GNU Smalltalk Library Reference

Version 3.2.5
24 November 2017

by Paolo Bonzini

1 Graphical users interfaces with BLOX

1.1 Tree

Classes documented in this manual are **boldfaced**.

- *Object*
  - **BLOX.BEventTarget**
    - **BLOX.BCanvasObject**
      - **BLOX.BBoundingBox**
        - **BLOX.BEmbeddedImage**
        - **BLOX.BEmbeddedText**
        - **BLOX.BLine**
        - **BLOX.BRectangle**
          - **BLOX.BOval**
            - **BLOX.BArc**
      - **BLOX.BPolyline**
        - **BLOX.BSpline**
    - **BLOX.BEventSet**
      - **BLOX.BBalloon**
    - **BLOX.Blox**
      - **BLOX.BMenuObject**
        - **BLOX.BMenu**
          - **BLOX.BPopupMenu**
        - **BLOX.BMenuBar**
        - **BLOX.BMenuItem**
          - **BLOX.BCheckMenuItem**
      - **BLOX.BWidget**
        - **BLOX.BExtended**
          - **BLOX.BButtonLike**
            - **BLOX.BColorButton**
          - **BLOX.BDropDown**
            - **BLOX.BDropDownEdit**
            - **BLOX.BDropDownList**
          - **BLOX.BProgress**
        - **BLOX.BPrimitive**
          - **BLOX.BButton**
            - **BLOX.BRadioButton**
            - **BLOX.BToggle**
          - **BLOX.BEdit**
          - **BLOX.BForm**
            - **BLOX.BContainer**
              - **BLOX.BRadioGroup**
            - **BLOX.BDialog**
            - **BLOX.BWindow**

```
              BLOX.BPopupWindow
              BLOX.BTransientWindow
            BLOX.BImage
            BLOX.BLabel
            BLOX.BViewport
              BLOX.BCanvas
                BLOX.BScrolledCanvas
              BLOX.BList
              BLOX.BText
       BLOX.BTextBindings
     BLOX.BTextAttributes
     BLOX.BTextTags
     BLOX.Gui
```

1.2 BLOX.BArc

Defined in namespace BLOX
Superclass: BLOX.BOval
Category: Graphics-Windows

I can draw arcs, pie slices (don't eat them!!), chords, and... nothing more.

1.2.1 BLOX.BArc: accessing

endAngle
Answer the ending of the angular range that is occupied by the arc, expressed in degrees

endAngle: angle
Set the ending of the angular range that is occupied by the arc, expressed in degrees

fillChord
Specify that the arc will be filled by painting an area delimited by the arc and the chord that joins the arc's endpoints.

fillSlice
Specify that the arc will be filled by painting an area delimited by the arc and the two radii joins the center of the arc with each of the endpoints (that is, that a pie slice will be drawn).

from
Answer the starting point of the arc in cartesian coordinates

from: aPoint
Set the starting point of the arc in cartesian coordinates

from: start to: end
Set the two starting points of the arc in cartesian coordinates

startAngle
Answer the beginning of the angular range that is occupied by the arc, expressed in degrees

startAngle: angle
Set the beginning of the angular range that is occupied by the arc, expressed in degrees

sweepAngle
Answer the size of the angular range that is occupied by the arc, expressed in degrees

sweepAngle: angle
Set the size of the angular range that is occupied by the arc, expressed in degrees

to Answer the ending point of the arc in cartesian coordinates

to: aPoint Set the ending point of the arc in cartesian coordinates

1.3 BLOX.BBalloon

Defined in namespace BLOX
Superclass: BLOX.BEventSet
Category: Graphics-Examples
This event set allows a widget to show explanatory information when the mouse lingers over it for a while.

1.3.1 BLOX.BBalloon class: accessing

balloonDelayTime
Answer the time after which the balloon is shown (default is half a second).

balloonDelayTime: milliseconds
Set the time after which the balloon is shown.

shown Answer whether a balloon is displayed

1.3.2 BLOX.BBalloon: accessing

shown Answer whether the receiver's balloon is displayed

text Answer the text displayed in the balloon

text: aString
Set the text displayed in the balloon to aString

1.3.3 BLOX.BBalloon: initializing

initialize: aBWidget
Initialize the event sets for the receiver

1.4 BLOX.BBoundingBox

Defined in namespace BLOX
Superclass: BLOX.BCanvasObject
Category: Graphics-Windows
I am the ultimate ancestor of all items that you can put in a BCanvas and which are well defined by their bounding box - i.e. everything except BPolylines and BSplines.

1.4.1 BLOX.BBoundingBox: accessing

boundingBox
Answer a Rectangle enclosing all of the receiver

center Answer the center point of the receiver

center: center extent: extent
Move the object so that it is centered around the center Point and its size is given by the extent Point. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

corner Answer the Point specifying the lower-right corner of the receiver

corner: pointOrArray
Set the Point specifying the lower-right corner of the receiver; pointOrArray can be a Point or a two-item Array. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

extent Answer a Point specifying the size of the receiver

extent: pointOrArray
Set the Point specifying the size of the receiver; pointOrArray can be a Point or a two-item Array. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

moveBy: pointOrArray
Move the object by the amount indicated by pointOrArray: that is, its whole bounding box is shifted by that amount. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

origin Answer the Point specifying the top-left corner of the receiver

origin: pointOrArray
Set the Point specifying the top-left corner of the receiver; pointOrArray can be a Point or a two-item Array. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

origin: originPointOrArray corner: cornerPointOrArray
Set the bounding box of the object, based on a Point specifying the top-left corner of the receiver and another specifying the bottom-right corner; the two parameters can both be Points or two-item Arrays. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

origin: originPointOrArray extent: extentPointOrArray
Set the bounding box of the object, based on a Point specifying the top-left corner of the receiver and another specifying its size; the two parameters can both be Points or two-item Arrays. No changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

1.5 BLOX.BButton

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I am a button that a user can click. In fact I am at the head of a small hierarchy of objects which exhibit button-like look and behavior

1.5.1 BLOX.BButton class: instance creation

new: parent label: label
Answer a new BButton widget laid inside the given parent widget, showing by default the 'label' String.

1.5.2 BLOX.BButton: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

callback Answer a DirectedMessage that is sent when the receiver is clicked, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is clicked. If the method accepts an argument, the receiver is passed.

font Answer the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value
Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor
Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value
Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

invokeCallback
Generate a synthetic callback

label Answer the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

label: value
Set the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

1.6 BLOX.BButtonLike

Defined in namespace BLOX
Superclass: BLOX.BExtended
Category: Graphics-Examples
I am an object whose 3-D appearance resembles that of buttons.

1.6.1 BLOX.BButtonLike: accessing

callback Answer a DirectedMessage that is sent when the receiver is clicked, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is clicked. If the method accepts an argument, the receiver is passed.

invokeCallback
Generate a synthetic callback

pressed
This is the default callback for the widget; it does nothing if you don't override it. Of course if a subclass overriddes this you (user of the class) might desire to call this method from your own callback.

1.7 BLOX.BCanvas

Defined in namespace BLOX
Superclass: BLOX.BViewport
Category: Graphics-Windows

I am an host for whatever geometric shape you want. If you want to do some fancy graphics with Smalltalk, I'll be happy to help. My friends derived from BCanvasObject ask me all sort of things to do, so I am the real worker, not they!

BCanvasObject: I am BCanvas: No I am BCanvasObject: No I am BCanvas: No I am

well, you know, he always has something to object.

1.7.1 BLOX.BCanvas: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

foregroundColor
Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value
Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

1.7.2 BLOX.BCanvas: geometry management

addChild: child
The widget identified by child has been added to the receiver. This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to either the superclass implementation or #basicAddChild:, to perform some initialization on the children just added. Answer the new child.

child: child height: value
Set the given child's height.

child: child heightOffset: value
Offset the given child's height by value pixels.

child: child width: value
Set the given child's width.

child: child widthOffset: value
Offset the given child's width by value pixels.

child: child x: value
Set the given child's top-left corner's x coordinate, in pixels in the canvas' coordinate system.

child: child xOffset: value
Offset the given child's top-left x by value pixels.

child: child y: value
Set the given child's top-left corner's y coordinate, in pixels in the canvas' coordinate system.

child: child yOffset: value
Offset the given child's top-left y by value pixels.

heightChild: child
Answer the given child's height in pixels.

widthChild: child
Answer the given child's width in pixels.

xChild: child
Answer the given child's top-left corner's x coordinate, in pixels in the canvas' coordinate system.

yChild: child
Answer the given child's top-left corner's y coordinate, in pixels in the canvas' coordinate system.

1.7.3 BLOX.BCanvas: widget protocol

at: aPoint Selects the topmost item in the canvas overlapping the point given by aPoint.

between: origin and: corner do: aBlock
Evaluate aBlock for each item whose bounding box intersects the rectangle between the two Points, origin and corner. Pass the item to the block.

boundingBox
Answer the bounding box of all the items in the canvas

destroyed The widget has been destroyed. Tell all of its items about this fact.

do: aBlock
Evaluate aBlock, passing each item to it.

empty Remove all the items from the canvas, leaving it empty

extraSpace
Answer the amount of space that is left as a border around the canvas items.

extraSpace: aPoint
Set the amount of space that is left as a border around the canvas items.

items Answer an Array containing all the items in the canvas

mapPoint: aPoint
Given aPoint, a point expressed in window coordinates, answer the corresponding canvas coordinates that are displayed at that location.

1.8 BLOX.BCanvasObject

Defined in namespace BLOX
Superclass: BLOX.BEventTarget
Category: Graphics-Windows
I am the ultimate ancestor of all items that you can put in a BCanvas. I provide some general methods to my concrete offspring.

1.8.1 BLOX.BCanvasObject class: instance creation

new This method should not be called for instances of this class.

new: parentCanvas
Answer a new instance of the receiver, displayed into the given parentCanvas.

1.8.2 BLOX.BCanvasObject: accessing

blox Answer the parent canvas of the receiver

boundingBox
Answer a Rectangle enclosing all of the receiver

color Answer the color to be used to fill this item's area.

color: color
Set the color to be used to fill this item's area.

copyInto: newCanvas
Answer a new BCanvasObject identical to this but displayed into another canvas, newCanvas. The new instance is not created at the time it is returned.

copyObject
Answer a new BCanvasObject identical to this. Unlike #copy, which merely creates a new Smalltalk object with the same data and referring to the same canvas item, the object created with #copyObject is physically distinct from the original. The new instance is not created at the time it is returned.

createCopy
Answer a new BCanvasObject identical to this. Unlike #copy, which merely creates a new Smalltalk object with the same data and referring to the same canvas item, the object created with #copyObject is physically distinct from the original. The new instance has already been created at the time it is returned.

createCopyInto: newCanvas
Answer a new BCanvasObject identical to this but displayed into another canvas, newCanvas. The new instance has already been created at the time it is returned.

deepCopy It does not make sense to make a copy, because it would make data inconsistent across different objects; so answer the receiver

grayOut Apply a 50% gray stippling pattern to the object

shallowCopy
It does not make sense to make a copy, because it would make data inconsistent across different objects; so answer the receiver

1.8.3 BLOX.BCanvasObject: widget protocol

create If the object has not been created yet and has been initialized correctly, insert it for real in the parent canvas

created Answer whether the object is just a placeholder or has already been inserted for real in the parent canvas

lower Move the item to the lowest position in the display list. Child widgets always obscure other item types, and the stacking order of window items is determined by sending methods to the widget object directly.

raise Move the item to the highest position in the display list. Child widgets always obscure other item types, and the stacking order of window items is determined by sending methods to the widget object directly.

redraw Force the object to be displayed in the parent canvas, creating it if it has not been inserted for real in the parent, and refresh its position if it has changed.

remove Remove the object from the canvas

show Ensure that the object is visible in the center of the canvas, scrolling it if necessary.

1.9 BLOX.BCheckMenuItem

Defined in namespace BLOX
Superclass: BLOX.BMenuItem
Category: Graphics-Windows
I am a menu item which can be toggled between two states, marked and unmarked.

1.9.1 BLOX.BCheckMenuItem class: instance creation

new: parent
This method should not be called for instances of this class.

1.9.2 BLOX.BCheckMenuItem: accessing

invokeCallback
Generate a synthetic callback

value Answer whether the menu item is in a selected (checked) state.

value: aBoolean
Set whether the button is in a selected (checked) state and generates a callback accordingly.

1.10 BLOX.BColorButton

Defined in namespace BLOX
Superclass: BLOX.BButtonLike
Category: Graphics-Examples
I am a button that shows a color and that, unless a different callback is used, lets you choose a color when it is clicked.

1.10.1 BLOX.BColorButton: accessing

color Set the color that the receiver is painted in.

color: aString
Set the color that the receiver is painted in.

pressed This is the default callback; it brings up a 'choose-a-color' window and, if 'Ok' is pressed in the window, sets the receiver to be painted in the chosen color.

1.11 BLOX.BContainer

Defined in namespace BLOX
Superclass: BLOX.BForm
Category: Graphics-Windows
I am used to group many widgets together. I can perform simple management by putting widgets next to each other, from left to right or from top to bottom.

1.11.1 BLOX.BContainer: accessing

setVerticalLayout: aBoolean
Answer whether the container will align the widgets vertically or horizontally. Horizontal alignment means that widgets are packed from left to right, while vertical alignment means that widgets are packed from the top to the bottom of the widget.

Widgets that are set to be "stretched" will share all the space that is not allocated to non-stretched widgets.

The layout of the widget can only be set before the first child is inserted in the widget.

1.12 BLOX.BDialog

Defined in namespace BLOX
Superclass: BLOX.BForm
Category: Graphics-Windows

I am a facility for implementing dialogs with many possible choices and requests. In addition I provide support for a few platform native common dialog boxes, such as choose-a-file and choose-a-color.

1.12.1 BLOX.BDialog class: instance creation

new: parent

Answer a new dialog handler (containing a label widget and some button widgets) laid out within the given parent window. The label widget, when it is created, is empty.

new: parent label: aLabel

Answer a new dialog handler (containing a label widget and some button widgets) laid out within the given parent window. The label widget, when it is created, contains aLabel.

new: parent label: aLabel prompt: aString

Answer a new dialog handler (containing a label widget, some button widgets, and an edit window showing aString by default) laid out within the given parent window. The label widget, when it is created, contains aLabel.

1.12.2 BLOX.BDialog class: prompters

chooseColor: parent label: aLabel default: color

Prompt for a color. The dialog box is created with the given parent window and with aLabel as its title bar text, and initially it selects the color given in the color parameter.

If the dialog box is canceled, nil is answered, else the selected color is returned as a String with its RGB value.

chooseFileToOpen: parent label: aLabel default: name defaultExtension: ext types: typeList

Pop up a dialog box for the user to select a file to open. Its purpose is for the user to select an existing file only. If the user enters an non-existent file, the dialog box gives the user an error prompt and requires the user to give an alternative selection or to cancel the selection. If an application allows the user to create new files, it should do so by providing a separate New menu command.

If the dialog box is canceled, nil is answered, else the selected file name is returned as a String.

The dialog box is created with the given parent window and with aLabel as its title bar text. The name parameter indicates which file is initially selected, and the default extension specifies a string that will be appended to the filename if the user enters a filename without an extension.

The typeList parameter is an array of arrays, like #((’Text files’ ’.txt’ ’.diz’) (’Smalltalk files’ ’.st’)), and is used to construct a listbox of file types. When the user chooses a file type in the listbox, only the files of that type are listed. Each item in the array contains a list of strings: the first one is the name of the file type described by a particular file pattern, and is the text string that appears in the File types listbox, while the other ones are the possible extensions that belong to this particular file type.

chooseFileToSave: parent label: aLabel default: name defaultExtension: ext types: typeList

Pop up a dialog box for the user to select a file to save; this differs from the file open dialog box in that non-existent file names are accepted and existing file names trigger a confirmation dialog box, asking the user whether the file should be overwritten or not.

If the dialog box is canceled, nil is answered, else the selected file name is returned as a String.

The dialog box is created with the given parent window and with aLabel as its title bar text. The name parameter indicates which file is initially selected, and the default extension specifies a string that will be appended to the filename if the user enters a filename without an extension.

The typeList parameter is an array of arrays, like #(('Text files' '.txt' '.diz') ('Smalltalk files' '.st')), and is used to construct a listbox of file types. When the user chooses a file type in the listbox, only the files of that type are listed. Each item in the array contains a list of strings: the first one is the name of the file type described by a particular file pattern, and is the text string that appears in the File types listbox, while the other ones are the possible extensions that belong to this particular file type.

1.12.3 BLOX.BDialog: accessing

addButton: aLabel receiver: anObject index: anInt

Add a button to the dialog box that, when clicked, will cause the #dispatch: method to be triggered in anObject, passing anInt as the argument of the callback. The caption of the button is set to aLabel.

addButton: aLabel receiver: anObject message: aSymbol

Add a button to the dialog box that, when clicked, will cause the aSymbol unary selector to be sent to anObject. The caption of the button is set to aLabel.

addButton: aLabel receiver: anObject message: aSymbol argument: arg

Add a button to the dialog box that, when clicked, will cause the aSymbol one-argument selector to be sent to anObject, passing arg as the argument of the callback. The caption of the button is set to aLabel.

contents

Answer the text that is displayed in the entry widget associated to the dialog box.

contents: newText

Display newText in the entry widget associated to the dialog box.

1.12.4 BLOX.BDialog: widget protocol

center Center the dialog box's parent window in the screen

centerIn: view
Center the dialog box's parent window in the given widget

destroyed Private - The receiver has been destroyed, clear the corresponding Tcl variable to avoid memory leaks.

invokeCallback: index
Generate a synthetic callback corresponding to the index-th button being pressed, and destroy the parent window (triggering its callback if one was established).

loop Map the parent window modally. In other words, an event loop is started that ends only after the window has been destroyed. For more information on the treatment of events for modal windows, refer to BWindow>>#modalMap.

1.13 BLOX.BDropDown

Defined in namespace BLOX
Superclass: BLOX.BExtended
Category: Graphics-Examples
This class is an abstract superclass for widgets offering the ability to pick items from a pre-built list. The list is usually hidden, but a button on the right of this widgets makes it pop up. This widget is thus composed of three parts: an unspecified text widget (shown on the left of the button and always visible), the button widget (shown on the right, it depicts a down arrow, and is always visible), and the pop-up list widget.

1.13.1 BLOX.BDropDown: accessing

backgroundColor
Answer the value of the backgroundColor for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the normal background color to use when displaying the widget.

backgroundColor: aColor
Set the value of the backgroundColor for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the normal background color to use when displaying the widget.

droppedRows
Answer the number of items that are visible at any time in the listbox.

droppedRows: anInteger
Set the number of items that are visible at any time in the listbox.

font
Answer the value of the font option for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value
Set the value of the font option for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor
Answer the value of the foregroundColor for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: aColor
Set the value of the foregroundColor for the widget, which in this class is only set for the list widget (that is, the pop-up widget). Subclasses should override this method so that the color is set properly for the text widget as well.

Specifies the normal foreground color to use when displaying the widget.

highlightBackground
Answer the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the list widget.

highlightBackground: aColor
Set the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the list widget.

highlightForeground
Answer the value of the highlightForeground option for the widget.

Specifies the foreground color to use when displaying selected items in the list widget.

highlightForeground: aColor
Set the value of the highlightForeground option for the widget.

Specifies the foreground color to use when displaying selected items in the list widget.

1.13.2 BLOX.BDropDown: callbacks

callback Answer a DirectedMessage that is sent when the receiver is clicked, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is clicked. If the method accepts an argument, the receiver is passed.

invokeCallback
Generate a synthetic callback

1.13.3 BLOX.BDropDown: flexibility

createList Create the popup widget to be used for the ‘drop-down list’. It is a BList by default, but you can use any other widget, overriding the ‘list box accessing’ methods if necessary.

createTextWidget
Create the widget that will hold the string chosen from the list box and answer it. The widget must be a child of ‘self primitive’.

itemHeight
Answer the height of an item in the drop-down list. The default implementation assumes that the receiver understands #font, but you can modify it if you want.

listCallback
Called when an item of the listbox is highlighted. Do nothing by default

listSelectAt: aPoint
: Select the item lying at the given position in the list box. The default implementation assumes that list is a BList, but you can modify it if you want.

listText
: Answer the text currently chosen in the list box. The default implementation assumes that list is a BList, but you can modify it if you want.

text
: Answer the text that the user has picked from the widget and/or typed in the control (the exact way the text is entered will be established by subclasses, since this is an abstract method).

text: aString
: Set the text widget to aString

1.13.4 BLOX.BDropDown: list box accessing

add: anObject afterIndex: index
: Add an element with the given value after another element whose index is contained in the index parameter. The label displayed in the widget is anObject's displayString. Answer anObject.

add: aString element: anObject afterIndex: index
: Add an element with the aString label after another element whose index is contained in the index parameter. This method allows the client to decide autonomously the label that the widget will display.

  If anObject is nil, then string is used as the element as well. If aString is nil, then the element's displayString is used as the label.

  Answer anObject or, if it is nil, aString.

addLast: anObject
: Add an element with the given value at the end of the listbox. The label displayed in the widget is anObject's displayString. Answer anObject.

addLast: aString element: anObject
: Add an element with the given value at the end of the listbox. This method allows the client to decide autonomously the label that the widget will display.

  If anObject is nil, then string is used as the element as well. If aString is nil, then the element's displayString is used as the label.

  Answer anObject or, if it is nil, aString.

associationAt: anIndex
: Answer an association whose key is the item at the given position in the listbox and whose value is the label used to display that item.

at: anIndex
: Answer the element displayed at the given position in the list box.

contents: stringCollection
: Set the elements displayed in the listbox, and set the labels to be their displayStrings.

contents: stringCollection elements: elementList
Set the elements displayed in the listbox to be those in elementList, and set the labels to be the corresponding elements in stringCollection. The two collections must have the same size.

do: aBlock
Iterate over each element of the listbox and pass it to aBlock.

elements: elementList
Set the elements displayed in the listbox, and set the labels to be their displayStrings.

index: newIndex
Highlight the item at the given position in the listbox, and transfer the text in the list box to the text widget.

labelAt: anIndex
Answer the label displayed at the given position in the list box.

labelsDo: aBlock
Iterate over the labels in the list widget and pass each of them to aBlock.

numberOfStrings
Answer the number of items in the list box

removeAtIndex: index
Remove the item at the given index in the list box, answering the object associated to the element (i.e. the value that #at: would have returned for the given index)

size Answer the number of items in the list box

1.13.5 BLOX.BDropDown: widget protocol

dropRectangle
Answer the rectangle in which the list widget will pop-up. If possible, this is situated below the drop-down widget's bottom side, but if the screen space there is not enough it could be above the drop-down widget's above side. If there is no screen space above as well, we pick the side where we can offer the greatest number of lines in the pop-up widget.

dropdown Force the pop-up list widget to be visible.

isDropdownVisible
Answer whether the pop-up widget is visible

toggle Toggle the visibility of the pop-up widget.

unmapList
Unmap the pop-up widget from the screen, transfer its selected item to the always visible text widget, and generate a callback.

1.14 BLOX.BDropDownEdit

Defined in namespace BLOX
Superclass: BLOX.BDropDown
Category: Graphics-Examples

This class resembles an edit widget, but it has an arrow button that allows the user to pick an item from a pre-built list.

1.14.1 BLOX.BDropDownEdit: accessing

backgroundColor: aColor

Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

font: aString

Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor: aColor

Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

highlightBackground: aColor

Set the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the list widget and the selection in the text widget.

highlightForeground: aColor

Set the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the list widget and the selection in the text widget.

1.14.2 BLOX.BDropDownEdit: accessing-overrides

text Answer the text shown in the widget

1.14.3 BLOX.BDropDownEdit: text accessing

insertAtEnd: aString
Clear the selection and append aString at the end of the text widget.

replaceSelection: aString
Insert aString in the text widget at the current insertion point, replacing the currently selected text (if any), and leaving the text selected.

selectAll
Select the whole contents of the text widget

selectFrom: first to: last
Sets the selection of the text widget to include the characters starting with the one indexed by first (the very first character in the widget having index 1) and ending with the one just before last. If last refers to the same character as first or an earlier one, then the text widget's selection is cleared.

selection
Answer an empty string if the text widget has no selection, else answer the currently selected text

selectionRange
Answer nil if the text widget has no selection, else answer an Interval object whose first item is the index of the first character in the selection, and whose last item is the index of the character just after the last one in the selection.

text: aString
Set the contents of the text widget and select them.

1.15 BLOX.BDropDownList

Defined in namespace BLOX
Superclass: BLOX.BDropDown
Category: Graphics-Examples
This class resembles a list box widget, but its actual list shows up only when you click the arrow button beside the currently selected item.

1.15.1 BLOX.BDropDownList: accessing

backgroundColor: aColor
Set the value of the backgroundColor for the widget, which in this class is set for the list widget and, when the focus is outside the control, for the text widget as well.

Specifies the normal background color to use when displaying the widget.

font: aString
Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are "Helvetica 10 Bold", "Times -14", "Futura Bold Underline". You must enclose the font family in braces if it is made of two or more words.

foregroundColor: aColor

Set the value of the foregroundColor for the widget, which in this class is set for the list widget and, when the focus is outside the control, for the text widget as well.

Specifies the normal foreground color to use when displaying the widget.

highlightBackground: aColor

Answer the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the list widget and, when the focus is inside the control, for the text widget as well.

highlightForeground: aColor

Answer the value of the highlightForeground option for the widget.

Specifies the foreground color to use when displaying selected items in the list widget and, when the focus is inside the control, for the text widget as well.

text

Answer the text that the user has picked from the widget and/or typed in the control (the exact way the text is entered will be established by subclasses, since this is an abstract method).

1.15.2 BLOX.BDropDownList: callbacks

callback: aReceiver message: aSymbol

Set up so that aReceiver is sent the aSymbol message (the name of a selector with at most two arguemtnts) when the active item in the receiver changegs. If the method accepts two arguments, the receiver is passed as the first parameter. If the method accepts one or two arguments, the selected index is passed as the last parameter.

invokeCallback

Generate a synthetic callback.

1.15.3 BLOX.BDropDownList: list box accessing

index

Answer the value of the index option for the widget. Since it is not possible to modify an item once it has been picked from the list widget, this is always defined for BDropDownList widgets.

1.16 BLOX.BEdit

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I am a widget showing one line of modifiable text.

1.16.1 BLOX.BEdit class: instance creation

new: parent contents: aString
Answer a new BEdit widget laid inside the given parent widget, with a default content of aString

1.16.2 BLOX.BEdit: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

callback
Answer a DirectedMessage that is sent when the receiver is modified, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is modified. If the method accepts an argument, the receiver is passed.

contents
Return the contents of the widget

contents: newText
Set the contents of the widget

font
Answer the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value
: Set the value of the font option for the widget.

  Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

  X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

  Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are "Helvetica 10 Bold", "Times -14", "Futura Bold Underline". You must enclose the font family in braces if it is made of two or more words.

foregroundColor
: Answer the value of the foregroundColor option for the widget.

  Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value
: Set the value of the foregroundColor option for the widget.

