Hibernation of Advanced Railroad Trains (ArrT)

Schritt Drei – ..... and Communicate them (aCt)

SRR Objects

Dieser Hibernation Report bemüht sich, einen Überblick über alle SRR Objekte zu geben.

THIS HR WILL BE TRANSLATED TO ENGLISH LANGUAGE SOON

1 Einleitung

SRR Objekte sind nichts anderes als MIDAS[1] Objekte.

Sie haben jedoch die besondere Eigenschaft, dass sie die Train Manager Extension des SMUOS Frameworks benötigen, sind also eng mit dem SRR Framework verbunden.

- SRR Framework = SMUOS Framework + Train Manager Extension

SRR Objekte helfen dabei,

- Gleise und Weichen zu modellieren,
- Schienenfahrzeuge zu modellieren (not yet implemented),
- und all diese miteinander funktional zu verknüpfen (Eisenbahnkinetik/-kinematik).

SRR Objekte gab es bereits bei der ersten LAN Party im März 2010 und durch das Re-Design, das seitdem im Gange ist, gibt es in bezug auf sie keine großen Neuerungen.

Es wird auch am Ende von Step 0033 nicht möglich sein Fahrzeuge zu kuppeln, Prellböcke und Kreuzungen wird es noch nicht geben und Kollisionen werden nicht berücksichtigt werden.

Trotzdem wird es bei der LAN Party #3 einige kleine Ergänzungen geben, die im März 2010 noch nicht vorhanden waren.

1. The master avatar container will be abandoned (moved to Simple Scene Controller)
2. Dynamic Modules will be supported
3. "Roles" will be abandoned and completely replaced by "Keys"
4. A MIDAS Base (MIB) will exist to ease implementation of MIDAS Objects
5. A "Beam to meeting place" function will exist
6. The SRR Framework will output status messages occasionally
7. A function will exist to reset all keys
8. There will be changes in the documentation and in the blogs
9. ~~??? It will be possible to use the SRR Framework from web spaces (monolithic layouts)~~
10. The base module of the SRR Framework will be replaced by the SMUOS Framework

1 MIDAS = Multiuser Interactivity Driven Animation and Simulation

2 Inhalt

Inhaltsverzeichnis

3 Literatur

[1] Dissertation; Hürlimann, D.; 2002;
Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen;
Institut für Verkehrsplanung und Transportsysteme, ETH Zürich, Zürich.

4 Was ist ein Doppelpunktgraph?

Wenn man Betriebsvorgänge im Eisenbahnwesen simulieren möchte, dann ist die Dissertation von Dr. D. Hürlimann an der ETH Zürich, [1], immer ein guter Anhaltspunkt.

Diesen Ansätzen folgend modelliert das SRR Framework die Gleistopologie durch einen sogenannten Doppelpunktgraphen.

Dabei wird jeder Gleisabschnitt durch eine Kante (Edge) dargestellt und jede Kante hat an ihren beiden Enden je genau einen Knoten (Node).

Jeder Knoten hat genau einen Nachbarknoten und jeder Knoten terminiert 0 oder mehr Kanten.

Damit kann man die wichtigsten Gleistypen darstellen.

Folgende Gleistypen werden in Step 0033 verfügbar sein:

- Gleisabschnitt,
- Zweiwegweiche.

Folgende Gleistypen sind mit diesem Ansatz ebenfalls darstellbar, werden aber erst in einem späteren Step realisiert werden:

- Dreiwegweiche,
- Kreuzung,
- Kreuzungsweiche,
- Prellbock.

Weitere Gleistypen, für die weiterführende Konzepte notwendig sein werden, umfassen zum Beispiel:

- Drehscheibe, Schiebebühne,
- Gleise auf Schienenfahrzeugen (z.B. um ein anderes Schienenfahrzeug zu transportieren).

Abbildung 1 zeigt ein Beispiel eines Doppelpunktgraphen für zwei Gleisabschnitte, eine Weiche und einen Prellbock.

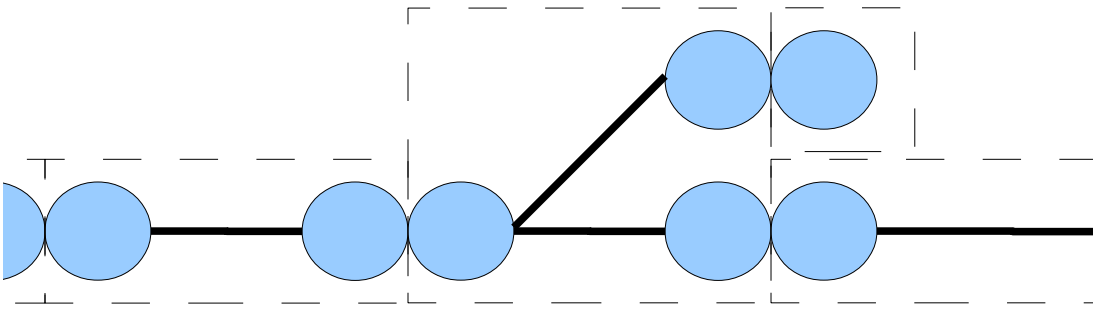

Abbildung 1: Gleiselemente im Doppelpunktgraphen

Der Knoten, der den Prellbock darstellt, terminiert keine einzige Kante, die Endknoten eines normalen Gleisabschnittes terminieren je eine Kante und der Knoten, der eine Weichenspitze modelliert, terminiert hier zwei Kanten.

Um solche Doppelpunktgraphen modellieren zu können, stellt das SRR Framework die SRR Objekte für Gleise und Weichen zur Verfügung.

