EggToolbar and EggMenu

GUADEC 2003
Dublin, Ireland

James Henstridge <james@daa.com.au>

Introduction

- What are they?
- EggMenu in depth.
  - Comparison with existing APIs.
  - Some concepts used by the system.
  - Description.

What Are They?

EggToolbar:

- A new toolbar implementation for GTK.
- Aims to be mostly backward compatible with the existing GtkToolbar and the BonoboUIToolbar.

EggMenu:

- A new menu and toolbar handling API.
- Action based.

Both will hopefully land in GTK 2.4.

Why Target GTK?

- While some people would like to see all GTK application developers use Gnome libraries, this probably won't happen soon for a number of reasons:
  - Portability to Windows.
  - Limit number of dependencies.
- Adding new features at the wrong conceptual level to encourage application developers doesn't seem to work.
- Need a better strategy.
  - Provide a good set of APIs in GTK for people to use.
  - Make those APIs extensible so that Gnome (and others) can add some value.
- This has the following benefits:
  - GTK only application developers can also use the best of our APIs.
  - If they want to use some Gnome libraries, they don't need to rewrite half their application.

EggToolbar History

- Existing GtkToolbar acts quite different to most containers.
  - children are usually created while adding to the parent.
  - All interfaces for manipulating buttons work with child indices.
    - a pain to keep track of if you are removing or inserting items.
  - Some desired toolbar layouts are impossible.
    - Right aligned items.
    - items that expand (such as the location bar in a web browser).
- BonoboToolbar also has its problems:
  - It was written due to limitations of GtkToolbar.
  - It is not exposed directly to the programmer.
  - It acts subtly different to GtkToolbar.

EggToolbar Features

- Primarily written to provide features needed by the new menu code.
- Designed to provide an API backward compatible with GtkToolbar.
  - If you never use any of the new APIs, it should act almost identical to the old API.
  - If you do use some of the new interfaces, some of the obscure parts of the old API will change.
- All toolbar child widgets are EggToolItems.
  - Items can be added and removed like normal widgets.
  - Some intelligence has been moved to the items, so that they are responsible for following the style of the toolbar.
- Items can be right aligned. (eg. throbbers)
- Items can expand to fill available space. (eg. location bars)
- Overflow is handled.

Why a New Menu Handling API?

- Existing menu API is insufficient for advanced programs.
- Bonobo addresses this to an extent, but has some issues:
  - API differs to the GTK one, so significant effort must be made to port to Gnome.
  - Requires use of CORBA, which some people don't like for some reason.
- We want an API in GTK that:
  - satisfies the requirements of larger applications.
  - satisfies needs of larger applications
  - simple to use so that it doesn't look too heavy for small applications
  - flexible enough that it can be extended to do what component systems and compound document systems need it to

Existing Menu/Toolbar API

- Code for creating menus looks a bit like this:
  - Create GtkMenuBar
  - Add GtkMenuItems for toplevel menus, and attach GtkMenus to them.
  - Add GtkMenuItems to the GtkMenus
  - Attach callbacks to the "activate" signal of the menu items
  - Repeat for sub-menus ...
- Toolbars are similar, although no sub-menus.
- The GtkItemFactory code can help here, but it is essentially a short hand for the above.

Problems With the Existing API

- Menu structure defined by code
  - if you want to rearrange menu structure, you need to rearrange the code
    - sometimes causes problems with signal connections, etc
- To enable or disable a user action, you must alter the state of the GtkMenuItem or toolbar button.
  - if you have multiple ways of performing the action, you need to alter all widgets.
- Doesn't even attempt to handle things like menu merging

Actions

- Represent something the user can do
  - a callback (or more than one)
  - a label to use in menu items
  - maybe a shorter label for toolbar buttons
  - an icon
  - state (sensitivity, visibility, etc)
- Can create an arbitrary number of menu items and toolbar buttons for an action
  - properties and state of menu/toolbar items mirror the action they represent
  - set action to disabled -> all widgets representing the action are disabled
- Different types of actions (extensible)

Action Groups

- Actions are grouped together into groups of related actions
  - actions that should be available in the same context
  - global actions: quit, new, open, etc
  - document specific actions: save
  - mode specific: actions needed when in a particular mode
    - table editing in a word processor
    - drawing layer in a spreadsheet
- Simple apps may have one action group
- Complex apps will have multiple groups.

