Frameworks

#WWDC14

Adopting Advanced Features of the New UI

Session 220

Chris Dreessen
AppKit Software Engineer

Corbin Dunn
AppKit Software Engineer

Finder File Edit View Go Window Help

Tue 9:41 AM

Mountain Biking with Jordan

Search

Favorites

All My Files

AirDrop

Applications

Desktop

Documents

Downloads

Devices

Shared

Tags

MTBwJordan1.jpg

MTBwJordan2.jpg

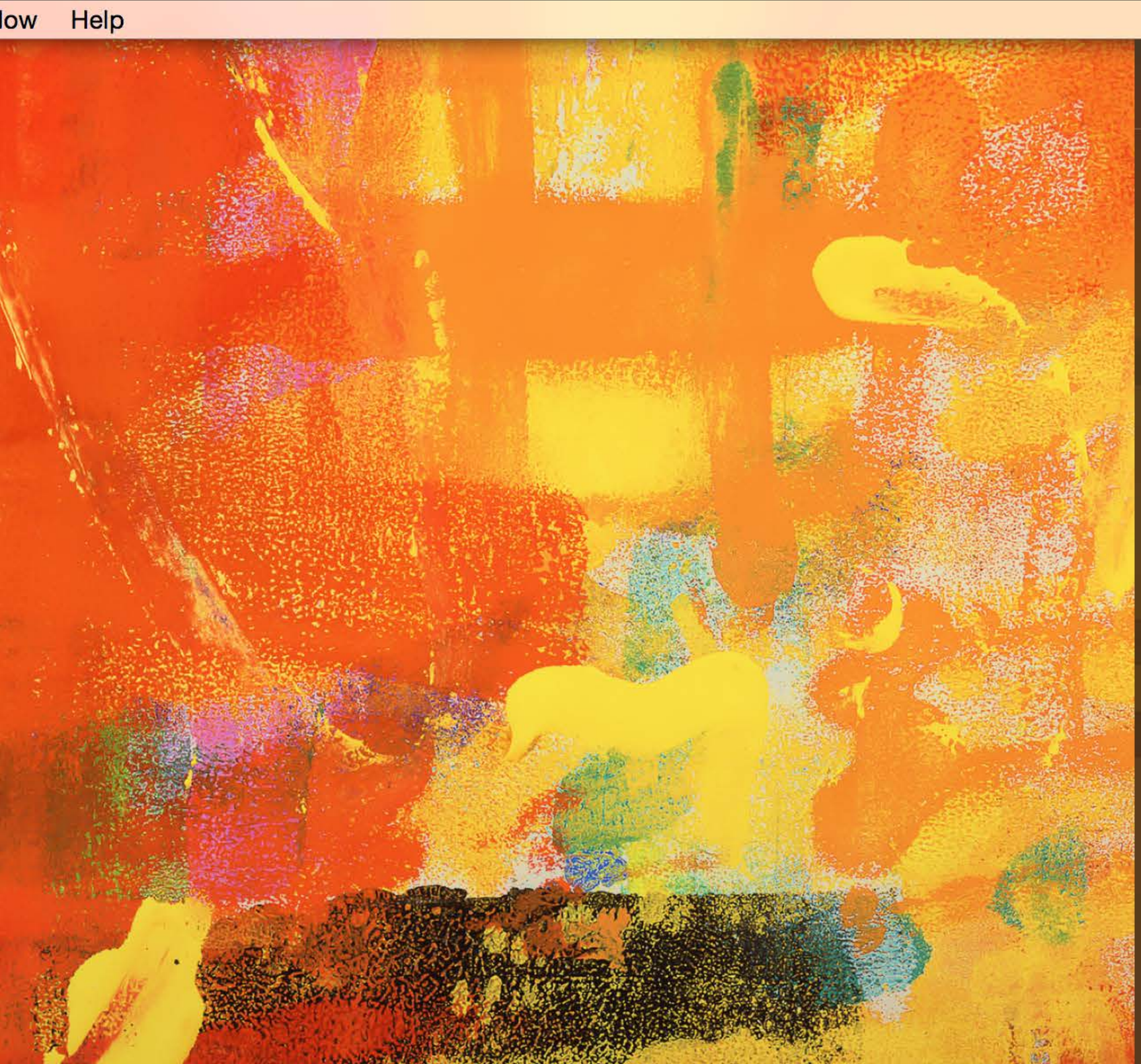
2 items, 21.16 GB available

Apple Inc.

WWDC14

Apple Worldwide Developers Conference. June 2–6, San Francisco.

Over the past six years, a massive cultural shift has occurred.
It's changed how we interact with one another. Learn new things.
Entertain ourselves. Do our work. And live our daily lives.
All because of developers and the apps they create.



low Help

Tue 9:41 AM

Today | Notifications

Tuesday,
June 3rd

Calendar

No Events

Reminders

Stocks

DOW J	16,675.50	+ 0.42%
NASDAQ	4,237.07	+ 1.22%
S&P 500	1,911.91	+ 0.60%
AAPL	625.63	+ 1.87%
GOOG	565.95	+ 2.40%

Overview

Overview

Significant changes in NSColor and NSImage

Overview

Significant changes in NSColor and NSImage

Titlebars and toolbars

Overview

Significant changes in NSColor and NSImage

Titlebars and toolbars

New visual effects and Vibrancy

Overview

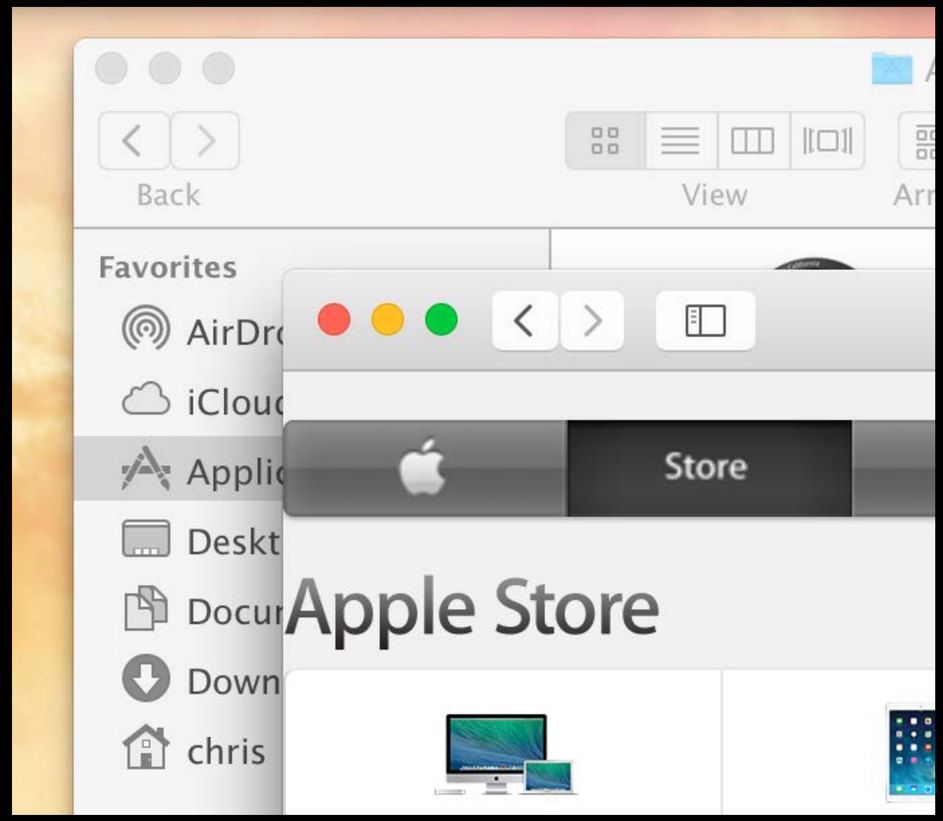
Significant changes in NSColor and NSImage

Titlebars and toolbars

New visual effects and Vibrancy

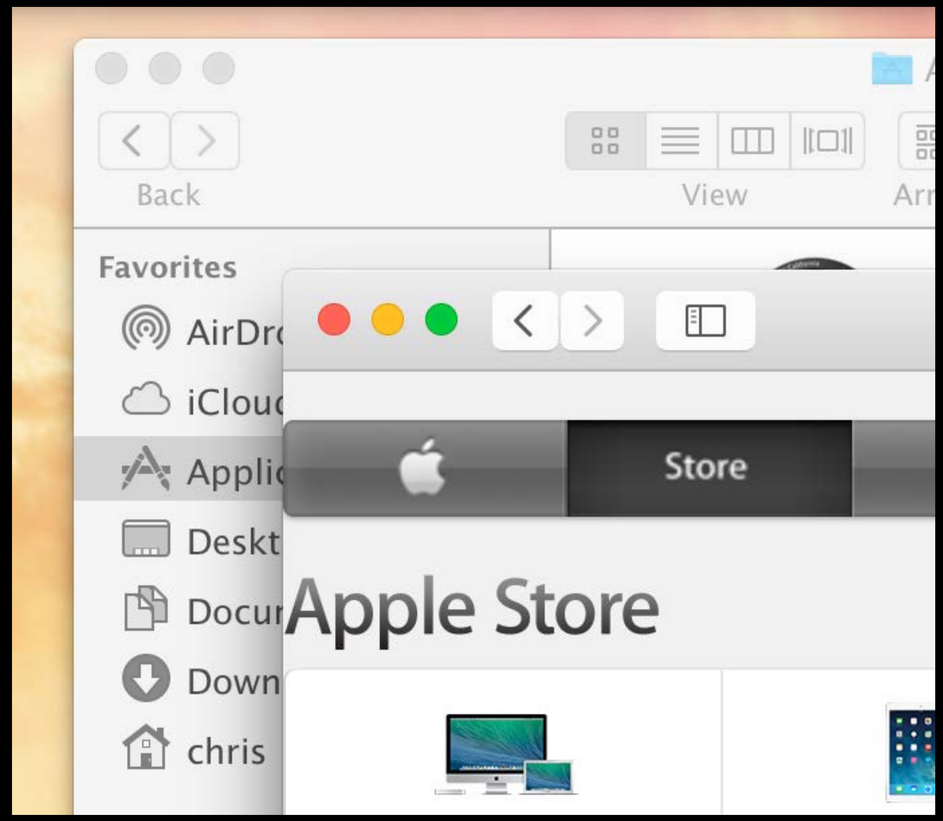
Performance

NSSegmentedControl



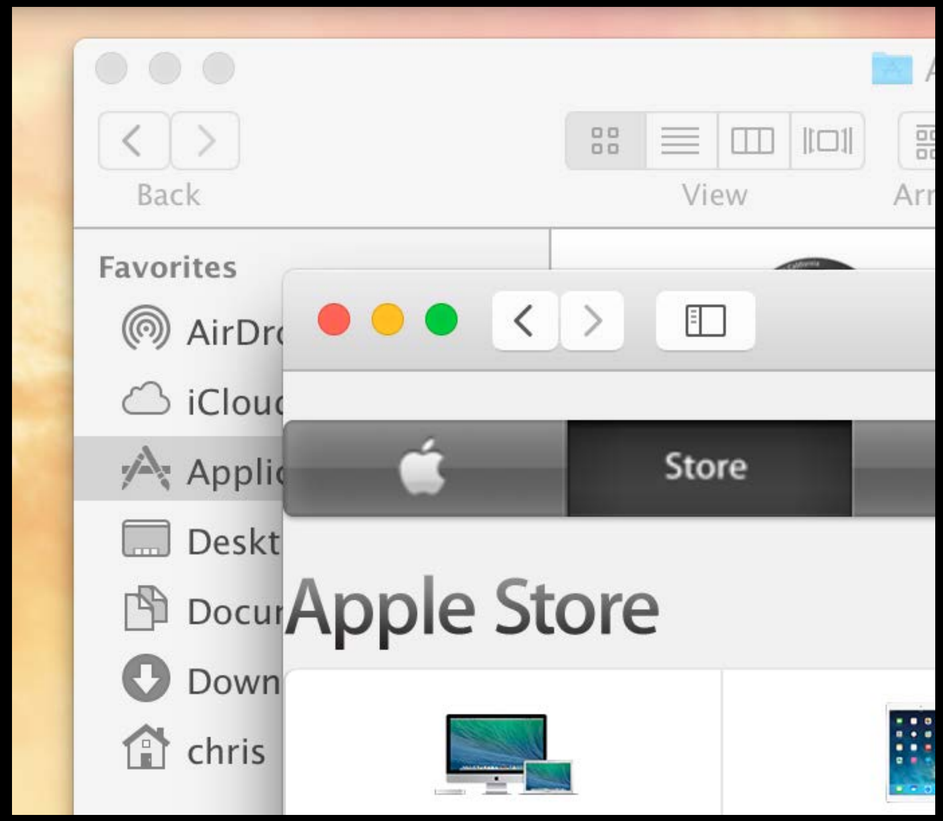
NSSegmentedControl

```
@interface NSSegmentedControl …
@property (NSSegmentStyle) segmentStyle;
…
```



NSSegmentedControl

```
@interface NSSegmentedControl …
@property (NSSegmentStyle) segmentStyle;
…

enum NSSegmentStyle {
…
    NSSegmentStyleSeparated;
}
```



NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

NSImage

```
@interface NSImage : NSObject …
@property NSEdgeInsets capInsets;
…
```

NSImage

```
@interface NSImage : NSObject …
@property NSEdgeInsets capInsets;
…
```

capInsets is in the image coordinate system

NSImage

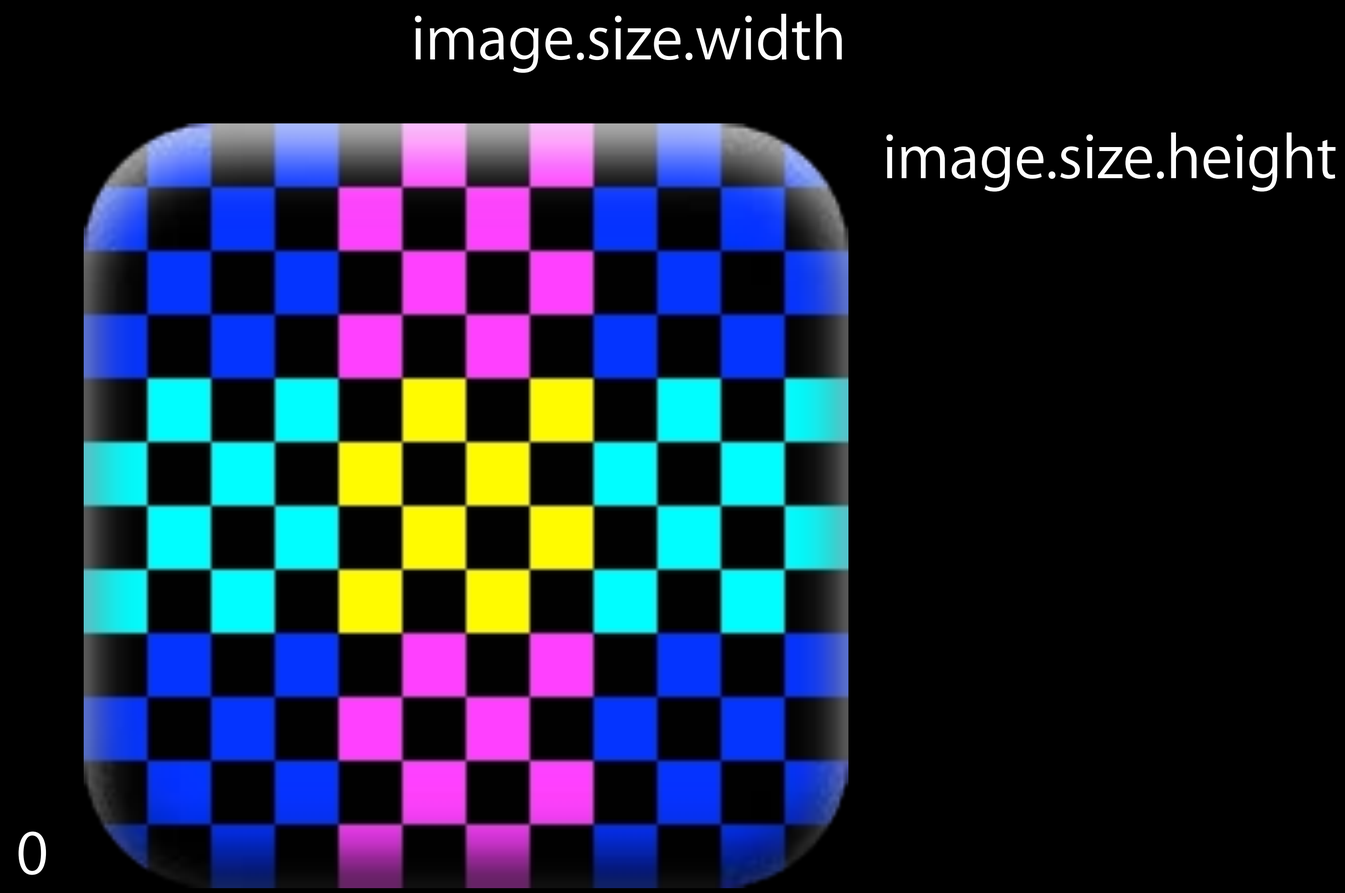
```
@interface NSImage : NSObject …
@property NSEdgeInsets capInsets;
…
```

capInsets is in the image coordinate system

NSImage

```
@interface NSImage : NSObject …
@property NSEdgeInsets capInsets;
…
```

capInsets is in the image coordinate system



NSImage

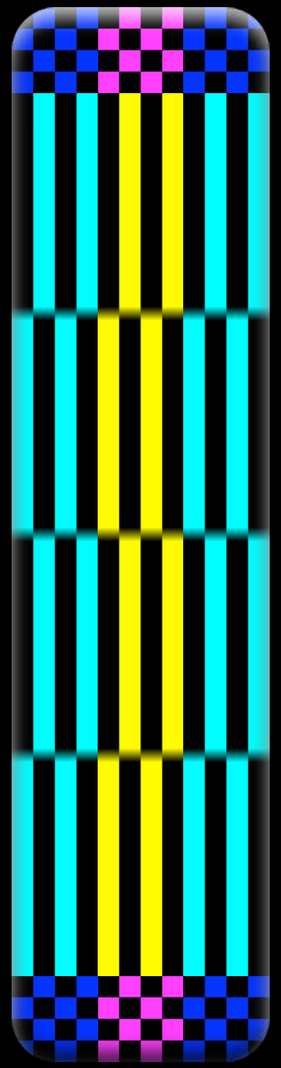
```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

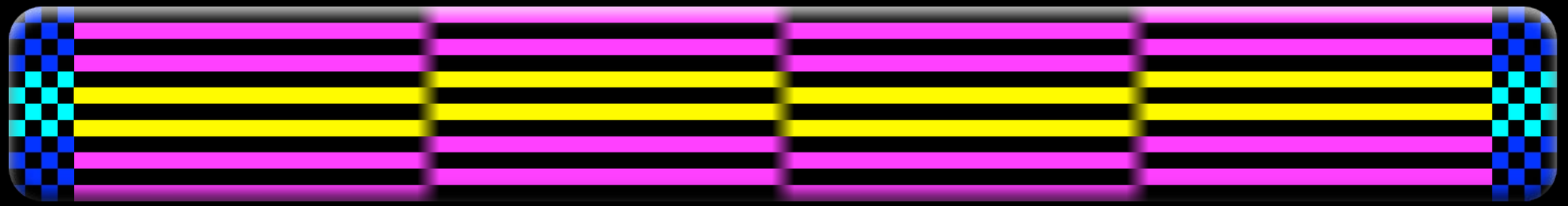


NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

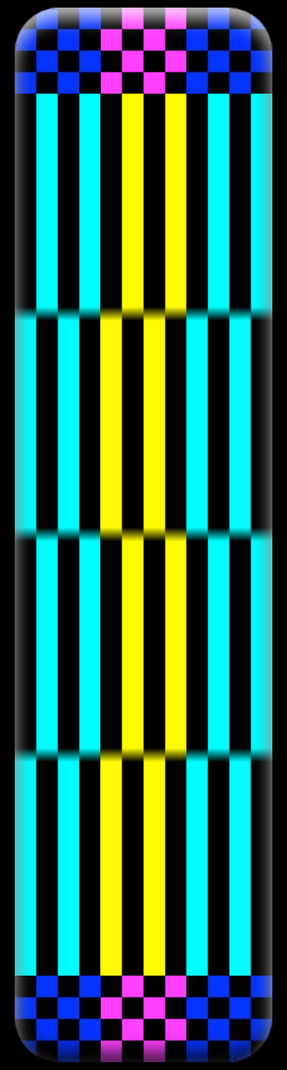


NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```



NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

```
image.capInsets = NSEdgeInsetsMake(64, 64, 64, 64);
```

NSImage

NSImage

```
enum NSImageResizingMode {
    NSImageResizingModeStretch
    NSImageResizingModeTile
}
```

NSImage

```
enum NSImageResizingMode {
    NSImageResizingModeStretch
    NSImageResizingModeTile
}
```

```
@interface NSImage : NSObject …
@property NSImageResizingMode resizingMode;


…
```

NSImage

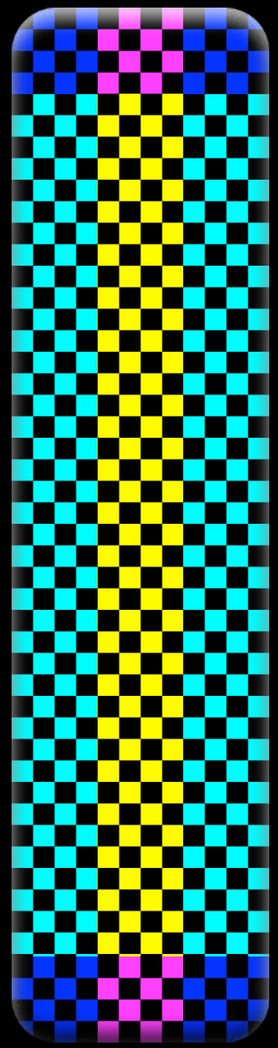
```
image.resizingMode = NSImageResizingModeTile
```

NSImage

```
image.resizingMode = NSImageResizingModeTile
```

NSImage

```
image.resizingMode = NSImageResizingModeTile
```

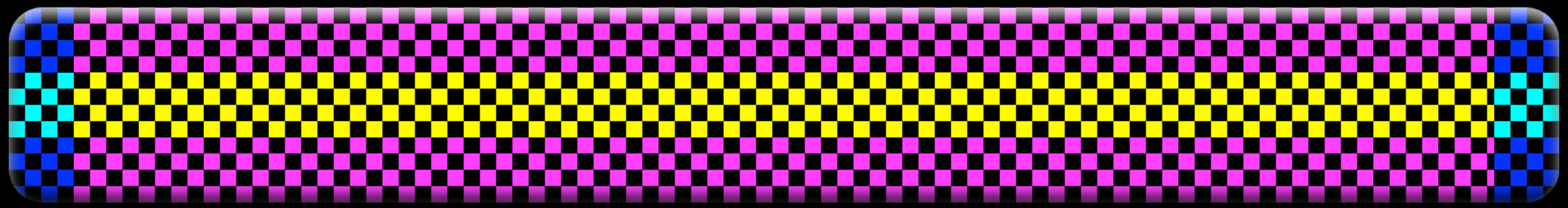


NSImage

```
image.resizingMode = NSImageResizingModeTile
```

NSImage

```
image.resizingMode = NSImageResizingModeTile
```

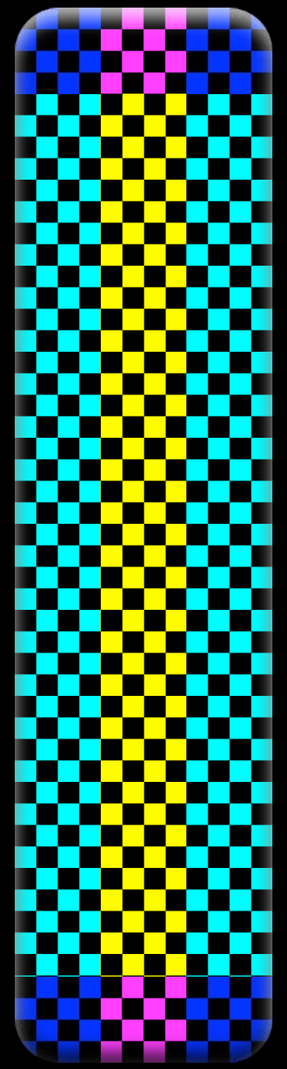


NSImage

```
image.resizingMode = NSImageResizingModeTile
```

NSImage

```
image.resizingMode = NSImageResizingModeTile
```

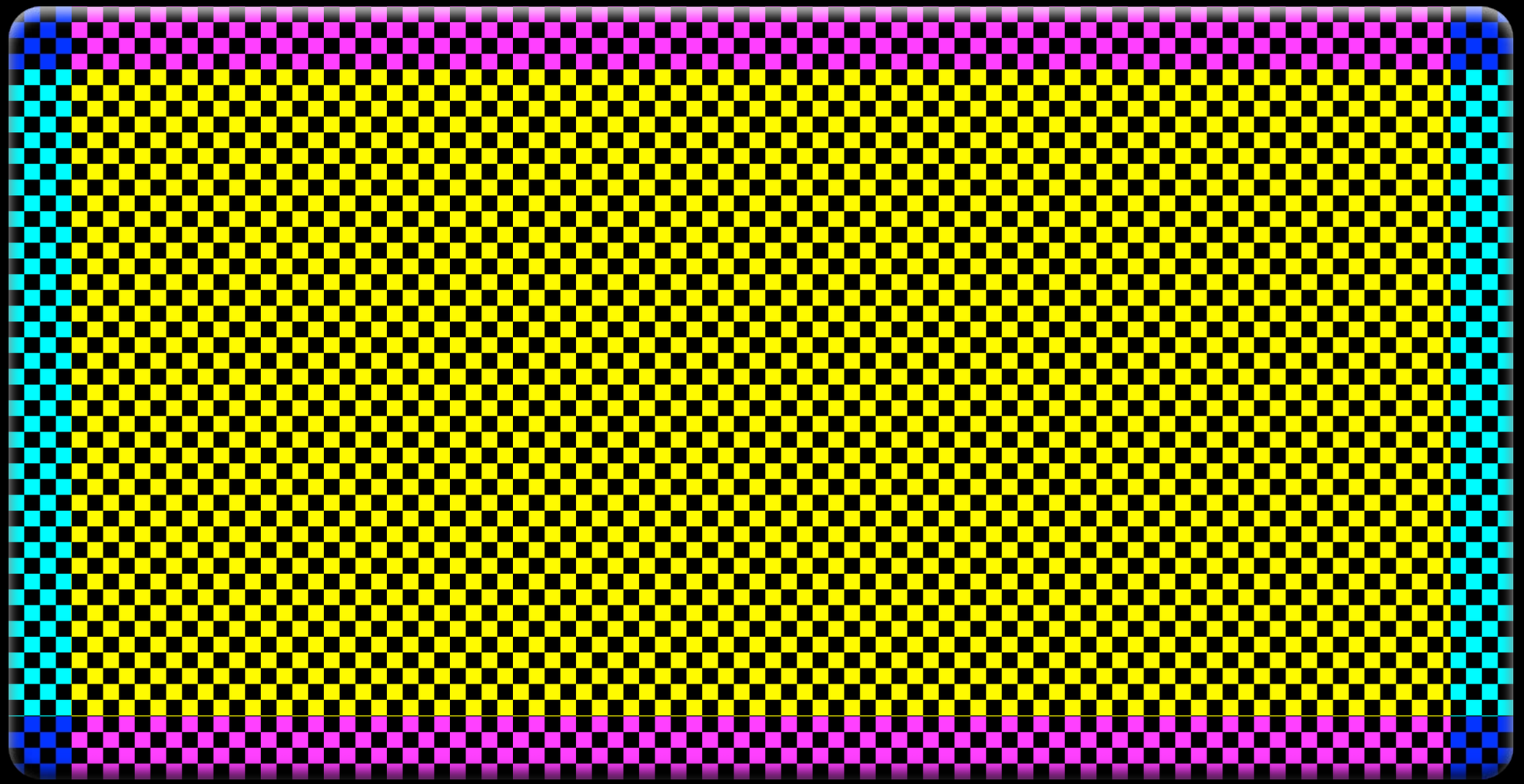


NSImage

```
image.resizingMode = NSImageResizingModeTile
```

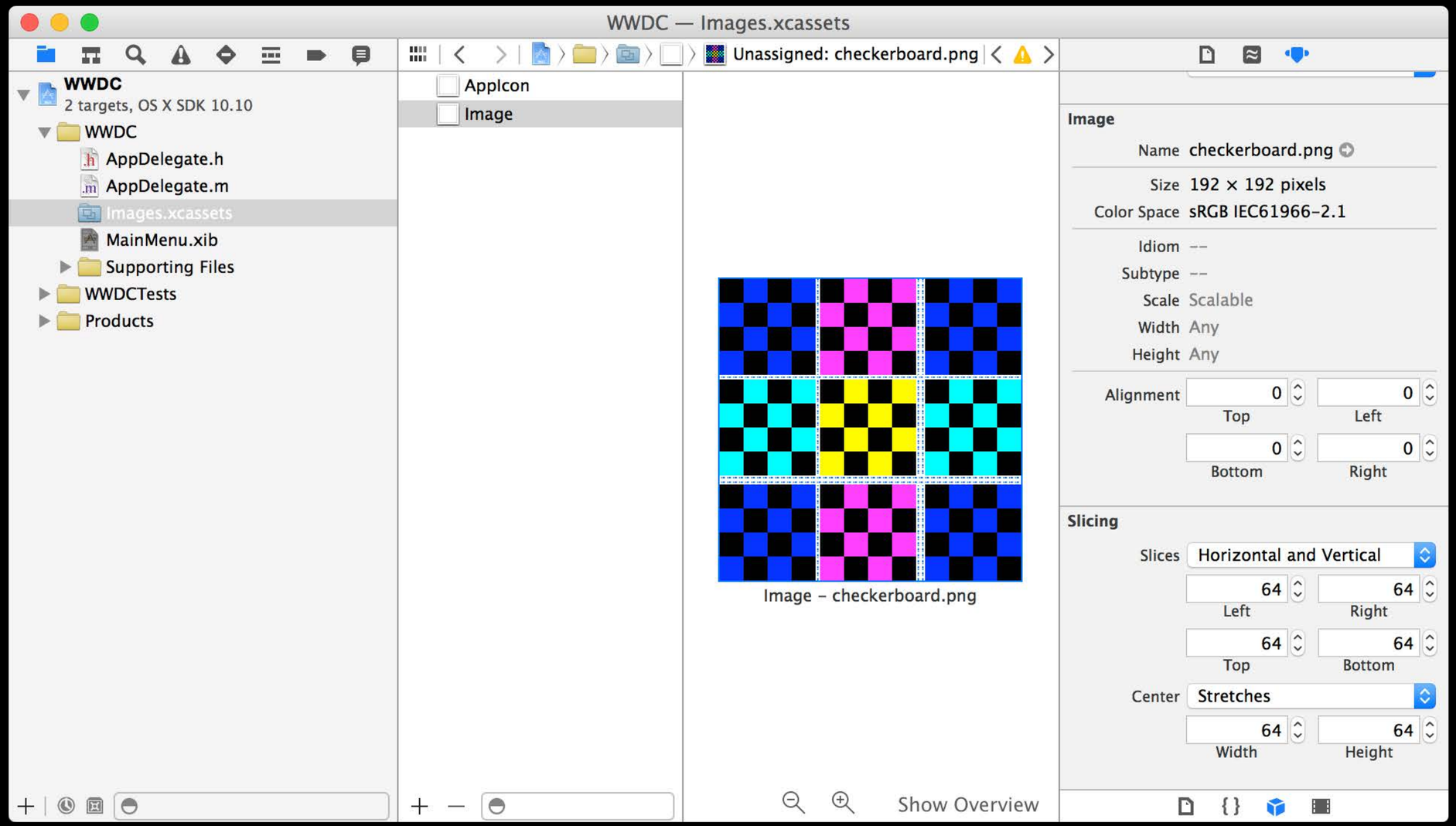
NSImage

