

# View Controller Advancements for iOS8

Session 214

Bruce D. Nilo

Manager, UIKit Fundamentals





viewWillTransitionToSize:withTransitionCoordinator:

<UICoordinateSpace>

UITraitCollection

Condensing Bars

UIPresentationController

UIUserInterfaceSizeClass

UISplitViewController

# Lots of new features

UIPopoverPresentationController

<UIContentContainer>

targetViewControllerForAction:sender:

UISearchController

UIAlertController

preferredContentSizeDidChangeForChildContainer:



splitViewController:shouldHideViewController:inOrientation:

willAnimateRotationToInterfaceOrientation:duration:

UIPopoverController

splitViewController:willHideController:withBarButtonItems:forPopoverController:

didRotateToInterfaceOrientation:

UIAlertView

UISearchDisplayController

InterfaceOrientation

UIActionSheet

willRotateToInterfaceOrientation:duration:



Some familiar APIs  
are getting retired

viewWillTransitionToSize:withTransitionCoordinator:

willRotateToInterfaceOrientation:duration:

interfaceOrientation

UIActionSheet

UIAlertView

UITraitCollection

<UICoordinateSpace>

Condensing Bars

UIPresentationController

UIUserInterfaceSizeClass

UISplitViewController

didRotateToInterfaceOrientation:

UISearchDisplayController

splitViewController:willHideController:withBarItem:forPopoverController:

UIPopoverPresentationController

<UIContentContainer>

targetViewControllerForAction:sender:

UIPopoverController

willAnimateRotationToInterfaceOrientation:duration:

UISearchController

UIAlertController

splitViewController:shouldHideViewControllers:inInterfaceOrientation:

preferredContentSizeDidChangeForChildContainer:

viewWillTransitionToSize:withTransitionCoordinator:

willRotateToInterfaceOrientation:duration:

interfaceOrientation

UIActionSheet

UIAlertView

UITraitCollection

<UICoordinateSpace>

Condensing Bars

UIPresentationController

UIUserInterfaceSizeClass

UISplitViewController

didRotateToInterfaceOrientation:

UISearchDisplayController

splitViewController:willHideController:withBarItem:forPopoverController:

UIPopoverPresentationController

<UIContentContainer>

targetViewControllerForAction:sender:

UIPopoverController

willAnimateRotationToInterfaceOrientation:duration:

UISearchController

UIAlertController

splitViewController:shouldHideViewControllers:inInterfaceOrientation:

preferredContentSizeDidChangeForChildContainer:

# Overview

A brief discussion about UIKit's new Adaptive APIs

New UISplitViewController features

Some new ways to condense and hide bars

Presentations and popovers

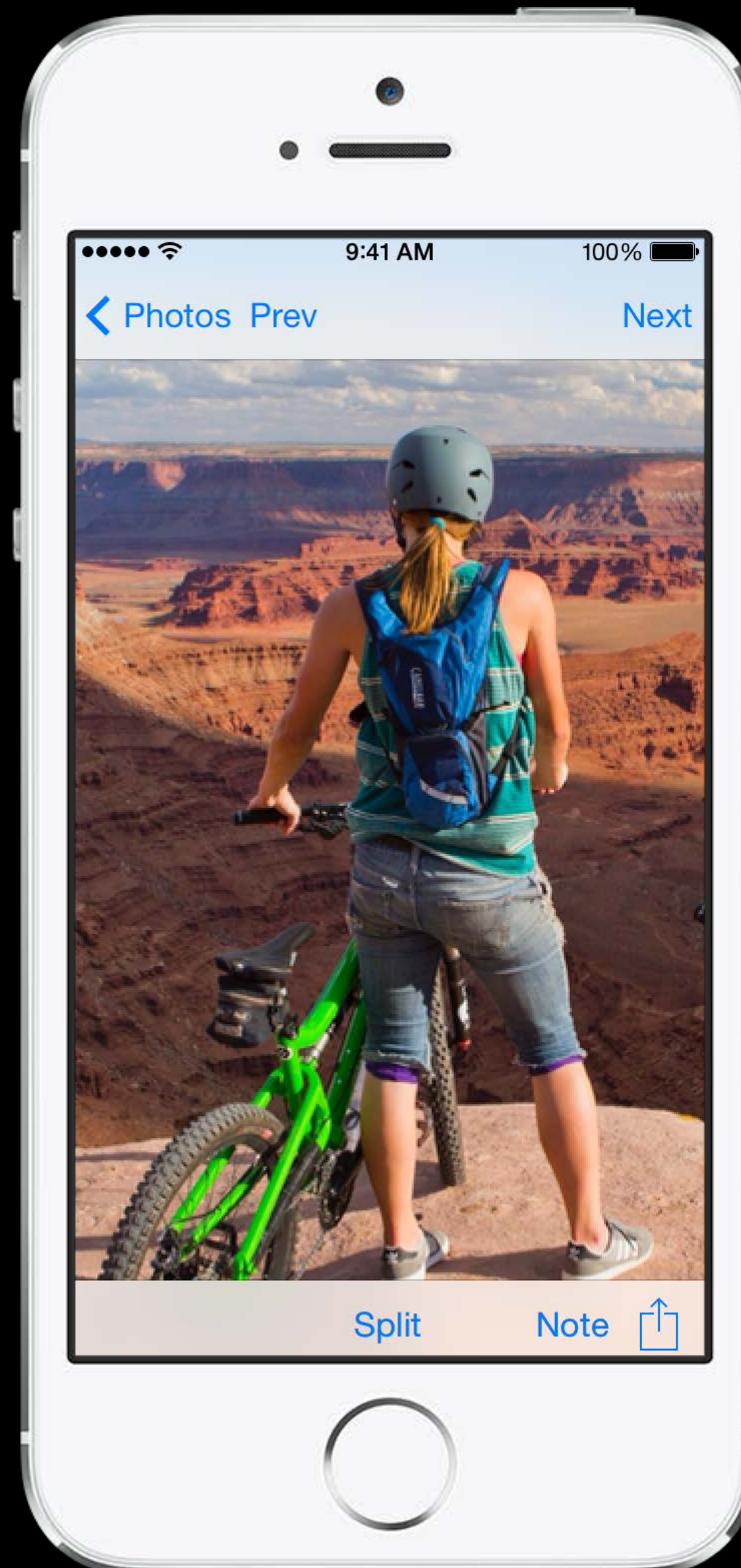
New API that uses transition coordinators

