Managing Geometry

The geometry of a widget consists of its size, location, and stacking order. Widgets often have preferred sizes and perhaps locations. For example, a Label widget may prefer to be just large enough to display the text of the label. But composite widgets usually have preferences or constraints in laying out their children, and these may conflict with the preferences of the child widgets. Furthermore, the user or the application can change a widget's geometry at any time; for example, by resizing the top−level window. Geometry management is the process by which the user, parent widgets, and child widgets negotiate the actual sizes and locations of the widgets in the application.

Following are some common occasions for geometry changes:

- The application manages or unmanages a child widget.
- The application sets a geometry resource.
- The application sets a resource that causes one of the geometry resources to change. For example, setting a new label for a Label widget may cause a geometry change.
- The user resizes a top−level window using the window manager.
- The user resizes a pane of a PanedWindow.

Following are the basic Core and RectObj resources that determine widget geometry:

XmNx

Specifies the x coordinate of the upper left outside corner (outside the border) of the widget's window. The value is relative to the upper left inside corner (inside the border) of the parent window.

XmNy

Specifies the y coordinate of the upper left outside corner (outside the border) of the widget's window. The value is relative to the upper left inside corner (inside the border) of the parent window.

XmNwidth

Specifies the inside width (excluding the border) of the widget's window.

XmNheight

Specifies the inside height (excluding the border) of the widget's window.

XmNborderWidth

Specifies the width of the border that surrounds the widget's window on all four sides. Note, however, that you should use resources like **XmNshadowThickness** and **XmNhighlightThickness** instead of **XmNborderWidth** to specify border widths.

Shells and Their Children

Shell widgets encapsulate application widgets, principally to communicate with the window manager. Motif has three shell classes based on Intrinsics shell classes:

VendorShell

Subclass of **WMShell** and superclass for other shell classes that contain both persistent top−level widgets and dialogs

XmDialogShell

Subclass of **TransientShell** (which is a subclass of **VendorShell**) used to contain dialog widgets, commonly subclasses of **XmBulletinBoard**

XmMenuShell

Subclass of **OverrideShell** used to contain RowColumn PulldownMenu and PopupMenu widgets

A shell has only one managed child. Except when a shell contains an off−the−spot input method, the shell's window is coincident with the child's window. The *geometry_manager* procedures of the shell classes treat geometry requests from the child as geometry requests for the shell, and the *resize* procedures of the shell classes make the child the same size as the shell. Applications should usually change the geometry of the child, not of the shell.

In particular, setting **XmNheight**, **XmNwidth**, or **XmNborderWidth** for either a shell or its child sets that resource to the same value in both the parent and the child. For a child of a shell, setting **XmNx** or **XmNy** sets the corresponding resource of the parent but does not change the child's position relative to the parent. **XtGetValues** for the child's **XmNx** or **XmNy** yields the value of the corresponding resource in the parent. The x and y coordinates of the child's upper left outside corner relative to the parent's upper left inside corner are both zero minus the value of **XmNborderWidth**.

The exception is a VendorShell or DialogShell that contains an off−the−spot input method. In this case, the input method appears inside the shell and below the application widget. The conventions for geometry parameters are the same as for other shells, except that the values of **XmNheight** for the child and the shell are not identical. The height of the shell is the sum of the height and border width of the application window and the height of the area occupied by the input method.

When the Shell resource **XmNallowShellResize** is False, a shell's *geometry_manager* procedure returns **XtGeometryNo** for all geometry requests from a realized child.

Manager Widgets and Their Children

Each Primitive widget has resources that determine its layout or contents. For example, the size of a Text widget depends on the values of the **XmNrows**, **XmNcolumns**, **XmNmarginHeight**, and **XmNmarginWidth** Text resources; the **XmNhighlightThickness** and **XmNshadowThickness** Primitive resources; and the basic Core geometry resources. In addition, when the Text **XmNresizeHeight** or **XmNresizeWidth** resource is True, the size of the widget can depend on the size of the text (the **XmNvalue** resource). Setting any of these resources can cause Text to generate a geometry request.

Manager widgets have their own layout policies, which they use in responding to geometry requests from their children or to resizing by their parents. These policies are determined by the Manager's own resources and, for some Managers, by its constraint resources.

Constraints are resources defined by the Manager but associated with each child. An application or user initializes, sets, or gets constraint resources for the child as if they were resources defined by the child's class. Initialization, **XtSetValues**, and **XtGetValues** for the child operate on the parent's constraint resources associated with that child. The Manager has constraint *initialize* and *set_values* procedures that allow it to set other constraints and recompute its layout.

