Controls Styles Readme

Label

Here is what the Help might say:

This dialog has several parts:

- Presets, which are combinations of styles which are known to work
- Styles groups, where Koda shows which styles are included in a Preset, and where you can choose your own styles
- Autoit parameters lists, that show which Windows constants will appear in generated code
- A list of Windows constants that AutoIt help says that it uses when it calls the Windows Create function, including those AutoIt forces
- Lists of the differences between what Windows constants AutoIt apparntly requests, and what Windows actually provided.
- A status indicator, which shows "Preset" and a number, "Custom", or "Failed".
- An optional picture of what the control will actually look like.

There are three ways of specifying styles for a control:

- Click on a Preset radio button: Presets are known to work; they cover most common requirements. The status indicator shows "Preset nn".
- Click on a Preset radio button, then check or uncheck styles: the status indicator usually shows "Custom", but may show "Failed".
- Click on the "None" radio button to uncheck all styles, then check styles as you wish: the status may be "Custom" or "Failed".

The style checkboxes you check are *requests* which Windows may fulfil (or may not): such are the ways of Windows!

While the "AutoIt or Windows ignored" list is an indicator that your form may not have the styles that you request, the Test feature shows exactly how your control will appear when the generated code is run. To turn this feature on, check "Show". Initially your control will appear for 2 seconds. You can show it for longer by changing the value in the spinner control.

If you hover the mouse over a style radio or check box, the Windows constant it provides is displayed.

“Left-aligned, one line only” expands tabs. It clips text that extends past the end of the line is clipped.

“Crude, fast text” left-aligns the text. If you update the text with shorter text, the end of the old text still shows. Disabling the control does not gray it.

“User’s code paints control” makes the user responsible for drawing the control. When the control is to be drawn, AutoIt expects user code to register and handle a `$WM_DRAWITEM` message.

“One-line text is vertically centered in the control” even when the form is resized.

“Like Edit control” duplicates the text-displaying characteristics of an Edit control: the average character width is calculated in the same way as in the Edit control, and a partially visible last line

is not displayed.

“& is a character, not an accelerator prefix”: When unchecked a single ampersand (&) displays the next character underlined (and this character is an accelerator). To display an ampersand, enter &&. When checked, ampersands are taken literally.

“Over-paint other controls”: If this control overlaps with another, this control shows partially.

“Is drag handle for Form”: Form can be repositioned by dragging this label

---

AutoIt defines the following SS_ constants:

```
Global Const $SS_LEFT = 0x0
Global Const $SS_CENTER = 0x1
Global Const $SS_RIGHT = 0x2
Global Const $SS_ICON = 0x3
Global Const $SS_BLACKRECT = 0x4
Global Const $SS_GRAYRECT = 0x5
Global Const $SS_WHITERECT = 0x6
Global Const $SS_BLACKFRAME = 0x7
Global Const $SS_GRAYFRAME = 0x8
Global Const $SS_WHITEFRAME = 0x9
Global Const $SS_SIMPLE = 0xB
Global Const $SS_LEFTNOWORDWRAP = 0xC
Global Const $SS_BITMAP = 0xE
Global Const $SS_ETCHEDHORZ = 0x10
Global Const $SS_ETCHEDVERT = 0x11
Global Const $SS_ETCHEDFRAME = 0x12

Global Const $SS_NOPREFIX = 0x0080
Global Const $SS_NOTIFY = 0x0100
Global Const $SS_CENTERIMAGE = 0x0200
Global Const $SS_RIGHTJUST = 0x0400
Global Const $SS_SUNKEN = 0x1000
```

`SS_LEFT` thru `SS_ETCHEDFRAME` require one group of Radios.

The AutoIt help says that the default is 0 and that `$SS_NOTIFY` and `$SS_LEFT` are forced. Testing shows that this is true. `$SS_LEFT`, being zero, forces nothing; why `$SS_NOTIFY` is forced is unknown to me. Testing shows that `$WS_CHILD`, `$WS_GROUP` and `$WS_VISIBLE` are also forced, although the AutoIt help does not mention them. Makes no sense to me.

`SS_PATHELLIPSIS` (0x8000) can be useful; `SS_ENDELLIPSIS` (0x4000) and `SS_WORDELLIPSIS` (0xc000) may be useful. They are missing from AutoIt, so Code Generator would need to generate them. They will require Radios. They do work, except `SS_WORDELLIPSIS` appears to do the same as `SS_ENDELLIPSIS`..

`SS_ENHMETAFILE` is missing from AutoIt and may possibly be useful, but I don’t understand it – so it is not in the dialog. `SS_OWNERDRAW` is also missing from AutoIt, but is included in the dialog.

`SS_CENTERIMAGE, SS_REALSIZECONTROL` and `SS_REALIZEIMAGE` do not pertain to Label

Testing shows that `WS_CLIPSIBLINGS` causes the Label not to clip other Labels!

`WS_HSCROLL` and `WS_VSCROLL` display, but are inactive.

The following I think only apply to a Form, so do not appear in the Label dialog:

- Form is toolbar (`WS_EX_TOOLWINDOW`),

- Avoid clipping children (WS_CLIPCHILDREN),
- Maximize initially (WS_MAXIMIZE),
- Minimize initially (WS_MINIMIZE),
- Always on top (WS_EX_TOPMOST),
- Layered effects (WS_EX_LAYERED),
- Scrollbars are on left/bottom (WS_EX_LEFTSCROLLBAR),
- Right-aligned text (WS_EX_RIGHT),
- Right-to-leeft reading order (WS_EX_RTLREADING),
- Double-buffer to reduce flicker (WS_EX_COMPOSITED),
- Form appears in task bar (WS_EX_APPWINDOW),
- Accept filenames in Input/Edit (WS_EX_ACCEPTFILES),
- Simulate MIDI (WS_EX_MIDICHILD),
- Tab key navigates among children (WS_EX_CONTROLPARENT),
- Modal(DS_MODALFRAME),
- Popup (WS_POPUP).

Child (WS_CHILD) is always true for a control, so does not need to be in the dialog.

Form is transparent: The AutoIt help says "To set the background to transparent, use GUICtrlSetBkColor(-1, $GUI_BKCOLOR_TRANSPARENT)." **I see background colour as a Property**. It would be nice to see background colours available for many controls.

I tried to replace ShowTestForm() with ShowTestLabel() but the test label did not work properly for the Force-fit feature, so I continue with ShowTestForm().

SS_EDITCONTROL pertains to Label, is not in Autoit, and is not present in this dialog. It may not be useful.

SS_ENHMETAFILE pertains to graphics, is not in Autoit, and is not present in this dialog.

Images

The following SS constants only apply to an image: SS_BITMAP, SS_ICON, SS_REALSIZECONTROL, SS_REALSIZEIMAGE and SS_RIGHTJUST. Which of these does Picture Editor handle? Which does the proposed Styles Editor need to handle?

I have learnt that ControlSetText() can be used to show text in an apparently image-only Static control. So the Styles Editor for Pic, Icon and Graphic could look much the same as that for Label. Would it be identical?

It occurs to me that if Koda will have a Styles Dialog for images that has styles for text, it would make sense to have Code Generator generate the call to ControlSetText().