Diese sind:

- SrrTrackEdge.....eine Gleiskante
- SrrTrackNode.....ein Gleisknoten
- MoosSwitchB.....ein Weichenantrieb
- SrrBasicTrackSection.....orchestriert zwei Gleisknoten und eine Gleiskante
- SrrBasicTurnout2Way.....orchestriert drei Gleisknoten, zwei Gleiskanten und einen Weichenantrieb

Über diese kann man im Kapitel 6 mehr erfahren.

5 Topologie versus Geometrie

Wie unschwer zu erkennen ist, stellt der Doppelpunktgraph bloss die logische Topologie einer Gleisanlage dar, nicht jedoch werden geometrische oder gar geographische Eigenschaften dargestellt, und sei es nur die geometrische Länge einer Kante in Metern.

Zu diesem Zweck definiert das SRR Framework ein Interface, mit dessen Hilfe man jeder Kante (also dem SRR Objekt SrrTrackEdge) geometrische Eigenschaften zuordnen kann, die sogenannte "track geometry".

Dem SRR Framework ist eine "beispielhafte Gleisgeometrie" beigelegt, die einen "track geometry node" (TGN) definiert und einige fertige Gleis- und Weichenmodelle, die auf diese "track geometry" aufbauen.

Über diese "beispielhafte Gleisgeometrie" findet man mehr im Kapitel 7 .

6 SRR Objekte für Gleise und Weichen (im Verzeichnis tmm/)

6.1 Überblick

Abbildung 2 zeigt einen Überblick über die Beziehungen zwischen den verschiedenen SRR Objekten für Gleise und Weichen

- Eine "Basic Track Section" orchestriert genau eine "Track Edge" und zwei "Track Nodes"
- Eine "Basic Turnout 2-Way" orchestriert genau zwei "Track Edges", drei "Track Nodes" und einen "Weichenantrieb"
- Jede "Track Edge" enthält genau einen TGN

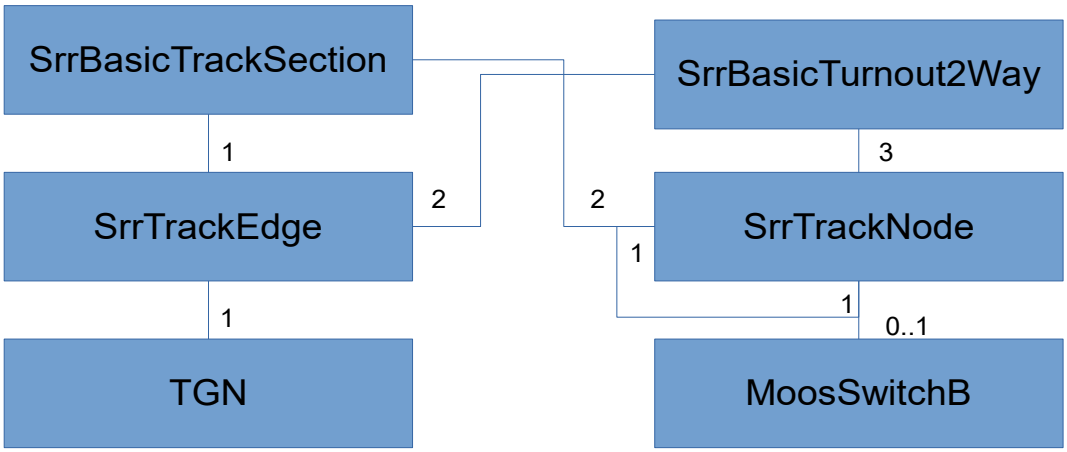

Abbildung 2: Überblick über alle SRR Objekte für Gleise und Weichen

Die SRR Objekte für Gleise und Weichen unterstützen nur die MOOs "LOADED", II und V, das heisst, sie lassen sich nur als gebundene Objekte einsetzen.

Von den Feldern des uiObj Interfaces werden also die folgenden Felder nicht angeboten:

"universalObjectClass", "commParam" und "initialized",

während folgende Felder tatsächlich angeboten werden:

"objId", "modParam", "disable", "parentObj", "attached", "enabledOut" und "dependentMobs".

6.2 SrrTrackNode

Ein "SrrTrackNode" SRR Objekt modelliert einen Gleisknoten im Doppelpunktgraphen.

Es hat immer eines der beiden Objekte SrrBasicTrackSection oder SrrBasicTurnout2Way als Elternobjekt und bietet folgende Parameter:

Sei N die Anzahl der Gleiskanten, die von diesem Knoten terminiert werden.

Das Feld **name='neighbourName' type='SFString' value=''**

enthält die "objId" des gewünschten Nachbarknotens, mit dem der Knoten bei der "Verlinkung der Gleise" verlinkt werden soll. Dieses Feld ist vom accessType = "initializeOnly" und kann somit nur einmal gesetzt werden.

Das Array **<field name='trackEdges' type='MFNode' value=''/>**

enthält Pointer auf alle N Kanten, die von diesem Knoten terminiert werden. Dieses Feld wird beim (Re-)Attachment des Elternobjektes auf die richtigen Werte gesetzt und ist bei der "Verlinkung der Gleise" bereits verwendbar.

Das Array **name='isNodeA' type='MFBool' value=''**

gibt für alle N Kanten, die von diesem Knoten terminiert werden, an, ob es sich um die A-Seite oder die B-Seite der Kante handelt. Dieses Feld wird beim (Re-)Attachment des Elternobjektes auf die richtigen Werte gesetzt und ist bei der "Verlinkung der Gleise" bereits verwendbar.

Das Feld **name='neighbour' type='SFNode' value='NULL'**

zeigt als Pointer auf den Nachbarknoten dieses Knotens. Dieses Feld ist genau dann NULL, wenn der Knoten mit keinem Nachbarn verlinkt ist. Dieses Feld wird durch das "Verlinken der Gleise" gesetzt und beim "Entlinken des Knotens" wieder auf NULL zurückgesetzt. Das gilt auch, wenn der Nachbarknoten von diesem Knoten entlinkt wird

Das Feld **name='turnoutSwitch' type='SFNode' value='NULL'**

enthält einen Pointer auf den "Weichenantrieb", wenn N größer als 1 ist. Wenn N kleiner oder gleich 1 ist, enthält dieser Pointer den Wert NULL.