```
image.resizingMode = NSImageResizingModeTile
```



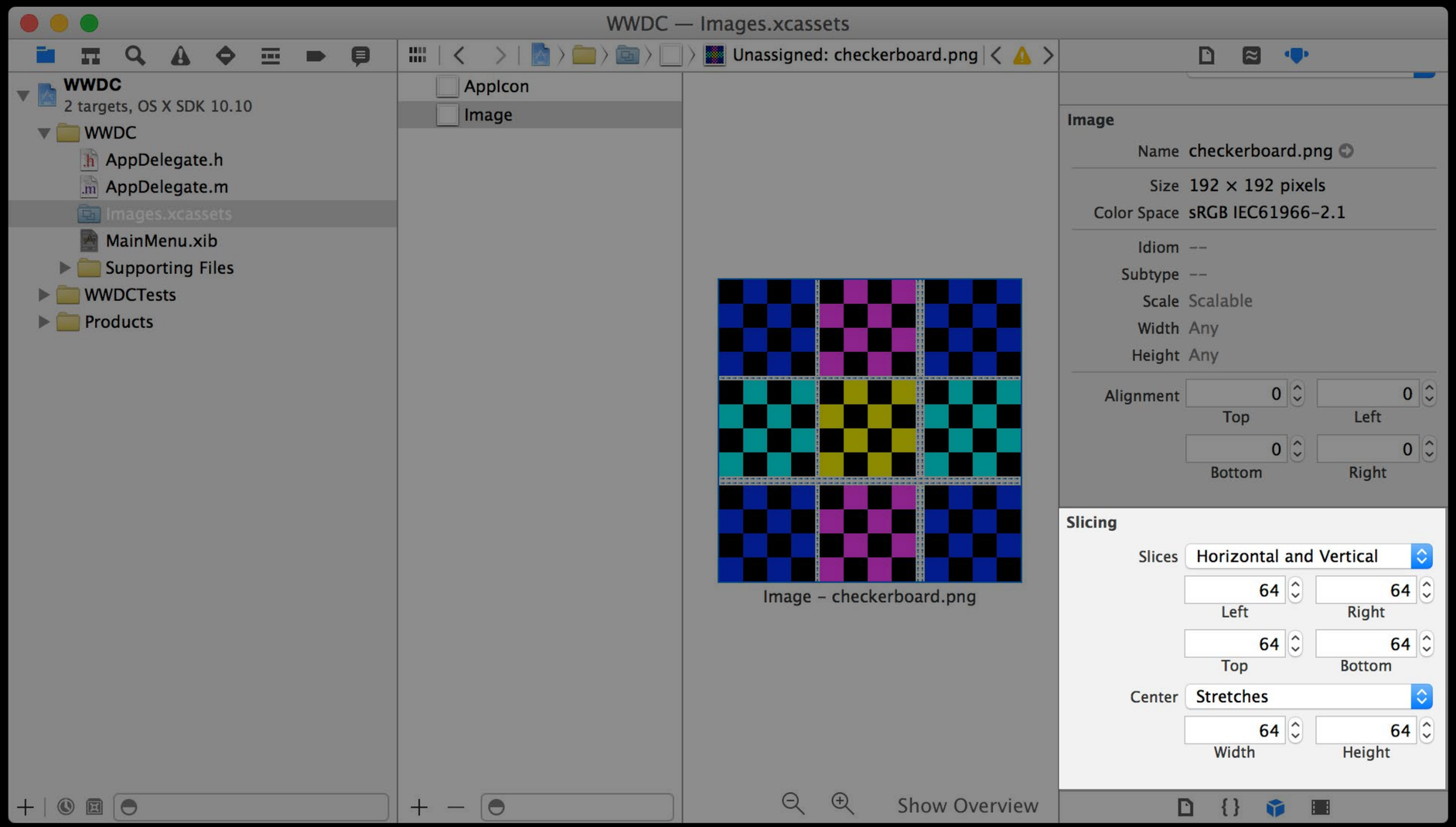
NSImage

Xcode



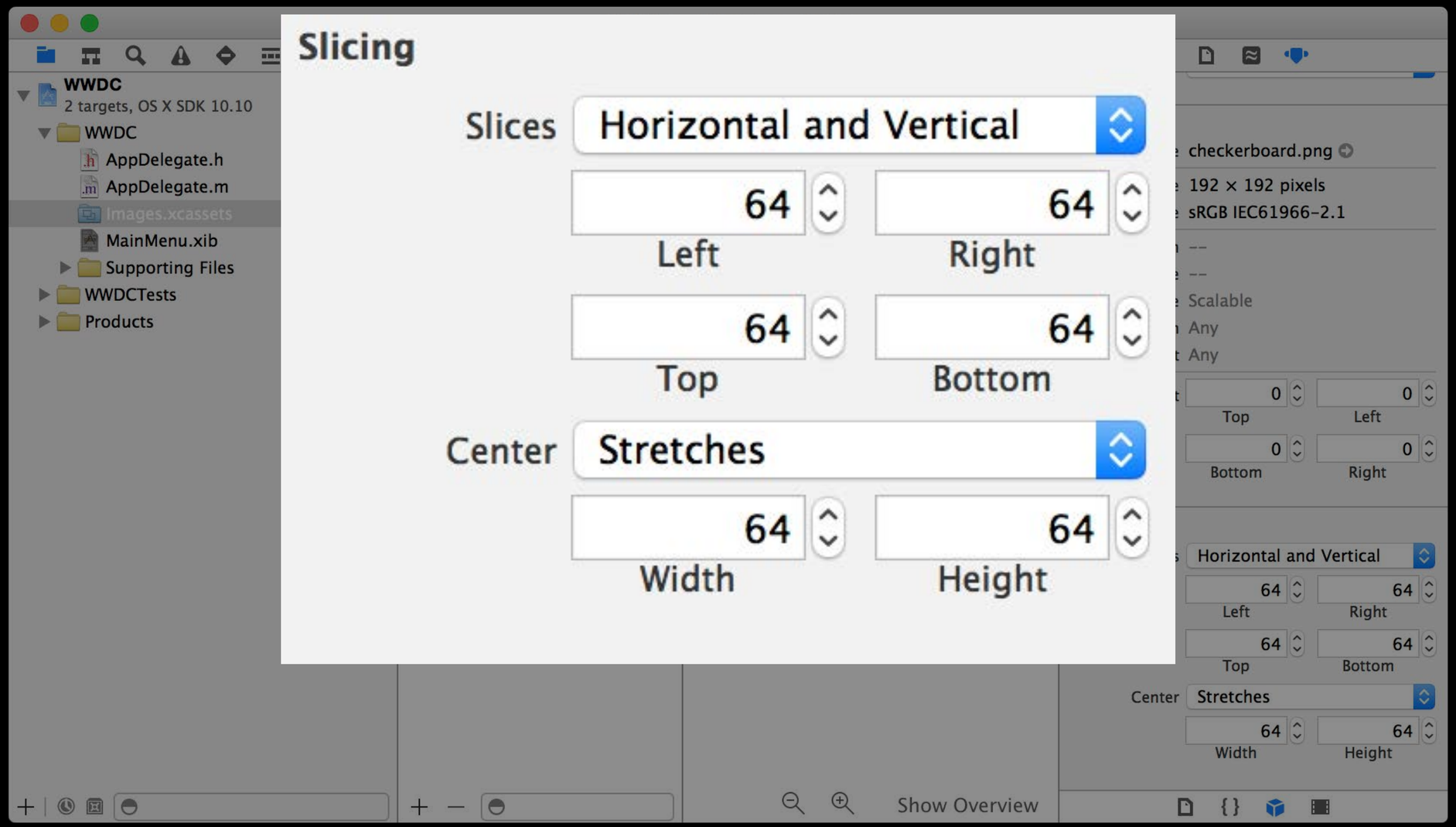
NSImage

Xcode



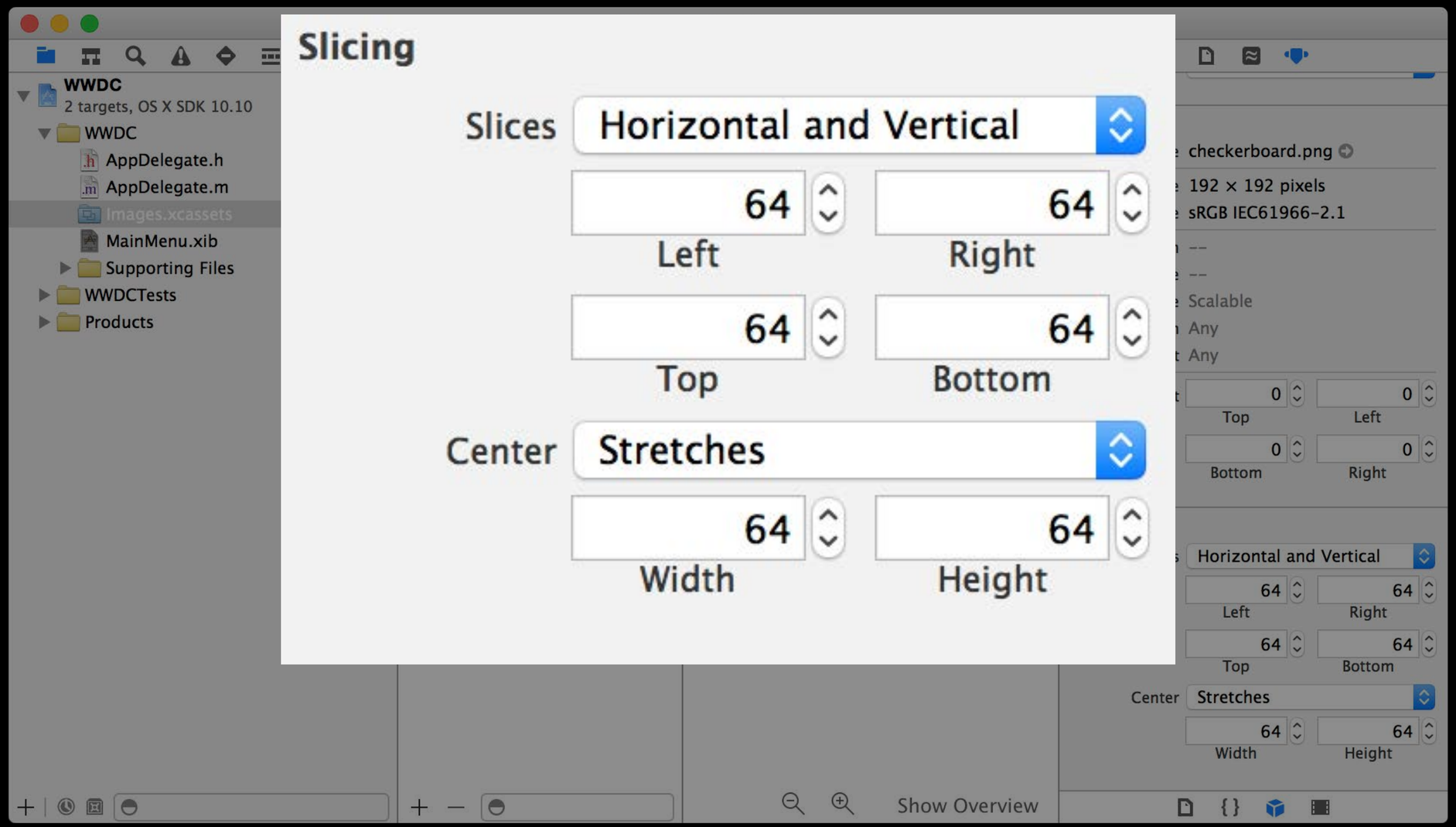
NSImage

Xcode



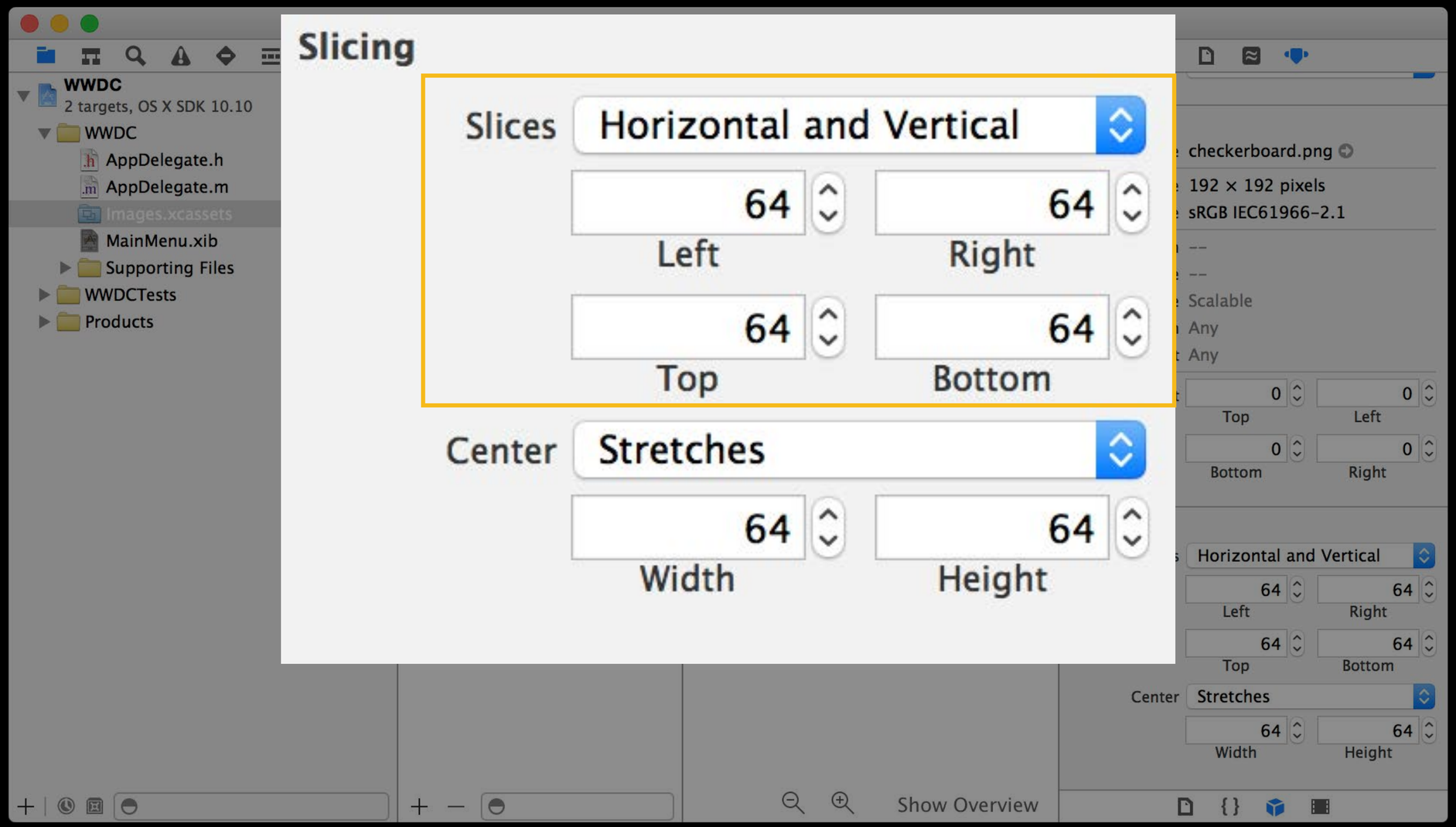
NSImage

Xcode



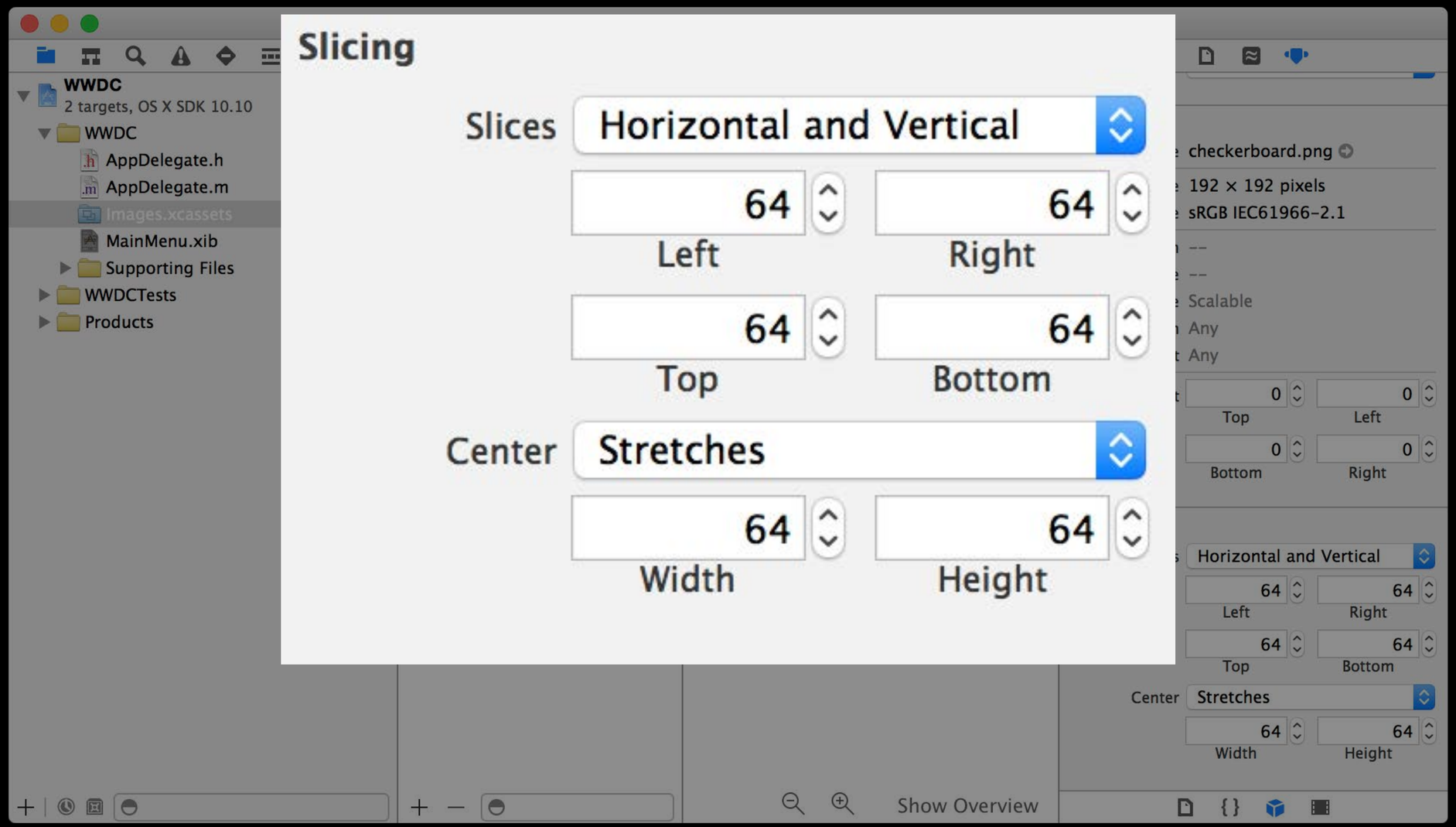
NSImage

Xcode



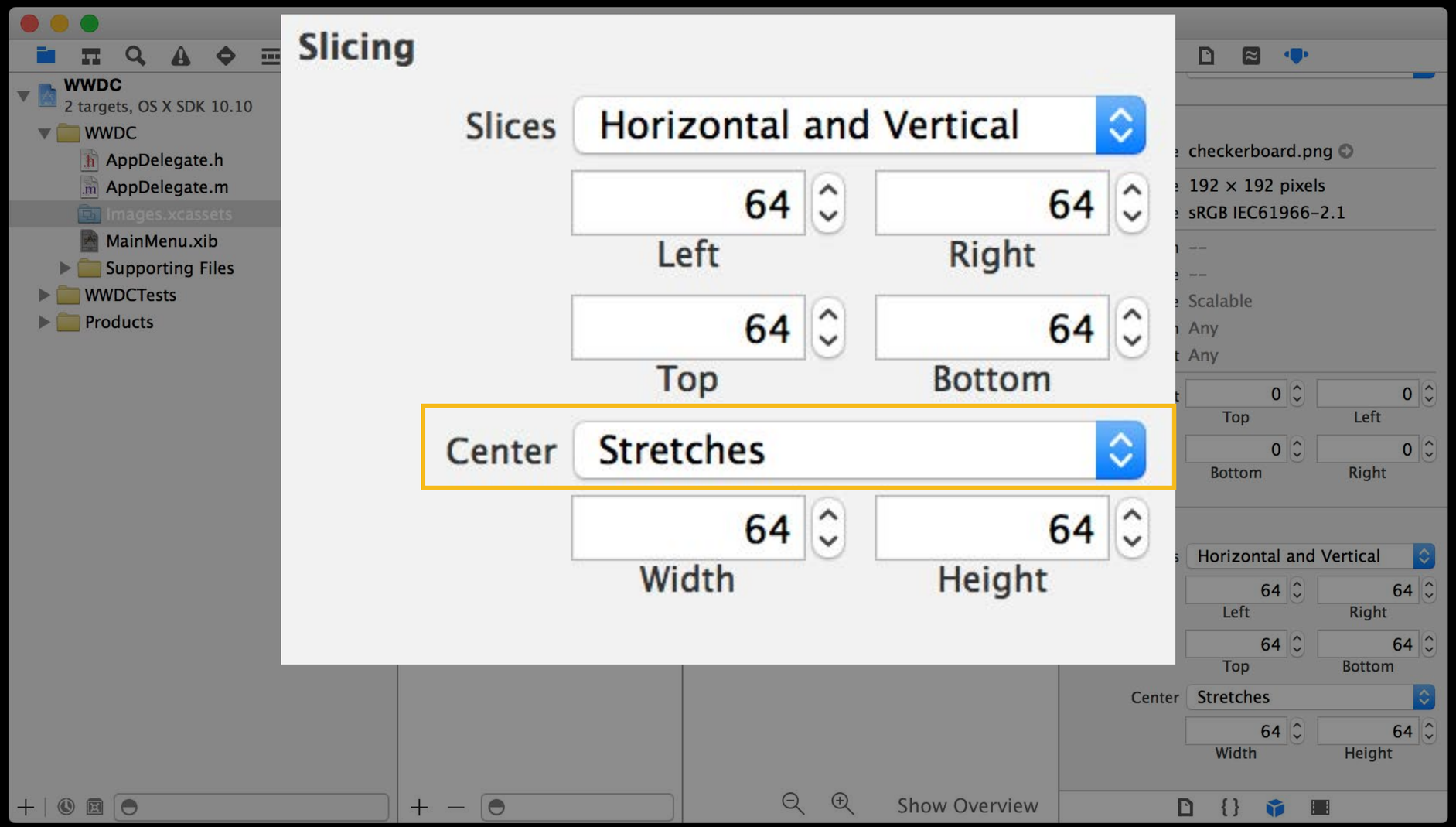
NSImage

Xcode



NSImage

Xcode



NSAppearance

NSAppearance

NSAppearance

```
+[NSAppearance appearanceNamed:]
```

NSAppearance

```
+[NSAppearance appearanceNamed:]
    NSAppearanceNameAqua
```

NSAppearance

```
+[NSAppearance appearanceNamed:]
    NSAppearanceNameAqua
    NSAppearanceNameLightContent
```

NSAppearance

```
+[NSAppearance appearanceNamed:]
    NSAppearanceNameAqua
    NSAppearanceNameLightContent
```

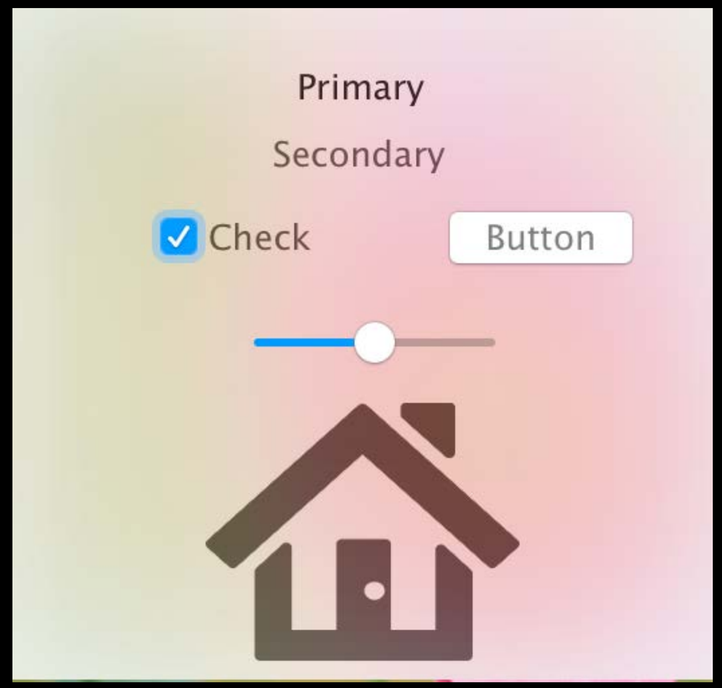
NSAppearance

```
+[NSAppearance appearanceNamed:]
    NSAppearanceNameAqua
    NSAppearanceNameLightContent
    NSAppearanceNameVibrantLight
```

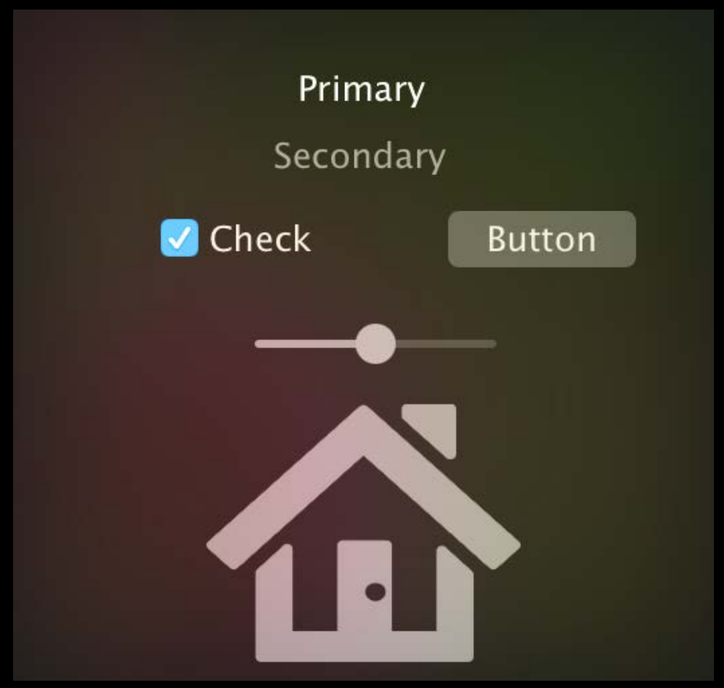
NSAppearance

```
+[NSAppearance appearanceNamed:]
    NSAppearanceNameAqua
    NSAppearanceNameLightContent
    NSAppearanceNameVibrantLight
    NSAppearanceNameVibrantDark
```

NSAppearance



NSAppearanceName`VibrantLight`



NSAppearanceName`VibrantDark`

NSColor

NSColor

NSColor

System colors are appearance sensitive

```
+[NSColor controlBackgroundColor]
+[NSColor controlTextColor]
+[NSColor highlightColor]
+[NSColor selectedTextColor]
```

NSColor

System colors are appearance sensitive

```
+[NSColor controlBackgroundColor]
+[NSColor controlTextColor]
+[NSColor highlightColor]
+[NSColor selectedTextColor]
...
```

NSColor

System colors are appearance sensitive

```
+[NSColor controlBackgroundColor]
+[NSColor controlTextColor]
+[NSColor highlightColor]
+[NSColor selectedTextColor]
...
```

New API

```
+[NSColor labelColor]
+[NSColor secondaryLabelColor]
```

Titlebars and Toolbars

Titlebars and Toolbars

Titlebars and Toolbars

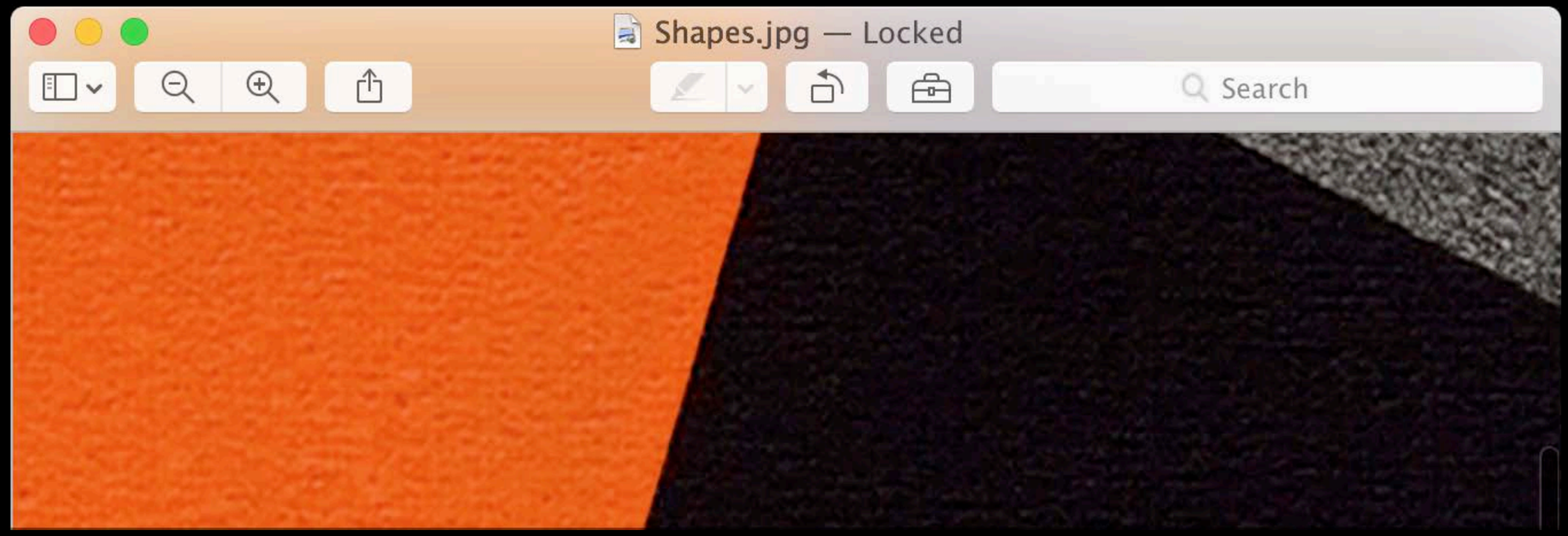
New default appearance

Titlebars and Toolbars

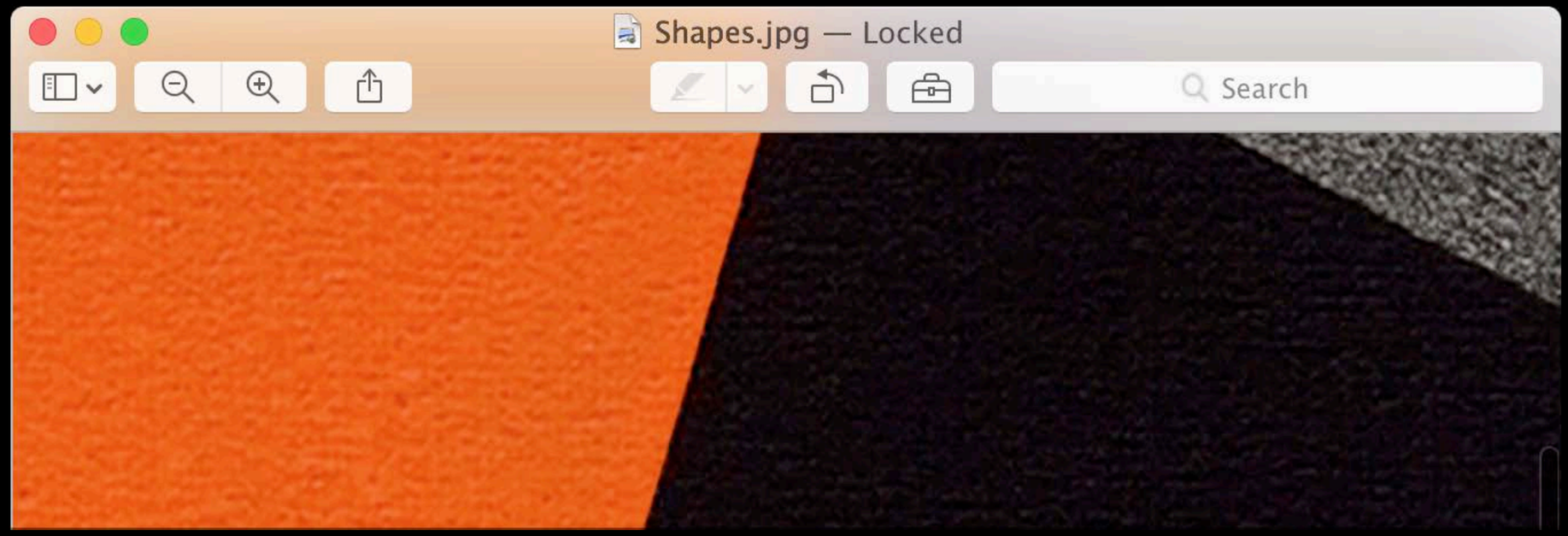
New default appearance

Blurs document content

Titlebars and Toolbars

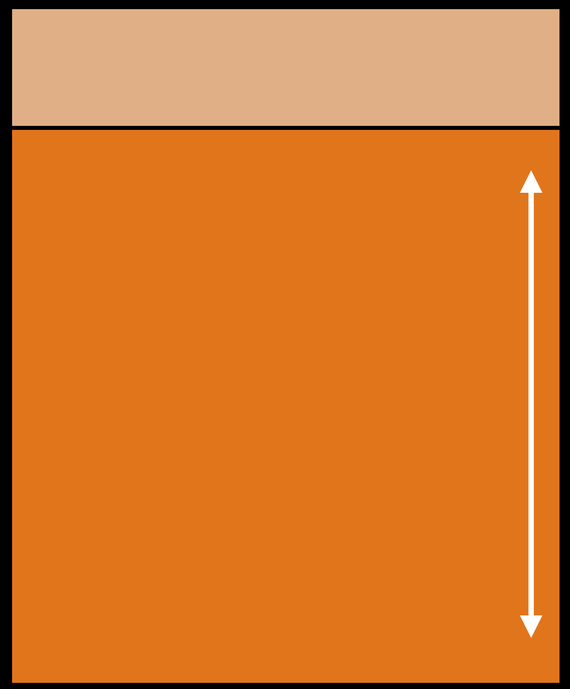


Titlebars and Toolbars

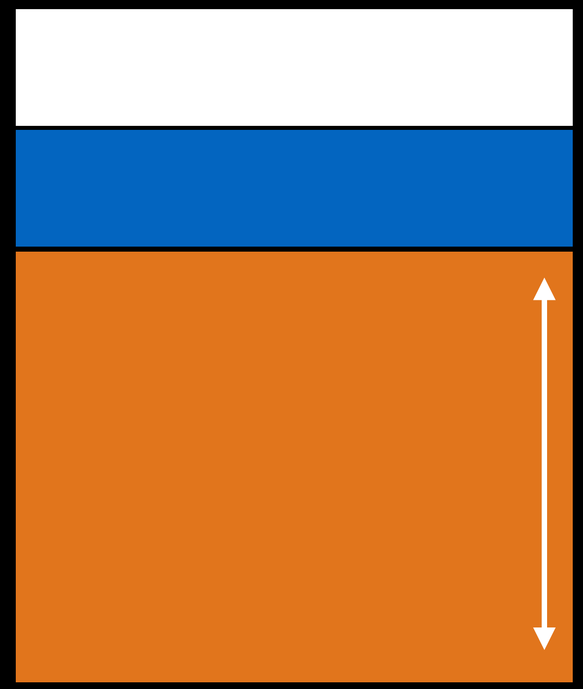


Titlebars and Toolbars

Extends scroll views adjacent to the titlebar or toolbar



Scroll view adjacent to titlebar



Scroll view not adjacent to titlebar

Titlebars and Toolbars

Titlebars and Toolbars

Some types of content require help

Titlebars and Toolbars

Some types of content require help

- Movie views

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers
  - Movie layers

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers
  - Movie layers
  - OpenGL layers

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers
  - Movie layers
  - OpenGL layers
  - Layers with 3D transforms

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers
  - Movie layers
  - OpenGL layers
  - Layers with 3D transforms
  - Layers with CIFilters

Titlebars and Toolbars

Some types of content require help

- Movie views
- OpenGL views
  - Some types of layers
  - Movie layers
  - OpenGL layers
  - Layers with 3D transforms
  - Layers with CIFilters

These need to be explicitly placed behind the titlebar and toolbar

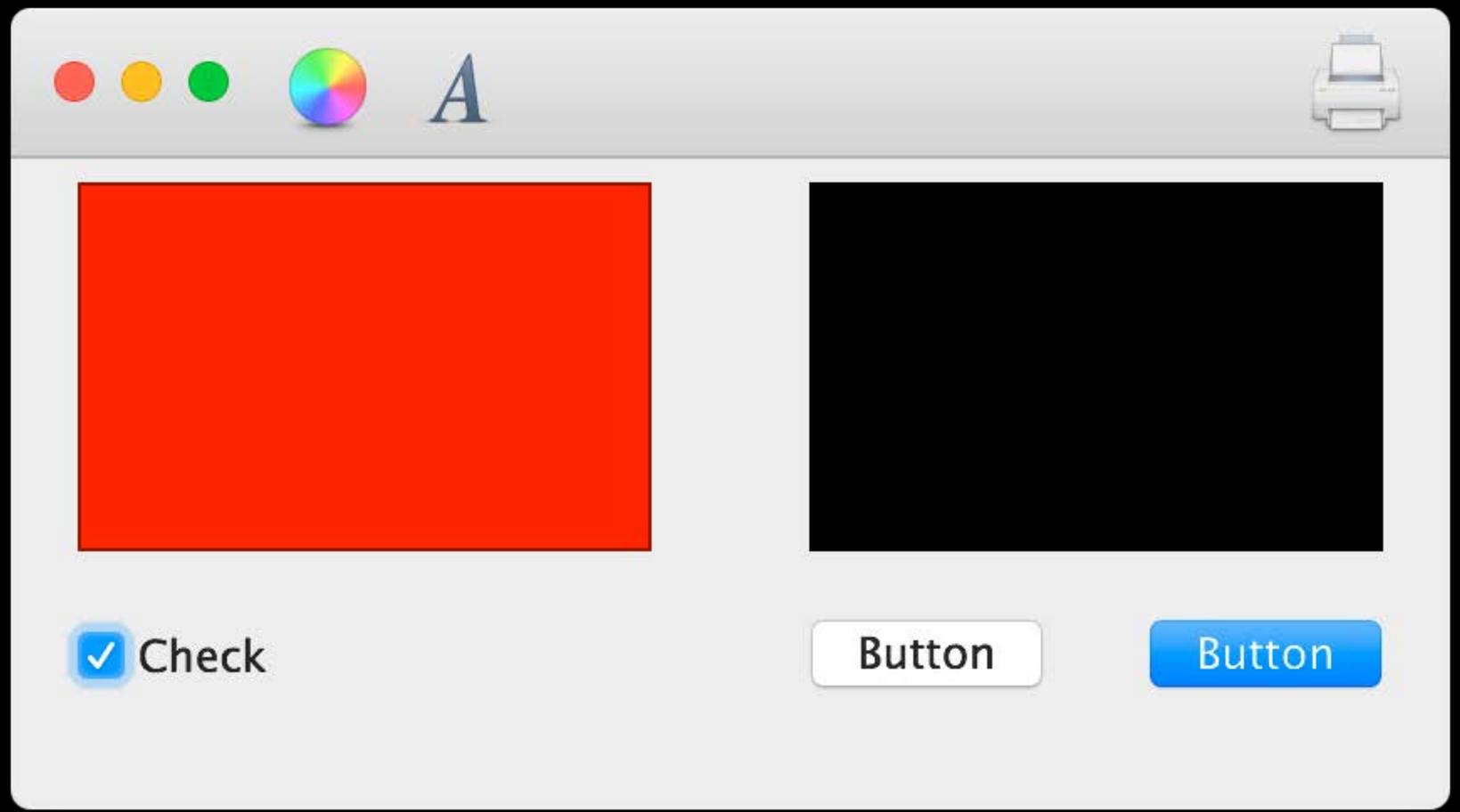
Titlebars and Toolbars

Titlebars and Toolbars

New window style mask: `NSFullSizeContentViewWindowMask`

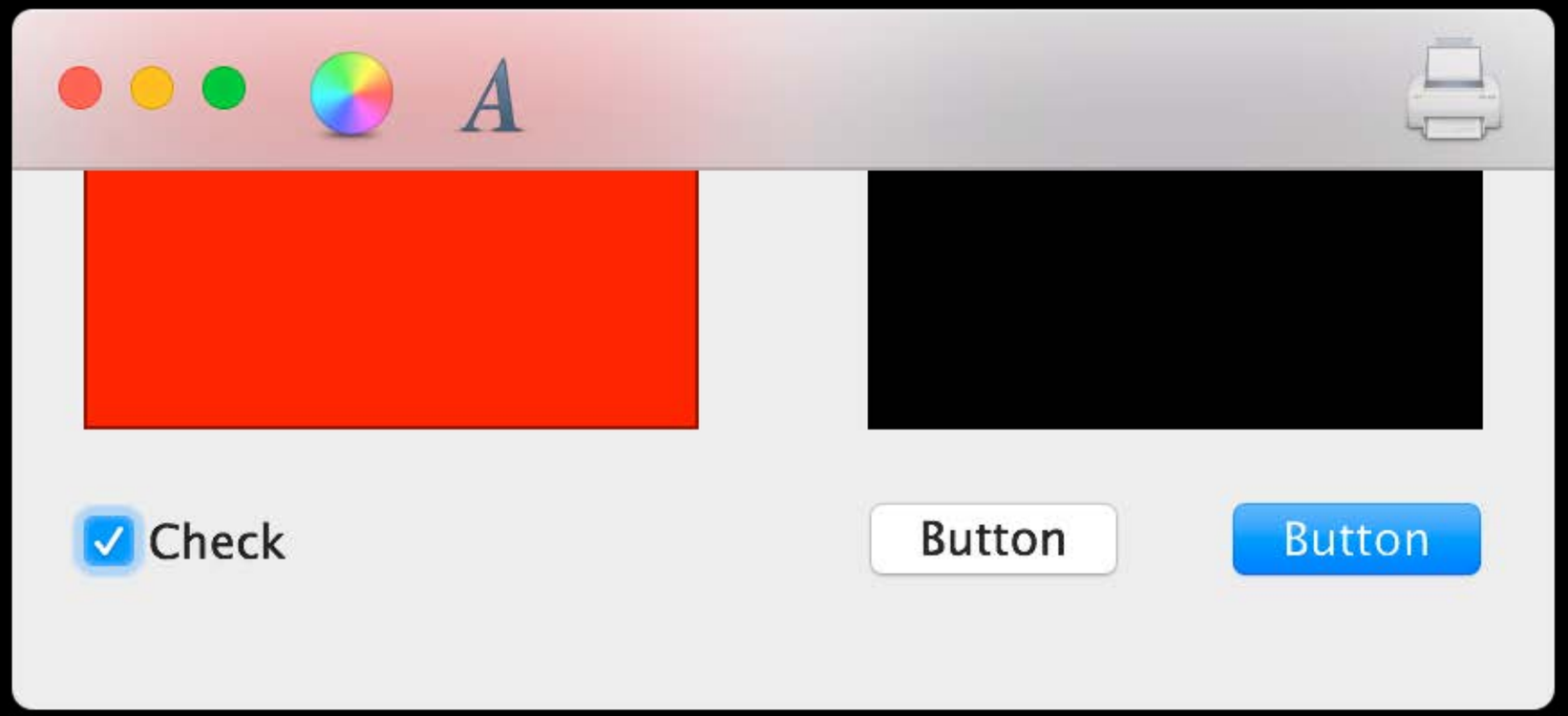
Titlebars and Toolbars

```
window.styleMask &= ~NSFullSizeContentViewWindowMask
```

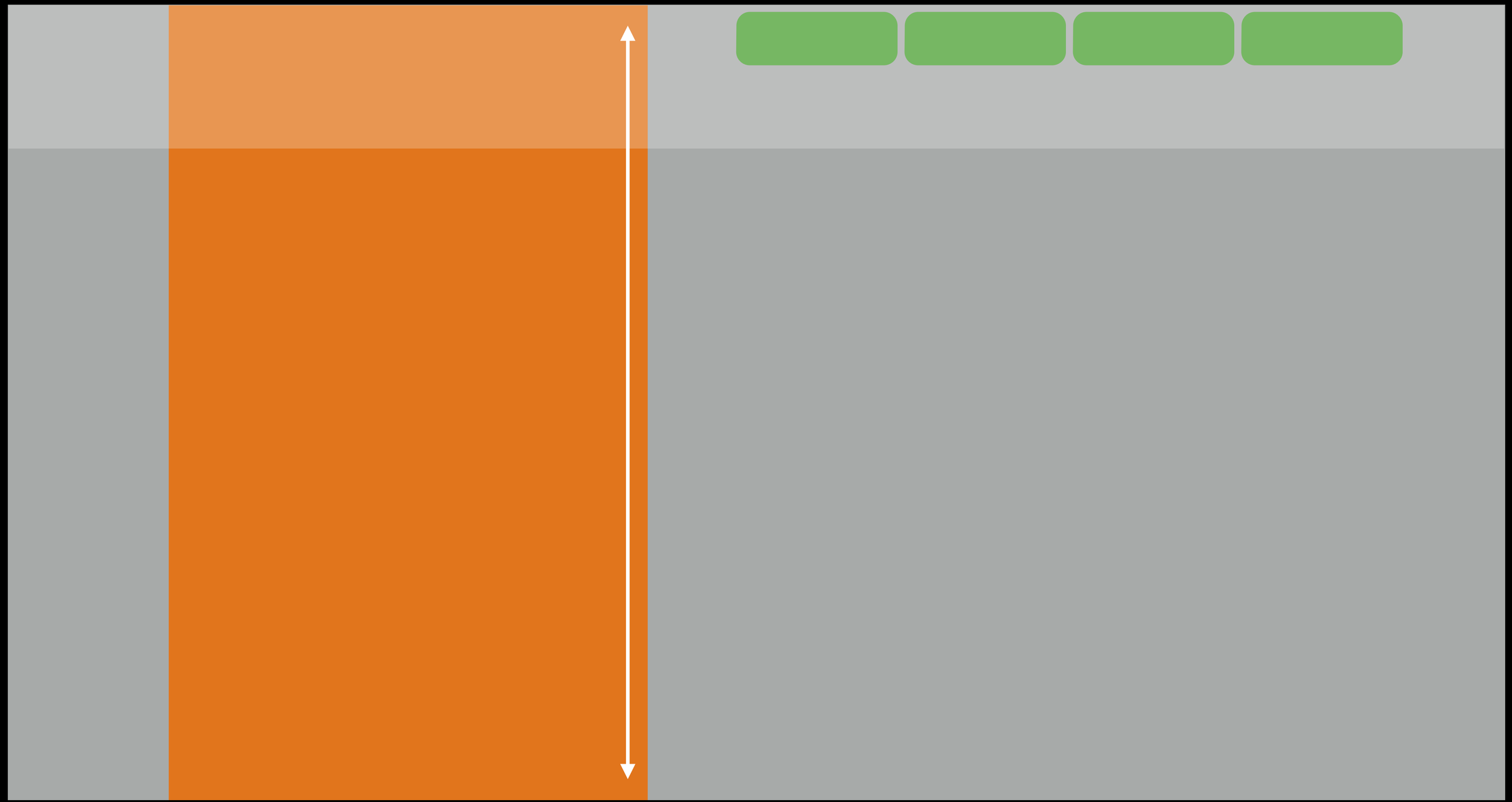


Titlebars and Toolbars

```
window.styleMask |= NSFullSizeContentViewWindowMask
```



Titlebars and Toolbars



Titlebars and Toolbars

Titlebars and Toolbars

This can be solved using Auto Layout

Titlebars and Toolbars

This can be solved using Auto Layout

```
@interface NSWindow : NSResponder …
@property (readonly) id contentLayoutGuide;

…
```

Titlebars and Toolbars

This can be solved using Auto Layout

```
@interface NSWindow : NSResponder …
@property (readonly) id contentLayoutGuide;

…
```

```
NSLayoutConstraint *topEdgeConstraint = [NSLayoutConstraint
constraintWithItem:myView attribute:NSLayoutAttributeTop
relatedBy:NSLayoutRelationEqual toItem:window.contentLayoutGuide
attribute:NSLayoutAttributeTop];
```

Titlebars and Toolbars

This can be solved using Auto Layout

```
@interface NSWindow : NSResponder …
@property (readonly) id contentLayoutGuide;

…
```

```
NSLayoutConstraint *topEdgeConstraint = [NSLayoutConstraint
constraintWithItem:myView attribute:NSLayoutAttributeTop
relatedBy:NSLayoutRelationEqual toItem:window.contentLayoutGuide
attribute:NSLayoutAttributeTop];

constraint.active = YES;
```

Titlebars and Toolbars

Titlebars and Toolbars

```
@interface NSWindow : NSResponder …
@property (readonly) NSRect contentLayoutRect;
…
```

Titlebars and Toolbars

```
@interface NSWindow : NSResponder …
@property (readonly) NSRect contentLayoutRect;
…
```

Key-value observing compliant

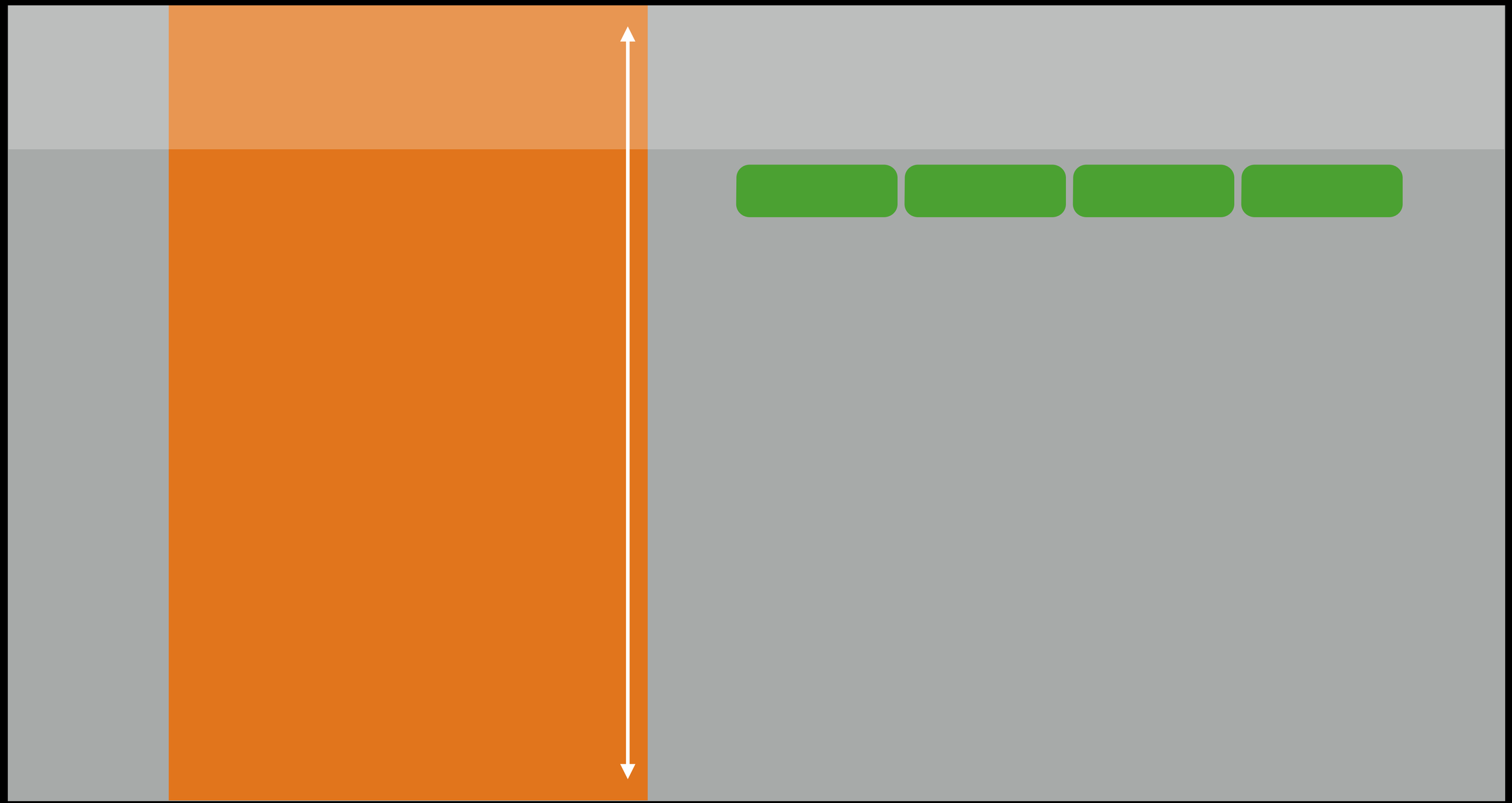
Titlebars and Toolbars

```
@interface NSWindow : NSResponder …
@property (readonly) NSRect contentLayoutRect;
…
```

Key-value observing compliant

Observe it! It can change

Titlebars and Toolbars



Titlebars and Toolbars

Titlebars and Toolbars

```
@interface NSScrollView : NSView …
@property NSEdgeInsets contentInsets; // Adds buffer zones around edges
```

Titlebars and Toolbars

```
@interface NSScrollView : NSView …
@property NSEdgeInsets contentInsets; // Adds buffer zones around edges

CGFloat offset = NSHeight(window.frame) - NSMaxY(window.contentLayoutRect);
scrollView.contentInsets = NSEdgeInsetsMake(offset, 0, 0, 0);
```

Titlebars and Toolbars

```
@interface NSScrollView : NSView …
@property NSEdgeInsets contentInsets; // Adds buffer zones around edges