  Specifies the normal foreground color to use when displaying the widget.

selectBackground
: Answer the value of the selectBackground option for the widget.

  Specifies the background color to use when displaying selected parts of the widget.

selectBackground: value
: Set the value of the selectBackground option for the widget.

  Specifies the background color to use when displaying selected parts of the widget.

selectForeground
: Answer the value of the selectForeground option for the widget.

  Specifies the foreground color to use when displaying selected parts of the widget.

selectForeground: value
: Set the value of the selectForeground option for the widget.

  Specifies the foreground color to use when displaying selected parts of the widget.

1.16.3 BLOX.BEdit: widget protocol

destroyed Private - The receiver has been destroyed, clear the corresponding Tcl variable to avoid memory leaks.

hasSelection
Answer whether there is selected text in the widget

insertAtEnd: aString
Clear the selection and append aString at the end of the widget.

insertText: aString
Insert aString in the widget at the current insertion point, replacing the currently selected text (if any).

invokeCallback
Generate a synthetic callback.

nextPut: aCharacter
Clear the selection and append aCharacter at the end of the widget.

nextPutAll: aString
Clear the selection and append aString at the end of the widget.

nl Clear the selection and append a linefeed character at the end of the widget.

replaceSelection: aString
Insert aString in the widget at the current insertion point, replacing the currently selected text (if any), and leaving the text selected.

selectAll Select the whole contents of the widget.

selectFrom: first to: last
Sets the selection to include the characters starting with the one indexed by first (the very first character in the widget having index 1) and ending with the one just before last. If last refers to the same character as first or an earlier one, then the widget's selection is cleared.

selection Answer an empty string if the widget has no selection, else answer the currently selected text

selectionRange
Answer nil if the widget has no selection, else answer an Interval object whose first item is the index of the first character in the selection, and whose last item is the index of the character just after the last one in the selection.

space Clear the selection and append a space at the end of the widget.

1.17 BLOX.BEmbeddedImage

Defined in namespace BLOX
Superclass: BLOX.BBoundingBox
Category: Graphics-Windows
I can draw a colorful image inside the canvas.

1.17.1 BLOX.BEmbeddedImage: accessing

copyInto: aBlox
Answer a new BCanvasObject identical to this but displayed into another canvas, newCanvas. The new instance is not created at the time it is returned.

data Answer the data of the image. The result will be a String containing image data either as Base-64 encoded GIF data, as XPM data, or as PPM data.

data: aString
Set the data of the image. aString may contain the data either as Base-64 encoded GIF data, as XPM data, or as PPM data. No changes are visible until you toggle a redraw using the appropriate method.

redraw Force the object to be displayed in the parent canvas, creating it if it has not been inserted for real in the parent, and refresh its position and image data if it has changed.

1.18 BLOX.BEmbeddedText

Defined in namespace BLOX
Superclass: BLOX.BBoundingBox
Category: Graphics-Windows
I can draw text in all sorts of colors, sizes and fonts.

1.18.1 BLOX.BEmbeddedText: accessing

font Answer the value of the font option for the canvas object.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: font Set the value of the font option for the canvas object.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

justify Answer how to justify the text within its bounding region.

justify: aSymbol
Sets how to justify the text within its bounding region. Can be #left, #right or #center (the default).

redraw Force the object to be displayed in the parent canvas, creating it if it has not been inserted for real in the parent, and refresh its position.

text Answer the text that is printed by the object

text: aString
Set the text that is printed by the object

1.19 BLOX.BEventSet

Defined in namespace BLOX
Superclass: BLOX.BEventTarget
Category: Graphics-Windows
I combine event handlers and let you apply them to many objects. Basically, you derive a class from me, override the #initialize: method to establish the handlers, then use the #addEventSet: method understood by every Blox class to add the event handlers specified by the receiver to the object.

1.19.1 BLOX.BEventSet class: initializing

new This method should not be called for instances of this class.

new: widget
Private - Create a new event set object that will attach to the given widget. Answer the object. Note: this method should be called by #addEventSet:, not directly

1.19.2 BLOX.BEventSet: accessing

widget Answer the widget to which the receiver is attached.

1.19.3 BLOX.BEventSet: initializing

initialize: aBWidget
Initialize the receiver’s event handlers to attach to aBWidget. You can override this of course, but don’t forget to call the superclass implementation first.

1.20 BLOX.BEventTarget

Defined in namespace BLOX
Superclass: Object
Category: Graphics-Windows
I track all the event handling procedures that you apply to an object.

1.20.1 BLOX.BEventTarget: intercepting events

addEventSet: aBEventSetSublass
: Add to the receiver the event handlers implemented by an instance of aBEventSetSubclass. Answer the new instance of aBEventSetSublass.

onAsciiKeyEventSend: aSelector to: anObject
: When an ASCII key is pressed and the receiver has the focus, send the 1-argument message identified by aSelector to anObject, passing to it a Character.

onDestroySend: aSelector to: anObject
: When the receiver is destroyed, send the unary message identified by aSelector to anObject.

onFocusEnterEventSend: aSelector to: anObject
: When the focus enters the receiver, send the unary message identified by aSelector to anObject.

onFocusLeaveEventSend: aSelector to: anObject
: When the focus leaves the receiver, send the unary message identified by aSelector to anObject.

onKeyEvent: key send: aSelector to: anObject
: When the given key is pressed and the receiver has the focus, send the unary message identified by aSelector to anObject. Examples for key are: 'Ctrl-1', 'Alt-X', 'Meta-plus', 'enter'. The last two cases include example of special key identifiers; these include: 'backslash', 'exclam', 'quotedbl', 'dollar', 'asterisk', 'less', 'greater', 'asciicircum' (caret), 'question', 'equal', 'parenleft', 'parenright', 'colon', 'semicolon', 'bar' (pipe sign), 'underscore', 'percent', 'minus', 'plus', 'BackSpace', 'Delete', 'Insert', 'Return', 'End', 'Home', 'Prior' (Pgup), 'Next' (Pgdn), 'F1'..'F24', 'Caps_Lock', 'Num_Lock', 'Tab', 'Left', 'Right', 'Up', 'Down'. There are in addition four special identifiers which map to platform-specific keys: '<Cut>', '<Copy>', '<Paste>', '<Clear>' (all with the angular brackets!).

onKeyEventSend: aSelector to: anObject
: When a key is pressed and the receiver has the focus, send the 1-argument message identified by aSelector to anObject. The pressed key will be passed as a String parameter; some of the keys will send special key identifiers such as those explained in the documentation for #onKeyEvent:send:to: Look at the #eventTest test program in the BloxTestSuite to find out the parameters passed to such an event procedure

onKeyUpEventSend: aSelector to: anObject
: When a key has been released and the receiver has the focus, send the 1-argument message identified by aSelector to anObject. The released key will be passed as a String parameter; some of the keys will send special key identifiers such as those explained in the documentation for #onKeyEvent:send:to: Look at the #eventTest test program in the BloxTestSuite to find out the parameters passed to such an event procedure

onMouseDoubleEvent: button send: aSelector to: anObject
When the given button is double-clicked on the mouse, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseDoubleEventSend: aSelector to: anObject
When a button is double-clicked on the mouse, send the 2-argument message identified by aSelector to anObject. The mouse position will be passed as a Point in the first parameter, the button number will be passed as an Integer in the second parameter.

onMouseDownEvent: button send: aSelector to: anObject
When the given button is pressed on the mouse, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseDownEventSend: aSelector to: anObject
When a button is pressed on the mouse, send the 2-argument message identified by aSelector to anObject. The mouse position will be passed as a Point in the first parameter, the button number will be passed as an Integer in the second parameter.

onMouseEnterEventSend: aSelector to: anObject
When the mouse enters the widget, send the unary message identified by aSelector to anObject.

onMouseLeaveEventSend: aSelector to: anObject
When the mouse leaves the widget, send the unary message identified by aSelector to anObject.

onMouseMoveEvent: button send: aSelector to: anObject
When the mouse is moved while the given button is pressed on the mouse, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseMoveEventSend: aSelector to: anObject
When the mouse is moved, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseTripleEvent: button send: aSelector to: anObject
When the given button is triple-clicked on the mouse, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseTripleEventSend: aSelector to: anObject
When a button is triple-clicked on the mouse, send the 2-argument message identified by aSelector to anObject. The mouse position will be passed as a Point in the first parameter, the button number will be passed as an Integer in the second parameter.

onMouseUpEvent: button send: aSelector to: anObject
When the given button is released on the mouse, send the 1-argument message identified by aSelector to anObject. The mouse position will be passed as a Point.

onMouseUpEventSend: aSelector to: anObject
When a button is released on the mouse, send the 2-argument message identified by aSelector to anObject. The mouse position will be passed as a Point in the first parameter, the button number will be passed as an Integer in the second parameter.

onResizeSend: aSelector to: anObject
When the receiver is resized, send the 1-argument message identified by aSelector to anObject. The new size will be passed as a Point.

1.21 BLOX.BExtended

Defined in namespace BLOX
Superclass: BLOX.BWidget
Category: Graphics-Windows
Just like Gui, I serve as a base for complex objects which expose an individual protocol but internally use a Blox widget for creating their user interface. Unlike Gui, however, the instances of my subclasses understand the standard widget protocol. Just override my newPrimitive method to return another widget, and you'll get a class which interacts with the user like that widget (a list box, a text box, or even a label) but exposes a different protocol.

1.21.1 BLOX.BExtended: accessing

asPrimitiveWidget
Answer the primitive widget that implements the receiver.

1.21.2 BLOX.BExtended: customization

create After this method is called (the call is made automatically) the receiver will be attached to a 'primitive' widget (which can be in turn another extended widget). This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to super (which only calls #newPrimitive and saves the result), to perform some initialization on the primitive widget just created; overriding #create is in fact more generic than overriding #newPrimitive. For an example of this, see the implementation of BButtonLike.

newPrimitive
Create and answer a new widget on which the implementation of the receiver will be based. You should not call this method directly; instead you must override it in BExtended's subclasses.

1.22 BLOX.BForm

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I am used to group many widgets together. I leave the heavy task of managing their position to the user.

1.22.1 BLOX.BForm: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

defaultHeight
Answer the value of the defaultHeight option for the widget.

Specifies the desired height for the form in pixels. If this option is less than or equal to zero then the window will not request any size at all.

defaultHeight: value
Set the value of the defaultHeight option for the widget.

Specifies the desired height for the form in pixels. If this option is less than or equal to zero then the window will not request any size at all.

defaultWidth
Answer the value of the defaultWidth option for the widget.

Specifies the desired width for the form in pixels. If this option is less than or equal to zero then the window will not request any size at all.

defaultWidth: value
Set the value of the defaultWidth option for the widget.

Specifies the desired width for the form in pixels. If this option is less than or equal to zero then the window will not request any size at all.

1.23 BLOX.BImage

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I can display colorful images.

1.23.1 BLOX.BImage class: arrows

downArrow
Answer the XPM representation of a 12x12 arrow pointing downwards.

leftArrow Answer the XPM representation of a 12x12 arrow pointing leftwards.

rightArrow
Answer the XPM representation of a 12x12 arrow pointing rightwards.

upArrow Answer the XPM representation of a 12x12 arrow pointing upwards.

1.23.2 BLOX.BImage class: GNU

gnu Answer the XPM representation of a 48x48 GNU.

1.23.3 BLOX.BImage class: icons

exclaim Answer the XPM representation of a 32x32 exclamation mark icon.

info Answer the XPM representation of a 32x32 ‘information’ icon.

question Answer the XPM representation of a 32x32 question mark icon.

stop Answer the XPM representation of a 32x32 ‘critical stop’ icon.

1.23.4 BLOX.BImage class: instance creation

new: parent data: aString
Answer a new BImage widget laid inside the given parent widget, loading data from the given string (Base-64 encoded GIF, XPM, PPM are supported).

new: parent image: aFileStream
Answer a new BImage widget laid inside the given parent widget, loading data from the given file (GIF, XPM, PPM are supported).

new: parent size: aPoint
Answer a new BImage widget laid inside the given parent widget, showing by default a transparent image of aPoint size.

1.23.5 BLOX.BImage class: small icons

directory Answer the Base-64 GIF representation of a ‘directory folder’ icon.

file Answer the Base-64 GIF representation of a ‘file’ icon.

1.23.6 BLOX.BImage: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

displayHeight
Answer the value of the displayHeight option for the widget.

Specifies the height of the image in pixels. This is not the height of the widget, but specifies the area of the widget that will be taken by the image.

displayHeight: value
Set the value of the displayHeight option for the widget.

Specifies the height of the image in pixels. This is not the height of the widget, but specifies the area of the widget that will be taken by the image.

displayWidth
Answer the value of the displayWidth option for the widget.

Specifies the width of the image in pixels. This is not the width of the widget, but specifies the area of the widget that will be taken by the image.

displayWidth: value
Set the value of the displayWidth option for the widget.

Specifies the width of the image in pixels. This is not the width of the widget, but specifies the area of the widget that will be taken by the image.

foregroundColor
Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value
Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

gamma
Answer the value of the gamma option for the widget.

Specifies that the colors allocated for displaying the image widget should be corrected for a non-linear display with the specified gamma exponent value. (The intensity produced by most CRT displays is a power function of the input value, to a good approximation; gamma is the exponent and is typically around 2). The value specified must be greater than zero. The default value is one (no correction). In general, values greater than one will make the image lighter, and values less than one will make it darker.

gamma: value
Set the value of the gamma option for the widget.

Specifies that the colors allocated for displaying the image widget should be corrected for a non-linear display with the specified gamma exponent value. (The intensity produced by most CRT displays is a power function of the input value, to a good approximation; gamma is the exponent and is typically around 2). The value specified must be greater than zero. The default value is one (no correction). In general, values greater than one will make the image lighter, and values less than one will make it darker.

1.23.7 BLOX.BImage: image management

blank
Blank the corresponding image

data: aString
Set the image to be drawn to aString, which can be a GIF in Base-64 representation or an X pixelmap.

dither Recalculate the dithered image in the window where the image is displayed. The dithering algorithm used in displaying images propagates quantization errors from one pixel to its neighbors. If the image data is supplied in pieces, the dithered image may not be exactly correct. Normally the difference is not noticeable, but if it is a problem, this command can be used to fix it.

fillFrom: origin extent: extent color: color
Fill a rectangle with the given origin and extent, using the given color.

fillFrom: origin to: corner color: color
Fill a rectangle between the given corners, using the given color.

fillRectangle: rectangle color: color
Fill a rectangle having the given bounding box, using the given color.

image: aFileStream
Read a GIF or XPM image from aFileStream. The whole contents of the file are read, not only from the file position.

imageHeight
Specifies the height of the image, in pixels. This option is useful primarily in situations where you wish to build up the contents of the image piece by piece. A value of zero (the default) allows the image to expand or shrink vertically to fit the data stored in it.

imageWidth
Specifies the width of the image, in pixels. This option is useful primarily in situations where you wish to build up the contents of the image piece by piece. A value of zero (the default) allows the image to expand or shrink horizontally to fit the data stored in it.

lineFrom: origin extent: extent color: color
Draw a line with the given origin and extent, using the given color.

lineFrom: origin to: corner color: color
This method's functionality has not been implemented yet.

lineFrom: origin toX: endX color: color
Draw an horizontal line between the given corners, using the given color.

lineFrom: origin toY: endY color: color
Draw a vertical line between the given corners, using the given color.

lineInside: rectangle color: color
Draw a line having the given bounding box, using the given color.

1.23.8 BLOX.BImage: widget protocol

destroyed Private - The receiver has been destroyed, clear the corresponding Tcl image to avoid memory leaks.

1.24 BLOX.BLabel

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I am a label showing static text.

1.24.1 BLOX.BLabel class: initialization

initialize Private - Initialize the receiver's class variables.

1.24.2 BLOX.BLabel class: instance creation

new: parent label: label
Answer a new BLabel widget laid inside the given parent widget, showing by default the 'label' String.

1.24.3 BLOX.BLabel: accessing

alignment Answer the value of the anchor option for the widget.

Specifies how the information in a widget (e.g. text or a bitmap) is to be displayed in the widget. Must be one of the symbols #topLeft, #topCenter, #topRight, #leftCenter, #center, #rightCenter, #bottomLeft, #bottomCenter, #bottomRight. For example, #topLeft means display the information such that its top-left corner is at the top-left corner of the widget.

alignment: aSymbol
Set the value of the anchor option for the widget.

Specifies how the information in a widget (e.g. text or a bitmap) is to be displayed in the widget. Must be one of the symbols #topLeft, #topCenter, #topRight, #leftCenter, #center, #rightCenter, #bottomLeft, #bottomCenter, #bottomRight. For example, #topLeft means display the information such that its top-left corner is at the top-left corner of the widget.

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

font Answer the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value

Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor

Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value

Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

label

Answer the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

label: value

Set the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

1.25 BLOX.BLine

Defined in namespace BLOX
Superclass: BLOX.BBoundingBox
Category: Graphics-Windows

I only draw straight lines but I can do that very well, even without a ruler...

1.25.1 BLOX.BLine: accessing

cap
Answer the way in which caps are to be drawn at the endpoints of the line. The answer may be #butt (the default), #projecting, or #round).

cap: aSymbol
Set the way in which caps are to be drawn at the endpoints of the line. aSymbol may be #butt (the default), #projecting, or #round).

width
Answer the width with which the line is drawn.

width: pixels
Set the width with which the line is drawn.

1.26 BLOX.BList

Defined in namespace BLOX
Superclass: BLOX.BViewport
Category: Graphics-Windows

I represent a list box from which you can choose one or more elements.

1.26.1 BLOX.BList: accessing

add: anObject afterIndex: index
Add an element with the given value after another element whose index is contained in the index parameter. The label displayed in the widget is anObject's displayString. Answer anObject.

add: aString element: anObject afterIndex: index
Add an element with the aString label after another element whose index is contained in the index parameter. This method allows the client to decide autonomously the label that the widget will display.

If anObject is nil, then string is used as the element as well. If aString is nil, then the element's displayString is used as the label.

Answer anObject or, if it is nil, aString.

addLast: anObject
Add an element with the given value at the end of the listbox. The label displayed in the widget is anObject's displayString. Answer anObject.

addLast: aString element: anObject
Add an element with the given value at the end of the listbox. This method allows the client to decide autonomously the label that the widget will display.

If anObject is nil, then string is used as the element as well. If aString is nil, then the element's displayString is used as the label.

Answer anObject or, if it is nil, aString.

associationAt: anIndex
Answer an association whose key is the item at the given position in the listbox and whose value is the label used to display that item.

at: anIndex
Answer the element displayed at the given position in the list box.

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

contents: elementList
Set the elements displayed in the listbox, and set the labels to be their displayStrings.

contents: stringCollection elements: elementList
Set the elements displayed in the listbox to be those in elementList, and set the labels to be the corresponding elements in stringCollection. The two collections must have the same size.

do: aBlock
Iterate over each element of the listbox and pass it to aBlock.

elements Answer the collection of objects that represent the elements displayed by the list box.

elements: elementList
Set the elements displayed in the listbox, and set the labels to be their displayStrings.

font Answer the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value

Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor

Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value

Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

highlightBackground

Answer the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the widget.

highlightBackground: value

Set the value of the highlightBackground option for the widget.

Specifies the background color to use when displaying selected items in the widget.

highlightForeground

Answer the value of the highlightForeground option for the widget.

Specifies the foreground color to use when displaying selected items in the widget.

highlightForeground: value

Set the value of the highlightForeground option for the widget.

Specifies the foreground color to use when displaying selected items in the widget.

index

Answer the value of the index option for the widget.

Indicates the element that has the location cursor. This item will be displayed in the highlightForeground color, and with the corresponding background color.

indexAt: point
Answer the index of the element that covers the point in the listbox window specified by x and y (in pixel coordinates). If no element covers that point, then the closest element to that point is used.

isSelected: index
Answer whether the element indicated by index is currently selected.

label Return nil, it is here for Gtk+ support

label: aString
Do nothing, it is here for Gtk+ support

labelAt: anIndex
Answer the label displayed at the given position in the list box.

labels Answer the labels displayed by the list box.

labelsDo: aBlock
Iterate over each listbox element's label and pass it to aBlock.

mode Answer the value of the mode option for the widget.

Specifies one of several styles for manipulating the selection. The value of the option may be either single, browse, multiple, or extended.

If the selection mode is single or browse, at most one element can be selected in the listbox at once. Clicking button 1 on an unselected element selects it and deselects any other selected item, while clicking on a selected element has no effect. In browse mode it is also possible to drag the selection with button 1. That is, moving the mouse while button 1 is pressed keeps the item under the cursor selected.

If the selection mode is multiple or extended, any number of elements may be selected at once, including discontiguous ranges. In multiple mode, clicking button 1 on an element toggles its selection state without affecting any other elements. In extended mode, pressing button 1 on an element selects it, deselects everything else, and sets the anchor to the element under the mouse; dragging the mouse with button 1 down extends the selection to include all the elements between the anchor and the element under the mouse, inclusive.

In extended mode, the selected range can be adjusted by pressing button 1 with the Shift key down: this modifies the selection to consist of the elements between the anchor and the element under the mouse, inclusive. The un-anchored end of this new selection can also be dragged with the button down. Also in extended mode, pressing button 1 with the Control key down starts a toggle operation: the anchor is set to the element under the mouse, and its selection state is reversed. The selection state of other elements is not changed. If the mouse is dragged with button 1 down, then the selection state of all elements between the anchor and the element under the mouse is set to match that of the anchor element; the selection state of all other elements remains what it was before the toggle operation began.

Most people will probably want to use browse mode for single selections and extended mode for multiple selections; the other modes appear to be useful only in special situations.

mode: value

Set the value of the mode option for the widget.

Specifies one of several styles for manipulating the selection. The value of the option may be either single, browse, multiple, or extended.

If the selection mode is single or browse, at most one element can be selected in the listbox at once. Clicking button 1 on an unselected element selects it and deselects any other selected item, while clicking on a selected element has no effect. In browse mode it is also possible to drag the selection with button 1. That is, moving the mouse while button 1 is pressed keeps the item under the cursor selected.

If the selection mode is multiple or extended, any number of elements may be selected at once, including discontiguous ranges. In multiple mode, clicking button 1 on an element toggles its selection state without affecting any other elements. In extended mode, pressing button 1 on an element selects it, deselects everything else, and sets the anchor to the element under the mouse; dragging the mouse with button 1 down extends the selection to include all the elements between the anchor and the element under the mouse, inclusive.

In extended mode, the selected range can be adjusted by pressing button 1 with the Shift key down: this modifies the selection to consist of the elements between the anchor and the element under the mouse, inclusive. The un-anchored end of this new selection can also be dragged with the button down. Also in extended mode, pressing button 1 with the Control key down starts a toggle operation: the anchor is set to the element under the mouse, and its selection state is reversed. The selection state of other elements is not changed. If the mouse is dragged with button 1 down, then the selection state of all elements between the anchor and the element under the mouse is set to match that of the anchor element; the selection state of all other elements remains what it was before the toggle operation began.

Most people will probably want to use browse mode for single selections and extended mode for multiple selections; the other modes appear to be useful only in special situations.

numberOfStrings

Answer the number of items in the list box

removeAtIndex: index

Remove the item at the given index in the list box, answering the object associated to the element (i.e. the value that #at: would have returned for the given index)

size Answer the number of items in the list box

1.26.2 BLOX.BList: widget protocol

callback Answer a DirectedMessage that is sent when the active item in the receiver changes, or nil if none has been set up.

callback: aReceiver message: aSymbol
: Set up so that aReceiver is sent the aSymbol message (the name of a selector with at most two arguemtnts) when the active item in the receiver changegs. If the method accepts two arguments, the receiver is passed as the first parameter. If the method accepts one or two arguments, the selected index is passed as the last parameter.

highlight: index
: Highlight the item at the given position in the listbox.

invokeCallback
: Generate a synthetic callback.

select: index
: Highlight the item at the given position in the listbox, without unhighlighting other items. This is meant for multiple- or extended-mode listboxes, but can be used with other selection mode in particular cases.

show: index
: Ensure that the item at the given position in the listbox is visible.

unhighlight
: Unhighlight all the items in the listbox.

unselect: index
: Unhighlight the item at the given position in the listbox, without affecting the state of the other items.

1.27 BLOX.Blox

Defined in namespace BLOX
Superclass: BLOX.BEventTarget
Category: Graphics-Windows

I am the superclass for every visible user interface object (excluding canvas items, which are pretty different). I provide common methods and a simple Tcl interface for internal use. In addition, I expose class methods that do many interesting event-handling things.

NOTE: some of the methods (notably geometry methods) may not be suitable for all Blox subclasses and may be included only for backwards compatibility towards 1.1.5 BLOX. You should use geometry methods only for subclasses of BWidget.

1.27.1 BLOX.Blox class: C call-outs

evalIn: interp tcl: cmd
: Not commented.

idle
: Not commented.

resultIn: interp
: Not commented.

tclInit
: Not commented.

1.27.2 BLOX.Blox class: event dispatching

dispatchEvents
: If this is the outermost dispatching loop that is started, dispatch events until the number of calls to #terminateMainLoop balances the number of calls to #dispatchEvents; return instantly if this is not the outermost dispatching loop that is started.

dispatchEvents: mainWindow
: Dispatch some events; return upon destruction of the 'mainWindow' widget (which can be any kind of BWidget, but will be typically a BWindow).

terminateMainLoop
: Terminate the event dispatching loop if this call to #terminateMainLoop balances the number of calls to #dispatchEvents.

update: aspect
: Initialize the Tcl and Blox environments; executed automatically on startup.

1.27.3 BLOX.Blox class: instance creation

new This method should not be called for instances of this class.

new: parent
: Create a new widget of the type identified by the receiver, inside the given parent widget. Answer the new widget

1.27.4 BLOX.Blox class: utility

active Answer the currently active Blox, or nil if the focus does not belong to a Smalltalk window.

at: aPoint Answer the Blox containing the given point on the screen, or nil if no Blox contains the given point (either because no Smalltalk window is there or because it is covered by another window).

atMouse Answer the Blox under the mouse cursor's hot spot, or nil if no Blox contains the given point (either because no Smalltalk window is there or because it is covered by another window).

beep Produce a bell

clearClipboard
: Clear the clipboard, answer its old contents.

clipboard Retrieve the text in the clipboard.

clipboard: aString
: Set the contents of the clipboard to aString (or empty the clipboard if aString is nil).

createColor: red green: green blue: blue
: Answer a color that can be passed to methods such as 'backgroundColor:'. The color will have the given RGB components (range is 0~65535).

createColor: cyan magenta: magenta yellow: yellow
: Answer a color that can be passed to methods such as ‘backgroundColor:’. The color will have the given CMY components (range is 0~65535).

createColor: cyan magenta: magenta yellow: yellow black: black
: Answer a color that can be passed to methods such as ‘backgroundColor:’. The color will have the given CMYK components (range is 0~65535).

createColor: hue saturation: sat value: value
: Answer a color that can be passed to methods such as ‘backgroundColor:’. The color will have the given HSV components (range is 0~65535).

defaultFont
: Answer the default font used by Blox.

fonts
: Answer the names of the font families in the system. Additionally, ‘Times’, ‘Courier’ and ‘Helvetica’ are always made available.

mousePointer
: If the mouse pointer is on the same screen as the application’s windows, returns a Point containing the pointer’s x and y coordinates measured in pixels in the screen’s root window (under X, if a virtual root window is in use on the screen, the position is computed in the whole desktop, not relative to the top-left corner of the currently shown portion). If the mouse pointer isn’t on the same screen as window then answer nil.

platform
: Answer the platform on which Blox is running; it can be either #unix, #macintosh or #windows.

screenOrigin
: Answer a Point indicating the coordinates of the upper left point of the screen in the virtual root window on which the application’s windows are drawn (under Windows and the Macintosh, that’s always 0 @ 0)

screenResolution
: Answer a Point containing the resolution in dots per inch of the screen, in the x and y directions.

screenSize
: Answer a Point containing the size of the virtual root window on which the application’s windows are drawn (under Windows and the Macintosh, that’s the size of the screen)

1.27.5 BLOX.Blox: accessing

state
: Answer the value of the state option for the widget.

