrrTrackNode node)
- trackNodeB (reference to an uninitialized SrrTrackNode node)

and initializes the referenced nodes.

The node trackEdge must contain an uninitialized track geometry node that will be initialized, too.

The example geometry prototype **SrrTrackSectionA** (not shown in Figure 2) will listen to the *trackCoordinates*, *alongVectors* and *normalVectors* events of the *trackGeometry* node to create the

graphical representation of the track section during initialization.

The node *trackEdge* will receive the object ID **<objId>**

The node *trackNodeA* will receive the object ID **<objId>.A**

The node *trackNodeB* will receive the object ID **<objId>.B**

SrrBasicTurnout2Way gets the parameters

- objId
- trackEdge0 (reference to an uninitialized SrrTrackEdge node)
- trackEdge1 (reference to an uninitialized SrrTrackEdge node)
- trackNodeA (reference to an uninitialized SrrTrackNode node)
- trackNodeB0 (reference to an uninitialized SrrTrackNode node)
- trackNodeB1 (reference to an uninitialized SrrTrackNode node)

and initializes the referenced nodes.

The nodes trackEdge0 and trackEdge1 must contain two uninitialized track geometry nodes that will be initialized, too.

The example geometry prototypes *SrrTurnoutLeftA* and *SrrTurnoutRightA* (not shown in Figure 2) will listen to the *trackCoordinates*, *alongVectors* and *normalVectors* events of the *trackGeometry* nodes to create the graphical representation of the turnout during initialization.

The node trackNodeA must contain an uninitialized SrrSwitchB node that will be initialized, too. *SrrTurnoutLeftA* and *SrrTurnoutRightA* will trigger the *toggle* field of the switch node to change the state of the turnout. SrrAxis (see next chapter) will use the *actualState* field of the switch to route the axis through the mesh of edges and nodes.

The node *trackEdge0* will receive the object ID **<objId>.0**

The node *trackEdge1* will receive the object ID **<objId>.1**

The node *trackNodeA* will receive the object ID **<objId>.A**

The node *trackNodeB0* will receive the object ID **<objId>.B0**

The node *trackNodeB1* will receive the object ID **<objId>.B1**

The contained SrrSwitchB node will receive the object ID **<objId>.Switch**

SrrSwitchB

is nearly identical to SrrSwitchA. The only difference is, that actualState is not reported as "true" or "false", but as SFInt32 "0" or "1" (needed by SrrAxis as index into the trackEdges[] and isNodeA[] arrays of SrrTrackNode). Three-way turnouts are not yet realized.

4 How Vehicles Interact with the Tracks and Turnouts

The interaction between vehicles and tracks is based on the prototype *SrrAxis*.

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

After having received the *deltaEss* event, *SrrAxis* will update all its properties

- parentEdge
- parentEdgeName
- isAtoB
- ess

For this purpose, *SrrAxis* needs some properties of the *trackGeometry* node contained in the parent edge.

After having updated the own properties, *SrrAxis* will send an *SFNode* event to the *trackGeometry node*, that is contained in the parentEdge. This event will be sent to the *calculateAxisTransformation* field of the track geometry node, which will in turn update the contents of the *transformation* property of the *SrrAxis* (using the *ess*, *inverse* and *isAtoB* properties of the *SrrAxis*).

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

5 The Example Track Geometry

SIMUL-RR comes with an example track geometry, that is contained in 5 prototypes:

- *SrrTrackGeometryABI*
- *SrrTransformation*
- *SrrTrackSectionA*
- *SrrTurnoutLeftA*
- *SrrTurnoutRightA*

5.1 *SrrTrackGeometryABI*

SrrTrackGeometryABI represents the geometric properties of a track edge.

During initialization it calculates the basic data to create a graphic representation of the edge.

Each time an axis receives a *deltaEss* event, *SrrTrackGeometryABI* supports the axis in calculating the new transformation matrix for the axis, based on the geometric parameters of the edge and the properties of the axis..

5.1.1 Initialization

The user sets the parameters

- vectorA (position vector of the starting point)
- vectorB (position vector of the end point)
- vectorI (position vector of the intermediate point)
- normalA (vector indicating the cross slope at point A)
- normalB (vector indicating the cross slope at point B)
- trackElementLength (number indicating the length of a track element)

The parameters are illustrated in Figure 3.

