Scrolling, Swiping, Dragging

Now with more animation!

Session 115

Raleigh Ledet
Cocoa Software Engineer

Scrolling, Swiping, Dragging

New

Only on Mac OS

- Scrolling
  - Scrollers
  - Elastic scrolling (aka rubber-banding)
- Fluid swiping
- Multi-image dragging

Scrolling

A content-focused redesign

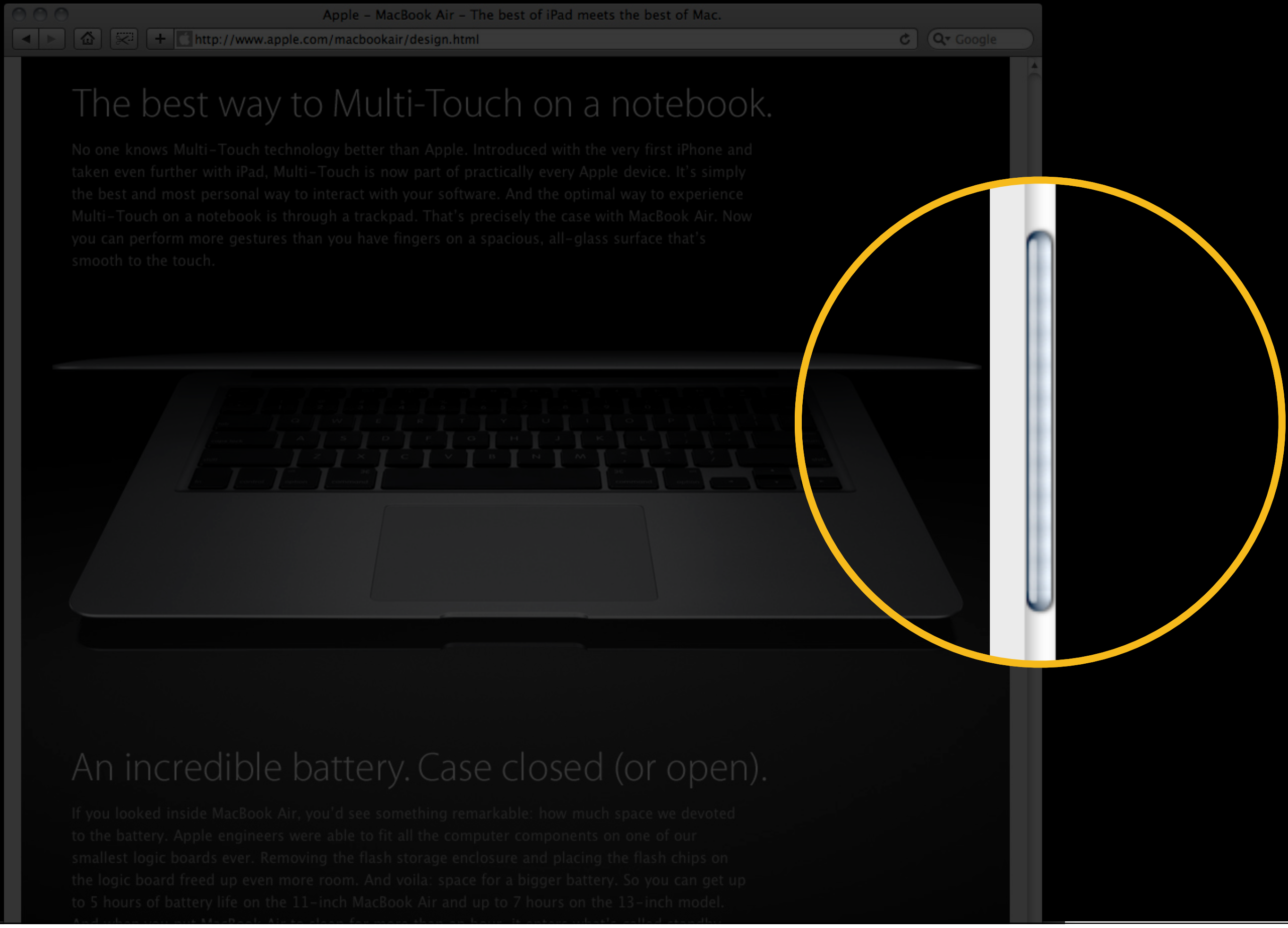
Lion’s New Scrollers

A content-focused redesign

Troy Stephens
Application Frameworks Engineer

Scrollers

Then and now


Apple – MacBook Air – The best of iPad meets the best of Mac.
http://www.apple.com/macbookair/design.html
Google

The best way to Multi-Touch on a notebook.

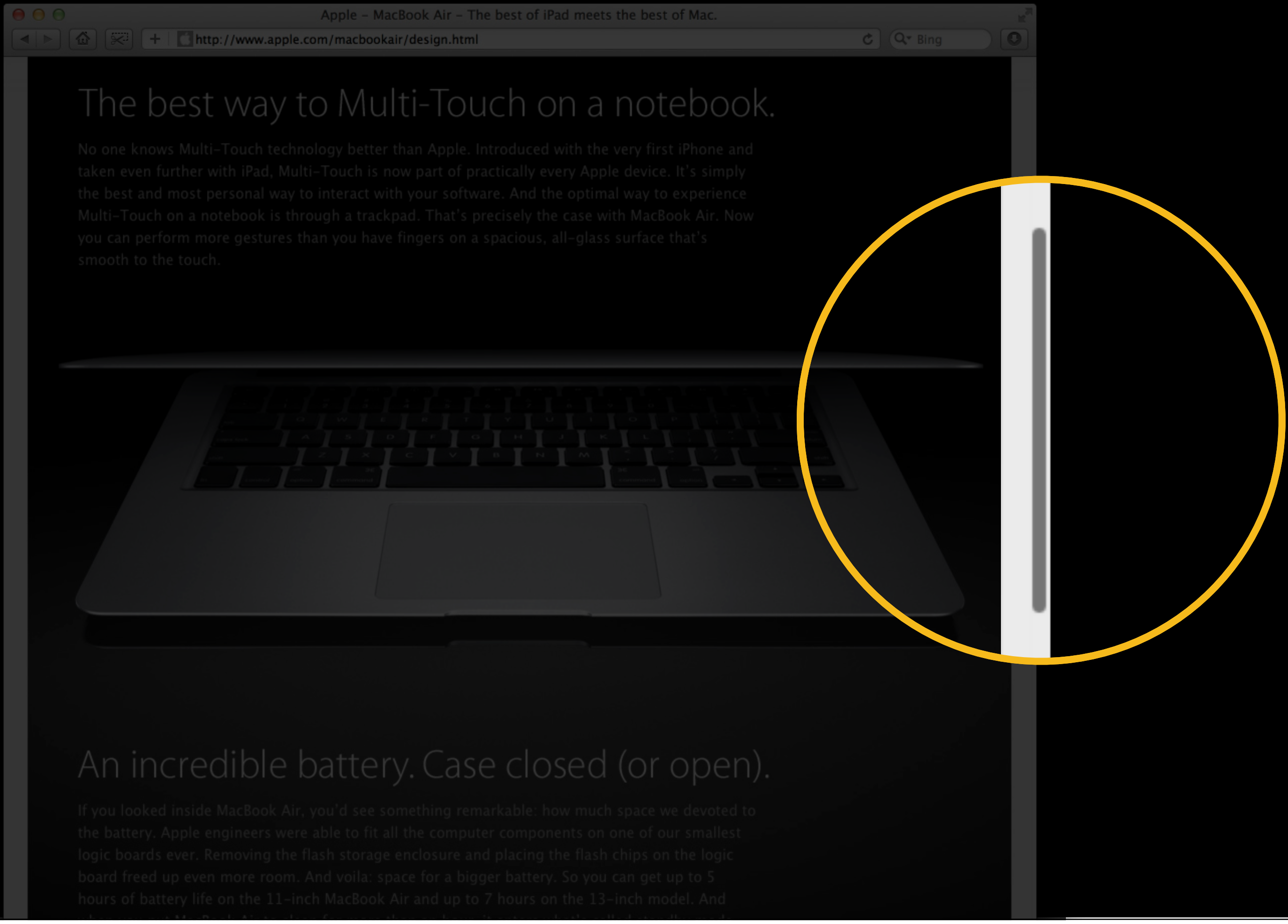
No one knows Multi-Touch technology better than Apple. Introduced with the very first iPhone and taken even further with iPad, Multi-Touch is now part of practically every Apple device. It's simply the best and most personal way to interact with your software. And the optimal way to experience Multi-Touch on a notebook is through a trackpad. That's precisely the case with MacBook Air. Now you can perform more gestures than you have fingers on a spacious, all-glass surface that's smooth to the touch.

An incredible battery. Case closed (or open).

If you looked inside MacBook Air, you'd see something remarkable: how much space we devoted to the battery. Apple engineers were able to fit all the computer components on one of our smallest logic boards ever. Removing the flash storage enclosure and placing the flash chips on the logic board freed up even more room. And voila: space for a bigger battery. So you can get up to 5 hours of battery life on the 11-inch MacBook Air and up to 7 hours on the 13-inch model.

Scrollers

Then and now


Apple - MacBook Air - The best of iPad meets the best of Mac.
http://www.apple.com/macbookair/design.html
Bing

The best way to Multi-Touch on a notebook.

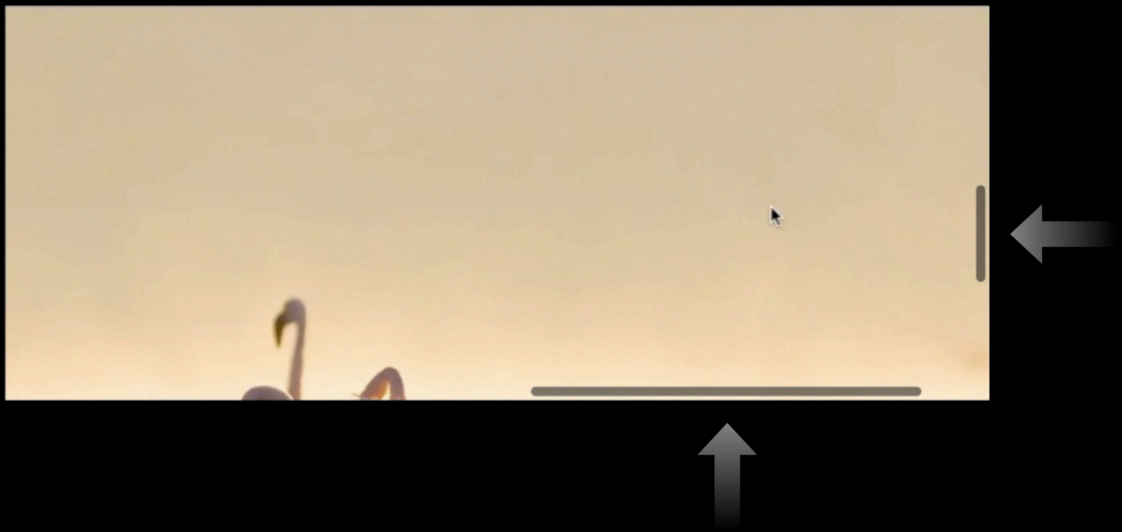
No one knows Multi-Touch technology better than Apple. Introduced with the very first iPhone and taken even further with iPad, Multi-Touch is now part of practically every Apple device. It's simply the best and most personal way to interact with your software. And the optimal way to experience Multi-Touch on a notebook is through a trackpad. That's precisely the case with MacBook Air. Now you can perform more gestures than you have fingers on a spacious, all-glass surface that's smooth to the touch.

An incredible battery. Case closed (or open).

