Basic OLIT Concepts 1

What is OLIT?

OLIT is an X Window System-based widget set and library used to create applications supporting the *OPEN LOOK* graphical user interface. *Widgets* are user interface objects such as buttons, scrollbars, control areas, text edit areas, and drawing areas. By creating and manipulating desired widgets with the *X Window System Toolkit*, the programmer can create an application user interface. Application code, i.e., the code that performs the actual work on the application's data, is attached to the user interface via callback programs that are executed when the user performs some action on a widget, such as a mouse gesture or keyboard input.

The X Window System Toolkit, also know as the X Toolkit, Xt Intrinsics, or simply Intrinsics, is an X consortium standard library that provides the structure and functions for assembling widgets into a user interface. The X Toolkit provides a few basic widget classes that are usually supplemented with a full-featured widget set such as OLIT. OLIT applications follow an object-oriented, event-driven programming model. For further information about OPEN LOOK, refer to *Open Look Graphical User Interface Guidelines*. For more information on the X Toolkit refer to the *X Window System Toolkit.*

1

The X Window System

Figure 1-1 shows a block diagram of two OLIT programs running in the X Window System. The *X Server* is a program that runs on machines controlling one or more displays. The X Server handles output from an application (called the *client*) to the screen(s), and sends keyboard and mouse input to the appropriate client for processing.

This programming model, called the *client-server computing model*, provides a device independent layer, via the X Server, between applications and the display hardware. Any machine running an X Server, be it a workstation, mainframe, or X terminal, can display X applications regardless of the hardware. In addition to device independence, the distributed architecture of the X Window System allows clients to be run on any machine in a network, and be displayed on any other machine(s) in that network.

The X Server communicates with clients using the *X Window System Protocol* or simply, the X protocol. A C library interface, called *Xlib*, isolates the programmer from the intricacies and detail of the X protocol. Providing another level of abstraction and programming simplicity is the toolkit, which consists of the Xt Intrinsics and the widget set (OLIT). The toolkit provides an object-oriented, event-driven interface via widgets.

