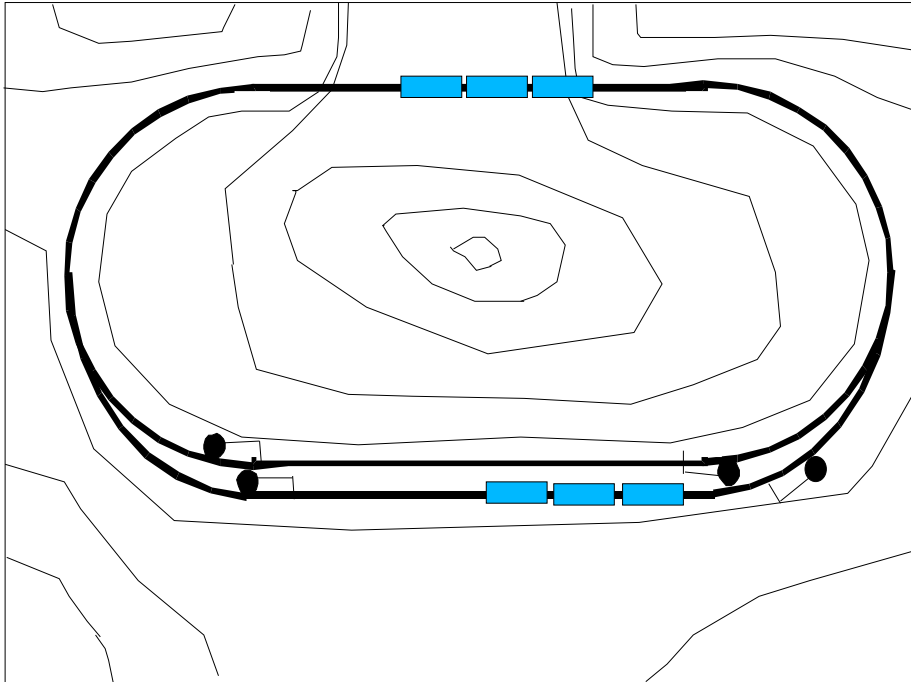


ANL



# **SIMUL-RR: VIRTUELLE MODELLEISENBAHN AUF X3D BASIS**

## **Wie baue ich meine eigenen SIMUL-RR**

### **Gleise und Weichen**

## 1 Einführung

Eines der wichtigsten Ziele von SIMUL-RR ist es, eine allgemein gültige Kinematik für Schienenfahrzeuge zur Verfügung zu stellen, die man in virtuellen Modelleisenbahnen auf X3D/VRML-Basis verwenden kann

Das "Rollercoaster"-Projekt von 2008 hat folgendes Konzept nahe gelegt:

- Die Gleisanlage wird durch einen Doppelpunktgraphen entsprechend der Dissertation von Daniel Hürlimann (ETH Zürich) modelliert
- Jede Achse wird unabhängig von den anderen Achsen über die Gleisanlage bewegt. Basis dafür ist das *deltaEss*-Ereignis
- Jede Achse hat eine Transformations-Eigenschaft (Position und Orientierung), die mit jedem Bild neu berechnet wird
- Position und Orientierung der Fahrzeuge wird von Position und Orientierung der Achsen abgeleitet
- Jede Kante des Doppelpunktgraphen hat auch *geometrische Eigenschaften*, die es ermöglichen, Position und Orientierung aller Achsen basierend auf deren Eigenschaften *ess*, *perentEdge*, *isAtoB* und *inverse* zu berechnen
- Weichen werden durch *Switch* Objekte modelliert, die als Kinder von Knoten des Doppelpunktgraphen auftreten. Die *actualState*-Eigenschaft der *Switch*-Objekte dient als Basis der Wegfindung der Achsen durch den Doppelpunktgraphen.

## 2 Modellierung der Gleisanlage

Abbildung 1 zeigt ein Beispiel eines Doppelpunktgraphen, wie er verwendet wird, um eine Gleisanlage zu modellieren.

