An Analysis of OpenMotif 2.2

March 4th, 2002

Contents

Introduction

This is a version of a brief report on Open Motif 2.2 released by ICS at the beginning of February 2002 for the Linux platform, as presented to OpenGroup. It was compiled after a detailed analysis of both the source code, and the Linux binary distribution.

The crucial findings of the analysis are presented in the Summary section, and this is followed by a more in-depth look at each widget or feature.

Explanation of Terms

Traits

A Trait is an encapsulation of generic behavior supported by a component of a given abstract type. It describes what the component does in logical terms, and exports the behavior in a standard form. The Trait includes methods for manipulating the exported behavior in a non-Widget Class specific manner. The Trait methods provide the means whereby a set of components with differing Widget Classes can be manipulated through a consistent and single interface.

For example, the XmQTaccessTextual Trait, supported by the Label, LabelGadget, Text, TextField, and any derived classes, provides a standard interface to the rest of the toolkit. Class-independent getter and setter methods are defined such that the text associated with any of the above components can be accessed without specific knowledge of the underlying classing. The same code works across the entire range of components supporting the Trait.

Render Tables

A Render Table is an encapsulation of abstract rendition information, independent of the class or content of the object to be displayed. It encapsulates not just font information, but also the general style of presentation, including the coloration and column layout aspects of the rendition process.

Render tables, and the Rendition objects of which they comprise, are fully reference-counted free standing pseudo-widgets supporting an external resource interface. As such, they allow the specification of generic style over and above any specific font specification one might like to make. Unlike the deprecated XmFontList, they also support attribute inheritance.

Tab Lists

A Tab List is a set of Tab objects, which define offsets across a widget or screen. Offsets may be at absolute or relative positions, and may be in a variety of measured units.

They are used as part of a Rendition object in order to specify multi-column aspects of the rendition process.

Uniform Transfer Model

In Motif 1.2, distinct codes were required in order to handle the various ways of transferring data from components to the application address space

- Clipboard
- Selections (Primary or otherwise)
- Cut and paste in general
- Drag and Drop

In Motif 2.1, this is rationalized, so that there is a single programming interface irrespective of the underlying cause of the data transfer.

Xme

In Motif 2.1, the Xme module defines a set of standard mechanisms shared by all components so that their look and feel is consistent. The Xme Drawing utilities are used in a standard way throughout, so that, for a trivial example, all Arrows in the system are rendered with the same interface, whether it be a standard ArrowButton itself, or a drawn artifact faked up by the ComboBox or SpinBox.

The Xme utilities also encapsulate other aspects of generic widget behavior, so that there is a standard method for configuring an object in geometric space, and so forth.

Summary

This release of Motif is found to be seriously flawed. It fails to maintain the high quality of code that has led Motif to be adopted in a large number of mission critical projects throughout many of the larger industrial, commercial and government organisations.

The key findings of the analysis are:

- **Open Motif 2.2 is not thread safe**
- **The new widgets do not conform to the Motif 2.1 Model**

  None of the widgets support Traits, Render Tables or Xme draw routines. They do not support the Uniform Transfer Model.

  In view of the fact that critical aspects of the system related to focus and navigation are implemented through the Trait system, Section 508 Accessibility is made considerably more difficult to implement.

- **Widgets duplicate functionality already present in Motif 2.1**

  For nearly all the widgets, the equivalent functionality is already provided by Motif 2.1 widgets.

- **Wide use of deprecated and obsolete features**

  For instance the FontSelector is for XmFontList which is deprecated in Motif 2.1.

- **Backwards compatibility compromised.**

  Motif 2.1 applications will not dynamically run with an Open Motif 2.2 library. Binary incompatibility has been introduced by such additions as the VendorShell extension for ToolTips.

- **Suppression of Low level incompatibility messages**

  The XtVersionDontCheck flag is widely used to suppress toolkit compatibility checking.

- **Copyright notices to ICS and others throughout code**
- **The introduction of reliance on proprietary extensions in various forms**
- **Light on bug fixing**

ButtonBox

Description

A Constraint Manager, providing proportional layout of child (Button) objects in horizontal or vertical layout

General Observations

This widget provides very little in the way of additional functionality over and above the Motif 2.1 toolkit.

In the general case, the layout can be achieved through standard Motif techniques

- Dialog template (horizontal layout).
- Form with suitable positioning and fraction base (either orientation).
- Container configured for Balanced Spatial Grid layout. This also supports flow balancing which the ButtonBox provides.

Whilst it is true that the component has one edge over the first two of the alternatives listed above, namely built-in dynamic modification of the layout in vertical or horizontal orientation, in the general case this is unlikely to be of use since ButtonBoxes tend to be statically preconfigured one way or the other. This is not true by comparison with the Container, whose layout is dynamic and flexible. As a layout manager, the component is therefore technically redundant, particularly as the Container is the preferred Motif 2.1 application interface, interfacing as it does with the Uniform Transfer Model, which this component does not.

Scope

Does not support Gadgets.

Thread Safety

No attention to multi-thread safety has been observed whatsoever. Since the sources access widget class structures throughout, this widget is particularly unsafe.

Motif 2.1 Compatibility

No consideration has been given to Motif 2.1 concepts such as the Trait or Xme sub-systems.

