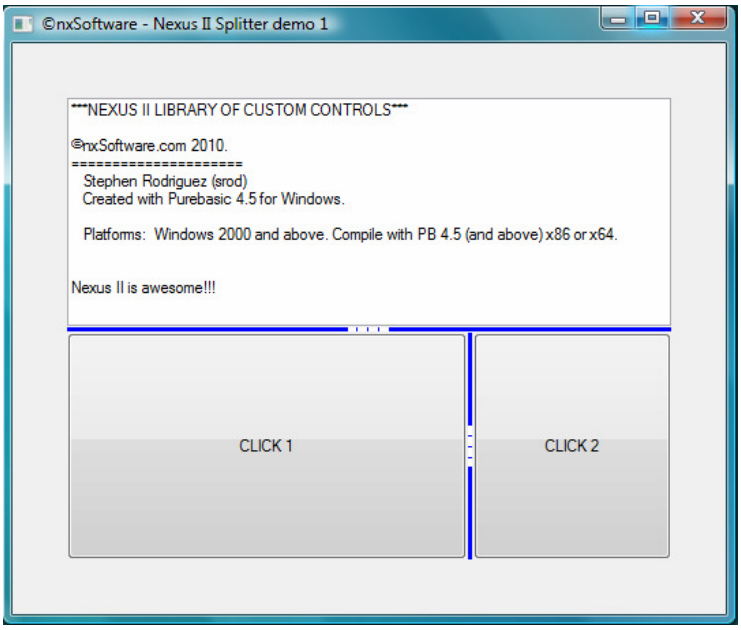
Nexus II

Splitter control

by Stephen Rodriguez



General.

The Nexus II splitter control is a more powerful and flexible version of the familiar PureBasic splitter gadget with two important differences :

1. The two 'split' gadgets are not resized dynamically as the user moves the 'slider'. This generally makes for a much smoother operation.
2. There is the option of adding a 'gripper' to the slider which allows the user to 'anchor' one of the two 'split' gadgets. Indeed, you can even prevent the slider from being dragged whilst one of the gadgets is anchored etc. (See the second of the two splitter demo programs.)

On top of this, and on top of being able to customise the splitter in various ways, it functions in a very similar fashion to the PureBasic splitter in that it allows you to specify minimum sizes for the 'split' gadgets and indeed keep one gadget fixed whilst the entire control is being resized etc.

Using the Nexus II splitter control within your own programs.

First ensure that you have correctly installed the Nexus II source files. See the Nexus II .chm user guide for more details of this.

Any program requiring the services of the Nexus II splitter control needs to declare the constant :

```
#INCLUDE_NEXUSII_SPLITTER = 1
```

at the top of the source, before adding the main Nexus II source file into the mix.

At this point, when you compile your program, the Nexus II splitter control will be included within the compilation.

Creating an instance of the Nexus II splitter control.

To create a new instance of a Nexus II splitter control, we use the NexusII_CreateSplitter() function.

This function has the following Purebasic prototype :

```
Procedure.i NexusII_CreateSplitter(x, y, width, height, gadget1, gadget2,
sliderWidth, styles=0, callback=0)
```

with the following parameters :

x, *y*, *width*, *height*

location and dimensions of the control.

gadget1, *gadget2*

the Purebasic gadget# of the two child gadgets which will be automatically resized by the splitter. These gadgets will be embedded within the splitter (as child controls).

Note that the splitter will still function if one (or both) of the specified gadget# do not actually identify a gadget! This can be useful for those wishing to employ some custom painting directly within the splitter control.

sliderWidth

Pixels. Defaults to 3 pixels if an invalid value is given. Equates to a slider height in the case of a horizontal splitter.

styles

A combination of the following values :

- *#NEXUSIISPLITTER_VERTICAL*

  The default is a horizontal splitter.

- *#NEXUSIISPLITTER_GRIPPERWITHDRAG*

  Adds a *gripper* to the slider, one which can be dragged along with the slider itself. When the user clicks a gripper, one or the other of the two child gadgets is collapsed down to it's specified minimum size.

- *#NEXUSIISPLITTER_GRIPPERWITHNODRAG*

  As above, but with this style, the gripper cannot be dragged. Only that part of the slider other than the gripper can be dragged to resize the child controls.