Dabei gelten folgende Regeln:

- Jede Kante referenziert zwei Endknoten, *Knoten "A"* und *Knoten "B"*
- Jeder Knoten hat genau einen Nachbarknoten
- Jeder Knoten referenziert 0, 1 oder mehrere Kanten
- Wenn ein Knoten auf mehr als eine Kante zeigt, enthält er einen *Schalter(Switch)* für die Weiche

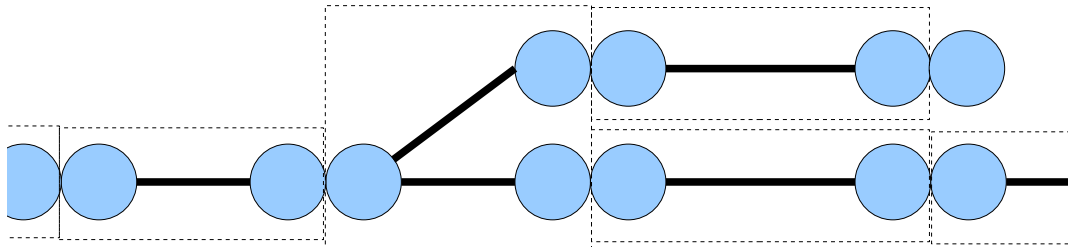


Abbildung 1: Modellierung der Gleisanlage als Doppelpunktgraph

Der Doppelpunktgraph wird mit Hilfe der SRR-Objekte *SrrTrackNode* und *SrrTrackEdge* aufgebaut.

Kanten und Knoten werden durch die übergeordneten Elemente *Gleisabschnitt* und *Weiche* organisiert (siehe die gestrichelten Kästchen in Abbildung 1).

SIMUL-RR enthält eine *beispielhafte Gleisgeometrie*, die durch eine andere ersetzt werden könnte. Diese Gleisgeometrie stellt Prototypen zur Verfügung, die bereits die graphische Repräsentation von Schienen und Weichen enthalten und einfach in einem Modul vom Modulautor instanziiert werden können.

### 3 Klassendiagramm für Gleise und Weichen

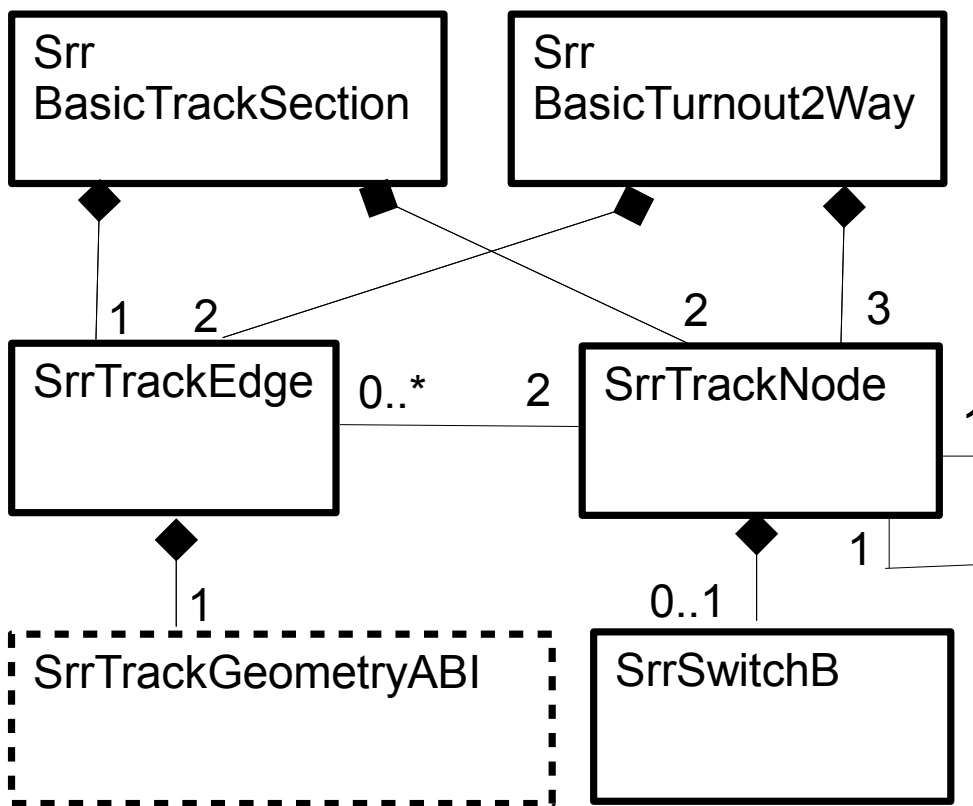


Abbildung 2: Klassendiagramm für Gleise und Weichen

Abbildung 2 zeigt die Beziehungen der SRR-Objekte, die benötigt werden, um eine Gleisanlage zu modellieren. SrrTrackGeometryABI ist gestrichelt umrandet, da dieser Prototyp eigentlich kein SRR-Objekt ist, sondern ein Teil der *beispielhaften Gleisgeometrie*.

Der Benutzer (Modulautor) muss nicht alle Details dieses Diagramms kennen, da es genügt, die *beispielhafte Gleisgeometrie* zu benutzen, um Gleise und Weichen anzulegen (siehe Kapitel 5).