  Specifies one of three states for the button: normal, active, or disabled. In normal state the button is displayed using the foreground and background options. The active state is typically used when the pointer is over the button. In active state the button is displayed using the activeForeground and activeBackground options. Disabled state means that the button should be insensitive: the application will refuse to activate the widget and will ignore mouse button presses.

state: value

Set the value of the state option for the widget.

Specifies one of three states for the button: normal, active, or disabled. In normal state the button is displayed using the foreground and background options. The active state is typically used when the pointer is over the button. In active state the button is displayed using the activeForeground and activeBackground options. Disabled state means that the button should be insensitive: the application will refuse to activate the widget and will ignore mouse button presses.

1.27.6 BLOX.Blox: basic

deepCopy It does not make sense to make a copy, because it would make data inconsistent across different objects; so answer the receiver

release Destroy the receiver if it still exists, then perform the usual task of removing the dependency links

shallowCopy

It does not make sense to make a copy, because it would make data inconsistent across different objects; so answer the receiver

1.27.7 BLOX.Blox: creating children

make: array

Create children of the receiver. Answer a Dictionary of the children. Each element of array is an Array including: a string which becomes the Dictionary's key, a binding like #{Blox.BWindow} identifying the class name, an array with the parameters to be set (for example #(#width: 50 #height: 30 #backgroundColor: 'blue')), and afterwards the children of the widget, described as arrays with this same format.

make: array on: result

Private - Create children of the receiver, adding them to result; answer result. array has the format described in the comment to #make:

makeChild: each on: result

Private - Create a child of the receiver, adding them to result; each is a single element of the array described in the comment to #make:

1.27.8 BLOX.Blox: customization

addChild: child

The widget identified by child has been added to the receiver. This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to either the superclass implementation or #basicAddChild:, to perform some initialization on the children just added. Answer the new child.

basicAddChild: child

The widget identified by child has been added to the receiver. Add it to the children collection and answer the new child. This method is public because you can call it from #addChild:.

1.27.9 BLOX.Blox: widget protocol

asPrimitiveWidget
Answer the primitive widget that implements the receiver.

childrenCount
Answer how many children the receiver has

childrenDo: aBlock
Evaluate aBlock once for each of the receiver's child widgets, passing the widget to aBlock as a parameter

destroy
Destroy the receiver

drawingArea
Answer a Rectangle identifying the receiver's drawing area. The rectangle's corners specify the upper-left and lower-right corners of the client area. Because coordinates are relative to the upper-left corner of a window's drawing area, the coordinates of the rectangle's corner are (0,0).

enabled
Answer whether the receiver is enabled to input. Although defined here, this method is only used for widgets that define a #state method

enabled: enabled
Set whether the receiver is enabled to input (enabled is a boolean). Although defined here, this method is only used for widgets that define a #state: method

exists
Answer whether the receiver has been destroyed or not (answer false in the former case, true in the latter).

fontHeight: aString
Answer the height of aString in pixels, when displayed in the same font as the receiver. Although defined here, this method is only used for widgets that define a #font method

fontWidth: aString
Answer the width of aString in pixels, when displayed in the same font as the receiver. Although defined here, this method is only used for widgets that define a #font method

isWindow
Answer whether the receiver represents a window on the screen.

parent
Answer the receiver's parent (or nil for a top-level window).

toplevel
Answer the top-level object (typically a BWindow or BPopupWindow) connected to the receiver.

window
Answer the window in which the receiver stays. Note that while #toplevel won't answer a BTransientWindow, this method will.

withChildrenDo: aBlock
Evaluate aBlock passing the receiver, and then once for each of the receiver's child widgets.

1.28 BLOX.BMenu

Defined in namespace BLOX
Superclass: BLOX.BMenuObject
Category: Graphics-Windows

I am a Menu that is part of a menu bar.

1.28.1 BLOX.BMenu class: instance creation

new: parent label: label
Add a new menu to the parent window's menu bar, with 'label' as its caption (for popup menus, parent is the widget over which the menu pops up as the right button is pressed).

1.28.2 BLOX.BMenu: accessing

label Answer the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

label: value
Set the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

1.28.3 BLOX.BMenu: callback registration

addLine Add a separator item at the end of the menu

addMenuItemFor: anArray notifying: receiver
Add a menu item described by anArray at the end of the menu. If anArray is empty, insert a separator line. If anArray has a single item, a menu item is created without a callback. If anArray has two or three items, the second one is used as the selector sent to receiver, and the third one (if present) is passed to the selector.

callback: receiver using: selectorPairs
Add menu items described by anArray at the end of the menu. Each element of selectorPairs must be in the format described in BMenu>>#addMenuItemFor:-notifying:. All the callbacks will be sent to receiver.

destroy Destroy the menu widget; that is, simply remove ourselves from the parent menu bar.

empty Empty the menu widget; that is, remove all the children

1.29 BLOX.BMenuBar

Defined in namespace BLOX
Superclass: BLOX.BMenuObject
Category: Graphics-Windows

I am the Menu Bar, the top widget in a full menu structure.

1.29.1 BLOX.BMenuBar: accessing

add: aMenu
Add aMenu to the menu bar

remove: aMenu
Remove aMenu from the menu bar

1.30 BLOX.BMenuItem

Defined in namespace BLOX
Superclass: BLOX.BMenuObject
Category: Graphics-Windows

I am the tiny and humble Menu Item, a single command choice in the menu structure. But if it wasn't for me, nothing could be done... eh eh eh!!

1.30.1 BLOX.BMenuItem class: instance creation

new: parent
Add a new separator item to the specified menu.

new: parent label: label
Add a new menu item to the specified menu (parent) , with 'label' as its caption.

1.30.2 BLOX.BMenuItem: accessing

label
Answer the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

label: value
Set the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

1.31 BLOX.BMenuObject

Defined in namespace BLOX
Superclass: BLOX.Blox
Category: Graphics-Windows

I am an abstract superclass for widgets which make up a menu structure.

1.31.1 BLOX.BMenuObject: accessing

activeBackground
Answer the value of the activeBackground option for the widget.

Specifies background color to use when drawing active elements. An element (a widget or portion of a widget) is active if the mouse cursor is positioned over the element and pressing a mouse button will cause some action to occur. For some elements on Windows and Macintosh systems, the active color will only be used while mouse button 1 is pressed over the element.

activeBackground: value
Set the value of the activeBackground option for the widget.

Specifies background color to use when drawing active elements. An element (a widget or portion of a widget) is active if the mouse cursor is positioned over the element and pressing a mouse button will cause some action to occur. For some elements on Windows and Macintosh systems, the active color will only be used while mouse button 1 is pressed over the element.

activeForeground
Answer the value of the activeForeground option for the widget.

Specifies foreground color to use when drawing active elements. See above for definition of active elements.

activeForeground: value
Set the value of the activeForeground option for the widget.

Specifies foreground color to use when drawing active elements. See above for definition of active elements.

asPrimitiveWidget
Answer the primitive widget that implements the receiver.

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

foregroundColor
Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value
Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

1.31.2 BLOX.BMenuObject: callback

callback
Answer a DirectedMessage that is sent when the receiver is modified, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is clicked. If the method accepts an argument, the receiver is passed.

callback: aReceiver message: aSymbol argument: anObject
Set up so that aReceiver is sent the aSymbol message (the name of a one- or two-argument selector) when the receiver is clicked. If the method accepts two argument, the receiver is passed together with anObject; if it accepts a single one, instead, only anObject is passed.

invokeCallback
Generate a synthetic callback

1.32 BLOX.BOval

Defined in namespace BLOX
Superclass: BLOX.BRectangle
Category: Graphics-Windows
I can draw ovals (ok, if you're a mathematic, they're really ellipses), or even circles.

1.33 BLOX.BPolyline

Defined in namespace BLOX
Superclass: BLOX.BCanvasObject
Category: Graphics-Windows
I can draw closed or open polylines, and even fill them!

1.33.1 BLOX.BPolyline: accessing

boundingBox
Answer 'boundingBox'.

cap Answer the way in which caps are to be drawn at the endpoints of the line.

This option is only available for open polylines. If you want to set it for a closed polylines, draw an open one on top of it.

cap: aSymbol
Set the way in which caps are to be drawn at the endpoints of the line. aSymbol may be #butt (the default), #projecting, or #round).

This option is only available for open polylines. If you want to set it for a closed polylines, draw an open one on top of it.

closed Answer whether the polyline is an open or a closed one.

closed: aBoolean
Set whether the polyline is an open or a closed one. This option may be set only once.

join Answer the way in which joints are to be drawn at the vertices of the line.

This option is only available for open polylines. If you want to set it for a closed polylines, draw an open one on top of it.

join: aSymbol
Answer the way in which joints are to be drawn at the vertices of the line. aSymbol can be #bevel, #miter (the default) or #round.

This option is only available for open polylines. If you want to set it for a closed polylines, draw an open one on top of it.

outlineColor
Answer the color with which the outline of the polyline is drawn. This option is only available for closed polylines.

outlineColor: color
Set the color with which the outline of the polyline is drawn. This option is only available for closed polylines.

points Answer the points that are vertices of the polyline.

points: arrayOfPointsOrArrays
Set the points that are vertices of the polyline. Each of the items of arrayOfPointsOrArrays can be a Point or a two-element Array. Note that no changes take place until you invoke the #create (if the object has not been inserted in the canvas yet) or the #redraw method.

width Answer the width with which the polyline (or its outline if it is a closed one) is drawn.

width: pixels
Set the width with which the polyline (or its outline if it is a closed one) is drawn.

1.34 BLOX.BPopupMenu

Defined in namespace BLOX
Superclass: BLOX.BMenu
Category: Graphics-Windows
I am a class that provides the ability to show popup menus when the right button (Button 3) is clicked on another window.

1.34.1 BLOX.BPopupMenu: widget protocol

popup Generate a synthetic menu popup event

1.35 BLOX.BPopupWindow

Defined in namespace BLOX
Superclass: BLOX.BWindow
Category: Graphics-Windows
I am a pseudo-window that has no decorations and no ability to interact with the user. My main usage, as my name says, is to provide pop-up functionality for other widgets. Actually there should be no need to directly use me - always rely on the #new and #popup: class methods.

1.35.1 BLOX.BPopupWindow: geometry management

addChild: w
: Private - The widget identified by child has been added to the receiver. This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to either the superclass implementation or #basicAddChild:, to perform some initialization on the children just added. Answer the new child.

child: child height: value
: Set the given child's height. This is done by setting its parent window's (that is, our) height.

child: child heightOffset: value
: This method should not be called for instances of this class.

child: child width: value
: Set the given child's width. This is done by setting its parent window's (that is, our) width.

child: child widthOffset: value
: This method should not be called for instances of this class.

child: child x: value
: Set the x coordinate of the given child's top-left corner. This is done by setting its parent window's (that is, our) x.

child: child xOffset: value
: This method should not be called for instances of this class.

child: child y: value
: Set the y coordinate of the given child's top-left corner. This is done by setting its parent window's (that is, our) y.

child: child yOffset: value
: This method should not be called for instances of this class.

heightChild: child
: Answer the given child's height, which is the height that was imposed on the popup window.

widthChild: child
: Answer the given child's width in pixels, which is the width that was imposed on the popup window.

xChild: child
: Answer the x coordinate of the given child's top-left corner, which is desumed by the position of the popup window.

yChild: child
: Answer the y coordinate of the given child's top-left corner, which is desumed by the position of the popup window.

1.36 BLOX.BPrimitive

Defined in namespace BLOX
Superclass: BLOX.BWidget
Category: Graphics-Windows

I am the superclass for every widget (except menus) directly provided by the underlying GUI system.

1.36.1 BLOX.BPrimitive: accessing

asPrimitiveWidget
Answer the primitive widget that implements the receiver.

1.37 BLOX.BProgress

Defined in namespace BLOX
Superclass: BLOX.BExtended
Category: Graphics-Examples

I show how much of a task has been completed.

1.37.1 BLOX.BProgress: accessing

backgroundColor
Answer the background color of the widget. This is used for the background of the non-filled part, as well as for the foreground of the filled part.

backgroundColor: aColor
Set the background color of the widget. This is used for the background of the non-filled part, as well as for the foreground of the filled part.

filledColor
Answer the background color of the widget's filled part.

filledColor: aColor
Set the background color of the widget's filled part.

foregroundColor
Set the foreground color of the widget. This is used for the non-filled part, while the background color also works as the foreground of the filled part.

foregroundColor: aColor
Set the foreground color of the widget. This is used for the non-filled part, while the background color also works as the foreground of the filled part.

value
Answer the filled percentage of the receiver (0..1)

value: newValue
Set the filled percentage of the receiver and update the appearance. newValue must be between 0 and 1.

1.38 BLOX.BRadioButton

Defined in namespace BLOX
Superclass: BLOX.BButton
Category: Graphics-Windows

I am just one in a group of mutually exclusive buttons.

1.38.1 BLOX.BRadioButton: accessing

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a selector accepting at most two arguments) when the receiver is clicked. If the method accepts two arguments, the receiver is passed as the first parameter. If the method accepts one or two arguments, true is passed as the last parameter for interoperability with BToggle widgets.

value Answer whether this widget is the selected one in its radio button group.

value: aBoolean
Answer whether this widget is the selected one in its radio button group. Setting this property to false for a group's currently selected button unhighlights all the buttons in that group.

1.39 BLOX.BRadioGroup

Defined in namespace BLOX
Superclass: BLOX.BContainer
Category: Graphics-Windows

I am used to group many mutually-exclusive radio buttons together. In addition, just like every BContainer I can perform simple management by putting widgets next to each other, from left to right or (which is more useful in this particular case...) from top to bottom.

1.39.1 BLOX.BRadioGroup: accessing

value Answer the index of the button that is currently selected, 1 being the first button added to the radio button group. 0 means that no button is selected

value: value
Force the value-th button added to the radio button group to be the selected one.

1.39.2 BLOX.BRadioGroup: widget protocol

destroyed Private - The receiver has been destroyed, clear the corresponding Tcl variable to avoid memory leaks.

1.40 BLOX.BRectangle

Defined in namespace BLOX
Superclass: BLOX.BBoundingBox
Category: Graphics-Windows

I only draw rectangles but I can do that very well.

1.40.1 BLOX.BRectangle: accessing

outlineColor
Answer the color with which the outline of the rectangle is drawn.

outlineColor: color
Set the color with which the outline of the rectangle is drawn.

width Answer the width with which the outline of the rectangle is drawn.

width: pixels
Set the width with which the outline of the rectangle is drawn.

1.41 BLOX.BScrolledCanvas

Defined in namespace BLOX
Superclass: BLOX.BCanvas
Category: Graphics-Windows

I am much similar to BCanvas, but I sport, in addition, two fancy scroll bars. This is just a convenience, since it could be easily done when creating the canvas...

1.42 BLOX.BSpline

Defined in namespace BLOX
Superclass: BLOX.BPolyline
Category: Graphics-Windows

Unlike my father BPolyline, I am more smooth at doing my job.

1.42.1 BLOX.BSpline: accessing

smoothness
Answer the degree of smoothness desired for curves. Each spline will be approximated with this number of line segments.

smoothness: anInteger
Set the degree of smoothness desired for curves. Each spline will be approximated with this number of line segments.

1.43 BLOX.BText

Defined in namespace BLOX
Superclass: BLOX.BViewport
Category: Graphics-Windows

I represent a text viewer with pretty good formatting options.

1.43.1 BLOX.BText class: accessing

emacsLike Answer whether we are using Emacs or Motif key bindings.

emacsLike: aBoolean
Set whether we are using Emacs or Motif key bindings.

1.43.2 BLOX.BText class: instance creation

newReadOnly: parent
Answer a new read-only text widget (read-only is achieved simply by setting its state to be disabled)

1.43.3 BLOX.BText: accessing

backgroundColor
Answer the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

backgroundColor: value
Set the value of the backgroundColor option for the widget.

Specifies the normal background color to use when displaying the widget.

callback Answer a DirectedMessage that is sent when the receiver is modified, or nil if none has been set up.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the receiver is modified. If the method accepts an argument, the receiver is passed.

contents Return the contents of the widget

contents: aString
Set the contents of the widget

font Answer the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: value

Set the value of the font option for the widget.

Specifies the font to use when drawing text inside the widget. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor

Answer the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

foregroundColor: value

Set the value of the foregroundColor option for the widget.

Specifies the normal foreground color to use when displaying the widget.

getSelection

Answer an empty string if the widget has no selection, else answer the currently selected text

selectBackground

Answer the value of the selectBackground option for the widget.

Specifies the background color to use when displaying selected parts of the widget.

selectBackground: value

Set the value of the selectBackground option for the widget.

Specifies the background color to use when displaying selected parts of the widget.

selectForeground

Answer the value of the selectForeground option for the widget.

Specifies the foreground color to use when displaying selected parts of the widget.

selectForeground: value

Set the value of the selectForeground option for the widget.

Specifies the foreground color to use when displaying selected parts of the widget.

wrap Answer the value of the wrap option for the widget.

Specifies how to handle lines in the text that are too long to be displayed in a single line of the text's window. The value must be #none or #char or #word. A wrap mode of none means that each line of text appears as exactly one line on the screen; extra characters that do not fit on the screen are not displayed. In the other modes each line of text will be broken up into several screen lines if necessary to keep all the characters visible. In char mode a screen line break may occur after any character; in word mode a line break will only be made at word boundaries.

wrap: value
Set the value of the wrap option for the widget.

Specifies how to handle lines in the text that are too long to be displayed in a single line of the text's window. The value must be #none or #char or #word. A wrap mode of none means that each line of text appears as exactly one line on the screen; extra characters that do not fit on the screen are not displayed. In the other modes each line of text will be broken up into several screen lines if necessary to keep all the characters visible. In char mode a screen line break may occur after any character; in word mode a line break will only be made at word boundaries.

1.43.4 BLOX.BText: attributes

insertAtEnd: aString attribute: attr
Clear the selection and append aString at the end of the widget. Use the given attributes to format the text.

insertText: aString attribute: attr
Insert aString in the widget at the current insertion point, replacing the currently selected text (if any). Use the given attributes to format the text.

removeAttributes
Remove any kind of formatting from the text in the widget

removeAttributesFrom: aPoint to: endPoint
Remove any kind of formatting from the text in the widget between the given endpoints. The two endpoints are Point objects in which both coordinates are 1-based: the first line is line 1, and the first character in the first line is character 1.

setAttributes: attr from: aPoint to: endPoint
Add the formatting given by attr to the text in the widget between the given endpoints. The two endpoints are Point objects in which both coordinates are 1-based: the first line is line 1, and the first character in the first line is character 1.

1.43.5 BLOX.BText: geometry management

child: child height: value
Set the height of the given child to be 'value' pixels.

child: child heightOffset: value
Adjust the height of the given child to be given by 'value' more pixels.

child: child width: value
Set the width of the given child to be 'value' pixels.

child: child widthOffset: value
Adjust the width of the given child to be given by 'value' more pixels.

child: child x: value
Never fail and do nothing, the children stay where the text ended at the time each child was added in the widget

child: child xOffset: value
This method should not be called for instances of this class.

child: child y: value
Never fail and do nothing, the children stay where the text ended at the time each child was added in the widget

child: child yOffset: value
This method should not be called for instances of this class.

heightChild: child
Answer the given child's height in pixels.

widthChild: child
Answer the given child's width in pixels.

xChild: child
Answer the given child's top-left border's x coordinate. We always answer 0 since the children actually move when the text widget scrolls

yChild: child
Answer the given child's top-left border's y coordinate. We always answer 0 since the children actually move when the text widget scrolls

1.43.6 BLOX.BText: images

insertImage: anObject
Insert an image where the insertion point currently lies in the widget. anObject can be a String containing image data (either Base-64 encoded GIF data, XPM data, or PPM data), or the result or registering an image with #registerImage:

insertImage: anObject at: position
Insert an image at the given position in the widget. The position is a Point object in which both coordinates are 1-based: the first line is line 1, and the first character in the first line is character 1.

anObject can be a String containing image data (either Base-64 encoded GIF data, XPM data, or PPM data), or the result or registering an image with #registerImage:

insertImageAtEnd: anObject
Insert an image at the end of the widgets text. anObject can be a String containing image data (either Base-64 encoded GIF data, XPM data, or PPM data), or the result or registering an image with #registerImage:

registerImage: anObject
: Register an image (whose data is in anObject, a String including Base-64 encoded GIF data, XPM data, or PPM data) to be used in the widget. If the same image must be used a lot of times, it is better to register it once and then pass the result of #registerImage: to the image insertion methods.

  Registered image are private within each BText widget. Registering an image with a widget and using it with another could give unpredictable results.

1.43.7 BLOX.BText: inserting text

insertAtEnd: aString
: Clear the selection and append aString at the end of the widget.

insertSelectedText: aString
: Insert aString in the widget at the current insertion point, leaving the currently selected text (if any) in place, and selecting the text.

insertText: aString
: Insert aString in the widget at the current insertion point, replacing the currently selected text (if any).

insertText: aString at: position
: Insert aString in the widget at the given position, replacing the currently selected text (if any). The position is a Point object in which both coordinates are 1-based: the first line is line 1, and the first character in the first line is character 1.

insertTextSelection: aString
: Insert aString in the widget after the current selection, leaving the currently selected text (if any) intact.

invokeCallback
: Generate a synthetic callback.

nextPut: aCharacter
: Clear the selection and append aCharacter at the end of the widget.

nextPutAll: aString
: Clear the selection and append aString at the end of the widget.

nl Clear the selection and append a linefeed character at the end of the widget.

refuseTabs
: Arrange so that Tab characters, instead of being inserted in the widget, traverse the widgets in the parent window.

replaceSelection: aString
: Insert aString in the widget at the current insertion point, replacing the currently selected text (if any), and leaving the text selected.

searchString: aString
: Search aString in the widget. If it is not found, answer zero, else answer the 1-based line number and move the insertion point to the place where the string was found.

space Clear the selection and append a space at the end of the widget.

1.43.8 BLOX.BText: position & lines

charsInLine: number
Answer how many characters are there in the number-th line

currentColumn
Answer the 1-based column number where the insertion point currently lies.

currentLine
Answer the 1-based line number where the insertion point currently lies.

currentPosition
Answer a Point representing where the insertion point currently lies. Both coordinates in the answer are 1-based: the first line is line 1, and the first character in the first line is character 1.

currentPosition: aPoint
Move the insertion point to the position given by aPoint. Both coordinates in aPoint are interpreted as 1-based: the first line is line 1, and the first character in the first line is character 1.

gotoLine: line end: aBoolean
If aBoolean is true, move the insertion point to the last character of the line-th line (1 being the first line in the widget); if aBoolean is false, move it to the start of the line-th line.

indexAt: point
Answer the position of the character that covers the pixel whose coordinates within the text's window are given by the supplied Point object.

lineAt: number
Answer the number-th line of text in the widget

numberOfLines
Answer the number of lines in the widget

selectFrom: first to: last
Select the text between the given endpoints. The two endpoints are Point objects in which both coordinates are 1-based: the first line is line 1, and the first character in the first line is character 1.

setToEnd Move the insertion point to the end of the widget

1.44 BLOX.BTextAttributes

Defined in namespace BLOX
Superclass: Object
Category: Graphics-Windows
I help you creating wonderful, colorful BTexts.

1.44.1 BLOX.BTextAttributes class: instance-creation shortcuts

backgroundColor: color
: Create a new BTextAttributes object resulting in text with the given background color.

black
: Create a new BTextAttributes object resulting in black text.

blue
: Create a new BTextAttributes object resulting in blue text.

center
: Create a new BTextAttributes object resulting in centered paragraphs.

cyan
: Create a new BTextAttributes object resulting in cyan text.

darkCyan
: Create a new BTextAttributes object resulting in dark cyan text.

darkGreen
: Create a new BTextAttributes object resulting in dark green text.

darkMagenta
: Create a new BTextAttributes object resulting in dark purple text.

events: aBTextBindings
: Create a new BTextAttributes object for text that responds to events according to the callbacks established in aBTextBindings.

font: font
: Create a new BTextAttributes object resulting in text with the given font. The font can be given as either an X font name or a Blox font description string.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor: color
: Create a new BTextAttributes object resulting in text with the given foreground color.

green
: Create a new BTextAttributes object resulting in green text.

magenta
: Create a new BTextAttributes object resulting in magenta text.

red
: Create a new BTextAttributes object resulting in red text.

strikeout
: Create a new BTextAttributes object resulting in struck-out text.

underline
: Create a new BTextAttributes object resulting in underlined text.

white
: Create a new BTextAttributes object resulting in white text.

yellow
: Create a new BTextAttributes object resulting in yellow text.

1.44.2 BLOX.BTextAttributes: colors

black Set the receiver so that applying it results in black text.

blue Set the receiver so that applying it results in blue text.

cyan Set the receiver so that applying it results in cyan text.

darkCyan Set the receiver so that applying it results in dark cyan text.

darkGreen
Set the receiver so that applying it results in dark green text.

darkMagenta
Set the receiver so that applying it results in dark magenta text.

green Set the receiver so that applying it results in green text.

magenta Set the receiver so that applying it results in magenta text.

red Set the receiver so that applying it results in red text.

white Set the receiver so that applying it results in white text.

yellow Set the receiver so that applying it results in black text.

1.44.3 BLOX.BTextAttributes: setting attributes

backgroundColor
Answer the value of the backgroundColor option for the text.

Specifies the background color to use when displaying text with these attributes. nil indicates that the default value is not overridden.

backgroundColor: color
Set the value of the backgroundColor option for the text.

Specifies the background color to use when displaying text with these attributes. nil indicates that the default value is not overridden.

center Center the text to which these attributes are applied

events Answer the event bindings which apply to text subject to these attributes

events: aBTextBindings
Set the event bindings which apply to text subject to these attributes

font Answer the value of the font option for the text. The font can be given as either an X font name or a Blox font description string, or nil if you want the widget's default font to apply.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

font: fontName
Set the value of the font option for the text. The font can be given as either an X font name or a Blox font description string, or nil if you want the widget’s default font to apply.