The sources use deprecated methods throughout, in particular there is no respect for the Xme utilities, which deprecates all of the rendering and resolution independent code found in these sources For example, the synthetic resource converters use _XmToVerticalPixels, where the Motif 2.1 routine is correctly XmeToVerticalPixels.

It might be reasonable to expect that consideration should be given to supporting the Trait XmQTtakesDefault. Otherwise, default button highlight emphasis is not supported in the standard manner.

No consideration has been given as to whether the orientation algorithms here should not also be sensitive to Motif 2.1 layout direction.

The visuals which the component provides will be non-standard. By comparison, the XmBulletinBoard explicitly recalculates shadow considerations when child manage/ insertion state is modified. No such algorithms exist for this component.

Construction

The widget has not been embedded into the Motif toolkit in general in the standard manner. For example, installing XmRepType handlers through the ClassInitialize method, although reasonable for stand-alone additional components, is not the way that the integrated Motif components are constructed. In addition, the header files contain type definitions which in standard usage would be embedded into the Xm.h public header.

No consideration has been given to the resolution independence of synthetic constraint resources.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

This widget is not Motif 2.1 compatible, is severely deprecated and unsafe, and has no place in the standard toolkit. Even if technically reworked for compatibility and thread-safety, is of limited additional benefit because of the range of alternative techniques which have become the standard way of achieving the effect.

ColorSelector

Description

A composite component which facilitates the selection of a color from either the X color database namespace, or via direct slider control.

General Observations

In principle, a color selection tool is a useful addition to any toolkit.

- Java Swing implements the JColorChooser class
- MFC implements the CColorDialog class

Scope

The functionality which the component offers is crude.

- There is no support for Color Models of any kind

  HSL, HSB, CMY, YIQ, HSV, HLS?

- The bare minimum which one would expect would be color, saturation, brightness controls.
- In this day and age, a graphical representation of cubic color space is not an unreasonable assumption. Both the Java and MFC offerings support this.

Thread Safety

No attention to multi-thread safety has been observed whatsoever. Since the sources access widget class structures, this widget is potentially unsafe.

Motif 2.1 Compatibility

There is no consideration given to the Motif 2.1 Xme standard interface at any level of detail.

- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.
- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.

No consideration has been given to the Motif 2.1 Trait mechanisms. It is not clear whether consideration of XmQTaccessColors could be indicated.

It uses deprecated XmStringGetLtoR utilities to fetch the selected item from the Color namespace List, where it should be using XmStringUnparse. It uses XmStringConcat/ XmStringFree combinations where the XmStringConcatAndFree is to be preferred.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, installing XmRepType handlers through the ClassInitialize method, although reasonable for stand-alone additional components, is not the way that the integrated Motif components are constructed.

It is not clear that the differences between decomposed/composed read/read-write Colormaps has been completely considered. The component allocates colors only on a single plane of the Colormap; the re-entrant PrivateColormaps method from the slider callbacks has the potential to iteratively allocate cells in the Colormap, but only one cell is finally deallocated in the Destroy method of the widget.

Conclusion

In principle, there is no philosophical objection to a Color Selection component being added to the standard toolkit.

In practice, it should not be this one; the component is thread-unsafe, deprecated, insufficiently capable, and not Motif 2.1 compatible.

Column

Description

A Constraint Manager which aligns and labels children in a two-column arrangement

General Observations

The Manager dynamically constructs the label on behalf of the child, but hides the result from the caller. Whilst in itself not necessarily an error, it does restrict the functionality of the component; public access would have made the whole rather more useful, particularly where in a typical situation the label associated with the child needs to be dynamically configured.

The layout it provides is however redundant, since a multi-column arrangement of labelled components can be achieved through a standard Motif Form. The Form is considerably more flexible in this respect in any case. Whereas in Motif 1.2 expertise with the general programming public with the Form was limited, and it made sense to provide an easier route to the layout, this is no longer the case: the public is more sophisticated, and the shorthand which the component provides is of rapidly decreasing usefulness.

A convenience only; it does not add to the range of logical layouts available through standard Managers in the toolkit. Reasonable for Motif 1.2, but now redundant.

Scope

Only appropriate for static labelling situations - the constructed Label is marked as private to the implementation.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The sources use the deprecated XmFontList throughout - no consideration has been given to the Motif 2.1 RenderTable type.

No consideration has been given to support for the Motif 2.1 Trait mechanisms.

No consideration has been given to Motif 2.1 layout direction in the algorithms.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, installing XmRepType handlers through the ClassInitialize method, although reasonable for stand-alone additional components, is not the way that the integrated Motif components are constructed. In addition, the header files contain type definitions which in standard usage would be embedded into the Xm.h public header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

This widget is not Motif 2.1 compatible, is deprecated, and thread unsafe, and has no place in the standard toolkit. Even if technically reworked for compatibility and thread-safety, the widget remains only a shorthand convenience for a layout which can be achieved rather more flexibly by standard Motif components.

ComboBox2

Description

A TextField with an associated drop-down list of validset selections.

General Observations

Redundant in the Motif 2.1 toolkit: the standard ComboBox more than suffices for the task to hand.