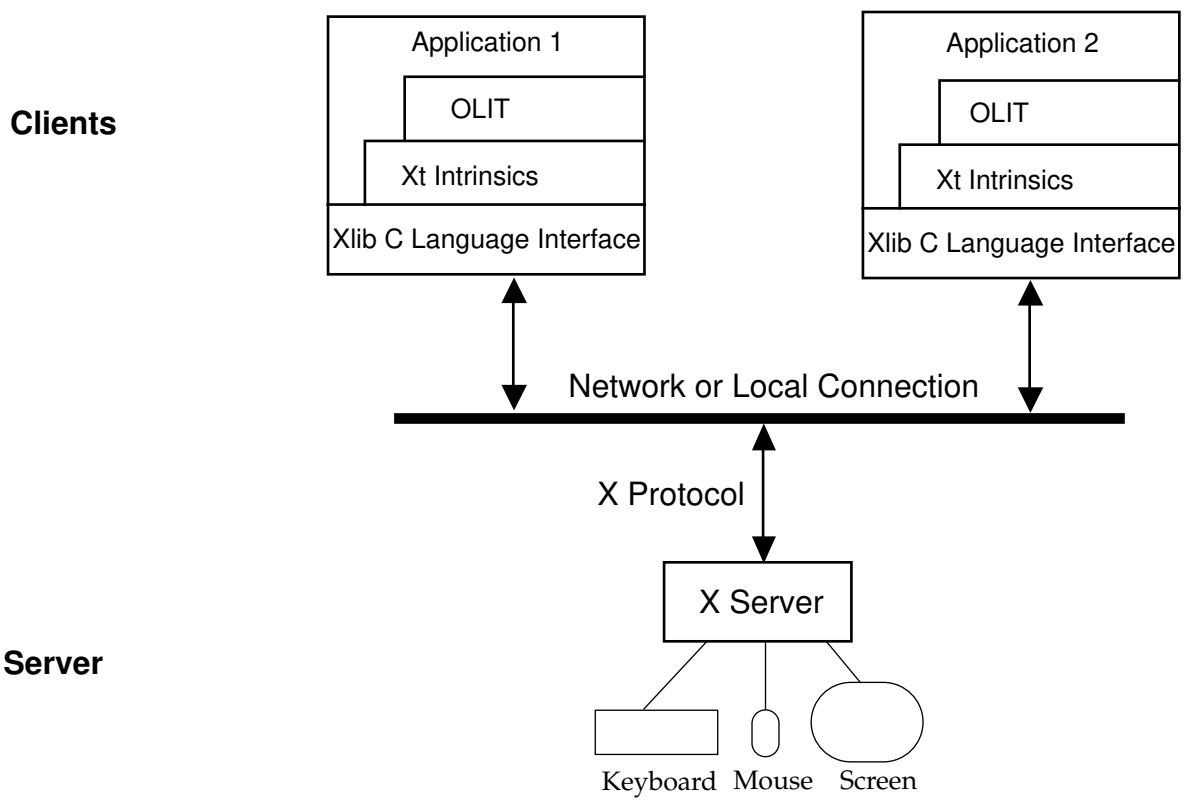

Figure 1-1 OLIT Applications in Client-Server Network

Widgets

Widgets, shown in Figure 1-2, are user interface objects. In the context of the X Window System, widgets are simply specialized X windows that can be created, manipulated, and destroyed. Programmatically, widgets are instantiated data structure types.

Widget types are organized by *class*. For example, there is the ScrollBar widget class, TextEdit widget class, OblongButton widget class, etc. When discussing a widget type, we refer to it as a *widget class*. When discussing a particular widget, we refer to it as a widget *instance*. If we want to create a widget of a particular class, we say we want to *instantiate* a widget.