X font names are given as many fields, each led by a minus, and each of which can be replaced by an * to indicate a default value is ok: foundry, family, weight, slant, setwidth, addstyle, pixel size, point size (the same as pixel size for historical reasons), horizontal resolution, vertical resolution, spacing, width, charset and character encoding.

Blox font description strings have three fields, which must be separated by a space and of which only the first is mandatory: the font family, the font size in points (or in pixels if a negative value is supplied), and a number of styles separated by a space (valid styles are normal, bold, italic, underline and overstrike). Examples of valid fonts are “Helvetica 10 Bold”, “Times -14”, “Futura Bold Underline”. You must enclose the font family in braces if it is made of two or more words.

foregroundColor
Answer the value of the foregroundColor option for the text.

Specifies the foreground color to use when displaying text with these attributes. nil indicates that the default value is not overridden.

foregroundColor: color
Set the value of the foregroundColor option for the text.

Specifies the foreground color to use when displaying text with these attributes. nil indicates that the default value is not overridden.

isCentered
Answer whether the text to which these attributes are applied is centered

isStruckout
Answer whether the text to which these attributes are applied is struckout

isUnderlined
Answer whether the text to which these attributes are applied is underlined

strikeout Strike out the text to which these attributes are applied

underline Underline the text to which these attributes are applied

1.45 BLOX.BTextBindings

Defined in namespace BLOX
Superclass: BLOX.BEventTarget
Category: Graphics-Windows

This object is used to assign event handlers to particular sections of text in a BText widget. To use it, you simply have to add event handlers to it, and then create a BTextAttributes object that refers to it.

1.45.1 BLOX.BTextBindings class: instance creation

new Create a new instance of the receiver.

1.46 BLOX.BTextTags

Defined in namespace BLOX
Superclass: Object
Category: Graphics-Windows

I am a private class. I sit between a BText and BTextAttributes, helping the latter in telling the former which attributes to use.

1.47 BLOX.BToggle

Defined in namespace BLOX
Superclass: BLOX.BButton
Category: Graphics-Windows

I represent a button whose choice can be included (by checking me) or excluded (by leaving me unchecked).

1.47.1 BLOX.BToggle: accessing

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a selector accepting at most two arguments) when the receiver is clicked. If the method accepts two arguments, the receiver is passed as the first parameter. If the method accepts one or two arguments, the state of the widget (true if it is selected, false if it is not) is passed as the last parameter.

invokeCallback
Generate a synthetic callback.

value Answer whether the button is in a selected (checked) state.

value: aBoolean
Set whether the button is in a selected (checked) state and generates a callback accordingly.

variable: value
Set the value of Tk's variable option for the widget.

1.48 BLOX.BTransientWindow

Defined in namespace BLOX
Superclass: BLOX.BWindow
Category: Graphics-Windows

I am almost a boss. I represent a window which is logically linked to another which sits higher in the widget hierarchy, e.g. a dialog box

1.48.1 BLOX.BTransientWindow class: instance creation

new This method should not be called for instances of this class.

new: parentWindow
Answer a new transient window attached to the given parent window and with nothing in its title bar caption.

new: label in: parentWindow
Answer a new transient window attached to the given parent window and with 'label' as its title bar caption.

1.48.2 BLOX.BTransientWindow: widget protocol

map Map the window and inform the windows manager that the receiver is a transient window working on behalf of its parent. The window is also put in its parent window's window group: the window manager might use this information, for example, to unmap all of the windows in a group when the group's leader is iconified.

1.49 BLOX.BViewport

Defined in namespace BLOX
Superclass: BLOX.BPrimitive
Category: Graphics-Windows

I represent an interface which is common to widgets that can be scrolled, like list boxes or text widgets.

1.49.1 BLOX.BViewport: accessing

connected Private - Answer the name of Tk widget for the connected widget.

1.49.2 BLOX.BViewport: scrollbars

horizontal Answer whether an horizontal scrollbar is drawn in the widget if needed.

horizontal: aBoolean
Set whether an horizontal scrollbar is drawn in the widget if needed.

horizontalNeeded
Answer whether an horizontal scrollbar is needed to show all the information in the widget.

horizontalShown
Answer whether an horizontal scrollbar is drawn in the widget.

vertical Answer whether a vertical scrollbar is drawn in the widget if needed.

vertical: aBoolean
Set whether a vertical scrollbar is drawn in the widget if needed.

verticalNeeded
Answer whether a vertical scrollbar is needed to show all the information in the widget.

verticalShown
Answer whether a vertical scrollbar is drawn in the widget.

1.50 BLOX.BWidget

Defined in namespace BLOX
Superclass: BLOX.Blox
Category: Graphics-Windows
I am the superclass for every widget except those related to menus. I provide more common methods and geometry management

1.50.1 BLOX.BWidget class: popups

new Create an instance of the receiver inside a BPopupWindow; do not map the window, answer the new widget. The created widget will become a child of the window and be completely attached to it (e.g. the geometry methods will modify the window's geometry). Note that while the widget *seems* to be directly painted on the root window, it actually belongs to the BPopupWindow; so don't send #destroy to the widget to remove it, but rather to the window.

popup: initializationBlock
Create an instance of the receiver inside a BPopupWindow; before returning, pass the widget to the supplied initializationBlock, then map the window. Answer the new widget. The created widget will become a child of the window and be completely attached to it (e.g. the geometry methods will modify the window's geometry). Note that while the widget *seems* to be directly painted on the root window, it actually belongs to the BPopupWindow; so don't send #destroy to the widget to remove it, but rather to the window.

1.50.2 BLOX.BWidget: accessing

borderWidth
Answer the value of the borderWidth option for the widget.

Specifies a non-negative value indicating the width of the 3-D border to draw around the outside of the widget (if such a border is being drawn; the effect option typically determines this). The value may also be used when drawing 3-D effects in the interior of the widget. The value is measured in pixels.

borderWidth: value
Set the value of the borderWidth option for the widget.

Specifies a non-negative value indicating the width of the 3-D border to draw around the outside of the widget (if such a border is being drawn; the effect option typically determines this). The value may also be used when drawing 3-D effects in the interior of the widget. The value is measured in pixels.

cursor Answer the value of the cursor option for the widget.

Specifies the mouse cursor to be used for the widget. The value of the option is given by the standard X cursor cursor, i.e., any of the names defined in cursorcursor.h, without the leading XC_.

cursor: value

Set the value of the cursor option for the widget.

Specifies the mouse cursor to be used for the widget. The value of the option is given by the standard X cursor cursor, i.e., any of the names defined in cursorcursor.h, without the leading XC_.

effect Answer the value of the effect option for the widget.

Specifies the effect desired for the widget's border. Acceptable values are raised, sunken, flat, ridge, solid, and groove. The value indicates how the interior of the widget should appear relative to its exterior; for example, raised means the interior of the widget should appear to protrude from the screen, relative to the exterior of the widget. Raised and sunken give the traditional 3-D appearance (for example, that of Xaw3D), while ridge and groove give a "chiseled" appearance like that of Swing or GTK+'s Metal theme. Flat and solid are not 3-D.

effect: value

Set the value of the effect option for the widget.

Specifies the effect desired for the widget's border. Acceptable values are raised, sunken, flat, ridge, solid, and groove. The value indicates how the interior of the widget should appear relative to its exterior; for example, raised means the interior of the widget should appear to protrude from the screen, relative to the exterior of the widget. Raised and sunken give the traditional 3-D appearance (for example, that of Xaw3D), while ridge and groove give a "chiseled" appearance like that of Swing or GTK+'s Metal theme. Flat and solid are not 3-D.

tabStop Answer the value of the tabStop option for the widget.

Determines whether the window accepts the focus during keyboard traversal (e.g., Tab and Shift-Tab). Before setting the focus to a window, Blox consults the value of the tabStop option. A value of false means that the window should be skipped entirely during keyboard traversal. true means that the window should receive the input focus as long as it is viewable (it and all of its ancestors are mapped). If you do not set this option, Blox makes the decision about whether or not to focus on the window: the current algorithm is to skip the window if it is disabled, it has no key bindings, or if it is not viewable. Of the standard widgets, BForm, BContainer, BLabel and BImage have no key bindings by default.

tabStop: value

Set the value of the tabStop option for the widget.

Determines whether the window accepts the focus during keyboard traversal (e.g., Tab and Shift-Tab). Before setting the focus to a window, Blox consults the value of the tabStop option. A value of false means that the window should be skipped entirely during keyboard traversal. true means that the window should receive the input focus as long as it is viewable (it and all of its ancestors are mapped). If you do not set this option, Blox makes the decision about whether or not to focus on the window: the current algorithm is to skip the window if it is disabled, it has no key bindings, or if it is not viewable. Of the standard widgets, BForm, BContainer, BLabel and BImage have no key bindings by default.

1.50.3 BLOX.BWidget: customization

addChild: child
The widget identified by child has been added to the receiver. This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to basicAddChild, to perform some initialization on the children just added. Answer the new child.

create
Make the receiver able to respond to its widget protocol. This method is public not because you can call it, but because it can be useful to override it, not forgetting the call to super, to perform some initialization on the primitive widget just created; for an example of this, see the implementation of BButtonLike.

initialize: parentWidget
This is called by #new: to initialize the widget (as the name says...). The default implementation calls all the other methods in the 'customization' protocol and some private ones that take care of making the receiver's status consistent, so you should usually call it instead of doing everything by hand. This method is public not because you can call it, but because it might be useful to override it. Always answer the receiver.

setInitialSize
This is called by #initialize: to set the widget's initial size. The whole area is occupied by default. This method is public not because you can call it, but because it can be useful to override it.

1.50.4 BLOX.BWidget: geometry management

boundingBox
Answer a Rectangle containing the bounding box of the receiver

boundingBox: rect
Set the bounding box of the receiver to rect (a Rectangle).

child: child height: value
Set the given child's height to value. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #height method. You should not use this method, which is automatically called by the child's #height: method, but you might want to override it. The child's property slots whose name ends with 'Geom' are

reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just do nothing.

child: child heightOffset: value
Adjust the given child's height by a fixed amount of value pixel. This is meaningful for the default implementation, using 'rubber-sheet' geometry management as explained in the comment to BWidget's #height and #heightOffset: methods. You should not use this method, which is automatically called by the child's #heightOffset: method, but you might want to override it. if it doesn't apply to the kind of geometry management that the receiver does, just add value to the current height of the widget.

child: child stretch: aBoolean
This method is only used when on the path from the receiver to its toplevel there is a BContainer. It decides whether child is among the widgets that are stretched to fill the entire width of the BContainer; if this has not been set for this widget, it is propagated along the widget hierarchy.

child: child width: value
Set the given child's width to value. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #width method. You should not use this method, which is automatically called by the child's #width: method, but you might want to override it. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just do nothing.

child: child widthOffset: value
Adjust the given child's width by a fixed amount of value pixel. This is meaningful for the default implementation, using 'rubber-sheet' geometry management as explained in the comment to BWidget's #width and #widthOffset: methods. You should not use this method, which is automatically called by the child's #widthOffset: method, but you might want to override it. if it doesn't apply to the kind of geometry management that the receiver does, just add value to the current width of the widget.

child: child x: value
Set the given child's x to value. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #x method. You should not use this method, which is automatically called by the child's #x: method, but you might want to override it. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just do nothing.

child: child xOffset: value
Adjust the given child's x by a fixed amount of value pixel. This is meaningful for the default implementation, using 'rubber-sheet' geometry management as explained in the comment to BWidget's #x and #xOffset: methods. You should not use this method, which is automatically called by the child's

#xOffset: method, but you might want to override it. if it doesn't apply to the kind of geometry management that the receiver does, just add value to the current x of the widget.

child: child y: value
Set the given child's y to value. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #y method. You should not use this method, which is automatically called by the child's #y: method, but you might want to override it. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just do nothing.

child: child yOffset: value
Adjust the given child's y by a fixed amount of value pixel. This is meaningful for the default implementation, using 'rubber-sheet' geometry management as explained in the comment to BWidget's #y and #yOffset: methods. You should not use this method, which is automatically called by the child's #yOffset: method, but you might want to override it. if it doesn't apply to the kind of geometry management that the receiver does, just add value to the current y of the widget.

extent Answer a Point containing the receiver's size

extent: extent
Set the receiver's size to the width and height contained in extent (a Point).

height Answer the 'variable' part of the receiver's height within the parent widget. The value returned does not include any fixed amount of pixels indicated by #heightOffset: and must be interpreted in a relative fashion: the ratio of the returned value to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management. Behavior if the left or right edges are not within the client area of the parent is not defined – the window might be clamped or might be positioned according to the specification.

height: value
Set to 'value' the height of the widget within the parent widget. The value is specified in a relative fashion as an integer, so that the ratio of 'value' to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management.

heightAbsolute
Force a recalculation of the layout of widgets in the receiver's parent, then answer the current height of the receiver in pixels.

heightChild: child
Answer the given child's height. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #height method. You should not use this method, which is automatically called by the child's #height method, but you might want to override.

The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just return 0.

heightOffset
Private - Answer the pixels to be added or subtracted to the height of the receiver, with respect to the value set in a relative fashion through the #height: method.

heightOffset: value
Add or subtract to the height of the receiver a fixed amount of 'value' pixels, with respect to the value set in a relative fashion through the #height: method. Usage of this method is deprecated; use #inset: and BContainers instead.

heightPixels: value
Set the current height of the receiver to 'value' pixels. Note that, after calling this method, #height will answer 0, which is logical considering that there is no 'variable' part of the size (refer to #height and #height: for more explanations).

inset: pixels
Inset the receiver's bounding box by the specified amount.

left: left top: top right: right bottom: bottom
Set the bounding box of the receiver through its components.

pos: position
Set the receiver's origin to the width and height contained in position (a Point).

posHoriz: aBlox
Position the receiver immediately to the right of aBlox.

posVert: aBlox
Position the receiver just below aBlox.

stretch: aBoolean
This method is only considered when on the path from the receiver to its toplevel there is a BContainer. It decides whether we are among the widgets that are stretched to fill the entire width of the BContainer.

width Answer the 'variable' part of the receiver's width within the parent widget. The value returned does not include any fixed amount of pixels indicated by #widthOffset: and must be interpreted in a relative fashion: the ratio of the returned value to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management. Behavior if the left or right edges are not within the client area of the parent is not defined – the window might be clamped or might be positioned according to the specification.

width: value
Set to 'value' the width of the widget within the parent widget. The value is specified in a relative fashion as an integer, so that the ratio of 'value' to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management.

width: xSize height: ySize
Set the size of the receiver through its components xSize and ySize.

widthAbsolute
Force a recalculation of the layout of widgets in the receiver's parent, then answer the current width of the receiver in pixels.

widthChild: child
Answer the given child's width. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #width method. You should not use this method, which is automatically called by the child's #width method, but you might want to override. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just return 0.

widthOffset
Private - Answer the pixels to be added or subtracted to the width of the receiver, with respect to the value set in a relative fashion through the #width: method.

widthOffset: value
Add or subtract to the width of the receiver a fixed amount of 'value' pixels, with respect to the value set in a relative fashion through the #width: method. Usage of this method is deprecated; use #inset: and BContainers instead.

widthPixels: value
Set the current width of the receiver to 'value' pixels. Note that, after calling this method, #width will answer 0, which is logical considering that there is no 'variable' part of the size (refer to #width and #width: for more explanations).

x
Answer the 'variable' part of the receiver's x within the parent widget. The value returned does not include any fixed amount of pixels indicated by #xOffset: and must be interpreted in a relative fashion: the ratio of the returned value to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management. Behavior if the left or right edges are not within the client area of the parent is not defined – the window might be clamped or might be positioned according to the specification.

x: value
Set to 'value' the x of the widget within the parent widget. The value is specified in a relative fashion as an integer, so that the ratio of 'value' to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management.

x: xPos y: yPos
Set the origin of the receiver through its components xPos and yPos.

x: xPos y: yPos width: xSize height: ySize
Set the bounding box of the receiver through its origin and size.

xAbsolute
Force a recalculation of the layout of widgets in the receiver's parent, then answer the current x of the receiver in pixels.

xChild: child
Answer the given child's x. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #x method. You should not use this method, which is automatically called by the child's #x method, but you might want to override. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just return 0.

xOffset Private - Answer the pixels to be added or subtracted to the x of the receiver, with respect to the value set in a relative fashion through the #x: method.

xOffset: value
Add or subtract to the x of the receiver a fixed amount of 'value' pixels, with respect to the value set in a relative fashion through the #x: method. Usage of this method is deprecated; use #inset: and BContainers instead.

xPixels: value
Set the current x of the receiver to 'value' pixels. Note that, after calling this method, #x will answer 0, which is logical considering that there is no 'variable' part of the size (refer to #x and #x: for more explanations).

xRoot Answer the x position of the receiver with respect to the top-left corner of the desktop (including the offset of the virtual root window under X).

y Answer the 'variable' part of the receiver's y within the parent widget. The value returned does not include any fixed amount of pixels indicated by #yOffset: and must be interpreted in a relative fashion: the ratio of the returned value to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management. Behavior if the left or right edges are not within the client area of the parent is not defined – the window might be clamped or might be positioned according to the specification.

y: value Set to 'value' the y of the widget within the parent widget. The value is specified in a relative fashion as an integer, so that the ratio of 'value' to the current size of the parent will be preserved upon resize. This apparently complicated method is known as 'rubber sheet' geometry management.

yAbsolute Force a recalculation of the layout of widgets in the receiver's parent, then answer the current y of the receiver in pixels.

yChild: child
Answer the given child's y. The default implementation of this method uses 'rubber-sheet' geometry management as explained in the comment to BWidget's #y method. You should not use this method, which is automatically called by the child's #y method, but you might want to override. The child's property slots whose name ends with 'Geom' are reserved for this method. This method should never fail – if it doesn't apply to the kind of geometry management that the receiver does, just return 0.

yOffset Private - Answer the pixels to be added or subtracted to the y of the receiver, with respect to the value set in a relative fashion through the #y: method.

yOffset: value
Add or subtract to the y of the receiver a fixed amount of 'value' pixels, with respect to the value set in a relative fashion through the #y: method. Usage of this method is deprecated; use #inset: and BContainers instead.

yPixels: value
Set the current y of the receiver to 'value' pixels. Note that, after calling this method, #y will answer 0, which is logical considering that there is no 'variable' part of the size (refer to #y and #y: for more explanations).

yRoot Answer the y position of the receiver with respect to the top-left corner of the desktop (including the offset of the virtual root window under X).

1.50.5 BLOX.BWidget: widget protocol

activate At any given time, one window on each display is designated as the focus window; any key press or key release events for the display are sent to that window. This method allows one to choose which window will have the focus in the receiver's display

If the application currently has the input focus on the receiver's display, this method resets the input focus for the receiver's display to the receiver. If the application doesn't currently have the input focus on the receiver's display, Blox will remember the receiver as the focus for its top-level; the next time the focus arrives at the top-level, it will be redirected to the receiver (this is because most window managers will set the focus only to top-level windows, leaving it up to the application to redirect the focus among the children of the top-level).

activateNext
Activate the next widget in the focus 'tabbing' order. The focus order depends on the widget creation order; you can set which widgets are in the order with the #tabStop: method.

activatePrevious
Activate the previous widget in the focus 'tabbing' order. The focus order depends on the widget creation order; you can set which widgets are in the order with the #tabStop: method.

bringToTop
Raise the receiver so that it is above all of its siblings in the widgets' z-order; the receiver will not be obscured by any siblings and will obscure any siblings that overlap it.

isActive Return whether the receiver is the window that currently owns the focus on its display.

sendToBack
Lower the receiver so that it is below all of its siblings in the widgets' z-order; the receiver will be obscured by any siblings that overlap it and will not obscure any siblings.

1.51 BLOX.BWindow

Defined in namespace BLOX
Superclass: BLOX.BForm
Category: Graphics-Windows

I am the boss. Nothing else could be viewed or interacted with if it wasn't for me...)):->

1.51.1 BLOX.BWindow class: instance creation

new Answer a new top-level window.

new: label Answer a new top-level window with 'label' as its title bar caption.

popup: initializationBlock
This method should not be called for instances of this class.

1.51.2 BLOX.BWindow: accessing

callback Answer a DirectedMessage that is sent to verify whether the receiver must be destroyed when the user asks to unmap it.

callback: aReceiver message: aSymbol
Set up so that aReceiver is sent the aSymbol message (the name of a zero- or one-argument selector) when the user asks to unmap the receiver. If the method accepts an argument, the receiver is passed.

If the method returns true, the window and its children are destroyed (which is the default action, taken if no callback is set up). If the method returns false, the window is left in place.

invokeCallback
Generate a synthetic callback, destroying the window if no callback was set up or if the callback method answers true.

label Answer the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

label: value
Set the value of the label option for the widget.

Specifies a string to be displayed inside the widget. The way in which the string is displayed depends on the particular widget and may be determined by other options, such as anchor. For windows, this is the title of the window.

menu: value
Set the value of the menu option for the widget.

Specifies a menu widget to be used as a menubar. On the Macintosh, the menubar will be displayed accross the top of the main monitor. On Microsoft Windows and all UNIX platforms, the menu will appear accross the toplevel window as part of the window dressing maintained by the window manager.

resizable Answer the value of the resizable option for the widget.

Answer whether the user can be resize the window or not. If resizing is disabled, then the window's size will be the size from the most recent interactive resize or geometry-setting method. If there has been no such operation then the window's natural size will be used.

resizable: value
Set the value of the resizable option for the widget.

Answer whether the user can be resize the window or not. If resizing is disabled, then the window's size will be the size from the most recent interactive resize or geometry-setting method. If there has been no such operation then the window's natural size will be used.

1.51.3 BLOX.BWindow: widget protocol

center Center the window in the screen

centerIn: view
Center the window in the given widget

height Answer the height of the window, as deduced from the geometry that the window manager imposed on the window.

height: anInteger
Ask the window manager to give the given height to the window.

heightAbsolute
Answer the height of the window, as deduced from the geometry that the window manager imposed on the window.

heightOffset: value
This method should not be called for instances of this class.

iconify Map a window and in iconified state. If a window has not been mapped yet, this is achieved by mapping the window in withdrawn state first, and then iconifying it.

isMapped Answer whether the window is mapped

isWindow Answer 'true'.

map Map the window and bring it to the topmost position in the Z-order.

modalMap
Map the window while establishing an application-local grab for it. An event loop is started that ends only after the window has been destroyed.

When a grab is set for a particular window, all pointer events are restructed to the grab window and its descendants in Blox's window hierarchy. Whenever the pointer is within the grab window's subtree, the pointer will behave exactly the same as if there had been no grab grab at all and all events will be reported in the normal fashion. When the pointer is outside the window's tree, button presses and releases and mouse motion events are reported to the grabbing window, and window entry and window exit events are ignored. In other words, windows outside the grab subtree will be visible on the screen but they will be insensitive until the grab is released. The tree of windows underneath the grab window can include top-level windows, in which case all of those top-level windows and their descendants will continue to receive mouse events during the grab. Keyboard events (key presses and key releases) are delivered as usual: the window manager controls which application receives keyboard events, and if they are sent to any window in the grabbing application then they are redirected to the window owning the focus.

state Set the value of the state option for the window.

Specifies one of four states for the window: either normal, iconic, withdrawn, or (Windows only) zoomed.

state: aSymbol
Raise an error. To set a BWindow's state, use #map and #unmap.

unmap Unmap a window, causing it to be forgotten about by the window manager

width Answer the width of the window, as deduced from the geometry that the window manager imposed on the window.

width: anInteger
Ask the window manager to give the given width to the window.

width: xSize height: ySize
Ask the window manager to give the given width and height to the window.

widthAbsolute
Answer the width of the window, as deduced from the geometry that the window manager imposed on the window.

widthOffset: value
This method should not be called for instances of this class.

window Answer the receiver.

x Answer the x coordinate of the window's top-left corner, as deduced from the geometry that the window manager imposed on the window.

x: anInteger
Ask the window manager to move the window's left border to the given x coordinate, keeping the size unchanged

x: xPos y: yPos
Ask the window manager to move the window's top-left corner to the given coordinates, keeping the size unchanged

x: xPos y: yPos width: xSize height: ySize
Ask the window manager to give the requested geometry to the window.

xAbsolute Answer the x coordinate of the window's top-left corner, as deduced from the geometry that the window manager imposed on the window.

xOffset: value
This method should not be called for instances of this class.

y Answer the y coordinate of the window's top-left corner, as deduced from the geometry that the window manager imposed on the window.

y: anInteger
Ask the window manager to move the window's left border to the given y coordinate, keeping the size unchanged

yAbsolute Answer the y coordinate of the window's top-left corner, as deduced from the geometry that the window manager imposed on the window.

yOffset: value
This method should not be called for instances of this class.

1.52 BLOX.Gui

Defined in namespace BLOX
Superclass: Object
Category: Graphics-Windows
I am a small class which serves as a base for complex objects which expose an individual protocol but internally use a Blox widget for creating their user interface.

1.52.1 BLOX.Gui: accessing

blox Return instance of blox subclass which implements window

blox: aBlox
Set instance of blox subclass which implements window

2 Complex number computations

2.1 Tree

Classes documented in this manual are **boldfaced**.

```
Object
  Magnitude
    Number
      Complex
```

2.2 Complex

Defined in namespace Smalltalk
Superclass: Number
Category: Examples-Useful

I provide complex numbers, with full interoperability with other kinds of numbers. Complex numbers can be created from imaginary numbers, which in turn are created with ‘Complex i’ or the #i method (e.g. ‘3 i’). Alternatively, they can be created from polar numbers.

2.2.1 Complex class: instance creation

i Return the imaginary unit, -1 sqrt.

initialize Initialize some common complex numbers.

new This method should not be called for instances of this class.

real: re imaginary: im
Return a complex number with the given real and imaginary parts.

realResult: re imaginary: im
Private - Return a new complex number knowing that re and im have the same generality.

rho: dist theta: angle
Return a complex number whose absolute value is dist and whose argument is angle.

2.2.2 Complex: comparing

< aNumber
Not commented.

<= aNumber
Not commented.

= aNumber
Not commented.

> aNumber
Not commented.

>= aNumber
Not commented.

hash Not commented.

~= aNumber
Not commented.

2.2.3 Complex: converting

asExactFraction
Not commented.

asFloat Not commented.

asFloatD Not commented.

asFloatE Not commented.

asFloatQ Not commented.

asFraction Not commented.

ceiling Not commented.

floor Not commented.

rounded Not commented.

truncated Not commented.

2.2.4 Complex: creation/coercion

coerce: aNumber
Not commented.

generality Not commented.

i Return the receiver multiplied by the imaginary unit.

imaginary Answer 'im'.

isComplex Answer 'true'.

one Answer 'One'.

real Answer 're'.

setReal: real imaginary: imag
Not commented.

zero Answer 'Zero'.

2.2.5 Complex: math

*** z**	Multiply the receiver by the (real or complex) number z.
+ z	Sum the receiver with the (real or complex) number z.
- z	Subtract the (real or complex) number z from the receiver.
/ z	Divide the receiver by the (real or complex) number z.
abs	Return the absolute value of the receiver.
absSquared	Return the squared absolute value of the receiver.
conjugate	Return the complex conjugate of the receiver.
reciprocal	Return the reciprocal of the receiver.