Scope

The component is String based in functional interface, not XmString based, which will have side effects for anyone attempting to use the component in an internationalized environment.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The popup shell which the component utilizes for internal operation is not Motif 2.1 compatible. While creating a suitably configured TopLevelShell would be acceptable in a Motif 1.2 toolkit, the preferred interface is now the Motif 2.1 GrabShell.

There is no consideration given to the Xme standard interface at any level of detail.

- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.
- It uses deprecated _XmDraw methods for rendition. This will give an inconsistent look to the component compared to the standard toolkit.
- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.

No consideration has been given to the Motif 2.1 Trait mechanisms. It is reasonable to expect that a component of this type should support XmQTaccessTextual at a minimum. The standard Motif toolkit will therefore not recognise the functionality which the component provides.

The XmString handling performed on behalf of the constituent List is not standard. There are no XmStringGenerate/XmStringParse calls. The routine _XmGetMBStringFromXmString() is an additional procedure, foreign to the standard toolkit, which replicates in a non-standard way the core standard technology.

Construction

The component relies on intentionally undocumented features, shared by other components in the proposed canon, for its operation.

Although the component copies down resources on behalf of constituent dynamically-created children, there is no corresponding specification of these in the top level resource table. There is no attempt to implement a properly constructed resource *mirror* - it replicates only those child resources which require an override for the proper operation of the component.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

Redundant, deprecated, non-internationalized, not Motif 2.1 compatible, and unsafe.

DataField

Description

A Text component offering programmatic field validation services, with bespoke imaged input method handling

General Observations

The component is from an abortive Hewlett-Packard Motif 1.2.4 offering which never made the standard Motif toolkit.

The release contains no manual pages for the component, and was thus difficult to evaluate and interpret. It would be difficult for the general programmer to use.

Scope

The standard methods for data field validation involve adding modify verify and value changed callbacks. Here these are added on behalf of the programmer, and validation is redirected to a bespoke validate callback. The programmer, however, still has to supply the validation code, but in the new form. This therefore gains the programmer nothing: as it stands, the component validates nothing, and is logically no more useful than the standard Text components in this respect.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The component uses non-standard methods for image handling: _XmInstallPixmap() is not just deprecated, but strictly obsolete.

No consideration has being given to any Motif 2.1 utilities.

- deprecated XmFontList handling throughout
- deprecated XmString handling throughout
- deprecated drop-site handling which does not use the XmTransfer Uniform Transfer Model functionality

No consideration has been given to the Motif 2.1 Trait mechanisms. It is reasonable to expect that a component of this type should support XmQTaccessTextual at a minimum. The standard Motif toolkit will therefore not recognise the functionality which the component provides, particularly as in the construction the normal super class inheritance chain has been subverted (see below).

Construction

Within the Xt class record, the object inherits from the Primitive class, but declares its size as the sum of TextField and DataField records. This looks highly dubious at best, unstable at worst. Consider the implications in a multi-abi environment where there are differences in structure byte alignment between compilation of the application base and the toolkit. This does not look like Xt-compatible construction. In addition, mangling the inheritance in this manner will short-circuit any Traits which might be expected to be inherited from a proper XmTextField superclassing.

Messages which should be defined in the central XmString repository are defined locally to the sources.

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the component header files contain callback type definitions which in standard usage would be embedded into the Xm.h public header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

The component re-implements input methodology directly, which should have been inherited from the TextField superclass. The input methodology which it provides is likely to be non-standard, and non-portable. In addition, since the widget re-implements the translations for input interaction, the user interaction with the input methodology is also rendered non-standard.

Conclusion

It appears that this component formed the basis of an abortive attempt at internationalization for input methodology, made redundant by standard techniques added to Motif at about the same time.

Not only redundant in terms of the field validation, and non-standard in terms of input methodology and interaction, this component is not Motif 2.1 compatible, and has a highly unstable construction.

Ext18List

Description

A multi-column multi-color List selection component, with embedded search functionality.

General Observations

Multi-column, multi color Lists are fully available in the standard Motif 2.1 toolkit; the XmRendition system provides the coloration, the XmTab sub-system gives the column layout. The search functionality is specious when the Motif 2.1 List allows quick navigation from the keyboard.

Scope

The user interaction is limited to single selection. There is no browse/extended/multiple selection implementation, and thus the components range of usefulness is considerably less generic than the standard offering.

Thread Safety

The component does not reference through widget class members, contain static data, or use routines known to be thread-unsafe. However, it relies on the I18List for implementation, and this subsidiary component is thread unsafe, thereby rendering this component as thread-unsafe in turn.

Motif 2.1 Compatibility

The component uses deprecated XmString utilities for the construction of the Find object label.

There is no consideration given to the Xme standard interface at any level of detail.

- It uses _XmWarning where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component. In addition, _XmWarning is strictly obsolete.

No consideration has been given to Motif 2.1 Trait system. One would reasonably expect a component of this type to support the XmQTtransfer Trait, either directly or indirectly in the sub-components. This trait interfaces to the Uniform Transfer Model, which is therefore unsupported.

The side effect is that the rest of the Motif toolkit will not recognise the functionality which this component provides. One might also reasonably expect support for the Trait XmQTscrollFrame somewhere in the hierarchy. Nor will the component participate in consistent data transfer.

Construction

The widget has not been embedded into the Motif toolkit in general in the standard manner. For example, the component header files contain callback and enumerated type definitions which in standard usage would be embedded into the Xm.h public header. The

header declarations lack consistent storage specification: extern is not specified for all methods listed.