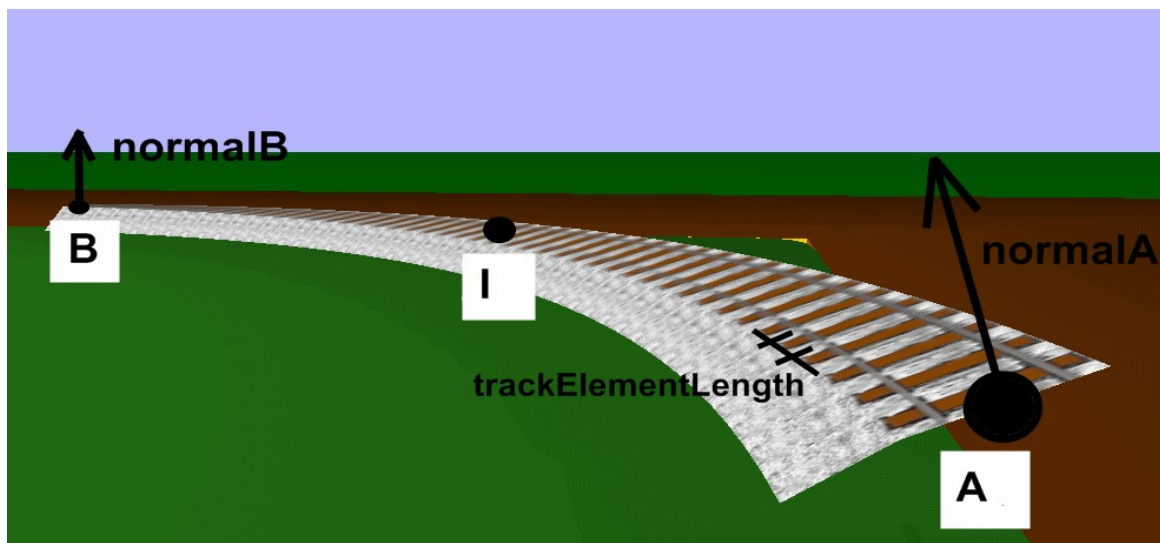


Figure 3: Geometric parameters of a track edge

SrrTrackGeometryABI updates

- trackElementLength (an integer number of track elements shall fit into the edge)

and calculates the external parameters

- length (length of the curve from point A over point I to point B)
- trackCoordinates ("numberOfTrackElements + 1" position vectors)
- alongVectors ("numberOfTrackElements + 1" unit vectors pointing from A to B)
- normalVectors("numberOfTrackElements + 1" unit vectors pointing "up")

The parameters

- center
- radius
- crossSlopeA
- crossSlopeB

are calculated and stored internally for faster calculation of axis transformations.

5.1.2 Calculation of transformation

SrrAxis will send an SFNode event which points to itself, to the *calculateAxisTransformation* field of the SrrTrackGeometryABI of the parent edge, every time it need to update its transformation.

The *isAtoB* property determines, if a positive deltaEss points from A to B (true) or from B to A (false). In case of *isAtoB* true it means, that the x-axis of the train points from A to B.

The *inverse* property determines, if an axis has been inserted into a train in forward direction or in backward direction.

5.2 SrrTransformation

For each deltaEss event, SrrTrackGeometry calculates a transformation according to SIMUL-RR standard.

The field *transformation* is an *MFVec3f* with 4 elements:

- *transformation*[0] is the *translation* of the axis
- *transformation*[1] is the *"forward" unit vector* of the axis
- *transformation*[2] is the *"up" unit vector* of the axis
- *transformation*[3] is the *"right" unit vector* of the axis

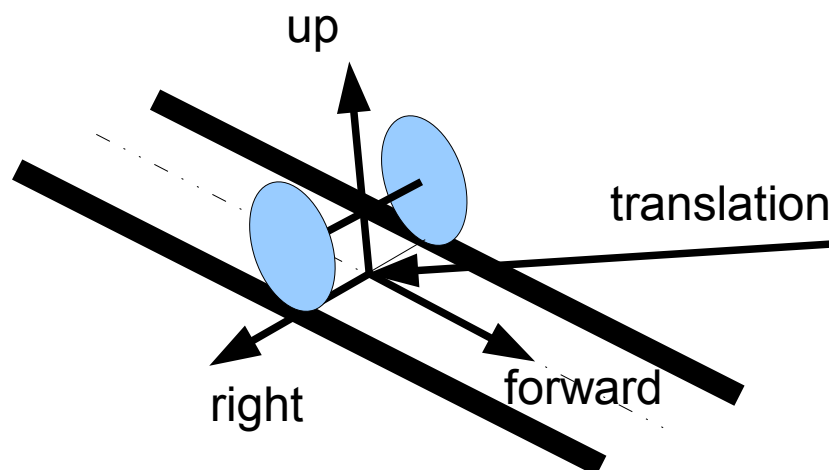


Figure 4: Transformation of an axis

In case of $(isAtoB \ \&\& \ inverse) \ || \ (!isAtoB \ \&\& \ !inverse)$ the "forward" vector will point from B to A, otherwise it will point from A to B.

SrrTransformation provides a *translation function transfToX3D* that translates the four vectors into three X3D values, that can be used as input for two nested X3D/VRML <Transform> nodes:

- translation
- rotation1
- rotation2

The values *translation* and *rotation1* are used as input for the outer transformation. The outer transformation moves the origin of the axis object to the correct place and rotate it in a way, that the local "x"-axis of the object points into the same direction as the "forward" vector.

The value *rotation2* is used as input for the inner transformation and rotates the object around the local "x"-axis such that it takes on the correct bank.

5.3 SrrTrackSectionA

SrrTrackSectionA takes the parameters

- objId
- vectorA
- vectorB
- vectorI
- normalA
- normalB
- neighbourA
- neighbourB

in order to create one SrrTrackEdge (containing a SrrTrackGeometryABI) and two SrrTrackNode together with an SrrBasicTrackSection.

SrrTrackSectionA contains a <TriangleStripSet>, whose point coordinates and texture coordinates will be set based on the *trackCoordinates*, *alongVectors* and *normalVectors* reported by SrrTrackGeometryABI.

A predefined texture file will be used to display the track section.

5.4 SrrTurnoutLeftA, SrrTurnoutRightA

These prototypes use the parameters

- objId
- vectorA
- vectorB0
- R
- neighbourA

- neighbourB0
- neighbourB1

to create turnouts in a 30° geometry. One track is straight (from A to B0), the other track is a 30°-curve with radius R.

A lever with a red knob is created besides the turnout, connected to SrrSwitchB that is responsible for the state of the turnout (0...straight, 1...curve).