Optimizing Drawing and Scrolling

On Mac OS X

Session 215

Corbin Dunn
AppKit Software Engineer

Raleigh Ledet
AppKit Software Engineer

Optimizing AppKit Drawing

Layer-Backed View Drawing
with Core Animation

Responsive Scrolling

Magnification

Optimizing AppKit Drawing

Best practices

Optimizing AppKit Drawing

Optimize -drawRect:

- You are probably already doing this

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

    [NSColor.redColor set];
    NSRectFill(dirtyRect);

}
```

Optimizing AppKit Drawing

Optimize -drawRect:

- And dirtying just the appropriate areas

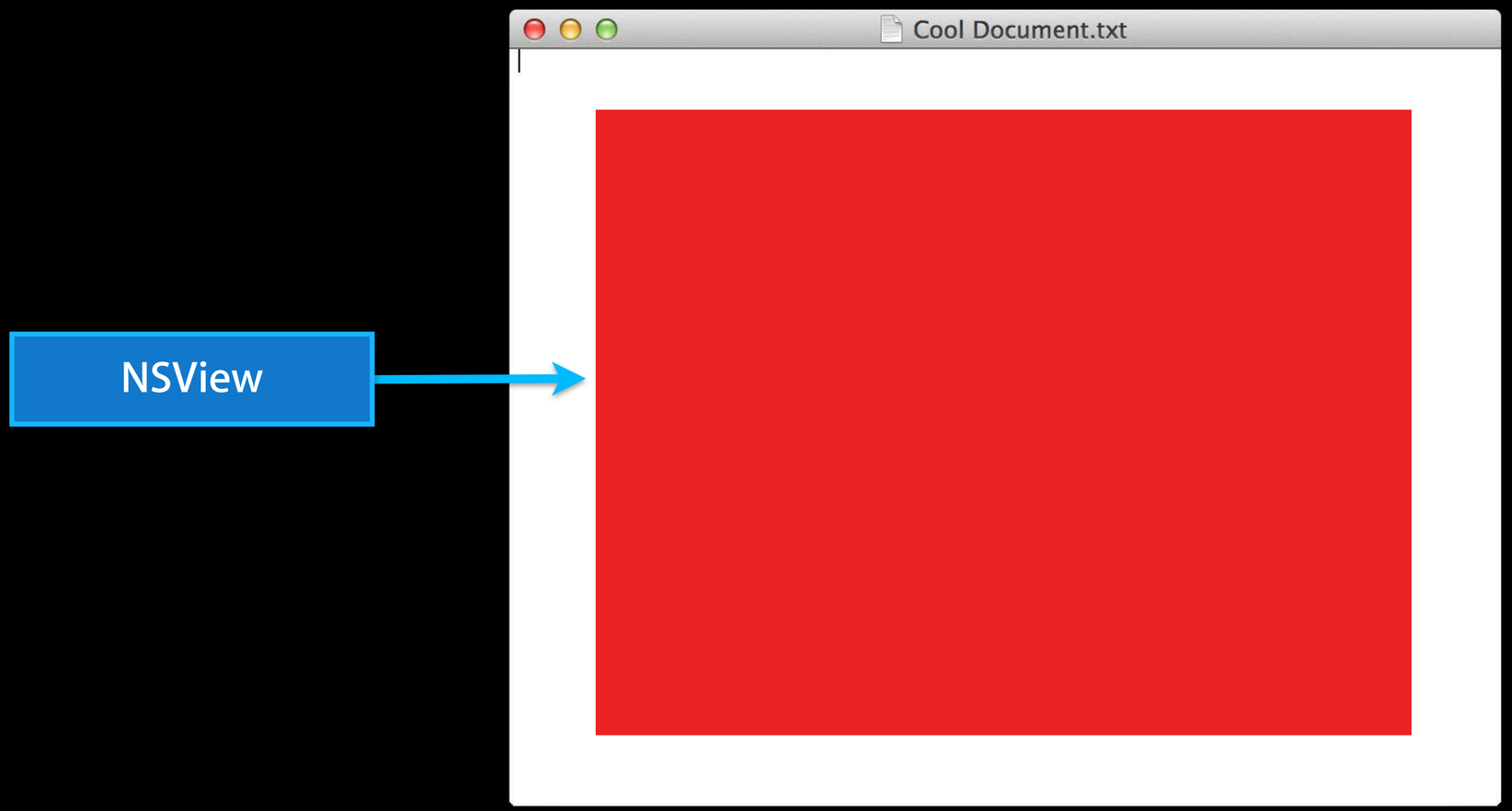
```
[myView setNeedsDisplayInRect:smallDirtyRect];
```

- And not

~~[myView setNeedsDisplay:YES];~~

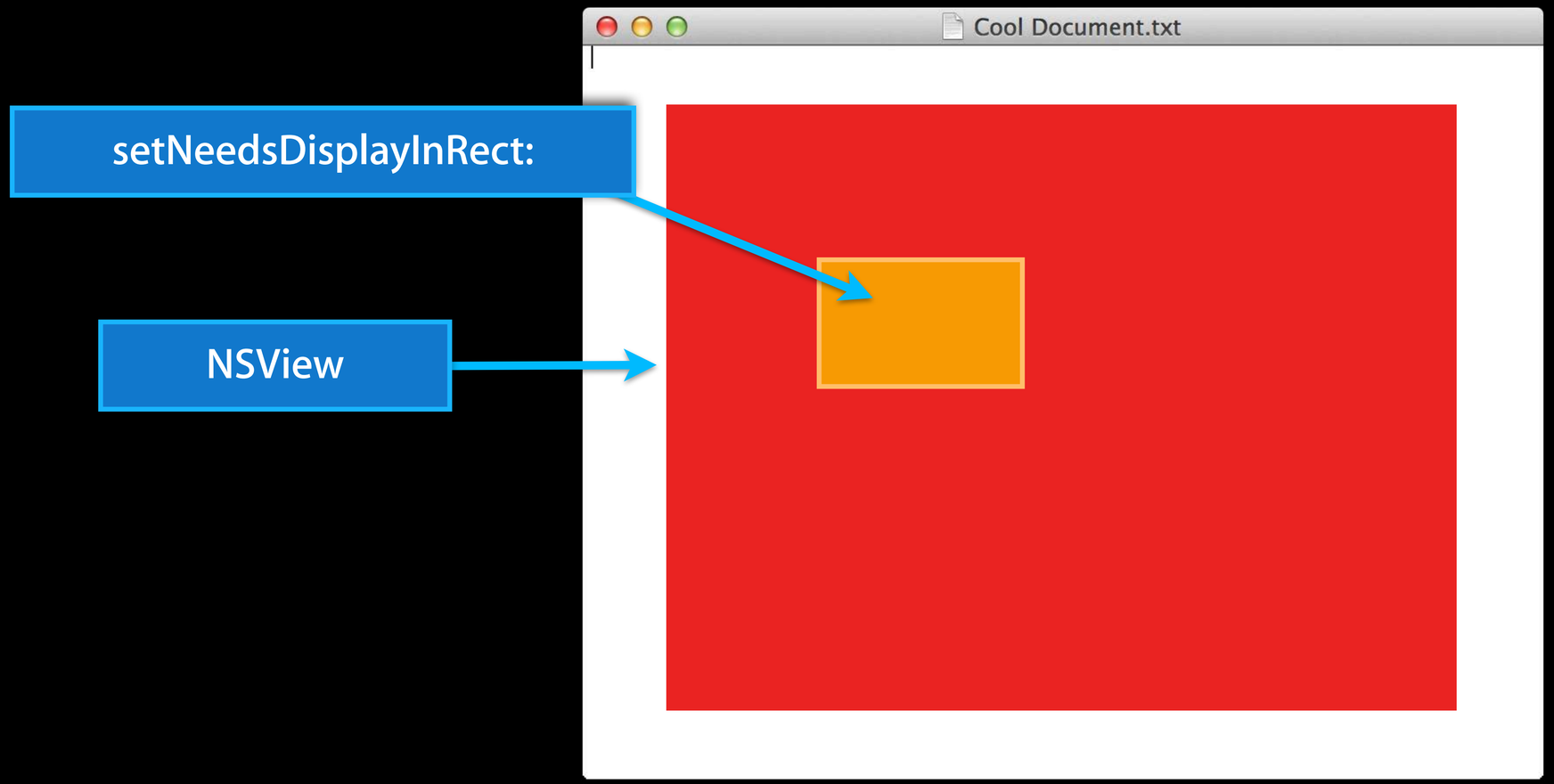
Optimizing AppKit Drawing

Optimize -drawRect:



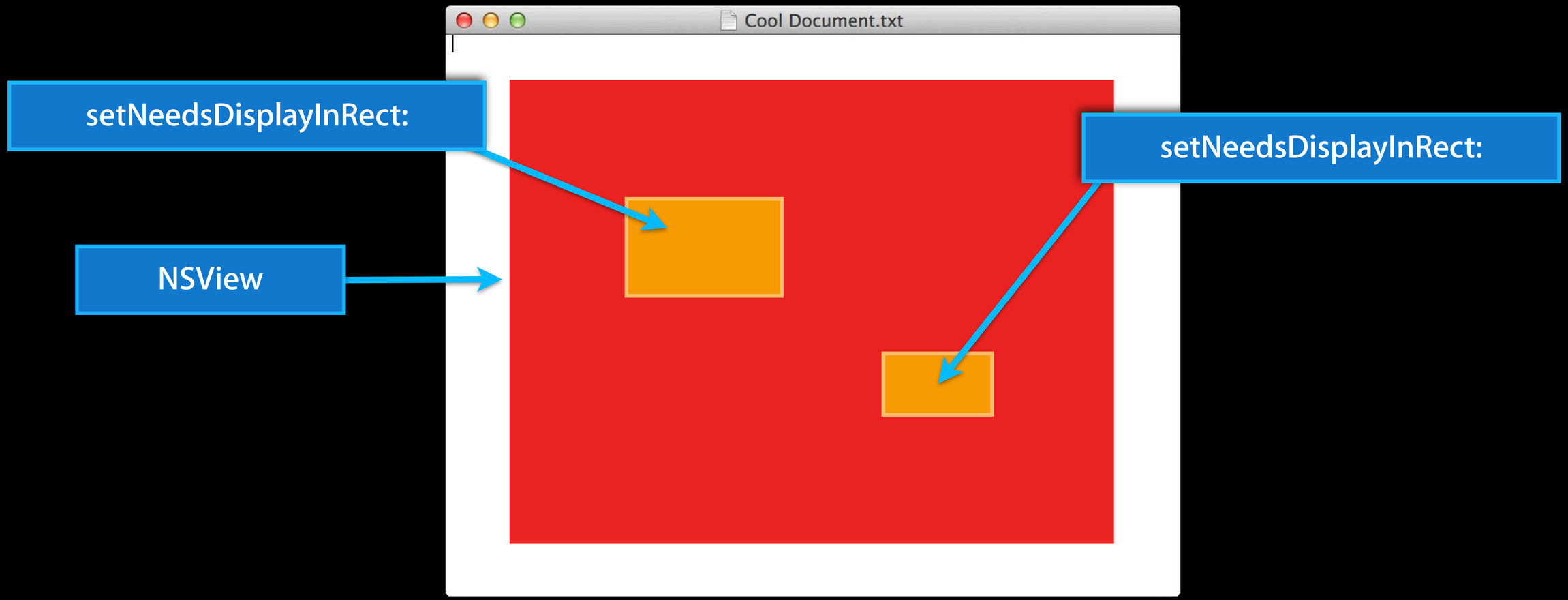
Optimizing AppKit Drawing

Optimize -drawRect:



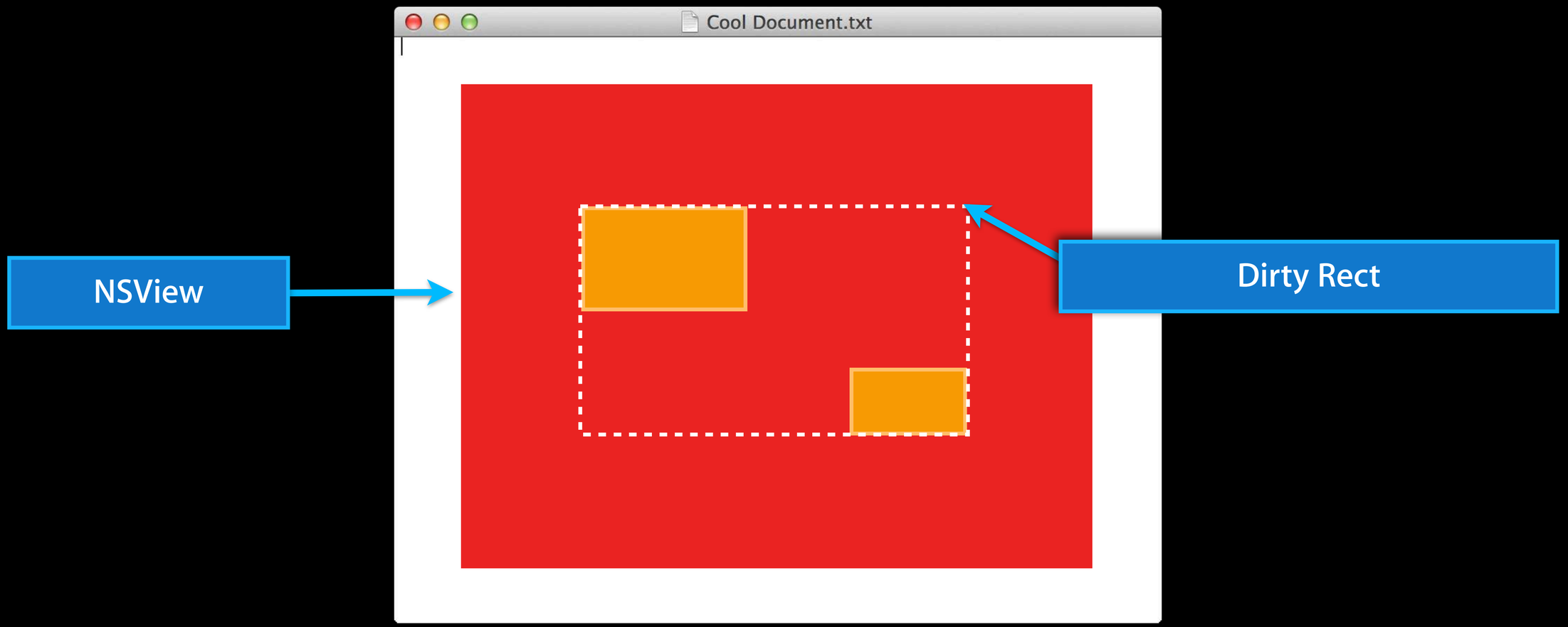
Optimizing AppKit Drawing

Optimize -drawRect:



Optimizing AppKit Drawing

Optimize -drawRect:



Optimizing AppKit Drawing

Optimize -drawRect:

- Utilize –[NSView getRectsBeingDrawn:count:]

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

  const NSRect *rectsBeingDrawn = NULL;
  NSInteger rectsBeingDrawnCount = 0;
  [self getRectsBeingDrawn:&rectsBeingDrawn count:&rectsBeingDrawnCount];
  [NSColor.redColor set]; // Set invariants outside of a loop
  for (NSInteger i = 0; i < rectsBeingDrawnCount; i++) {
    NSRectFill(rectsBeingDrawn[i]);
  }

}
```

Optimizing AppKit Drawing

Optimize -drawRect:

- Or use -needsToDrawRect:

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

    NSRect redRect = NSMakeRect(...);
    if ([self needsToDrawRect:redRect]) {
        [NSColor.redColor set];
        NSRectFill(redRect);
    }

}
```

Optimizing AppKit Drawing

Performant operations

- Only do drawing in -drawRect:
  - No network calls
  - No image allocation or loading
  - No file access
  - No layout (adding/removing subviews)

Optimizing AppKit Drawing

Performant operations

- Only do drawing in -drawRect:
  - No network calls
  - No image allocation or loading
  - No file access
  - No layout (adding/removing subviews)
- Hiding views may be faster than adding/removing them
  - Utilize setHidden: when necessary
  - Exceptions: Layer-backed views

Optimizing AppKit Drawing

Cache images loaded with -imageNamed:

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

    if (_myImage == nil) {
      _myImage = [[NSImage imageNamed:@"MyImage"] retain];
    }
    [_myImage drawInRect:self.imageRect];

}
```

Optimizing AppKit Drawing

Avoid image allocation when drawing

- Use NSOperationQueue to asynchronously load images

```
[MyOperationQueue addOperationWithBlock:^(void) {
   NSImage *image = [[NSImage alloc] initWithContentsOfURL:url];
   // Access the CGImage to pre-warm it and fault it in
   [image CGImageForProposedRect:... context: hints:];

   // Do the update and redisplay on the main thread
   [[NSOperationQueue mainQueue] addOperationWithBlock:^(void) {
      myView.image = image;
      [myView setNeedsDisplayInRect:myView.imageRect];
   }];

}];
```

Optimizing AppKit Drawing

Avoid image allocation when drawing

- Use NSOperationQueue to asynchronously load images

```
[MyOperationQueue addOperationWithBlock:^(void) {
   NSImage *image = [[NSImage alloc] initWithContentsOfURL:url];
   // Access the CGImage to pre-warm it and fault it in
   [image CGImageForProposedRect:... context: hints:];

   // Do the update and redisplay on the main thread
   [[NSOperationQueue mainQueue] addOperationWithBlock:^(void) {
      my
      [myView setNeedsDisplayInRect:myView.imageRect];
   }];

}];
```

Expensive work done on background thread

Optimizing AppKit Drawing

Avoid image allocation when drawing

- Use NSOperationQueue to asynchronously load images

```
[MyOperationQueue addOperationWithBlock:^(void) {
    NSImage *image = [[NSImage
    // Access the CGImage to p
    [image CGImageForProposedRect:... context: hints:];

    // Do the update and redisplay on the main thread
    [[NSOperationQueue mainQueue] addOperationWithBlock:^(void) {
        myView.image = image;
        [myView setNeedsDisplayInRect:myView.imageRect];
    }];

}];
```

Dispatch UI work done to the main thread

Optimizing AppKit Drawing

Don’t do layout or invalidation in drawing

```
- (void)viewWillDraw {
    [self addSubview:newSubview];
    [self setNeedsDisplayInRect:coolRect];
}
```

```
- (void)drawRect:(NSRect)dirtyRect {
    [self addSubview:newSubview];
    [self setNeedsDisplayInRect:coolRect];
    [NSColor.redColor set];
    NSRectFill(coolRect);
}
```

Optimizing AppKit Drawing

Faster compositing

- Say YES to isOpaque when possible
  - Assuming the view is really opaque!

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

Optimizing AppKit Drawing

Override -wantsDefaultClipping

- -wantsDefaultClipping defaults to returning YES
- Return NO if you don’t need clipping
  - Must constrain drawing to the -getRectsBeingDrawn:count:

```
- (BOOL)wantsDefaultClipping {
    return NO;
}
```

Avoid Overriding Certain Methods

Methods AppKit frequently calls

- All of the "gState" methods

```
- (NSInteger)gState;
- (void)allocateGState;
- (oneway void)releaseGState;
- (void)setUpGState;
- (void)renewGState;
```

- Sometimes used to know when some state changes
  - Such as the view global position in the window
- Prefer to use:
  - `NSViewFrameDidChangeNotification`
  - `NSViewBoundsDidChangeNotification`

Layer-Backed View Drawing

Best practices with Core Animation

Effectively Using Layer-Backed NSViews

Utilize Lion and Mountain Lion API

- See “WWDC 2012 Layer-Backed Views”
- layerContentsRedrawPolicy
- updateLayer / wantsUpdateLayer

Redrawing Layer-Backed Views

Lion introduced – `[NSView layerContentsRedrawPolicy]`

- This property tells when AppKit should mark the layer as needing display
  - NSViewLayerContentsRedrawDuringViewResize
  - NSViewLayerContentsRedrawOnSetNeedsDisplay
  - NSViewLayerContentsRedrawBeforeViewResize
  - NSViewLayerContentsRedrawNever

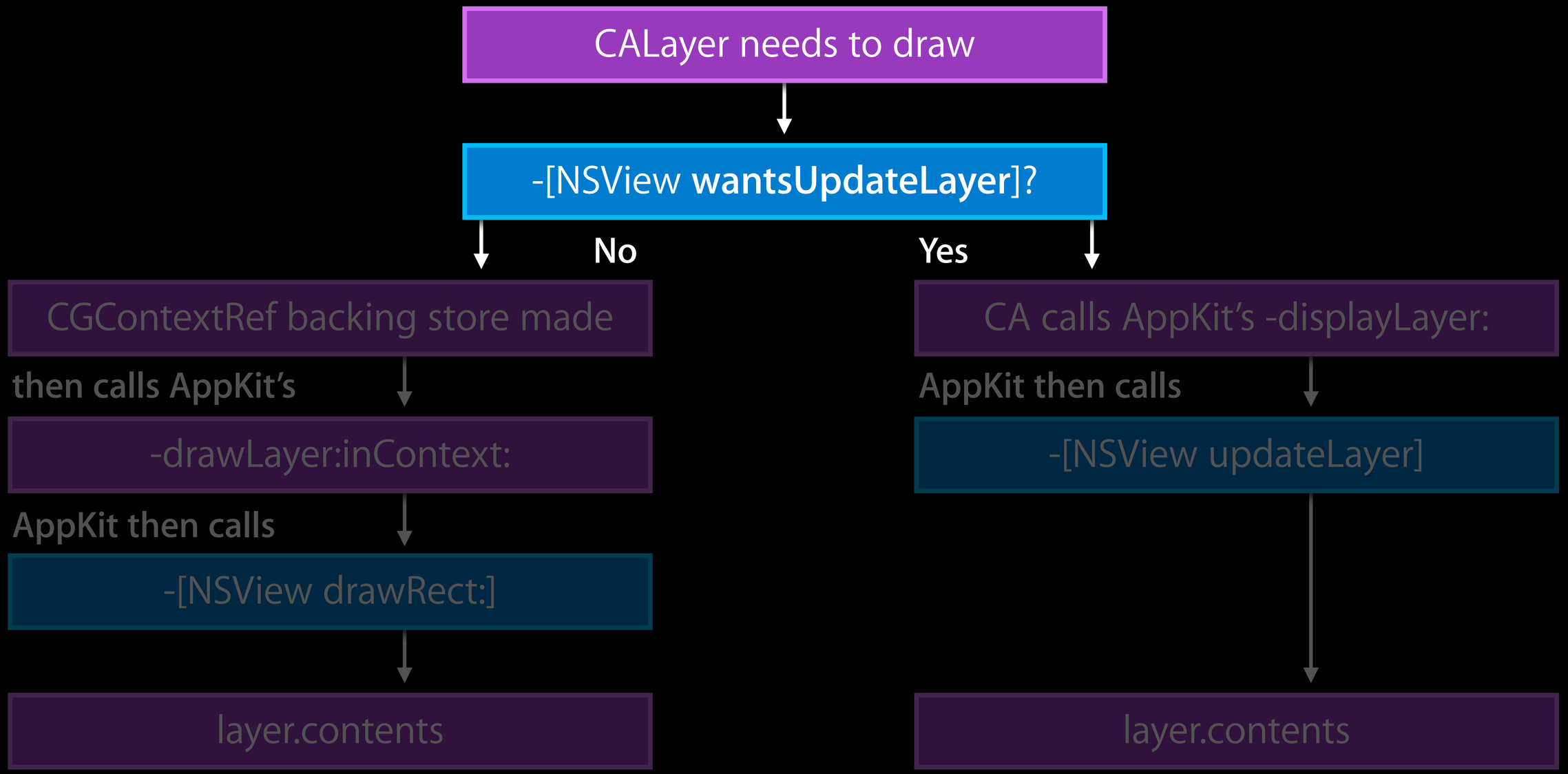
Redrawing Layer-Backed Views

`NSViewLayerContentsRedrawOnSetNeedsDisplay`

- Doing: `[view setNeedsDisplay:YES]`
  - Means “invalidate the layer and lazily redraw”
- AppKit does not call `setNeedsDisplay:` when the frame changes!
- NOT the default value
  - Therefore, you MUST set it!