CGFloat offset = NSHeight(window.frame) - NSMaxY(window.contentLayoutRect);
scrollView.contentInsets = NSEdgeInsetsMake(offset, 0, 0, 0);
```

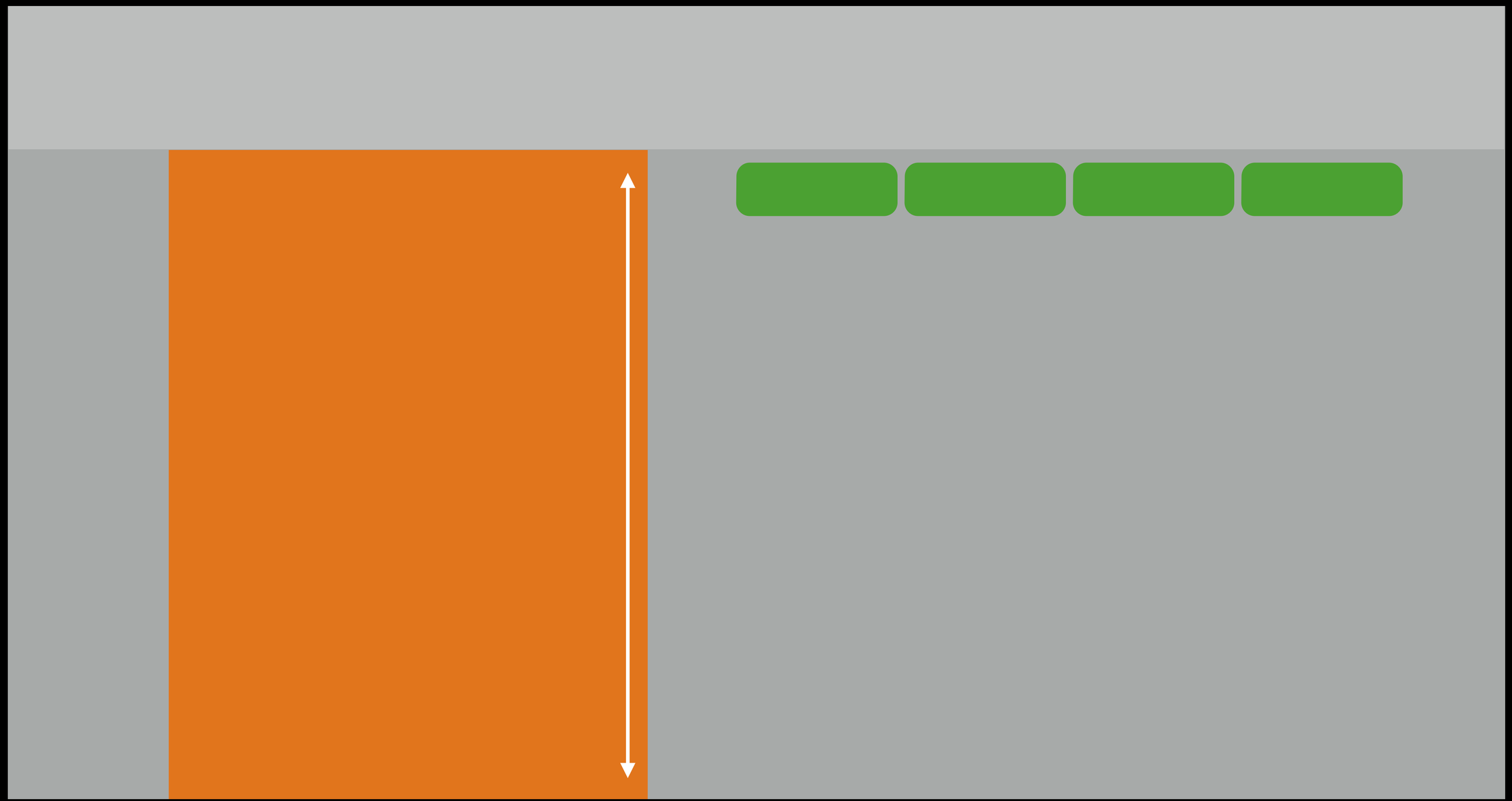
```
@interface NSScrollView : NSView …
@property BOOL automaticallyAdjustsContentInsets;
…
```

Titlebars and Toolbars

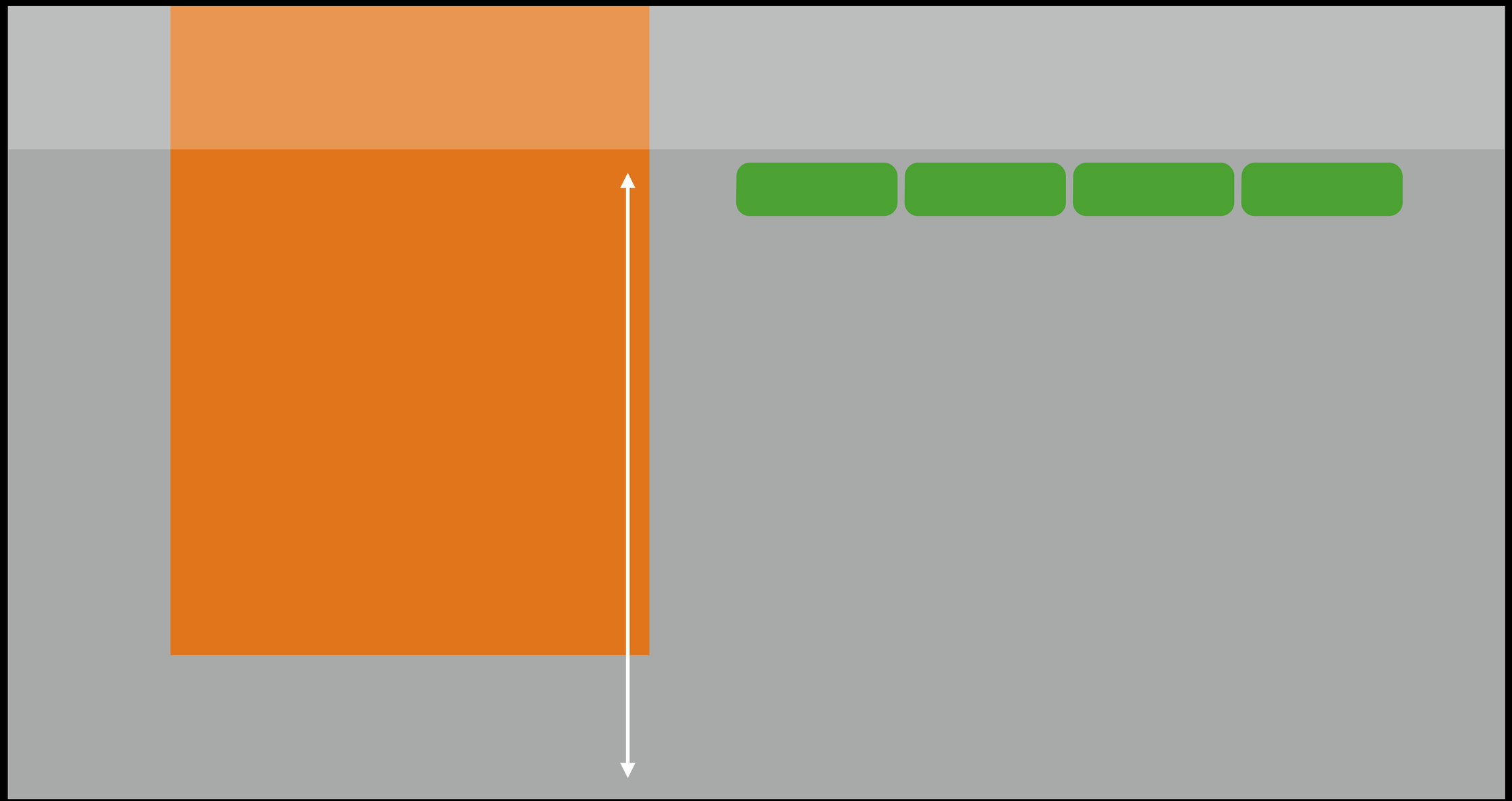
```
@interface NSScrollView : NSView …
@property NSEdgeInsets contentInsets; // Adds buffer zones around edges

CGFloat offset = NSHeight(window.frame) - NSMaxY(window.contentLayoutRect);
scrollView.contentInsets = NSEdgeInsetsMake(offset, 0, 0, 0);
```

```
@interface NSScrollView : NSView …
@property BOOL automaticallyAdjustsContentInsets;
…

scrollView.automaticallyAdjustsContentInsets = YES
```

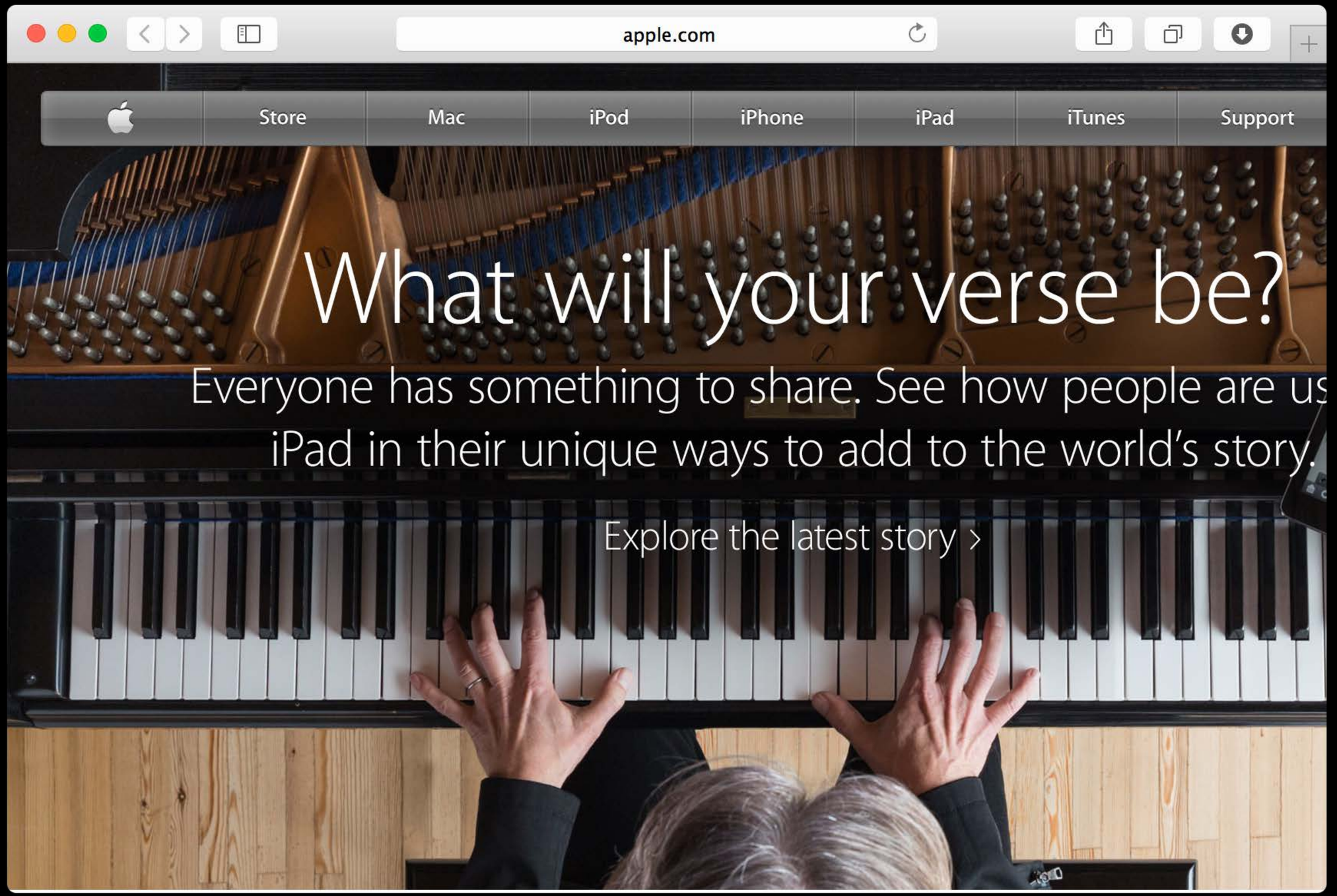
Titlebars and Toolbars



Titlebars and Toolbars



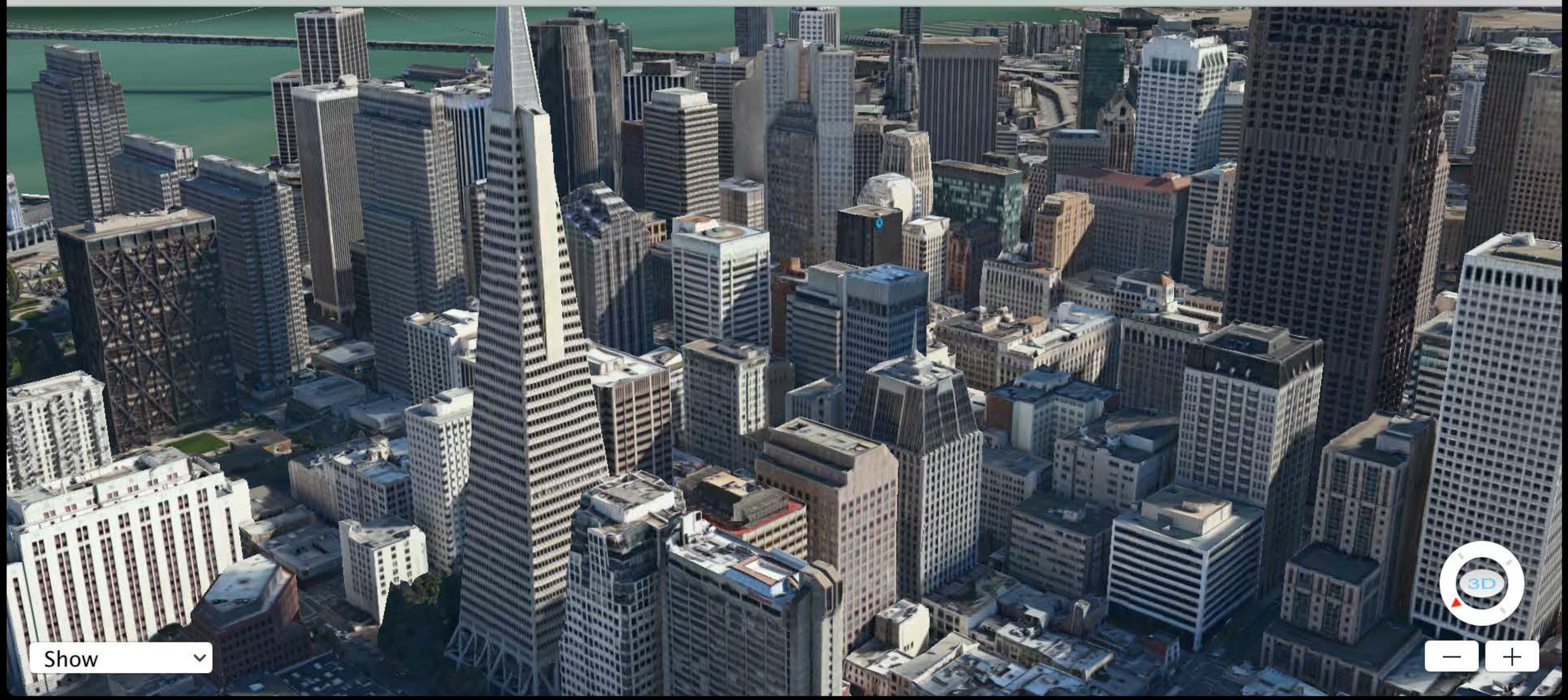
Titlebars and Toolbars


apple.com
Store
Mac
iPod
iPhone
iPad
iTunes
Support
What will your verse be?
Everyone has something to share. See how people are us
iPad in their unique ways to add to the world's story.
Explore the latest story ›

Directions

Search or enter an address

Map | Satellite



Titlebars and Toolbars

Titlebars and Toolbars

```
enum NSWindowTitleVisibility {
    NSWindowTitleVisible
    NSWindowTitleHidden
    NSWindowTitleHiddenWhenActive
}
```

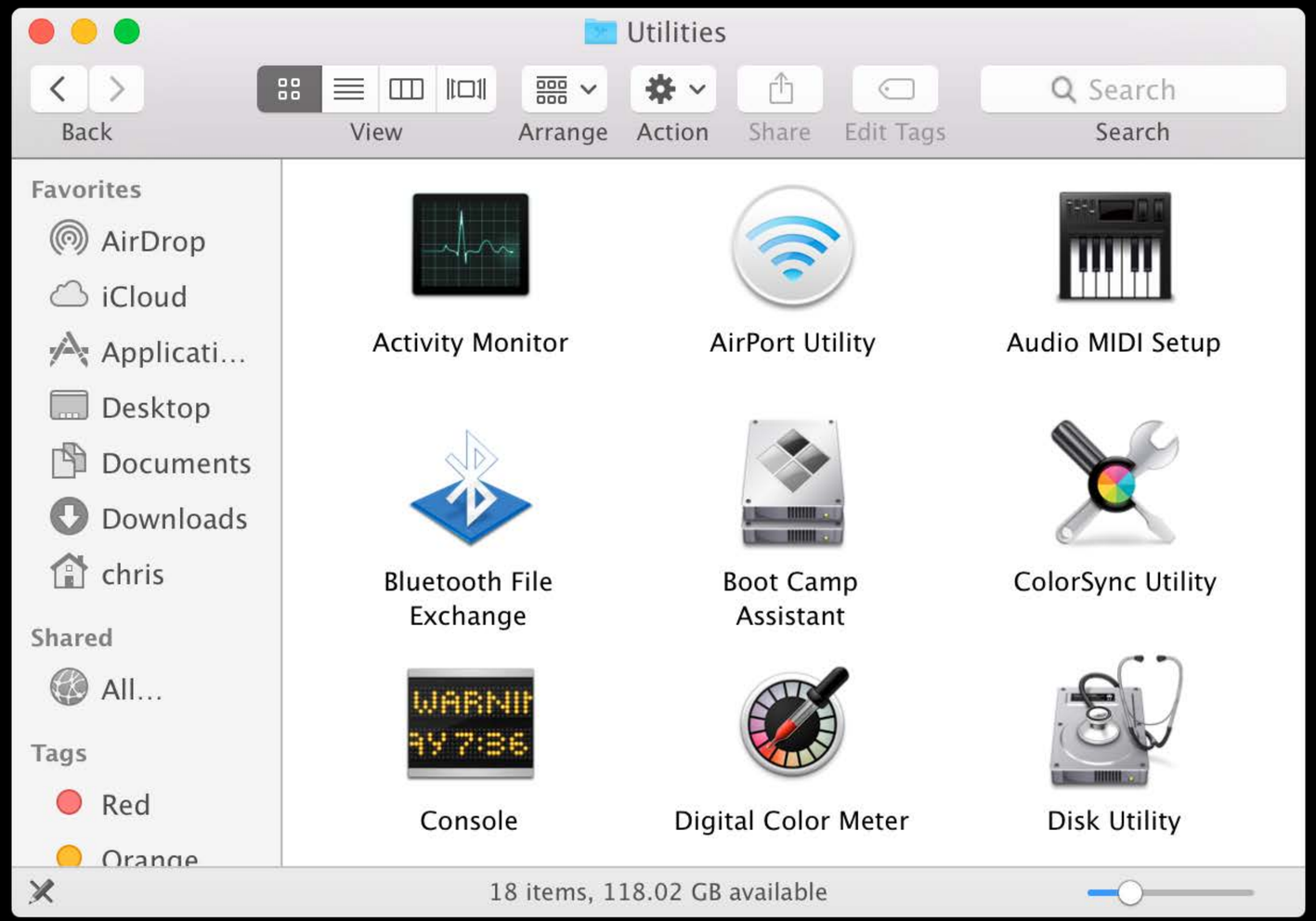
Titlebars and Toolbars

```
enum NSWindowTitleVisibility {
    NSWindowTitleVisible
    NSWindowTitleHidden
    NSWindowTitleHiddenWhenActive
}
```

```
@interface NSWindow : NSResponder …
@property NSWindowTitleVisibility titleVisibility;
…
```

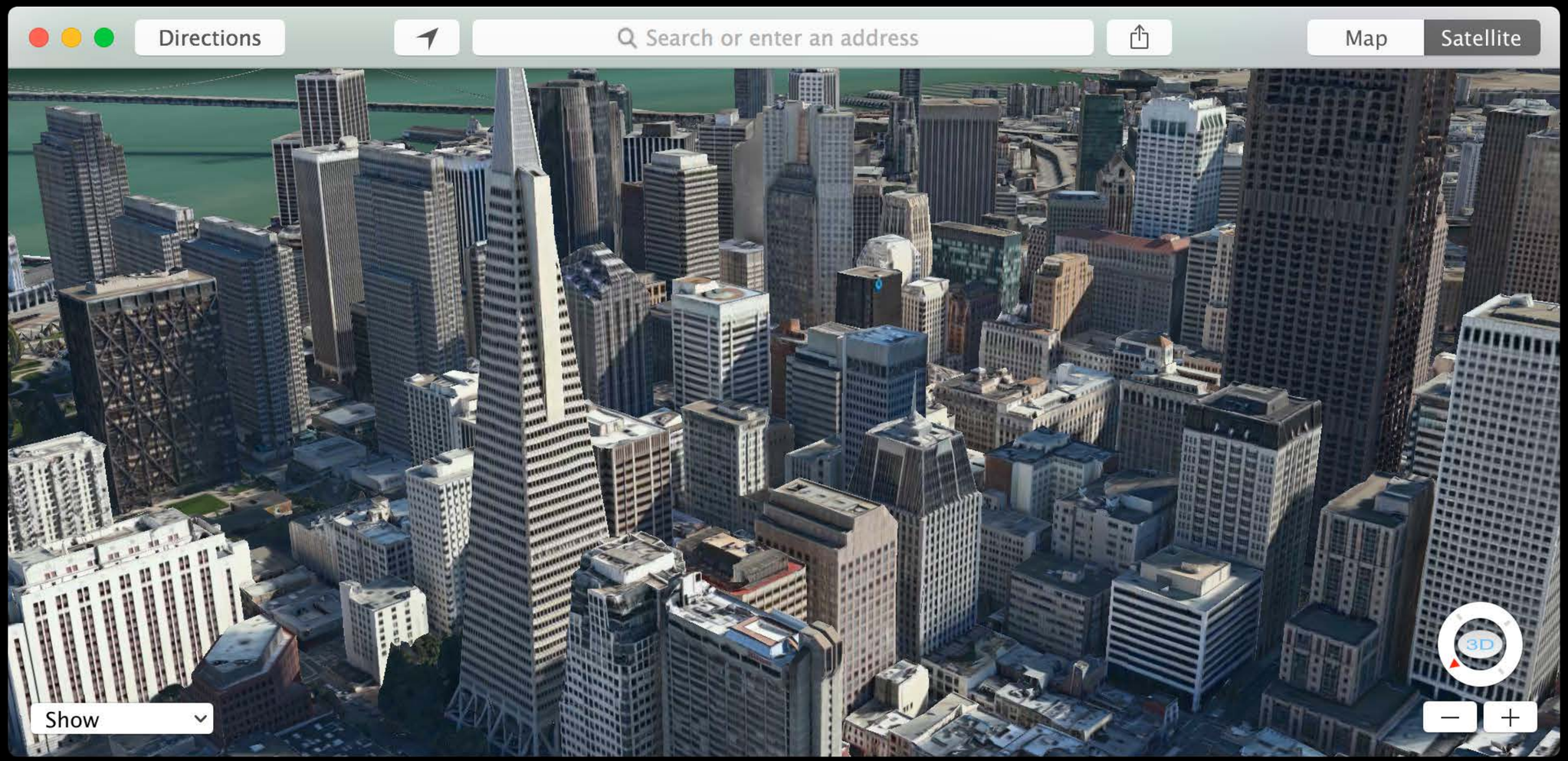
Titlebars and Toolbars

```
window.titleVisibility = NSWindowTitleVisible
```



Titlebars and Toolbars

```
window.titleVisibility = NSWindowTitleHidden
```



Titlebars and Toolbars

Search

Scheduled

On My Mac

Reminders

Add List

Reminders

- [ ] Call Corbin
- [ ] Get conference badge
- [ ] Take screenshots

Search

NSColor system colors ar…
Yesterday

NSAppearance summary:
Yesterday

NSImage supports slicing!
Yesterday

May 28, 2014, 9:06 AM

NSColor system colors are now appearance-sensitive

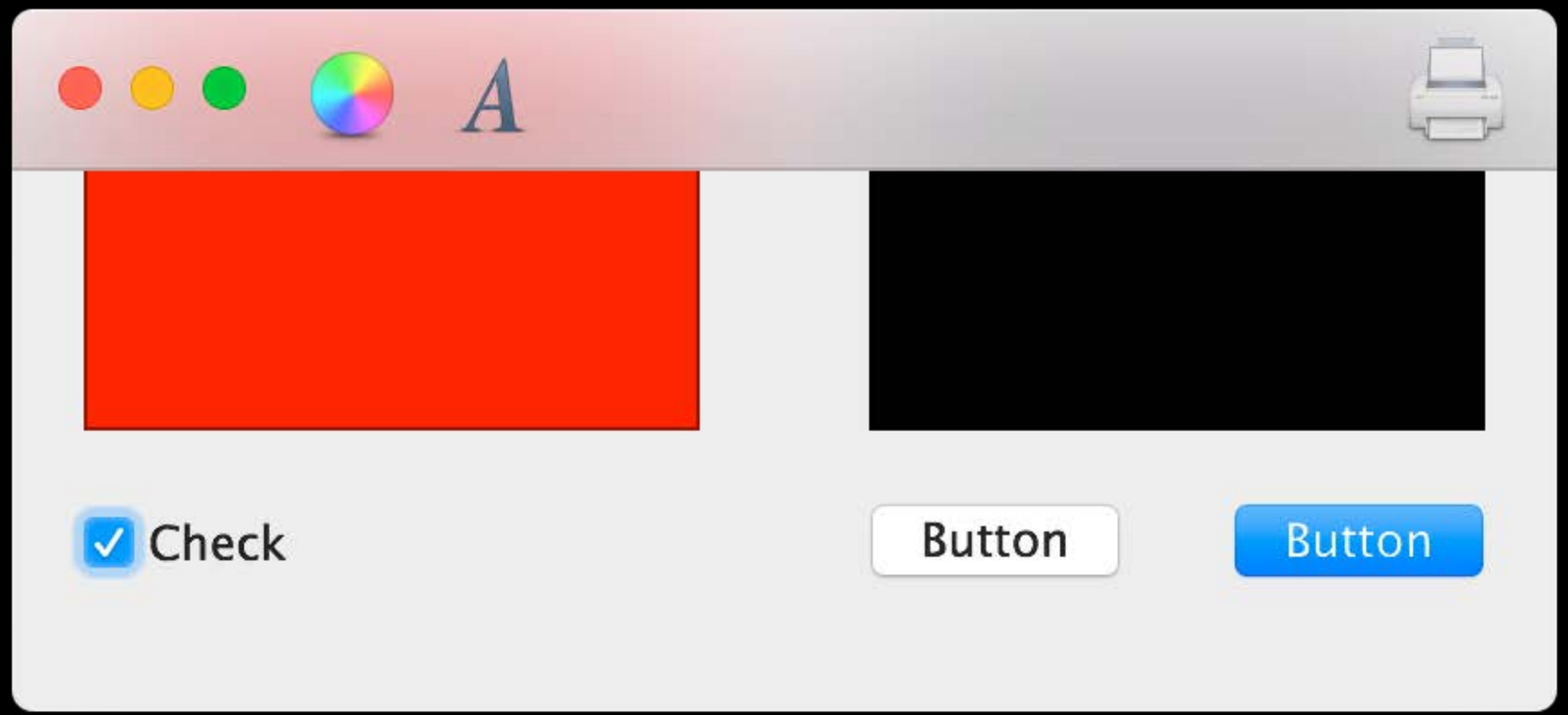
Titlebars and Toolbars

Titlebars and Toolbars

```
@interface NSWindow : NSResponder …
@property BOOL titlebarAppearsTransparent;
…
```

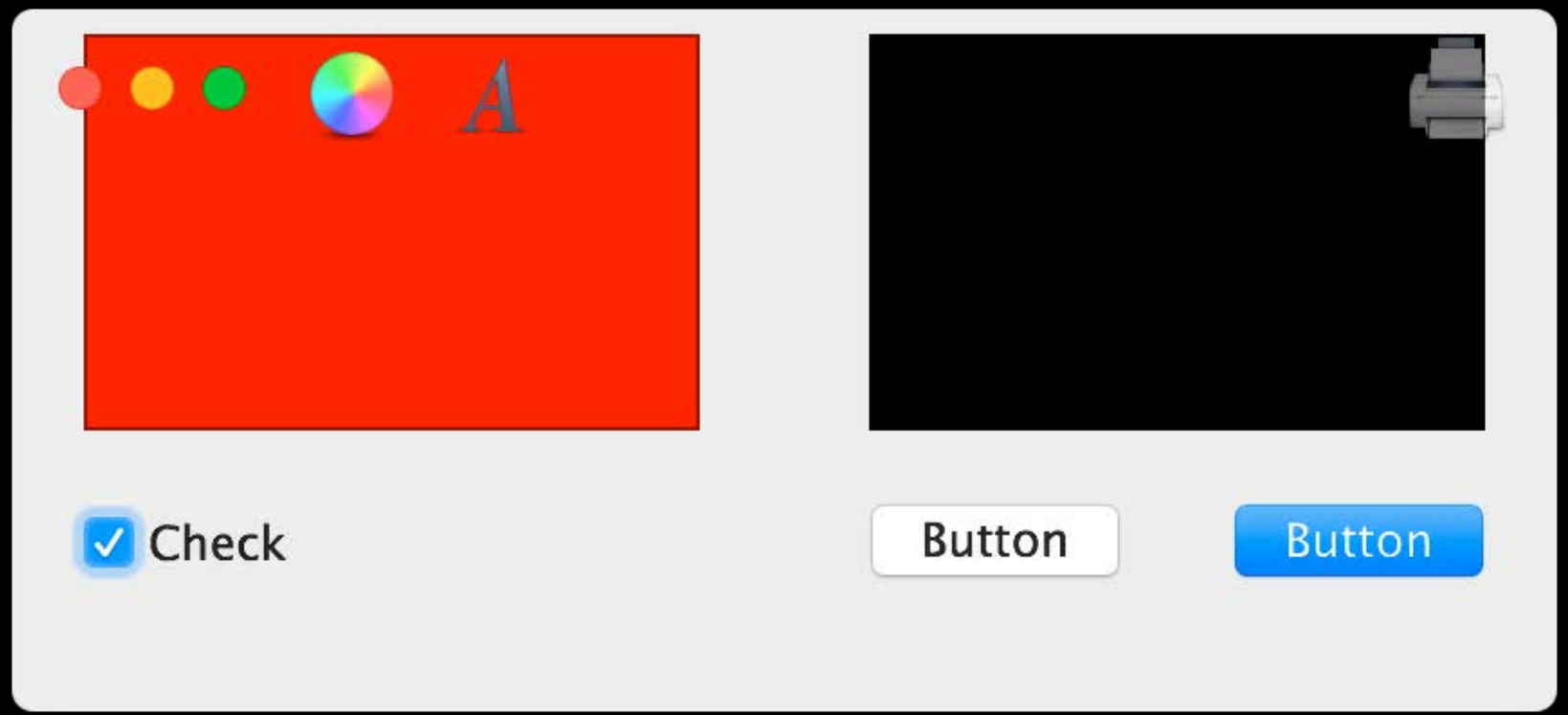
Titlebars and Toolbars

```
NSWindow.titlebarAppearsTransparent = NO
```



Titlebars and Toolbars

```
NSWindow.titlebarAppearsTransparent = YES
```



New Visual Effects and Vibrancy

Corbin Dunn
AppKit Software Engineer

New Visual Effects and Vibrancy

NSVisualEffectView

NSAppearance and Materials

Vibrancy and Custom Controls

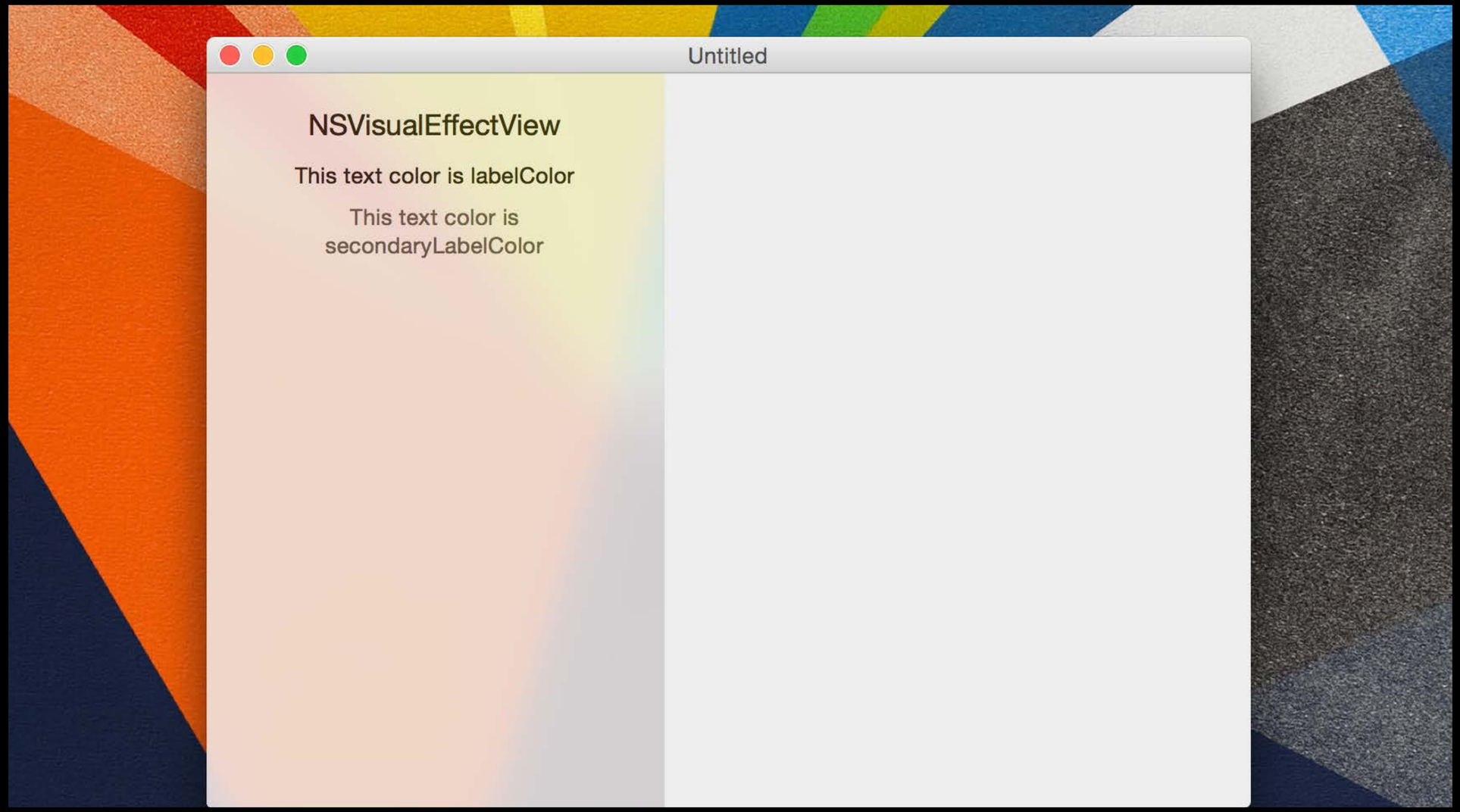
Standard AppKit Views and Vibrancy

Demo

VisualEffectPlayground

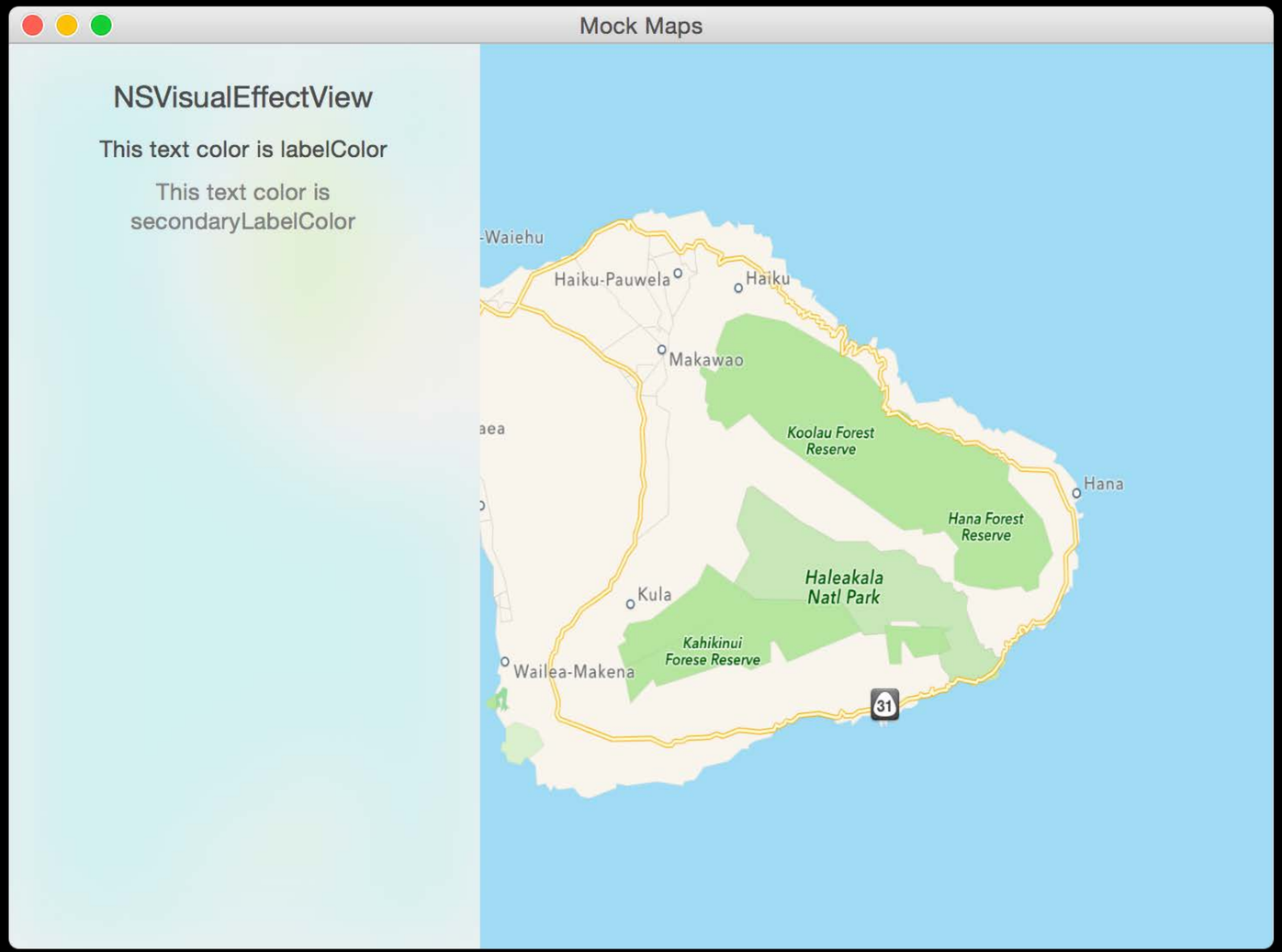
New Visual Effects

NSVisualEffectView



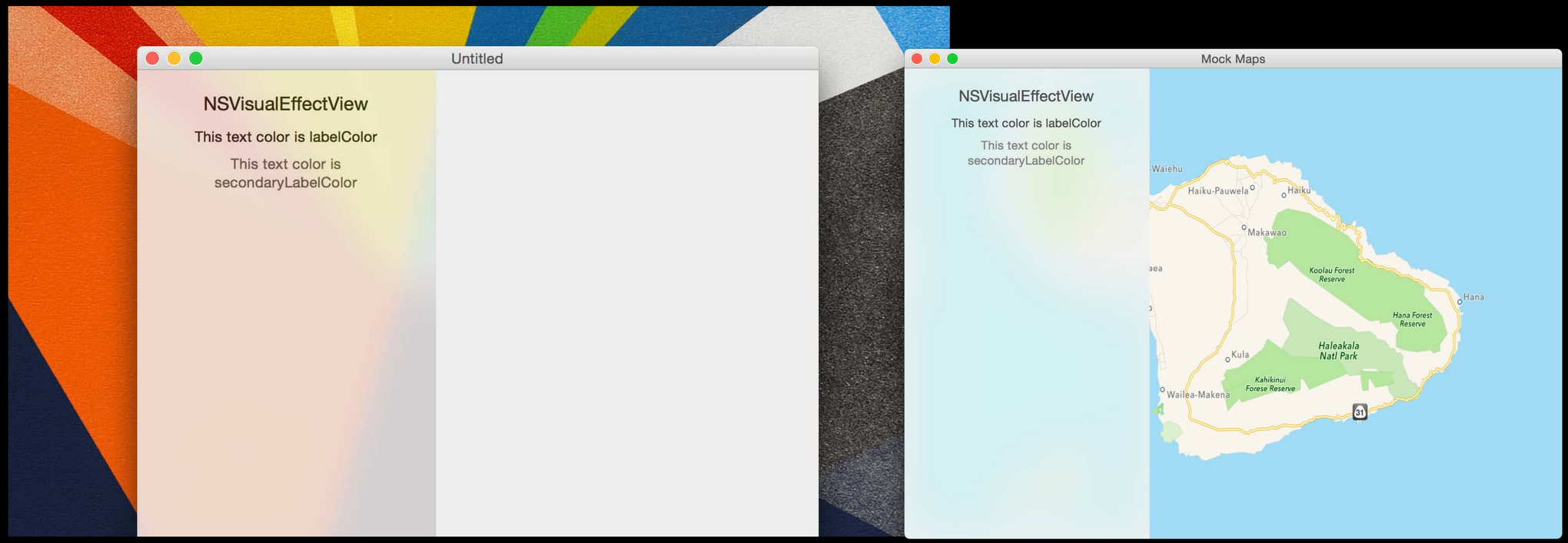
New Visual Effects

NSVisualEffectView



New Visual Effects

NSVisualEffectView



Behind window blending

Within window blending

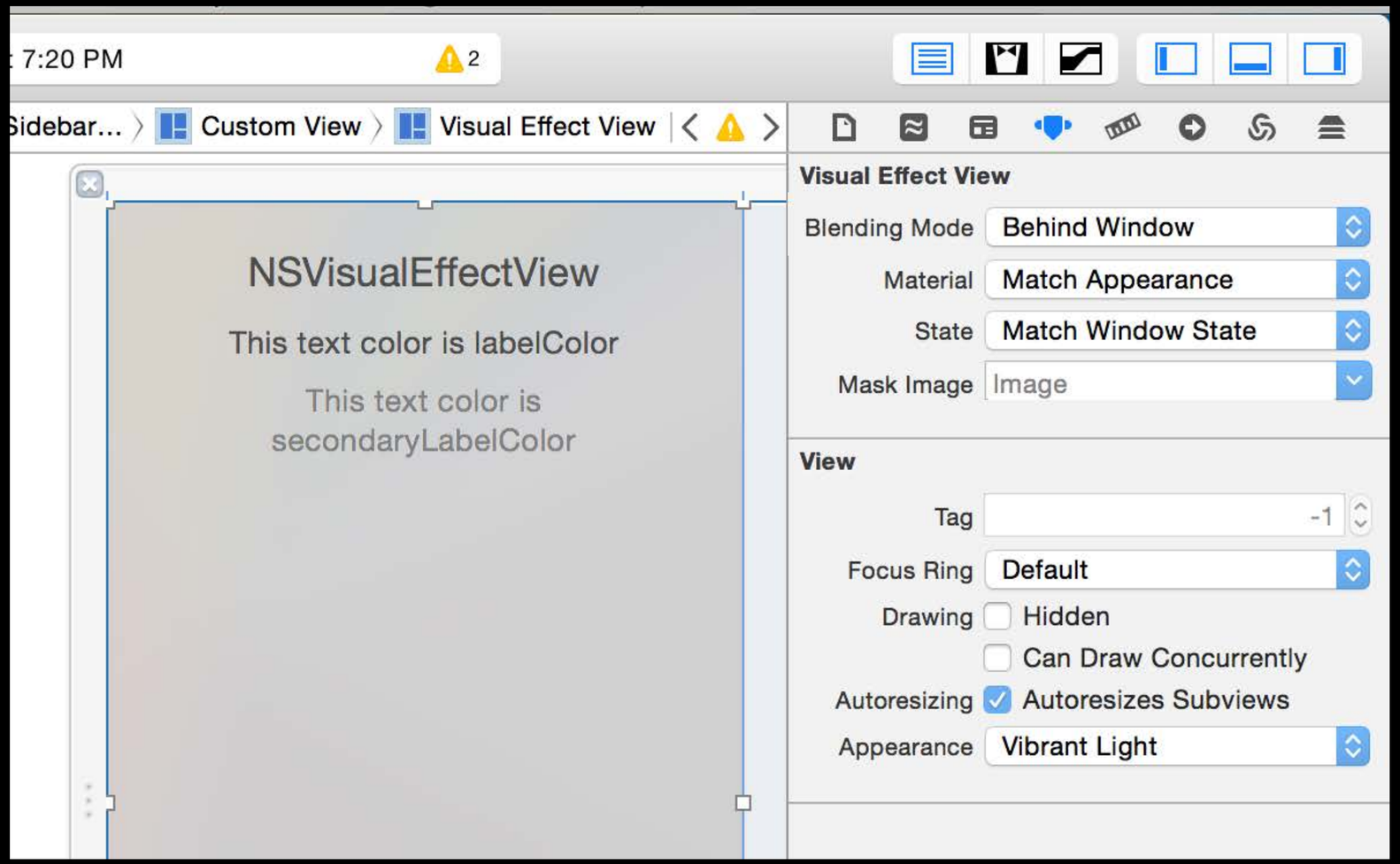
New Visual Effects

NSVisualEffectView API