- *#NEXUSIISPLITTER_FIRSTFIXED*

  When the entire splitter control is resized, then this style bit ensures that the *first* child control remains fixed in size whilst the *second* child control will be resized (and the slider repositioned accordingly).

- *#NEXUSIISPLITTER_SECONDFIXED*

  As above, but this time the *second* child control remains fixed in size etc.

- *#NEXUSIISPLITTER_DONOTERASECHILDRECTS*

  To reduce flicker when employing dynamic resizing of controls (e.g. resizing a Nexus II splitter when it's parent window is resized by the user) then you can consider using this style bit.

  This style will remove all erasing from those parts of the splitter control in which the two child controls reside (or would reside in the case that you opt not to add any child controls).

  For example, if you use a Nexus II splitter with two button gadgets (as at least one of our demo programs does) then you would be advised to use this style bit simply because there is no point adding an additional layer of erasing to the part of the splitter occupied by the button gadgets! On the other hand, you would be advised not to use this style if using a splitter directly on a ComboBox gadget as these gadgets will typically not expand to fill the entire space allotted

to it by the splitter and thus removing erasing will lead to all kinds of redrawing issues!

This style bit can also be useful for advanced users who employ one of our splitters without any child controls and instead utilise custom painting techniques.

callback

Optional.

The address of a Nexus II call-back function residing within your own program. This function will receive notifications and messages as described in the 'Event call-back' section below.

The function, if successful, will return a fully instantiated NexusIIControl object as described in the Nexus II .chm user guide. Use the appropriate methods of this object to administer the control as appropriate, -and as described in the next section.

NexusIIControl methods for use with a splitter control object.

After a successful invocation of the above function, you will be left with a fully instantiated NexusIIControl object pointer.

Now, not all of the NexusIIControl class methods are appropriate for use with a splitter control object and so you must refrain from attempting to use any NexusIIControl method which is not specifically listed here as being appropriate for such a control object. Any attempt to do so will be met with a very unseemly crash!

List of NexusIIControl class methods for use with a splitter control object.

We offer up detailed explanations for a method's use only where warranted as many of the following methods are entirely self explanatory.

\Destroy()
Use only if destroying a splitter object before freeing it's parent control/window as otherwise this method will be called automatically.

This method will obviously also free any child controls.

\GetControlType()
Returns *#NEXUSII_SPLITTER*.

\Disable(state)

\GetAttribute(attribute)
Retrieves the appropriate attribute for the splitter control.

See the \SetAttribute() method below for a list of appropriate values for the *attribute* parameter.

\GetColor(colorType)
Retrieves the specified colour for the underlying control.

See the \SetColor() method below for a list of appropriate colour types.

\GetData()

\GetHeight()

\GethWnd()

\GetID()
Returns the Purebasic gadget# of the container gadget used as the basis for the Nexus II splitter.

\GetState()
Returns one of the following :

- *#NEXUSIISPLITTER_ANCHORFIRSTGADGET*
  Indicates that the first gadget has been anchored (due either to the user clicking the gripper or through the \SetState() method).

- *#NEXUSIISPLITTER_ANCHORSECONDGADGET*
  Indicates that the second gadget has been anchored (due either to the user clicking the gripper or through the \SetState() method).

- A positive value to indicate the position of the slider. This position equates to the width of the first child gadget + the size of the internal margin (which can be retrieved with the \GetAttribute() method).

\GetWidth()

\GetX()

\GetY()

\Hide(state)

\Resize(x, y, width, height)

\SetAttribute(attribute, value)
Sets the appropriate attribute for the splitter control.

Possible attributes are :

- *#NEXUSIISPLITTER_FIRSTMINIMUMSIZE*
  (Get and Set.)

- *#NEXUSIISPLITTER_SECONDMINIMUMSIZE*
  (Get and Set.)

- *#NEXUSIISPLITTER_FIRSTGADGET*
  (Get and Set.)
  Use this attribute to change the *first* child control which is subject to the splitter's auto-sizing etc. Any existing child gadget will be returned to the splitter's parent.

  When retrieving, a value of -1 is returned if there is no first child control.

- *#NEXUSIISPLITTER_SECONDGADGET*
  (Get and Set.)

Use this attribute to change the *second* child control which is subject to the splitter's auto-sizing etc. Any existing child gadget will be returned to the splitter's parent.

When retrieving, a value of -1 is returned if there is no second child control.

- *#NEXUSIISPLITTER_GRIPPERLENGTH*
  (Get and Set.)
  Pixels.

