Widget Reference (A–C) 6

Chapters 6 to 10 describe the widgets in the OLIT widget set and functions that augment the specific widgets. (For functions that are not specific to a widget, see Chapter 5, "Toolkit Functions.")

AbbrevMenuButton Widget

Class

Class Name: `AbbrevMenuButton`
Class Pointer: `abbrevMenuButtonWidgetClass`

Ancestry

Core-Primitive-AbbrevMenuButton

Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/AbbrevMenu.h>
```

Description

The AbbrevMenuButton widget is used to create a popup menu that also provides current selection viewing to the user. When the user invokes the

6

AbbrevMenuButton Widget

MENU command on the AbbrevMenuButton, a menu pops up. Once the user makes a selection off the menu, the selected item should be displayed next to the AbbrevMenuButton. The AbbrevMenuButton also provides the features of the MenuButton widget (menu default selection, menu previewing, menu selection).

Components

The AbbrevMenuButton consists of a square button containing an arrow (menumark) with a popup menu attached. An application should create and identify an additional component, the Current Selection Widget, which is used to display previewing and also to display the current selection off the menu. Each AbbrevMenuButton also has the components of the MenuShell widget.

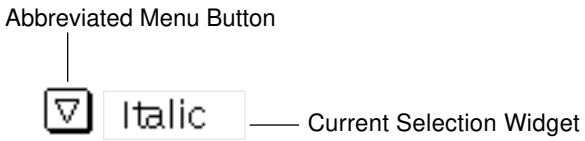

Figure 6-1 AbbrevMenuButton Widget

Subwidget

The AbbrevMenuButton widget automatically creates and attaches a MenuShell widget. An application can add menu items to this menu by obtaining the value of the `XtNmenuPane` resource and adding children to this widget.

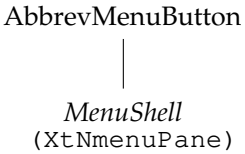

Figure 6-2 AbbrevMenuButton Subwidget

Current Selection Widget

The Current Selection Widget is created by the application. This widget should be a StaticText, TextField, or TextLine widget. Typically, the Current Selection Widget and the AbbrevMenuButton widget are placed together in a composite

widget that manages their side-by-side placement. The AbbrevMenuButton widget uses the Current Selection Widget only for previewing the default item in the menu. The application is responsible for using it to display the current selection, and if the Current Selection Widget is an editable field, for adding the new item to the menu as appropriate.

Coloration

For 3D, AbbrevMenuButton coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNbackground` and `XtNforeground` are used to render the AbbrevMenuButton as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to OL_INPUT_FOCUS_COLOR, then the background of the AbbrevMenuButton will be drawn with the value of `XtNinputFocusColor` when the widget receives input focus. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the widget inverts `XtNforeground` and `XtNbackground`. Once the input focus leaves the widget, the original coloration is restored.

Keyboard Traversal

The default value of the `XtNtraversalOn` resource is TRUE. The AbbrevMenuButton responds to the following navigation keys:

- SELECTKEY: The response depends on the value of the toolkit resource `XtNselectDoesPreview`. If `XtNselectDoesPreview` is TRUE, this key activates the default item in the Menu. If `XtNselectDoesPreview` is FALSE, this key pops up the Menu and transfers the keyboard focus to the default item in the Menu.
- MENUKEY: Pops up the Menu and transfers the keyboard focus to the default item in the Menu.
- MOVEDOWN: Pops up the Menu and transfers the keyboard focus to the default item in the Menu.

- NEXTFIELD and MOVERIGHT move to the next traversable widget in the shell.
- PREVFIELD, MOVEUP, and MOVELEFT move to the previous traversable widget in the shell.
- NEXTWINDOW moves to the next window in the application.
- PREVWINDOW moves to the previous window in the application.
- NEXTAPP moves to the first window in the next application.
- PREVAPP moves to the first window in the previous application.
- The response of the Menu associated with the AbbrevMenuButton is specified in the MenuShell widget's description. See page 414.

Keyboard Mnemonic Display

The AbbrevMenuButton does not display the mnemonic accelerator. If the AbbrevMenuButton is the child of a Caption widget, the Caption widget can be used to display the mnemonic.

Keyboard Accelerator Display

The AbbrevMenuButton does not display the keyboard accelerator. If the AbbrevMenuButton is a child of a Caption widget, the Caption widget can be used to display the accelerator as part of the label.

Resources

Table 6-1 AbbrevMenuButton Core Resources

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	G
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI

Table 6-1 AbbrevMenuButton Core Resources (Continued)

Name	Type	Default	Access
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	(calculated)	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive	Boolean	TRUE	GIO
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	(calculated)	SGI
XtNx	Position	0	SGI
XtNy	Position	0	SGI

Table 6-2 AbbrevMenuButton Primitive Resources

Name	Type	Default	Access
XtNaccelerator	String	NULL	SGI
XtNacceleratorText	String	NULL	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNfont	OlFont	XtDefaultFont	SGID
XtNfontColor	Pixel	XtDefaultForeground	SGID
XtNforeground	Pixel	XtDefaultForeground	SGID
XtNinputFocusColor	Pixel	Red	SGID
XtNmnemonic	unsigned char	'\0'	SGI
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNscale	int	12	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNuserData	XtPointer	NULL	SGI

Table 6-3 AbbrevMenuButton Resources

Name	Type	Default	Access
XtNmenuPane	Widget	(special)	G
XtNpreviewWidget	Widget	NULL	SGI

The following table lists the AbbrevMenuButton resources that are propagated to the MenuShell subwidget.

Table 6-4 AbbrevMenuButton Subwidget Resources[1]

Name	Type	Default	Access
XtNcenter	Boolean	TRUE	I
XtNhPad	Dimension	4	I
XtNhSpace	Dimension	4	I
XtNlayoutType	OlDefine	OL_FIXEDROWS	I[2]
XtNmeasure	int	1	I[2]
XtNpushpin	OlDefine	OL_NONE	I
XtNpushpinDefault	Boolean	FALSE	I
XtNsameSize	OlDefine	OL_COLUMNS	I
XtNshellTitle	OlStr	(widget name)	SGI
XtNvPad	Dimension	4	I
XtNvSpace	Dimension	4	I

1. These subwidget resources are described in the sections "ControlArea Widget" on page 249 and "MenuShell Widget" on page 414.

2. These resources can only be set programmatically via XtCreateWidget, XtVaCreateWidget, etc. Application resource file settings will not apply to these resources.

XtNmenuPane

Class	Type	Default	Access
XtCMenuPane	Widget	(special)	G

Synopsis: The widget where menu items can be added.
Values: ID of the menupane widget contained in the AbbrevMenuButton's MenuShell.

The value of this resource is available once the AbbrevMenuButton widget has been created.

XtNpreviewWidget

Class	Type	Default	Access
XtCPreviewWidget	Widget	NULL	SGI

Synopsis: The Current Selection Widget that the AbbrevMenuButton can use for previewing the Default Item.

Values: ID of an existing widget; this should be a StaticText, TextField, or TextLine widget.

When the user presses SELECT over the AbbrevMenuButton widget, it uses the location and size of the Current Selection Widget to display the label of the Default Item. The preview is constrained to be within the height and width of the Current Selection Widget. If the Current Selection Widget is not defined or is not mapped, previewing does not take place.

Activation Types

The following table lists the activation types used by the AbbrevMenuButton.

Table 6-5 AbbrevMenuButton Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MENU	MENU	XtNmenuBtn
OL_MENUKEY	MENU	XtNmenuKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_SELECT	SELECT	XtNselectBtn
OL_SELECTKEY	SELECT	XtNselectKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

Activation types not described in the following list are described in “Common Activation Types” on page 68.

OL_MENU

The OL_MENU activation type can be used to pop up the menu in two different modes: press-drag-release and click-move-click. These modes are described in the *OPEN LOOK GUI Functional Specification* section “Using Menu Buttons” in Chapter 15. The position of the menu depends on the space available on the

screen and is described in the *OPEN LOOK GUI Functional Specification* section "Menu Placement" in Chapter 15.

OL_MENUKEY

The OL_MENUKEY activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

OL_SELECT

The activation of the AbbrevMenuButton widget with the SELECT button depends on the value of the toolkit resources `XtNselectDoesPreview`. When the resource `XtNselectDoesPreview` is FALSE, this activation type will behave exactly as the OL_MENU activation type described previously. When `XtNselectDoesPreview` is TRUE, the SELECT action can be used as a shortcut to display and activate the menu default as described in the *OPEN LOOK GUI Functional Specification* section "Button Controls" in Chapter 4.

OL_SELECTKEY

When the AbbrevMenuButton has keyboard focus, the OL_SELECTKEY activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

See Also

"ControlArea Widget" on page 249,
"MenuButton Widget" on page 403,
"MenuShell Widget" on page 414.

BulletinBoard Widget

Class

Class Name: `BulletinBoard`
Class Pointer: `bulletinBoardWidgetClass`

Ancestry

Core-Composite-Constraint-Manager-BulletinBoard

Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/BulletinBo.h>
```