Das Feld **name='gauge' type='SFFloat' value='1.435'**

enthält die Spurweite des Gleises, die an diesem Knoten gültig ist. Nachbarknoten sollten immer dieselbe Spurweite haben, aber entlang einer Kante kann eine verlaufende Spurweite modelliert werden, indem man am A-Knoten und am B-Knoten der Kante unterschiedliche Spurweiten setzt.

6.3 SrrTrackEdge

Ein "SrrTrackEdge" SRR Objekt modelliert eine Gleiskante im Doppelpunktgraphen.

Es hat immer eines der beiden Objekte SrrBasicTrackSection oder SrrBasicTurnout2Way als Elternobjekt und bietet folgende Parameter:

Das Feld **name='trackNodeA' type='SFNode' value='NULL'**

enthält einen Pointer auf den Gleisknoten an der A-Seite der Gleiskante. Dieses Feld wird bei der "Verlinkung der Gleise" gesetzt, wenn der Knoten mit seinem Nachbarknoten verlinkt wird. Bei der "Entlinkung des Knotens" wird dieses Feld auf den Wert NULL zurückgesetzt. Das gilt auch, wenn der Nachbarknoten von diesem Knoten entlinkt wird.

Das Feld **name='trackNodeB' type='SFNode' value='NULL'**

enthält einen Pointer auf den Gleisknoten an der B-Seite der Gleiskante. Dieses Feld wird bei der "Verlinkung der Gleise" gesetzt, wenn der Knoten mit seinem Nachbarknoten verlinkt wird. Bei der "Entlinkung des Knotens" wird dieses Feld auf den Wert NULL zurückgesetzt. Das gilt auch, wenn der Nachbarknoten von diesem Knoten entlinkt wird.

Das Feld **name='trackGeometry' type='SFNode' value='NULL'**

enthält einen Pointer auf einen TGN. Jeder "normalen" Gleiskante muss genau ein TGN zugeordnet werden, welcher die geometrischen und geographischen Eigenschaften der Gleiskante behandelt.

Das Feld **name='exitViewpoint' type='SFNode' value='NULL'**

enthält einen Pointer auf einen X3D <Viewpoint> Knoten.

Wenn ein User ein Schienenfahrzeug betritt, dann muss man einen <Viewpoint> "binden", der sich mit dem Schienenfahrzeug mitbewegt, damit der User vom Fahrzeug "mitgenommen" wird. Wenn nun der User das Fahrzeug verläßt, muss man wiederum einen <Viewpoint> binden, der gegenüber der Landschaft ruht.

Dieses Feld gibt dem Szenenautor die Möglichkeit, jeder Gleiskante einen <Viewpoint> zuzuordnen, der vom SRR Framework gebunden wird, sobald der User das Fahrzeug verläßt und wenn das Fahrzeug soeben diese Gleiskante befährt.

6.4 Track Geometry Nodes (TGNs) am Beispiel SrrTrackSectionA.x3d

Über den TGN greift das SRR Framework auf die geometrischen Eigenschaften einer Gleiskante zu.

Der TGN kann auch Berechnungen durchführen, um dem umgebenden Gleismodell eine Basis für die graphische Darstellung des Gleises zu liefern.

Das wird am besten sichtbar, wenn man einen Blick in die Datei tg/SrrTrackSectionA.x3d macht.

Diese implementiert ein Modell eines Gleisabschnitts, der einer Gleiskante entspricht mit Hilfe der "ABI Gleisgeometrie". Die "ABI Gleisgeometrie" ist im TGN "SrrTrackGeometryABI.x3d" festgelegt.

Hier sieht man zuerst die SRR Objekte und danach die graphischen Elemente, die dann wirklich sichtbar werden:

```
<!--  ***** SRR objects to instrument the track section *****  -->
          <ProtoInstance DEF='TrackSection' name='SrrBasicTrackSection'>
            <fieldValue name="dependentMobs">
              <ProtoInstance DEF="TrackNodeA" name="SrrTrackNode">
                <fieldValue name="gauge" value="0.545"/>
                <IS>
                  <connect nodeField="neighbourName" protoField="neighbourA"/>
                </IS>
              </ProtoInstance>
              <ProtoInstance DEF="TrackNodeB" name="SrrTrackNode">
                <fieldValue name="gauge" value="0.545"/>
                <IS>
                  <connect nodeField="neighbourName" protoField="neighbourB"/>
                </IS>
              </ProtoInstance>
              <ProtoInstance DEF="TrackEdge" name="SrrTrackEdge">
                <fieldValue name="trackGeometry">
                  <ProtoInstance DEF='TrackGeometry' name="SrrTrackGeometryABI">
                    <fieldValue name="trackElementLength" value="0.5"/>
                    <IS>
                      <connect nodeField="vectorA" protoField="vectorA"/>
                      <connect nodeField="vectorB" protoField="vectorB"/>
                      <connect nodeField="vectorI" protoField="vectorI"/>
                      <connect nodeField="normalA" protoField="normalA"/>
                      <connect nodeField="normalB" protoField="normalB"/>
                    </IS>
                  </ProtoInstance>
                </fieldValue>
                <IS>
                  <connect nodeField="exitViewpoint" protoField="exitViewpoint"/>
                </IS>
              </ProtoInstance>
            </fieldValue>
          </ProtoInstance>
<!--  ***** graphical representation of the track section *****  -->
          <Switch DEF="OnOffSwitch">
            <Shape>
              <TriangleStripSet DEF='VisualTrackGeometry' stripCount="3">
                <Coordinate DEF='VisualTrackCoordinates' point='0 0 0, 1 0 0, 0 1 0'/>
                <TextureCoordinate DEF='VisualTrackTextureCoordinates' point='0 0, 1 0, 0 1'/>
              </TriangleStripSet>
              <Appearance>
                <Material />
                <ImageTexture url='"../tg/trackElementA.jpg"'/>
              </Appearance>
            </Shape>
          </Switch>
        </Group>
```

Der TGN (rot markiert) ist hier als Kind des SrrTrackEdge-Knotens eingefügt und bekommt vom User des Modells die Werte "vectorA", "vectorB", "vectorI", "normalA" und "normalB" übermittelt.