Coordinate spaces

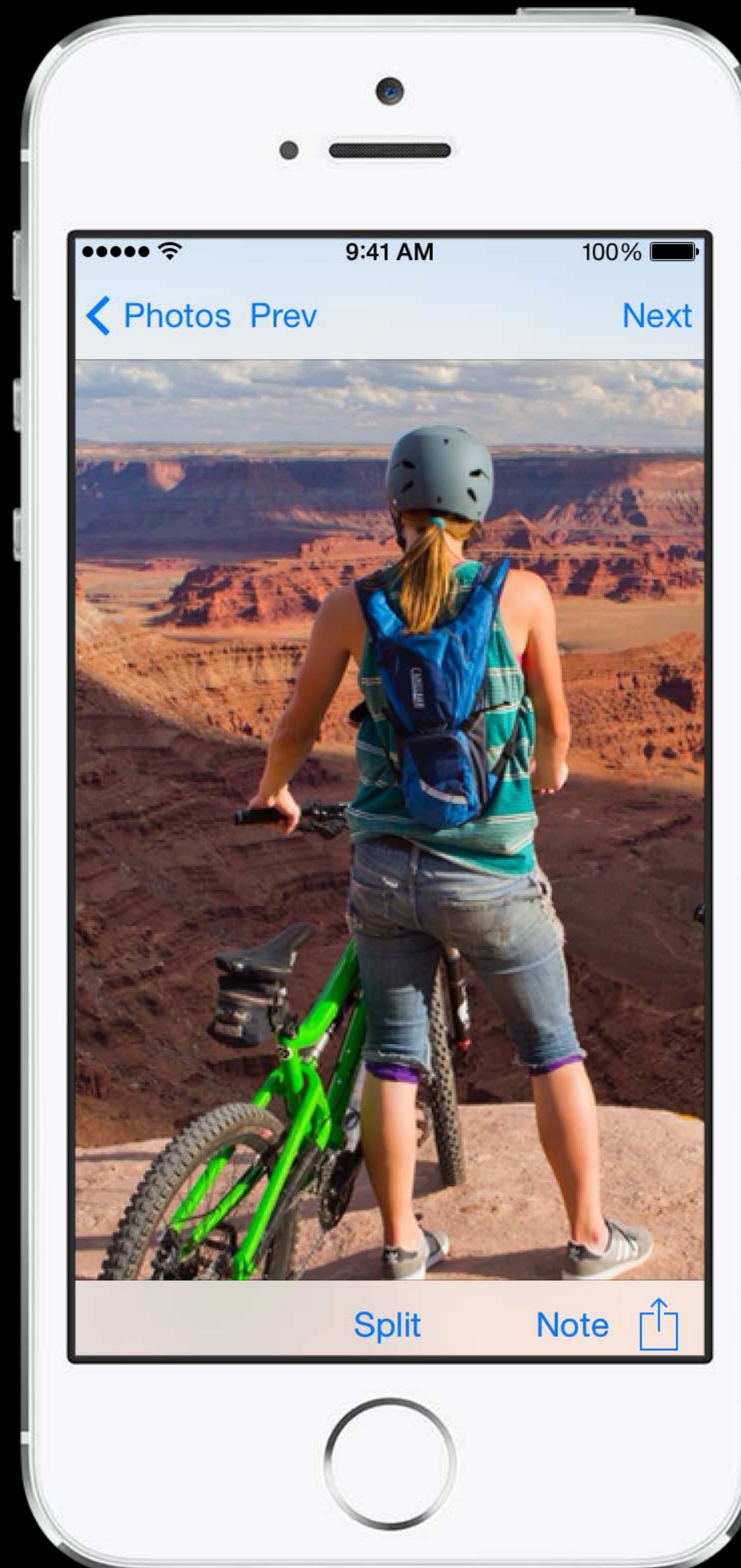


# Support for Adaptive User Interfaces

# Application Structure and Layout

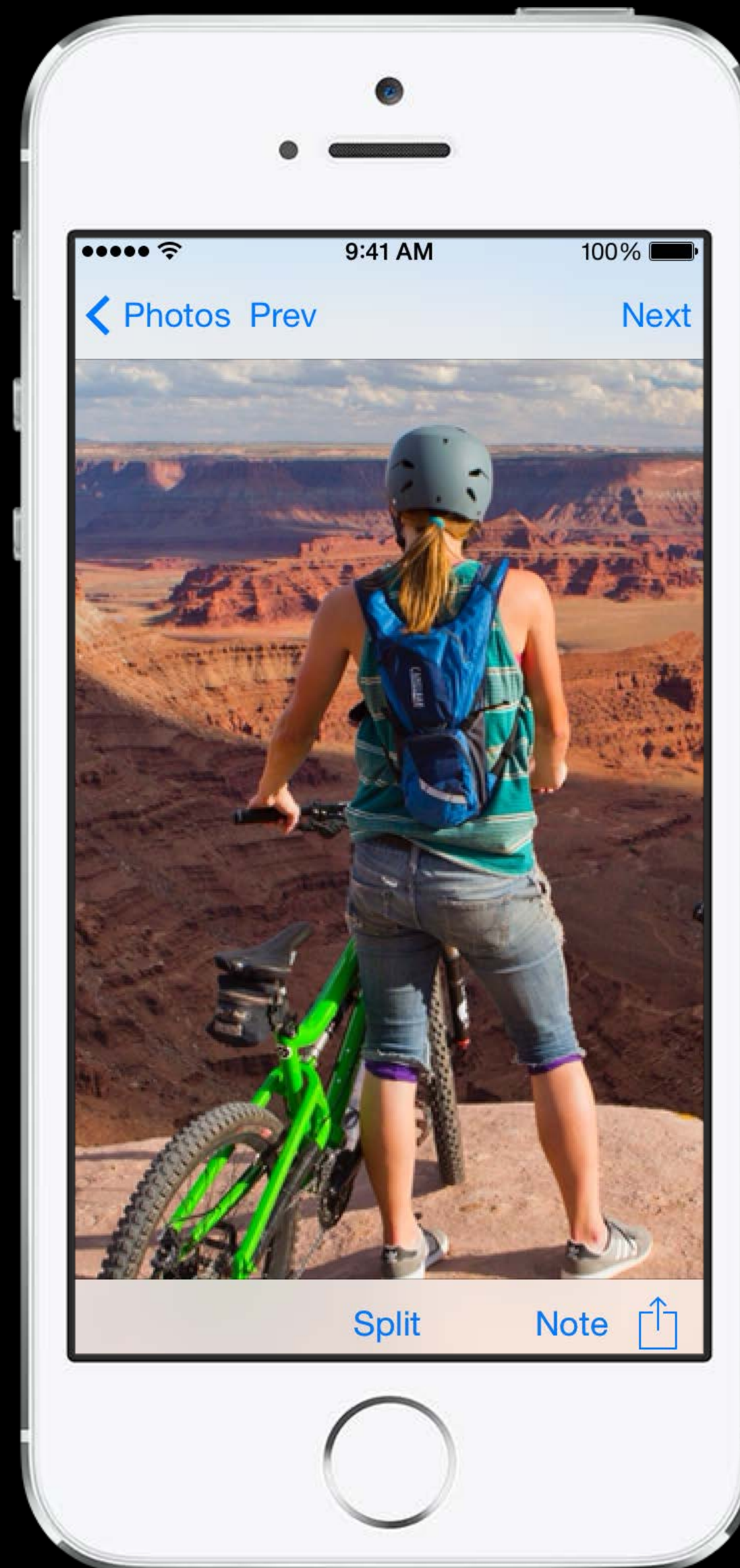


# Application Structure and Layout

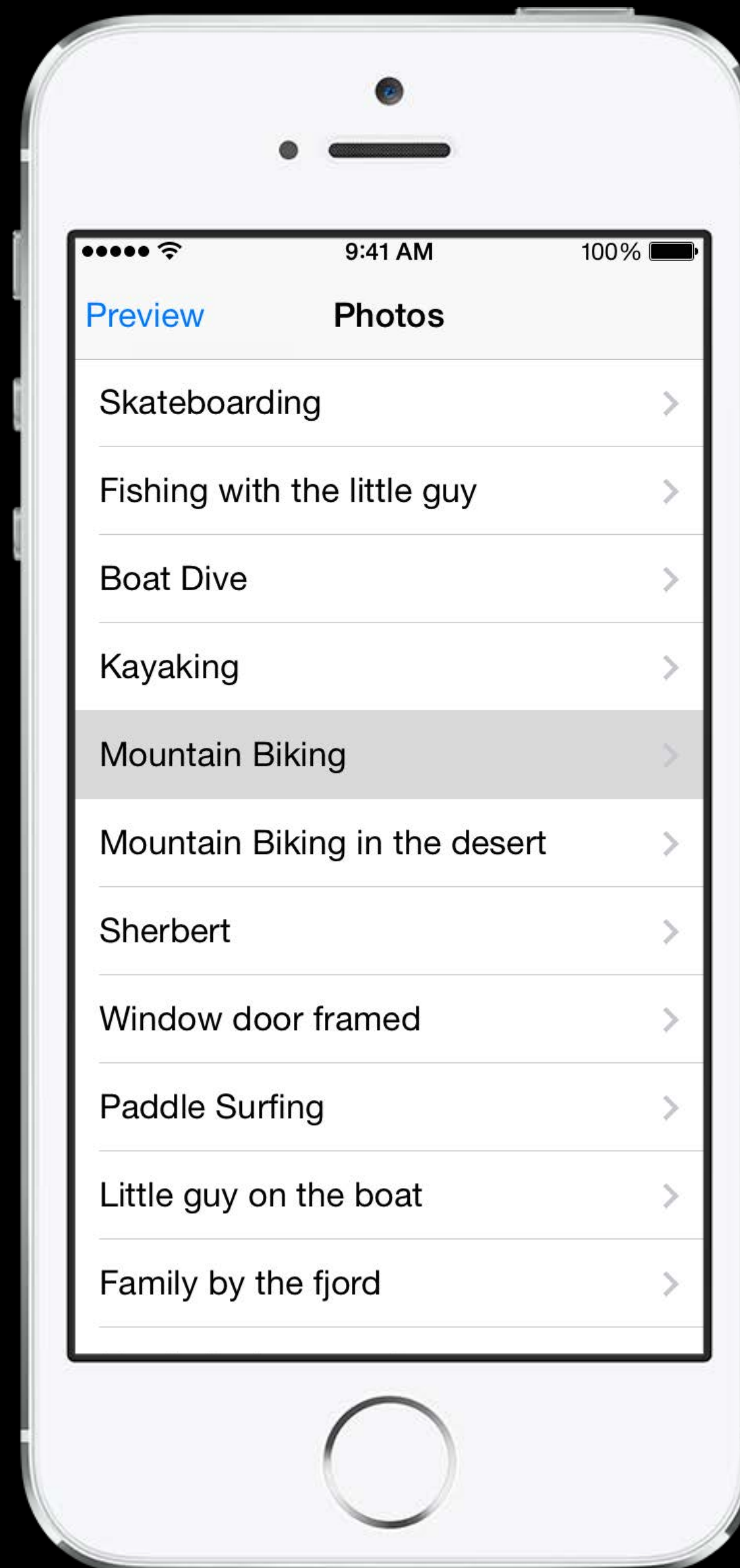




# Application Structure and Layout

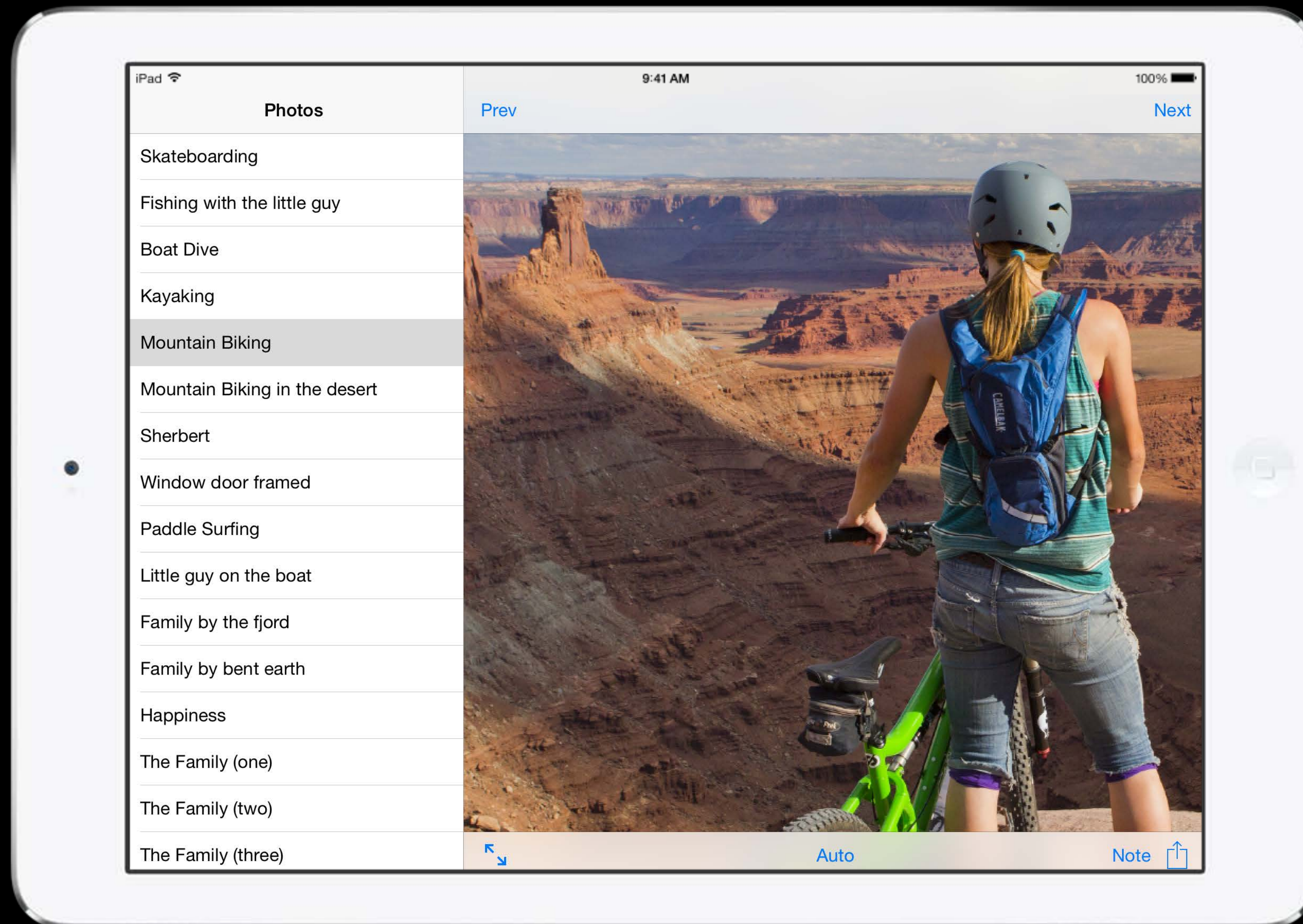


# Application Structure and Layout



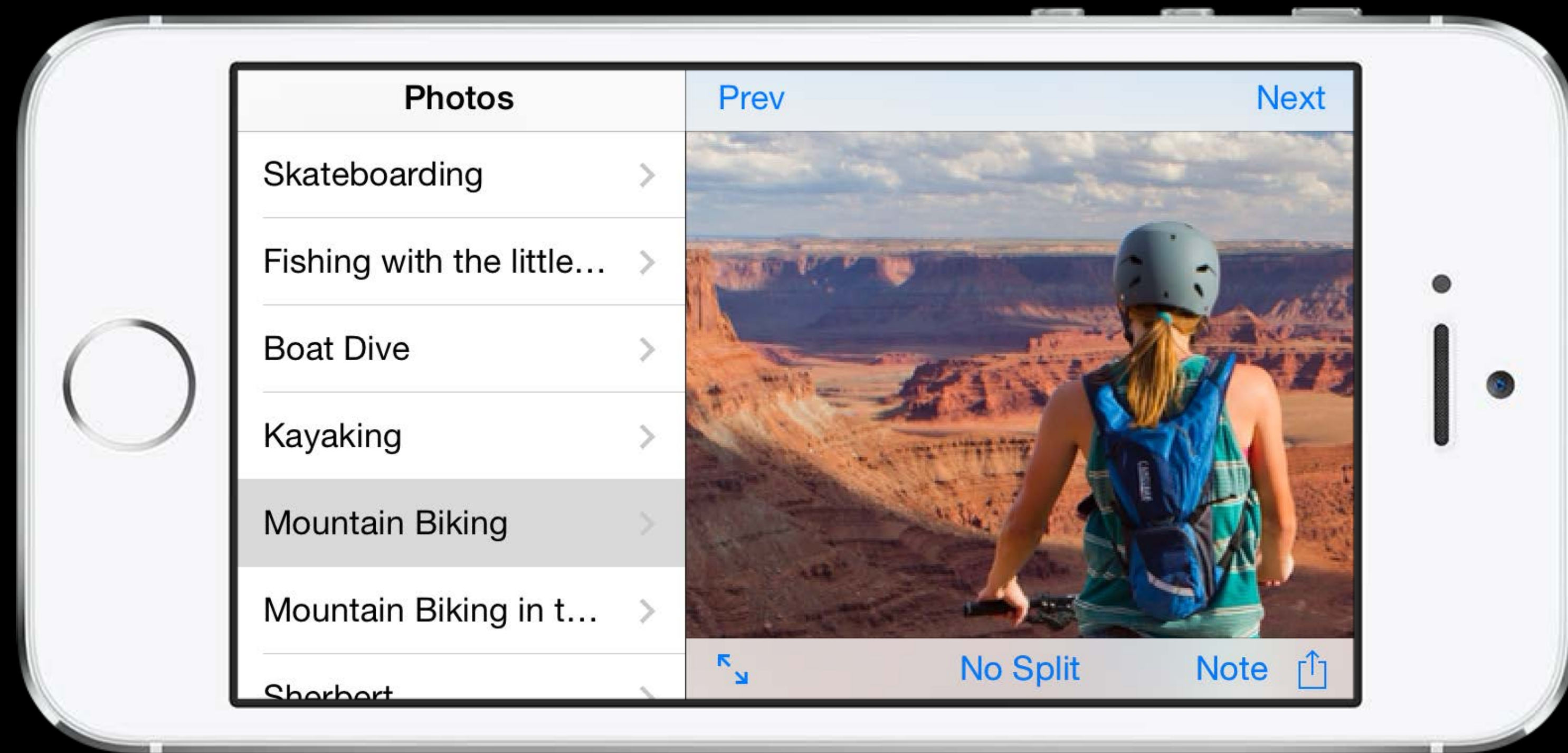


# Application Structure and Layout





# Application Structure and Layout



# Application Structure and Layout



# Application Structure and Layout

Before iOS 8

- Device type
- Interface Orientation
- Size

# Application Structure and Layout

## Before iOS 8

- Device type
- Interface Orientation
- Size

## iOS 8 and After

- Traits and trait collections
- Size

What's a "trait collection"?

What's a "trait collection"?

It's a bag of traits

# Trait Collections

A bag of traits

---

horizontalSizeClass **Compact**

---

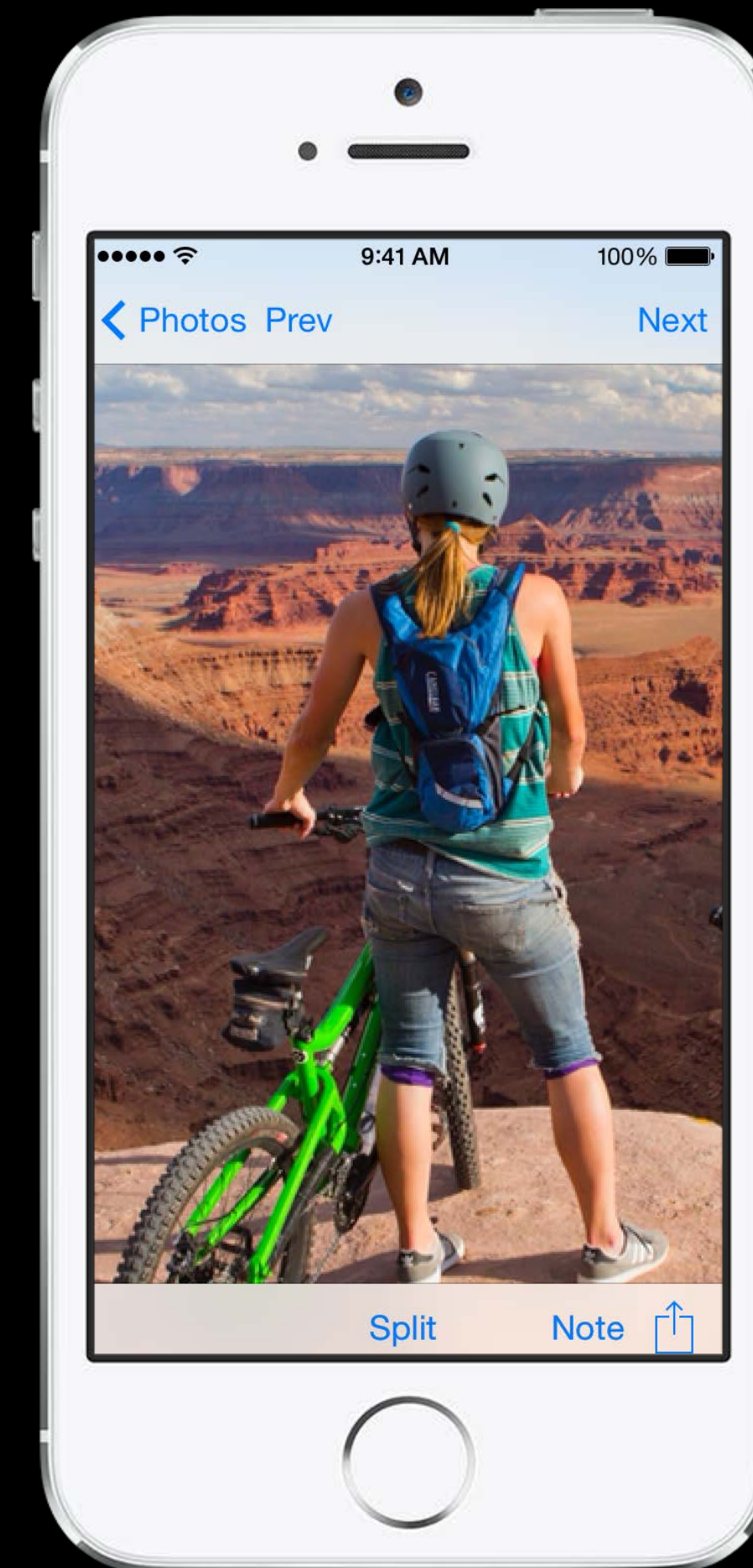
verticalSizeClass **Regular**

---

userInterfaceIdiom **Phone**

---

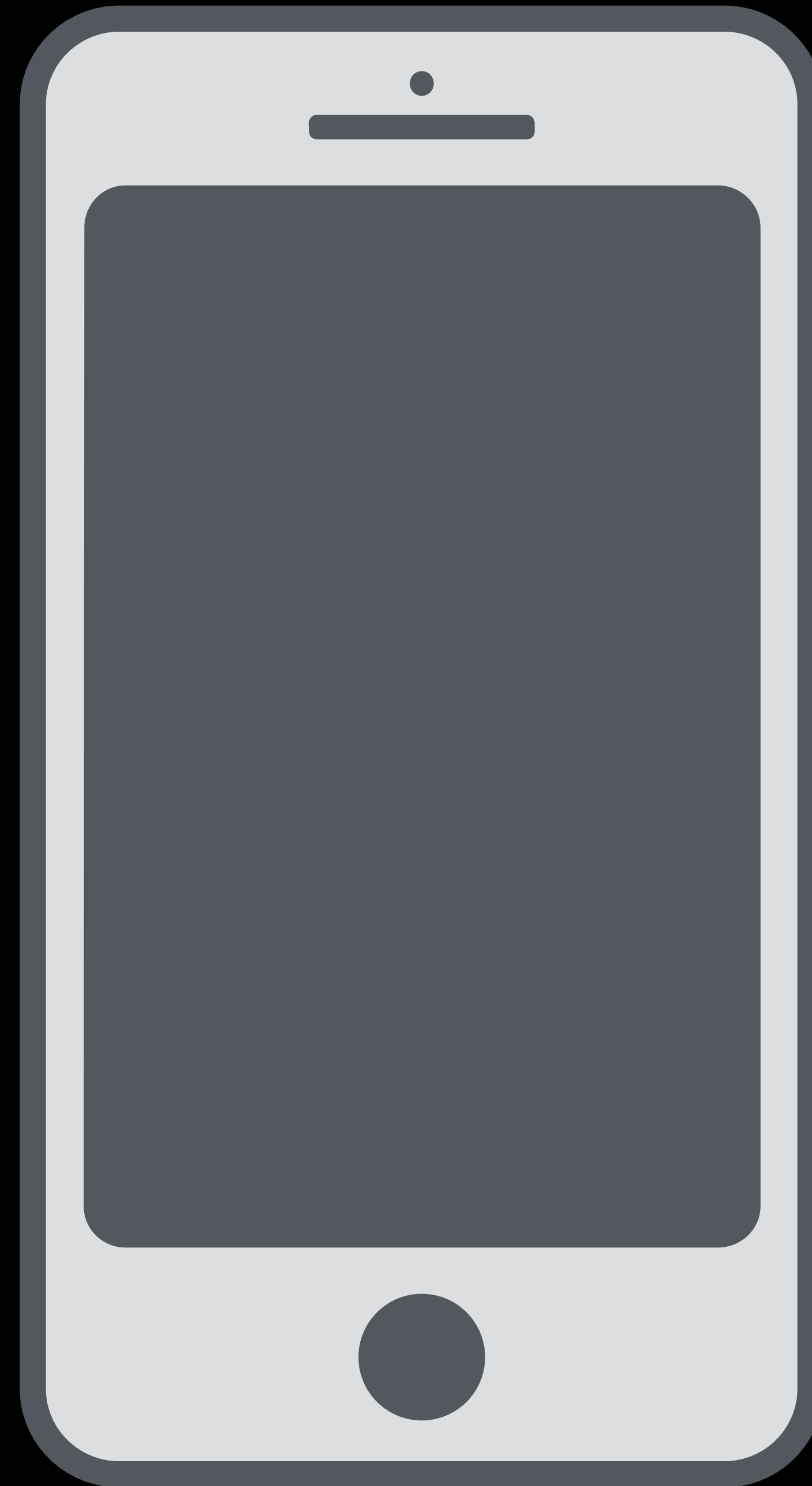
displayScale **2.0**



What's a "size class"?

What's a "size class"?

It's a trait that coarsely defines the space available

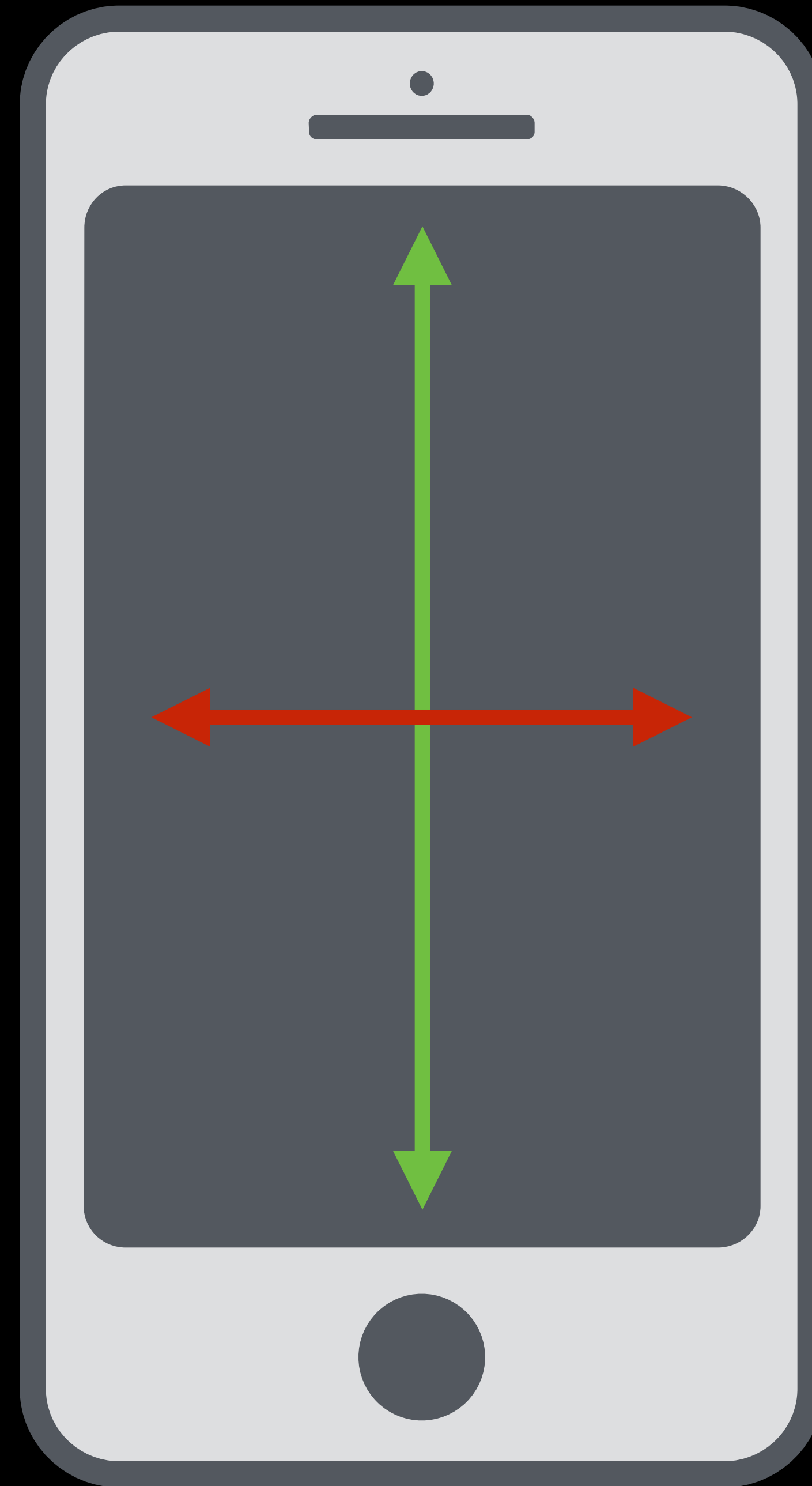




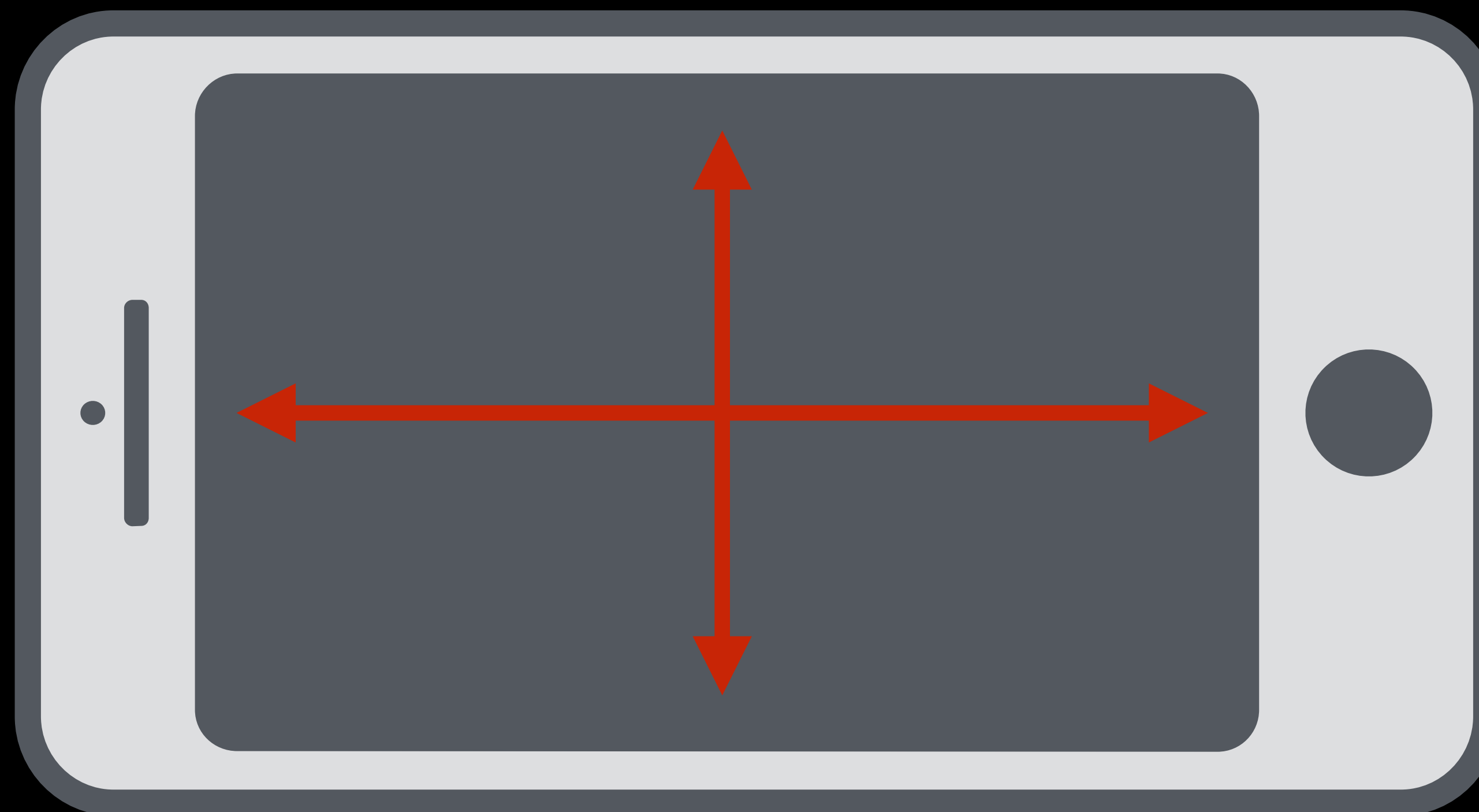
---

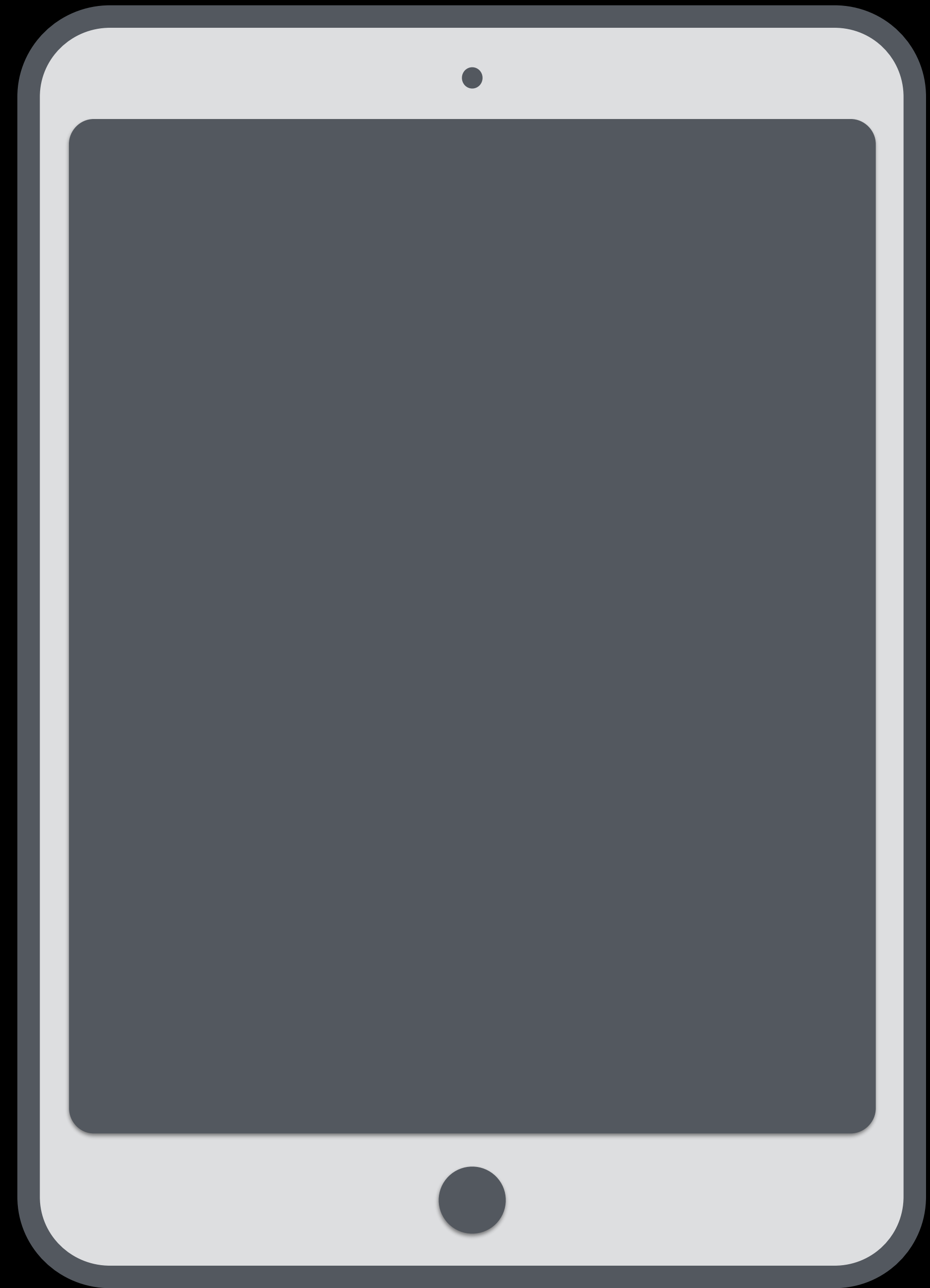
x-sizeClass	Compact
y-sizeClass	Regular
idiom	Phone
scale	2.0