2.2.6 Complex: printing

printOn: aStream	Not commented.
storeOn: aStream	Not commented.

2.2.7 Complex: testing

isExact	Answer whether the receiver performs exact arithmetic. Complex numbers do so as long as both parts, real and imaginary, are exact.

2.2.8 Complex: transcendental functions

arcTan	Return the arc-tangent of the receiver.
arcTan: aNumber	Return the arc-tangent of aNumber divided by the receiver.
arg	Return the argument of the receiver.
cos	Return the cosine of the receiver.
cosh	Return the hyperbolic cosine of the receiver.
exp	Return e raised to the receiver.
ln	Return the natural logarithm of the receiver.
log	Return the base-10 logarithm of the receiver.
sin	Return the sine of the receiver.
sinh	Return the hyperbolic sine of the receiver.
sqrt	Return the square root of the receiver. Can be improved!
tan	Return the tangent of the receiver.
tanh	Return the hyperbolic tangent of the receiver.

3 Database connectivity with DBI

3.1 Tree

Classes documented in this manual are **boldfaced**.

```
DBI.ROE.RASQLRelation
  DBI.Table
Object
  DBI.ColumnInfo
  DBI.Connection
  DBI.ConnectionInfo
  DBI.FieldConverter
  DBI.Row
  DBI.Statement
  Iterable
    Stream
      DBI.ResultSet
```

3.2 DBI.ColumnInfo

Defined in namespace DBI
Superclass: Object
Category: DBI-Framework

3.2.1 DBI.ColumnInfo: accessing

index Return the 1-based index of the column in the result set (abstract).

isNullable Return whether the column can be NULL (always returns true in ColumnInfo).

name Return the name of the column (abstract).

size Return the size of the column (abstract).

type Return a string containing the type of the column (abstract).

3.2.2 DBI.ColumnInfo: printing

displayOn: aStream
Print a representation of the receiver on aStream.

printOn: aStream
Print a representation of the receiver on aStream.

3.3 DBI.Connection

Defined in namespace DBI
Superclass: Object
Category: DBI-Framework
I represent a connection to a database.

3.3.1 DBI.Connection class: connecting

connect: aDSN user: aUserName password: aPassword
Connect to the database server identified by aDSN using the given username and password. The DSN is in the format dbi:DriverName:-dbname=database_name;host=hostname;port=port Where dbi is constant, DriverName is the name of the driver, and everything else is parameters in the form name1=value1;name2=value2;...

Individual drivers may parse the parameters differently, though the existing ones all support parameters dbname, host and port.

paramConnect: params user: aUserName password: aPassword
Connect to the database server using the parameters in params (a Dictionary) and the given username and password (abstract).

3.3.2 DBI.Connection class: initialization

updateDriverList
Private - Look for new subclasses of Connection.

3.3.3 DBI.Connection: accessing

>> aString Returns a Table object corresponding to the given table.

database Returns the database name for this connection. This corresponds to the catalog in SQL standard parlance (abstract).

fieldConverter
Returns a FieldConverter that can be used to insert Smalltalk objects into queries.

tableAt: aString
Returns a Table object corresponding to the given table.

tableAt: aString ifAbsent: aBlock
Returns a Table object corresponding to the given table.

3.3.4 DBI.Connection: connecting

close Close the connection now; should happen on GC too (abstract).

3.3.5 DBI.Connection: querying

do: aSQLQuery
Executes a SQL statement (usually one that doesn't return a result set). Return value is a ResultSet, to which you can send #rowsAffected (abstract).

prepare: aSQLQuery
Creates a statement object, that can be executed (with parameters, if applicable) repeatedly (abstract).

primTableAt: aString ifAbsent: aBlock
Returns a Table object corresponding to the given table. Should be overridden by subclasses.

select: aSQLQuery
Prepares and executes a SQL statement. Returns the result set or throws an exception on failure (abstract).

3.4 DBI.ConnectionInfo

Defined in namespace DBI
Superclass: Object
Category: DBI-Framework
A utility class to contain connection info.

3.4.1 DBI.ConnectionInfo class: instance creation

fromDSN: aDSN
Parse a DSN in the format dbi:DriverName:dbname=database_name;host=hostname;port=port where dbi is constant, DriverName is the name of the driver, and everything else is parameters in the form name1=value1;name2=value2;...

3.4.2 DBI.ConnectionInfo: accessing

driver Answer the driver; this is not the driver class.

driver: aString
Set the driver; this is not the driver class.

paramString: aString
Set the parameter list.

params Return the parsed parameters in a Dictionary.

scheme Answer the scheme; the only supported one is 'dbi'.

scheme: aString
Set the scheme; the only supported one is 'dbi'.

3.5 DBI.FieldConverter

Defined in namespace DBI
Superclass: Object
Category: DBI

3.5.1 DBI.FieldConverter class: instance creation

new Not commented.

uniqueInstance
Not commented.

3.5.2 DBI.FieldConverter: actions

print: aValue on: aStream
Not commented.

printString: aValue
Not commented.

3.5.3 DBI.FieldConverter: converting-smalltalk

writeBoolean: aBoolean on: aStream
Not commented.

writeDate: aDate on: aStream
Not commented.

writeDateTime: aDateTime on: aStream
Not commented.

writeFloat: aFloat on: aStream
Not commented.

writeInteger: anInteger on: aStream
Not commented.

writeQuotedDate: aDate on: aStream
Not commented.

writeQuotedTime: aDate on: aStream
Not commented.

writeTime: aTime on: aStream
Not commented.

3.6 DBI.ResultSet

Defined in namespace DBI
Superclass: Stream
Category: DBI-Framework
I represent a result set, ie. the set of rows returned from a SELECT statement. I may also be returned for DML statements (INSERT, UPDATE, DELETE), in which case I only hold the number of rows affected.

3.6.1 DBI.ResultSet: accessing

columnAt: aIndex
Answer the aIndex'th column name.

columnNames
Answer an array of column names in order (abstract).

columns Answer a Dictionary of column name -> ColumnInfo pairs (abstract).

isDML Returns true if the statement was not a SELECT or similar operation (e.g. SHOW, DESCRIBE, EXPLAIN).

isSelect Returns true if the statement was a SELECT or similar operation (e.g. SHOW, DESCRIBE, EXPLAIN), false otherwise.

rowCount Returns the number of rows in the result set; error for DML statements.

rows Answer the contents of the execution result as array of Rows.

rowsAffected
For DML statments, returns the number of rows affected; error for SELECT statements.

statement Return the Statement, if any, that generated the result set.

3.6.2 DBI.ResultSet: cursor access

atEnd Return whether all the rows in the result set have been consumed. (abstract).

fetch Return the next row, or nil if at the end of the result set.

next Return the next row, or raise an error if at the end of the stream (abstract).

3.6.3 DBI.ResultSet: printing

printOn: aStream
Print a representation of the receiver on aStream.

3.6.4 DBI.ResultSet: stream protocol

position Returns the current row index (0-based) in the result set (abstract).

position: anInteger
Sets the current row index (0-based) in the result set (abstract).

size Returns the number of rows in the result set.

3.7 DBI.Row

Defined in namespace DBI
Superclass: Object
Category: DBI-Framework
I represent a row in a result set.

3.7.1 DBI.Row: accessing

asArray Return the values of the columns.

asDictionary
Return the names and values of the columns as a dictionary.

at: aColumnName
Return the value of the named column (abstract).

atIndex: aColumnIndex
Return the value of the column at the given 1-based index (abstract).

columnAt: aIndex
Return a ColumnInfo object for the aIndex-th column in the row.

columnCount
Return the number of columns in the row.

columnNames
Return an array of column names for the columns in the row.

columns Return a Dictionary of ColumnInfo objects for the columns in the row, where the keys are the column names.

keysAndValuesDo: aBlock
Pass to aBlock each column name and the corresponding value.

resultSet Return the result set that includes the receiver.

3.7.2 DBI.Row: printing

printOn: aStream
Print a representation of the receiver on aStream.

3.8 DBI.Statement

Defined in namespace DBI
Superclass: Object
Category: DBI-Framework
I represent a prepared statement.

3.8.1 DBI.Statement class: instance creation

on: aConnection
Return a new statement for this connection.

3.8.2 DBI.Statement: querying

execute Execute with no parameters (abstract).

executeWith: aParameter
Execute with one parameters.

executeWith: aParam1 with: aParam2
Execute with two parameters.

executeWith: aParam1 with: aParam2 with: aParam3
Execute with three parameters.

executeWithAll: aParams
Execute taking parameters from the Collection aParams (abstract).

3.9 DBI.Table

Defined in namespace DBI
Superclass: DBI.ROE.RASQLRelation
Category: DBI

3.9.1 DBI.Table: accessing

columnAt: aIndex
Answer the aIndex'th column name.

columnNames
Answer an array of column names in order (abstract).

columns Not commented.

database Returns the database name for this table. This corresponds to the catalog in SQL standard parlance.

3.9.2 DBI.Table: core

size Not commented.

3.9.3 DBI.Table: printing

print: anObject on: aStream
Not commented.

4 Controlling Smalltalk processes with DebugTools

4.1 Tree

Classes documented in this manual are **boldfaced**.

Object
  Debugger

4.2 Debugger

Defined in namespace Smalltalk
Superclass: Object
Category: System-Debugging

I provide debugging facilities for another inferior process. I have methods that allow the controlled process to proceed with varying granularity. In addition, I keep a cache mapping instruction pointer bytecodes to line numbers.

4.2.1 Debugger class: disabling debugging

debuggerClass
Answer 'nil'.

4.2.2 Debugger class: instance creation

on: aProcess
Suspend aProcess and return a new Debugger that controls aProcess. aProcess must not be the currently running process.

4.2.3 Debugger class: source code

currentLineIn: aContext
Not commented.

4.2.4 Debugger: inferior process properties

currentLine
Return the line number in traced process.

isActive Answer true if the inferior process is still running.

process Answer the inferior process.

suspendedContext
Answer the suspended execution state of the inferior process.

4.2.5 Debugger: stepping commands

continue Terminate the controlling process and continue execution of the traced process.

finish Run to the next return.

finish: aContext
Run up until aContext returns.

next Run to the end of the current line in the inferior process, skipping over message sends.

slowFinish Run in single-step mode up to the next return.

slowFinish: aContext
Run in single-step mode until aContext returns.

step Run to the end of the current line in the inferior process or to the next message send.

stepBytecode
Run a single bytecode in the inferior process.

stopInferior
Suspend the inferior process and raise a DebuggerReentered notification in the controlling process.

stopInferior: anObject
Suspend the inferior process and raise a DebuggerReentered notification in the controlling process with anObject as the exception's message.

5 Multilingual and international support with Iconv and I18N

5.1 Tree

Classes documented in this manual are **boldfaced**.

- *Object*
  - *Exception*
    - *Error*
      - **I18N.IncompleteSequenceError**
      - **I18N.InvalidSequenceError**
      - *SystemExceptions.InvalidValue*
        - *SystemExceptions.SystemExceptions.InvalidArgument*
          - **I18N.InvalidCharsetError**
  - *FileSegment*
    - **I18N.FileStreamSegment**
  - **I18N.EncodedStringFactory**
  - **I18N.LocaleData**
    - **I18N.LcMessagesDomain**
      - **I18N.LcMessagesCatalog**
        - **I18N.LcMessagesMoFileVersion0**
      - **I18N.LcMessagesDummyDomain**
      - **I18N.LcMessagesTerritoryDomain**
    - **I18N.Locale**
    - **I18N.LocaleConventions**
      - **I18N.LcMessages**
      - **I18N.LcPrintFormats**
        - **I18N.LcNumeric**
          - **I18N.LcMonetary**
            - **I18N.LcMonetaryISO**
        - **I18N.LcTime**
  - **I18N.RunTimeExpression**
    - **I18N.RTEAlternativeNode**
    - **I18N.RTEBinaryNode**
    - **I18N.RTELiteralNode**
    - **I18N.RTENegationNode**
    - **I18N.RTEParameterNode**
  - *Iterable*
    - *Collection*
      - *SequenceableCollection*
        - *ArrayedCollection*
          - *CharacterArray*
            - **I18N.EncodedString**
    - *Stream*
      - *FileDescriptor*

FileStream
I18N.BigEndianFileStream
I18N.EncodedStream
I18N.Encoder

5.2 I18N.BigEndianFileStream

Defined in namespace I18N
Superclass: FileStream
Category: i18n-Messages

Unlike ByteStream and FileStream, this retrieves integer numbers in big-endian (68000, PowerPC, SPARC) order.

5.3 I18N.EncodedStream

Defined in namespace I18N
Superclass: Stream
Category: i18n-Character sets

This class is a factory for subclasses of Encoder. Encoders act as parts of a pipe, hence this class provides methods that construct an appropriate pipe.

5.3.1 I18N.EncodedStream class: initializing

initialize Initialize the registry of the encoders to include the standard encoders contained in the library.

registerEncoderFor: arrayOfAliases toUTF32: toUTF32Class fromUTF32: fromUTF32Class

Register the two classes that will respectively convert from the charsets in arrayOfAliases to UTF-32 and vice versa.

The former class is a stream that accepts characters and returns (via #next) integers representing UTF-32 character codes, while the latter accepts UTF-32 character codes and converts them to characters. For an example see respectively FromUTF7 and ToUTF7 (I admit it is not a trivial example).

5.3.2 I18N.EncodedStream class: instance creation

encoding: anUnicodeString

Answer a pipe of encoders that converts anUnicodeString to default encoding for strings (the current locale's default charset if none is specified).

encoding: aStringOrStream as: toEncoding

Answer a pipe of encoders that converts anUnicodeString (which contains to the supplied encoding (which can be an ASCII String or Symbol).

on: aStringOrStream from: fromEncoding

Answer a pipe of encoders that converts aStringOrStream (which can be a string or another stream) from the given encoding to the default locale's default charset.

on: aStringOrStream from: fromEncoding to: toEncoding
Answer a pipe of encoders that converts aStringOrStream (which can be a string or another stream) between the two supplied encodings (which can be ASCII Strings or Symbols)

on: aStringOrStream to: toEncoding
Answer a pipe of encoders that converts aStringOrStream (which can be a string or another stream) from the default locale's default charset to the given encoding.

unicodeOn: aStringOrStream
Answer a pipe of encoders that converts aStringOrStream (which can be a string or another stream) from its encoding (or the current locale's default charset, if the encoding cannot be determined) to integers representing Unicode character codes.

unicodeOn: aStringOrStream encoding: fromEncoding
Answer a pipe of encoders that converts aStringOrStream (which can be a string or another stream) from the supplied encoding (which can be an ASCII String or Symbol) to integers representing Unicode character codes.

5.4 I18N.EncodedString

Defined in namespace I18N
Superclass: CharacterArray
Category: i18n-Character sets
An EncodedString, like a String, is a sequence of bytes representing a specific encoding of a UnicodeString. Unlike a String, however, the encoding name is known, rather than detected, irrelevant or assumed to be the system default.

5.4.1 I18N.EncodedString class: accessing

isUnicode Answer false; the receiver stores bytes (i.e. an encoded form), not characters.

5.4.2 I18N.EncodedString class: instance creation

fromString: aString
Not commented.

fromString: aString encoding: encoding
Not commented.

new This method should not be called for instances of this class.

new: size This method should not be called for instances of this class.

5.4.3 I18N.EncodedString: accessing

asString Answer 'string'.

asUnicodeString
Not commented.

at: anIndex
Not commented.

at: anIndex put: anObject
Not commented.

do: aBlock
Not commented.

encoding Not commented.

hash Not commented.

size Not commented.

species Not commented.

utf16Encoding
Not commented.

utf32Encoding
Not commented.

valueAt: anIndex
Not commented.

valueAt: anIndex put: anObject
Not commented.

5.4.4 I18N.EncodedString: copying

copy Not commented.

copyEmpty
Not commented.

copyEmpty: size
Not commented.

5.4.5 I18N.EncodedString: initializing

encoding: aString
Not commented.

setString: aString
Not commented.

5.4.6 I18N.EncodedString: printing

displayOn: aStream
Print a representation of the receiver on aStream. Unlike #printOn:, this method does not display the encoding and enclosing quotes.

printOn: aStream
Print a representation of the receiver on aStream.

5.5 I18N.EncodedStringFactory

Defined in namespace I18N
Superclass: Object
Category: i18n-Character sets

An EncodedStringFactory is used (in place of class objects) so that Encoders can return EncodedString objects with the correct encoding.

5.5.1 I18N.EncodedStringFactory class: instance creation

encoding: aString
Answer a new EncodedStringFactory, creating strings with the given encoding.

5.5.2 I18N.EncodedStringFactory: accessing

isUnicode Answer false; the receiver stores bytes (i.e. an encoded form), not characters.

5.5.3 I18N.EncodedStringFactory: instance creation

encoding Answer the encoding used for the created Strings.

encoding: aString
Set the encoding used for the created Strings.

fromString: aString
Answer an EncodedString based on aString and in the encoding represented by the receiver.

new Answer a new, empty EncodedString using the encoding represented by the receiver.

new: size Answer a new EncodedString of the given size, using the encoding represented by the receiver.

5.6 I18N.Encoder

Defined in namespace I18N
Superclass: Stream
Category: i18n-Character sets

This class is the superclass of streams that take an origin and encode it to another character set. The subclasses are are for internal use unless you are writing support for your own encodings.

5.6.1 I18N.Encoder class: instance creation

on: aStringOrStream from: fromEncoding to: toEncoding
Answer a new encoder that translates from fromEncoding to toEncoding. The encodings are guaranteed to be those for which the encoder was registered.

5.6.2 I18N.Encoder: stream operations

atEnd Return whether the receiver can produce another character in the receiver; by default, this is true if there is another character in the origin.

atEndOfInput
Return whether there is another character in the origin. This method is for private use by encoders, calling it outside won't corrupt the internal state of the encoder but the result probably won't be meaningful (depending on the innards of the encoder).

next Return the next character in the receiver; by default, this is the next character in the origin.

nextInput Return the next character in the origin. This method is for private use by encoders, calling it outside may corrupt the internal state of the encoder.

nextInputAvailable: n into: aCollection startingAt: pos
Place up to N characters from the origin in aCollection. This method is for private use by encoders, calling it outside may corrupt the internal state of the encoder.

peekInput Return the next character in the origin without advancing it.

species We answer a string of Characters encoded in our destination encoding.

5.7 I18N.FileStreamSegment

Defined in namespace I18N
Superclass: FileSegment
Category: i18n-Messages
Unlike FileSegment, this object assumes that the 'file' instance variable is a FileStream, not a file name.

5.7.1 I18N.FileStreamSegment: basic

fileName Answer the name of the file containing the segment

withFileDo: aBlock
Evaluate aBlock, passing a FileStream corresponding to the file

5.8 I18N.IncompleteSequenceError

Defined in namespace I18N
Superclass: Error
Category: i18n-Character sets
I am raised if an invalid sequence is found while converting a string from a charset to another. In particular, I am raised if the input stream ends abruptly in the middle of a multi-byte sequence.

5.8.1 I18N.IncompleteSequenceError: accessing

description
Answer a textual description of the exception.

5.9 I18N.InvalidCharsetError

Defined in namespace I18N
Superclass: SystemExceptions.SystemExceptions.InvalidArgument
Category: i18n-Character sets

I am raised if the user tries to encode from or to an unknown encoding

5.9.1 I18N.InvalidCharsetError: accessing

description
Answer a textual description of the exception.

5.10 I18N.InvalidSequenceError

Defined in namespace I18N
Superclass: Error
Category: i18n-Character sets

I am raised if an invalid sequence is found while converting a string from a charset to another

5.10.1 I18N.InvalidSequenceError: accessing

description
Answer a textual description of the exception.

5.11 I18N.LcMessages

Defined in namespace I18N
Superclass: I18N.LocaleConventions
Category: i18n-Messages

This object is a factory of LcMessagesDomain objects

5.11.1 I18N.LcMessages class: accessing

category Answer the environment variable used to determine the default locale

selector Answer the selector that accesses the receiver when sent to a Locale object.

5.11.2 I18N.LcMessages: accessing

languageDirectory
Answer the directory holding MO files for the language

languageDirectory: rootDirectory
Answer the directory holding MO files for the language, given the root directory of the locale data.

territoryDirectory
Answer the directory holding MO files for the language, specific to the territory

territoryDirectory: rootDirectory
Answer the directory holding MO files for the language, specific to the territory, given the root directory of the locale data.

5.11.3 I18N.LcMessages: opening MO files

? aString Answer an object for the aString domain, querying both the language catalog (e.g. pt) and the territory catalog (e.g. pt_BR or pt_PT).

domain: aString
Answer an object for the aString domain, querying both the language catalog (e.g. pt) and the territory catalog (e.g. pt_BR or pt_PT).

domain: aString localeDirectory: rootDirectory
Answer an object for the aString domain, querying both the language catalog (e.g. pt) and the territory catalog (e.g. pt_BR or pt_PT). The localeDirectory is usually '<installprefix>/share/locale'.

5.12 I18N.LcMessagesCatalog

Defined in namespace I18N
Superclass: I18N.LcMessagesDomain
Category: i18n-Messages
This object is an abstract superclass of objects that retrieve translated strings from a file.

5.13 I18N.LcMessagesDomain

Defined in namespace I18N
Superclass: I18N.LocaleData
Category: i18n-Messages
This object is an abstract superclass for message domains (catalogs). It contains methods to create instances of its subclasses, but they are commonly used only by LcMessages.

Translations are accessed using either #at: or the shortcut binary messages '?'. This way, common idioms to access translated strings will be

string := NLS? 'abc'. string := self? 'abc'.

(in the first case NLS is a class variable, in the second the receiver implements #? through delegation) which is only five or six characters longer than the traditional

string := 'abc'.

(cfr. the _("abc") idiom used by GNU gettext)

5.13.1 I18N.LcMessagesDomain class: opening MO files

id: anArray on: aFileName
Create an instance of the receiver with a given locale identifier from a path to the MO file

5.13.2 I18N.LcMessagesDomain: handling the cache

flush Flush the receiver's cache of translations

shouldCache
Answer whether translations should be cached. Never override this method to always answer false, because that would cause bugs when transliteration is being used.

5.13.3 I18N.LcMessagesDomain: querying

? aString Answer the translation of 'aString', or answer aString itself if none is available.

at: aString
Answer the translation of 'aString', or answer aString itself if none is available.

at: singularString plural: pluralString with: n
Answer either the translation of pluralString with '%1' replaced by n if n ~= 1, or the translation of singularString if n = 1.

at: aString put: anotherString
This method should not be called for instances of this class.

translatorInformation
Answer information on the translation, or nil if there is none. This information is stored as the 'translation' of an empty string.

translatorInformationAt: key
Answer information on the translation associated to a given key

translatorInformationAt: key at: subkey
Answer information on the translation associated to a given key and to a subkey of the key

5.14 I18N.LcMessagesDummyDomain

Defined in namespace I18N
Superclass: I18N.LcMessagesDomain
Category: i18n-Messages
This object does no attempt to translate strings, returning instead the same string passed as an argument to #?.

5.15 I18N.LcMessagesMoFileVersion0

Defined in namespace I18N
Superclass: I18N.LcMessagesCatalog
Category: i18n-Messages
This object is an concrete class that retrieves translated strings from a GNU gettext MO file. The class method #fileFormatDescription contains an explanation of the file format.

5.15.1 I18N.LcMessagesMoFileVersion0 class: documentation

fileFormatDescription
The Format of GNU MO Files (excerpt of the GNU gettext manual) ==

The format of the generated MO files is best described by a picture, which appears below.

The first two words serve the identification of the file. The magic number will always signal GNU MO files. The number is stored in the byte order of the generating machine, so the magic number really is two numbers: '0x950412de' and '0xde120495'. The second word describes the current revision of the file format. For now the revision is 0. This might change in future versions, and ensures that the readers of MO files can distinguish new formats from old ones, so that both can be handled correctly. The version is kept separate from the magic number, instead of using different magic numbers for different formats, mainly because '/etc/magic' is not updated often. It might be better to have magic separated from internal format version identification.

Follow a number of pointers to later tables in the file, allowing for the extension of the prefix part of MO files without having to recompile programs reading them. This might become useful for later inserting a few flag bits, indication about the charset used, new tables, or other things.

Then, at offset O and offset T in the picture, two tables of string descriptors can be found. In both tables, each string descriptor uses two 32 bits integers, one for the string length, another for the offset of the string in the MO file, counting in bytes from the start of the file. The first table contains descriptors for the original strings, and is sorted so the original strings are in increasing lexicographical order. The second table contains descriptors for the translated strings, and is parallel to the first table: to find the corresponding translation one has to access the array slot in the second array with the same index.

Having the original strings sorted enables the use of simple binary search, for when the MO file does not contain an hashing table, or for when it is not practical to use the hashing table provided in the MO file. This also has another advantage, as the empty string in a PO file GNU 'gettext' is usually *translated* into some system information attached to that particular MO file, and the empty string necessarily becomes the first in both the original and translated tables, making the system information very easy to find.

The size S of the hash table can be zero. In this case, the hash table itself is not contained in the MO file. Some people might prefer this because a precomputed hashing table takes disk space, and does not win *that* much speed. The hash table contains indices to the sorted array of strings in the MO file. Conflict resolution is done by double hashing. The precise hashing algorithm used is fairly dependent of GNU 'gettext' code, and is not documented here.

As for the strings themselves, they follow the hash file, and each is terminated with a <NUL>, and this <NUL> is not counted in the length which appears in the string descriptor. The 'msgfmt' program has an option selecting the alignment for MO file strings. With this option, each string is separately aligned so it starts at an offset which is a multiple of the alignment value. On some RISC machines, a correct alignment will speed things up.

Nothing prevents a MO file from having embedded <NUL>s in strings. However, the program interface currently used already presumes that strings are <NUL>

terminated, so embedded <NUL>s are somewhat useless. But MO file format is general enough so other interfaces would be later possible, if for example, we ever want to implement wide characters right in MO files, where <NUL> bytes may accidently appear.

This particular issue has been strongly debated in the GNU 'gettext' development forum, and it is expectable that MO file format will evolve or change over time. It is even possible that many formats may later be supported concurrently. But surely, we have to start somewhere, and the MO file format described here is a good start. Nothing is cast in concrete, and the format may later evolve fairly easily, so we should feel comfortable with the current approach.

byte +——————————————————+ 0 | magic number = 0x950412de | | | 4 | file format revision = 0 | | | 8 | number of strings | == N | | 12 | offset of table with original strings | == O | | 16 | offset of table with translation strings | == T | | 20 | size of hashing table | == S | | 24 | offset of hashing table | == H | | . . . (possibly more entries later) . . . | | O | length & offset 0th string ———————-. O + 8 | length & offset 1st string ————————. ... ... | | O + ((N-1)*8)| length & offset (N-1)th string | | | | | | | T | length & offset 0th translation ————————. T + 8 | length & offset 1st translation ————————. ... ... | | | | T + ((N-1)*8)| length & offset (N-1)th translation | | | | | | | | | | | H | start hash table | | | | | ... ... | | | | H + S * 4 | end hash table | | | | | | | | | | | | NUL terminated 0th string <———————-' | | | | | | | | | NUL terminated 1st string <————————' | | | | | | ... ... | | | | | | | NUL terminated 0th translation <———————' | | | | | NUL terminated 1st translation <————————' | | ... ... | | +——————————————————————+

Locating Message Catalog Files —————————————

Because many different languages for many different packages have to be stored we need some way to add these information to file message catalog files. The way usually used in Unix environments is have this encoding in the file name. This is also done here. The directory name given in 'bindtextdomain's second argument (or the default directory), followed by the value and name of the locale and the domain name are concatenated:

DIR_NAME/LOCALE/LC_CATEGORY/DOMAIN_NAME.mo

The default value for DIR_NAME is system specific. For the GNU library, and for packages adhering to its conventions, it's: /usr/local/share/locale

LOCALE is the value of the locale whose name is this 'LC_CATEGORY'. For 'gettext' and 'dgettext' this locale is always 'LC_MESSAGES'.

