Subject: Re: V1.00 RC1: switchRef/crossingRef
Posted by Volker Knollmann on Tue, 05 Oct 2004 06:53:05 GMT
View Forum Message <> Reply to Message

On 04.10.2004 14:57, Matthias Hengartner wrote:
>> I suggest an additional value "straight" (which perfectly coincides with
>> the possible values for "trackContinueCourse")
>
>
> In case (1a), I'd take "outgoing" for _both_ connected tracks (1b:
> "incoming"). The attribute "course" would have the value "straight" for the
> second connected track (and "left"/"right" for the first [1a/1b]).
> So in my opinion, we _could_ introduce the value "straight" in the
> "orientation"-attribute, but there's no need for it.

Waaaaaaaaaaaaaaaah

Shame on me! I didn't see the attribute "course" in the
<connection>-element. With the proper setting of "course", no further
attributes or values are required, you are right. Thanks for pointing me
on that...

>> and the __convention__ to
>> let the <switch>-element be part of track at the switch's tip. Thus,
>> role of every track is unambiguous.
>
>
> I agree fully with you!
> [...]
> But this is of course a very "dirty" implementation. And I don't think that
> the possibility to implement case (2) is really needed (It can easily be
> avoided).

Full ack. But the thing is to take EVERY case into account that CAN be
realized in railML and therefore must be handled by the interpreting
software.
Of course there's no need to implement switches in such a queer way like
case (2). But sooner or later, someone will fail to withstand the
temptations of the dark side of the force^W^W^W eeeeer railML... ;-)

> Another possibility... We could abandon the special treatment of
> switches/crossings which are placed on trackBegin/trackEnd (I feel a little
> uncomfortable about saying this, because this idea is penned by me...).
> However, then we'd have a <simpleConnection> and a <switch> which have the
> same position (on trackBegin/trackEnd).

We are brothers in mind.
That's exactly the way I defined the implementation of switches for data

exchange in our lab. Given a switch like this:

```
           o
          /
         /
        /
o-----------------o
```

I transformed this into two <track>-elements in railML:

```
     o
    /
   /
  /
o
o------------------o
```

with a switch of zero size at pos="0.000" of the lower track. This
implementation has another advantage: you can easily set up different
speed restrictions for the branching and the straight part, because you
have two <track>-elements (I know that there is a "vMax"-attribute for
the <connection>-element, but it was introduced too late for me).

> Or, final idea (for the moment ;-)): We could combine these 2 approaches:
> We could have a <simpleConnection> with a reference to a
> <switch>/<crossing>.

Hmm? That would be the above mentioned way: define a switch at
pos="0.000" or pos="[length]" and refer to it from the branching
element. The straight element is implicitly given by the parent of
<switch>. This avoids some extra attributes and the handling of special
cases....

Best regards from Braunschweig,
Volker Knollmann