---

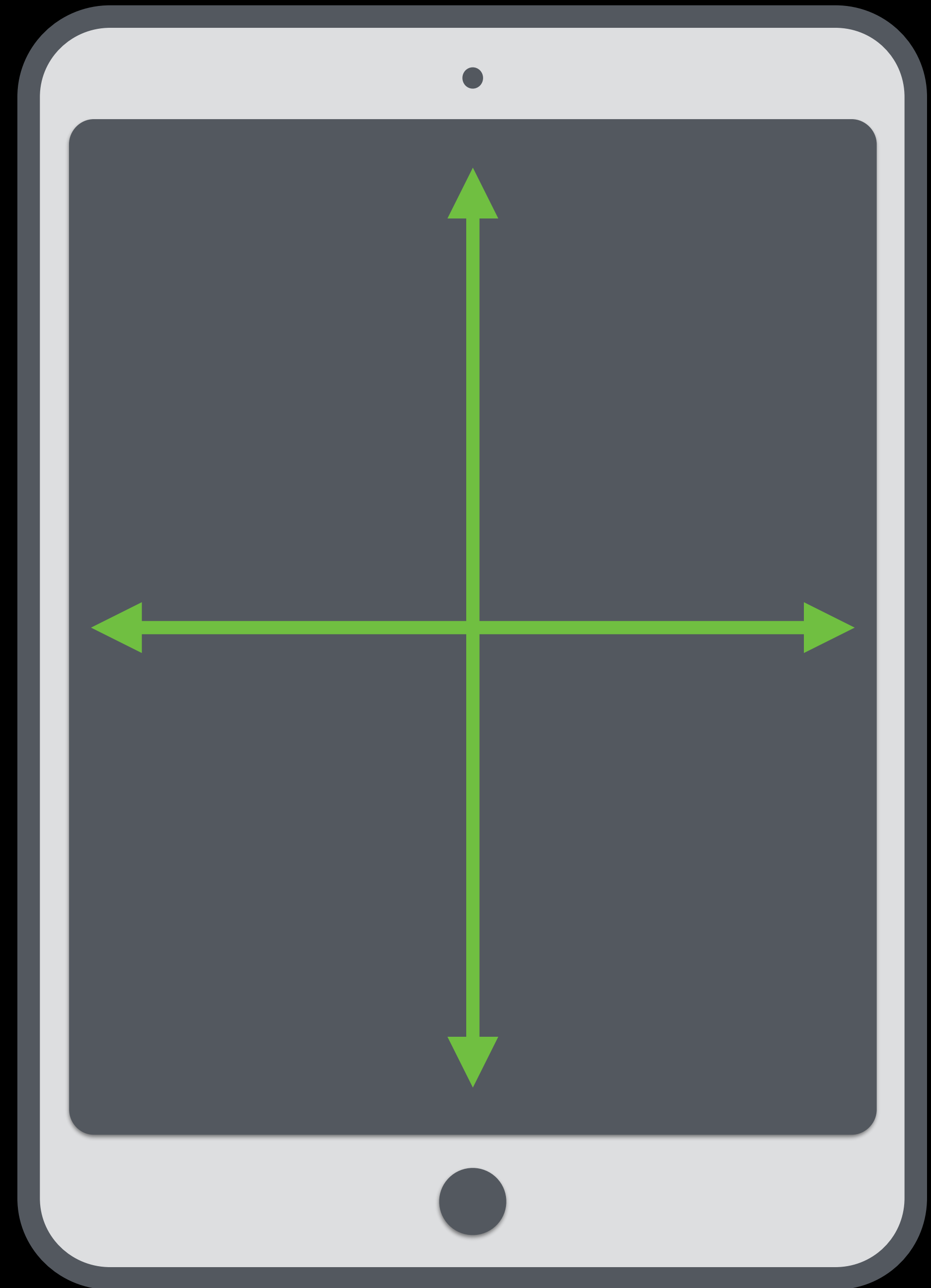


x-sizeClass	Compact
y-sizeClass	Compact
idiom	Phone
scale	2.0

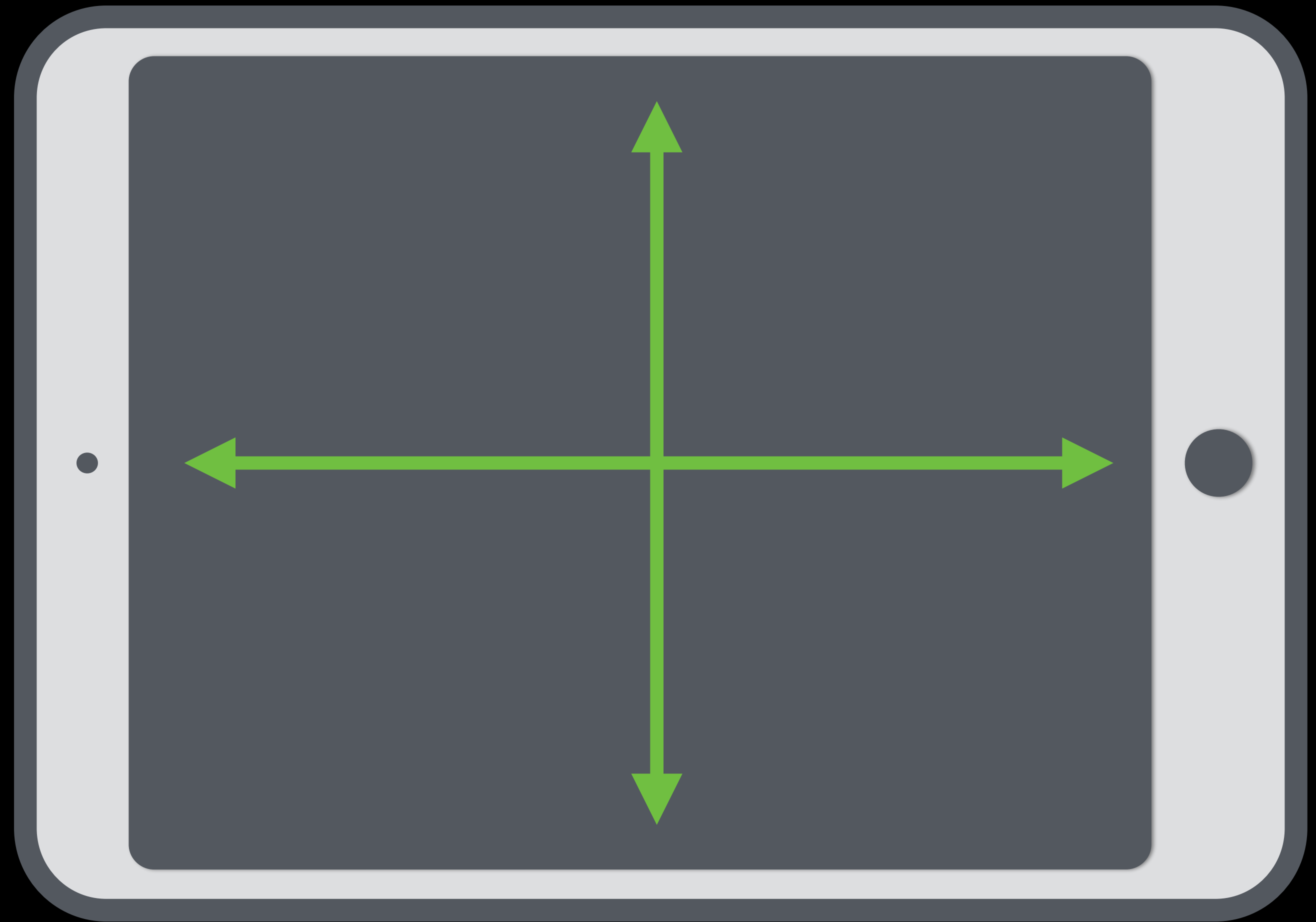




x-sizeClass	Regular
y-sizeClass	Regular
idiom	Pad
scale	1.0



x-sizeClass	Regular
y-sizeClass	Regular
idiom	Pad
scale	1.0



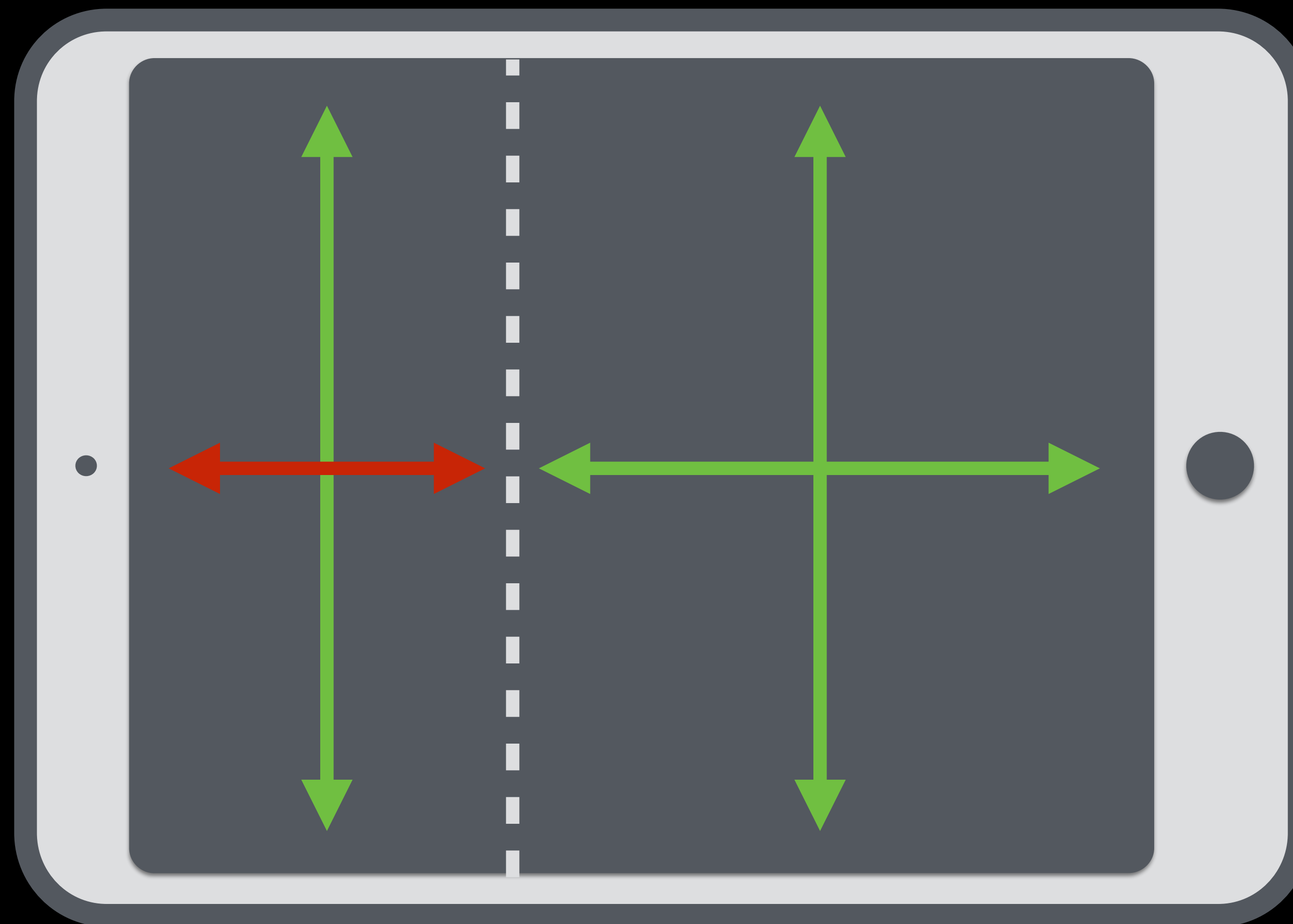
---

x-sizeClass	Regular
y-sizeClass	Regular
idiom	Pad
scale	1.0

---



x-sizeClass	Regular
y-sizeClass	Regular
idiom	Pad
scale	1.0
+	
x-sizeClass	Compact



# Application Structure and Layout



# Application Structure and Layout

A size class coarsely categorizes available space

- Horizontally
- Vertically

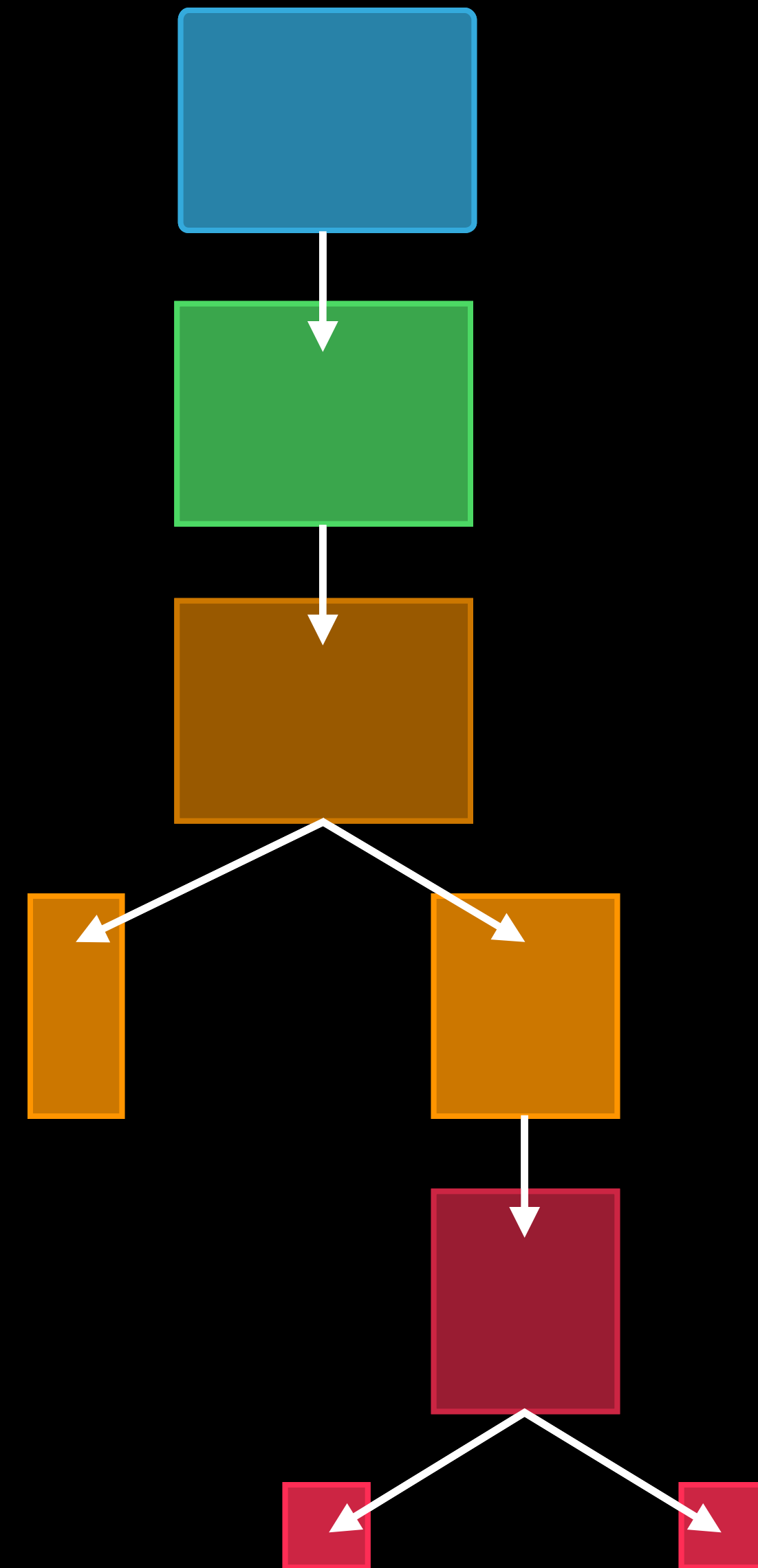
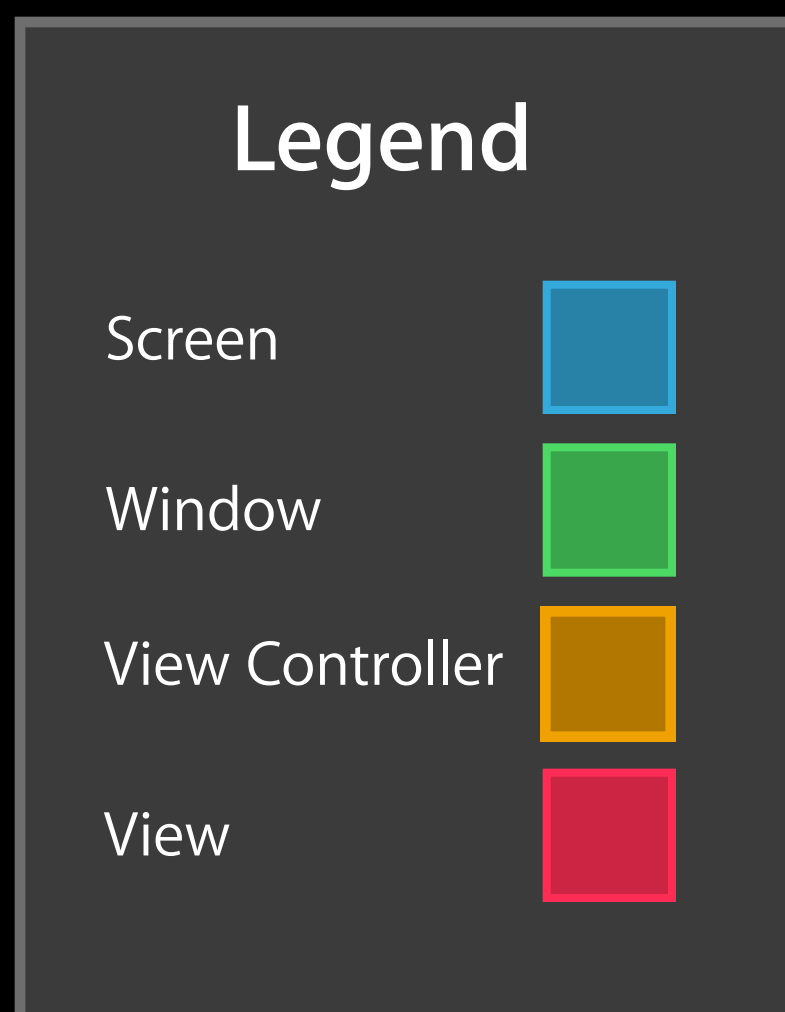
# Application Structure and Layout

A size class coarsely categorizes available space

- Horizontally
- Vertically

Trait collection's vendored from trait containers have both a horizontal and vertical size class trait