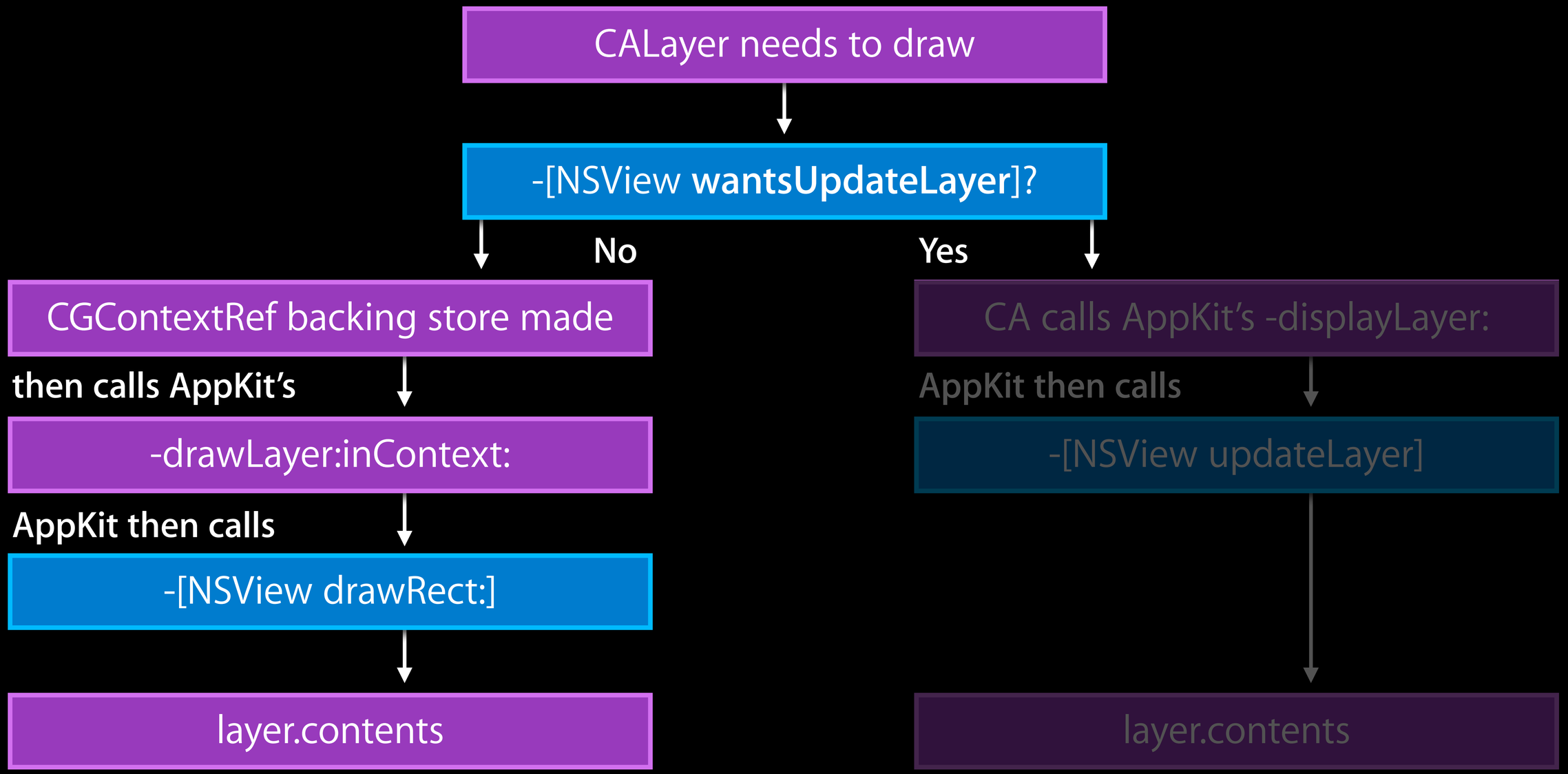
AppKit Layer Drawing/Updating

Since adding -wantsUpdateLayer



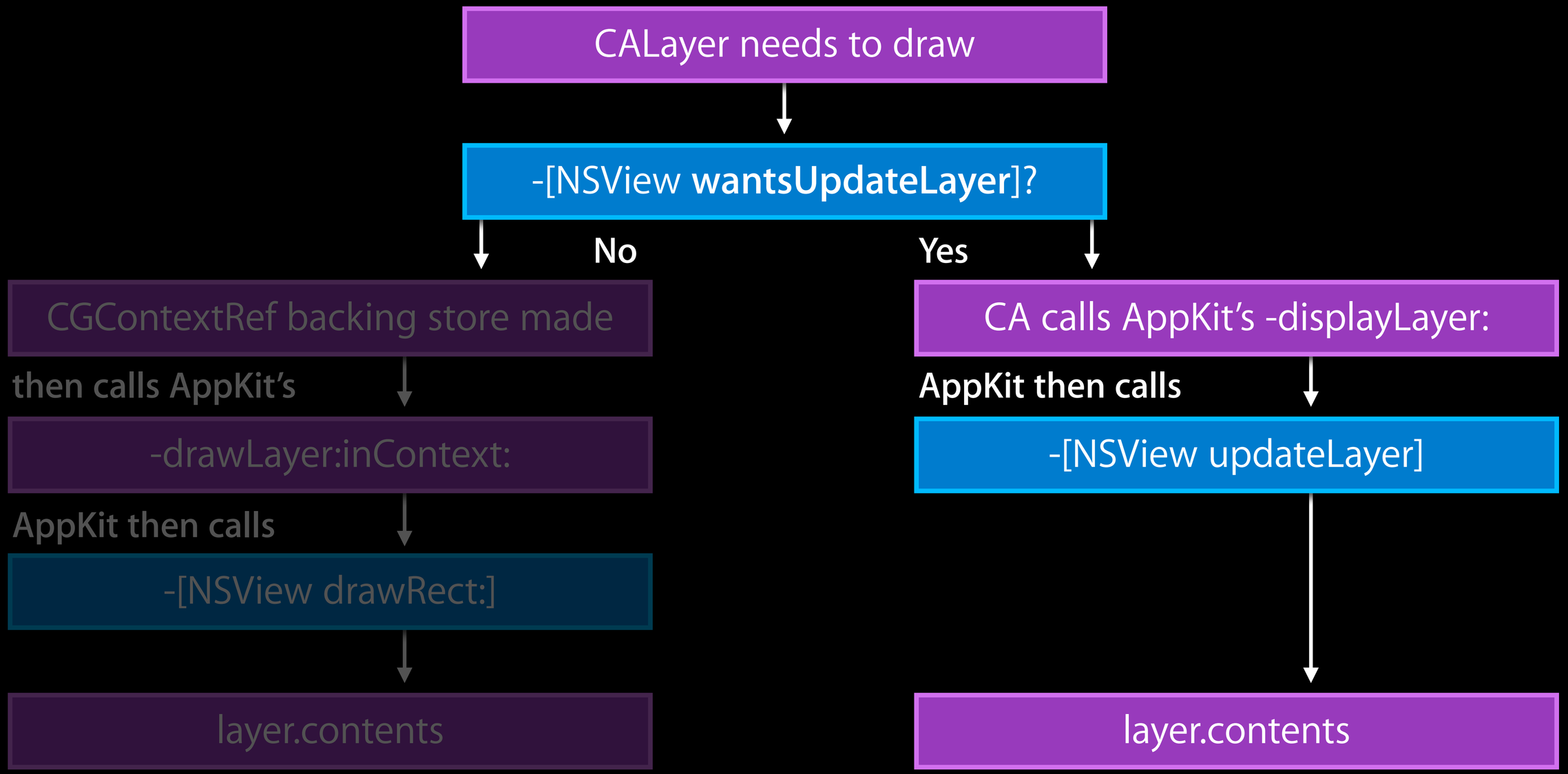
AppKit Layer Drawing/Updating

Since adding -wantsUpdateLayer



AppKit Layer Drawing/Updating

Since adding -wantsUpdateLayer



Improving Layer-Backed Memory Use

Use -wantsUpdateLayer and -updateLayer

```
- (BOOL)wantsUpdateLayer {
    return YES;
}
- (void)updateLayer {
    self.layer.backgroundColor = NSColor.whiteColor.CGColor;
    self.layer.borderColor = NSColor.redColor.CGColor;
}
```

Avoid Expensive Core Animation Properties

- Avoid these properties if possible

```
@property CGFloat cornerRadius;
@property(retain) CALayer *mask;
@property(copy) NSArray *filters;
@property(copy) NSArray *backgroundFilters;
```

Utilize Opaque Views When Possible

- Return YES from `[NSView isOpaque]`

```
layer.opaque = YES; // Implicitly set for you
```

Large Layer Drawing in AppKit

You can still use -drawRect:

Large Layer Drawing in AppKit

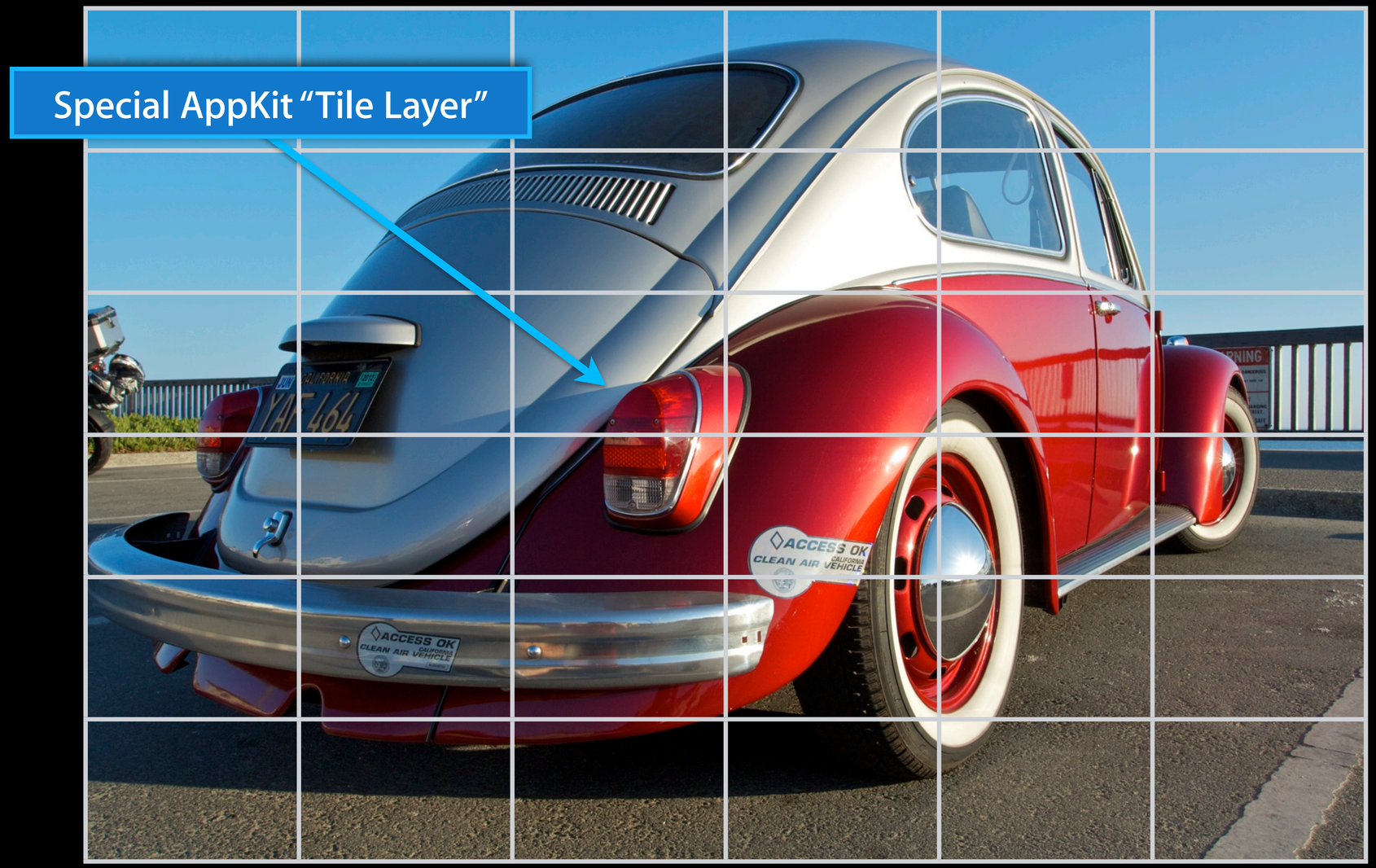
You can still use -drawRect:

Large Layer Drawing in AppKit

You can still use -drawRect:

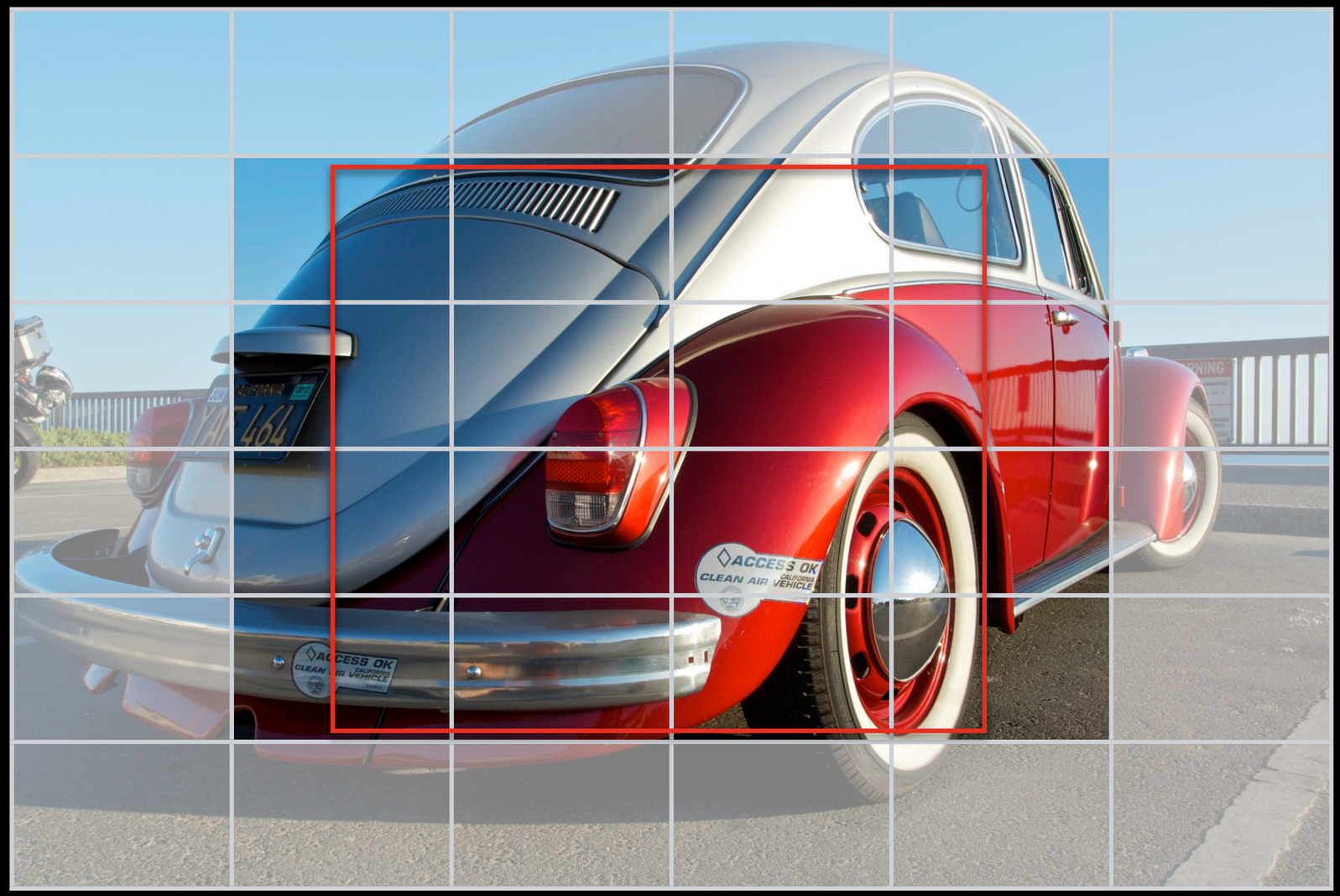
Large Layer Drawing in AppKit

You can still use -drawRect:



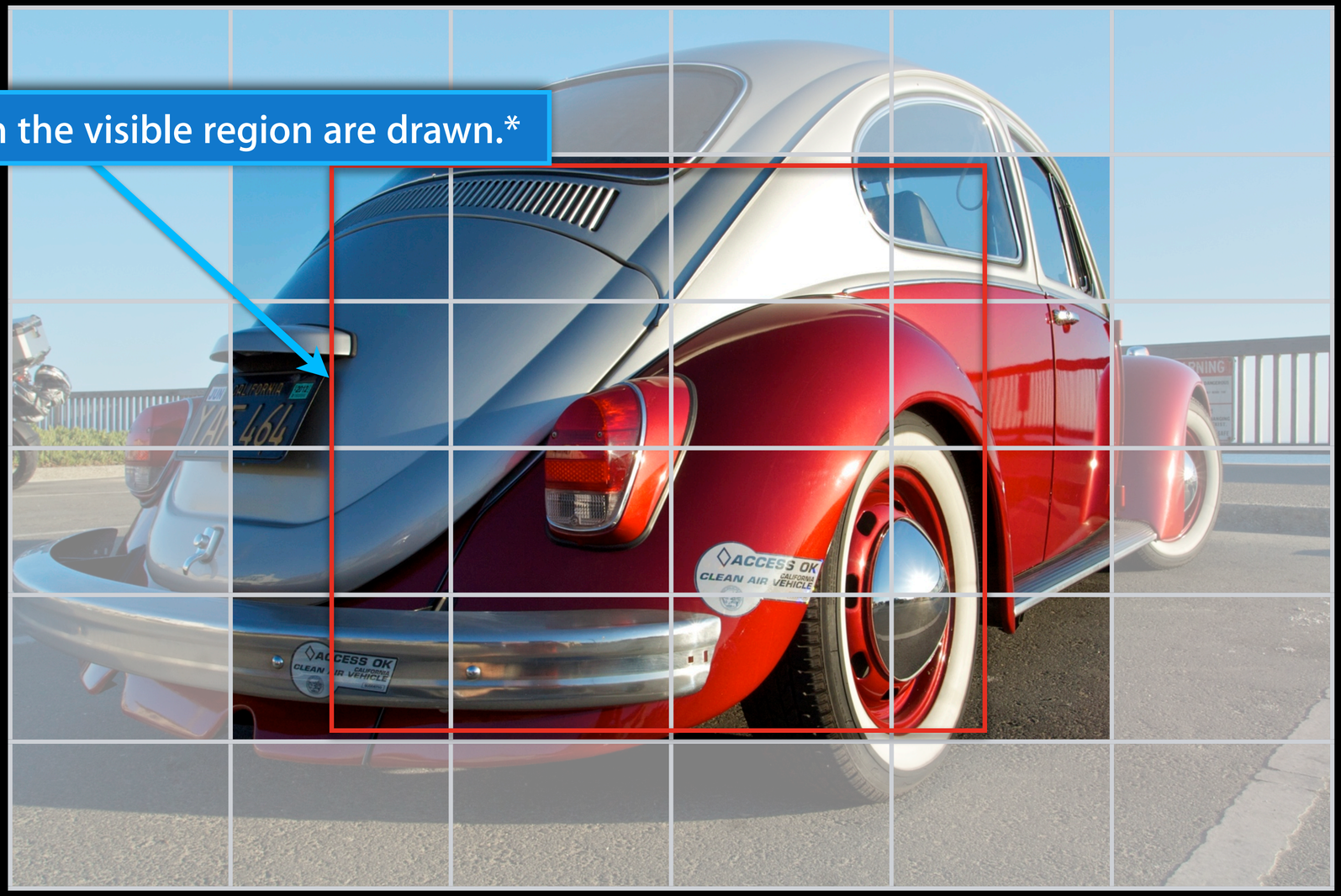
Large Layer Drawing in AppKit

You can still use -drawRect:



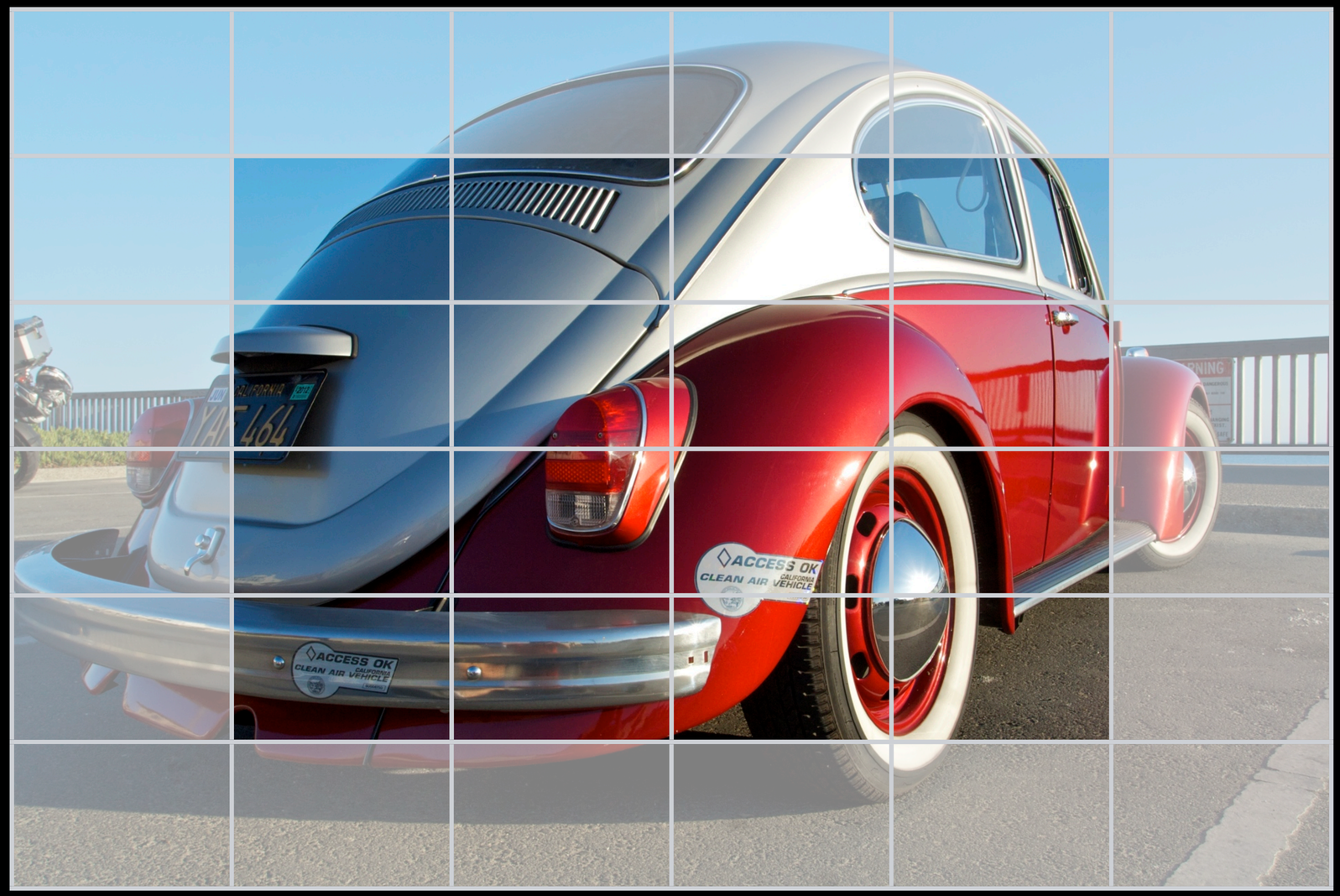
Large Layer Drawing in AppKit

You can still use -drawRect:



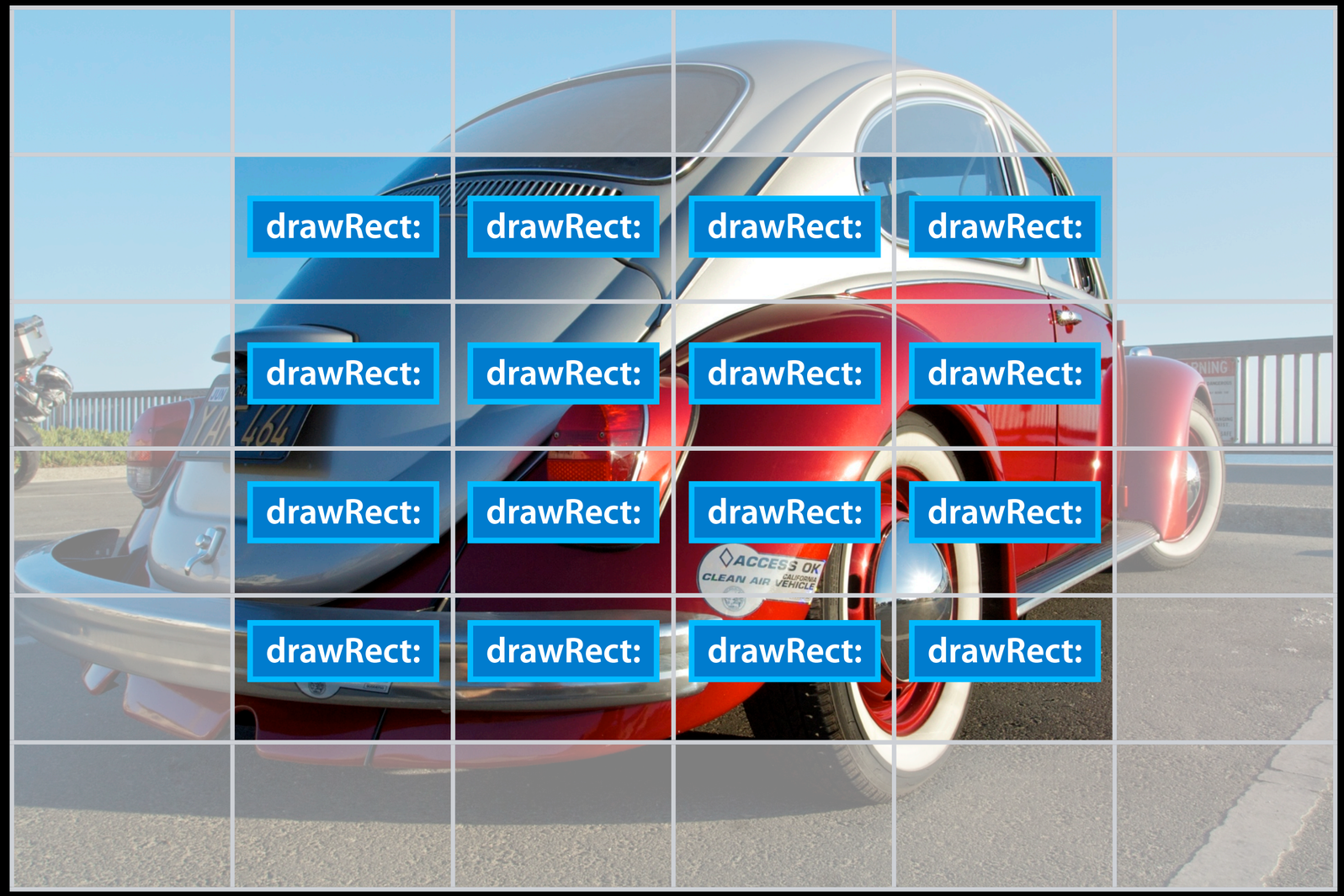
Large Layer Drawing in AppKit

You can still use -drawRect:



Large Layer Drawing in AppKit

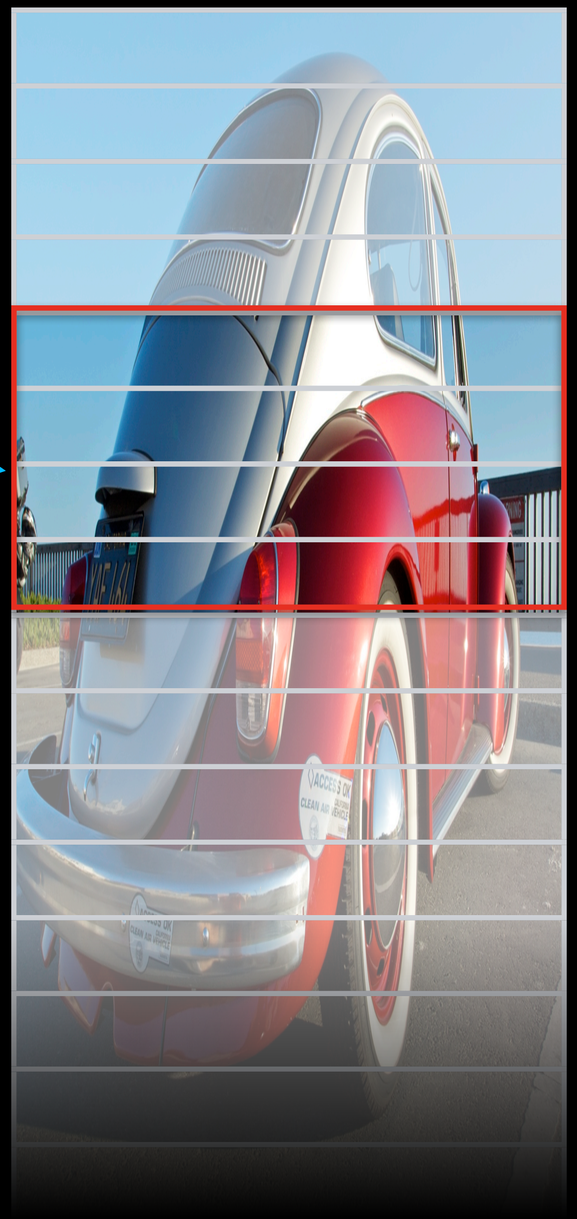
You can still use -drawRect:



Large Layer Drawing in AppKit

Tiles are intelligent sizes

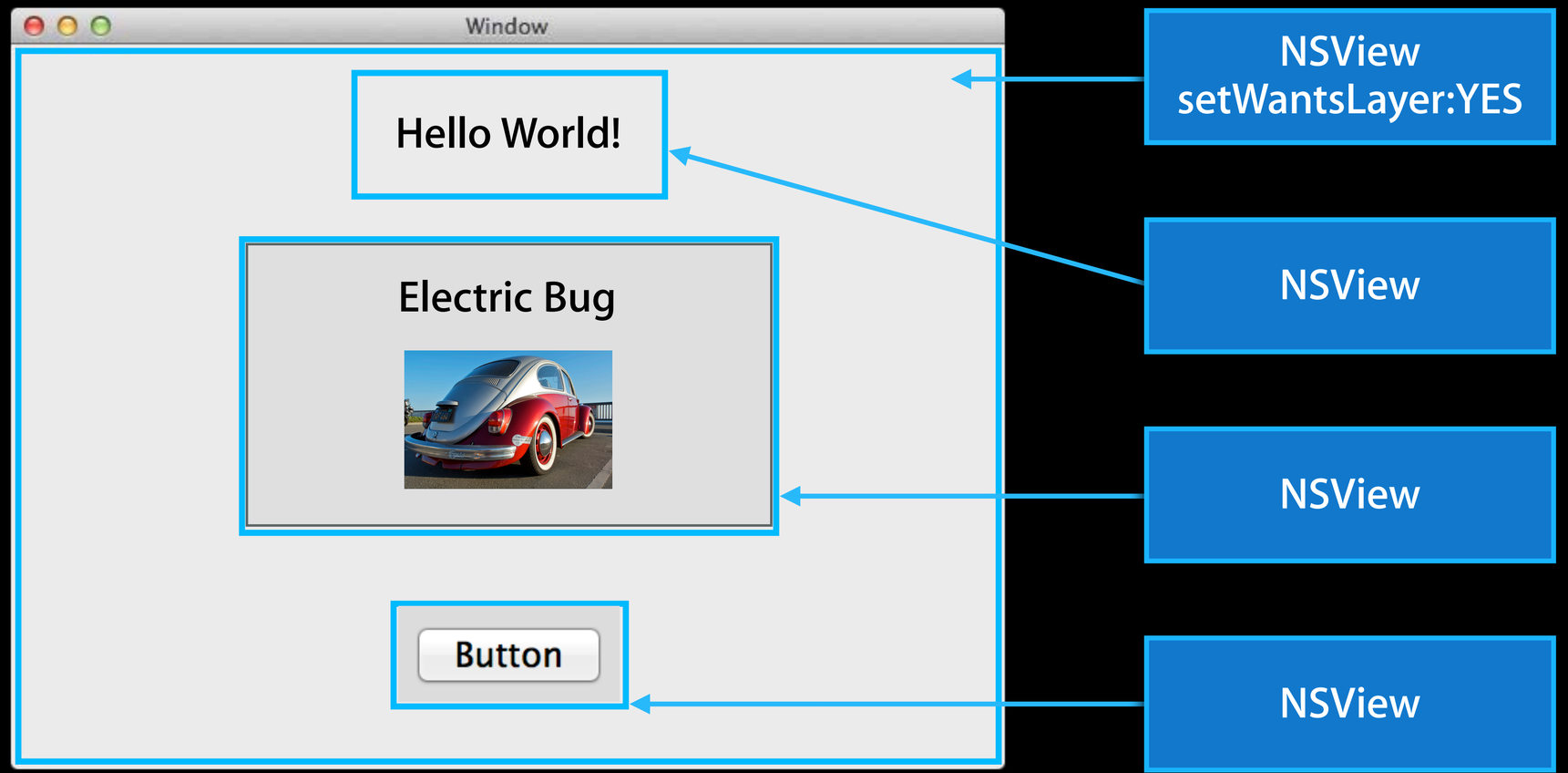
NSScrollView



Reduce Your Layer Count

Typical layer-backed views

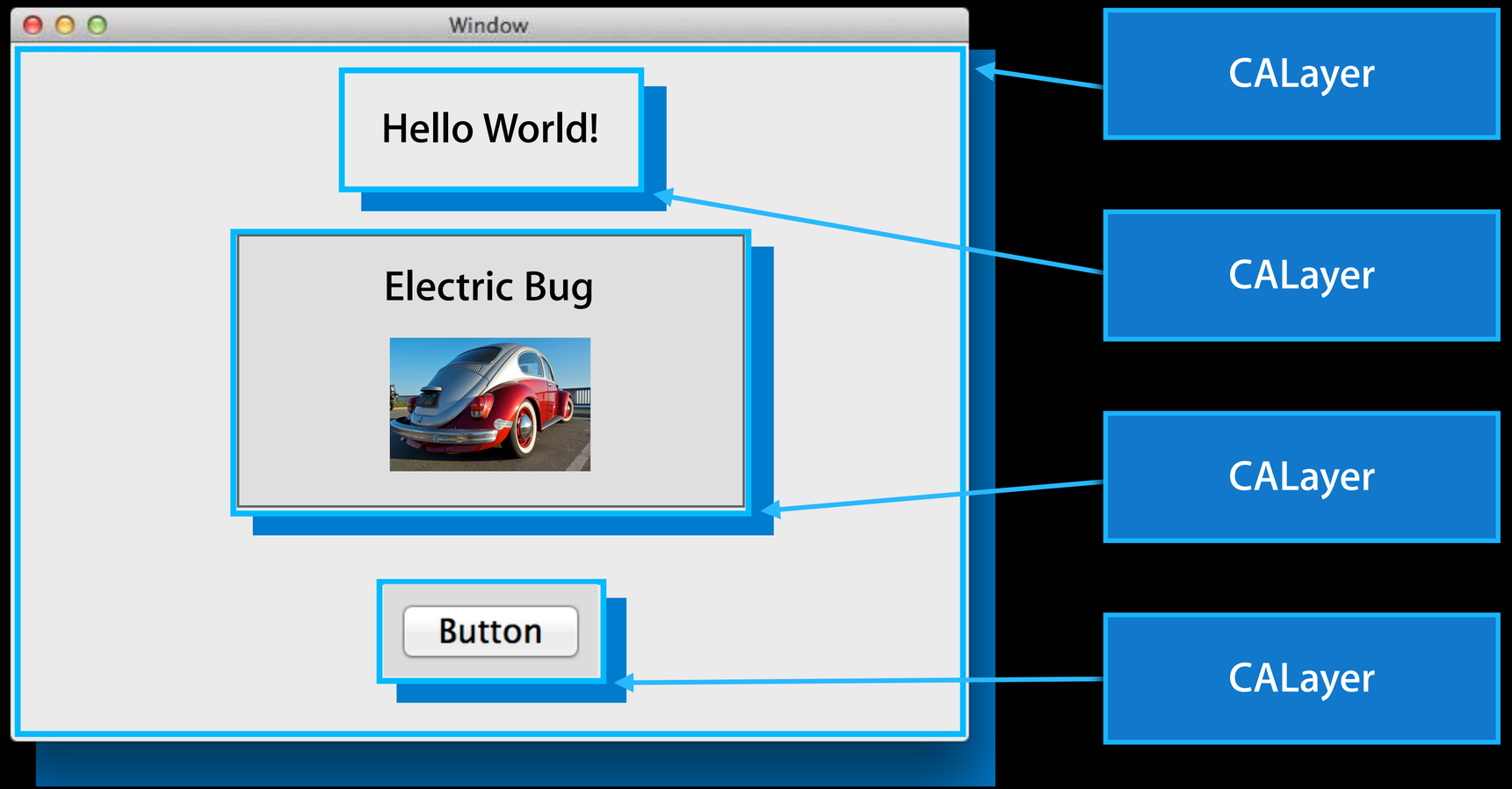
- Layer-backing a parent view implicitly creates layers for children views



Reduce Your Layer Count

Typical layer-backed views

- Layer-backing a parent view implicitly creates layers for children views



Reduce Your Layer Count

Issues with having lots of layers

- Potentially a high memory cost
  - Each subview may have its own backing store (image)
  - Overlapping subviews can waste memory
- Potentially high composition cost
  - Hidden layers still have a composition cost
    - Removing them may be better than hiding
    - One or two is okay, but hiding hundreds is not good

Reduce Your Layer Count

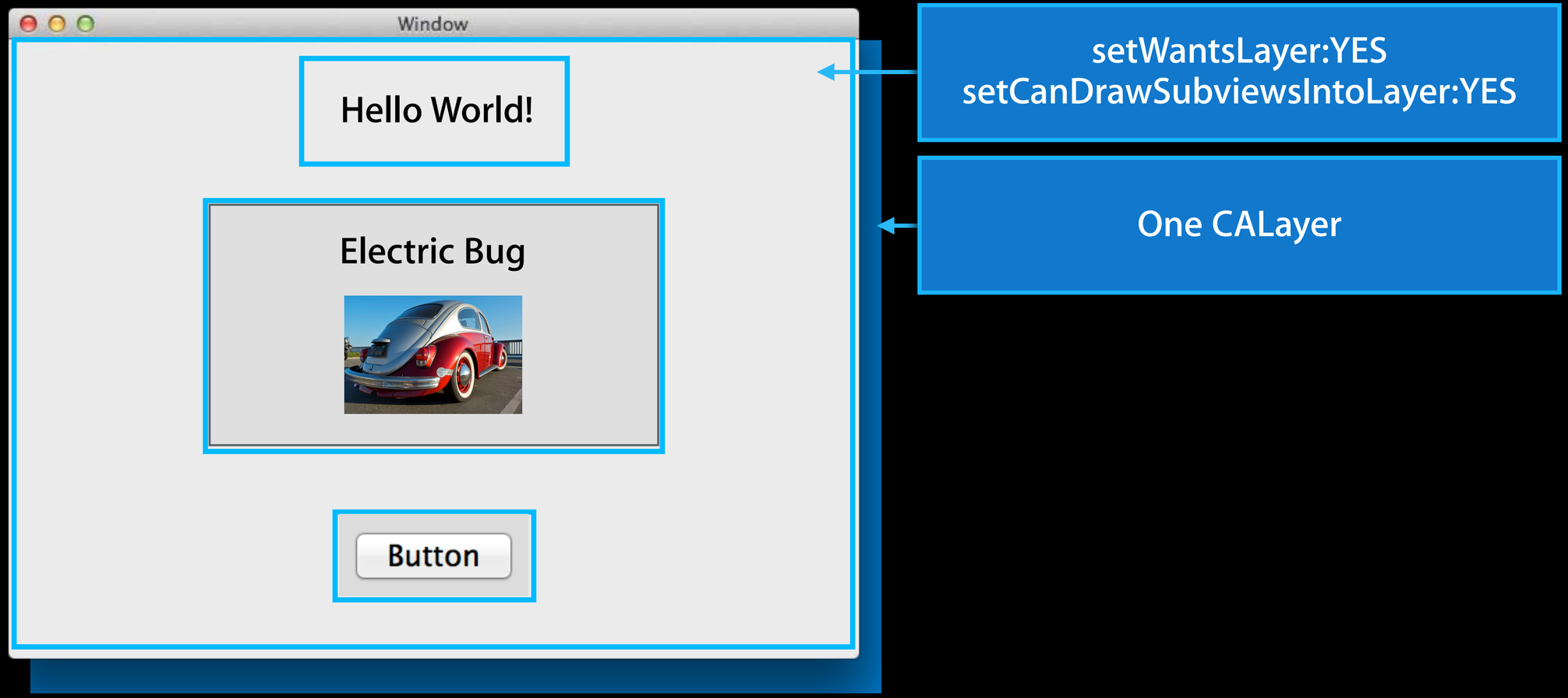
New API: canDrawSubviewsIntoLayer


```
@interface NSView ...

- (void)setCanDrawSubviewsIntoLayer:(BOOL)flag NS_AVAILABLE_MAC(10_9);
- (BOOL)canDrawSubviewsIntoLayer NS_AVAILABLE_MAC(10_9);

@end
```

Reduce Your Layer Count

New API: canDrawSubviewsIntoLayer

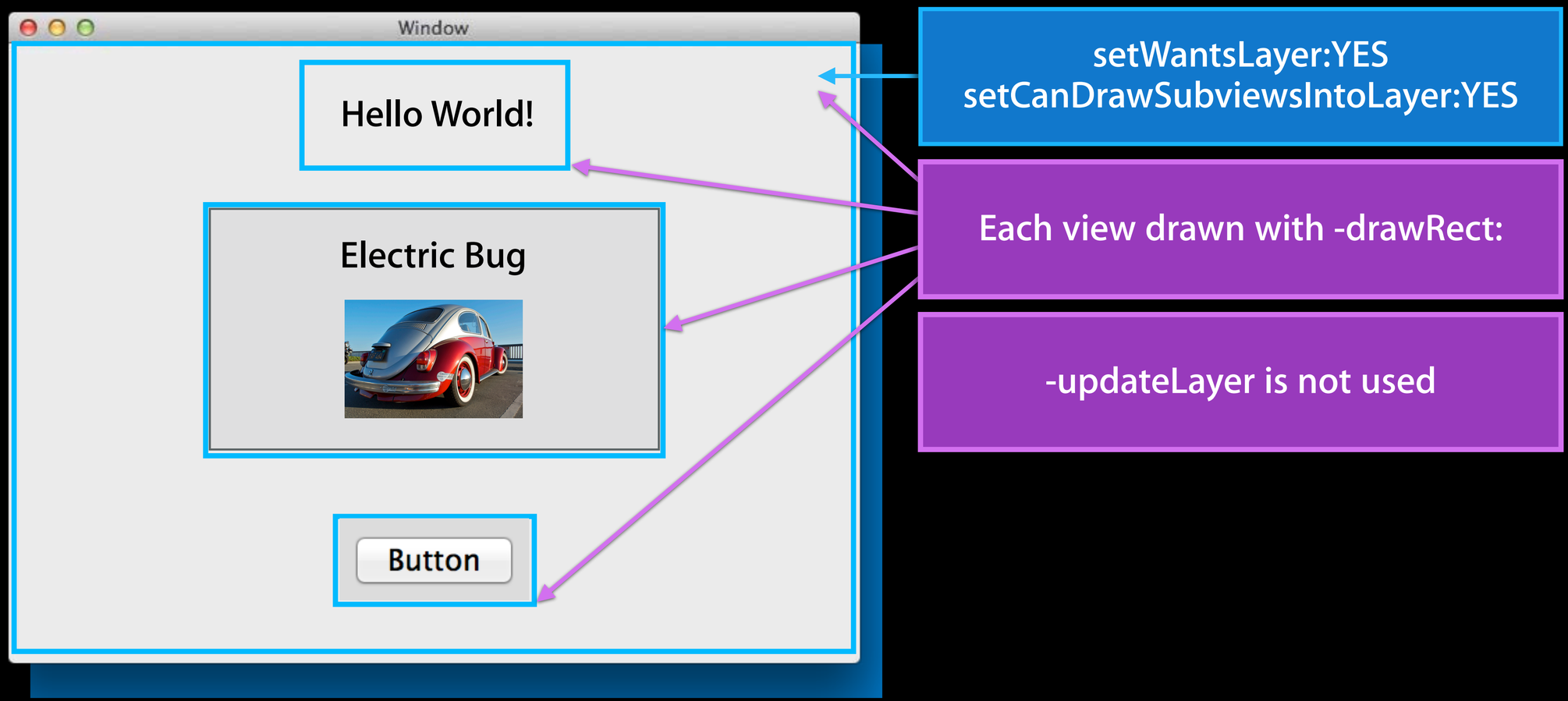
- All children NSViews are drawn into a single CALayer



Reduce Your Layer Count

New API: canDrawSubviewsIntoLayer

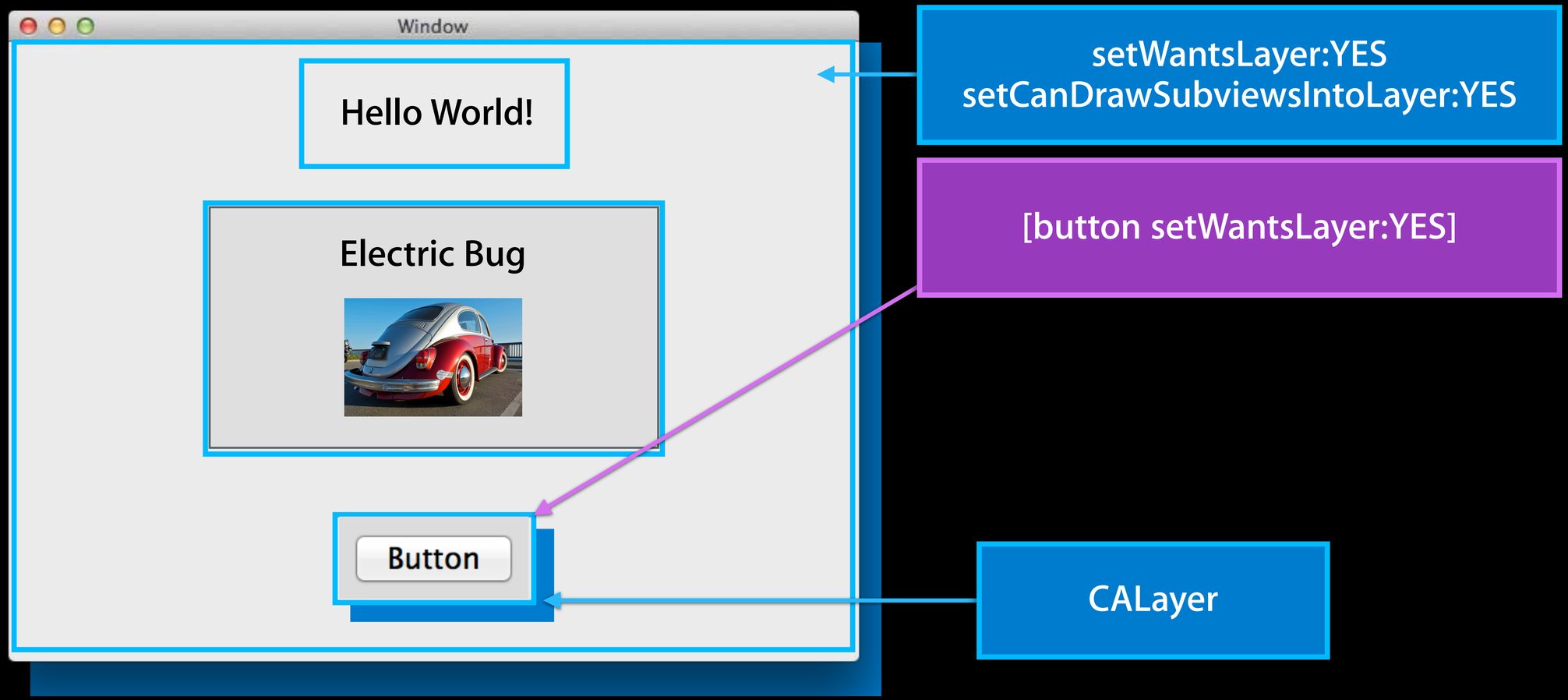
- -drawRect: is utilized for every view!



Reduce Your Layer Count

New API: canDrawSubviewsIntoLayer

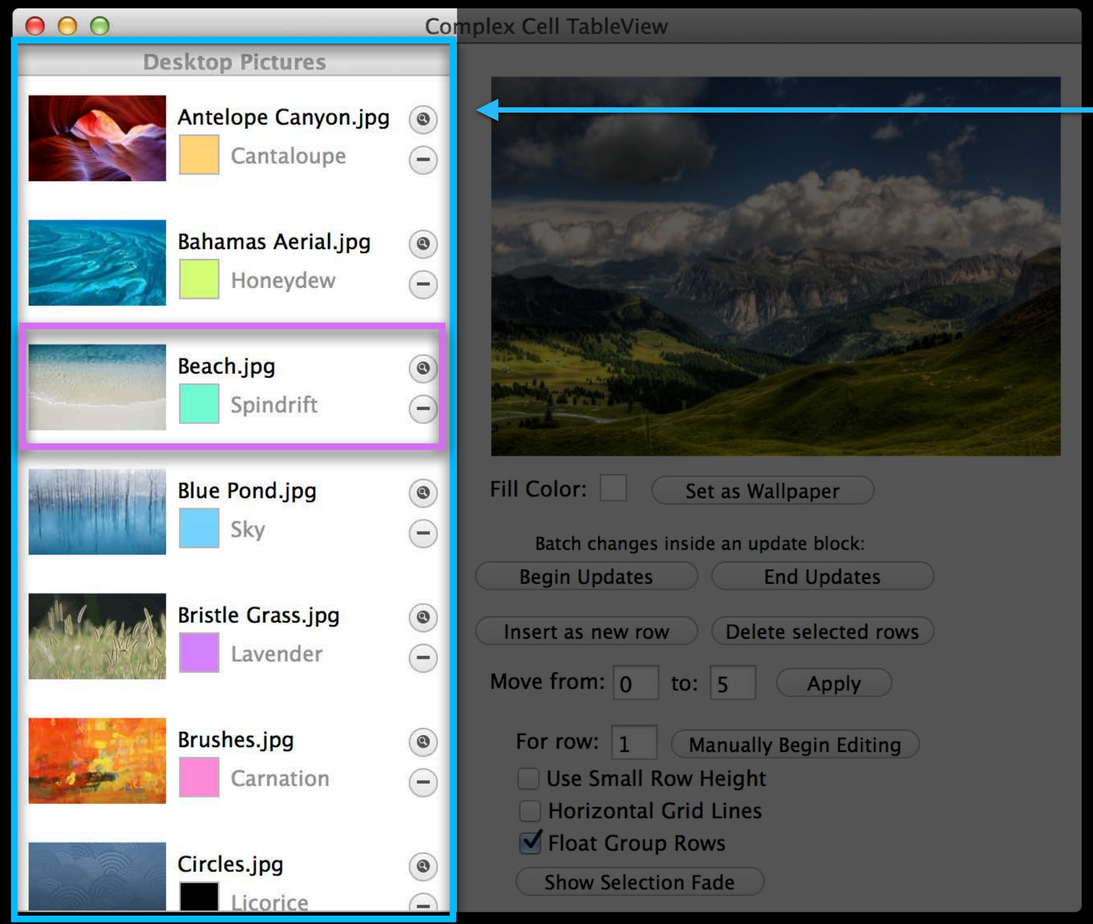
- Individual subviews can opt-in to having their own layer



Reduce Your Layer Count

Useful in NSTableView

- Reduces all row subviews into a single layer
- Row animations will be done with Core Animation

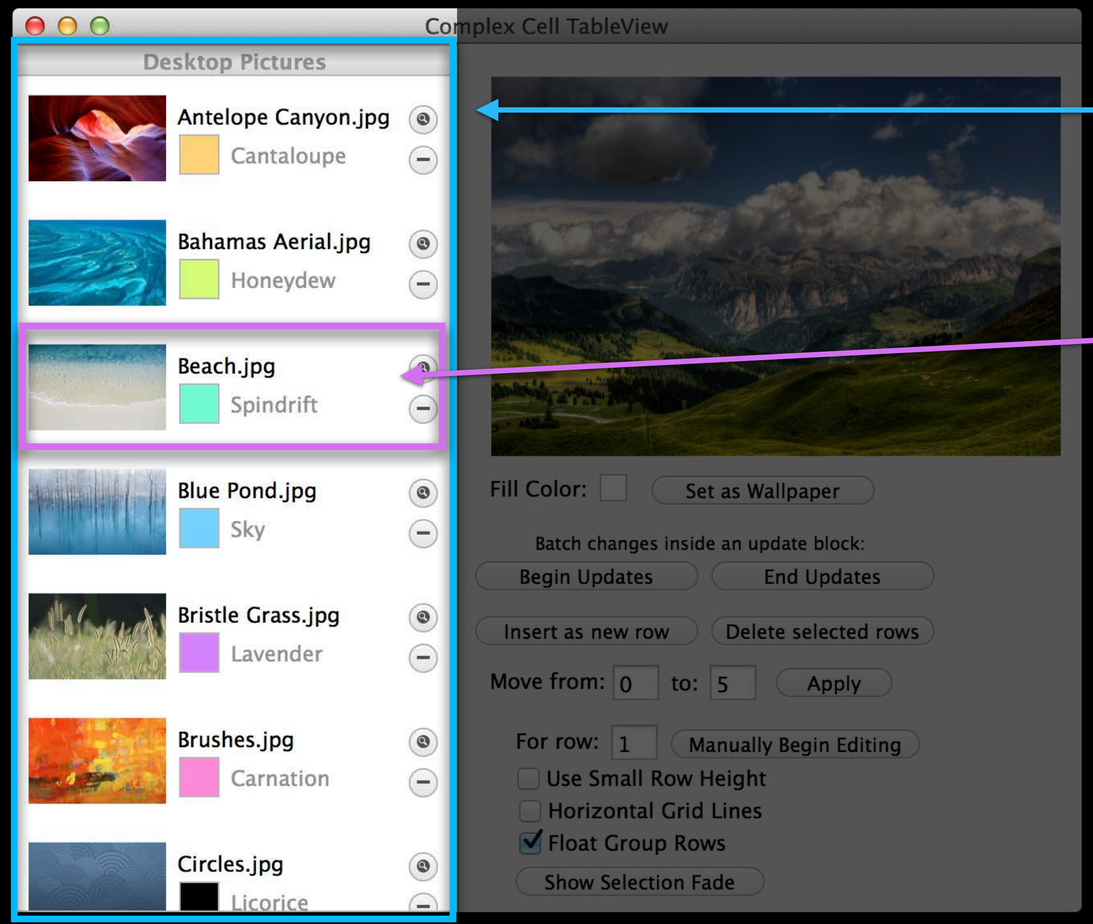


setWantsLayer:YES on the NSScrollView

Reduce Your Layer Count

Useful in NSTableView

- Reduces all row subviews into a single layer
- Row animations will be done with Core Animation