Description

The BulletinBoard widget is a composite widget that enforces no ordering on its children. It is up to the application to specify the x- and y-coordinates of each child inserted. Otherwise, it will be placed in the upper left corner of the BulletinBoard widget. The BulletinBoard can be mapped with no children. It displays an empty space which can be surrounded by a border.

Keyboard Traversal

The BulletinBoard widget is a Composite widget and cannot be accessed via keyboard traversal. Input focus moves between the Primitive children of this widget.

Coloration

The following diagram illustrates the resources used for BulletinBoard coloration.

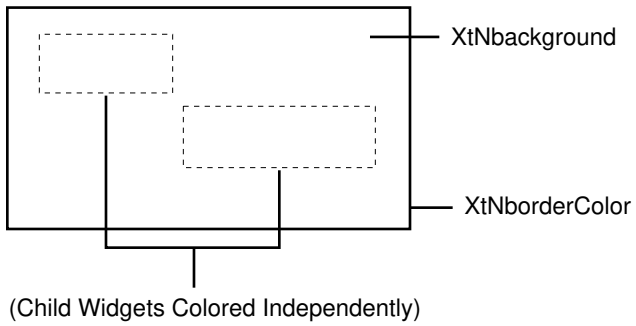

Figure 6-3 BulletinBoard Coloration

Resources

Table 6-6 BulletinBoard Core Resources

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	G
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive	Boolean	TRUE	GIO
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	0	SGI

Table 6-6 BulletinBoard Core Resources (Continued)

Name	Type	Default	Access
XtNx	Position	0	SGI
XtNy	Position	0	SGI

Table 6-7 BulletinBoard Composite Resources

Name	Type	Default	Access
XtNchildren	WidgetList	NULL	G
XtNinsertPosition	XtOrderProc	NULL	SGI
XtNnumChildren	Cardinal	0	G

Table 6-8 BulletinBoard Manager Resources

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	n/a
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNunrealizeCallback	XtCallbackList	NULL	SGIO
XtNuserData	XtPointer	NULL	SGI

Table 6-9 BulletinBoard Resources

Name	Type	Default	Access
XtNlayout	OlDefine	OL_MINIMIZE	SGI

XtNlayout

Class	Type	Default	Access
XtCLayout	OlDefine	OL_MINIMIZE	SGI

Synopsis: The layout policy the BulletinBoard widget is to follow.

Values: `OL_MINIMIZE/"minimize"` – The BulletinBoard widget will always be just large enough to contain all its children, regardless of any provided width and height values. Thus, the BulletinBoard widget will grow and shrink depending on the size needs of its children.
`OL_IGNORE/"ignore"` – The BulletinBoard widget will honor its own width and height; it will not grow or shrink in response to the addition, deletion, or alteration of its children.
`OL_MAXIMIZE/"maximize"` – The BulletinBoard widget will grow as required due to new or altered children, but will not shrink.

