

# Preparing For Pollock

Number 1, 1950 (Lavender Mist)

Jackson Pollock 1912-1956



# Preparing For Pollock

**The Future of the  
VisualWorks GUI**

# *Preparing For Pollock*

- ➡ Why a new GUI Framework
- ➡ Pollock Philosophy and Goals
- ➡ The Path
- ➡ The Metaphor
- ➡ The Basics
- ➡ Pollock & The Trigger Events Framework
- ➡ Models & ValueEvents

# *Preparing For Pollock*

- ☞ Pollock & DragDrop
- ☞ Pollock & Edit Keys
- ☞ Coding Today With The Future In Mind
- ☞ Pollock Tools
- ☞ Building A New Widget – An Example
- ☞ Q&A

# *Why a new GUI Framework*

- ☞ What needs to be obsolete
  - ☞ Wrappers
  - ☞ Look Specific Widgets
  - ☞ Ball Of Mud Controllers
  - ☞ Static Edit Keys
  - ☞ Fragile & Noisy Change/Updates in the Guts
  - ☞ Do Too Much Builder



# *Why a new GUI Framework*

- ☞ What needs to be added
  - ☞ Configurable Hotkeys
  - ☞ Dynamic Look Changing
  - ☞ Stupid Controllers
  - ☞ Full Use of the Trigger Event System
  - ☞ Separate Builder and Widget Inventory
  - ☞ E-Z Developer Widget Creation
  - ☞ E-Z Developer Widget Extension Capability

# *Why a new GUI Framework*

## ☞ Widgets In General

### ☞ Out of the box they need

☞ To Do More

☞ To Do It Faster

☞ To Do It Better

☞ Where Required: To Do It More Platform Faithfully

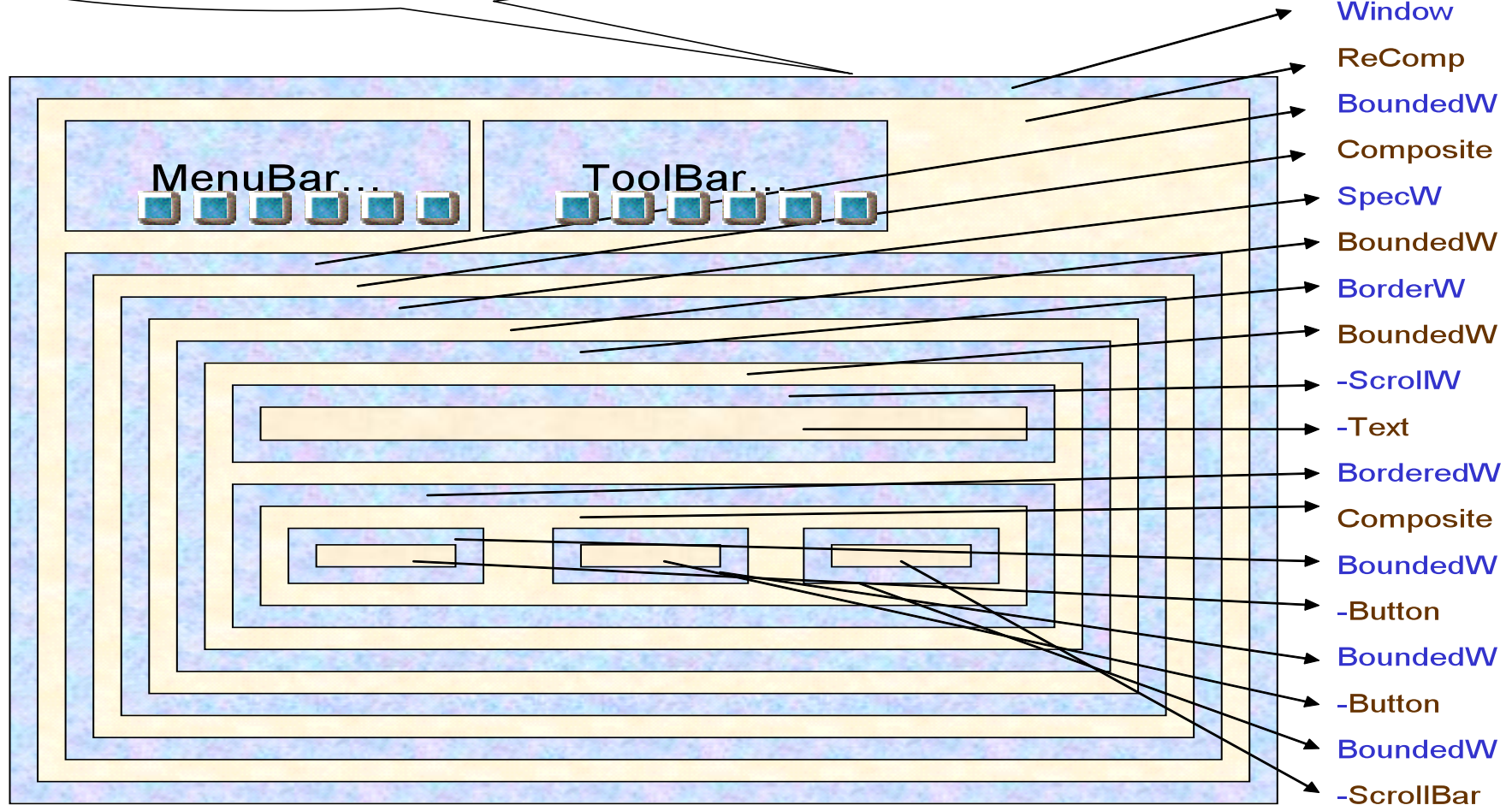
### ☞ And if not:

☞ E-Z To Make Them Do It

# What needs to be obsolete:

## Wrappers Today

Visual Launcher





# What needs to be obsolete: Wrappers Today

```
an ApplicationWindow 3
├─ a ReComposingComposite
│   ├─ a MenuBarWrapper on: a Win98MenuBar
│   │   └─ a Win98MenuBar
│   ├─ a BoundedWrapper on: a Win98ToolBar
│   │   └─ a Win98ToolBar
│   └─ a BoundedWrapper on: a CompositePart
│       └─ a CompositePart
│           └─ a SpecWrapper on: a BoundedWrapper on: a BorderDecorator
│               └─ a BoundedWrapper on: a BorderDecorator
│                   └─ a BorderDecorator
│                       └─ a BorderedWrapper on: a ScrollWrapper on: a TextCollectorView
│                           └─ a ScrollWrapper on: a TextCollectorView
│                               └─ a TextCollectorView
│                                   └─ a BorderedWrapper on: a CompositePart
│                                       └─ a CompositePart
│                                           └─ a BoundedWrapper on: a VisualPairButton
│                                               └─ a VisualPairButton
│                                           └─ a BoundedWrapper on: a VisualPairButton
│                                               └─ a VisualPairButton
│                                           └─ a BoundedWrapper on: a Win95ScrollBar
│                                               └─ a Win95ScrollBar
```

# What needs to be obsolete:

## ☞ Wrappers

- ☞ SpecWrappers: Hold on to too much runtime information, waste space with the rest
- ☞ Bounded, Bounding, Bordered... Baloney!
- ☞ Scrollbars : The result of composition run rampant
- ☞ Menus : Not much better
- ☞ Bounds calculation : The progenitor of modern C obfuscation contests

# What needs to be obsolete:

- ☞ Look Specific Widgets
  - ☞ Can't change their look on the fly without rebuilding whole window
  - ☞ Complex build rules (<widget>:into:)
  - ☞ Forces duplicated build code
  - ☞ Commingled common and look specific code

# What needs to be obsolete:

- ➡ Ball Of Mud Controllers
  - ➡ Controllers have lost their “MVC” meaning
  - ➡ They have become sophomoric
  - ➡ Because of wrappers are involved in downcasting & upcasting events
  - ➡ Knows too much about view

# What needs to be obsolete:

## ☞ Static Edit Keys

- ☞ Controllers do too much
- ☞ Controllers contain too much feel specific code
- ☞ Dispatch tables are code based
- ☞ The Developer knows best (i.e. least)



# What needs to be obsolete:

- ☞ **Fragile & Noisy Change/Updates in the Guts**
  - ☞ All dependents get all update information, even when they don't care (which in the guts of the GUI, is most of the time)
  - ☞ Updates fling up and down the wrapper hierarchy
  - ☞ Changing models and views is fraught with danger, making existing Widgets fragile
  - ☞ Debugging is a nightmare

# What needs to be obsolete:







- ☞ Do Too Much Builder
  - ☞ Builds then sticks around
  - ☞ “self builder componentAt:” (*nuff said!*)
  - ☞ Can't properly get nested components (sub-canvases)
  - ☞ Throws out important build time information

# What needs to be added (redux):

- ☞ Configurable Hotkeys
- ☞ Dynamic Look Changing
- ☞ Stupid Controllers
- ☞ Full Use of the Trigger Event System
- ☞ Separate Builder and Widget Inventory
- ☞ E-Z Developer Widget Creation
- ☞ E-Z Developer Widget Extension Capability