If you looked inside MacBook Air, you'd see something remarkable: how much space we devoted to the battery. Apple engineers were able to fit all the computer components on one of our smallest logic boards ever. Removing the flash storage enclosure and placing the flash chips on the logic board freed up even more room. And voila: space for a bigger battery. So you can get up to 5 hours of battery life on the 11-inch MacBook Air and up to 7 hours on the 13-inch model. And

Lion's Two Scroller Styles

NSScrollerStyleOverlay



Lion's Two Scroller Styles

NSScrollerStyleOverlay

Lion’s Two Scroller Styles

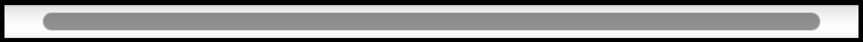
NSScrollerStyleOverlay

Lion’s Two Scroller Styles

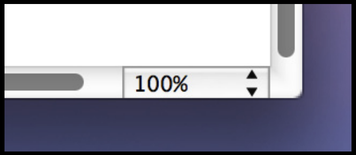
NSScrollerStyleOverlay

Lion's Two Scroller Styles

NSScrollerStyleLegacy



- For compatibility, and to accommodate user preferences
- Used when user asks for scrollers to be shown "Always"
- Used when AppKit detects an accessory view in a ScrollView's margins

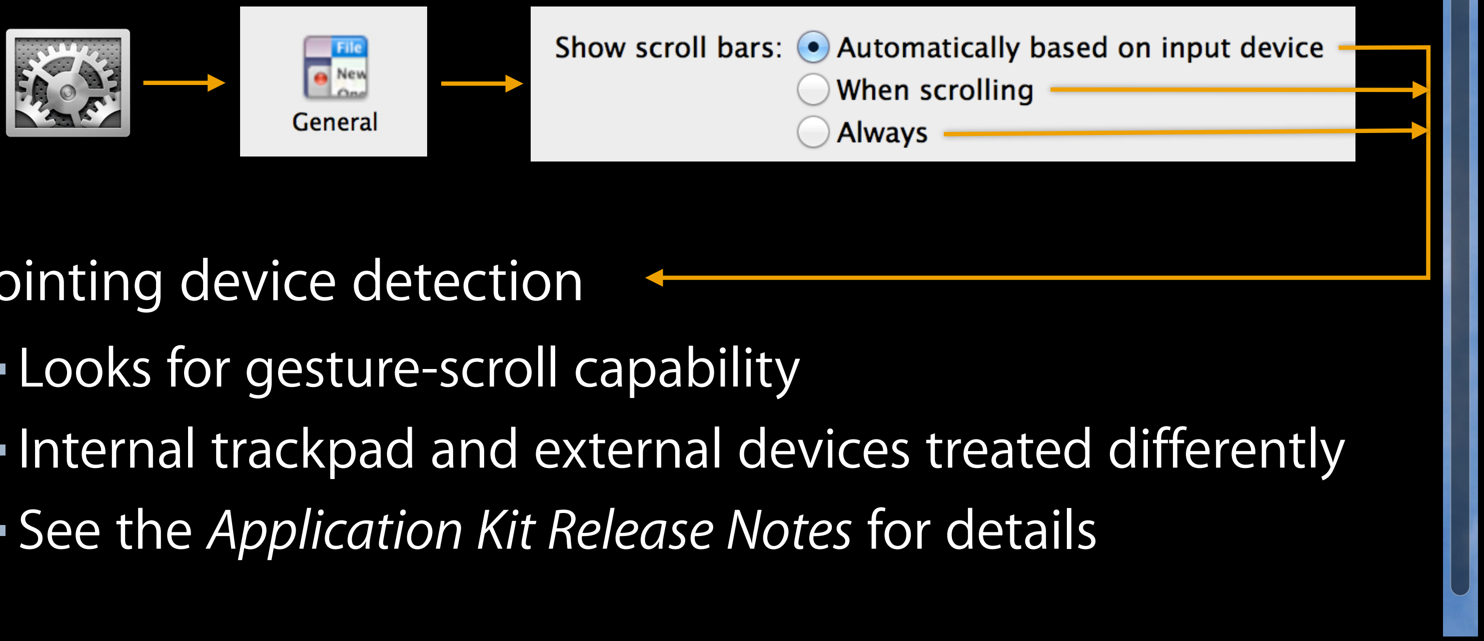


- Used when AppKit is not sure an NSScroller subclass is compatible

Scrollers

How scroller style is determined on Lion

- User's "General" preference



Show scroll bars: [✓ Automatically based on input device] When scrolling Always

- Pointing device detection
  - Looks for gesture-scroll capability
  - Internal trackpad and external devices treated differently
  - See the *Application Kit Release Notes* for details

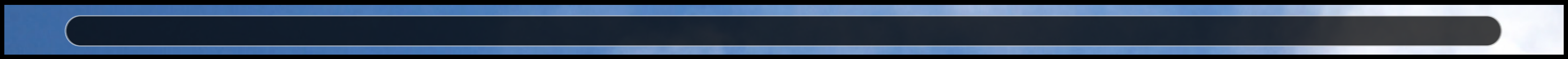
Scrollers

Reacting to scroller style changes

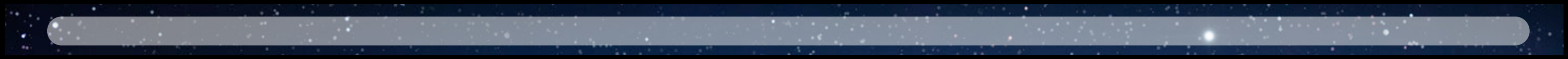
- Apps must be prepared to work with user's choice of scroller style
- AppKit updates all NSScrollViews automatically
  - Sends `-setScrollerStyle:` to each NSScrollView instance
- If you have additional code that needs to respond to a style change:
  - NSScroller `+preferredScrollerStyle` returns the preferred style
  - `NSPreferredScrollerStyleDidChangeNotification`

Scrollers

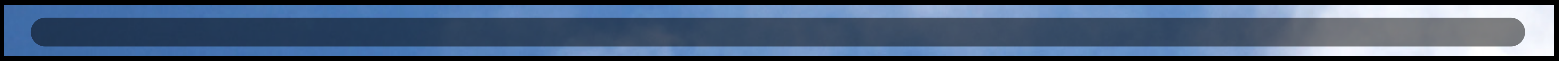
Overlay scroller knob styles



```
[scrollView setScrollerKnobStyle:NSScrollerKnobStyleDefault];
```



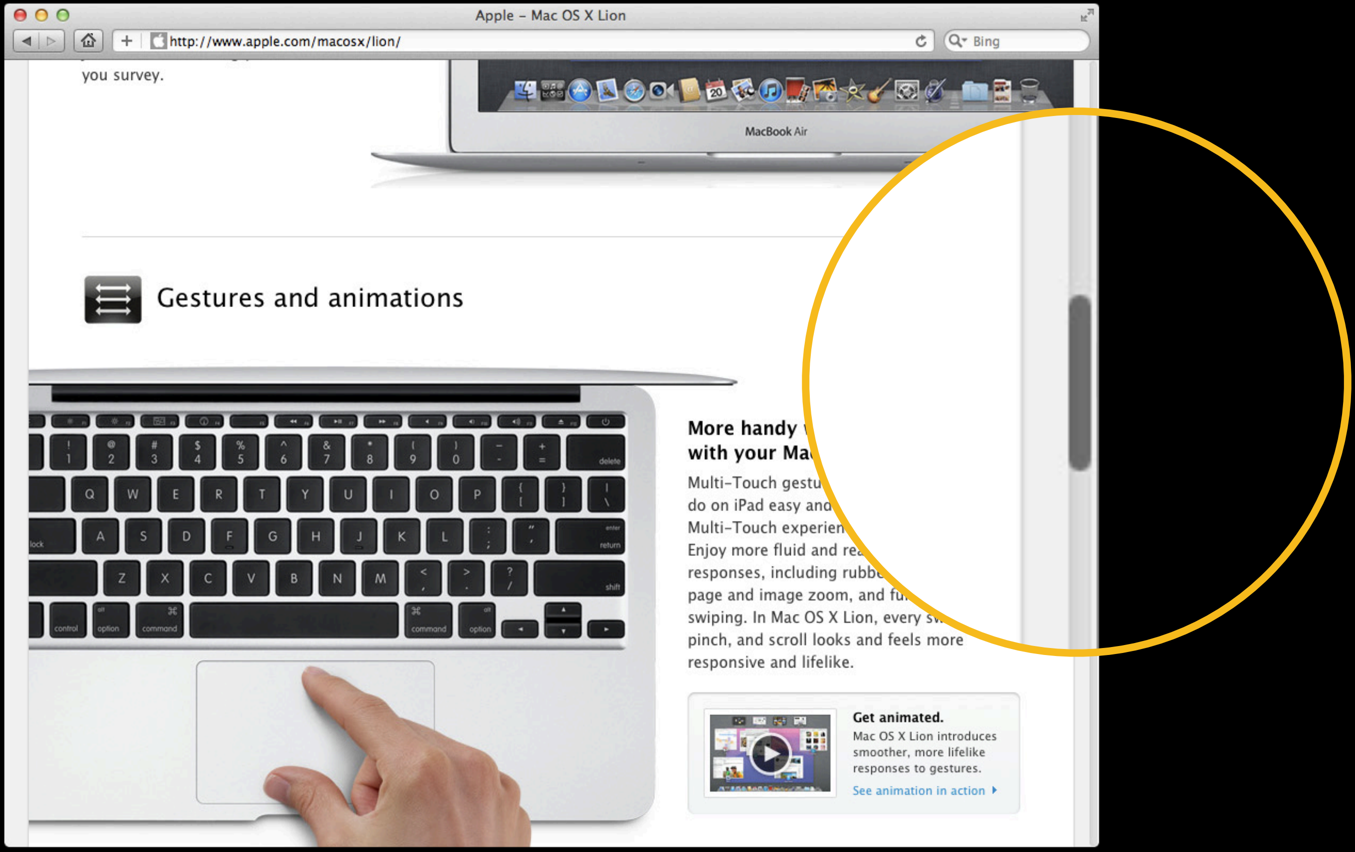
```
[scrollView setScrollerKnobStyle:NSScrollerKnobStyleLight];
```



```
[scrollView setScrollerKnobStyle:NSScrollerKnobStyleDark];
```

Scrollers

Overlay scroller knob styles

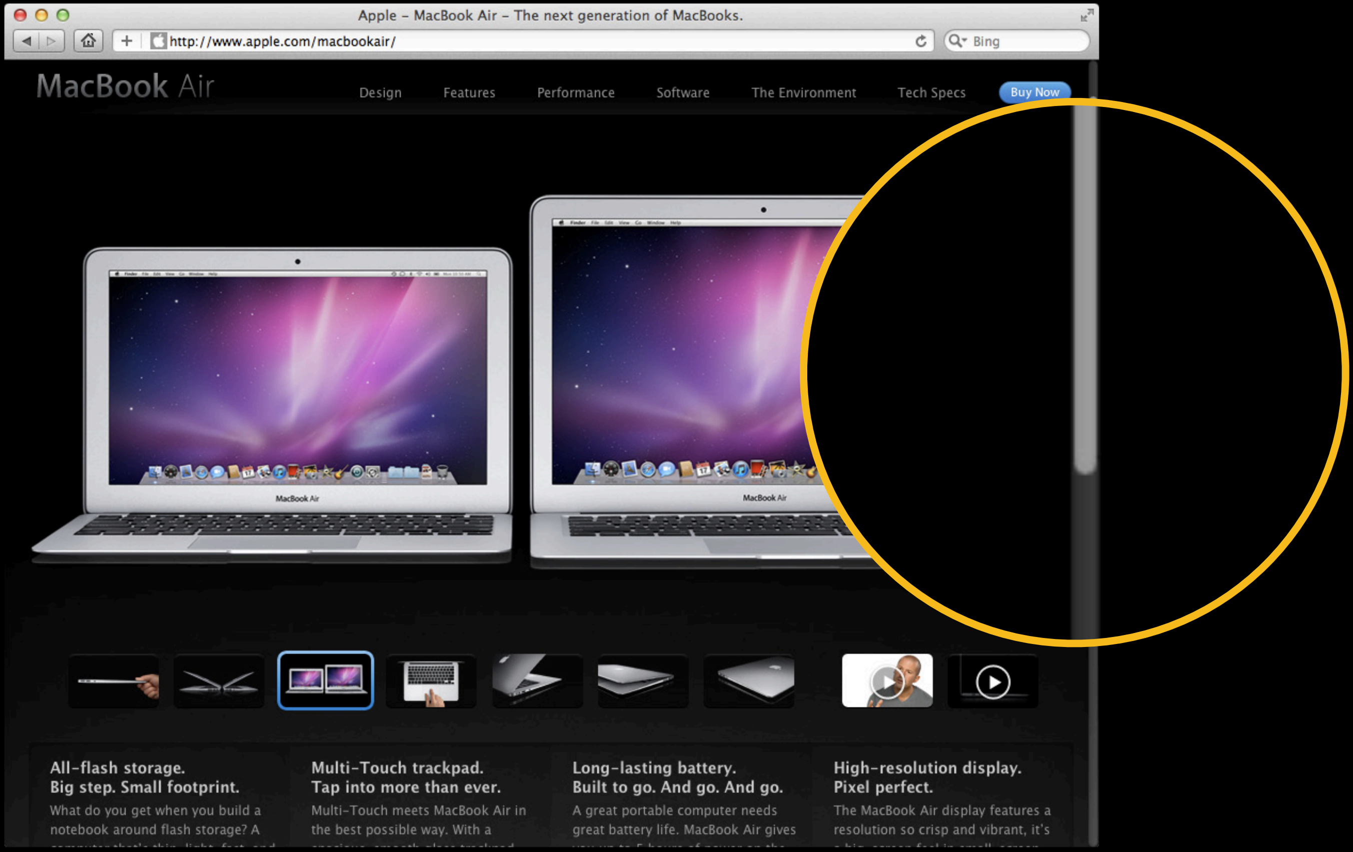


More handy
with your Ma
Multi-Touch gestu
do on iPad easy and
Multi-Touch experien
Enjoy more fluid and rea
responses, including rubbe
page and image zoom, and fu
swiping. In Mac OS X Lion, every sv
pinch, and scroll looks and feels more
responsive and lifelike.