Activation Types

The following table lists the activation types used by the BulletinBoard.

Table 6-10 BulletinBoard Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

The BulletinBoard widget has no activation types besides the ones in "Common Activation Types" on page 68.

Caption Widget

Class

Class Name: `Caption`
Class Pointer: `captionWidgetClass`

Ancestry

Core-Composite-Constraint-Manager-Caption

Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/Caption.h>
```

Description

The Caption composite widget provides a convenient way to label an arbitrary widget.

Components

The Caption widget has two parts: the label and the child widget.

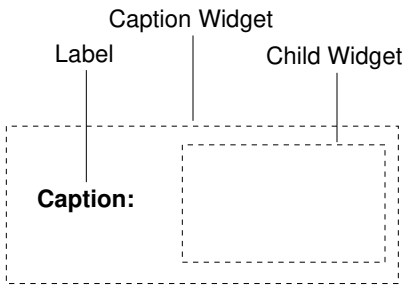

Figure 6-4 Caption Widget

Layout Control

The application can specify that the label goes above, below, to the left, or to the right of the child, and how far away the label is to be placed.

Child Constraints

The Caption composite allows only one child; attempts to add more than one are refused with a warning. If the Caption widget is mapped without a child widget, or if the child widget is not managed, only the label is shown.

Coloration

The following diagram illustrates the resources used for Caption coloration.

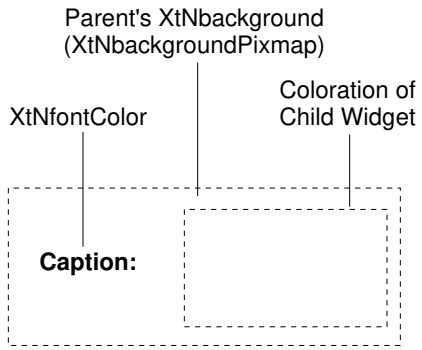

Figure 6-5 Caption Coloration

Keyboard Traversal

The Caption is a special Manager widget that can be used to display the mnemonic for its single child. However, the label used as a caption to the child is not accessible via keyboard traversal.

The action of a mnemonic on a Caption widget is used for traversal as well as performing a SELECT on the Caption widget's child widget.

Keyboard Mnemonic Display

The Caption widget displays its mnemonic as part of its label. If the mnemonic character is in the label, then that character is marked according to the value of the toolkit resource `XtNshowMnemonics`. If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and marked according to the value of `XtNshowMnemonics`.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

Keyboard Accelerator Display

The Caption widget displays the keyboard accelerator for its child as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic), right justified, and separated by at least one space.

If truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

Resources

Table 6-11 Caption Core Resources

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	G
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive	Boolean	TRUE	GIO

Table 6-11 Caption Core Resources (Continued)

Name	Type	Default	Access
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	0	SGI
XtNx	Position	0	SGI
XtNy	Position	0	SGI

Table 6-12 Caption Composite Resources

Name	Type	Default	Access
XtNchildren	WidgetList	NULL	G
XtNinsertPosition	XtOrderProc	NULL	SGI
XtNnumChildren	Cardinal	0	G

Table 6-13 Caption Manager Resources

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	SGID
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNunrealizeCallback	XtCallbackList	NULL	SGIO
XtNuserData	XtPointer	NULL	SGI

Table 6-14 Caption Resources

Name	Type	Default	Access
XtNalignment	OlDefine	OL_CENTER	SGI
XtNfont	OlFont	OlDefaultBoldFont	SGID
XtNfontColor	Pixel	XtDefaultForeground	SGID
XtNlabel	OlStr	(instance name)	SGI
XtNmnemonic	unsigned char	'\0'	SGI
XtNposition	OlDefine	OL_LEFT	SGI

Table 6-14 Caption Resources (Continued)

Name	Type	Default	Access
XtNrecomputeSize	Boolean	TRUE	SGI
XtNspace	Dimension	4	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI

XtNalignment

Class	Type	Default	Access
XtCAlignment	OlDefine	OL_CENTER	SGI

Synopsis: The alignment of the label relative to the child widget.

Values: `OL_BOTTOM/"bottom"` – Align the bottom edge of the label with the bottom edge of the child widget.
`OL_CENTER/"center"` – Align the center of the label with the center of the child widget.
`OL_LEFT/"left"` – Align the left edge of the label with the left edge of the child widget.
`OL_RIGHT/"right"` – Align the right edge of the label with the right edge of the child widget.
`OL_TOP/"top"` – Align the top edge of the label with the top edge of the child widget.

The `XtNalignment` and `XtNposition` resources interact in the following way. If `XtNposition` is OL_LEFT or OL_RIGHT, then the alignment can be OL_TOP, OL_CENTER, or OL_BOTTOM. If `XtNposition` is OL_TOP or OL_BOTTOM, then the alignment can be OL_LEFT, OL_CENTER, or OL_RIGHT.

XtNfont

Class	Type	Default	Access
XtCFont	OlFont	OlDefaultBoldFont	SGID

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfont" on page 26.

XtNfontColor

Class	Type	Default	Access
XtCFontColor	Pixel	XtDefaultForeground	SGID

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfontColor" on page 27.

XtNlabel

Class	Type	Default	Access
XtCLabel	OlStr	(instance name)	SGI

Synopsis: The label text.
Values: Any `OlStr` value valid in the current locale. NULL is the same as the empty string.

The label is displayed as given; no punctuation (such as a colon) is added. Control characters (other than spaces) are ignored without warning. For example, embedded newlines do not cause line breaks.

XtNmnemonic

Class	Type	Default	Access
XtCMnemonic	unsigned char	'\0'	SGI

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNmnemonic" on page 28.

XtNposition

Class	Type	Default	Access
XtCPosition	OlDefine	OL_LEFT	SGI

Synopsis: The placement of the label in relation to the child widget.
Values: `OL_BOTTOM/"bottom"` – The label is below the child widget.
`OL_LEFT/"left"` – The label is to the left of the child widget.
`OL_RIGHT/"right"` – The label is to the right of the child widget.
`OL_TOP/"top"` – The label is above the child widget.

XtNrecomputeSize

Class	Type	Default	Access
XtCRecomputeSize	Boolean	TRUE	SGI

Synopsis: The widget resize policy.
Values: `TRUE/"true"` – The widget resizes itself to accommodate changes in its children's sizes due to changes in resources such as fonts or labels.
`FALSE/"false"` – The widget does not resize itself.

XtNspace

Class	Type	Default	Access
XtCSpace	Dimension	4	SGI

Synopsis: The separation of the label from the child widget, in pixels.
Values: 0 ≤ `XtNspace`

The separation of the label and child widget is shown in the following figure.

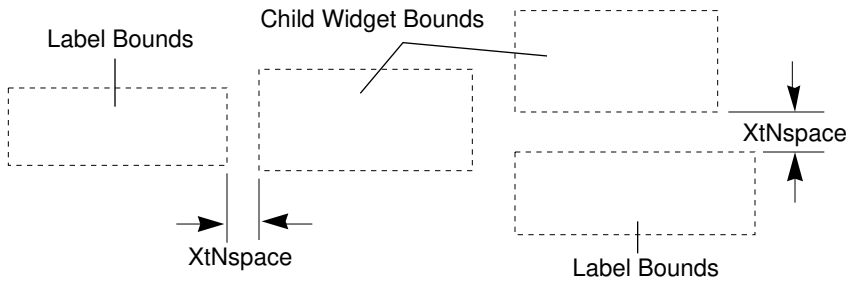

Figure 6-6 Label and Child Widget Spacing

XtNtextFormat

Class	Type	Default	Access
XtCTextFormat	OlStrRep	OL_SB_STR_REP	GI

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNtextFormat" on page 29.

Activation Types

The following table lists the activation types used by the Caption.

Table 6-15 Caption Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey

Table 6-15 Caption Activation Types (Continued)

Activation Type	Semantics	Resource Name
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_SELECT	SELECT	XtNselectBtn
OL_SELECTKEY	SELECT	XtNselectKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

Activation types not described in the following list are described in "Common Activation Types" on page 68.

OL_SELECT/
OL_SELECTKEY

The Caption widget does not respond to any user gestures, but a client can activate it with `OlActivateWidget()` and an activation type of OL_SELECT or OL_SELECTKEY. When so activated, the Caption widget will move focus to its child widget and then activate the child with the OL_SELECT activation type.

CheckBox Widget

Class

Class Name: `CheckBox`
Class Pointer: `checkBoxWidgetClass`

Ancestry

Core-Composite-Constraint-Manager-CheckBox

Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/CheckBox.h>
```

