CES321918

Generating, Transforming, and Analyzing Railway Design Data in Civil 3D and Dynamo

Wouter Bulens
TUC RAIL

Learning Objectives

- Learn how to design dynamic blocks that are digital representations of local standards and better fit your design process.
- Learn how to organize Dynamo nodes to connect different design data in AutoCAD and Civil 3D.
- Learn how to analyze corridor data and other design objects more directly and iteratively.
- Learn about the need for design objects with the correct data and interactivity configuration.

Description

Railway projects have a lot of regulations, diverse technical fields, and project members coming from many different backgrounds. These factors and many more all come together during the design process and need to be managed to make the project a success. This class will show how connecting and using design data in Civil 3D software, AutoCAD software, and Dynamo can help us meet this challenge. Attendees will see examples of how to make regulations interactive and more easily accessible, and how to transform design data depending on the required technical field or output. We will also demonstrate tools to assist the designers so they can better analyze their creations, and in doing so, motivate them to start interacting and designing in a new way.

Speaker

Wouter Bulens has been an active user of Autodesk AEC software solutions for 19 years, he holds a degree as a professional drafter and a master's degree in industrial science – construction. In his professional career he has been active on numerous civil engineering projects in Belgium, France, the Netherlands and some international tenders, with a role of road- and rail designer. He currently works as BIM manager for the Belgian rail engineering firm TUC RAIL. He has focused his career on 3D modeling for multidisciplinary civil projects and process optimization for diverse fields: drafting, render visualization, 4D animations, real-time applications, project system engineering and design platform integration. Besides his design experience, he is also a .NET developer (API), having made tools for 3DS Max, AutoCAD and Civil 3D. He is an active participant of the Civil 3D Rail development group, helping to push not just rail design but transport and infrastructure design forward.

Table of contents

Introduction

After 11 years of being active in rail and civil design, I have accepted the general rules that are intrinsic to the work we do every day:

- No civil project is 100 % alike, there is always something different;
- Many different viewpoints, different languages, different understanding of the subject;
- Aligning, translating and explaining takes time;
- Our industry is stable and reliable, but tentative regarding change.

To manage these rules during a project we need:

- Tools that have a degree of flexibility built-in;
- To talk about the objects and their purpose;
- To automate the connections between objects and people;
- Solutions with the right interactivity for the user.

This class will demonstrate how we can combine AutoCAD and Civil 3D with Dynamo, using the principles of design objects, data and analysis to meet these challenges. Each principle will be demonstrated in practical workflows that hopefully teach you how to induce workflows that suit your project.

Overview

This document is composed of four learning objectives: three practically approached objectives that contain examples, and one principle which is explained throughout the entire document. Our understanding is being gradually built-up as we tackle each objective. The goal is not to just be able to reproduce the workflows described in this document, but to create your own solutions.

Design using Objects/Data/Analysis

Learn about the need for design objects with the correct data and interactivity configuration:

In this first part, the need for objects, data and analysis are illustrated with four examples derived from everyday situations on a project. The definition of our solutions is explained in each following section/learning objective in combination with specific examples.

Design Object

Learn how to design dynamic blocks that are digital representations of local standards and better fit your design process:

The definition of a design object is explained so that its meaning is clear for the rest of the document. To better understand the definition, two examples are described: a **platform edge** and a **switch** (railway turnout).

The creation of the switch design object is explained in detail. The final dynamic block is deconstructed into individual examples to zoom in to each individual function.

Transforming Design Data

Learn how to organize Dynamo nodes to connect different design data in AutoCAD and Civil 3D:

The definition of design data is explained with a simple AutoCAD circle object. I also discuss the difference between an object and data, to clarify as to why one or the other is needed in certain situations.

Secondly, Autodesk Dynamo for Civil 3D is introduced as a tool to manage and interact with all design objects and data present in AutoCAD and Civil 3D. Dynamo also offers the possibility to connect with data sources outside of the AutoCAD environment, however I don't cover that in this class handout.

Finally, three Dynamo scripts and their custom nodes will illustrate the transformation and use of data between design objects:

- Alignment - Profile - Cant <> Switch Dynamic Block;
- Switch Dynamic Block <> Profile – ProfileView;
- Switch Dynamic Block <> Corridor.

Design Analysis

Learn how to analyze corridor data more directly and iteratively:

I approach this final section describing what design analysis should look like and give two day to day examples that need to change. I also describe the necessity of this analysis in more detail.

Two analysis methods are enriched with the possibilities that Dynamo has to offer:

- AutoCAD Data Extraction;
- Civil 3D Corridor.

For each analysis a Dynamo script and custom nodes are explained in detail.

Software Requirements

For the exercises in this class, we will use the following software:

- Civil 3D 2020;
- Dynamo;
- AutoCAD 2020.

All custom nodes were created using Visual Studio 2019.

Design using Objects/Data/Analysis

The easiest way to explain the need for objects, data and analysis in design that I can think of, is to describe four situations or challenges I see on a daily basis when I look around the office. I can only hope that by the end of this document you will conclude, as did I, that objects, data and analysis are the solutions to these situations.

Medium ≠ Design

I am amazed every time I pass through the corridors of our office or participate in project meetings. It is incredible how often the subject on people's lips is not the object they are designing, rather the paper plan or document that they have made to represent the object or just a part of it.

There are entire schedules, standardizations and workgroups dedicated to following, defining and evaluating the creation of these paper information media.

FIGURE 1: MEDIUM ≠ DESIGN

It is not that paper is a bad medium to communicate between project participants. The problem is, at a certain point, we are only looking at the piece of paper and not the design. In this digital age it is also true that construction is using digital design data, so paper is not always used to build (example: LandXML files for a railway-tamping machine).

Tool ≠ Design

FIGURE 2: TOOL ≠ DESIGN

Just because we have been building something according to a standard, procedure or tool for a long time, does not mean that it will never change. The tool needs to be able to evolve independently of the project.

At one time hanging a frame required a hand drill, a screwdriver, manual labor and time. Today both tools are combined in an electric drill and the job can be done rapidly with minimum effort.

We should not reinvent the wheel, but we should also not stop checking whether the current tool is right for the job or relevant for a specific situation. After all, it is not guaranteed that the tool covers all your current design needs (maybe we should glue the frame).

Individual disciplines ≠ Design

Under the banner of efficiency, it is tempting to solve a design puzzle by separating it into its composing disciplines.

The individual parts of the puzzle can then be solved and at the end put together. With this method it is easy to optimize your discipline design, but not the overall design. It promotes the use of standard methods and does not question if they are right for the job.

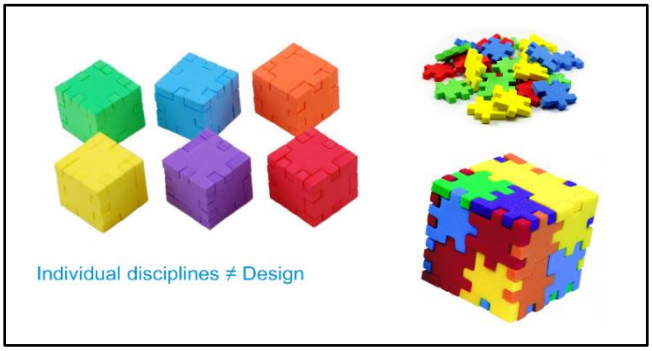


FIGURE 3: INDIVIDUAL DISCIPLINES ≠ DESIGN

To solve the puzzle correctly, to get the best outcome for the project, we need to connect disciplines that would otherwise be separated. This could lead to a solution that is completely different from the standard.

We only trust the ruler

FIGURE 4: WE ONLY TRUST THE RULER

"A CAD designer has used Civil 3D to design a new alignment and created a corridor that connects to the existing terrain. To validate the design, the engineer asks for a plan with sections of the corridor. The engineer places the plan on a table and takes out a scaling ruler to measure whether the design is acceptable."

Is this a familiar situation? It is still a far too common phenomenon in the workplace. We are already using parametric intelligent objects like alignments and corridors. However, when these need to be evaluated we revert to more classic tools. To build more trust with these objects we need other design analysis methods.

Recap

To achieve the goals of being medium independent, tool flexible, to connect individual disciplines and evaluate our design in new ways, we need to manage our design in a different way.

The concept of using objects, data and analysis in design is already widely spread in the IT industry, where almost every object is fictional digital code. In the civil industry where we build objects in the physical world, it is not yet common practice. There is work yet to be done.

But what are design objects, data and analysis? Furthermore, how do you actually use them in design today? In the next sections, each subject is explained and illustrated with examples.

Design Object

What is a design object in the context of this document?

In this handout we are not referring to the “Object Type Library” or “Object Breakdown Structure” as known in system engineering. No, we are referring to the object with which the drafter, engineer or designer interacts during his design work. This object can contribute to system engineering and it can represent a true object in the field, but what interests us even more, is the interaction during the design.

An example:

When working in AutoCAD a drafter might place simple circles on a floorplan. It is true that these circles might represent structural posts or pipes or a million other things. However, our drafter is interacting with an AutoCAD circle and all the data this object can contain. This circle is the design object we are talking about. Its relation to what it represents is important but so is the design interaction between the drafter and the circle.

How does a design object work?

First, we need to think about what part or sort of design the object needs to support. Where does it fit in the general process or lifecycle?

Secondly, we need to know that every object is different, but they should follow the input-process-output (IPO) model:

- Input:
  - Design decision;
  - Interaction with another design object;
  - Result from a process.
- Process:
  - Formula;
  - Decision tree/algorithm.
- Output:
  - Graphical/non-graphical;
  - Number/text/yes-no/choice.

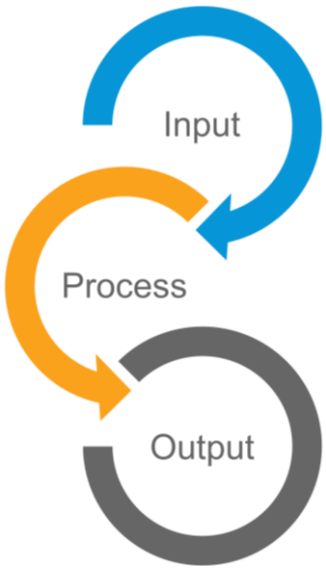


FIGURE 5: IPO MODEL

Applied to the simple AutoCAD circle:

The drafter needs to place structural posts on a floorplan. There are different types and dimensions of posts, so the design object that is used needs to be able to remember and display the difference. The defined posts are then used by someone else to create a list of location, circumference, type, diameter.

- Input:
  - Location X,Y on the floorplan;
  - Type (layer);
  - Dimension.
- Process:
  - Calculate the circumference; (π *d = 2π *r)
  - Circle geometry.
- Output:
  - A graphical circle on the floorplan;
  - Circle object with all the information.

The question whether or not this design object (AutoCAD circle) is optimally suited for the job, is open for debate and improvement. It is however clear that this design object gets the job done.

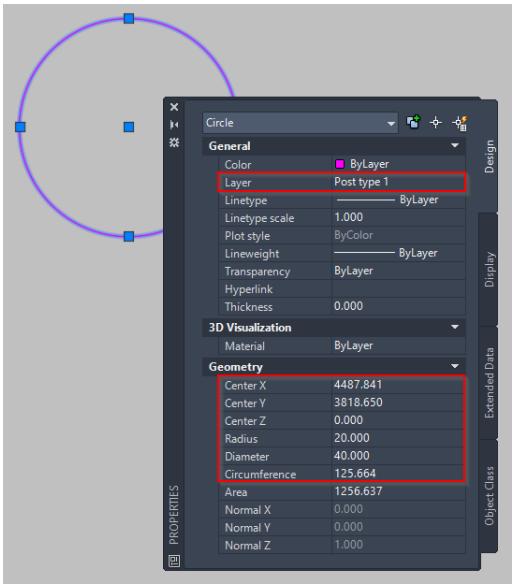


FIGURE 6: CIRCLE DESIGN OBJECT

Platform Edge

Before creating our own design object, let us take a closer look at another example that was added in 2019, the platform edge. Before this object was added in Civil 3D, the drafter would use an Excel template for the calculation and AutoCAD/Civil 3D to construct a 3D polyline. While its correct design is very important for the safe passage of the trains and the on- and off boarding of passengers, the calculation of the edge was considered a tedious job. For this reason, the amount of iterations was limited as much as possible, potentially limiting creativity.

To design a platform edge, the drafter must combine the national standard that describes the algorithm to calculate the points of the edge, with a region of the alignment/profile of his design or the existing situation. After this, he must validate the design with all other rail disciplines and adapt it when necessary. After validation, he must create a coordinate list to construct the platform onsite.

- Input:
  - Alignment/profile – station range;
  - Track gauge;
  - Calculation interval;
  - Side;
  - Rail type;
  - Platform type (national standard).
- Process:
  - Calculate the exact track center;
  - Calculate the offset according (national standard);
  - Combine track center and offset to determine platform edge.

- Output:
  - Linked Feature Line;
  - Platform edge data (.csv-file).

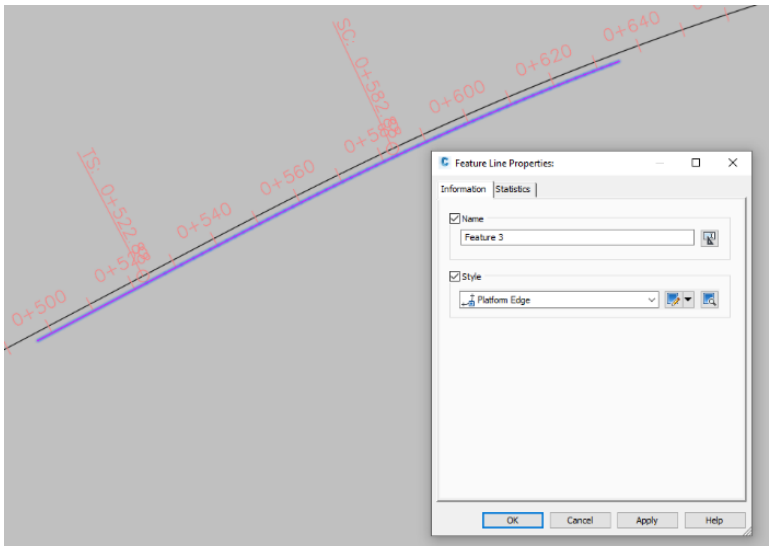


FIGURE 8: LINKED FEATURE LINE

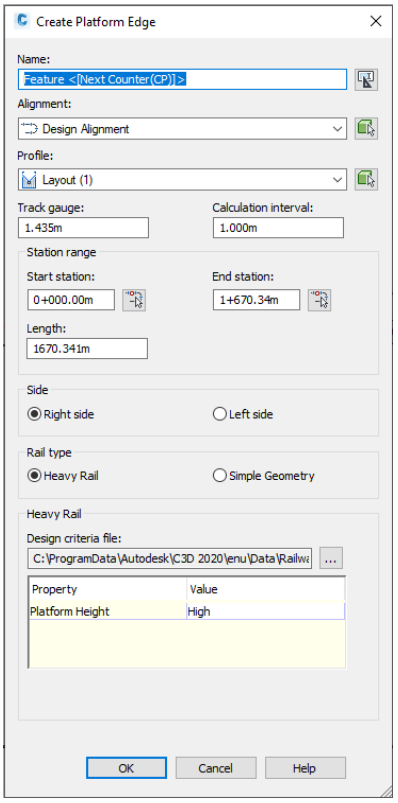


FIGURE 7: CREATE PLATFORM EDGE INTERFACE

In essence, the way we design a platform edge has not changed: the national standard is the same, the alignment is the same and the resulting coordinates are the same. Only now we are using a different design object (this is a great example of “Tool ≠ Design”) which improves parts of the work and frees up time to concentrate on other aspects of platform design (capacity, access, maintainability…).

An additional advantage is that this object retains all knowledge of the edge calculation. Because of this, it remains dynamically linked to the alignment, so it can change with the design and can be altered at any time.

Switch/turnout

The design of a railway switch is very similar to a platform edge. The rail designer chooses a correct switch type from a manufacturer and places it on the alignment. After adapting and validating the design, he creates plans and an order form for construction.

- Input:
  - Manufacturing configuration (ID, order number);
  - 3D placement (alignment/profile/cant).
- Process:
  - Combine design decisions;
  - Calculate graphical and non-graphical data.
- Output:
  - Geometry for drawing production/design validation;
  - Material order information;
  - Coordinates for on-site construction.

A suitable design object is currently under development at Autodesk. So here's the question: how can we improve this design process today? Well … It so happens that AutoCAD has a powerful framework to build your own design objects: the (dynamic) block.

When TUC RAIL made the transition to AutoCAD, we started looking for a suitable design object to better support switch design. Like many other disciplines at the time, the track department constructed a large library of blocks to support all the different tasks (mainly paper plan production).

"A block is essentially a block definition that includes the block name, the block geometry, the location of the base point to be used for aligning the block when you insert it, and any associated attribute data."

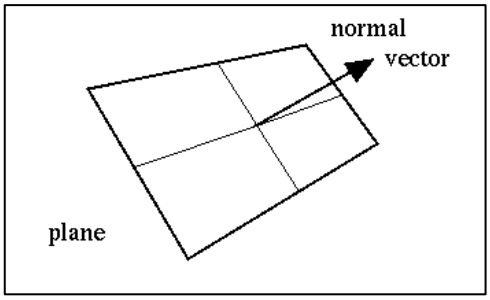


FIGURE 9: BLOCK ALIGNMENT

The fact that a block uses one insertion point and a direction to be placed in 3D, makes it suitable for the placement of most switches but also many other railway assets. The name identifier, contained geometry and associated attributes make it perfect to create a library or catalogue of railway assets.

The biggest limitation of these blocks is the fact that you need to alternate through many of them during your design process to work correctly and efficiently. Ultimately, one drafter's design task transformed into many different design objects, being far from ideal. It was for this and other reasons that we started using dynamic blocks.

"Dynamic blocks contain rules and restrictions that control the appearance and behavior of a block when it is inserted into a drawing or when it's later modified."

Dynamic blocks brought a new dimension to our library, we were now able to combine blocks and make design decisions interactive. We started combining not just graphical presentations but also switch variations of the same base type. By doing this, the user was now able to change the switch type by using grips and controls and not by selecting a new block.

For all of its possibilities there is one important restriction to take into account. The grips and controls of a dynamic block are limited to 2D operations only. Some take this as an indication that dynamic blocks are only 2D, this is wrong. A dynamic block can hold 3D geometry, but it can only manipulate the geometry in 2D. We will show you how far you can push these 3D aspects later.

Going through all the possibilities of a dynamic block would take us too far. For this class we will focus on four functions that are the most essential for our switch design object:

- Graphical presentation;
- Variants;
- 3D geometry;
- Data management.

Graphical presentation

The graphical presentation of our design object is incredibly important. It is through the graphical interface that designers work and communicate with others. In presentations we must make the same difference that AutoCAD made between Model and Layout Space, some graphical elements are for the designer only and some are for sharing. Secondly, it is possible to view the same object in different ways (multiple layouts of the same model). We conclude that we will need a system to manage what we see.

Let us start at the beginning: how many and what type of presentations do we need?

Switch Design Views	
New Switch Theoretical Placement	The minimum presentation needed to correctly place the switch or turnout, better known as the theoretical triangle.
Existing Switch Theoretical Triangle Placement	Not all switches are new, so we need to be able to make a distinction with existing switches.
New Switch Theoretical Triangle Design Aids	To aid applying the placement regulations. These aids can also be used in the verification process.
New Switch Theoretical Triangle Placement Details	When a switch is placed during design, the coordinates are less important, but for later execution these details are needed to physically place the switch.
New Switch Assembly Material Order	Being able to display material details is needed to place the correct order but also to validate the design.
New Switch Clearance 3D Coordination	To verify that there is enough space to place the switch and check possible spatial interfaces with other disciplines.

In our first version, we chose to make use of “Visibility States” in the dynamic block. For each view in the table above, a visibility state was made. Geometry can be turned on or off in each state. The result for the user is a grip that gives a dropdown list from which a Visibility State can be selected. They can also be selected in the properties window of the block.

It worked great and the drafters liked the easy grip, until we tried to make a layout with two different views. We could not use a different visibility state in another viewport. One solution would be to just copy the dynamic block and put both blocks in different states, but then we would not have one object. I have to point out that this only works because of the Layer system. That system makes it possible to turn the layers of a different visibility state invisible. Conclusively this was not the solution (example see Block: Switch Visibility strategy1).

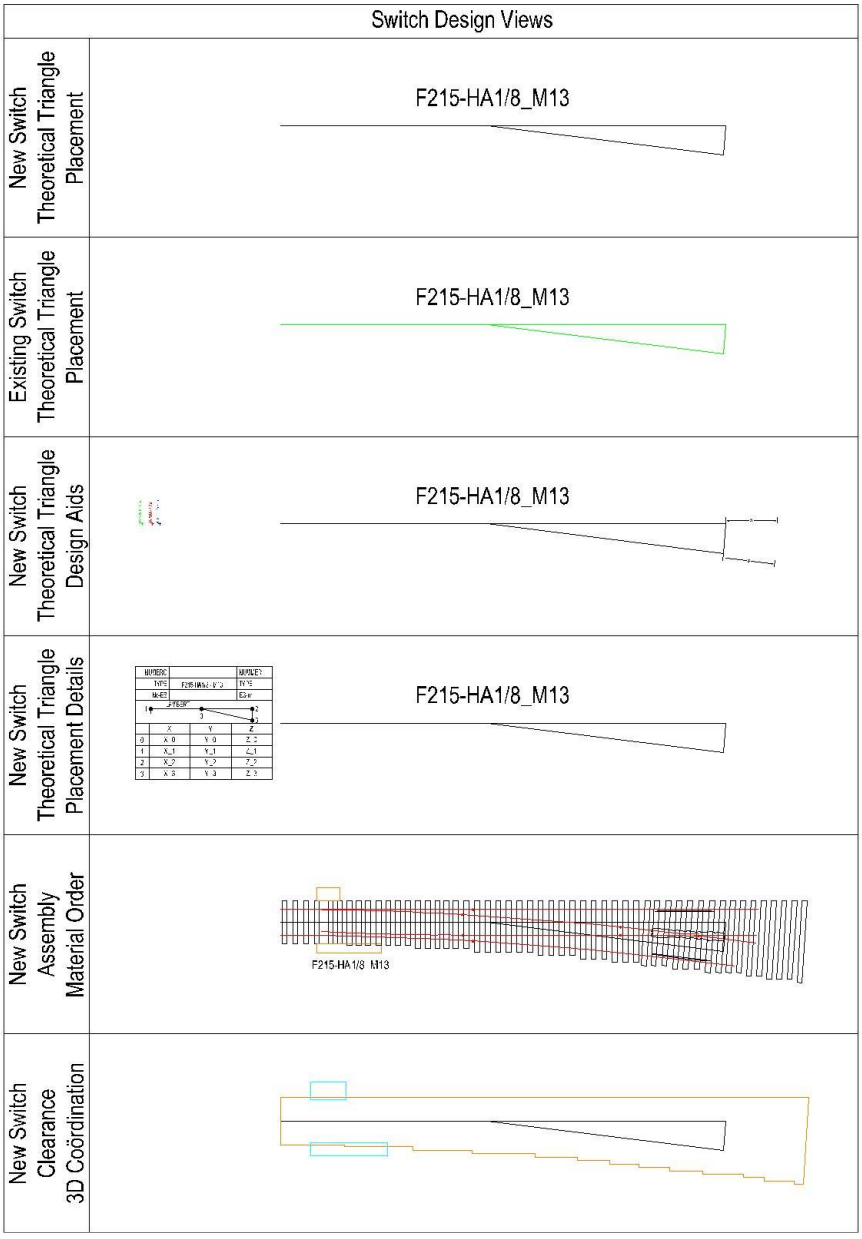


FIGURE 10: SWITCH DESIGN VIEWS

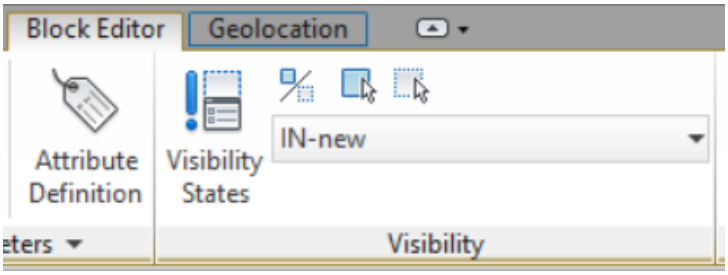


FIGURE 11: BLOCK EDITOR VISIBILITY

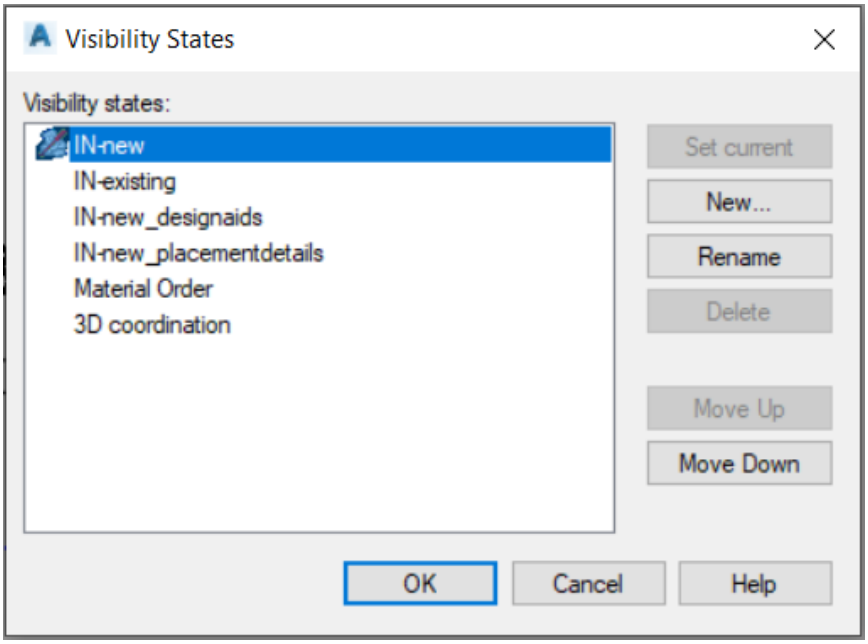


FIGURE 12: VISIBILITY STATES

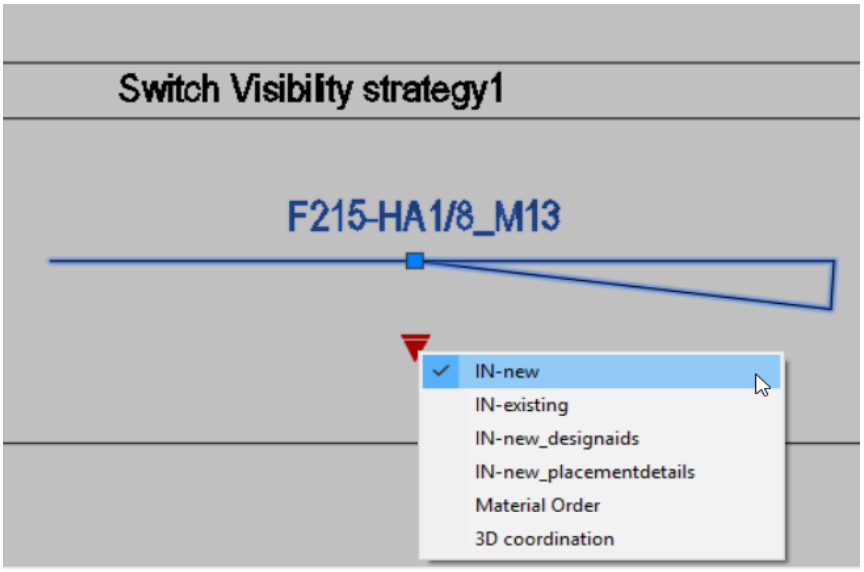
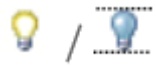


FIGURE 13: SWITCH VISIBILITY STRATEGY1

The solution is not intrinsic to dynamic blocks. For this, we need something basic that we know all too well: “Layers”. AutoCAD already has a robust system that controls the presentation of its objects. We just need to refresh the rules needed to apply when using Layers in combination with blocks.

Layer 0 used in a block	Every object in the block on Layer 0, will use the active Layer when the block is inserted.
Object properties ByLayer	The block objects will use the same properties as defined in the layer properties. The block follows the layer system.
Object properties ByBlock	The block object properties can be overridden after the block is placed. Each block can have its own settings.
Object properties other than ByLayer or ByBlock	These properties cannot be overridden after the block is placed.

Next are the Layer States:

On/Off		When a layer is on, it is visible and available for plotting. When a layer is off, it is invisible and not available for plotting (even with Plot On).
Thaw/Freeze		Thaws and freezes layer in all viewports. AutoCAD does not display, plot, hide, render, or regenerates objects on a frozen layer.
Lock/Unlock		You cannot edit objects on a locked layer.
Plot/Do not Plot		If you turn off plotting for a layer, the object on that layer are still displayed (layers that are on and thawed).
Current VP Thaw/Freeze		Freezes selected layers in the current layout viewport.
New VP Thaw/Freeze		Thaws and freezes selected layers in new layout viewports.

After this refresh, we can recreate the block. At this point, it reverts to a basic block (not dynamic). The grip has disappeared, the Layers now completely manage the visibility of the block (example see Block: Switch Visibility strategy2).
To help manage visibility, drafters can make use of Group Filters and Properties Filters (LAYERPROPERTIES MANAGER) or use the Layer States Manager. These tools facilitate changing the presentation, same as the grip did in our first strategy.

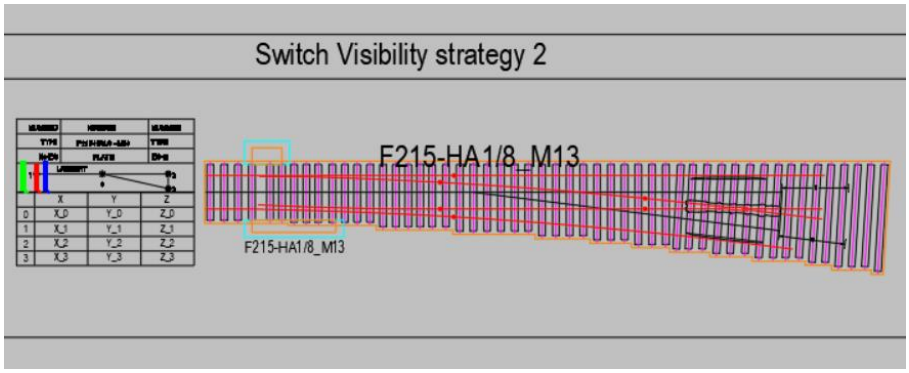


FIGURE 14: SWITCH VISIBILITY STRATEGY2

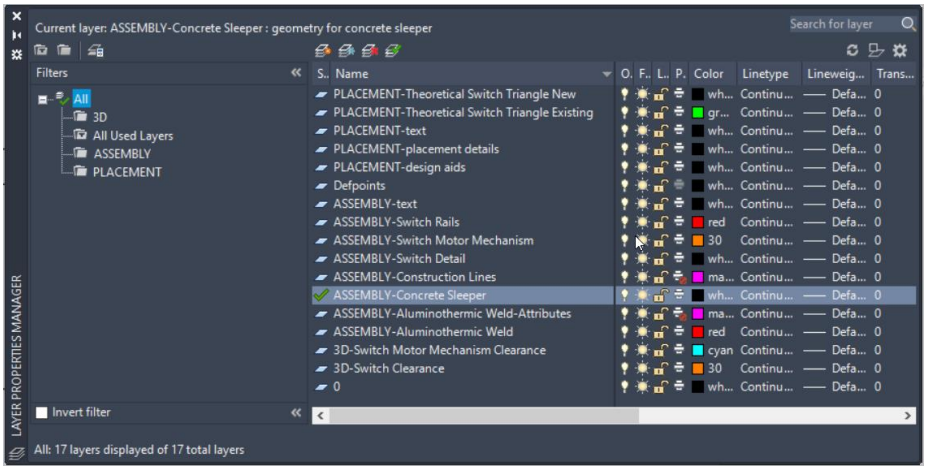


FIGURE 15: LAYER PROPERTIES MANAGER - GROUP AND PROPERTIES FILTER

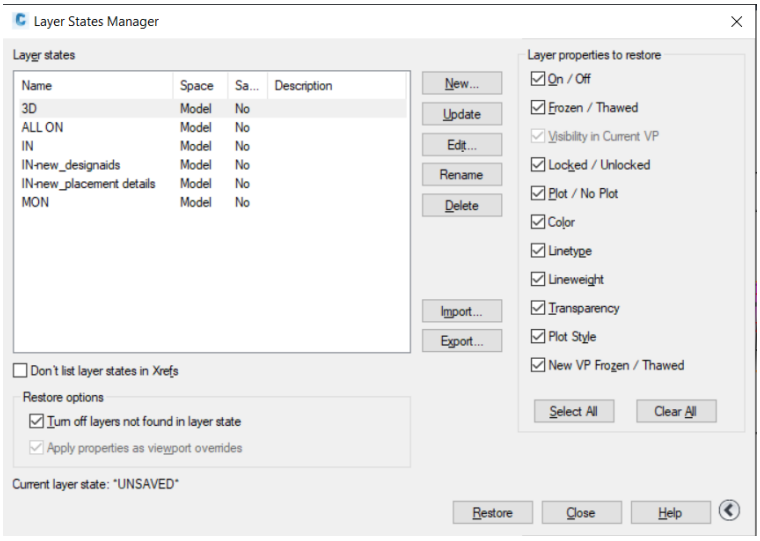


FIGURE 16: LAYER STATES MANAGER

Conclusion: as a golden rule, we can say that different presentations of the same object are best managed by the AutoCAD Layer system. The Visibility States of a dynamic block can in certain situations work, but they are not as powerful as using layers and will need layers to solve some display restrictions.

Variants

Let us look at the definition of the word variant to begin with:

“Something that is slightly different from other similar things.”

Therefore, a variant is not a different presentation of the same object. It’s another object that is only slightly different.

