EtoileUI

FOSDEM 2011

http://www.etoleos.com
Smalltalk vs ObjC Memo

Smalltalk

tulip witherWithSpeed: 54
color: NSColor redColor.

Objective-C

[tulip witherWithSpeed: 54
color: [NSColor redColor]];
Étoilé

A desktop environment built around:

• Pervasive Data Sharing & Versioning
• Composite Document
• Collaboration
• Light & Focused Applications (1000 loc max per app)
Well, presently more or less a development platform centered around

- LanguageKit
- CoreObject
- EtoileUI
Bird View

- EtoileUI
- CoreObject
- EtoileFoundation
- GNUstep

EtoileUI

Drawing Support  Widget Backend
Surprisingly Small

• Found on Digg (in 2007)…

• Konqueror itself is really a surprisingly small app: approx 40k lines of code. Not tiny, by any stretch of the imagination, but way, way smaller than people seem to think it is.

• 40x what is allowed in Étoilé :-/

From: http://digg.com/linux_unix/Nautilus_vs_Dolphin_vs_Konqueror
Code Compression

- Étoilé Generic Object Manager
  - 700 loc
- Étoilé Model Builder
  - 1000 loc
Model Builder
Editing a package & browsing a repository
EtoileUI

• Post-WIMP Toolkit
• Inspired by Morphic, HotDraw, Taligent and OpenDPI
• Kinda related to CoreAnimation, Clutter, GEGL, WPF, HTML etc.
Post-WIMP?

• No special assumptions about the UI
• EtoileUI doesn’t require:
  • windows
  • menus
  • a mouse or a keyboard
Post-WIMP?

• From the whole screen to a single row in a list view…
• It’s just an uniform tree structure
• No special window, list or row node
Visual Search

Fog Vision
expense report
Visual Search Demo

Table Example

- Copy Window
- Paste Window twice (Don’t use Edit > Paste)
- Click last window background and Inspect
- Switch to Browser
- Close Inspector

- Visual Search
- First ‘Red’
- Then ‘NS’
- Then ‘ñ’
- Finally Nothing
Why?

• An existing application should be easy to retarget
  • personal computer
  • mobile phone
  • tablet
  • web
How it began

• Why UI in Photoshop and IDE are so rigid?
• Why UI development is getting easier on the Web than on the Desktop?
• Why not make UI programming really easy ;-)
Why a “new” UI toolkit?

• Everything can be changed at runtime
• Simple, compact and highly polymorphic API
• Write less code and develop faster
• Feeling of manipulating real objects
Screen Layout Demo
Collage & Markup Editor Example

Collage
- Switch window group to Free
- Move things around
- Ungroup the items that represent windows
- Move subitems between windows
- Switch back to Window layout

Markup Editor
- Open some documents
- Switch window group to Master Detail
- Navigate a bit
(Don't switch back to Window layout)
What does it solve?

• Generic protocol for Structured Document
• Building blocks for Graphics Editor
• Custom widget development
• As little code as possible
• Plasticity
Separation of Concerns

• No monolithic view/widget, but rather…
• UI aspects
  • Styles, Decorators, Layouts
  • Tools, Action Handlers
• Widgets
• Model Objects, Controllers
- Cut Red in Top Left
- Paste in Bottom Left
- Drag Icon from Bottom Left to Top Right
- Move Icon back to Bottom Left
- Click window background then Inspect button
- Switch main item to Outline
- Drag Top Left into Icon (of Bottom Left)
- Switch main item back to Fixed
- Show Top Left in Icon
- Drag Top Left on window background
- Click Bottom Left background
- Copy Bottom Left
- Paste it into Bottom Left
- Grow the Window
- Switch Bottom left to Free
- Rearrange items in Bottom Left (to show the subitem using Outline)
- Switch Bottom Left from Free to Browser
- Navigate a bit
Turtles all the way down

- Many things are just items
  - selection rectangle
  - handles
  - shapes
  - windows
  - layers
Collage Demo
Composite Layout

• Packaging entire UI as a component

• reusable on an arbitrary item

• Basic Idea

• Wrap an item tree in a layout

• Inject this item tree when the layout is used

• Undoable at any time
Composite Layout Features

- Arbitrary nesting
- Content routing
  - e.g. to the item that represents a source list
- Delegating top item arrangement to a positional layout
  -ETFFlowLayout, ETFixedLayout etc.
Basic Example

ETLayout *originalLayout = [mainItem layout];

// Make mainItem looks like a Markup Editor
// Can be usable or not based on the model bound to mainItem
[mainItem setLayout: [ETMarkupEditorLayout layout]];

// Restore mainItem original UI
[mainItem setLayout: originalLayout];
Why is possible?

• Each item is pretty much a black blox
• Controllers are clearly owned
• Items must not interact with parent item aspects
• Force the developer to
  • organize the objects around the item tree
• use strict ownership rules

The two first points are important to dispatch actions from menu.
Pane-based UI

- Tabs
- Two Panes
  - Master Detail
  - Master Content
- Slideshow
- Drilldown
- Wizard
Pane Taxonomy

- Two Panes
  - Tabs, Master Detail, Master Content
- One Pane *(implicit pane)*
  - Slideshow *(photo source list)*
  - Drilldown *(path)*
  - Wizard *(step list)*
Pane Layout

- Provides various pane-based UI patterns
- ETCompositeLayout subclass
  - Easy to apply and revert with -setLayout:
  - Ability to route content between panes
  - Package a item tree with two nodes
Pane Layout

- Two pane items
- bar item
- content item
- Internal layout control
- bar thickness
- bar position
- Navigation actions (go to item, go back etc.)
Three Panes?