```
typedef NS_ENUM(NSInteger, NSVisualEffectBlendingMode) {
    NSVisualEffectBlendingModeBehindWindow,
    NSVisualEffectBlendingModeWithinWindow,
};

@interface NSVisualEffectView …

@property NSVisualEffectBlendingMode blendingMode;

@end
```

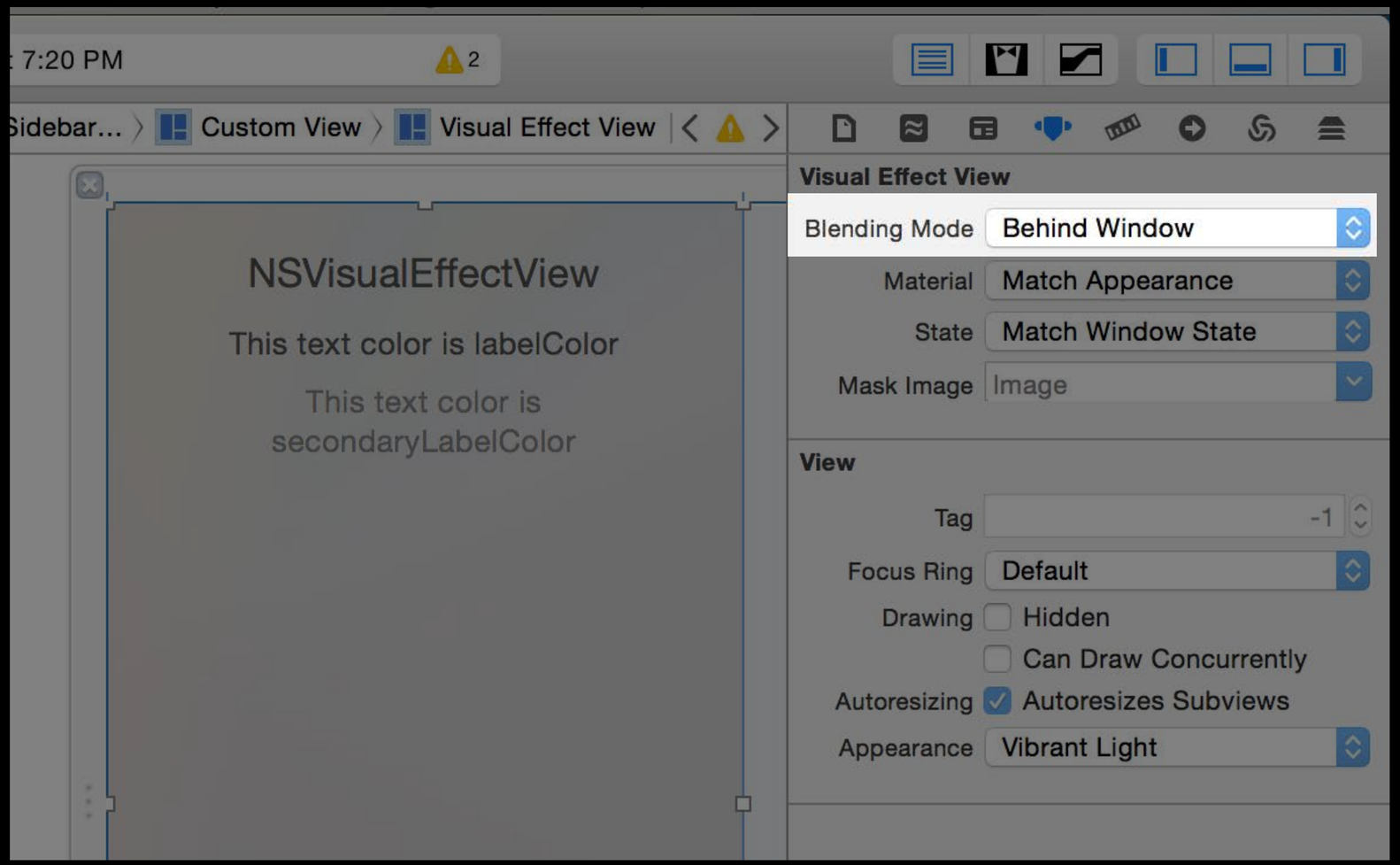
New Visual Effects

Set the Blending Mode in IB



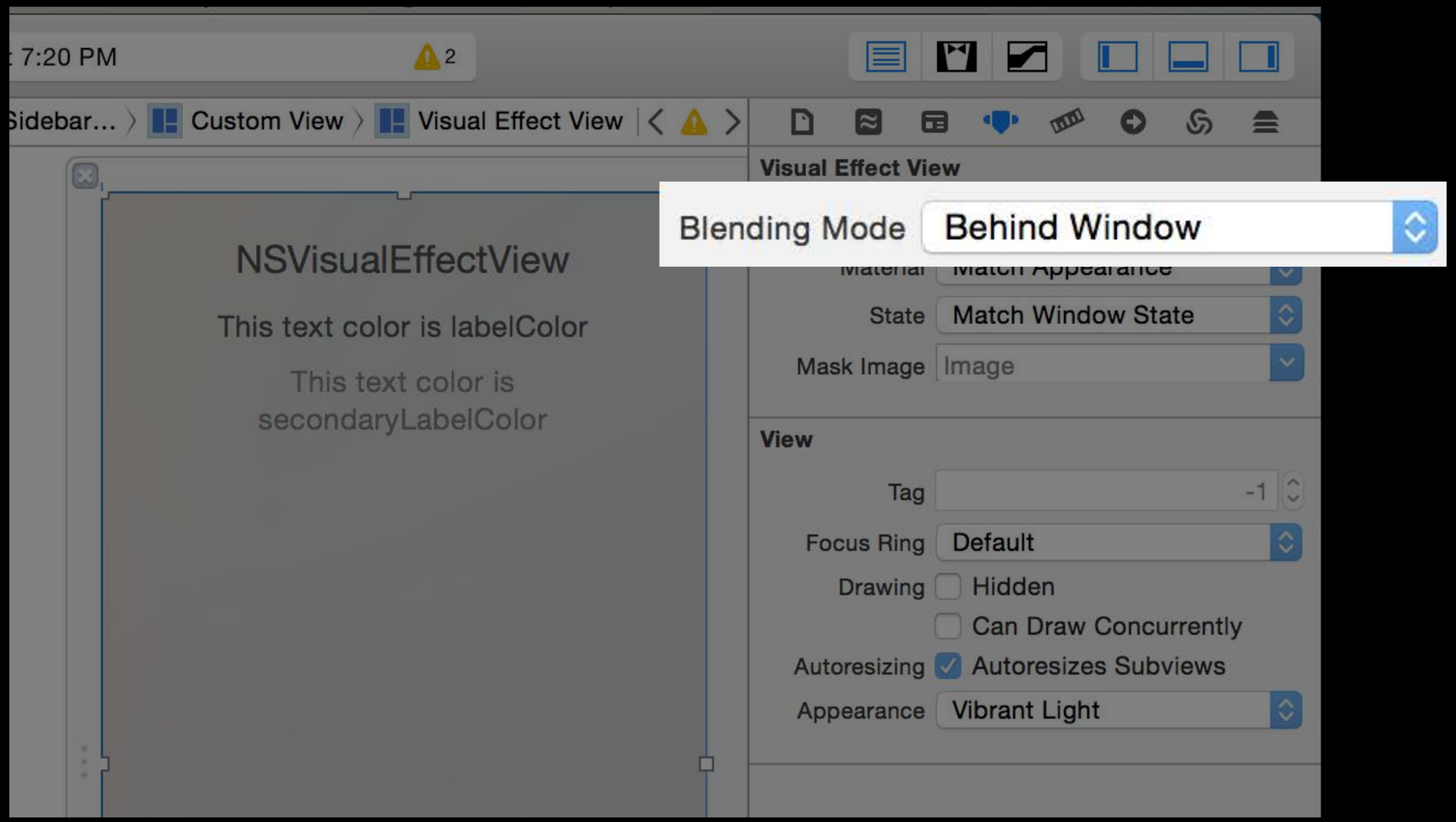
New Visual Effects

Set the Blending Mode in IB



New Visual Effects

Set the Blending Mode in IB



New Visual Effects

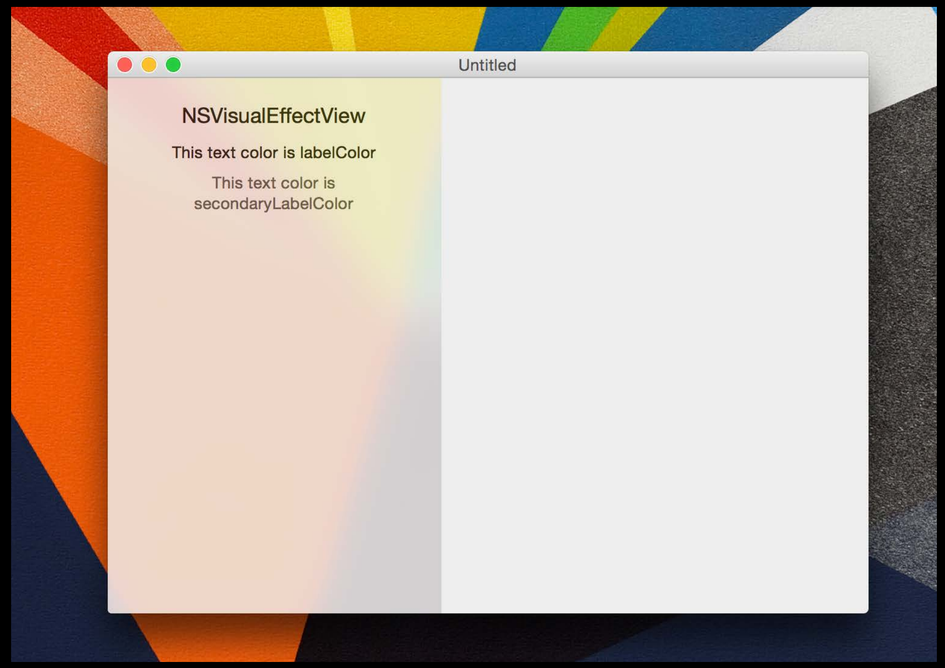
How “behind window” blending works

Implemented in Core Graphics/Window Server

Moving the window does not redraw any window content

NSVisualEffectView defines regions for the Window Server to show with the effect

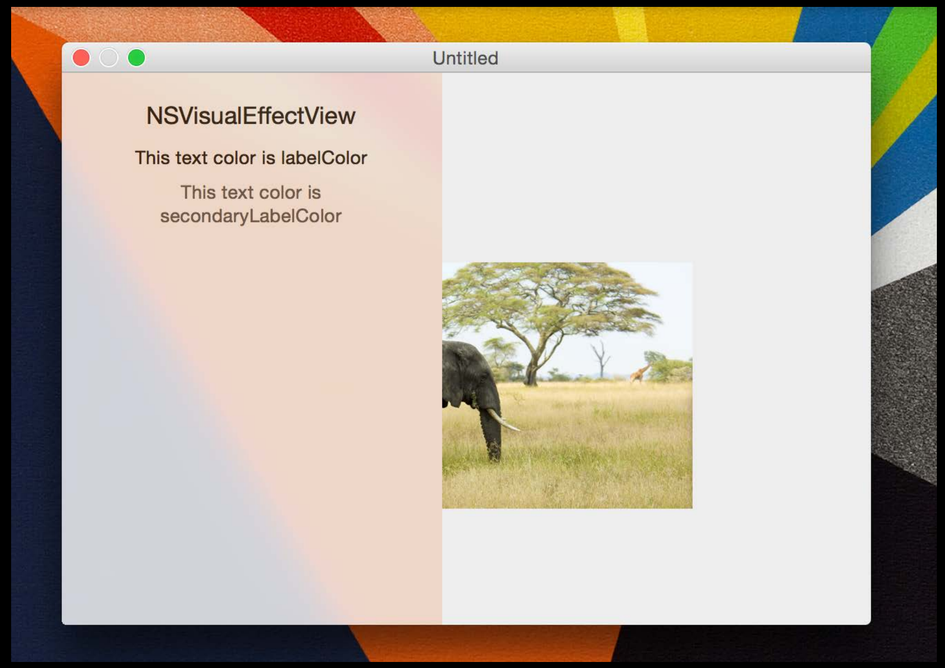
Implements Vibrancy



New Visual Effects

How “behind window” blending works

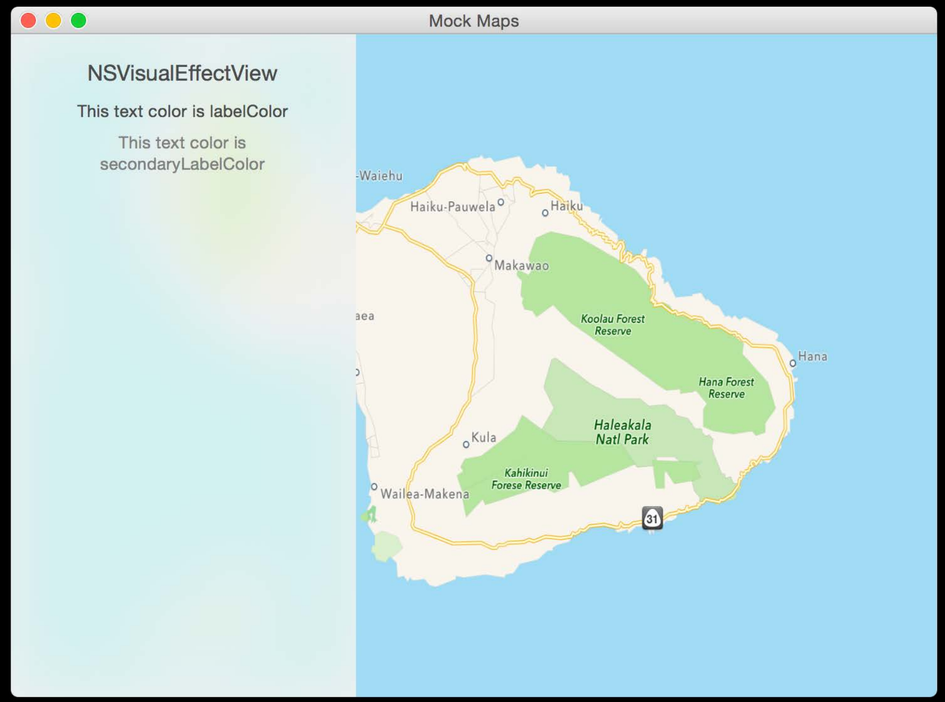
Any area behind the NSVisualEffectView in the window will be “knocked out”



New Visual Effects

How “within window” blending works

- Implemented using Core Animation
- NSVisualEffectView uses special layer filters to perform blurring
- Core Animation layer filters implement Vibrancy



New Visual Effects

How “within window” blending works

`NSVisualEffectBlendingMode`**`WithinWindow`**

Requires layer-backing on the container view

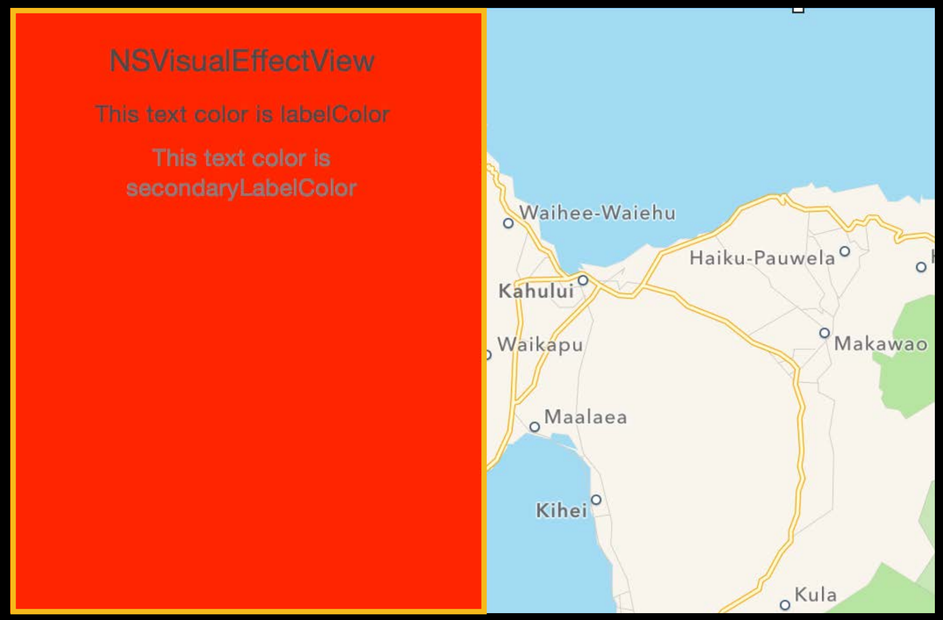
New Visual Effects

How “within window” blending works

`NSVisualEffectBlendingMode`**`WithinWindow`**

Requires layer-backing on the container view

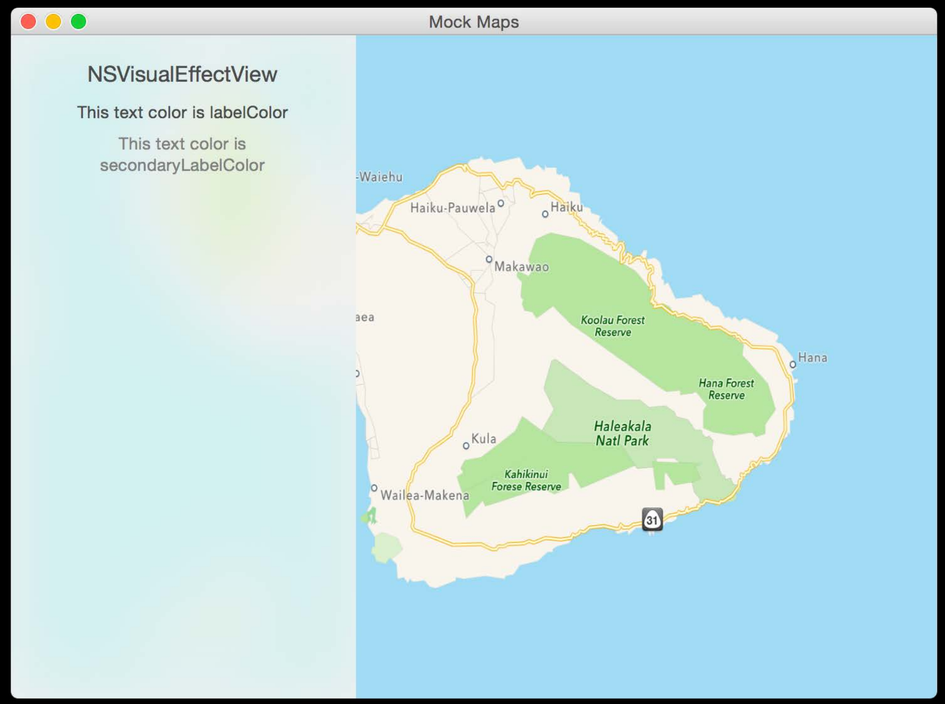
This view is red in IB to indicate it needs to have a layer →



New Visual Effects

How “within window” blending works

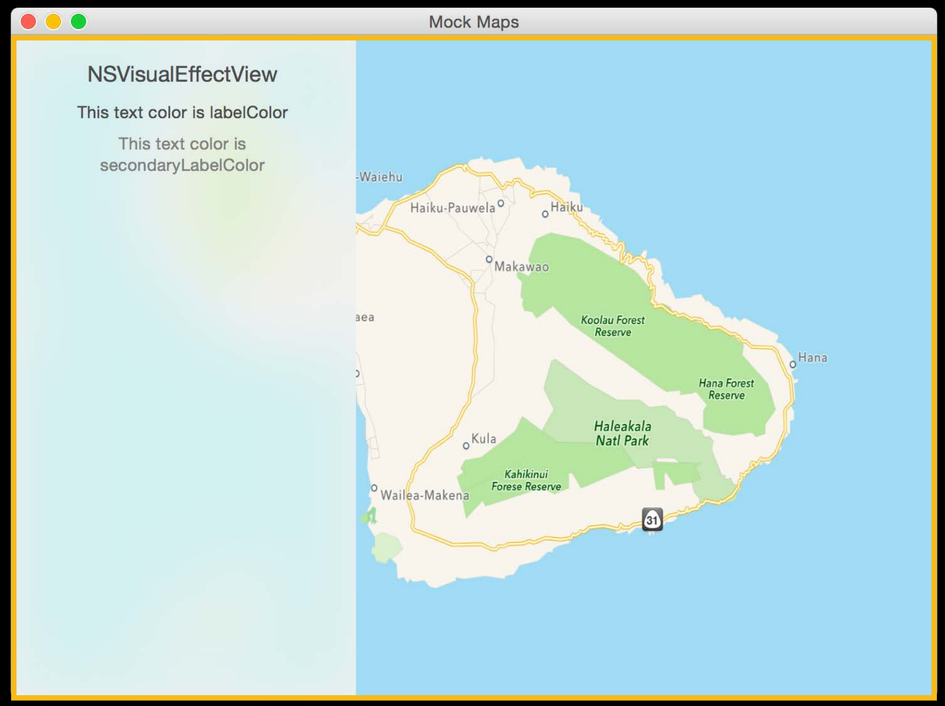
```
NSVisualEffectBlendingModeWithinWindow
containingView.wantsLayer = YES;
```



New Visual Effects

How “within window” blending works

```
NSVisualEffectBlendingModeWithinWindow
containingView.wantsLayer = YES;
```

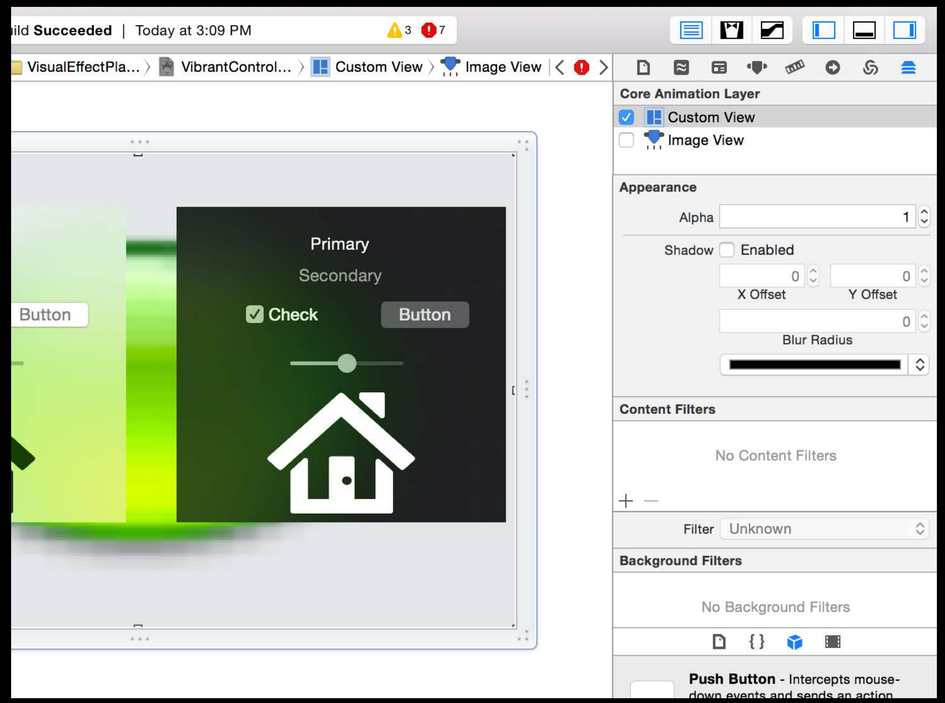


New Visual Effects

How “within window” blending works

`NSVisualEffectBlendingMode`**`WithinWindow`**

Or set it in Xcode/IB:

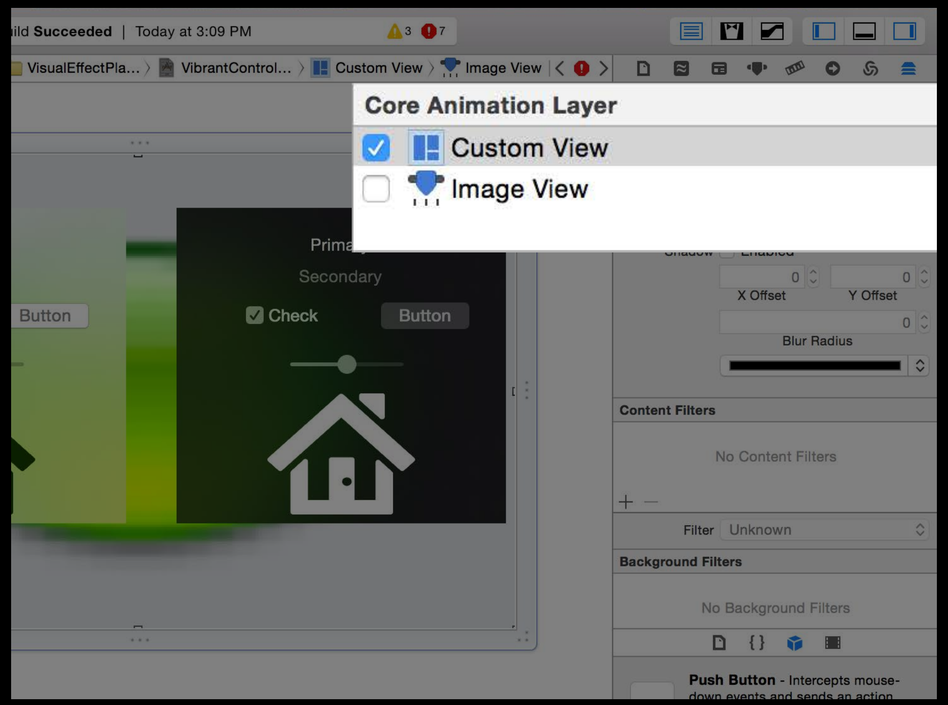


New Visual Effects

How “within window” blending works

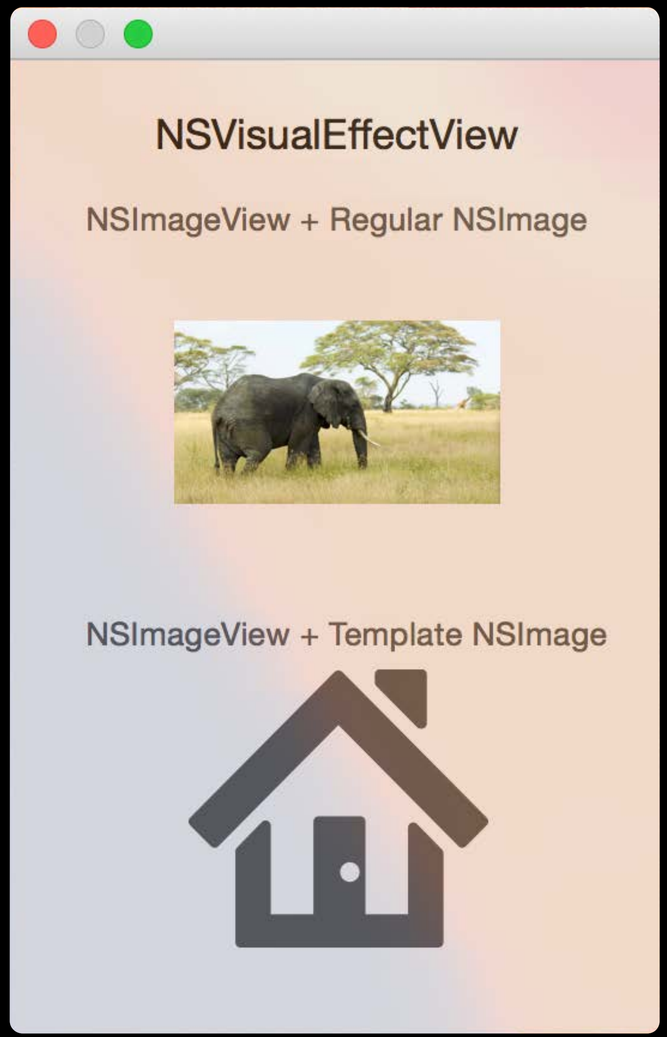
`NSVisualEffectBlendingMode`**`WithinWindow`**

Or set it in Xcode/IB:



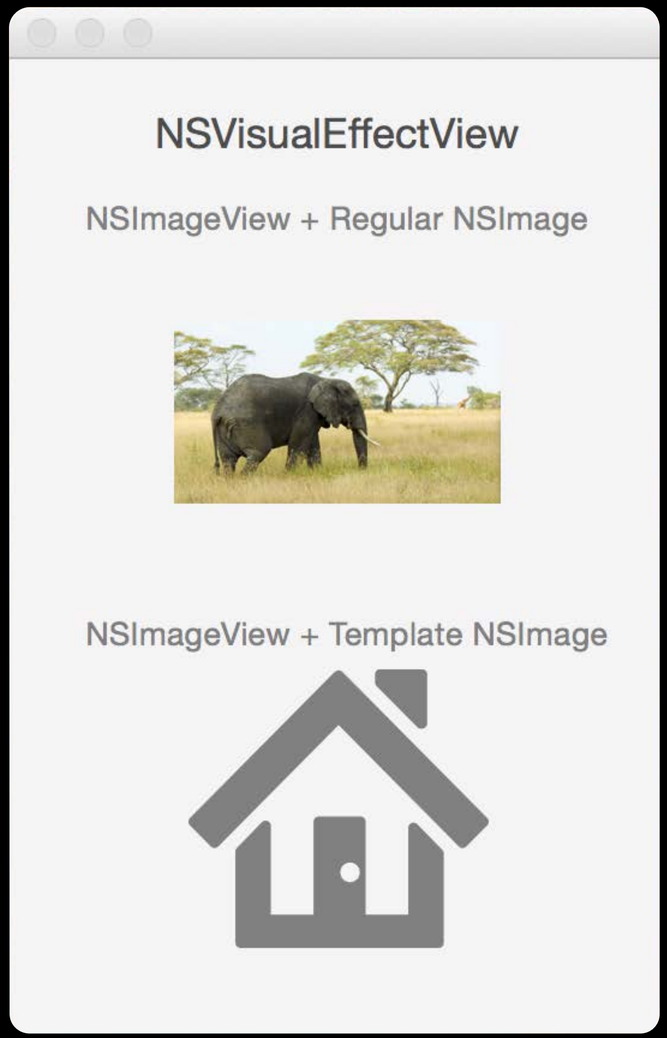
New Visual Effects

Key window/active state



New Visual Effects

Non-key window/inactive state



New Visual Effects

State

```
typedef NS_ENUM(NSInteger, NSVisualEffectState) {
    NSVisualEffectStateFollowsWindowActiveState,
    NSVisualEffectStateActive,
    NSVisualEffectStateInactive,
};

@property NSVisualEffectState state;
```

New Visual Effects

State

```
typedef NS_ENUM(NSInteger, NSVisualEffectState) {
    NSVisualEffectStateFollowsWindowActiveState,
    NSVisualEffectStateActive,
    NSVisualEffectStateInactive,
};

@property NSVisualEffectState state;
```

New Visual Effects

State

```
typedef NS_ENUM(NSInteger, NSVisualEffectState) {
    NSVisualEffectStateFollowsWindowActiveState,
    NSVisualEffectStateActive,
    NSVisualEffectStateInactive,
};

@property NSVisualEffectState state;
```

New Visual Effects

State

```
typedef NS_ENUM(NSInteger, NSVisualEffectState) {
    NSVisualEffectStateFollowsWindowActiveState,
    NSVisualEffectStateActive,
    NSVisualEffectStateInactive,
};

@property NSVisualEffectState state;
```

New Visual Effects

State

Use `NSVisualEffectStateFollowsWindowActiveState` the majority of the time

Use `NSVisualEffectStateActive` sparingly

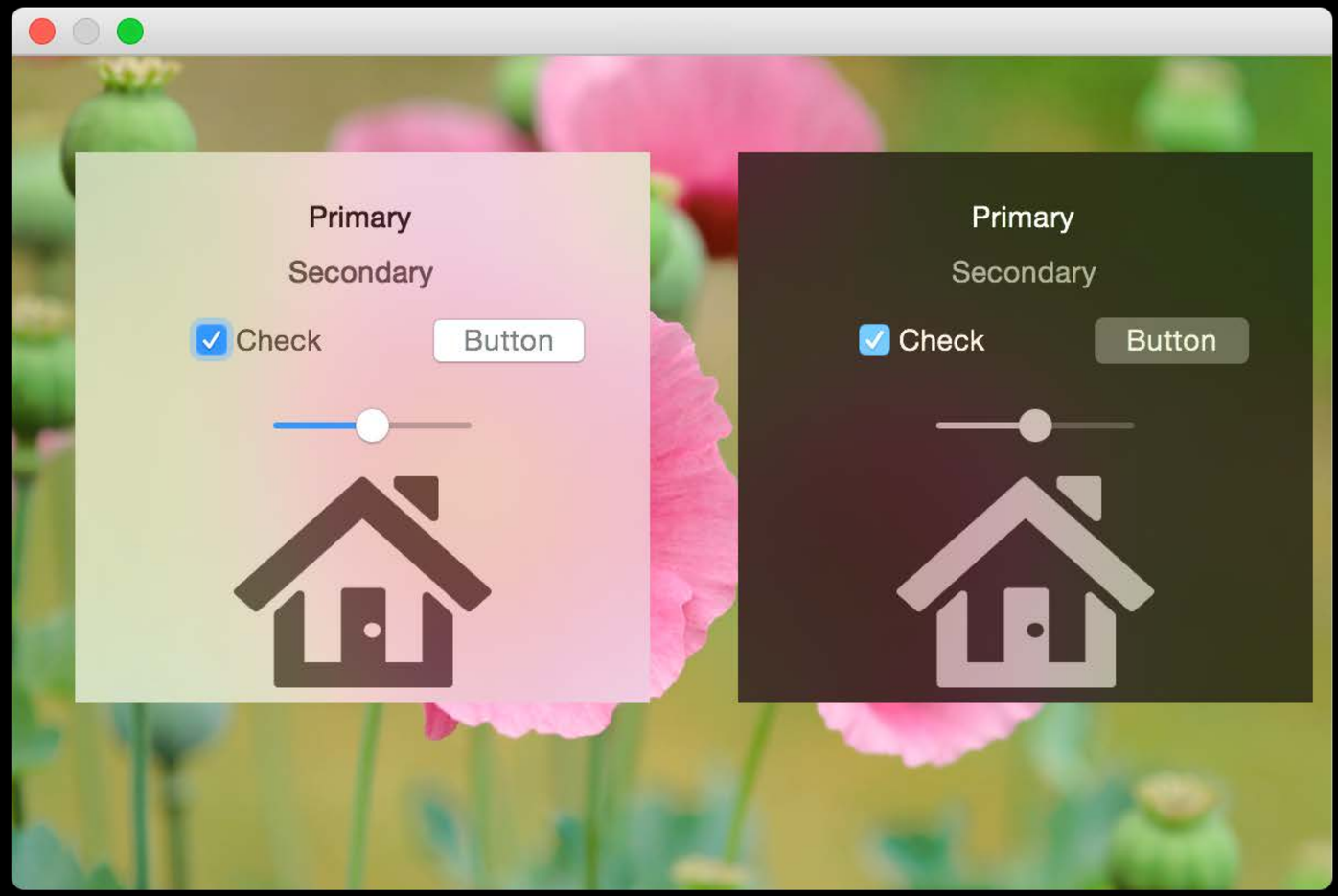
- It can affect performance and battery life
- Use in windows that never become key
- Use in windows that should always have an active look
  - Popovers and sheets

NSAppearance and Materials

Using NSVisualEffectView

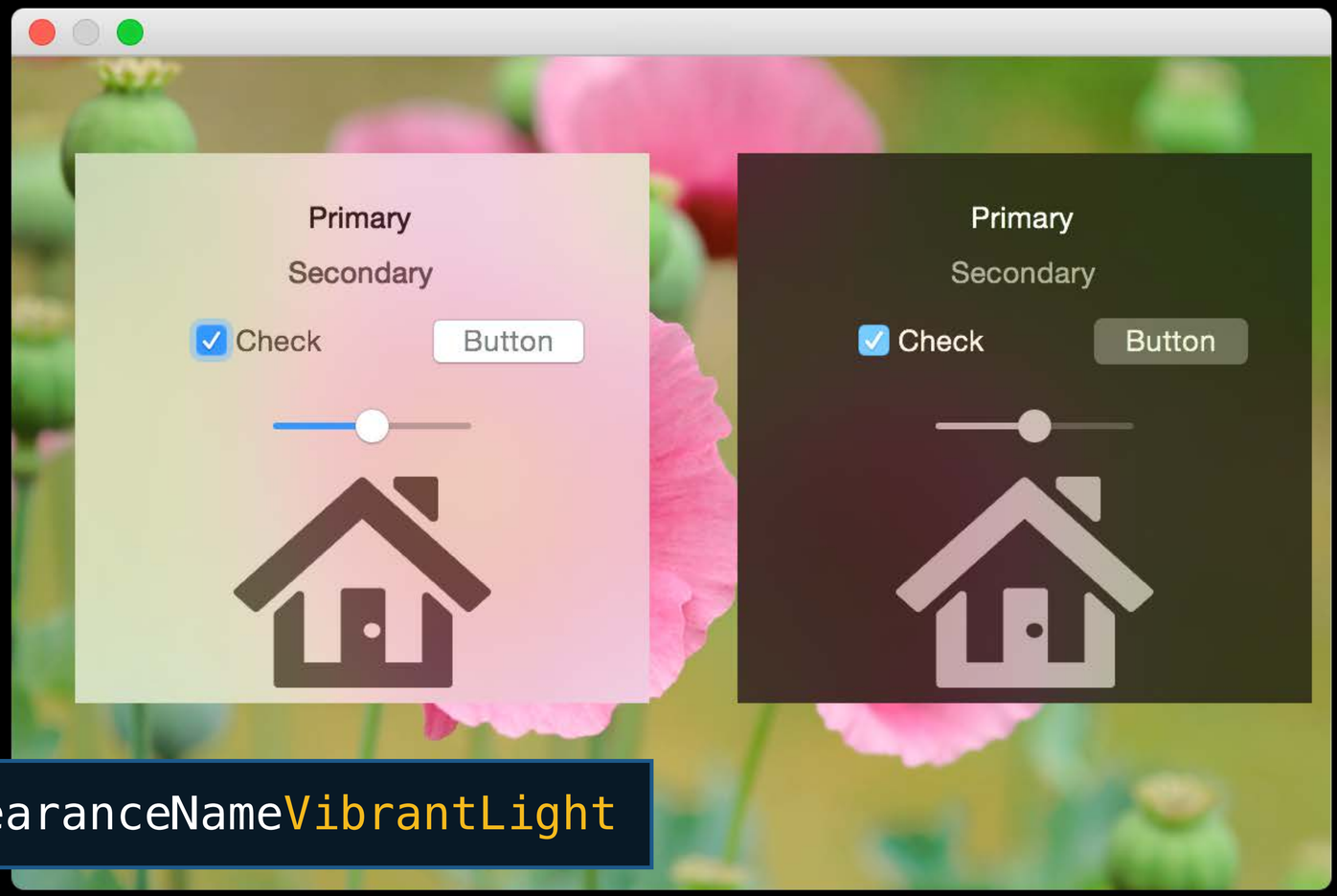
New Visual Effects

NSAppearance and materials



New Visual Effects

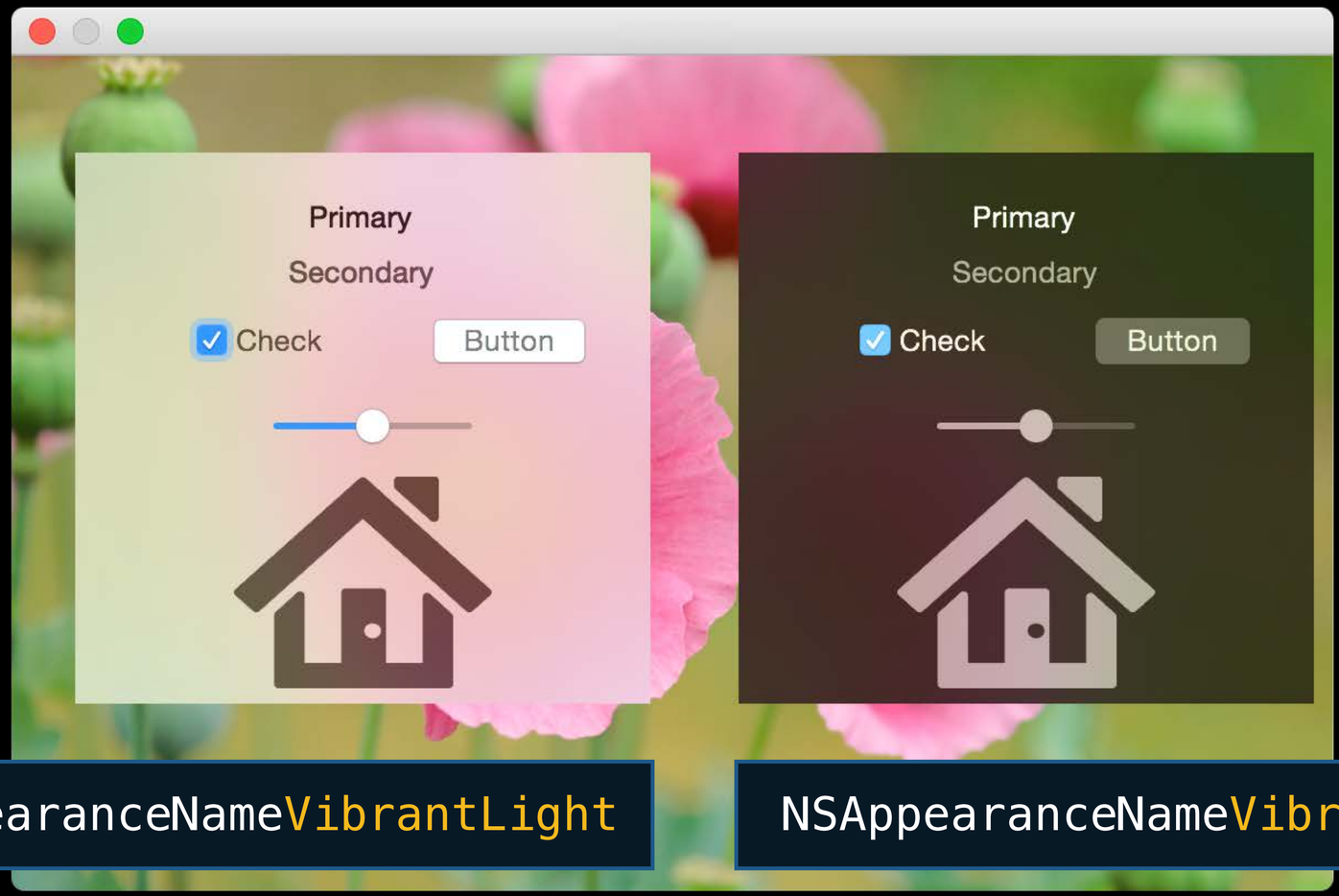
NSAppearance and materials