In the case of a railway switch, a variant can be defined using their type and deviation angle. Switches of the same type (single, symmetrical, double…) and same deviation angle are considered variants. In our case all variants come from the national switch catalog. This catalog contains a large number of possibilities and restrictions that impact which switch can be used where. By imbedding these standards into a dynamic block, the designer is able to interact with them more efficiently and the chance of mistakes decreases.

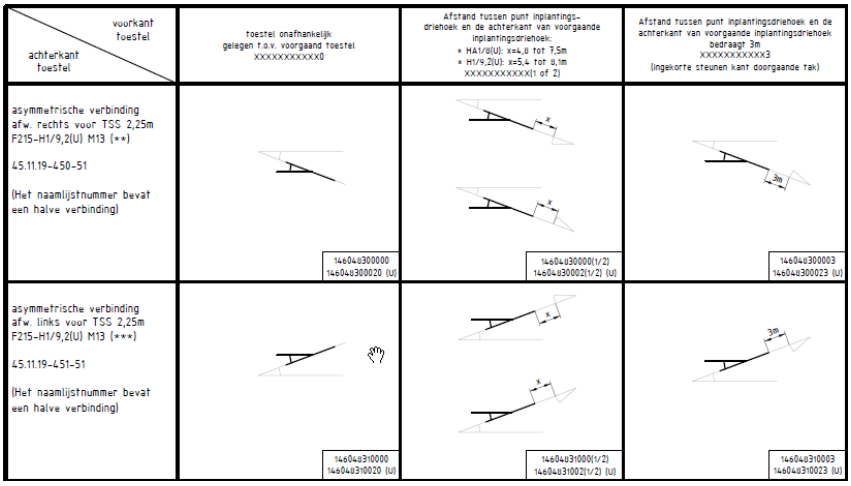


FIGURE 17: EXAMPLE NATIONAL SWITCH CATALOG

The differences between the variants are:

- The chosen combination of point (front of the switch) and the frog (back of the switch).
- Left or right deviation-motor position (defined by looking from point to frog).
- Yes or no second drive mechanism.

Depending on the design object you are making, it can have more or less differences, all dividable into two categories:

- PD - parametrical differences (position, scale, size, rotation, mirror, group,…).
- CG - different composing geometry (geometry that only exists in some variants).

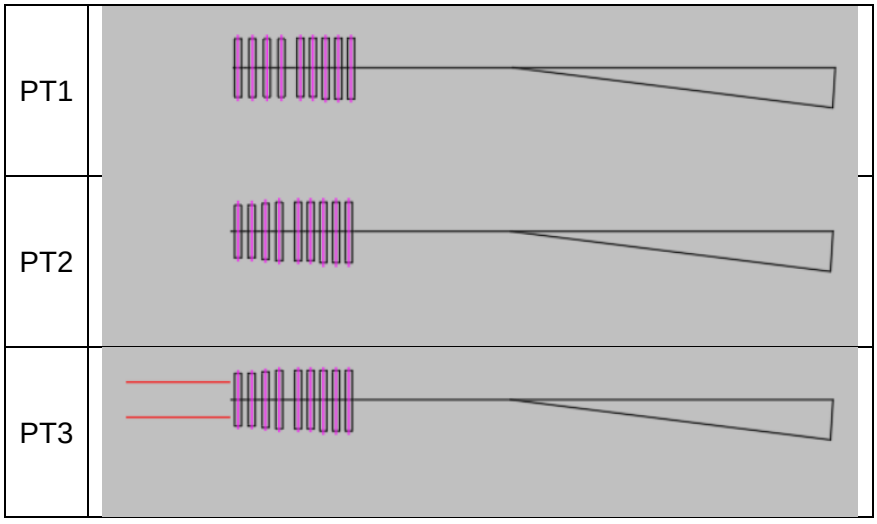
The first group can be controlled using the Geometric Constraints and Actions available in a dynamic block. The second group require Visibility States. The grid below shows what tools from dynamic blocks are used to create the switch variant.

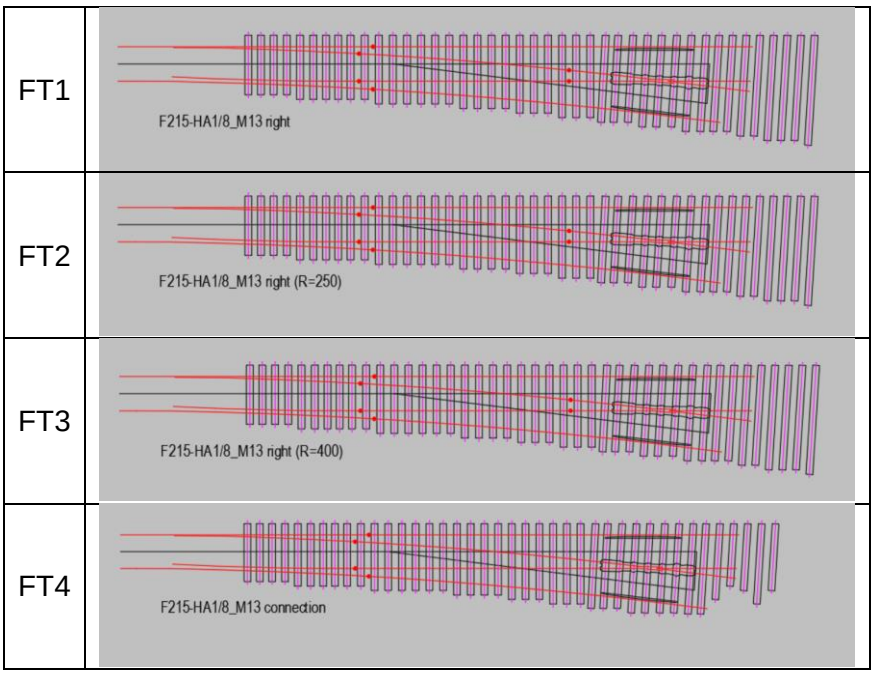
1	Point/Frog combination	CG	Visibility States controlled by a Double Lookup or a Block Table (user preference).
2	Left/Right deviation-motor	PD	Flip Action controlled by a Lookup.
3	Yes/No second drive	PD	Stretch Action controlled by a Lookup.

1. Point/Frog combination

To create all the different Visibility States, it is best practice to prepare all the geometry ahead of time. That way you can just copy it into the correct Visibility State.

In our example, we will use an object with 3 point and 4 frog variations. This already gives 12 Visibility States. Imaginge if we were to use Visibility States for all the differences, then we would have 96 Visibility States (3*4*2*2*2). Because it is not easy to manage a large amount of Visibility States, it is preferable to use Geometric Constraints and Actions or consider making a separate dynamic block. There is no fixed rule but try to make it user and maintenance friendly.





To control the Visibility States using the different point and frog variations, there are two options: a Double Lookup or a Block Table. They both do technically the same, only the interface is different and in the configuration screen of the Block Table you are able to copy/paste multiple entries at once (prepare everything in a spreadsheet and then copy it over).

A Double Lookup does not, by default, exist in the Block Editor. To add it, you must open the Block Editor and go to the Authoring Palettes, Parameter Sets tab. Right click on the Lookup set and select copy, next paste in the same pallet. Select the new Lookup Set and select properties. Now change according to the table below:

Name	Double Lookup Set
Description	Creates a lookup parameter with no Lookup grip associated with two Lookup action
Type	Lookup
Number of grips	0
Actions	Lookup, Lookup

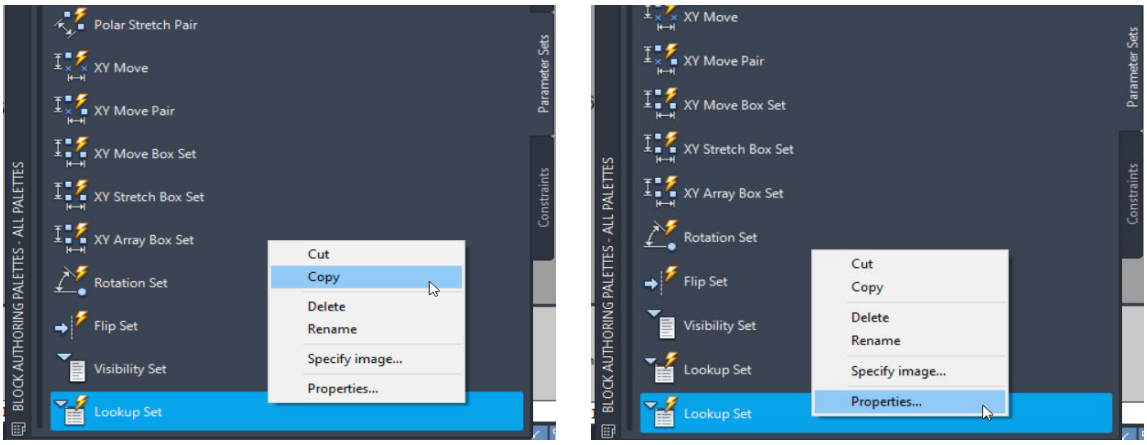


FIGURE 18: AUTHORING PALETTES - PARAMETER SETS - LOOKUP SET

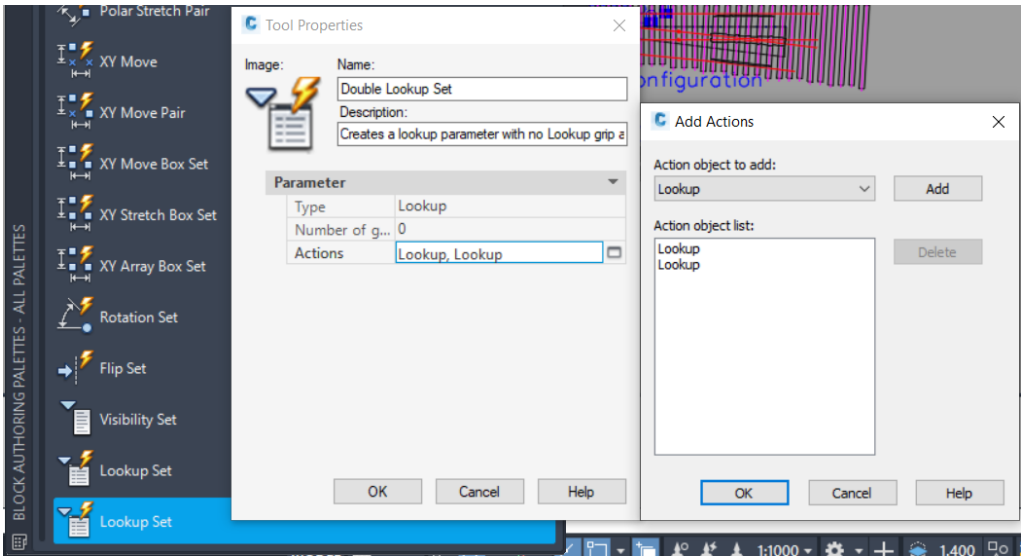


FIGURE 19: DOUBLE LOOKUP SET

You can now use a Double Lookup, but what does it do and why do we need it? The idea is that we want separate Lookups for the point and frog, but these two need to be combined into one visibility state (this is where the Double Lookup comes into play).

To make everything work, we need 2 Lookup sets (PointType and FrogType), 2 Linear Parameters (PT and FT) and 1 Double Lookup (SwitchType).

The first Lookups (Pointtype and Frogtype) will make it possible for the user to select the type of point and frog that is needed. The lookup makes it easy to add a logical description to the selection. Each Lookup effects the value of its linked Linear Parameter (PT and FT).

The Double Lookup SwitchType combines all the Linear Parameter combinations with all the possible Visibility States. One Lookup Action controls the Linear Parameters (PointFrogActions), the other the Visibility States (Visibility States). Because they are connected to the same Lookup Parameter, the Actions will connect and work simultaneously.

The result is a dynamic block whose Visibility States can be controlled by individual lookups that can be placed in logical locations.

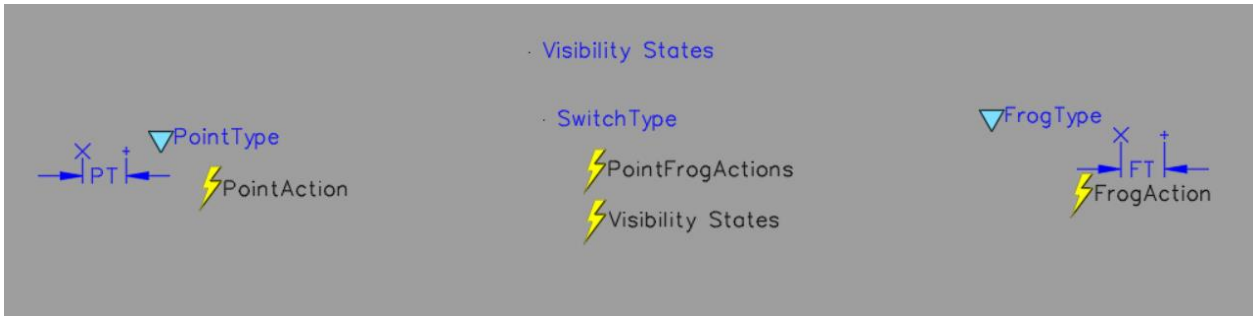


FIGURE 20: DOUBLE LOOKUP LINKING TWO LOOKUPS AND VISIBILITY STATES

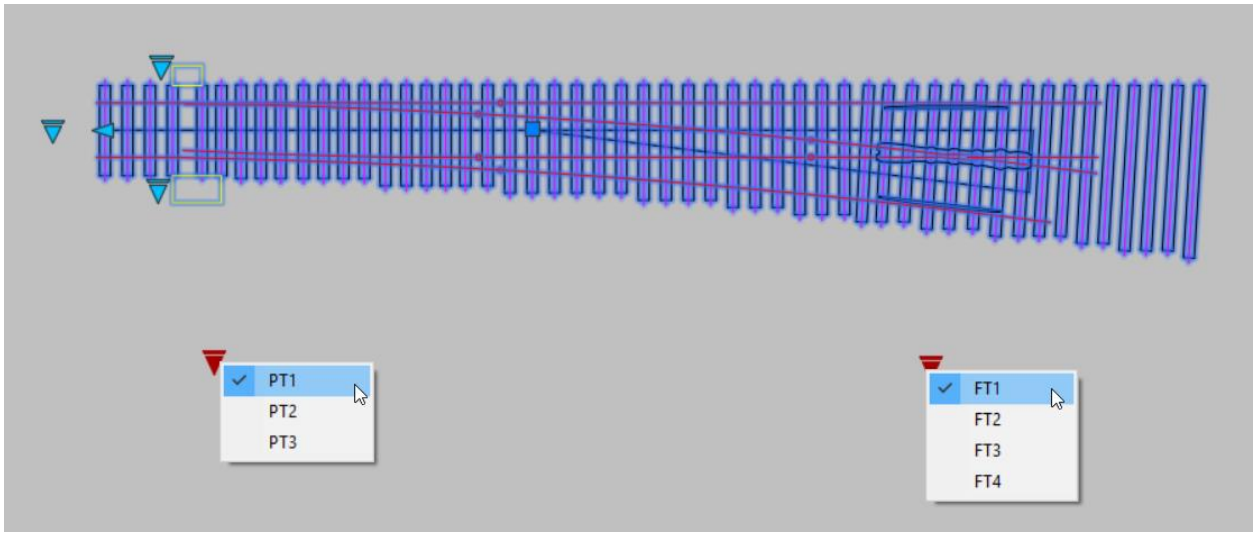


FIGURE 21: TWO LOOKUPS CONTROLLING DYNAMIC BLOCK VISIBILITY STATES

An alternative for this method is to use a Block Table. This Action is more straightforward to configure but only gives one grip to control all the variations in one dropdown list or table. There can only be one Block Table in the dynamic block.

In the Authoring Palettes under the Actions tab select the Block Properties Table. Double click the table icon or select Block Table (Dimensional) in the Block Editor Ribbon.

You are given an empty table; you can add new parameters or add existing. We need to add the Visibility States and define two new string parameters (PointType and FrogType) that will be uses to select the correct Visibility State. Once these are added, all combinations can be defined in the grid (you can also prepare this in Excel and copy all fields in at the same time).

The result is a dynamic block with one grip from which the user can select all the combinations defined in the Block Properties Table. Double Row Values are combined to make a selection cascade. It is also possible to open the Properties Table and continue the selection from there.

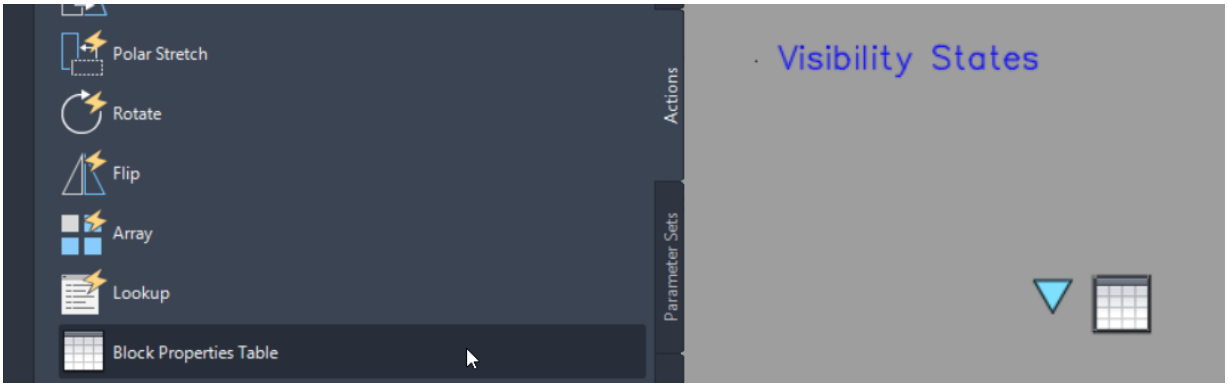


FIGURE 22: BLOCK PROPERTIES TABLE ACTION

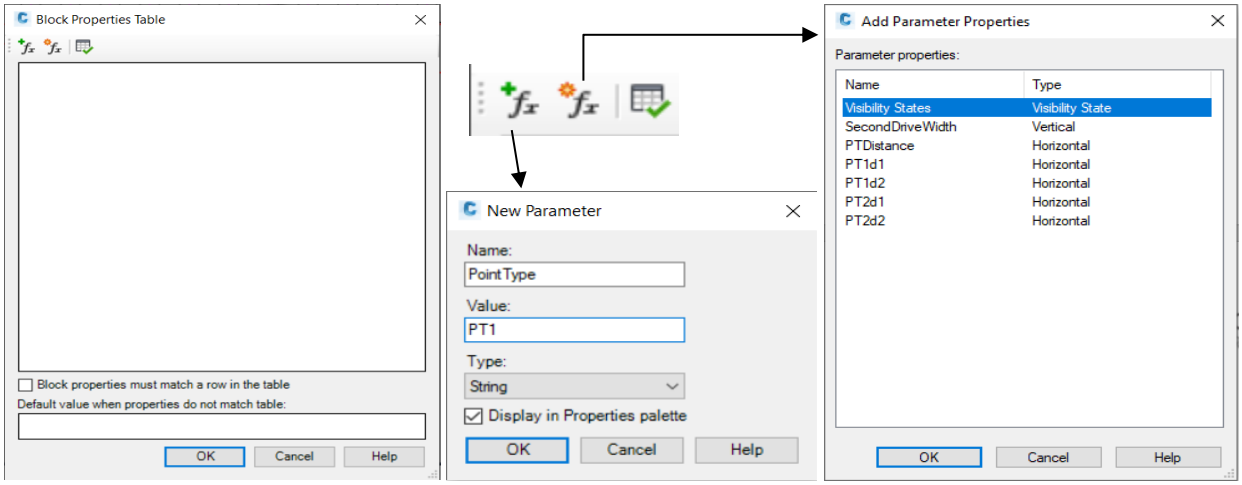


FIGURE 23: BLOCK PROPERTIES TABLE CONFIGURATION

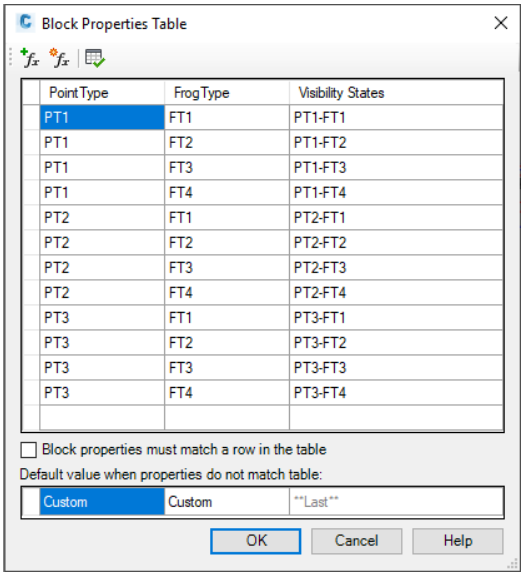


FIGURE 24: POINTTYPE-FROGTYPE-VISIBILITY STATE COMBINATIONS

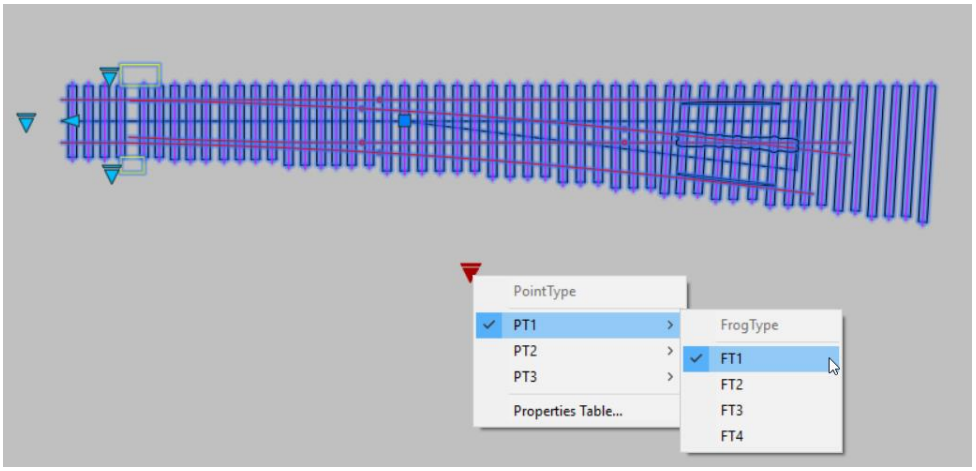


FIGURE 25: BLOCK TABLE CONTROLLING DYNAMIC BLOCK VISIBILITY STATES

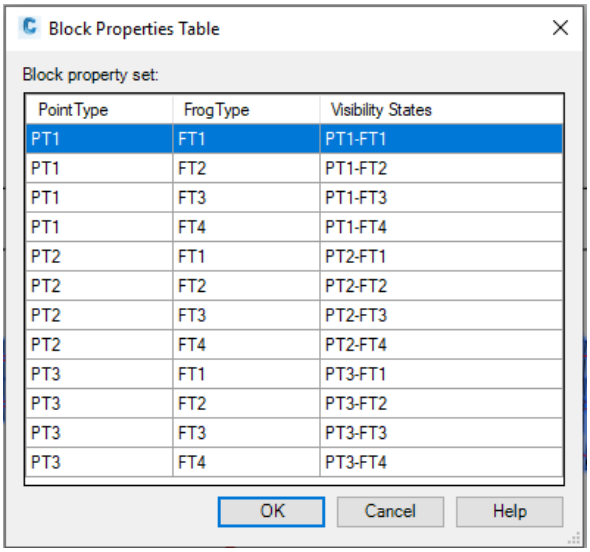


FIGURE 26: BLOCK PROPERTIES TABLE - BLOCK PROPERTY SET

2. Left/Right deviation-motor

The Flip Action/Parameter is specifically created for any mirror variations you need, no need to use Visibility States. In the switch block we use multiple flips at the same time, one for the switch as a whole and one for the motor and second drive.

This corresponds completely with the reality, depending on local conditions the motor might need to be placed on the other side (maintenance access). This is possible because each Flip has its own Action Selection Set (selection of geometry that it affects).

One downside of the Flip parameter is that it uses the terms “Flipped” and “Not flipped”. These terms can be completely meaningless in your case and could even form a hindrance for the users. To remedy this, you can use a Lookup to control the Flip parameter. By doing this, you can choose your own terms to change the flip position.

So, to start just add a flip set from the Parameters Sets tab in the Authoring Palettes to your block. Because we will be controlling the flip status with a Lookup, we can remove the grips from the Flip Parameter. To remove the grips, go to the properties of the Flip Parameter and set the Number of Grips to 0.

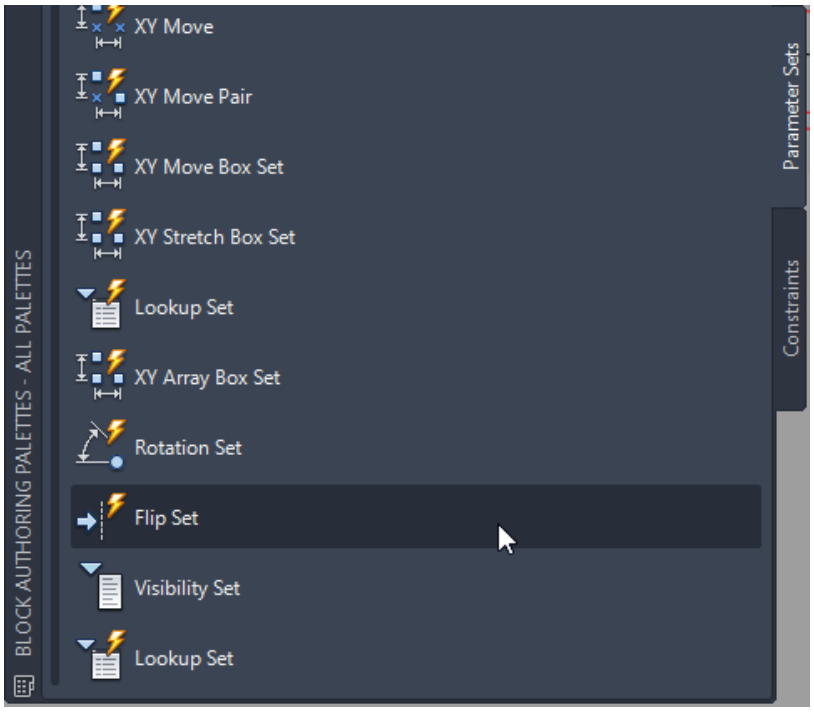


FIGURE 27: AUTHORING PALETTES – PARAMETER SETS - FLIP SET

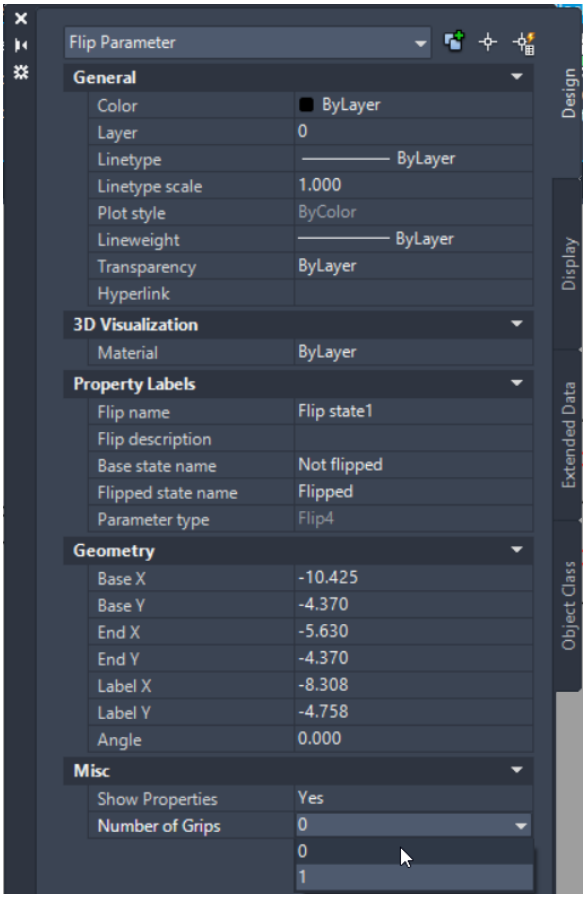


FIGURE 28: FLIP PARAMETER PROPERTIES

Next, we add a Lookup Set and add the flip parameter to the Lookup. We only need to define two situations: Flipped and Not flipped. Because we drew the switch with a right deviation, “Right” will be equal to “Not flipped” and “Left” will be “Flipped”.

The Name of the Lookup parameter can be used to indicate which geometry it affects, its position can be freely chosen to best suit the users’ needs.

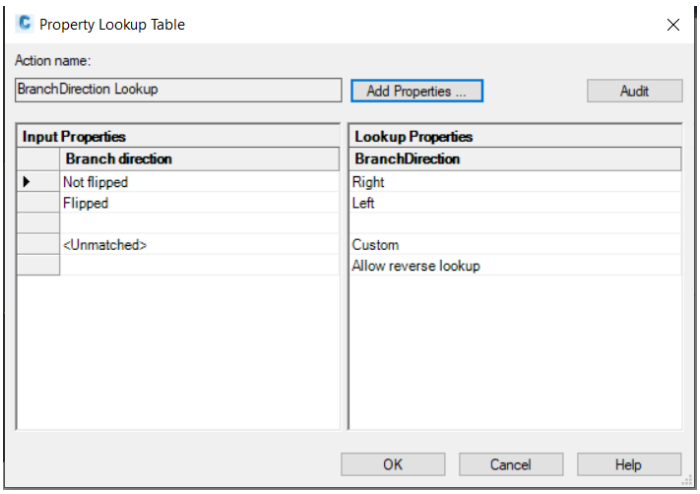


FIGURE 29: PROPERTY LOOKUP TABLE

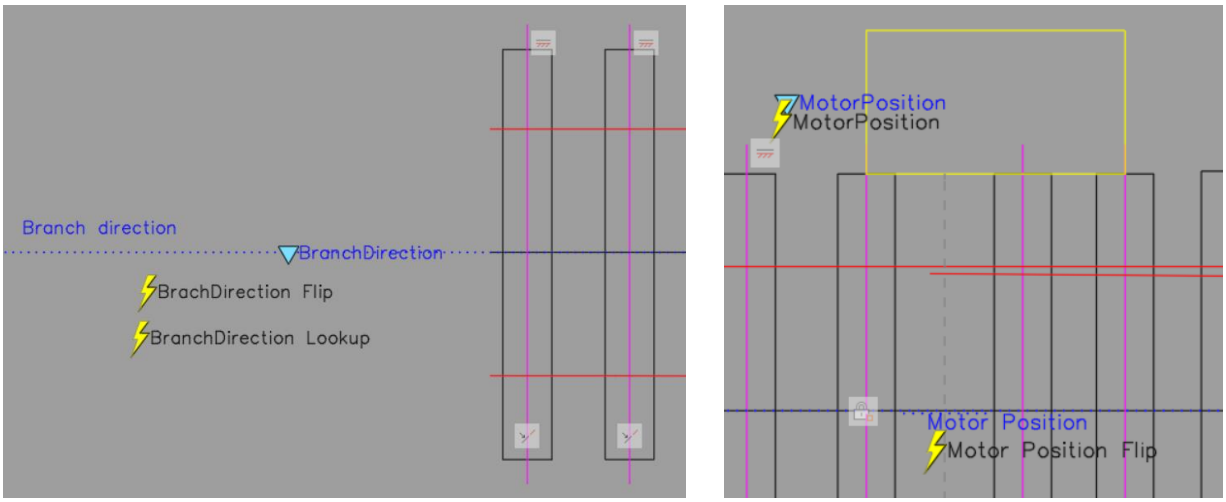


FIGURE 30: PARAMETER AND ACTION POSITIONS

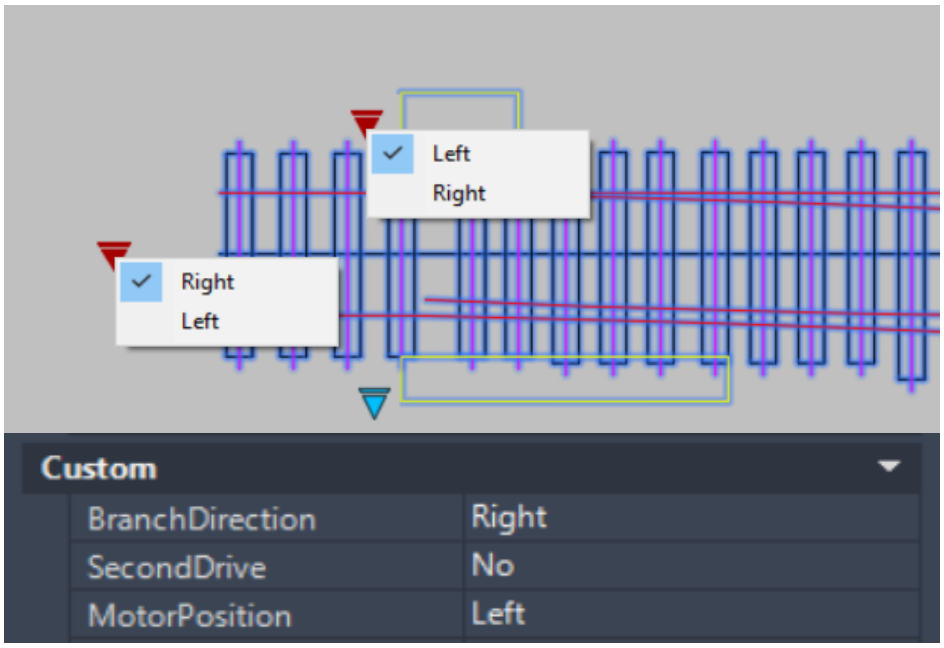


FIGURE 31: DYNAMIC BLOCK GRIPS AND CUSTOM PROPERTIES

3. Yes/No second drive

A dynamic block has too many possibilities to cover them all, so take some time to look at the help or look online. As a rule, before using visibility states look at all the Actions and Constraints. By combining constrains and actions you might not need visibility states.