Description

The CheckBox widget is similar in function to the RectButton widget. Several CheckBoxes are typically used together to provide the user with a set of options that can be toggled on or off.

Components

The CheckBox widget consists of a label next to a Check Box; the Check Box will have a Check Mark, if selected.

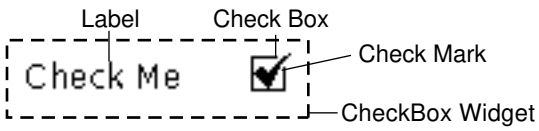

Figure 6-7 CheckBox Widget

The following figure shows several buttons, in unselected and selected, as well as normal and dim states.

Figure 6-8 Check Boxes in Various States

Typical Use

Check Boxes may be used alone, but are usually used in the Nonexclusives composite widget, where they are used to implement a several-of-many selection. Making the CheckBox widget a child of a different composite widget will not produce an error, but proper behavior is not guaranteed.

Operations

A CheckBox widget has two states: "set" and "not set." When set, the Check Mark is visible. Toggling this state alternates a resource (`XtNset`) between logical TRUE and FALSE and starts an action associated with the check box. Clicking SELECT on a check box toggles the state associated with it. Pressing SELECT, or moving the pointer into the check box while SELECT is pressed, adds or removes the Check Mark to reflect the state the check box would be in if SELECT was released. Releasing SELECT toggles the state. Moving the pointer off the check box before releasing SELECT restores the original Check Box, but does not toggle the state. Clicking or pressing MENU does not do anything in the CheckBox widget; the event is passed up to an ancestor widget.

Bounds on SELECT

Only the Check Box and Check Mark respond to SELECT, as shown in the following figure.

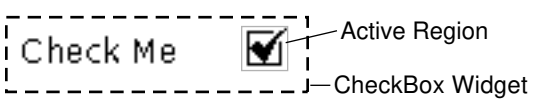

Figure 6-9 CheckBox Widget

Coloration

For both 3D and 2D, the background of the CheckBox widget is drawn in the parent's `XtNbackground` resource. The label is drawn using `XtNfontColor`. The checkmark is drawn using `XtNforeground`.

For 3D, the check box component coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." The parent's `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNforeground` is used to render the outline of the check box component as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to OL_INPUT_FOCUS_COLOR, then the background of the check box component will be drawn with the value of `XtNinputFocusColor` when the widget receives input focus. However, if `XtNinputFocusColor` is the same as the parent's `XtNbackground`, then the widget inverts `XtNforeground` and the parent's `XtNbackground` inside the check box component. Once the input focus leaves the widget, the original coloration is restored.

