Acta Polytechnica CTU Proceedings **43**:18–27, 2023
https://doi.org/10.14311/APP.2023.43.0018

DETERMINING 3D COORDINATES BASED ON THE TRACK GEOMETRY DESCRIPTION

Adam Hlubuček

Czech Technical University in Prague, Faculty of Transportation Sciences, Department of Transport Telematics, Konviktská 20, 110 00 Prague 1, Czech Republic

correspondence: hlubuada@fd.cvut.cz

Abstract. This paper is focused on the data description of the railway infrastructure. Its aim is to present the possibilities of constructing of linear elements representing tracks in 2D and their subsequent transformation into 3D using the parametric description of horizontal and vertical curves. The transformation is based on determining the spatial coordinates of the newly emerging 3D linear elements. This process is supposed to be implemented in such a graphical editor environment that allows the relevant data to be transferred to the the database based on the Multipurpose Railway Infrastructure Model. With the use of this data, it is possible to perform, among other things, the visualization of the railway infrastructure.

Keywords: Railway infrastructure, track geometry, data description, Multipurpose Railway Infrastructure Model, RailTopoModel.

1. Introduction

In recent years, the data description of railway infrastructure has become more important, as many intelligent transport systems are being developed, for which it is an essential input. It is desirable to address the question of how to describe the infrastructure in such a way that the description can be used for the widest possible range of target applications [1]. This is what also the Multipurpose Railway Infrastructure Model aims for.

The Multipurpose Railway Infrastructure Model is a data model reflecting some principles of the UIC RailTopoModel [2, 3] with data stored in the form of a relational database. It is gradually being developed in the Railway Laboratory at CTU in Prague, Faculty of Transport Sciences [4], whereas its latest version, referenced in this paper, is the version 12.2.

One of the fundamental aspects of the railway infrastructure description in an expression of the track geometry are the spatial characteristics of the railway line which are the basis for the location of many other infrastructure facilities. When expressing the track geometry, there are several different ways to do this. The Multipurpose Railway Infrastructure Model allows us to express the individual points of the centre line of the track described by coordinates as well as to describe its geometric parameters analytically, using the appropriate attributes [5, 6].

Both these approaches have their specific advantages and cases in which it is appropriate to apply them. Therefore, in order to create a consistent data description, it is necessary to design software tools that allow both of these approaches to be used so that the outputs provided are in accordance with each other.

2. Consistent filling the model with data

For the purpose of filling the Multipurpose Railway Infrastructure Model database with data, a specialized editing script working in the environment of a computer-aided design software is being developed. In the following text, this script together with the computer-aided design software will be referred to as the graphical editor.

The graphical editor allows individual instances of the model classes to be visualized as graphical objects arranged into corresponding layers. These graphical objects can also be described by relevant data. Each graphical object is described by several items corresponding to the attributes of the respective class.

Within a specific graphical object, each item is expressed using a record of the structure

$$[table][attribute][attributeValue],$$

where *table* expresses the name of the table, in which the value of the respective attribute is to be stored, *attribute* expresses the name of the attribute to be stored and *attributeValue* expresses its value. On the basis of such a description of all graphical object belonging to layers corresponding to the Multipurpose Railway Infrastructure Model classes, the data from the computer-aided design software can be uploaded to the relational database, based on its structure.

The mutual consistency of individual data items, which cannot be verified at the database level, is supposed to be ensured on the basis of functionalities of the graphical editor. Among other things, the graphical editor provides the creation of graphical object representing individual data objects of the Multipurpose Railway Infrastructure Model and describing them with respective data items. Some of the data is entered by the graphical editor user, other can be obtained based on the spatial aspects of these objects in the environment of the computer-aided design software. The way the data objects are graphically visualized usually depends on the class which they belong to.

In addition, the model makes it possible to assign some data objects to individual network levels. According to the RailTopoModel, a network level is a data object expressed by an instance of the *LevelNetwork* class. It can be described by the *descriptionLevel* attribute expressing the respective level of detail [2, 3].

The Multipurpose Railway Infrastructure Model introduces the *dimension* and *representation* attributes, additionally. These attributes make it possible to distinguish between the description of the network using different spatial dimensions and whether the network is described only schematically or realistically. Of course, the way of visualization of individual data objects also depends on the attribute description of the respective network level which these data objects are assigned to.

When interested in creating the track geometry data description, it is therefore necessary to pay attention to the development of such software tools of the graphical editor that allow coherent transformation across the different description methods and levels. Providing these transformations is one of the other functionalities of the editing script. Since the track geometry data provide considerably detailed information, it is appropriate to express them at a detailed level corresponding to the value *micro* of the *descriptionLevel* attribute of the *LevelNetwork* class.

Graphical representation of the Multipurpose Railway Infrastructure Model data is suitable either in 2D or in 3D which can match the values *xy* and *xyz* of the *dimension* attribute. In order to enable the derivation of attribute values based on the spatial aspects of the visualized data objects, it is desirable to work with such network levels described by the value *realistic* of the *representation* attribute.

3. Linear elements and associated positions

In accordance with the RailTopoModel, the basic units of the topological network description are the so-called net elements which can be classified into non-linear and linear elements [2, 3]. According to the Multipurpose Railway Infrastructure Model, each linear element is also additionally described by its length in meters, expressed by the *length* attribute. The interest in describing linear elements by its length was originally promoted by the railML community in order to allow the position within a linear element to be expressed in a more practical way than by specifying the relative value of the *intrinsicCoord* attribute [7]. In the latest RailTopoModel version 1.4, the *length* attribute attribute was also added to the *NetElement* class [8].

