Subject: Re: V1.00 RC1: open discussion points, part 2
Posted by Matthias Hengartner on Tue, 02 Nov 2004 11:25:57 GMT
View Forum Message <> Reply to Message

Hello,

> I'm afraid I don't understand well the idea of 0...3 connections. 0
> connections are for crossings, 1 for single crossing switches (EKW) and 2
> for double crossing switches (DKW), am I right? But 3 connections?
> This means that a EKW has 1 connection ID (Right or wrong?), but for
> interlocking it needs 2 because it has 2 independent point machines with 2
> independent pairs of blades.

*** <answer> ***

If we have a <track> (Track1, see below) with any <crossing> (+), there are 2 <connection>-elements, one "incoming" and connected with Track2, the other "outgoing" and connected with Track3. So it doesn't matter if it's a single/double crossing switch or a simple crossing.

```
                   / Track3
                  /
Track1           /
  -----------+----------
            /
           /
          / Track2
```

The possibility to have 3 <connection>-elements is (as far as I know) only needed if we have a crossing on a <trackBegin>/<trackEnd>. In the case which is illustrated below, there is a crossing at the end of Track1, and we have 2 "outgoing" connections to Track3 and Track4 and 1 "incoming" connection to Track2.

```
                    / Track3
                   /
Track1            /
   -----------+---------- Track 4
             /
            /
           / Track2
```

*** </answer> ***

*** <idea> ***

Some days ago, I had an idea of another way of modelling a crossing,
which should also be possible:
If two tracks (see below) cross each other, it should be possible, that
BOTH of them are continuous.

```
                    /
                  /
Track1          /
   ----------+-----------
              /
            /
          / Track2
```

At least in the case of a "simpleCrossing" it should be possible to do it
this way.

How to implement this?

One possible way is the following:

Given the picture above and assuming that Track1 goes from the left to the
right and Track2 from bottom (left) up (right)

So we could have the following implementation in railML:

in Track1:

```
<crossing>
   <connection connectionID="C1a" branchIDRef="C2b"
branchTrackIDRef="Track2" orientation="outgoing"/>
   <connection connectionID="C1b" branchIDRef="C2a"
branchTrackIDRef="Track2" orientation="incoming"/>
</crossing>
```

in Track2:

```
<crossing>
   <connection connectionID="C2a" branchIDRef="C1b"
branchTrackIDRef="Track1" orientation="outgoing"/>
```

```
   <connection connectionID="C2b" branchIDRef="C1a"
branchTrackIDRef="Track1" orientation="incoming"/>
</crossing>
```

When the crossing is a right-angled "simpleCrossing", the orientation would be "rightAngled", of course.

Advantage of this solution:
There is a <crossing>-element in both tracks, so it's "symmetric" and both track are treated equally.

Disadvantages:
1) There are 2 <crossing>-elements for 1 crossing
2) Due to 1), there is some redundancy (which could also result in inconsistency)

I'll give you another possible way of implementing this later (in another context).

*** </idea> ***

Best regards
Matthias Hengartner

Tel G: +41 1 633 68 16
hengartner@ivt.baug.ethz.ch