1

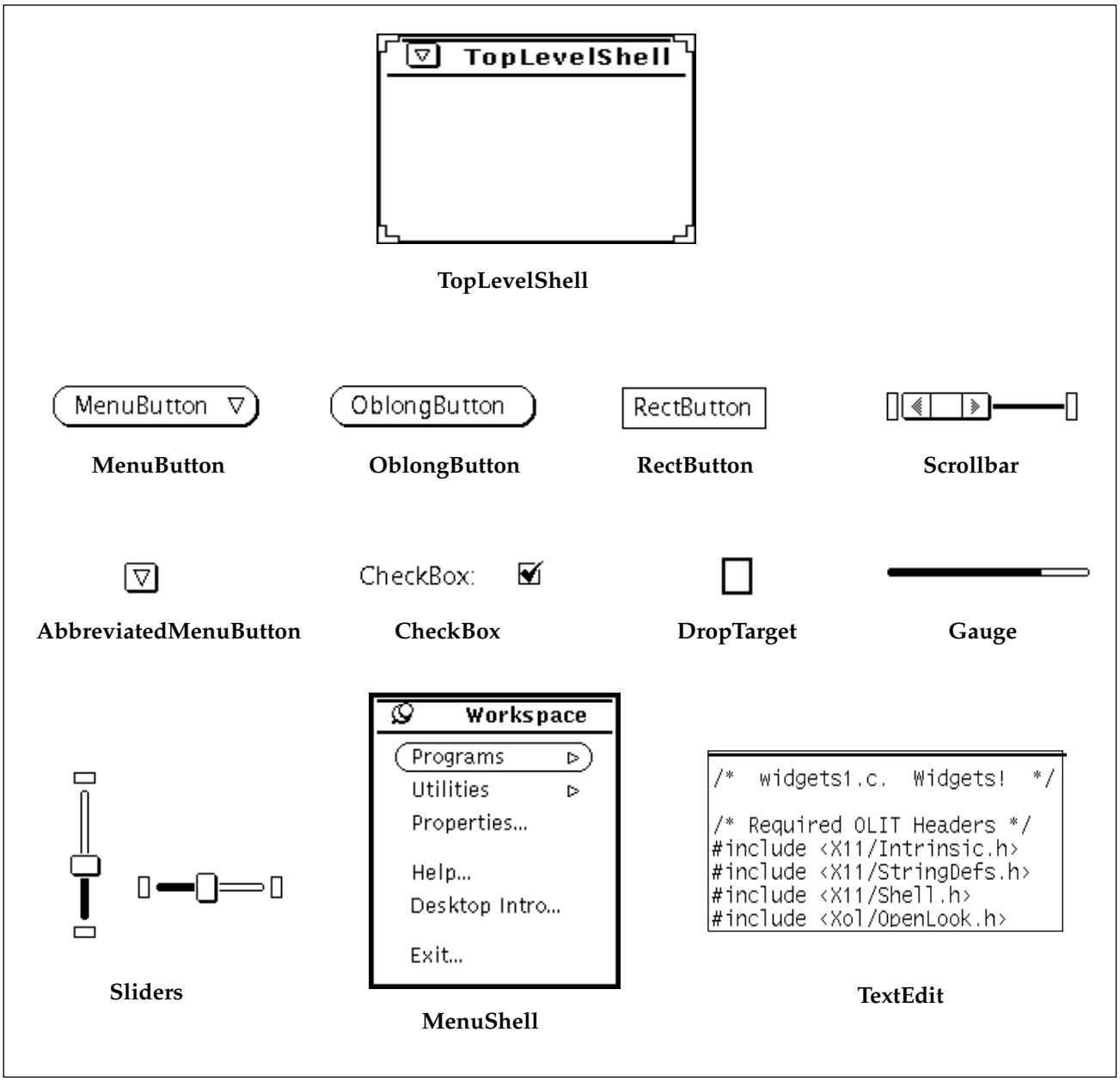

Figure 1-2 Widgets