Since the information of the geometrical track description has mainly linear features and because the linear elements make it possible to better express the information about the admissible routing within the network, the recommended way of expressing the network structure for the needs of describing track geometry is to use linear elements. In that case, the line elements represent individual track sections (routes) between nodes. After all, such an approach implies the micro description level of the network.

Each linear element is represented by an instance of the *LinearElement* class, within RailTopoModel version 1.4 renamed to *LinearNetElement*. Individual net elements can be connected by positioned relations which are the instances of the *PositionedRelation* class. Each positioned relation connects exactly two net elements and in the case of a linear element, it can be bound either at its beginning or at its end. Each positioned relation is also described by the *navigability* attribute, expressing whether it is passable and, if so, in which direction [2, 3, 8].

The Multipurpose Railway Infrastructure Model allows us to define any number of associated positions bound to a particular linear element. Each associated position, represented by an instance of the *AssociatedPosition* class, is described by its *intrinsicReference* and *deltaPosition* attributes.

The *intrinsicReference* attribute expresses the intrinsic coordinate within the relevant net element. In the basic concept, for a linear element, it can either take the value of 0 (at the beginning of the element) or the value of 1 (at the end of the element).

The *deltaPosition* attribute expresses the difference in position measured along the net element from the position expressed by the *intrinsicReference* attribute to the resulting position in meters. This attribute can also take the values of negative numbers. Therefore, the resulting position parameter p of each associated position can be calculated as

$$p \quad = \quad c \cdot l + \Delta p, \tag{1}$$

where c expresses the value of the *intrinsicReference* attribute, Δp expresses the value of the *deltaPosition* and l expresses the value of the *length* attribute of the linear element to which the respective associated position is bound.

Whereas net elements can appear in different dimensions, the values of the resulting positions calculated from the attributes of corresponding associated positions bound to the linear elements of different dimensions matching each other are generally slightly different. This fact can be demonstrated on the case of two network levels, which will be referred to as the *2D* network level and the *3D* network level according to the respective dimension.

Let the linear elements expressed at the *2D* network level are visualized as the perpendicular projection of the centre line of the real objects that they represent into the *xy* horizontal plane and the respective instance of the *LevelNetwork* class is described by the attribute values as follows:

- *descriptionLevel* ← *micro*,
- *dimension* ← *xy*,
- *representation* ← *realistic*.

Let the linear elements expressed at the *3D* network level are visualized basically accurately representing the centre line of the real object in three-dimensional space *xyz* and the respective instance of the *LevelNetwork* class is described by the attribute values as follows:

- *descriptionLevel* ← *micro*,
- *dimension* ← *xyz*,
- *representation* ← *realistic*.

When we compare the linear element expressed at the *2D* network level with the linear element expressed at the *3D* network level, they have different lengths (except for cases where they express a horizontal track section). This difference is also reflected in the calculation of the corresponding resulting positions on the linear elements and caused by the gradient profile, which is not taken into account when determining resulting positions at the *2D* network level.

The described ways of graphic visualization can also be reasonably applied to objects of other classes assigned to the stated network levels. The direct description of linear elements does not say anything about their shape. They can nevertheless be visualized based on the geometric entities that are localized to them. These are instances of the terminal classes of the *GeometryEntity* module, which is one of the specific Multipurpose Railway Infrastructure Model extensions used by many projects [5, 6]. The *GeometryEntity* module was designed using some aspects of the railML® 3.1 data format [9], which is a exchange format based on the RailTopoModel, so that portability can be ensured in the future. Nevertheless, the module has some specific features of its own. Its structure can be seen in the Figure 1.

4. Horizontal curves

When filling in the data description of the Multipurpose Railway Infrastructure Model using the graphical editor, it is advantageously feasible to construct the linear elements of the micro level first in 2D using horizontal curves. Horizontal curves are instances of all the terminal classes which are specializations of the *HorizontalCurve* class contained in the *GeometryEntity* module. Together with the *HorizontalCurve* abstract class, the module includes also the *VerticalCurves* and the *Superelevation* abstract classes, which all are specializations of the *GeometryEntity* abstract class, the top class of the *GeometryEntity* module, derived from the *NetEntity* abstract class.

Generally, net entities, as introduced by the RailTopoModel, are those objects representing the facilities and properties of the railway infrastructure. They can be localized to individual net elements. The RailTopoModel is so general that it does not define specific classes of net entities [2, 3, 8].

Although the railML® specifications do so, the railML® 3.1 data format implements the *HorizontalCurve* class as a common class for which instances are horizontal curves of all types [9]. Nevertheless, the Multipurpose Railway Infrastructure Model introduces a separate class for each curve type, allowing instances of these classes to be described with more specific attributes.

The currently used version of the *GeometryEntity* module includes the following terminal classes of horizontal curves:

- *StraightHC* – describing straight horizontal curves,
- *CircularArcHC* – describing horizontal curves of the shape of a circular arc,
- *CubicParabolaHC* – describing horizontal transition curves of the shape of a cubic parabola,
- *ClothoidHC* – describing horizontal transition curves of the shape of a clothoid.