# Pollock Philosophy and Goals

## Goals

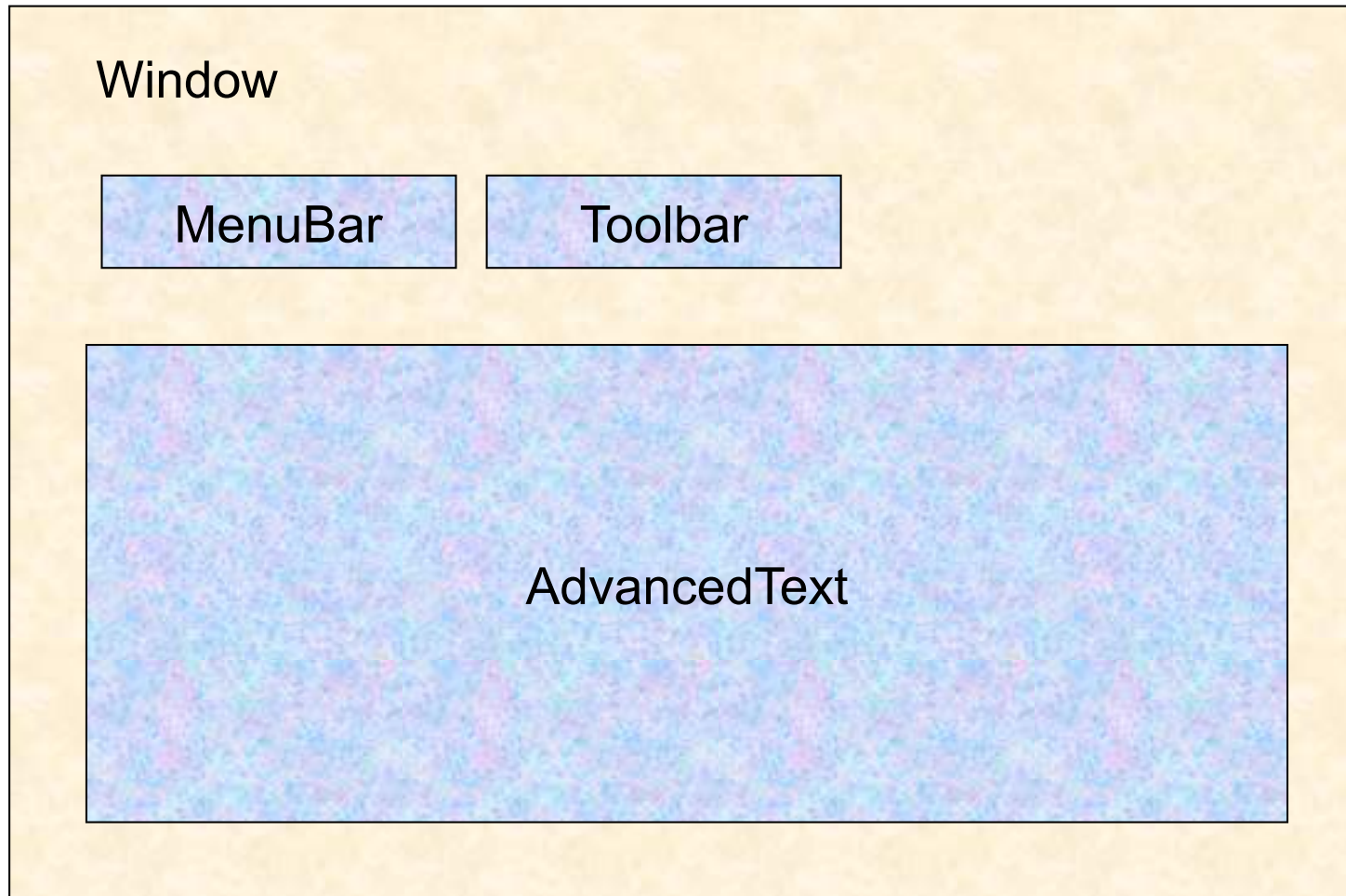
-  Must be more accessible to both novice and expert developers
-  Must be more modular
-  Must be more adaptable to new looks and feels
-  Must have tools to migrate from old framework
-  Must have comprehensive unit tests
-  Must be developed “Out In The Open”

# Pollock Philosophy and Goals

- ☞ Philosophy : XP (as much as possible)
  - ☞ Make it work,  
Make it right,  
Make it fast  
(in that order)
  - ☞ You aren't going to need it
  - ☞ The simplest thing that can possibly work
  - ☞ Let the code tell me what to do
  - ☞ Test first
  - ☞ Continuous integration



# Visual Launcher under Pollock



# The Path

## ☞ Beta 1: VW 7

- ☞ ApplicationWindow, Button, Label & Image
- ☞ Simple layouts (OriginExtent, Alignment)
- ☞ Win9x, MacOSX & Motif looks
- ☞ Build, Read & Write from XML and Array
- ☞ DTD for XML
- ☞ Window opening styles
- ☞ Full Trigger Event usage
- ☞ Builder, UserInterface, WidgetInventory

# The Path

## ☞ Beta 2 : VW 7.1

☞ CheckBox, Radio, InputField, List, Menu, DropDown (list and menu), AdvancedText, Grid (Table & Dataset combined), Dialog, Toolbar & Toolbar panes, TreeView, TabControl, GroupBox

☞ DragDrop

☞ Fractional Layouts

☞ UIPainter written in Pollock

# The Path

- Production 1 : VW 7.2
  - SubpaneHolder, Resizer, SpinButton, Divider, ClickWidget, Region, Progress, Slider
  - UIPainter, MenuEditor, ToolbarEditor
  - Translation & Conversion tools
  - Co-exist in image with existing framework
  - Existing framework default user interface creation tool
  - Last changes to existing framework
  - All widgets have at least basic functionality

# The Path

- Production 2 : VW “7.2”
  - All widgets have full capability
  - WinXP Look & Feel
  - Some internal tools use Pollock
  - Co-exist in image with existing framework
  - Pollock default new user interface creation tool
  - Existing framework “Soft” obsolete



# The Path

- ➡ Production 3 : VW “7.3”
  - ➡ All tools use Pollock
  - ➡ Existing framework “Hard” obsolete
  - ➡ Only Pollock in image
  - ➡ Existing framework fully loadable from Obsolete parcel

# The Metaphor

## ☞ Panes & Frames, Agents & Artists

### ☞ Pane

- ☞ Like a VisualComponent
- ☞ Some Panes have Subpanes
- ☞ Window is a Pane
- ☞ Screen is a Pane
- ☞ All subclass from AbstractPane

# The Metaphor

## ☞ Panes & Frames, Agents & Artists (cont.)

### ☞ Frame

- ☞ Like a Layout
- ☞ visible bounds
- ☞ displayable bounds
- ☞ origin
- ☞ [alignment]
- ☞ [fractional positions]
- ☞ [relation positions]
- ☞ [viewport]

# The Metaphor

## ☞ Panes & Frames, Agents & Artists

### ☞ Agent

- ☞ Offload behavior from controller
- ☞ Tell the artist when to draw
- ☞ Maintain most state (visibility, enablement, selected, pushedness, cursor position)
- ☞ Behavior for mouse and keyboard (keyboard processor, focus, “handler for mouse event”)