Get animated.
Mac OS X Lion introduces smoother, more lifelike responses to gestures.
See animation in action

Scrollers

Overlay scroller knob styles



Apple – MacBook Air – The next generation of MacBooks.

http://www.apple.com/macbookair/

All-flash storage. Big step. Small footprint.
What do you get when you build a notebook around flash storage? A

Multi-Touch trackpad. Tap into more than ever.
Multi-Touch meets MacBook Air in the best possible way. With a

Long-lasting battery. Built to go. And go. And go.
A great portable computer needs great battery life. MacBook Air gives

High-resolution display. Pixel perfect.
The MacBook Air display features a resolution so crisp and vibrant, it's

Scrollers

Making scroller subclasses Overlay-scroller-compatible

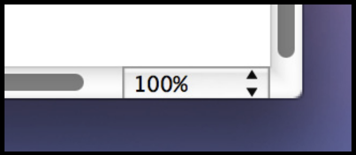
- Add this method override to your NSScroller subclass:

```
@implementation MyCustomScroller
+ (BOOL)isCompatibleWithOverlayScrollers {
    return self == [MyCustomScroller class];
}
@end
```

- Requirements:
  - Override `-drawKnob` and `-drawKnobSlotInRect:highlight:`, not `-drawRect:`
  - Override `-testPart:` and `-trackKnob:`, not `-mouseDown:`
  - OK with empty rects for the arrow parts
  - OK with potentially different size and layout metrics

Scrollers

Migrating “accessory” views



- Lion’s “Overlay” scrollers give space back to the user’s content
  - Composited atop the content area when shown
  - Do not interfere with user-content interaction when hidden
- What to do with accessory views?
  - Cannot leave them obscuring the user’s content
  - Hiding them along with scrollers probably is not right
  - Therefore, accessory views cause fallback to Legacy scroller style
  - To get Overlay scrollers, move your accessory view UI elsewhere

Scrollers

API deprecations and usage improvements

- AppKit now consistently uses NSScroller’s `-rectForPart:` method!
- NSScroller methods and constants dealing with arrows are deprecated
- Choice of blue/graphite NSControlTint no longer affects scrollers

Scrollers

Layout methods

```
@interface NSScroller
+ (CGFloat)scrollerWidth
+ (CGFloat)scrollerWidthForControlSize:(NSSize)controlSize

+ (CGFloat)scrollerWidthForControlSize:(NSSize)controlSize
                          scrollerStyle:(NSScrollerStyle)scrollerStyle
```

Scrollers

Layout methods

```
@interface NSScrollView
+ (NSSize)frameSizeForContentSize:(NSSize)cSize
            hasHorizontalScroller:(BOOL)hFlag
              hasVerticalScroller:(BOOL)vFlag
                       borderType:(NSBorderType)aType
```

Layout Methods

NSScrollView

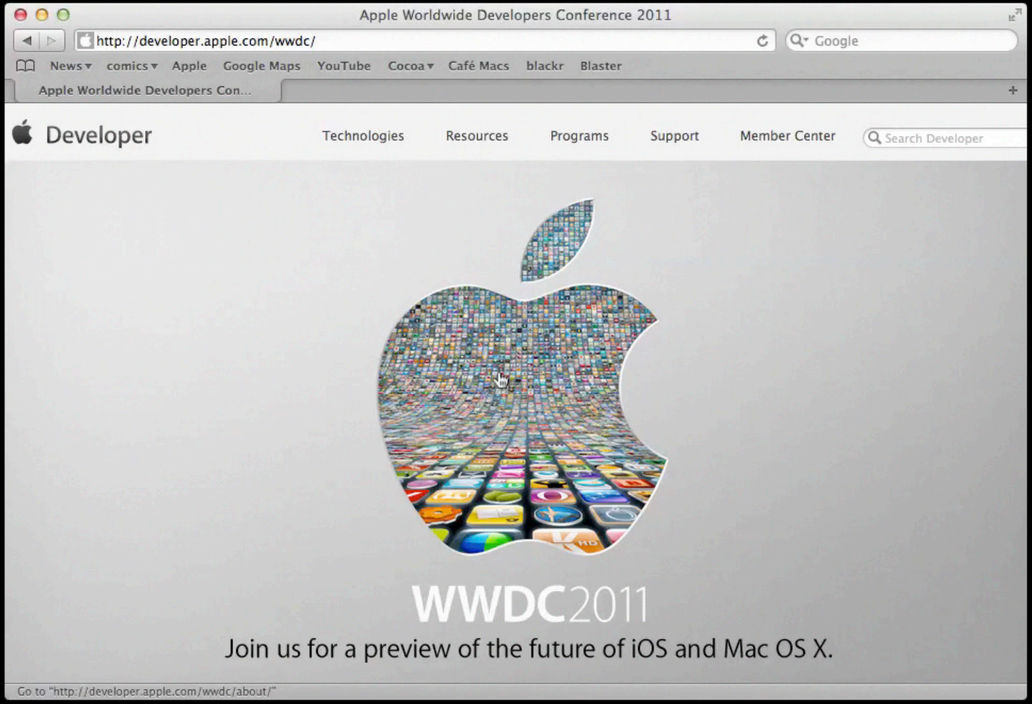
```
@interface NSScrollView
+ (NSSize)frameSizeForContentSize:(NSSize)cSize
          horizontalScrollerClass:(Class)horizontalScrollerClass
            verticalScrollerClass:(Class)verticalScrollerClass
                       borderType:(NSBorderType)aType
                      controlSize:(NSControlSize)controlSize
                    scrollerStyle:(NSScrollerStyle)scrollerStyle
```

Elastic Scrolling

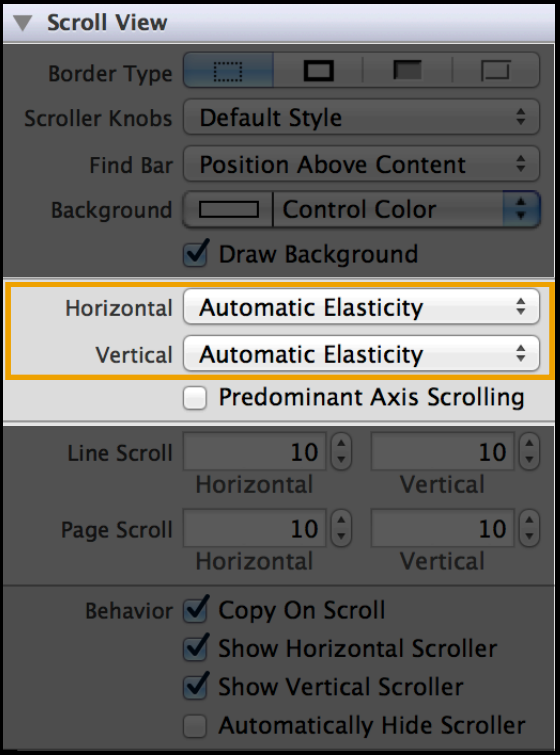
A content-focused redesign

Raleigh Ledet

Elastic Scrolling


Apple Worldwide Developers Conference 2011
http://developer.apple.com/wwdc/
Google
News comics Apple Google Maps YouTube Cocoa Café Macs blackr Blaster
Apple Worldwide Developers Con...
Developer
Technologies
Resources
Programs
Support
Member Center
Search Developer
WWDC2011
Join us for a preview of the future of iOS and Mac OS X.
Go to "http://developer.apple.com/wwdc/about/"

Elastic Scrolling


Scroll View

Border Type

Scroller Knobs: Default Style

Find Bar: Position Above Content

Background: Control Color

[✓] Draw Background

Horizontal: Automatic Elasticity

Vertical: Automatic Elasticity

[ ] Predominant Axis Scrolling

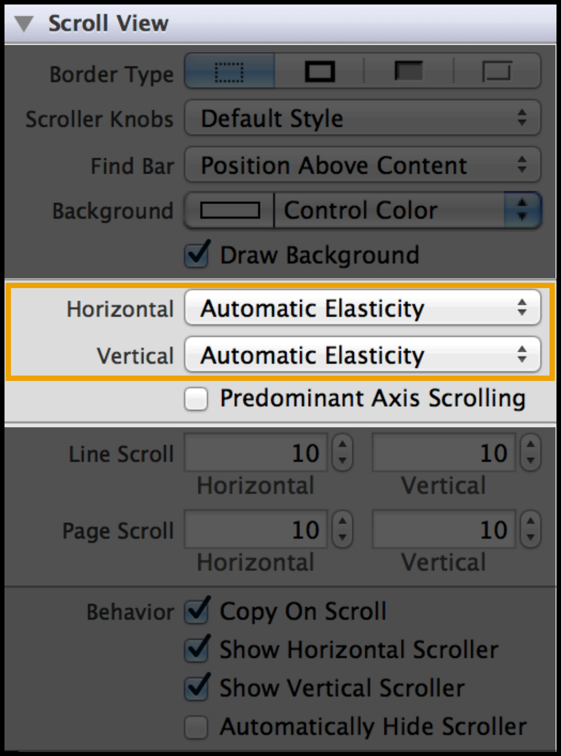
Line Scroll: 10 Horizontal, 10 Vertical

Page Scroll: 10 Horizontal, 10 Vertical

Behavior:
[✓] Copy On Scroll
[✓] Show Horizontal Scroller
[✓] Show Vertical Scroller
[ ] Automatically Hide Scroller

```
-(NSScrollElasticity)horizontalScrollElasticity;
-(NSScrollElasticity)verticalScrollElasticity;
```

Elastic Scrolling


Scroll View

Border Type

Scroller Knobs: Default Style

Find Bar: Position Above Content

Background: Control Color

[✓] Draw Background

Horizontal: Automatic Elasticity

Vertical: Automatic Elasticity

[ ] Predominant Axis Scrolling

Line Scroll: 10 Horizontal, 10 Vertical

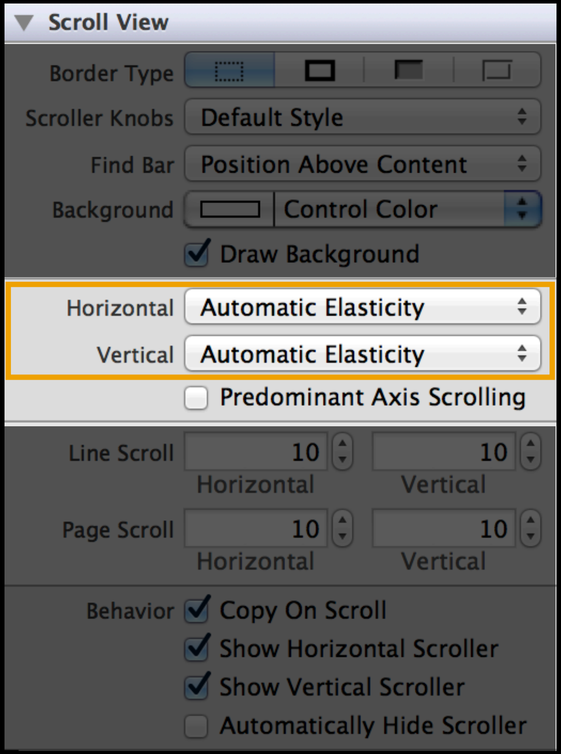
Page Scroll: 10 Horizontal, 10 Vertical

Behavior:
[✓] Copy On Scroll
[✓] Show Horizontal Scroller
[✓] Show Vertical Scroller
[ ] Automatically Hide Scroller

```
-(NSScrollElasticity)horizontalScrollElasticity;
-(NSScrollElasticity)verticalScrollElasticity;
```

```
NSScrollElasticityAutomatic
```