Each of these specialized classes of horizontal curves has several attributes defined. The only attribute common to all Multipurpose Railway Infrastructure Model classes except association classes is *id* inherited from the *BaseObject* class. This attribute has the meaning of a unique identifier across all objects of these classes. Based on its value, records of all relational database tables belonging to one data object are joined. Attributes that are common to a significant number of named object classes representing the formalized and user-defined naming of the relevant objects are the *name* and *longname* attributes inherited from the *NamedResource* class.

The attributes *azimuth0* and *deltaAzimuth* are the attributes that are common to all specialized classes inherited from the *HorizontalCurve* class. The *azimuth0* attribute determines the azimuth at the starting point of the respective horizontal curve expressed in degrees and it can take values from 0 to 360. The *deltaAzimuth* attribute expresses the difference between the azimuth at the end and the azimuth at the beginning of the respective horizontal curve. It take the value of a positive number for right-turning curves and the value of a negative number for left-turning curves. For each instance of the *StraightHC* class it takes the value of 0.

In order to express the azimuth also at the horizontal curve end point, we can define the *azimuth1* parameter. For each horizontal curve, this parameter can be calculated as follows:

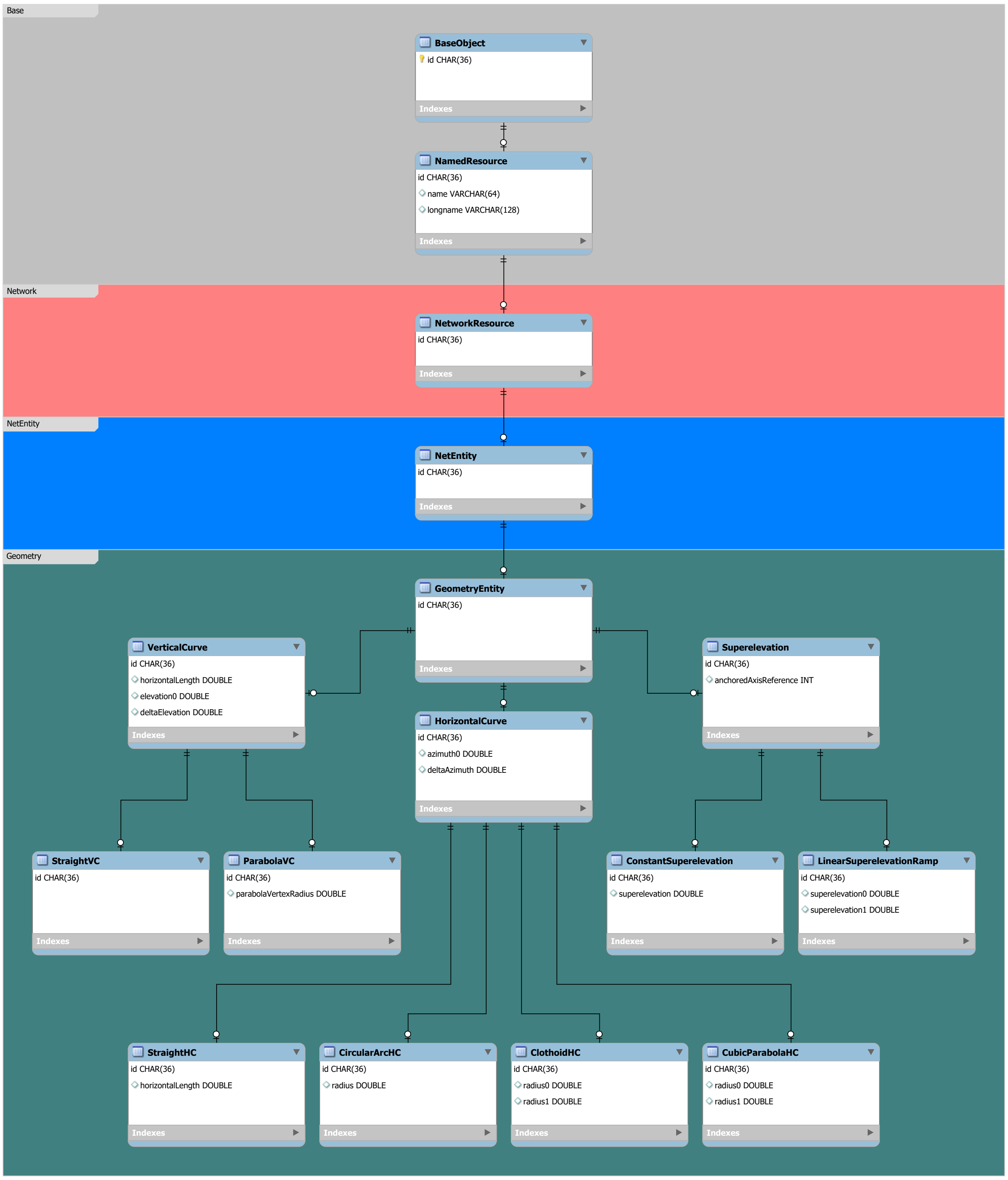

Figure 1. The *GeometryEntity* module as a specific extension of the Multipurpose Railway Infrastructure Model.

$$\alpha_0 + \Delta\alpha < 0 \quad \Rightarrow \quad \alpha_1 = \alpha_0 + \Delta\alpha + 360, \quad (2)$$

$$0 \leq \alpha_0 + \Delta\alpha < 360 \quad \Rightarrow \quad \alpha_1 = \alpha_0 + \Delta\alpha, \quad (3)$$

$$360 \leq \alpha_0 + \Delta\alpha \quad \Rightarrow \quad \alpha_1 = \alpha_0 + \Delta\alpha - 360, \quad (4)$$

where α_0 expresses the value of the *azimuth0* attribute, $\Delta\alpha$ expresses the value of the *deltaAzimuth* and α_1 expresses the value of the *azimuth1* parameter.