**SrrBasicTrackSection** bekommt die Parameter

- objId
- trackEdge (Referenz auf einen uninitialisierten SrrTrackEdge-Knoten)
- trackNodeA (Referenz auf einen uninitialisierten SrrTrackNode-Knoten)
- trackNodeB (Referenz auf einen uninitialisierten SrrTrackNode-Knoten)

und initialisiert die referenzierten Knoten.

Der Knoten trackEdge muss einen trackGeometry Knoten enthalten, welcher ebenfalls initialisiert wird.

Der Prototyp *SrrTrackSectionA* aus der beispielhaften Gleisgeometrie (in Abbildung 2 nicht gezeigt) reagiert auf die *trackCoordinates-*, *alongVectors-* und *normalVectors-*Ereignisse des *trackGeometry*-Knotens, um die graphische Repräsentation des Gleisabschnitts zu erzeugen.

Der Knoten *trackEdge* bekommt die Objekt-ID **<objId>**

Der Knoten *trackNodeA* bekommt die Objekt-ID **<objId>.A**

Der Knoten *trackNodeB* bekommt die Objekt-ID **<objId>.B**

**SrrBasicTurnout2Way** bekommt die Parameter

- objId
- trackEdge0 (Referenz auf einen uninitialisierten SrrTrackEdge-Knoten)
- trackEdge1 (Referenz auf einen uninitialisierten SrrTrackEdge-Knoten)
- trackNodeA (Referenz auf einen uninitialisierten SrrTrackNode-Knoten)
- trackNodeB0 (Referenz auf einen uninitialisierten SrrTrackNode-Knoten)
- trackNodeB1 (Referenz auf einen uninitialisierten SrrTrackNode-Knoten)

und initialisiert die referenzierten Knoten.

Die Knoten trackEdge0 und trackEdge1 müssen je einen trackGeometry Knoten enthalten, welcher ebenfalls initialisiert wird.

Die Prototypen *SrrTurnoutRightA* und *SrrTurnoutLeftA* aus der beispielhaften Gleisgeometrie (in Abbildung 2 nicht gezeigt) reagieren auf die *trackCoordinates-*, *alongVectors-* und *normalVectors-*Ereignisse der *trackGeometry*-Knoten, um die graphische Repräsentation der Weiche zu erzeugen.

Der Knoten *trackEdge0* bekommt die Objekt-ID **<objId>.0**

Der Knoten *trackEdge1* bekommt die Objekt-ID **<objId>.1**

Der Knoten *trackNodeA* bekommt die Objekt-ID **<objId>.A**

Der Knoten *trackNodeB0* bekommt die Objekt-ID **<objId>.B0**

Der Knoten *trackNodeB1* bekommt die Objekt-ID **<objId>.B1**

### **SrrSwitchB**

ist fast identisch zu SrrSwitchA. Einzig das *actualState*-Ereignis wird nicht als SFBool (true oder false), sondern als SFInt32 (0 oder 1) geliefert. SrrAxis benötigt dies, um damit die Arrays trackEdges[] und isNodeA[] (Eigenschaften des SrrTrackNode Prototypen) zu indizieren.

Dreiwegweichen sind noch nicht realisiert.

## 4 Wie Fahrzeuge mit Gleisen und Weichen interagieren

Die Interaktion zwischen Fahrzeugen und der Gleisanlage beruht auf dem Prototypen *SrrAxis*.

Jedes Fahrzeug/jeder Zug ist verantwortlich, alle seine Achsen zu initialisieren und ihnen bei jedem Frame ein *deltaEss*-Ereignis zu schicken.

Nachdem *SrrAxis* das *deltaEss*-Ereignis erhalten hat, berechnet es seine Eigenschaften neu

- parentEdge
- parentEdgeName
- isAtoB
- ess

Dafür benötigt *SrrAxis* einige Eigenschaften des *trackGeometry* Knotens, der in der Elternkante enthalten ist.

Danach wird *SrrAxis* ein *calculateAxisTransformation*-Ereignis an den *trackGeometry* Knoten der Elternkante senden.

Dieser wird den Inhalt der *transformation*-Eigenschaft der Achse neu berechnen (unter Benutzung der *ess*, *inverse* und *isAtoB* Eigenschaften der Achse).

Das Fahrzeug wird auf die Änderung des Inhalts der *transformation*-Eigenschaft reagieren und die Achsen und das Fahrzeug an der neuen Position und mit der neuen Orientierung darstellen.

