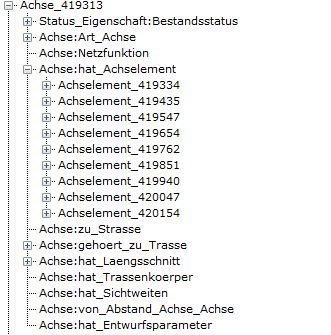
# OKSTRA

# Alignment

OKTRA defines alignments as *Achse* objects which store the horizontal alignment elements as well as the vertical alignments.



Vertical alignment

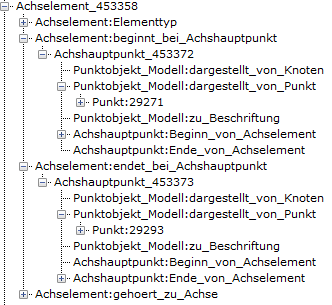
Horizontal alignment elements

# Horizontal Alignment

Reading the horizontal alignment is done by iterating through the alignment elements (in OKSTRA “Achselement”) and generating the horizontal alignment elements.

For every element:

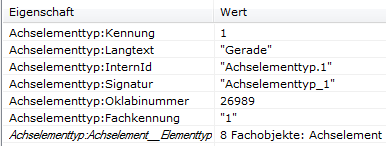
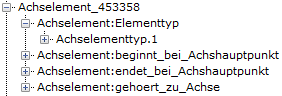
1. **Read start and end position:**



End position

Start position

1. **Determine element type:**



Type

Possible types:

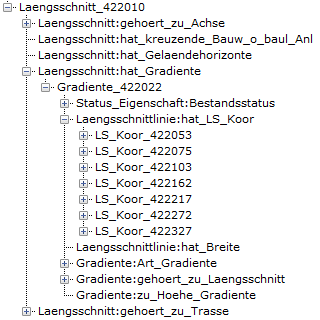
* *Gerade* -> line
* *Kreisbogen, tangential* -> arc
* *Klothoide* -> clothoid

1. **Obtain further parameters from Okstra:**
   * Arc:
     1. radius = „achselement“->“Radius\_zu\_Beginn”
     2. start direction = „achselement“->“Richtung“
   * Clothoid:
     1. radius start = “achselement”->”Radius\_zu\_Beginn“
     2. radius end = „achselement“->“Radius\_am\_Ende“
     3. length = „achselement“->“Laenge“
     4. clothoid constant = “achselement”->”Parameter”
     5. start direction = “achselement”->”Richtung” Vertical Alignment
2. **Generate alignment elements:**
   * Line (start, end):
     1. Start and end position already given
   * Arc (center, start, end, clockwise):
     1. Rotate start direction by 90° to get a vector that points from the start position to the center position
     2. Scaling this vector and adding it to the start position result in the center position
     3. The arc runs clockwise if the radius is greater than 0
   * Clothoid(start, end, pi, length, start radius, end radius, clockwise)
     1. Calculate vectors that point from start/end position to the pi position with the help of start/end direction
     2. Calculate intersection of these vectors to get pi position
     3. The clothoid runs clockwise if start and end radius are greater than 0

# Vertical Alignment

## Reading the OKSTRA 1.014 File

The vertical alignment is represented by a List of PVIs stored in a *Gradiente* object which is held by a *Laengsschnitt* object:



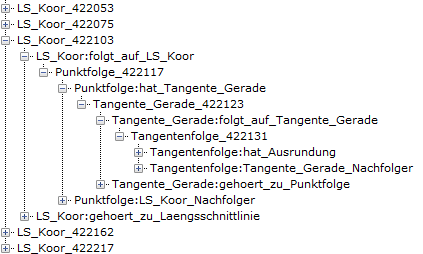
The positions of the PVIs can be obtained by inspecting the LS\_Koors:



However, no information about the type and additional parameters of the PVI is given.

Start and end gradient can be calculated by looking at the preceding and following PVIs.

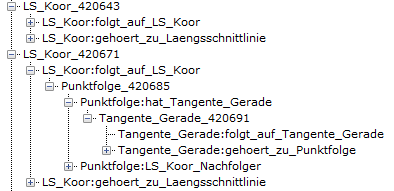
The parabola parameter can be found by inspecting the next PVI:



The type of the PVI is determined by the following rules:

* **Start (Starting point of the vertical alignment):**
  + The PVI has no predecessor

No predecessor

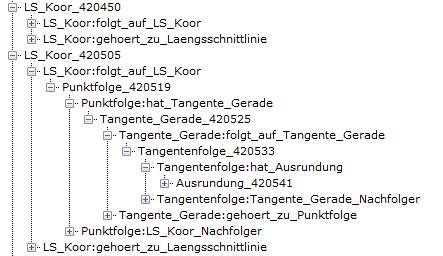
* + 
* **Line (Endpoint of a line segment):**
  + The PVI has a predecessor, but has no parabola parameter
  + 

No parabola parameter

Predecessor

* **Parabola (Intersection point of two tangents):**
  + The PVI has a predecessor and has a parabola parameter

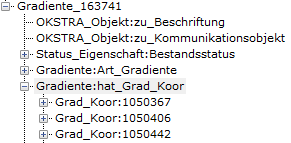
Predecessor

* + 

Parabola parameter

## Reading the OKSTRA 2.016 File

Compared to Okstra 1.014, getting the parabola parameter in Oktsra 2.016 is a lot easier. The L*aengsschnitt* object has been removed and the *LS\_Koor*s have been replaced by *Grad\_Koors*:



This time everything we need is stored in *Grad\_Koor*:



Parabola parameter



The parabola can be found in the *Ausrundung* object and the pvi-type determined by the following rules:

* Start:
  + First pvi
* Line:
  + *Grad\_Koor* doesn’t have a *Ausrundung* object
* Parabola
  + *Grad\_Koor* does have a *Ausrundung* object

### Resulting PVI type definition:

enum class OKSTRA\_PVI\_Type

{

Start,

Line,

Parabola

};

struct OKSTRA\_PVI

{

OKSTRA\_PVI\_Type pvi\_type;

double Station;

double Höhe;

double Ausrundung;

Only used if pvi\_type is Parabola

double Anfangssteigung;

double Endsteigung;

};

## Generating Alignment Elements

Given a vector of PVIs, we have to generate alignment elements (segments).

This is achieved by iterating through the OKSTRA\_PVIs:

* Start:
  + Temporarily save position
* Parabola:
  + Calculate start and end position of the parabola
  + Generate line element from the last position to the start position
  + Generate parabola element between start and end position
  + Temporarily save end position
* Line:
  + Generate line element from the last position to the current position
  + Temporarily save position

Parabola

Start

Line

Parabola

Line