Elastic Scrolling



Scroll View

Border Type

Scroller Knobs: Default Style

Find Bar: Position Above Content

Background: Control Color

[✓] Draw Background

Horizontal: Automatic Elasticity

Vertical: Automatic Elasticity

[ ] Predominant Axis Scrolling

Line Scroll: 10 Horizontal, 10 Vertical

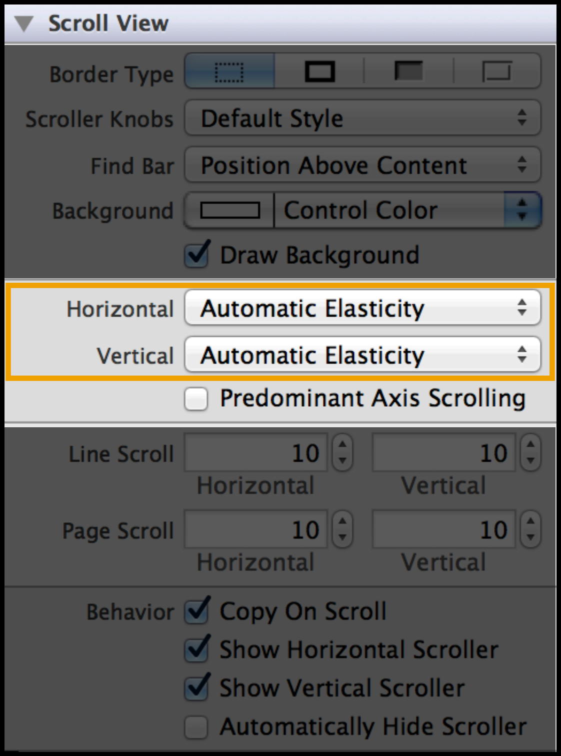
Page Scroll: 10 Horizontal, 10 Vertical

Behavior:
[✓] Copy On Scroll
[✓] Show Horizontal Scroller
[✓] Show Vertical Scroller
[ ] Automatically Hide Scroller

```
-(NSScrollElasticity)horizontalScrollElasticity;
-(NSScrollElasticity)verticalScrollElasticity;
```

```
NSScrollElasticityAutomatic
NSScrollElasticityNone
```

Elastic Scrolling



Scroll View

Border Type

Scroller Knobs: Default Style

Find Bar: Position Above Content

Background: Control Color

✓ Draw Background

Horizontal: Automatic Elasticity

Vertical: Automatic Elasticity

☐ Predominant Axis Scrolling

Line Scroll: 10 Horizontal, 10 Vertical

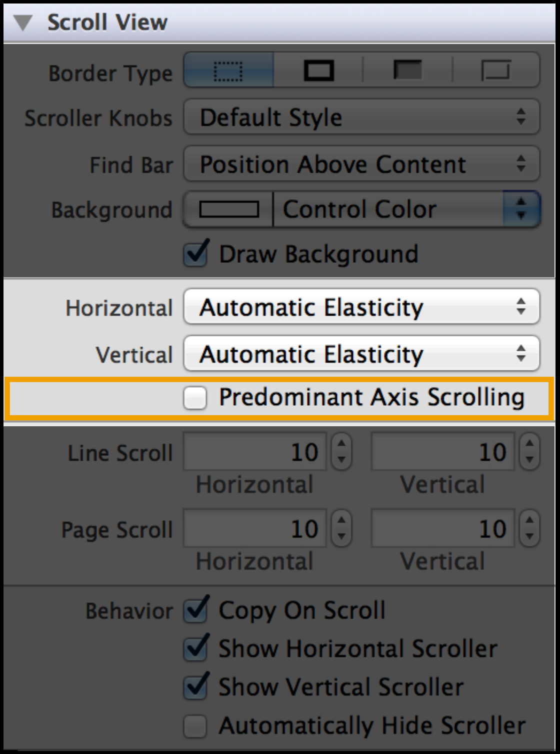
Page Scroll: 10 Horizontal, 10 Vertical

Behavior:
✓ Copy On Scroll
✓ Show Horizontal Scroller
✓ Show Vertical Scroller
☐ Automatically Hide Scroller

```
-(NSScrollElasticity)horizontalScrollElasticity;
-(NSScrollElasticity)verticalScrollElasticity;
```

```
NSScrollElasticityAutomatic
NSScrollElasticityNone
NSScrollElasticityAllowed
```

Elastic Scrolling



Scroll View

Border Type

Scroller Knobs: Default Style

Find Bar: Position Above Content

Background: Control Color

✓ Draw Background

Horizontal: Automatic Elasticity

Vertical: Automatic Elasticity

☐ Predominant Axis Scrolling

Line Scroll: Horizontal 10, Vertical 10

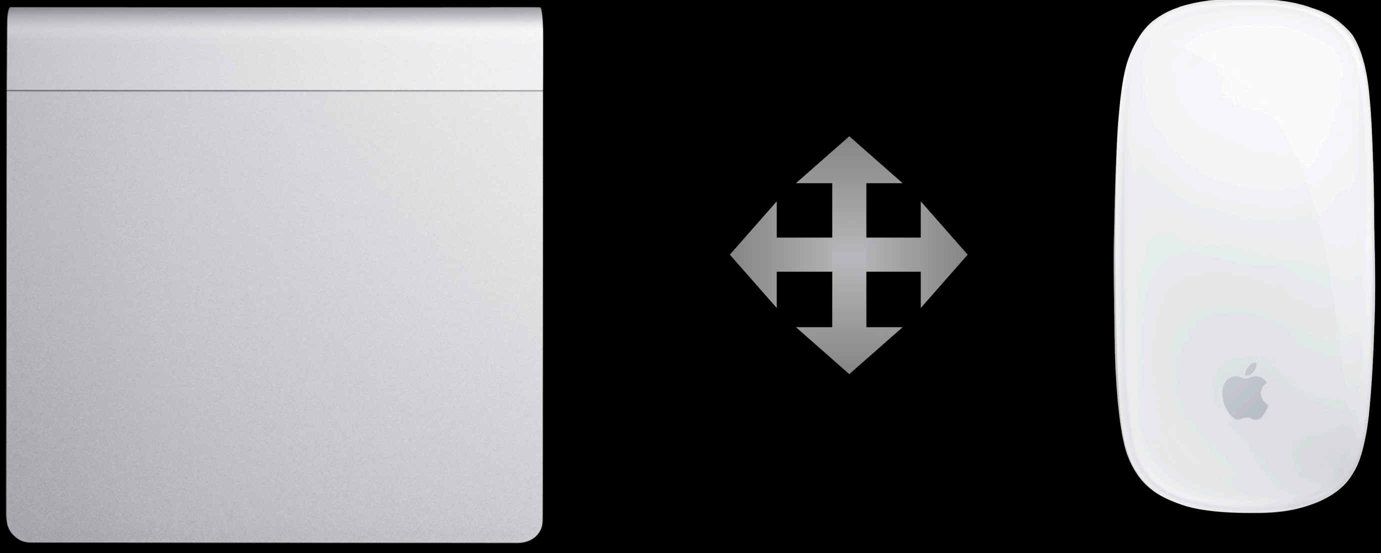
Page Scroll: Horizontal 10, Vertical 10

Behavior:
✓ Copy On Scroll
✓ Show Horizontal Scroller
✓ Show Vertical Scroller
☐ Automatically Hide Scroller

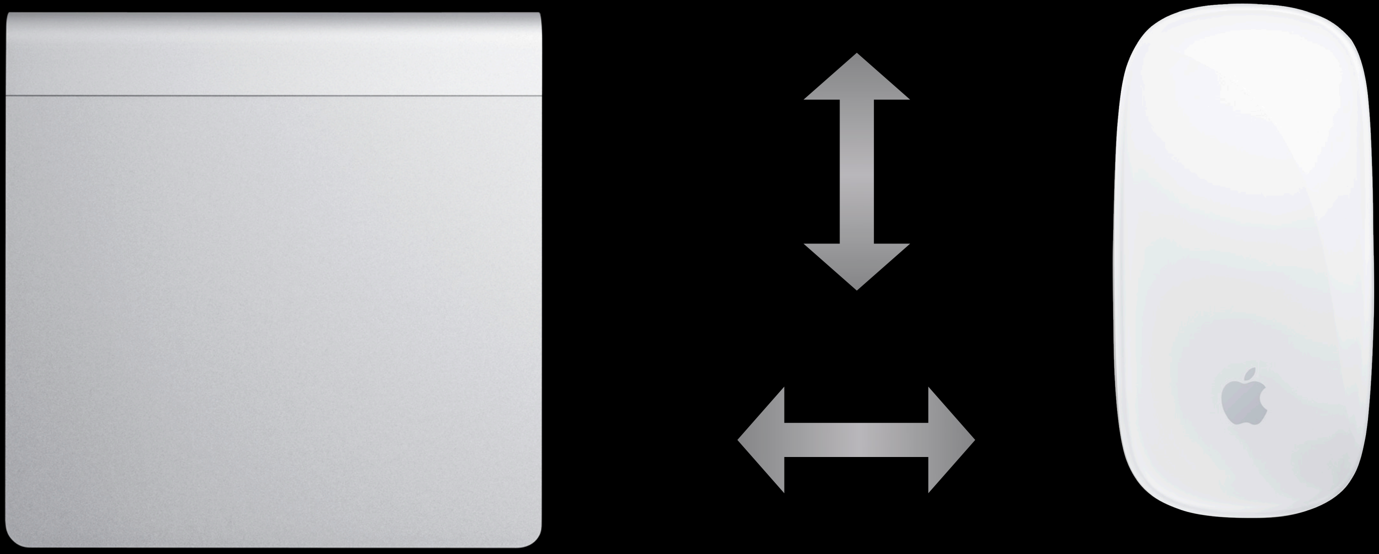
```
-(BOOL)usesPredominantAxisScrolling;
-(void)setUsesPredominantAxisScrolling:(BOOL)b;
```

Elastic Scrolling

Elastic Scrolling

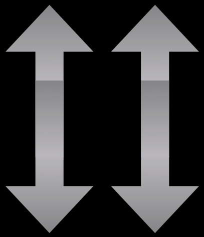


Elastic Scrolling



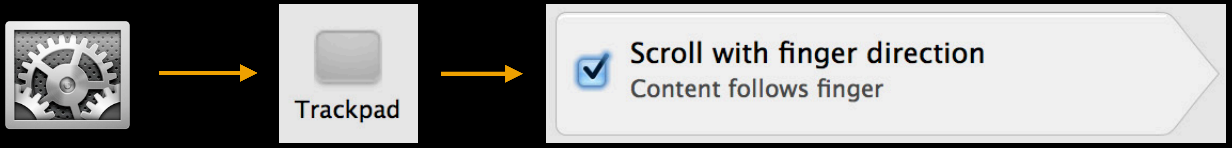
```
-(void)setUsesPredominantAxisScrolling:(BOOL)predominantAxisScrolling;
```

Scrolling



Apple Worldwide Developers Conference 2011
http://developer.apple.com/wwdc/
lorem ipsum
Developer
Technologies
Resources
Programs
Support
Member Center
Search Developer
WWDC2011
Join us for a preview of the future of iOS and Mac OS X.