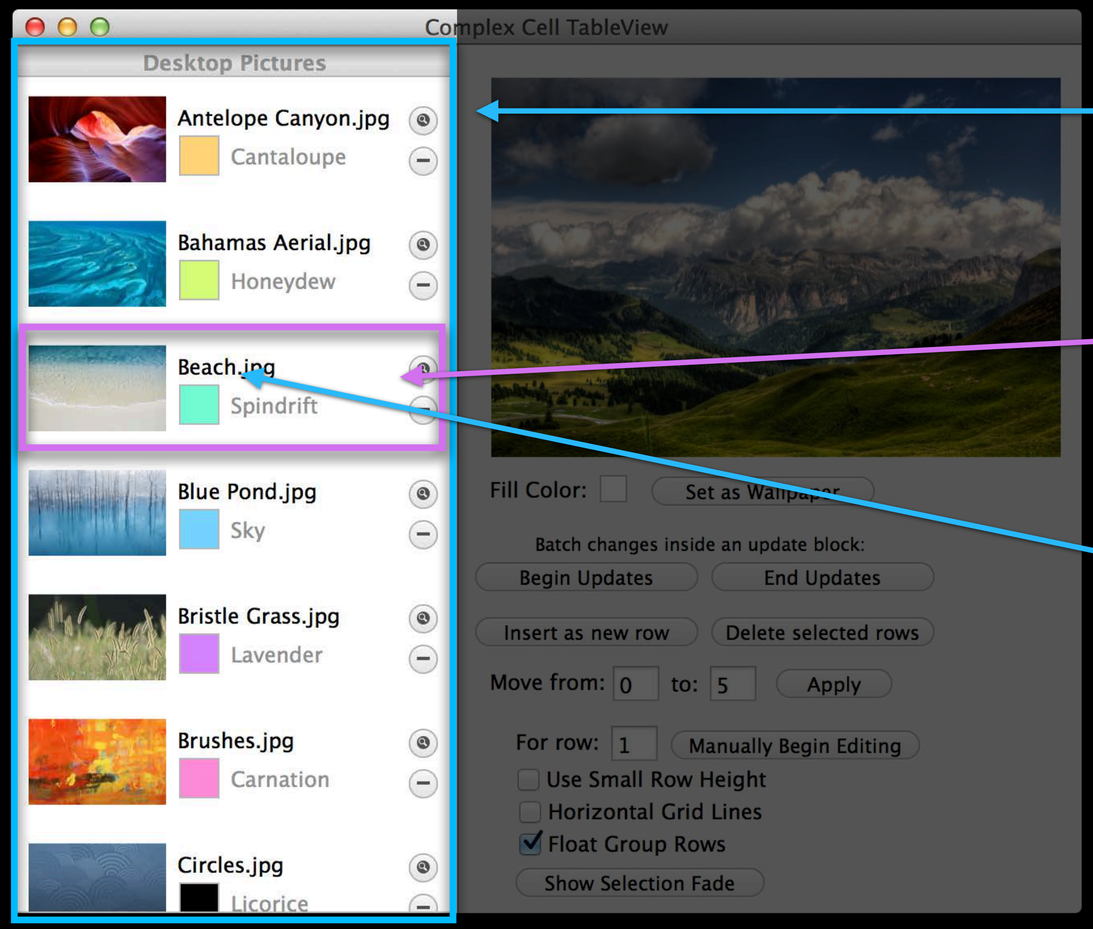
setWantsLayer:YES on the NSScrollView

setCanDrawSubviewsIntoLayer:YES on each NSTableRowView

Reduce Your Layer Count

Useful in NSTableView

- Reduces all row subviews into a single layer
- Row animations will be done with Core Animation



setWantsLayer:YES on the NSScrollView

setCanDrawSubviewsIntoLayer:YES on each NSTableRowView

For text to have font smoothing, the text must be drawn into an opaque area

Responsive Scrolling

Raleigh Ledet

Demo

Responsive Scrolling

Goals

- Fluid
- Smooth
- Non-stuttering

Responsive Scrolling

Overview


Bryant Lake Gardens

Bryant Lakes Gardens Executive Summary

Objectives

Build an environment-friendly housing community that is energy efficient, self-sustaining, and cost-effective. We will use modular designs to accelerate implementation and provide significant savings. Bryant Lakes Gardens will become a national showcase for housing communities that help the environment while reducing development costs and providing long-term savings.

The housing community should be aesthetically pleasing and modern, completely wired, and comfortable in every way. Living in an eco-friendly home should be a bonus to prospective residents without sacrificing comfort or appearance.

The design should make the community blend in with the surrounding environment.

Goals

Make full use of renewable energy sources.

Harness solar and wind power for electric power and hot water.

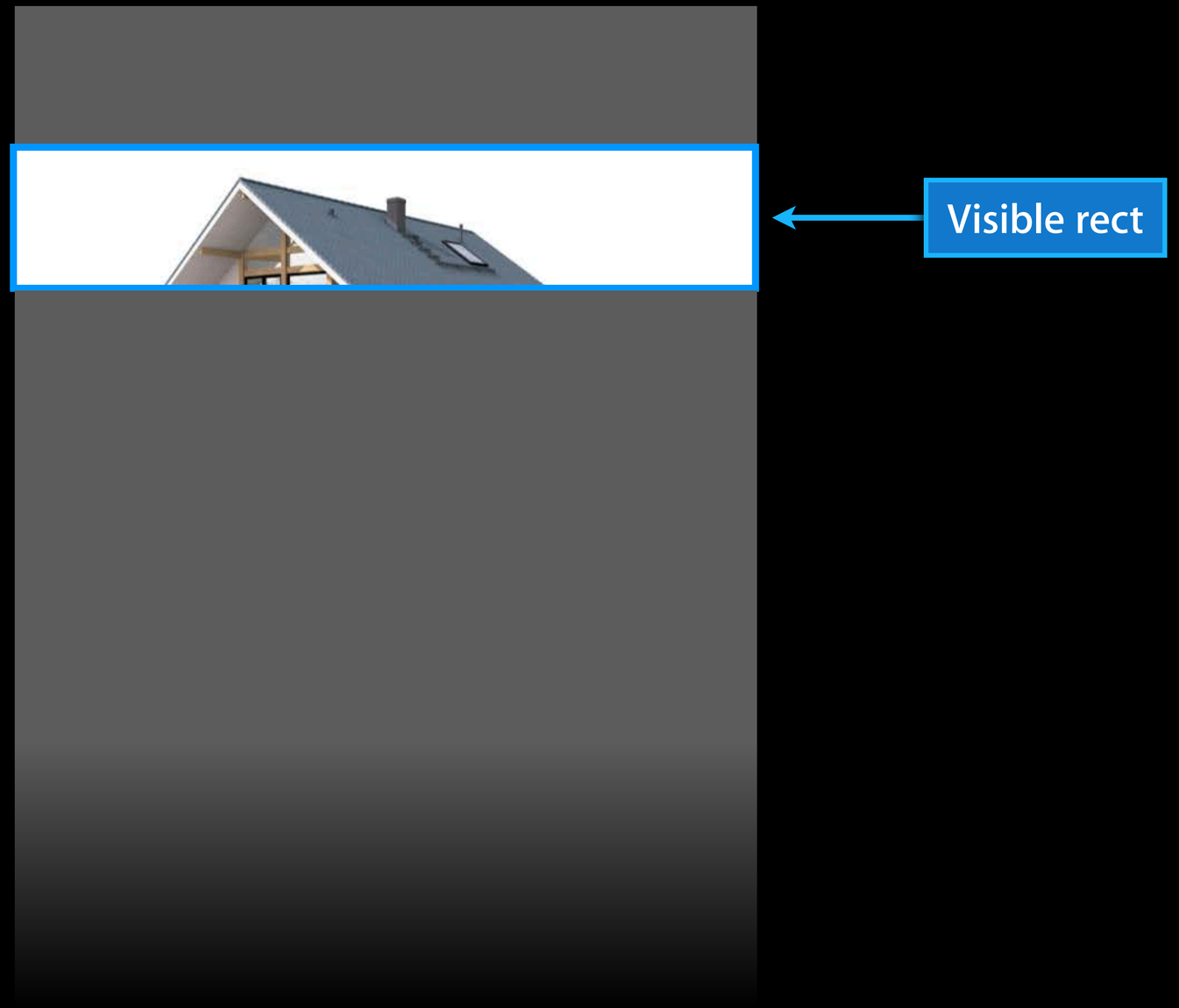
Capture and store rain water for drinking and cooking.

Design to save money via energy conservation and setbacks.

Generate surplus energy to sell back to the electric company.

Responsive Scrolling

Overview



Responsive Scrolling

Overview

Overdraw

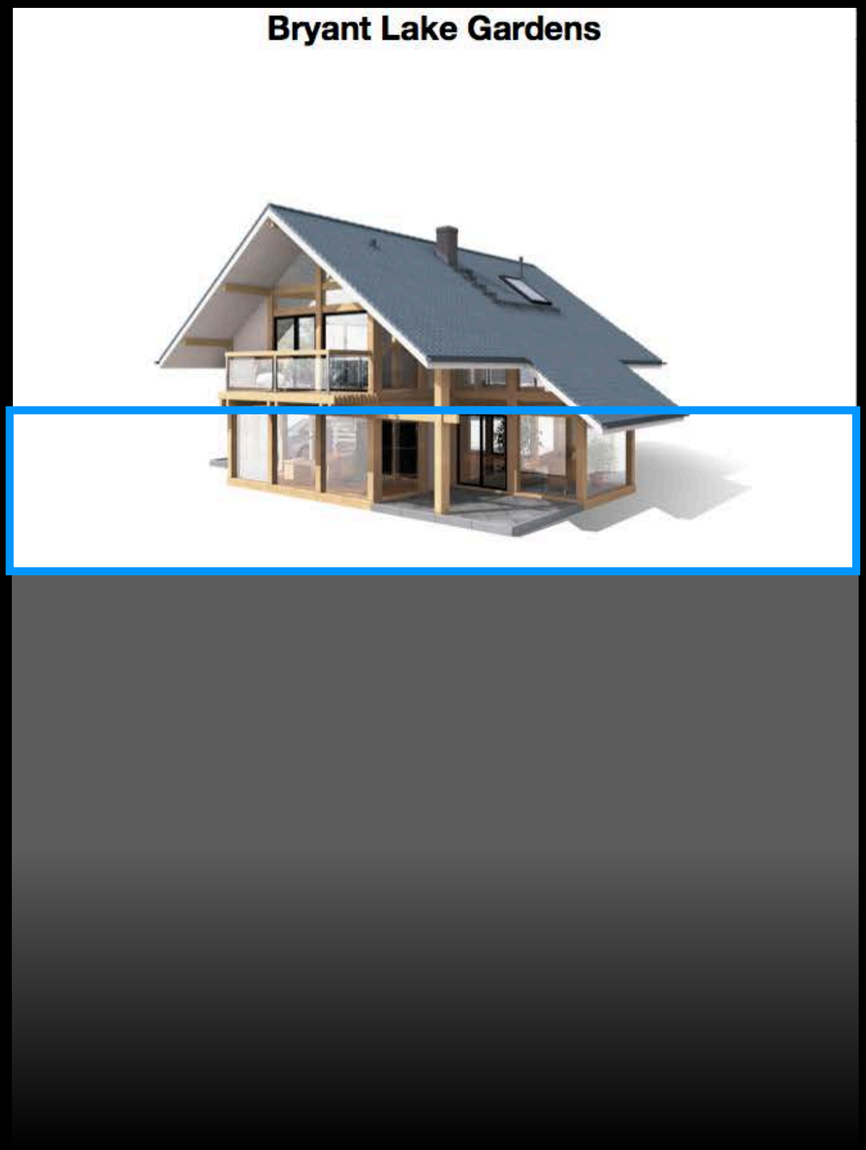
Visible rect

Overdraw

Responsive Scrolling

Overview

Overdraw



Visible rect

Overdraw

Responsive Scrolling

Overview

- Overdraw
- Event Model
- API
- Adoption

Overdraw

Responsive scrolling

Overdraw

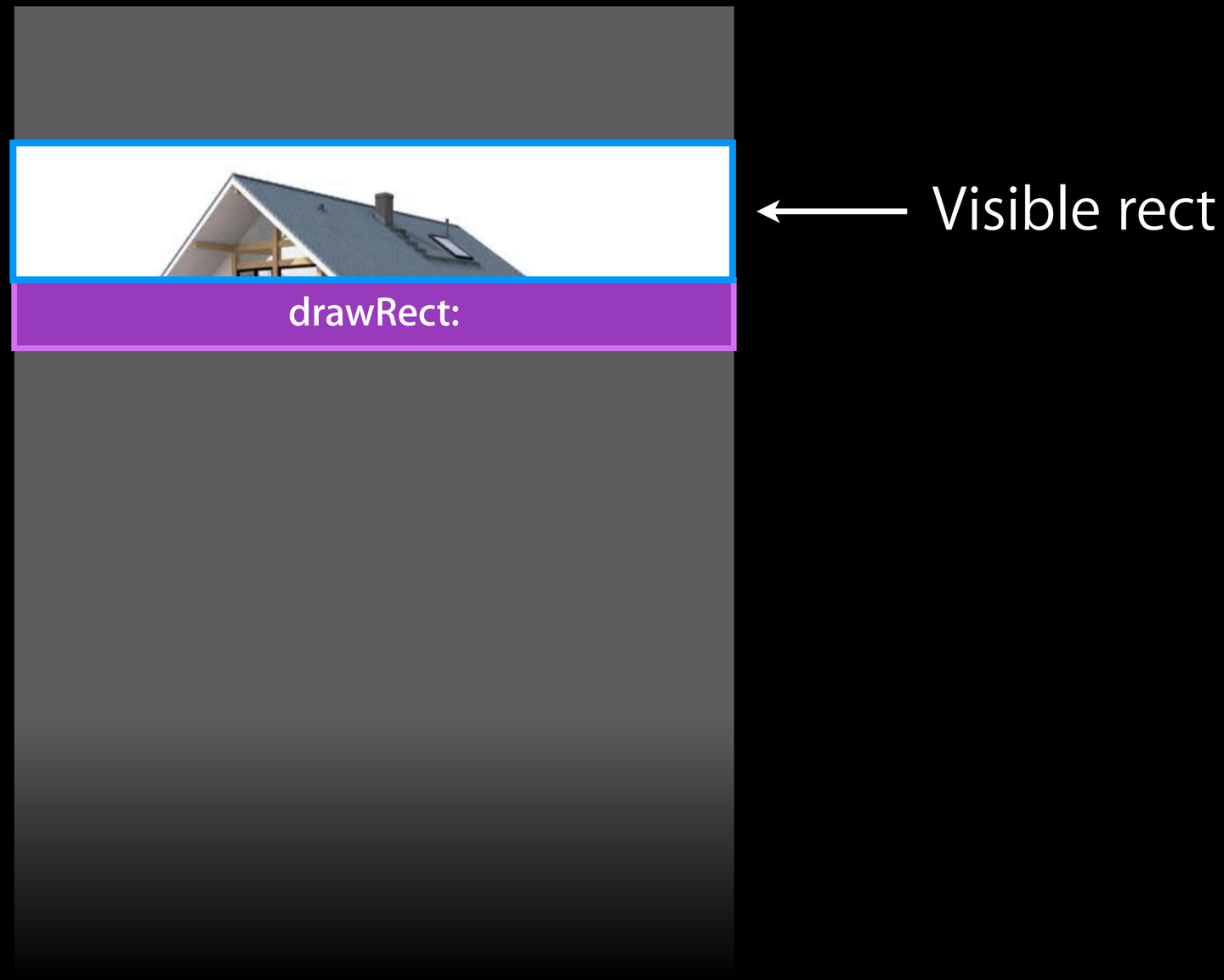
- Main thread driven
- -drawRect: called with non-visible rects

Overdraw

Idle prefetch

Overdraw

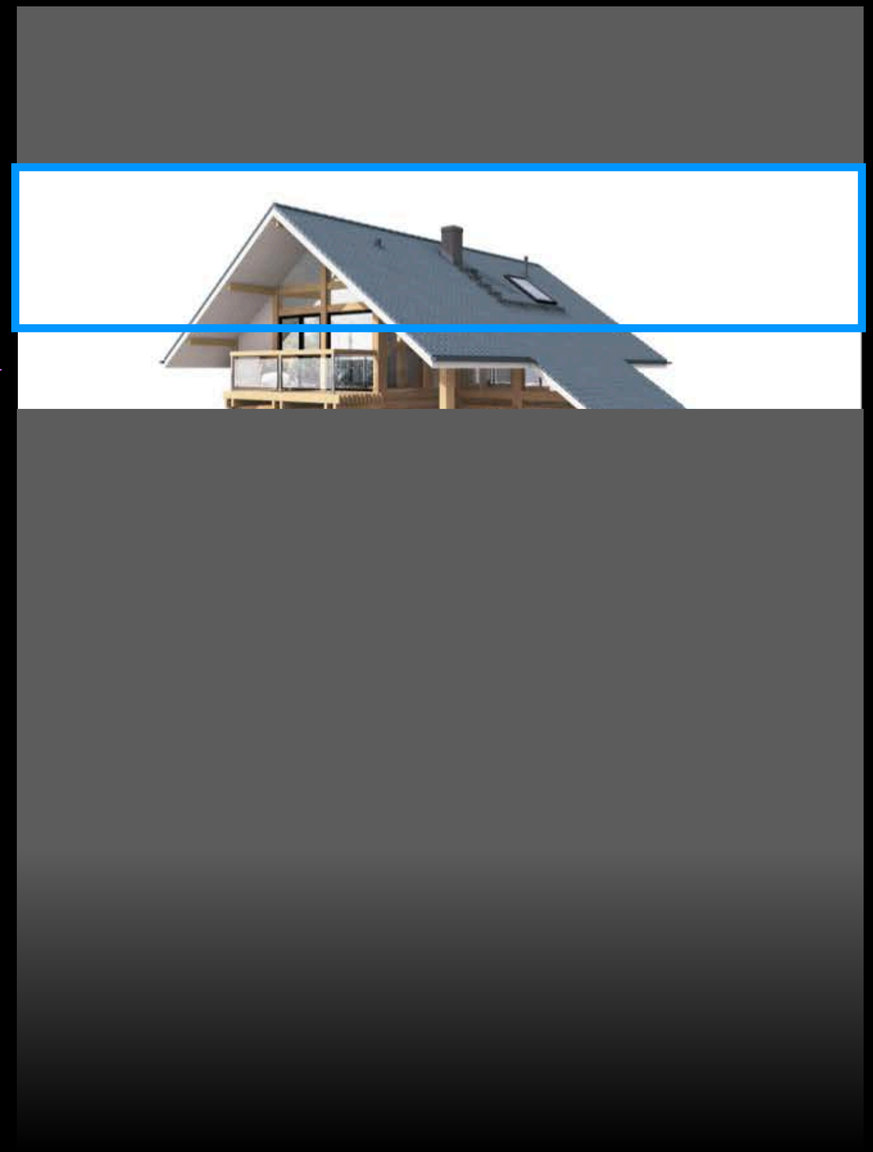
Idle prefetch



Overdraw

Idle prefetch

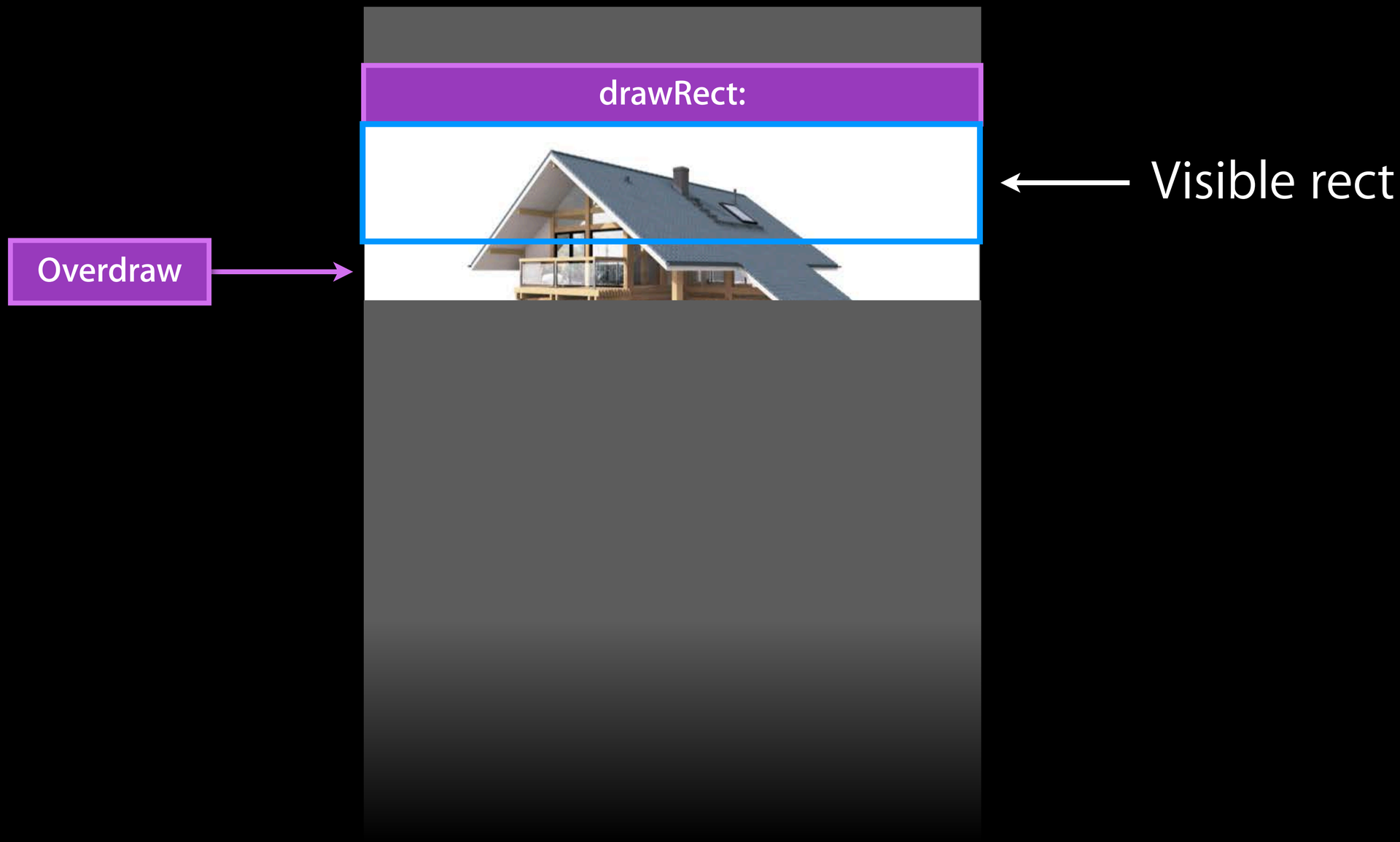
Overdraw



Visible rect

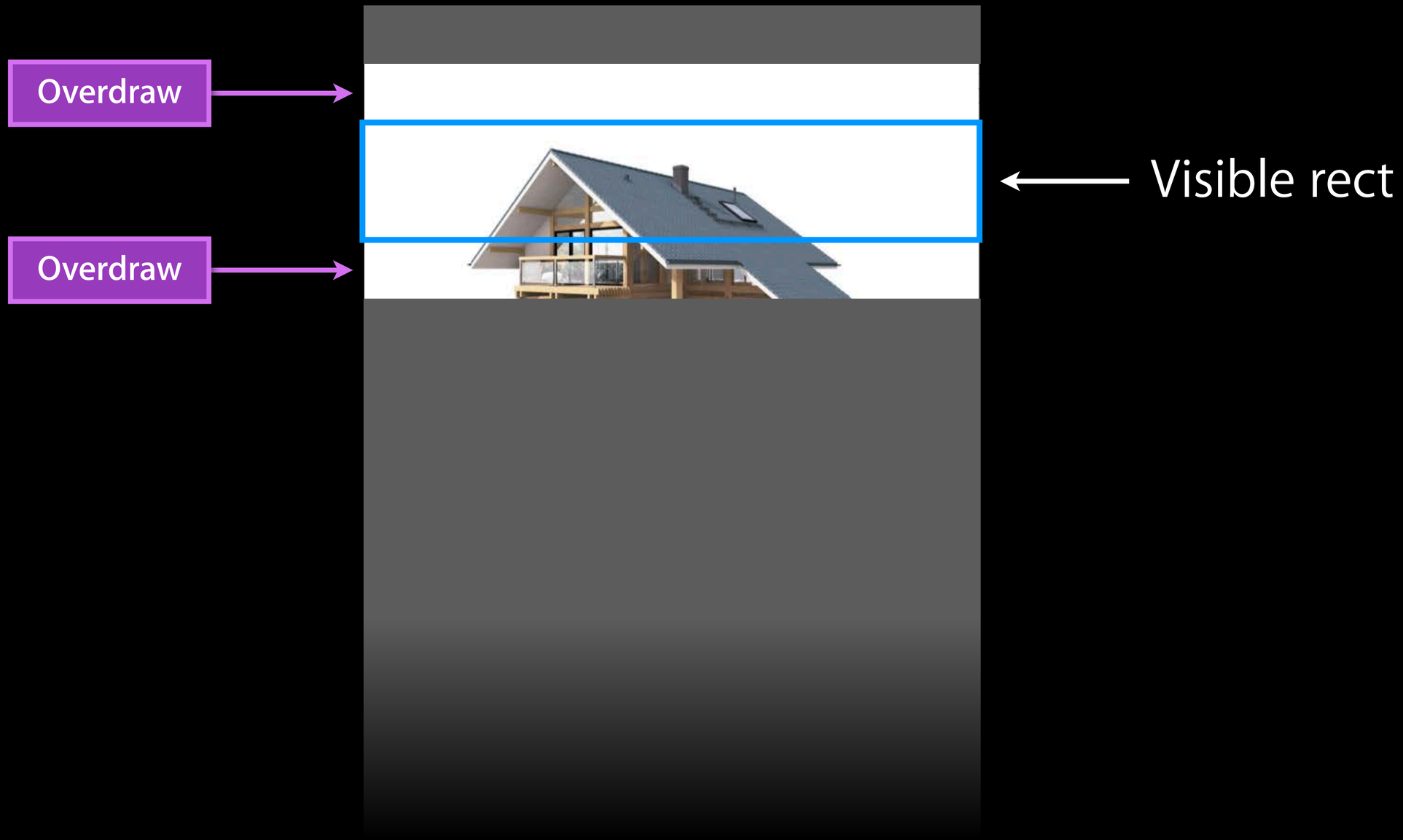
Overdraw

Idle prefetch



Overdraw

Idle prefetch



Overdraw

Idle prefetch

Overdraw

Overdraw

Visible rect

Overdraw

- Main thread driven
- -drawRect: called with non-visible rects
- AppKit balances overdraw amount with memory and power usage