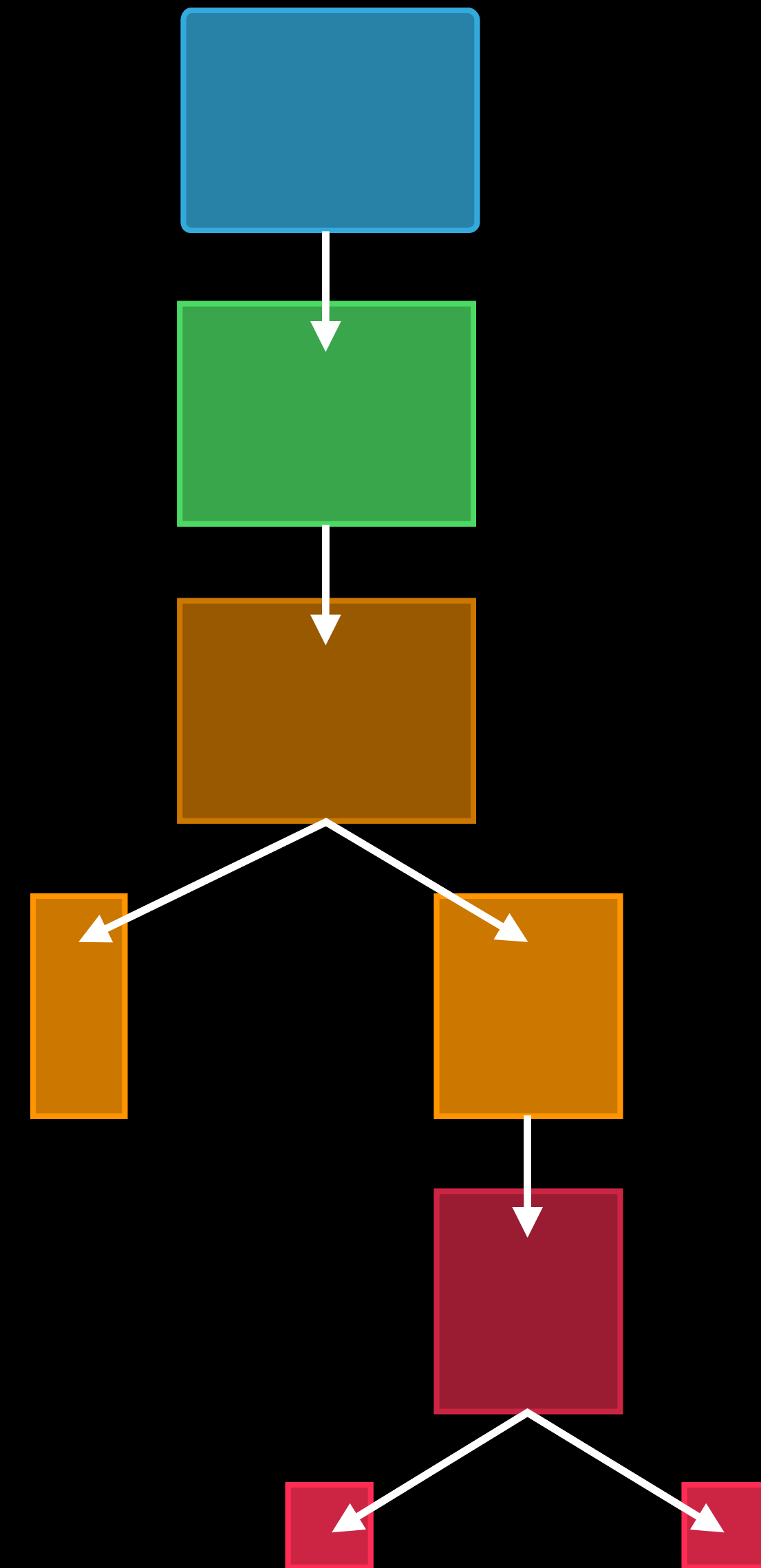
# Trait Environments



# Trait Environments

NEW

```
@protocol UITraitEnvironment <NSObject>  
  
@property UITraitCollection *traitCollection;  
  
- (void)traitCollectionDidChange:  
  
@end
```



## Legend

Screen



Window



View Controller



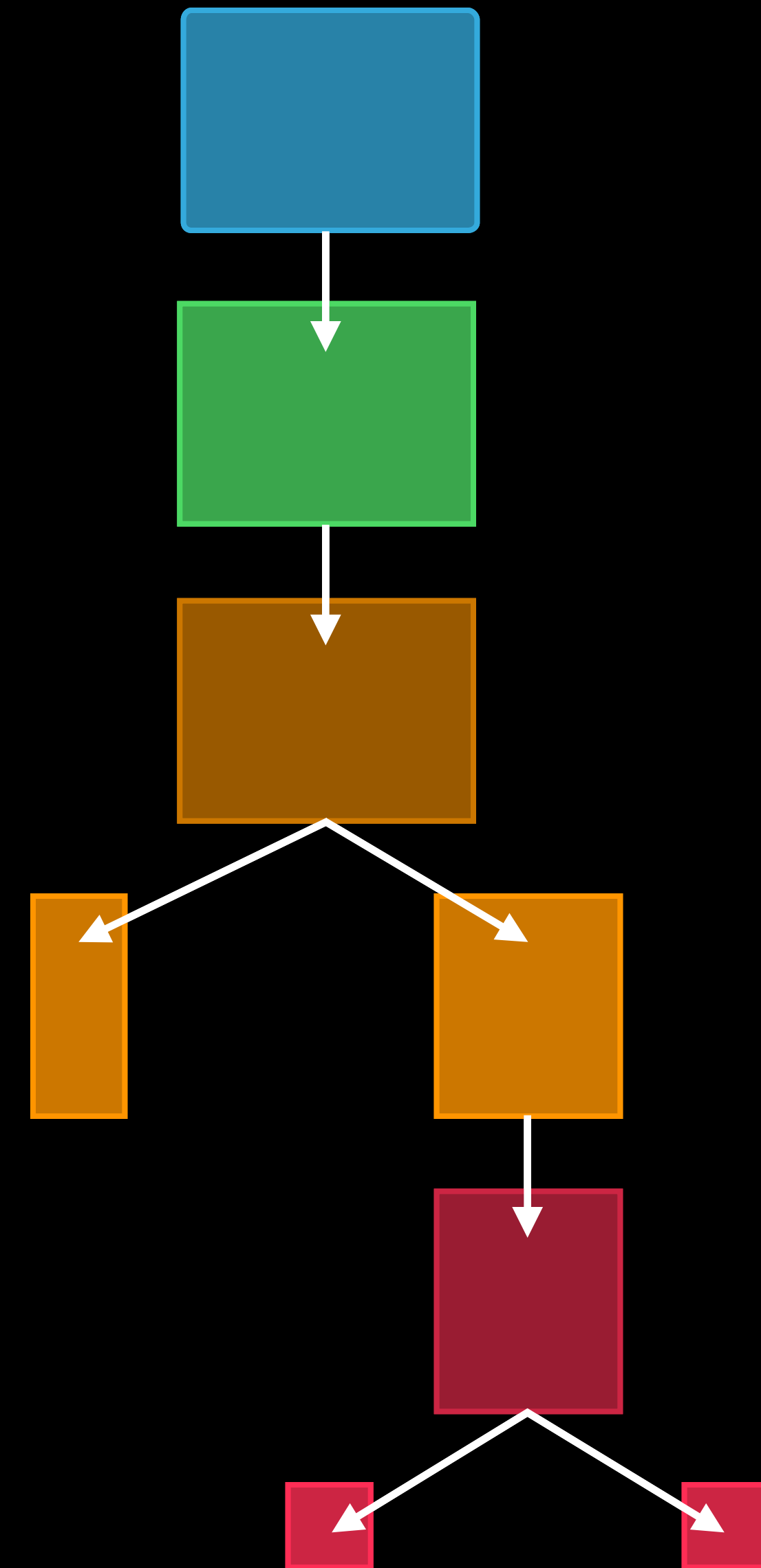
View



# Trait Environments

NEW

```
@protocol UITraitEnvironment <NSObject>  
  
@property UITraitCollection *traitCollection;  
  
- (void)traitCollectionDidChange:  
  
@end
```



## Legend

Screen



Window

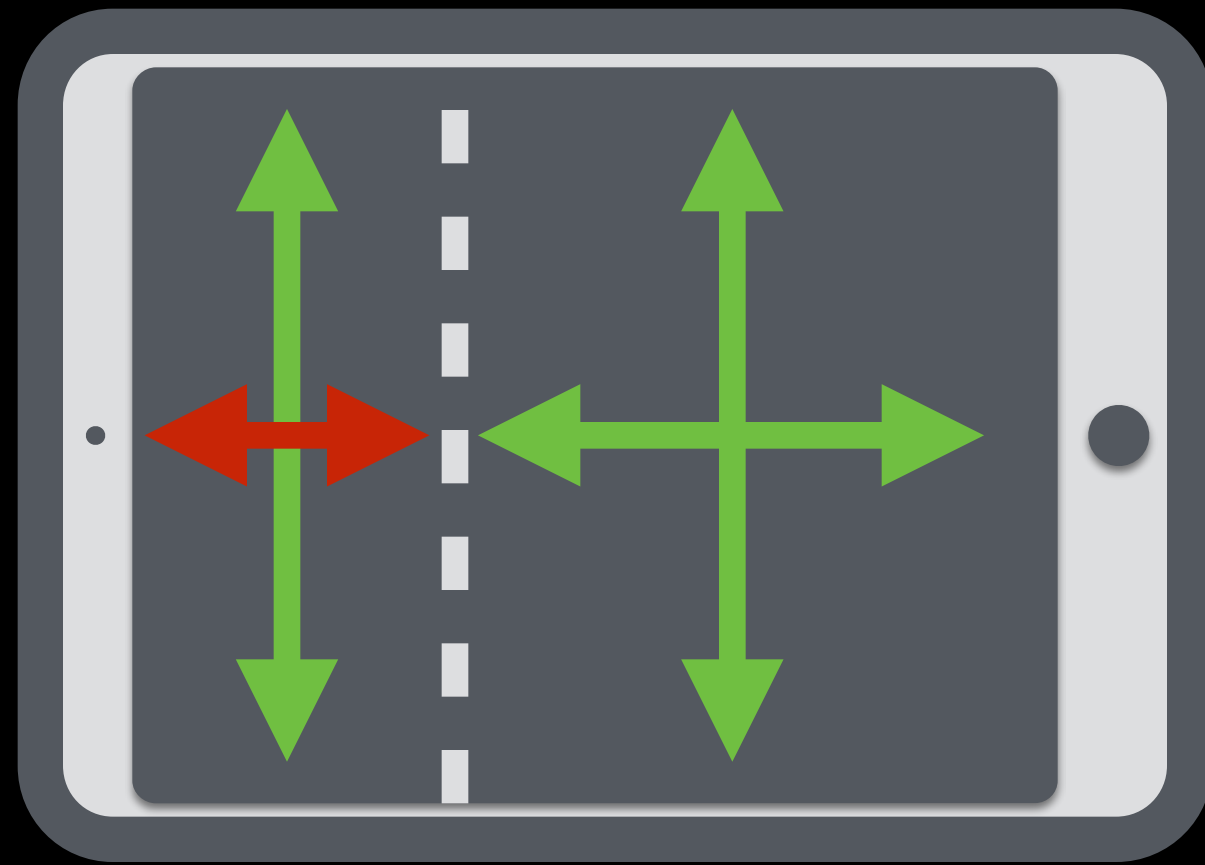


View Controller



View





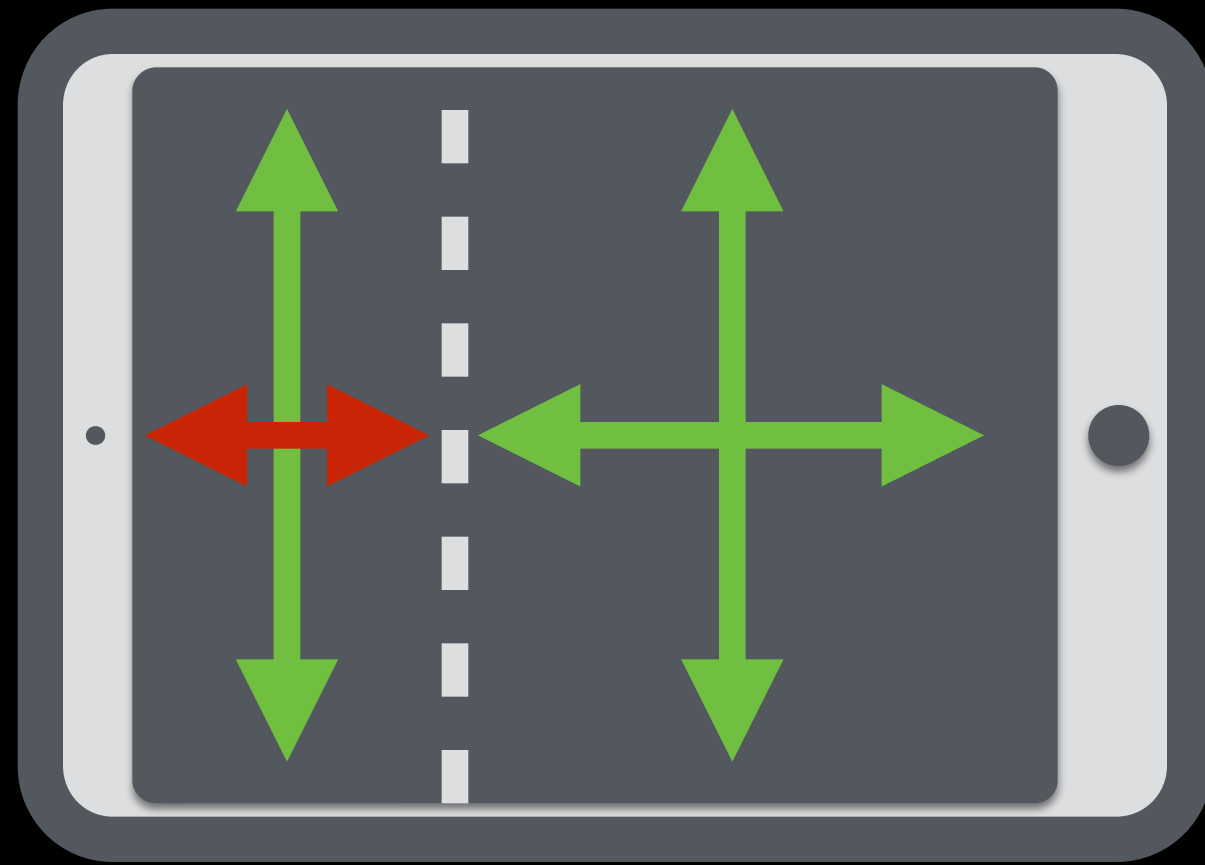
x-sizeClass	Regular
y-sizeClass	Regular
idiom	Pad
scale	1.0

+

x-sizeClass	Compact
-------------	---------

How does a parent view controller  
override the traits for a child?

NEW



x-sizeClass	Regular
-------------	---------

y-sizeClass	Regular
-------------	---------

idiom	Pad
-------	-----

scale	1.0
-------	-----

+

x-sizeClass	Compact
-------------	---------

```
@interface UIViewController <UITraitEnvironment>
```

```
- (void)setOverrideTraitCollection: forChildViewController:
```

```
- (UITraitCollection *)overrideTraitCollectionForChildViewController:
```

```
@end
```

*Demo*

UISplitViewController

DisplayMode and More



UISplitViewController

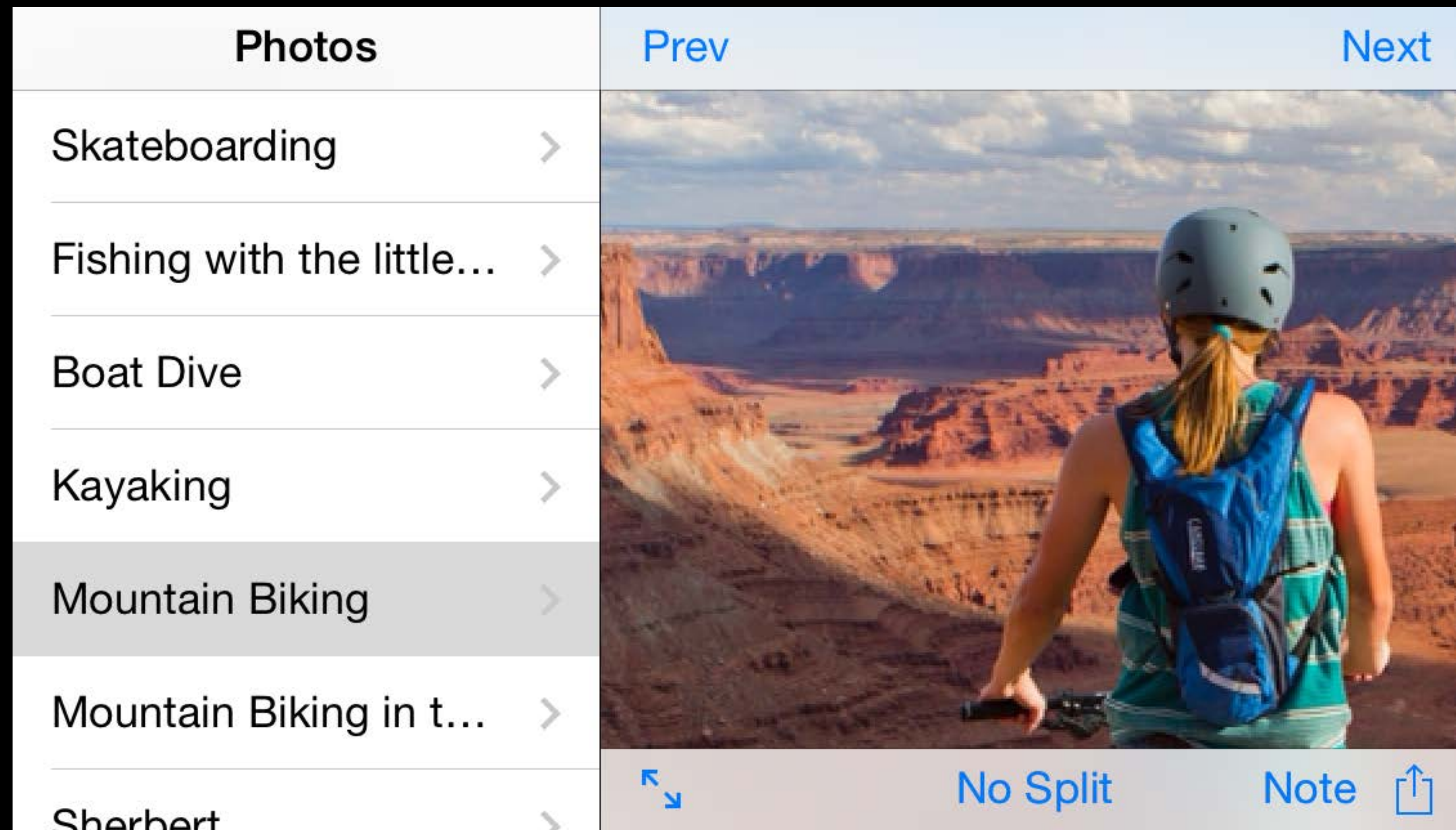
UISplitViewControllers can now  
be used on the phone

```
@interface UISplitViewController
```

```
@property(getter=isCollapsed) BOOL collapsed;
```