In a switch, you could envision numerous geometric variations: sleeper lengths, sleeper spacing, rail lengths and many more. For this document, we will work out a method for a second drive on the motor. This second drive will need additional free space to work correctly and its presence is important information to order the correct motor.

For a start, we simply need a square polyline that we constrain to hold its shape (perpendicular corners, equal sides). The width is constraint and locked and for its length, a linear stretch is added. In this case, the grip of the Linear Parameter is removed and a Lookup is added. The Linear Parameter is added to the Lookup and value descriptions are added for the size variations (Yes or No second drive).

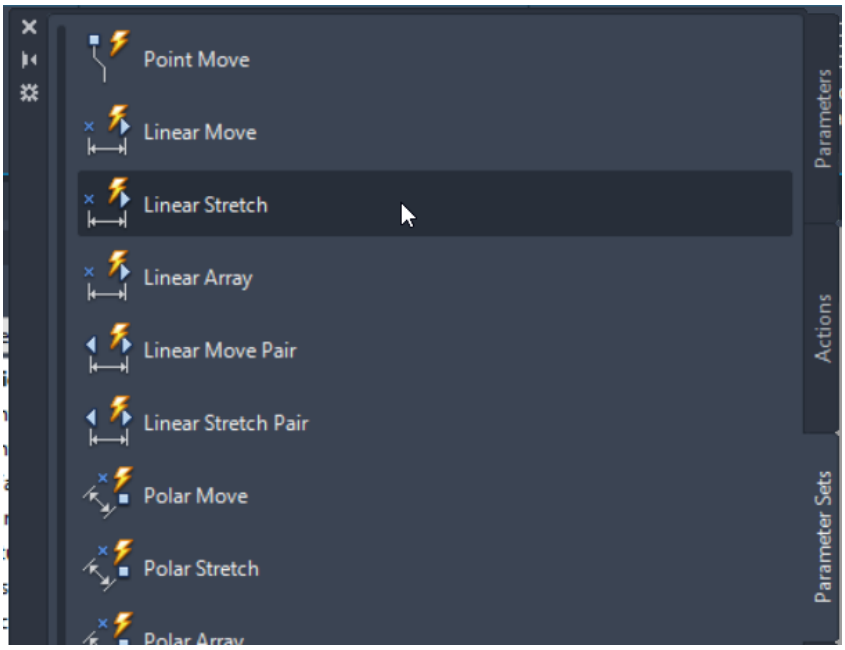


FIGURE 32: AUTHORING PALETTES – PARAMETER SETS - LINEAR STRETCH

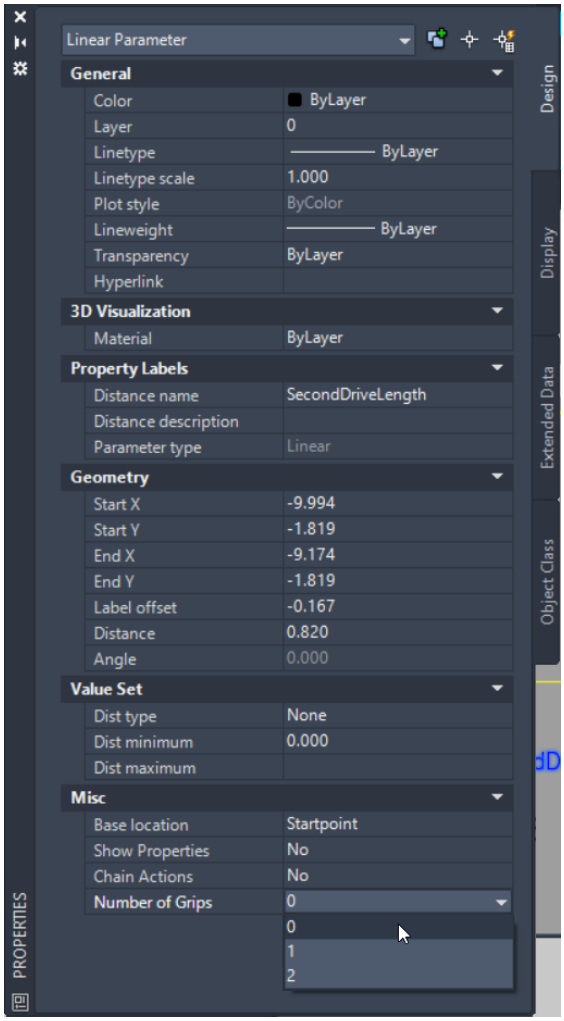


FIGURE 33: LINEAR PARAMETER PROPERTIES

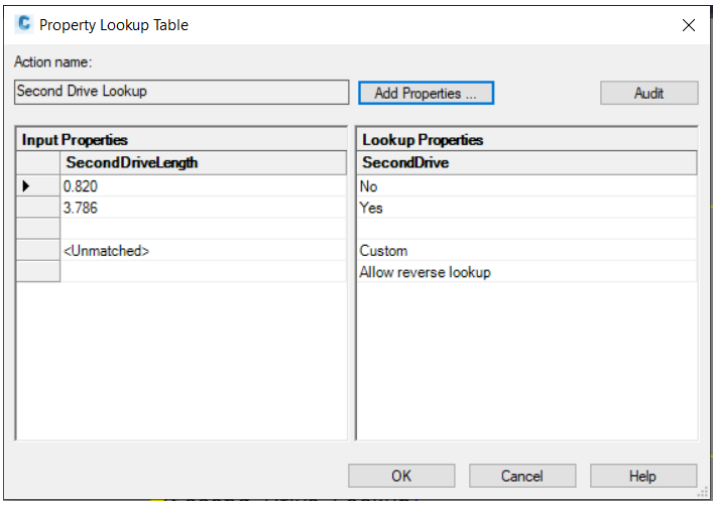


Figure 34: PROPERTY LOOKUP TABLE

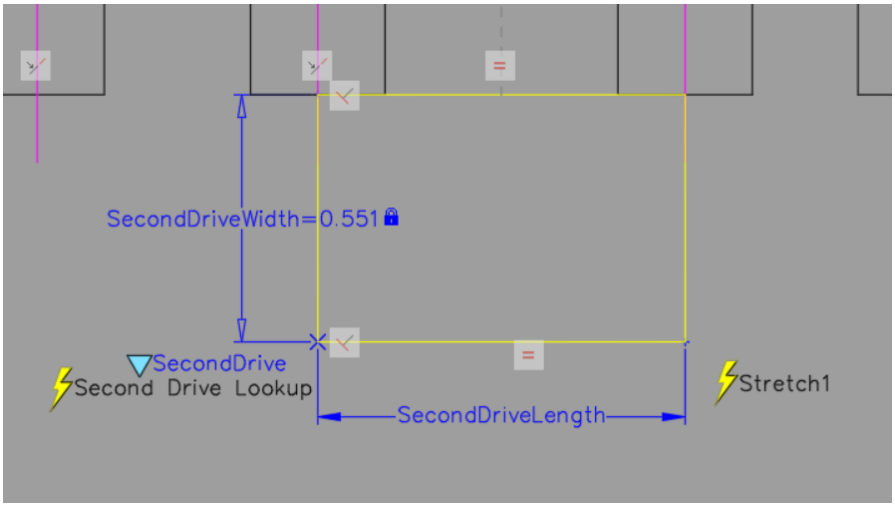


FIGURE 35: PARAMETER AND ACTION POSITIONS

The result is a block with a Lookup to indicate whether a second drive is used or not.

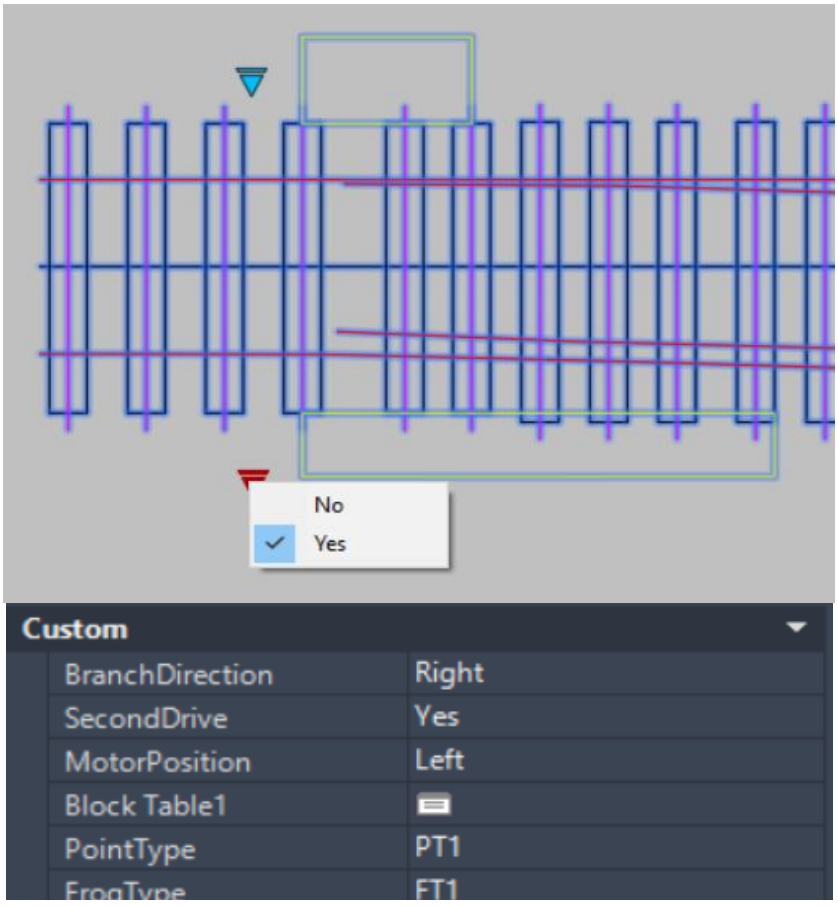


FIGURE 36: DYNAMIC BLOCK GRIPS AND CUSTOM PROPERTIES

3D geometry

Can you use 3D geometry in a dynamic block?

Yes, a block and therefore a dynamic block has no problem with containing 3D geometry. There are, however, restrictions on the impact that actions and controls have on the contained 3D geometry. So, depending on the situation, a dynamic block might be enough for your 3D needs. However, if you need full 3D parametric capabilities (AutoCAD solid modeling, Inventor, Revit), then a dynamic block is not the solution.

To begin with, all controls and grips in a dynamic block are 2D. They will be displayed in the block base plane and nowhere else. Therefore, grips cannot be placed on different heights and you are not able to rotate in other plains. In general, there are no 3D interfaces to manipulate the geometry.

Secondly, not all Actions are able to influence 3D geometry. Move, Scale, Rotate, Flip and Array will work, but Stretch and Polar Stretch do not have the desired effect. This is because they are unable to affect certain 3D geometry types (solids).

Thirdly, visibility states work just the same as for 2D geometry. Keep in mind that variations quickly add up to a long list of states.

These restrictions lead to the conclusion that we can do basic manipulations on 3D geometry, but are not able to change their basic shape/form interactively. There are however two loopholes that can be exploited:
1. Extrude based on geometry in the base plane (+ possibility to cap the extrusion).
2. Use an expression to give access to the extrusion parameters.

To extrude geometry, we cannot use a solid extrusion, because then we lose the connection with the planar 2D geometry. Nevertheless, if we use surface mode in the extrusion command, then the connection persists. There is one additional restriction, the system variable SURFACEMODELINGMODE must be set to 0 (procedural surface), a setting of 1 (NURB surface) will not work for this loophole.

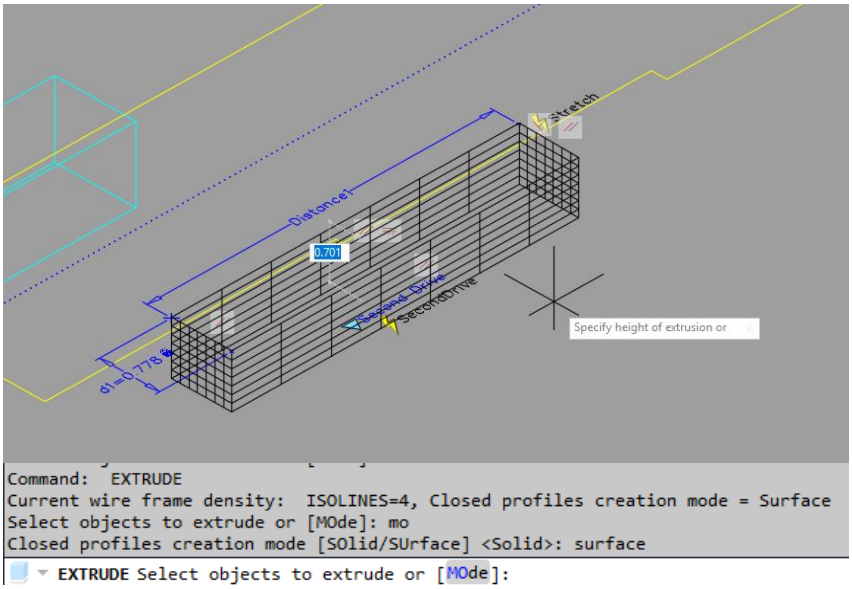


FIGURE 37: EXTRUDE - SURFACE MODE

The result are four surfaces that follow the changes made to the 2D geometry that was used as a base. Of course, this is not yet a closed shape. To do that we will need two additional commands that close the top and bottom: PLANESURF and SURFPATCH. With PLANESURF, we create a plane surface using the 2D base geometry and with SURFPATCH, we connect the bottom surface extrusion edges to form a surface. We end up with a completely encapsulated 3D shape.

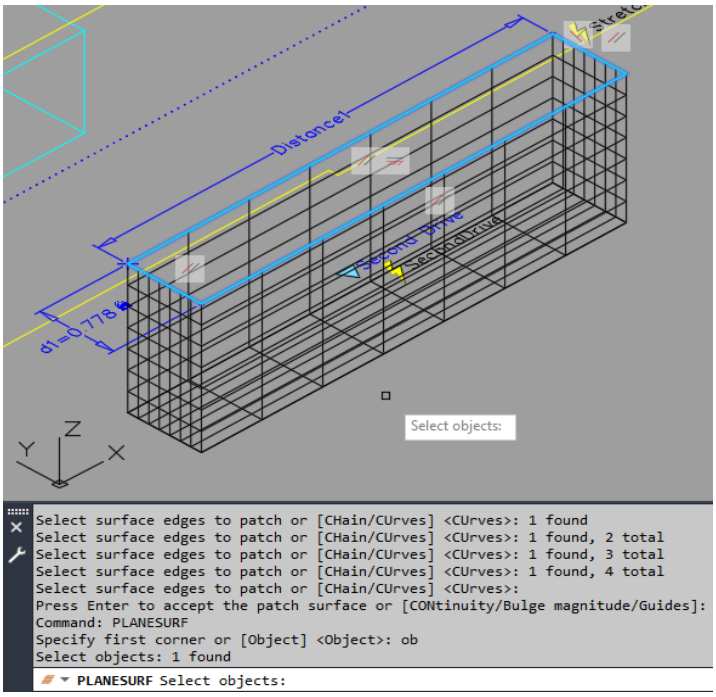


FIGURE 38: PLANESURF - PLANAR SURFACES

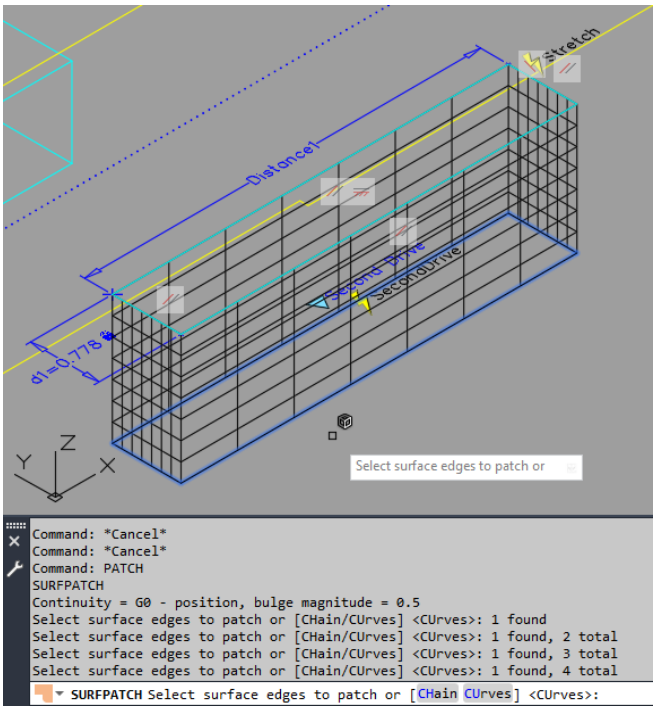


FIGURE 39: SURFPATCH - CAP SURFACE EDGE

Now the geometry is already parametrical in regards to its 2D base geometry, but the extrusion height is a manual value. This can be remedied by defining a new parameter (SwitchDepth) and then using this as an expression in the extrusion Height property.

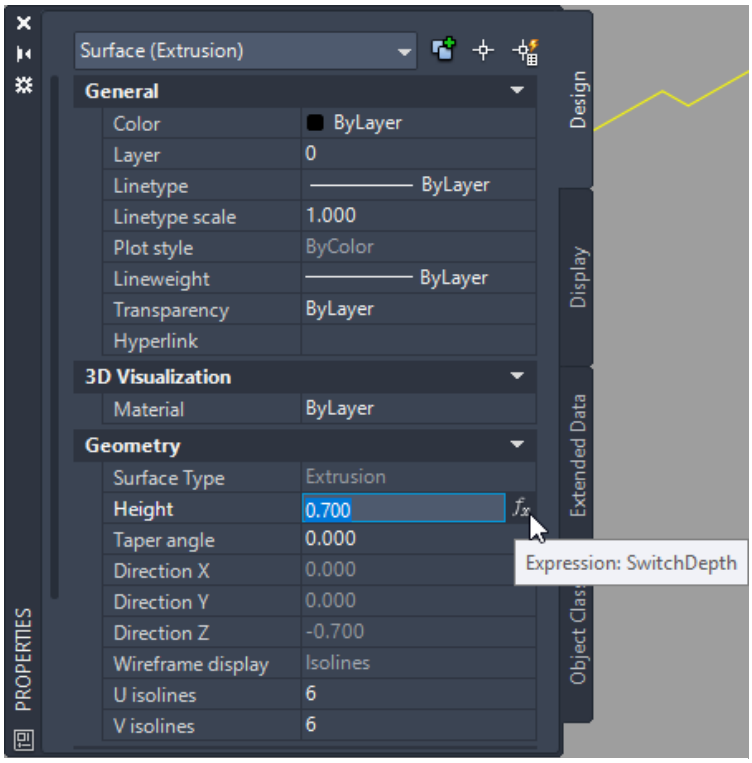


FIGURE 40: SURFACE(EXTRUSION) - HEIGHT EXPRESSION

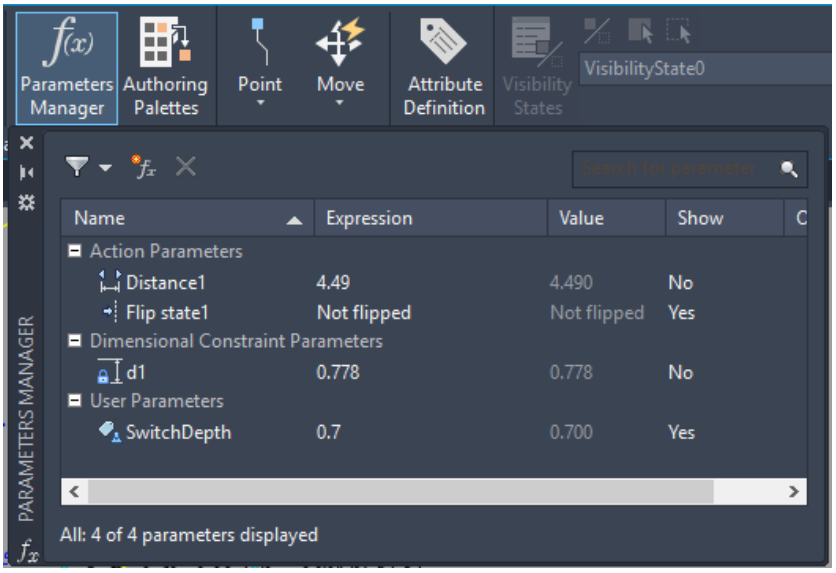


FIGURE 41: CUSTOM USER PARAMETER - SWITCHDEPTH

The result is a 3D extrusion geometry that can be controlled through the controls available on a dynamic block.

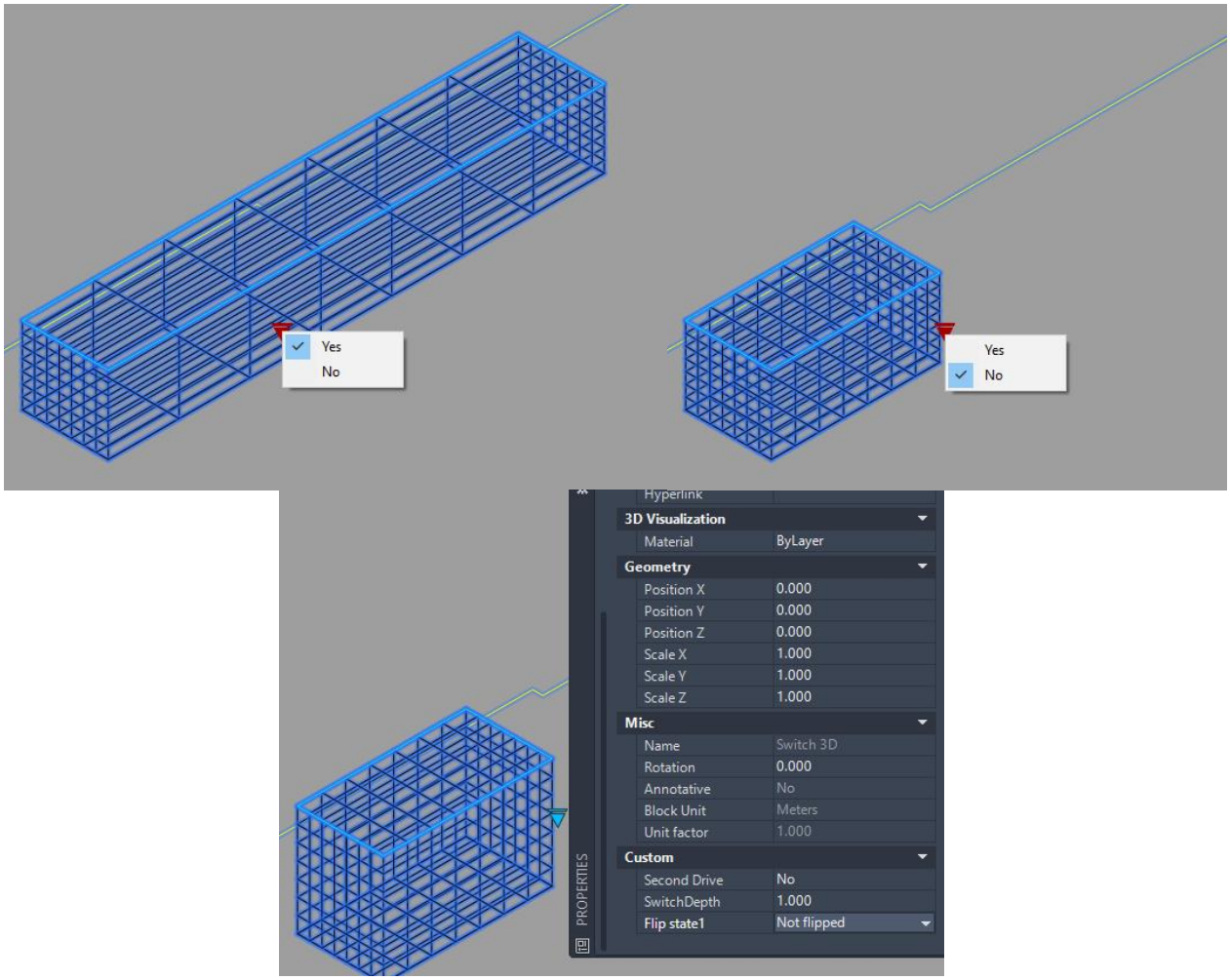


FIGURE 42: DYNAMIC BLOCK PARAMETRIC 3D GEOMETRY

There are some additional restrictions when these loopholes are combined with the general dynamic block actions (never made for this application). Our first loophole will work with all actions that work on 3D geometry, but the sequence of commands can give different results (test before you take in production). The second loophole does not combine with the flip and array action.

Data Management

The subject of Data Management is very important for the interaction with the user, safeguarding the functionality of our design object and exposing the data to interact with other design objects.

In all the earlier examples we have constantly been creating custom data for our design object, most of the time this custom data has been the digital representation of design decisions:

- Left/Right switch or motor;
- Yes/No for the placement of a second drive;
- Point/Frog combination.

Depending on how the dynamic block is configured, the data is invisible or visible and read or write. The data type and identification are also managed using the block editor.

Besides the standard block properties, all custom data can be split into two groups: Attributes and Parameters. Both are shown in the Parameters Manager and grouped together (Action Parameters, User Parameters and Attributes). Whether or not a Parameter is shown outside of the block editor is configured here, Attribute visibility is configured upon creation or in their respective property palette.

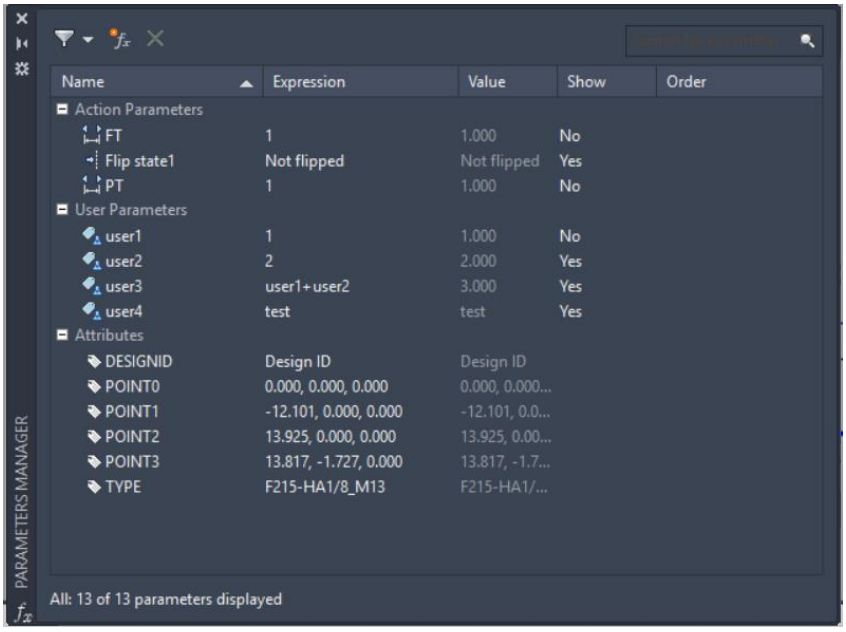


FIGURE 43: BLOCK EDITOR - PARAMETERS MANAGER

The expression (how the value is calculated) of an Action Parameter is directly related to the Action type and the configuration of the Action (done in its property pallet). For User Parameters, the type can be selected in the Block Properties Table (Real, Distance, Area, Volume, Angle, String). The Block Properties Table and grip can be deleted after making all the needed User parameters. If a numeric type is used, then the user can define a formula using other parameters. All Attributes expressions are of the type string (text), so an attribute can only give a text result (numeric values are digitally expressed as text). The expression can however contain a field, which can retrieve data from a variety of sources.

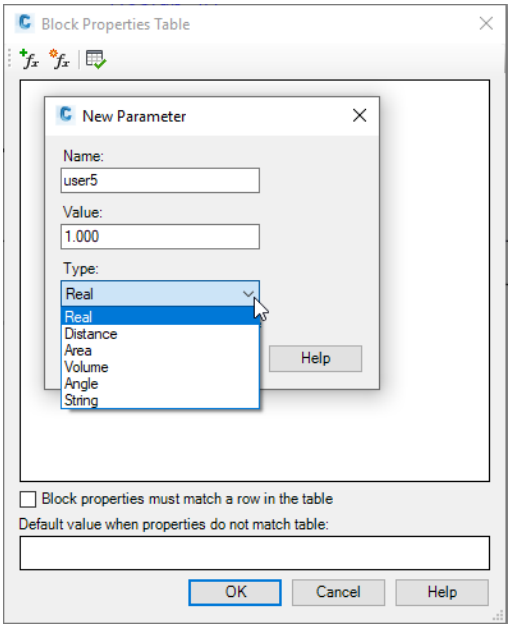


FIGURE 44: BLOCK PROPERTIES TABLE - USER PARAMETER TYPES

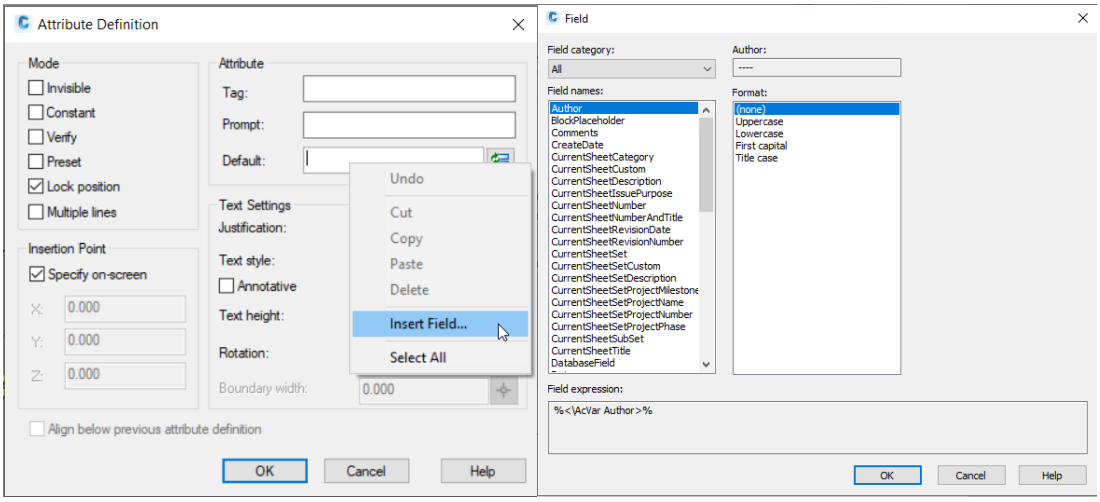


FIGURE 45: ATTRIBUTE - INSERT FIELD

The goal of this document is not to show all possibilities, but rather to demonstrate the potential. For this reason, the following subjects are demonstrated in this block:

- Combining multiple parameters into one unique identifier;
- Exposing geometric data;
- Configuring data access.

To combine multiple parameters into one identifier we can simply use a Lookup Set (without grip if we want it to be controlled by its connected parameters). By creating unique combinations of the input properties, we are able to connect a unique Lookup property. This can be very useful to facilitate the selection of one type in a large catalogue. The resulting ID can then be seen in the property pallet of the block reference or it can be used in an Attribute field definition (all of this is dependent on Parameters Show attribute being active).

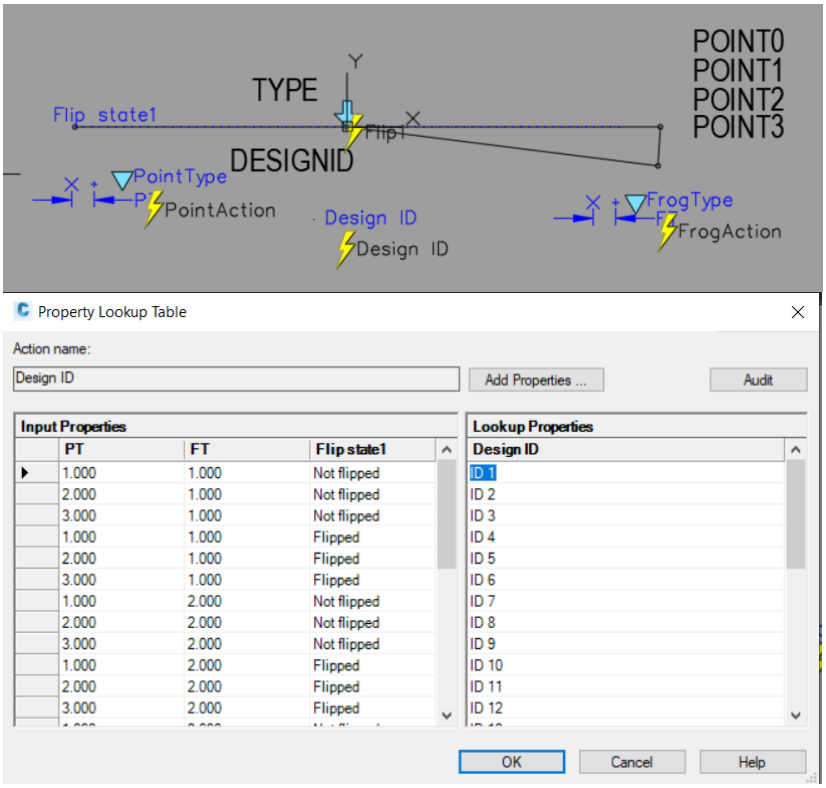


FIGURE 47: LOOKUP SET - UNIQUE ID

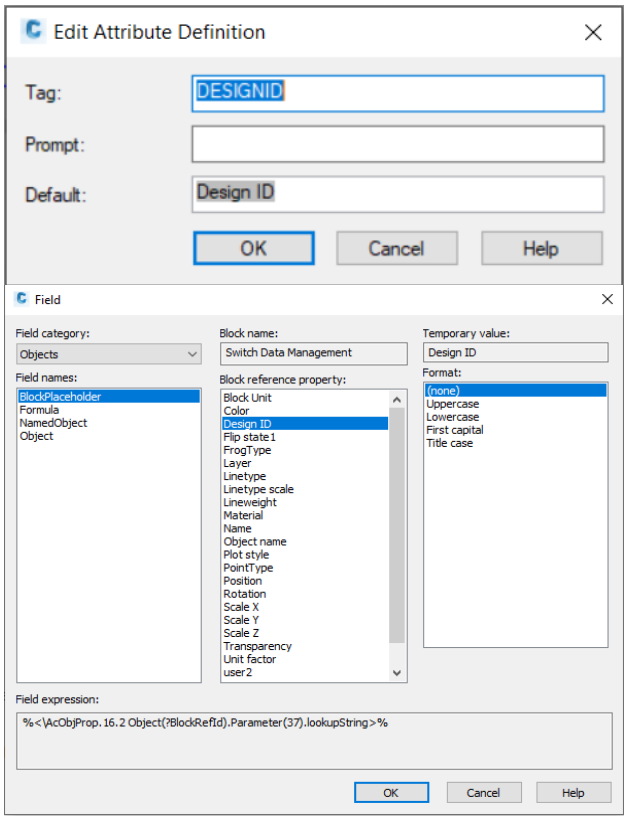


FIGURE 46: UNIQUE ID IN ATTRIBUTE FIELD

The goal of the second subject is to expose geometric data, but not using the coordinate system in the block editor. We want to give geometric data about where points are in the world coordinate system (an individually placed block). To do this we must first create an AutoCAD object, for example: a circle, a point, a square. We will then create a reference to this object in the field definition of an attribute. In this case, the center of the selected circle object is used to communicate the coordinates of a particular point. The AutoCAD object can be placed on an invisible and non-plotting layer. Do not forget to “REGEN” before evaluating Attributes that use fields.