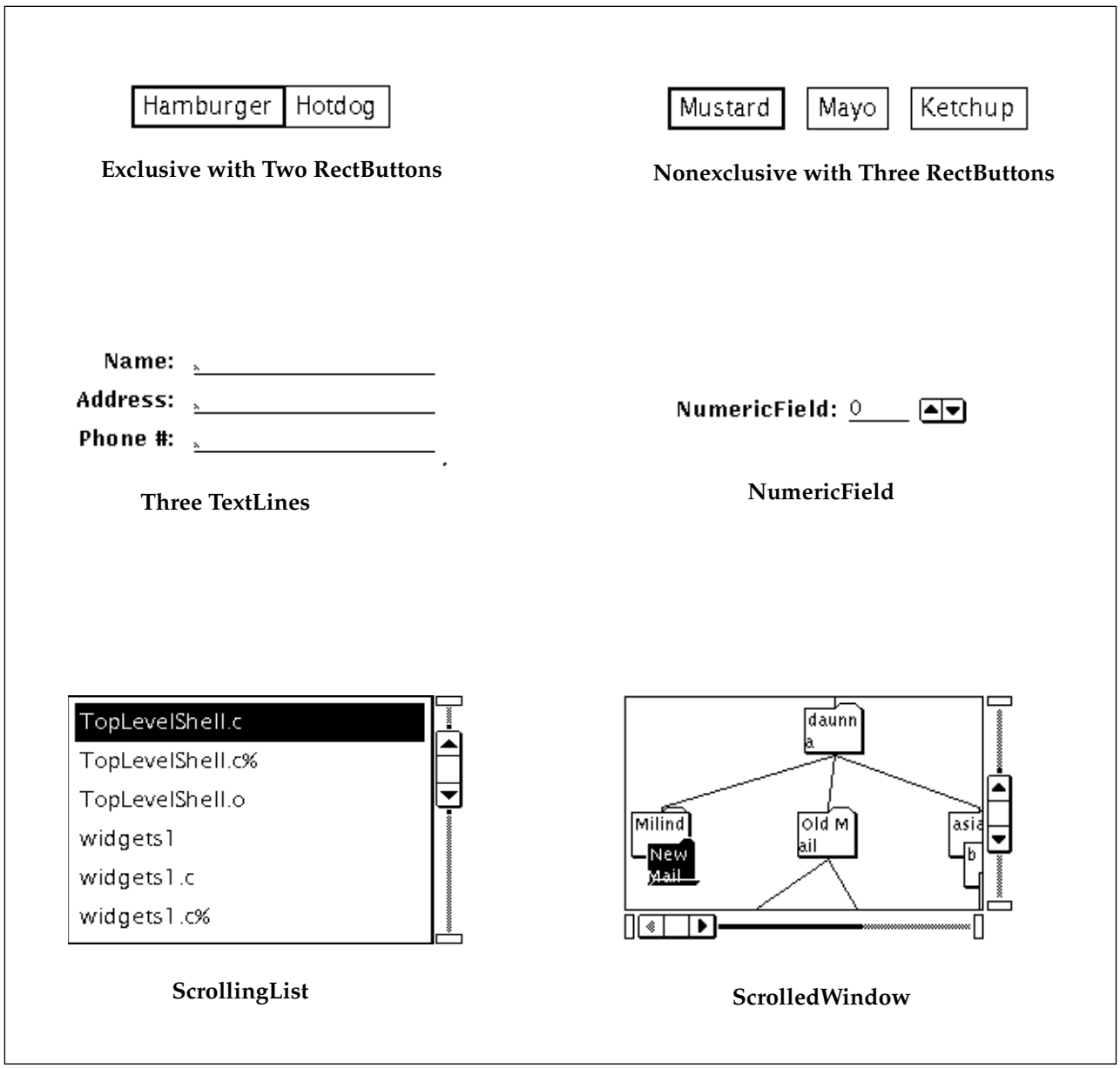

Figure 1-3 Widgets (continued)