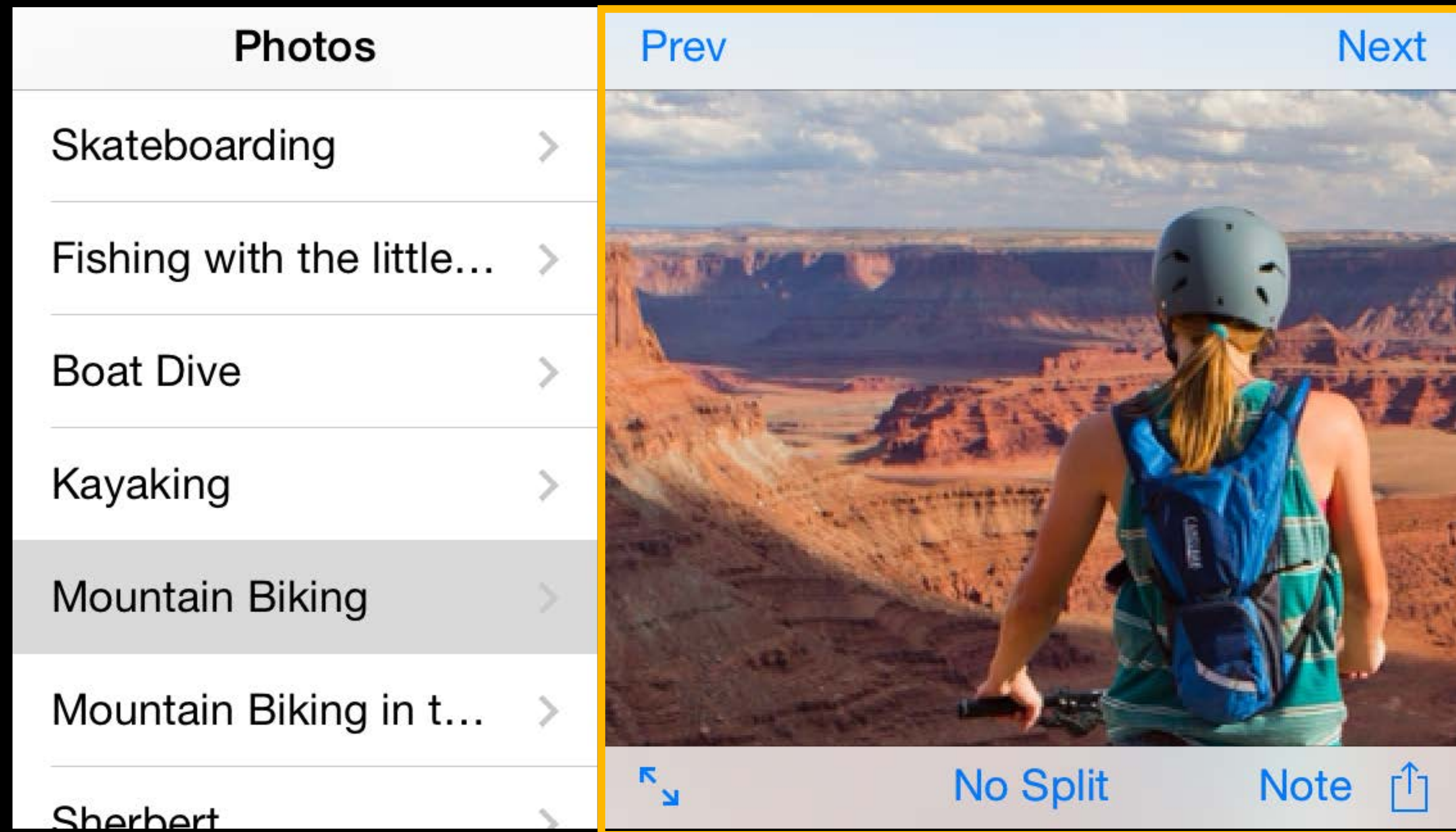
```
@end
```

# An expanded split view controller



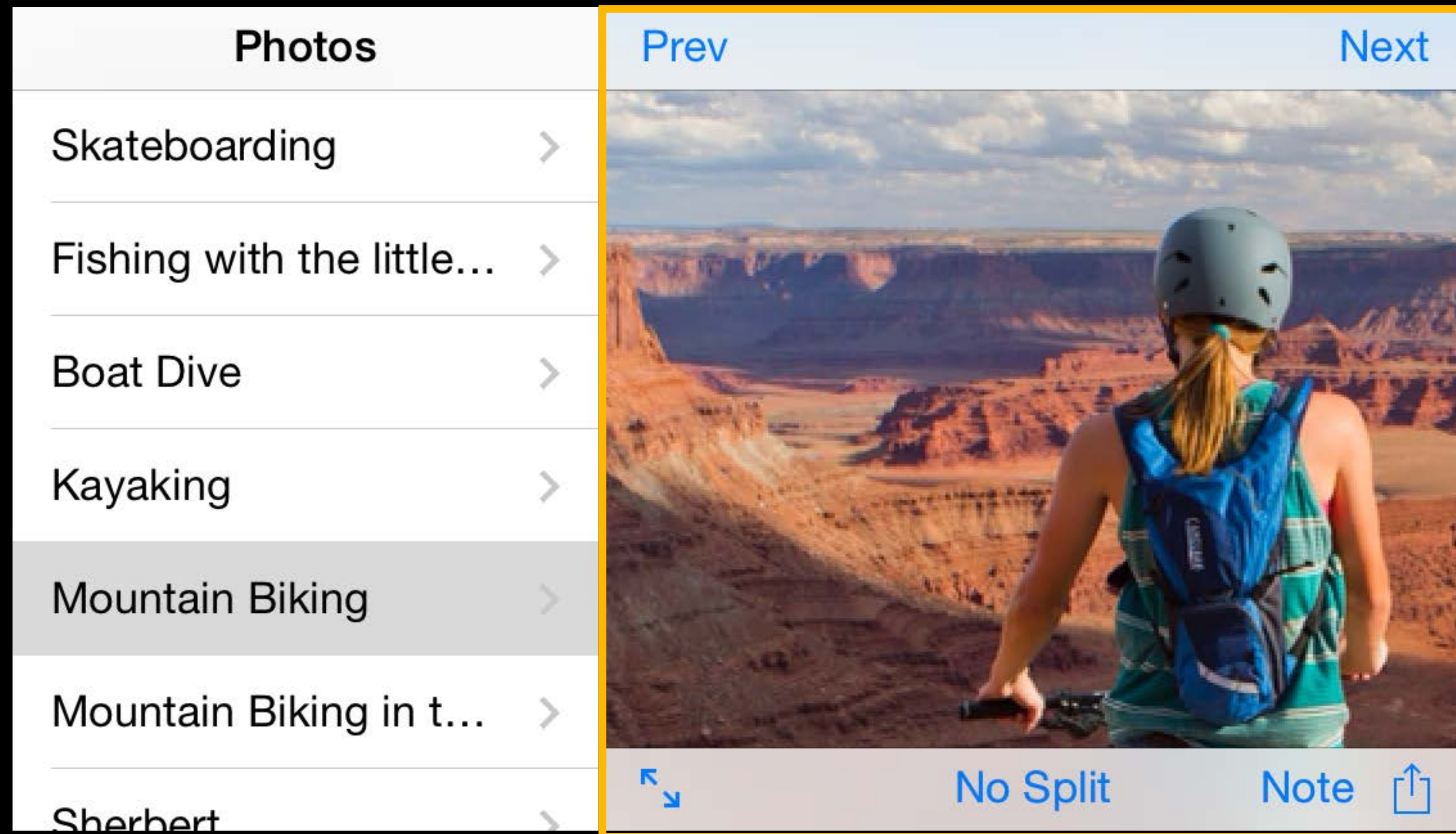


# An expanded split view controller



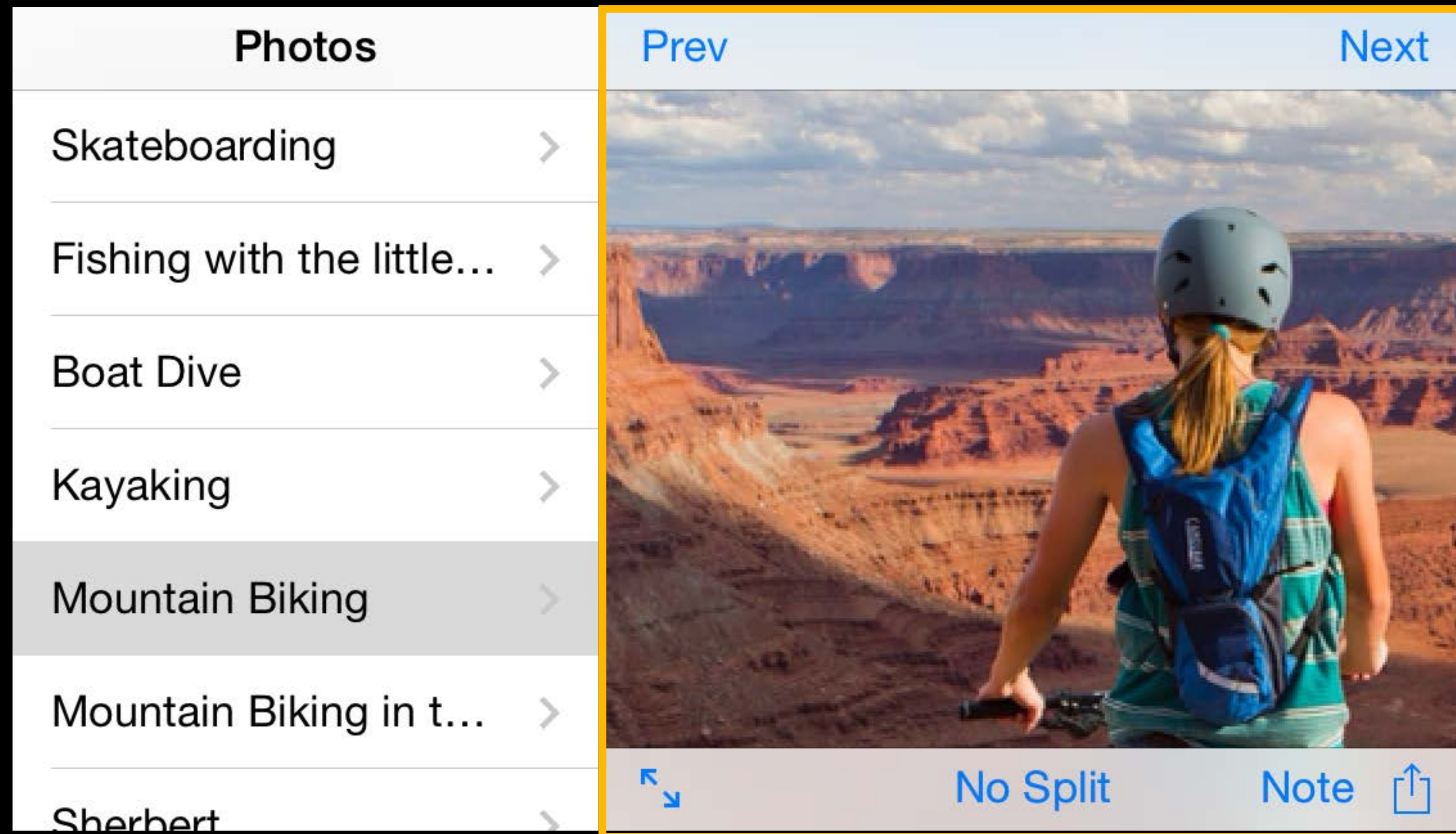


# An expanded split view controller



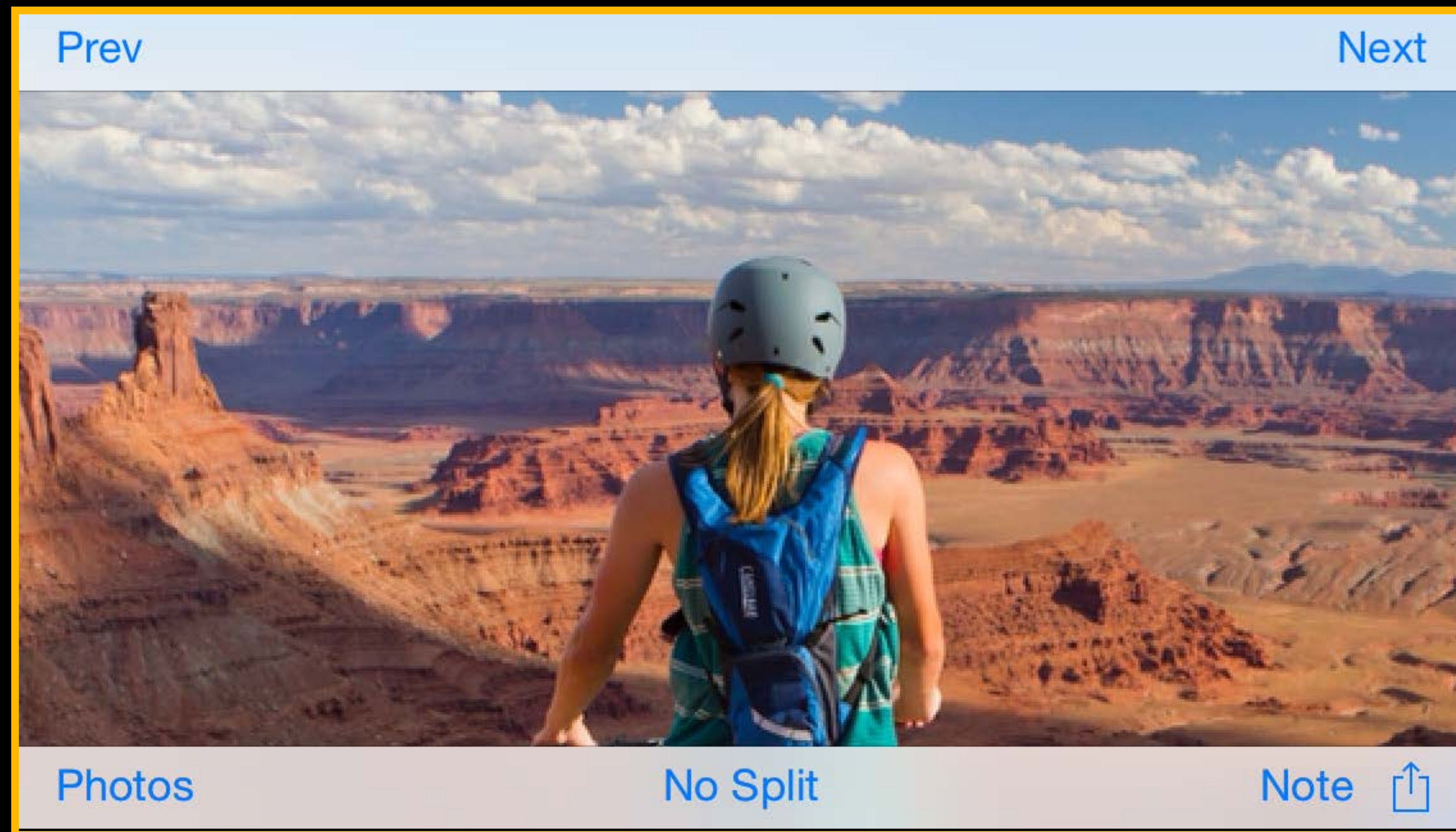


# An expanded split view controller



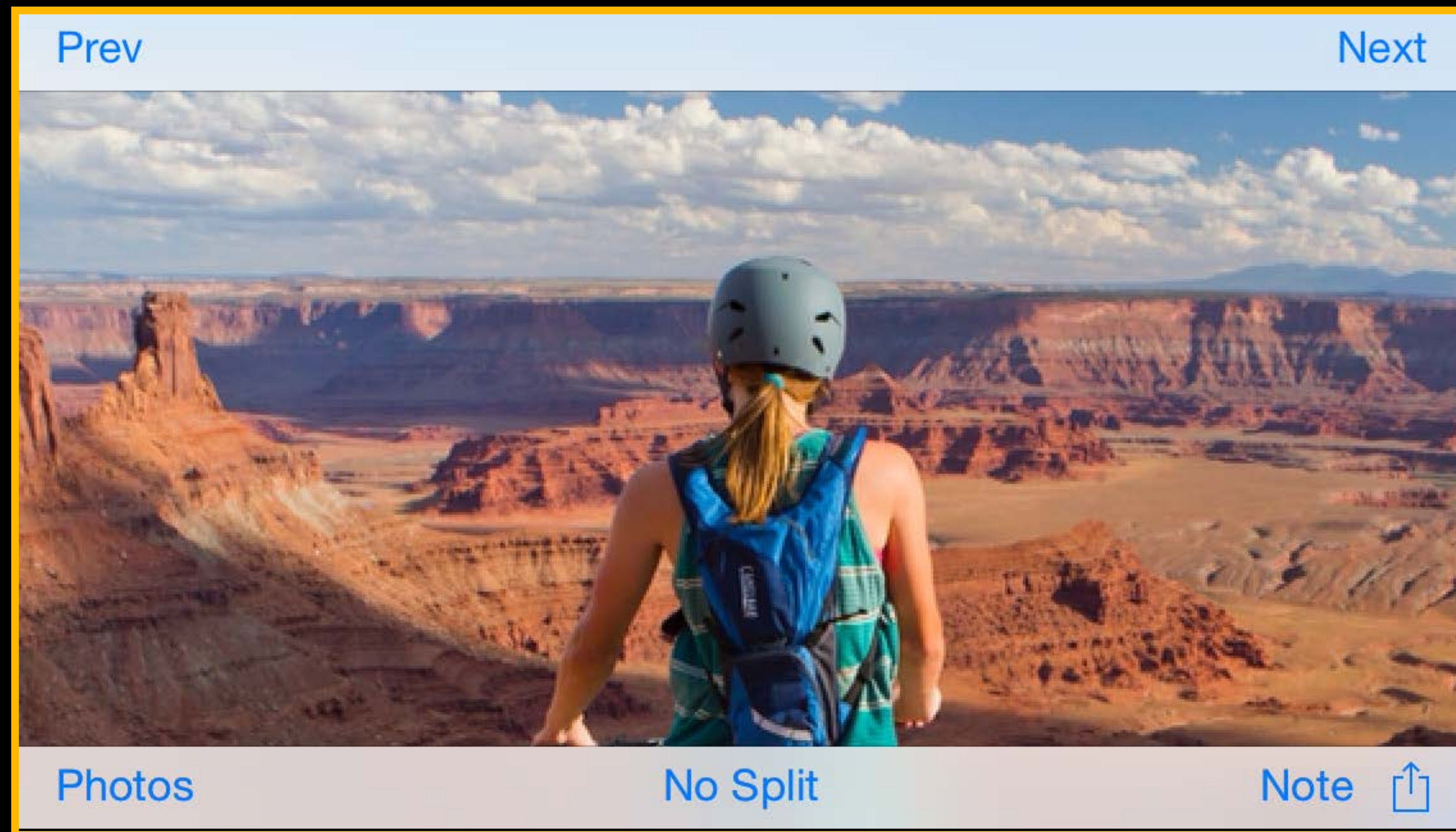


# An expanded split view controller



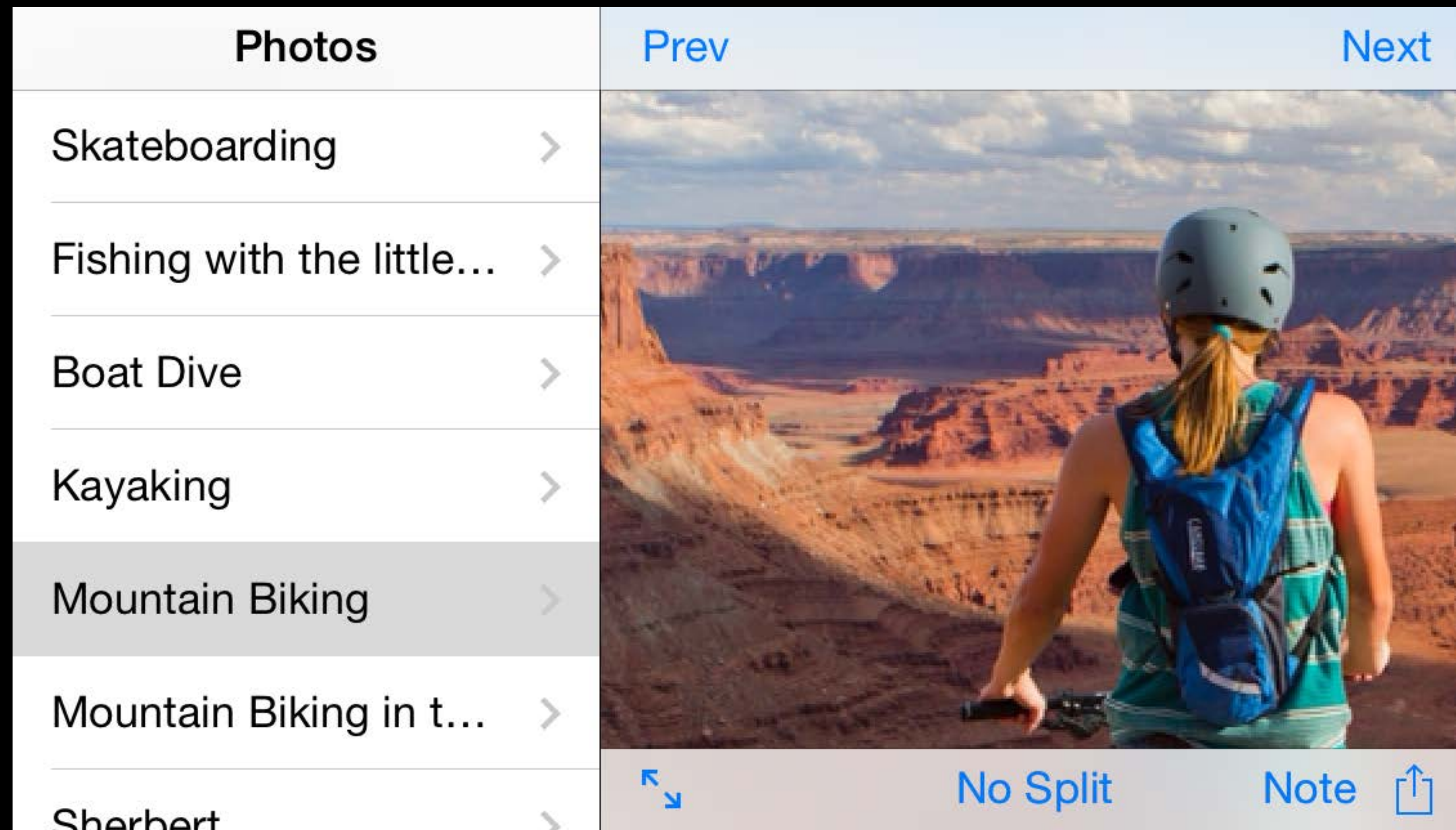


# An expanded split view controller



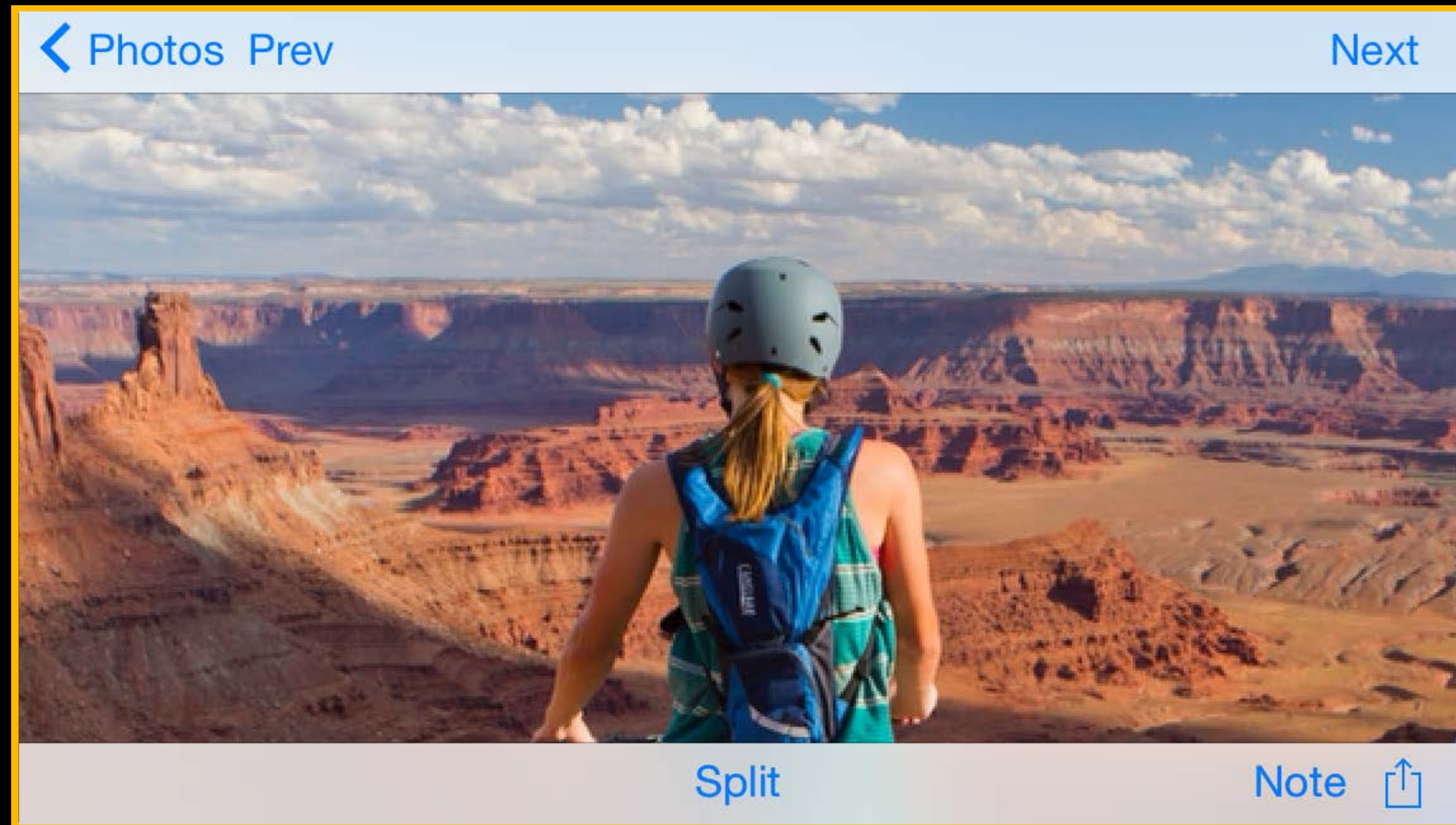


# An expanded split view controller





# A collapsed split view controller



# UISplitViewController

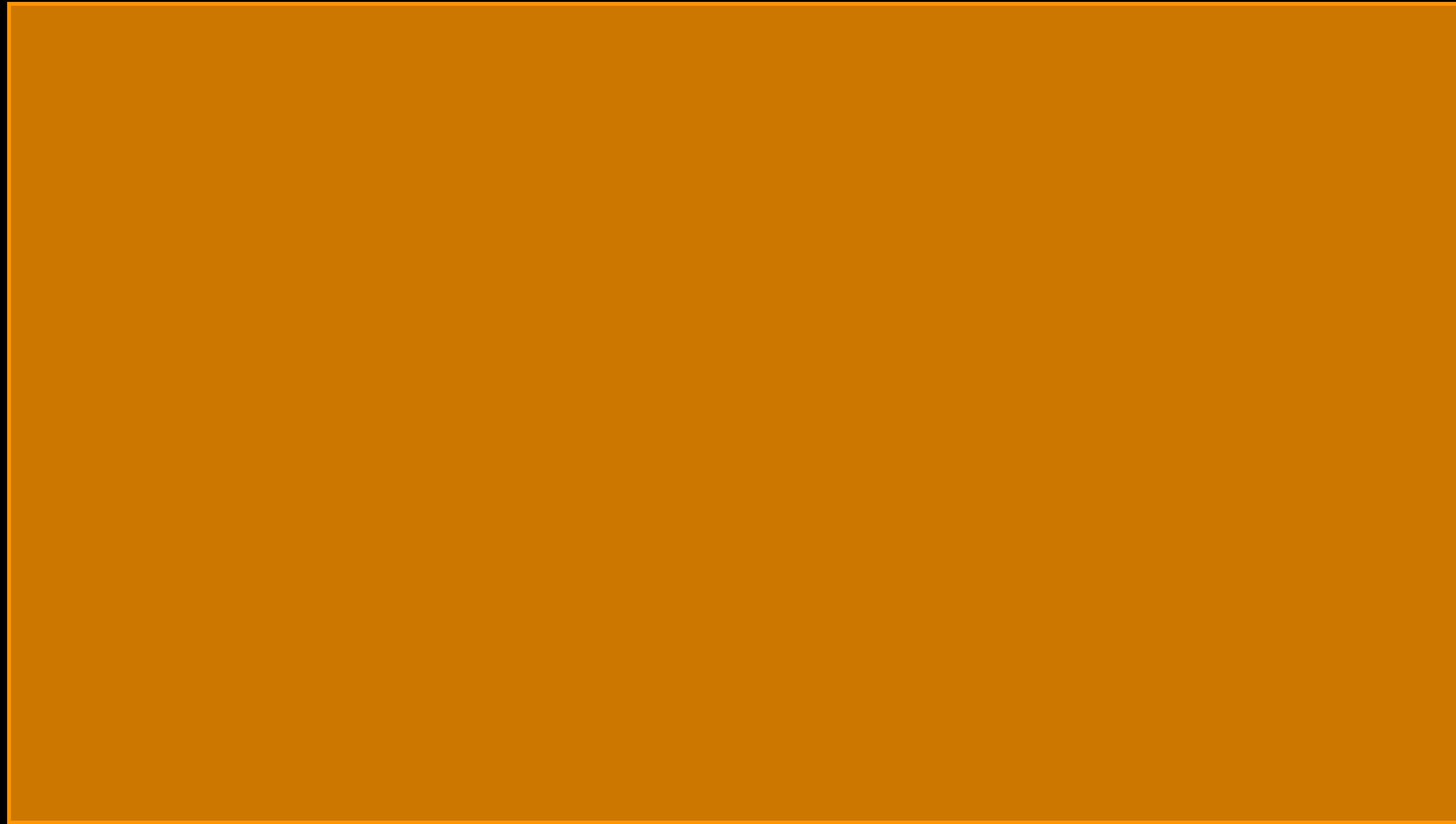