Keyboard Traversal

The default value of the `XtNtraversalOn` resource is TRUE. The CheckBox widget responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVFIELD moves to the previous traversable widget in the window
- MOVEUP moves to the CheckBox above the current widget in the Nonexclusives composite
- MOVEDOWN moves to the CheckBox below the current widget in the Nonexclusives composite
- MOVELEFT moves to the CheckBox to the left of the current widget in the Nonexclusives composite
- MOVERIGHT moves to the CheckBox to the right of the current widget in the Nonexclusives composite
- NEXTWINDOW moves to the next window in the application
- PREVWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVAPP moves to the first window in the previous application

Keyboard Mnemonic Display

The CheckBox widget displays its mnemonic as part of its label. If the mnemonic character is in the label, then that character is marked according to the value of the toolkit resource `XtNshowMnemonics`. If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and marked according to the value of `XtNshowMnemonics`.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

Keyboard Accelerator Display

The CheckBox widget displays the keyboard accelerator as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic) separated by at least one space. The accelerator text is right justified.

If truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

CheckBox Appearance

The `XtNdim` and `XtNset` resources can be set independently, as shown in the following state table.

Table 6-16 CheckBox Appearance with Set/Default/Dim

XtNset	XtNdim	Check Box Appearance
TRUE	TRUE	
TRUE	FALSE	
FALSE	TRUE	
FALSE	FALSE	

Label Resource Interactions

The `XtNwidth`, `XtNheight`, `XtNrecomputeSize`, and `XtNlabelJustify` resources interact to produce a truncated, clipped, centered, left-justified, or right-justified Label and Check Box as shown in the table below.

Table 6-17 CheckBox Label and Check Box Appearance

XtNwidth	XtNrecomputeSize	XtNlabelJustify	Result
any value	TRUE	any	Just Fits
> needed for label	FALSE	OL_LEFT	Left Justified
> needed for label	FALSE	OL_RIGHT	Right Justified
< needed for label	FALSE	any	Truncated
XtNheight	**XtNrecomputerSize**	**XtNlabelJustify**	**Result**
any value	TRUE	any	Just Fits
> needed for label	FALSE	any	Centered
< needed for label	FALSE	any	Clipped

When the label is left-justified, right-justified, or centered the extra space is filled with the background color of the CheckBox widget's parent, as determined by the `XtNbackground` and `XtNbackgroundPixmap` resources of the parent. See also the `XtNlabelTile` resource for how it affects the appearance of a label.

Resources

Table 6-18 CheckBox Core Resources

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	G
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI

Table 6-18 CheckBox Core Resources (Continued)

Name	Type	Default	Access
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive	Boolean	TRUE	GIO
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	0	SGI
XtNx	Position	0	SGI
XtNy	Position	0	SGI

Table 6-19 CheckBox Composite Resources

Name	Type	Default	Access
XtNchildren	WidgetList	NULL	G
XtNinsertPosition	XtOrderProc	NULL	SGI
XtNnumChildren	Cardinal	0	G

Table 6-20 CheckBox Manager Resources

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	SGID
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNunrealizeCallback	XtCallbackList	NULL	SGIO
XtNuserData	XtPointer	NULL	SGI

Table 6-21 CheckBox Resources

Name	Type	Default	Access
XtNaccelerator	String	NULL	SGI
XtNacceleratorText	String	NULL	SGI
XtNdim	Boolean	FALSE	SGI
XtNfont	OlFont	XtDefaultFont	SGID
XtNfontColor	Pixel	XtDefaultForeground	SGID
XtNforeground	Pixel	XtDefaultForeground	SGID
XtNlabel	OlStr	(instance name)	SGI
XtNlabelImage	XImage *	NULL	SGI
XtNlabelJustify	OlDefine	OL_LEFT	SGI
XtNlabelTile	Boolean	FALSE	SGI
XtNlabelType	OlDefine	OL_STRING	SGI
XtNmnemonic	unsigned char	'\0'	SGI
XtNposition	OlDefine	OL_LEFT	SGI
XtNrecomputeSize	Boolean	TRUE	SGI
XtNscale	int	12	SGI
XtNselect	XtCallbackList	NULL	SGIO
XtNset	Boolean	FALSE	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNunselect	XtCallbackList	NULL	SGIO

XtNaccelerator

Class	Type	Default	Access
XtCAccelerator	String	NULL	SGI

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNaccelerator" on page 25.

XtNacceleratorText

Class	Type	Default	Access
XtCAcceleratorText	String	NULL	SGI

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNacceleratorText" on page 25.

XtNdim

Class	Type	Default	Access
XtCDim	Boolean	FALSE	SGID

Synopsis: The visual appearance of the CheckBox in reflecting the state of associated objects.

Values: `TRUE/"true"` – The check box border is dimmed to show that the check box represents the state of one or more of several objects that, as a group, are in different states.
`FALSE/"false"` – The border does not show the state of underlying objects.

XtNfont

Class	Type	Default	Access
XtCFont	OlFont	XtDefaultFont	SGID

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfont" on page 26.

XtNfontColor

Class	Type	Default	Access
XtCFontColor	Pixel	XtDefaultForeground	SGID

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfontColor" on page 27.

XtNforeground

Class	Type	Default	Access
XtCForeground	Pixel	XtDefaultForeground	SGID

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNforeground" on page 27.

XtNlabel

Class	Type	Default	Access
XtCLabel	OlStr	(instance name)	SGI

Synopsis: The text for the label.

Values: Any `OlStr` value valid in the current locale.

The default value is the name of the widget as specified in the `XtCreateWidget()` routine. This resource will be ignored if the `XtNlabelType` resource has the value OL_IMAGE.

XtNlabelImage

Class	Type	Default	Access
XtCLabelImage	XImage *	NULL	SGI

Synopsis: The image for the label of the CheckBox widget.

This resource will be ignored unless the `XtNlabelType` resource has the value OL_IMAGE. If the image is smaller than the space available for it next to the check box, it will be right- or left-justified horizontally, depending on the value of the `XtNlabelJustify` resource. If the image is larger than the space available for it, it will be clipped so that it does not display outside the space. See, however, `XtNlabelTile` for alternative behavior.

XtNlabelJustify

Class	Type	Default	Access
XtCLabelJustify	OlDefine	OL_LEFT	SGI

Synopsis: The justification of the label, if the `XtNwidth` resource gives more space than needed.