The public interface is String, and not XmString-based, which has implications for internationalized environments. In the same vein, the default labelling on the constituent parts of the whole is hard-coded English, and not dynamically constructed from the component name, thus renders the whole non-internationalized.

Although the component copies down sub-part resources to constituent children, there is no consideration given to the construction of a fully working resource mirror. This would matter where an application dynamically queries the widget class in order to deduce the public interface to the component.

No consideration has been given to synthetic resolution independent conversion of the width/height resources of the component.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

This component is not Motif 2.1 compatible, not internationalized, is thread unsafe, and redundant.

Font Selector

Description

A dialog component for selecting and choosing an XLFD font name.

General Observations

In general, a component which assists in the construction of complex data types such as the XmRenderTable is a more than worthwhile proposition. Both Java and MFC support such components in their underlying toolkits.

Specifically in Motif 2.1, the rendition process involves considerably more than just the name of a single XLFD font. As part of a larger rendition editor, the component could see service, provided that objections to the limited scope of the functionality can be overcome, particularly with respect to internationalized contexts.

Scope

The dialog does not support the notion of multi-font, or X Font Set selection. The search criteria which it offers does not allow selection from all parts of the XLFD specification.

The sources hard code the range of valid font sizes. A platform containing, for example, a 33 pixel font would be marked as invalid by the component.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The XmFontList is deprecated in Motif 2.1.

There is no consideration given to the Xme standard interface at any level of detail.

- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, header declarations lack consistent storage specification: extern is not specified for all methods listed.

The routine _XmGetMBStringFromXmString() is an additional procedure, foreign to the standard toolkit, which replicates in a non-standard way the core standard technology.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

Since there is no attempt to construct multi-font or font set objects, no attempt has been made to ensure that selections satisfy the requirements of the locale. The dialog therefore is of no assistance in internationalized applications.

If Motif 1.2 formed the basis of the underlying toolkit, the component might be worthy of reconsideration, provided that support for multi-font environments is implemented.

For Motif 2.1, the component is strictly deprecated, unsafe, and incompatible.

I18List

Description

A composite component which supports List selection accompanied by simultaneous textual input and font specification, arranged in a bespoke Paned hierarchy. Parts of the whole structure are configurable on request. The component underpins the Ext18List.

General Observations

The Font aspects are redundant in an environment which supports standard internationalized input methods. Furthermore, no consideration has been given to internationalized Font Sets, or to satisfy the requirements of the locale in general.

It should not be incumbent on general users that they should have to construct the font piecemeal by hand; in the general case, what is explicit here should be automatic and pre-configured.

No supporting documentation for the component can be found in the supplied materials.

Scope

The component has very limited application given the specific requirements for internationalized applications.

The List aspects of the structure supports single, but not multiple or extended selection, which therefore is considerably less capable than the standard Motif offering, particularly given that the requirements of internationalization require the construction of a multi-font font set.

Nowhere in the sources is the X Font Set taken into consideration.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

There is no consideration given to the Xme standard interface at any level of detail.

- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.
- It uses deprecated drawing methods to render itself. This will give an inconsistent look to the component.
- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.

There is no consideration given to the XmTrait system. A reasonable assumption would be that the component supports XmQTtransfer.

Not only does the component not support the Uniform Transfer Model, but no consideration for inter-component data transfer has been given: the Motif 1.2 Drag and Drop functionality is also absent.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the component header files contain callback and enumerated type definitions which in standard usage would be embedded into the Xm.h public header. The header declarations lack consistent storage specification: extern is not specified for all methods listed.

The routine _XmGetMBStringFromXmString() is an additional procedure, foreign to the standard toolkit, which replicates in a non-standard way the core standard technology.

The composite hierarchy relies on components which are strictly redundant and foreign to the standard toolkit.

The List translation and action specifications are not consistent with the Motif toolkit.

The component is a strictly private implementation to the Ext18List - there is no separate header specification for the component.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

The component is unsafe, deprecated throughout, Motif 2.1 incompatible, and not consistently internationalized, irrespective of the Font aspects of the hierarchy. Those aspects which it provides do not satisfy the general requirements.

IconBox

Description

A Constraint Manager which lays out children in a balanced grid.

General Observations

The component provides no functionality not available in the standard toolkit. The Motif 2.1 Container and IconGadget combination, configured for Spatial/Balanced Grid Layout is explicitly designed for the purpose.

Scope

The component does not support Gadgets.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

There is no consideration given to the Xme standard interface at any level of detail.

- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.
- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.

The component does not consider data transfer in any aspects. Consideration should be given as to whether a component of this type should support the XmQTtransfer and XmQTtraversalControl Traits. The standard Container/IconGadget combination is therefore functionally superior, given the support for the Uniform Transfer Model. Clearly, the presence of the XmIconBoxDropData structure type in the header file would leave one to believe that consideration has been given; however, the type is unreferenced and the functionality unimplemented.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the component header files contain data type definitions which in standard usage would be embedded into the Xm.h public header. The header declarations lack consistent storage specification: extern is not specified for all methods listed.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

The component is deprecated, unsafe, incompatible, and redundant.

IconButton

Description

A component which displays an image and label simultaneously.