Split view controllers have a collapsed layout

- within horizontally compact containers
- e.g. phones
- otherwise they are expanded

# UISplitViewController

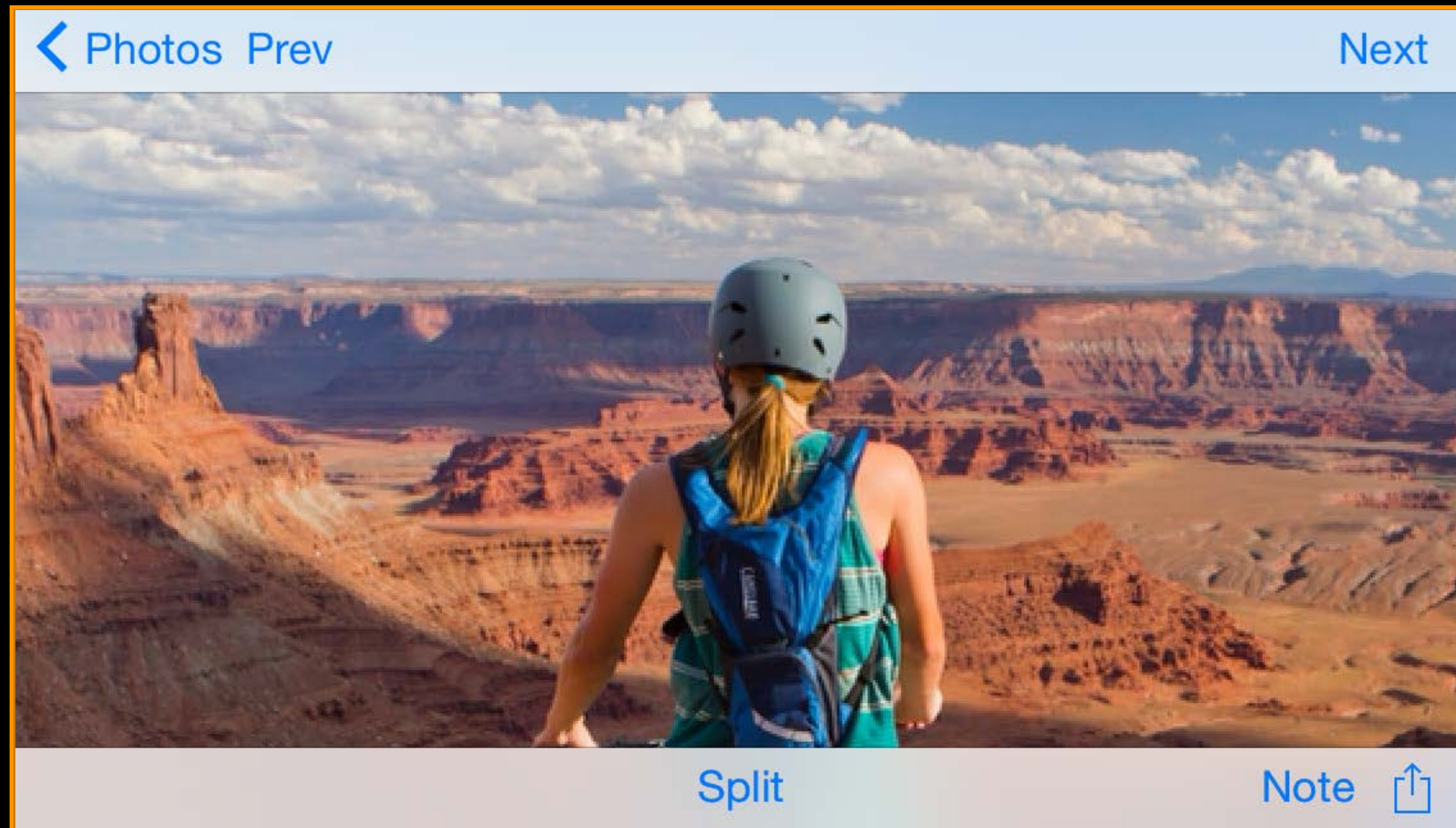
Can one enable an expanded split view controller on an iPhone?

Embed a UISVC inside a containerVC



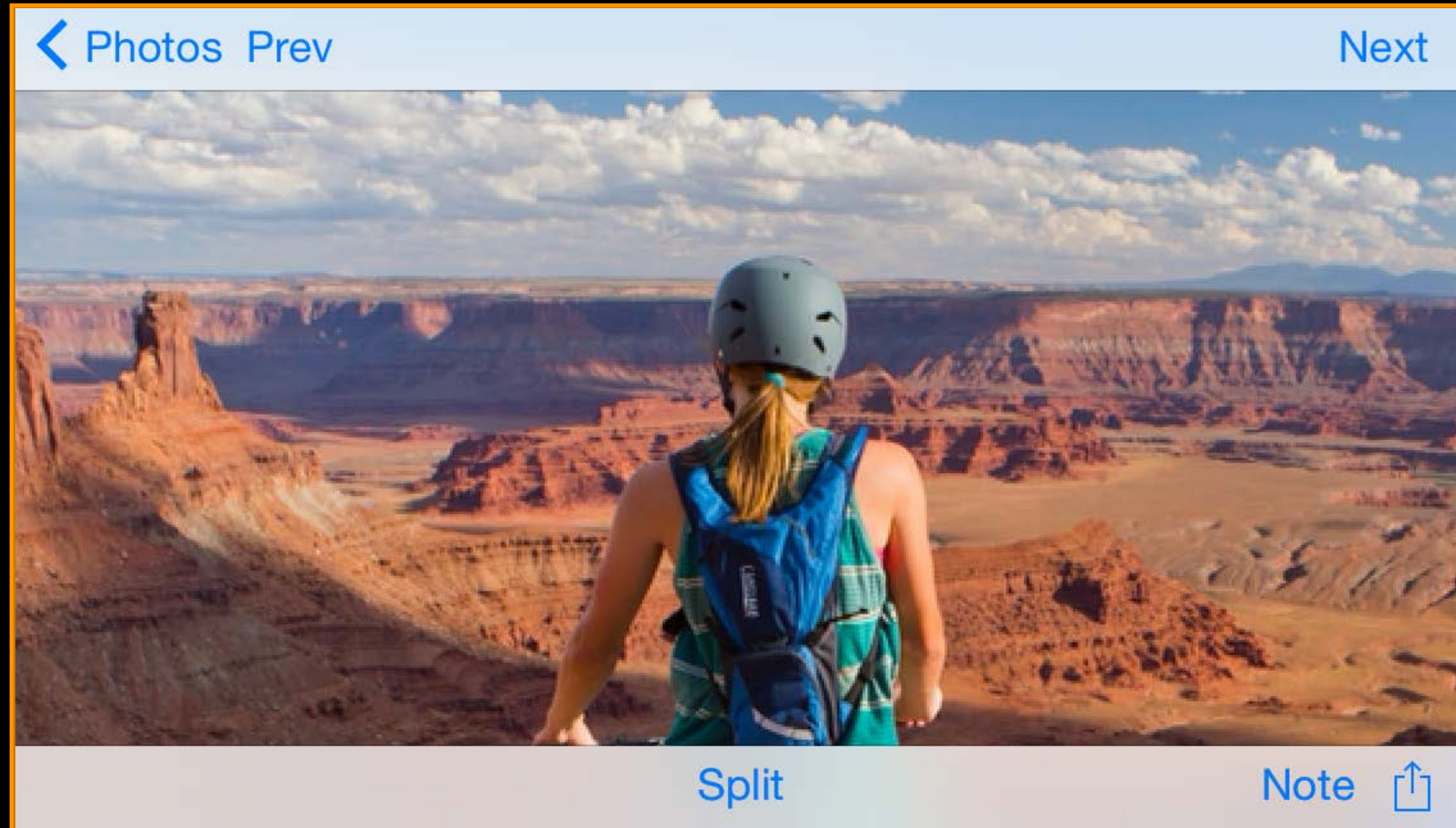


# Embed a UISVC inside a containerVC





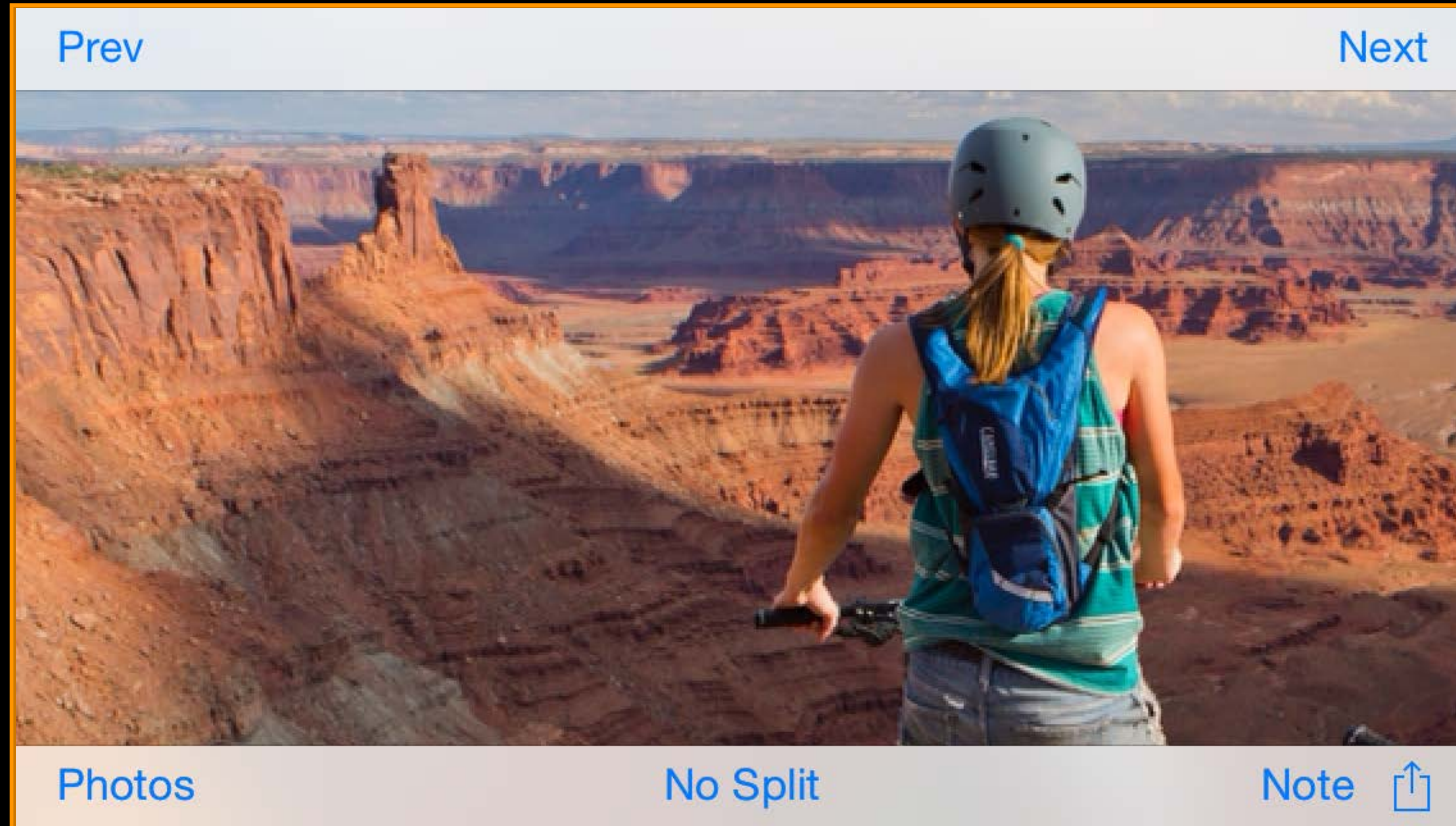
# Embed a UISVC inside a containerVC



```
[containerVC setOverrideTraitCollection: c forChildViewController:svc];
```



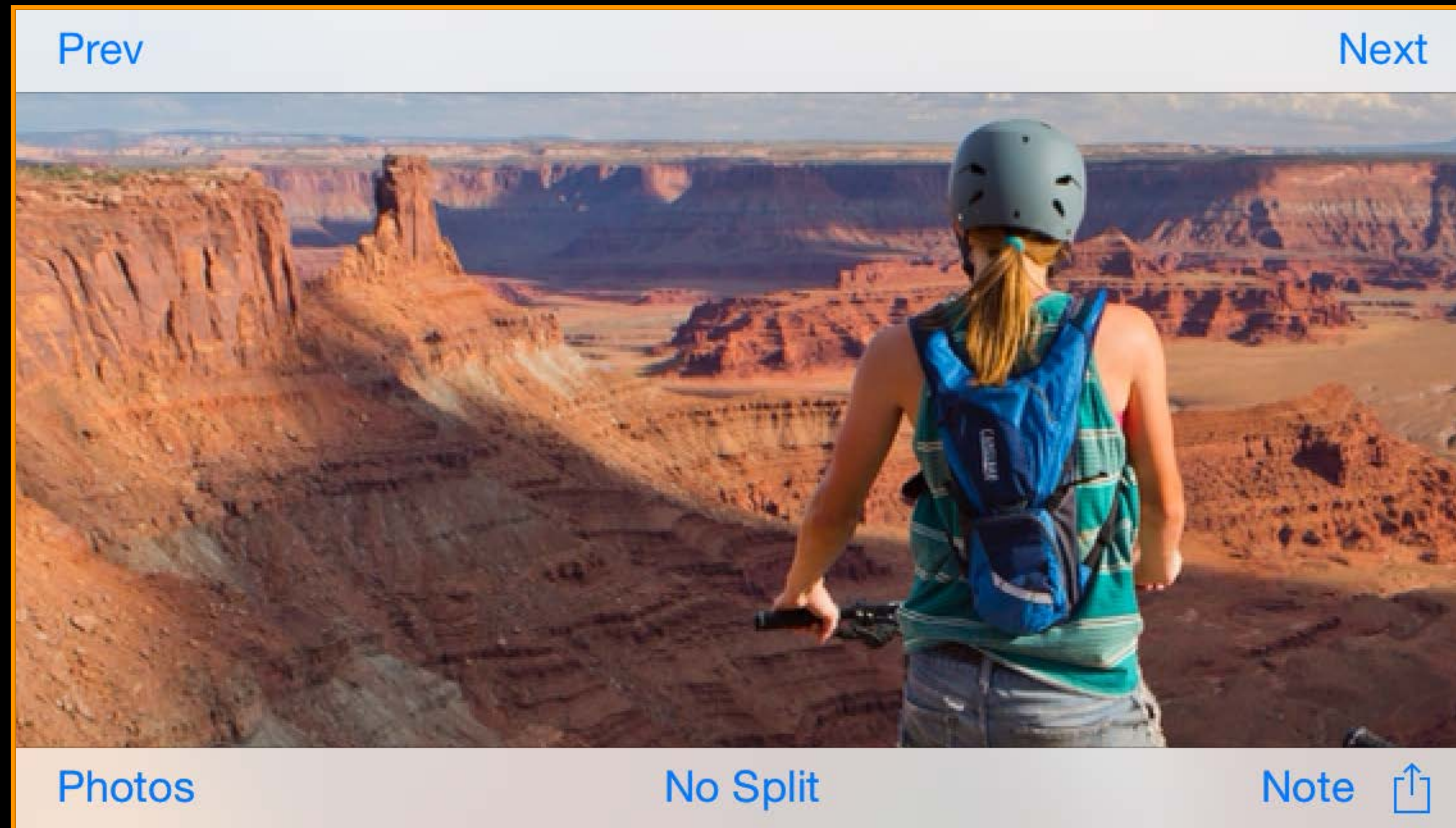
# Embed a UISVC inside a containerVC



```
[containerVC setOverrideTraitCollection: c forChildViewController:svc];
```