Scrolling

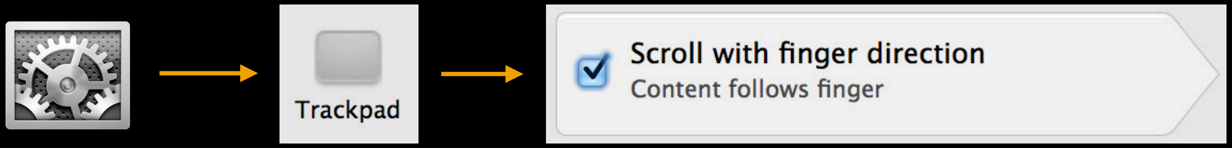


☑ Scroll with finger direction
Content follows finger

Scrolling

- NSEvent

```
-(BOOL)isDirectionInvertedFromDevice;
```

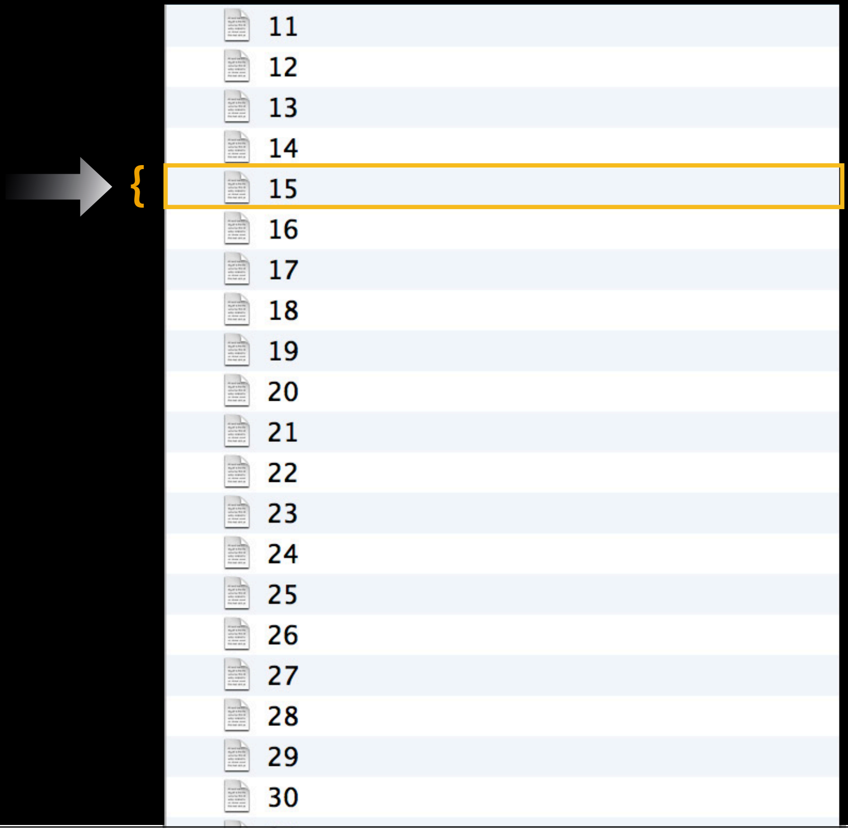


☑ Scroll with finger direction
Content follows finger

Gesture and Momentum NSEvent Properties

```
-(BOOL)isDirectionInvertedFromDevice;
-(CGFloat)scrollingDeltaX;
-(CGFloat)scrollingDeltaY;
-(BOOL)hasPreciseScrollingDeltas;
```

Gesture and Momentum NSEvent Properties



Gesture and Momentum NSEvent Properties

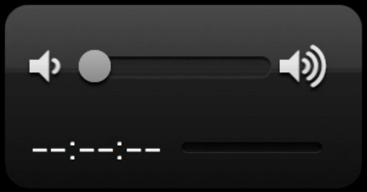
```
-(BOOL)isDirectionInvertedFromDevice;
-(CGFloat)scrollingDeltaX;
-(CGFloat)scrollingDeltaY;
-(BOOL)hasPreciseScrollingDeltas;
```

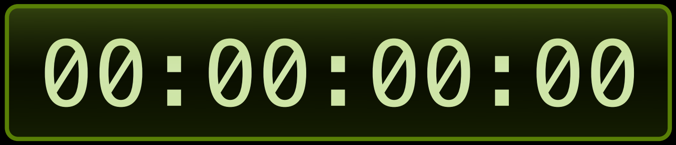
Gesture and Momentum NSEvent Properties

```
-(BOOL)isDirectionInvertedFromDevice;
-(CGFloat)scrollingDeltaX;
-(CGFloat)scrollingDeltaY;
-(BOOL)hasPreciseScrollingDeltas;
-(NSEventPhase)phase;
-(NSEventPhase)momentumPhase;
```

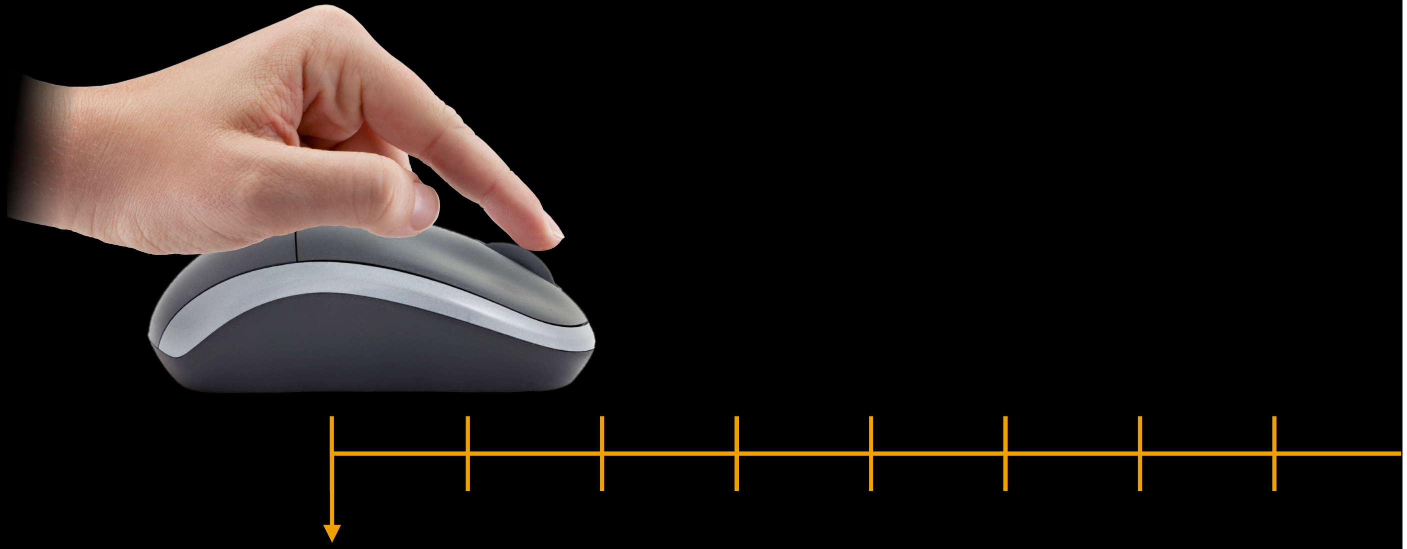
Gesture Scroll Sequence

Uses



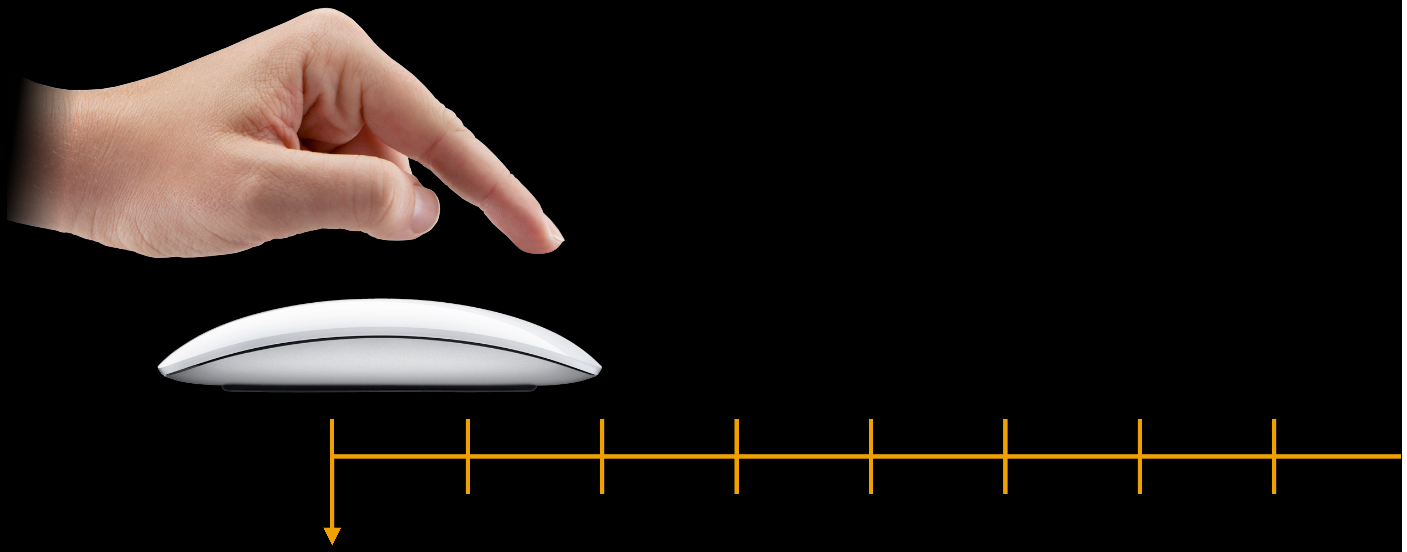


Gesture Scroll Sequence



```
       -phase NSEventPhaseNone
-momentumPhase NSEventPhaseNone
```

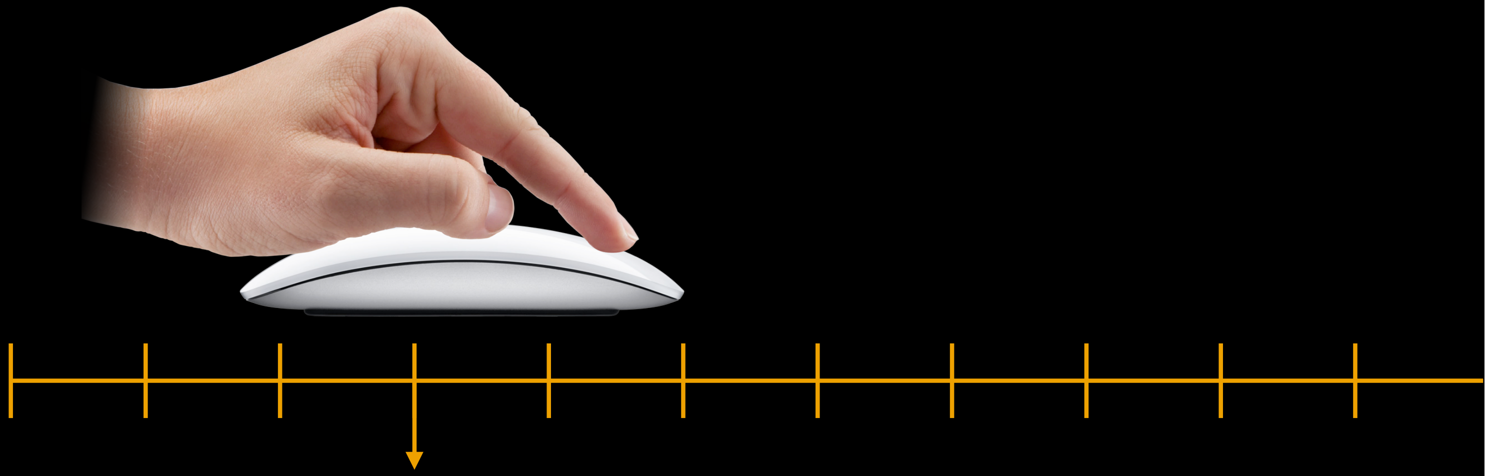
Gesture Scroll Sequence



−phase NSEventPhaseChanged

−momentumPhase NSEventPhaseNone

Gesture Scroll Sequence

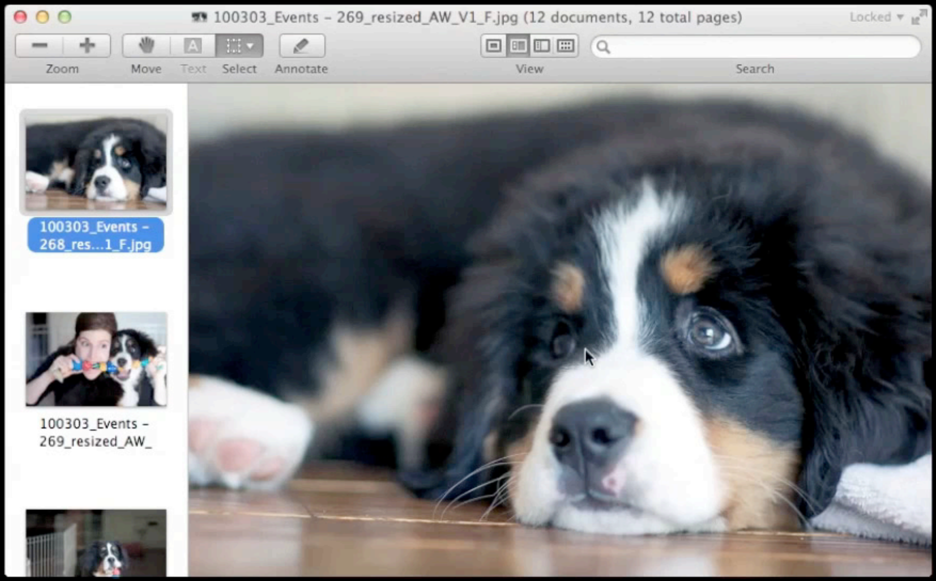


```
        -phase NSEventPhaseEnded
-momentumPhase NSEventPhaseBegan
```

