"Best Practices", Visualization

Version: 2.0
Template: templ_tecdoc_en_V3.0.docx
File name: Best Practices Visualization EN.docx

CONTENT

1 Advice

1.1 Keeping static elements in the background

The visualization can best optimize elements without linking to variables when these are in the visualization behind the elements linked to variables (whenever possible).

The optimizations result in better performance and a lower memory requirement for the visualization.

1.2 Design frequently changing elements as small as possible

A large box element with frequently changing small text causes poorer performance than a box of more appropriate size.

1.3 Selecting hidden elements

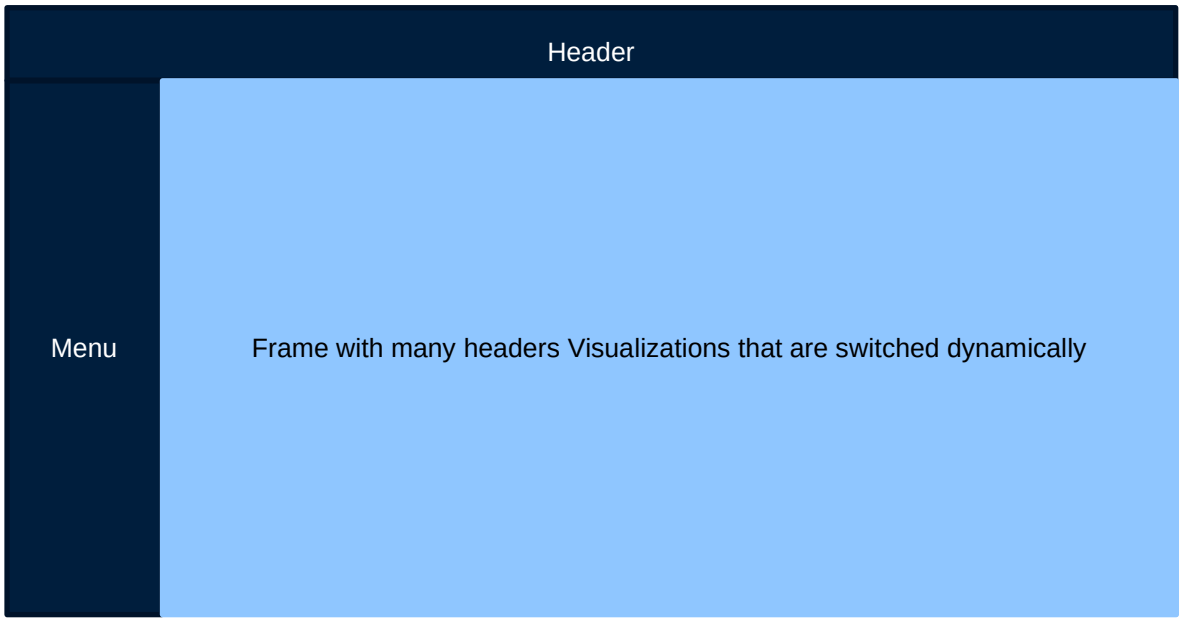
All elements can be selected at a mouse position by means of <Ctrl>+<Shift>+Click.

1.4 Avoid many invisible elements

The performance of the visualization is better when a group or frame is configured invisible rather than individual elements being invisible. When there are many individual invisible elements, performance can be improved by grouping.

1.5 Visualization with header and menu

It is better for performance when only part of the entire display area is redrawn. It is better to switch the contents into a frame than to switch the entire visualization. Do not copy and paste headers and menus to many main visualizations. Instead, provide a landing page (with header and menu) with subpages that switch within the frame.



1.6 Visualization objects for function blocks – Structuring the main pages into subdivisions

A visualization provides an interface editor. This is used for passing parameters to a visualization. In this way, visualization objects can be created that can be reused by means of the frame element and can be linked in other visualizations.

The following applies for interface parameters:

- It is better to pass one parameter (for example a function block instance) than multiple scalar parameters.
- VAR_IN_OUT should be used for this because the instance is not copied.

One elegant form of input processing is to use "Execute ST code" in an event (for example: OnMouseClick) in the input configuration and to call a method or property of the passed function block instance in the ST code. This makes debugging easier and improves the structure.

1.7 Visualization states vs. instance states

If the visualization is structured as recommended (assignment FB->Visu via interface/VAR_IN_OUT, see Section 1.6), then variables that contain the state of the object (for example, operating mode: operational/starting-up/in error) are created in the FB.

Only variables that contain just the state of the visualization should be declared itself as VAR in the visualization.

Example: The selection should be performed in an own drawn list. Exception: The list contains ten cells with box elements and a text that comes from the "pou" function block.

```
VAR_IN_OUT
  pouIn : POUList;
END_VAR
VAR
  arrSelection : ARRAY[VISU_MIN_NUMBER_OF_CLIENTS..VISU_MAX_NUMBER_OF_CLIENTS] OF
INT;
END_VAR
```

The constants VISU_MIN_NUMBER_OF_CLIENT/VISU_MAX_NUMBER_OF_CLIENTS are available in CODESYS Version V3.5 SP10 and later.

The cells should be configured as follows, where INDEX must be replaced by the respective index of the cell.

Texts – Text:%s

Text variables – Text variable: pouIn.arr[INDEX] → The character string that should be displayed in the cell.