In terms of individual specialized classes of horizontal curves, the *StraightHC* class has the *horizontalLength* attribute that expresses the length of the line segment representing the straight horizontal curve in meters, in addition. It takes the value of a positive number. The *CircularArcHC* class has the *radius* attribute expressing the radius in meters, instead. It takes the value of a positive number for right-turning curves and the value of a negative number for left-turning curves.

For transition curves, which are instances of the *ClothoidHC* and *CubicParabolaHC* classes, however, two radius values must be expressed, both at the beginning and at the end of the respective curve. This is provided by the *radius0* and *radius1* attributes. One of these points is often a point with zero curvature. For such a point, the corresponding attribute takes the value of 0, although it does not express the radius.

When constructing horizontal curves in the xy plane (where $z = 0$) using the graphical editor tools, we can express the starting point of each horizontal curve using the x_0 and y_0 coordinates and its end point using the x_1 and y_1 coordinates as follows:

$$x_1 = x_0 + \Delta x, \quad (5)$$
$$y_1 = y_0 + \Delta y. \quad (6)$$

The Δx and Δy values can be calculated for each horizontal curve based on knowledge of the class of which it is an instance and the set of values of the following attributes: *azimuth0*, *deltaAzimuth* and the specific attributes of individual specialized classes of horizontal curves. Based on knowledge of the horizontal curve specialized class and the attribute values, the horizontal length of the respective curve can also be calculated.

5. Creating linear elements in 2D using horizontal curves

If we choose a specialized horizontal curve class and enter the x_0 and y_0 coordinates (for example by selecting a point of the xy plane) and the values of the relevant attributes in the graphical editor, we are able to plot a curve representing one instance of the given specialization of the *HorizontalCurve* class. After plotting the horizontal curve of specified attribute values in the base position (which can be starting from the beginning of the coordinate system at the azimuth of 90°, i. e. not yet taking into account the value of the *azimuth0* attribute), it is necessary to move it to the determined starting point with the x_0 and y_0 coordinates and rotate it so that the azimuth at the beginning of it matches the value of the *azimuth0* attribute. In that case, the x_1 and y_1 coordinates can also be calculated by means of the graphical editor.

If we declare the x_1 and y_1 coordinates of the current horizontal curve to be the x_0 and y_0 coordinates of the consecutive horizontal curve, it is then possible to construct the consecutive horizontal curve in the same way. Whereas the value of the *azimuth0* attribute of the newly constructed horizontal curve should be equal to the *azimuth1* parameter of the already constructed horizontal curve. This procedure can be used to plot the graphical representation of the entire linear element which these horizontal curves belong to. In that case, the mentioned steps must be repeated until all the horizontal curves belonging to the linear element has been constructed.

Although all the horizontal curves are considered net entities connected to a net element, the graphical representation of the linear element created using the graphical editor at the *2D* network level it is based on the spatial aspects of horizontal curves. While net entities of all other classes are localized to an already existing element along with creating their graphical representation, horizontal curves are supposed to be exceptionally inserted (not yet located using associated location) in advance of the linear element itself. The relevant 2D linear element is created by enclosing selected consecutive horizontal curves, then. Horizontal curves that can be enclosed into several linear elements can be seen in the Figure 2.

In order for a linear element to be enclosed, several conditions must be met. If the linear element is to be created from n horizontal curve the $(k+1)$th horizontal curve must start at the point where the kth curve ends and the azimuth at the beginning of the $(k+1)$th curve must be equal to the azimuth at the end of the kth curve, where can k can take the value of a natural number from 1 to n. When plotting subsequent horizontal curves in the above-mentioned manner, these conditions are already ensured.

The *length* attribute of the linear element is also calculated based on the particular horizontal curves used to enclose it. If we denote the horizontal length of the kth horizontal curve from the total number of n horizontal curves forming the linear element as s_k, we can express the linear element length l as follows:

$$l = \sum_{k=1}^{n} s_k. \quad (7)$$

These horizontal curves are retrospectively located to the linear element, which was created on their spatial basis, then. Each horizontal curve is to be located to the linear element using the line associated location that uses individual associated sections. Each associated section is defined by two associated positions