Motif uses constraints in determining the layout of Form, PanedWindow, and Frame widgets. Motif also uses constraints to adjust the positions of child widgets in PanedWindow and RowColumn. The Form widget is discussed in Section 14.5. PanedWindow and Frame are discussed in Chapter 8.

Managing Geometry Using RowColumn

In addition to its role as the menu widget, RowColumn provides general−purpose layout and geometry management for child widgets arranged in rows, columns, or grids. The default RowColumn type, **XmWORK_AREA**, provides the layout features but not the menu semantics.

RowColumn's layout is controlled by two sets of resources. One set determines the position of children within the parent. The other set specifies whether RowColumn adjusts the internal layout characteristics of the children, such as margins and text alignment.

The two primary resources that control child positioning are **XmNorientation** and **XmNpacking**. **XmNorientation** determines whether RowColumn lays out its children in rows or columns. When **XmNorientation** is **XmVERTICAL**−−the default for a WorkArea−−the layout is column−major. When **XmNorientation** is **XmHORIZONTAL** the layout is row−major.

XmNpacking controls the general style of the layout. The resource has three possible values:

XmPACK_TIGHT

> RowColumn places children one after the other along the major dimension (for example, in a column when **XmNorientation** is **XmVERTICAL**). It proceeds until no more children fit along that dimension and then begins a new row or column. When **XmNorientation** is **XmVERTICAL** and the vertical distance remaining in the current column is too small to accommodate the child being placed, RowColumn begins a new column if **XmNresizeHeight** is False or the RowColumn cannot become larger. When placing children in a column, RowColumn does not alter their heights, but it makes the width of each child in the column equal to the width of the widest child in that column. Analogous rules apply to row−major layouts. **XmPACK_TIGHT** is the default value for **XmNpacking** in a WorkArea.

XmPACK_COLUMN

> RowColumn makes the width and height of each child identical. The width is the maximum width of all children, and the height is the maximum height. RowColumn uses the value of **XmNnumColumns** to determine the maximum number of columns (in **XmVERTICAL** orientation) or rows (in **XmHORIZONTAL** orientation) to produce. RowColumn tries to create **XmNnumColumns** columns (or rows) with an equal number of children in each column (or row).

XmPACK_NONE

> RowColumn does not change the position of any child. Unless **XmNresizeWidth** is False, it tries to grow large enough to enclose the greatest x extent of any child. Unless **XmNresizeHeight** is False, it tries to grow large enough to enclose the greatest y extent of any child.

Several other resources influence the position and size of children:

XmNadjustLast

> This resource applies only when **XmNpacking** is **XmPACK_TIGHT** or **XmPACK_COLUMN**. When this resource is True and the orientation is vertical, RowColumn increases the widths of children in the last column when necessary so that all children extend to the right edge of the RowColumn. When this resource is True and the orientation is horizontal, RowColumn increases the heights of children in the last row when necessary so that all children extend to the bottom edge of the RowColumn.

XmNentryBorder

> When this resource is nonzero, it specifies the border width for all children of the RowColumn. When this resource is zero, RowColumn does not alter the border width of its children.

XmNmarginHeight

> This resource specifies the amount of space between the top edge of the RowColumn and the first item in each column, and between the bottom edge of the RowColumn and the last item in each column.

XmNmarginWidth

> This resource specifies the amount of space between the left edge of the RowColumn and the first item in each row, and between the right edge of the RowColumn and the last item in each row.

XmNresizeHeight

When this resource is True, RowColumn adjusts its own height when possible to accommodate its children. When this resource is False, RowColumn does not request a new height during layout.

XmNresizeWidth

When this resource is True, RowColumn adjusts its own width when possible to accommodate its children. When this resource is False, RowColumn does not request a new width during layout.

XmNspacing

This resource applies only when **XmNpacking** is **XmPACK_TIGHT** or **XmPACK_COLUMN**. It specifies the amount of vertical space between each child in a vertical orientation and the amount of horizontal space between each child in a horizontal orientation.

RowColumn also has several resources that can cause the RowColumn to change the internal layout of some classes of children:

XmNadjustMargin

This resource applies only to children that are subclasses of **XmLabel** and **XmLabelGadget**. When this resource is True and the orientation is vertical, RowColumn sets the **XmNmarginLeft** and **XmNmarginRight** for all children to the maximum values for those resources among all children. When this resource is True and the orientation is horizontal, RowColumn sets the **XmNmarginTop** and **XmNmarginBottom** for all children to the maximum values for those resources among all children. In PopupMenus and PulldownMenus this resource adjusts the margins only for button children, not for labels.