Overdraw

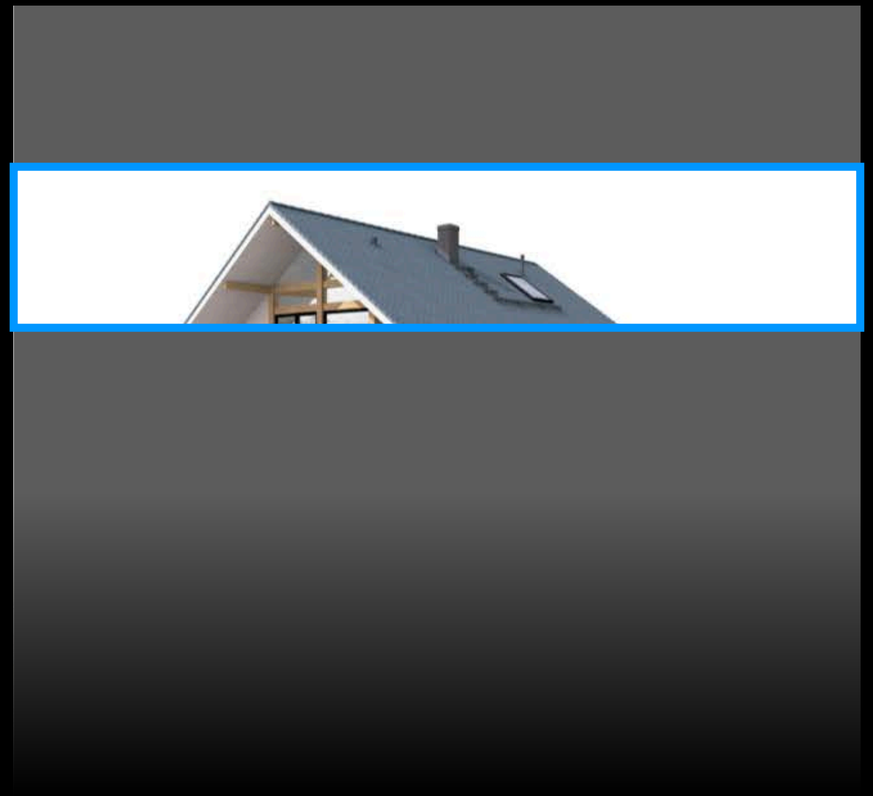
- Main thread driven
- -drawRect: called with non-visible rects
- AppKit balances overdraw amount with memory and power usage
- API if you need more control

```
@property NSRect preparedContentRect;
- (void)prepareContentInRect:(NSRect)rect;
```

Overdraw

API - Controlling overdraw

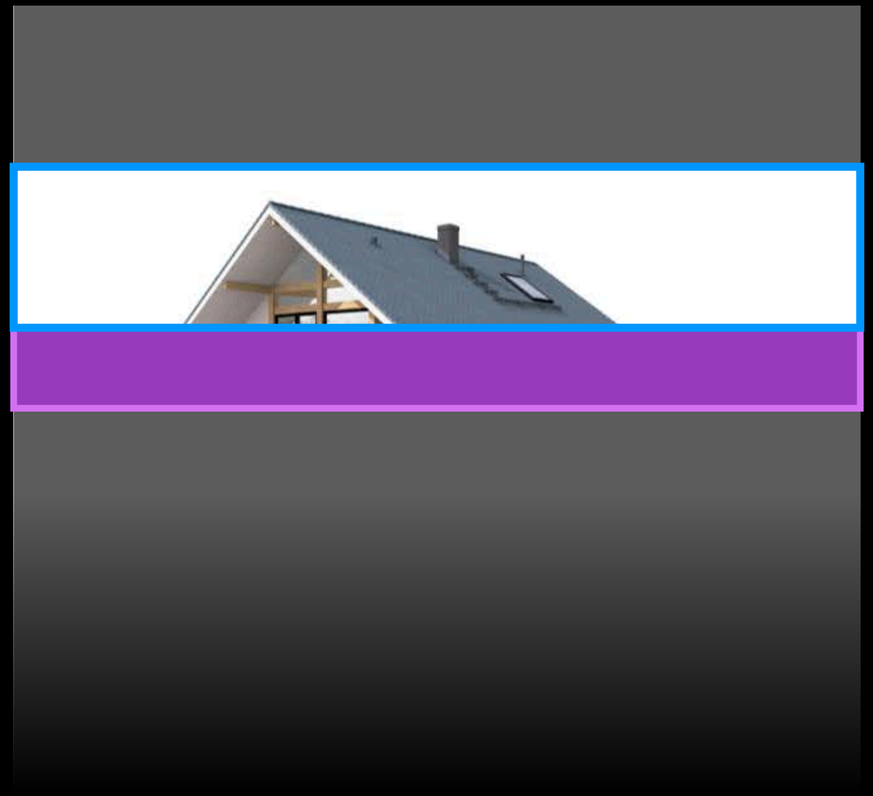
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

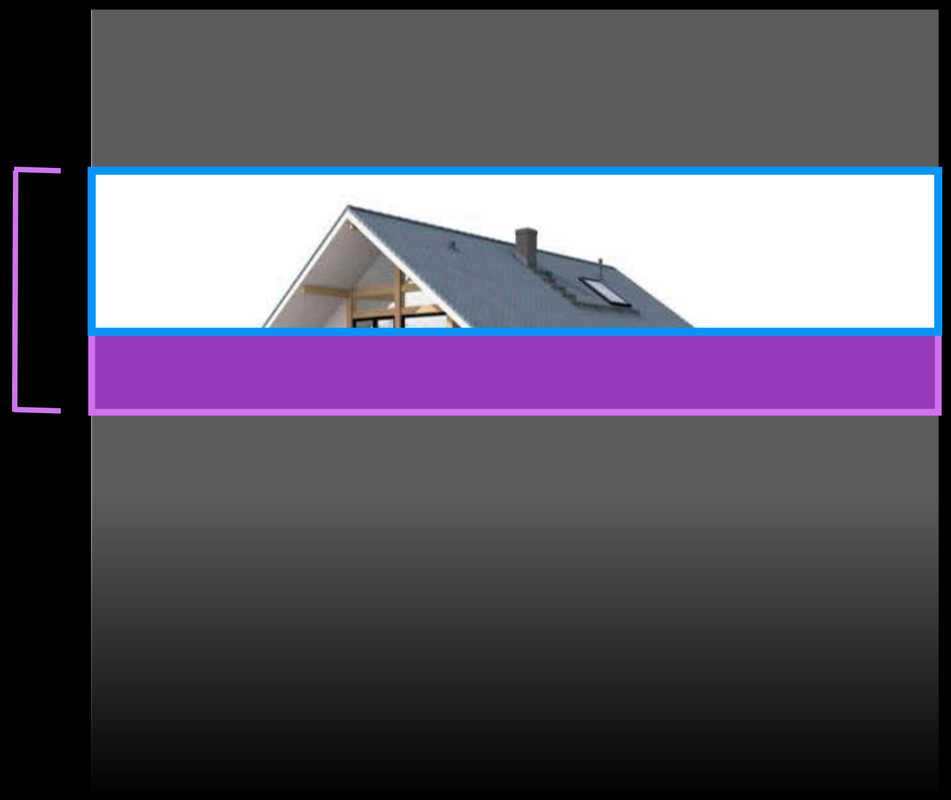
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

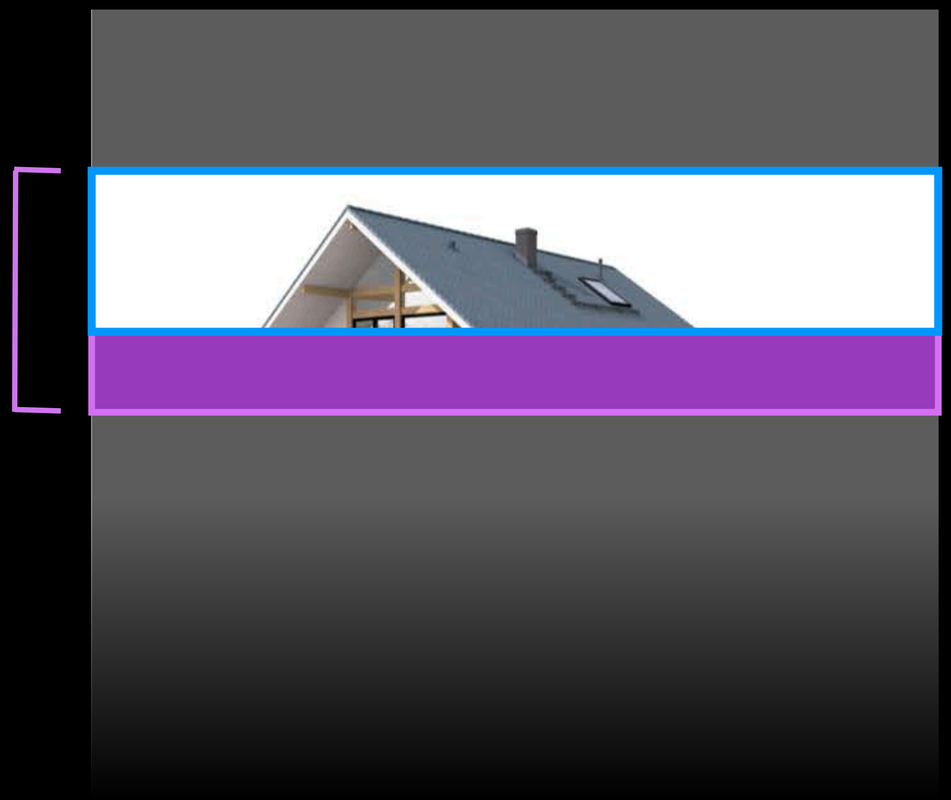
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

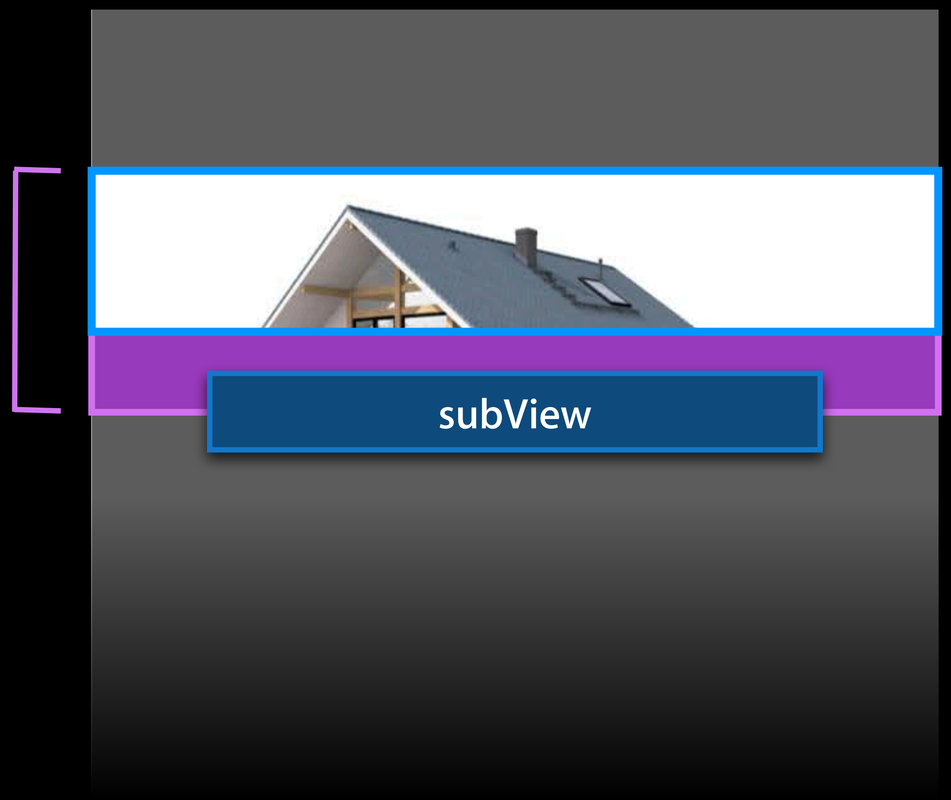
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

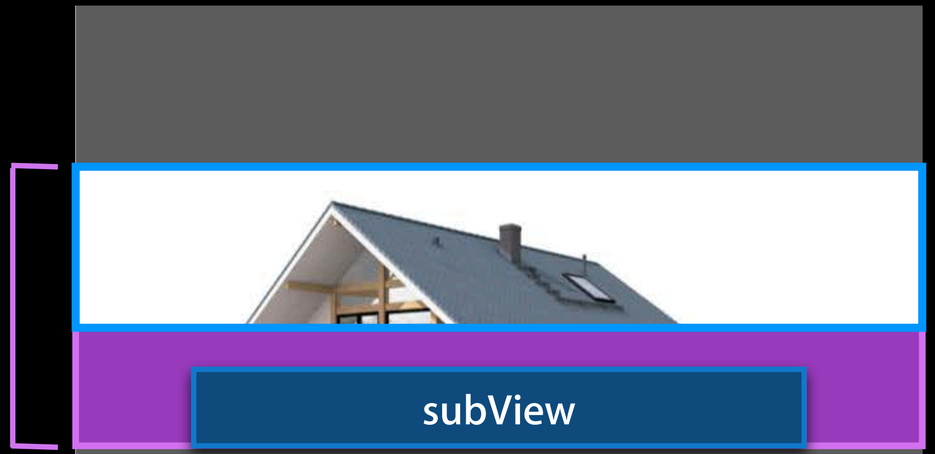
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

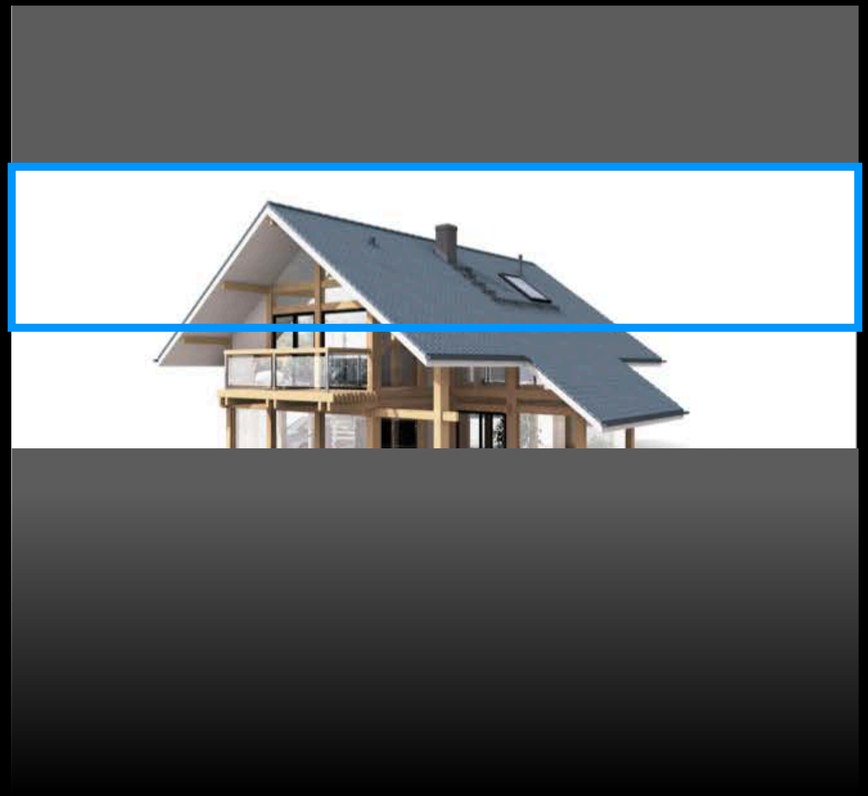
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

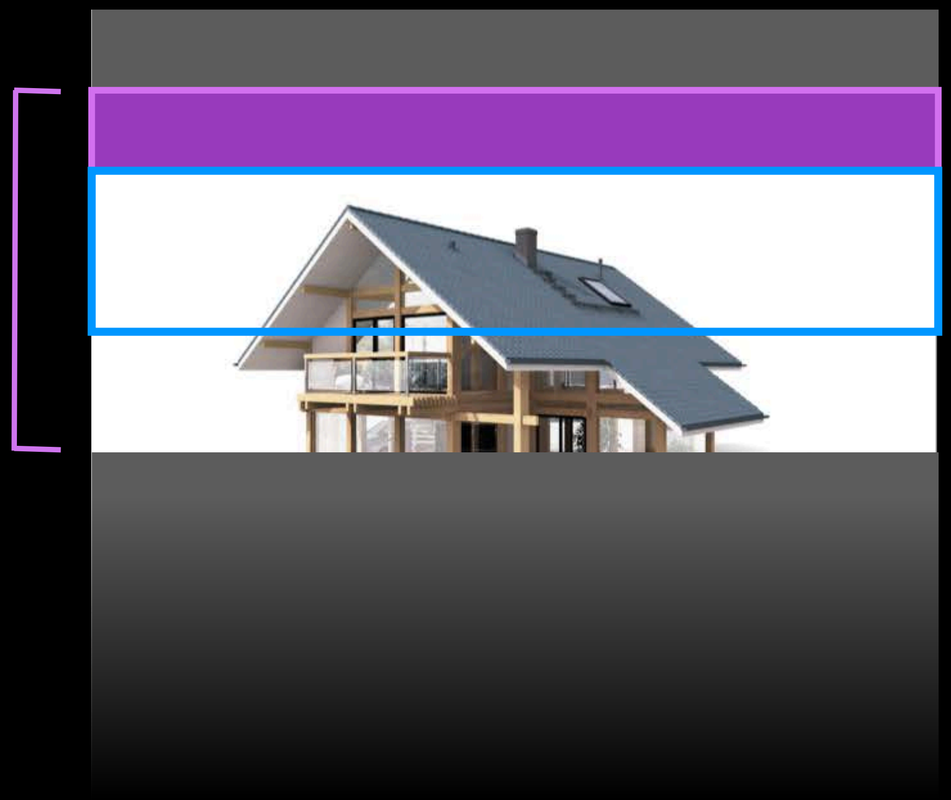
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:rect];
}
```



Overdraw

API - Controlling overdraw

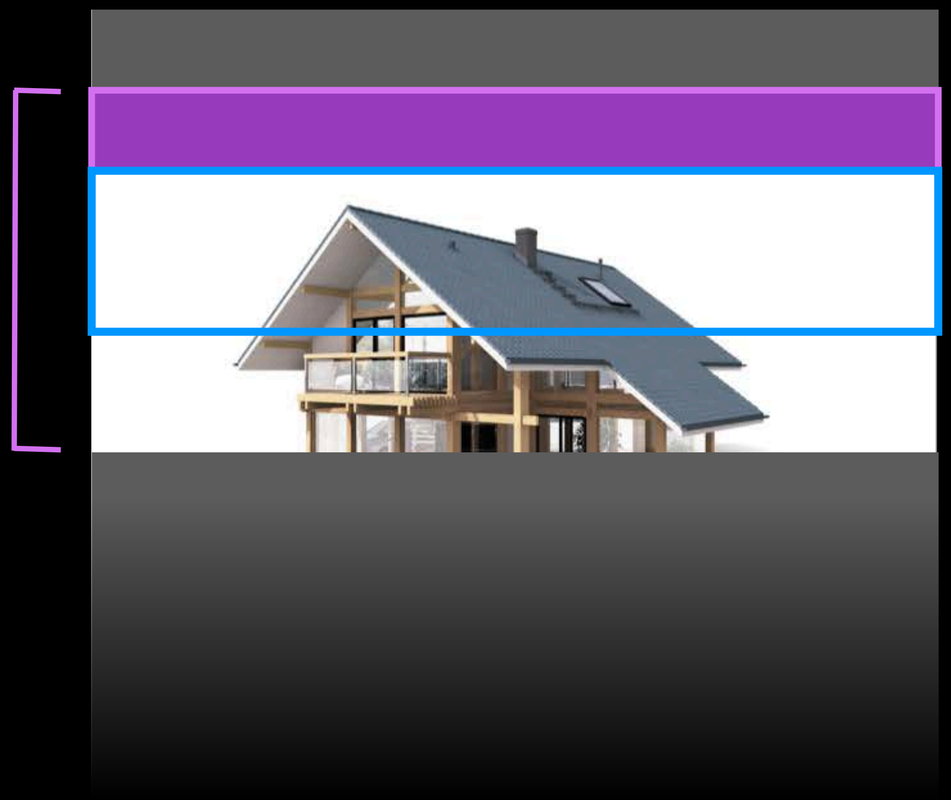
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:previousRect];
}
```



Overdraw

API - Controlling overdraw

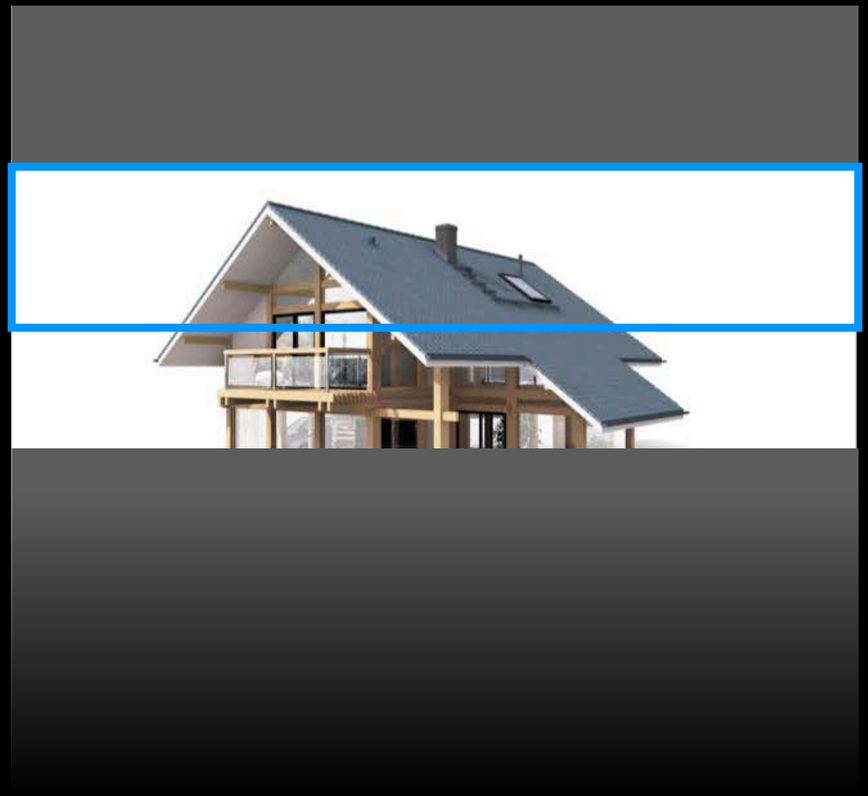
```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:previousRect];
}
```



Overdraw

API - Controlling overdraw

```
- (void)prepareContentInRect:(NSRect)rect {
  // prepare as needed
  [super prepareContentInRect:previousRect];
}
```



Overdraw

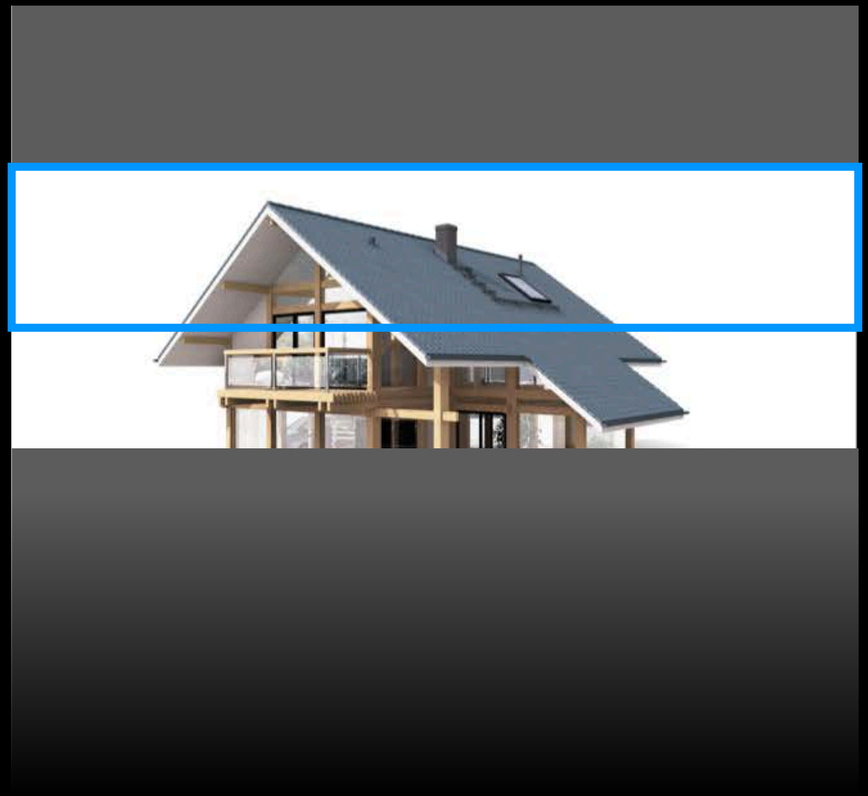
API - Invalidating non-visible content

```
[documentView setNeedsDisplayInRect:rect];
```