# The Metaphor

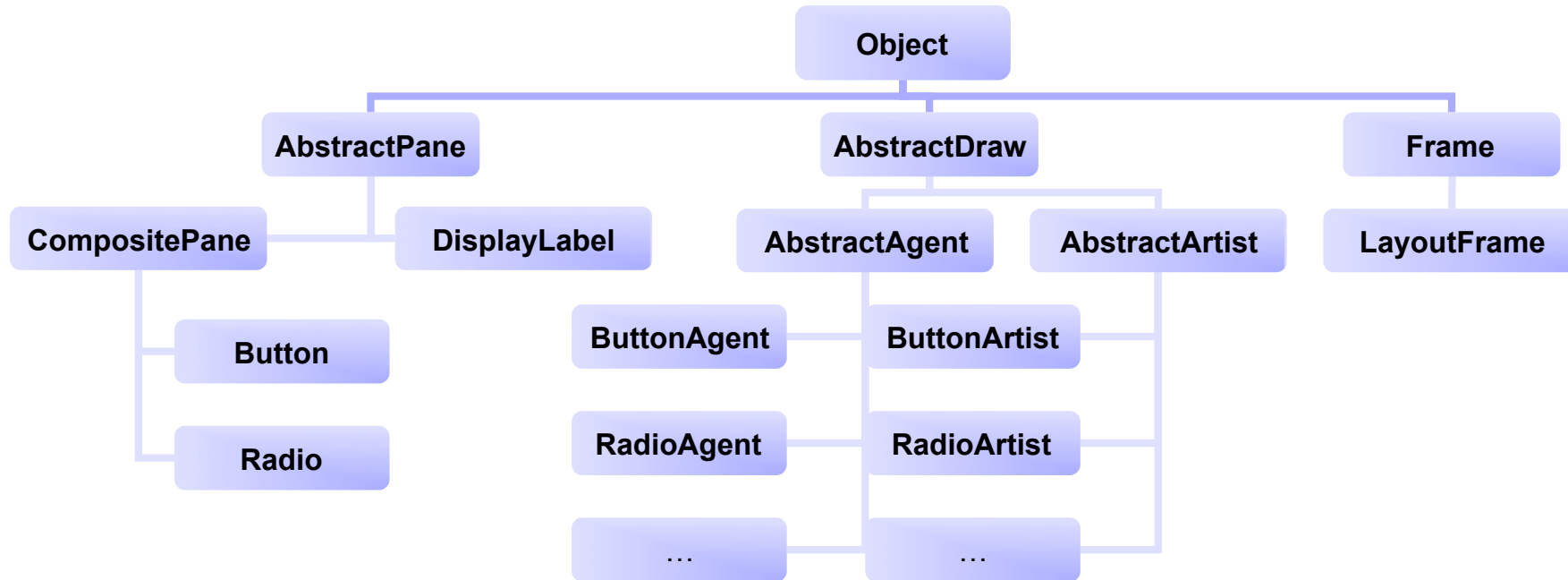
## ☞ Panes & Frames, Agents & Artists

### ☞ Artist

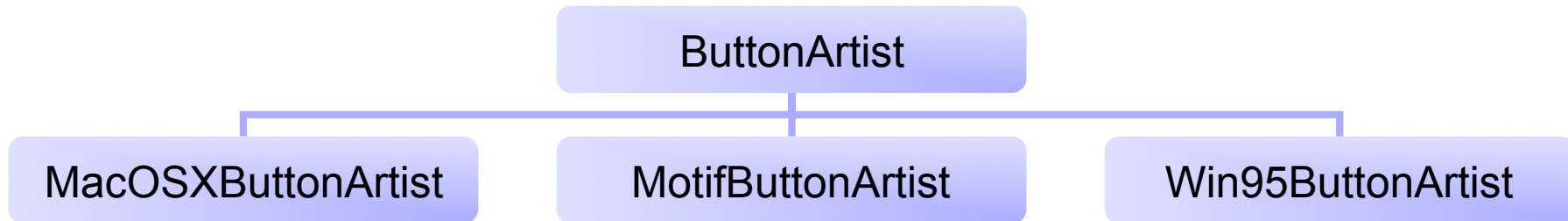
- ☞ Does all drawing

- ☞ Interacts with Agent and Pane to get information about what to draw, and when

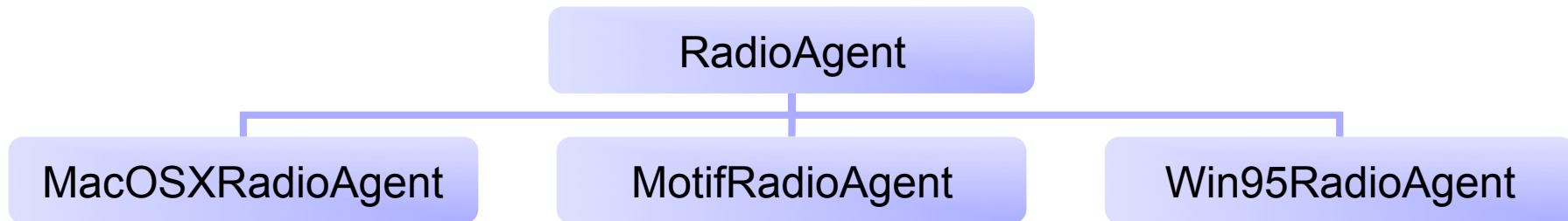
# The Basics



# The Basics : Looks



# The Basics : Feels





# The Basics

👉 Creating, configuring and opening a window

```
| window |  
window := Pollock.ScheduledWindow new.  
window frame: ((WindowFrame new)  
    maximumSize: 400 @ 400;  
    openingPosition: 100 @ 100;  
    yourself).  
window border: #etched.  
window open
```