XmNentryAlignment

This resource applies only to children that are subclasses of **XmLabel** and **XmLabelGadget**. When **XmNisAligned** is True, RowColumn sets the **XmNalignment** of all children to the value specified by **XmNentryAlignment**. Following are the possible values:

XmALIGNMENT_BEGINNING

The child's text or pixmap is aligned with the left edge of the child's window.

XmALIGNMENT_CENTER

The child's text or pixmap is aligned with the center of the child's window.

XmALIGNMENT_END

The child's text or pixmap is aligned with the right edge of the child's window.

In menus, RowColumn sets the alignment only for button children, not for labels.

XmNentryVerticalAlignment

This resource applies only to children that are subclasses of **XmLabel**, **XmLabelGadget**, **XmText**, and **XmTextField**. It also applies only when **XmNpacking** is **XmPACK_COLUMN** (in either orientation) or when **XmNpacking** is **XmPACK_TIGHT** and the orientation is horizontal. The value specifies a reference point for aligning the children in any row:

XmALIGNMENT_BASELINE_BOTTOM

Causes the last baseline of each child in a row to align with the last baseline of the tallest child in the row. This value is applicable only when all children in a row contain textual data.

XmALIGNMENT_BASELINE_TOP

Causes the first baseline of each child in a row to align with the first baseline of the tallest child in the row. This value is applicable only when all children in a row contain textual data.

XmALIGNMENT_BOTTOM

Causes the bottom edge of the last line of text contained in each child to align with the bottom edge of the last line of text of the tallest child in the row.

XmALIGNMENT_CENTER

Causes the center of each child to align vertically with the center point established by the tallest child in the row.

XmALIGNMENT_TOP

Causes the top edge of the first line of text contained in each child to align with the top edge of the first line of text of the tallest child in the row.

XmNisAligned

When True, RowColumn sets the **XmNalignment** resources of children that are subclasses of **XmLabel** or **XmLabelGadget** to the value specified by **XmNentryAlignment**.

Managing Geometry Using BulletinBoard and DrawingArea

BulletinBoard and DrawingArea are two container widgets with similar geometry policies. These widgets have three geometry−related resources in common:

XmNmarginHeight

Specifies the amount of space between the top shadow of the widget and the top edge of any child, and between the bottom shadow of the widget and the bottom edge of any child. When the value of this resource is greater than 0, the widget ensures that the top edges of all children are below the widget's top margin.

XmNmarginWidth

Specifies the amount of space between the left shadow of the widget and the left edge of any child, and between the right shadow of the widget and the right edge of any child. When the value of this resource is greater than 0, the widget ensures that the left edges of all children are to the right of the widget's left margin.

XmNresizePolicy

Determines the widget's policy with regard to resize requests from its children. Following are the possible values:

XmRESIZE_NONE

The widget has a fixed size determined by its **XmNwidth** and **XmNheight**. The widget does not accept any geometry requests that would cause it to grow, but it may accept requests (without changing its own size) that would not cause it to grow. The widget also reports its current size as its own preferred size.

XmRESIZE_GROW

The widget can grow but not shrink. If its own parent approves, the widget accepts geometry requests that cause it to grow in order to enclose its children. It may accept requests (without changing its own size) that would not cause it to grow. When queried about its own preferred size, the widget calculates its layout and reports as its preference the greater of the calculated width and height and the current width and height.

XmRESIZE_ANY

The widget tries to accommodate geometry requests that would cause it to grow or shrink in order to enclose its children, requesting changes to its own size when necessary. When queried about its own preferred size, the widget calculates its layout and reports the calculated width and height as its preference.

In addition to these policies, BulletinBoard has geometry facilities that allow it to interact with subclasses in laying out complex collections of children. For example, SelectionBox has a List containing choices, a Text selection area, labels for the list and selection area, and three or four buttons. Usually the list appears above the selection area. The buttons appear equally spaced in a row below the selection area.

Additional children may be added to the SelectionBox after creation. The first child is used as a work area. The value of **XmNchildPlacement** determines if the work area is placed above or below the Text area, or above or below the List area. Additional children are laid out in the following manner:

MenuBar

The first MenuBar child is placed at the top of the window.

