EtoileUI

at FOSDEM 2010

Smalltalk vs ObjC Memo

Smalltak

```
tulip witherWithSpeed: 54
                        color: NSColor redColor.
```

Objective-C

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

Étoilé

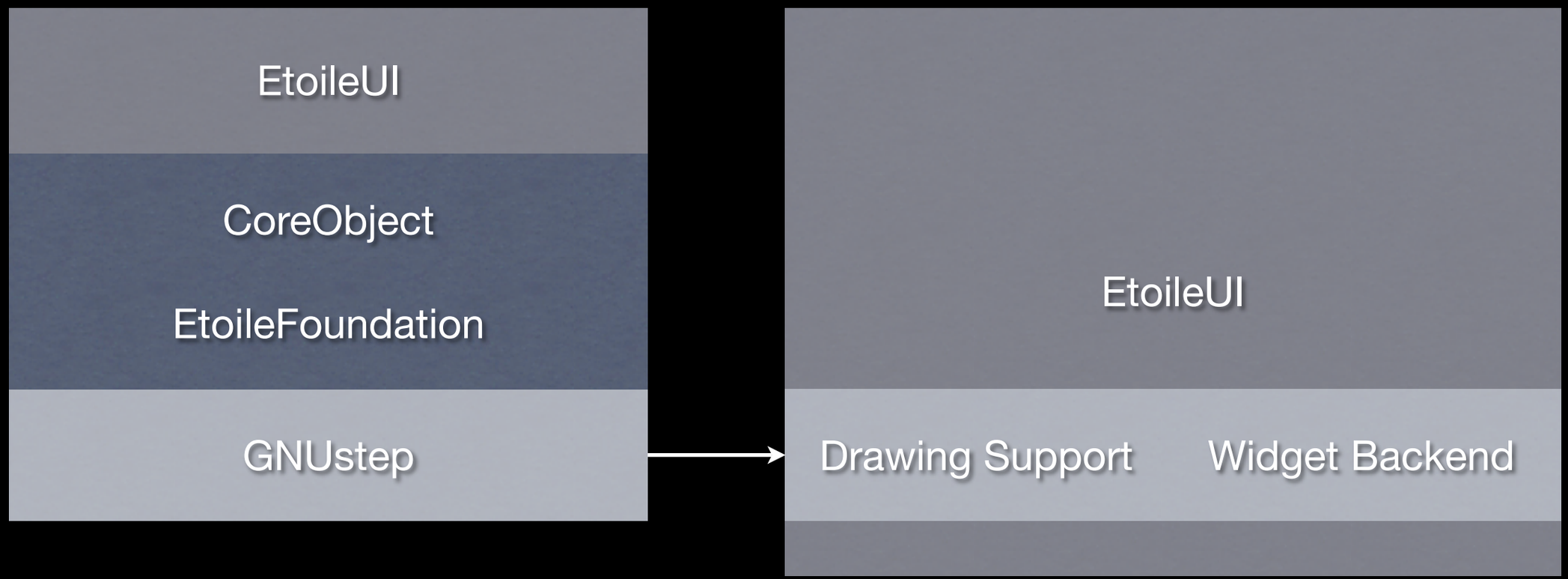
A desktop environment built around:

- Pervasive Data Sharing & Versioning
- Composite Document
- Collaboration
- Light & Focused Applications (1000 loc max per app)

Étoilé Today

- 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 !