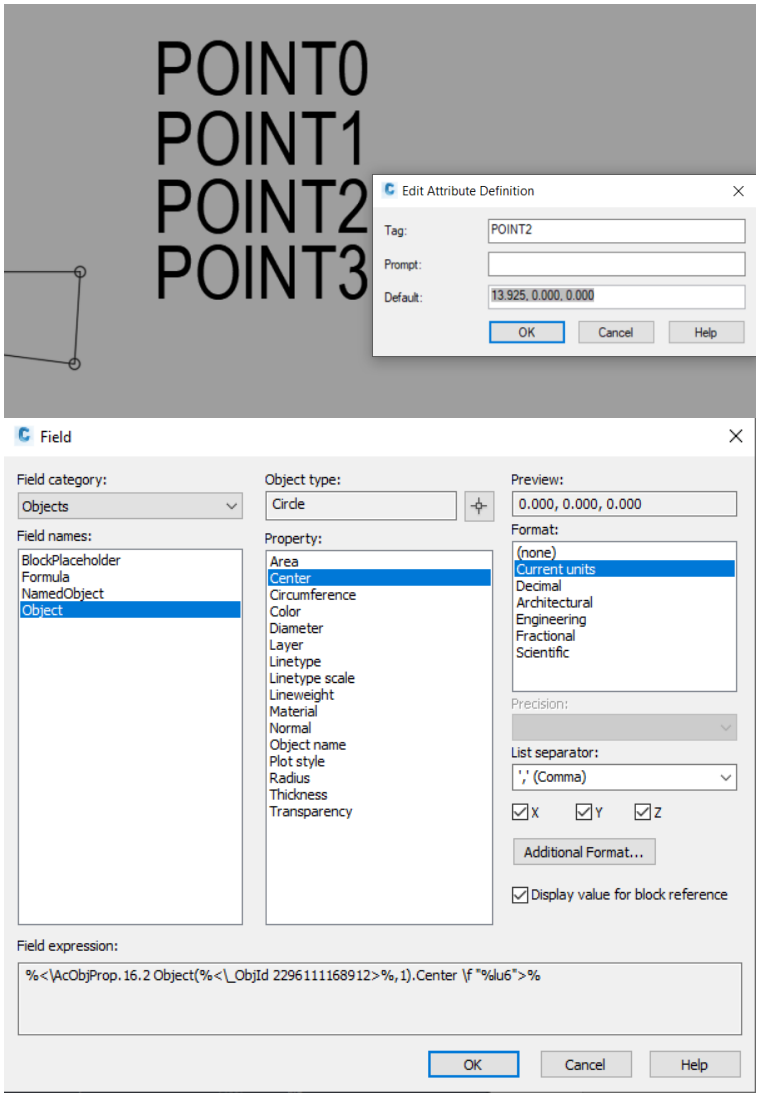


FIGURE 48: BLOCK GEOMETRIC DATA - ATTRIBUTE FIELD - OBJECT

There is no true data protection in a dynamic block, but there are ways to make it harder to change things by configuring your data access correctly.

On a single parameter you can only choose to make it invisible for the user, that way he cannot change the value. The downside is that he also does not know what the value is. A solution is to create a second value that uses the first one as a reference. A formula cannot be edited by the user, in this way you are able to build calculations that use user input but also have fixed values.

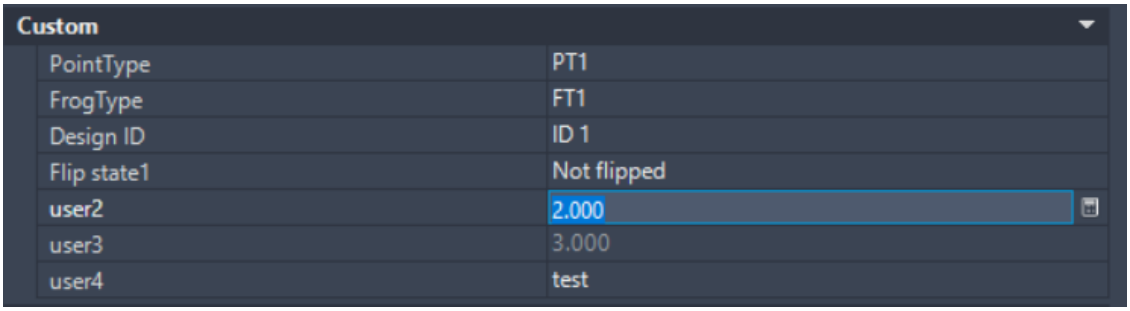


FIGURE 49: PARAMETER ROEAD ONLY - USER3 = USER1 + USER2

For Attributes, you can define a constant value to guarantee that the user cannot change the value. The downside of a constant value is that it converts any field value to a fixed value, so it does not update. An alternative is to lock the layer on which the attribute is placed. After doing this, the user is no longer able to edit the attribute value. The only thing you should not forget is that the layer needs to be unlocked on the moment of creation of the block; else, the field relations will be lost and cannot be recreated.

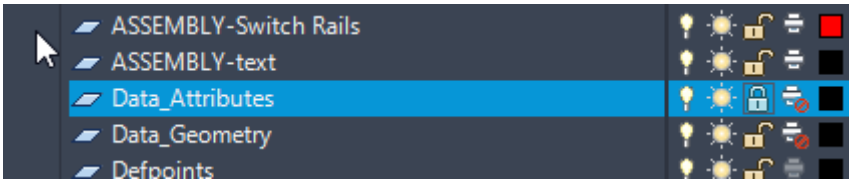


FIGURE 50: ATTRIBUTE LOCK LAYER

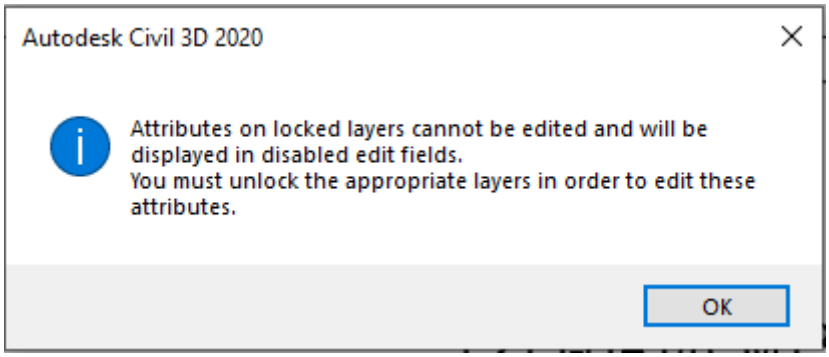


FIGURE 51: ERROR ATTRIBUTE ON LOCKED LAYERS

Transforming Design Data

What is design data and how does it relate to a design object?

Design data is simply the information that is contained in a design object. The input data that is needed to create the object, the process data that is handled inside the object and the output data. On the image below you can see an illustration of an AutoCAD circle design object and all the design data it contains.

FIGURE 52: AUTOCAD CIRCLE OBJECT

This data is used:

- To create the object and potential following objects.
- To connect with other design objects.
- To analyze design objects.

Why shouldn't we use the entire design object?

As defined earlier, a design object is composed for a specific task. Using it in its totality for other tasks causes specific cooperation issues:

- Communicating with the entire object is not clear and precise. There is unused data that is irrelevant to the current task. This data takes up space and time.
- By using the object, we are in essence creating a copy of that object. Copies can lead to versioning issues and a desynchronization in our design.
- If the entire object is shared, then it becomes unclear who is the owner of the object and there is less control over who has access to the data in the object.

Autodesk Dynamo for Civil 3D

To use design object and data properly, we need tools that make the interaction with them intuitive and efficient. In Civil 3D and AutoCAD this is done by the user interface (ribbons, buttons, command line) and all the functions that ship with the product.

It is all but certain that the standard solutions will not cover all the processes and needs in your design. For this reason, Autodesk made an API (application programming interface) available. People with enough programming skills can make custom solutions to satisfy the design needs. However, the required programming knowledge and the fact that you yourself are making a complete function, becomes a threshold that limits the amount of people that choose this solution.

Dynamo changes this dynamic. Because Dynamo is a visual programming tool, you do not need training in traditional programming to use it. It visualizes the design objects and data, that are already available in the form of code in the API, and makes it less complex for the user by using simple nodes and wires. However, these simple nodes and wires in no way limit the potential of the API. It is simply a different representation that makes the creation of custom workflows easy and flexible. Rather than building complete functions, programmers only need to build nodes. Combining the nodes into a workflow can be done by the people who are directly involved in the design. The result is the potential of a very large user base.

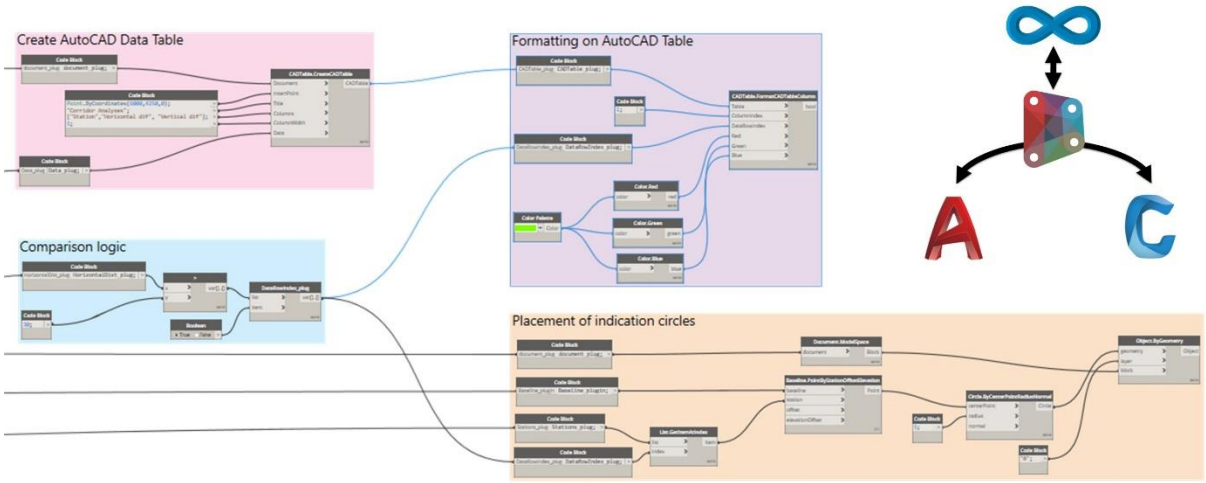


FIGURE 53: DYNAMO FOR AUTOCAD AND CIVIL 3D

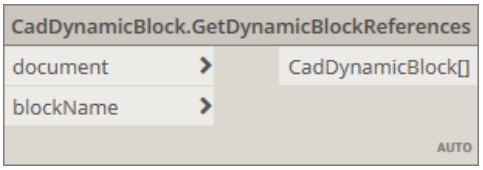
Using Dynamo, we can now make new connections between AutoCAD and Civil 3D objects and design completely new workflows. We are also able to build connections with other external sources; the possibilities are only limited to the creativity of the user.

To illustrate this, three new rail workflows have been developed using the switch design object defined in the previous section:

- Switch (asset) Placement System;
- Switch – Profile;
- Switch – Corridor.

Custom Dynamo Nodes

CadDynamicBlock.GetDynamicBlockReferences

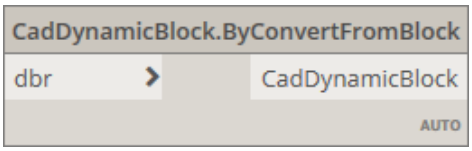


This node not only exposes basic Blocks but also dynamic Blocks. It retrieves Block references by name in a document.

Input:
- document: document from which to retrieve Block references (document);
- blockName: name of the Block definition (string);

Output:
- CadDynamicBlock: list of (dynamic) Block references (CadDynamicBlock).

CadDynamicBlock.ByConvertFromBlock

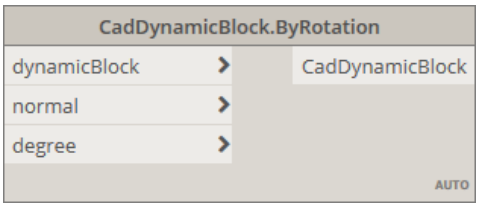


This node converts a default block reference into a custom CADDynamicBlock. This node is needed to interact with other custom nodes.

Input:
- dbr: default block reference (BlockReference).

Output:
- CadDynamicBlock: list of (dynamic) Block references (CadDynamicBlock).

CadDynamicBlock.ByRotation

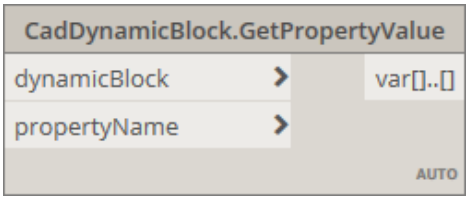


This node rotates a (dynamic) block around a chosen normal for a specified amount of degrees.

Input:
- dynamicBlock: list of (dynamic) Block references (CadDynamicBlock).
- normal: vector representing the normal around which to rotate (vector).
- degree: amount of degrees to rotate (double).

Output:
- CadDynamicBlock: list of (dynamic) Block references (CadDynamicBlock).

CadDynamicBlock .GetPropertyValue

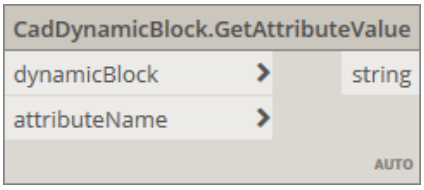


This node gets a property value by name from a dynamic block reference and returns a list of the values.

Input:
- dynamicBlock: list of (dynamic) Block references (CadDynamicBlock).
- propertyName: name of the Block property (string).

Output:
- var: list of property values (var).

CadDynamicBlock.GetAttributeValue

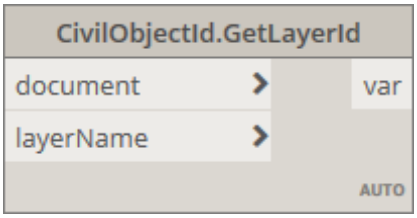


This node gets an attribute value by name from a dynamic block reference and return a list of strings.

Input:
- dynamicBlock: list of (dynamic) Block references (CadDynamicBlock).
- attributeName: name of the Block attribute (string).

Output:
- string: list of attribute values (string).

CivilObjectId.GetLayerId

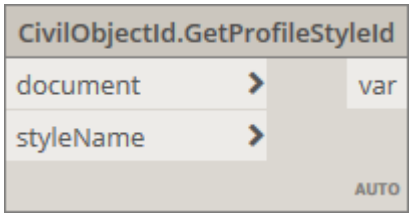


This node finds the ObjectId of a Layer using its name.

Input:
- document: current document (document).
- layerName: name of the layer (string).

Output:
- var: ObjectId (var).

CivilObjectId.GetProfileStyleId

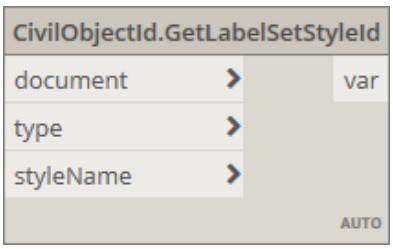


This node finds the ObjectId of a ProfileStyle using its name.

Input:
- document: current document (document).
- styleName: name of the ProfileStyle (string).

Output:
- var: ObjectId (var).

CivilObjectId.GetLabelSetStyleId

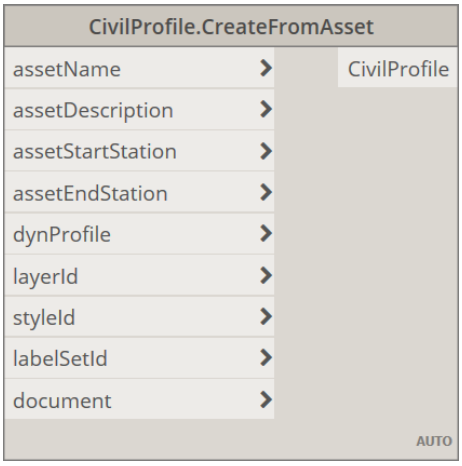


This node finds the ObjectId of a LabelSetStyle using its name and type.

Input:
- document: current document (document).
- type: LabelSet type (string).
- styleName: name of the LabelSetStyle (string).

Output:
- var: ObjectId (var).

CivilProfile.CreateFromAsset



This node creates a new profile to represent the asset position on design alignment and profile. The new profiles name, description, beginning and end come from the asset. Its elevation comes from the design profile. All style settings can be configured using the Label, Style and LabelSet ObjectIds.

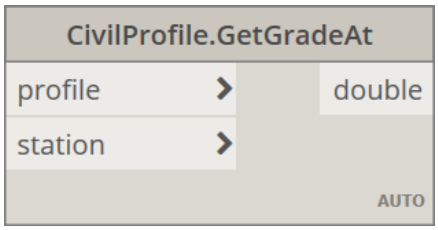
Input:

- assetName: name of the asset (string).
- assetDescription: description of the asset (string).
- assetStartStation: start station of the asset (double).
- assetEndStation: end station of the asset (double).
- dynProfile: design profile (Profile).
- layerId: ObjectId of the chosen layer (ObjectId).
- styleId: ObjectId of the chosen style (ObjectId).
- labelSetId: ObjectId of the chosen LabelSet (ObjectId).
- document: current document (document).

Output:

- CivilProfile: profile representing the asset (CivilProfile).

CivilProfile.GetGradeAt



This node retrieves the grade value on a specified station of a profile.

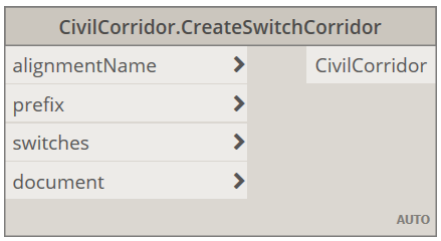
Input:

- profile: alignment profile object (Profile)
- station: specific station value (double)

Output:

- double: grade value on the specified station of the given profile

CivilCorridor.CreateSwitchCorridor



This node creates a specific switch corridor. Its name is based on its connected alignment and profile combined with a prefix and its description lists all contained switches.

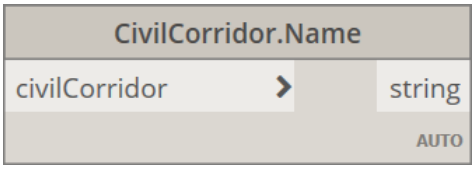
Input:

- alignmentName: alignment name (string).
- prefix: name prefix (string).
- switches: switch names (string).
- document: current document (document).

Output:

- CivilCorridor: corridor (CivilCorridor).

CivilCorridor.Name

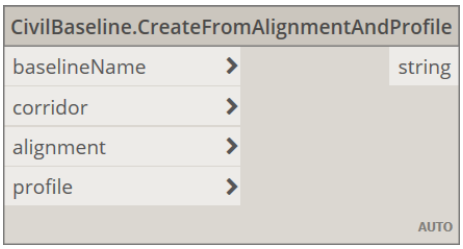


This node returns the name of the CivilCorridor.

Input:
- civilCorridor: corridor (CivilCorridor).

Output:
- string: corridor name (string).

CivilBaseline.CreateFromAlignmentAndProfile

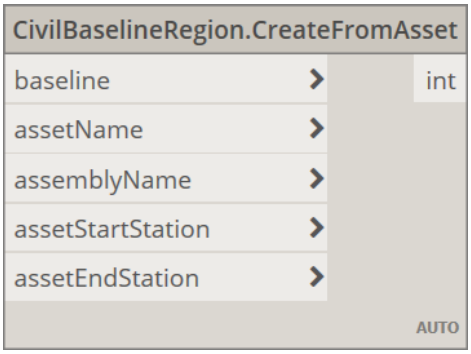


This node creates a new Baseline in an existing corridor object.

Input:
- baselineName: name for the new baseline (string).
- corridor: corridor (CivilCorridor).
- alignment: alignment (Alignment).
- profile: profile (Profile).

Output:
- string: corridor name (string).

CivilBaselineRegion.CreateFromAsset

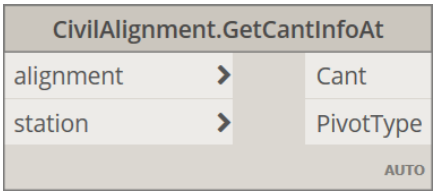


This node creates a new BaselineRegion in an existing baseline using an asset.

Input:
- baseline: existing baseline (CivilBaseline).
- assetName: name of the asset, will be used as region name (CivilCorridor).
- assemblyName: name of the assembly that is to be used in the region (string).
- assetStartStation: start station of the asset, start of the region (double).
- assetEndStation: end station of the asset, end of the region (double).

Output:
- int: BaselineRegion index (integer).

CivilAlignment.GetCantInfoAt



This node gets the Cant information on a specified station of an Alignment.

Input:
- alignment: Alignment object (Alignment, with cant data)
- station: specific station value (double)

Output:

- Cant: cant value on the specified station of the given alignment (var)
- PivotType: value representing pivottype (None, LeftRail, RightRail, Centerline) (var)

FIGURE 54: TRANSFORMING DESIGN DATA - CUSTOM NODES

Switch (asset) Placement System

There is no method to place a switch object (dynamic block), in its correct 3D position, using the civil 3D data that is available in the DWG file. To place the switch correctly, a feature line (3D line or polyline) of the central axis is needed for both alignments in the switch. Then the AutoCAD “ALIGN” command is used to position the switch on both lines. When the design changes this entire workflow needs to be repeated.

In this script, the civil 3D data is used to position the switch object directly. The alignment, profile and cant define the insertion point of the block, but also give the direction (XY plane), longitudinal slope (YZ plane) and cross slope (XZ plane).

Workflow

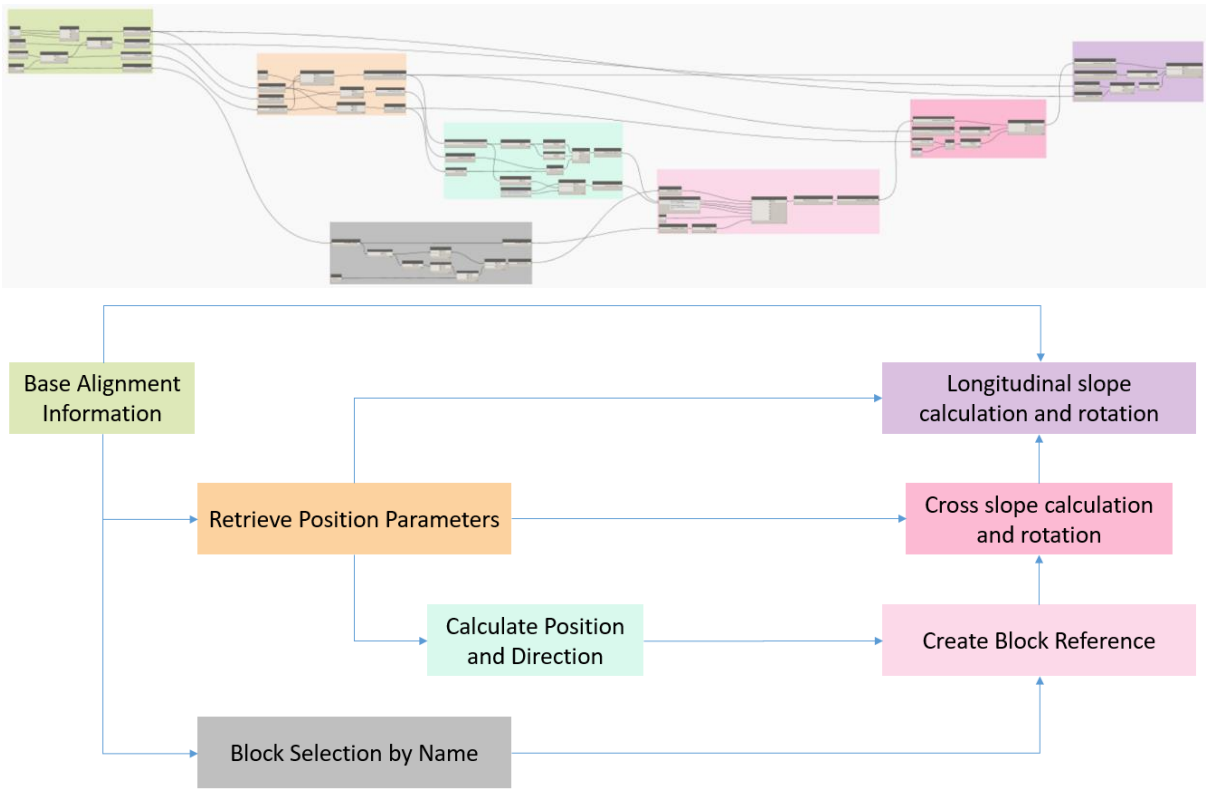


FIGURE 55: SWITCH (ASSET) PLACEMENT SYSTEM

Base Alignment Information:

To start the workflow all the base information is retrieved, to place the asset on the alignment. The alignment and profile are found using their name. For the stations a range is generated from the beginning until the end of the alignment with a station value every 100 units.

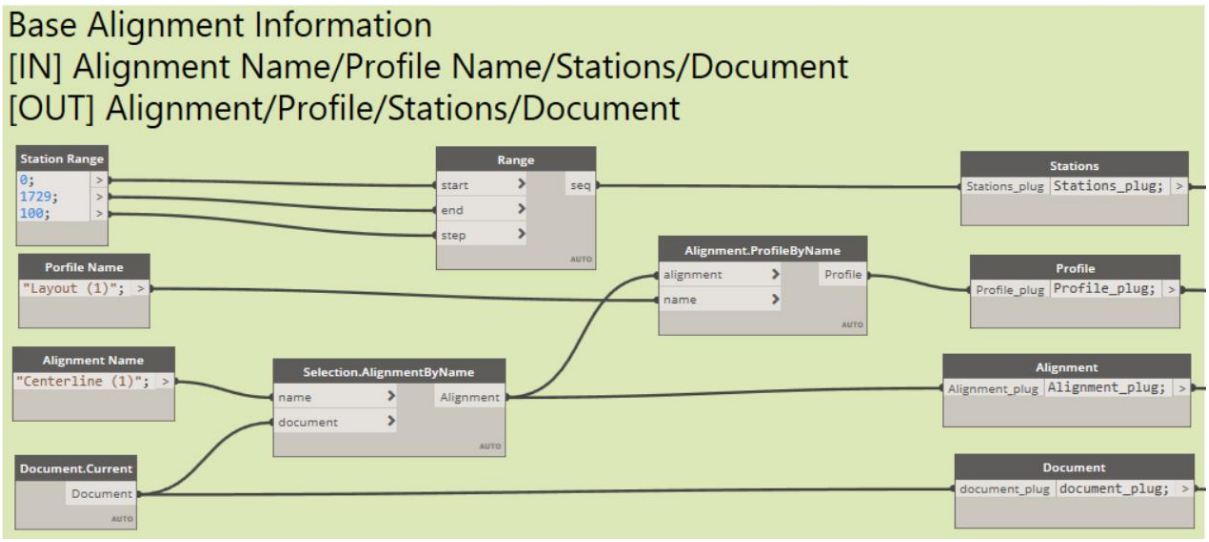


FIGURE 56: SPS - BASE ALIGNMENT INFORMATION

Block Selection by Name:

In this group, a Block definition is selected by its name. First, all block definitions and their names are found. Next, the index of the Block name that is the same as the user-defined name is found. This index is then used to select the correct block.

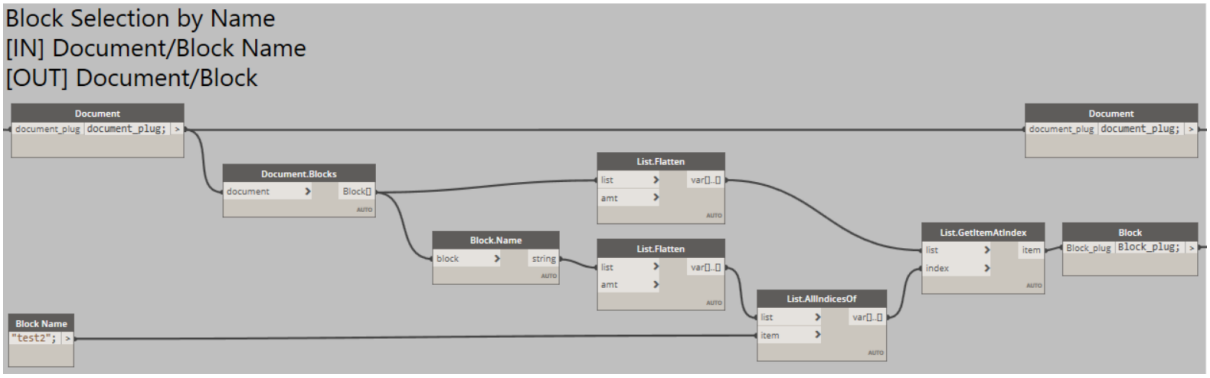


FIGURE 57: SPS - BLOCK SELECTION BY NAME

Retrieve Position Parameters:

For each station value (point), the corresponding coordinate system on the alignment is found as well as the elevation and cant value.

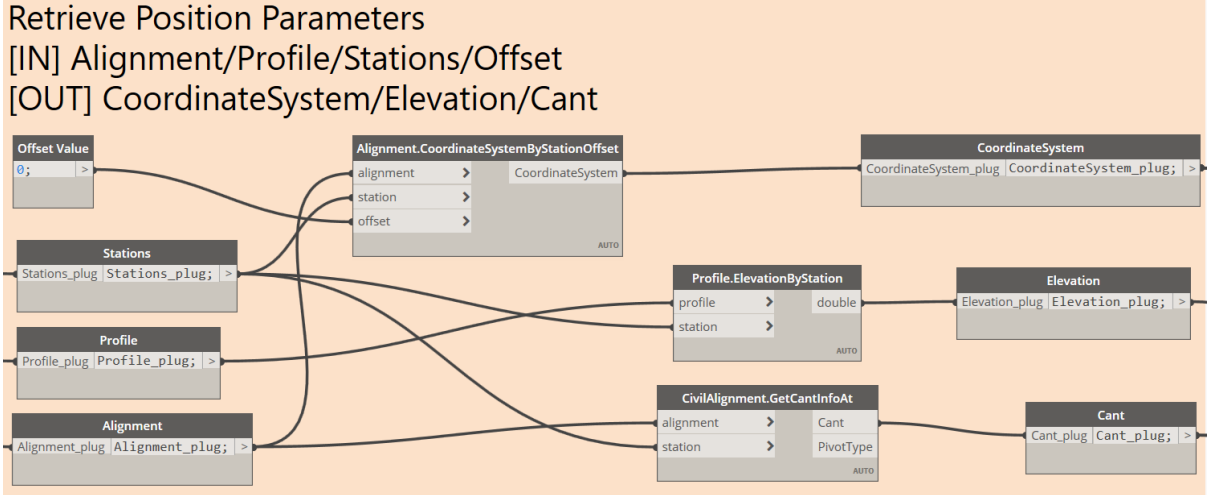


FIGURE 58: SPS - RETRIEVE POSITION PARAMETERS

Calculate Position and Direction:

Using the coordinate system, elevation and cant value a correct 3D point position for each station is composed. Keep in mind that on the center axis, half the cant value needs to be added to the elevation (rail design using the low rail principle).

In addition, the direction is calculated using the coordinate system. The angle is defined to the X-axis and rotating in the XY plane.

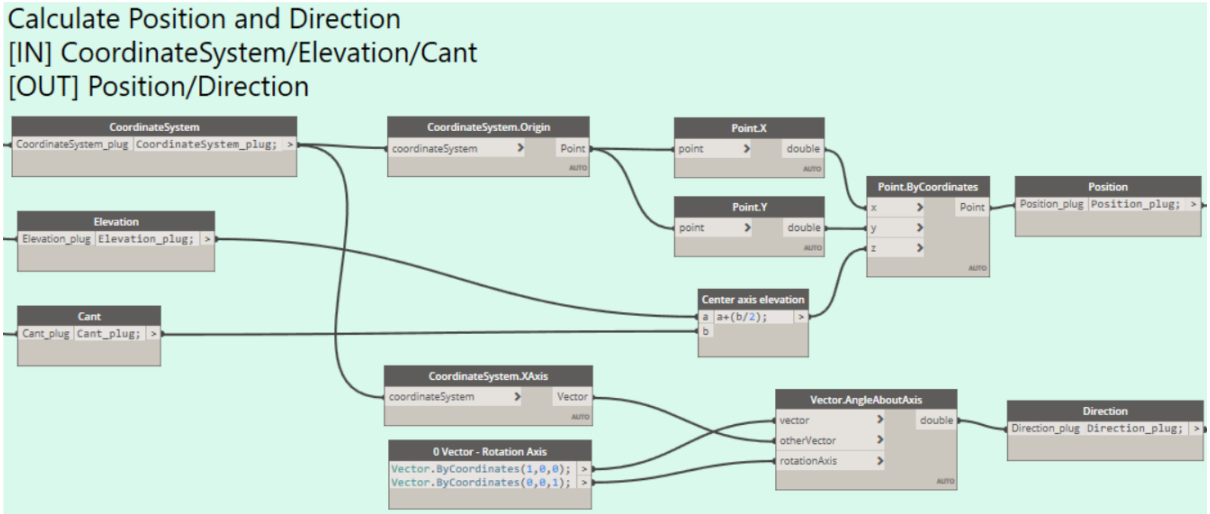


FIGURE 59: SPS - CALCULATE POSITION AND DIRECTION

Create Block Reference:

In this group, all previous calculations are combined to place a Block reference using the chosen Block definition. The calculated position point and direction are combined with a normal defined on the Z-axis and a scaling factor of one. This block is placed on layer “0”. At the end, the resulting block reference is converted into a custom block reference definition to interact with other custom nodes.