# The Basics

## ☞ Creating, configuring and adding a Pane

```
| window radio |
```

```
window := self openWindow.
```

```
radio := Radio new.
```

```
radio frame: (OriginExtentFrame origin: 10 @ 10 extent: 30 @ 30).
```

```
window addComponent: radio.
```

# The Basics

☞ Labels are NOT part of panes, but can be subpanes

```
| window radio displayLabel labelModel |  
window := self openWindow.  
radio := self createRadio.  
displayLabel := Pollock.DisplayLabel new.  
labelModel := Pollock.Label string: 'Hi&Ho'.  
displayLabel label: labelModel.  
displayLabel frame origin: 0 @ 0.  
radio addComponent: displayLabel.  
window addComponent: radio
```

# The Basics

## ☞ CompositePanels

- ☞ Any subclass can have subpanels
- ☞ Can have any number of subpanels
- ☞ Can have any kind of subpanels

## ☞ Common subpanels

- ☞ Buttons : Labels and DisplayImages
- ☞ Radios : Labels

# The Basics

- Controllers are much more dumb
  - Controllers and Trackers still exist
  - New hierarchy
  - Forward all events to the Agent
- The Agent holds the Keyboard Processor
  - Because of compatibility, controller passes it to the Agent

# The Basics

## ☞ Decorations and Borders

☞ Decorations : Things inside a pane

☞ Scrollbars in an Advanced Text pane

☞ The “3D Look / Raised” portion of a Button

☞ Clip the drawable bounds

☞ Borders : Things that surround a pane

☞ Effectively clip the visible bounds

☞ Most panes have borders

☞ Some panes have decorations

☞ Borders and Decorations have Artists

# The Basics

☞ A Button with a Border (ridged) and a decoration:

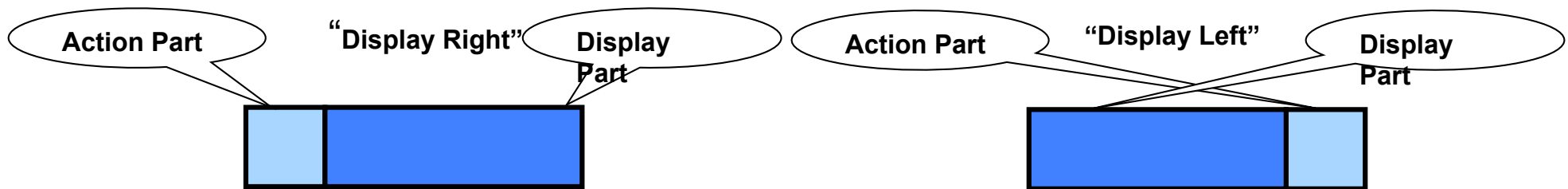


☞ A Button with a Border (ridged) – No decoration:



# The Basics

## ☞ ActionDisplay pane



## ☞ Examples:

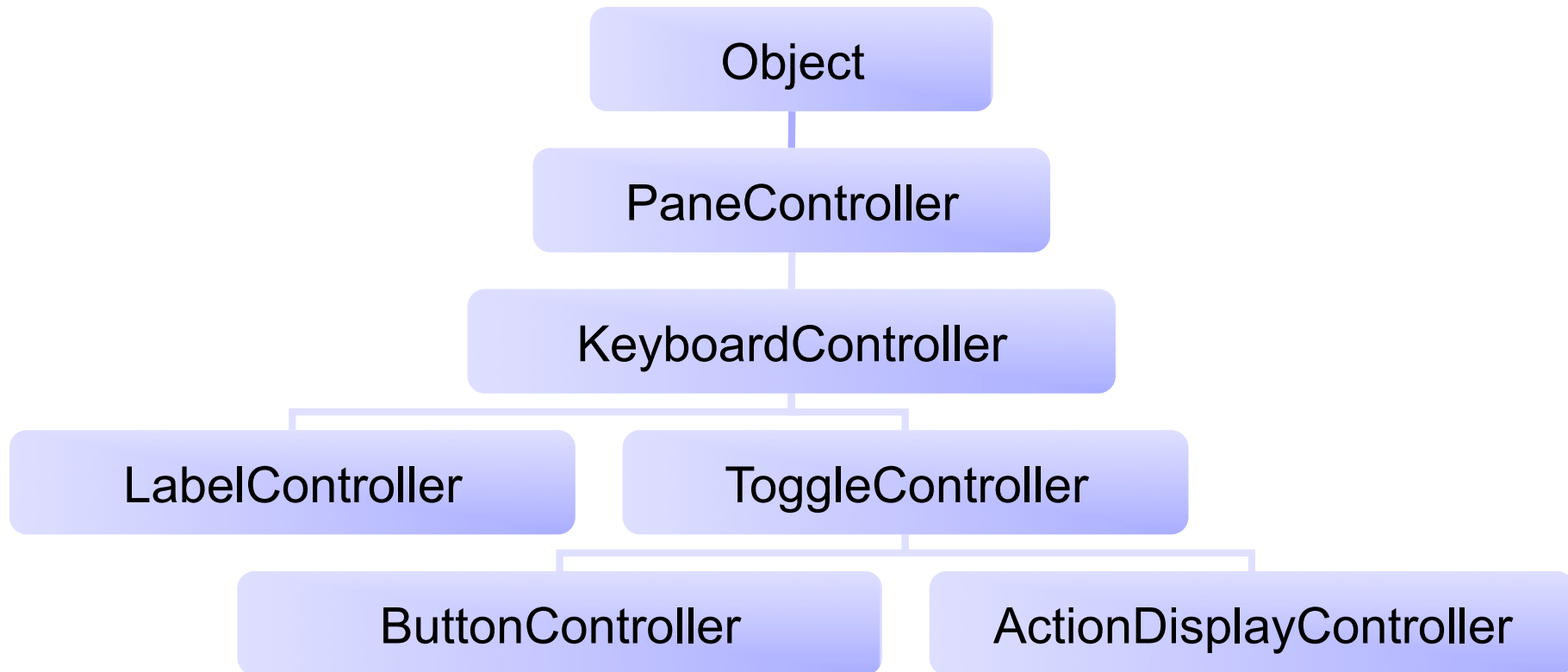
This block shows a vertical stack of GUI controls within a light brown frame. At the top is a 'Check Box' with an unchecked checkbox and the text 'Check Box'. Below it is a 'Radio' button with an unselected radio button and the text 'Radio'. A horizontal red line separates these from the controls below. Below the line is a 'Spin Button' with the text 'Spin Button' and two small red arrows (up and down). At the bottom is a 'Drop Down' menu with the text 'Drop Down' and a small red arrow pointing down. To the right of the frame, two white boxes with black borders point to the 'Check Box' and 'Drop Down' controls. The top box is labeled 'Display Right' and points to the 'Check Box'. The bottom box is labeled 'Display Left' and points to the 'Drop Down'.