Buttons

All **XmArrowButton**, **XmDrawnButton**, **XmPushButton**, and **XmToggleButton** widgets or gadgets, and their subclasses are placed after the OK button in the order of their creation.

Others

The layout of additional children that are not in the above categories is undefined.

Managing Geometry Using Form

Form is a container widget that provides the most comprehensive facilities for controlling the layout of children. Constraints are placed on children of the Form to define attachments for each of the child's four sides. These attachments can be to the Form, to another child widget or gadget, to a relative position within the Form, or to the initial position of the child. The attachments determine the layout behavior of the Form when resizing occurs. Form is a subclass of BulletinBoard, so the resources and general geometry policies of BulletinBoard apply to Form as well.

Each child has 17 Form constraint resources, four for each side of the child and one, **XmNresizable**, that applies to the child as a whole. Following is a description of **XmNresizable** and the constraint resources that apply to the top side of a child:

XmNresizable

This Boolean resource specifies whether or not a child's request for a new size is (conditionally) granted by the Form. If this resource is set to True, the request is granted if possible. If this resource is set to False, the request is always refused.

If a child has both left and right attachments, its width is completely controlled by the Form, regardless of the value of the child's **XmNresizable** resource. Similarly, if a child has both top and bottom attachments, its height is completely controlled by the Form, regardless of the value of the child's **XmNresizable** resource. If a child has a left or right attachment but not both, the child's **XmNwidth** is used in setting its width if the value of the child's **XmNresizable** resource is True. These conditions are also true for top and bottom attachments, with height acting like width.

XmNtopAttachment

Specifies attachment of the top side of the child. It can have the following values:

XmATTACH_NONE

Do not attach the top side of the child. If **XmNbottomAttachment** is also **XmATTACH_NONE**, this value is ignored and the child is given a default top attachment.

XmATTACH_FORM

Attach the top side of the child to the top side of the Form.

XmATTACH_OPPOSITE_FORM

Attach the top side of the child to the bottom side of the Form. **XmNtopOffset** can be used to determine the visibility of the child.

XmATTACH_WIDGET

Attach the top side of the child to the bottom side of the widget or gadget specified in the **XmNtopWidget** resource. If **XmNtopWidget** is NULL, **XmATTACH_WIDGET** is replaced by **XmATTACH_FORM**, and the child is attached to the top side of the Form.

XmATTACH_OPPOSITE_WIDGET

Attach the top side of the child to the top side of the widget or gadget specified in the **XmNtopWidget** resource.

XmATTACH_POSITION

Attach the top side of the child to a position that is relative to the top side of the Form and in proportion to the height of the Form. This position is determined by the **XmNtopPosition** and **XmNfractionBase** resources.

XmATTACH_SELF

Attach the top side of the child to a position that is proportional to the current y value of the child divided by the height of the Form. This position is determined by the **XmNtopPosition** and **XmNfractionBase** resources. **XmNtopPosition** is set to a value proportional to the current y value of the child divided by the height of the Form.

XmNtopOffset

Specifies the constant offset between the top side of the child and the object to which it is attached. The relationship established remains, regardless of any resizing operations that occur.

XmNtopPosition

This resource is used to determine the position of the top side of the child when the child's **XmNtopAttachment** is set to **XmATTACH_POSITION**. In this case, the position of the top side of the child is relative to the top side of the Form and is a fraction of the height of the Form. This fraction is the value of the child's **XmNtopPosition** resource divided by the value of the Form's **XmNfractionBase**. For example, if the child's **XmNtopPosition** is 50, the Form's **XmNfractionBase** is 100, and the Form's height is 200, the position of the top side of the child is 100.

XmNtopWidget

Specifies the widget or gadget to which the top side of the child is attached. This resource is used if **XmNtopAttachment** is set to either **XmATTACH_WIDGET** or **XmATTACH_OPPOSITE_WIDGET**.

These constraint resources interact with the following resources of the Form itself:

XmNfractionBase

Specifies the denominator used in calculating the relative position of a child widget using **XmATTACH_POSITION** constraints. The value must not be 0.

If the value of a child's **XmNleftAttachment** (or **XmNrightAttachment**) is **XmATTACH_POSITION**, the position of the left (or right) side of the child is relative to the left side of the Form and is a fraction of the width of the Form. This fraction is the value of the child's **XmNleftPosition** (or **XmNrightPosition**) resource divided by the value of the Form's **XmNfractionBase**.

