Subject: Re: Switch: usage of attribute @course
Posted by Thomas Nygreen JBD on Mon, 09 Apr 2018 12:57:16 GMT
View Forum Message <> Reply to Message

Thomas Nygreen wrote on Thu, 05 April 2018 17:38I have read Claus Feyling's post, and, as I have argued above, I disagree with the proposal to change the interpretation of @course. Also, I am not convinces about the need for @geometry, as it in most cases follows from the topology, and in remaining cases, such as the one he gives, can be described by switch/@trackContinueRadius and connection/@radius. Although, as the attachments are missing, there could be something I am missing.

I have now read the Q&A attachment (thanks to Christian Rahmig for sending it to me), and given some more thought to the suggestion of a new geometry attribute. As I will show below, @geometry is not strictly necessary. However, I now find @geometry to be a good addition, since is does not depend on using topology to deduce geometry, but separates the two. It also allows a more strict interpretation of the enumerated values for @course.

Before I continue, I would like to state that, regarding @course, I still prefer the current interpretation of "right" and "left" as seen in the "up" direction. This question is, however, completely separable from the other suggestions by Claus Feyling.

First, regarding stricter use, I have found no documentation of the meaning of the values "right", "left" and "straight" for @trackContinueCourse. Claus Feyling only uses "straight" for three-way switches, while I have always assumed that it can also be used for two-way switches. If all three values can be used also for two-way switches, the following combinations are all topologically equivalent, as they all have a branch further left than the continuing track:

@trackContinueCourse = "straight" and connection[1]/@course = "left"@trackContinueCourse = "right" and connection[1]/@course = "left"@trackContinueCourse = "right" and connection[1]/@course = "straight"

All of these are valid under the railML XSD, but they might not all be desirable. They can be used to reflect the physical geometry of the switch, but again, that might not be desirable. Rather, I prefer to separate the geometry from the topology, and include the suggested @geometry. This also allows reducing the three possible alternatives above to just one.

Furthermore, I suggest:

making connection/@course mandatory, as it is not possible to determine topology without itdeprecating @trackContinueCourse, as it is given by the remaining direction not used by any connection

Best regards
Thomas