Fluid Swiping

Raleigh Ledet

Fluid Swiping


100303_Events – 269_resized_AW_V1_F.jpg (12 documents, 12 total pages)
Locked
Zoom
Move
Text
Select
Annotate
View
Search
100303_Events – 268_res...1_F.jpg
100303_Events – 269_resized_AW_

Safari

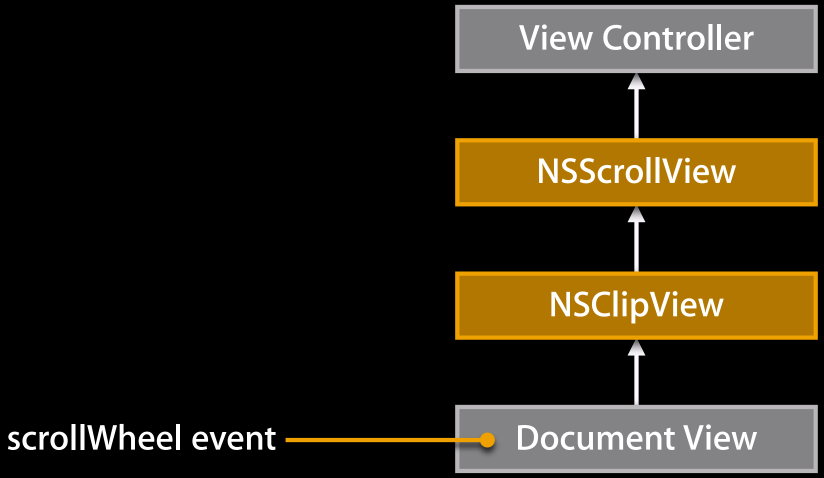
Preview

iCal

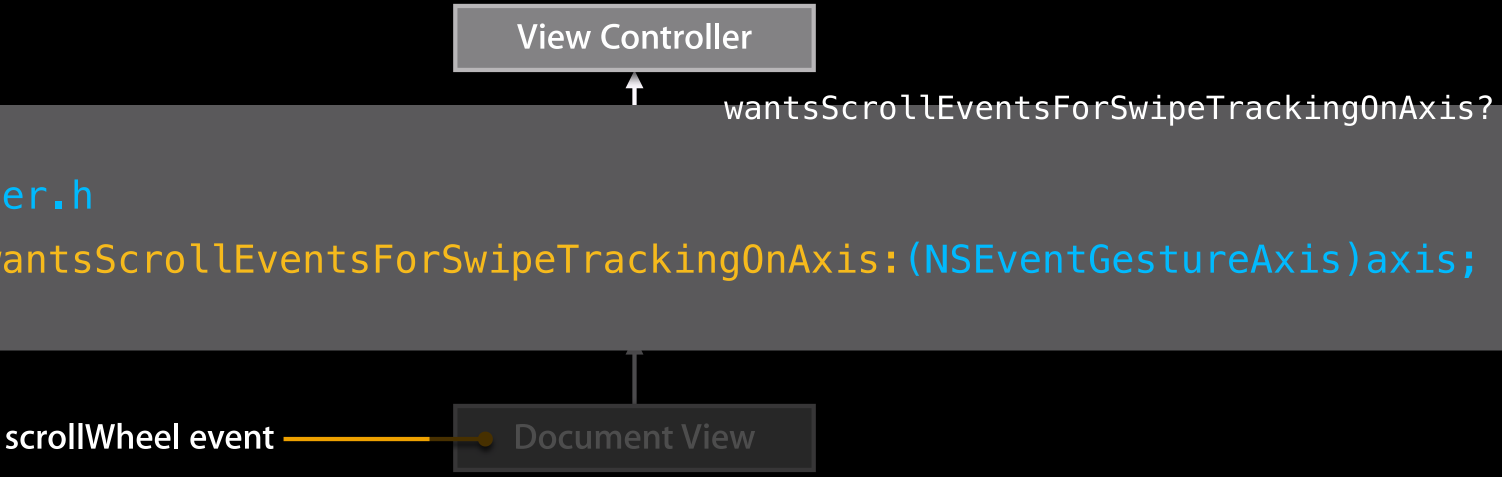
Quick Look

It’s just scrolling!

Fluid Swiping

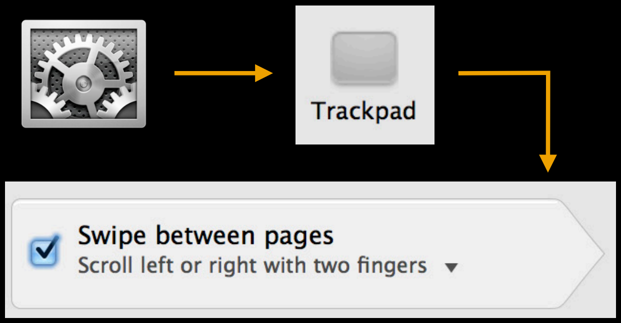


Fluid Swiping

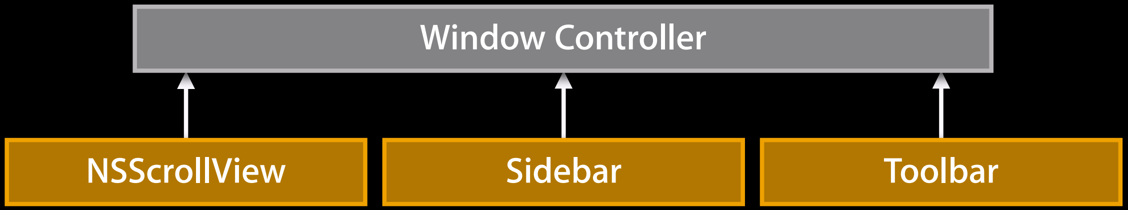


```
NSResponder.h
- (BOOL)wantsScrollEventsForSwipeTrackingOnAxis:(NSEventGestureAxis)axis;
```

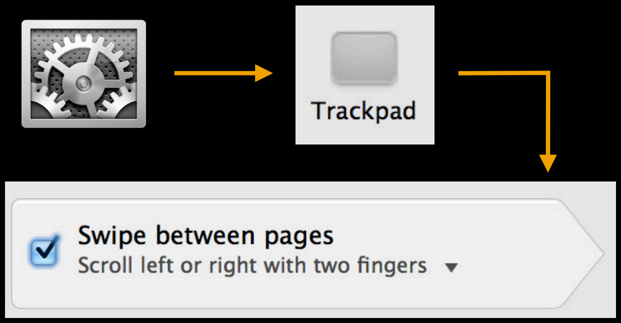
Fluid Swiping



☑ Swipe between pages
Scroll left or right with two fingers



Fluid Swiping



☑ Swipe between pages
Scroll left or right with two fingers ▾

```
NSEvent
+ (BOOL)isSwipeTrackingFromScrollEventsEnabled
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
            max:nextCount usingHandler:^(CGFloat gestureAmount,
            NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
            max:nextCount usingHandler:^(CGFloat gestureAmount,
            NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
          max:nextCount usingHandler:^(CGFloat gestureAmount,
          NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
          max:nextCount usingHandler:^(CGFloat gestureAmount,
          NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
          max:nextCount usingHandler:^(CGFloat gestureAmount,
          NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

Fluid Swiping