```
NSAppearanceNameVibrantLight
```

New Visual Effects

NSAppearance and materials



`NSAppearanceNameVibrantLight`

`NSAppearanceNameVibrantDark`

New Visual Effects

NSAppearance and materials

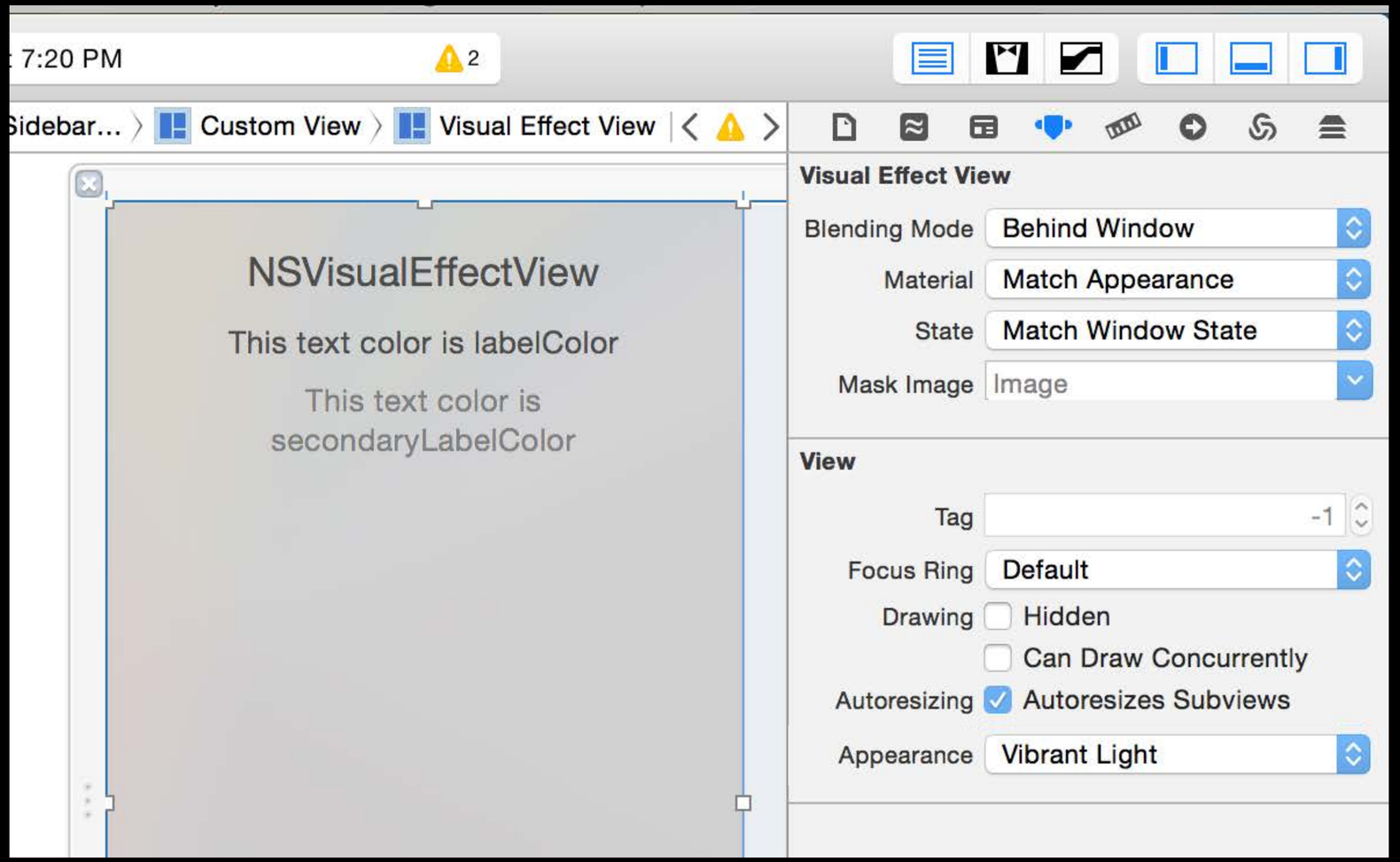
```
visualEffectView.appearance =
                [NSAppearance appearanceNamed:NSAppearanceNameVibrantLight];
```

or

```
visualEffectView.appearance =
                [NSAppearance appearanceNamed:NSAppearanceNameVibrantDark];
```

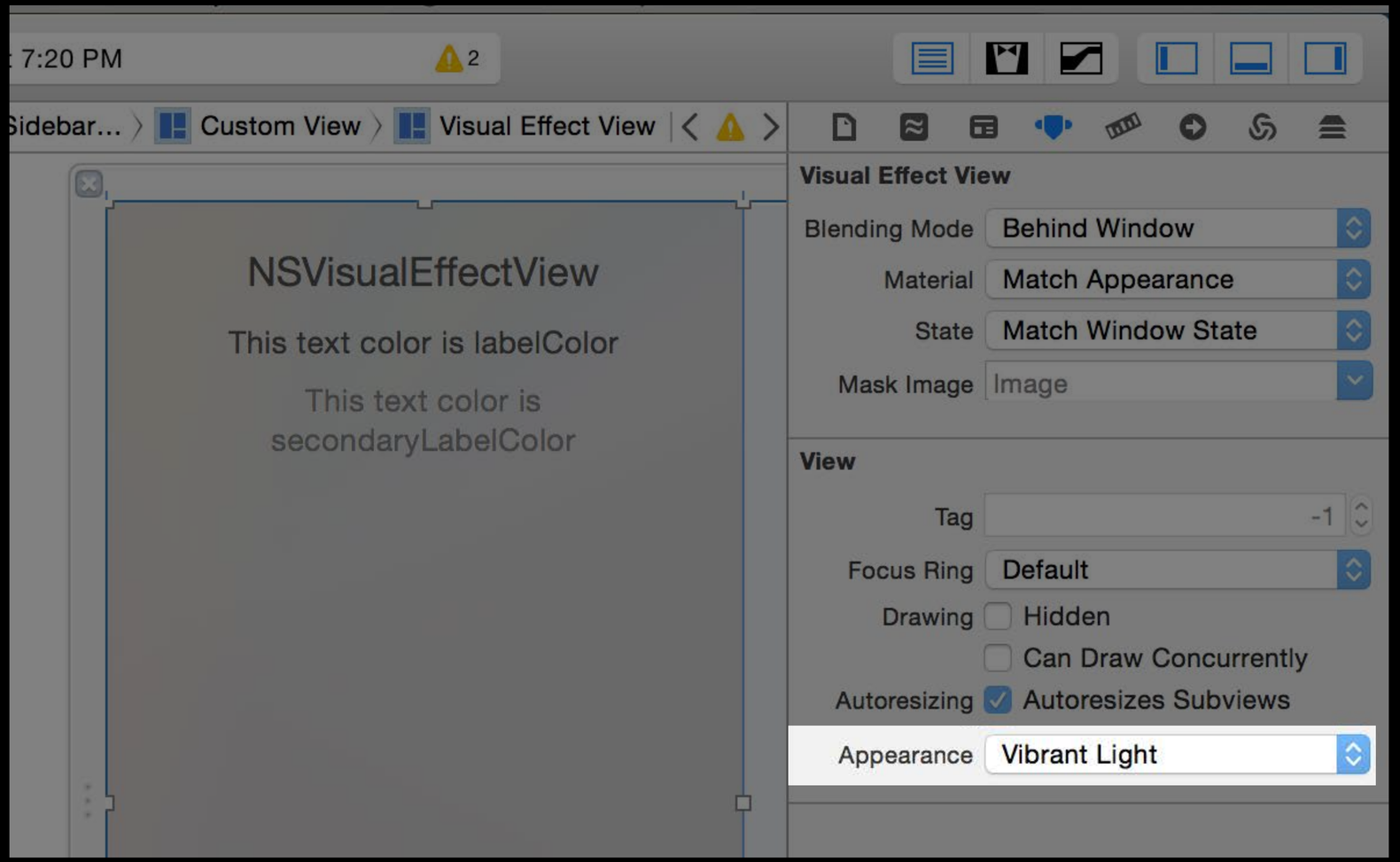
New Visual Effects

Set the Appearance in IB



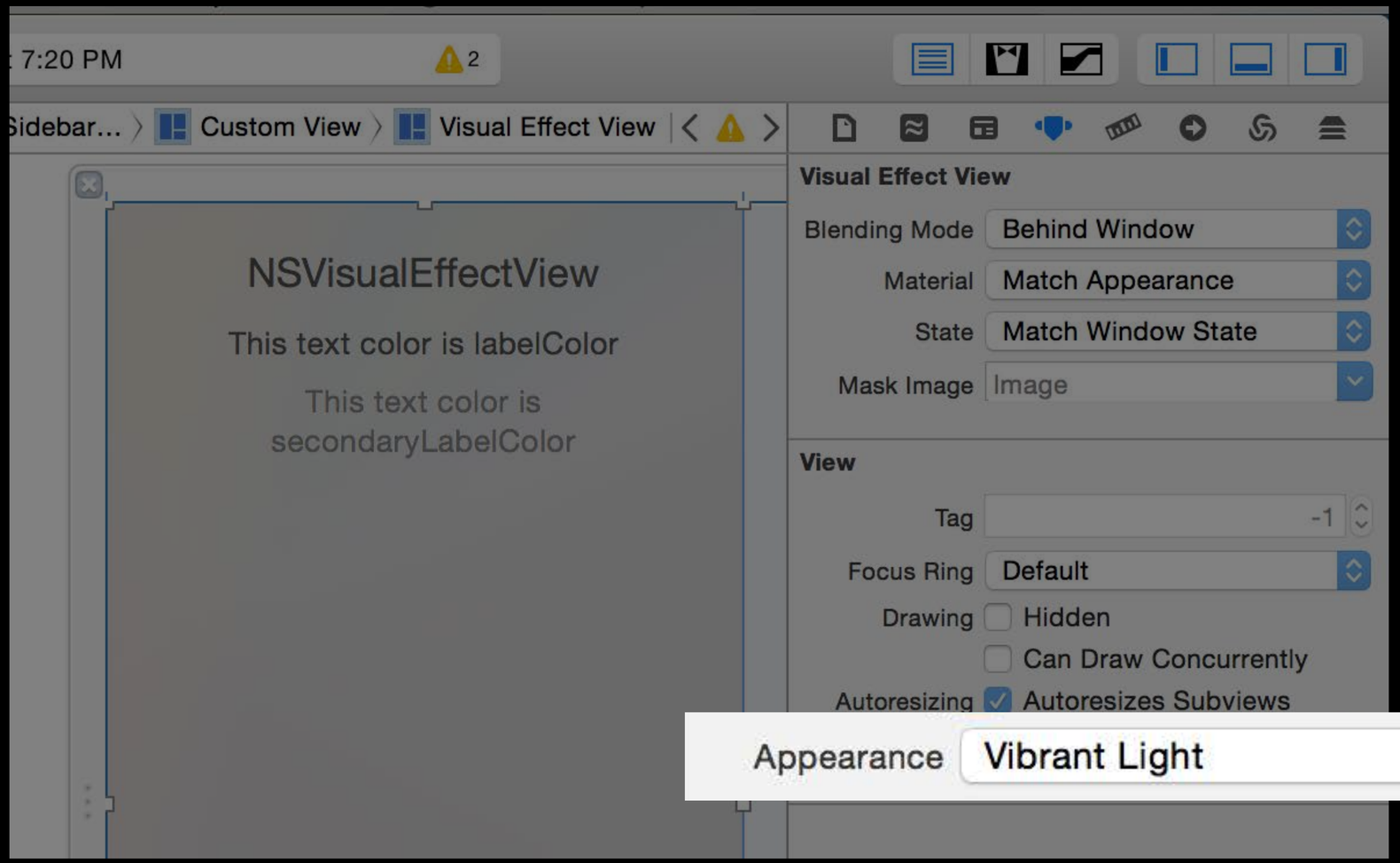
New Visual Effects

Set the Appearance in IB



New Visual Effects

Set the Appearance in IB



New Visual Effects

NSAppearance and materials

The vibrant appearances require an NSVisualEffectView

The NSVisualEffectView must be a direct ancestor of the view with the vibrant appearance applied

The material shown is not necessarily controlled by the appearance

New Visual Effects

NSAppearance and materials

```
@interface NSVisualEffectView ...
@property NSVisualEffectMaterial material;
@end
```

New Visual Effects

NSAppearance and materials

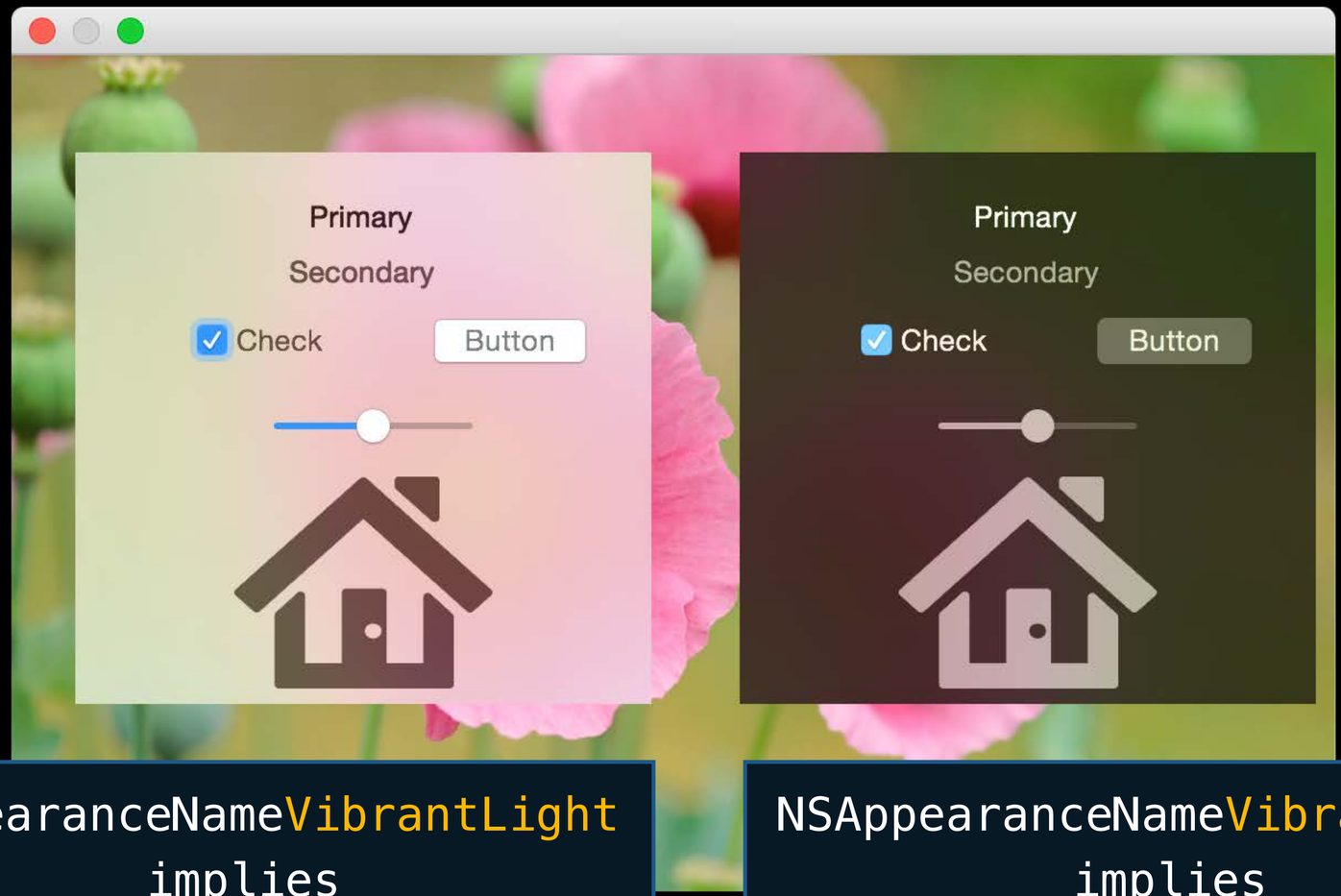
```
@interface NSVisualEffectView ...
@property NSVisualEffectMaterial material;
@end

typedef NS_ENUM(NSInteger, NSVisualEffectMaterial) {
    NSVisualEffectMaterialAppearanceBased,
     NSVisualEffectMaterialLight,
     NSVisualEffectMaterialDark,
     NSVisualEffectMaterialTitlebar
};
```

New Visual Effects

NSVisualEffectMaterialAppearanceBased

```
@interface NSVisualEffectView ...
@property NSVisualEffectMaterial material;
@end

typedef NS_ENUM(NSInteger, NSVisualEffectMaterial) {
    NSVisualEffectMaterialAppearanceBased,
     NSVisualEffectMaterialLight,
     NSVisualEffectMaterialDark,
     NSVisualEffectMaterialTitlebar
};
```

NSVisualEffectMaterialAppearanceBased



NSAppearanceNameVibrantLight
implies
NSVisualEffectMaterialLight

NSAppearanceNameVibrantDark
implies
NSVisualEffectMaterialDark

New Visual Effects

Titlebar material

```
@interface NSVisualEffectView …
@property NSVisualEffectMaterial material;
@end

typedef NS_ENUM(NSInteger, NSVisualEffectMaterial) {
    NSVisualEffectMaterialAppearanceBased,
     NSVisualEffectMaterialLight,
     NSVisualEffectMaterialDark,
     NSVisualEffectMaterialTitlebar
};
```

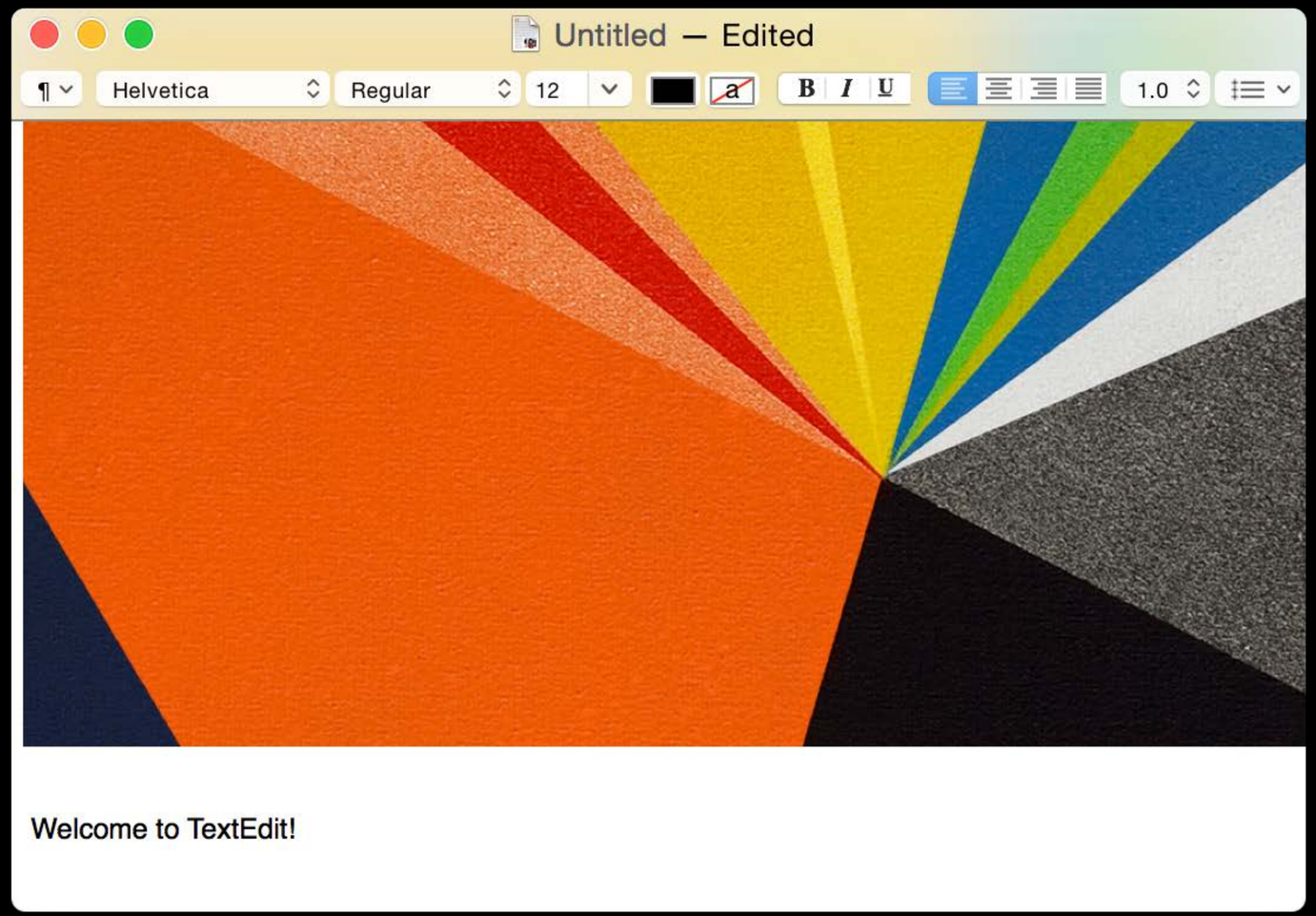
New Visual Effects

Titlebar material

```
@interface NSVisualEffectView …
@property NSVisualEffectMaterial material;
@end

typedef NS_ENUM(NSInteger, NSVisualEffectMaterial) {
    NSVisualEffectMaterialAppearanceBased,
     NSVisualEffectMaterialLight,
     NSVisualEffectMaterialDark,
     NSVisualEffectMaterialTitlebar
 };
```

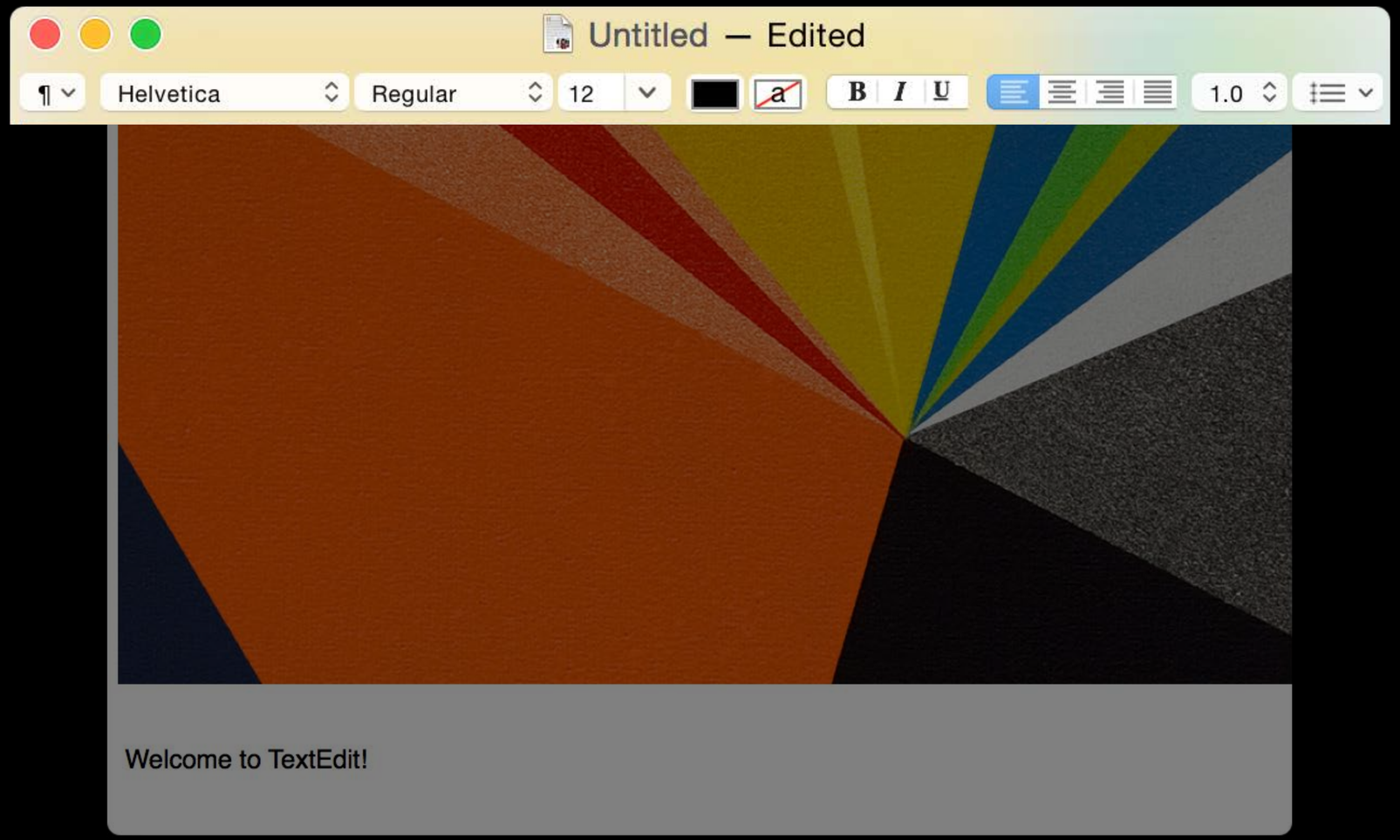
New Visual Effects

NSVisualEffectMaterialTitlebar



New Visual Effects

NSVisualEffectMaterialTitlebar



Vibrancy and Custom Controls

What is Vibrancy?

Vibrancy is a special blend mode for how a source and destination pixel are combined together

Vibrancy is provided by AppKit as an abstraction

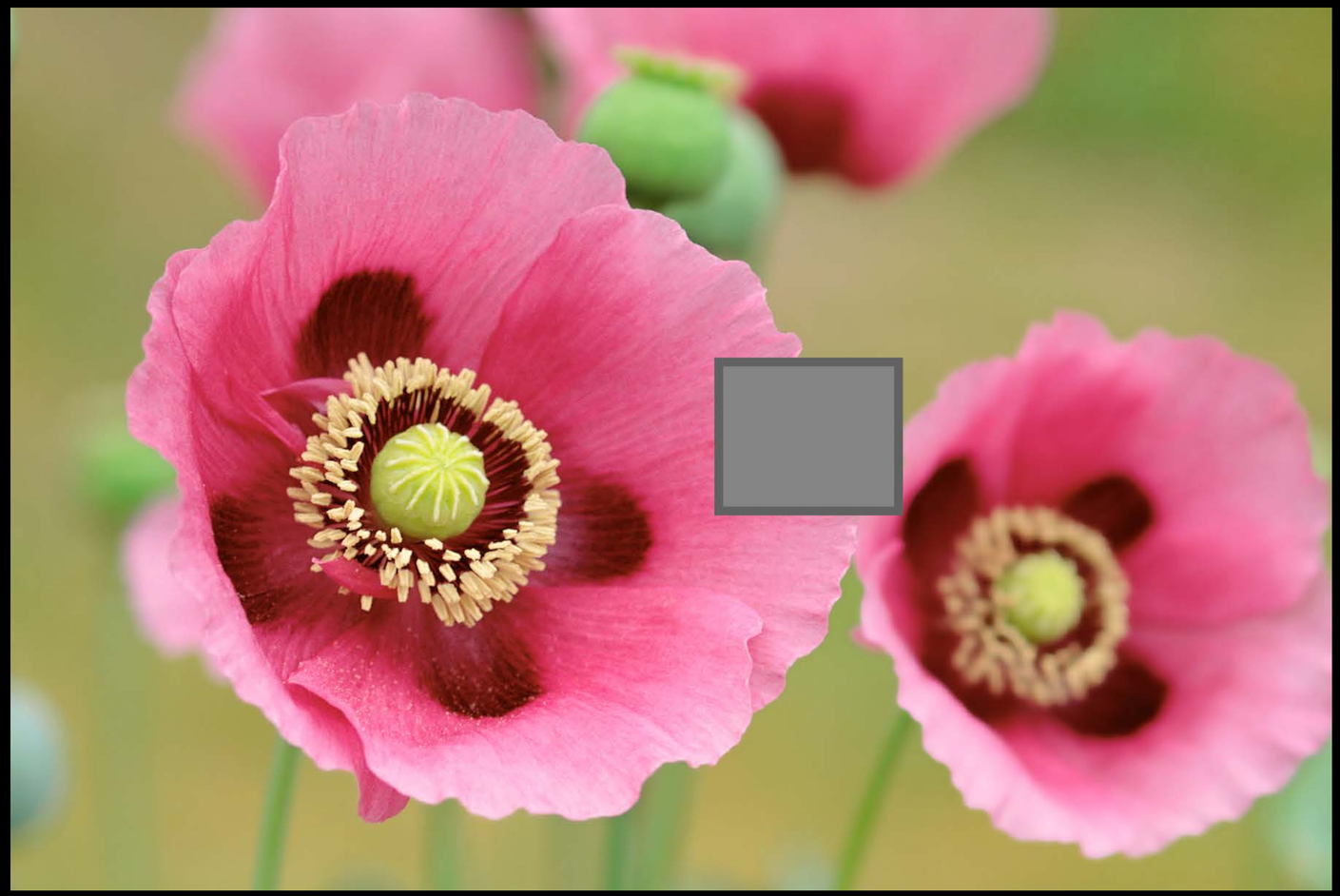
The actual blend mode may be Linear Burn, Color Dodge, PlusD, PlusL, etc.

What is Vibrancy?

Example—color burn

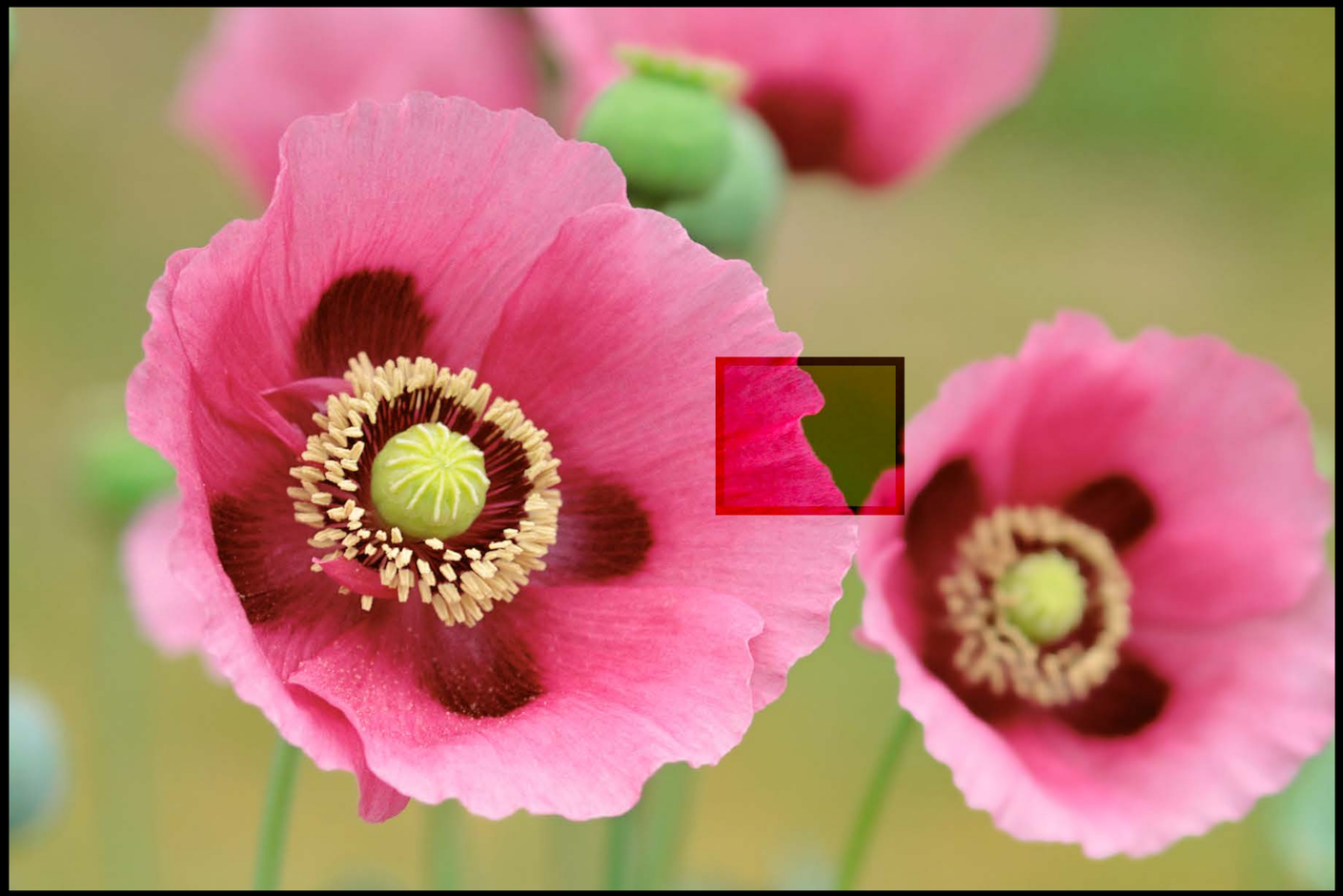
What is Vibrancy?

Example—color burn



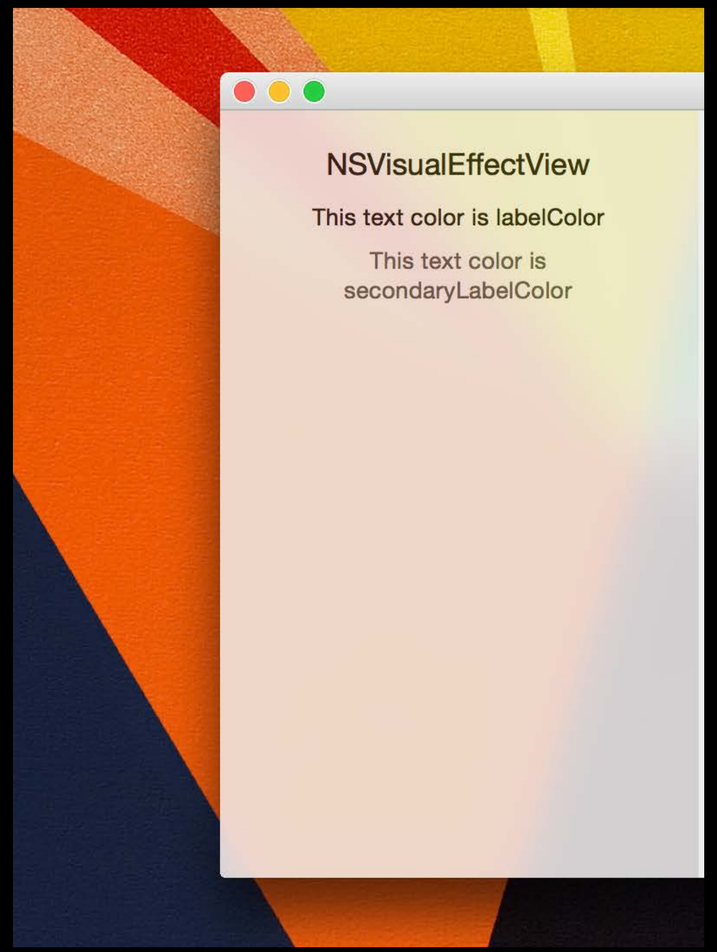
What is Vibrancy?

Example—color burn



New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text

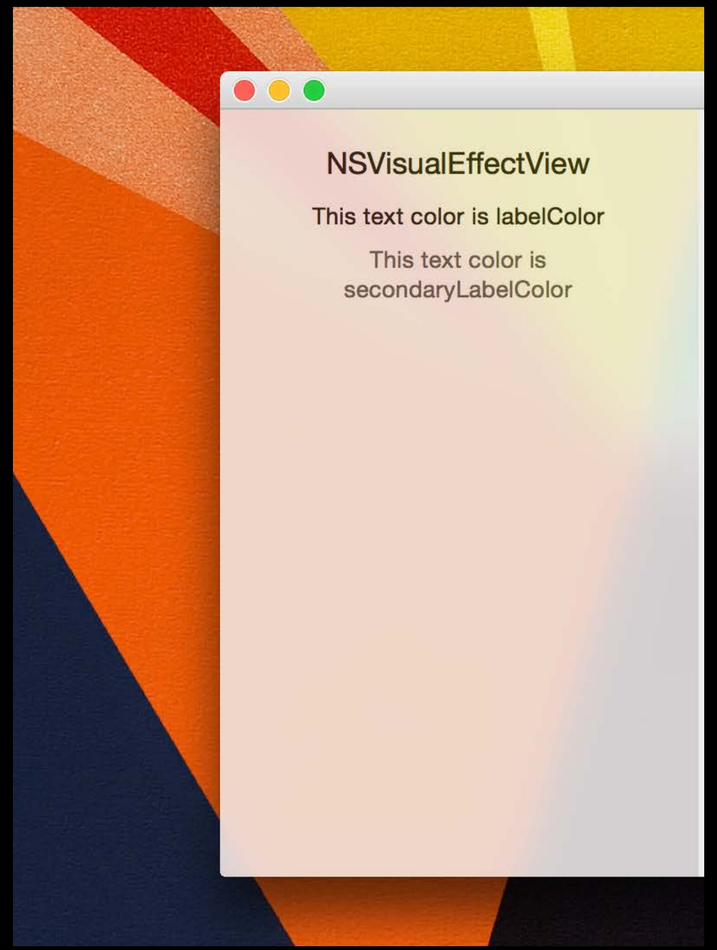


New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text

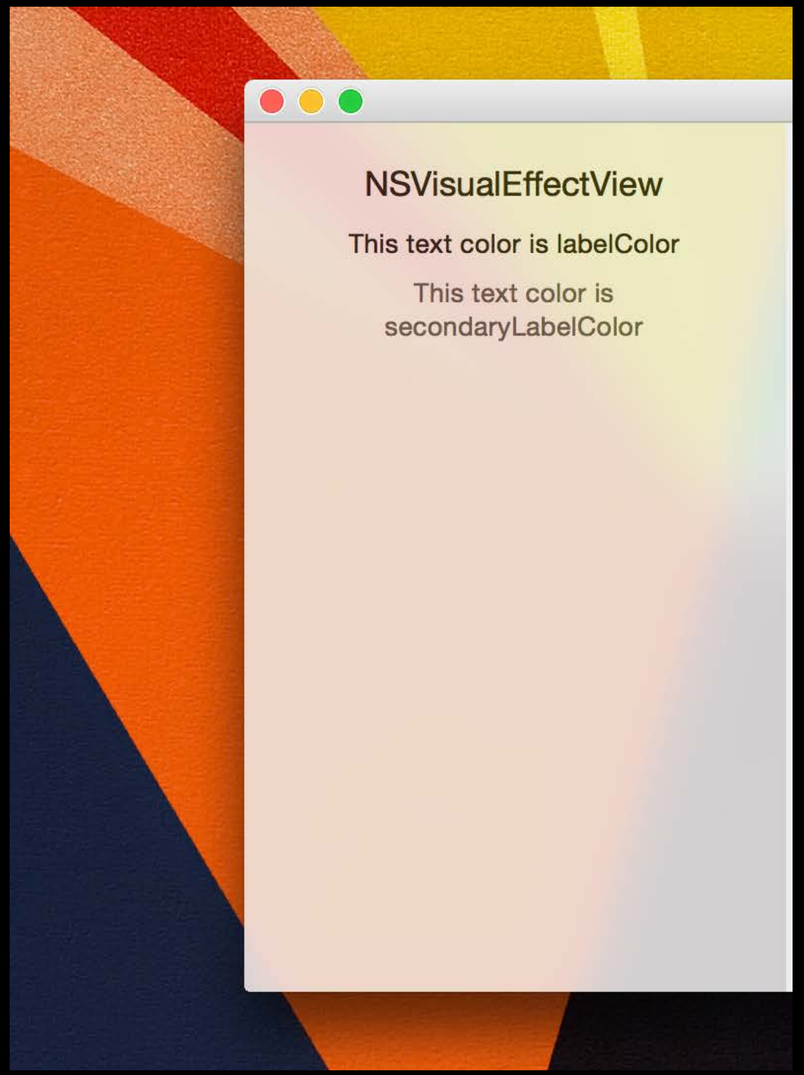
NSAppearanceNamedVibrantLight or NSAppearanceNamedVibrantDark must be used

Vibrancy is abstractly applied



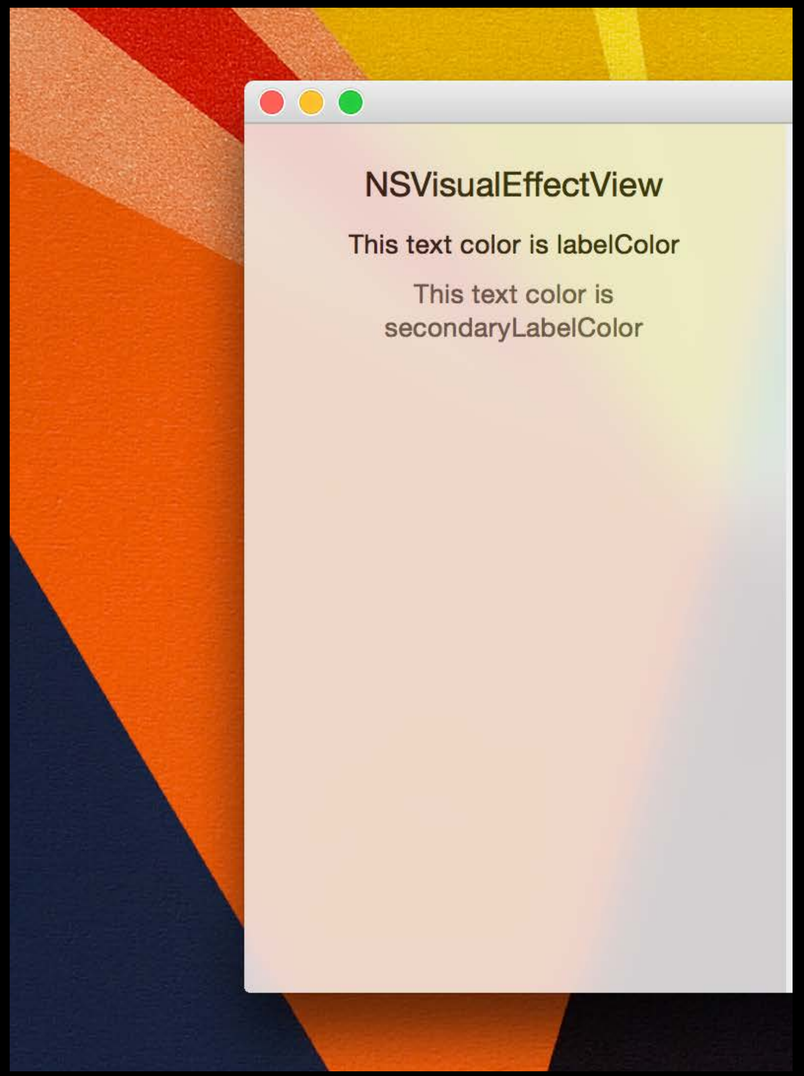
New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text



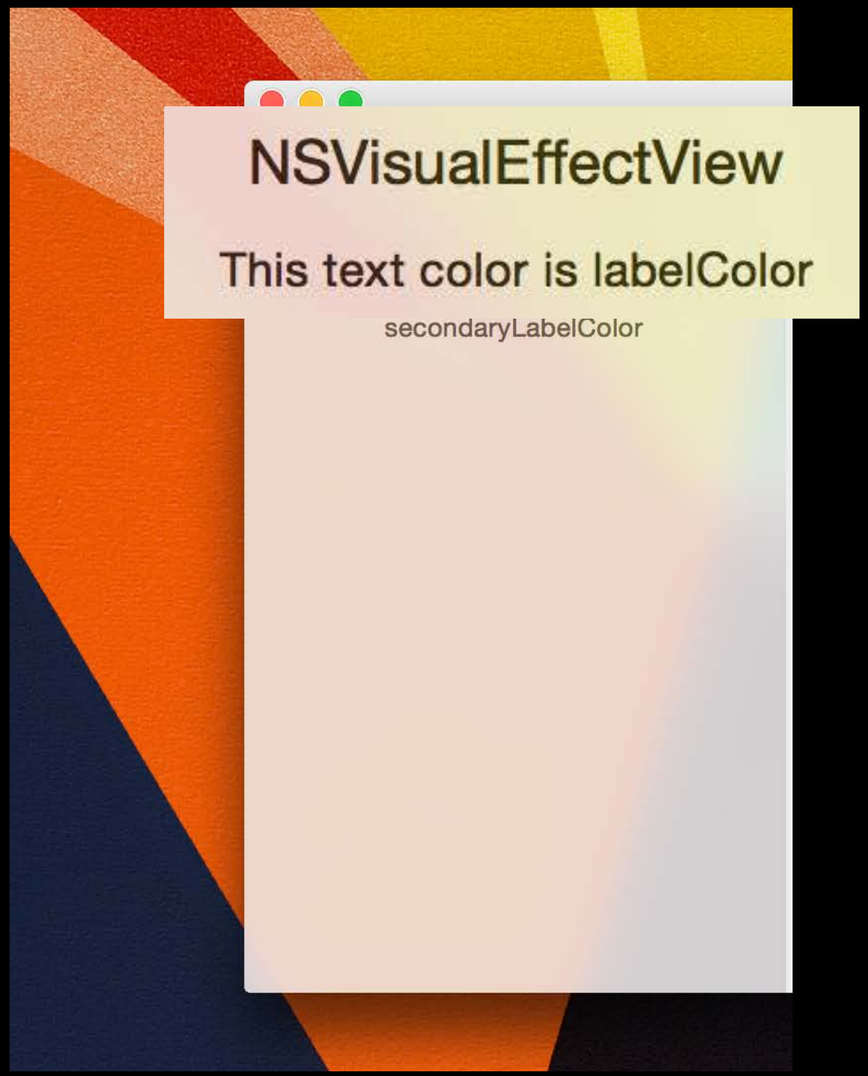
New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text



New Visual Effects and Vibrancy

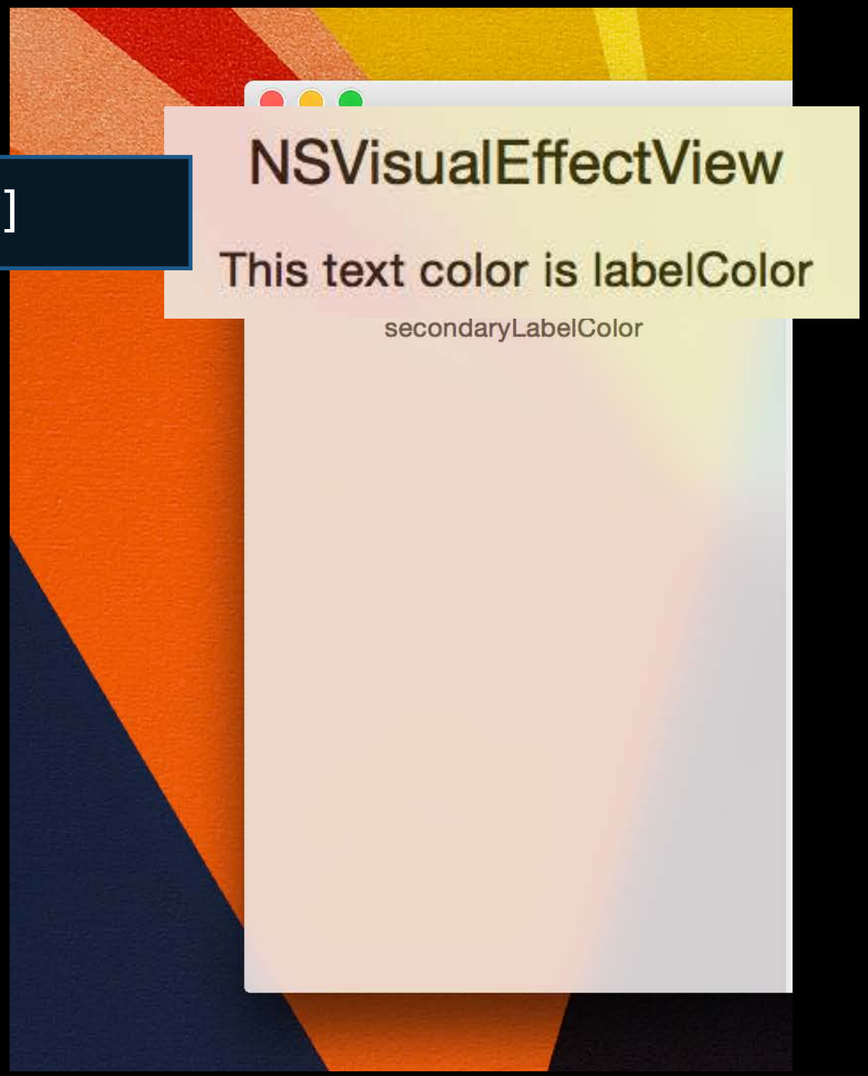
NSVisualEffectView and vibrant text



New Visual Effects and Vibrancy

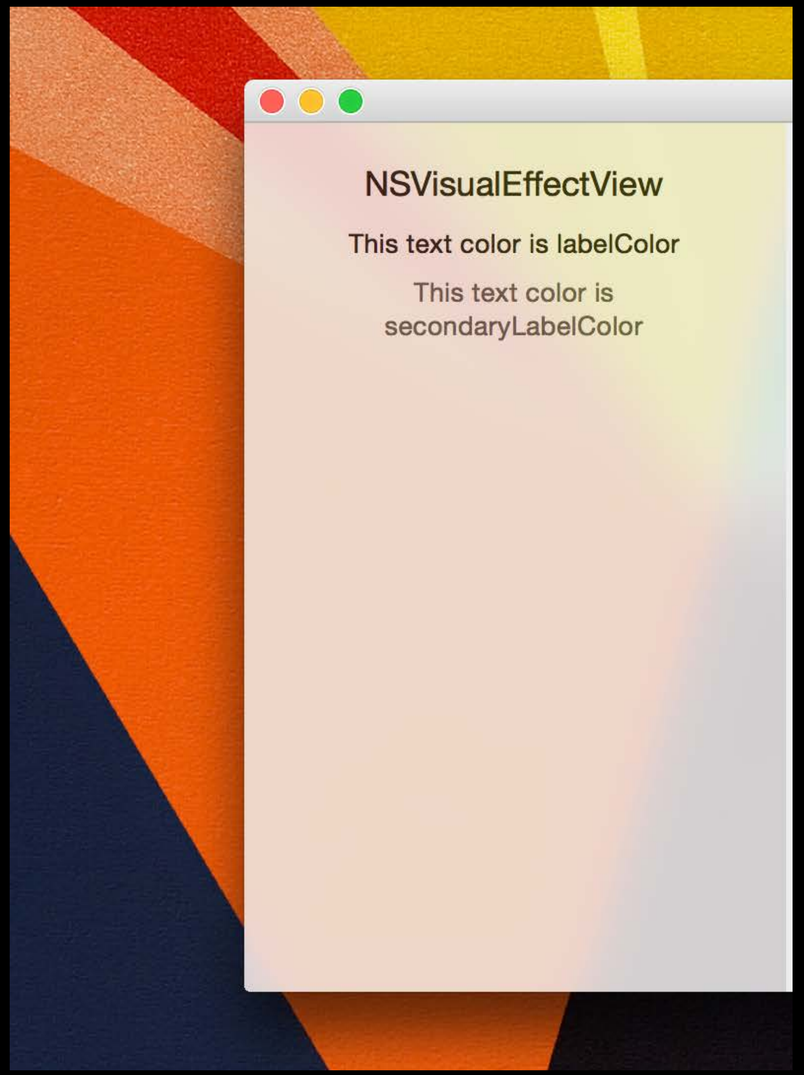
NSVisualEffectView and vibrant text