Diese Werte werden dann benützt, um einen Kreisabschnitt zu berechnen, der die Basis des Gleisabschnitts darstellt.

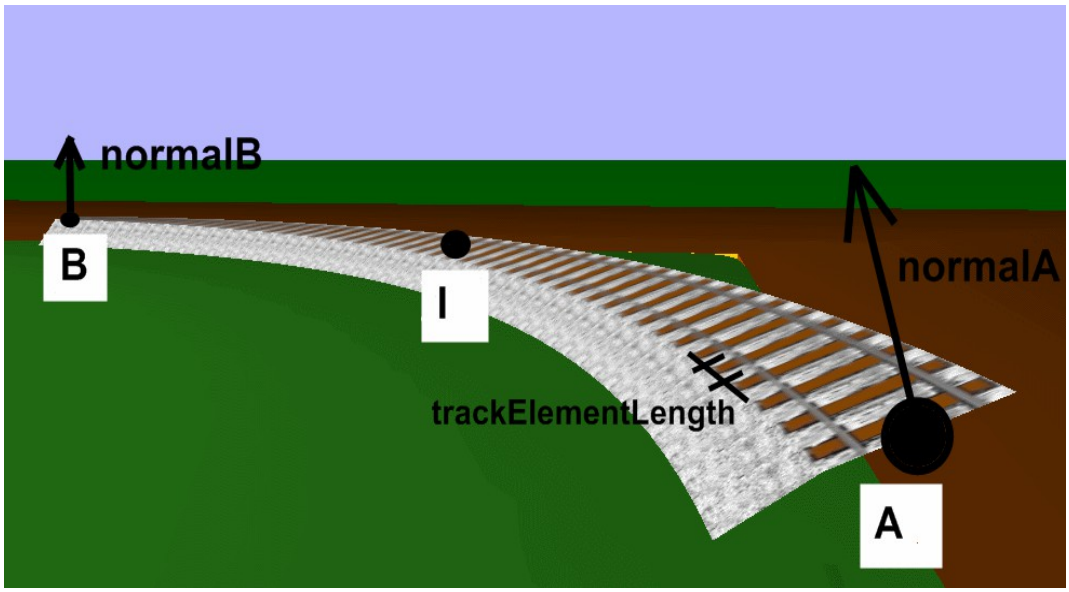

Abbildung 3: Parameter der ABI Gleisgeometrie

Wenn man den TGN (re-)initialisiert, dann nimmt er diese Parameter und den Parameter "trackElementLength", berechnet zuerst den Kreisabschnitt intern und den externen Parameter "length", zerteilt die Länge "length" in eine ganze Anzahl von Gleiselementen (hier Schwellen) und gibt die Parameter "trackCoordinates", "alongVectors" und "normalVectors" aus.

Diese Ausgabeparameter sind Arrays von Vektoren, die von einem Skript (hier nicht dargestellt) verwendet werden, um die geometrischen Eigenschaften des <TriangleStripSet/> festzulegen, bevor dieses mit Hilfe des Schalters "OnOffSwitch" sichtbar gemacht wird.

Der <ImageTexture/> Knoten wird dabei so skaliert, dass das Bild "tg/trackElementA.jpg" genau "length" / "trackElementLength" mal dargestellt wird.

Jeder TGN (also nicht nur der TGN der "ABI Gleisgeometrie") MUSS folgende Parameter anbieten:

Das Feld **name='objId' type='SFString' value=''**

dient dazu, dem TGN eine <objId> zuzuordnen.

Das Feld **name='parentObj' type='SFNode' value='NULL'**

dient dazu, die <objId>s der Elternobjekte festzustellen. Damit kann dann eine zusammengesetzte <objId> erzeugt werden, z.B. für Tracer Ausgaben.

Das Feld **name='modParam' type='SFNode' value='NULL'**

wird verwendet, um den TGN zu (re-)initialisieren. Hier wird der Begriff "(Re-)Initialisierung" verwendet, da ein TGN eigentlich kein SRR Objekt ist.

Mit dem Feld **name='attached' type='SFNode'**

muss der TGN mitteilen, ob die (Re-)Initialisierung erfolgreich war. Der Wert initialized=NULL bedeutet, dass die (Re-)Initialisierung schief gegangen ist.

Das Feld **name='enabledOut' type='SFBool'**

muss auf den Wert "false" gesetzt werden, falls der TGN disablet worden ist.

Der TGN muss das Feld **name='length' type='SFFloat' value='0.0'**

spätestens bei der Initialisierung auf den richtigen Wert setzen.

Mit dem Feld **name='calculateAxleTransformation' type='SFNode'**

beauftragt man einen TGN, eine Achse relativ zur Elternkante dieses TGNs darzustellen. Wenn man dem TGN an diesem Feld den Pointer "Value" auf eine Achse übergibt, dann muss der TGN die Eigenschaften der Achse auswerten und dann die Eigenschaft Value.transformation.transformation dieser Achse auf den richtigen Wert setzen.

Mit dem Feld **name='disable' type='SFTime'**

kann man den TGN disablen.