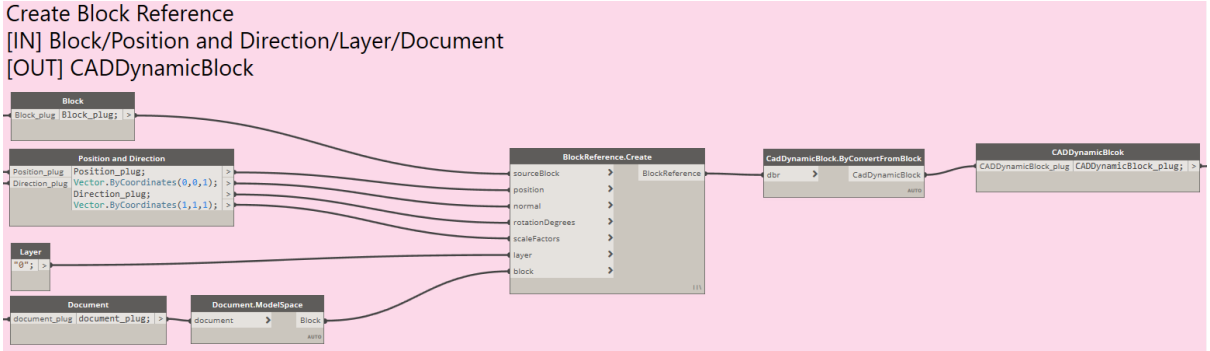


FIGURE 60: SPS - CREATE BLOCK REFERENCE

Cross slope calculation and rotation:

The created Block reference is rotated around its local Y-axis with the calculated cant angle. The cant angle is calculated using a user defined track gauge.

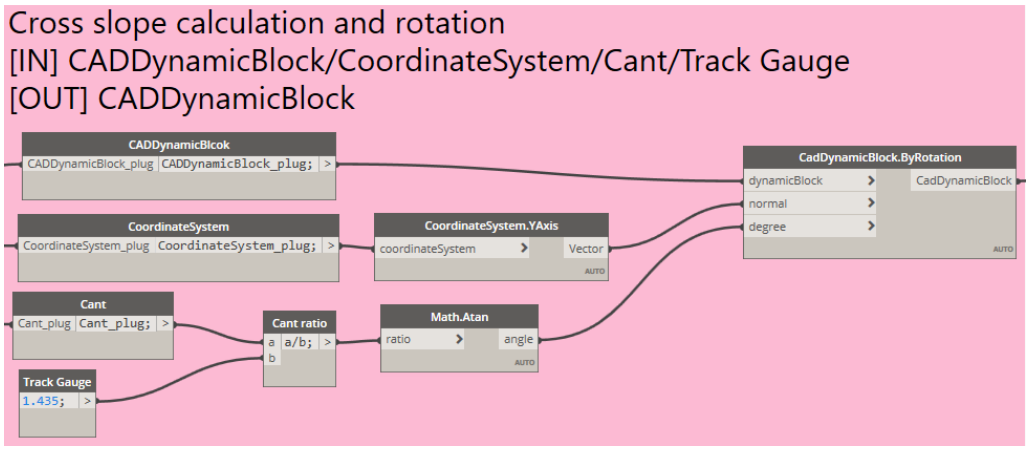


FIGURE 61: SPS - CROSS SLOPE CALCULATION AND ROTATION

Longitudinal slope calculation and rotation:

In the last step, the block reference is again rotated. This time around the local X axis using the profile grade value.

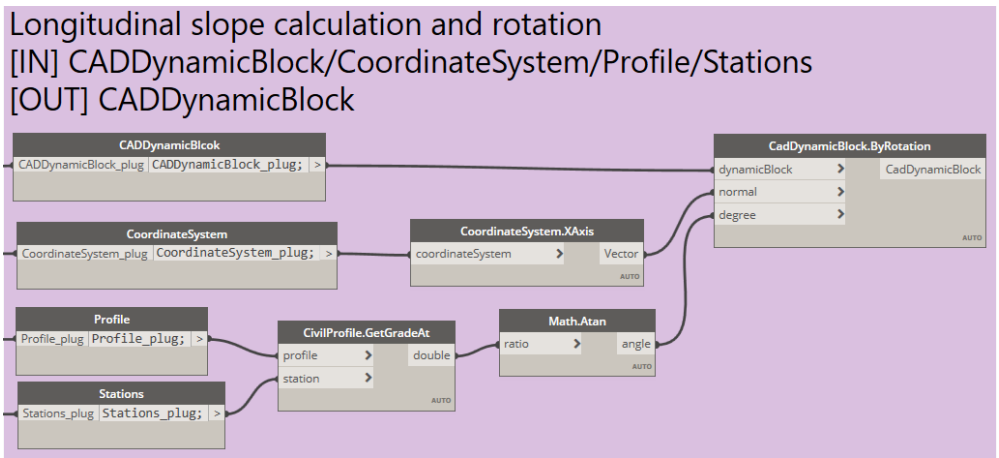


FIGURE 62: SPS - LONGITUDINAL SLOPE CALCULATION AND ROTATION

Result

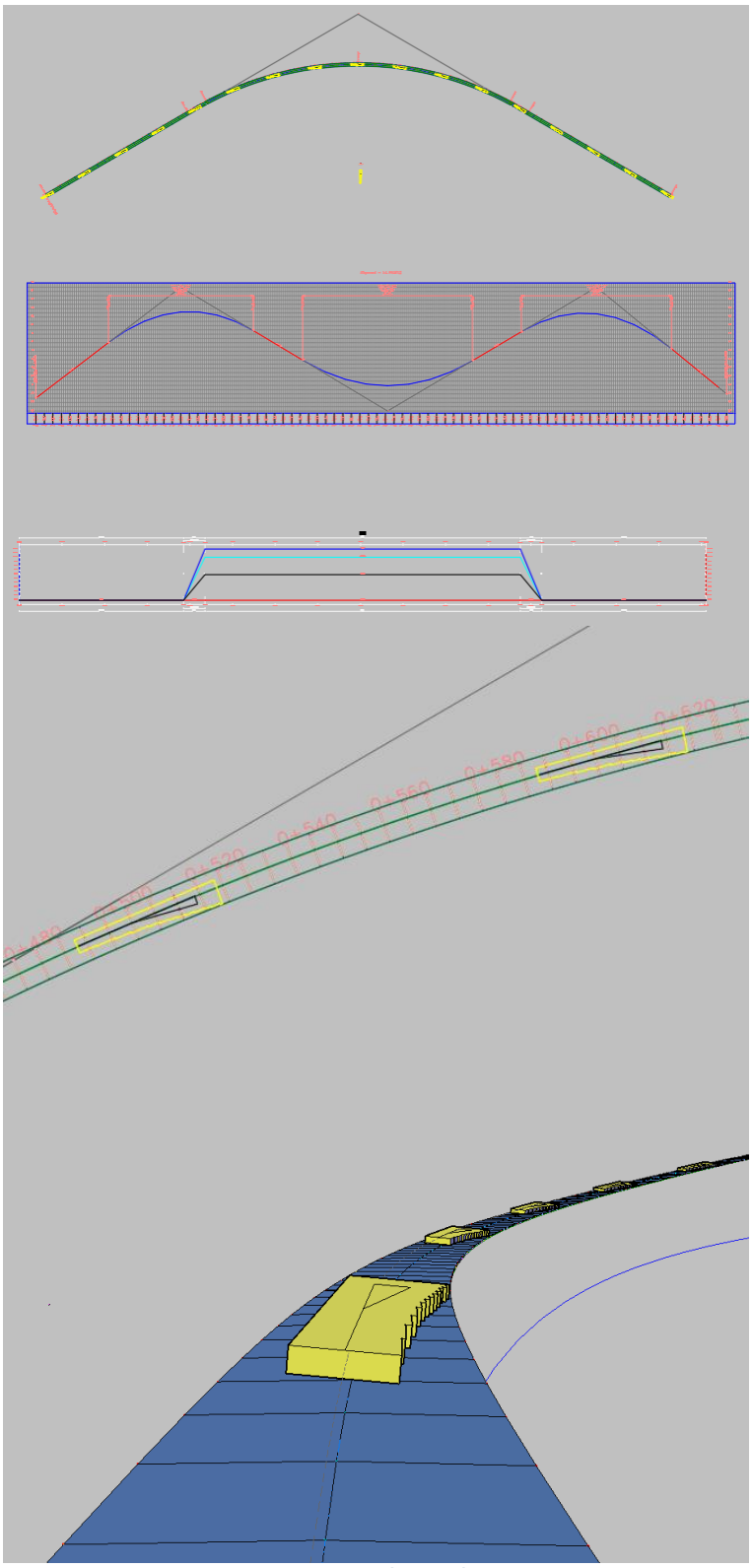


FIGURE 63: EXAMPLE SWITCH (ASSET) PLACEMENT SYSTEM

Switch - Profile

In today's workflow, there is no civil representation of the switch object. The dynamic block is correctly placed in 3D. However, when it comes to certain plan production (longitudinal profile), the switch is added manually. In communication with others using LandXML, there is no transfer of switch location information.

In the Switch - Profile Dynamo script an alternative workflow is created. The switch object (dynamic block) is used to create a profile on the alignment. Its relative placement is also determined and added into the description of the profile. The result is a profile for each switch on the alignment that can be used in Civil 3D for automated plan production. A profile is also part of the LandXML definition, so now a LandXML exports will not only contain alignment information but also switch location information (potentially very useful for alignment verification).

Workflow

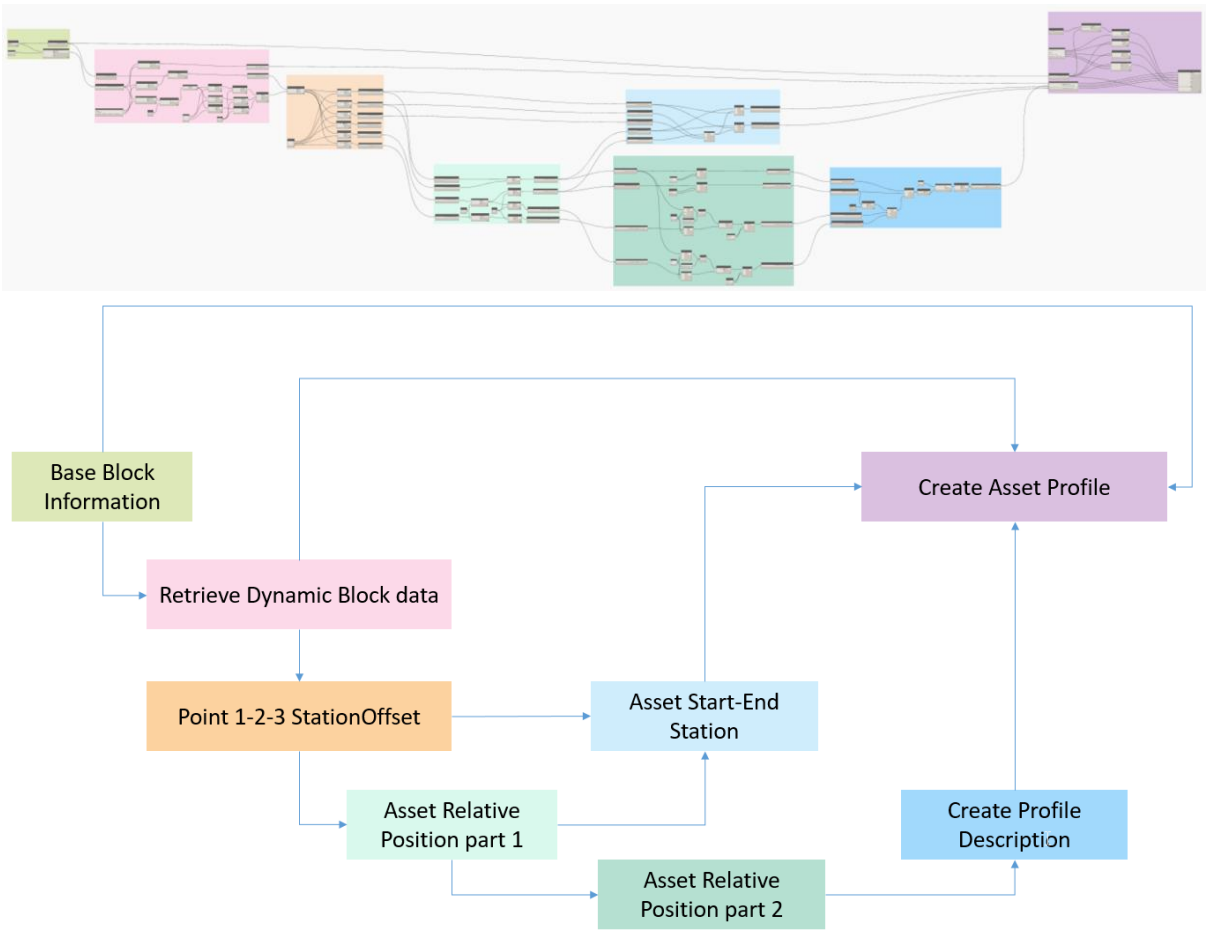


FIGURE 64: SWITCH - PROFILE

Base Block Information:

All dynamic blocks of a particular definition are retrieved from the current document. The document and block references are available for other groups.

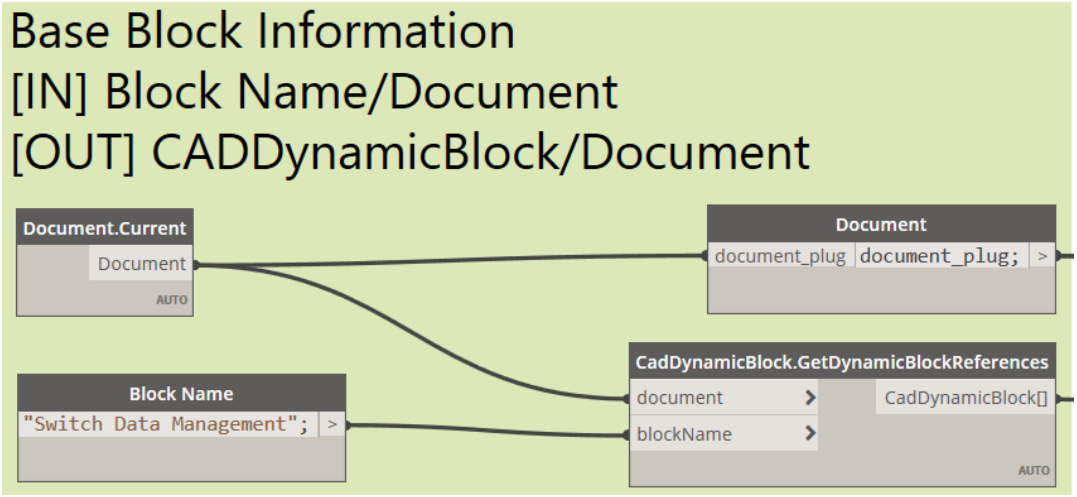


FIGURE 65: SP - BASE BLOCK INFORMATION

Retrieve Dynamic Block data:

In this group, specific attribute information is read from the retrieved block references and converted in preparation for other calculations. The “NAME” attribute will be used to name the new profile. The “ALIGNMENT” attribute is used in this group to retrieve the alignment object and the “POINT” attributes are converted to numbers and split to combine a list of points.

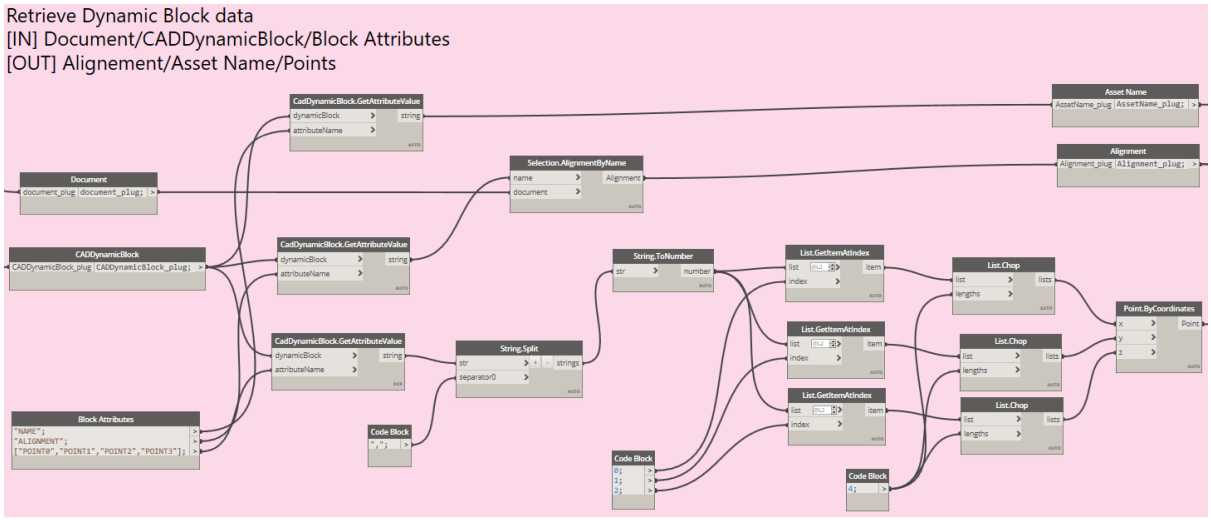


FIGURE 66: SP - RETRIEVE DYNAMIC BLOCK DATA

Point 1-2-3 StationOffset:

The alignment and point list are used to calculate the station and offset of the points 1, 2 and 3. These are the points that enable the evaluation of the relative position of the switch on the alignment.

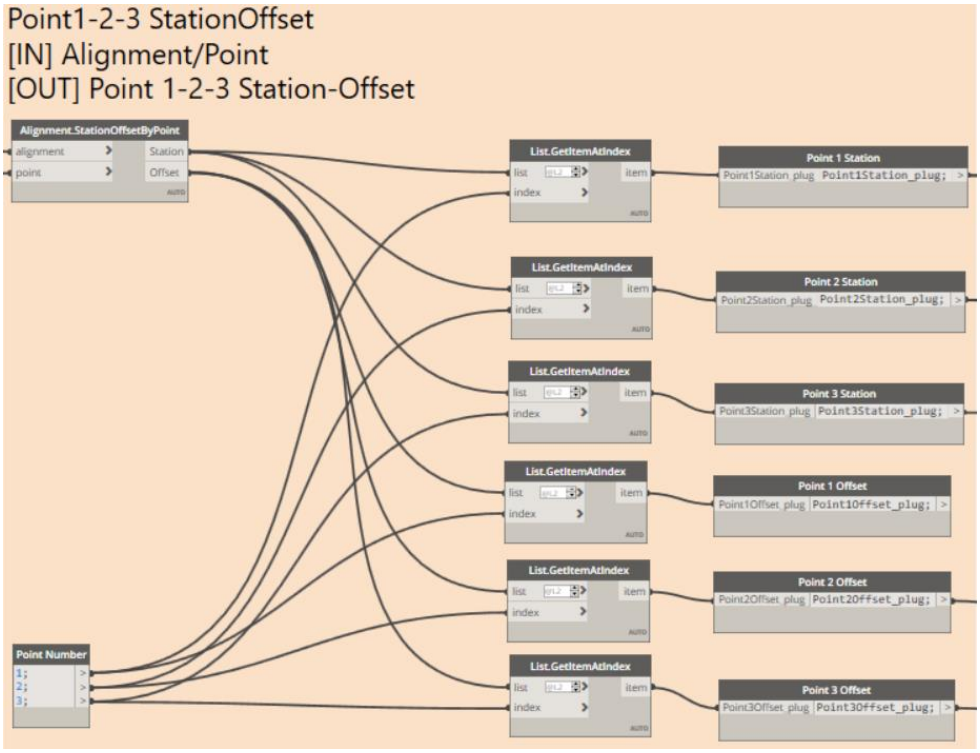


FIGURE 67: SP - POINT 1-2-3 STATIONOFFSET

Asset Relative Position part 1:

By comparing the station value of point 1 and 2, we are able to determine whether the switch is placed “Up” (station point 1 > station point 2) or “Down”. The offset of point 2 is used to indicate if the alignment runs through the main direction of the switch “Main” (offset point 2 = 0) or it’s “Branch”.

Next, the offset of point 2 and 3 is compared to zero to identify negative values. A negative value indicated that the point is left of the alignment (following increasing station direction).

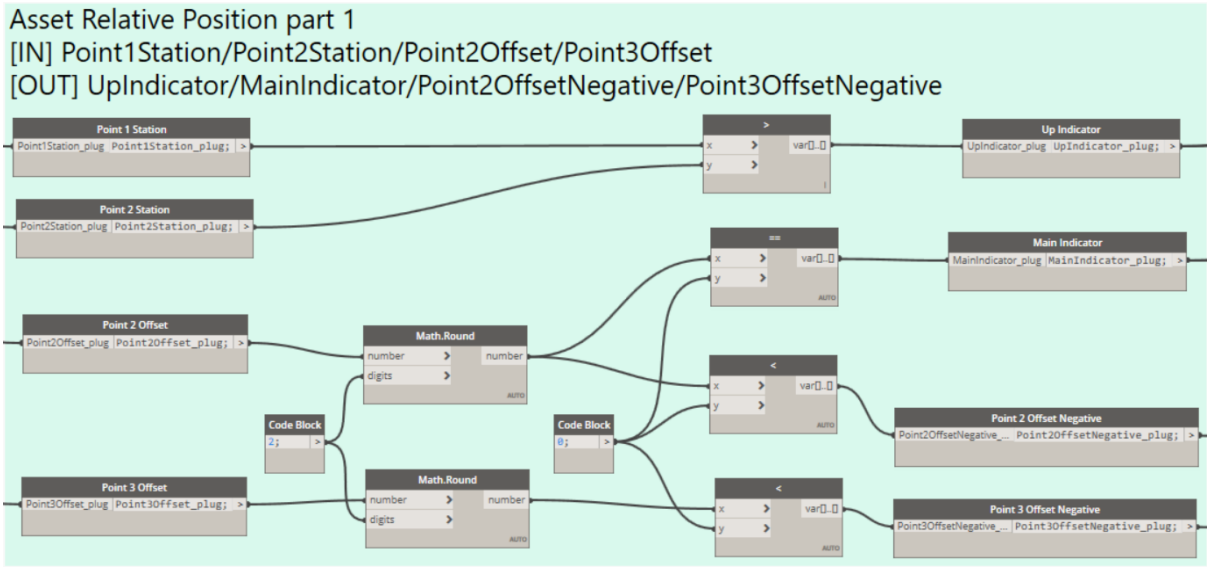


FIGURE 68: SP - ASSET RELATIVE POSITION PART 1

Asset Start-End Station:

Using the Up and Main indicator, the correct start and end station of the asset is determined.

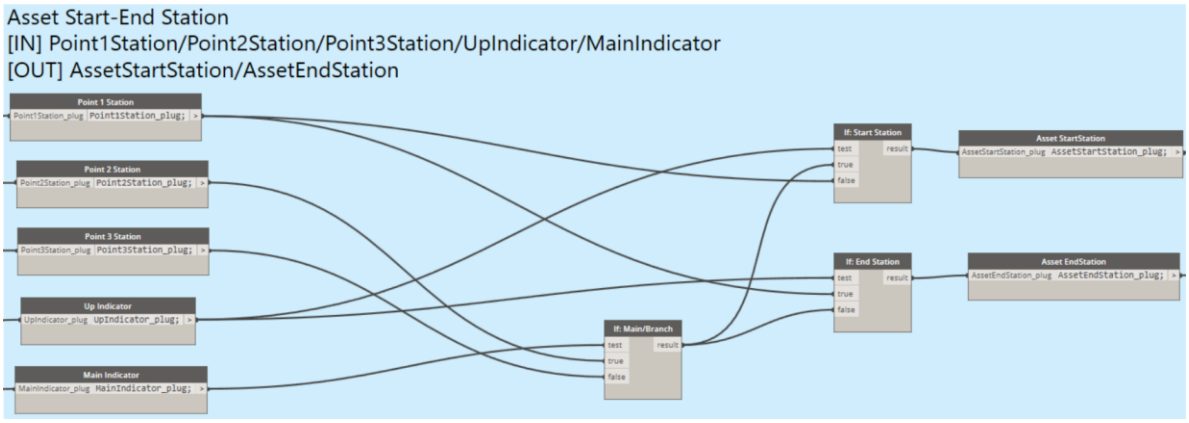


FIGURE 69: SP - ASSET START-END STATION

Asset Relative Position part 2:

The Up and Main indicator value are converted into string values, “Up” <> “Down” and “Main” <> “Branch”. Using the point 2 and 3 negative indicator a Main Right (Right/Left) and Branch Right (Right/Left) Value are calculated.

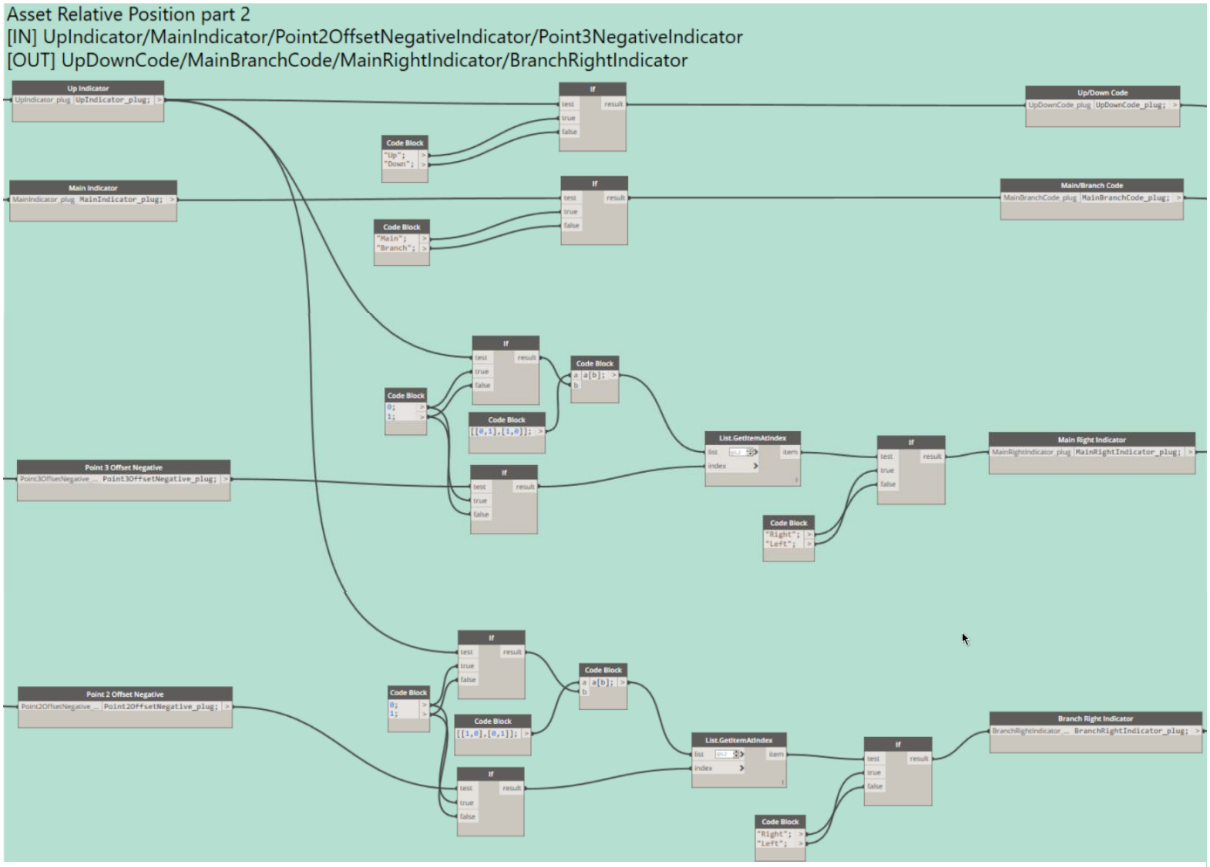


FIGURE 70: SP - ASSET RELATIVE POSITION PART 2

Create Profile Description:

In this group the finale relative position of the switch is determined and all string codes (“Up”<>”Down”, “Main”<>”Branch”, “Right”<>”Left”) are combined into one description string per switch.

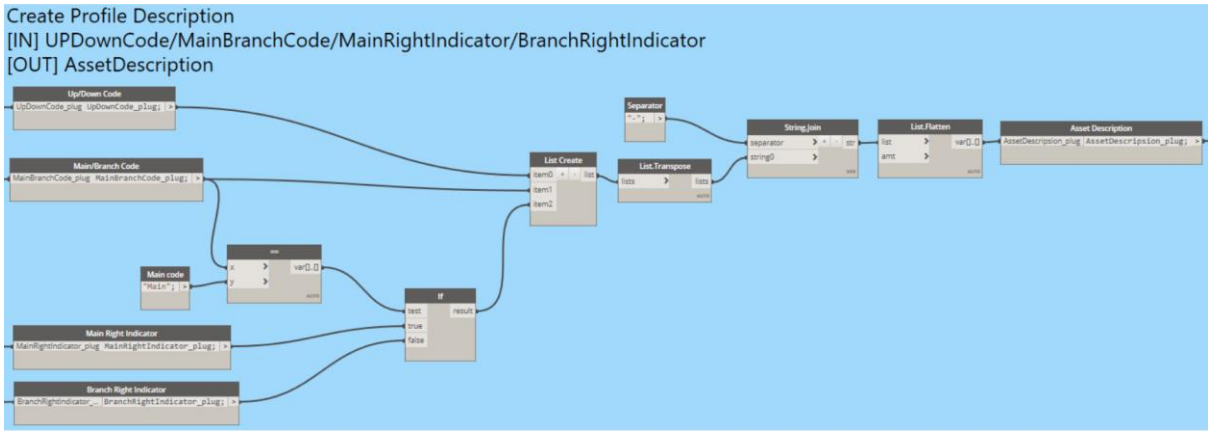


FIGURE 71: SP - CREATE PROFILE DESCRIPTION

Create Asset Profile:

Finally, all information is combined to create a profile that will represent the switch position on the alignment. The switch profile representation will follow a design profile chosen by the user. All style configurations can also be chosen by name. The asset name will be used as profile name, its relative position as description and its start- end station as the beginning and end of the profile.

FIGURE 72: SP - CREATE ASSET PROFILE

Result

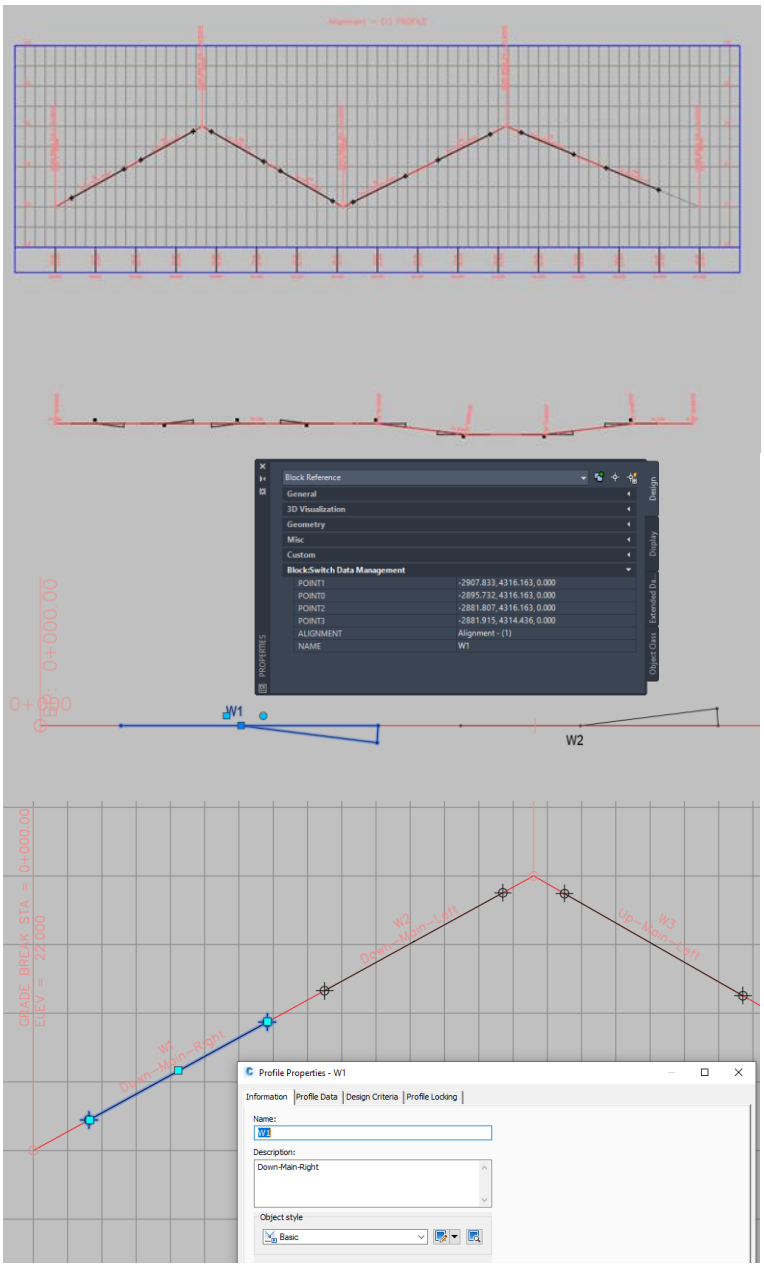


FIGURE 73:EXAMPLE SWITCH – PROFILE

Switch - Corridor

Many times, during design, a part of your section definition will be based on an asset placed on the track alignment. A railway switch is such an asset. For example, the deviation of the switch will change the platform construction. However, in the current design workflow, we are not able to use the switch as a direct input to create the corridor and its sub objects: Baseline and BaselineRegion.