General Observations

The Motif 2.1 IconGadget is designed for this purpose, and therefore the component is strictly redundant.

Scope

The class contains resource definitions which are specific to a proprietary interface builder.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, and since the component defines unprotected static cache data, this is likely to be highly unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The component implements an internal Pixmap cache outside the auspices of the standard cache mechanisms.

There is no consideration given to the Xme standard interface at any level of detail.

- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.
- It uses deprecated methods to render itself and associated shadows.
- It uses deprecated methods to deduce the default font.

The component is not derived from the standard PushButton component; it installs bespoke translations and actions which will give the component an inconsistent feel to standard Button components.

No consideration has been given to the Motif 2.1 Trait mechanisms. One might expect a component of this type to support the XmQTactivatable and XmQTcareParentVisual Traits.

No consideration has been given to interfacing with the data transfer mechanisms of the toolkit, whether deprecated Drag and Drop or otherwise.

The component does not support Motif 2.1 render table or layout direction functionality.

The string direction resource which it relies on for layout is deprecated in the Motif 2.1 toolkit.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the component header files contain data type and callback definitions which in standard usage would be embedded into the Xm.h public header. The header declarations lack consistent storage specification: extern is not specified for all methods listed.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

The component is unsafe, incompatible, and strictly redundant.

Paned

Description

A Constraint Manager which provides a horizontal and vertical adjustable viewport for its children.

General Observations

In Motif 1.2, the standard PanedWindow did not officially support orientation of the viewport. In this context, the component was potentially a useful addition to the toolkit.

In Motif 2.1, the standard PanedWindow fully supports variable orientation; this component is therefore now strictly redundant.

The component is not documented in the supplied manual pages.

The component, originating from the Athena widget set, is strictly Copyright MIT X Consortium, Copyright Digital Equipment Corporation. It now appears distributed with Copyright 1992 Integrated Computer Solutions, Inc.

The original Athena notice reads as follows:

> Copyright (c) 1987, 1988, 1994 X Consortium
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> *The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

This notification with respect of the X Consortium’s rights is entirely missing from the supplied sources, and therefore constitutes a serious material breach of the original copyright.

Scope

The component places an upper bound on the number of children which it is willing to support. This is more restrictive than the Motif 2.1 component.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

There is no consideration given to the Xme standard interface at any level of detail.

- It uses deprecated methods to perform resolution independent conversion of synthetic resources. XmeToVerticalPixels/XmeToHorizontalPixels is the correct technique.
- It uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.

The component has faulty code which attempts to mimic the Motif specification.

- It hard codes a specific and constant navigation type, where the Motif toolkit only overrides in the absence of an explicit setting.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the component header files contain data type and callback definitions which in standard usage would be embedded into the Xm.h public header. The header declarations lack consistent storage specification: extern is not specified for all methods listed.

The InsertChild method ***aborts*** the application where the current child exceeds the number of children it is willing to support. This is not acceptable in a production quality toolkit supporting mission-critical domains.

Conclusion

The component is unsafe, incompatible, and redundant.

TabBox, TabCanvas

Description

Private components, in support of the TabStack.

General Observations

The TabManager functionality is redundant in the face of a suitably configured standard Notebook component.

Scope

The functionality is private and specific to a particular manager.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

There is no consideration given to the Xme standard interface at any level of detail.

- It uses idiosyncratic methods to render itself and associated shadows, and not the standard Xme drawing utilities.
- It uses deprecated methods to deduce the default font.

The component orientation and general layout is not sensitive to the Motif 2.1 layout direction methodology.

The component supports only the deprecated XmFontList, and not the Motif 2.1 XmRenderTable.

Construction

The widget has not been embedded into the Motif toolkit in general in the standard manner. For example, the representation types are installed in the ClassInitialize method, and not through the central XmRepType manager. Callback and enumerated types are defined in the header files, where the correct place for private toolkit implementation would be the XmP.h header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusions

The functionality which the components provide on behalf of a redundant manager is also redundant, incompatible, unsafe, and deprecated.

Table

Description

A multi-columned Table component

General Observations

In principle, a worthy idea for consideration: it offers functionality not present in the standard toolkit.

In practice, this may not be politically the best idea at this stage of the toolkit:

- There are too many commercial concerns who have a proprietary interest in a component of this type.

  XRT/Table

  INT/EditTable

- The commercial offerings are fully functioning, robust, and well respected.
- There are respectable, well-known freeware components of excellent pedigree which are logically rich in functionality, and which are Motif-compatible.

  Xbae/Matrix

Technically, the component would have to be spectacularly good by comparison, or should not be considered.

The release contains no documentation for the component.

Scope

The component requires a distinct cell editor child per column of the Table - the calculations are all based on this assumption.

The component is therefore heavyweight, and will not scale.

There is no complete support for either MVC through a TableModel, nor for standard concepts such as frozen rows/columns, although some attempt has been made to extract the problem through cell render procedures. The functionality it provides is therefore considerably thin by comparison with the available compatible components.

There are built-in assumptions concerning what constitutes a valid cell editor: Text, TextField, the proposed extension ComboBox2.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

There is no consideration given to the Xme standard interface at any level of detail.

- It uses deprecated methods to render itself and associated shadows, and not the standard Xme drawing utilities.