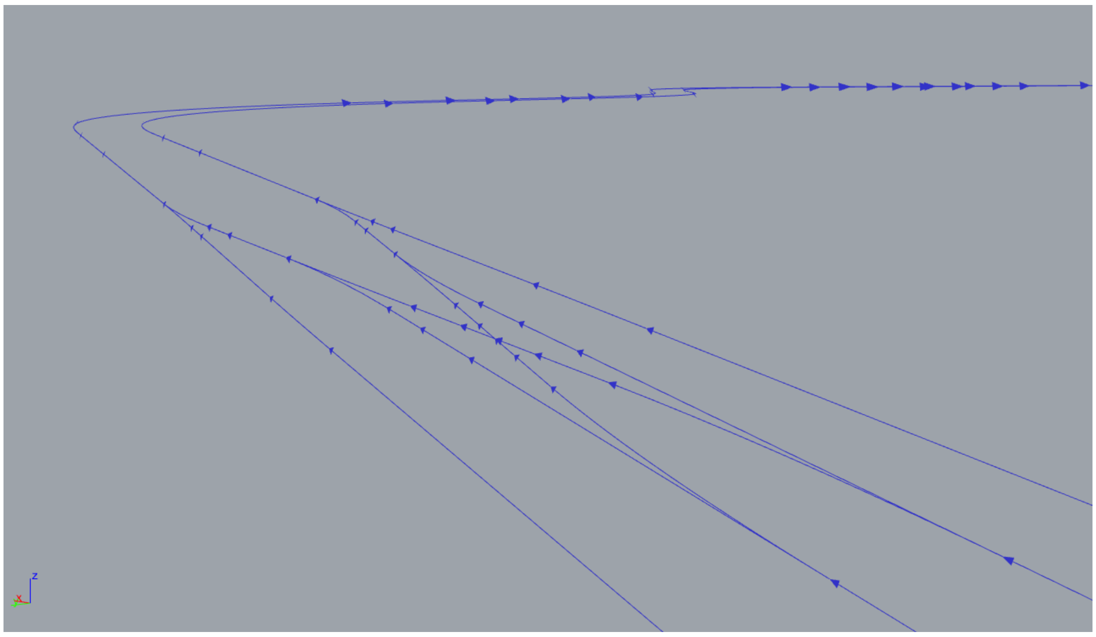

Figure 2. Graphically expressed horizontal curves of a station throat area to be enclosed into 2D linear net elements.

bound to the same linear element. In order to create associated sections required to create associated locations intended to locate horizontal curves, individual associated positions at the boundaries and interfaces of the individual horizontal curves must be created.

For the stated purpose, we need to define $n + 1$ associated positions. Since the associated positions are instances of the *AssociatedPosition* class, it is necessary to set the values of their attributes *intrinsicReference* and *deltaPosition* as well. Although this can be done in various ways, the following one can be recommended:

$$c_1 = 0, \tag{8}$$
$$\Delta p_1 = 0, \tag{9}$$
$$c_j = 0, \tag{10}$$
$$\Delta p_j = \sum_{i=1}^{j-1} s_i, \tag{11}$$
$$c_{n+1} = 1, \tag{12}$$
$$\Delta p_{n+1} = 0, \tag{13}$$

where j takes the value of a natural number from 2 to n, c_k expresses the value of the *intrinsicReference* attribute of the kth created instance of the *AssociatedPosition* class and Δp_k expresses the value of the *deltaPosition* attribute of the kth created instance of the *AssociatedPosition* class.

Within the Multipurpose Railway Infrastructure Model, each associated position can be linked to a geo-point. Each geo-point can have its coordinates assigned within each defined coordinate system. In the case of geometric or geographical coordinates, this is carried out using the *GeoPointGeoCoordinate* association class, while the respective coordinates are expressed by the x, y and z attributes.

Taking into account the coordinate system of the graphic editor, we can fill the x and y attributes related to individual geo-points connected to the associated positions created in order to locate horizontal curves with the x_0 and y_0 (eventually x_1 and y_1) values of the respective horizontal curves in meters. Since we are describing a linear element in 2D, we are supposed to set the value of the each z attribute to 0.

6. Vertical Curves

After a linear element has been successfully created, other net entities can be located to it. This usually brings with it the creation of new associated positions and associated sections used to define their associated locations. In some cases, existing associated features can also be used. We can also calculate the coordinate values with which the attributes of the *GeoPointGeoCoordinate* class instances are to be filled, when creating new geo-points based on the new associated positions. The means of the graphical editor can also be used to fulfill this task.

In terms of track geometry, the *GeometryEntity* extension module of the Multipurpose Railway Infrastructure Model allows us to describe selected types of vertical curves and separable superelevation sections.

The currently used version of the *GeometryEntity* module includes the following terminal classes of vertical curves:

- *StraightVC* – describing straight vertical curves, i. e. sections of constant slope,
- *ParabolaHC* – describing vertical curves in the shape of a parabola.

Both of these specialized classes of vertical curves has several attributes defined. The common ones of them are the *id*, *name* and *longname* attributes again and in addition the attributes of the *VerticalCurve* class, which are the *elevation0*, *deltaElevation* and *horizontalLength* attributes. The *elevation0* attribute determines the elevation at the starting point of the respective vertical curve expressed in meters above the reference level (e. g. above sea level) and it can take the value of a real number. The *deltaElevation* attribute expresses the difference between the elevation at the end and the elevation at the beginning of the respective vertical curve.

In order to express the elevation also at the vertical curve end point, we can define the *elevation1* parameter. For each vertical curve, this parameter is calculated as follows:

$$z_1 = z_0 + \Delta z, \tag{14}$$

where z_0 expresses the value of the *elevation0* attribute, Δz expresses the value of the *deltaElevation* attribute and z_1 expresses the value of the *elevation1* parameter.