```
[NSColor labelColor]
```



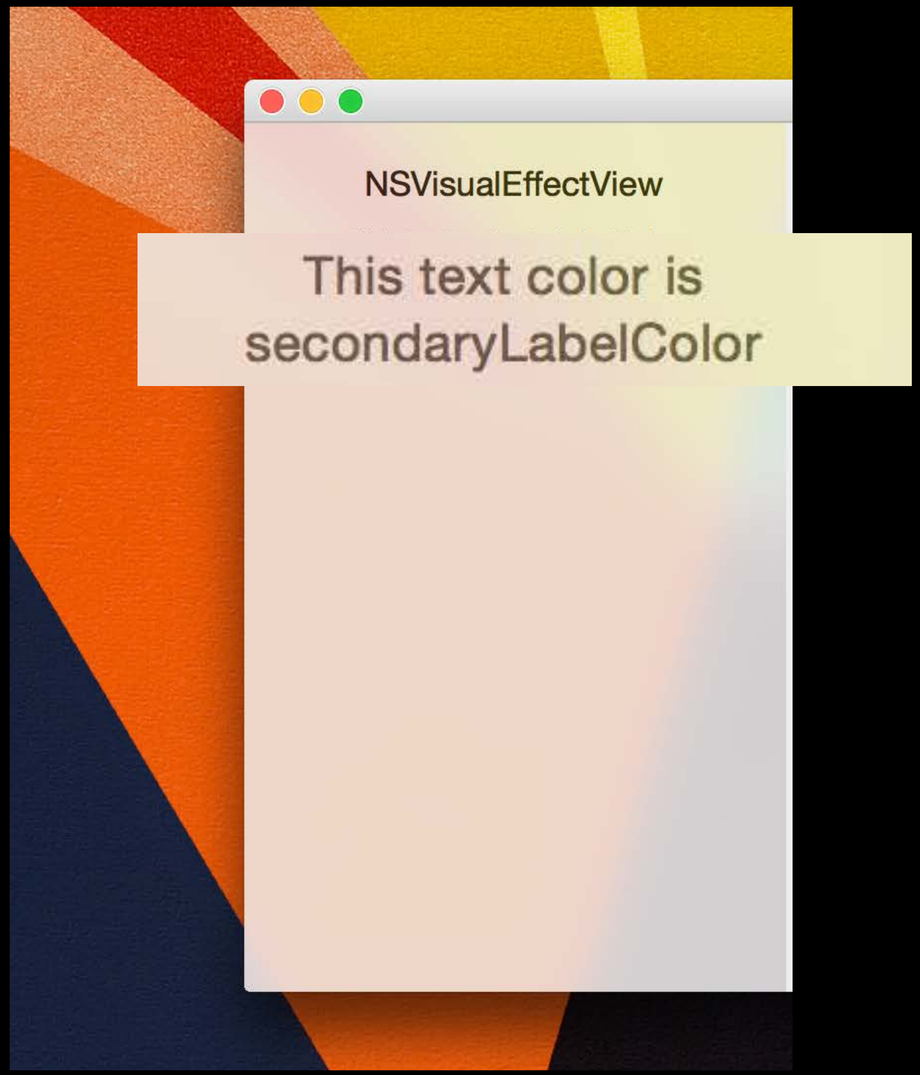
New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text



New Visual Effects and Vibrancy

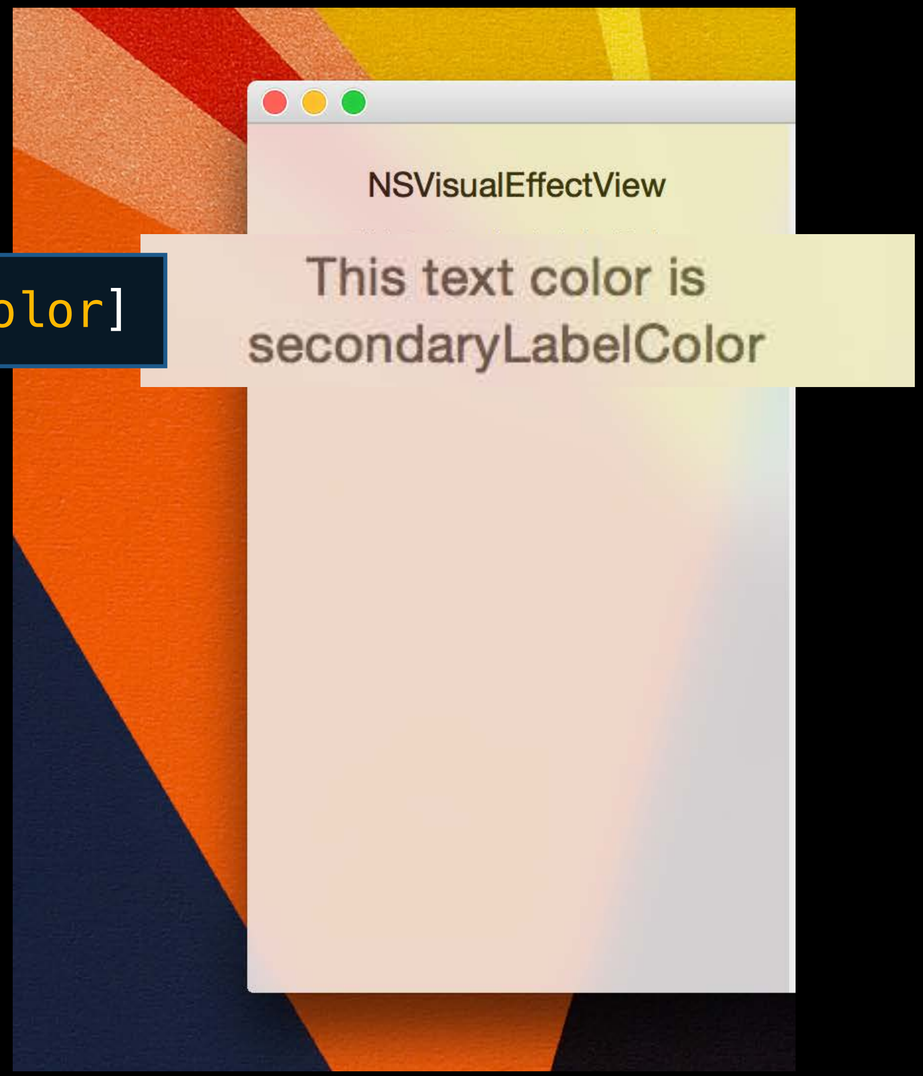
NSVisualEffectView and vibrant text



New Visual Effects and Vibrancy

NSVisualEffectView and vibrant text

```
[NSColor secondaryLabelColor]
```



New Visual Effects and Vibrancy

Reminder about template images

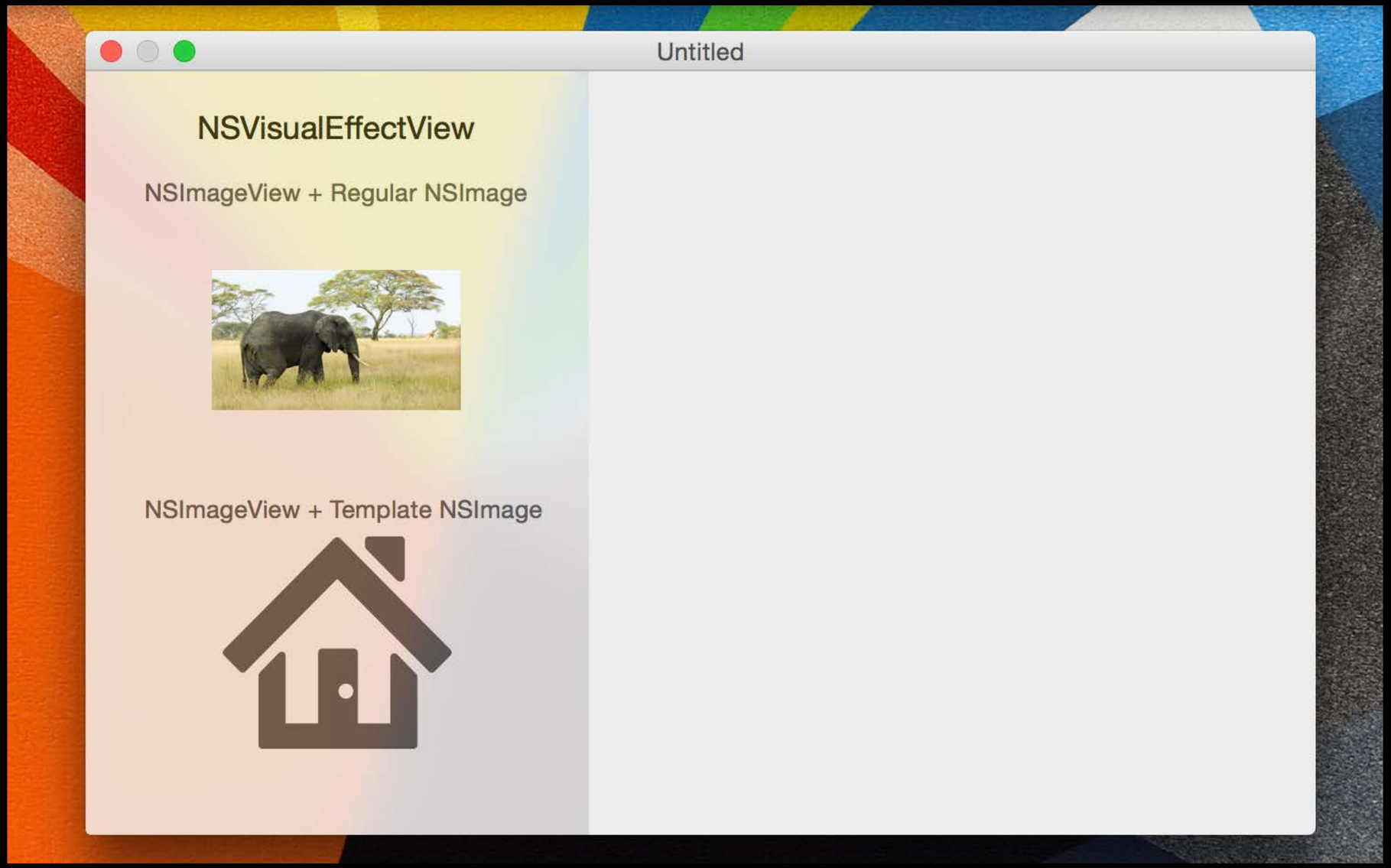
Template images are black and white

NSImage has API to set any image to be marked as a template image

```
@interface NSImage ...
- (BOOL)isTemplate NS_AVAILABLE_MAC(10_5);
- (void)setTemplate:(BOOL)isTemplate NS_AVAILABLE_MAC(10_5);
@end
```

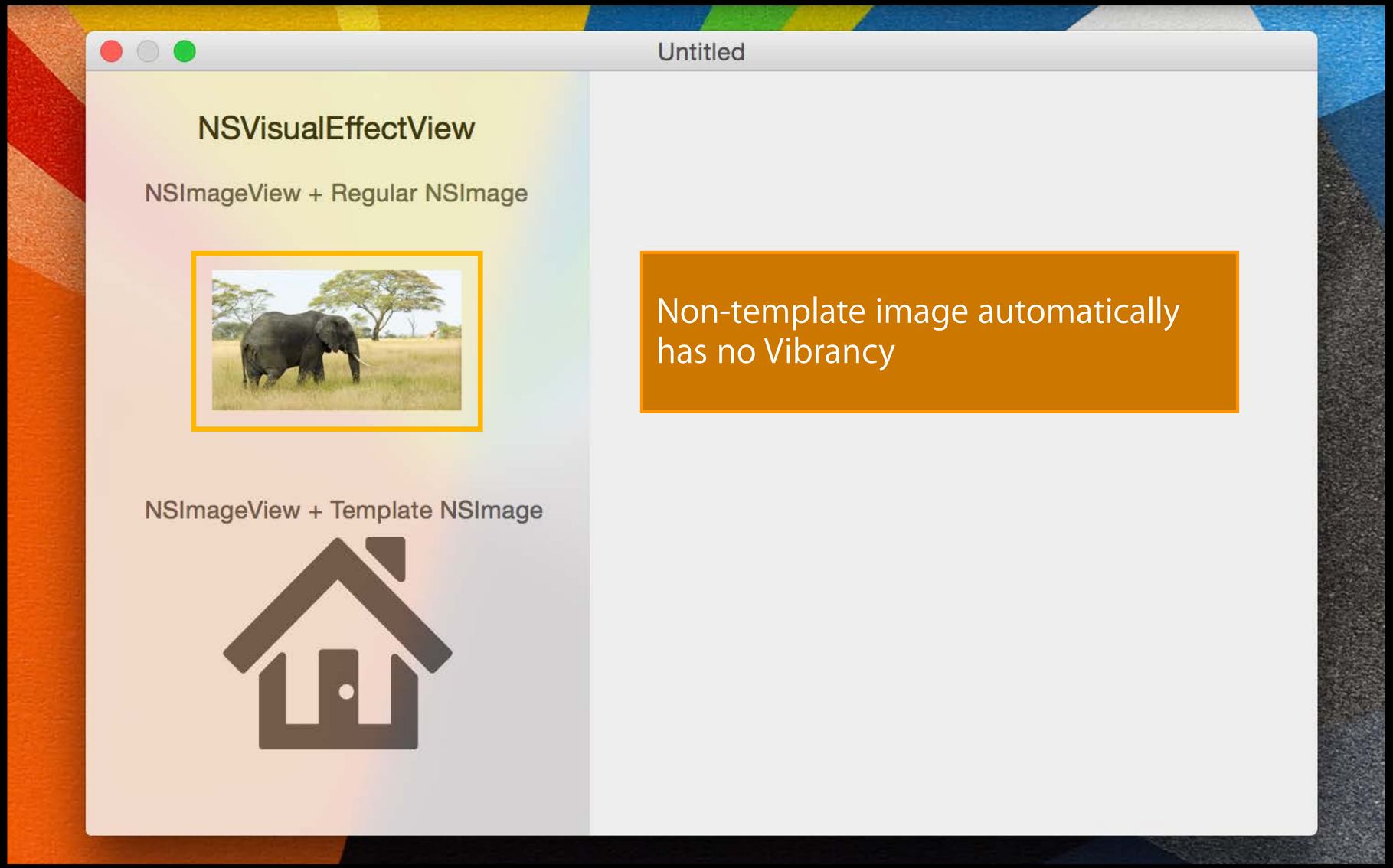
New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView



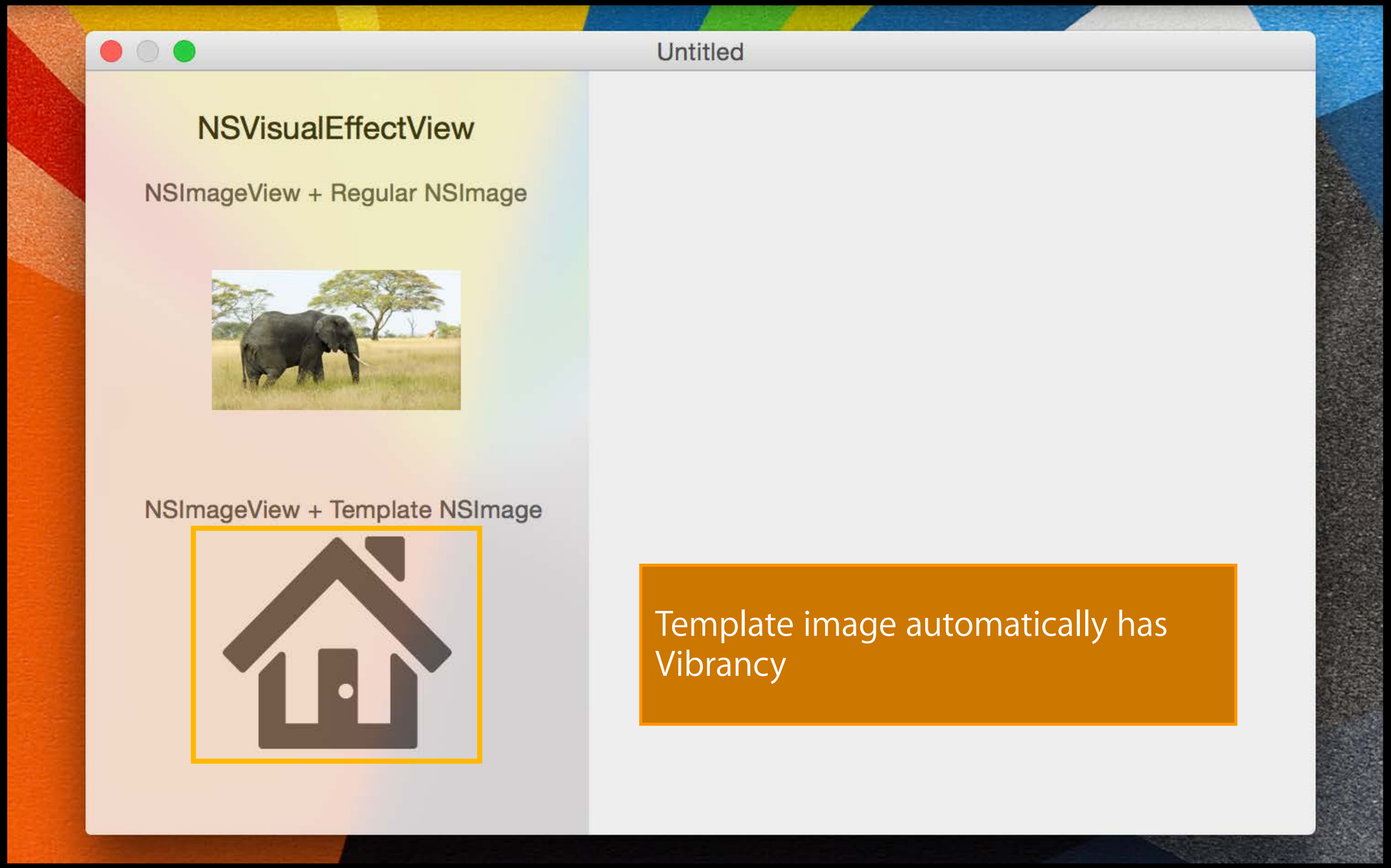
New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView



New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView



New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView

Question—How does NSImageView draw vibrant or not?

New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView

Question—How does NSImageView draw vibrant or not?

Answer—It overrides new NSView API:

```
- (BOOL)allowsVibrancy {
    return self.image.isTemplate;
}
```

New Visual Effects and Vibrancy

NSVisualEffectView and NSImageView

Question—How to invalidate the state of allowsVibrancy?

Answer—Call setNeedsDisplay:YES

```
[view setNeedsDisplay:YES];
// allowsVibrancy will be called again and vibrancy will be updated
```

New Visual Effects and Vibrancy

NSTextField and Vibrancy

NSTextField also behaves like NSImageView

It says YES to allowsVibrancy only when specific system colors are use

```
[NSColor labelColor]
[NSColor secondaryLabelColor]
```

Utilizing [NSColor controlTextColor] will make it not be vibrant

New Visual Effects and Vibrancy

NSAppearance and Vibrancy

NSAppearance also exposes the same API to know when a given appearance is vibrant or not

```
@interface NSAppearance …
@property (readonly) BOOL allowsVibrancy;
@end
```

New Visual Effects and Vibrancy

Views and Vibrancy

Any NSView will automatically become vibrant when all of the following are true

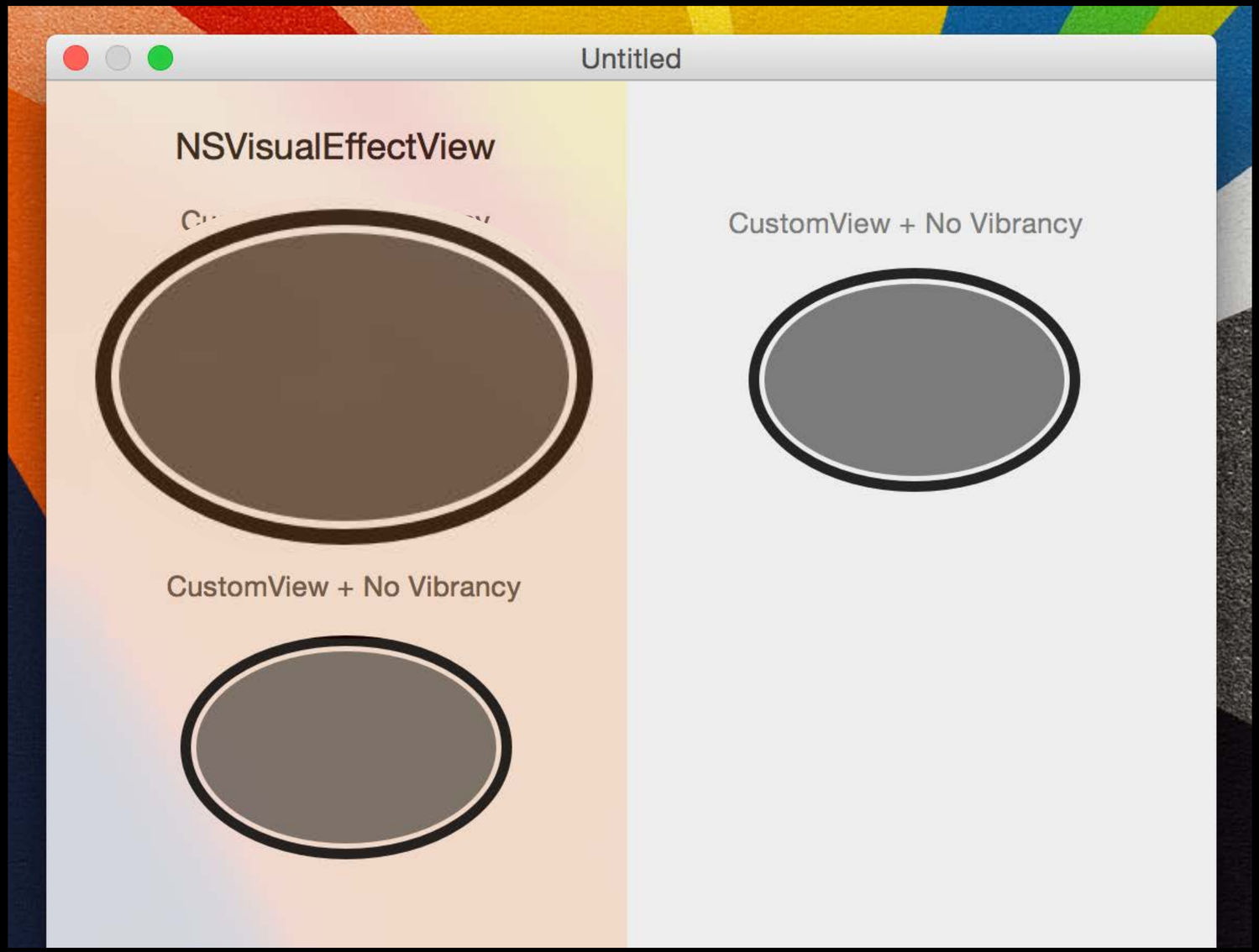
- The NSAppearance returns YES from allowsVibrancy
  - NSAppearanceNameVibrantLight or NSAppearanceNameVibrantDark
- The NSView subclass returns YES from allowsVibrancy
- The NSView is contained inside an NSVisualEffectView

New Visual Effects and Vibrancy

Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy



New Visual Effects and Vibrancy

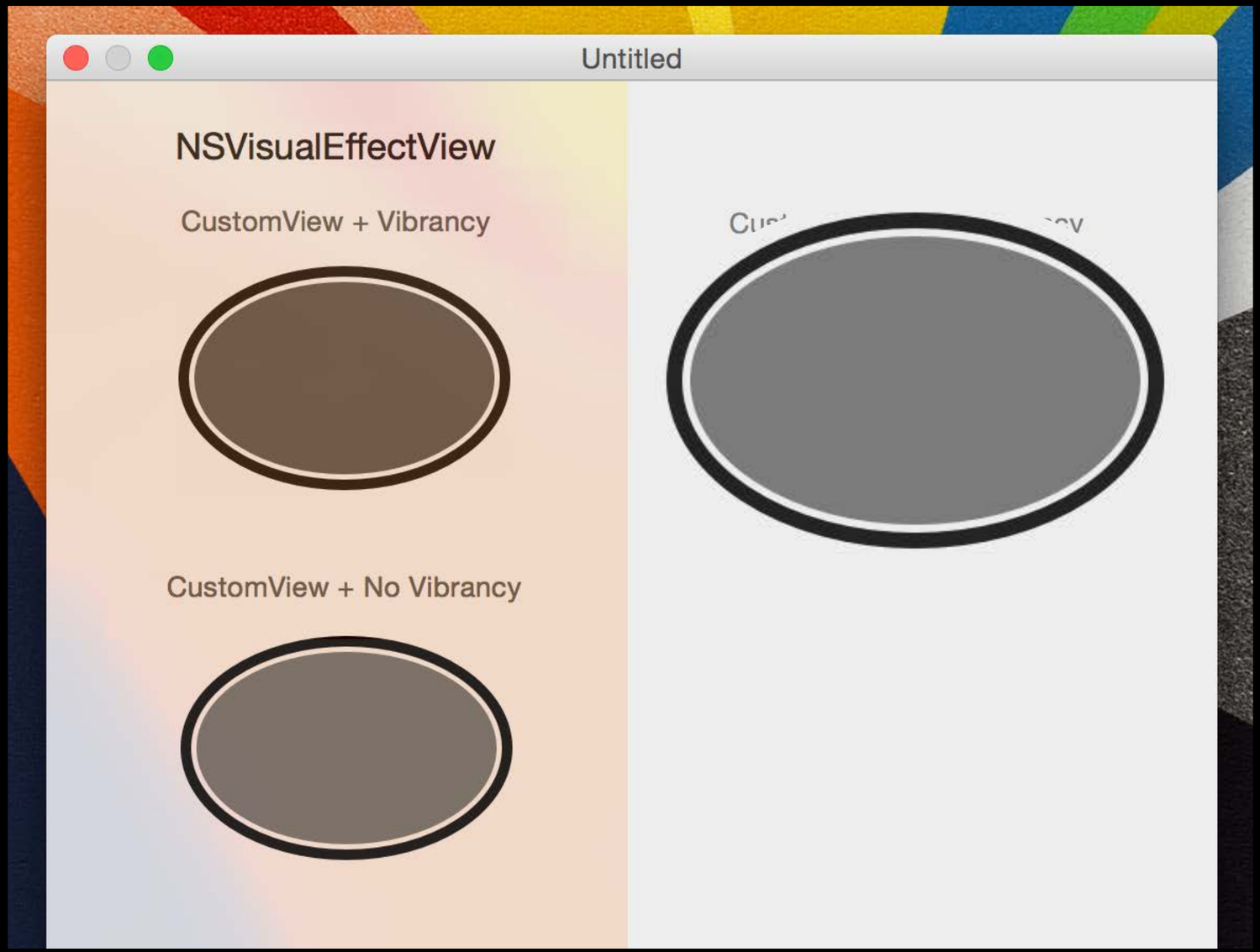
Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy



New Visual Effects and Vibrancy

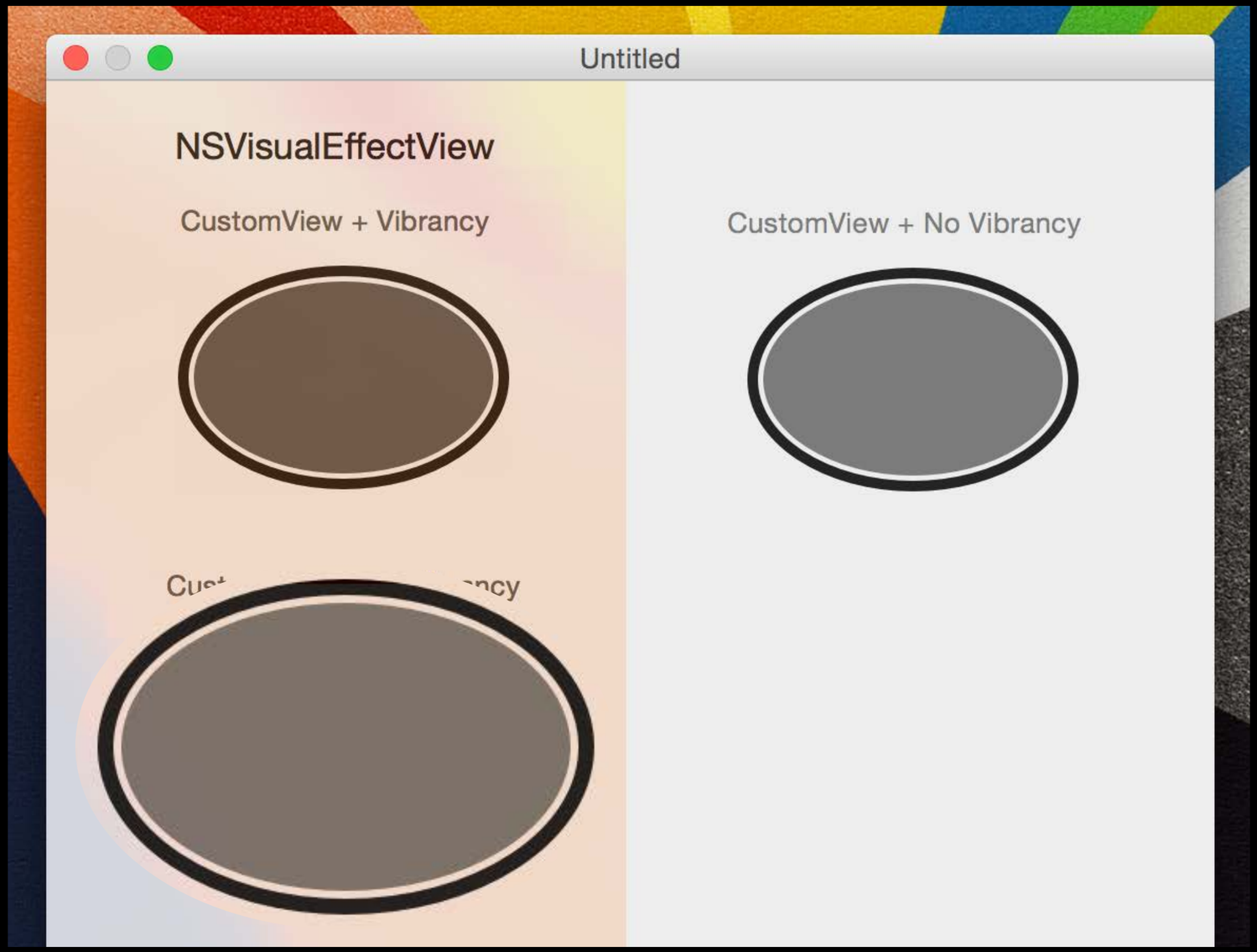
Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy



New Visual Effects and Vibrancy

Custom views and Vibrancy

New Visual Effects and Vibrancy

Custom views and Vibrancy

```
@implementation CustomView

- (BOOL)allowsVibrancy {
    return YES;
}
```

```
- (void)drawRect:(NSRect)dirtyRect {
    ...
```

New Visual Effects and Vibrancy

Custom views and Vibrancy

```
@implementation CustomView

- (void)drawRect:(NSRect)dirtyRect {
   [[NSColor colorWithDeviceWhite:0 alpha:0.85] set];
    NSBezierPath *path = [NSBezierPath bezierPathWithOvalInRect:
                                          NSInsetRect(self.bounds, 5, 5)];
   [path stroke];