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

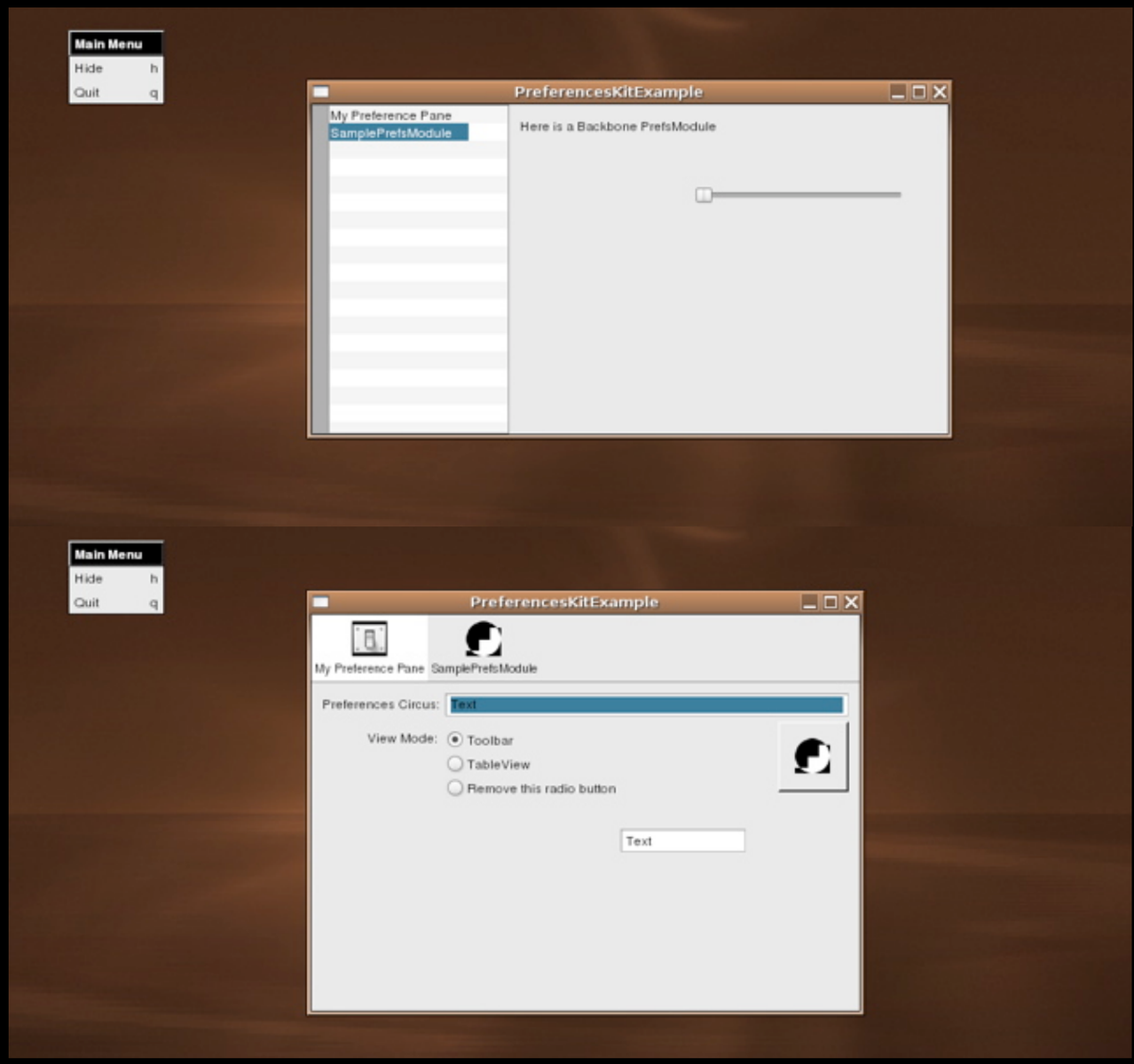
Why?

- An existing application should be easy to retarget:
  - personal computer
  - mobile phone
  - tablet

How it began

- Why UI in Photoshop and IDE are so rigid?
- PaneKit
- AppKit is great but:
  - still low-level
  - NSView and NSCell hierarchy doesn't scale
- Why not make UI programming really easy ;-)

PaneKit Example



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

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/wigdet, but rather…
- UI aspects
  - Styles, Decorators, Layouts
  - Tools, Action Handlers
  - Widgets
  - Model Objects, Controllers

Turtles all the way down

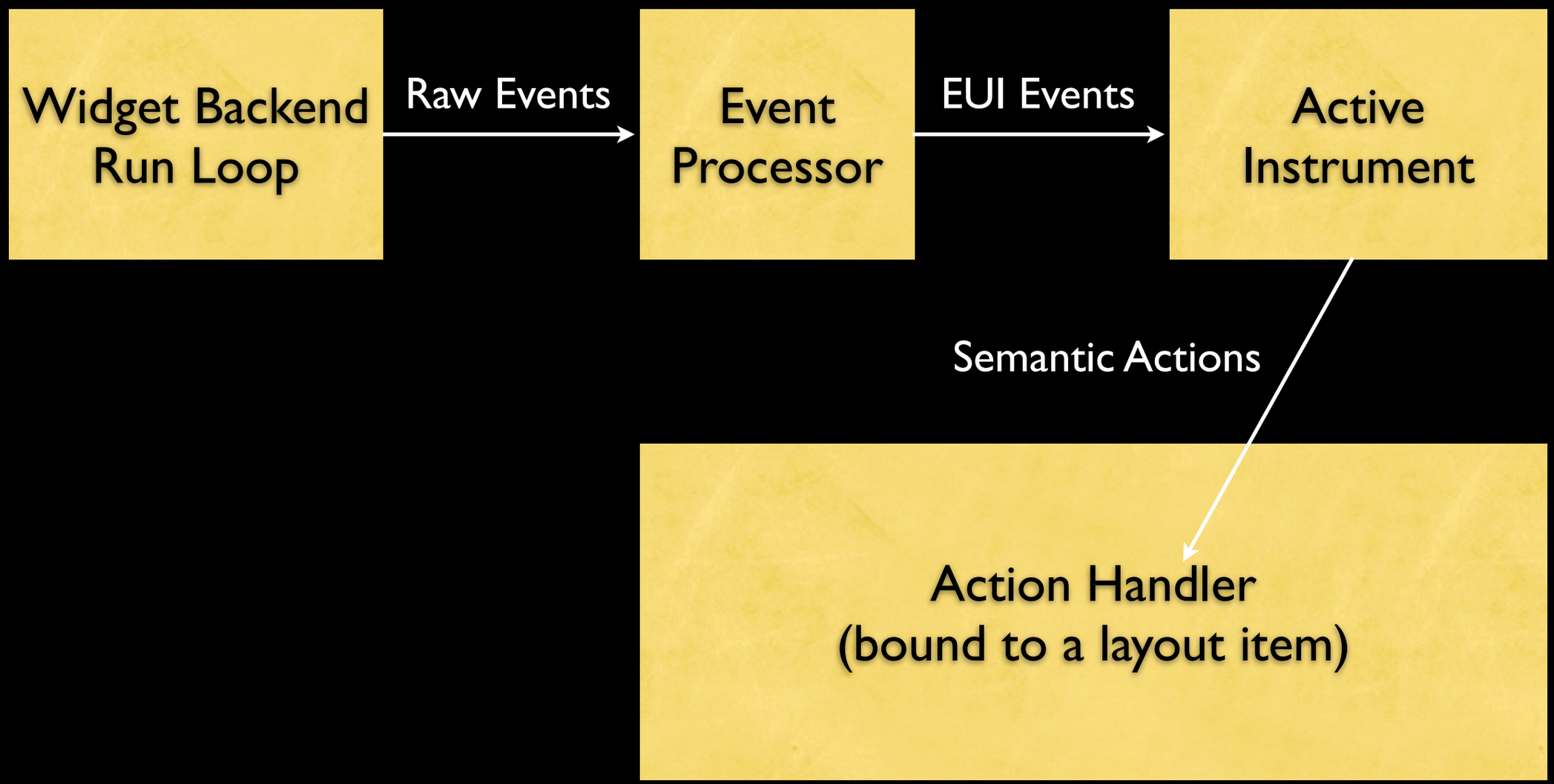
- Many things are just items:
  - selection rectangle
  - handles
  - shapes
  - windows
  - layers