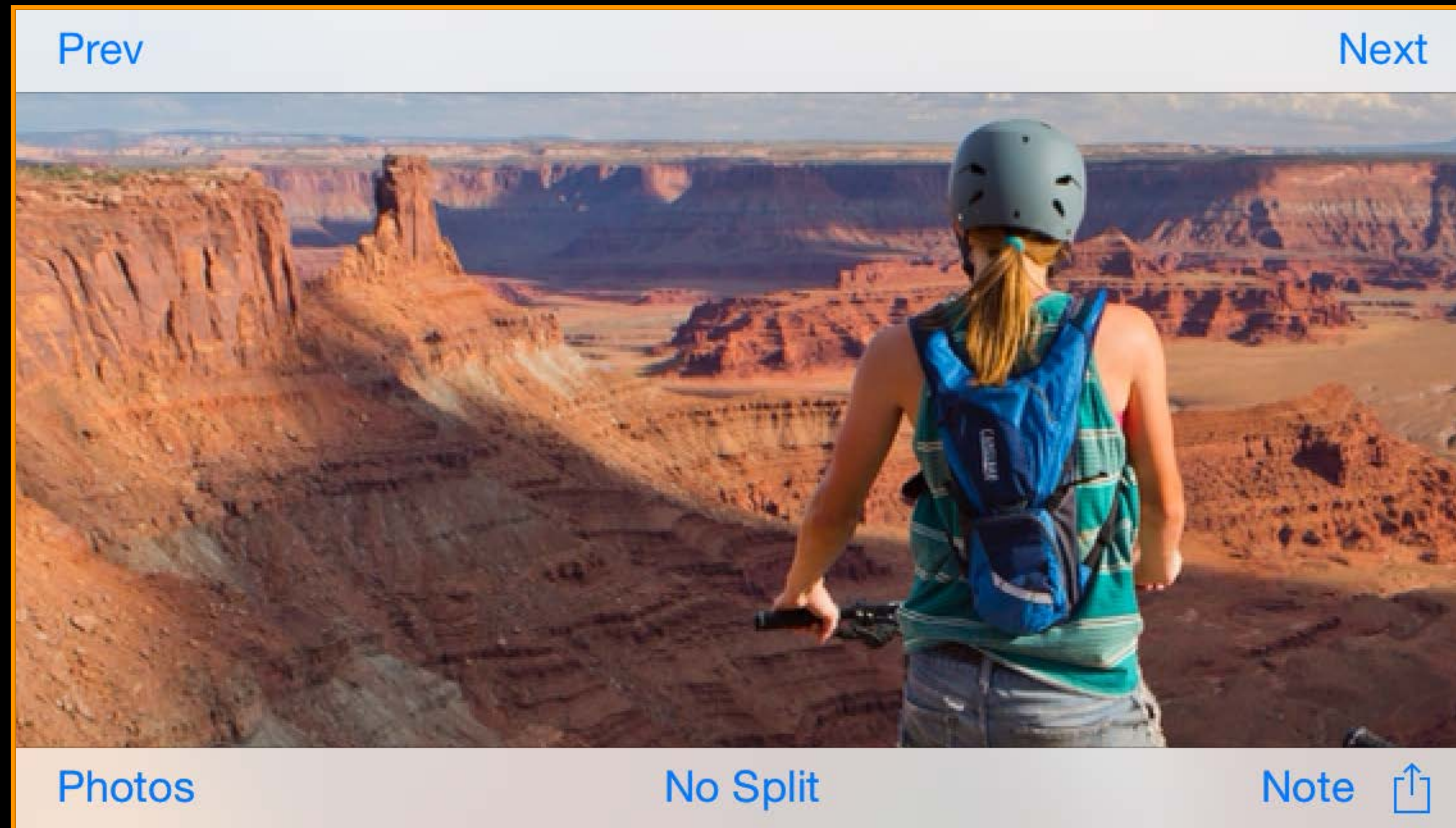


# Once expanded use preferredDisplayMode





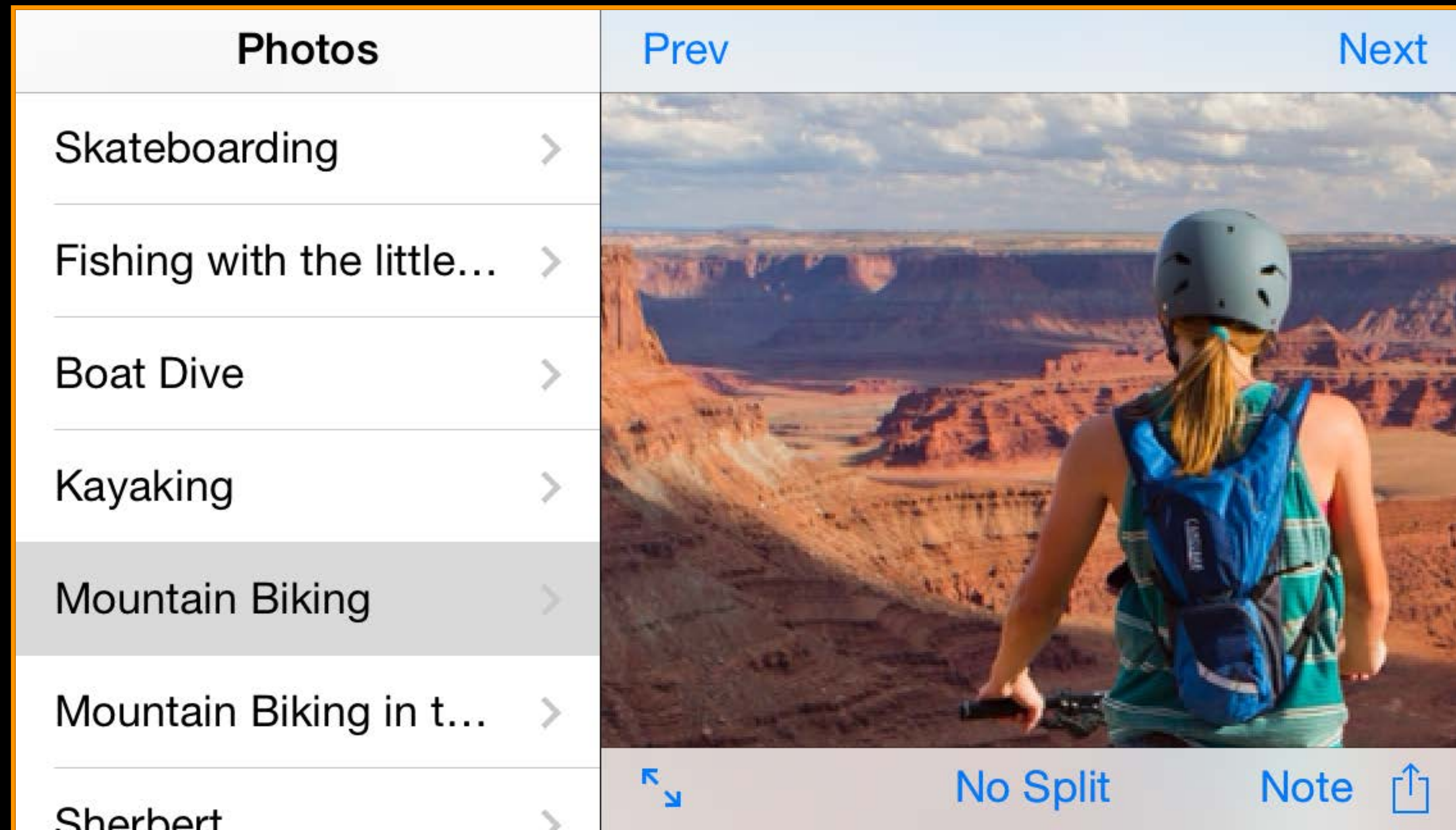
# Once expanded use preferredDisplayMode



```
svc.preferredDisplayMode = UISplitViewControllerDisplayModeAllVisible;
```



# Once expanded use preferredDisplayMode



```
svc.preferredDisplayMode = UISplitViewControllerDisplayModeAllVisible;
```



# UISplitViewController



```
@property (...) UISplitViewControllerDisplayMode preferredDisplayMode;

typedef NS_ENUM(NSInteger, UISplitViewControllerDisplayMode) {
    UISplitViewControllerDisplayModeAutomatic,
    UISplitViewControllerDisplayModePrimaryHidden,
    UISplitViewControllerDisplayModeAllVisible,
    UISplitViewControllerDisplayModePrimaryOverlay,
};
```

# UISplitViewController

A white rounded square badge with the word "NEW" in a colorful, outlined font.

```
@property (...) UISplitViewControllerDisplayMode preferredDisplayMode;
```

```
typedef NS_ENUM(NSInteger, UISplitViewControllerDisplayMode) {  
    UISplitViewControllerDisplayModeAutomatic,  
    UISplitViewControllerDisplayModePrimaryHidden,  
    UISplitViewControllerDisplayModeAllVisible,  
    UISplitViewControllerDisplayModePrimaryOverlay,  
};
```

# UISplitViewController



```
@property (...) UISplitViewControllerDisplayMode preferredDisplayMode;
```

```
typedef NS_ENUM(NSInteger, UISplitViewControllerDisplayMode) {
```

```
    UISplitViewControllerDisplayModeAutomatic,  
    UISplitViewControllerDisplayModePrimaryHidden,  
    UISplitViewControllerDisplayModeAllVisible,  
    UISplitViewControllerDisplayModePrimaryOverlay,
```

```
};
```

# UISplitViewController



```
@property (...) UISplitViewControllerDisplayMode preferredDisplayMode;
```


```
typedef NS_ENUM(NSInteger, UISplitViewControllerDisplayMode) {  
    UISplitViewControllerDisplayModeAutomatic,  
    UISplitViewControllerDisplayModePrimaryHidden,  
    UISplitViewControllerDisplayModeAllVisible,  
    UISplitViewControllerDisplayModePrimaryOverlay,  
};
```

```
@property (readonly) UISplitViewControllerDisplayMode displayMode;
```

```
- (UIBarButtonItem *)displayModeButtonItem;
```

You may also control the width of the split

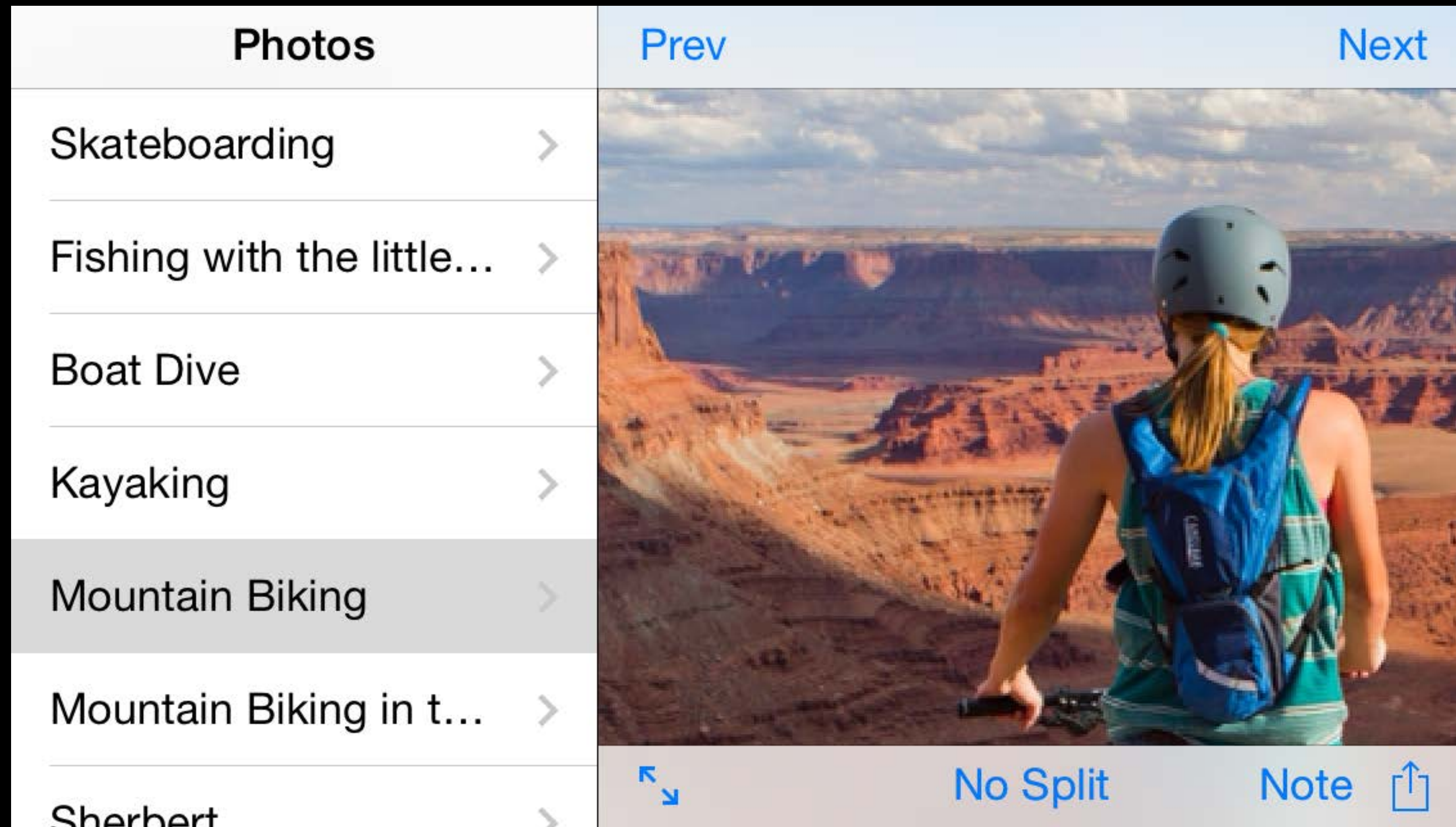


Photos	Prev <span style="float: right;">Next</span>
Skateboarding >	
Fishing with the little... >	
Boat Dive >	
Kayaking >	
Mountain Biking >	
Mountain Biking in t... >	
Sherbert >	

↶ ↷ No Split Note 📄

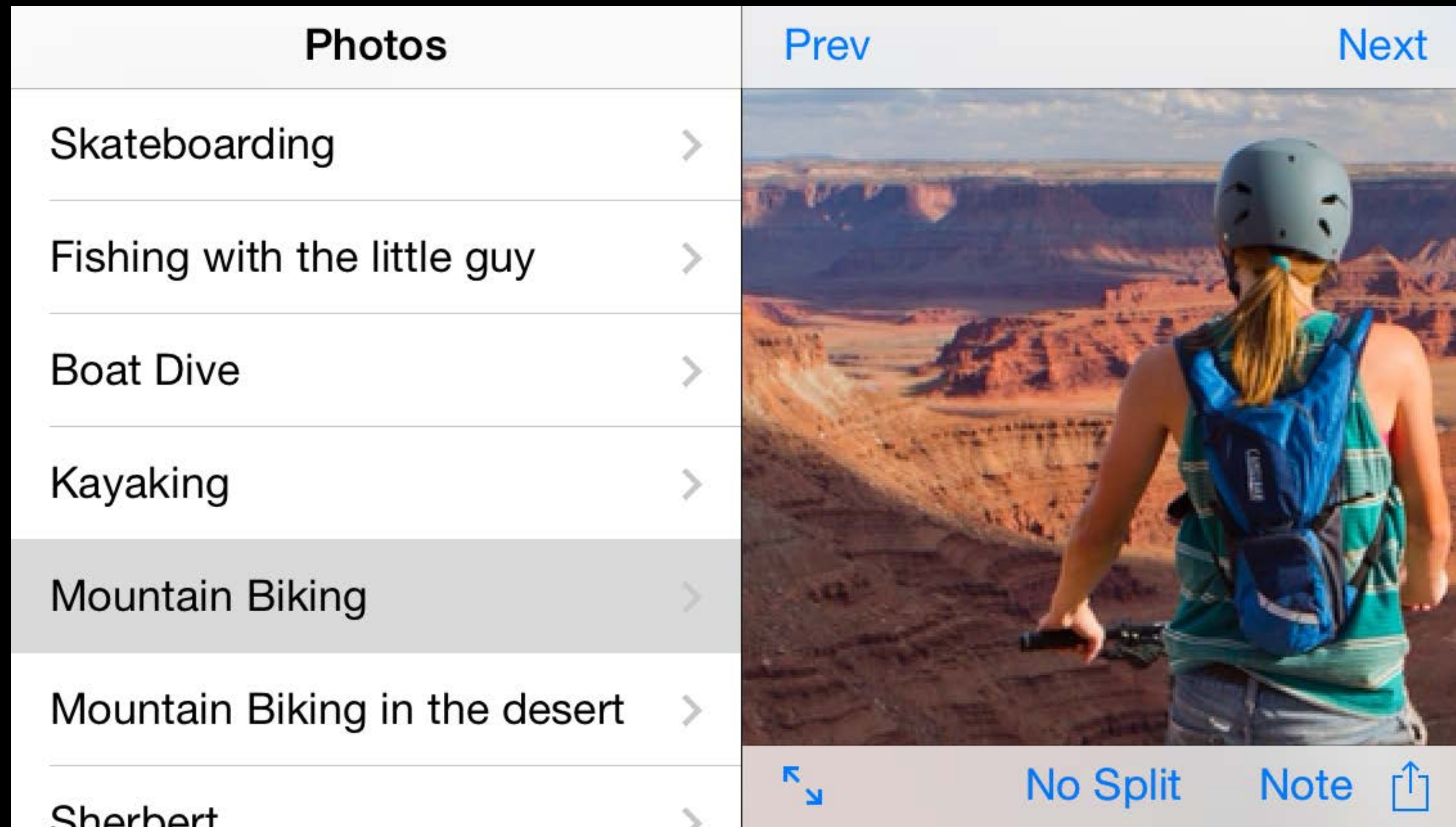


```
svc.preferredPrimaryColumnWidthFraction = .5;
```





```
svc.preferredPrimaryColumnWidthFraction = .5;
```



# UISplitViewController



# UISplitViewController



```
@property (assign) CGFloat preferredPrimaryColumnWidthFraction;  
@property (assign) CGFloat minimumPrimaryColumnWidth;  
@property(assign) CGFloat maximumPrimaryColumnWidth;  
// The current primary view controller's column width.  
@property(nonatomic, readonly) CGFloat primaryColumnWidth;
```



# UISplitViewController

What have we learned?

# UISplitViewController

What have we learned?

Can be used on both the iPhone AND iPad

# UISplitViewController

What have we learned?

Can be used on both the iPhone AND iPad

Collapsed by default in horizontally compact environments

# UISplitViewController

What have we learned?

Can be used on both the iPhone AND iPad

Collapsed by default in horizontally compact environments

The animatable displayMode property controls the appearance of the primary



# UISplitViewController

What have we learned?

