EggMenu

James Henstridge <james@daa.com.au>

Data Analysis Australia

Gnome Foundation

Introduction

- What is EggMenu?
- Gnome development processes
- EggMenu in depth
  - Comparison with existing APIs
  - Description

What is EggMenu?

- A new menu and toolbar handling API
- Currently in development
- Will hopefully land in GTK 2.4

Development Process (overview)

- Pre-Gnome 1.0: development is fairly unstructured
- Gnome 1.x: commit to providing binary compatibility
  - new features occasionally added
- Gnome 2.x: move closer to providing forward and backward compatibility in a minor series of versions
  - 2.0.x releases are bug fixes
  - 2.2.x is bug fixes + new features (backward compatible).

Pre-1.0 Gnome Development

- Development driven by what hackers were interested in
- Many people with check in privileges, but no strong standards for what to check in to the development platform (other than "doesn’t break the build").
- Resulted in a lot of good ideas along with many bad ideas
- Was not a great platform to recommend other people base apps

The 1.x Platform

- Something that other people could develop against.
- Some cruft removed from platform libraries
  - Keep stuff that was actually being used, and we would be able to maintain
- Maintained backward compatibility throughout 1.x series.
  - most new features were implemented in new releases of the platform, but not all.
    - (eg. some packages would require gnome-libs-1.2.3 because they used a 5 line function that was added in that release).

2.x platform

- Major upgrade to GTK (3 years in development)
- Taken as an opportunity to fix many of the problems in the interface that could not be fixed without breaking compatibility.
- Harder guarantees of binary compatibility
  - Within a minor series (2.0.x, 2.2.x, etc), forward and backward binary compatibility is maintained
- Has required us to develop new ways to introduce APIs
  - adding a bad API may mean managing that API for years
  - libegg is part of the new development process

Example: GnomeLamp

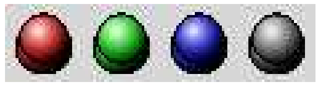
- A then unknown loon called Bowie posts to gnome-list about his "Color Reactive GUIs"

```
I propose that the Gnome desktop not only -feature- this design
innovation, but figure it prominently in the general layout of each
window as per the recommendations listed above, and shown in the
appendices.
```

- Two days later, an implementation turns up in gnome-libs
- API:

```
lamp = gnome_lamp_new_with_color(color);
gnome_lamp_set_type(GNOME_LAMP(lamp), GNOME_LAMP_BUSY);
```

- Example lamps:



GnomeLamp (continued)

- Usability problems:
  - not obvious what the widget is, does or represents
- I18N problems:
  - new_with_color() and set_color() are bad, because different colours have different meanings to people.
  - set_type() has the potential for localisation
- Accessibility problems:
  - colours is not a good medium to communicate information to people
    - blind users, colour blind users, etc.
  - The set_color() variant does not follow desktop theme.
- There are almost always better ways to provide information

Example: GtkTreeModel/GtkTreeView

- Was developed as a new tree widget for GTK 2.0
- Initially developed as a standalone module in CVS
- Design looked at existing widgets that performed the same task
  - Java Swing
  - Qt
- Initial implementation reviewed on gtk-devel-list
- After design issues found during review were fixed, moved to GTK.

EggMenu

- Existing menu and toolbar handling API in GTK is not sufficient for advanced programs.
- Bonobo has a more full featured API, but has some issues
  - API is quite different to GTK API, so is a barrier to porting apps to Gnome
  - Requires use of CORBA, which is not always desired.
- Want an API in GTK that:
  - 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 ...
- There is GtkItemFactory to simplify things, but it does essentially the same thing as this
- Toolbars are similar, although no sub-menus.

Problems with this setup

- 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 the action concept without menu merging).
- Used to overlay a set of menu/tool items onto another set.
  - and demerge them
- 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 Bonobo UI format
  - translatable strings kept out of the XML file

Merge Example

file1.ui

```
Root
  menu
    submenu: FileMenu
      menuitem: Open
    placeholder: TestPlaceholder
    submenu: HelpMenu
      menuitem: About
  dockitem: toolbar1
    toolitem: NewButton
```

file2.ui

```
Root
  menu
    submenu: FileMenu
      separator
      menuitem: Quit
    placeholder: TestPlaceholder
      submenu: EditMenu
        menuitem: Cut
  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

Future

- API for adding dynamic menu items
  - something better than bonobo’s API
- Get more apps to test EggMenu API
  - and fix problems.
- prepare for GEP process

Conclusions

- Gnome Enhancement Proposals
  - http://developer.gnome.org/gep/
- Code available in Gnome CVS
  - libegg module, libegg/menu subdirectory