Color variable – Toggle color: arrSelection[CurrentClientId] = INDEX

Input configuration – OnMouseEnter – Execute ST code: arrSelection[CurrentClientId] := INDEX;

Note: Moreover, the library "VisuGlobalClientManager" must also be linked in order to work with the variable "CurrentClientId".

This example demonstrates how to use client-specific data directly in the visualization.

1.8 Visualization styles

The visualization style that was used for designing the project should be defined from the start. If it is preferable that the visualization elements behave accordingly when the style is modified, then the element properties should be set to style values, if possible. Fixed colors or fonts do not change when the style is modified.

Typical style entries include the following:

- List of colors
- List of fonts
- Images for visualization elements
- Selection of the base style for the visualization elements

<AdditionalStyles>
<!-- Styledefinition Style1-->
<Value name="Style" type="string">STYLE1_XP</Value>

- STYLE1_XP
- STYLE2_W7
- STYLE3_GRADIENT1_LINEAR1
- STYLE4_GRADIENT2_LINEAR2
- STYLE5_GRADIENT3_AXIAL1
- STYLE6_GRADIENT4_AXIAL2
- STYLE7_GRADIENT5_DOUBLELINEAR1
- STYLE8_GRADIENT6_DOUBLELINEAR2
- STYLE9_FLAT
- STYLE_WHITE

2 Typical use cases

Many visualization examples are displayed when searching the "Visualization" category in the CODESYS Store.

For example:

- Global Client Manager
- Multitouch Example
- Recipe Management
- Visu Event Handler
- Sound Demo
- Selection Manager
- Responsive Design Example
- etc.

2.1 Dynamic image toggling

Requirement:

Different images (pixel or vector graphics) should be toggled dynamically in a visualization.

Implementation:

Add image element. In the properties of the element in "Image ID variable", specify a string variable for the "Image ID". This string variable can come from an IEC 61131-3 POU with the respective character string. The POU corresponds to the name of the image from the image pool (example: "ImagePoolStates.Run").

There is an image pool named "ImagePoolStates". It includes the following entries in the ID column: Run, Stop,…, as well as the corresponding file names for the images in the file name column.

2.2 Dynamic text toggling

Requirement:

The text of the current error should be displayed in a box in the visualization. The error is contained in a DWORD variable.

Implementation:

Insert a box element. The name of the text list (example: "Error texts") and the text index is specified in the properties of the element in "Dynamic texts". The text index is a string variable that can come from an IEC 61131-3 POU. If the error number is in a DWORD variable, then DWORD_TO_STRING(fb.dwError) must be configured, for example.

A text list named ""Error texts" provides an error text for each error value. The error texts can be provided in multiple languages.

2.3 Mode switching with different levels

Requirement:

Different operating modes can be activated in the visualization.

Implementation:

Inserting a potentiometer element and corresponding configuration in the properties:

- Change the number of selectable modes: In the "Scale" area, change the values for "Main scale" and "Subscale“ to "1". Set "Scale start" and "Scale end" so that the number of values corresponds to the required operating modes.
- Input of the writing variable in ""Variable"
- As appropriate, set default entry "Scale format" to "%s" (positive integers only) and set fixed text before it: for example, "Mode %s" ➔ "Mode 1", "Mode2", etc.
- As appropriate, modify the switch points in the "Arrow" area with new values for "Arrow start" and "Arrow end".

2.4 Dynamic switching to different visualizations from one main page

Requirement:

A main page should be linked to subpages. A menu for toggling should be created ion the left edge (see project structure from Section 1.6 or sample project "VisuDemoFactory" in the installation directory `\CODESYS\Projects\Visu\Examples`).

Implementation:

First, create the visualizations for the subpages. Then, you can add a frame element to the main page and include the subpages as a list in the frame element from the context menu in "Frame selection".

The menu can be created with button elements. The following configuration is performed for this kind of button:

Input configuration – OnMouseDown – Toggle frame visualization.

Then, the frame element should be displayed on the main page.

Now you can select this kind of visualization from the frame to switch to by means of this button. Then select "Assign selection".

2.5 Customizing visualization dialogs

Requirement:

The design of the visualization dialogs "Numpad/Keypad" or "Login/ChangePassword" for the user management of the visualization should be modified according to the corporate design of the company.

Implementation:

The source libraries of the visualization dialogs are located in the installation directory `\CODESYS\Projects\Visu\Dialogs`. These libraries can be opened and the contained visualizations can be customized. Then you must also modify the project information of the library (own company name, own version, etc.).

Note: All dialogs are visualizations that have been identified specifically in the visualization object properties, category "Visualization" by means of the selection "Dialog" or "Numpad/Keyboard/Dialog for input configuration".

Now you can include the newly created, company-specific library in the library manager of the project. Then the dialogs of this library can be selected, as well as in the visualization manager in "Settings"/"Settings for default text input" or "Settings for user management dialogs".

These specific dialogs are used then in runtime mode.

Change History

Version	Description		Date
0.1	Issued		07.12.2016
0.2	Formal Review and Rework		05.01.2017
1.0	Formal Release		10.01.2017
1.1	Update to Template V3.0		16.11.2023
2.0	Formal Review and Release		23.11.2023