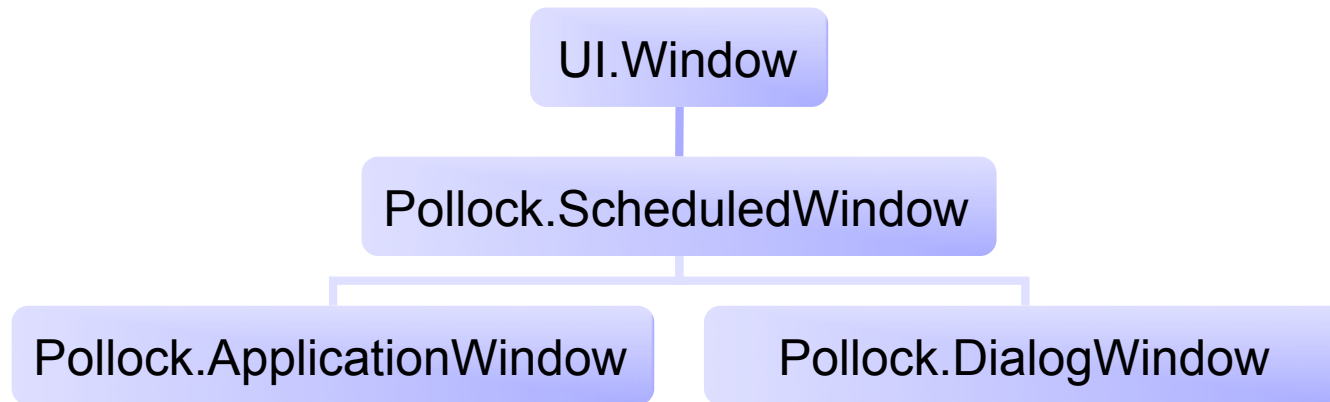
# The Basics

- Agents, Artists and Frames are pluggable at runtime
- Update/Change behavior is banned from the Pollock internals
- WidgetPolicy, BorderPolicy, LookPolicy
- FeelPolicy replaced by KeyboardPolicy
- New Controller hierarchy
- New ScheduledWindow hierarchy

# New Controller Hierarchy






# New ScheduledWindow hierarchy



# The Basics – Widget Changes

## Grid

-  Combines Dataset & Table
-  Will support Dataset models, as well as Table models
-  Will provide Multi-Column list mode

# The Basics – Widget Changes

## DropDown

 Combines DropDown, Combo and MenuButton

 Will support Lists as well as Menus as models

# The Basics – Unsure Future

## ☞ TabControl

☞ Will exist

☞ Should it add “Minor” tabs (right and/or bottom)?

☞ Should it add “Stacked” tabs?

## ☞ Notebook

☞ Not scheduled to exist (should it?)

# Pollock & The Trigger Events Framework

- ☞ The Trigger Events Framework
  - ☞ Fully fleshed out in VW 7
  - ☞ All events ultimately triggered by “views”
- ☞ Event Checking
  - ☞ Ambivalent vs. Strict
  - ☞ Object – Default is Ambivalent
  - ☞ Views & Panes – Default is Strict
- ☞ Using it now
  - ☞ In General
  - ☞ ApplicationModel


# Trigger Events – Using it now

## General

 eventTable

 canTriggerEvent: anEventNameSymbol

 hasActionForEvent: anEventNameSymbol

 actionListForEvent: anEventNameSymbol





# Trigger Events – Using it now

## ☞ General

- ☞ when: anEventNameSymbol do: aBlock
- ☞ when: anEventNameSymbol evaluate: anAction
- ☞ when: anEventNameSymbol send: aSelectorSymbol to: anObject
- ☞ when: anEventNameSymbol send: aSelectorSymbol to: anObject with: anArgumentObject
- ☞ when: anEventNameSymbol send: aSelectorSymbol to: anObject with: firstArgumentObject with: secondArgumentObject
- ☞ when: anEventNameSymbol send: aSelectorSymbol to: anObject withArguments: anArgumentCollection
- ☞ whenAny:....