```
- (void)scrollWheel:(NSEvent *)event {
    [event trackSwipeEventWithOptions:0 dampenAmountThresholdMin:-prevCount
           max:nextCount usingHandler:^(CGFloat gestureAmount,
           NSEventPhase phase, BOOL isComplete, BOOL *stop) {
        if (phase == NSEventPhaseBegan) [self showSwipeOverlay];
        [self moveOverlayContentTo:gestureAmount];
        if (phase == NSEventPhaseEnded) {
            currentPictureIndex += (gestureAmount > 0) ? 1 : -1;
        }
        if (isComplete) [self hideSwipeOverlay];
    }];
}
```

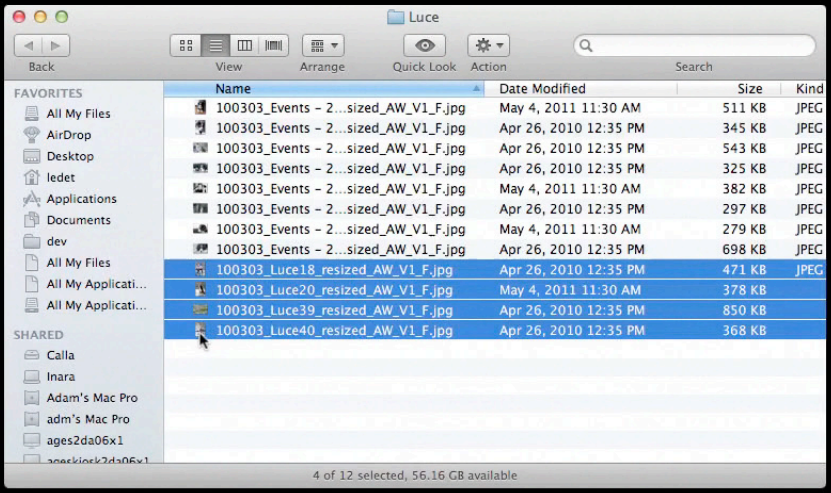
Demo

Raleigh Ledet

Multi-Image Dragging

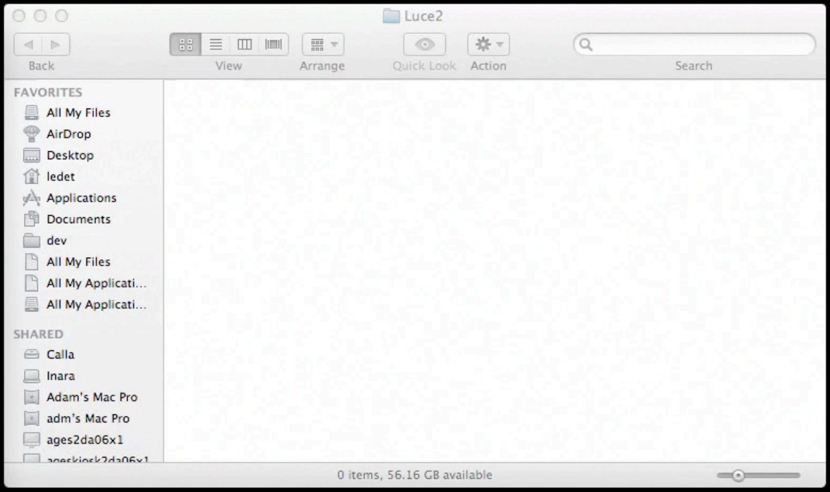
Raleigh Ledet

Multi-Image Dragging



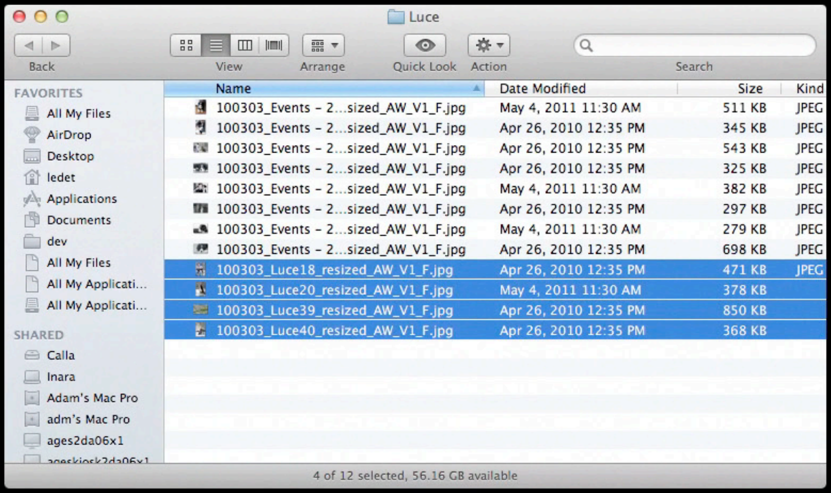
Name	Date Modified	Size	Kind
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	511 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	345 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	543 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	325 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	382 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	297 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	279 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	698 KB	JPEG
100303_Luce18_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	471 KB	JPEG
100303_Luce20_resized_AW_V1_F.jpg	May 4, 2011 11:30 AM	378 KB	
100303_Luce39_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	850 KB	
100303_Luce40_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	368 KB	

4 of 12 selected, 56.16 GB available



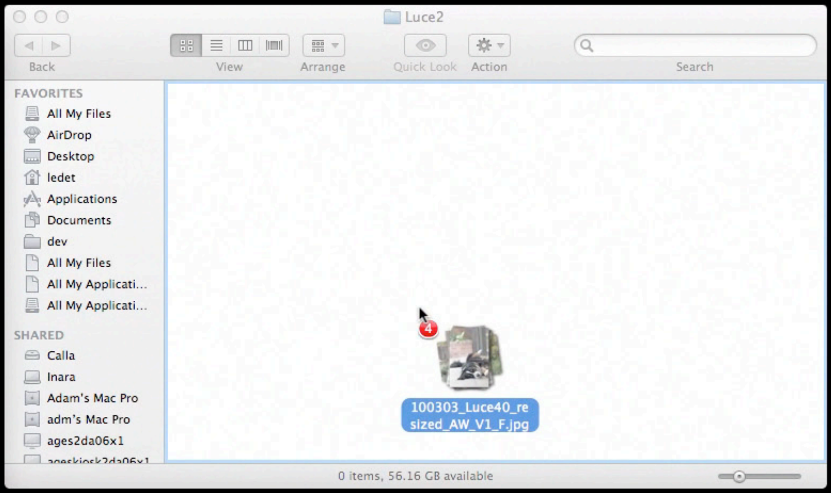
0 items, 56.16 GB available

Multi-Image Dragging



Name	Date Modified	Size	Kind
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	511 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	345 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	543 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	325 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	382 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	297 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	May 4, 2011 11:30 AM	279 KB	JPEG
100303_Events – 2...sized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	698 KB	JPEG
100303_Luce18_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	471 KB	JPEG
100303_Luce20_resized_AW_V1_F.jpg	May 4, 2011 11:30 AM	378 KB	
100303_Luce39_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	850 KB	
100303_Luce40_resized_AW_V1_F.jpg	Apr 26, 2010 12:35 PM	368 KB	

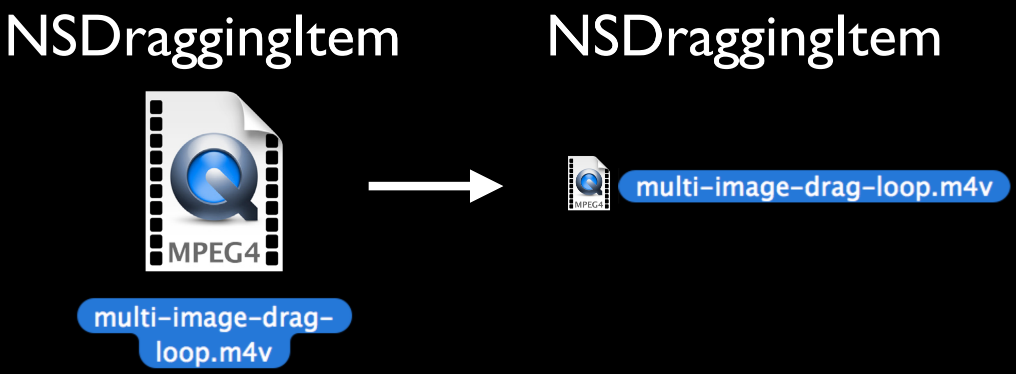
4 of 12 selected, 56.16 GB available



0 items, 56.16 GB available

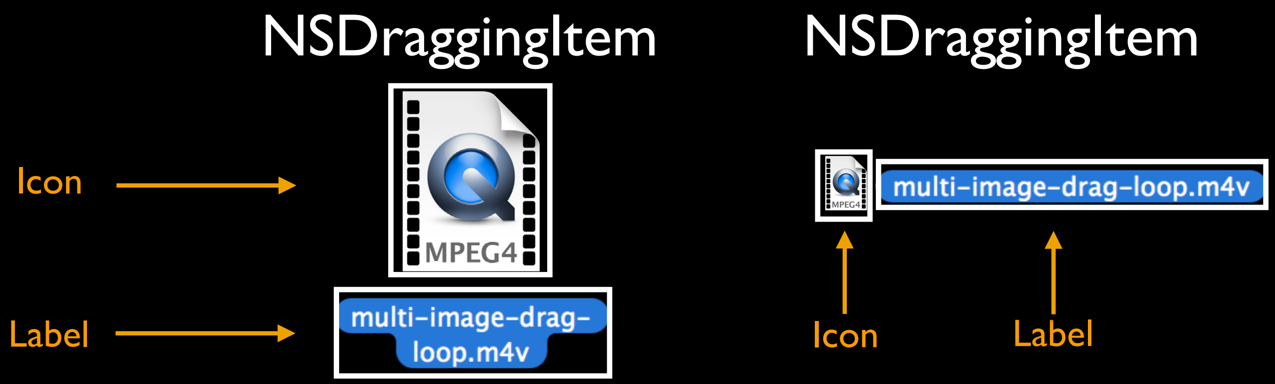
Multi-Image Dragging

How it works



Multi-Image Dragging

How it works



Multi-Image Dragging

How it works

NSDraggingItem

NSDraggingItem

Multi-Image Dragging

Starting a drag in Snow Leopard

```
NSPasteboard *pboard = [NSPasteboard pasteboardWithName:NSDragPboard];
[pboard clearContents];
[pboard writeObjects:pboardWriters];

[self dragImage:anImage at:location offset:NSZeroPoint event:event
      pasteboard:pboard source:self slideBack:YES];
```

Multi-Image Dragging

Starting a drag in Snow Leopard

```
NSPasteboard *pboard = [NSPasteboard pasteboardWithName:NSDragPboard];
[pboard clearContents];
[pboard writeObjects:pboardWriters];

[self dragImage:anImage at:location offset:NSZeroPoint event:event
      pasteboard:pboard source:self slideBack:YES];
```

Multi-Image Dragging

Starting a drag in Snow Leopard

```
NSPasteboard *pboard = [NSPasteboard pasteboardWithName:NSDragPboard];
[pboard clearContents];
[pboard writeObjects:pboardWriters];

[self dragImage:anImage at:location offset:NSZeroPoint event:event
      pasteboard:pboard source:self slideBack:YES];
```

Multi-Image Dragging

Starting a drag in Lion

```
NSArray *draggingItems = [self draggingItemsFromSelection];

NSDraggingSession *dragSession;
dragSession = [self beginDraggingSessionWithItems:draggingItems
                    event:event source:self];
dragSession.animatesToStartingPositionsOnCancelOrFail = YES;
```

Multi-Image Dragging

Starting a drag in Lion

```
NSArray *draggingItems = [self draggingItemsFromSelection];

NSDraggingSession *dragSession;
dragSession = [self beginDraggingSessionWithItems:draggingItems
                    event:event source:self];
dragSession.animatesToStartingPositionsOnCancelOrFail = YES;
```

Multi-Image Dragging

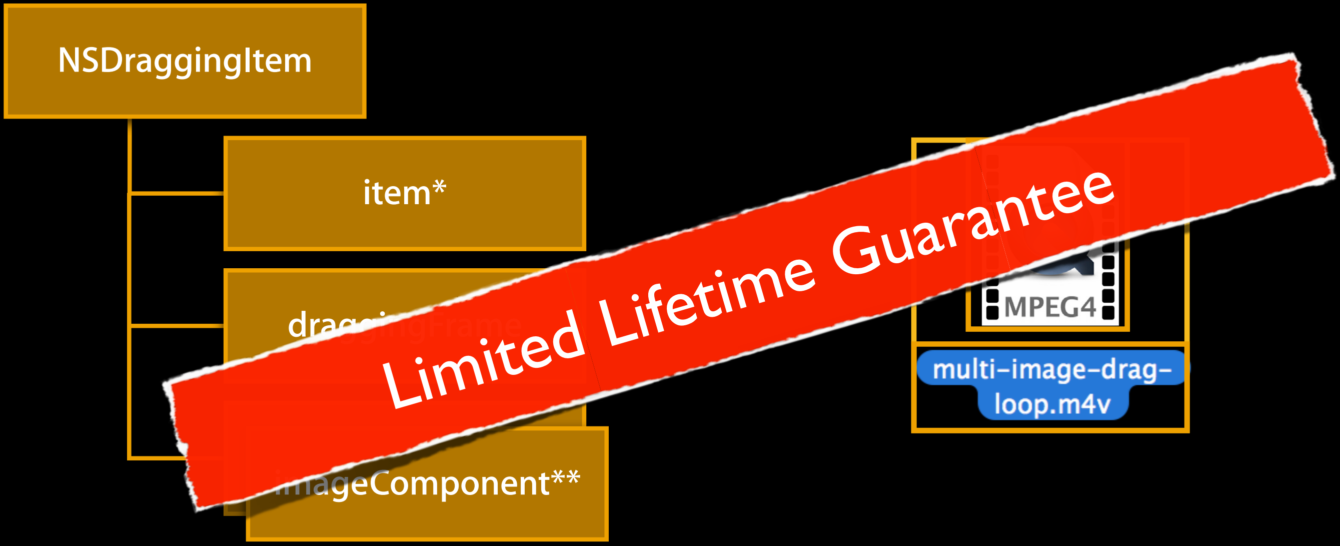
Starting a drag in Lion

```
NSArray *draggingItems = [self draggingItemsFromSelection];

NSDraggingSession *dragSession;
dragSession = [self beginDraggingSessionWithItems:draggingItems
                    event:event source:self];
dragSession.animatesToStartingPositionsOnCancelOrFail = YES;
```

Multi-Image Dragging

Creating an NSDraggingItem



Multi-Image Dragging

Creating an NSDraggingItem

```
id pbWriter = [self pasteboardWriterForObjectAtIndex:idx];
NSDraggingItem *dragItem = [[[NSDraggingItem alloc]
                             initWithPasteboardWriter:pbWriter] autorelease];
NSRect dragFrame = [[self rootViewAtIndex:idx] frame];
[dragItem setDraggingFrame:dragFrame contents:anImage];
```

Multi-Image Dragging

Creating an NSDraggingItem

```
 id pbWriter = [self pasteboardWriterForObjectAtIndex:idx];
NSDraggingItem *dragItem = [[[NSDraggingItem alloc]
                              initWithPasteboardWriter:pbWriter] autorelease];
NSRect dragFrame = [[self rootViewAtIndex:idx] frame];
[dragItem setDraggingFrame:dragFrame contents:anImage];
```

Multi-Image Dragging

Creating an NSDraggingItem

```
draggingItem.imageComponentsProvider = ^ {
    NSMutableArray *componentsArray = [NSMutableArray arrayWithCapacity:2];
    NSDraggingImageComponent *component;

    component = [NSDraggingImageComponent
            draggingImageComponentWithKey:NSDraggingImageComponentIconKey];
    component.frame = [iconView frame];
    component.contents = ImageOfView(iconView);
    [componentsArray addObject:component];
    ...
    return componentsArray;
};
```

Multi-Image Dragging

Creating an NSDraggingItem

```
draggingItem.imageComponentsProvider = ^ {
    NSMutableArray *componentsArray = [NSMutableArray arrayWithCapacity:2];
    NSDraggingImageComponent *component;

    component = [NSDraggingImageComponent
            draggingImageComponentWithKey:NSDraggingImageComponentIconKey];
    component.frame = [iconView frame];
    component.contents = ImageOfView(iconView);
    [componentsArray addObject:component];
    ...
    return componentsArray;
};
```

Multi-Image Dragging

Starting a drag in Lion

```
NSArray *draggingItems = [self draggingItemsFromSelection];

NSDraggingSession *dragSession;
dragSession = [self beginDraggingSessionWithItems:draggingItems
                    event:event source:self];
dragSession.animatesToStartingPositionsOnCancelOrFail = YES;
```

Do not access draggingItems beyond this point

Multi-Image Dragging

Dragging source—NSDragging.h