6.5 SrrBasicTrackSection und SrrBasicTurnout2Way

Diese SRR Objekte sind die "Elternobjekte", um alle SRR Objekte je eines Gleistyps zu "orchestrieren".

SrrBasicTrackSection orchestriert 2 Gleisknoten und eine Gleiskante zu einem generellen Gleisabschnitt. Die Gleiskante muss dabei einen TGN enthalten.

SrrBasicTurnout2Way orchestriert 3 Gleisknoten und 2 Gleiskanten zu einer Zweiwegweiche. Die Gleiskanten müssen dabei je einen TGN enthalten und der Gleisknoten an der Weichenspitze muss einen Weichenantrieb enthalten.

6.6 Orchestrierung der Gleise und Weichen

Die Systematik der MIDAS Objekte (in Form des Prototypen MibCore) sorgt dafür, dass zuerst die dependent MOBs initialisiert und attached werden – also

- TGN (Track Geometry Node) – einer pro Track Edge,
- MoosSwitchB (Weichenantrieb) – einer pro Weiche,
- SrrTrackNode – 2 Stück bei einem Gleisabschnitt und 3 Stück bei einer Weiche und
- SrrTrackEdge – eine bei einem Gleisabschnitt und zwei bei einer Weiche,

bevor die "Elternobjekte" initialisiert und attached werden, also

- SrrBasicTrackSection
- SrrBasicTurnout2Way

Da der Modulkoordinator (genau gesagt die TMM Extension des Modulkoordinators) sowohl eine Liste aller Track Edges des Moduls als auch eine Liste aller Track Nodes des Modul führt,

announcen sich diese beiden Knoten während ihres (Re-)Attachments beim MC und warten auch auf die Erfolgsmeldung, bevor das "Elternobjekt" weiter initialisiert und attached werden kann.

SrrTrackEdge überprüft zusätzlich, ob es tatsächlich einen TGN enthält.

Danach – während des Attachment des "Elternobjektes" – setzt dieses einige Pointer in den Track Edge und Track Node Objekten, damit dann die "Verlinkung der Gleise" ordnungsgemäß stattfinden kann.

Noch sind die Gleise nämlich nicht miteinander verlinkt, eine "Rollen der Räder" ist noch nicht möglich.

6.6.1 Verlinkung der Gleise

Die "Verlinkung der Gleise" wird in Step 0033.11 des SrrTrains v0.01 Projektes implementiert werden.

6.6.2 Entlinkung eines Knotens

Die "Entlinkung von Knoten" und "Neuverlinkung von Gleisen" wird erst in einem Step nach Step 0033 implementiert werden.

7 Die "Example Track Geometry" (im Verzeichnis tg/)

Das SRR Framework (das auf das SMUOS Framework aufbaut), enthält eine beabsichtigte Lücke.

Wir haben von anfang an nur eine sehr simple, man könnte sagen rudimentäre, Gleisgeometrie implementiert, die jeden Gleisabschnitt auf einen Kreisbogen reduziert.

Dieser Kreisbogen ist zwar immerhin räumlich ausgelegt – sodass man auch Kuppen und Täler im Gleis modellieren kann – bis hin zu Achterbahnen – aber dem Eisenbahnfachmann fehlen die geliebten Übergangsbögen. Diese sind zur Zeit nicht modellierbar.

Um hier Abhilfe zu schaffen, haben wir die Gleisgeometrie sozusagen "ausgelagert".

Das heisst: wir haben im SRR Framework eine Lücke gelassen, in die jeder seine eigene Gleisgeometrie einsetzen kann, haben aber dem SRR Framework eine "Example Track Geometry" samt einigen Modellen für Gleisabschnitte und Weichen beigelegt, eben jene "rudimentäre Gleisgeometrie".

Diese befindet sich im Verzeichnis tg/ und ist ebenfalls mit einer LGPL lizenziert.

8 SRR Objekte für Schienenfahrzeuge (im Verzeichnis tmm/)

Diese werden erst im Step 0033.11 des SrrTrains v0.01 Projektes implementiert.

Erste Ideen sollen im folgenden dargestellt sein.

8.1 Grundlegende Gedanken

8.1.1 Der Zustand eines Zuges

Prinzipiell sollen Schienenfahrzeuge und Züge über die Gleise bewegt werden, indem man einzelne Achsen über die Gleise bewegt, nämlich mit Hilfe des SRR Objektes SrrAxle.

Aus den translativen und den rotatorischen Positionen der Achsen sollen dann die Positionen der Drehgestelle und Fahrzeuge hergeleitet werden.

Dabei sollen alle Achsen eines Zuges insoferne gleichgeschaltet sein, als sie sich relativ zum Gleis mit derselben Geschwindigkeit bewegen. In Längsrichtung sei der Zug also ein ***starrer Stab***.

Diese Geschwindigkeit – die Geschwindigkeit des Zuges – und die translative Position der Achsen relativ zum Gleis soll von einem SRR Objekt berechnet werden, das für jeden Zug genau einmal existiert. Es ist das das SRR Objekt ***SrrTrainState***.

SrrTrainState entspricht einer "unsichtbaren Referenzachse", die den Bezugspunkt des Zuges definiert. Sobald und solange der SrrTrainState aktiv ist, liefert er gültige Werte für

- Modul,
- Track Edge,
- isAtoB und
- ess

der unsichtbaren Referenzachse. Damit ist die absolute Position der Referenzachse, also ***absPosTrain*** definiert.