There is no consideration given to the Motif 2.1 Trait system. One might expect a component of this type to support both XmQTtraversalControl, and XmQTtransfer Traits.

The component supports the deprecated Motif 1.2 XmFontList, and not the Motif 2.1 XmRenderTable.

The component by default supports the proposed ComboBox2 as a cell editor, but not the standard Motif 2.1 ComboBox. Embedding specific class knowledge bypasses the whole point of the Trait system - one would reasonably expect a Motif 2.1 compatible component to use the XmQTaccessTextual Trait for this purpose, thereby immediately supporting any derivative classes supplied from the component manufacturers.

The component relies on an internal bespoke clipping object, whereas there is a standard XmClipWindow object which could be adopted for the purpose. Where this standard offering does not provide the functionality required, the derivative component should at least support the XmQTclipWindow Trait.

Construction

The widget has not been embedded into the Motif toolkit in general in the standard manner. For example, the representation types are installed in the ClassInitialize method, and not through the central XmRepType manager. Callback and enumerated types are defined in the header files, where the correct place for private toolkit implementation would be the XmP.h header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

Although the component provides functionality not present in the toolkit, there are serious issues with the implementation which necessitate significant rework before the component as it stands should be considered for inclusion.

In particular, it remains thread unsafe and Motif 2.1 incompatible.

TabStack

Description

A Constraint Manager which manages overlapping children such that only one logical page (child) is visible at any given time.

General Observations

Relies on the TabBox/TabCanvas for operation.

Redundant. The Motif 2.1 Notebook is designed for precisely this purpose. Although the default Notebook configuration is over-decorative, one might say florid, in appearance, it is generically configurable so that a traditional TabManager appearance and orientation can be achieved using the standard component.

Scope

Does not support XmQTnavigator in any way, so alternative SpinBox page control is absent from the combination, unlike the Motif 2.1 Notebook.

Thread Safety

No consideration has been given to the thread safety of this component. Since this component references through widget class members, this is likely to be unstable in a multi-threaded environment.

Motif 2.1 Compatibility

The component implements bespoke Tab string direction, which will conflict with Motif 2.1 layout direction.

The components use deprecated XmFontList resources, and not Motif 2.1 render table support. In addition, the components use deprecated methods to deduce the default font.

The components do not use XmeDraw utilities, thereby giving an inconsistent look to the rest of the standard toolkit.

The component uses XtAppWarningMsg where the standard interface is XmeWarning. This will given an inconsistent feel to the messaging emanating from the component.

The components use deprecated methods for performing synthetic resolution independent resource conversion.

No consideration of Traits has been implemented in the component. One might reasonably expect that a combination of this type gave consideration to whether XmQTscrollFrame, XmQTtraversalControl, XmQTnavigator are appropriate.

The components use the Motif 1.2 XmDropSite system, and not the replacement Motif 2.1 Uniform Transfer Model for data transfer.

Construction

The widgets have not been embedded into the Motif toolkit in general in the standard manner. For example, the representation types are installed in the ClassInitialize method, and not through the central XmRepType manager. Callback and enumerated types are defined in the header files, where the correct place for private toolkit implementation would be the XmP.h header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

The components offer no new functionality to the Motif 2.1 toolkit,. are thread unsafe, incompatible with the standard toolkit, and are fully redundant.

ToolTip

Description

An assistive text utility, which displays popup context sensitive information.

General Observations

In principle, a more than worthy idea, which provides functionality missing from the standard Motif 2.1 toolkit.

The functionality which it provides is not widget-based; the component itself is therefore intrinsically not fully accessible to programmer via code, or the general user through external sub-part resources.

The implementation relies on extensions to the VendorShell, Primitive, and Gadget classes. While additional Primitive and Gadget resourcing is acceptable, modifying the VendorShell extension record is likely to render the toolkit non-backwards compatible at the binary level.

Scope

Works with both Widgets and Gadgets, but not Managers or Drawables.

Does not allow per-target rendition specification, or per-target enable/disable controls.

Does not allow specific default rendition characteristics, but merely inherits the default font from the VendorShell.

Does not cater for alternative assistive text methodology.

Thread Safety

No consideration has been given to the thread safety of this component. Although the routines which create the tooltip hierarchy do not reference class members, the public support routines reference through widget instance structures without locking the application context. This is unstable in a multi-threaded environment.

Motif 2.1 Compatibility

No consideration is given to the Motif 2.1 Trait system. A new Trait associated with assistive technology in general is indicated here.

No consideration is given to Motif 2.1 layout direction in positioning the tooltip popup.

Construction

The functionality is implemented through the VendorShell, and is not widget-based. Public access to the hierarchy is hidden from the programmer.

There is no associated public header file for the functionality.

The component does not take into account the edges of the screen, but doggedly attempts to locate the tooltip at a fixed offset and location to the source of the help. This will

truncate the assistive text. No consideration is given to configuring the relative positioning of the popup.

The implementation assumes one tooltip at the top of the widget hierarchy. This may not be a valid assumption given the requirements of internationalized applications.

Conclusion

There is a lack of generalization here, which misses the point of the Motif 2.1 Trait mechanisms. The Motif 2.1 toolkit attempted to isolate the system from specific details of implementation, so that for example a Manager no longer operates by expecting specific Text components in specific configurations, but queries and sets through the XmQTaccessTextual Trait, which is supported by a variety of components.