Graphics Editing

- Shapes
- Layers
- Tools
- Styles
- Generic handle, control point support etc.
- Layouts

Collage Demo

From Events to Actions



Instrument Examaple

```
tool := ETSelectTool tool.

tool setShouldRemoveItemsAtPickTime: true.

tool selectionAreaItem setShape: ETShape circle.

item layout setAttachedInstrument: tool.
```

What do we gain?

- Input Device Indepent
- Multi-instruments Interaction
  - one per input device (e.g bimanual interaction)
  - one per user (e.g. collaboration)
- Ability to operate over process boundaries

Bloated?

- Hard to be objective but…
- 30 000 loc
  - code reuse
  - small, consistent and polymorphic API
  - layout item all the way down…

Hello World

```
NSObject subclass: HelloWorld [
    run [
        ETApplication sharedApplication setDelegate: self.
        ETApplication sharedApplication run.
    ]

    applicationDidFinishLaunching: notif  [ | itemFactory helloItem |
        itemFactory := ETLayoutItemFactory factory.
        helloItem := factory itemWithValue: 'Hello World!'.
        factory windowGroup addItem:  helloItem.
    ]
]
```

FileManager Example

```
CODirectory extend
[
    +fsRoot
    [
        ^ self objectWithURL: (NSURL fileURLWithPath: '/').
    ]
]
```

FileManager in 70 loc

```
ETController subclass: ObjectManagerMainController
[
    dateFormatter [ 1 loc ]

    visit: sender [ 2 loc ]

    awakeFromNib
    [ | itemFactory managerItem controller |
        30 loc -- presented in the next 4 slides
    ]
]
```

FileManager Window

```
managerItem := itemFactory itemGroupWithRepresentedObject:
                                                CODirectory fsRoot.

managerItem setSource: managerItem;
        setFrame: (NSValue rectWithX: 300 Y: 150
                                        width: 500 height: 400);
        setDoubleAction: 'visit:';
        setTarget: self;
        setController: ETController new;
        setHasVerticalScroller: true;
        setLayout: ETOutlineLayout layout.
```

FileManager List View

```
managerItem layout setDisplayedProperties:
                                        { 'icon'. 'displayName'. 'date'. 'size ' };

        setEditable: true forProperty: 'displayName';

        setDisplayName: 'Modification Date' forProperty: 'date ';
        setFormatter: self dateFormatter forProperty: 'date';

        setDisplayName: 'Size' forProperty: 'size';
        setFormatter: ETByteSizeFormatter new forProperty: 'size'.
```

FileManager Drag and Drop

```
managerItem setShouldMutateRepresentedObject: true.

fileUTI := ETUTI typeWithClass: (COFile class)
dirUTI := ETUTI typeWithClass: (CODirectory class)

managerItem controller setAllowedPickTypes: {  fileUTI }.
managerItem controller setAllowedDropTypes: { fileUTI }
                                        forTargetType: dirUTI.
```

FileManager Multi-Window

managerItem reloadAndUpdateLayout.

itemFactory windowGroup setController: self.
self setTemplateItemGroup: managerItem.

self addNewGroup: nil.