5.15.2 I18N.LcMessagesMoFileVersion0 class: plurals

initialize Initialize a table with the expressions computing the plurals for the most common languages

pluralExpressionFor: locale ifAbsent: aBlock
Answer a RunTimeExpression yielding the plural form for the given language and territory, if one is known, else evaluate aBlock and answer it.

5.15.3 I18N.LcMessagesMoFileVersion0: flushing the cache

flush Flush the cache and reread the catalog's metadata.

shouldCache
Answer true, we always cache translations if they are read from a file

5.16 I18N.LcMessagesTerritoryDomain

Defined in namespace I18N
Superclass: I18N.LcMessagesDomain
Category: i18n-Messages
This object asks for strings to a primary domain (e.g. it_IT) and a secondary one (e.g. it).

5.16.1 I18N.LcMessagesTerritoryDomain class: instance creation

primary: domain1 secondary: domain2
Answer an instance of the receiver that queries, in sequence, domain1 and domain2

5.17 I18N.LcMonetary

Defined in namespace I18N
Superclass: I18N.LcNumeric
Category: i18n-Printing
Sending either #?, #printString: or #print:on: converts a Number to a String according to the rules that are mandated by ISO for printing currency amounts in the current locale.

5.17.1 I18N.LcMonetary class: accessing

category Answer the environment variable used to determine the default locale

selector Answer the selector that accesses the receiver when sent to a Locale object.

5.17.2 I18N.LcMonetary: printing

print: aNumber on: aStream
Print aNumber on aStream according to the receiver's formatting conventions. Always print a currency sign and don't force to print negative numbers by putting parentheses around them.

print: aNumber on: aStream currency: currency parentheses: p
Print aNumber on aStream according to the receiver's formatting conventions. If currency is true, print a currency sign, and if p is true force to print negative numbers by putting parentheses around them. If p is true, for positive numbers spaces are put around the number to keep them aligned.

5.18 I18N.LcMonetaryISO

Defined in namespace I18N
Superclass: I18N.LcMonetary
Category: i18n-Printing

5.18.1 I18N.LcMonetaryISO class: accessing

selector Answer the selector that accesses the receiver when sent to a Locale object.

5.19 I18N.LcNumeric

Defined in namespace I18N
Superclass: I18N.LcPrintFormats
Category: i18n-Printing

Sending either #?, #printString: or #print:on: converts a Number to a String according to the rules that are used in the given locale.

5.19.1 I18N.LcNumeric class: accessing

category Answer the environment variable used to determine the default locale

selector Answer the selector that accesses the receiver when sent to a Locale object.

5.19.2 I18N.LcNumeric: printing

basicPrint: aNumber on: aStream
Print aNumber on aStream according to the receiver's formatting conventions, without currency signs or anything like that. This method must not be overridden.

print: aNumber on: aStream
Print aNumber on aStream according to the receiver's formatting conventions.

5.20 I18N.LcPrintFormats

Defined in namespace I18N
Superclass: I18N.LocaleConventions
Category: i18n-Messages

LcPrintFormats subclasses have instances that understand #?, #printString: and #print:on: (the last of which is abstract) which provide a means to convert miscellaneous objects to Strings according to the rules that are used in the given locale.

5.20.1 I18N.LcPrintFormats: printing

? anObject
Answer how anObject must be printed according to the receiver's formatting conventions.

print: anObject on: aStream
Print anObject on aStream according to the receiver's formatting conventions.

printString: anObject
Answer how anObject must be printed according to the receiver's formatting conventions.

5.21 I18N.LcTime

Defined in namespace I18N
Superclass: I18N.LcPrintFormats
Category: i18n-Printing
Sending either #?, #printString: or #print:on: converts a Date or Time to a String according to the rules that are used in the given locale.

5.21.1 I18N.LcTime class: accessing

category Answer the environment variable used to determine the default locale

selector Answer the selector that accesses the receiver when sent to a Locale object.

5.21.2 I18N.LcTime: printing

print: aDateOrTimeOrArray on: aStream
Print aDateOrTimeOrArray on aStream according to the receiver's formatting conventions. It can be a Date, Time, DateTime, or an array made of a Date and a Time

print: aDateOrTimeOrArray on: aStream ifFull: fullFmt ifDate: dateFmt ifTime: timeFmt
Print aDateOrTimeOrArray on aStream according to the receiver's formatting conventions. It can be a Date, Time, DateTime, or an array made of a Date and a Time: Date is printed with dateFmt and Time with timeFmt, while in the other cases fullFmt is used. For information on the formatting codes, see #print:time:format:on:.

print: aDate time: aTime format: aString on: aStream
Print the specified date and time on aStream according to the receiver's formatting conventions, using the given format. The valid abbreviations are the same used by the C function strftime: abbreviated weekday (%a) weekday (%A) abbreviated month (%b) month (%B) date & time (%c) century (%C) day of the month (%d) date (US) (%D) day of the month (%e) year for the ISO week (%g) year for the ISO week (%G) abbreviated month (%h) hours (%H) hours (AM/PM) (%I) day of the year (%j) hours (%k) hours (AM/PM) (%l) month (%m) minutes (%M) AM/PM (%p) lowercase AM/PM (%P) AM/PM time (%r) time (US) (%R) time_t (%s) seconds (%S) time (US) (%T) day of the week (%u) week number starting at Sun (%U) week number starting at Thu (%V) day of the week, Sunday=0 (%w) week number starting at Mon (%W) date (%x) time (%X) year (2-digit) (%y) year (4-digit) (%Y).

5.21.3 I18N.LcTime: tests

allFormatsExample
Answer a long string that includes all the possible formats

5.22 I18N.Locale

Defined in namespace I18N
Superclass: I18N.LocaleData
Category: i18n-Messages

This object is an abstract superclass of objects related to the territory and language in which the program is being used. Instances of it are asked about information on the current locale, and provide a means to be asked for things with a common idiom, the #? binary message.

5.22.1 I18N.Locale class: C call-outs

primRootDirectory
Not commented.

5.22.2 I18N.Locale class: initialization

rootDirectory
Answer the directory under which locale definition files are found.

rootDirectory: aString
Set under which directory locale definition files are found.

5.22.3 I18N.Locale class: instance creation

default Answer an instance of the receiver that accesses the default locale.

flush Flush the information on locales that are not valid across an image save/load.

fromString: aString
Answer an instance of the receiver that accesses the given locale (in the form language[_territory][.charset]).

posix Answer an instance of the receiver that accesses the POSIX locale.

5.22.4 I18N.Locale: C call-outs

load: name
Not commented.

5.22.5 I18N.Locale: subobjects

messages Answer the LcMessages object for the locale represented by the receiver.

monetary Answer the LcMonetary object for the locale represented by the receiver.

monetaryIso
Answer the LcMonetaryISO object for the locale represented by the receiver.

numeric Answer the LcNumeric object for the locale represented by the receiver.

time Answer the LcTime object for the locale represented by the receiver.

5.23 I18N.LocaleConventions

Defined in namespace I18N
Superclass: I18N.LocaleData
Category: i18n-Messages

I am an abstract superclass of objects that are referred to by a Locale object.

5.23.1 I18N.LocaleConventions class: accessing

? anObject
Query the default object, forwarding the message to it.

default Answer an instance of the receiver that accesses the default locale.

fromString: aString
Answer an instance of the receiver that accesses the given locale (in the form language[_territory][.charset]).

posix Answer an instance of the receiver that accesses the POSIX locale.

selector This method's functionality should be implemented by subclasses of LocaleConventions

5.23.2 I18N.LocaleConventions: accessing

? anObject
This method's functionality should be implemented by subclasses of LocaleConventions

5.24 I18N.LocaleData

Defined in namespace I18N
Superclass: Object
Category: i18n-Messages

I am an abstract superclass of objects that represent localization information.

5.24.1 I18N.LocaleData class: accessing

category Answer 'nil'.

default This method's functionality should be implemented by subclasses of LocaleData

flush Flush the contents of the instances of each subclass of LocaleData.

fromString: lang
This method's functionality should be implemented by subclasses of LocaleData

language: lang
Answer the local object for the given language.

language: lang territory: territory
Answer the local object for the given language and territory.

language: lang territory: territory charset: charset
Answer the local object for the given language, territory and charset.

new This method should not be called for instances of this class.

posix This method's functionality should be implemented by subclasses of LocaleData

update: aspect
Flush instances of the receiver when an image is loaded.

5.24.2 I18N.LocaleData class: database

defaultCharset
Answer the default charset used when nothing is specified.

defaultCharset: aString
Set the default charset used when nothing is specified.

defaults Answer the default territory-language and language-charset associations.

initialize Initialize the receiver's class variables.

languages ISO639 language codes

territories ISO3166 territory codes

5.24.3 I18N.LocaleData: accessing

charset Return the charset supported by the receiver.

id Return the identifier of the locale supported by the receiver.

isPosixLocale
Answer whether the receiver implements the default POSIX behavior for a locale.

language Return the language supported by the receiver.

languageDirectory
Answer the directory where data files for the current language reside.

languageDirectory: rootDirectory
Answer the directory where data files for the current language reside, given the root directory of the locale data.

territory Return the territory supported by the receiver.

territoryDirectory
Answer the directory where data files for the current language, specific to the territory, reside.

territoryDirectory: rootDirectory
Answer the directory where data files for the current language, specific to the territory, reside, given the root directory of the locale data.

5.24.4 I18N.LocaleData: initialization

id: anArray
Private - Set which locale the receiver contains data for

initialize: aString
Set which locale the receiver contains data for, starting from a string describing the locale.

5.25 I18N.RTEAlternativeNode

Defined in namespace I18N
Superclass: I18N.RunTimeExpression
Category: i18n-Messages

5.25.1 I18N.RTEAlternativeNode class: compiling

condition: cond ifTrue: trueNode ifFalse: falseNode
Private - Create a node in the parse tree for the run-time expression, mapping s to a Smalltalk arithmetic selector

5.25.2 I18N.RTEAlternativeNode: computing

condition: condNode ifTrue: trueNode ifFalse: falseNode
Initialize the children of the receiver and the conditional expression to choose between them

printOn: aStream
Print a representation of the receiver on aStream

send: parameter
Evaluate the receiver by conditionally choosing one of its children and evaluating it

5.26 I18N.RTEBinaryNode

Defined in namespace I18N
Superclass: I18N.RunTimeExpression
Category: i18n-Messages

5.26.1 I18N.RTEBinaryNode class: compiling

lhs: lhs op: op rhs: rhs
Private - Create a node in the parse tree for the run-time expression, mapping s to a Smalltalk arithmetic selector

5.26.2 I18N.RTEBinaryNode: compiling

lhs Answer 'lhs'.

op Answer 'op'.

rhs Answer 'rhs'.

5.26.3 I18N.RTEBinaryNode: computing

lhs: lhsNode op: aSymbol rhs: rhsNode
Initialize the children of the receiver and the operation to be done between them

printOn: aStream
Print a representation of the receiver on aStream

send: parameter
Private - Evaluate the receiver by evaluating both children and performing an arithmetic operation between them.

5.27 I18N.RTELiteralNode

Defined in namespace I18N
Superclass: I18N.RunTimeExpression
Category: i18n-Messages

5.27.1 I18N.RTELiteralNode class: initializing

parseFrom: aStream
Parse a literal number from aStream and return a new node

5.27.2 I18N.RTELiteralNode: computing

n: value Set the value of the literal that the node represents

printOn: aStream
Print a representation of the receiver on aStream

send: parameter
Answer a fixed value, the literal encoded in the node

5.28 I18N.RTENegationNode

Defined in namespace I18N
Superclass: I18N.RunTimeExpression
Category: i18n-Messages

5.28.1 I18N.RTENegationNode class: initializing

child: aNode
Answer a new node representing the logical negation of aNode

5.28.2 I18N.RTENegationNode: computing

child: value
Set the child of which the receiver will compute the negation

printOn: aStream
Print a representation of the receiver on aStream

send: parameter
Evaluate the receiver by computing the child's logical negation

5.29 I18N.RTEParameterNode

Defined in namespace I18N
Superclass: I18N.RunTimeExpression
Category: i18n-Messages

5.29.1 I18N.RTEParameterNode: computing

printOn: aStream
: Print a representation of the receiver on aStream

send: parameter
: Evaluate the receiver by answering the parameter

5.30 I18N.RunTimeExpression

Defined in namespace I18N
Superclass: Object
Category: i18n-Messages

5.30.1 I18N.RunTimeExpression class: compiling

parseExpression: stream
: Private - Compile the expression in the stream

parseOperand: stream
: Parse an operand from the stream (i.e. an unary negation, a parenthesized subexpression, ‘n’ or a number) and answer the corresponding parse node.

parseOperator: stream
: Answer a Symbol for an operator read from stream, or nil if something else is found.

5.30.2 I18N.RunTimeExpression class: initializing

initialize Private - Initialize internal tables for the parser

5.30.3 I18N.RunTimeExpression class: instance creation

on: aString
: Compile aString and answer a RunTimeExpression

5.30.4 I18N.RunTimeExpression: computing

send: parameter
: This method’s functionality should be implemented by subclasses of RunTime-Expression

value: parameter
: Evaluate the receiver, and answer its value as an integer

6 Network programming with Sockets

6.1 Tree

Classes documented in this manual are **boldfaced**.

- *Object*
  - *CObject*
    - *CCompound*
      - *CStruct*
        - **Sockets.CAddrInfoStruct**
        - **Sockets.CSockAddrIn6Struct**
  - *Iterable*
    - *Stream*
      - *FileDescriptor*
        - **Sockets.AbstractSocketImpl**
          - **Sockets.DatagramSocketImpl**
            - **Sockets.MulticastSocketImpl**
              - **Sockets.UDPSocketImpl**
            - **Sockets.OOBSocketImpl**
            - **Sockets.RawSocketImpl**
              - **Sockets.ICMP6SocketImpl**
              - **Sockets.ICMPSocketImpl**
            - **Sockets.UnixDatagramSocketImpl**
          - **Sockets.SocketImpl**
            - **Sockets.TCPSocketImpl**
            - **Sockets.UnixSocketImpl**
      - *PositionableStream*
        - *ReadStream*
          - **Sockets.ReadBuffer**
        - *WriteStream*
          - **Sockets.WriteBuffer**
      - **Sockets.AbstractSocket**
        - **Sockets.DatagramSocket**
          - **Sockets.MulticastSocket**
        - **Sockets.ServerSocket**
        - **Sockets.StreamSocket**
          - **Sockets.Socket**
      - **Sockets.DummyStream**
  - **Sockets.Datagram**
  - **Sockets.SocketAddress**
    - **Sockets.IP6Address**
    - **Sockets.IPAddress**
    - **Sockets.UnixAddress**

6.2 Sockets.AbstractSocket

Defined in namespace Sockets
Superclass: Stream
Category: Sockets-Streams

This class models a client site socket. A socket is a TCP/IP endpoint for network communications conceptually similar to a file handle.

This class only takes care of buffering and blocking if requested. It uses an underlying socket implementation object which is a subclass of AbstractSocketImpl. This is necessary to hide some methods in FileDescriptor that are not relevant to sockets, as well as to implement buffering independently of the implementation nuances required by the different address families. The address family class (a subclass of SocketAddress) acts as a factory for socket implementation objects.

6.2.1 Sockets.AbstractSocket class: defaults

defaultAddressClass
Answer the default address family to be used. In the library, the address family is represented by a subclass of SocketAddress which is by default IPAddress.

defaultAddressClass: class
Set the default address family to be used. In the library, the address family is represented by a subclass of SocketAddress which is by default IPAddress.

defaultImplementationClassFor: aSocketAddressClass
Answer the default implementation class. Depending on the subclass, this might be the default stream socket implementation class of the given address class, or rather its default datagram socket implementation class.

6.2.2 Sockets.AbstractSocket class: instance creation

new This method should not be called for instances of this class.

new: implementation
Answer a new instance of the receiver, using as the underlying layer the object passed as the 'implementation' parameter; the object is probably going to be some kind of AbstractSocketImpl.

new: implClass addressClass: addressClass
Answer a new instance of the receiver, using as the underlying layer a new instance of 'implementationClass' and using the protocol family of 'addressClass'.

6.2.3 Sockets.AbstractSocket class: timed-out operations

checkPeriod
Answer the period that is to elapse between socket polls if data data is not ready and the connection is still open (in milliseconds)

checkPeriod: anInteger
Set the period that is to elapse between socket polls if data data is not ready and the connection is still open (in milliseconds)

timeout Answer the period that is to elapse between the request for (yet unavailable) data and the moment when the connection is considered dead (in milliseconds)

timeout: anInteger
Set the period that is to elapse between the request for (yet unavailable) data and the moment when the connection is considered dead (in milliseconds)

6.2.4 Sockets.AbstractSocket class: well known ports

defaultPortAt: protocol
Answer the port that is used (by default) for the given service (high level protocol)

defaultPortAt: protocol ifAbsent: port
Answer the port that is used (by default) for the given service (high level protocol), or the specified port if none is registered.

defaultPortAt: protocol put: port
Associate the given port to the service specified by 'protocol'.

portCmdServer
Answer the port on which the rsh daemon listens

portDNS Answer the port on which the DNS listens

portDayTime
Answer the port on which the TOD service listens

portDiscard
Answer the port on which the DISCARD service listens

portEcho Answer the port on which the ECHO service listens

portExecServer
Answer the port on which the exec server listens

portFTP Answer the port on which the FTP daemon listens

portFinger
Answer the port on which the finger daemon listens

portGopher
Answer the port on which the Gopher daemon listens

portHTTP
Answer the port on which the http daemon listens

portLoginServer
Answer the port on which the rlogin daemon listens

portNNTP
Answer the port on which the nntp daemon listens

portNetStat
Answer the port on which the NETSTAT service listens

portPOP3 Answer the port on which the pop3 daemon listens

portReserved
Answer the last port reserved to privileged processes

portSMTP
Answer the port on which the SMTP daemon listens

portSSH Answer the port on which the SSH daemon listens

portSystat
Answer the port on which the SYSTAT service listens

portTelnet
Answer the port on which the TELNET daemon listens

portTimeServer
Answer the port on which the time server listens

portWhois
Answer the port on which the WHOIS daemon listens

6.2.5 Sockets.AbstractSocket: accessing

address Answer an IP address that is of common interest (this can be either the local or the remote address, according to the definition in the subclass).

available Answer whether there is data available on the socket. Same as #canRead, present for backwards compatibility.

canRead Answer whether there is data available on the socket.

canWrite Answer whether there is free space in the socket's write buffer.

close Close the socket represented by the receiver.

flush Flush any buffers used by the receiver.

isOpen Answer whether the connection between the receiver and the remote endpoint is still alive.

isPeerAlive
Answer whether the connection with the peer remote machine is still valid.

localAddress
Answer the local IP address of the socket.

localPort Answer the local IP port of the socket.

port Answer an IP port that is of common interest (this can be the port for either the local or remote endpoint, according to the definitions in the subclass

remoteAddress
Answer the IP address of the socket's remote endpoint.

remotePort
Answer the IP port of the socket's remote endpoint.

6.2.6 Sockets.AbstractSocket: printing

printOn: aStream
Print a representation of the receiver on aStream

6.2.7 Sockets.AbstractSocket: socket options

soLinger
Answer the number of seconds that the socket is allowed to wait if it promises reliable delivery but has unacknowledged/untransmitted packets when it is closed, or nil if those packets are left to their destiny or discarded.

soLinger: linger
Set the number of seconds that the socket is allowed to wait if it promises reliable delivery but has unacknowledged/untransmitted packets when it is closed.

soLingerOff
Specify that, even if the socket promises reliable delivery, any packets that are unacknowledged/untransmitted when it is closed are to be left to their destiny or discarded.

species
Answer 'String'.

6.2.8 Sockets.AbstractSocket: stream protocol

atEnd
By default, answer whether the connection is still open.

next
Read another character from the socket, failing if the connection is dead.

next: n putAll: aCollection startingAt: pos
Write 'char' to the socket, failing if the connection is dead. The SIGPIPE signal is automatically caught and ignored by the system.

nextPut: char
Write 'char' to the socket, failing if the connection is dead. The SIGPIPE signal is automatically caught and ignored by the system.

6.2.9 Sockets.AbstractSocket: testing

isExternalStream
Answer whether the receiver streams on a file or socket.

6.3 Sockets.AbstractSocketImpl

Defined in namespace Sockets
Superclass: FileDescriptor
Category: Sockets-Protocols

This abstract class serves as the parent class for socket implementations. The implementation class serves an intermediary to routines that perform the actual socket operations. It hides the buffering and blocking behavior of the Socket classes.

A default implementation is provided by each address family, but this can be changed by class methods on SocketAddress sublcasses.

6.3.1 Sockets.AbstractSocketImpl class: abstract

addressClass
Answer the class responsible for handling addresses for the receiver

protocol Answer the protocol parameter for 'create'

socketType
Answer the socket type parameter for 'create'.

6.3.2 Sockets.AbstractSocketImpl class: C call-outs

accept: socket peer: peer addrLen: len
Not commented.

bind: socket to: addr addrLen: len
Not commented.

connect: socket to: addr addrLen: len
Not commented.

create: family type: type protocol: protocol
Not commented.

getPeerName: socket addr: addr addrLen: len
Not commented.

getSockName: socket addr: addr addrLen: len
Not commented.

listen: socket log: len
Not commented.

option: socket level: level at: name get: value size: len
Not commented.

option: socket level: level at: name put: value size: len
Not commented.

receive: socket buffer: buf size: len flags: flags from: addr size: addrLen
Not commented.

send: socket buffer: buf size: len flags: flags to: addr size: addrLen
Not commented.

6.3.3 Sockets.AbstractSocketImpl class: C constants

soLinger Not commented.

soReuseAddr
Not commented.

sockDgram
Not commented.

sockRDM Not commented.

sockRaw Not commented.

sockStream
Not commented.

solSocket Not commented.

6.3.4 Sockets.AbstractSocketImpl class: socket creation

newFor: addressClass
Create a socket for the receiver.

6.3.5 Sockets.AbstractSocketImpl: accessing

connectTo: ipAddress port: port
Connect the receiver to the given IP address and port. ‘Connecting’ means attaching the remote endpoint of the socket.

localAddress
Answer the address of the local endpoint of the socket (even if IP is not being used, this identifies the machine that is bound to the socket).

localPort Answer the port of the local endpoint of the socket (even if IP is not being used, this identifies the service or process that is bound to the socket).

remoteAddress
Answer the address of the remote endpoint of the socket (even if IP is not being used, this identifies the machine to which the socket is connected).

remotePort
Answer the port of the remote endpoint of the socket (even if IP is not being used, this identifies the service or process to which the socket is connected).

6.3.6 Sockets.AbstractSocketImpl: asynchronous operations

ensureReadable
If the file is open, wait until data can be read from it. The wait allows other Processes to run.

ensureWriteable
If the file is open, wait until we can write to it. The wait allows other Processes to run.

waitForException
If the file is open, wait until an exceptional condition (such as presence of out of band data) has occurred on it. The wait allows other Processes to run.

6.3.7 Sockets.AbstractSocketImpl: C call-outs

accept: socket peer: peer addrLen: len
Not commented.

bind: socket to: addr addrLen: len
Not commented.

connect: socket to: addr addrLen: len
Not commented.

create: family type: type protocol: protocol
Not commented.

getPeerName: socket addr: addr addrLen: len
Not commented.

getSockName: socket addr: addr addrLen: len
Not commented.

listen: socket log: len
Not commented.

option: socket level: level at: name get: value size: len
Not commented.

option: socket level: level at: name put: value size: len
Not commented.

receive: socket buffer: buf size: len flags: flags from: addr size: addrLen
Not commented.

send: socket buffer: buf size: len flags: flags to: addr size: addrLen
Not commented.

6.3.8 Sockets.AbstractSocketImpl: C constants

soError: socket
Not commented.

6.3.9 Sockets.AbstractSocketImpl: socket operations

accept: implementationClass
Accept a connection on the receiver, and create a new instance of implementationClass that will deal with the newly created active server socket.

bindTo: ipAddress port: port
Bind the receiver to the given IP address and port. 'Binding' means attaching the local endpoint of the socket.

fileOp: ioFuncIndex
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1 with: arg2
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 with: arg2 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

getSockName
: Retrieve a ByteArray containing a sockaddr_in struct for the local endpoint of the socket.

listen: backlog
: Make the receiver a passive server socket with a pending connections queue of the given size.

6.3.10 Sockets.AbstractSocketImpl: socket options

optionAt: opt level: level put: anObject
: Modify the value of a socket option. The option identifier is in 'opt' and the level is in 'level'. anObject can be a boolean, integer, socket address or ByteArray. A layer over this method is provided for the most common socket options, so this will be rarely used.

optionAt: opt level: level size: size
: Answer in a ByteArray of the given size the value of a socket option. The option identifier is in 'opt' and the level is in 'level'. A layer over this method is provided for the most common socket options, so this will be rarely used.

soLinger
: Answer the number of seconds by which a 'close' operation can block to ensure that all the packets have reliably reached the destination, or nil if those packets are left to their destiny.

soLinger: linger
: Set the number of seconds by which a 'close' operation can block to ensure that all the packets have reliably reached the destination. If linger is nil, those packets are left to their destiny.

soReuseAddr
: Answer whether another socket can be bound the same local address as this one. If you enable this option, you can actually have two sockets with the same Internet port number; but the system won't allow you to use the two identically-named sockets in a way that would confuse the Internet. The reason for this option is that some higher-level Internet protocols, including FTP, require you to keep reusing the same socket number.

soReuseAddr: aBoolean
: Set whether another socket can be bound the same local address as this one.

valueWithoutBuffering: aBlock
: Evaluate aBlock, ensuring that any data that it writes to the socket is sent immediately to the network.

6.4 Sockets.CAddrInfoStruct

Defined in namespace Sockets
Superclass: CStruct
Category:

6.4.1 Sockets.CAddrInfoStruct class: C call-outs

getaddrinfo: name service: servname hints: hints result: res
Not commented.

6.4.2 Sockets.CAddrInfoStruct: C call-outs

aiAddr Not commented.

aiCanonname
Not commented.

free Not commented.

6.4.3 Sockets.CAddrInfoStruct: C function wrappers

getaddrinfo: name
Not commented.

getaddrinfo: name service: service
Not commented.