## 5 Die beispielhafte Gleisgeometrie

SIMUL-RR enthält eine beispielhafte Gleisgeometrie, die aus folgenden Prototypen besteht:

- SrrTrackGeometryABI
- SrrTransformation
- SrrTrackSectionA
- SrrTurnoutLeftA
- SrrTurnoutRightA

### 5.1 SrrTrackGeometryABI

*SrrTrackGeometryABI* repräsentiert die geometrischen Eigenschaften einer Kante.

Während der Initialisierung berechnet dieser Prototyp auch die grundlegenden Daten, um die graphische Darstellung der Kante zu ermöglichen.

Bei jedem *deltaEss*-Ereignis unterstützt *SrrTrackGeometryABI* das *SrrAxis*-Objekt durch die Berechnung der neuen Transformation für die Achse, basierend auf den geometrischen Eigenschaften der Kante und den aktuellen Eigenschaften der Achse.

#### 5.1.1 Initialisierung

Der Benutzer setzt die Parameter

- vectorA (Ortsvektor des Ausgangspunktes)
- vectorB (Ortsvektor des Endpunktes)
- vectorI (Ortsvektor eines Zwischenpunktes)
- normalA (Vektor, um die Querneigung am Punkt A festzulegen)
- normalB (Vektor, um die Querneigung am Punkt B festzulegen)
- trackElementLength (Länge eines Gleiselementes)

Die Bedeutung dieser Parameter ist in Abbildung 3 veranschaulicht.

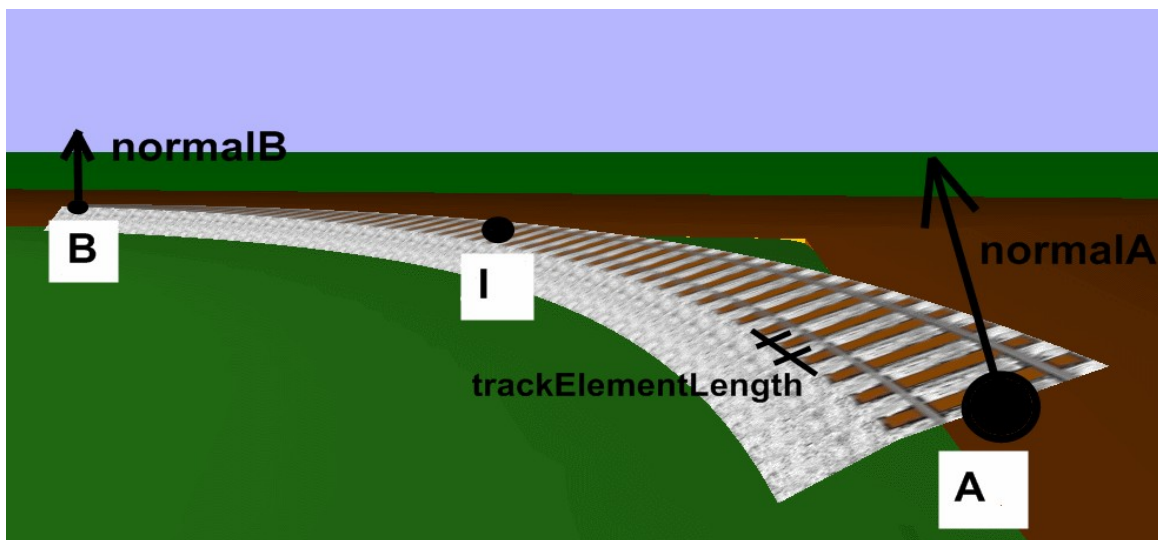


Abbildung 3: Geometrische Eigenschaften einer Kante

SrrTrackGeometryABI berechnet in der Initialisierung

- trackElementLength (sodass eine ganze Zahl von Gleiselementen in die Kante passt)

sowie die weiteren externen Parameter

- length (Länge der Kurve von A über I nach B)
- trackCoordinates ("numberOfTrackElements + 1" Ortsvektoren)
- alongVectors ("numberOfTrackElements + 1" Einheitsvektoren, die "von A nach B" zeigen)
- normalVectors("numberOfTrackElements + 1" Einheitsvektoren, die "aufwärts" zeigen)

Die Parameter

- center
- radius
- crossSlopeA
- crossSlopeB

werden berechnet und intern gespeichert, um später schneller die Transformationen der Achsen zu

berechnen.

### 5.1.2 Berechnung der Transformation

Immer, wenn SrrAxis die eigene Transformation anpassen möchte, sendet es ein SFNode-Ereignis an das Feld *calculateAxisTransformation* des *trackGeometry* Knotens der Elternkante..