If the value of a child's **XmNtopAttachment** (or **XmNbottomAttachment**) is **XmATTACH_POSITION**, the position of the top (or bottom) side of the child is relative to the top side of the Form and is a fraction of the height of the Form. This fraction is the value of the child's **XmNtopPosition** (or **XmNbottomPosition**) resource divided by the value of the Form's **XmNfractionBase**.

XmNhorizontalSpacing

Specifies the offset for right and left attachments.

XmNrubberPositioning

Indicates the default near (left) and top attachments for a child of the Form.

Note: Whether this resource actually applies to the left or right side of the child and its attachment may depend on the value of the **XmNstringDirection** resource.)

The default left attachment is applied whenever initialization or **XtSetValues** leaves the child without either a left or right attachment. The default top attachment is applied whenever initialization or **XtSetValues** leaves the child without either a top or bottom attachment.

If this Boolean resource is set to False, **XmNleftAttachment** and **XmNtopAttachment** default to **XmATTACH_FORM**, **XmNleftOffset** defaults to the current x value of the left side of the child, and **XmNtopOffset** defaults to the current y value of the child. The effect is to position the child according to its absolute distance from the left or top side of the Form.

If this resource is set to True, **XmNleftAttachment** and **XmNtopAttachment** default to **XmATTACH_POSITION**, **XmNleftPosition** defaults to a value proportional to the current x value of the left side of the child divided by the width of the Form, and **XmNtopPosition** defaults to a value proportional to the current y value of the child divided by the height of the Form. The effect is to position the child relative to the left or top side of the Form and in proportion to the width or height of the Form.

XmNverticalSpacing

Specifies the offset for top and bottom attachments.

Following are some important considerations in using a Form:

- Every child must have an attachment on either the left or the right. If initialization or **XtSetValues** leaves a widget without such an attachment, the result depends upon the value of **XmNrubberPositioning**.
  If **XmNrubberPositioning** is False, the child is given an **XmNleftAttachment** of **XmATTACH_FORM** and an **XmNleftOffset** equal to its current x value.

If **XmNrubberPositioning** is True, the child is given an **XmNleftAttachment** of **XmATTACH_POSITION** and an **XmNleftPosition** proportional to the current x value divided by the width of the Form.

In either case, if the child has not been previously given an x value, its x value is taken to be 0, which places the child at the left side of the Form.

- If you want to create a child without any attachments, and then later (for example, after creating and managing it, but before realizing it) give it a right attachment using **XtSetValues**, you must set its **XmNleftAttachment** to **XmATTACH_NONE** at the same time.
- The **XmNresizable** resource controls only whether a geometry request by the child will be granted. It has no effect on whether the child's size can be changed because of changes in geometry of the Form or of other children.
- Every child has a preferred width, based on geometry requests it makes (whether they are granted or not).
- If a child has attachments on both the left and the right sides, its width is completely controlled by the Form. It can be shrunk below its preferred width or enlarged above it, if necessary, due to other constraints. In addition, the child's geometry requests to change its own width may be refused.
- If a child has attachments on only its left or right side, it will always be at its preferred width (if resizable, otherwise at is current width). This may cause it to be clipped by the Form or by other children.
- If a child's left (or right) attachment is set to **XmATTACH_SELF**, its corresponding left (or right) offset is forced to 0. The attachment is then changed to **XmATTACH_POSITION**, with a position that corresponds to the x value of the child's left (or right) edge. To fix the position of a side at a specific x value, use **XmATTACH_FORM** or **XmATTACH_OPPOSITE_FORM** with the x value as the left (or right) offset.
- Unmapping a child has no effect on the Form except that the child is not mapped.
- Unmanaging a child unmaps it. If no other child is attached to it, or if all children attached to it and all children recursively attached to them are also all unmanaged, all of those children are treated as if they did not exist in determining the size of the Form.
- When using **XtSetValues** to change the **XmNx** resource of a child, you must simultaneously set its left attachment to either **XmATTACH_SELF** or **XmATTACH_NONE**. Otherwise, the request is not granted. If **XmNresizable** is False, the request is granted only if the child's size can remain the same.
- A left (or right) attachment of **XmATTACH_WIDGET**, where **XmNleftWidget** (or **XmNrightWidget**) is NULL, acts like an attachment of **XmATTACH_FORM**.
- If an attachment is made to a widget that is not a child of the Form, but an ancestor of the widget is a child of the Form, the attachment is made to the ancestor.

All these considerations are true of top and bottom attachments as well, with top acting like left, bottom acting like right, y acting like x, and height acting like width.