6.5 Sockets.CSockAddrIn6Struct

Defined in namespace Sockets
Superclass: CStruct
Category:

6.6 Sockets.Datagram

Defined in namespace Sockets
Superclass: Object
Category: Sockets-Protocols
This class models a packet of data that is to be sent across the network using a connectionless protocol such as UDP. It contains the data to be send, as well as the destination address and port. Note that datagram packets can arrive in any order and are not guaranteed to be delivered at all.

This class can also be used for receiving data from the network.

6.6.1 Sockets.Datagram class: instance creation

data: aByteArray
Answer a new datagram with the specified data.

data: aByteArray address: ipAddress port: port
Answer a new datagram with the specified target socket, and aByteArray as its data.

object: object address: ipAddress port: port
Serialize the object onto a ByteArray, and create a Datagram with the object as its contents, and the specified receiver. Note that each invocation of this method creates a separate ObjectDumper; if different objects that you're

sending are likely to contain references to the same objects, you should use #object:objectDumper:address:port:.

object: object objectDumper: od address: ipAddress port: port
Serialize the object onto a ByteArray, and create a Datagram with the object as its contents, and the specified receiver. Serialization takes place through ObjectDumper passed as 'od', and the stream attached to the ObjectDumper is resetted every time. Using this method is indicated if different objects that you're sending are likely to contain references to the same objects.

6.6.2 Sockets.Datagram: accessing

address Answer the address of the target socket

address: ipAddress
Set the address of the target socket

data Answer the data attached to the datagram

data: aByteArray
Set the data attached to the datagram

dataSize Answer the size of the message.

dataSize: aSize
I am called to update the size...

get Parse the data attached to the datagram through a newly created ObjectDumper, and answer the resulting object. This method is complementary to #object:address:port:.

getThrough: objectDumper
Parse the data attached to the datagram through the given ObjectDumper without touching the stream to which it is attached, and answer the resulting object. The state of the ObjectDumper, though, is updated. This method is complementary to #object:objectDumper:address:port:.

port Answer the IP port of the target socket

port: thePort
Set the IP port of the target socket

size I determine the size of the datagram. It is either an explicitly specified dataSize, or the size of the whole collection.

6.7 Sockets.DatagramSocket

Defined in namespace Sockets
Superclass: Sockets.AbstractSocket
Category: Sockets-Streams
This class models a connectionless datagram socket that sends individual packets of data across the network. In the TCP/IP world, this means UDP. Datagram packets do not have guaranteed delivery, or any guarantee about the order the data will be received on the remote host.

This class uses an underlying socket implementation object which is a subclass of DatagramSocketImpl. This is less necessary for datagram sockets than for stream sockets (except for hiding some methods in FileDescriptor that are not relevant to sockets), but it is done for cleanliness and symmetry.

6.7.1 Sockets.DatagramSocket class: accessing

defaultBufferSize
Answer the default maximum size for input datagrams.

defaultBufferSize: size
Set the default maximum size for input datagrams.

defaultImplementationClassFor: aSocketAddressClass
Answer the default implementation class. Depending on the subclass, this might be the default stream socket implementation class of the given address class, or rather its default datagram socket implementation class.

6.7.2 Sockets.DatagramSocket class: initialization

initialize Initialize the class to use an input datagram size of 128.

6.7.3 Sockets.DatagramSocket class: instance creation

local: ipAddressOrString port: remotePort
Create a new socket and bind it to the given host (passed as a String to be resolved or as an IPAddress), on the given port.

new Answer a new datagram socket (by default an UDP socket), without a specified local address and port.

port: localPort
Create a new socket and bind it to the local host on the given port.

remote: ipAddressOrString port: remotePort local: ipAddress port: localPort
Create a new socket and bind it to the given host (passed as a String to be resolved or as a SocketAddress), and to the given remotePort. The default destination for the datagrams will be ipAddressOrString (if not nil), on the remotePort port.

6.7.4 Sockets.DatagramSocket: accessing

address Answer the local address.

bufferSize Answer the size of the buffer in which datagrams are stored.

bufferSize: size
Set the size of the buffer in which datagrams are stored.

datagramClass
Answer the class used by the socket to return datagrams.

next Read a datagram on the socket and answer it.

nextPut: aDatagram
Send the given datagram on the socket.

peek Peek for a datagram on the socket and answer it.

peek: datagram
Peek for a datagram on the socket, store it in 'datagram', and answer the datagram itself.

port Answer the local port.

receive: datagram
Read a datagram from the socket, store it in 'datagram', and answer the datagram itself.

6.7.5 Sockets.DatagramSocket: direct operations

nextFrom: ipAddress port: port
Answer the next datagram from the given address and port.

6.8 Sockets.DatagramSocketImpl

Defined in namespace Sockets
Superclass: Sockets.AbstractSocketImpl
Category: Sockets-Protocols
This abstract class serves as the parent class for datagram socket implementations.

6.8.1 Sockets.DatagramSocketImpl class: parameters

datagramClass
Answer the datagram class returned by default by instances of this class.

socketType
Answer the socket type parameter for 'create'.

6.8.2 Sockets.DatagramSocketImpl: accessing

bufferSize Answer the size of the buffer in which datagrams are stored.

bufferSize: size
Set the size of the buffer in which datagrams are stored.

6.8.3 Sockets.DatagramSocketImpl: C constants

ipAddMembership
Not commented.

ipDropMembership
Not commented.

ipMulticastIf
Not commented.

ipMulticastTtl
Not commented.

msgPeek Not commented.

6.8.4 Sockets.DatagramSocketImpl: socket operations

next Retrieve a datagram from the receiver, answer a new object of the receiver's datagram class.

nextPut: aDatagram
Send aDatagram on the socket

peek Peek for a datagram on the receiver, answer a new object of the receiver's datagram class.

peek: aDatagram
Peek for a datagram on the receiver, answer aDatagram modified to contain information on the newly received datagram.

receive: aDatagram
Retrieve a datagram from the receiver, answer aDatagram modified to contain information on the newly received datagram.

receive: flags datagram: aDatagram
Receive a new datagram into 'datagram', with the given flags, and answer 'datagram' itself; this is an abstract method. The flags can be zero to receive the datagram, or 'self msgPeek' to only peek for it without removing it from the queue.

send: aDatagram to: theReceiver port: port
Send aDatagram on the socket to the given receiver and port

6.9 Sockets.DummyStream

Defined in namespace Sockets
Superclass: Stream
Category: Sockets-Tests

6.10 Sockets.ICMP6SocketImpl

Defined in namespace Sockets
Superclass: Sockets.RawSocketImpl
Category: Sockets-Protocols
Unless the application installs its own implementation, this is the default socket implementation that will be used for IPv6 raw sockets. It uses C call-outs to implement standard BSD style sockets of family AF_INET, type SOCK_RAW, protocol IPPROTO_ICMPV6.

6.10.1 Sockets.ICMP6SocketImpl class: C constants

protocol Not commented.

6.11 Sockets.ICMPSocketImpl

Defined in namespace Sockets
Superclass: Sockets.RawSocketImpl
Category: Sockets-Protocols

Unless the application installs its own implementation, this is the default socket implementation that will be used for IPv4 raw sockets. It uses C call-outs to implement standard BSD style sockets of family AF_INET, type SOCK_RAW, protocol IPPROTO_ICMP.

6.11.1 Sockets.ICMPSocketImpl class: C constants

protocol Not commented.

6.12 Sockets.IP6Address

Defined in namespace Sockets
Superclass: Sockets.SocketAddress
Category: Sockets-Protocols

This class models an IPv6 address. It also acts as a factory for IPv6 stream (TCP), datagram (UDP) and raw sockets.

6.12.1 Sockets.IP6Address class: C constants

addressFamily
Not commented.

aiAll Not commented.

aiV4mapped
Not commented.

protocolFamily
Not commented.

6.12.2 Sockets.IP6Address class: constants

addressSize
Answer the size of an IPv4 address.

version Answer the version of IP that the receiver implements.

6.12.3 Sockets.IP6Address class: initialization

createLoopbackHost
Answer an object representing the loopback host in the address family for the receiver. This is ::1 for IPv4.

createUnknownAddress
Answer an object representing an unkown address in the address family for the receiver

initialize Set up the default implementation classes for the receiver

6.12.4 Sockets.IP6Address class: instance creation

fromArray: parts
Answer a new IP6Address from an array of numbers; the numbers are to be thought as the colon-separated numbers in the standard numbers-and-colons notation for IPv4 addresses.

fromBytes: aByteArray
Answer a new IP6Address from a ByteArray containing the bytes in the same order as the digit form: 131.175.6.2 would be represented as #[131 175 6 2].

fromSockAddr: aByteArray port: portAdaptor
Private - Answer a new IP6Address from a ByteArray containing a C sockaddr_in structure. The portAdaptor's value is changed to contain the port that the structure refers to.

fromString: aString
Answer a new IP6Address from a String containing the requested address in digit form.

new This method should not be called for instances of this class.

6.12.5 Sockets.IP6Address: accessing

asByteArray
Answer a read-only ByteArray of size four containing the receiver's bytes in network order (big-endian)

isMulticast
Answer whether the receiver reprensents an address reserved for multicast datagram connections

6.12.6 Sockets.IP6Address: printing

printOn: aStream
Print the receiver in dot notation.

6.13 Sockets.IPAddress

Defined in namespace Sockets
Superclass: Sockets.SocketAddress
Category: Sockets-Protocols
This class models an IPv4 address. It also acts as a factory for IPv4 stream (TCP), datagram (UDP) and raw sockets.

6.13.1 Sockets.IPAddress class: C constants

addressFamily
Not commented.

protocolFamily
Not commented.

6.13.2 Sockets.IPAddress class: constants

addressSize
Answer the size of an IPv4 address.

version Answer the version of IP that the receiver implements.

6.13.3 Sockets.IPAddress class: initialization

createLoopbackHost
Answer an object representing the loopback host in the address family for the receiver. This is 127.0.0.1 for IPv4.

createUnknownAddress
Answer an object representing an unkown address in the address family for the receiver

initialize Set up the default implementation classes for the receiver

6.13.4 Sockets.IPAddress class: instance creation

fromArray: parts
Answer a new IPAddress from an array of numbers; the numbers are to be thought as the dot-separated numbers in the standard numbers-and-dots notation for IPv4 addresses.

fromBytes: aByteArray
Answer a new IPAddress from a ByteArray containing the bytes in the same order as the digit form: 131.175.6.2 would be represented as #[131 175 6 2].

fromSockAddr: aByteArray port: portAdaptor
Private - Answer a new IPAddress from a ByteArray containing a C sockaddr_in structure. The portAdaptor's value is changed to contain the port that the structure refers to.

fromString: aString
Answer a new IPAddress from a String containing the requested address in digit form. Hexadecimal forms are not allowed.

An Internet host address is a number containing four bytes of data. These are divided into two parts, a network number and a local network address number within that network. The network number consists of the first one, two or three bytes; the rest of the bytes are the local address.

Network numbers are registered with the Network Information Center (NIC), and are divided into three classes–A, B, and C. The local network address numbers of individual machines are registered with the administrator of the particular network.

Class A networks have single-byte numbers in the range 0 to 127. There are only a small number of Class A networks, but they can each support a very large number of hosts (several millions). Medium-sized Class B networks have two-byte network numbers, with the first byte in the range 128 to 191; they support several thousands of host, but are almost exhausted. Class C networks are the smallest and the most commonly available; they have three-byte network numbers, with the first byte in the range 192-223. Class D (multicast, 224.0.0.0 to 239.255.255.255) and E (research, 240.0.0.0 to 255.255.255.255) also have three-byte network numbers.

Thus, the first 1, 2, or 3 bytes of an Internet address specifies a network. The remaining bytes of the Internet address specify the address within that network. The Class A network 0 is reserved for broadcast to all networks. In addition, the host number 0 within each network is reserved for broadcast to all hosts in that network. The Class A network 127 is reserved for loopback; you can always use the Internet address '127.0.0.1' to refer to the host machine (this is answered by the #loopbackHost class method).

Since a single machine can be a member of multiple networks, it can have multiple Internet host addresses. However, there is never supposed to be more than one machine with the same host address.

There are four forms of the standard numbers-and-dots notation for Internet addresses: a.b.c.d specifies all four bytes of the address individually; a.b.c interprets as a 2-byte quantity, which is useful for specifying host addresses in a Class B network with network address number a.b; a.b intrprets the last part of the address as a 3-byte quantity, which is useful for specifying host addresses in a Class A network with network address number a.

If only one part is given, this corresponds directly to the host address number.

new This method should not be called for instances of this class.

with: b1 with: b2 with: b3 with: b4
Answer a new IPAddress whose bytes (from most-significant to least-significant) are in the parameters.

6.13.5 Sockets.IPAddress: accessing

addressClass
Answer the 'address class' of the receiver (see IPAddress class>>#fromString:)

asByteArray
Answer a read-only ByteArray of size four containing the receiver's bytes in network order (big-endian)

host Answer an host number for the receiver; this is given by the last three bytes for class A addresses, by the last two bytes for class B addresses, else by the last byte.

isMulticast
Answer whether the receiver reprensents an address reserved for multicast datagram connections

network Answer a network number for the receiver; this is given by the first three bytes for class C/D/E addresses, by the first two bytes for class B addresses, else by the first byte.

subnet Answer an host number for the receiver; this is 0 for class A addresses, while it is given by the last byte of the network number for class B/C/D/E addresses.

6.13.6 Sockets.IPAddress: printing

printOn: aStream
Print the receiver in dot notation.

6.14 Sockets.MulticastSocket

Defined in namespace Sockets
Superclass: Sockets.DatagramSocket
Category: Sockets-Streams
This class models a multicast socket that sends packets to a multicast group. All members of the group listening on that address and port will receive all the messages sent to the group.

In the TCP/IP world, these sockets are UDP-based and a multicast group consists of a multicast address (a class D internet address, i.e. one whose most significant bits are 1110), and a well known port number.

6.14.1 Sockets.MulticastSocket: instance creation

interface
Answer the local device supporting the multicast socket. This is usually set to any local address.

interface: ipAddress
Set the local device supporting the multicast socket. This is usually set to any local address.

join: ipAddress
Join the multicast socket at the given IP address

leave: ipAddress
Leave the multicast socket at the given IP address

nextPut: packet timeToLive: timeToLive
Send the datagram with a specific TTL (time-to-live)

timeToLive
Answer the socket's datagrams' default time-to-live

timeToLive: newTTL
Set the default time-to-live for the socket's datagrams

6.15 Sockets.MulticastSocketImpl

Defined in namespace Sockets
Superclass: Sockets.DatagramSocketImpl
Category: Sockets-Protocols
This abstract class serves as the parent class for datagram socket implementations that support multicast.

6.15.1 Sockets.MulticastSocketImpl: multicasting

ipMulticastIf
Answer the local device for a multicast socket (in the form of an address)

ipMulticastIf: interface
Set the local device for a multicast socket (in the form of an address, usually anyLocalAddress)

join: ipAddress
Join the multicast socket at the given address

leave: ipAddress
Leave the multicast socket at the given address

timeToLive
Answer the time to live of the datagrams sent through the receiver to a multicast socket.

timeToLive: ttl
Set the time to live of the datagrams sent through the receiver to a multicast socket.

6.16 Sockets.OOBSocketImpl

Defined in namespace Sockets
Superclass: Sockets.DatagramSocketImpl
Category: Sockets-Protocols
This abstract class serves as the parent class for socket implementations that send out-of-band data over a stream socket.

6.16.1 Sockets.OOBSocketImpl: C constants

msgOOB Not commented.

6.16.2 Sockets.OOBSocketImpl: implementation

canRead Answer whether out-of-band data is available on the socket

ensureReadable
Stop the process until an error occurs or out-of-band data becomes available on the socket

6.17 Sockets.RawSocketImpl

Defined in namespace Sockets
Superclass: Sockets.DatagramSocketImpl
Category: Sockets-Protocols
This abstract class serves as the parent class for raw socket implementations. Raw socket packets are modeled as datagrams.

6.17.1 Sockets.RawSocketImpl class: parameters

socketType
Answer the socket type parameter for ‘create’.

6.18 Sockets.ReadBuffer

Defined in namespace Sockets
Superclass: ReadStream
Category: Examples-Useful tools

I'm a ReadStream that, when the end of the stream is reached, evaluates an user defined block to try to get some more data.

6.18.1 Sockets.ReadBuffer class: instance creation

on: aCollection
Answer a Stream that uses aCollection as a buffer. You should ensure that the fillBlock is set before the first operation, because the buffer will report that the data has ended until you set the fillBlock.

6.18.2 Sockets.ReadBuffer: accessing-reading

nextAvailable: anInteger into: aCollection startingAt: pos
Place the next anInteger objects from the receiver into aCollection, starting at position pos. Return the number of items stored.

nextAvailable: anInteger putAllOn: aStream
Copy the next anInteger objects from the receiver to aStream. Return the number of items stored.

upTo: anObject
Returns a collection of the same type that the stream accesses, up to but not including the object anObject. Returns the entire rest of the stream's contents if anObject is not present.

upToEnd Returns a collection of the same type that the stream accesses, up to but not including the object anObject. Returns the entire rest of the stream's contents if anObject is not present.

6.18.3 Sockets.ReadBuffer: buffer handling

atEnd Answer whether the data stream has ended.

availableBytes
Answer how many bytes are available in the buffer.

bufferContents
Answer the data that is in the buffer, and empty it.

close Not commented.

fill Fill the buffer with more data if it is empty, and answer true if the fill block was able to read more data.

fillBlock: block
Set the block that fills the buffer. It receives a collection and the number of bytes to fill in it, and must return the number of bytes actually read

isEmpty Answer whether the next input operation will force a buffer fill

isFull Answer whether the buffer has been just filled

notEmpty Check whether the next input operation will force a buffer fill and answer true if it will not.

pastEnd Try to fill the buffer if the data stream has ended.

6.19 Sockets.ServerSocket

Defined in namespace Sockets
Superclass: Sockets.AbstractSocket
Category: Sockets-Streams

This class models server side sockets. The basic model is that the server socket is created and bound to some well known port. It then listens for and accepts connections. At that point the client and server sockets are ready to communicate with one another utilizing whatever application layer protocol they desire.

As with the other AbstractSocket subclasses, most instance methods of this class simply redirect their calls to an implementation class.

6.19.1 Sockets.ServerSocket class: accessing

defaultImplementationClassFor: aSocketAddressClass
Answer the default implementation class.

6.19.2 Sockets.ServerSocket class: instance creation

defaultQueueSize
Answer the default length of the queue for pending connections. When the queue fills, new clients attempting to connect fail until the server has sent #accept to accept a connection from the queue.

port: anInteger
Answer a new ServerSocket serving on any local address, on the given port, with a pending connections queue of the default length.

port: anInteger bindTo: ipAddress
Answer a new ServerSocket serving on the given address and port, with a pending connections queue of the default length.

port: anInteger queueSize: backlog
Answer a new ServerSocket serving on any local address, on the given port, with a pending connections queue of the given length.

port: anInteger queueSize: backlog bindTo: ipAddress
Answer a new ServerSocket serving on the given address and port, and with a pending connections queue of the given length.

queueSize: backlog
Answer a new ServerSocket serving on any local address and port, with a pending connections queue of the given length.

queueSize: backlog bindTo: ipAddress
Answer a new ServerSocket serving on the given local address, and on any port, with a pending connections queue of the given length.

6.19.3 Sockets.ServerSocket: accessing

accept Accept a new connection and create a new instance of Socket if there is one, else answer nil.

accept: socketClass
Accept a new connection and create a new instance of socketClass if there is one, else answer nil. This is usually needed only to create DatagramSockets.

address Answer the local address

port Answer the local port (the port that the passive socket is listening on).

primAccept: socketClass
Accept a new connection and create a new instance of Socket if there is one, else fail.

waitForConnection
Wait for a connection to be available, and suspend the currently executing process in the meanwhile.

6.19.4 Sockets.ServerSocket: initializing

port: anInteger queueSize: backlog bindTo: localAddr
Initialize the ServerSocket so that it serves on the given address and port, and has a pending connections queue of the given length.

6.20 Sockets.Socket

Defined in namespace Sockets
Superclass: Sockets.StreamSocket
Category: Sockets-Streams
This class adds read and write buffers to the basic model of AbstractSocket.

6.20.1 Sockets.Socket class: accessing

writeBufferSize
Answer the size of the write buffer for newly-created sockets

writeBufferSize: anInteger
Set the size of the write buffer for newly-created sockets

6.20.2 Sockets.Socket class: tests

datagramLoopbackTest
Send data from one datagram socket to another on the local machine. Tests most of the socket primitives and works with different processes.

datagramLoopbackTestOn: addressClass
Send data from one datagram socket to another on the local machine. Tests most of the socket primitives and works with different processes.

loopbackTest
Send data from one socket to another on the local machine. Tests most of the socket primitives.

loopbackTest: bufferSizes
Send data from one socket to another on the local machine. Tests most of the socket primitives. The parameter is the size of the input and output buffer sizes.

loopbackTest: bufferSizes addressClass: addressClass
Send data from one socket to another on the local machine. Tests most of the socket primitives. The parameters are the size of the input and output buffer sizes, and the address class (family) to use.

loopbackTestOn: addressClass
Send data from one socket to another on the local machine. Tests most of the socket primitives. The parameter is the address class (family) to use.

microTest Extremely small test (try to receive SMTP header)

producerConsumerTest
Send data from one datagram socket to another on the local machine. Tests most of the socket primitives and works with different processes.

producerConsumerTestOn: addressClass
Send data from one socket to another on the local machine. Tests most of the socket primitives and works with different processes.

sendTest Send data to the 'discard' socket of localhost.

sendTest: host
Send data to the 'discard' socket of the given host. Tests the speed of one-way data transfers across the network to the given host. Note that many hosts do not run a discard server.

testPort2For: anAddressClass
Not commented.

testPortFor: anAddressClass
Not commented.

tweakedLoopbackTest
Send data from one socket to another on the local machine, trying to avoid buffering overhead. Tests most of the socket primitives. Comparison of the results of loopbackTest and tweakedLoopbackTest should give a measure of the overhead of buffering when sending/receiving large quantities of data.

6.20.3 Sockets.Socket class: well known ports

initialize Initialize the receiver's defaults

6.20.4 Sockets.Socket: stream protocol

canWrite Answer whether more data is available in the socket's read buffer or from the operating system.

ensureWriteable
Answer whether more data is available in the socket's read buffer or from the operating system.

flush Flush the write buffer to the operating system

next: n putAll: aCollection startingAt: pos
Write aString to the socket; this acts as a bit-bucket when the socket is closed. This might yield control to other Smalltalk Processes.

nextPut: char
Write a character to the socket; this acts as a bit-bucket when the socket is closed. This might yield control to other Smalltalk Processes.

writeBufferSize: size
Create a new write buffer of the given size, flushing the old one is needed. This might yield control to other Smalltalk Processes.

6.21 Sockets.SocketAddress

Defined in namespace Sockets
Superclass: Object
Category: Sockets-Protocols
This class is the abstract class for machine addresses over a network. It also fulfills the function of the C style functions gethostname(), gethostbyname(), and gethostbyaddr(), resolves machine names into their corresponding numeric addresses (via DNS, /etc/hosts, or other mechanisms) and vice versa.

6.21.1 Sockets.SocketAddress class: abstract

extractFromSockAddr: aByteArray port: portAdaptor
Private - Answer a new SocketAddress from a ByteArray containing a C sockaddr structure. The portAdaptor's value is changed to contain the port that the structure refers to.

fromSockAddr: aByteArray port: portAdaptor
Private - Answer a new IPAddress from a ByteArray containing a C sockaddr structure. The portAdaptor's value is changed to contain the port that the structure refers to. Raise an error if the address family is unknown.

6.21.2 Sockets.SocketAddress class: accessing

anyLocalAddress
Answer an IPAddress representing a local address.

at: host cache: aBlock
Private - Answer the list of addresses associated to the given host in the cache. If the host is not cached yet, evaluate aBlock and cache and answer the result.

defaultDatagramSocketImplClass
Answer the class that, by default, is used to map between the Socket's protocol and a low-level C interface.

defaultDatagramSocketImplClass: aClass
Set which class will be used by default to map between the receiver's protocol and a low-level C interface.

defaultRawSocketImplClass
Answer the class that, by default, is used to map between the Socket's protocol and a low-level C interface.

defaultRawSocketImplClass: aClass
Set which class will be used by default to map between the receiver's protocol and a low-level C interface.

defaultStreamSocketImplClass
Answer the class that, by default, is used to map between the Socket's protocol and a low-level C interface.

defaultStreamSocketImplClass: aClass
Set which class will be used by default to map between the receiver's protocol and a low-level C interface.

isDigitAddress: aString
Answer whether the receiver can interpret aString as a valid address without going through a resolver.

localHostName
Answer the name of the local machine.

loopbackHost
Answer an instance of the receiver representing the local machine (127.0.0.1 in the IPv4 family).

unknownAddress
Answer an instance of the receiver representing an unknown machine (0.0.0.0 in the IPv4 family).

6.21.3 Sockets.SocketAddress class: C call-outs

primLocalName
Not commented.

primName: address len: len type: addressFamily
Not commented.

6.21.4 Sockets.SocketAddress class: C constants

addressFamily
Not commented.

aiAddrconfig
Not commented.

aiCanonname
Not commented.

protocolFamily
Not commented.

6.21.5 Sockets.SocketAddress class: creating sockets

newRawSocket
Create a new raw socket, providing access to low-level network protocols and interfaces for the protocol family represented by the receiver (for example, the C protocol family PF_INET for the IPAddress class) Ordinary user programs usually have no need to use this method.

6.21.6 Sockets.SocketAddress class: host name lookup

allByName: aString
Answer all the IP addresses that refer to the the given host. If a digit address is passed in aString, the result is an array containing the single passed address. If the host could not be resolved to an IP address, answer nil.

byName: aString
Answer a single IP address that refer to the the given host. If a digit address is passed in aString, the result is the same as using #fromString:. If the host could not be resolved to an IP address, answer nil.

6.21.7 Sockets.SocketAddress class: initialization

anyLocalAddress: anObject
Private - Store an object representing a local address in the address family for the receiver

createLoopbackHost
Answer an object representing the loopback host in the address family for the receiver.

createUnknownAddress
Answer an object representing an unkown address in the address family for the receiver

flush Flush the cached IP addresses.

initLocalAddresses
Private - Initialize the anyLocalAddress class-instance variable for the entire hierarchy.

update: aspect
Flush all the caches for IPAddress subclasses

6.21.8 Sockets.SocketAddress: accessing

= aSocketAddress
Answer whether the receiver and aSocketAddress represent the same machine. The host name is not checked because an IPAddress created before a DNS is activated is named after its numbers-and-dots notation, while the same IPAddress, created when a DNS is active, is named after its resolved name.

asByteArray
Convert the receiver to a ByteArray passed to the operating system's socket functions)

hash Answer an hash value for the receiver

name Answer the host name (or the digit notation if the DNS could not resolve the address). If the DNS answers a different IP address for the same name, the second response is not cached and the digit notation is also returned (somebody's likely playing strange jokes with your DNS).