SrrTrainState liefert darüber hinaus Werte für die Geschwindigkeit und die Beschleunigung des Zuges:

- ***velocityTrain***
- ***accelerationTrain***

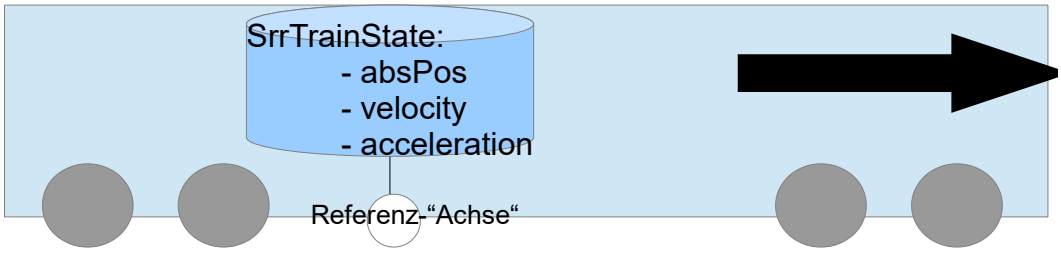

Abbildung 4: Zustand eines Zuges, bezogen auf eine "unsichtbare Referenzachse"

8.1.2 Zugriff auf alle Teile des Zuges

Jeder Zug hat ein astrales SRR Objekt ***SrrTrainBus***, das es allen Teilen des Zuges ermöglicht, innerhalb einer Szeneninstanz direkt miteinander in Kontakt zu treten.

Der SrrTrainBus speichert unter anderem einen MFFloat Wert, nämlich die "axleOffsets", sodass sich die absolute Position einer Achse [i] jederzeit zu

- absPosAxles[i] = absPosTrain + axleOffsets[i]

ergibt.

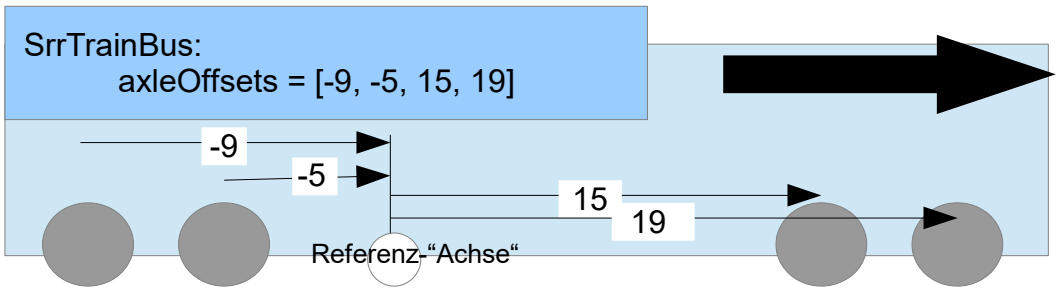

Abbildung 5: Beispiel für axleOffsets[]

Dieser MFFloat Wert "axleOffsets" und andere Parameterwerte, die der SrrTrainBus speichert, können sich immer dann ändern, wenn

- ein Zug "gegründet" wird, wenn
- zwei Züge zu einem Zug "vereinigt" werden, oder wenn
- ein Zug in zwei Züge "aufgetrennt" wird.

In SrrTrains gibt es keine Fahrzeuge, die nicht Teil eines Zuges sind.

Es kann aber natürlich Züge geben, die nur aus einem Fahrzeug bestehen, und das auch noch ohne Antrieb (z.B. abgestellte Waggons).

Die oben genannten "Prozeduren", also ***Gründung, Vereinigung und Trennung von Zügen***, werden aus sogenannten "Sub-Prozeduren" zusammengesetzt. Darüber später mehr.

8.1.3 Antriebe und Führerstände

Nun wollen wir aber auch in der Lage sein, ein Schleudern oder ein Blockieren einzelner Antriebe – oder sogar einzelner Achsen – zu modellieren.

Wir werden also die SrrAxle Objekte nicht direkt vom SrrTrainState ansteuern.

Deshalb fassen wir immer eine oder mehrere Achsen mit Hilfe eines Antriebs zusammen.

Das SRR Objekt ***SrrDrive*** enthält also immer ein oder mehrere SRR Objekte ***SrrAxle***.

Jedes SrrAxle Objekt ist in genau einem SrrDrive Objekt enthalten, ausgenommen die Referenzachse, die direkt dem SrrTrainState Objekt zugeordnet ist.

- Der SrrTrainState verteilt die deltaEss Events also an alle SrrDrives, die sie dann an die SrrAxles weiterleiten.
- Die rotatorische Position bekommen die SrrAxle Objekte direkt von den SrrDrive Objekten.

Jetzt bleibt noch die Frage, wie die Beschleunigung des Zuges zustande kommt.

- Dazu liefert jedes SrrDrive Objekt eine Kraft eff an den SrrTrainState

Jeder Antrieb wird von höchstens einem Führerstand gesteuert. Jeder Führerstand kann beliebig viele Antriebe steuern.

Wir haben also folgenden Ablauf der Eisenbahnkinetik:

A) Jedes SrrDrive Objekt wird von höchstens einem Führerstand gesteuert und berechnet kontinuierlich den Kraftbeitrag zur Beschleunigung des Zuges.
Dabei wird jeder Achse ein bestimmter Anteil der Gesamtmasse zugeordnet, damit die Wirkung der Schwerkraft gleich subtrahiert werden kann

B) Das SrrTrainState Objekt berechnet aus den Kraftbeiträgen und der Gesamtmasse des Zuges eine Beschleunigung und eine Geschwindigkeit, daraus ein deltaEss Event

C) Das deltaEss Event wird vom SrrTrainState an alle SrrDrives und damit an alle SrrAxles verteilt. Die Referenzachse wird von SrrTrainState extra berücksichtigt

D) Aus den deltaEss Events ergeben sich die translatorischen und rotatorischen Positionen der Achsen

E) Die translatorischen und rotatorischen Positionen der Drehgestelle, Fahrzeuge und sonstigen Teile des Zuges werden aus den translatorischen und rotatorischen Positionen der Achsen berechnet

F) Um die rotatorische Position der Achsen kümmern sich die SrrDrive Objekte

8.1.4 SrrVehicleTrains versus SrrTrains

Da wir in Step 0033.11 das Kuppeln und Entkuppeln von Fahrzeugen noch gar nicht implementieren, können wir uns vorerst auf "one-vehicle-trains" zurückziehen.