The Switch Corridor Dynamo script shows how data from a switch object (dynamic block) can be used as input for the direct creation of a corridor, Baseline and BaselineRegion. The switch data is also used to choose which assembly needs to be applied.

Workflow

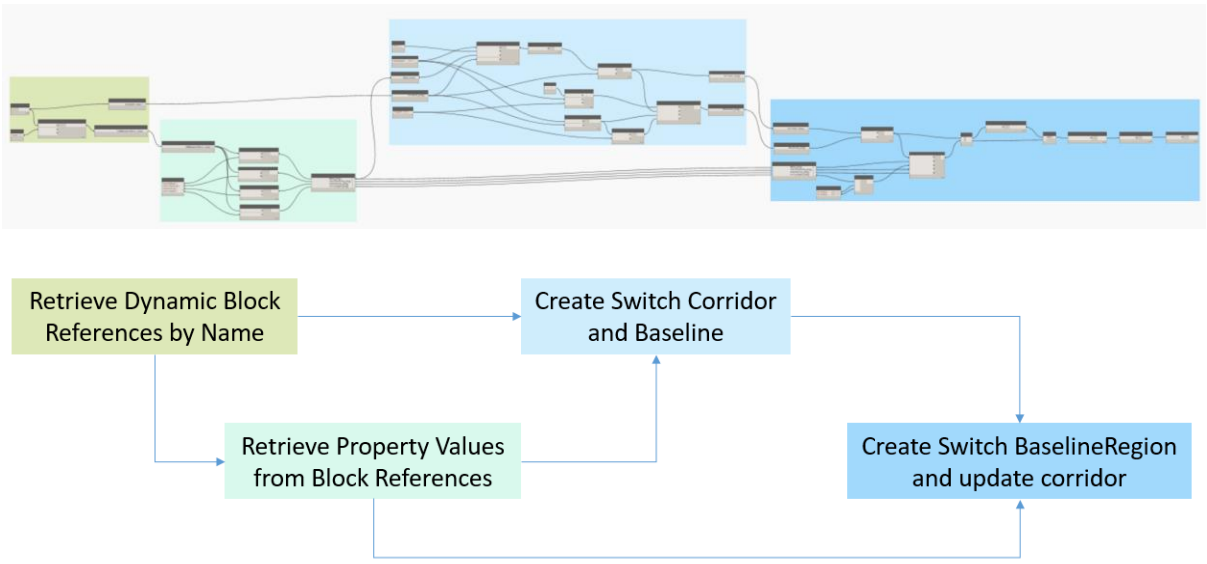


FIGURE 74: SWITCH – CORRIDOR

Retrieve Dynamic Block References by Name:

In this group dynamic Block references are retrieved from the current document by name. The document and block references are made available for other groups.

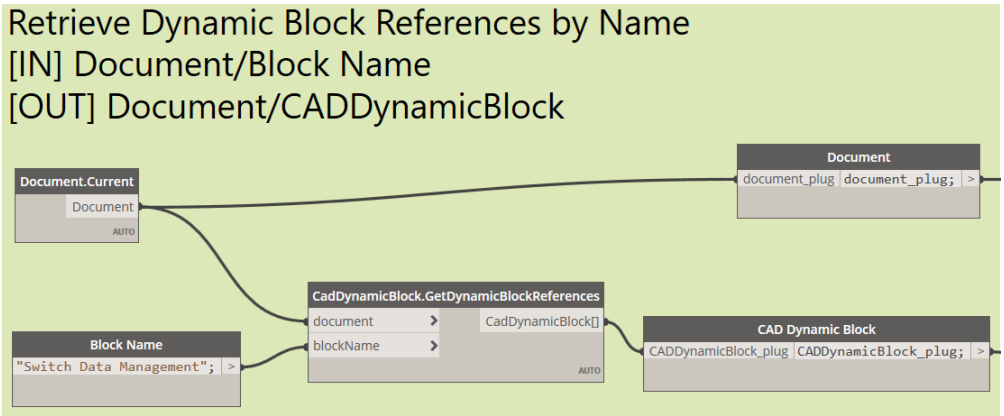


FIGURE 75: SC - RETRIEVE BLOCK REFERENCES BY NAME

Retrieve Property Values from Block References:

From a list of block references specific property values are retrieved (SwitchName, StartStation, EndStation, Left_Right) and converted into lists. These lists will be used as input for corridor, Baseline and BaselineRegion creation.

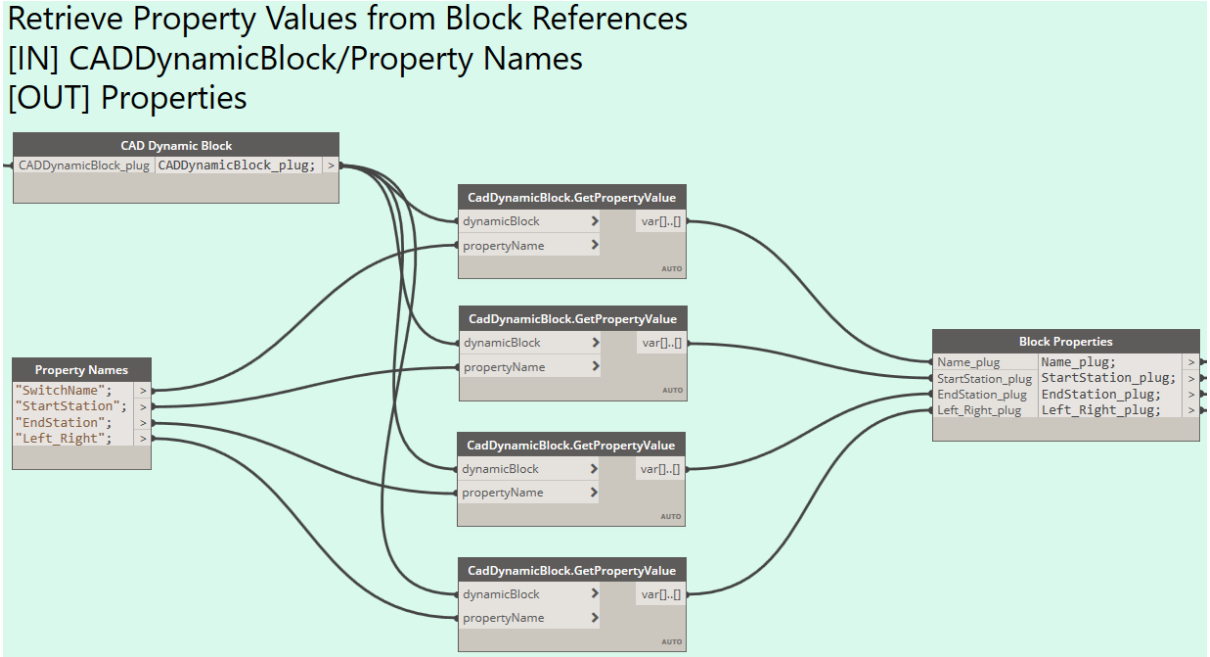


FIGURE 76: SC - RETRIEVE PROPERTY VALUES FROM BLOCK REFERENCES

Create Switch Corridor and Baseline:

In this group, a corridor is created using the Alignment and Profile name. For the corridor name, a prefix is combined with the alignment and profile name. The switch names are added to the description of the corridor. Next, a baseline is created in corridor with the alignment and profile as name.

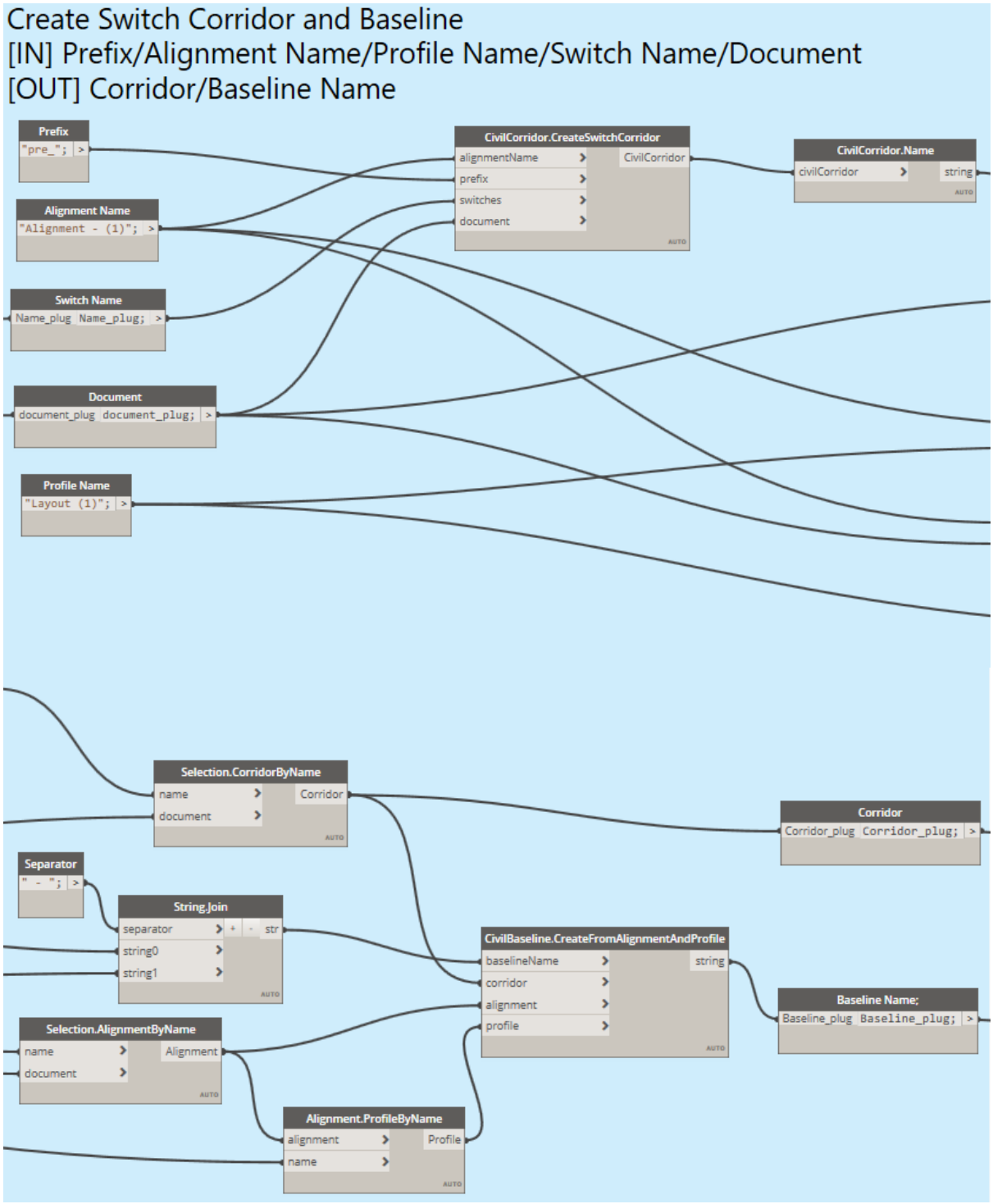


FIGURE 77: SC - CREATE SWITCH CORRIDOR AND BASELINE

Create Switch BaselineRegion and update corridor:

In this last group, the block properties are used to determine which assembly is to be used in the creation of a BaselineRegion per asset (switch). Next, the corridor is updated to calculate the new regions.

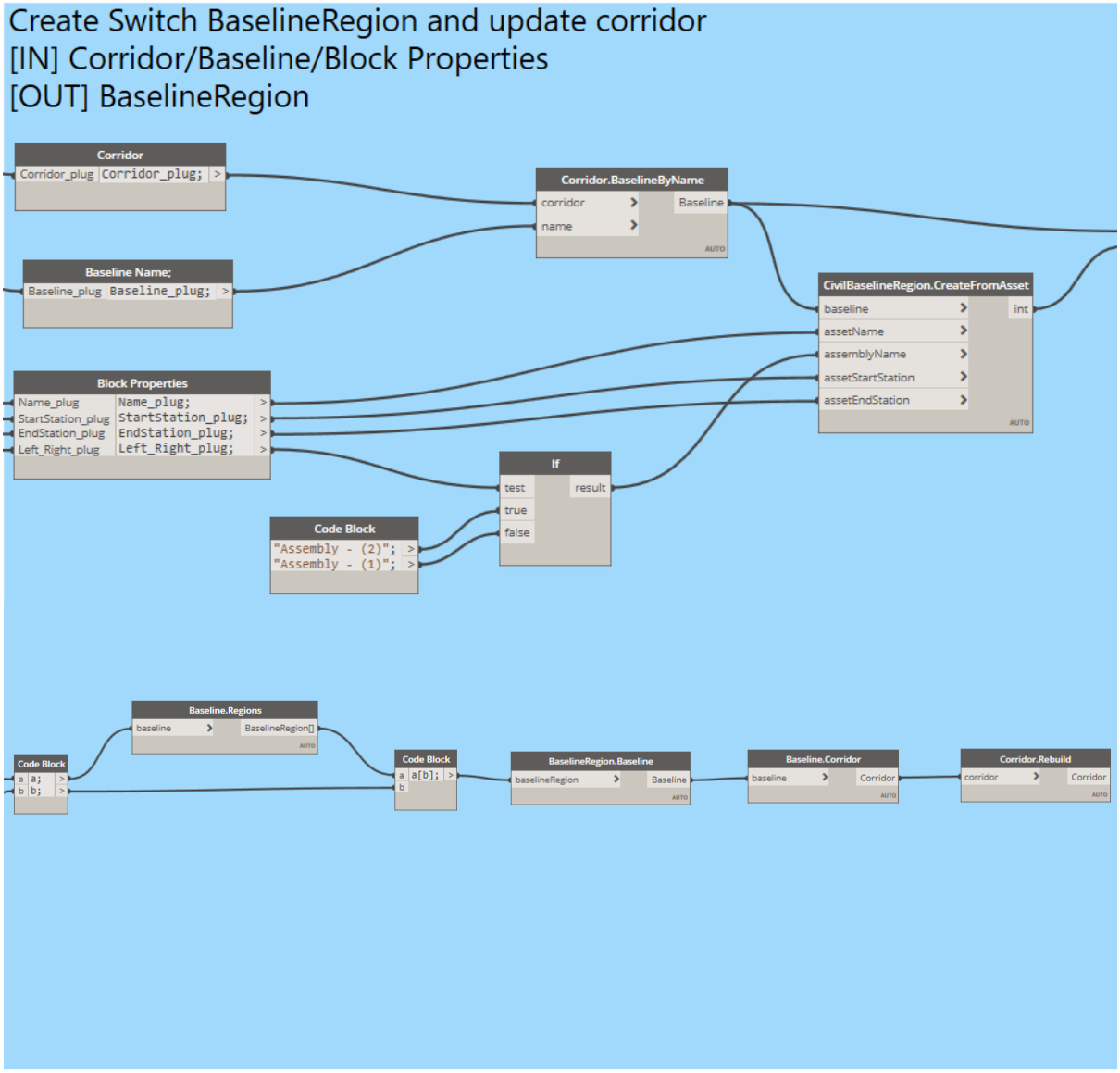


FIGURE 78: SC - CREATE SWITCH BASELINEREGION AND UPDATE CORRIDOR

Result

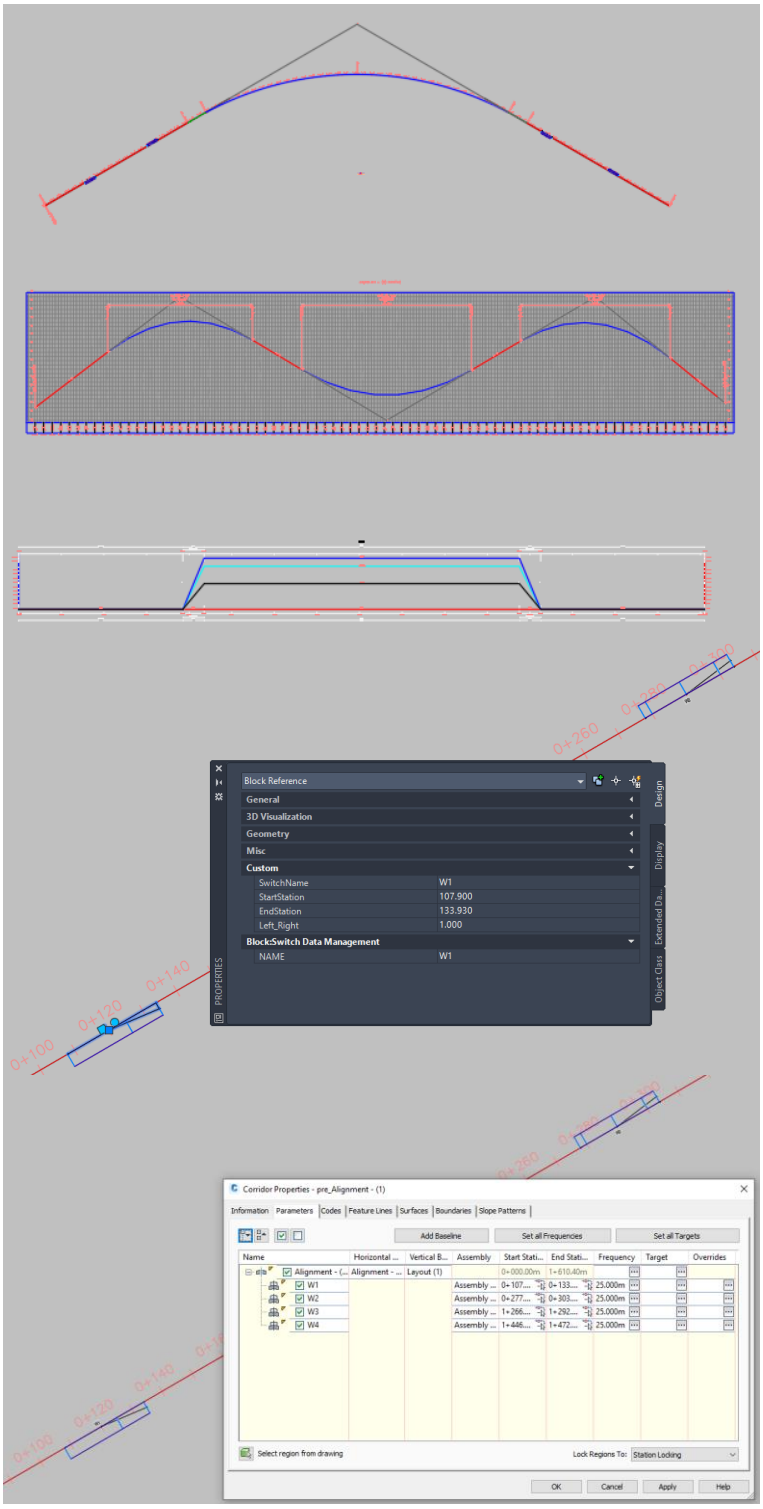


FIGURE 79: EXAMPLE SWITCH – CORRIDOR

Design Analysis

What is design analysis?

Analyzing a design can be described as taking an overall and/or a detailed (macro-micro) look at the design.

We do this because it is a part of the iterative design process: model our first idea, take a closer look, adjusting something, taking a step back to see the impact on the overall design and repeat this until we are satisfied with the design. An analysis is also a means of communication between people to facilitate a better understanding of the design.
Last but not least, it is needed to build trust in the accuracy of the design. Because models are so big and many parts are created using automated tools, the designer needs feedback to be sure that his vision is correctly translated into the model.

Examples:

Design analysis today usually means creating reports with long lists of numerical values. These values are imported into Excel so we can use the conditional formatting to identify the locations where a value does not comply with our requirements. Finally, we return to our design environment to find the listed locations and make adjustments.

Another example is the creation of a corridor using intelligent parametrical subassemblies. To then, create sample lines and sections. Plot these sections on paper and verify the corridor using a scaling ruler on paper, make notes on paper. Return to Civil 3D and correct the corridor where needed.

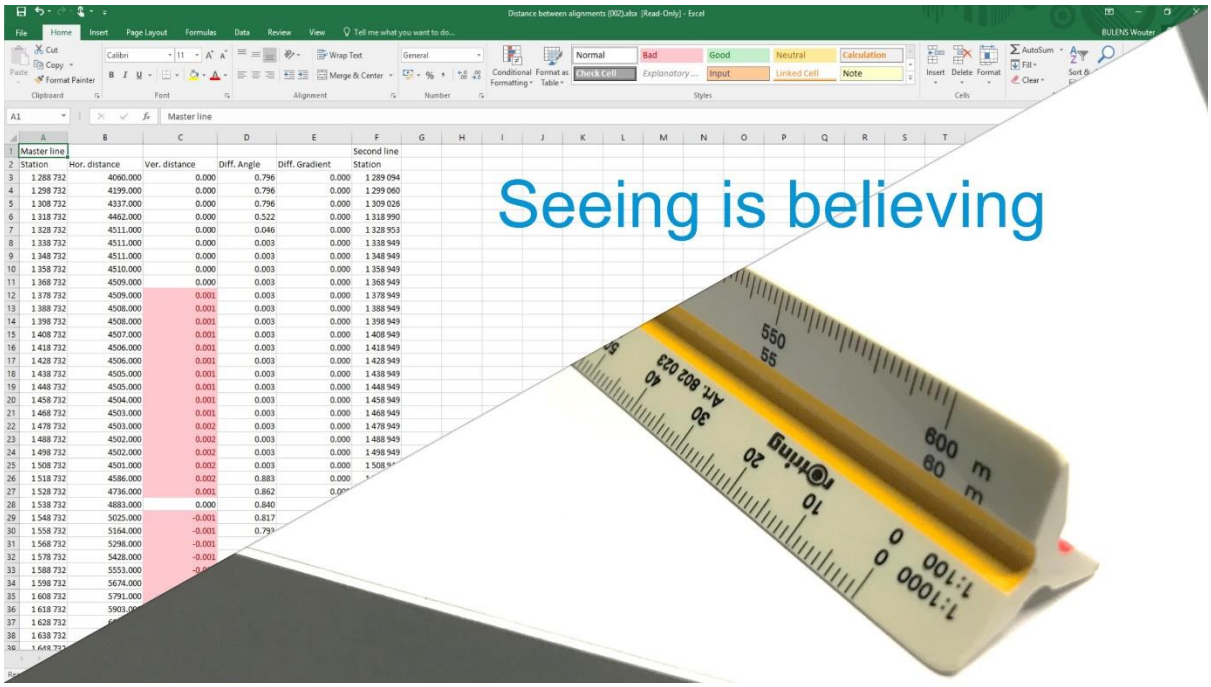


FIGURE 80: SEEING IS BELIEVING

Why do we need advanced or automated analysis?

Because today's creation tools allow us to make so much data. It is simply not efficient to use automated design tools in combination with manual analysis tools. Certainly because analysis should be a part of the design process.

To make a direct connection between the data and design decisions. So that we can better evaluate the impact of our design decisions and study more variants to improve the design.

To gain new insight in our design but also in the design process. The amount of analysis results can lead to entirely different conclusions and viewpoints that can change the way we design.

Automated design analysis is the stepping-stone to true analytics and generative design.

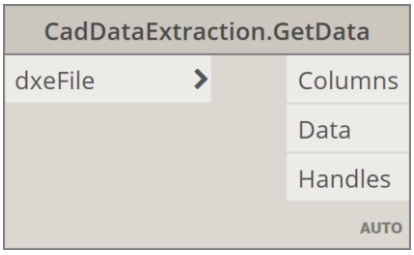
FIGURE 81: MANUAL ANALYSIS

To illustrate the power of automated analysis, three Dynamo scripts have been developed to bring existing analysis tools to a new level and make it possible to interact in a new way with existing design objects:

- Dynamo Data Extraction.
- Corridor Parameter Analyzer.
- Corridor Automated Ruler.

Custom Dynamo Nodes

CadDataExtraction.GetData

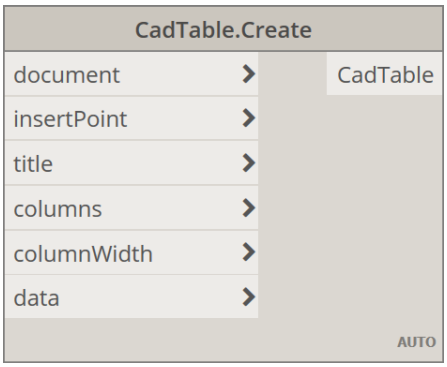


Extending the existing Data Extraction in AutoCAD, this node retrieves the extraction result (before processing) using a DXE file.

Input:
- dxeFile: file path of an DXE file (string).

Output:
- Columns: configured object property names (string).
- Data: configured object data result (string).
- Handles: object handles of the found objects (string).

CadTable.Create

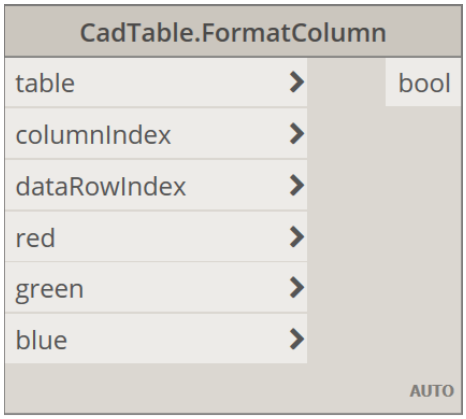


With this custom node the user is able to create an AutoCAD Table.

Input:
- document: document in which to place the table (document).
- insertPoint: geometry point that represents the place of insertion (point).
- title: text that will be placed in the title row of the table (string).
- columns: list of column titles (string).
- columnWidth: number to set the column width (integer).
- data: list of row values (string).

Output:
- CadTable: AutoCAD Table, Autodesk.AutoCAD.DatabaseServices.Table.

CadTable.FormatColumn

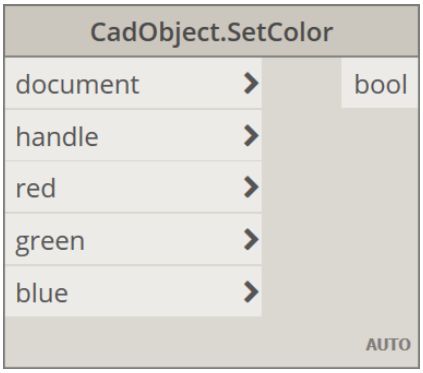


Based on the principles of conditional formatting in Excel, this node changes the color of specific rows in one column. The color is controlled by its RGB values.

Input:
- table: table object to format (CADTable).
- columnIndex: index of the column to edit (integer).
- dataRowIndex: index of the rows to edit (integer).
- red: red indicator value 0-255 (integer).
- green: green indicator value 0-255 (integer).
- blue: blue indicator value 0-255 (integer).

Output:
- bool: process indicator (Boolean).

CadObject.SetColor



This node enables the coloring of AutoCAD Objects. It is created to extend the conditional formatting possibilities in AutoCAD beyond a Table.

Input:

- document: document in which to place the table (document).
- handle: AutoCAD object handle (string).
- red: red indicator value 0-255 (integer).
- green: green indicator value 0-255 (integer).
- blue: blue indicator value 0-255 (integer).

Output:

- bool: process indicator (Boolean).

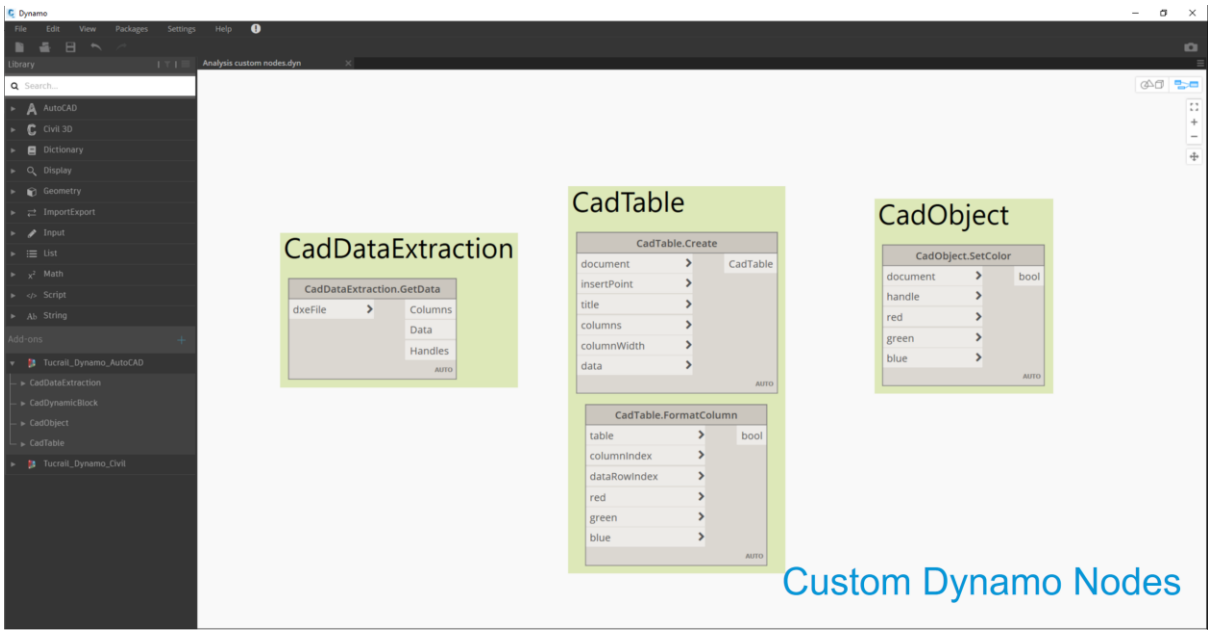


FIGURE 82: DESIGN ANALYSIS - CUSTOM NODES

Dynamo Data Extraction

AutoCAD already has a great analysis tool that can “query” the objects in a DWG file, Data Extraction:

“The ability to extract data from objects in one or more drawings. It searches for the objects you want, looks up the required attributes, links to an external file to add additional data, makes a table with a flexible format and updates”

This function has three major drawbacks:

Firstly, the ability to add additional data is limited to Excel (Excel Data Link, .XLSX no macro). Other data sources must first be converted to Excel, which is not a preferable data workflow (risk of desynchronization).

Secondly, the standard tool has data refinement possibilities, but these can be insufficient. To perform complex refinements, the user must first go to Excel and then re-import using a Data Link.

And, data extraction only gives limited output types and one data output at a time. Here the solution is also to go to Excel and distribute multiple outputs from there.

This script resolves these limitations by exposing the Data Extraction to Dynamo. It creates a table, and then performs data refinement and conditional formatting on the table and the source objects.

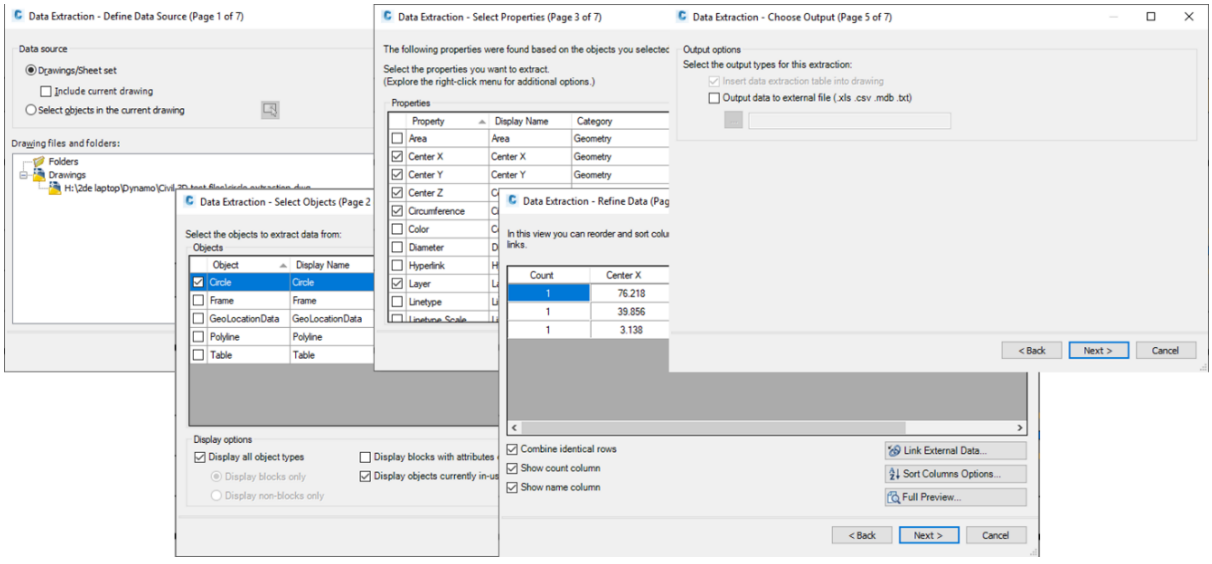


FIGURE 83: AUTOCAD DATA EXTRACTION

Workflow

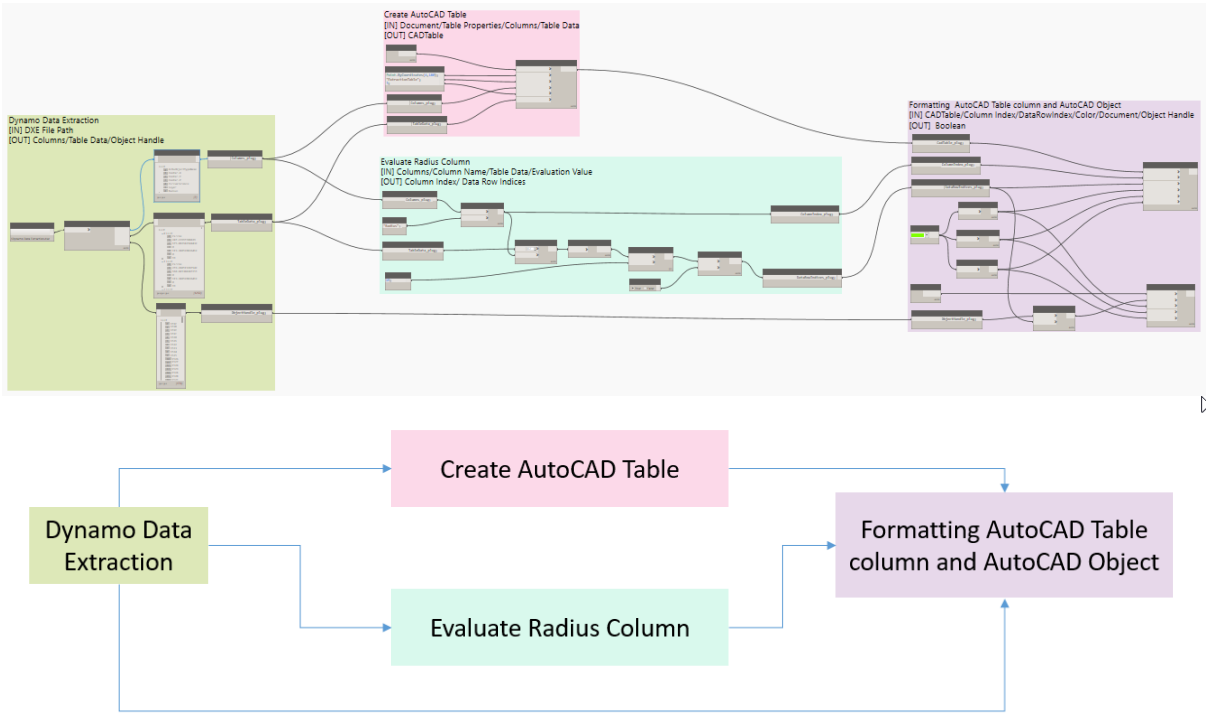


FIGURE 84: DYNAMO DATA EXTRACTION

Dynamo Data Extraction:

To start, the DXE file is selected and used to perform a data extraction. The resulting columns, data and object handles are available as output.