- Three Panes (or more) can be built by nesting Two Panes

```swift
[mainItem setLayout: [ETPaneLayout layout]];
[[[mainItem layout] contentItem]
    setLayout: [ETPaneLayout layout]];
// We could continue to nest panes more deeply
```
- Switch the subcontent item (first child) to Master Content
- Switch the subcontent item (first child) to Table
- Select the top group item (window level)
- Switch it to Master Detail
- Navigate a bit
- Compile without THREE_LAYOUT_PANES
- 'MarkupItemFactory'
- Open a document
- Don't click anything
- Inspect Item
- Switch second subnode to Master Content
- Switch the content item (first child) to Table
- Switch the content item to Text Editor and back to Table
- Switch the content item (first child) to Master Content
- Switch the subcontent item (first child) to...
What’s New?

- Cover style to draw on top
- True Automatic Layout
- Semantic Separator at layout level
- Template support at controller level
- Document Controller
- Nib Integration
Automatic Layout Previously

- Triggered by
  - item insertion or removal
  - a layout change
- Layout updates were all
  - immediate
  - recursive
  - never coalesced
Automatic Layout Now

• Triggered by
  • a geometry change too

• When the current event handling ends, layout updates are
  • coalesced
  • reordered (to ensure children are sized)
  • executed
What does that mean?

• An item can be resized by its layout transparently
• No redundant layout updates
• No -updateLayout use

[itemGroup setWidth: 300];
[itemGroup updateLayout];
Nib Example 1

- (void) applicationWillFinishLaunching: (NSNotification *)notif
{

    // Turn the nib views and windows into layout item trees
    [ETApp rebuildMainNib];
}

Nib Example 2

ETLayoutItemGroup *windowGroup =
    [[[ETLayoutItemFactory factory] windowGroup];
ETNibOwner *nibOwner = [ETNibOwner new];

[NSBundle loadNibNamed: @"test" owner: nibOwner];
nibOwner rebuildTopLevelObjectsWithBuilder:
    [[[ETEtoileUIBuilder builder] [windowGroup addItem: [[nibOwner topLevelItems] firstObject]];]};
// In ‘test’ nib, the controller mainContent outlet is connected
// to a view or window

ETController *controller = [ETController new];
[NSBundle loadNibNamed: @"test" owner: controller];
[windowGroup addItem: [controller content]];
Document Controller

• Adding, removing documents on screen is the same than adding, removing anywhere else

• No document architecture

• No UI expectations

• such as each document is a window
What is this Controller then?

- ETDocumentController is a tiny ETController subclass
- 150 sloc
- less than 15 methods
- Open and save in addition to New
- Managing items by URL
Document Controller

• Usually bound to the window group
• But can be bound to any item group
  • Single window document editor or related
• Should be extendable to manage items downwards in the item tree
  • Tabbed documents in multiple windows
Document Controller Example

ETItemTemplate *template =

    [PListItemTemplate templateWithItem: [itemFactory editor]
    objectClass: mutableDictClass];

    [self setTemplate: template forType: plistUTI];
    [self setCurrentObjectType: plistUTI];

    // self is the Document controller instantiated in the nib
    [[itemFactory windowGroup] setController: self];
    [self newDocument: nil];
PListItemTemplate Code

- (ETLayoutItem *) contentItem {
    return [[self item] itemForIdentifier: @"documentContent"];
}

- (ETLayoutItem *) newItemReadFromURL: (NSURL *)URL options: (NSDictionary *)options {
    // Reading plist file (six lines omitted)
    return [self newItemWithRepresentedObject: plistNode options: options];
}
// In -initWithNibName:bundle:

ETItemTemplate *template =
    [ETItemTemplate templateWithItem: item
        objectClass: mutableDictClass];

[self setTemplate: template forType: [self currentObjectType]];
Document Demo

- Compile with THREE_PANES_LAYOUT
- Launch
- Show the code and play a bit
- Recompile without WINDOWGROUP_DOCUMENT_CONTROLLER
- Open some documents
- Navigate a bit
Work-in-progress

• Form layout
  • including Model-driven item tree generation
• Viewpoints
  • to create custom “views” on object graphs
  • to model content flow
• Automatic reload
  • when a model collection is mutated
What’s Next?

• Aspect Repository
• Object Graph Styling
• Generic Template Picker and Object Palette
• Compound Document Editor
  • can be used as a UI builder
• CoreObject Integration
Compound Document Editor

- Can be reduced to
  - Inspectors + Object Palettes
- Inspectors
- Aspects + Metamodel
- Object Palettes
- Aspect Repository + Object Picker Layout
Long Run

- Constraint Solver
- Additional Backends
  - Web, GTK, UIKit, OpenGL etc.
- Minimal EtoileUI packaging a tiny AppKit subset
Long Run continued

• Animation (CoreAnimation probably)
• Diagram editing
• Hybrid vector and bitmap editing (EtoilePaint)
• Generative Art a la Processing, Field, Nodebox etc.
Composite Demo

Markup Editor Example

Cancelled
Code Compression continued