Values: `OL_LEFT/"left"` – Left-justify the label.
`OL_RIGHT/"right"` – Right-justify the label.

The label will be justified within the space available next to the Check Box.

XtNlabelTile

Class	Type	Default	Access
XtCLabelTile	Boolean	FALSE	SGI

Synopsis: The tiling of the background of the CheckBox rectangle.

Values: `TRUE/"true"` – The label area is tiled with the image to fill the background if the image is smaller than the background of the subobject.
`FALSE/"false"` – The label area is placed as described by the `XtNlabelImage` resource.

This resource augments the `XtNlabelImage` resource to allow tiling of the background. The `XtNlabelTile` resource is ignored for text labels.

XtNlabelType

Class	Type	Default	Access
XtCLabelType	OlDefine	OL_STRING	SGI

Synopsis: The form that the label takes.
Values: `OL_STRING/"string"` – The label is a text string.
`OL_IMAGE/"image"` – The label is an image.

XtNmnemonic

Class	Type	Default	Access
XtCMnemonic	unsigned char	'\0'	SGI

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNmnemonic" on page 28.

XtNposition

Class	Type	Default	Access
XtCPosition	OlDefine	OL_LEFT	SGI

Synopsis: On which side of the CheckBox the label will be placed.
Values: `OL_LEFT/"left"` – Place the label on the left.
`OL_RIGHT/"right"` – Place the label on the right.

XtNrecomputeSize

Class	Type	Default	Access
XtCRecomputeSize	Boolean	TRUE	SGI

Synopsis: Whether the CheckBox widget should calculate its size.
Values: `TRUE/"true"` – The CheckBox widget will do normal size calculations that may cause its geometry to change, and automatically set the `XtNheight` and `XtNwidth` resources.
`FALSE/"false"` – The CheckBox widget will leave its size unchanged; this may cause truncation of the visible image being shown by the CheckBox widget if the fixed size is too small, or may cause padding if the fixed size is too large. The location of the padding is determined by the `XtNlabelJustify` resource.

XtNscale

Class	Type	Default	Access
XtCScale	int	12	SGI

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNscale" on page 29.

XtNselect

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked when the user toggles the widgetinto "set" mode, making `XtNset` be TRUE. Simply setting `XtNset` to TRUE with a call to `XtSetValues()` does not issue the `XtNselect` callbacks.

XtNset

Class	Type	Default	Access
XtCSet	Boolean	FALSE	SGI

Synopsis: The current state of the check box.
Values: `TRUE/"true"` – The check mark will be present.
`FALSE/"false"` – The check mark will not be present.

XtNtextFormat

Class	Type	Default	Access
XtCTextFormat	OlStrRep	OL_SB_STR_REP	GI

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNtextFormat" on page 29.

XtNunselect

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked when the user toggles the widget into "unset" mode, making `XtNset` be FALSE. Simply setting `XtNset` to FALSE with a call to `XtSetValues()` does not issue the `XtNunselect` callbacks.

Activation Types

The following table lists the activation types used by the CheckBox.

Table 6-22 CheckBox Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_SELECT	SELECT	XtNselectBtn
OL_SELECTKEY	SELECT	XtNselectKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

Activation types not described in the following list are described in "Common Activation Types" on page 68.

OL_SELECT/
OL_SELECTKEY

The activation of a CheckBox is described in the *OPEN LOOK GUI Functional Specification* section "Check Boxes" in Chapter 4. When the CheckBox is activated with either OL_SELECT or OL_SELECTKEY, the state of the `XtNset` resource will be reversed. When the `XtNset` resource goes to FALSE, the `XtNunselect` callback will be called; when the `XtNset` resource goes to TRUE, the `XtNselect` callback will be called.

See Also

"RectButton Widget" on page 489.