   [[NSColor colorWithDeviceWhite:0 alpha:0.48] set];
    path = [NSBezierPath bezierPathWithOvalInRect:
                                      NSInsetRect(self.bounds, 10, 10)];
   [path fill];
}
```

New Visual Effects and Vibrancy

Custom views and Vibrancy

If a view says YES to allowsVibrancy then

- Everything drawn in drawRect: will be vibrant
- Including any color and any image drawn will be vibrant
- Don’t have to use [NSColor labelColor] or [NSColor secondaryLabelColor] to get Vibrancy

New Visual Effects and Vibrancy

Utilizing named system colors

System colors will dynamically change based on the effectiveAppearance

- For instance, [NSColor controlTextColor] is dark on light appearances and white on dark appearances
- Note that converting a CGColor, converting it to a colorspace or extracting components will loose the dynamic nature of the color

Changing the appearance automatically invalidates views

New Visual Effects and Vibrancy

Utilizing named system colors

```
- (void)drawRect:(NSRect)dirtyRect {
    NSColorList *colors = [NSColorList colorListNamed:@"System"];
    NSRect rect = self.bounds;
    rect.size.height = 12;
    NSDictionary *attrs = ...; // sets an 8 point font size
    for (NSString *key in colors.allKeys) {
        NSColor *color = [colors colorWithKey:key];
        [color set];
        NSRectFill(rect);
        [key drawInRect:rect withAttributes:attrs];
        rect.origin.y += 12;
    }
}
```

New Visual Effects and Vibrancy

Utilizing named system colors

```
- (void)drawRect:(NSRect)dirtyRect {
    NSColorList *colors = [NSColorList colorListNamed:@"System"];
    NSRect rect = self.bounds;
    rect.size.height = 12;
    NSDictionary *attrs = ...; // sets an 8 point font size
    for (NSString *key in colors.allKeys) {
        NSColor *color = [colors colorWithKey:key];
        [color set];
        NSRectFill(rect);
        [key drawInRect:rect withAttributes:attrs];
        rect.origin.y += 12;
    }
}
```

New Visual Effects and Vibrancy

Utilizing named system colors

```
- (void)drawRect:(NSRect)dirtyRect {
    NSColorList *colors = [NSColorList colorListNamed:@"System"];
    NSRect rect = self.bounds;
    rect.size.height = 12;
    NSDictionary *attrs = ...; // sets an 8 point font size
    for (NSString *key in colors.allKeys) {
        NSColor *color = [colors colorWithKey:key];
        [color set];
        NSRectFill(rect);
        [key drawInRect:rect withAttributes:attrs];
        rect.origin.y += 12;
    }
}
```

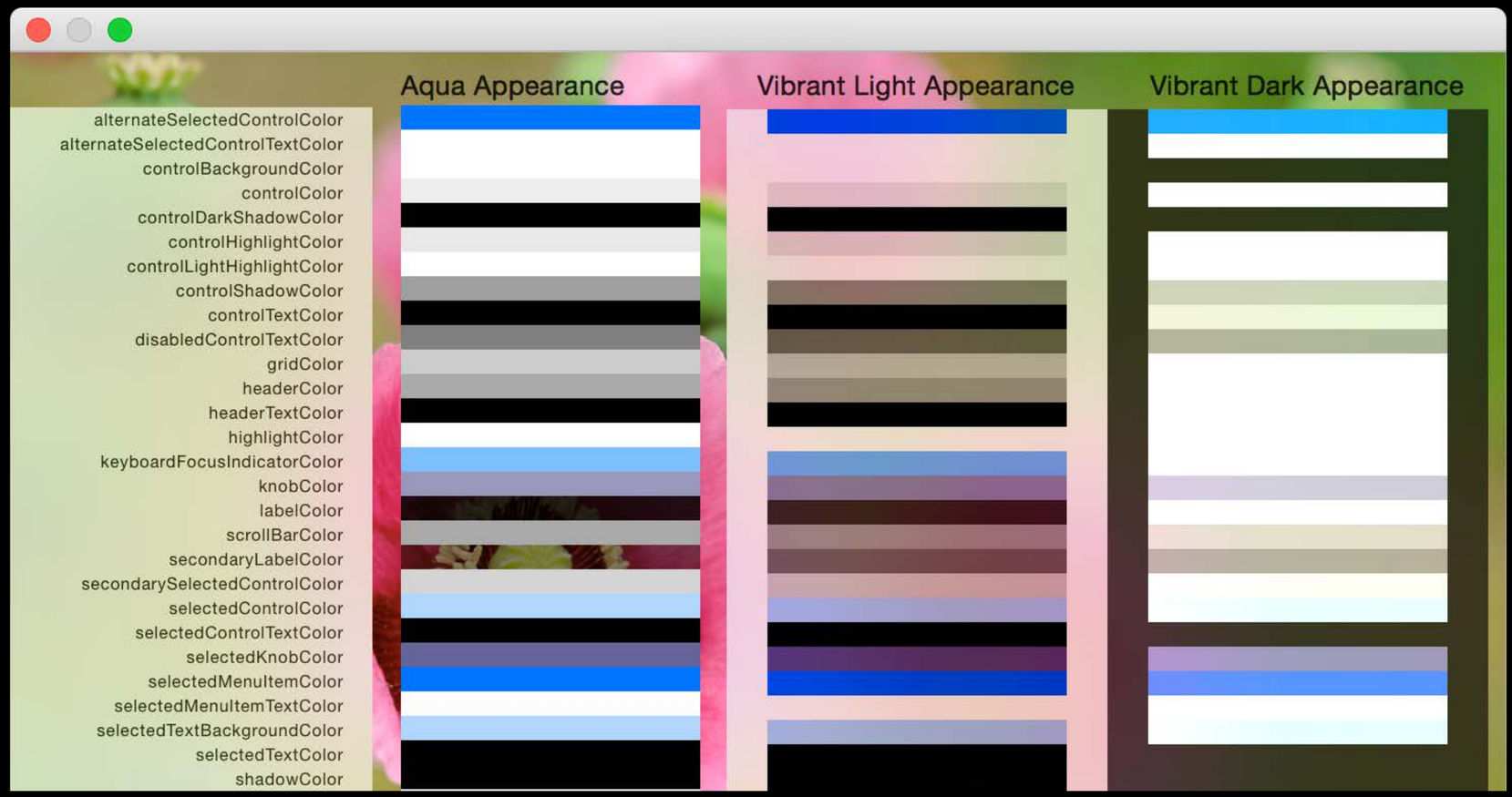
New Visual Effects and Vibrancy

Utilizing named system colors

```
- (void)drawRect:(NSRect)dirtyRect {
    NSColorList *colors = [NSColorList colorListNamed:@"System"];
    NSRect rect = self.bounds;
    rect.size.height = 12;
    NSDictionary *attrs = ...; // sets an 8 point font size
    for (NSString *key in colors.allKeys) {
        NSColor *color = [colors colorWithKey:key];
        [color set];
        NSRectFill(rect);
        [key drawInRect:rect withAttributes:attrs];
        rect.origin.y += 12;
    }
}
```

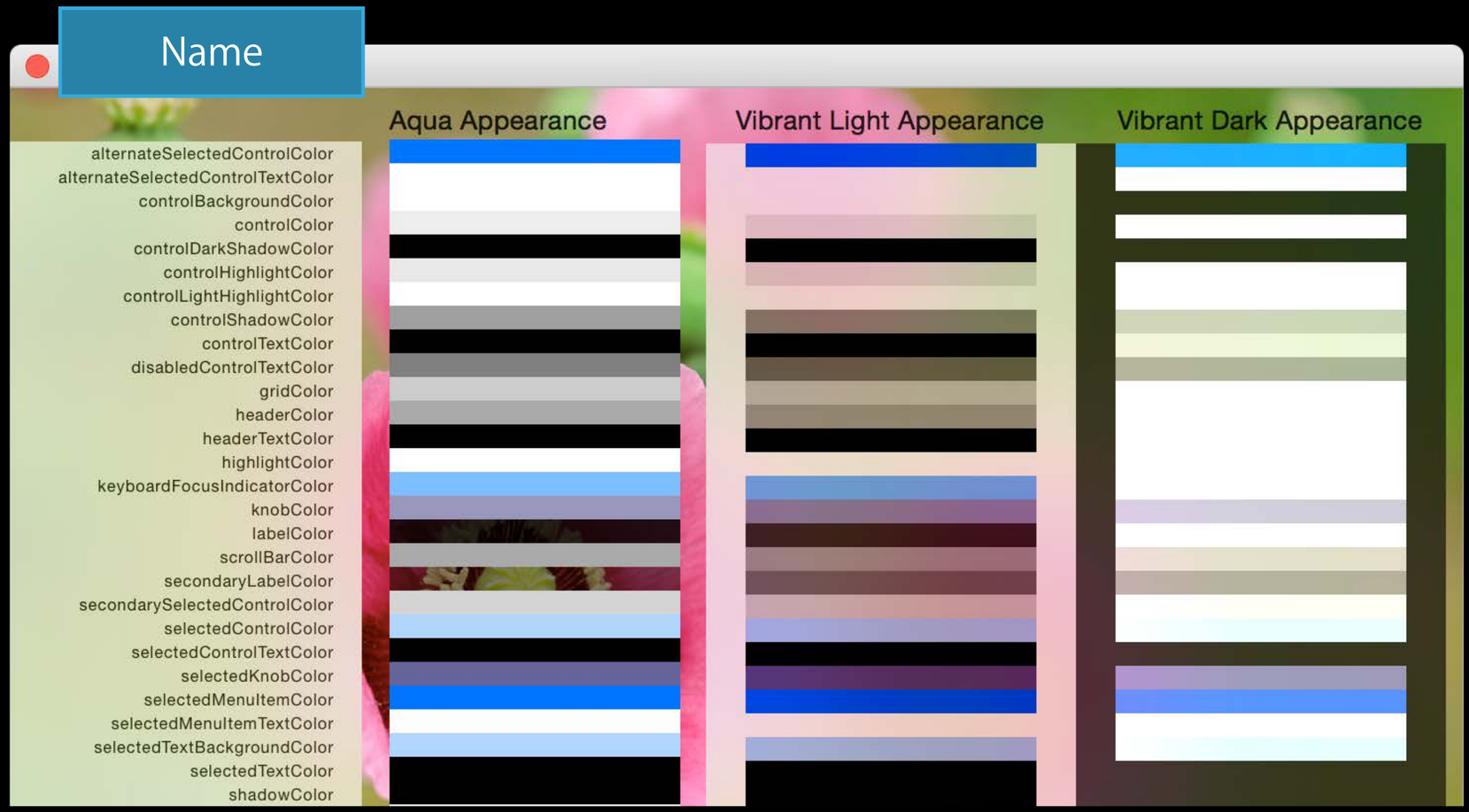
New Visual Effects and Vibrancy

Utilizing named system colors



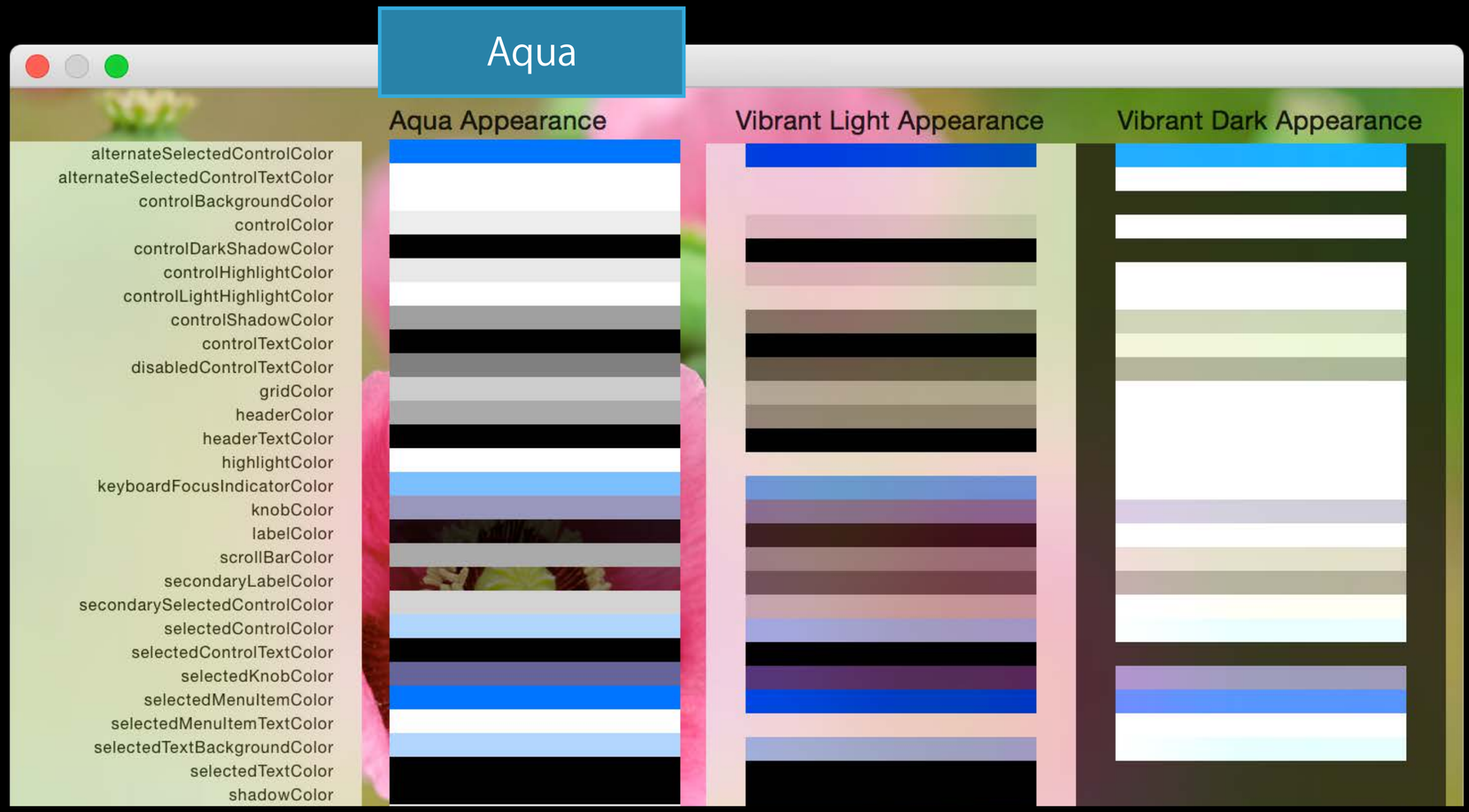
New Visual Effects and Vibrancy

Utilizing named system colors



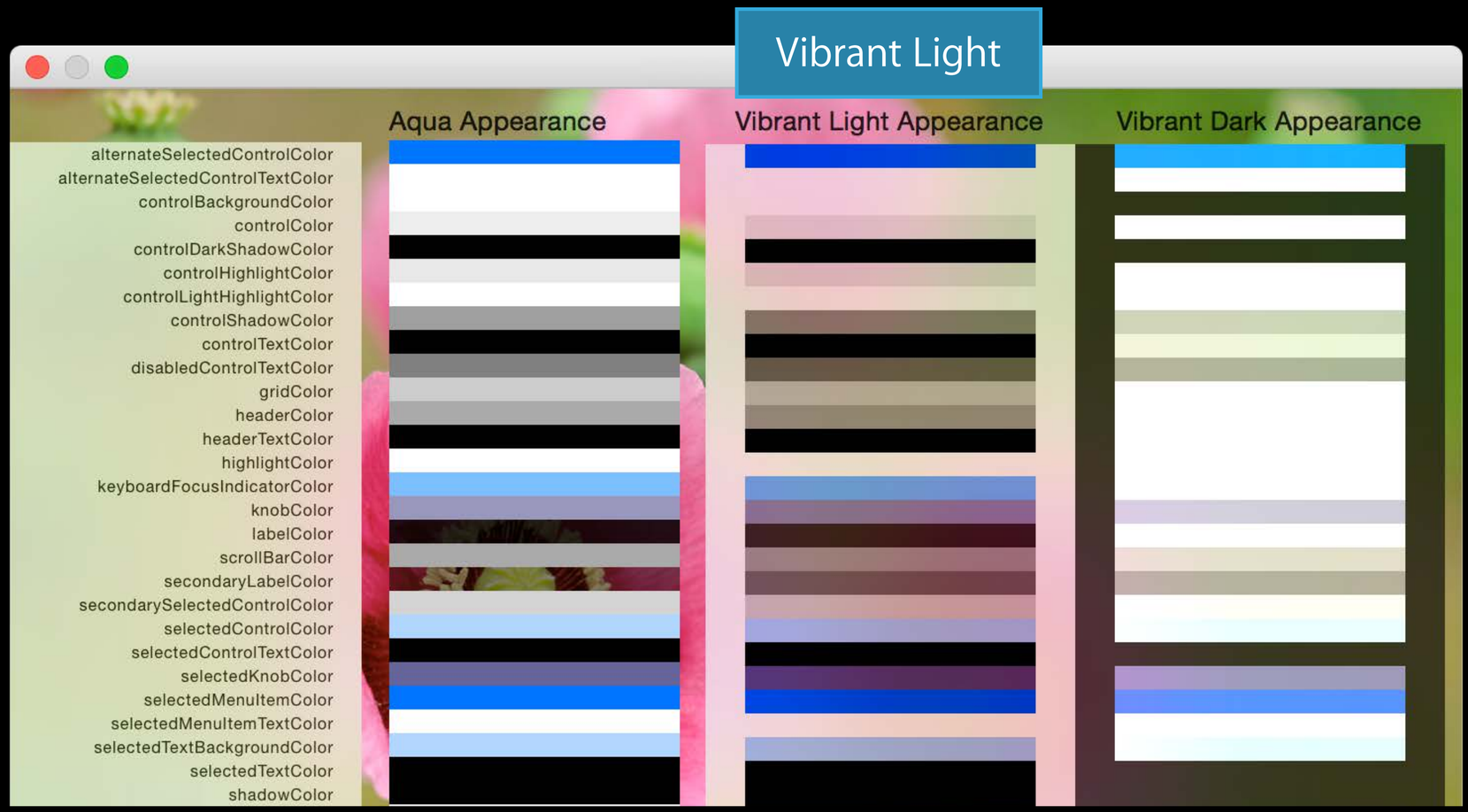
New Visual Effects and Vibrancy

Utilizing named system colors



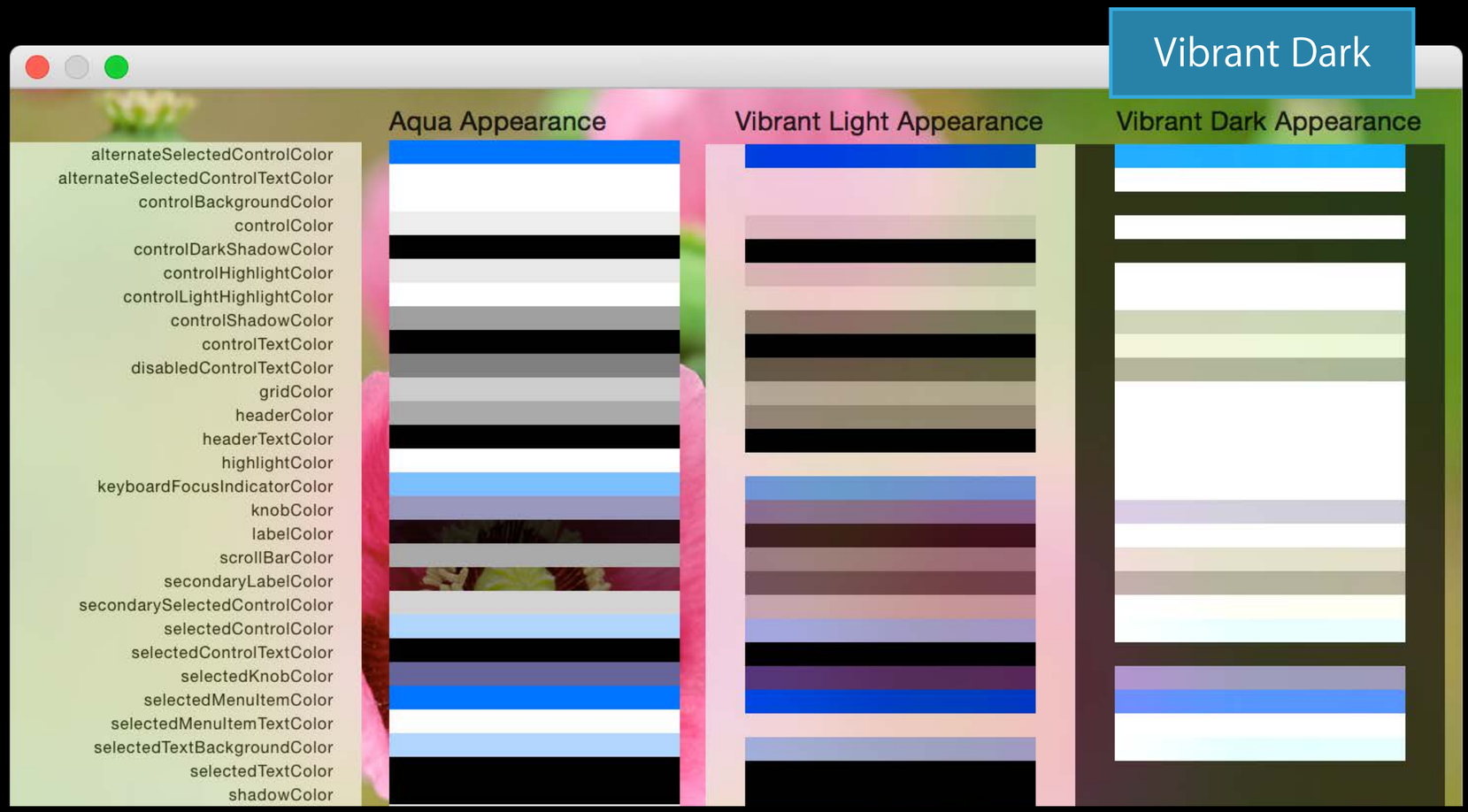
New Visual Effects and Vibrancy

Utilizing named system colors



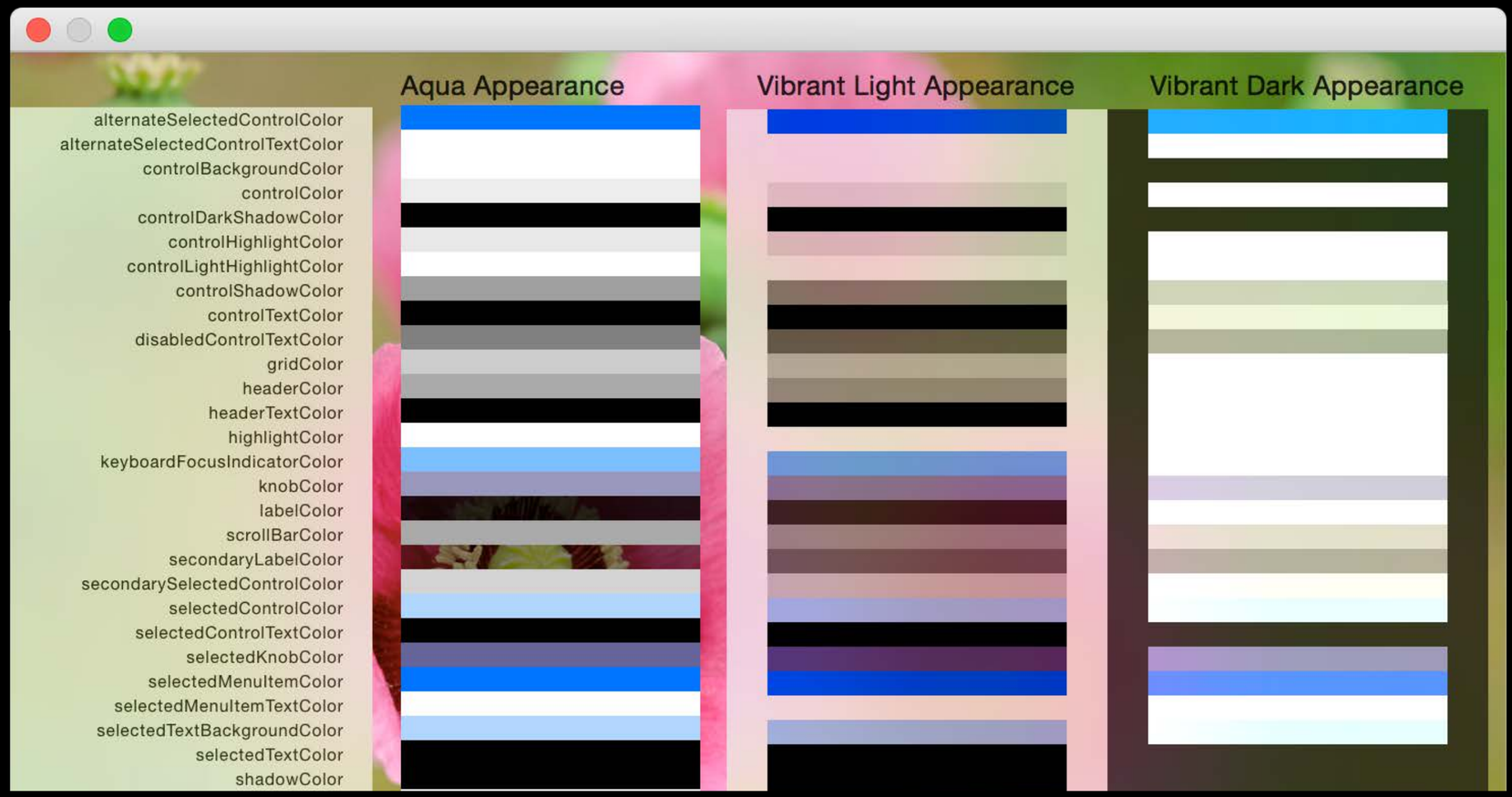
New Visual Effects and Vibrancy

Utilizing named system colors



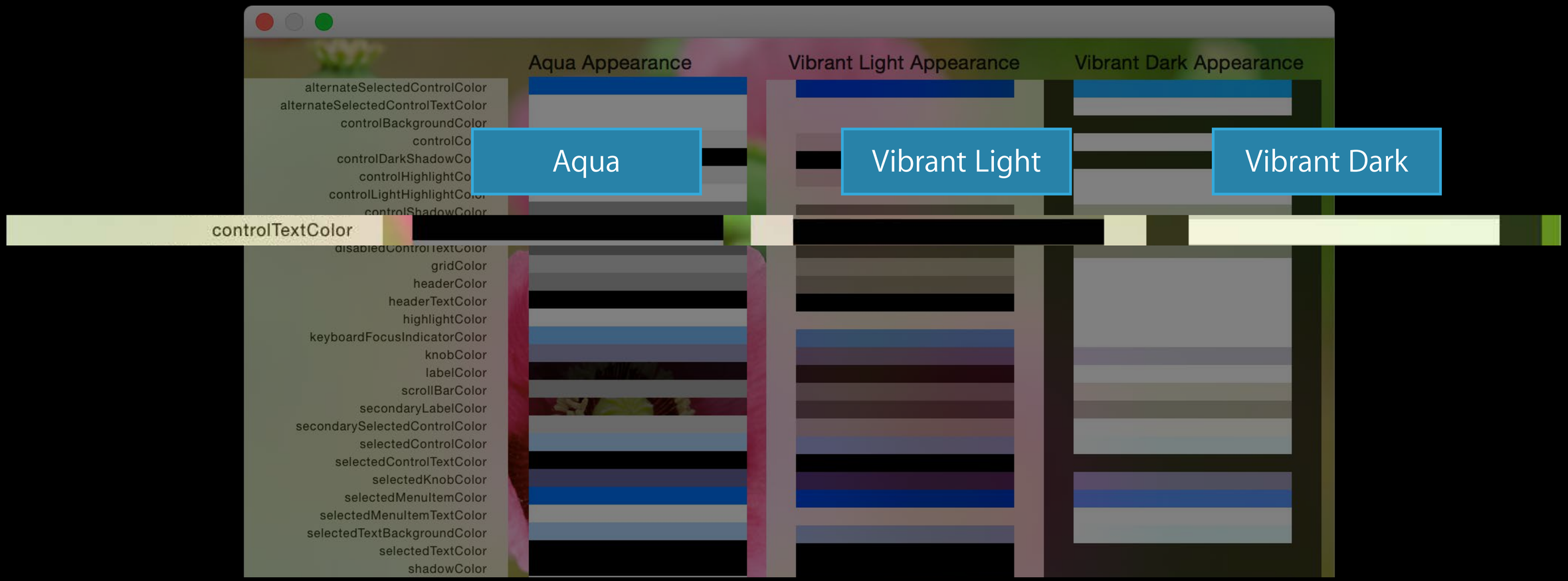
New Visual Effects and Vibrancy

Utilizing named system colors



New Visual Effects and Vibrancy

Utilizing named system colors



New Visual Effects and Vibrancy

Works on a dark and light appearance

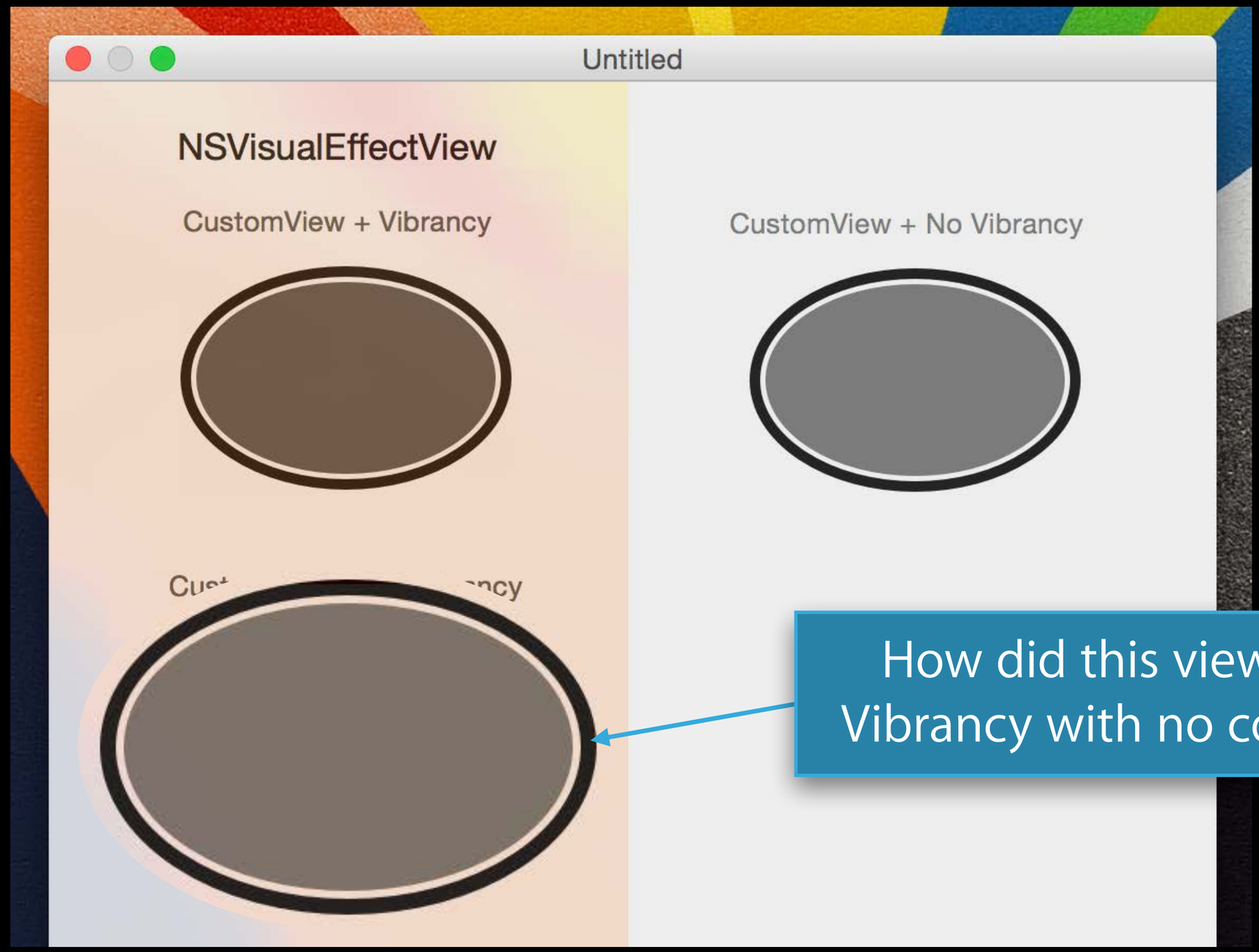
```
@implementation CustomView

- (void)drawRect:(NSRect)dirtyRect {
   [[NSColor labelColor] set];
    NSBezierPath *path = [NSBezierPath bezierPathWithOvalInRect:
                                                NSInsetRect(self.bounds, 5, 5)];
   [path stroke];

   [[NSColor secondaryLabelColor] set];
    path = [NSBezierPath bezierPathWithOvalInRect:
                                            NSInsetRect(self.bounds, 10, 10)];
   [path fill];
}
```

New Visual Effects and Vibrancy

Opting out of Vibrancy

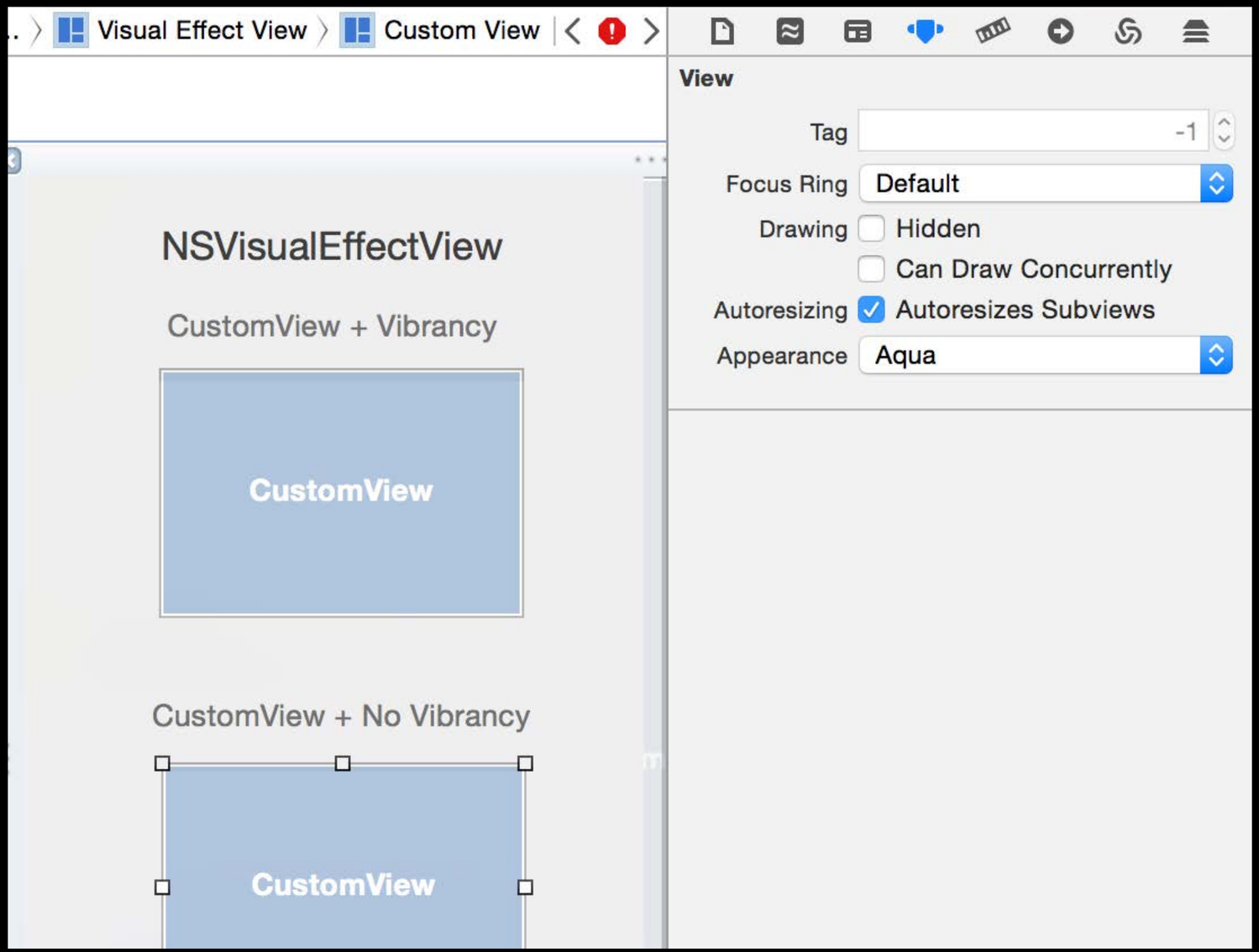
New Visual Effects and Vibrancy

Opting out of Vibrancy



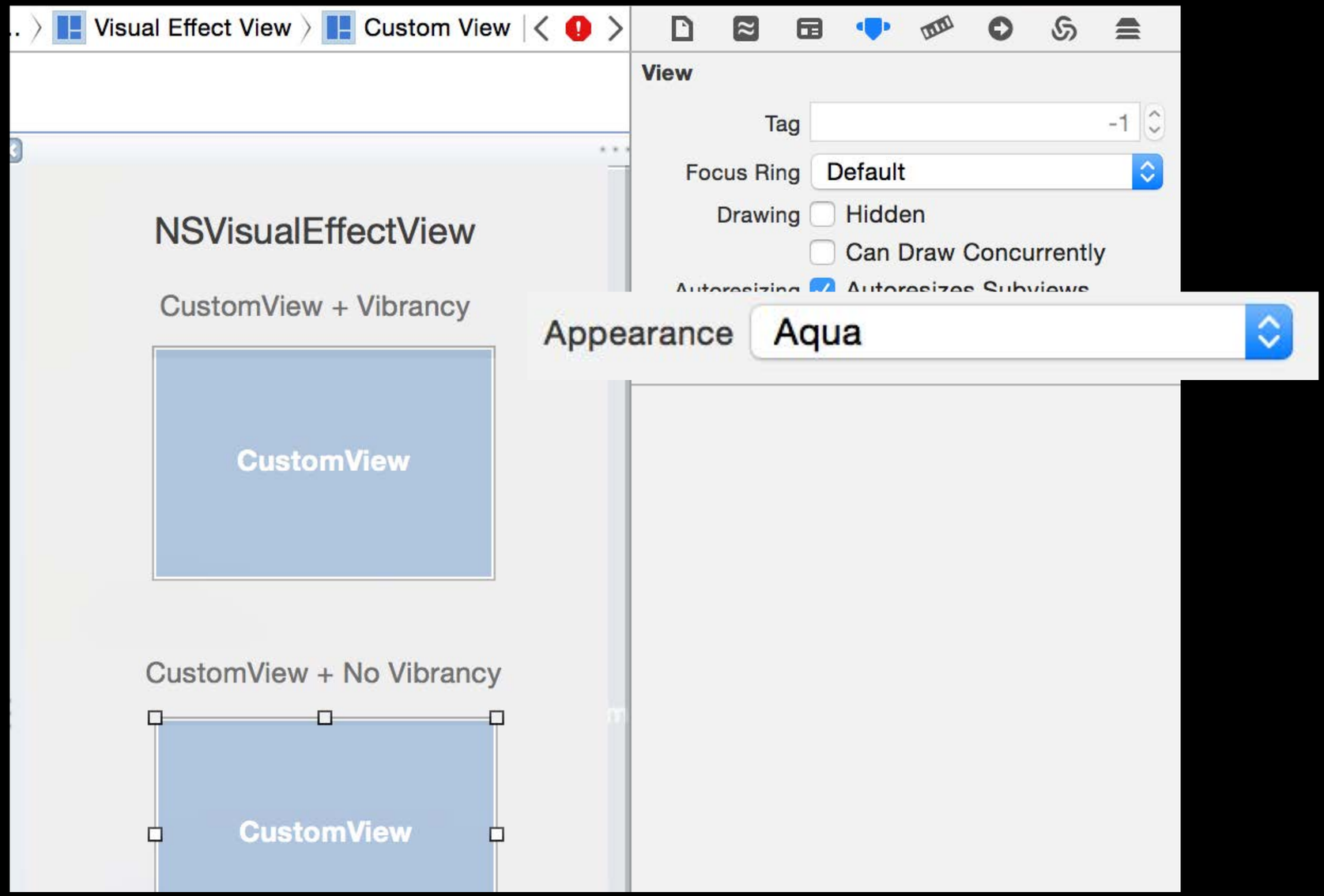
New Visual Effects and Vibrancy

Opting out of Vibrancy



New Visual Effects and Vibrancy

Opting out of Vibrancy



New Visual Effects and Vibrancy

Opting out of Vibrancy

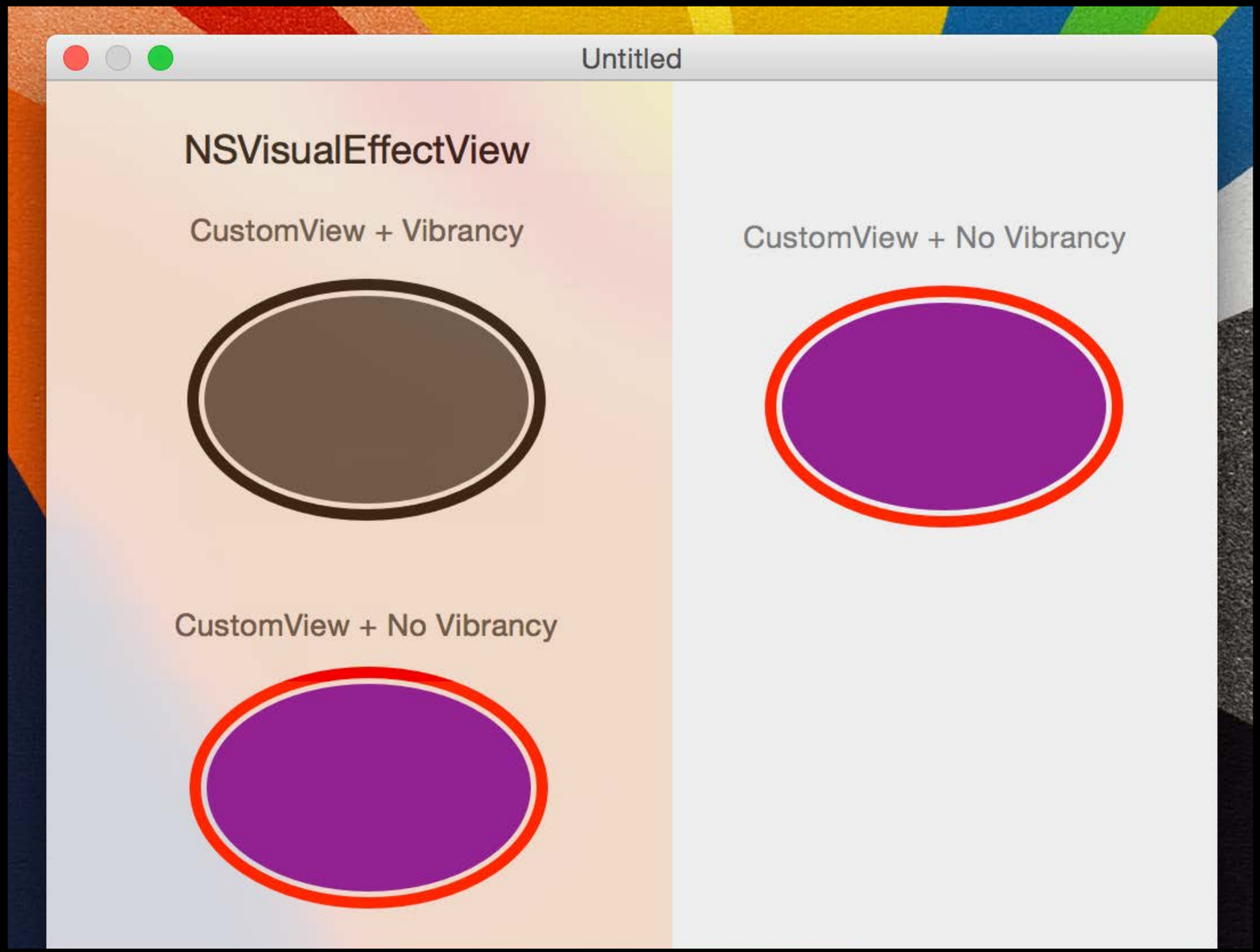
Set appearance back to NSAppearanceNamedAqua on the individual view

Or set the appearance on a container view

- All children will opt-out of Vibrancy
- Can’t be a subview of a view that already was vibrant (allowsVibrancy=YES)

New Visual Effects and Vibrancy

Drastically different vibrant code drawing



New Visual Effects and Vibrancy

Custom views and Vibrancy

Desired changes could include

- Different artwork when vibrant vs. non-vibrant
- Different colors when vibrant vs. non-vibrant

New Visual Effects and Vibrancy

Custom views and Vibrancy