The *horizontalLength* attribute determines the length of the perpendicular projection of the respective vertical curve to the horizontal xy plane in meters. It must have the same value as the difference between resulting positions of the associated positions defining the associated section intended to locate the respective vertical curve to the corresponding 2D linear element.

In terms of individual specialized classes of vertical curves, the *StraightVC* class has no additional attributes, while the *ParabolaHC* class has the *parabolaVertexRadius* additional attribute expressing the radius of the respective parabola in meters defined. It takes the value of a positive number for sag roundings and the value of a negative number for crest roundings.

7. Separable superelevation sections

The currently used version of the *GeometryEntity* module includes the following terminal classes of separable superelevation sections:

- *ConstantSuperelevation* – describing sections of constant superelevation,
- *LinearSuperelevationRamp* – describing linear superelevation ramps.

Both of these specialized classes of separable superelevation sections has several attributes defined. The common ones of them are the *id*, *name* and *longname* attributes again and the attribute of the *Superelevation* class, which is the *anchoredAxisReference* attribute, in addition. The *anchoredAxisReference* attribute determines which axis remains at its original height even after the elevation is constructed. If it is the track axis, the attribute takes the value of 0. If it is the left rail axis, the attribute takes the value of −1. If it is the right rail axis, the attribute takes the value of 1.

In terms of individual specialized classes of separable superelevation sections, the *ConstantSuperelevation* class has the *superelevation* attribute expressing the height difference between the track rails in millimeters, in addition. It takes the value of a positive number if the left rail is higher, the value of a negative number if the right rail is higher a the value of 0 if both rails are at the same height level.

For superelevation ramps, which are instances of the *LinearSuperelevationRamp* class, however, two superelevation values must be expressed, both at the beginning and at the end of the respective separable section. This is provided by the *superelevation0* and *superelevation1* attributes.

8. Transformation of 2D linear elements to 3D

The 2D linear element that is coherently, completely and unambiguously described by vertical curves can be transformed into the corresponding 3D linear element in the graphical editor. This involves creating a new graphical representation of the newly emerging linear element. The default assumption for doing this is the ability to express the x and y coordinates and corresponding resulting position p at each point of the 2D linear element graphical representation (we can imagine an instance of the *AssociatedPosition* class to be created in each of these points). Since the element is already plotted in the graphical editor which has the tools to obtain these values, this assumption can be considered fulfilled.

In order to plot the corresponding 3D linear element, it is necessary to calculate the z coordinate belonging to the individual points of the 2D linear element. It is advisable to proceed according to the associated sections to which the individual vertical curves are located. The determination of the z coordinate is carried out depending on the class of the respective vertical curve and the resulting position at the relevant point. It is based on calculations related to the design of rail transport structures [10, 11].

For straight vertical curves, the z coordinate of each point within the associated location of the respective vertical curve can be calculated based on its resulting position within the source 2D linear element simply by interpolating the stated values of the *elevation0* attribute and the *elevation1* parameter of the respective vertical curve in the following manner:

$$z_p = z_0 + (p_p - p_0) \cdot \frac{z_1 - z_0}{p_1 - p_0}, \tag{15}$$

where p_p expresses the resulting position of the given point, p_0 represents the resulting position at the beginning of the vertical curve, p_1 expresses the resulting position at the end of the vertical curve, z_0 expresses the value of the *elevation0* attribute, z_1

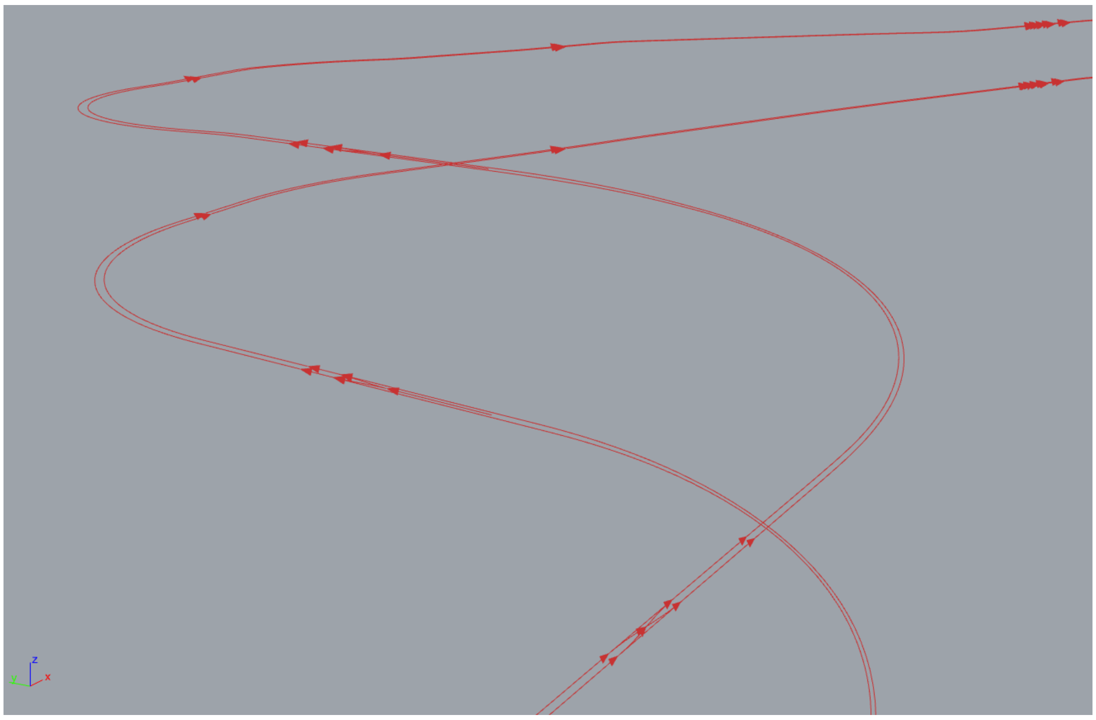

Figure 3. Graphically expressed linear elements representing line and station tracks in 2D and in 3D.