# Trigger Events – Using it now

## General

-  `removeAction: anAction forEvent: anEventNameSymbol`
-  `removeActionsForEvent: anEventNameSymbol`

# Trigger Events – Using it now

## ☞ ApplicationModel

- ☞ widget: aWidgetIDSymbol when: anEventSymbol do: aBlock
- ☞ widget: aWidgetIDSymbol when: anEventSymbol evaluate: anAction
- ☞ widget: aWidgetIDSymbol when: anEventSymbol send: anAction to: anObject
- ☞ widget: aWidgetIDSymbol when: anEventSymbol send: anAction to: anObject with: anArgument
- ☞ widget: aWidgetIDSymbol when: anEventSymbol send: anAction to: anObject with: firstArgument with: secondArgument
- ☞ widget: aWidgetIDSymbol when: anEventSymbol send: anAction to: anObject withArguments: aCollection

# Trigger Events – Using it now

👉 New behavior in `Object>>changed:with:`

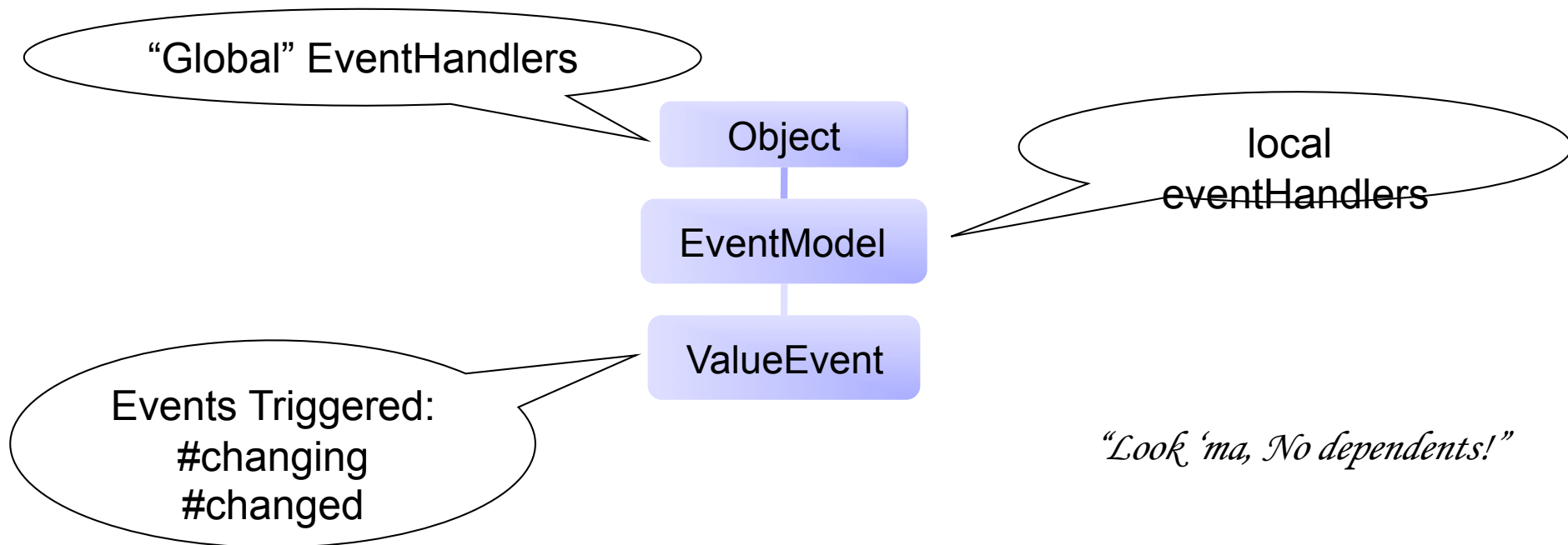
```
changed: anAspectSymbol with: aParameter
```

"The receiver changed. The change is denoted by the argument `anAspectSymbol`. Usually the argument is a `Symbol` that is part of the dependent's change protocol, that is, some aspect of the object's behavior, and `aParameter` is additional information. Inform all of the dependents."

```
self myDependents update: anAspectSymbol with: aParameter from: self.  
anAspectSymbol isSymbol ifTrue: [self triggerEvent: anAspectSymbol]
```

# Models & ValueEvents

- ❏ The Model Hierarchy will not be used
- ❏ Replaced by the EventModel Hierarchy



# Pollock – New DragDrop System

## State driven engine

For single select lists, the following are the rule states.

- 1) If you click on an unselected item, and you do NOT move the mouse more than 2@2 while you have the mouse down, nothing happens until you release the mouse, at which time the target item is selected.
- 2) If you click on an unselected item, and you DO move the mouse more than 2@2, the target item is selected, and you are put into Drag mode for that item if it is turned on, if not, the selection changes as you drag the mouse over successive unselected items.
- 3) If you click on a selected item and do NOT move the mouse more than 2@2, nothing happens until you release the mouse, at which time the target item is unselected.
- 4) If you click on a selected item and DO move the mouse more than 2@2, you are put into Drag mode for that item if it is turned on, if not, the selection changes as you drag the mouse over successive unselected items (as in rule #2)

# Pollock – New DragDrop System

## State driven engine (continued)

For multi select lists, the following are the rule states.

- 1) If you click on an unselected item, and you do NOT move the mouse more than 2@2 while you have the mouse down, nothing happens until you release the mouse, at which time the target item is selected. (Rules of Shift & Ctrl apply to "items" selected)
- 2) If you click on an unselected item, and you DO move the mouse more than 2@2, the current item is selected and any additional items you drag the mouse over get added to the current selections. (Rules of Shift & Ctrl apply to "items" selected).
- 3) If you click on a selected item, and you do NOT move the mouse more than 2@2 while you have the mouse down, nothing happens until you release the mouse, at which time the item is unselected. (Rules of Shift & Ctrl apply to "items" selected/unselected)
- 4) If you click on a selected item, and you DO move the mouse more than 2@2, you are now in Drag mode if it is turned on (Rules of Ctrl & Shift apply to move/copy mode of Drag, and not to selection modes), if not, rule #2 kicks in.