Die *isAtoB* Eigenschaft der Achse entscheidet, ob ein positives deltaEss von A nach B oder von B nach A zeigt. D.h., wenn isAtoB gleich true ist, zeigt die x-Achse des Zuges auf dieser Kante von A nach B.

Die *inverse* Eigenschaft entscheidet, ob eine Achse relativ zum Zug vorwärts oder rückwärts eingefügt worden ist.

## 5.2 SrrTransformation

SrrTrackGeometry berechnet für jedes deltaEss Ereignis eine SIMUL-RR Standardtransformationsmatrix, die aus 4 Vektoren besteht.

- *transformation*[0] ist der Ortsvektor des Ursprungs des lokalen Koordinatensystems
- *transformation*[1] ist die *x-Achse* des lokalen Koordinatensystems ("*vorwärts*")
- *transformation*[2] ist die *y-Achse* des lokalen Koordinatensystems ("*aufwärts*")
- *transformation*[3] ist die *z-Achse* des lokalen Koordinatensystems ("*rechts*")

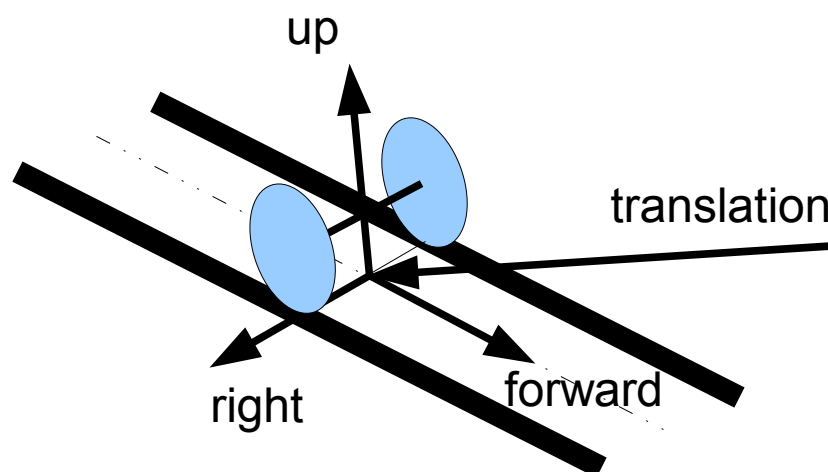


Abbildung 4: Transformation einer Achse

Im Fall von *(isAtoB && inverse)* || *(!isAtoB && !inverse)* wird der "forward" – Vektor (x-Achse) von B nach A zeigen, ansonsten von A nach B.

SrrTransformation bietet eine **Übersetzungsfunktion *transfToX3D***, die die vier Vektoren in 3 X3D-Werte umrechnet, welche als Eingabewerte für zwei geschachtelte <Transform>-Knoten verwendet werden können:

- translation
- rotation1
- rotation2

Die Werte ***translation*** und ***rotation1*** werden als Eingabe für die äußere Transformation verwendet, welche den Ursprung an den richtigen Punkt in der Gleisachse transformiert und die x-Achse in die Längsachse des Gleises rotiert.

Der Wert ***rotation2*** wird als Eingabe für die innere Transformation verwendet, welche das Objekt um die lokale x-Achse in die korrekte Lage dreht.

### 5.3 SrrTrackSectionA

SrrTrackSectionA nimmt die Parameter

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

entgegen, um eine SrrTrackEdge (mit einer SrrTrackGeometryABI) zwei SrrTrackNode zusammen mit einem ***SrrBasicTrackSection*** zu kreieren.

SrrTrackSectionA enthält eine <TriangleStripSet>, deren Koordinaten und Texturkoordinaten mit Hilfe der Ausgabewerte ***trackCoordinates***, ***alongVectors*** und ***normalVectors*** von SrrTrackGeometryABI berechnet werden.

Die Gleiselemente werden dabei durch eine vordefinierte Texturdatei realisiert.

### 5.4 SrrTurnoutLeftA, SrrTurnoutRightA

Diese Prototypen nehmen die Parameter

- objId
- vectorA
- vectorB0

- R
- neighbourA
- neighbourB0
- neighbourB1

entgegen, um 30°-Weichen zu kreieren (***SrrBasicTurnout2Way***). Ein Gleis ist gerade (von A zu B0), das andere ist eine 30°-Kurve mit Radius R.

Ein Hebel mit einem roten Knauf und touch sensor wird kreiert, um das Umstellen der Weiche (des enthaltenen SrrSwitchB-Knotens) zu ermöglichen.