- *#NEXUSIISPLITTER_SLIDERTYPE*
  (Get and Set.)
  Value = *#NEXUSIISPLITTER_GRIPPERWITHDRAG* or *#NEXUSIISPLITTER_GRIPPERWITHNODRAG* or zero to indicate that there is just a basic slider with no gripper.

- *#NEXUSIISPLITTER_SLIDERWIDTH*
  (Get only.)
  Pixels. This really equates to a slider height in the case of a horizontal slider.

- *#NEXUSIISPLITTER_SIZEOFSLIDERMARGIN*
  (Get only.)
  The number of pixels between a child control and the actual slider. Useful for advanced users employing some kind of custom painting within the splitter control.

\SetColor(colorType, color)
Sets the specified colour for the underlying splitter control.

Possible values for the colorType parameter are :

- *#NEXUSIISPLITTER_BACKCOLOR*
  This colour is used when erasing parts of the splitter control and also to fill any margins between the slider and the child controls.

- *#NEXUSIISPLITTER_SLIDERCOLOR*
- *#NEXUSIISPLITTER_ANCHOREDSLIDERCOLOR*

\SetData(value)

\SetState(state)
Use this method to set the slider position via the *state* parameter as follows :

- *#NEXUSIISPLITTER_ANCHORFIRSTGADGET*
  Anchor the first gadget (collapses the first gadget to its' minimum size).

- *#NEXUSIISPLITTER_ANCHORSECONDGADGET*
  Anchor the second gadget (collapses the second gadget to its' minimum size).

- *#NEXUSIISPLITTER_RESTORESLIDER*
  Restores the slider from a position of being anchored (if indeed the slider was anchored).

- A positive value giving the absolute position of the slider. This position equates to the width of the first child gadget + the size of the internal margin (which can be retrieved with the \GetAttribute() method).

Event call-back function.

Specifying a call-back function when creating an instance of a splitter object allows your application(s) to receive notifications and other messages from the control itself.

Your call-back must take the form of all Nexus II call-back functions :

Procedure.i *SplitterCallback*(splitter.NexusIIControl, uMsg, wParam, lParam, xParam)

as described in the Nexus II .chm user guide.

The messages pertinent to a splitter control object are detailed below.

#NEXUSIISPLITTER_SLIDERANCHORED

Notification that the slider has been anchored (either through the user clicking the gripper or through the \SetState() method) and one of the two child gadgets has been collapsed to its' minimum specified size.

wParam = 1 or 2 to indicate which child gadget has been collapsed.

The return value is ignored.

#NEXUSIISPLITTER_ANCHORRELEASED

Notification that a previously anchored slider has been released (either through the user clicking the gripper or through the \SetState() method) and returned to it's pre-anchor position.

wParam = 1 or 2 to indicate which child gadget has been *released* from its collapsed state.

The return value is ignored.

#NEXUSIISPLITTER_GRIPPERCLICKED

Notification that the user has clicked a gripper.

wParam = 1 or 2 to indicate which child gadget will be collapsed if the default behaviour of collapsing alternating gadgets is allowed.

The message doubles as a request for additional information regarding which child gadget should be collapsed. Return one of the following to direct the action which should be taken :

- *#NEXUSIISPLITTER_ANCHORFIRSTGADGET*
- *#NEXUSIISPLITTER_ANCHORSECONDGADGET*
- non-zero to alternate the collapsing of the two child gadgets. This is the default behaviour.
- Zero to halt the processing of this message. No gadget will be collapsed.

#NEXUSIISPLITTER_GADGETSRESIZED

Notification that the two child gadgets have been resized due to user action (e.g. repositioning the slider) or through code.

wParam = height (or width if using a vertical splitter) of the first gadget (or it's allotted space if there is no such gadget).

lParam = height (or width if using a vertical splitter) of the second gadget (or it's allotted space if there is no such gadget).

The return value is ignored.

#NEXUSIISPLITTER_CUSTOMPAINT

Advanced users only.

Sent immediately prior to Nexus II painting the slider margins (and the slider itself) and allows the host application to paint directly into the splitter window. It is sent after the BeginPaint_() call.

This is useful for splitters which do not utilise child gadgets, but paint their content directly etc.

wParam = HDC (a handle to a device context) into which the host application should paint.

lParam = *ps.PAINTSTRUCT.

The return value is ignored.