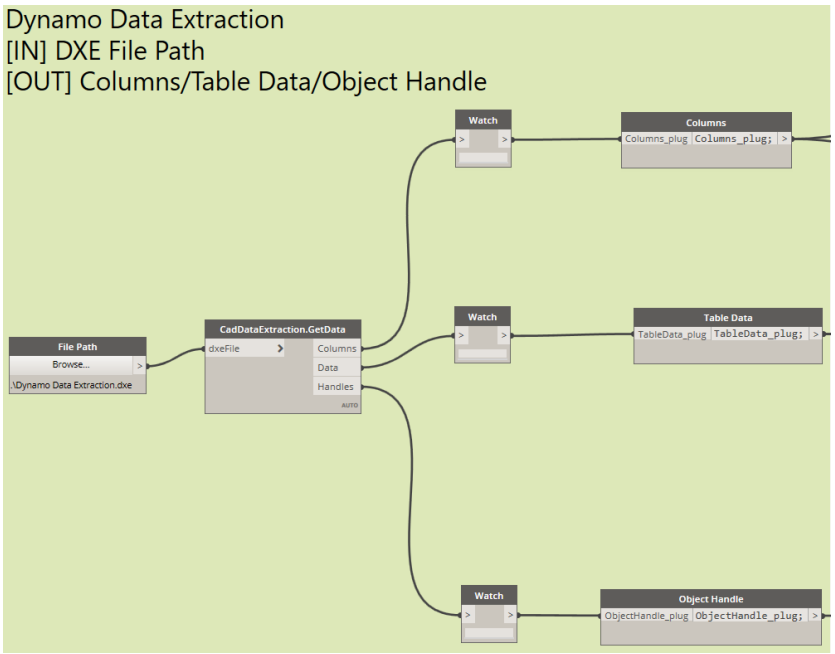


FIGURE 85: DDE - DYNAMO DATA EXTRACTION

Create AutoCAD Table:

In this group, an AutoCAD Table is made in the current document. The table's location and style settings are defined separately; the column titles are taken from the data extraction results. The data input consists out of a list of rows and each row consists of out of a list of strings (size corresponding with the number of columns).

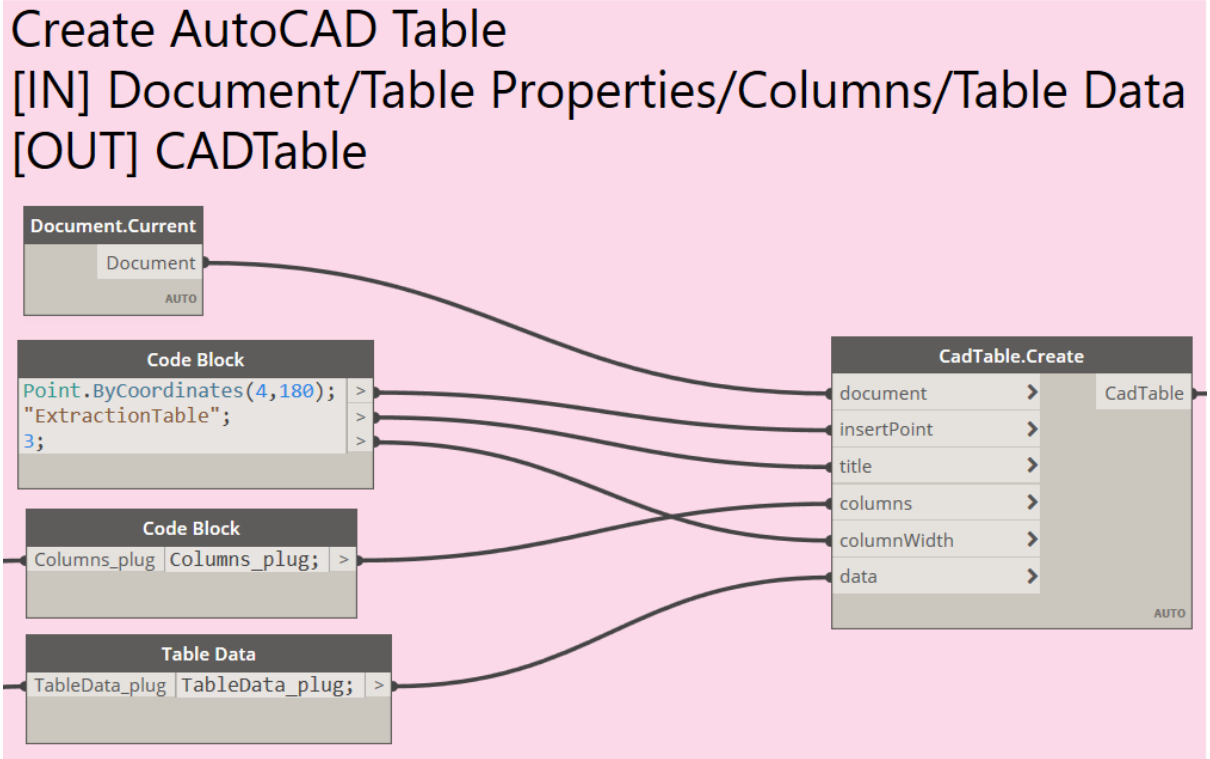


FIGURE 86: DDE - CREATE AUTOCAD TABLE

Evaluate Radius Column:

From the columns, result list the index of the "Radius" column is identified and all column values are converted to numbers. These numbers are then compared to an evaluation value. The column index and data row indices of positive result are available for other groups.

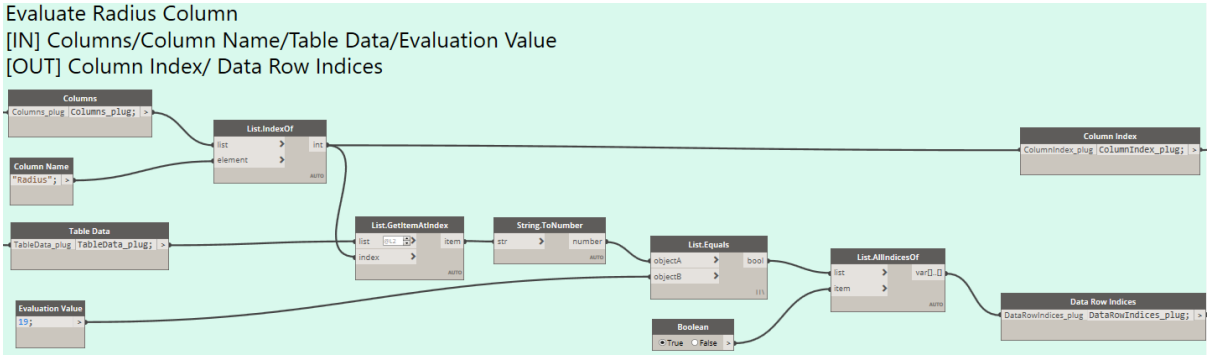


FIGURE 87: DDE - EVALUATE RADIUS COLUMN

Formatting AutoCAD Table column and AutoCAD Object:

This group contains the color formatting of the AutoCAD objects and the created table. The column index and data row indices are used to select the right cells in the table to change color. The data row indices are also used to select the right object from the object handle list. For the color selection, a Color Palette node is used.

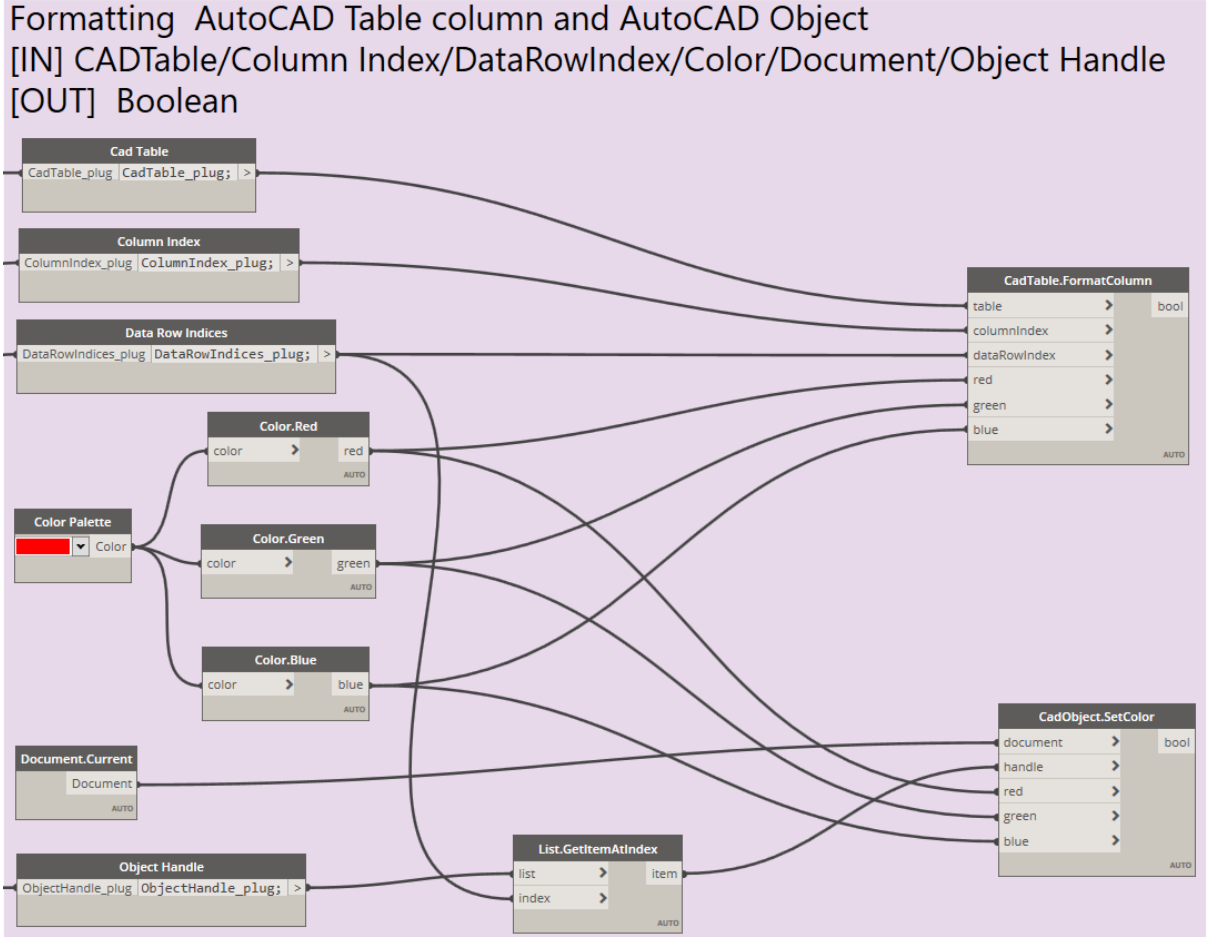
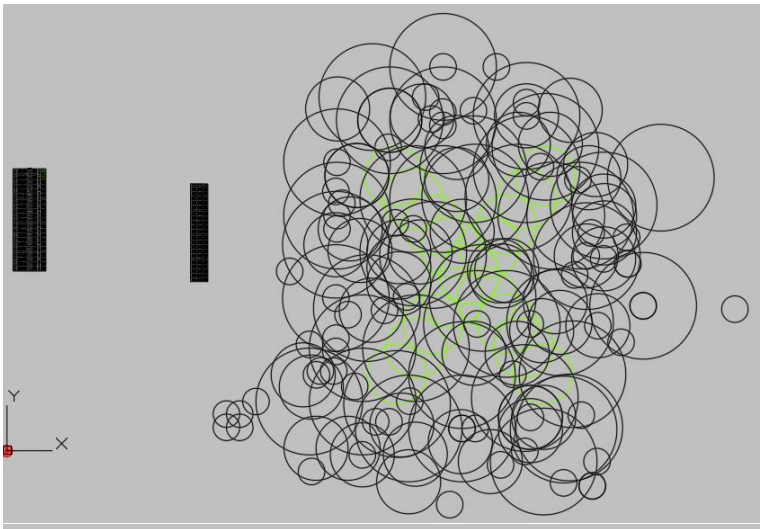


FIGURE 88: DDE - FORMATTING AUTOCAD TABLE COLUMN AND AUTOCAD OBJECT

Result



Count	Center X	Center Y	Center Z	Circumference	Name	Layer	test	Radius
1	362.195	112.336	0.000	53.149	Circle	0	26.574	8.459
1	376.998	79.947	0.000	53.149	Circle	0	26.574	8.459
1	299.910	81.080	0.000	53.149	Circle	0	26.574	8.459
1	140.149	15.160	0.000	53.149	Circle	0	26.574	8.459
1	140.149	23.619	0.000	53.149	Circle	0	26.574	8.459
1	270.915	211.723	0.000	53.149	Circle	0	26.574	8.459
1	297.674	216.960	0.000	53.149	Circle	0	26.574	8.459
1	395.326	136.275	0.000	53.149	Circle	0	26.574	8.459
1	333.999	21.507	0.000	53.149	Circle	0	26.574	8.459
1	376.465	-6.657	0.000	53.149	Circle	0	26.574	8.459
1	298.812	14.437	0.000	53.149	Circle	0	26.574	8.459
1	250.115	117.264	0.000	128.295	Circle	0	64.148	20.419
1	219.099	141.665	0.000	128.295	Circle	0	64.148	20.419
1	232.492	89.805	0.000	128.295	Circle	0	64.148	20.419

ExtractionTable

AcDbObjectTypeName	Center.X	Center.Y	Center.Z	Circumference	Layer	Radius
Circle	247.23995508839	175.002942944432	0	119.380520836412	0	19
Circle	259.280519287142	160.685388385555	0	119.380520836412	0	19
Circle	271.894445043312	144.077024344589	0	119.380520836412	0	19
Circle	282.788288718777	129.759468844875	0	119.380520836412	0	19
Circle	295.402214474947	112.578403374223	0	119.380520836412	0	19
Circle	306.296056815041 1	97.6881464448239	0	119.380520836412	0	19
Circle	318.909983906581	81.0797824038574	0	119.380520836412	0	19
Circle	329.803827582045	65.6168231039358	0	119.380520836412	0	19
Circle	341.844394372345	49.0084590629692	0	119.380520836412	0	19
Circle	343.564473861503	175.002942944432	0	119.380520836412	0	19
Circle	331.523909662751	160.685388385555	0	119.380520836412	0	19
Circle	318.909983906581	144.077024344589	0	119.380520836412	0	19
Circle	308.016140231116	129.759468844875	0	119.380520836412	0	19
Circle	295.402214474947	112.578403374223	0	119.380520836412	0	19
Circle	284.508370799482	97.6881464448239	0	119.380520836412	0	19
Circle	271.894445043312	81.0797824038574	0	119.380520836412	0	19
Circle	261.000601367848	65.6168231039358	0	119.380520836412	0	19
Circle	248.960034577548	49.0084590629692	0	119.380520836412	0	19
Circle	259.280519287142	160.685388385555	0	213.671605602713	0	34.0068922300536
Circle	299.909983906581	81.0797824038574	0	213.671605602713	0	34.0068922300536
Circle	261.000601367848	65.6168231039358	0	213.671605602713	0	34.0068922300536
Circle	229.672632569763	91.2126993523061	0	213.671605602713	0	34.0068922300536
Circle	259.280519287142	126.678496155502	0	213.671605602713	0	34.0068922300536
Circle	271.894445043312	125.077024344589	0	213.671605602713	0	34.0068922300536
Circle	303.508370799482	97.6881464448239	0	213.671605602713	0	34.0068922300536
Circle	318.909983906581	81.0797824038574	0	213.671605602713	0	34.0068922300536
Circle	360.844394372345	49.0084590629692	0	213.671605602713	0	34.0068922300536
Circle	369.949188723698	124.395309962667	0	213.671605602713	0	34.0068922300536
Circle	347.317713043756	150.123391663243	0	213.671605602713	0	34.0068922300536
Circle	331.523909662751	179.685388385555	0	213.671605602713	0	34.0068922300536
Circle	298.753511909883	161.622490933893	0	213.671605602713	0	34.0068922300536
Circle	278.280519287142	160.685388385555	0	213.671605602713	0	34.0068922300536

FIGURE 89: EXAMPLE DYNAMO DATA EXTRACTION

Corridor Parameter Analyzer

Based on the Civil3D_ReadAndWriteSubassemblyProperties script that comes by default with the Civil 3D Dynamo installation, this script retrieves the AppliedSubassemblies and a specific parameter by name. The result parameters are then combined and published in an AutoCAD Table.

These specific parameters are already available using the section editor but can only be reviewed section by section. With this script, the entire corridor is evaluated in one run of the script.

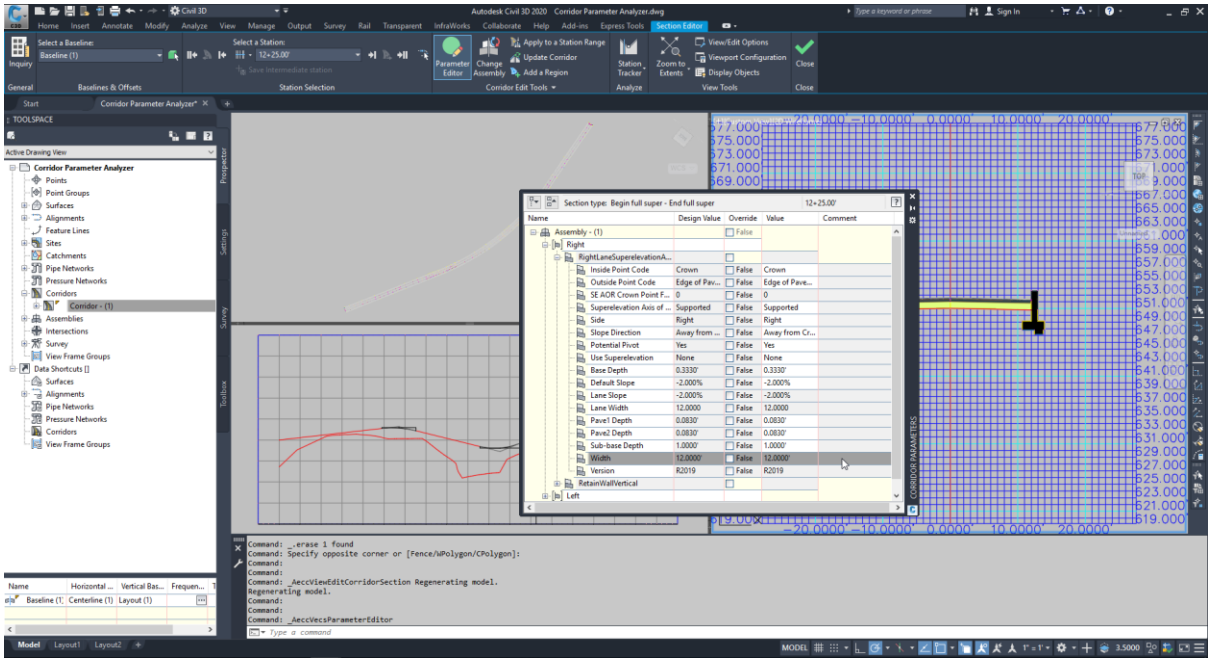


FIGURE 90: CIVIL 3D SECTION EDITOR – PARAMETER EDITOR

Workflow

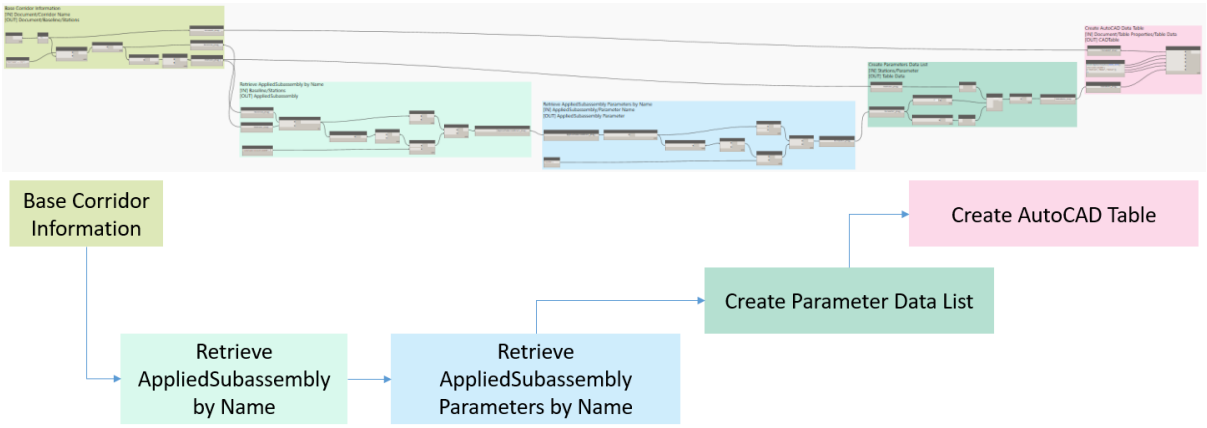


FIGURE 91: CORRIDOR PARAMETER ANALYZER

Base Corridor Information:

To start the corridor is retrieved by name from the current document. Next the corridor baseline and its stations are retrieved. The current document, the baseline and a flattened list of the stations is made available for other groups.

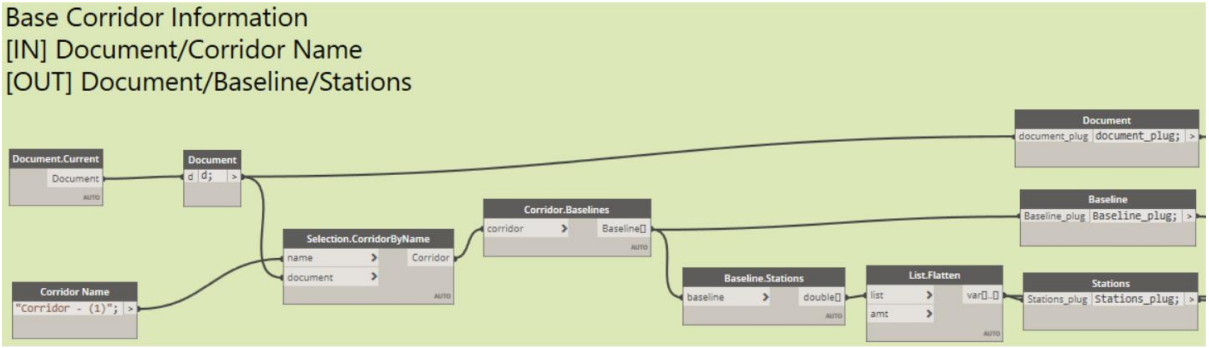


FIGURE 92: CPA - BASE CORRIDOR INFORMATION

Retrieve AppliedSubAssembly by Name:

Using the baseline and stations all AppliedSubassemblies are retrieved. Next, a list of the Subassemblies names is made and used to find the indices of a specific name. These indices are then used to select which subassemblies are to be used in the next group.

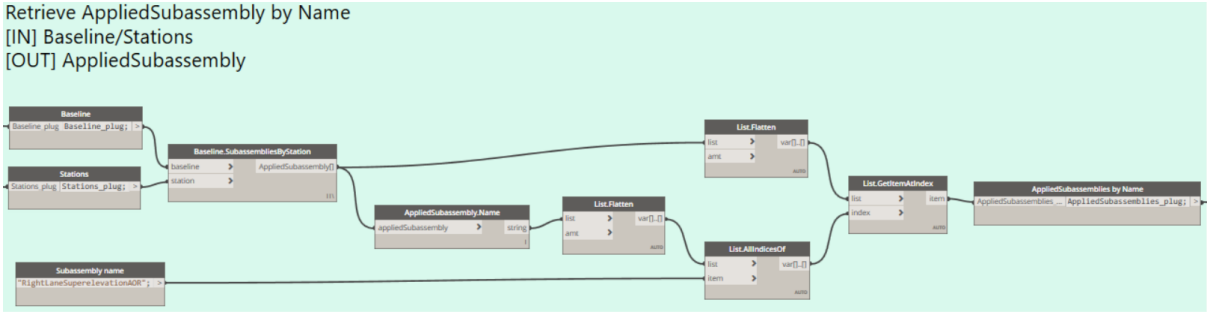


FIGURE 93: CPA - RETRIEVE APPLIEDSUBASSEMBLY BY NAME

Retrieve AppliedSubAssembly Parameters by Name:

With all the AppliedSubassemblies selected by name, we can retrieve all parameters. Next, a list of the parameter names is made and used to find the indices of a specific name. These indices are then used to select the desired parameters.

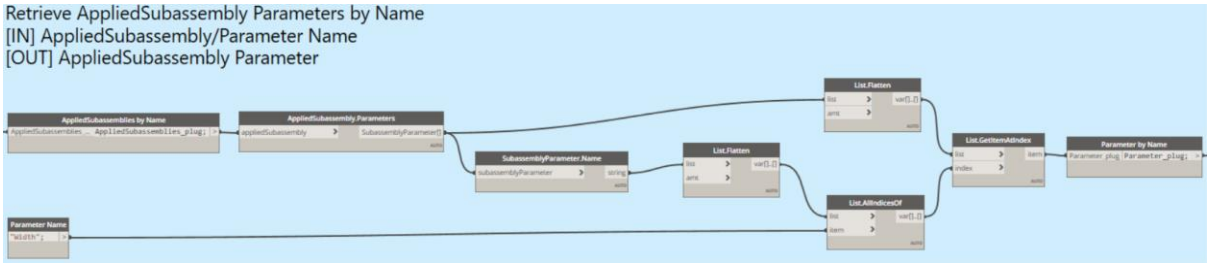


FIGURE 94: CPA - RETRIEVE APPLIEDSUBASSEMBLY PARAMETERS BY NAME

Create Parameter Data List:

The parameters name and value are combined with their corresponding station in one list and converted to text. This list is then transposed to define table rows.

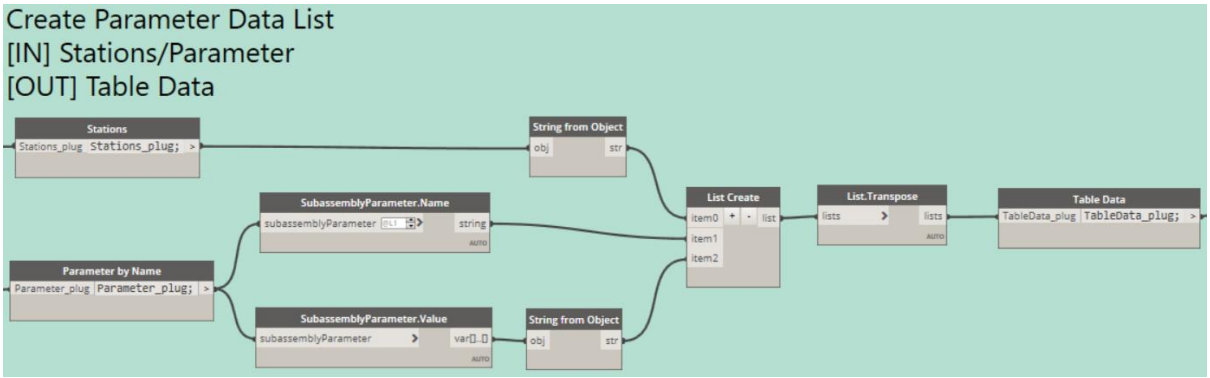


FIGURE 95: CPA - CREATE PARAMETER DATA LIST

Create AutoCAD Table:

In this group, an AutoCAD Table is made in the current document. The table's location and style settings are separately defined. The data input consists of a list of rows and each row consists of a list of strings (size corresponding with the number of columns).

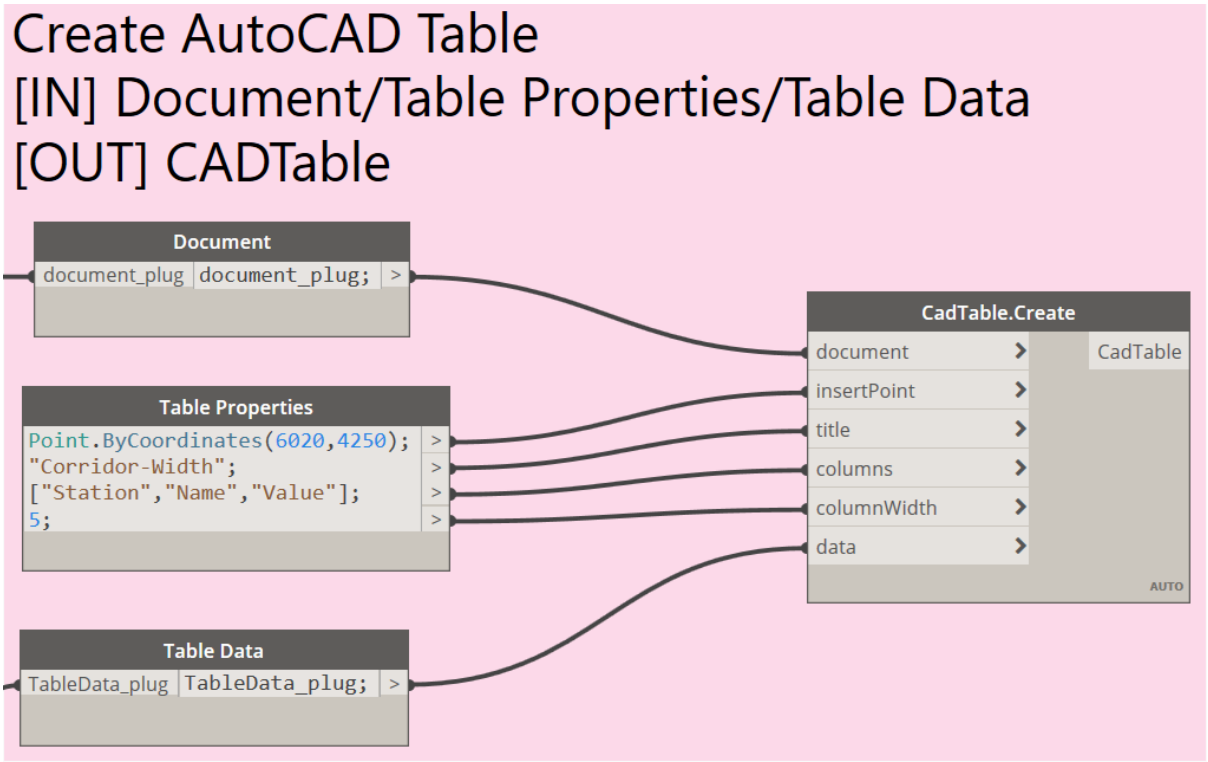
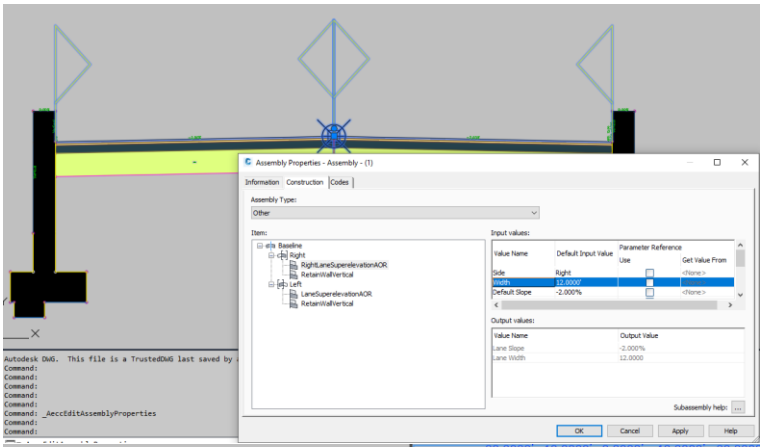
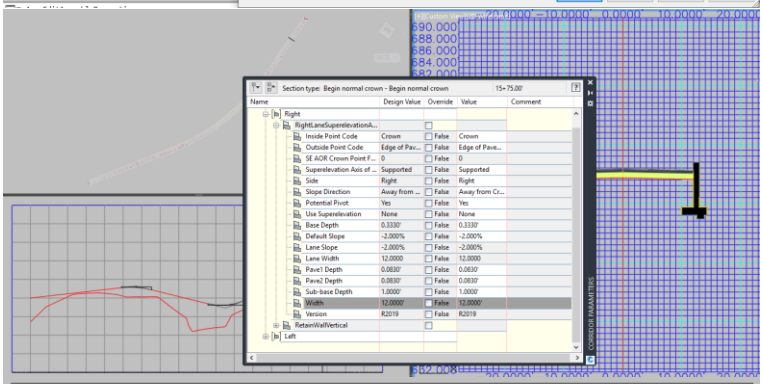


FIGURE 96: CPA - CREATE AUTOCAD TABLE

Result





Corridor – Width

Station	Name	Value
0.000000	Width	12.000000
25.000000	Width	12.000000
50.000000	Width	12.000000
75.000000	Width	12.000000
100.000000	Width	12.000000
125.000000	Width	12.000000
150.000000	Width	12.000000
175.000000	Width	12.000000
190.600000	Width	12.000000
200.000000	Width	12.000000
225.000000	Width	12.000000
250.000000	Width	12.000000
275.000000	Width	12.000000
300.000000	Width	12.000000
316.000000	Width	12.000000
325.000000	Width	12.000000
350.000000	Width	12.000000
375.000000	Width	12.000000
399.500000	Width	12.000000
400.000000	Width	12.000000
425.000000	Width	12.000000
450.000000	Width	12.000000
475.000000	Width	12.000000
483.100000	Width	12.000000
489.229322	Width	12.000000
490.000000	Width	12.000000
500.000000	Width	12.000000
510.000000	Width	12.000000
520.000000	Width	12.000000
525.000000	Width	12.000000
530.000000	Width	12.000000
540.000000	Width	12.000000
549.516995	Width	12.000000

FIGURE 97: CORRIDOR PARAMETER ANALYZER

Corridor Automated Ruler

This Dynamo script enables the user to automatically measure between two points in an Assembly. Both points need to have an identical point code for the script to work.