ControlArea Widget

Class

Class Name: `ControlArea`
Class Pointer: `controlAreaWidgetClass`

Ancestry

Core-Composite-Constraint-Manager-ControlArea

Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/ControlAre.h>
```

Description

The ControlArea is a composite widget that organizes its child controls in rows and columns.

Components

The ControlArea widget has zero or more child widgets and an optional border. The ControlArea can also provide changebars alongside a child. Changebars are provided by the *OPEN LOOK GUI Functional Specification* as a means of feedback that a change has occurred in the corresponding object.

Layout Control

The application can choose one of four simple layout schemes:

- Fixed number of columns in the control pane
- Fixed number of rows
- Fixed overall width of the control area
- Fixed overall height

The application can also specify the inter-control spacing and the size of the margin around the children.

The children in each row align at the top of the row. The distance between the top of one row and the next is the height of the tallest child in the row plus the application-specified inter-row spacing.

Coloration

The following diagram illustrates the resources used for ControlArea coloration.

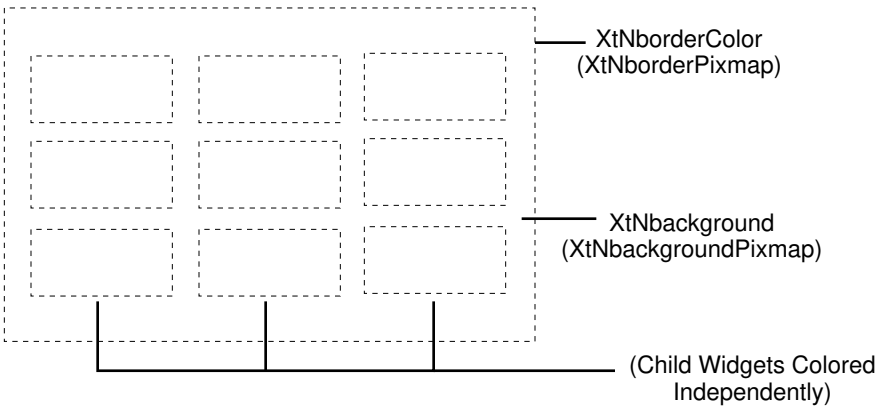

Figure 6-10 ControlArea Coloration

Resources

Table 6-23 ControlArea Core Resources

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	G
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO

Table 6-23 ControlArea Core Resources (Continued)

Name	Type	Default	Access
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive	Boolean	TRUE	GIO
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	0	SGI
XtNx	Position	0	SGI
XtNy	Position	0	SGI

Table 6-24 ControlArea Composite Resources

Name	Type	Default	Access
XtNchildren	WidgetList	NULL	G
XtNinsertPosition	XtOrderProc	NULL	SGI
XtNnumChildren	Cardinal	0	G

Table 6-25 ControlArea Manager Resources

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	SGID
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNunrealizeCallback	XtCallbackList	NULL	SGIO
XtNuserData	XtPointer	NULL	SGI

Table 6-26 ControlArea Resources

Name	Type	Default	Access
XtNalignCaptions	Boolean	FALSE	SGI
XtNallowChangeBars	Boolean	FALSE	SGI
XtNcenter	Boolean	FALSE	SGI

Table 6-26 ControlArea Resources (Continued)

Name	Type	Default	Access
XtNhPad	Dimension	4	SGI
XtNhSpace	Dimension	4	SGI
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNmeasure	int	1	SGI
XtNsameSize	OlDefine	OL_COLUMNS	SGI
XtNvPad	Dimension	4	SGI
XtNvSpace	Dimension	4	SGI

Table 6-27 ControlArea Subwidget Resources

Name	Type	Default	Access
XtNchangeBar	OlDefine	OL_NONE	SGI

XtNallowChangeBars

Class	Type	Default	Access
XtCAllowChangeBars	Boolean	FALSE	SGI

Synopsis: Whether the ControlArea displays changebars.
Values: `TRUE/"true"` – The ControlArea displays changebars.
`FALSE/"false"` – The ControlArea does not display changebars.

The ControlArea supports changebars only if the `XtNlayoutType` resource value is OL_FIXEDCOLS or OL_FIXEDROWS.

XtNalignCaptions

Class	Type	Default	Access
XtCAlignCaptions	Boolean	FALSE	SGI

Synopsis: The alignment of Caption widgets in the ControlArea.
Values: `TRUE/"true"` – The ControlArea aligns all Caption widgets in each column so that their captions are right- or left-justified, depending on the position of the Caption's label (see Figure 6-11 on page 253). This may affect the width calculation for a column.
`FALSE/"false"` – The ControlArea aligns all Caption widgets the same as other widgets—by their overall width.

`XtNalignCaptions` takes precedence over the `XtNcenter` resource, but only for Caption widgets.

The effective width for the Caption widgets in a column becomes the sum of the width of the widest caption, plus the largest caption/child widget separation and child widget width. This alignment is only for groups of Caption widgets with all their captions on the left or the right.

Mixed orientation, or captions above or below, cannot be aligned well.

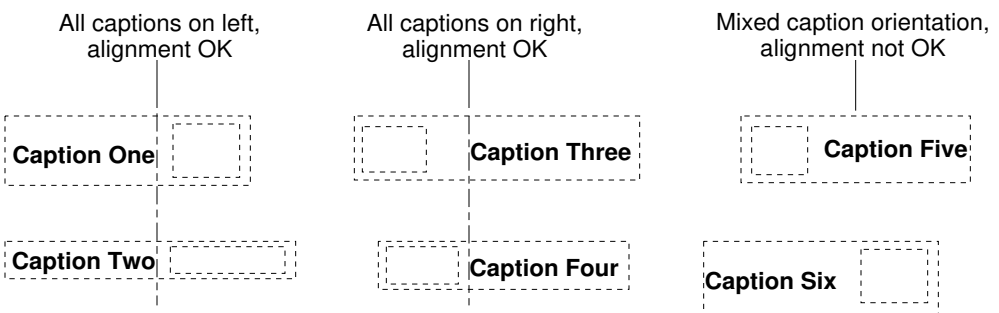

Figure 6-11 Aligning Captions

XtNchangeBar

Class	Type	Default	Access
XtCChangeBar	OlDefine	OL_NONE	SGI

Synopsis: The type of change bar to be displayed. This resource should be set for the child that requires the changebar.