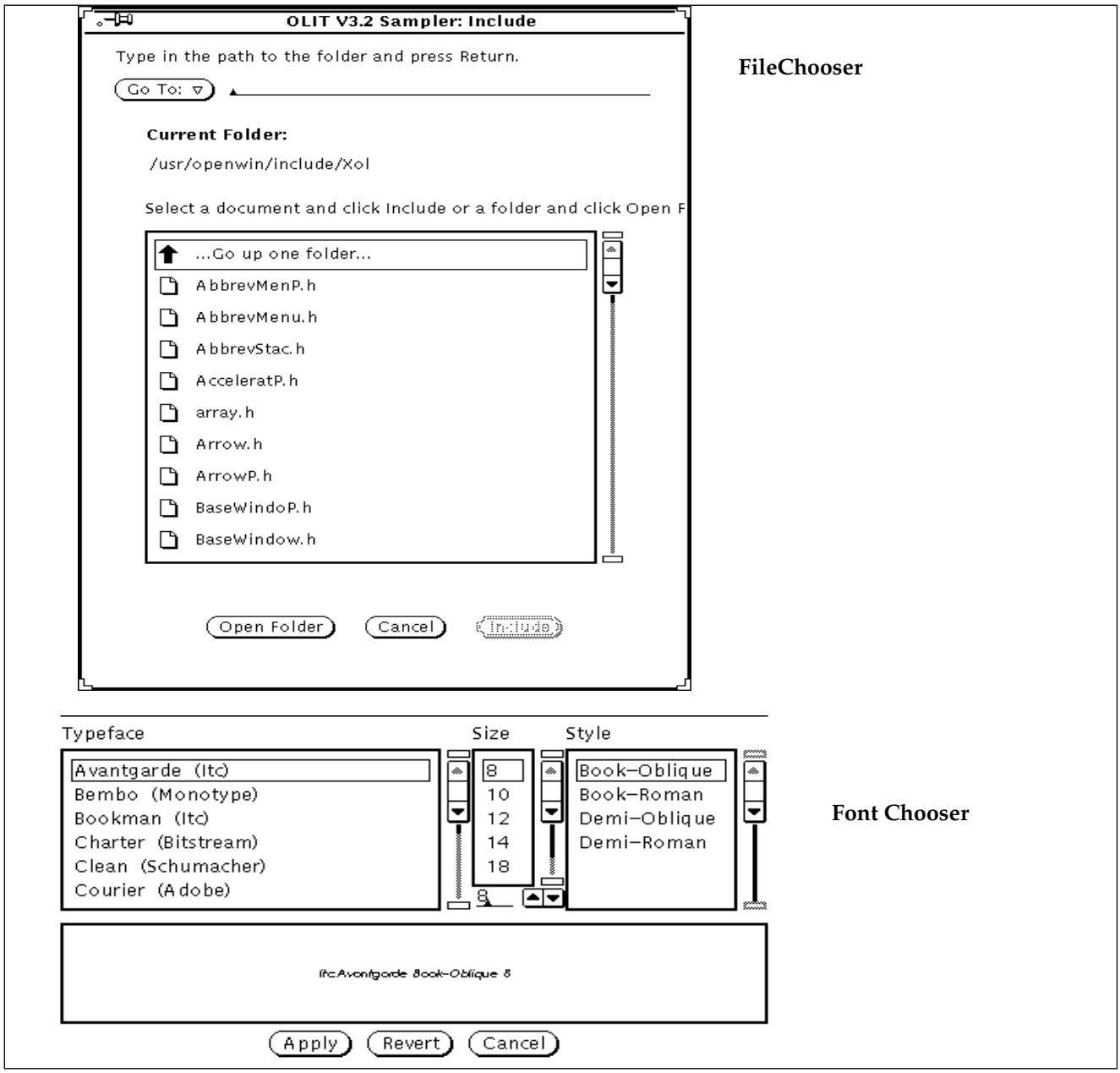

Figure 1-3 Widgets (continued)

1

John Pew's *X Window System: Programming and Applications with Xt/ OPEN LOOK Edition* categorizes widgets into five groups according to the type of functions they perform. *Action widgets* accept user or application input, and respond to the input with a programmatic action. *Text Control widgets* allow an application to receive and display text or numbers. *Popup widgets* are used to notify the user, bring up a pop-up menu, or prompt for input. *Container widgets* are specialized widgets that manage one or more child widgets. *Manager widgets* are used to combine all widgets into an entire application. A sixth category, called *Function Widgets*, perform specialize functions such as setting application fonts or importing/exporting files.

Table 1-1 OLIT Widgets

Widget Class	Characteristics and Usage
Action Widgets	
OblongButton	A button the user can "push" by pressing SELECT on it. When pushed, its border inverts, making it appear "pressed." Used to initiate application-defined actions.
RectButton	A toggle button that executes one set of actions when pressed and another when unpressed. When pressed, the button looks pressed. Button remains pressed until pressed again or unset programmatically.
CheckBox	A toggle that executes one set of actions when checked and another when unchecked. Similar to the RectButton widget.
MenuButton	An oblong button that creates a pop-up menu when user presses MENU on it.
AbbrevMenuButton	Provides a pulldown menu pane (same as the MenuButton widget) user presses MENU on it. Uses less screen space than MenuButton.
Slider	Lets user set a numeric value by sliding drag box along the bar.
Gauge	Graphically displays a read-only numeric value.
Scrollbar	Similar to the slider widget with additional features.
DropTarget	Creates a rectangle to be used as a drop site. Provides visual indication of the progress of a Drag and Drop operation.
Manager Widgets	
BulletinBoard	A composite widget that enforces no particular layout order on its children. Application must specify x- and y-coordinates of each child.
ControlArea	Widget that organizes its children controls in rows and columns.