The logic here reverses this: it hard codes a specific implementation. It does not consider the generic problem of assistance in the round. So for example, one might reasonably consider whether all assistive text is appropriate to be handled in this form: it would make sense to allow display of the assistive text in a Status Line at the bottom of the current dialog as an alternative mechanism, or delegate the interpretation of the assistive text to a generic assistance module.

A toolkit is about enabling technology, not specific technique, and the implementation here misses the point entirely, although one does not doubt that it works within its limitations. One can see the logic in parts of the implementation - Gadgets are an issue because of lack of public access to focus change, which the technology requires for operation. This should have been opened up in a generic way - add general purpose focus change/enter/leave callbacks to the component class, rather than make specific single-purpose modifications just to make one specific component work. This way, alternative assistive technology, such as the play of audio clips on entry, are also available to the programmer, and not just one specific means of displaying specific help in a specific way. A generic solution to the problems of Accessibility will therefore require that this code is all stripped out and reworked from the top down. Far from moving towards an Accessible Motif, the implementation paints the toolkit into a corner.

The implementation is also lacking in crucial aspects of configurability, particularly in the ability to specify rendition information on a per-target basis. This will have severe restrictions on the internationalization aspects of the technology.

Embedding specific technique into the toolkit will also have implications for those in the general community who have worked out alternative solutions to the problem outside the scope of the existing Motif toolkit. It is not clear whether the embedded assumptions are going to sit well with the alternative solutions.

Motif 2.1 will benefit from tooltip functionality, especially as an enabling technique towards generic assistive technology; the implementation here lets it down.

Tree/Hierarchy

Description

A Constraint Manager which lays out its children in indented outline/tree format, with connective lines.

General Observations

Redundant - the Container/IconGadget combination configured in Outline mode is specifically designed for this purpose.

Scope

Does not support Gadgets in any shape or form.

Thread Safety

No consideration has been given to the thread safety of these components. Both the Tree and the Hierarchy superclass reference widget class members without implementing an application context lock. This will render the component particularly unsafe in a multi-threaded environment.

Motif 2.1 Compatibility

No consideration is given to the Motif 2.1 Trait system. One might give consideration to XmQTtraversalControl in this context.

No consideration has been given to the XmeDraw utilities in these components. This will give an inconsistent look.

Synthetic resolution independent resources are calculated using deprecated _XmToVerticalPixels/_XmToHorizontalPixels code.

Layout is not sensitive to Motif 2.1 layout direction functionality.

The components use the subsumed Drag-and-drop utilities, and not the replacement Motif 2.1 Uniform Transfer Model.

Construction

The widget has not been embedded into the Motif toolkit in the standard manner. For example, the representation types are installed in the ClassInitialize method, and not through the central XmRepType manager. Enumerated types are defined in the header files, where the correct place for private toolkit implementation would be the XmP.h header.

XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with a Motif toolkit of uncertain version, but it is not the way that a component embedded within the toolkit itself is constructed.

Conclusion

The components are redundant in, and incompatible with, Motif 2.1, and thread unsafe.

Toolkit Support Utilities

Picture

The XmPicture sources, in support of the DataField, allocate a fixed size array for regular expression parsing. Long regular expressions will exceed the bounds and cause segmentation violation in the application.

This should be reworked before consideration should be given to integration. There are standard regular expression encapsulations in the general toolkit in any case, and thus this code should be marked as redundant.

XmExtUtils

Provides multi-byte translation on behalf of the proposed new components, and general component move/resize functionality. Redundant in the presence of the XmTextType support built into the Motif 2.1 XmParse utilities, and the Motif 2.1 XmeConfigure support.

EditRes

This should not be embedded in the general toolkit *on principle*.

This is not to say that the functionality it provides is not useful, or in any way worthy of inclusion somewhere. By all means supply any code required to interface to specific applications in a separate support or contributions section of the release.

You should not incorporate application-specific code in a generic application-independent toolkit, and therefore on principle, this should be removed from the library.

Toolkit Maintenance

There appears to be a very limited degree of bug patching and general maintenance in the toolkit over and above the base Motif 2.1.30 release.

There is nothing at this stage of the investigation to doubt the functional correctness of certain of the patches.

However, there is an issue with the implementation of patches associated with multi-byte support in the Darwin operating system. While not doubting that the problem has been isolated, and the solution may well work, there is a lamentable lack of principle in the implementation.

One would expect that the solution should be abstracted, so that the sources are not embedded with platform-specific conditional compilations.

The correct and principled solution here would be to generalize the problem, place the porting fix in modular form into the correct place (Xmos.c), and call the appropriate general solution from the rest of the sources.

There is therefore a basic lack of engineering and source hygiene technique evident in the workmanship of the port.