Values: `OL_NONE/"none"` – Do not display any change bar
`OL_NORMAL/"normal"` – Display a solid vertical line.
`OL_DIM/"dim"` – Display a dimmed vertical line.

XtNcenter

Class	Type	Default	Access
XtCCenter	Boolean	FALSE	SGI

Synopsis: The orientation of each widget within a ControlArea column.

Values: `TRUE/"true"` – Center each widget within each column.
`FALSE/"false"` – Left-justify each widget within each column, except where the `XtNalignCaptions` resource applies.

XtNhPad/ XtNvPad

Class	Type	Default	Access
XtCHPad	Dimension	4	SGI
XtCVPad	Dimension	4	SGI

Synopsis: The amount of padding, in pixels, to leave around the edges of the control area.

Values: 0 ≤ `XtNhPad`
0 ≤ `XtNvPad`

The action of these resources is illustrated by the following figure.

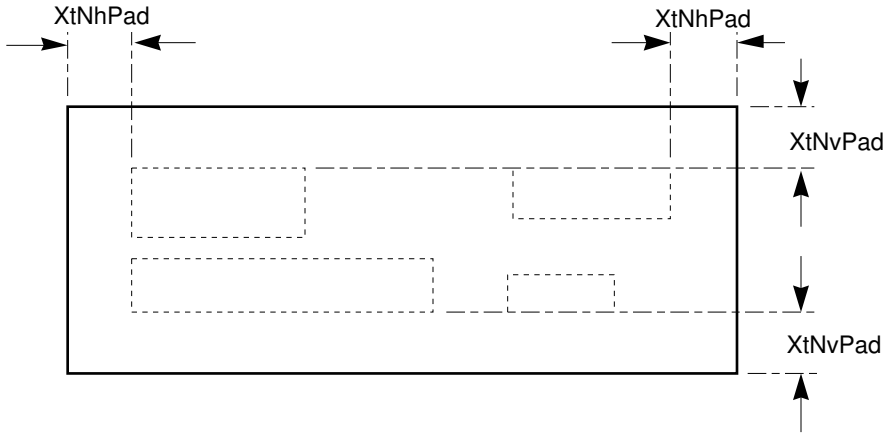

Figure 6-12 ControlArea Padding Around Controls

XtNhSpace/ XtNvSpace

Class	Type	Default	Access
XtCHSpace	Dimension	4	SGI
XtCVSpace	Dimension	4	SGI

Synopsis: The amount of space, in pixels, to leave between controls.

Values: 0 ≤ `XtNhSpace`
0 ≤ `XtNvSpace`

If the controls are of different sizes in a row or column, the spacing applies to the widest or tallest dimension of all the controls.

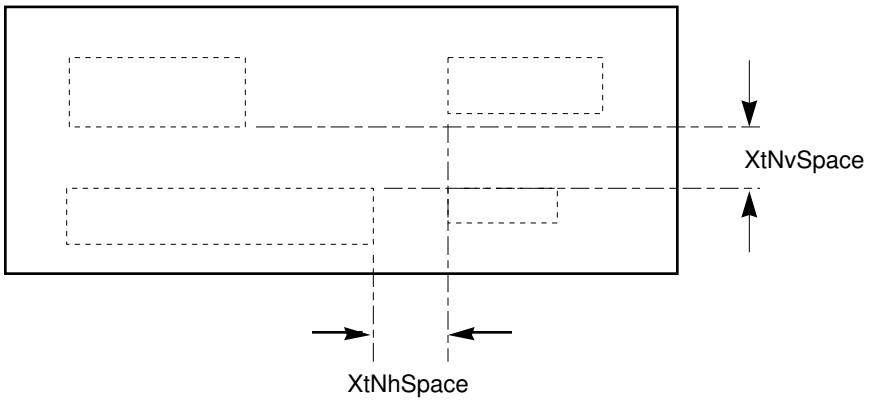

Figure 6-13 Spacing Between Controls

XtNlayoutType

Class	Type	Default	Access
XtCLayoutType	OlDefine	OL_FIXEDROWS	SGI

Synopsis: The layout of the child widgets by the ControlArea.

Values: `OL_FIXEDROWS/"fixedrows"` – The layout will have a fixed number of rows and enough columns to hold all the controls; `OL_FIXEDCOLS/"fixedcols"` – The layout will have a fixed number of columns and enough rows to hold all the controls; `OL_FIXEDWIDTH/"fixedwidth"` – The layout will be of a fixed width, but tall enough to hold all the controls; `OL_FIXEDHEIGHT/"fixedheight"` – The layout will be of a fixed height, but wide enough to hold all the controls.

The `XtNmeasure` resource gives the number of rows or columns or the fixed height or width, in pixels.

The choices are to specify the number of rows or columns, or to specify the overall height or width of the layout area. Only one of these dimensions can be specified directly; the other is determined by the number of controls added.

For instance, if the application specifies that the control area should have four columns, the number of rows will be the number of controls divided by four.

XtNmeasure

Class	Type	Default	Access
XtCMeasure	int	1	SGI

Synopsis: The number of rows or columns in the layout of the child widgets, or the fixed width or height of the control area.

Values: 0 < `XtNmeasure`

If `XtNlayoutType` is OL_FIXEDROWS or OL_FIXEDCOLS, then the default is 1. If `XtNlayoutType` is OL_FIXEDWIDTH or OL_FIXEDHEIGHT, then the default is the width or height of the widest or tallest widget, depending on `XtNlayoutType`.

When `XtNlayoutType` is OL_FIXEDWIDTH or OL_FIXEDHEIGHT, the measure includes the padding on both edges and the inter-control spacing, as suggested by the following figure.

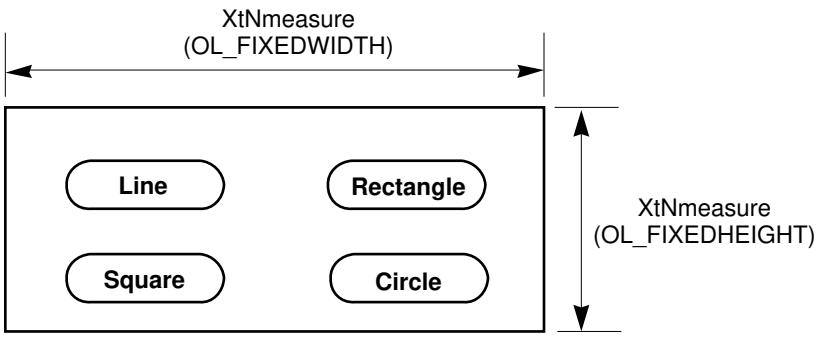

Figure 6-14 `XtNmeasure`

XtNsameSize

Class	Type	Default	Access
XtCSameSize	OlDefine	OL_COLUMNS	SGI

Synopsis: The controls within the ControlArea widget that are forced to be the same width (if any).

Values: `OL_NONE/"none"` – The controls are placed in fixed-width columns, but the size of each control is left alone. The width of each column is the width of the widest control in the column.
`OL_COLUMNS/"columns"` – Controls in each column are made the same width as the widest of them. The width of each column is

thus the width of the widest control in the column.
`OL_ALL/"all"` – All controls are made the same width, the width of the widest control in the ControlArea widget.

Note – The ControlArea widget does not resize its children when it gets resized. Use the Form or RubberTile widget if the children need to be resized.

Activation Types

The following table lists the activation types used by the ControlArea.

Table 6-28 ControlArea Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

The ControlArea widget has no activation types besides the ones in "Common Activation Types" on page 68.

See Also

"BulletinBoard Widget" on page 225,
"Caption Widget" on page 229,
"Form Widget" on page 385,
"RubberTile Widget" on page 502.

6

ControlArea Widget