UI Merging (continued)

- Orthogonal to actions
  - (some toolkits have actions, but do not provide a menu merge API).
- Used to overlay a set of menu/tool items onto another set, and demerge them later.
- A tree of menus and toolbars is maintained, with names attached to nodes.
- Nodes map to actions
  - if action is provided by multiple action groups, top action group wins
- Menu layouts described by XML files
  - based on a subset of the Bonobo UI format
  - translatable strings kept out of the XML file

Merge Example

file1.ui	file2.ui
`Root`	`Root`
`  menu`	`  menu`
`    submenu: FileMenu`	`    submenu: FileMenu`
`      menuitem: Open`	`      separator`
`    placeholder: TestPlaceholder`	`      menuitem: Quit`
`    submenu: HelpMenu`	`    placeholder: TestPlaceholder`
`      menuitem: About`	`      submenu: EditMenu`
`  dockitem: toolbar1`	`        menuitem: Cut`
`    toolitem: NewButton`	`  dockitem: toolbar1`
	`    toolitem: OpenButton`

Merged UI

```
Root
  menu
    submenu: FileMenu
      menuitem: Open
      separator
      menuitem: Quit
    placeholder: TestPlaceholder
      submenu: EditMenu
        menuitem: Cut
    submenu: HelpMenu
      menuitem: About
  dockitem: toolbar1
    toolitem: NewButton
    toolitem: OpenButton
```

UI Merging (continued)

- Nodes merged based on names
  - if node has no name, the node type is used as the name
- New nodes appended to containers
  - there is a flag to prepend instead
- Placeholders are "virtual containers" used to add ordering

Putting it into Practise

Creating an action:

```
action = g_object_new(EGG_TYPE_ACTION,
                      "name", "quit",
                      "label", _("Quit"),
                      "tooltip", _("Quits the application"),
                      "stock_id", GTK_STOCK_QUIT,
                      NULL);
```

Creating an action group:

```
action_group = egg_action_group_new("base_actions");
egg_action_group_add_action(action_group, action);
```

Putting it into Practise (continued)

Creating the `EggMenuMerge` object:

```
merge = egg_menu_merge_new();
egg_menu_merge_insert_action_group(merge, action_group, 0);
```

Adding some menus:

```
merge_id = egg_menu_merge_add_ui_from_file(merge, "menus.ui", &err);
```

Getting the menu bar:

```
menubar = egg_menu_merge_get_widget(merge, "Root/menu");
```

Removing menu items:

```
egg_menu_merge_remove_ui(merge, merge_id);
```

Accelerators

Accelerators are handled by the GTK accelerator map.

- All actions are assigned an accelerator path of the form:

```
<Actions>/group-name/action-name
```

- A shortcut is associated with this accelerator path.
- All menu and toolbar items take on this accelerator path.
- The GTK accel map code makes sure that the correct shortcut displays next to the menu item.

This has the following benefits:

- Two user interface elements representing the same action can be activated with the same keyboard shortcut.
- If dynamic shortcut editing is turned on, it will work correctly for EggMenu based menus.

Accessibility Concerns

- Hasn't been investigated much so far.
- We have the ability to provide a lot more information to accessibility tools compared to the older menu system:
  - If two widgets in the UI will perform the same action, we could make that relation explicit.
  - As an alternative to navigating all the menus, an AT could provide a way to trigger actions directly.

EggMenu To Do

- API for adding dynamic menu items. Will probably look like this:
  - create a merge ID.
  - manually create a node in the menu tree using that merge ID.
- Get more apps to test EggMenu API
  - Find out what is needed.
  - Make things more robust.