1

Table 1-1 OLIT Widgets

Widget Class	Characteristics and Usage
DrawArea	Provides a window on which to render images using Xlib calls.
RubberTile	Organizes its children vertically or horizontally in a single file. Assigns relative weights to each child so that it expands or contracts a certain percentage of size changes of the RubberTile.
Form	Organizes children relative to the position of other children. Manipulates these layout when Form is resized, new children are added, moved, resized, unmanaged, remanaged, rearranged, or destroyed.
Text Widgets	
FooterPanel	Attaching a footer message to the bottom of a base window.
NumericField	Provides a one-line input field for alphanumeric text.
StaticText	Displays an uneditable block of text.
TextEdit	Provides a multi-line text editing window.
TextLine	One-line input field for text data. Replaces TextField.
Container Widgets	
Caption	Provides a convenient way to label a widget.
Exclusives	Manages a set of rectangular buttons to create a one-of-many button selection object
Nonexclusives	Manages a set of rectangular buttons or check boxes to create a several-of-many button selection object.
FlatCheckBox	Same functionality as a `NonExclusives` widget managing CheckBox widgets, but instead of creating individual CheckBoxes as children of a container widget, it creates sub-objects that have the same behavior as the CheckBoxes. Improves performance since fewer widgets are created.
FlatExclusives	Same functionality as an `Exclusives` widget managing RectButtons, as the same advantages as described in FlatCheckBox.
FlatNonexclusives	Provides the same functionality as a `Nonexclusives` widget managing RectButtons. Has the same advantages as described in FlatCheckBox.
ScrolledWindow	Provides a scrolled window view of a text or graphics pane.
ScrollingList	Provides a list of items the user can scroll through and select. Items can be exclusive or multiple choice non-exclusive. Users can edit items.

Table 1-1 OLIT Widgets

Widget Class	Characteristics and Usage
Popup Widgets	
NoticeShell	Creates a pop-up user notification message. Warns user of a problem; allows user to confirm a choice. Interaction with the application stops until the `NoticeShell` is popped down.
MenuShell	Creates a pop-up menu not associated with `MenuButton` or `AbbrevMenuButton`. Has the same features as a `MenuButton` widget (except those related to menu creation).
PopupWindowShell	Creates a pop-up property window allowing users to set properties of an application.

By creating and manipulating the desired widgets, the developer assembles the application's user interface. After the user interface is assembled, the application code, i.e., code that performs computations on the application's data, is attached to the desired widget as *callback procedures*, or simply *callbacks*. A callback procedure is a operation that gets executed when the widget receives some event, such as a mouse selection, a scrollbar being moved, or a text field being entered. Attaching specific procedures to specific widgets allows you to produce modular source code.

Widget Resources

A widget resource is a named piece of data that sets a specific widget attribute. Resources set attributes such as the color, font, layout, alignment, label, or callback list of a widget. A widget's resource values are usually set in a resource file for flexibility, but can also be set in the application code during widget creation or as a command line option. Widget resources are specified in

resource/value pairs with the resource name followed by the resource value (see Code Example 1-1). The OLIT Reference Manual lists and describes all the widget class resources.

Resource/Value Pair in Application Code

```
hello_button = XtVaCreateManagedWidget( "button1",
                    oblongButtonWidgetClass, control_area,
                    XtNlabel, "Hello!",
                    NULL);
```

Resource/Value Pair in a Resource File

```
hello.controlarea.button1.label: Hello!
hello.controlarea.OblongButton.background: aquamarine
```

Code Example 1-1 OLIT Resource/Value Pairs (in **bold**)

Widget Class Hierarchy

The X Toolkit organizes widgets by class. Class defines the characteristics, operations, and resources of a widget type. This includes the resources a widget class uses. All widget classes are *subclassed* from other widget classes. That is, a widget class is created by modifying and specializing another widget class called its *superclass*. The subclass inherits some or all of the characteristics of it's superclass. Class architecture and inheritance make it easier to create new widgets because subclasses use much of the same code and declarations as its superclass.

From an programmer's standpoint, the OLIT widget class hierarchy and architecture would seem of minimal interest unless you were going to create a new widget class by subclassing another widget. Understanding a little about the architecture, however, will help understand the relationship and shared characteristics of between all widgets.

Figure 1-3 shows the OLIT widget class hierarchy. Note that Xt Intrinsics defines a number of widget classes that are superclasses of all other widget classes. For example, all widget classes are subclassed from the Core widget class, therefore, all widgets inherit the Core widget resources.