expresses the value of the *elevation1* parameter and z_p expresses the z coordinate of the given point.

For parabolic vertical curves, the calculation of the z coordinate of each point within the associated location of the respective vertical curve requires knowledge of the *parabolaVertexRadius* attribute value, below referred to as r_v, in addition. First of all, it is advisable to calculate the values of resulting position p_v and the elevation z_v at the top of the respective parabola:

$$p_v = p_1 - \frac{(p_1 - p_0)^2 + 2 \cdot (z_1 - z_0) \cdot r_v}{2 \cdot (p_1 - p_0)^2}, \quad (16)$$

$$z_v = z_0 - \frac{(p_v - p_0)^2}{2 \cdot r_v}. \quad (17)$$

With the use of these auxiliary values, the final calculation can already be performed:

$$z_p = z_v + \frac{(p_p - p_v)^2}{2 \cdot r_v}. \quad (18)$$

Once this procedure is done for all points of all vertical curves located to the 2D linear element, it is possible to plot the corresponding 3D linear element. How the result of the transformation looks for several linear elements can be seen in the Figure 3.

The transformation may also include creating 3D associated points bound to the 3D linear element based on 2D points bound to the corresponding 2D element. As individual associated positions may also correspond to individual geo-points with assigned geo-coordinates of defined geo-positioning system, this transformation may further include the creation of matching geo-points in 3D. If a suitable geo-coordinate system is used, this basically means to preserve the values of the x and y attributes and to change the z attribute value of the original 2D geo-point from 0 to the calculated value when assigning new geo-coordinates to the corresponding geo-points in 3D.

9. Visualization

The presented method can also be used to visualize the infrastructure data related to track geometry directly based on records in the database of the Multipurpose Railway Infrastructure Model. This procedure is nevertheless more complex, as it also includes the construction of the curve representing the 2D linear element. Indeed, this can be achieved using tools similar to those used to create the horizontal curves in the graphical editor environment.

The construction of the 2D linear element curve assumes that the linear element is coherently, completely and unambiguously described by horizontal curves. In case this data was obtained from the graphical editor where the 2D linear element was created by enclosing of inserted horizontal curves, this condition is already fulfilled. It only must be ensured that the resulting data description was not subsequently improperly tampered with.

At the beginning of the 2D curve construction, it is essential to determine the coordinates of the starting point. For this purpose, it is appropriate to find the geo-point which is assigned to the associated position of the values of its attributes *intrinsicCoordinate* $\leftarrow 0$ and *deltaPosition* $\leftarrow 0$ which is bound to the relevant 2D linear element. Its coordinates expressed in

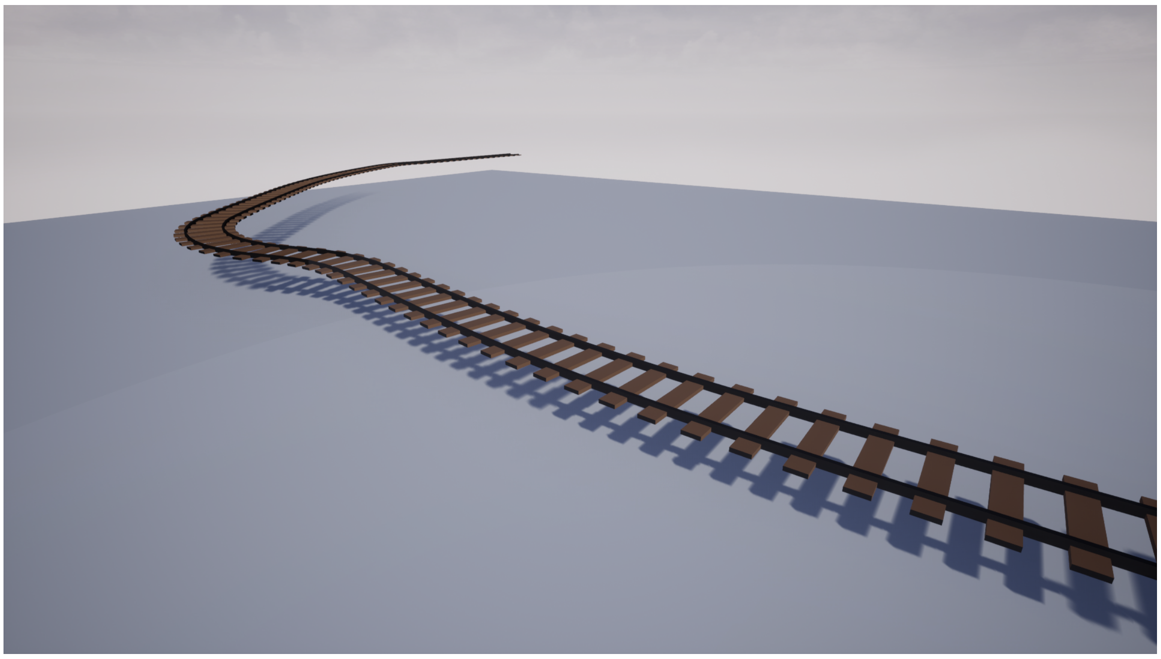

FIGURE 4. Visualization of the Multipurpose Railway Infrastructure Model data in software created by Martin Němec.

the appropriate geo-coordinate system determine the starting point.