Overdraw

API - Invalidating non-visible content

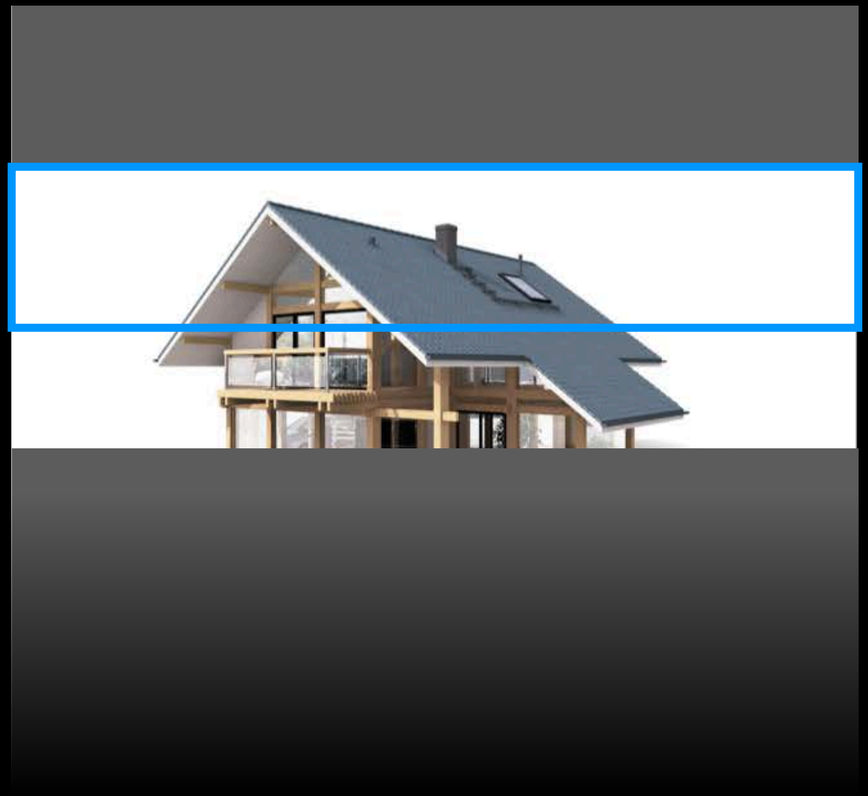
```
[documentView setNeedsDisplayInRect:rect];
```



Overdraw

API - Resetting overdraw

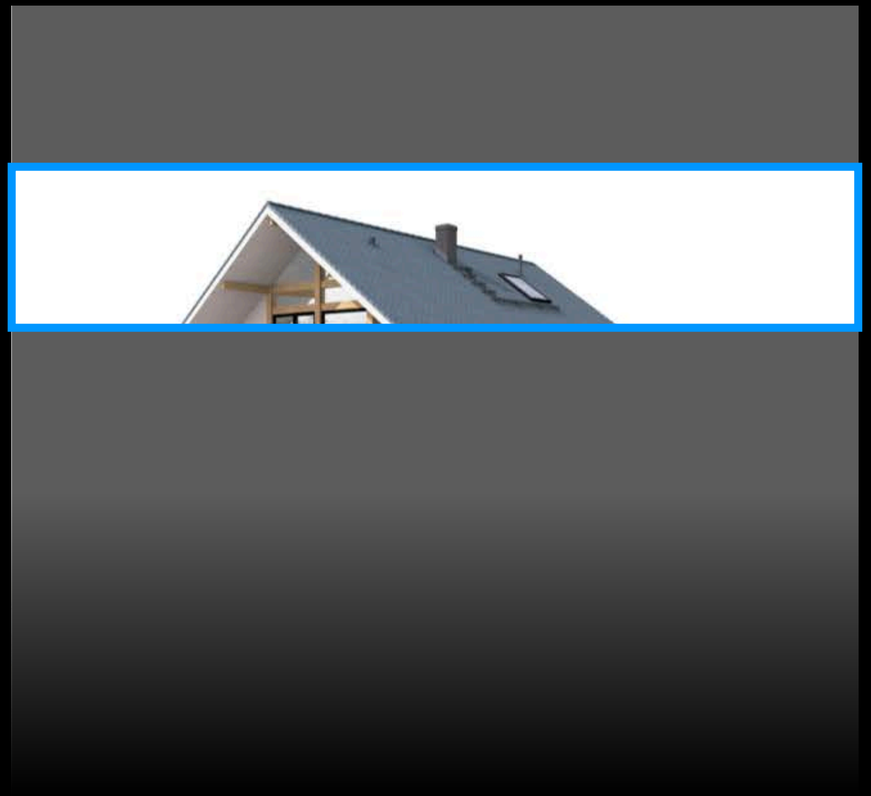
```
docView.preparedContentRect = [docView visibleRect];
```



Overdraw

API - Resetting overdraw

```
docView.preparedContentRect = [docView visibleRect];
```



Overdraw

- Main thread driven
- -drawRect: called with non-visible rects
- AppKit balances overdraw amount with memory and power usage
- API if you need more control

```
@property NSRect preparedContentRect;
- (void)prepareContentInRect:(NSRect)rect;
```

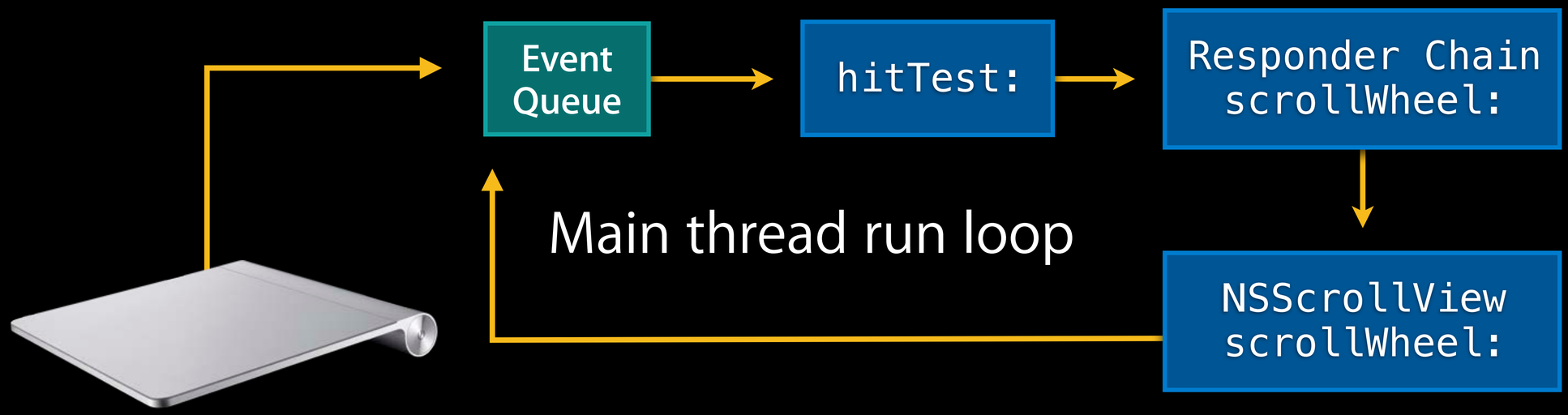
Event Model

Responsive scrolling

Event Model

Traditional

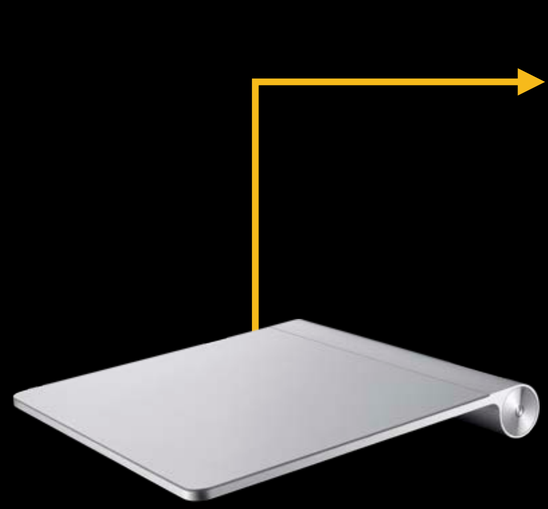
Each scroll wheel event is independent



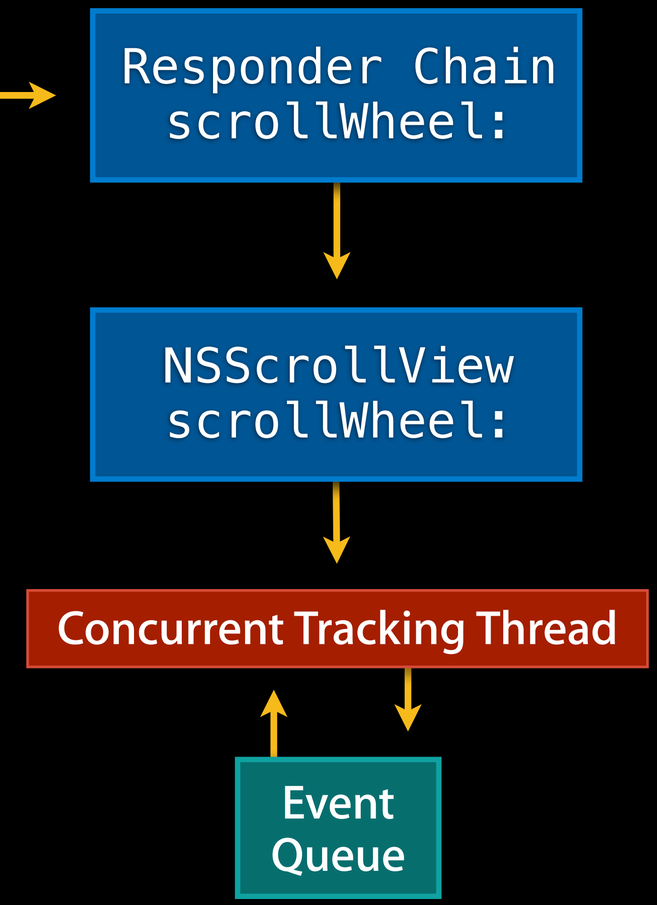
Event Model

Responsive

Scroll wheel events tracked concurrently



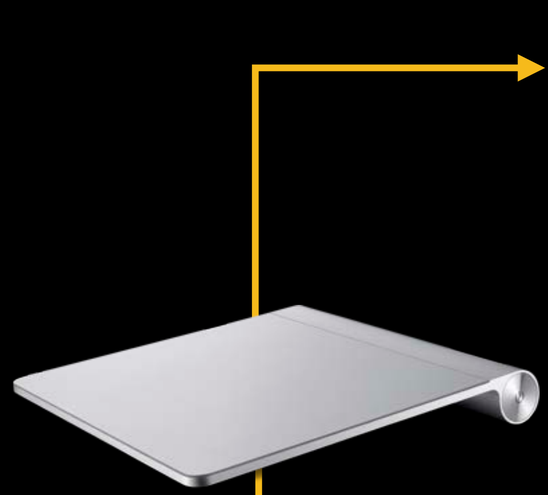
Main thread run loop



Event Model

Responsive

Scroll wheel events tracked concurrently



Event Queue

Main thread run loop

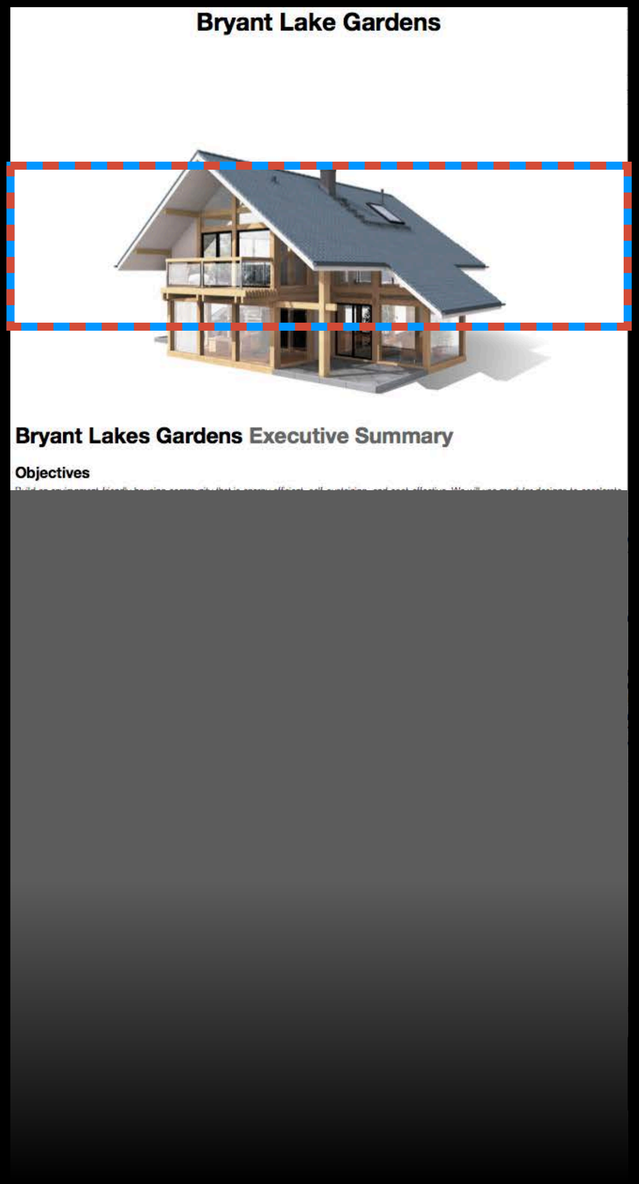
NSScrollView scrollWheel:

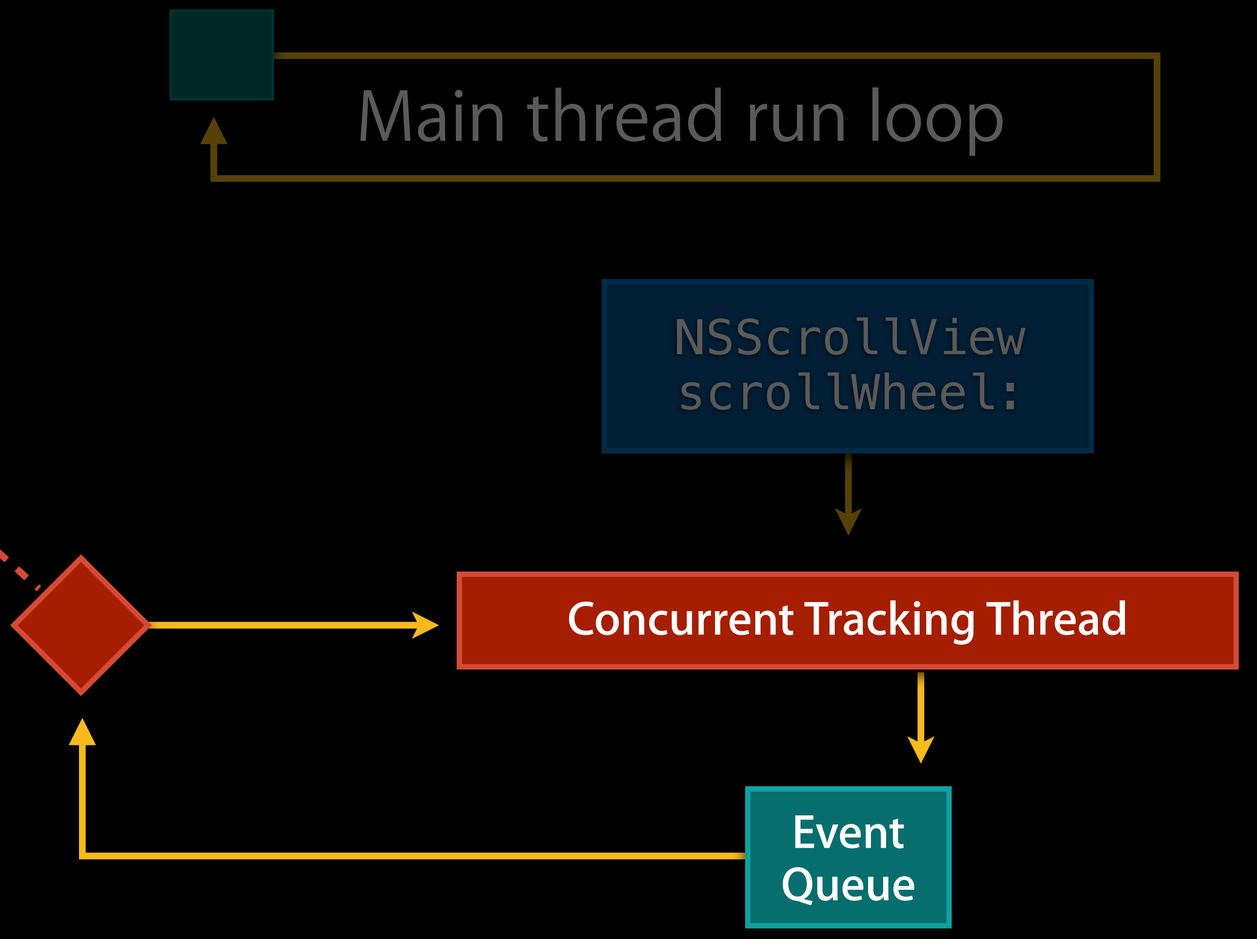
Concurrent Tracking Thread

Event Queue

Event Model

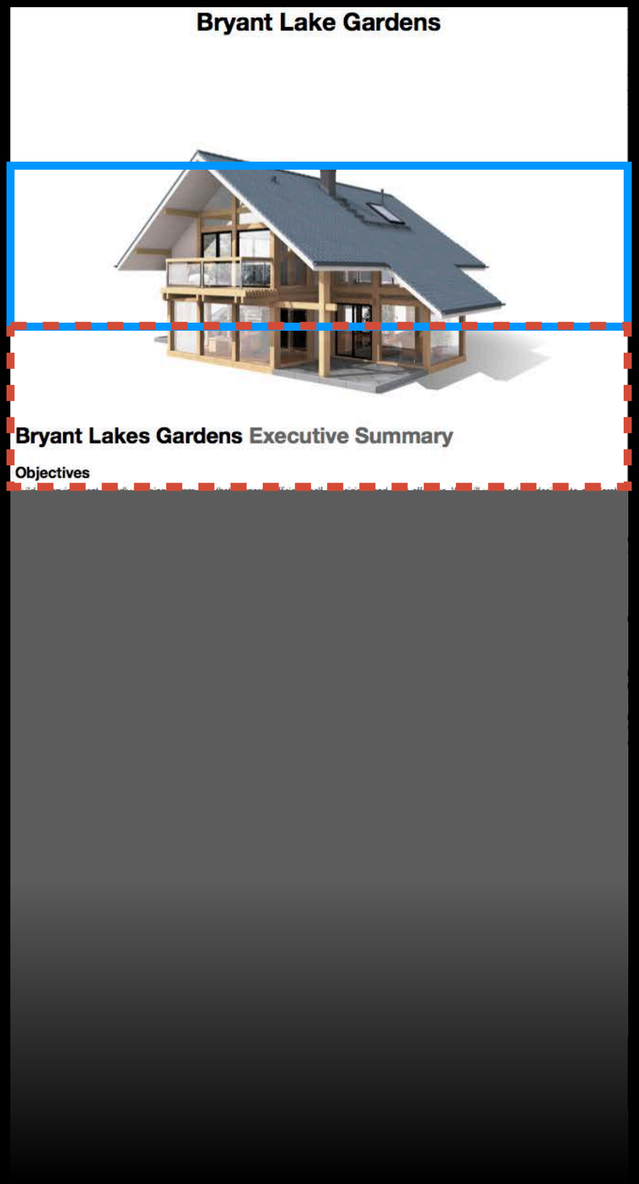
Responsive





Event Model

Responsive



Main thread run loop

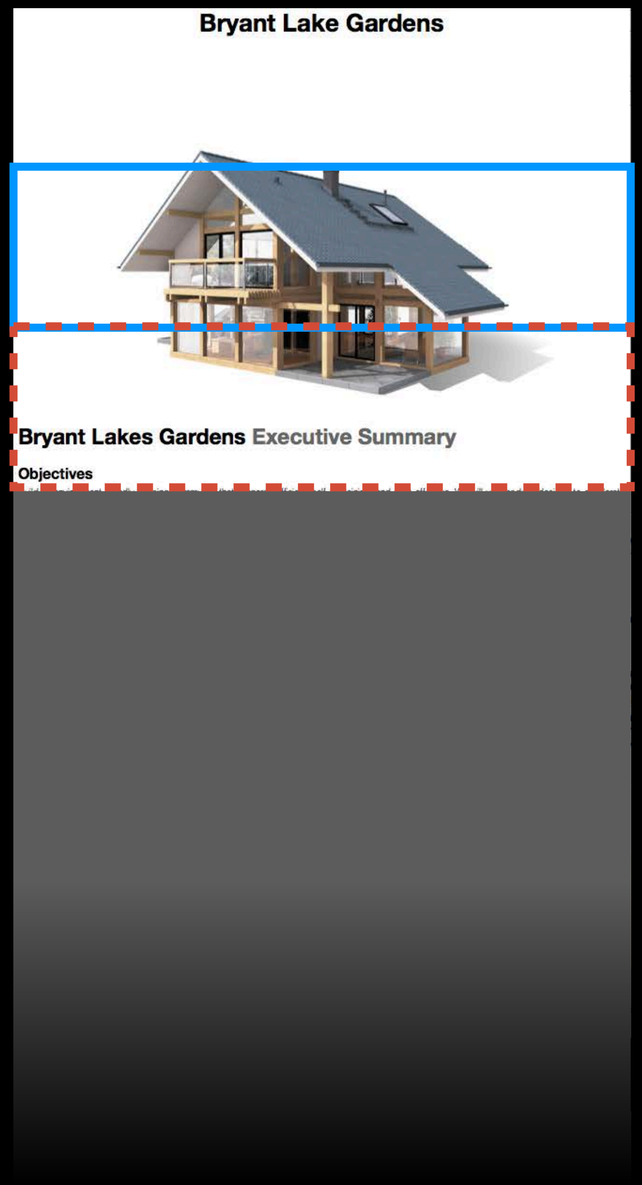
NSScrollView
scrollWheel:

Concurrent Tracking Thread

Event
Queue

Event Model

Responsive



Main thread run loop

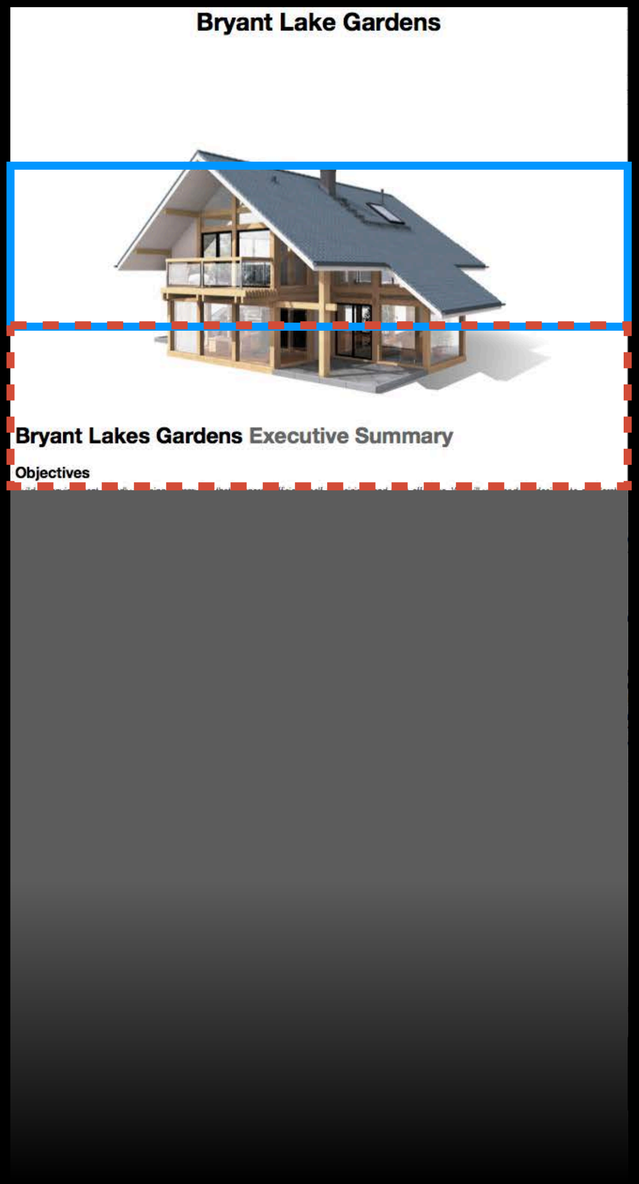
NSScrollView
scrollWheel:

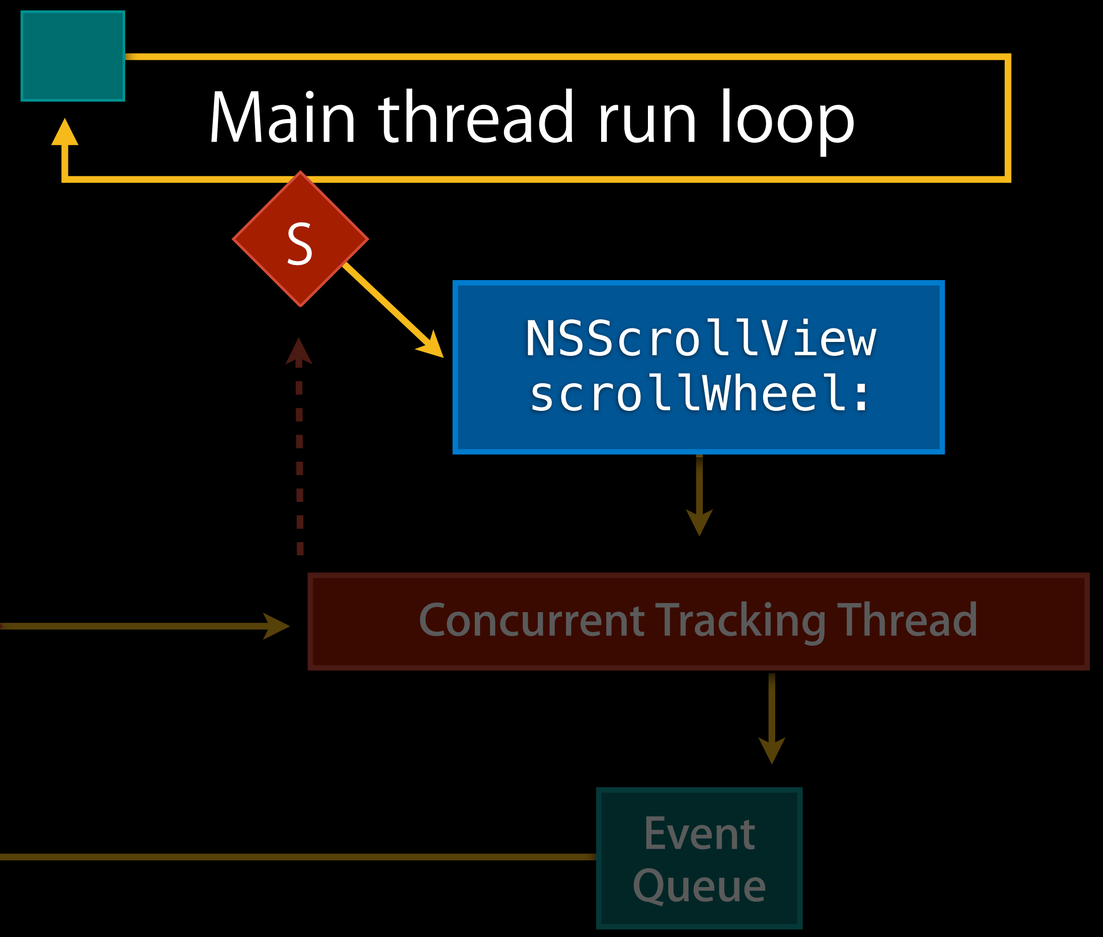
Concurrent Tracking Thread

Event
Queue

Event Model

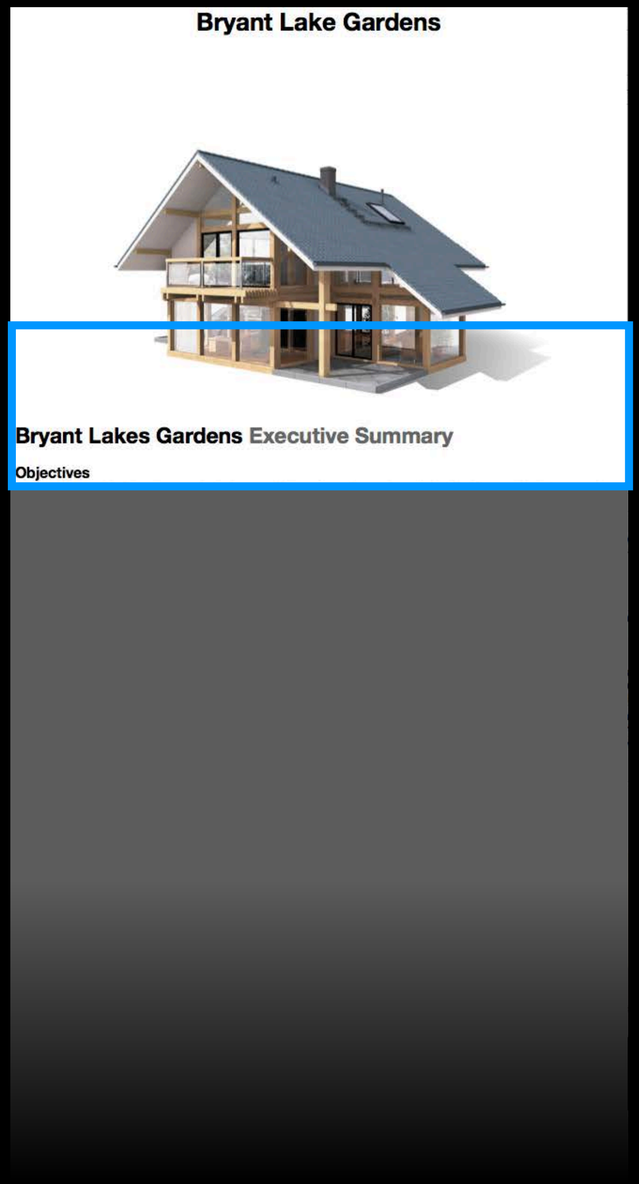
Responsive

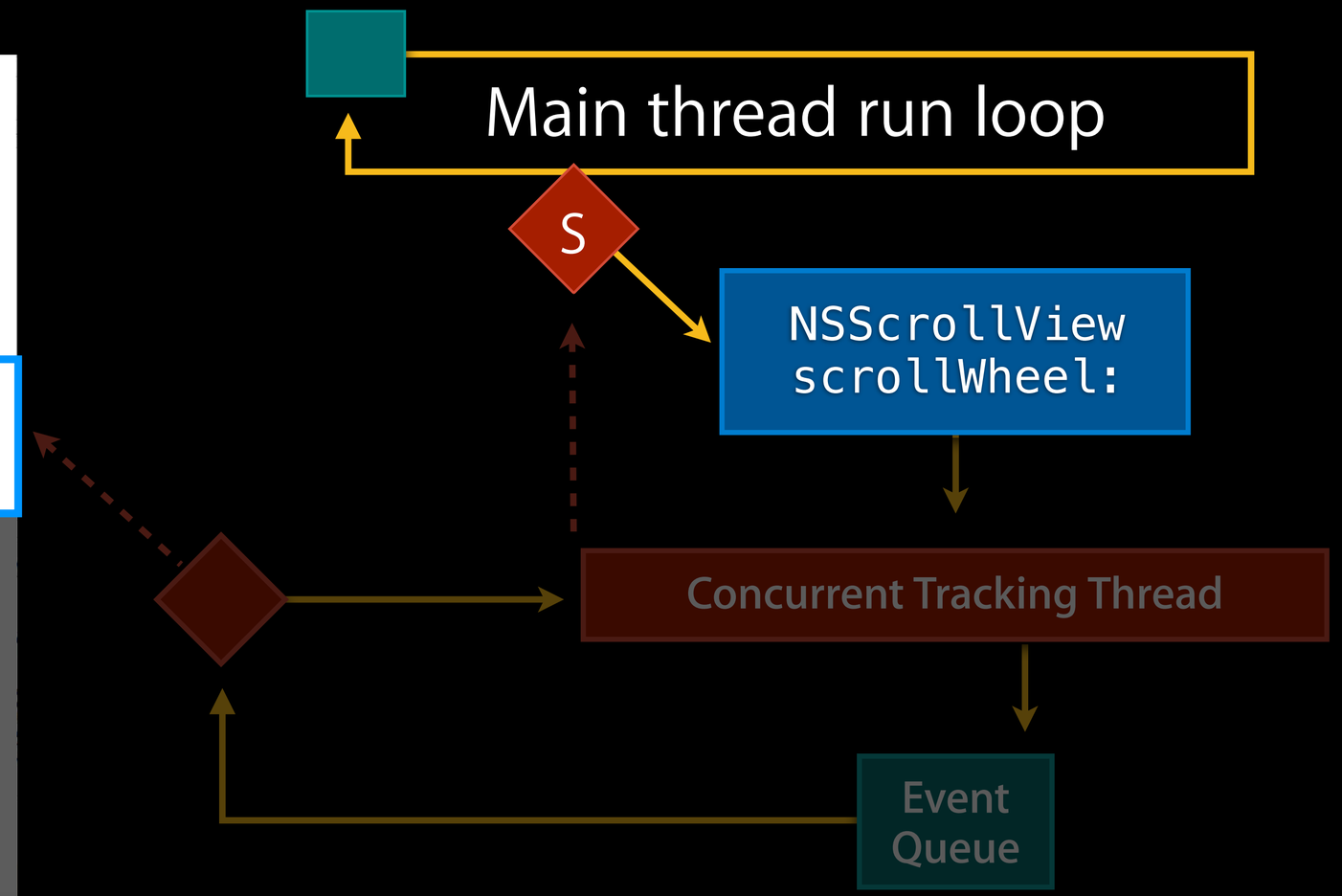




Event Model

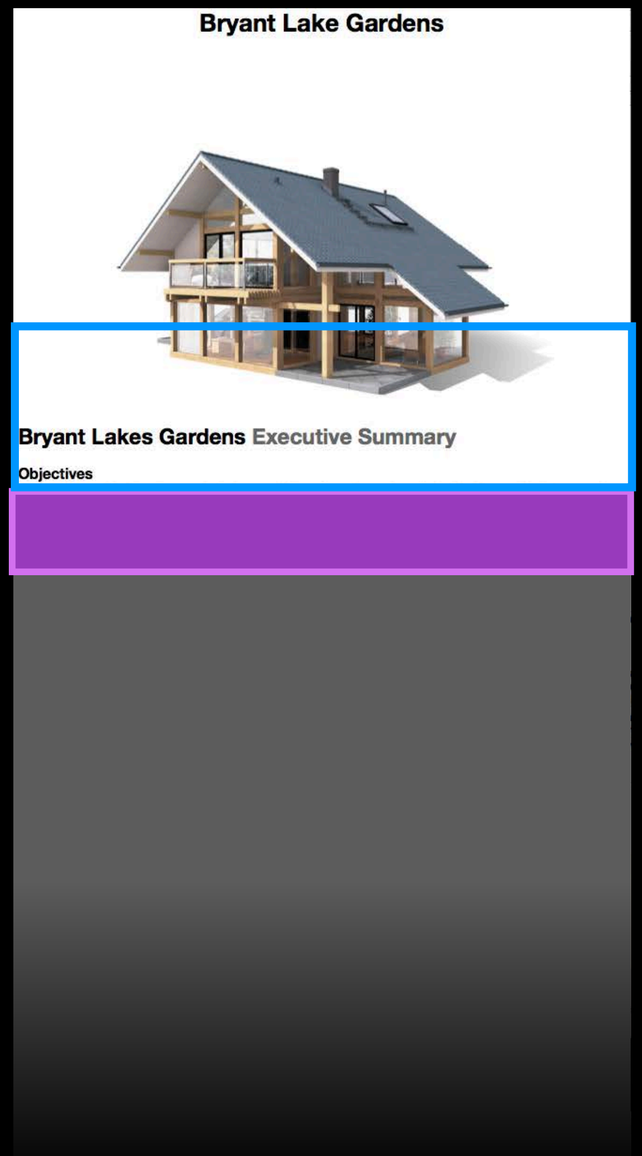
Responsive

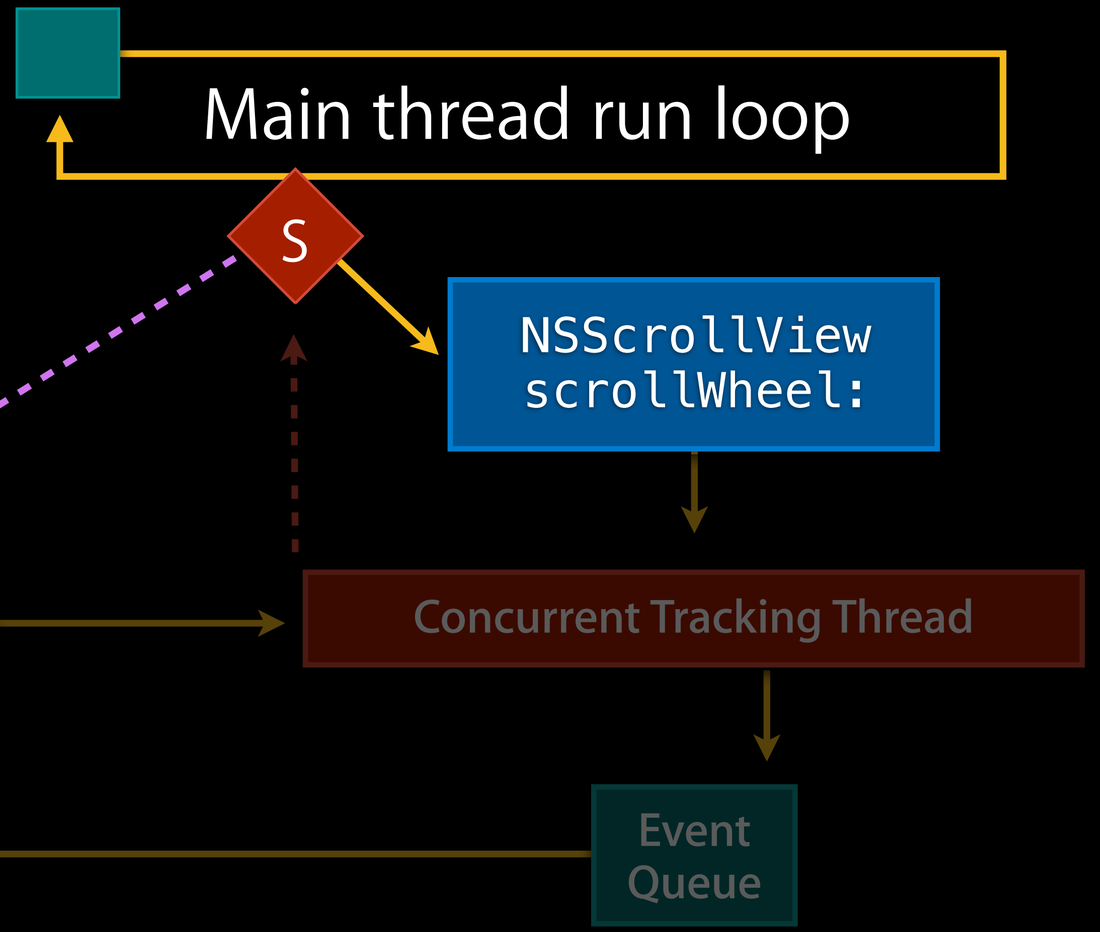




Event Model

Responsive





Responsive Scrolling

Overview

- Concurrent event tracking
- What is on screen may not match visibleRect
- Not a silver bullet

API

Responsive scrolling

API

Getting informed when scrolling occurs

- Observe clip view bounds change notifications

```
NSClipView *clipView = [scrollView contentView];
[clipView setPostsBoundsChangedNotifications:YES];
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(boundsChanged:)
                                             name:NSViewBoundsDidChangeNotification
                                           object: clipView];
```

API

Getting informed when scrolling occurs

- Observe clip view bounds change notifications

```
NSClipView *clipView = [scrollView contentView];
[clipView setPostsBoundsChangedNotifications:YES];
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(boundsChanged:)
                                             name:NSViewBoundsDidChangeNotification
                                           object: clipView];
```

API

Getting informed of user scrolling

- Live scroll notifications

```
NSScrollViewWillStartLiveScroll

NSScrollViewDidLiveScroll

NSScrollViewDidEndLiveScroll
```

API

Getting informed of user scrolling

- Live scroll notifications

```
NSScrollViewWillStartLiveScroll

NSScrollViewDidLiveScroll

NSScrollViewDidEndLiveScroll
```

API

Getting informed of user scrolling

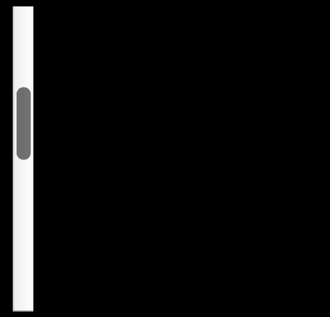
- Live scroll notifications

```
NSScrollViewWillStartLiveScroll

NSScrollViewDidLiveScroll

NSScrollViewDidEndLiveScroll
```

API

Getting informed of user scrolling

- Live scroll notifications



```
NSScrollViewWillStartLiveScroll

NSScrollViewDidLiveScroll

NSScrollViewDidEndLiveScroll
```

API

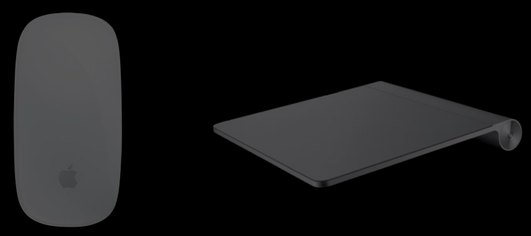
Getting informed of user scrolling

- Live scroll notifications

NSScrollViewWillStartLiveScroll

NSScrollViewDidLiveScroll

NSScrollViewDidEndLiveScroll

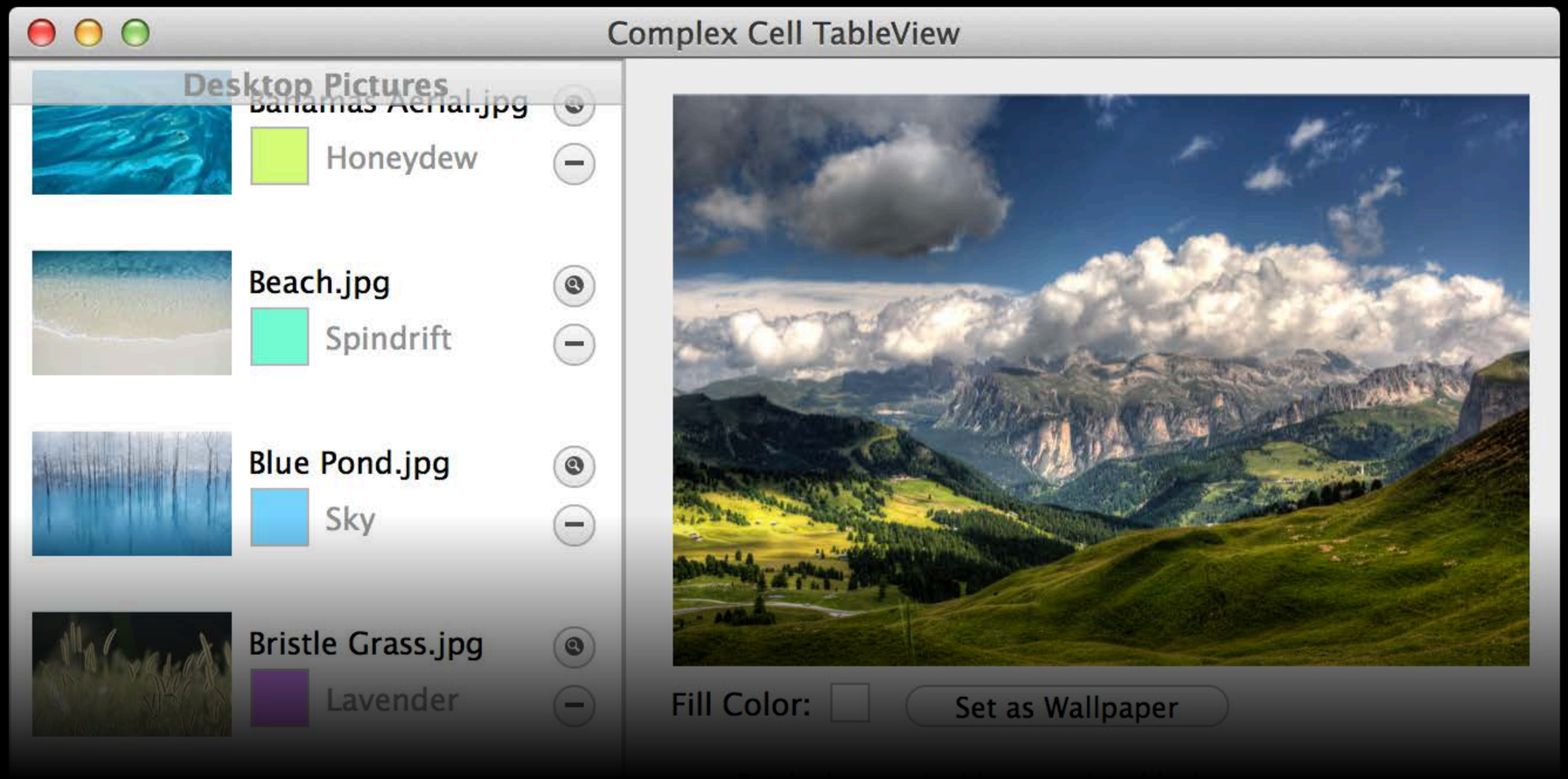


NSScrollView API

Floating content

- Floating subviews

```
- (void)addFloatingSubview:(NSView *)view forAxis:(NSEventGestureAxis)axis;
```

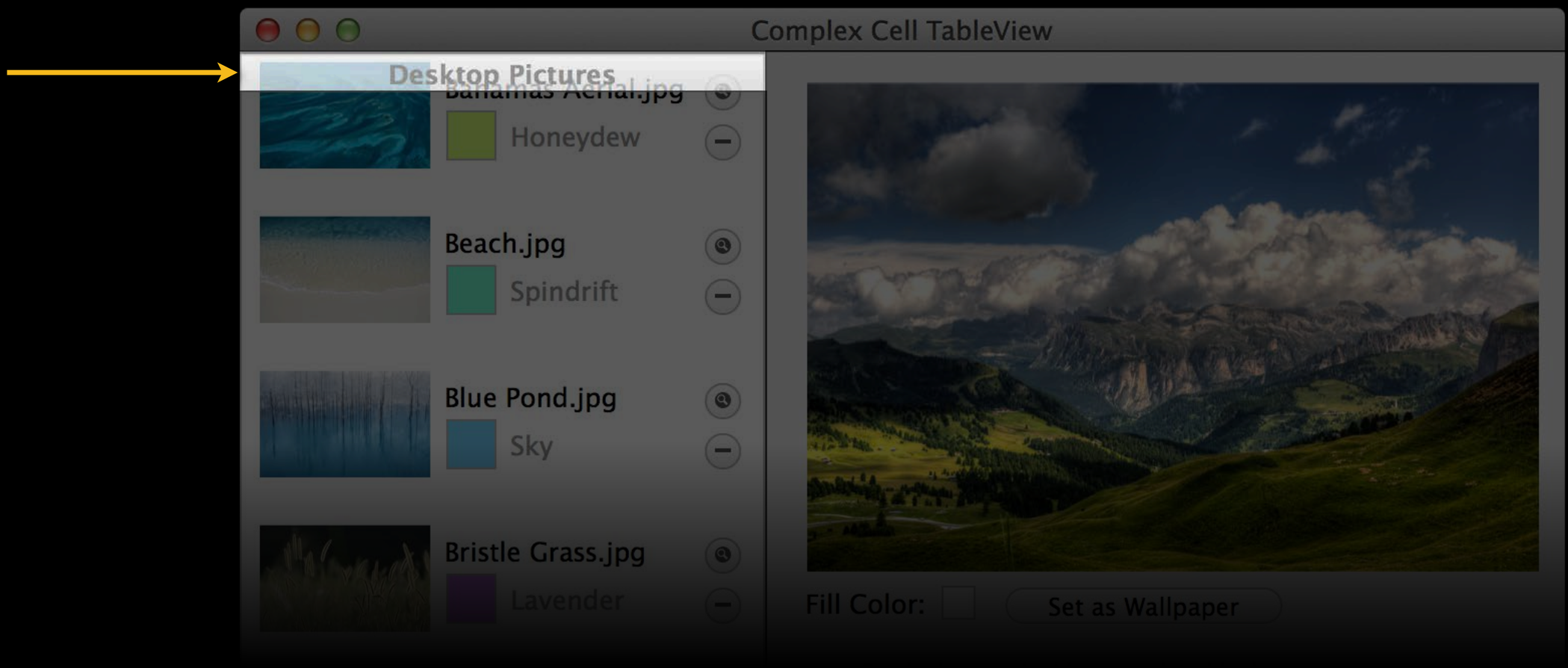


NSScrollView API

Floating content

- Floating subviews

```
- (void)addFloatingSubview:(NSView *)view forAxis:(NSEventGestureAxis)axis;
```



Adoption

Responsive scrolling

Adoption

Responsive scrolling

- Linked on 10.8 or later
- Window alpha must be 1.0
- Document must not have an OpenGL context

Adoption

Responsive scrolling

- Automatic

Adoption

Responsive scrolling

- Automatic

NSScrollView	NSClipView	Document View

Adoption

Responsive scrolling

- Automatic

NSScrollView	NSClipView	Document View

- Explicit API

```
+ (BOOL)isCompatibleWithResponsiveScrolling;
```

Adoption

Responsive scrolling

- Do not override

```
-scrollWheel:
-lockFocus:
```

Adoption

Responsive scrolling

- Traditional drawing
  - copiesOnScroll must be YES
  - isOpaque must return YES for document view