```
– draggingSourceOperationMaskForDraggingContext:
– draggedImage:beganAtPoint:
– draggedImage:movedToPoint:
– draggedImage:endedAtPoint:operation:
– ignoreModifierKeysWhileDragging:
```

Multi-Image Dragging

Dragging source—NSDragging.h

```
@protocol NSDraggingSource <NSObject>
- draggingSession:sourceOperationMaskForDraggingContext:
- draggingSession:beganAtPoint:
- draggingSession:movedToPoint:
- draggingSession:endedAtPoint:operation:
- ignoreModifierKeysForDraggingSession:
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
- (NSDragOperation)draggingEntered:(id)sender;
- (NSDragOperation)draggingUpdated:(id)sender;
- (void)draggingExited:(id)sender;
- (void)draggingEnded:(id)sender;
- (BOOL)prepareForDragOperation:(id)sender;
- (BOOL)performDragOperation:(id)sender;
- (void)concludeDragOperation:(id)sender;
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
@protocol NSDraggingDestination <NSObject>
- (NSDragOperation)draggingEntered:(id <NSDraggingInfo>)sender;
- (NSDragOperation)draggingUpdated:(id <NSDraggingInfo>)sender;
- (void)draggingExited:(id <NSDraggingInfo>)sender;
- (void)draggingEnded:(id <NSDraggingInfo>)sender;
- (BOOL)prepareForDragOperation:(id <NSDraggingInfo>)sender;
- (BOOL)performDragOperation:(id <NSDraggingInfo>)sender;
- (void)concludeDragOperation:(id <NSDraggingInfo>)sender;
- (void)updateDraggingItemsForDrag:(id <NSDraggingInfo>)sender;
```

Multi-Image Dragging

Dragging destination—NSDragging.h

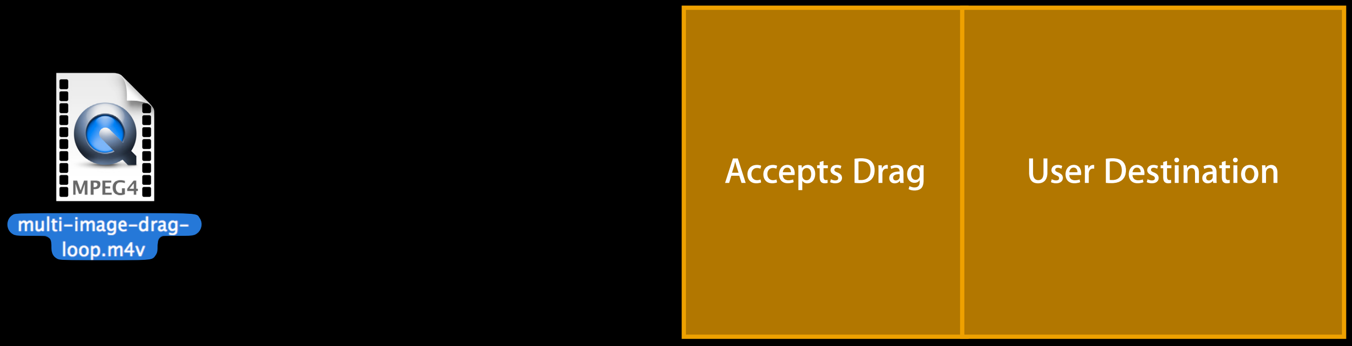
```
@protocol NSDraggingDestination <NSObject>



- (void)updateDraggingItemsForDrag:(id <NSDraggingInfo>)sender;
```

Multi-Image Dragging

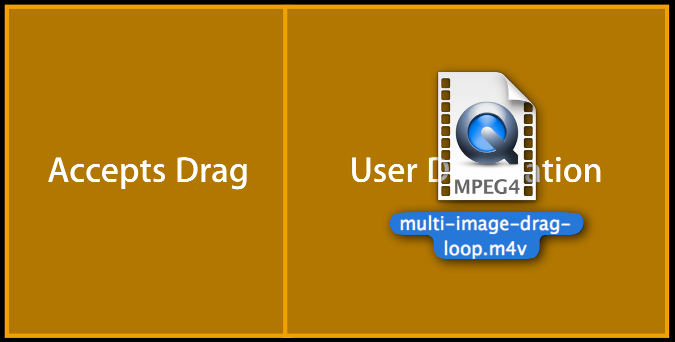
Dragging destination



```
— (NSDragOperation)draggingEntered:(id <NSDraggingInfo>)sender
— (void)updateDraggingItemsForDrag:(id <NSDraggingInfo>)sender
```

Multi-Image Dragging

Dragging destination



```
- (void)updateDraggingItemsForDrag:(id <NSDraggingInfo>)sender
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
NSArray *objectClasses = [NSArray arrayWithObject:[NSURL class]];
[dragInfo enumerateDraggingItemsWithOptions:0 forView:self
       classes:objectClasses searchOptions:nil
       usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
{
    NSURL *url = (NSURL *)draggingItem.item;
    NSImage *dragImage = // CREATE IMAGE HERE, but do it quickly
    NSRect newFrame = // determine new frame in self's coordinate space
    [draggingItem setDraggingFrame:newFrame contents:dragImage];
}];
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
NSArray *objectClasses = [NSArray arrayWithObject:[NSURL class]];
[dragInfo enumerateDraggingItemsWithOptions:0 forView:self
       classes:objectClasses searchOptions:nil
       usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
{
    NSURL *url = (NSURL *)draggingItem.item;
    NSImage *dragImage = // CREATE IMAGE HERE, but do it quickly
    NSRect newFrame = // determine new frame in self's coordinate space
    [draggingItem setDraggingFrame:newFrame contents:dragImage];
}];
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
NSArray *objectClasses = [NSArray arrayWithObject:[NSURL class]];
[dragInfo enumerateDraggingItemsWithOptions:0 forView:self
       classes:objectClasses searchOptions:nil
       usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
{
    NSURL *url = (NSURL *)draggingItem.item;
    NSImage *dragImage = // CREATE IMAGE HERE, but do it quickly
    NSRect newFrame = // determine new frame in self's coordinate space
    [draggingItem setDraggingFrame:newFrame contents:dragImage];
}];
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
NSArray *objectClasses = [NSArray arrayWithObject:[NSURL class]];
[dragInfo enumerateDraggingItemsWithOptions:0 forView:self
       classes:objectClasses searchOptions:nil
       usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
{
    NSURL *url = (NSURL *)draggingItem.item;
    NSImage *dragImage = // CREATE IMAGE HERE, but do it quickly
    NSRect newFrame = // determine new frame in self's coordinate space
    [draggingItem setDraggingFrame:newFrame contents:dragImage];
}];
```

Multi-Image Dragging

Dragging destination—NSDragging.h

```
NSArray *objectClasses = [NSArray arrayWithObject:[NSURL class]];
[dragInfo enumerateDraggingItemsWithOptions:0 forView:self
       classes:objectClasses searchOptions:nil
       usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
{
    NSURL *url = (NSURL *)draggingItem.item;
    NSImage *dragImage = // CREATE IMAGE HERE, but do it quickly
    NSRect newFrame = // determine new frame in self's coordinate space
    [draggingItem setDraggingFrame:newFrame contents:dragImage];
}];
```

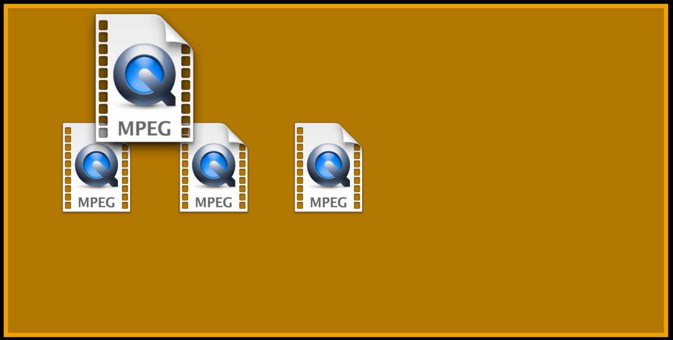
Multi-Image Dragging

Dragging destination—NSDragging.h

```
@protocol NSDraggingDestination <NSObject>
- draggingEntered:(id <NSDraggingInfo>)sender;
- draggingUpdated:(id <NSDraggingInfo>)sender;
- draggingExited:(id <NSDraggingInfo>)sender;
- draggingEnded:(id <NSDraggingInfo>)sender;
- (BOOL)prepareForDragOperation:(id <NSDraggingInfo>)sender;
- (BOOL)performDragOperation:(id <NSDraggingInfo>)sender;
- (void)concludeDragOperation:(id <NSDraggingInfo>)sender;
- updateDraggingItemsForDrag:(id <NSDraggingInfo>)sender
```

Multi-Image Dragging

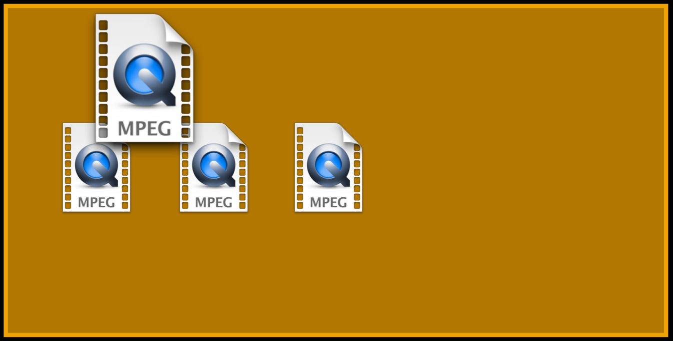
Dragging destination



```
- (BOOL)prepareForDragOperation:(id <NSDraggingInfo>)sender
```

Multi-Image Dragging

Dragging destination

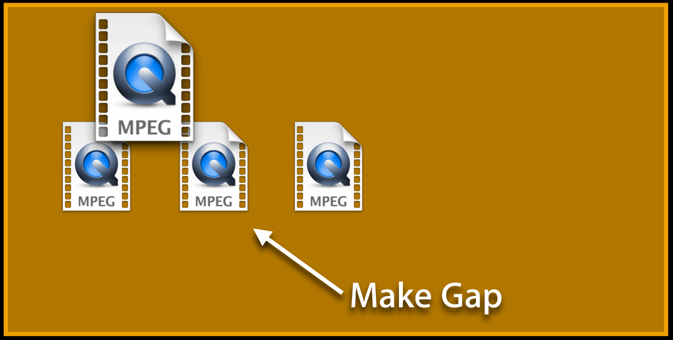


```
- (BOOL)prepareForDragOperation:(id <NSDraggingInfo>)sender

             sender.animatesToDestination = YES;
```

Multi-Image Dragging

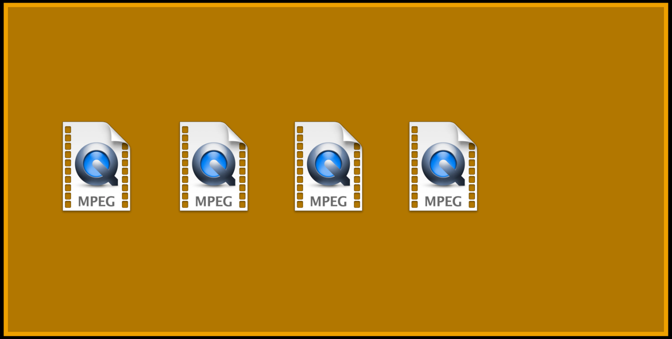
Dragging destination



```
- (BOOL)performDragOperation:(id <NSDraggingInfo>)sender
```

Multi-Image Dragging

Dragging destination



```
- (void)concludeDragOperation:(id <NSDraggingInfo>)sender
```

Demo

Raleigh Ledet

Multi-Image Dragging

- View based NSTableView makes this easy
- NSCollectionView makes this easy

Summary

- Scrolling
  - Content focused redesign
  - New scroll event properties
- Fluid swiping
  - Use the -trackSwipe… method
  - Respect user preferences
- Multi-image dragging

Related Sessions

Full Screen and Aqua Changes	Russian Hill Wednesday 10:15AM
View Based NSTableView Basic to Advanced	Nob Hill Thursday 10:15AM

Labs

Cocoa, Full Screen, and Aqua Lab	App Frameworks Lab A Wednesday 2:00-6:00PM
Cocoa, Auto Save, File Coordination, and Resume Lab	App Frameworks Lab A Thursday 2:00-4:15PM

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Documentation
Mac OS X Dev Center
http://developer.apple.com/devcenter/mac

Apple Developer Forums
http://devforums.apple.com

WWDC2011