The first horizontal curve, which associated location uses the stated associated position, is supposed to be plotted from the starting point. The subsequent procedure is similar to the one carried out when working in the graphical editor, only there is no need to enter individual attribute values, as these are loaded from the database. An example of visualization of a linear element representing a track based on data from the Multipurpose Railway Infrastructure Model database is shown in the Figure 4.

Net entities of other classes can also be used for visualization purposes. E. g., in terms of track geometry, these are the above-mentioned separable superelevation sections.

10. CONCLUSIONS

This paper introduced the possibilities of creating a realistic graphic representation of linear elements representing tracks in 2D and 3D. Related procedures and mathematical operations use data compatible with the Multipurpose Railway Infrastructure Model structure and the *GeometryEntity* module, which is its extension module focused on the track geometry description. Their implementation is supposed to be carried out in the graphical editor, which also serves to fill the database of the Multipurpose Railway Infrastructure Model with consistent data. The way of data description of horizontal curves, vertical curves and separable superelevation sections using the classes of this the *GeometryEntity* module was continuously presented.

The 2D linear element formation is based on the insertion of individual horizontal curves intended to be enclosed into it. The 3D linear element creation presupposes the transformation of the 2D linear element into 3D. This is based on the calculation of the z coordinates of the individual points of 2D linear element graphical representation. The method of the z coordinate calculation within a given associated section used to locate any of the vertical curves of the linear element varies based on the the class of the respective vertical curve. The resulting data description stored in the Multipurpose Railway Infrastructure Model database can be graphically presented with the use of dedicated visualization tools.

Further development of the described *GeometryEntity* module could include the introduction of other types of transitions curves and superelevation ramps, as well as the possibilities of expressing different values of track gauge and track gauge widening.

REFERENCES

[1] A. Hlubuček. Význam popisu infrastruktury pro inteligentní dopravní systémy na železnici. *Vědeckotechnický sborník ČD* **42**, 2016. [2020-08-31], `https://docplayer.cz/108533917-Vyznam-popisu-infrastruktury-pro-inteligentni-dopravni-systemy-na-zeleznici.html`.

[2] UIC. RailTopoModel v1.0. 2016, [2022-04-22], `http://www.railtopomodel.org/en/download/irs30100-apr16-7594BCA1524E14224D0.html?file=files/download/RailTopoModel/180416_uic_irs30100.pdf`.

[3] UIC. RailTopoModel v1.1. 2017, [2022-04-22], `https://www.railtopomodel.org/en/download/irs30100-apr16-7594BCA1524E14224D0.html?file=files/download/RailTopoModel/061117_uic_railtopomodel_v1-1.zip`.

[4] M. Leso, D. Kamenický, P. Koutecký, A. Hlubuček. Dopravní sál FD – železniční laboratoř pro výuku i výzkum. *VTS Správy železnic* **1**:76–85, 2019. [2020-08-31], `https://www.spravazeleznic.cz/documents/50004227/87152001/V%C4%9Bdeckotechnick%C3%BD+sborn%C3%ADk+Spr%C3%A1vy+%C5%BEeleznic+%C4%8D.1-2019/5743ca4b-17ca-4e95-8f51-cc179180fa64?version=1.0`.

[5] A. Hlubuček. Possibilities of high-speed railway turnout data description. *Acta Polytechnica CTU Proceedings* **31**:18–26, 2021. `https://doi.org/10.14311/APP.2021.31.0018`

[6] A. Hlubuček. Digital track map for the VEXA expert system. *Acta Polytechnica CTU Proceedings* **35**:8–13, 2022. `https://doi.org/10.14311/APP.2022.35.0008`

[7] V. P. Kolmorgen, C. Rahmig. railML 3.1beta Dissemination and Feedback Workshop. Berlin, 2018.

[8] UIC. RailTopoModel v1.4. 2022, [2022-04-22], `https://www.railtopomodel.org/en/download/irs30100-apr16-7594BCA1524E14224D0.html?file=files/download/RailTopoModel/2022-04-29_railml_railtopomodel_v1-4.zip`.

[9] railML. railML schema version 3.1, 2019.

[10] M. Lindahl. Track geometry for high-speed railways. Tech. rep., Royal Institute of Technology, Stockholm, 2001. [2022-04-22], `http://www.europakorridoren.se/spargeometri.pdf`.

[11] J. Procházka. Sylabus 10. přednášky z Inženýrské geodézie (Přechodnice, přechodnicové a výškové oblouky). Inženýrská geodézie, 2015, [2022-04-22], `https://k154.fsv.cvut.cz/wp-content/uploads/2022/01/Sylabus_IG_10.pdf`.