```
- (void)drawRect:(NSRect)dirtyRect {
    if (self.effectiveAppearance.allowsVibrancy) {
        [NSColor.labelColor set];
    } else {
        [NSColor.redColor set];
    }
    NSBezierPath *path = [NSBezierPath
                    bezierPathWithOvalInRect:NSInsetRect(self.bounds, 5, 5)];
    [path stroke];
    ...
```

New Visual Effects and Vibrancy

Caveats within window blending

New Visual Effects and Vibrancy

Caveats within window blending

allowsVibrancy == YES

New Visual Effects and Vibrancy

Caveats within window blending

allowsVibrancy == YES

Incorrectly vibrant NSImageView

New Visual Effects and Vibrancy

Caveats within window blending

allowsVibrancy == YES

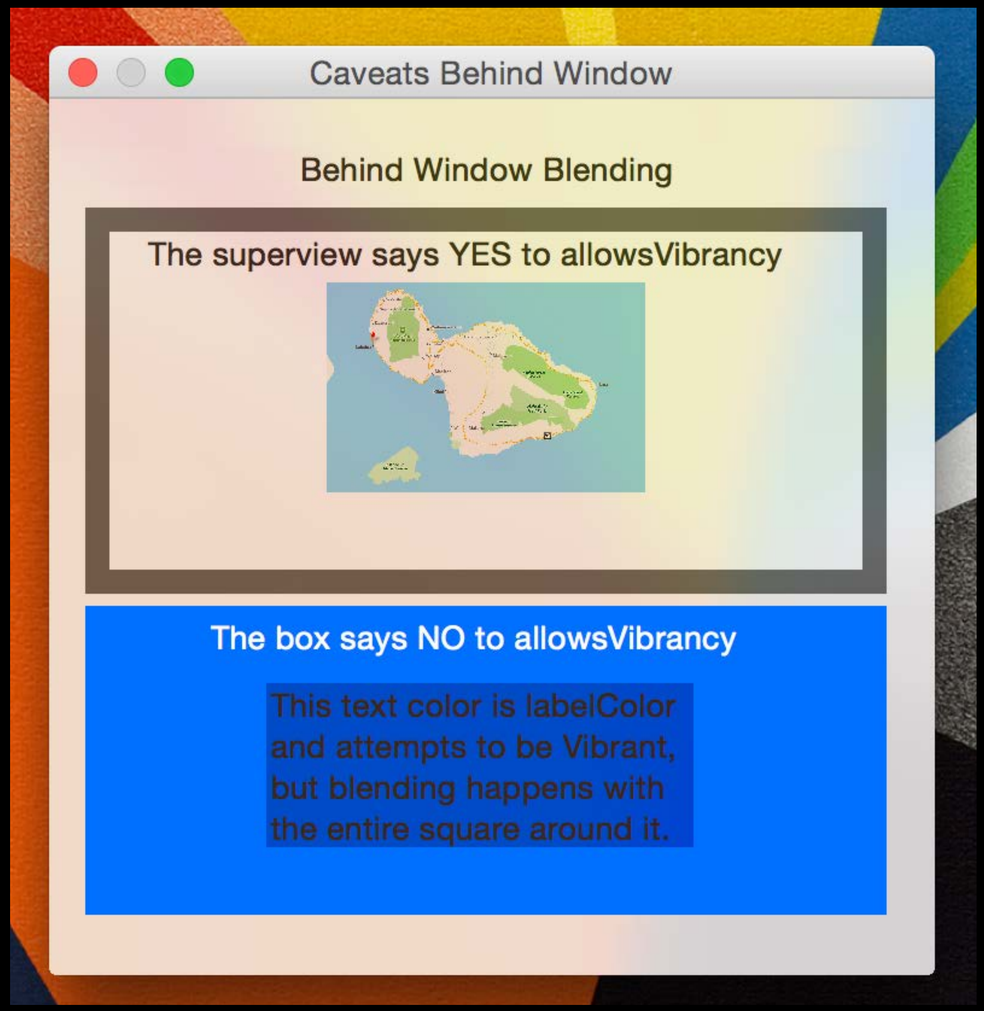
Custom view returns YES from allowsVibrancy

- All its subviews will be vibrant!
- If a parent is vibrant, then all children need to be designed with Vibrancy in mind

New Visual Effects and Vibrancy

Caveats with behind window blending

Once vibrant, always vibrant



New Visual Effects and Vibrancy

Caveats with behind window blending

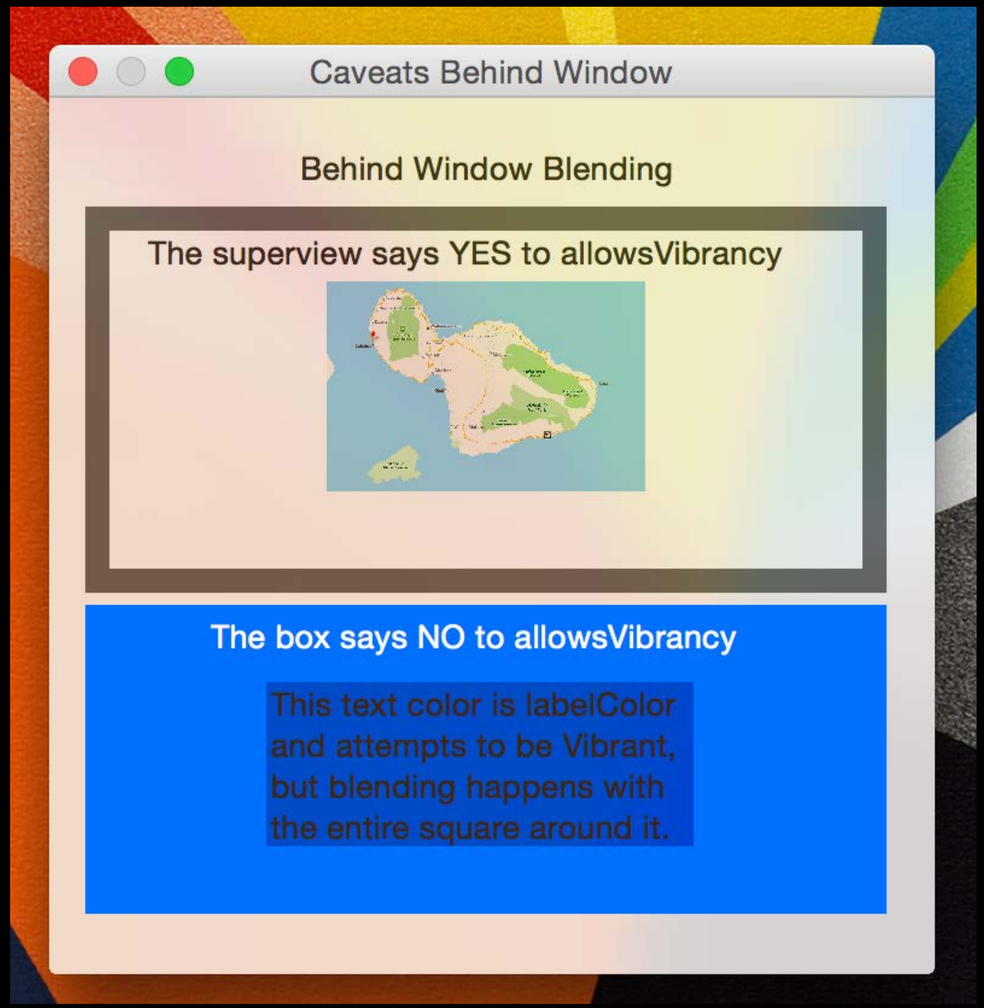
Additional problem with vibrant drawing over non-vibrant drawing

Don’t use vibrant text on a non-vibrant superview

- Vibrant Text: using labelColor or secondaryLabelColor

Instead, use controlTextColor

Or have the superview say YES to allowsVibrancy



New Visual Effects and Vibrancy

Caveats with behind window blending

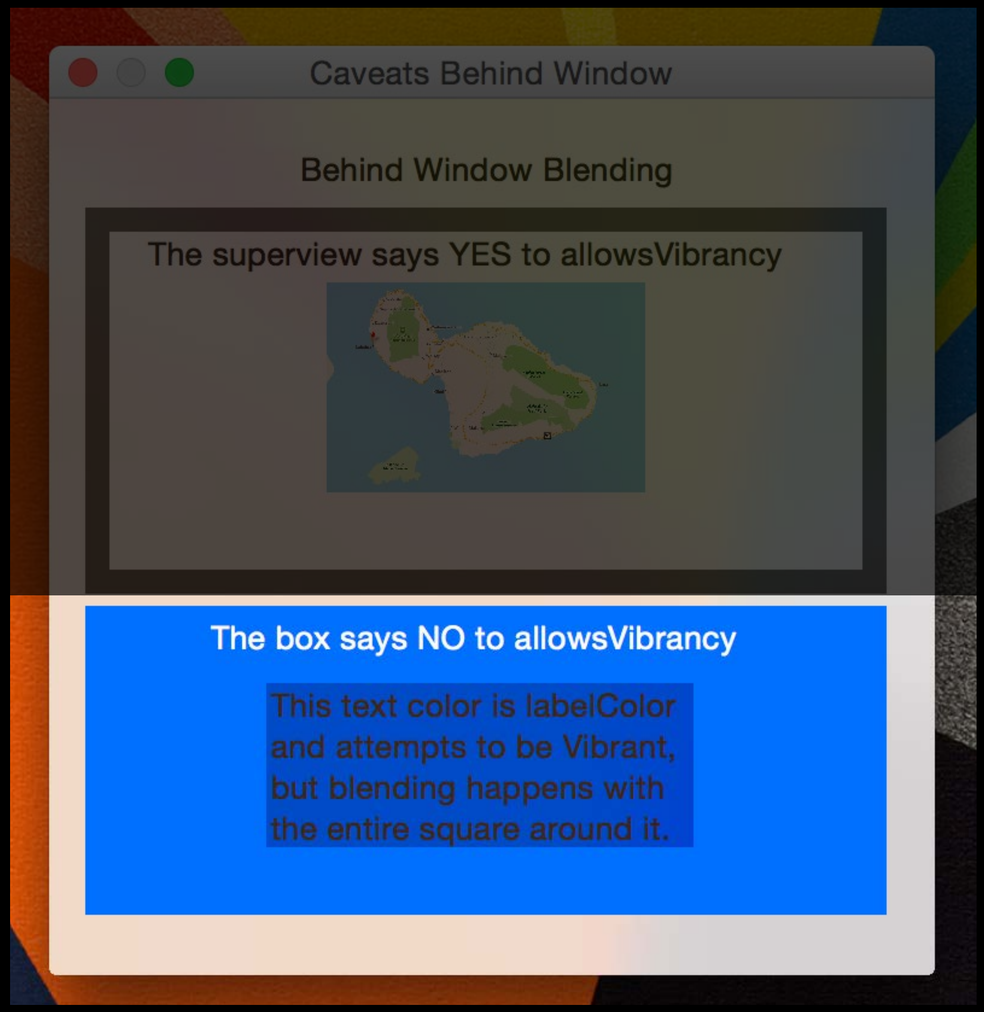
Additional problem with vibrant drawing over non-vibrant drawing

Don’t use vibrant text on a non-vibrant superview

- Vibrant Text: using labelColor or secondaryLabelColor

Instead, use controlTextColor

Or have the superview say YES to allowsVibrancy



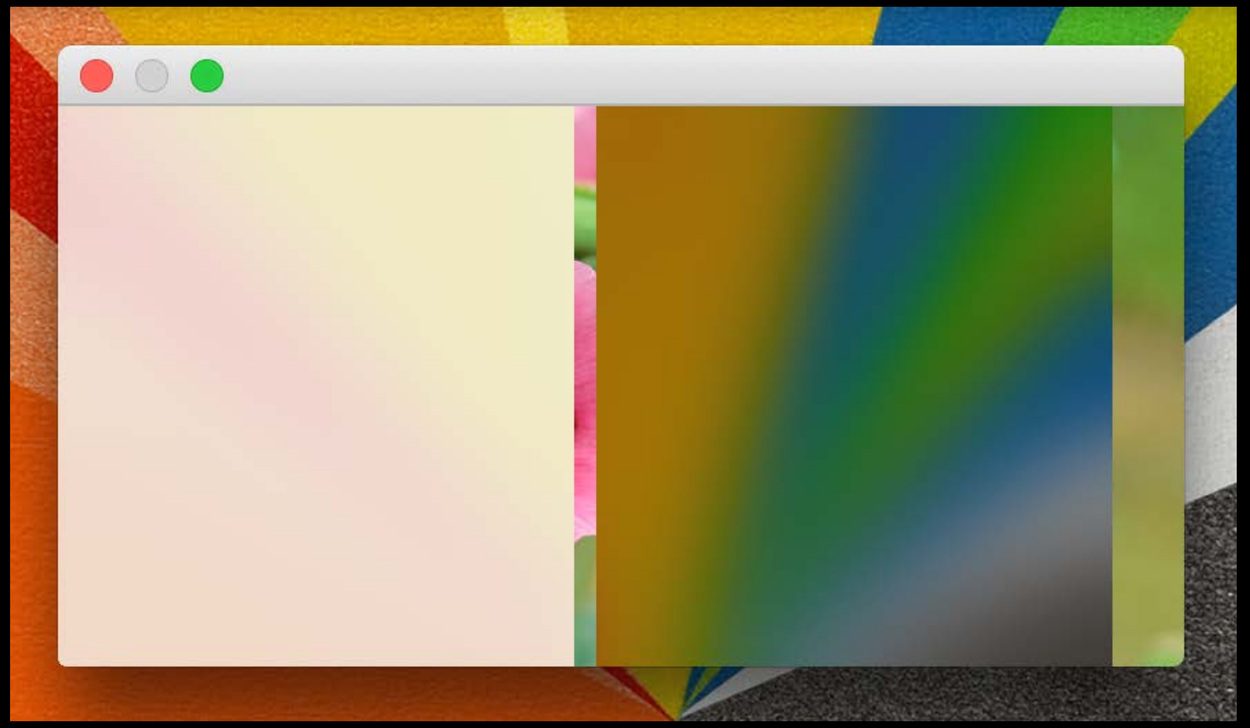
New Visual Effects and Vibrancy

Masks

```
@interface NSVisualEffectView : NSView ...

@property(retain) NSImage *maskImage;

...
```

New Visual Effects and Vibrancy

Masks with standard template images



New Visual Effects and Vibrancy

Masks with standard template images

New Visual Effects and Vibrancy

Masks with custom drawing

```
- (void)viewDidLoad {
    [super viewDidLoad]; // Good practice to call super

   NSRect bounds = self.view.bounds;
   self.visualEffectView.maskImage = [NSImage imageWithSize:bounds.size
                                                    flipped:YES drawingHandler:^BOOL(NSRect dstRect) {
       NSBezierPath *path = [NSBezierPath bezierPathWithRoundedRect:bounds
                                                            xRadius:10 yRadius:10];
       [path fill];
       return YES;
   }];
}
```

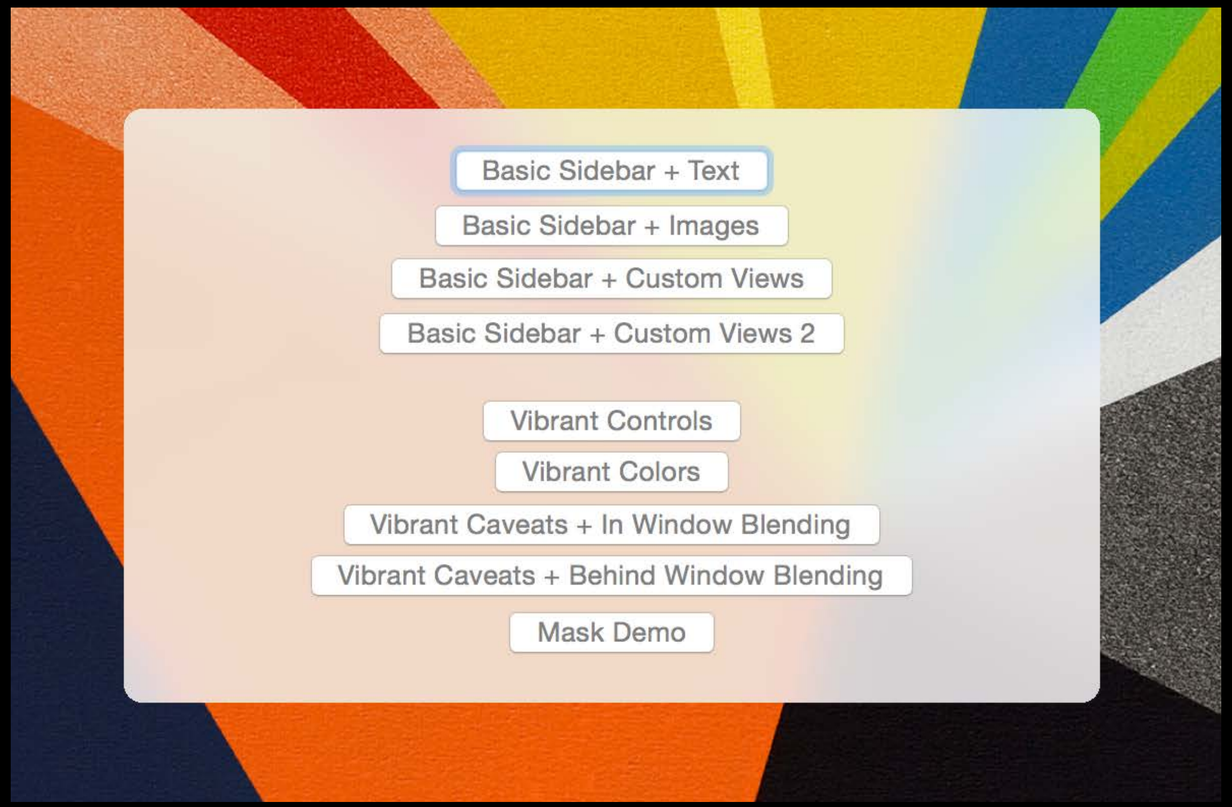
New Visual Effects and Vibrancy

Masks with custom drawing

```
- (void)viewDidLoad {
    [super viewDidLoad]; // Good practice to call super

   NSRect bounds = self.view.bounds;
   self.visualEffectView.maskImage = [NSImage imageWithSize:bounds.size
                                                    flipped:YES drawingHandler:^BOOL(NSRect dstRect) {
        NSBezierPath *path = [NSBezierPath bezierPathWithRoundedRect:bounds
                                                             xRadius:10 yRadius:10];
        [path fill];
        return YES;
    }];
}
```

New Visual Effects and Vibrancy

Masks with custom drawing



New Visual Effects and Vibrancy

Mask image notes

The mask image is drawn with [NSImage drawInRect:]

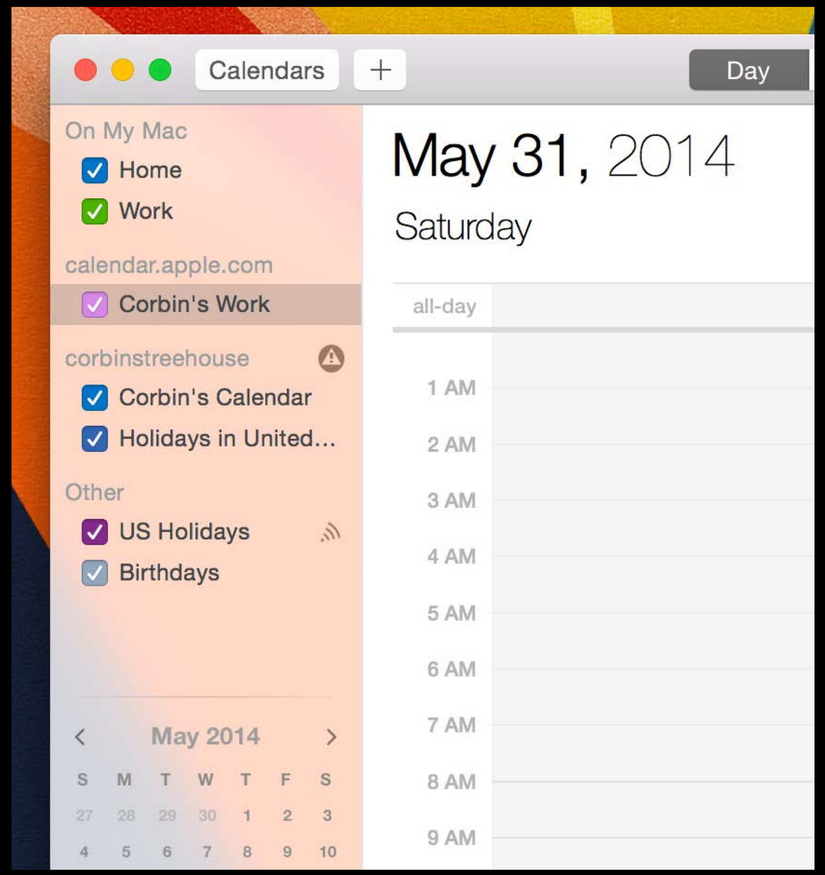
Utilize the new NSImage capInsets property to specify how it stretches

Standard AppKit Views and Vibrancy

Standard AppKit Views and Vibrancy

NSTableView/NSOutlineView

All source list sidebars automatically get an NSVisualEffectView and Vibrant appearance



Standard AppKit Views and Vibrancy

NSTableView/NSOutlineView

NSTableView and NSOutlineView are considered a source list when

selectionHighlightStyle == NSTableViewSelectionHighlightStyleSourceList

Side effects happen when setting the style

- NSOutlineView's indentation and spacing is affected
- intercellSpacing.height is affected (new to 10.10)
- The position and size of NSTableCellView's imageView and textField is controlled
- Attributed are applied to text
- The backgroundColor is changed to an internal color

Standard AppKit Views and Vibrancy

NSTableView/NSOutlineView on 10.10

When selectionHighlightStyle == NSTableViewSelectionHighlightStyleSourceList

- Selection is now a special blue material that does behind window blending
  - The material size and drawing can not be customized
- If the backgroundColor was not changed, an NSVisualEffectView is placed behind the table to do behind window blurring
  - Changing the backgroundColor allows you to opt-out of this behavior
- The appearance is set to NSAppearanceNameVibrantLight

Standard AppKit Views and Vibrancy

NSPopover on 10.10

NSPopover now utilizes Light and Dark materials

NSPopover will have its appearance set to the VibrantLight appearance

Performance

Chris Dreessen
AppKit Software Engineer

Performance

Performance

Blur is not free

Performance

Blur is not free

Impacts graphics performance and battery usage

Performance

Blur is not free

Impacts graphics performance and battery usage

Strike a balance between UI polish and resource utilization

Performance

Performance

Performance

Performance

Performance

Performance

For in window blurs, be aware of frequently updated content

Performance

For in window blurs, be aware of frequently updated content

- Movies
- Animations
- Blinking cursors

Performance

For in window blurs, be aware of frequently updated content

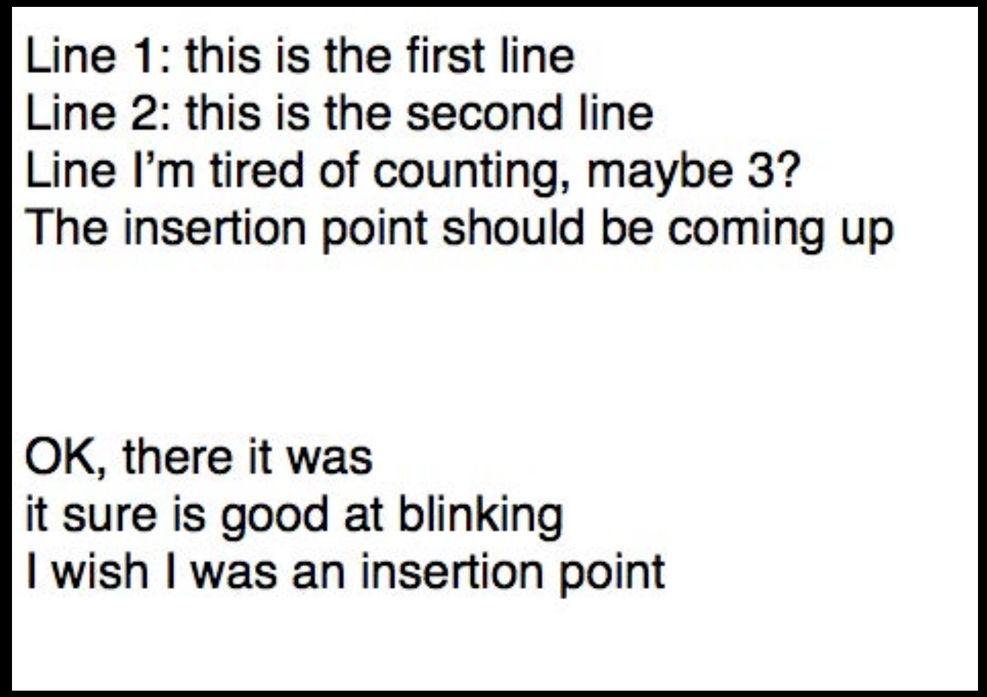
- Movies
- Animations
- Blinking cursors

This applies to behind window blurs as well, but you can’t control the content

Performance

Layers have different dirty shapes than non-layer backed views

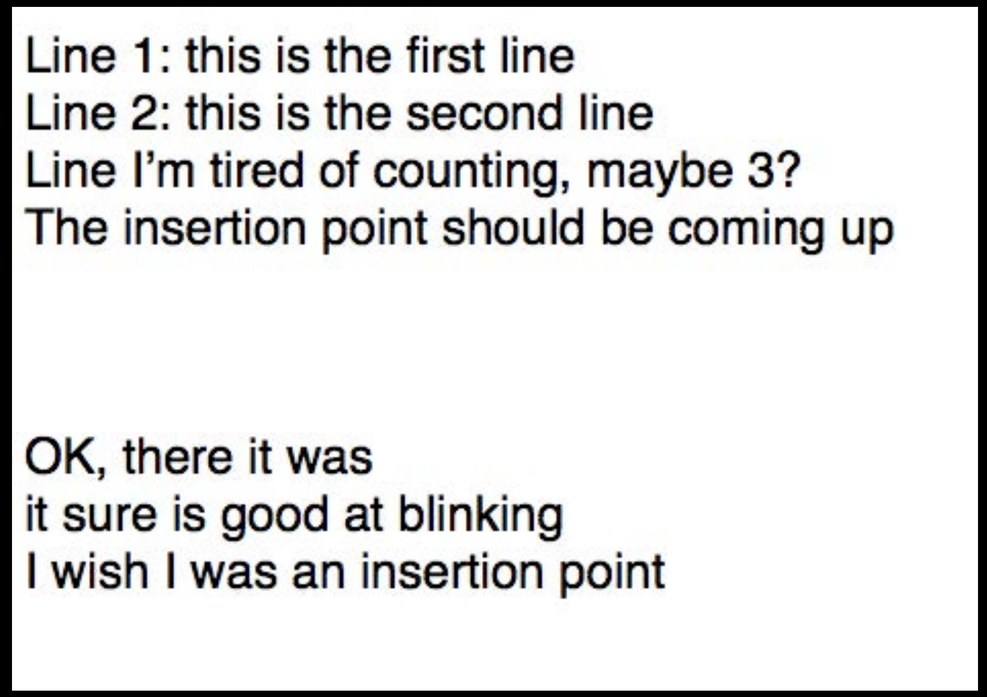
```
[view setWantsLayer:NO];
[view setNeedsDisplayInRect:r];
```



Performance

Layers have different dirty shapes than non-layer backed views

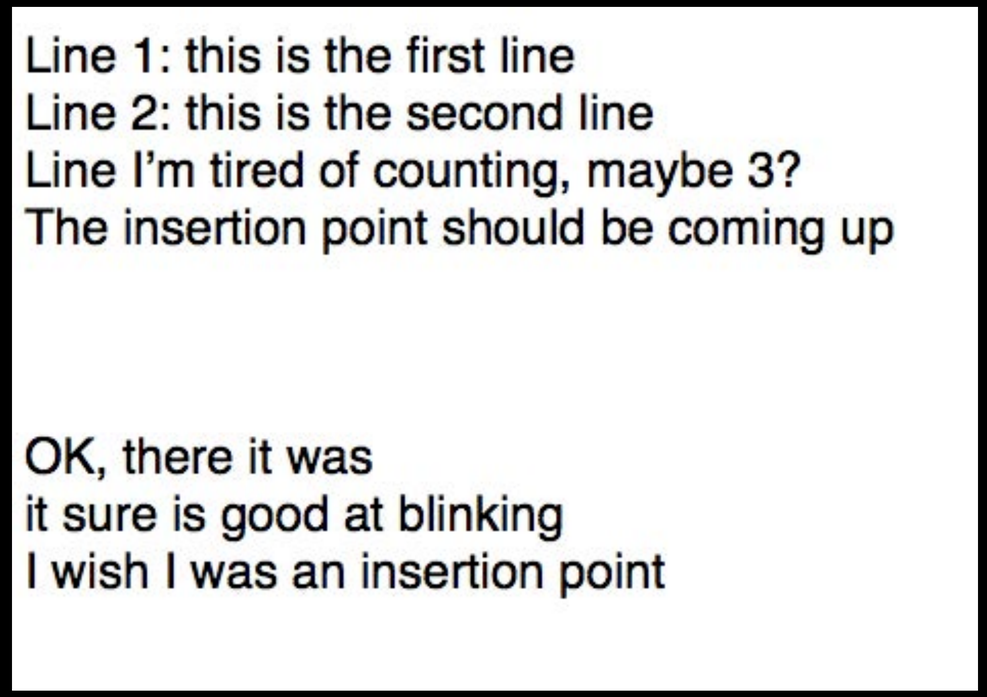
```
[view setWantsLayer:NO];
[view setNeedsDisplayInRect:r];
```



Performance

Layers have different dirty shapes than non-layer backed views

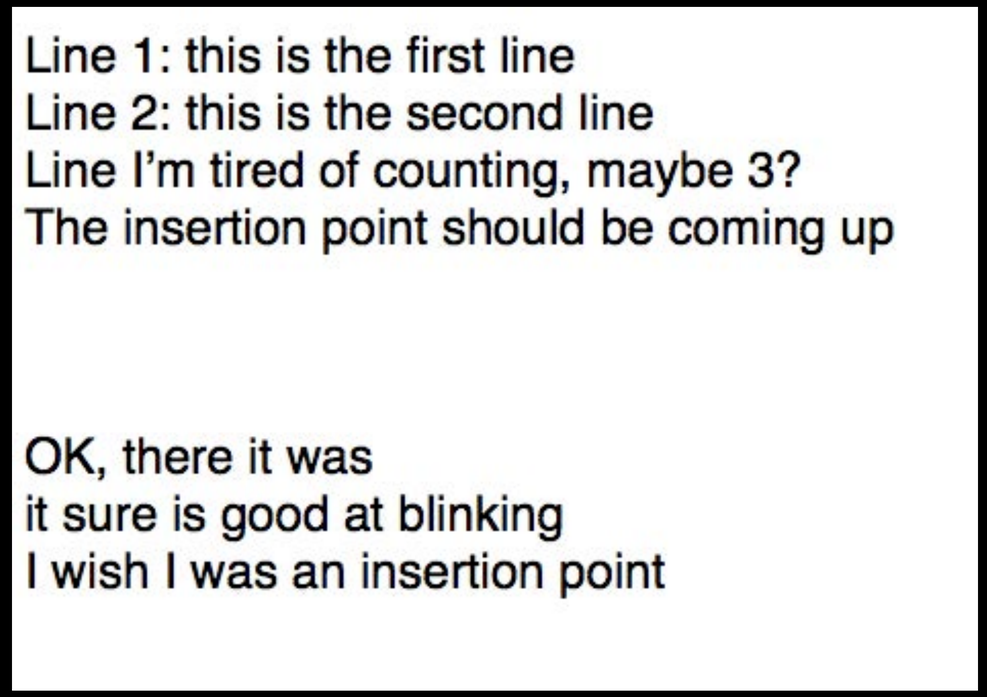
```
[view setWantsLayer:YES];
[view setNeedsDisplayInRect:r];
```



Performance

Layers have different dirty shapes than non-layer backed views

```
[view setWantsLayer:YES];
[view setNeedsDisplayInRect:r];
```



Performance

Performance

This is true even if you’re using -updateLayer or setting layer properties directly

Performance

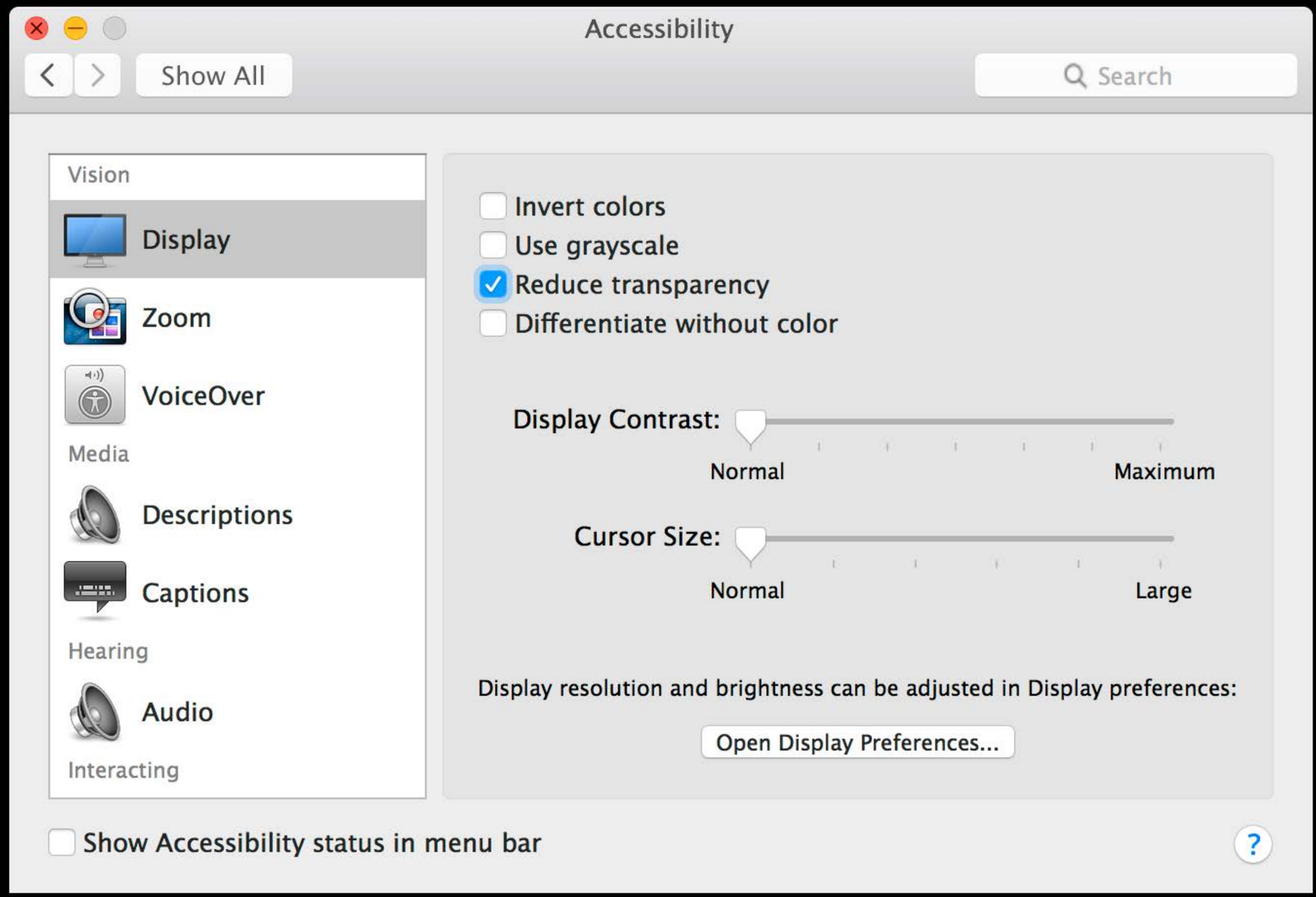
This is true even if you're using -updateLayer or setting layer properties directly

Sometimes you can refactor your view hierarchy to avoid this

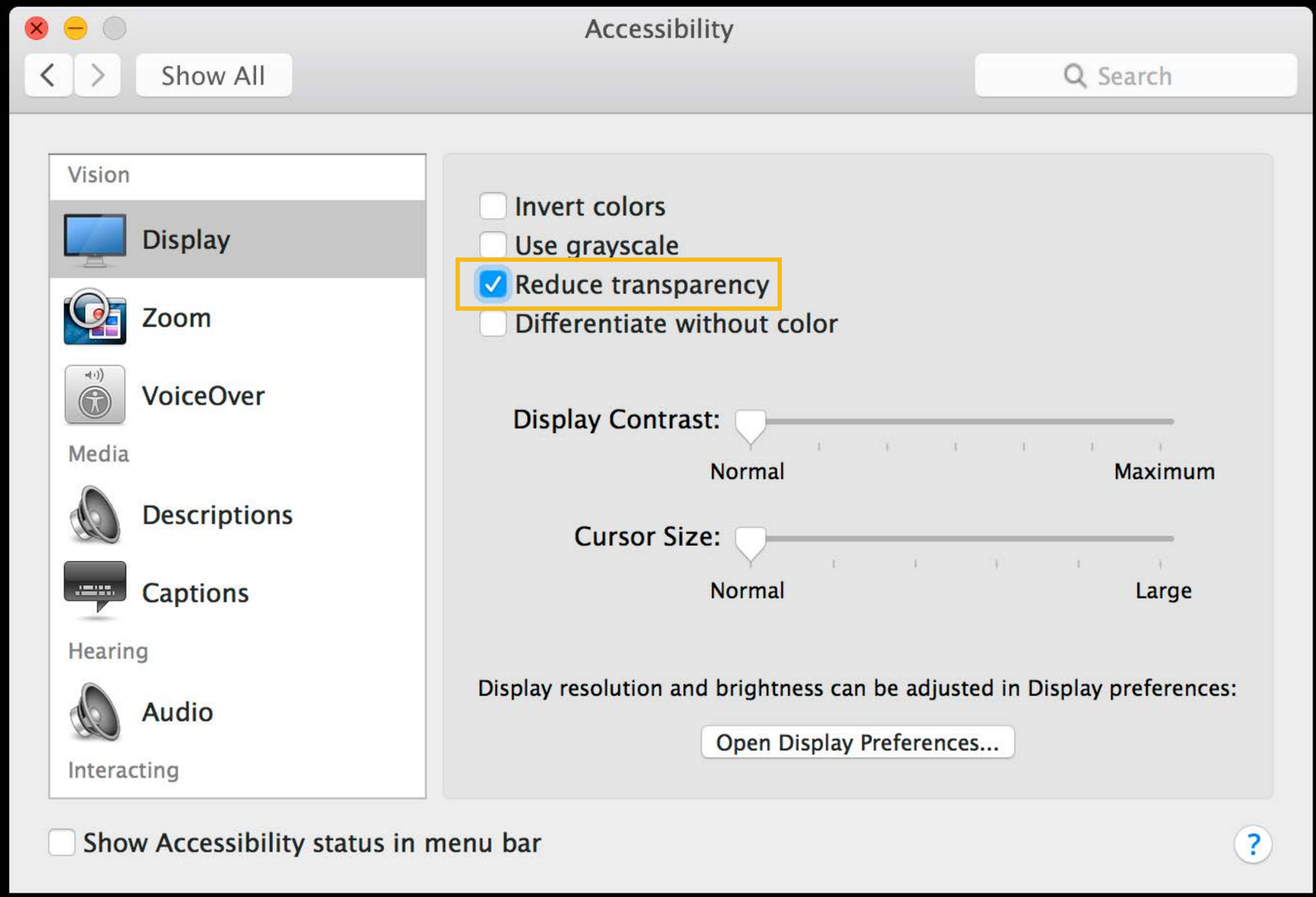
Performance

Don’t fiddle with `NSWindow.opaque`

Performance



Performance



Performance

Performance

Quartz Debug

Performance

Quartz Debug

Instruments

Performance

Quartz Debug

Instruments

Activity Monitor

Performance

Quartz Debug

Instruments

Activity Monitor

sample

Summary

Summary

NSImage drawing enhancements with capInsets

Summary

NSImage drawing enhancements with capInsets

New window features

Summary

NSImage drawing enhancements with capInsets

New window features

- Full content size

Summary

NSImage drawing enhancements with capInsets

New window features

- Full content size
- Title visibility options

Summary

NSImage drawing enhancements with capInsets

New window features

- Full content size
- Title visibility options
- Transparent title bar

Summary

NSImage drawing enhancements with capInsets

New window features

- Full content size
- Title visibility options
- Transparent title bar

New visual effects, appearances, Vibrancy

Summary

NSImage drawing enhancements with capInsets

New window features

- Full content size
- Title visibility options
- Transparent title bar

New visual effects, appearances, Vibrancy

Performance

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation
What's New in OS X
http://developer.apple.com/

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

Related Sessions

• Adapting Your App to the New UI	Pacific Heights	Tuesday 3:15PM
• What's New in Interface Builder	Mission	Wednesday 3:15PM
• Creating Modern Cocoa Apps	Marina	Thursday 10:15AM

Labs

• New UI and Cocoa Lab	Frameworks Lab B	Wednesday 3:15PM
• Xcode and Interface Builder Lab	Tools Lab C	Thursday 9:00AM
• View Controllers and Cocoa Lab	Frameworks Lab B	Thursday 11:30AM
• Cocoa Lab	Frameworks Lab B	Thursday 4:30PM
• Interface Builder and Auto Layout Lab	Tools Lab C	Friday 9:00AM

WWDC14