Anstatt also einen Zug aus mehreren UBOs zusammenzusetzen (aus einem "Zug", aus mehreren "Fahrzeugen" und aus "Fahrzeug-zu-Zug-Beziehungen"), werden wir immer nur einzelne UBOs erzeugen, löschen, laden oder entladen.

Trotzdem werden wir in diesem Paper einige Gedanken zu zusammengesetzten Zügen wälzen.

8.2 One-Vehicle-Trains

Ein "one-vehicle-train" ist ein UBO der UOC "SrrVehicleTrains", enthält aber auch schon SRR Objekte, die später für zusammengesetzte Züge Verwendung finden sollen.

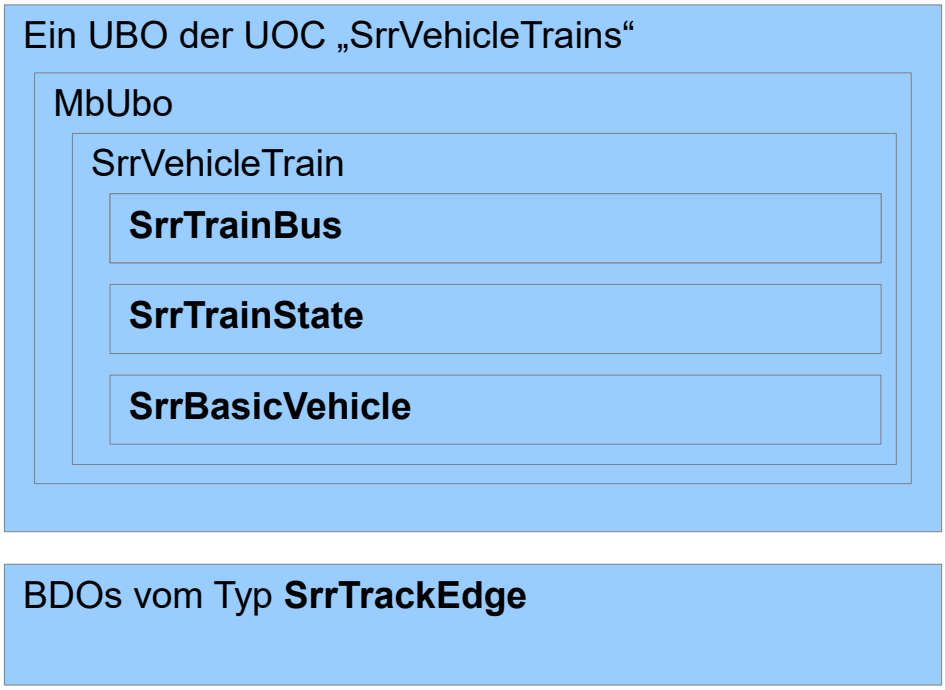

Figure 1: Prinzipieller Aufbau eines "SrrVehicleTrains" UBO

MbUbo ist die Model Base für UBOs.

SrrVehicleTrain ist das "Master" SRR Objekt für "one-vehicle-trains" und es orchestriert

- ein MIDAS Objekt ***SrrTrainBus***,
- ein MIDAS Objekt ***SrrTrainState*** und
- ein MIDAS Objekt ***SrrBasicVehicle*** mit seinen "üblichen" dependent objects
  - ***SrrCabs*** + ***SrrCab (N mal)***
  - ***SrrDrives*** + ***SrrDrive (N mal)***
  - ***SrrAxle (N mal)***

SrrTrainBus kann auf alle Elemente des Zuges innerhalb einer Szeneninstanz zugreifen. Alle Elemente des Zuges innerhalb einer Szeneninstanz können auf den SrrTrainBus zugreifen.

SrrCabs enthält N Cabs eines Fahrzeuges.

SrrDrives enthält N Drives eines Fahrzeuges.

Jedes SrrDrive enthält N SrrAxles.

8.3 Zusammengesetzte Züge (not yet implemented)

Dieses Kapitel hält erste Ideen für zusammengesetzte Züge, die auf folgende SRR Objekte aufbauen. Diese SRR Objekte sollen sowohl für "one-vehicle-trains" als auch für zusammengesetzte Züge verwendet werden:

- MbUbo.....generelle Basis für UBOs
- SrrTrainBus.....SRR Objekt genau ein mal für jeden Zug (MOO III)
- SrrTrainState.....SRR Objekt genau ein mal für jeden Zug (MOO IV)
- SrrBasicVehicle.....SRR Objekt genau einmal für jedes Schienenfahrzeug
- SrrDrives + SrrDrive + SrrAxle.....SRR Objekte für jedes Schienenfahrzeug
- SrrCabs + SrrCab.....SRR Objekte für jedes Schienenfahrzeug