General Considerations

The importation of the proposed new widgets into the Motif widget set is in non-standard form:

- The public enumerative types and definitions, and callback structures associated with the components are defined in header files outside the scope of the Xm.h header.
    - the Xm.h public header is the standard place for public type definitions.
- There are namespace issues associated with the Ext.h header which links the additional component namespace into the Motif.
    - the Xi namespace is usually associated with the standard low level Xi device driver library.
- The messages which are defined therein have not been relocated into the standard XmString message definition system.
    - this has side effects on internationalizing the toolkit.
- Not one of these components uses standard Motif Fast sub-classing.
    - the rest of the toolkit will therefore look on these components as foreign in crucial aspects. The RowColumn, for example, relies on Fast subclassing for various aspects of its behavior: an IconButton could not be placed in a homogenous RowColumn.
- XtVersionDontCheck/XmResolveAllPartOffsets might be a reasonable way of integrating a foreign component with an Xt-based toolkit of uncertain version and compilation, but it is not the way that a first-class integration within the toolkit itself should be constructed.
- Representation types for enumerations are installed through ClassInitialize methods, and not through the central XmRepType management system.
- Non-use of Traits means that components do not interface well with the rest of the Motif toolkit.
    - this will give an inconsistent feel, and since parts of the navigation system rely on this functionality, it will restrict the Accessibility of the toolkit as a whole.
- Non-use of Xme utilities will give an alternative rendering and appearance to these components.
    - this will give an inconsistent look.

- Non-use of the Uniform Transfer Model means that the transfer of data between these components and those elsewhere in the toolkit is potentially inconsistent in behavior.

    - this will give inconsistent application dynamics.

In many cases, the components implement bespoke caches outside the scope of the standard toolkit object caches. Pixmap caching is particularly prone to this. The reference counting of objects of this type is therefore suspect; one would not like to speculate what would happen if a Pixmap were shared between the IconButton and the general toolkit for example, where the Pixmap were to be deallocated by the standard mechanisms.

All in all, the merging of the proposed new components into the toolkit leaves much to be desired. It is not a seamless and consistently thought out integration.

The WML specification supplied with the toolkit does not include the proposed new components in any shape or form. UIL, and therefore indirectly Ada, application bases will not work with this toolkit at all levels of detail. Since the ICS BuilderXcessory GUI builder, through the EPak proprietary extensions, supports the extended toolkit, but reserves the additional UIL specification to itself, this toolkit only works at all levels of detail with a specific support tool from a specific proprietary location.

In addition, certain components contain resource specifications which are specific to the proprietary GUI builder. This toolkit cannot be regarded as truly open.

Conclusions

Irrespective of whatever statements which can be made about the workmanship gone into the construction of the toolkit, the compatibility of the components, or the appropriateness of the functionality, the multi-thread safety issue is paramount. This has potential to severely damage any application base built on top of this toolkit. It should not be used for any mission-critical applications.

Open Source notwithstanding, this version of the toolkit is a very long way from production quality, and is not worthy of the Motif heritage.

The workmanship of the release is more than poor. While the general model under which the proposed components are constructed would be reasonable for a Motif 1.2 environment, here there are no Traits, RenderTables, TabLists, Xme encapsulation, Uniform Transfer Model, Layout Direction, or indeed any other Motif 2.1 features. The components are not embedded into the toolkit in the standard manner. The components form an isolated island where they reference each others functionality, but not the rest of the toolkit in general. In reverse, the standard toolkit will not recognise the functionality which the proposed components offer at crucial levels of detail, because there is no Trait support offered. To summarize, they neither import from, nor export to, the standard Motif 2.1 toolkit.

The X FontSet is entirely absent from the additional sources, precisely where it might reasonably be expected. This sets back much of the work which has been performed to make the Motif toolkit fully internationalised, and usage of the same internationalizable.

Contrary to the impression given by the accompanying materials, the components are not Motif 2.1 toolkit compatible precisely *because* they use XmPartOffsets. A tight integration should use fast sub-classing.

In view of the dependence of the toolkit on a specific proprietary support utility, both in the embedding of proprietary resource constructs and in the absence of public WML specifications, the release should be marked explicitly by origin and purpose. Whatever else this is, the toolkit is *not open*.

Observations

The toolkit has been released without respect for due process.

No formal test suite exists for the new functionality.

Not one of the partners, operating system vendors, or any other interested parties associated with the OpenGroup or the Motif community in general were advised, consulted, or otherwise invited to give peer review of any of the contents of this release.

OpenGroup themselves were at no point given no prior warning of the release, and were completely ignorant of the contents and quality. when released ostensibly in their name.

The OpenGroup has removed all official status for the toolkit; the release has been downgraded accordingly.

OpenMotif 2.1.30 remains the officially supported release.

Recommendations

Wherever released, the toolkit should be accompanied with explicit and severe warnings as to its true status, particularly with respect to the thread-safety aspects of the release. Since the most mission- and enterprise- critical applications in the world are still Motif-based, distribution any other way would be irresponsible in the extreme. This version of the toolkit cannot and must not be used for such applications under any circumstances.

The vast majority of the proposed new functionality associated with this release is redundant; it is not the case that the shortcomings of the release should be fixed; there is far too much wrong with the workmanship of the release for it to form the basis of a proper officially supported version. The entire set of additions should be removed completely.

There is a more than significant lack of understanding of the Motif toolkit exhibited by those responsible for the release. A protracted period of learning and instruction should be undertaken, not just in the Motif 2.1 toolkit, but also in what is required to support fully internationalised mission critical applications, before any such further attempt to modify the standard toolkit is undertaken. Proprietary considerations above all must take a poor second place to any considerations as to the responsibility of the task at hand.

This must not under any circumstances form the basis of the next official release of Motif.