- Or -

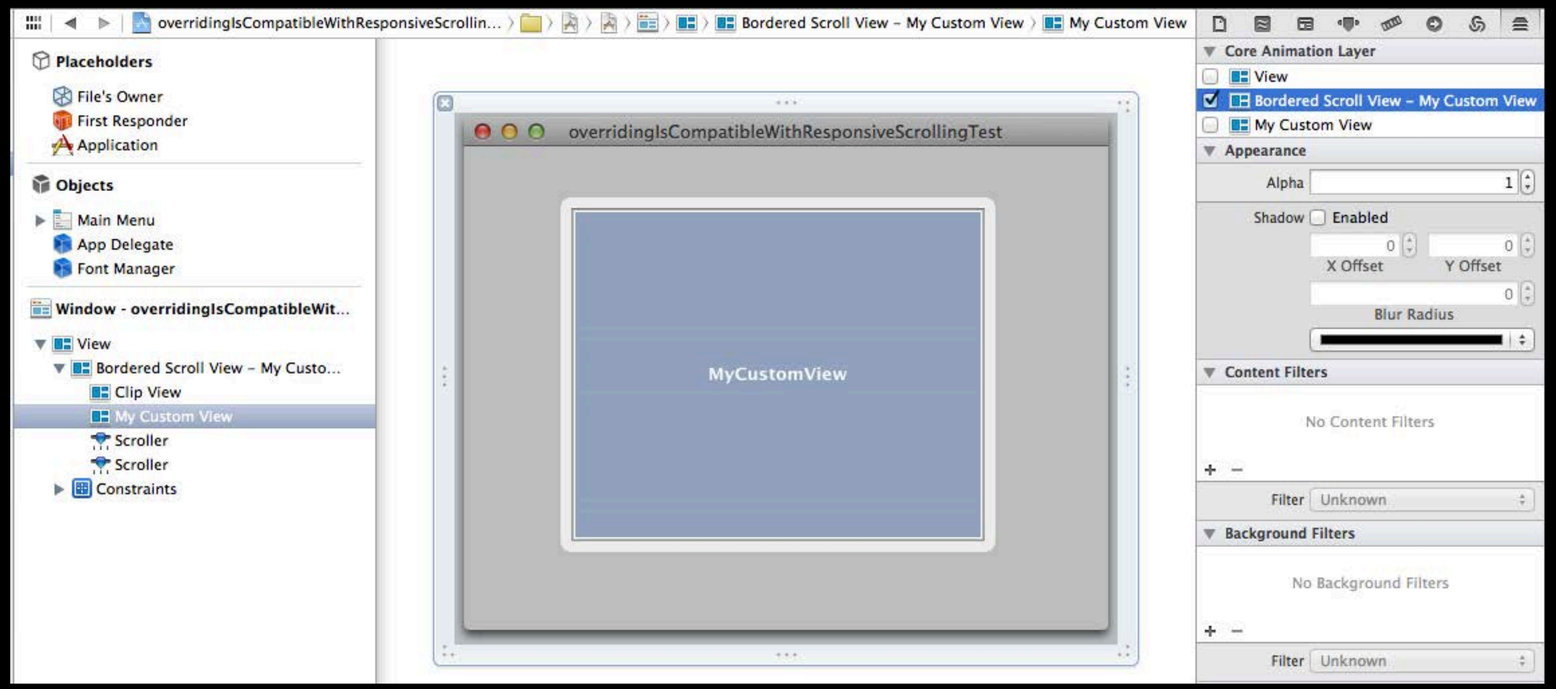
- Layer-back the scroll view

Adoption: Layer-Backing

Responsive scrolling

- NSScrollView or ancestor

```
- (void)setWantsLayer:(BOOL)flag;
- (BOOL)wantsLayer;
```

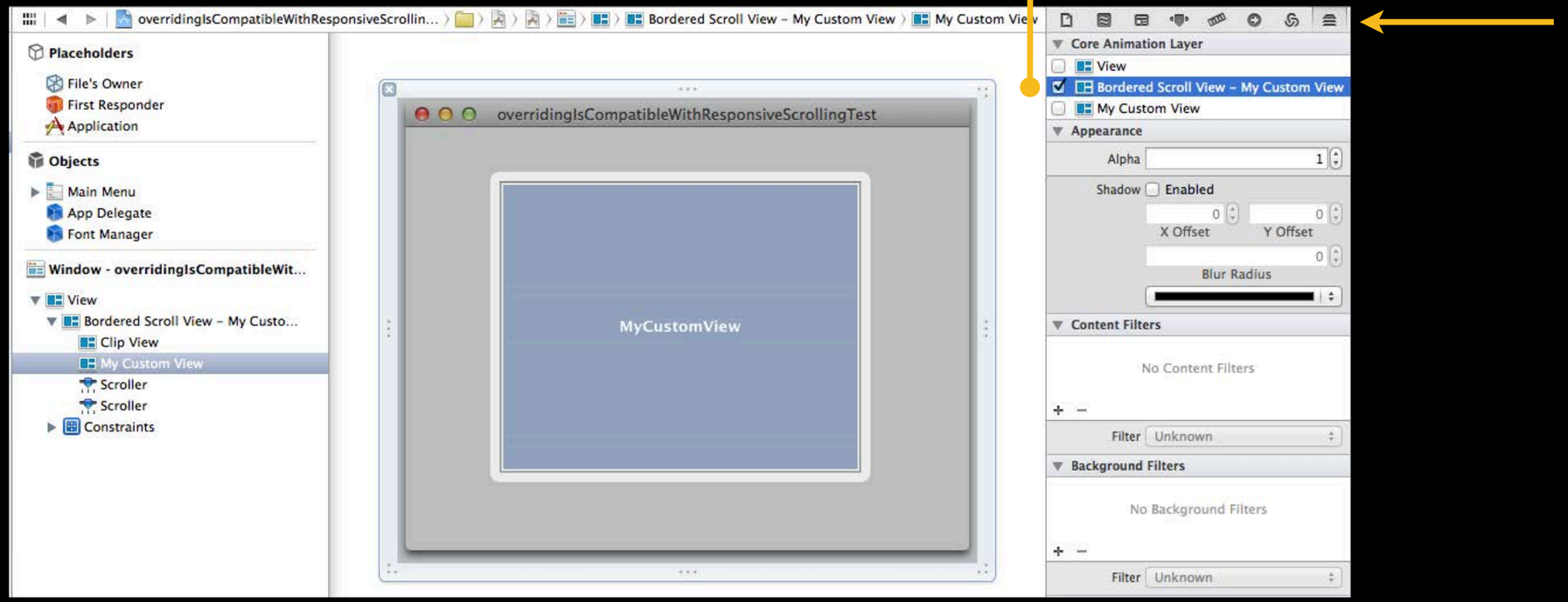


Adoption: Layer-Backing

Responsive scrolling

- NSScrollView or ancestor

```
- (void)setWantsLayer:(BOOL)flag;
- (BOOL)wantsLayer;
```



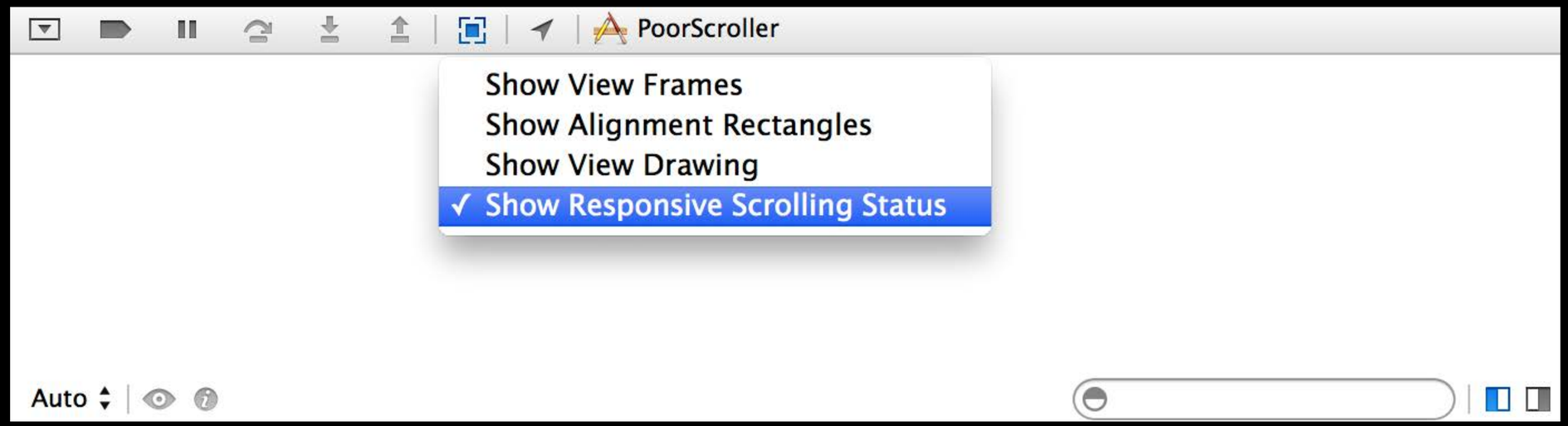
Adoption: Layer-Backing

Responsive scrolling

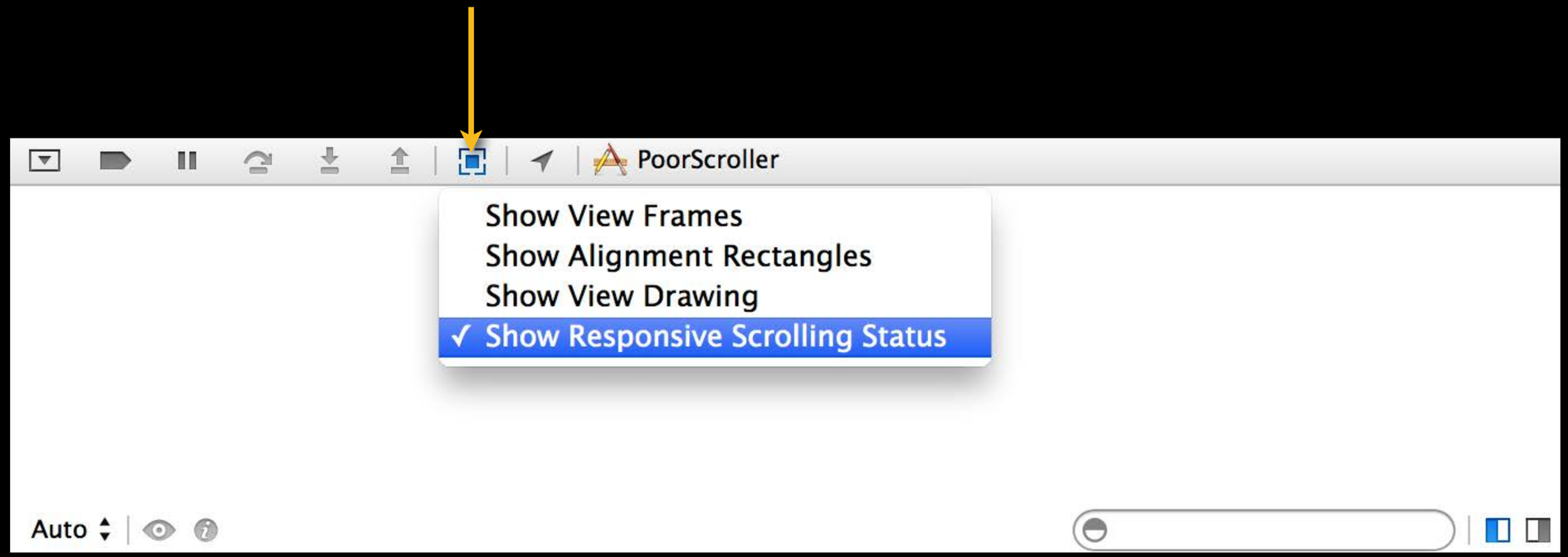
- Collapsing layers of document view or children

```
- (void)setCanDrawSubviewsIntoLayer:(BOOL)flag;
```

Xcode Support

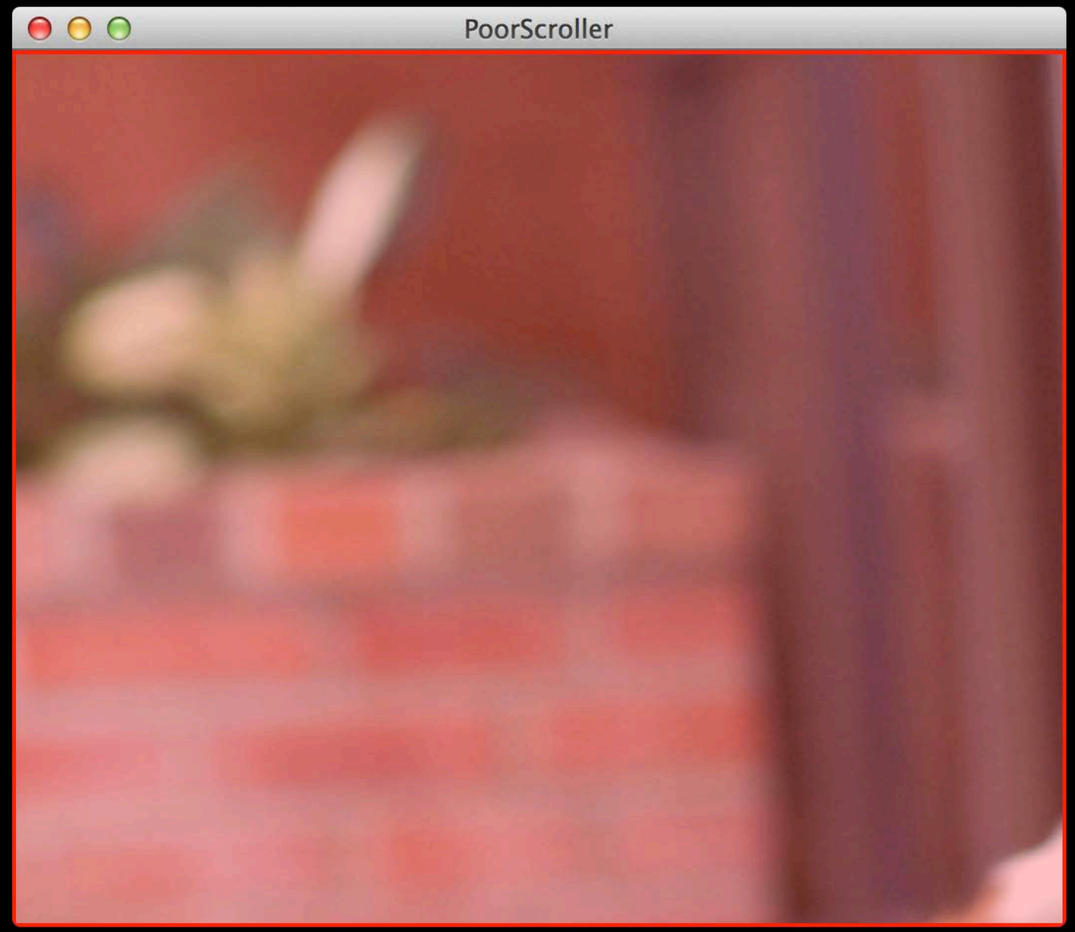


Xcode Support



Xcode Support

Traditional scrolling



Xcode Support

Responsive scrolling

Adoption

Summary

- Automatic when possible

Adoption

Summary

- Automatic when possible
- Explicitly opt in as last resort

Adoption

Summary

- Automatic when possible
- Explicitly opt in as last resort
- Layer-backed vs. traditional drawing

Adoption

Summary

- Automatic when possible
- Explicitly opt in as last resort
- Layer-backed vs. traditional drawing
- Use Xcode to verify

Magnification

NSScrollView

Magnification

Responsiveness

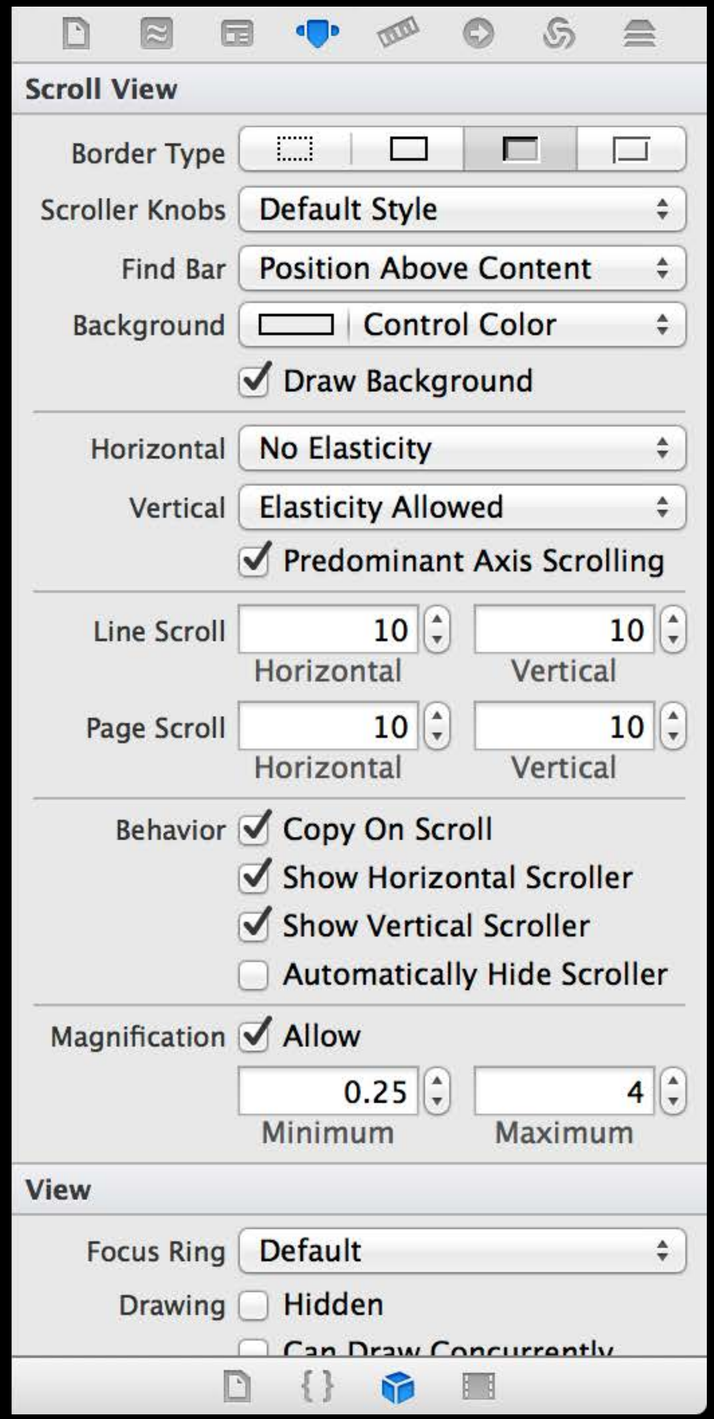
- NSScrollView supports magnification

```
@property BOOL allowsMagnification NS_AVAILABLE_MAC(10_8);
```

Magnification

Responsiveness

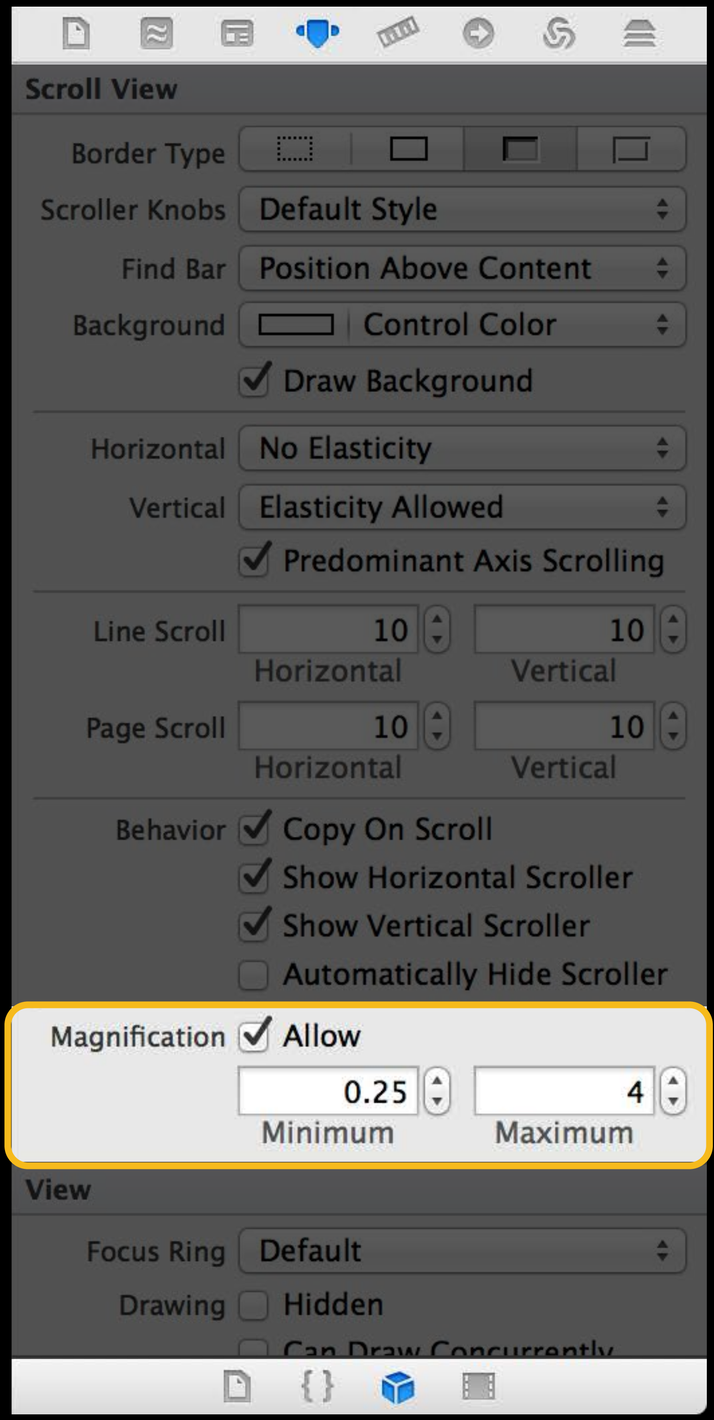
- NSScrollView supports magnification



Magnification

Responsiveness

- NSScrollView supports magnification



Magnification

Responsiveness

- Still main thread driven

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends

Magnification

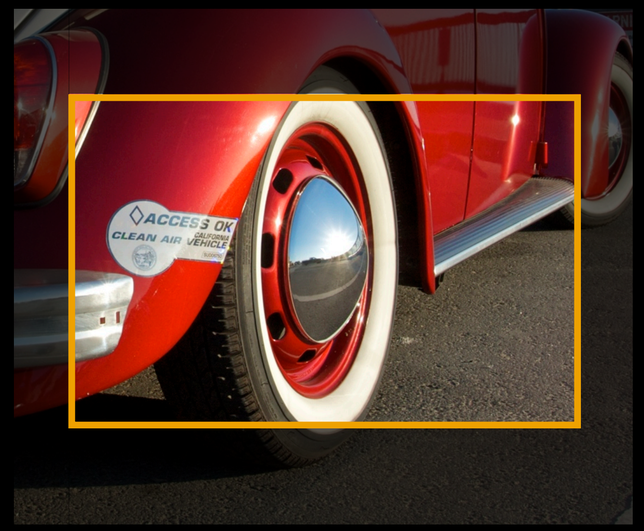
Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends
- Pause for new drawing



Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends
- Pause for new drawing

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends
- Pause for new drawing

Magnification

Responsiveness

- Still main thread driven
- Likely have overdraw
- During gesture we scale existing content
- Visible rect redrawn when gesture ends
- Pause for new drawing

Magnification

Responsiveness

- -drawRect: speed is crucial

Magnification

Responsiveness

- -drawRect: speed is crucial
- Live magnification notifications

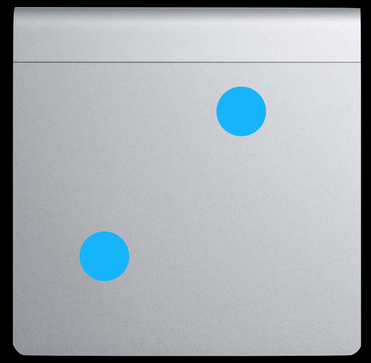
```
NSScrollViewWillStartLiveMagnifyNotification NS_AVAILABLE_MAC(10_8);
NSScrollViewDidEndLiveMagnifyNotification NS_AVAILABLE_MAC(10_8);
```

Magnification

Centering clip views

- Deprecated API

```
- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;
```

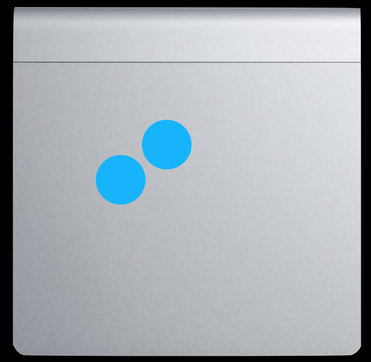


Magnification

Centering clip views

- Deprecated API

```
- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;
```



Magnification

Centering clip views

- Deprecated API

```
- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;
```

Magnification

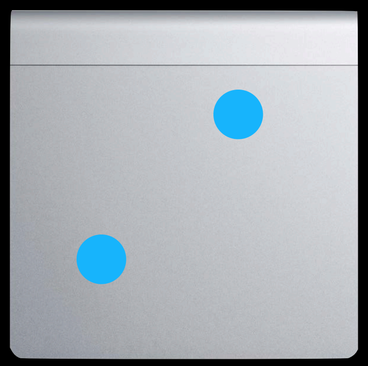
Centering clip views

- Deprecated API

```
- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;
```

- Replacement API

```
- (NSRect)constrainBoundsRect:(NSRect)proposedBounds;
```



Magnification

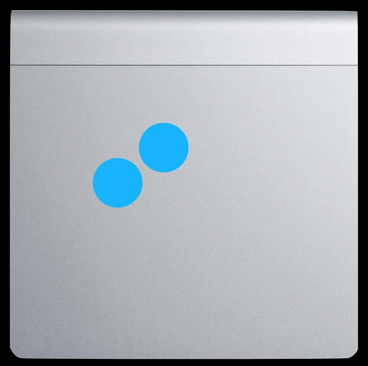
Centering clip views

- Deprecated API

~~- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;~~

- Replacement API

```
- (NSRect)constrainBoundsRect:(NSRect)proposedBounds;
```



Magnification

Centering clip views

- Deprecated API

~~- (NSPoint)constrainScrollPoint:(NSPoint)newOrigin;~~

- Replacement API

```
- (NSRect)constrainBoundsRect:(NSRect)proposedBounds;
```

Conclusion

Optimizing AppKit Drawing

Layer-Backed View Drawing
with Core Animation

Responsive Scrolling

Magnification

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation
Core Animation Programming Guide
http://developer.apple.com/

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

Related Sessions

Best Practices for Cocoa Animation	Marina Wednesday 2:00PM	

Labs

NSTableView, NSView, and Cocoa Lab	Frameworks Lab A Thursday 10:15AM	
Cocoa Animations, Drawing, and Cocoa Lab	Frameworks Lab A Friday 9:00AM	

WWDC2013