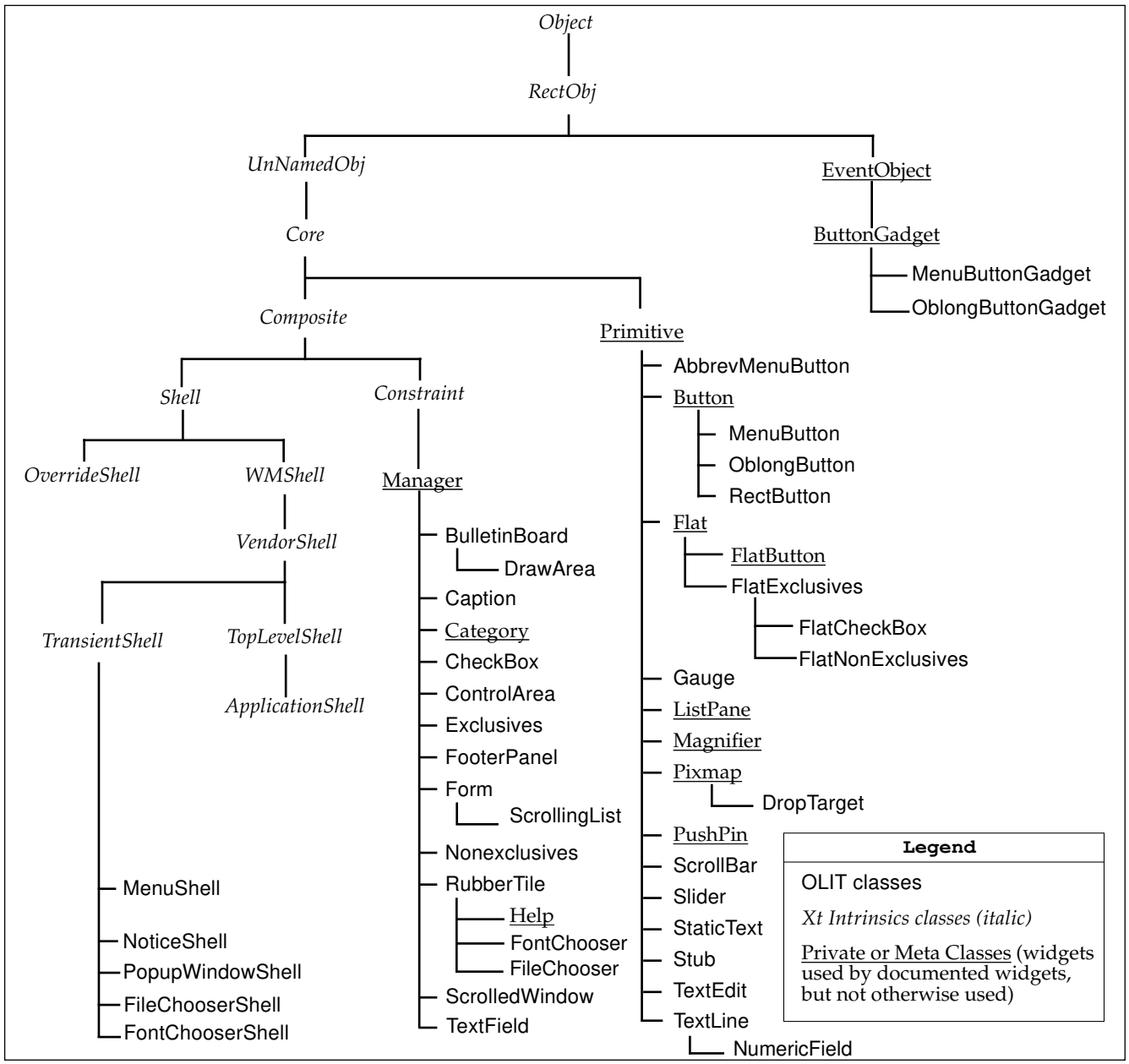

Figure 1-3 OLIT Class Hierarchy Tree

1