While it is true that a user can already do this manually with the Section Editor or the AutoCAD measuring tools, it is simply not efficient to do this for long corridors.

In addition to the measuring task, the script contains some additional functions:
- Editing of AppliedSubassembly parameters, to change the design.
- Creation of an AutoCAD table.
- Evaluation of the horizontal distance between the points.
- Formatting the AutoCAD table.
- Placement of indication objects.

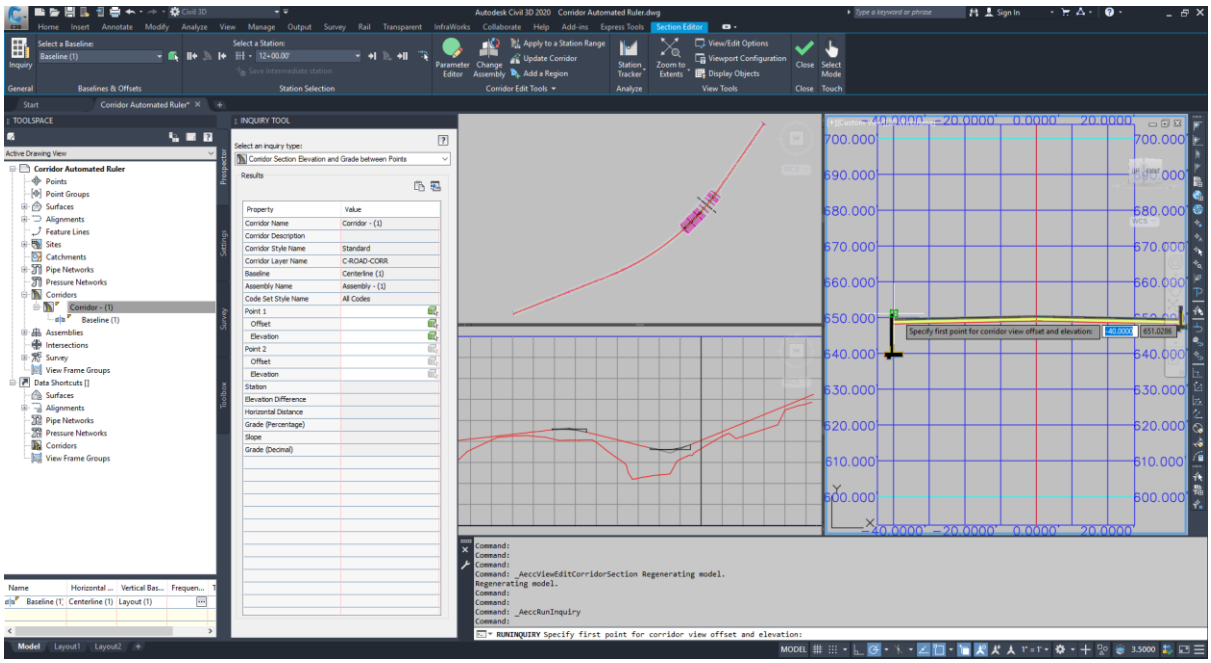


FIGURE 98: CIVIL 3D SECTION EDITOR - INQUIRY TOOL

Workflow

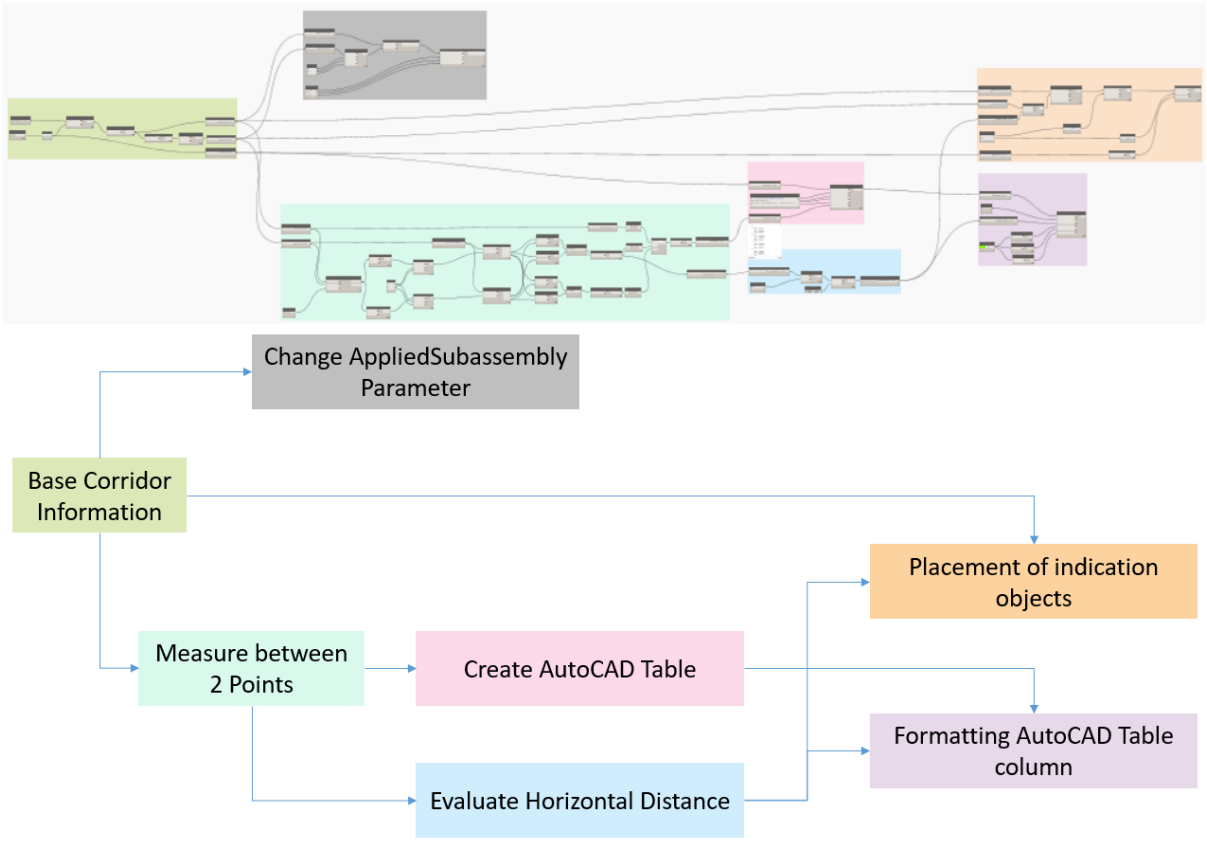


FIGURE 99: CORRIDOR AUTOMATED RULER

Base Corridor Information:

At the beginning, the corridor is retrieved by name from the current document. Next, the corridor baseline and its stations are retrieved. The current document, the baseline and a flattened list of the stations is made available for other groups.

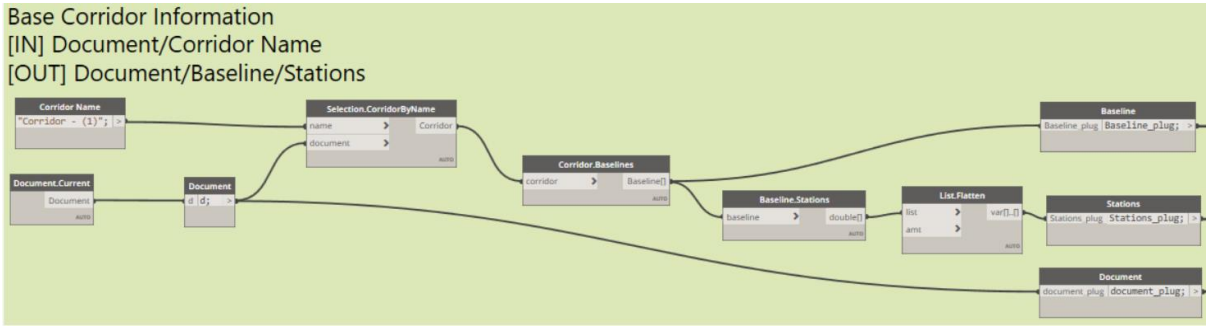


FIGURE 100: CAR - BASE CORRIDOR INFORMATION

Change AppliedSubassembly Parameter:

The Baseline station list is sliced to a user-defined zone. The resulting station list is used to get the AppliedSubassemblies and alter the chosen parameter by a specific value and rebuilt.

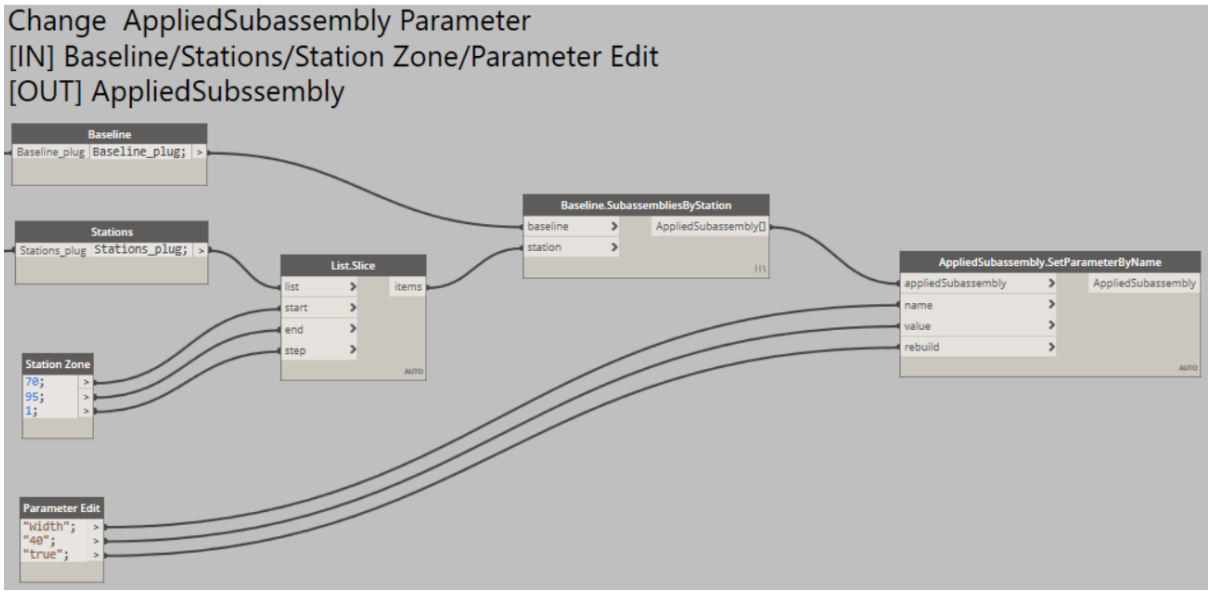


FIGURE 101: CAR - CHANGE APPLIEDSUBASSEMBLY PARAMETER

Measure between two Points:

First, the points with a user defined point code are retrieved for every section on the baseline. Next, the points are sorted into two lists (left and right). For each list, the station offset to the baseline is calculated. Then using some value comparisons and basic math the horizontal distance and vertical height difference is calculated. These values are combined with the corresponding station into one list. At the end the numeric horizontal distance and the combined data list is made available.

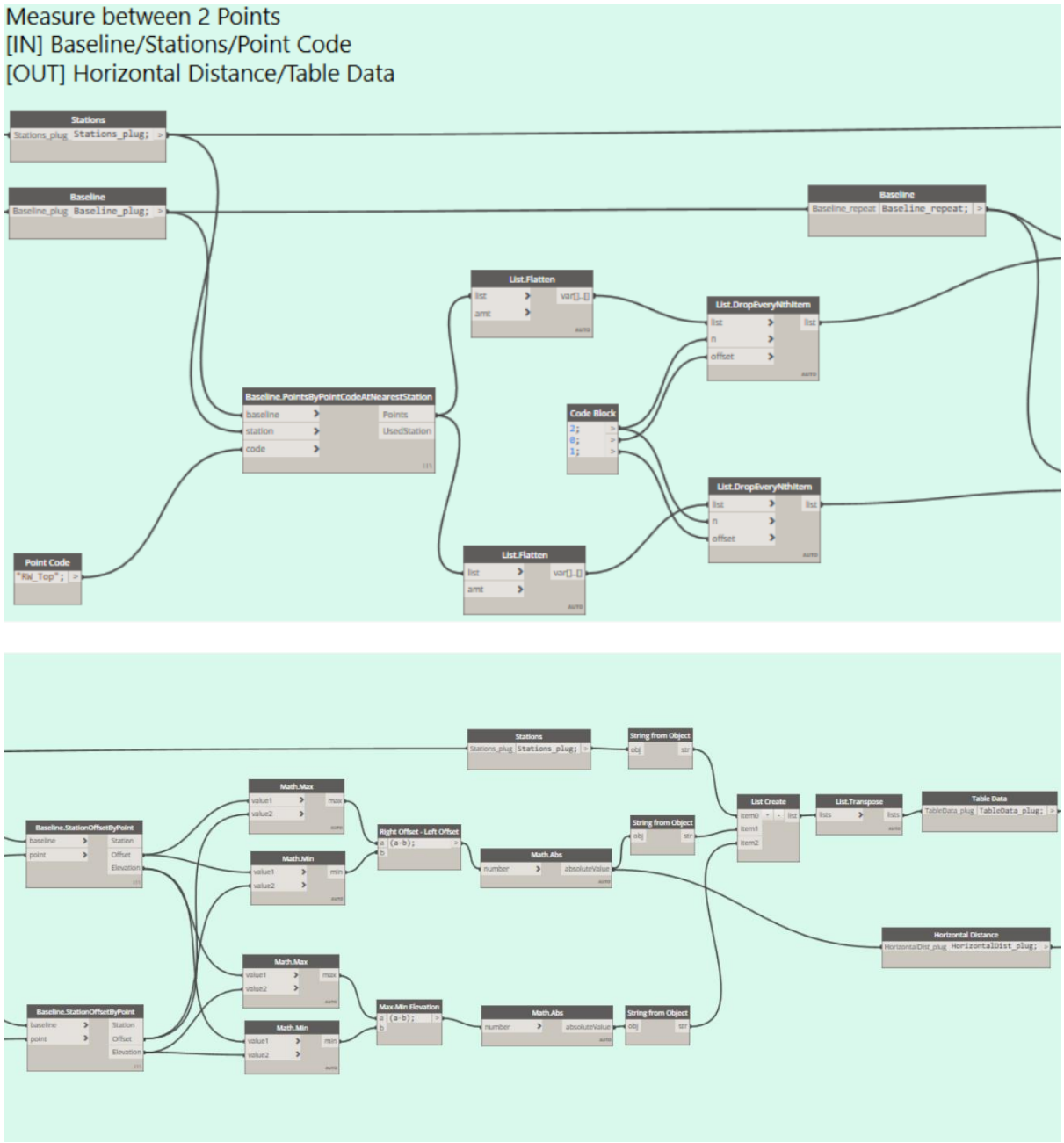


FIGURE 102: CAR - MEASURE BETWEEN 2 POINTS

Create AutoCAD Table:

In this group, an AutoCAD Table is made in the current document. The table's location and style settings are separately defined. The data input consists of a list of rows and each row consists of a list of strings (size corresponding with the number of columns).

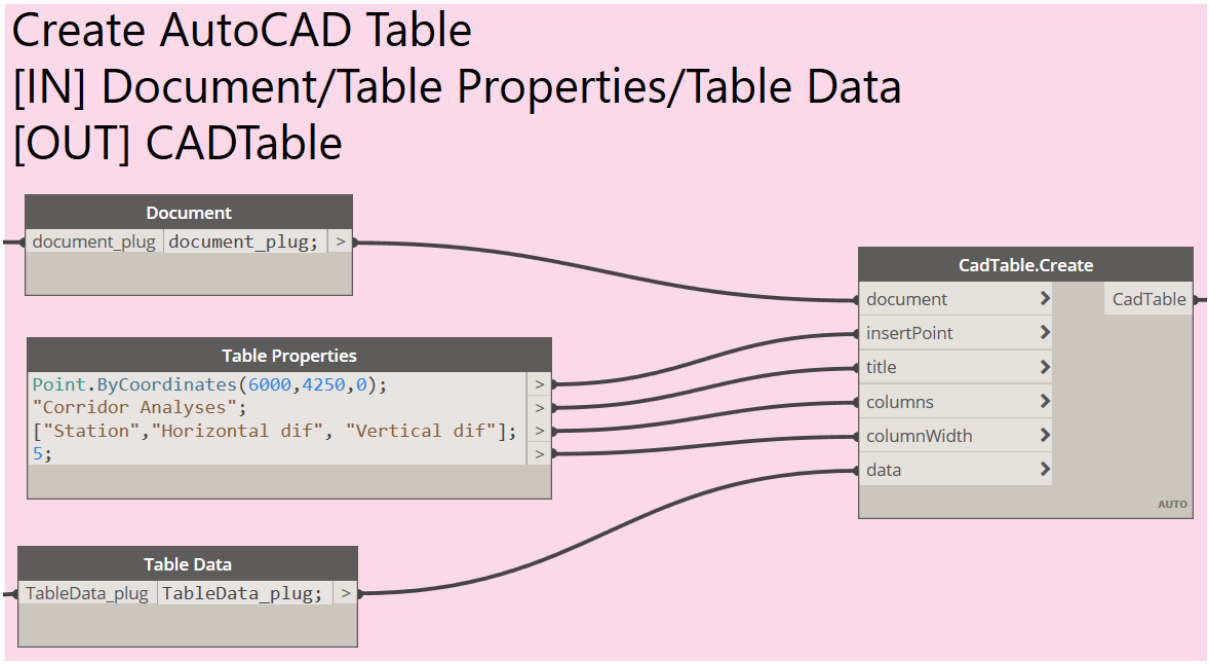


FIGURE 103: CAR - CREATE AUTOCAD TABLE

Evaluate Horizontal Distance:

In this example a basic evaluation is performed on the result horizontal distance (>30). All positive indices are combined into a list for further use in the script.

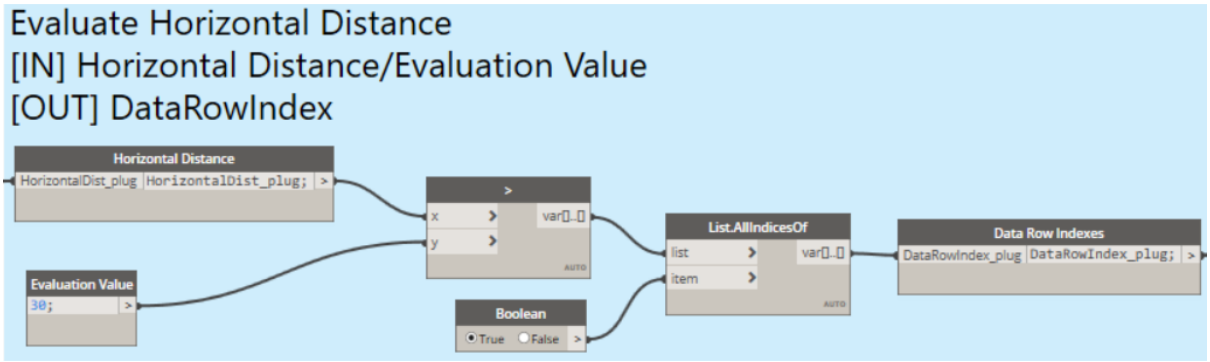


FIGURE 104: CAR - EVALUATE HORIZONTAL DISTANCE

Formatting AutoCAD Table column:

This group contains the actual conditional formatting. The created table is retrieved, and the user indicates which column and row indices need to change color. For the color selection, a Color Palette node is used.

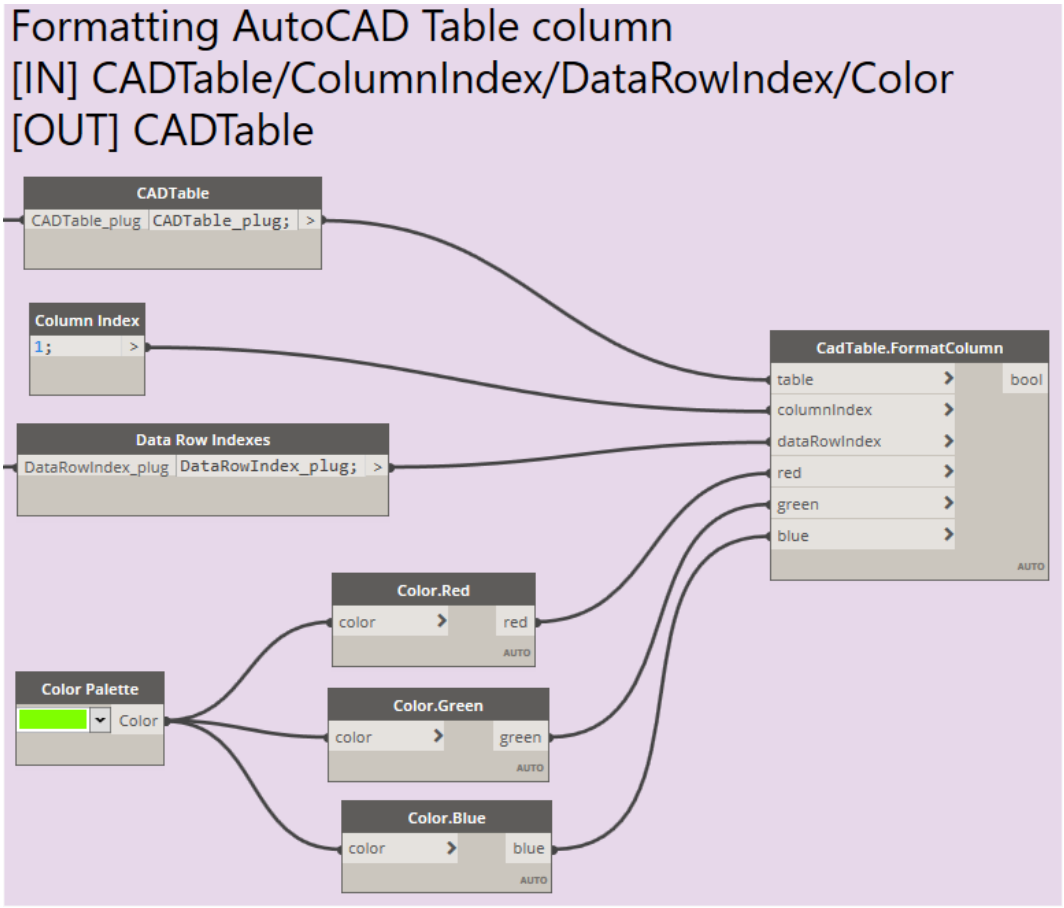


FIGURE 105: CAR - FORMATTING AUTOCAD TABLE COLUMN

Placement of indication objects:

As a second reporting method, the positive row indices are used to retrieve the corresponding station. Then using the station, the corresponding coordinate is retrieved on the baseline. Finally, a circle is created to indicate where the measured value is positive according to the evaluation group.

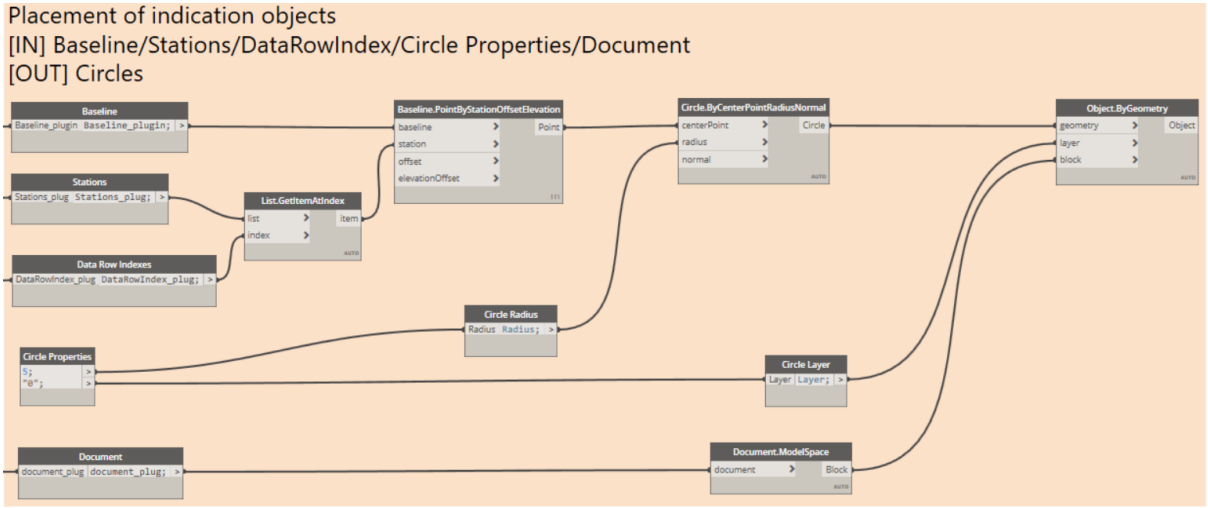
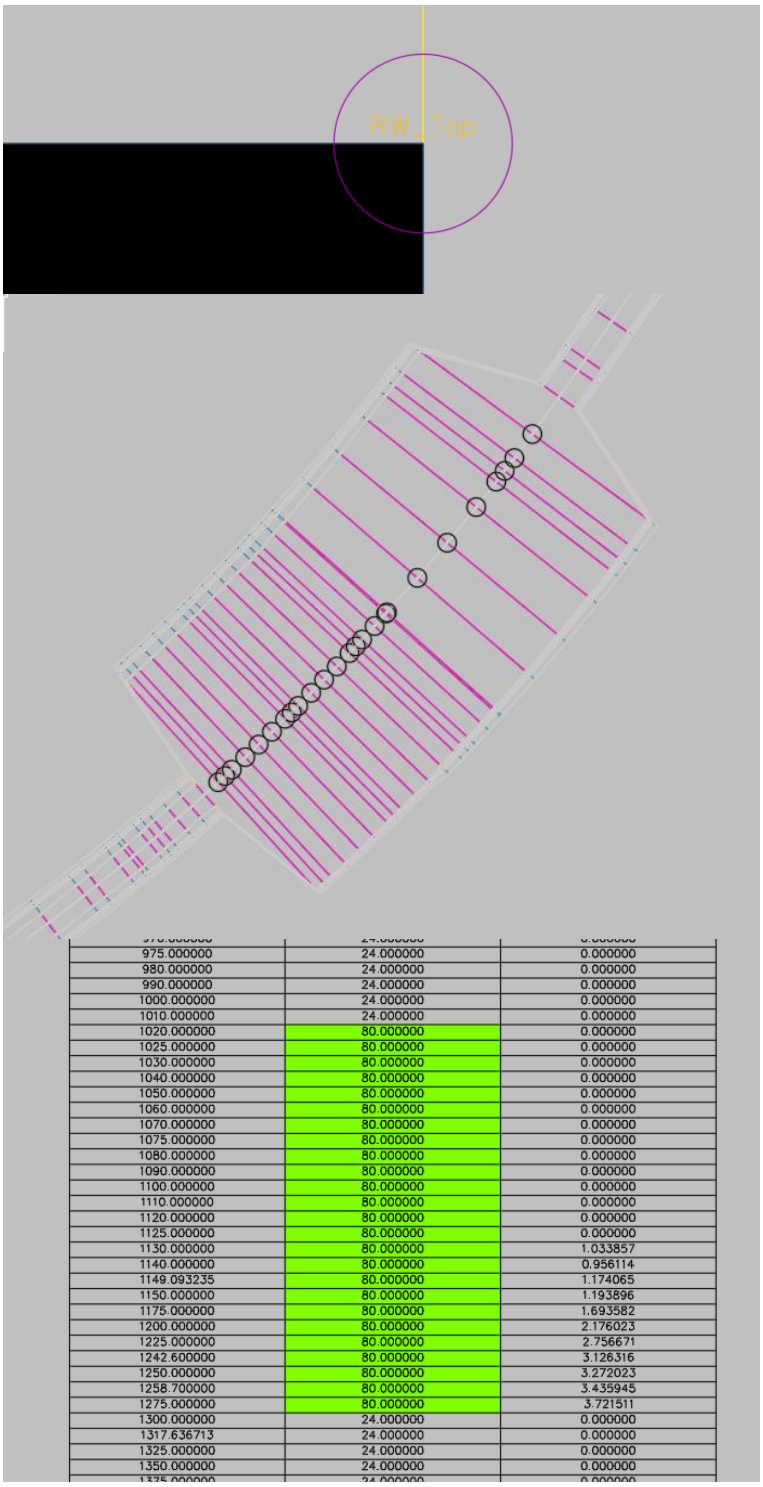


FIGURE 106: CAR - PLACEMENT OF INDICATION OBJECTS

Result



975.000000	24.000000	0.000000
980.000000	24.000000	0.000000
990.000000	24.000000	0.000000
1000.000000	24.000000	0.000000
1010.000000	24.000000	0.000000
1020.000000	80.000000	0.000000
1025.000000	80.000000	0.000000
1030.000000	80.000000	0.000000
1040.000000	80.000000	0.000000
1050.000000	80.000000	0.000000
1060.000000	80.000000	0.000000
1070.000000	80.000000	0.000000
1075.000000	80.000000	0.000000
1080.000000	80.000000	0.000000
1090.000000	80.000000	0.000000
1100.000000	80.000000	0.000000
1110.000000	80.000000	0.000000
1120.000000	80.000000	0.000000
1125.000000	80.000000	0.000000
1130.000000	80.000000	1.033857
1140.000000	80.000000	0.956114
1149.093235	80.000000	1.174065
1150.000000	80.000000	1.193896
1175.000000	80.000000	1.693582
1200.000000	80.000000	2.176023
1225.000000	80.000000	2.756671
1242.600000	80.000000	3.126316
1250.000000	80.000000	3.272023
1258.700000	80.000000	3.435945
1275.000000	80.000000	3.721511
1300.000000	24.000000	0.000000
1317.636713	24.000000	0.000000
1325.000000	24.000000	0.000000
1350.000000	24.000000	0.000000

FIGURE 107: EXAMPLE CORRIDOR AUTOMATED RULER

Conclusion

The railway industry and many others face the challenge of separating the used medium and tools from the design. Of integrating different disciplines into one design, rather than the combination of individual sub-designs. The challenge of learning to trust not only the ruler but start interacting and designing in a new way.

In this handout, we have explained how design objects, data and analysis can help us to meet these challenges and we have illustrated this with specific examples.

We have shown how a dynamic block can be used to create custom design objects that support your design process and standards. How to use Dynamo to connect this custom object with existing objects in AutoCAD, Civil 3D and potentially many others. And finally, use Dynamo to automate existing analysis tools like data extraction and create new feedback and measuring methods for a corridor.

I hope this handout will motivate and help you to set the first steps in creating your own design object, data and analysis. It is guaranteed that in the future new media and tools will come, but I am convinced that the principles applied in this handout will help us see the opportunities in the challenges we face.

Best of luck,

Wouter

Special thanks to Anneleen van Passel, Steve Crokaert, Michael Cox and KaDe King for proofreading and improving this handout.

References

Countless posts on Autodesk Forums and previous Autodesk University handouts have contributed to this handout. The following references deserve specific mention:

- Autodesk Forums: “How to Make a Double Lookup” https://forums.autodesk.com/t5/dynamic-blocks-forum/how-to-make-a-double-lookup/td-p/5785708
- Autodesk University 2008: “CP301-2L Using the New AutoCAD Data Extraction API with VB.NET”
- Autodesk University 2011: “CP4887 Navigating Through the Corridor Using AutoCAD Civil 3D .NET API”

Class Material

- This handout, the presentation and the exercise files (dwg’s and dynamo scripts) can be found on Autodesk University under AU2019 class CES321918 (Class Handout and Additional Class Materials)
- All videos and codes for the custom nodes can be found on GitHub. (dwg’s and dynamo scripts are also available there) https://github.com/TUCRAIL/AU2019

Table of Figures