6.21.9 Sockets.SocketAddress: testing

isMulticast
Answer whether an address is reserved for multicast connections.

6.22 Sockets.SocketImpl

Defined in namespace Sockets
Superclass: Sockets.AbstractSocketImpl
Category: Sockets-Protocols
This abstract class serves as the parent class for stream socket implementations.

6.22.1 Sockets.SocketImpl class: parameters

socketType
Answer the socket type parameter for 'create'.

6.22.2 Sockets.SocketImpl: abstract

outOfBandImplClass
Return an implementation class to be used for out-of-band data on the receiver.

6.22.3 Sockets.SocketImpl: socket operations

connectTo: ipAddress port: port
Try to connect the socket represented by the receiver to the given remote machine.

getPeerName
Retrieve a ByteArray containing a sockaddr_in struct for the remote endpoint of the socket.

6.23 Sockets.StreamSocket

Defined in namespace Sockets
Superclass: Sockets.AbstractSocket
Category: Sockets-Streams
This class adds a read buffer to the basic model of AbstractSocket.

6.23.1 Sockets.StreamSocket class: accessing

defaultImplementationClassFor: aSocketAddressClass
Answer the default implementation class. Depending on the subclass, this might be the default stream socket implementation class of the given address class, or rather its default datagram socket implementation class.

readBufferSize
Answer the size of the read buffer for newly-created sockets

readBufferSize: anInteger
Set the size of the read buffer for newly-created sockets

6.23.2 Sockets.StreamSocket class: initialize

initialize Initialize the receiver's defaults

6.23.3 Sockets.StreamSocket class: instance creation

remote: ipAddressOrString port: remotePort
Create a new socket and connect to the given host (passed as a String to be resolved or as a SocketAddress), and to the given port.

remote: ipAddressOrString port: remotePort local: ipAddress port: localPort
Create a new socket and connect to the given host (passed as a String to be resolved or as a SocketAddress), and to the given remotePort. Then bind it to the local address passed in ipAddress, on the localPort port; if the former is nil, any local address will do, and if the latter is 0, any local port will do.

6.23.4 Sockets.StreamSocket: accessing

address Answer the address of the remote endpoint

port Answer the port of the remote endpoint

6.23.5 Sockets.StreamSocket: accessing-reading

nextAvailable: anInteger into: aCollection startingAt: pos
Place up to anInteger objects from the receiver into aCollection, starting from position pos and stopping if no more data is available.

nextAvailable: anInteger putAllOn: aStream
Copy up to anInteger objects from the receiver to aStream, stopping if no more data is available.

6.23.6 Sockets.StreamSocket: out-of-band data

outOfBand
Return a datagram socket to be used for receiving out-of-band data on the receiver.

6.23.7 Sockets.StreamSocket: printing

printOn: aStream
Print a representation of the receiver on aStream

6.23.8 Sockets.StreamSocket: stream protocol

atEnd Answer whether more data is available on the socket

availableBytes
Answer how many bytes are available in the socket's read buffer or from the operating system.

bufferContents
Answer the current contents of the read buffer

canRead Answer whether more data is available in the socket's read buffer or from the operating system.

close Flush and close the socket.

fill Fill the read buffer with data read from the socket

isPeerAlive
Answer whether the connection with the peer remote machine is still valid.

next Read a byte from the socket. This might yield control to other Smalltalk Processes.

peek Read a byte from the socket, without advancing the buffer; answer nil if no more data is available. This might yield control to other Smalltalk Processes.

peekFor: anObject
Read a byte from the socket, advancing the buffer only if it matches anObject; answer whether they did match or not. This might yield control to other Smalltalk Processes.

readBufferSize: size
Create a new read buffer of the given size (which is only possible before the first read or if the current buffer is empty).

6.24 Sockets.TCPSocketImpl

Defined in namespace Sockets
Superclass: Sockets.SocketImpl
Category: Sockets-Protocols
Unless the application installs its own implementation, this is the default socket implementation that will be used for IPv4 stream sockets. It uses C callouts to implement standard BSD style sockets of family AF_INET and type SOCK_STREAM.

6.24.1 Sockets.TCPSocketImpl class: C constants

ipprotoTcp
Not commented.

protocol Not commented.

tcpNodelay
Not commented.

6.24.2 Sockets.TCPSocketImpl: socket options

valueWithoutBuffering: aBlock
Evaluate aBlock, ensuring that any data that it writes to the socket is sent immediately to the network.

6.25 Sockets.UDPSocketImpl

Defined in namespace Sockets
Superclass: Sockets.MulticastSocketImpl
Category: Sockets-Protocols
Unless the application installs its own implementation, this is the default socket implementation that will be used for IPv4 datagram sockets. It uses C callouts to implement standard BSD style sockets of family AF_INET and type SOCK_DGRAM.

6.25.1 Sockets.UDPSocketImpl class: C constants

ipprotoIp Not commented.

protocol Not commented.

6.25.2 Sockets.UDPSocketImpl: multicasting

ipMulticastIf
Answer the local device for a multicast socket (in the form of an address)

ipMulticastIf: interface
Set the local device for a multicast socket (in the form of an address, usually anyLocalAddress)

join: ipAddress
Join the multicast socket at the given address

leave: ipAddress
Leave the multicast socket at the given address

primJoinLeave: ipAddress option: opt
Private - Used to join or leave a multicast service.

timeToLive
Answer the time to live of the datagrams sent through the receiver to a multicast socket.

timeToLive: ttl
Set the time to live of the datagrams sent through the receiver to a multicast socket.

6.26 Sockets.UnixAddress

Defined in namespace Sockets
Superclass: Sockets.SocketAddress
Category: Sockets-Protocols

This class represents an address for a machine using the AF_UNIX address family. Since this address family is only used for local sockets, the class is a singleton; the filesystem path to the socket is represented using the port argument to socket functions, as either a String or a File object.

6.26.1 Sockets.UnixAddress class: C constants

addressFamily
Not commented.

protocolFamily
Not commented.

6.26.2 Sockets.UnixAddress class: initialization

createLoopbackHost
Answer an object representing the loopback host in the address family for the receiver. This is 127.0.0.1 for IPv4.

createUnknownAddress
Answer an object representing an unkown address in the address family for the receiver

initialize Set up the default implementation classes for the receiver

6.26.3 Sockets.UnixAddress class: instance creation

fromSockAddr: aByteArray port: portAdaptor
Private - Answer the unique UnixAddress instance, filling in the portAdaptor's value from a ByteArray containing a C sockaddr_in structure.

uniqueInstance
Not commented.

6.26.4 Sockets.UnixAddress: accessing

= aSocketAddress
Answer whether the receiver and aSocketAddress represent the same socket on the same machine.

hash Answer an hash value for the receiver

6.26.5 Sockets.UnixAddress: printing

printOn: aStream
Print the receiver in dot notation.

6.26.6 Sockets.UnixAddress: testing

isMulticast
Answer whether an address is reserved for multicast connections.

6.27 Sockets.UnixDatagramSocketImpl

Defined in namespace Sockets
Superclass: Sockets.DatagramSocketImpl
Category: Sockets-Protocols
This class represents a datagram socket using the AF_UNIX address family. It unlinks the filesystem path when the socket is closed.

6.27.1 Sockets.UnixDatagramSocketImpl: socket operations

close Not commented.

6.28 Sockets.UnixSocketImpl

Defined in namespace Sockets
Superclass: Sockets.SocketImpl
Category: Sockets-Protocols
This class represents a stream socket using the AF_UNIX address family. It unlinks the filesystem path when the socket is closed.

6.28.1 Sockets.UnixSocketImpl: socket operations

close Not commented.

6.29 Sockets.WriteBuffer

Defined in namespace Sockets
Superclass: WriteStream
Category: Examples-Useful tools
I'm a WriteStream that, instead of growing the collection, evaluates an user defined block and starts over with the same collection.

6.29.1 Sockets.WriteBuffer: accessing-writing

next: n putAll: aCollection startingAt: pos
Put n characters or bytes of aCollection, starting at the pos-th, in the collection buffer.

6.29.2 Sockets.WriteBuffer: buffer handling

close Not commented.

flush Evaluate the flushing block and reset the stream

flushBlock: block
Set which block will be used to flush the buffer. The block will be evaluated with a collection and an Integer n as parameters, and will have to write the first n elements of the collection.

6.29.3 Sockets.WriteBuffer: testing

isFull Not commented.

7 Compressing and decompressing data with ZLib

7.1 Tree

Classes documented in this manual are **boldfaced**.

```
Object
  Exception
    Error
      ZLib.ZlibError
  Iterable
    Stream
      ZLib.ZlibStream
        ZLib.ZlibReadStream
          ZLib.RawDeflateStream
            ZLib.DeflateStream
            ZLib.GZipDeflateStream
          ZLib.RawInflateStream
            ZLib.GZipInflateStream
            ZLib.InflateStream
        ZLib.ZlibWriteStream
          ZLib.RawDeflateWriteStream
            ZLib.DeflateWriteStream
            ZLib.GZipDeflateWriteStream
```

7.2 ZLib.DeflateStream

Defined in namespace ZLib
Superclass: ZLib.RawDeflateStream
Category: Examples-Useful

Instances of this class produce "standard" (zlib, RFC1950) deflated data.

7.2.1 ZLib.DeflateStream class: instance creation

compressingTo: aStream

Answer a stream that receives data via #nextPut: and compresses it onto aStream.

compressingTo: aStream level: level

Answer a stream that receives data via #nextPut: and compresses it onto aStream with the given compression level.

7.3 ZLib.DeflateWriteStream

Defined in namespace ZLib
Superclass: ZLib.RawDeflateWriteStream
Category: Examples-Useful

Instances of this class produce "standard" (zlib, RFC1950) deflated data.

7.4 ZLib.GZipDeflateStream

Defined in namespace ZLib
Superclass: ZLib.RawDeflateStream
Category: Examples-Useful

Instances of this class produce GZip (RFC1952) deflated data.

7.4.1 ZLib.GZipDeflateStream class: instance creation

compressingTo: aStream

Answer a stream that receives data via #nextPut: and compresses it onto aStream.

compressingTo: aStream level: level

Answer a stream that receives data via #nextPut: and compresses it onto aStream with the given compression level.

7.5 ZLib.GZipDeflateWriteStream

Defined in namespace ZLib
Superclass: ZLib.RawDeflateWriteStream
Category: Examples-Useful

Instances of this class produce GZip (RFC1952) deflated data.

7.6 ZLib.GZipInflateStream

Defined in namespace ZLib
Superclass: ZLib.RawInflateStream
Category: Examples-Useful

Instances of this class reinflate GZip (RFC1952) deflated data.

7.7 ZLib.InflateStream

Defined in namespace ZLib
Superclass: ZLib.RawInflateStream
Category: Examples-Useful

Instances of this class reinflate "standard" (zlib, RFC1950) deflated data.

7.8 ZLib.RawDeflateStream

Defined in namespace ZLib
Superclass: ZLib.ZlibReadStream
Category: Examples-Useful

Instances of this class produce "raw" (PKZIP) deflated data.

7.8.1 ZLib.RawDeflateStream class: instance creation

compressingTo: aStream

Answer a stream that receives data via #nextPut: and compresses it onto aStream.

compressingTo: aStream level: level

Answer a stream that receives data via #nextPut: and compresses it onto aStream with the given compression level.

on: aStream

Answer a stream that compresses the data in aStream with the default compression level.

on: aStream level: compressionLevel

Answer a stream that compresses the data in aStream with the given compression level.

7.9 ZLib.RawDeflateWriteStream

Defined in namespace ZLib
Superclass: ZLib.ZlibWriteStream
Category: Examples-Useful

Instances of this class produce "raw" (PKZIP) deflated data.

7.9.1 ZLib.RawDeflateWriteStream class: instance creation

on: aWriteStream

Answer a stream that compresses the data in aStream with the default compression level.

on: aWriteStream level: compressionLevel

Answer a stream that compresses the data in aStream with the given compression level.

7.10 ZLib.RawInflateStream

Defined in namespace ZLib
Superclass: ZLib.ZlibReadStream
Category: Examples-Useful

Instances of this class reinflate "raw" (PKZIP) deflated data.

7.10.1 ZLib.RawInflateStream: positioning

copyFrom: start to: end
: Answer the data on which the receiver is streaming, from the start-th item to the end-th. Note that this method is 0-based, unlike the one in Collection, because a Stream's #position method returns 0-based values. Notice that this class can only provide the illusion of random access, by appropriately rewinding the input stream or skipping compressed data.

isPositionable
: Answer true if the stream supports moving backwards with #skip:.

position: anInteger
: Set the current position in the stream to anInteger. Notice that this class can only provide the illusion of random access, by appropriately rewinding the input stream or skipping compressed data.

reset Reset the stream to the beginning of the compressed data.

skip: anInteger
: Move the current position by anInteger places, either forwards or backwards.

7.11 ZLib.ZlibError

Defined in namespace ZLib
Superclass: Error
Category: Examples-Useful
: This exception is raised whenever there is an error in a compressed stream.

7.11.1 ZLib.ZlibError: accessing

stream Answer the ZlibStream that caused the error.

stream: anObject
: Set the ZlibStream that caused the error.

7.12 ZLib.ZlibReadStream

Defined in namespace ZLib
Superclass: ZLib.ZlibStream
Category: Examples-Useful
: This abstract class implements the basic buffering that is used for communication with zlib.

7.12.1 ZLib.ZlibReadStream: accessing-reading

nextAvailable: anInteger into: aCollection startingAt: pos
: Place up to anInteger objects from the receiver into aCollection, starting from position pos and stopping if no more data is available.

nextAvailable: anInteger putAllOn: aStream
: Copy up to anInteger objects from the receiver to aStream, stopping if no more data is available.

7.12.2 ZLib.ZlibReadStream: streaming

atEnd Answer whether the stream has got to an end

next Return the next object (character or byte) in the receiver.

peek Returns the next element of the stream without moving the pointer. Returns nil when at end of stream.

peekFor: anObject
Returns true and gobbles the next element from the stream of it is equal to anObject, returns false and doesn't gobble the next element if the next element is not equal to anObject.

position Answer the current value of the stream pointer. Note that only inflating streams support random access to the stream data.

7.13 ZLib.ZlibStream

Defined in namespace ZLib
Superclass: Stream
Category: Examples-Useful
This abstract class implements the basic interface to the zlib module. Its layout matches what is expected by the C code.

7.13.1 ZLib.ZlibStream class: accessing

bufferSize Answer the size of the output buffers that are passed to zlib. Each zlib stream uses a buffer of this size.

bufferSize: anInteger
Set the size of the output buffers that are passed to zlib. Each zlib stream uses a buffer of this size.

defaultCompressionLevel
Return the default compression level used by deflating streams.

defaultCompressionLevel: anInteger
Set the default compression level used by deflating streams. It should be a number between 1 and 9.

7.13.2 ZLib.ZlibStream class: instance creation

new This method should not be called for instances of this class.

on: aStream
Answer an instance of the receiver that decorates aStream.

7.13.3 ZLib.ZlibStream: streaming

isExternalStream
Answer whether the receiver streams on a file or socket.

name Return the name of the underlying stream.

species Return the type of the collections returned by #upTo: etc.

stream Answer the wrapped stream.

7.14 ZLib.ZlibWriteStream

Defined in namespace ZLib
Superclass: ZLib.ZlibStream
Category: Examples-Useful

This abstract class implements the basic buffering that is used for communication with zlib in a WriteStream decorator.

7.14.1 ZLib.ZlibWriteStream: streaming

close Finish the deflated output to the destination stream using Z_FINISH. The destination stream is closed, which implies flushing.

contents Finish the deflated output to the destination stream using Z_FINISH and return the deflated data (requires the destination stream to support #contents).

finish Finish the deflated output to the destination stream using Z_FINISH. The destination stream is not flushed.

flush Flush the deflated output to the destination stream, and flush the destination stream.

flushBuffer
Flush the deflated output to the destination stream.

flushDictionary
Flush the deflated output to the destination stream using Z_FULL_FLUSH, and flush the destination stream.

next: n putAll: aCollection startingAt: pos
Put n characters or bytes of aCollection, starting at the pos-th, in the deflation buffer.

nextPut: aByte
Append a character or byte (depending on whether the destination stream works on a ByteArray or String) to the deflation buffer.

partialFlush
Flush the deflated output to the destination stream using Z_PARTIAL_FLUSH, and flush the destination stream.

position Answer the number of compressed bytes written.

readStream
Finish the deflated output to the destination stream using Z_FINISH and return a ReadStream on the deflated data (requires the destination stream to support #readStream).

syncFlush Flush the deflated output to the destination stream using Z_SYNC_FLUSH, and flush the destination stream. Note that this includes the four bytes 0/0/255/255 at the end of the flush.

8 Libraries for the SAX, DOM, XPath and XSLT standards

by Thomas Gagne, edited by Paolo Bonzini

8.1 Building a DOM from XML

If you're like me, the first thing you may be trying to do is build a Document Object Model (DOM) tree from some kind of XML input. Assuming you've got the XML in a String the following code will build an XML Document:

```
XML.SAXParser defaultParserClass processDocumentString: theXMLString
beforeScanDo: [ :p | p validate: false].
```

Though the code above appears as though it should be easy to use, there's some hidden features you should know about. First, `theXMLString` can not contain any null bytes. Depending on where your XML comes from it may have a NULL byte at the end (like mine did). Many languages implement strings as an array of bytes (usually printable ones) ending with a null (a character with integer value 0). In my case, the XML was coming from a remote client written in C using middleware to send the message to my server. Since the middleware doesn't assume to know anything about the message it received, it's received into a String, null-byte and all. To remove it I used:

```
XML.SAXParser defaultParserClass
    processDocumentString: (aString copyWithout: 0 asCharacter)
    beforeScanDo: [ :p | p validate: false].
```

Starting out, I didn't know much about the value of DTDs either (Document Type Definitions), so I wasn't using them (more on why you should later). What you need to know is XML comes in two flavors, (three if you include broken as a flavor) *well-formed* and *valid*.

Well-formed XML is simply XML following the basic rules, like only one top-level (the document's root), no overlapping tags, and a few other contraints. Valid XML means not only is the XML well-formed, but it's also compliant with some kind of rule base about which elements are allowed to follow which other ones, whether or not attributes are permitted and what their values and defaults should be, etc.

There's no way to get around well-formedness. Most XML tools complain vociferously about missing or open tags. What you may not have lying around, though, is a DTD describing how the XML should be assembled. If you need to skip validation for any reason you must include the selector:

```
beforeScanDo: [ :p | p validate: false].
```

Now that you have your XML document, you probably want to access its contents (why else would you want one, right?). Let's take the following (brief) XML as an example:

```
<porder porder_num="10351">
  <porder_head>
    <order_date>01/04/2000</order_date>
  </porder_head>
  <porder_line>
    <part>widget</part>
```

```
        <quantity>1.0000</quantity>
      </porder_line>
      <porder_line>
        <part>doodad</part>
        <quantity>2.0000</quantity>
      </porder_line>
    </porder>
```

The first thing you probably want to know is how to access the different tags, and more specifically, how to access the contents of those tags. First, by way of providing a roadmap to the elements I'll show you the Smalltalk code for getting different pieces of the document, assuming the variable you've assigned the document to is named *doc*. I'll also create instance variables for the various elements as I go along:

Element you want	*Code to get it*
porder element	`doc root`
porder_head	`doc root elementNamed: 'porder_head'`
order_date (as a String)	`(porderHead elementNamed: 'order_date') characterData`
order_date (as a Date)	`(Date readFrom: (porderHead elementNamed: 'order_date') characterData readStream)`
a collection with both porder_lines	`doc root elementsNamed: 'porder_line'`

I've deliberately left-out accessing `porder`'s attribute because accessing them is different from accessing other nodes. You can get an OrderedCollection of attributes using:

```
attributes := doc root attributes.
```

but the ordered collection isn't really useful. To access any single attribute you'd need to look for it in the collection:

```
porderNum := (attributes detect: [ :each | each key type = 'porder_num' ]) value.
```

But that's not a whole lot of fun, especially if there's a lot you need to get, and if there's any possibility the attribute may not exist. Then you have to do the whole `detect:ifNone:` thing, and boy, does that make the code readable! What I did instead was create a method in my objects' abstract:

```
dictionaryForAttributes: aCollection
    ^Dictionary withAll: (aCollection
collect: [ :each | each key type -> each value ])
```

Now what you have is an incrementally more useful method for getting attributes:

```
attributes := self dictionaryForAttributes: doc root attributes.
porderNum := attributes at: 'porder_num'.
```

At first this appears like more code, and for a single attribute it probably is. But if an element includes more than one attribute the payoff is fairly decent. Of course, you still need to handle the absence of an attribute in the dictionary but I think it reads a little better using a Dictionary than an OrderedCollection:

```
porderNum := attributes at: 'porder_num' ifAbsent: [].
```

8.2 Building XML

There's little reason to build an XML document if its not going to be processed by something down the road. Most XML tools require XML documents have a document root. A root is a tag inside which all other tags exist, or put another way, a single parent node from which all other nodes descend. In my case, a co-worker was attempting to use Sablot's sabcmd to transform the XML from my server into HTML. So start your document with the root ready to go:

```
replyDoc := XML.Document new.
replyDoc addNode: (XML.Element tag: 'response').
```

Before doing anything more complex, we can play with our new XML document. Assuming you're going to want to send the XML text to someone or write it to a file, you may first want to capture it in a string. Even if you don't want to first capture it into a string our example is going to:

```
replyStream := String new writeStream.
replyDoc printOn: replyStream.
```

If we examine'd the contents of our replyStream (`replyStream contents`) we'd see:

```
<response/>
```

Which is what an empty tag looks like.

Let's add some text to our XML document now. Let's say we want it to look like:

```
<response>Hello, world!</response>
```

Building this actually requires two nodes be added to a new XML document. The first node (or element) is named `response`. The second node adds text to the first:

```
replyDoc := XML.Document new.
replyDoc addNode: (XML.Element tag: response). "our root node"
replyDoc root addNode: (XML.Text text: 'Hello, world!').
```

Another way of writing it, and the way I've adopted in my code is to create the whole node before adding it. This is not just to reduce the appearance of assignments, but it suggests a template for cascading `#addNode:` messages to an element, which, if you're building any kind of nontrivial XML, you'll be doing a lot of:

```
replyDoc := XML.Document new.
replyDoc addNode: (
    (XML.Element tag: response)
        addNode: (XML.Text text: 'Hello, world!')
).
```

Unless you're absolutely sure you'll never accidentally add text nodes that have an ampersand (&) in them, you'll need to escape it to get past XML parsers. The way I got around this was to escape them whenever I added text nodes. To make it easier, I (again) created a method in my objects' abstract superclass:

```
asXMLElement: tag value: aValue
```

```
        | n |

        n := XML.Element tag: tag.
        aValue isNil ifFalse: [
    n addNode: (XML.Text
        text: (aValue displayString copyReplaceAll: '&' with: '&amp;'))].
        ^n
```

Calls to `self asXMLElement: 'sometagname' value: anInstanceVariable` are littered throughout my code.

Adding attributes to documents is, thankfully, easier than accessing them. If we wanted to add an attribute to our document above we can do so with a single statement:

```
replyDoc root addAttribute: (XML.Attribute name: 'isExample' value: 'yes').
```

Now, our XML looks like:

```
<response isExample="yes">Hello, world!</response>
```

8.3 Using DTDs

What I didn't appreciate in my first XML project (this one) was how much error checking I was doing just to verify the format of incoming XML. During testing I'd go looking for attributes or elements that *should* have been there but for various reasons were not. Because I was coding fast and furious I overlooked some and ignored others. Testing quickly ferreted out my carelessnes and my application started throwing exceptions faster than election officials throw chads.

The cure, at least for formatting, is having a DTD, or Document Type Definition describing the XML format. You can read more about the syntax of DTDs in the XML specification.

There's not a lot programmers are able to do with DTDs in VisualWorks, except requiring incoming XML to include DOCTYPE statements. There is something programmers need to do to handle the exceptions the XML parser throws when it finds errors.

I'm not an expert at writing Smalltalk exception handling code, and I haven't decided on what those exceptions should look like to the client who sent the poorly formatted XML in the first place. The code below does a decent job of catching the errors and putting the description of the error into an XML response. It's also a fairly decent example of XML document building as discussed earlier.

```
replyDoc := XML.Document new.
replyDoc addNode: (XML.Element tag: 'response').

[
    doc := XML.SAXParser defaultParserClass processDocumentString: (anIsdMessage messa
] on: Exception do: [ :ex |
    replyDoc root
        addAttribute: (XML.Attribute name: 'type' value: 'Exception');
        addNode: ((XML.Element tag: 'description')
            addNode: (XML.Text text: ex signal description));
        addNode: ((XML.Element tag: 'message')
            addNode: (XML.Text text: ex messageText))
].
```

I said before there's not a lot programmers can do with DTDs, but there are some things I wish the XML library would do:

- I'd like to make sure the documents I build are built correctly. It would be great if a DTD could be attached to an empty XML document so that exceptions could be thrown as misplaced elements were added.
- It would be great to specify which DTD the XML parser should use when parsing incoming XML so that the incoming XML wouldn't always have to include a <!DOCTYPE> tag. Though it's fairly easy to add the tag at the start of XML text, it's really not that simple. You need to know the XML's root element before adding the <!DOCTYPE> tag but you really don't know that until after you've parsed the XML You would have to parse the XML, determine the root tag, then parse the output of the first into a new XML document with validation turned-on.
- Another reason to be able to create a DTD document to use with subsequent parsing is to avoid the overhead of parsing the same DTD over and over again. In transaction processing systems this kind of redundant task could be eliminated and the spare CPU cycles put to better use.

8.4 XSL Processing

I spent a night the other week trying to figure out how to get the XSL libraries to do anything. I no longer need it now, but I did discover some things others with an immediate need may want to be aware of.

- Transforming an XML document requires you parse the XSL and XML documents separately first. After that, you tell the XSL.RuleDatabase to process the XML document. The result is another XML document with the transformations.

  A code snippet for doing just that appears below.

  ```
  | rules xmlDoc htmlDoc |

  rules := XSL.RuleDatabase new readFileNamed: 'paymentspending.xsl'.
  xmlDoc := XML.SAXParser defaultParserClass
               processDocumentInFilename: 'paymentspending.xml'
               beforeScanDo: [ :p | p validate: false ].
  htmlDoc := rules process: xmlDoc.
  ```

  There is also a `readString:` method which can be used instead of `readFileNamed:`.
- The XSL library doesn't use the W3-approved stylesheet, but instead uses the draft version (same one Microsoft uses). `<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">`
- The functions `position()` and `count()` aren't implemented, or if they are, aren't implemented in the way other XSL tools implement it.

8.5 Attributions

Cincom, for supporting Smalltalk and the Smalltalk community by making the library available for GNU Smalltalk under the LGPL.

Thanks also to Randy Ynchausti, Bijan Parsia, Reinout Heeck, and Joseph Bacanskas for answering many questions on the XML library.

Class index

Method index

Selector cross-reference

Table of Contents