# Pollock – New DragDrop System

- ☞ Trigger Event communication (based on VSE design)
- ☞ “Select When Down” dies a miserable death
- ☞ Configurable
  - ☞ Default – Right Button
  - ☞ Options : Left Button. Modifier Key



# Pollock – Edit Keys

☞ “FeelPolicy” and the dispatch table as we know it, goes the way of the dodo

## ☞ KeyboardPolicy

☞ Based on MagicKeys from Roel Wuyts (Thank You!)

☞ Assignable by “Application”:

☞ Transcript – EditPolicy

☞ Refactoring Browser – CodePolicy

☞ MyApplication – MyApplicationPolicy

☞ Can have “Platform” versions

☞ WindowsEditPolicy, WindowsCodePolicy

☞ MacEditPolicy, MacMyApplicationPolicy

# Coding with the future in mind

## ☞ ApplicationModel

☞ Don't write "self builder componentAt:"

☞ Use new (VW 7) protocols

☞ #mainWindow

☞ #windowMenuBar

☞ #controllerAt: aWidgetIDSymbol

☞ #widgetAt: aWidgetIDSymbol

☞ #wrapperAt: aWidgetIDSymbol

# Coding with the future in mind

- ☞ Use the Trigger Event system instead of update/change
- ☞ Stay away from the Smalltalk (“Default”) look
- ☞ Stay away from the GUI guts, but if you must: SUBCLASS!

# Pollock Tools

- ➡ ApplicationModel conversion tool
- ➡ WindowSpec conversion tool
- ➡ Enhanced MenuEditor
- ➡ CoolImage replaces ImageEditor (Thank You Travis Griggs!)
- ➡ Toolbar Editor



# Building A New Widget

- ☞ Our Example : The Region Pane
- ☞ What we have to do
  1. Write a Test
  2. Create the Pane subclass
  3. Write a Test
  4. Create the Agent subclass
  5. Write a Test
  6. Create the Artist subclass

# Pane – Required Methods

☞ agentClass

☞ artistClass

☞ defaultControllerClass

☞ frameClass

☞ getController

(because defaultControllerClass is nil)

☞ outerExtentChanged: aPoint

# First Uses

openWindowWithRegion

self openWindow.

region := Region new.

region frame: (OriginExtentFrame origin: 10 @ 10 extent: 100 @ 100).

window addComponent: region

openWindowWithRegionWithBorder

self openWindow.

region := Region new.

region frame: (OriginExtentFrame origin: 10 @ 10 extent: 100 @ 100).

region border: #ridged.

window addComponent: region

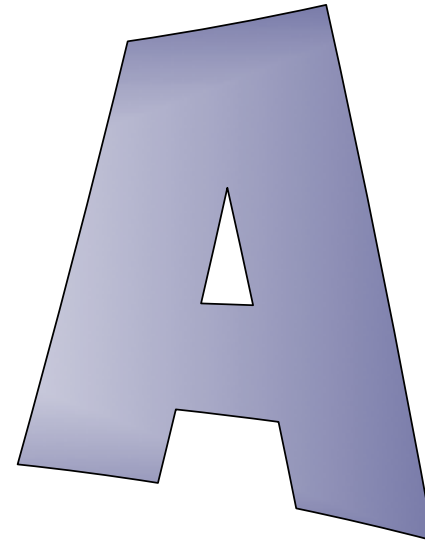
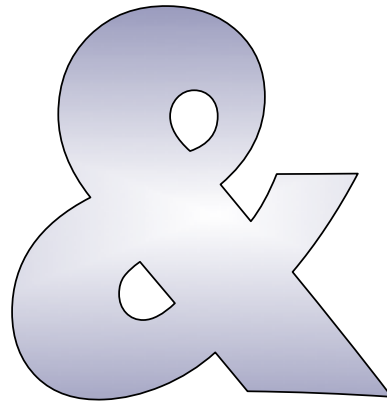
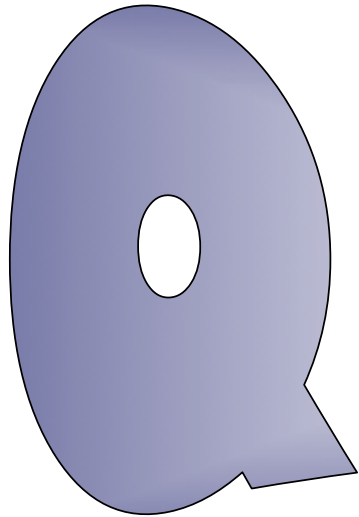
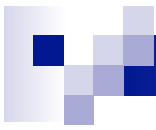


# Fleshing It Out

- ☞ rectangle instance variable (Pane)
- ☞ drawOn: (Artist)
- ☞ background (Artist & Pane)
- ☞ lineSize, lineColor, fillColor (Artist & Pane)
- ☞ visible, beVisible: (Agent & Pane)

# On your own

- ☞ Platform looks (if any)
- ☞ Additional shapes





**© 2002 Cincom Systems, Inc.  
All Rights Reserved  
Developed in the U.S.A.**

CINCOM, , and The Smart Choice are trademarks or registered trademarks of Cincom Systems, Inc

All other trademarks belong to their respective companies.