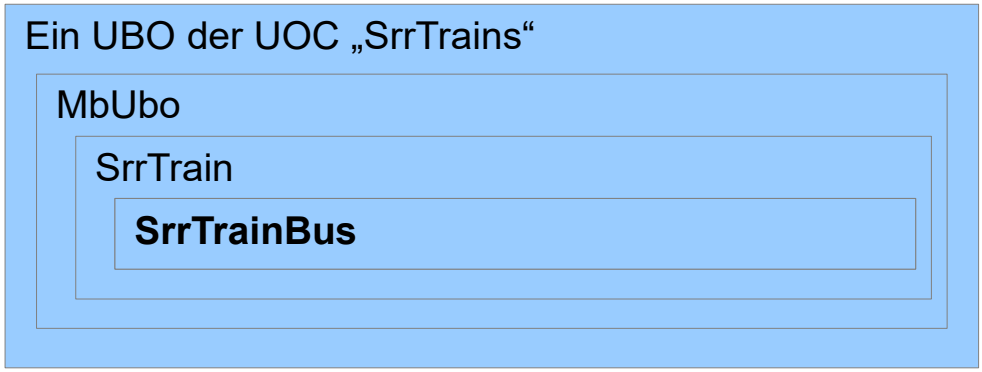

Abbildung 6: Prinzipieller Aufbau eines "SrrTrains" UBO

SrrTrain ist das "Master" SRR Objekt für zusammengesetzte Züge, es orchestriert

- ein SRR Objekt ***SrrTrainBus***,
- stellt die Verbindung zum SrrTrainState her und
- stellt die Verbindung zu allen SrrBasicVehicles her

Der Train State wird in einem externen UBO, dem sogenannten "***Abstract State Holder (ASH)***" gehalten (siehe unten). SrrTrain muss diesen externen SrrTrainState mit dem SrrTrainBus verknüpfen.

Die Fahrzeuge werden in externen UBOs der UOC "SrrRailVehicles" gehalten (siehe unten). Sie enthalten auch je ein SRR Objekt vom Typ SrrBasicVehicle, wobei SrrTrain den Kontakt zwischen den SrrBasicVehicles und dem SrrTrainBus herstellen muss.

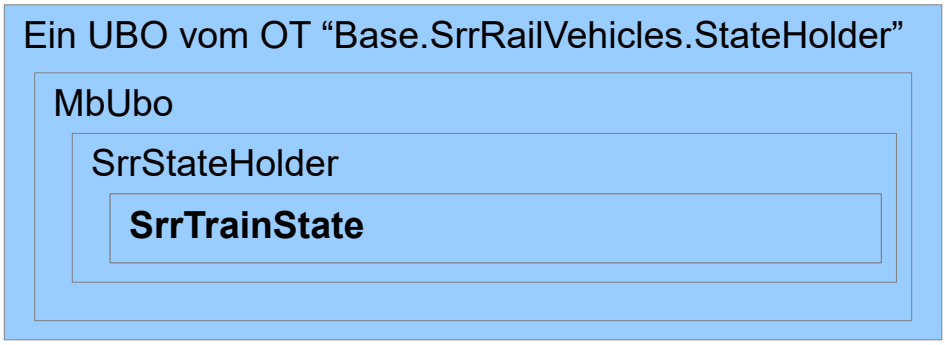

Abbildung 7: Prinzipieller Aufbau des "Abstract State Holder (ASH)"

SrrStateHolder ist das "Master" SRR Objekt für den "Abstract State Holder (ASH)".

Die Object ID des ASH wird von der Object ID des SrrTrain UBO hergeleitet.

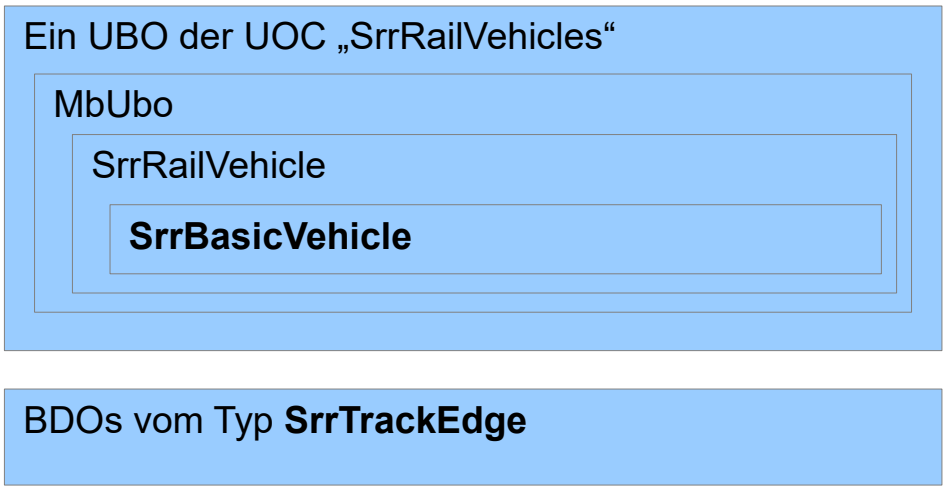

Abbildung 8: Prinzipieller Aufbau eines "SrrRailVehicles" UBO

SrrRailVehicle ist das "Master" SRR Objekt für Fahrzeuge in zusammengesetzten Zügen.

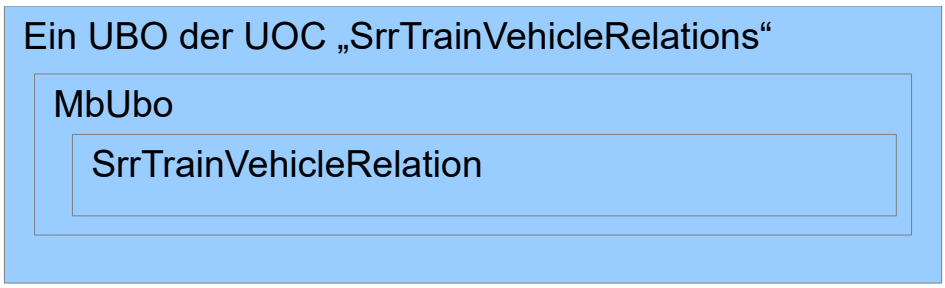

Abbildung 9: Prinzipieller Aufbau eines "SrrTrainVehicleRelations" UBO

SrrTrainVehicleRelation fügt Fahrzeuge (SrrRailVehicle) in Züge (SrrTrain) ein.