Can be used on both the iPhone AND iPad

Collapsed by default in horizontally compact environments

The animatable `displayMode` property controls the appearance of the primary

The split width can be specified

There are many new UISVC adaptive APIs  
which customize how the UISVC collapses  
and separates

# Condensing Bars


Prev

Next



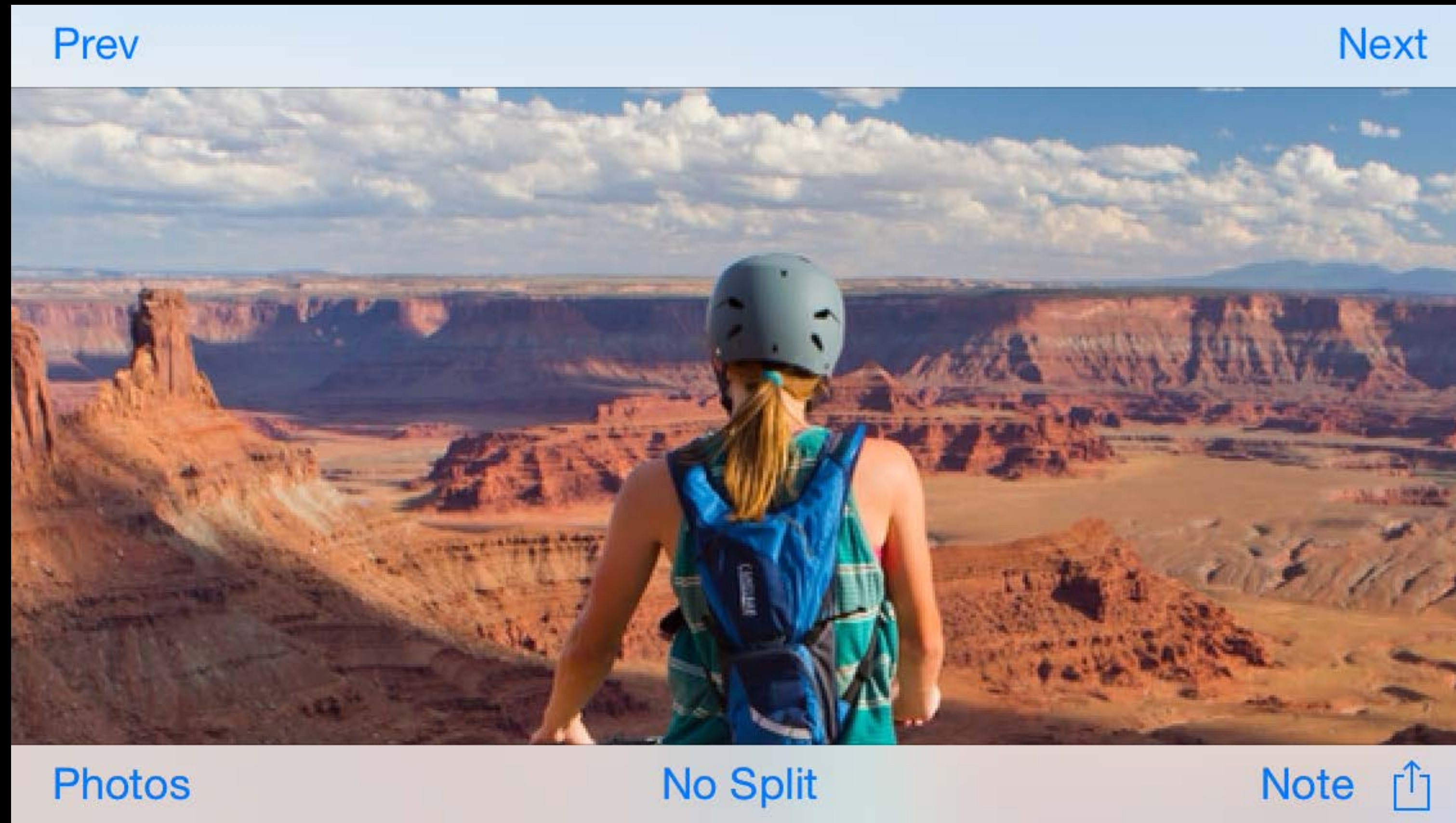
Photos

No Split

Note 

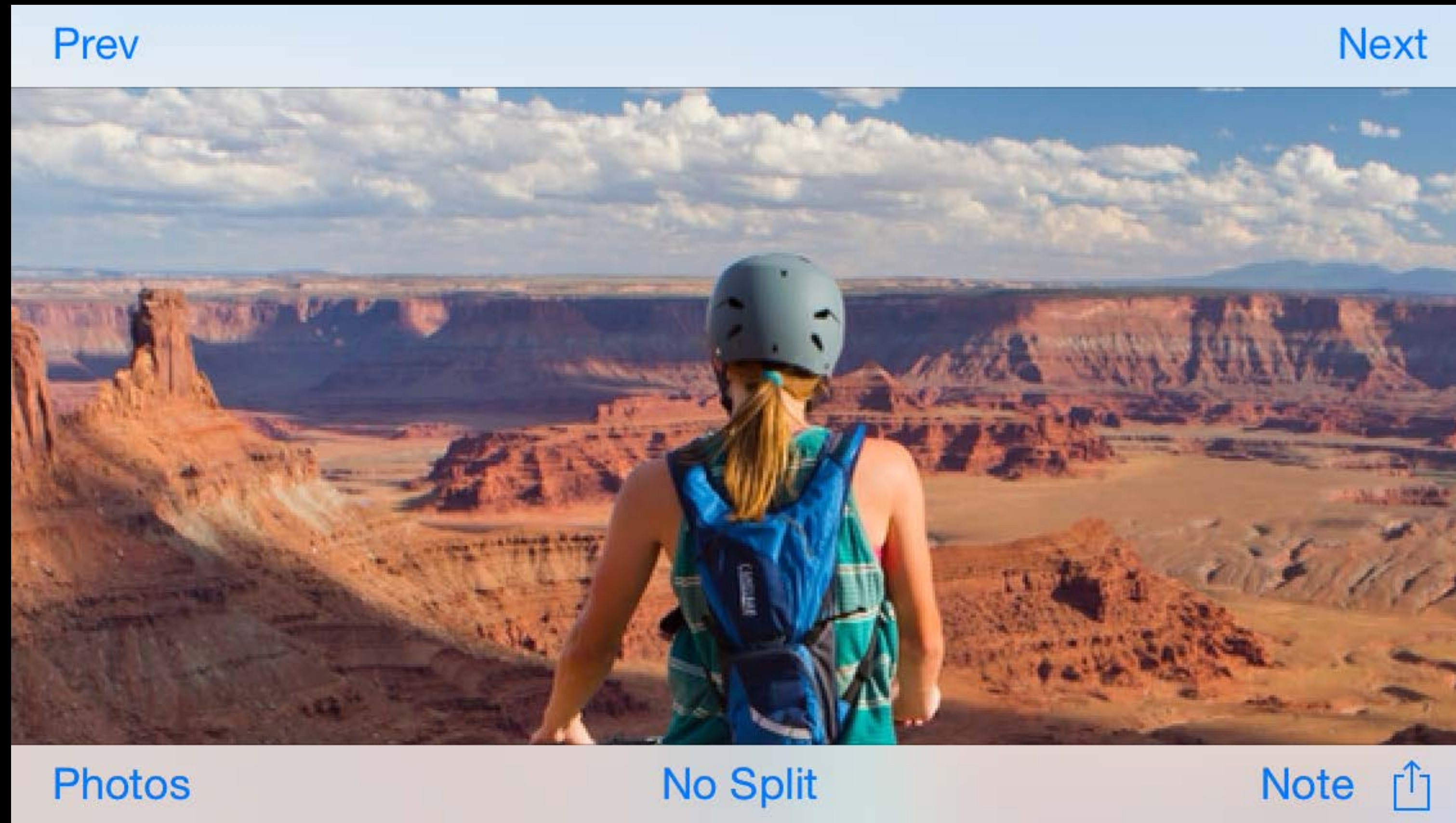


```
UINavigationController *navController;  
navController.hidesBarOnTap = YES;
```





```
UINavigationController *navController;  
navController.hidesBarOnTap = YES;
```














Prev

Next



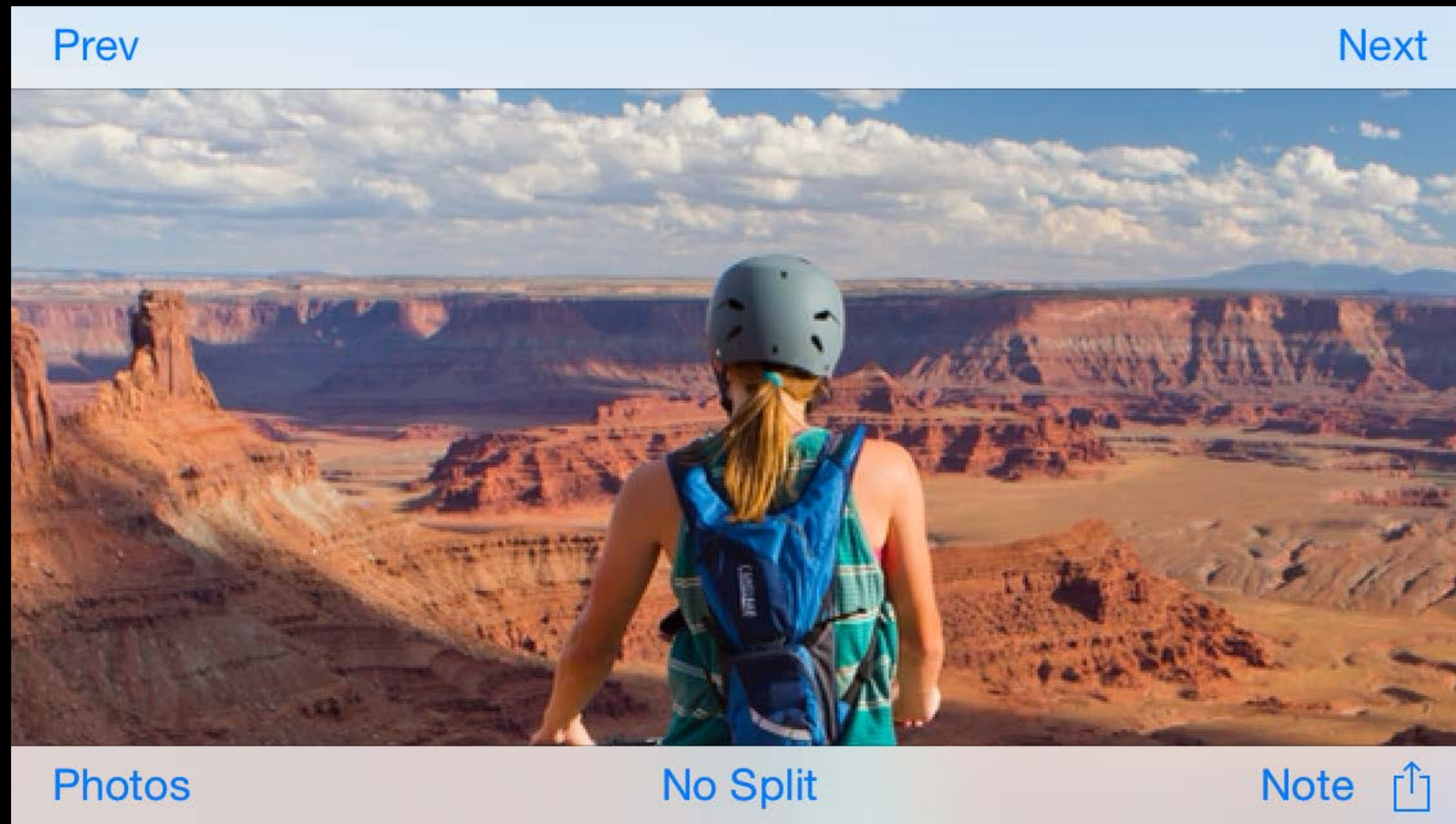
Photos

No Split

Note 

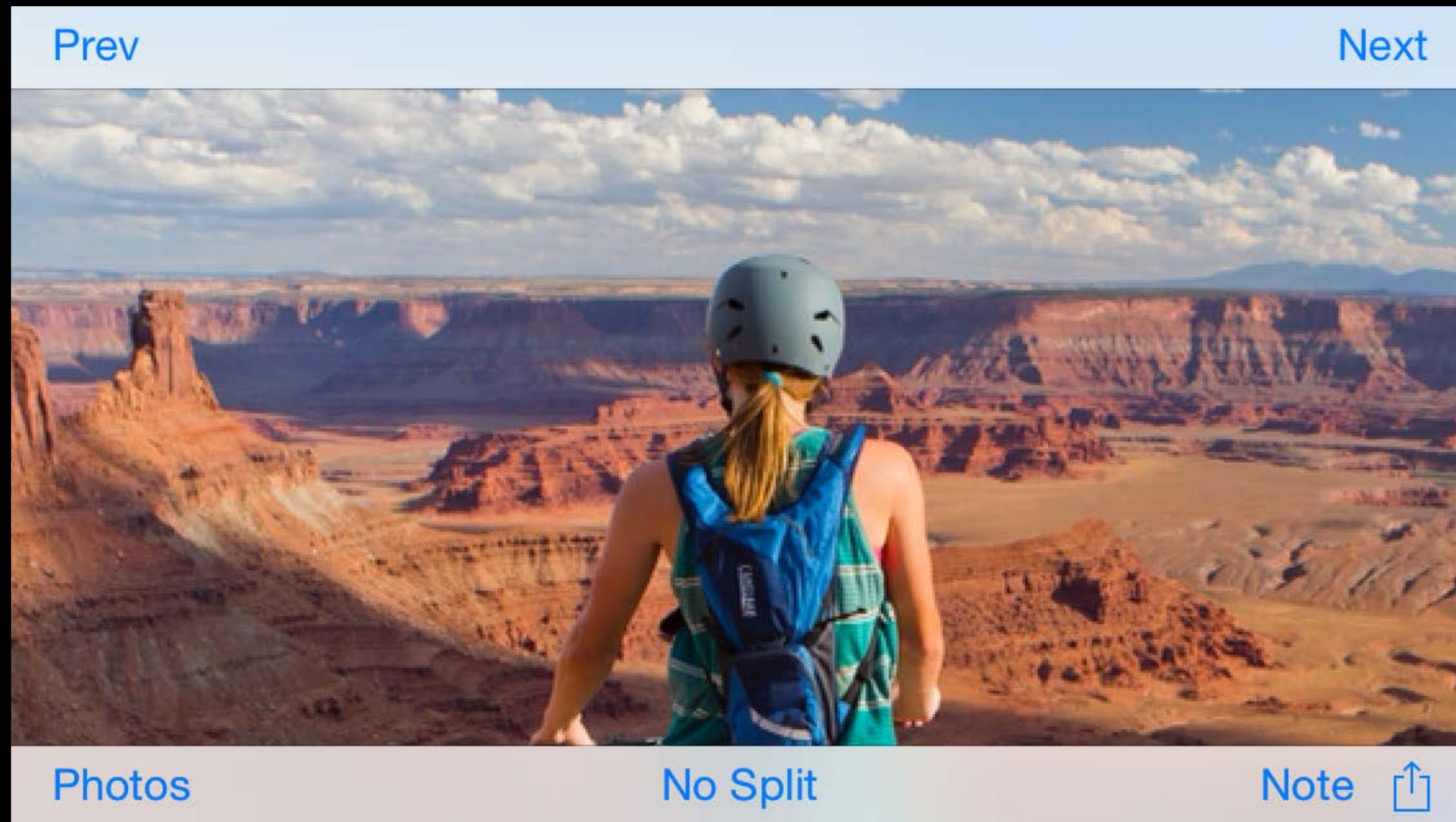


```
UINavigationController *navController;  
navController.condensesBarsOnSwiped = YES;
```





```
UINavigationController *navController;  
navController.condensesBarsOnSwiped = YES;
```





[Prev](#)

[Next](#)



# Condensing Bars

New UINavigationController API





# Condensing Bars

## New UINavigationController API



### Automatic Behavior

```
@property(assign) BOOL condensesBarsOnSwipe;  
@property(assign) BOOL hidesBarsOnTap;  
@property(assign) BOOL hidesBarWhenVerticallyCompact;  
@property(assign) BOOL condensesBarsWhenKeyboardAppears
```



# Condensing Bars

## New UINavigationController API



### Automatic Behavior

```
@property(assign) BOOL condensesBarsOnSwipe;  
@property(assign) BOOL hidesBarsOnTap;  
@property(assign) BOOL hidesBarWhenVerticallyCompact;  
@property(assign) BOOL condensesBarsWhenKeyboardAppears
```

### Manual Control

```
@property(getter=isNavigationBarCondensed) BOOL navigationBarCondensed;
```

# Presentation Controllers

# Presentation Controllers

iOS 7 custom presentations

Presenting View  
Controller

View Controller to be  
Presented



# Presentation Controllers

iOS 7 custom presentations

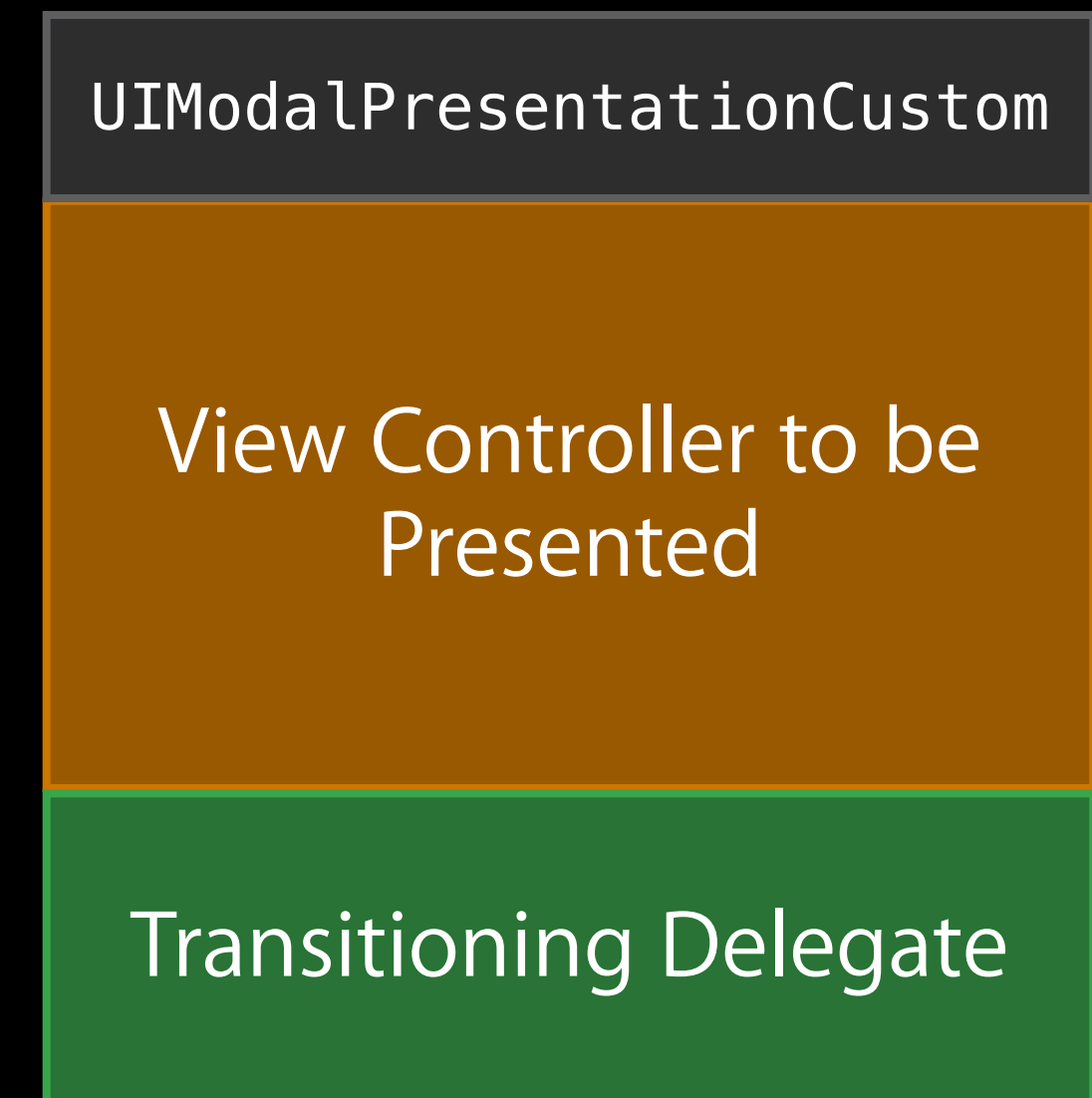
Presenting View  
Controller

UIModalPresentationCustom

View Controller to be  
Presented

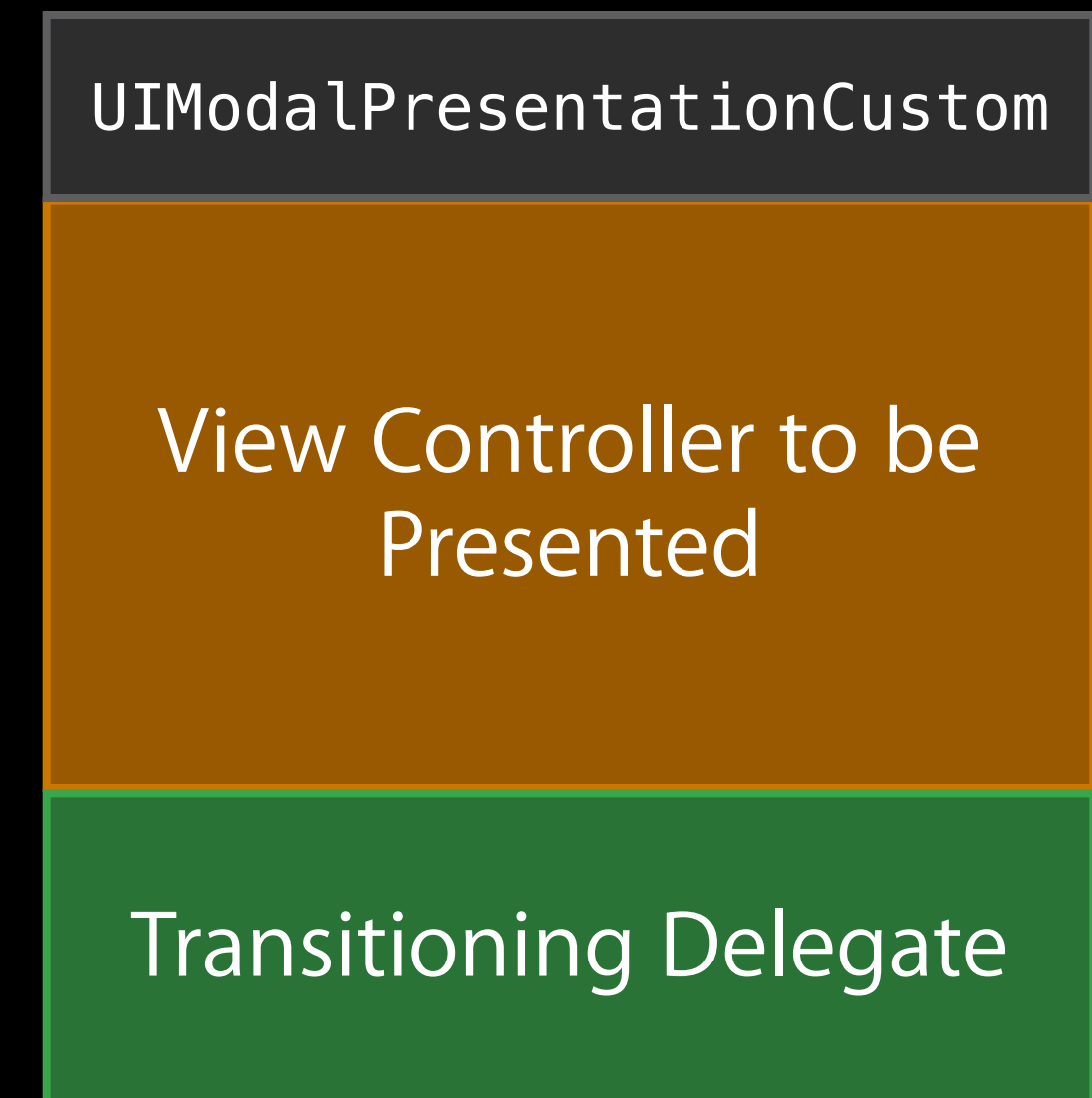
# Presentation Controllers

iOS 7 custom presentations



# Presentation Controllers

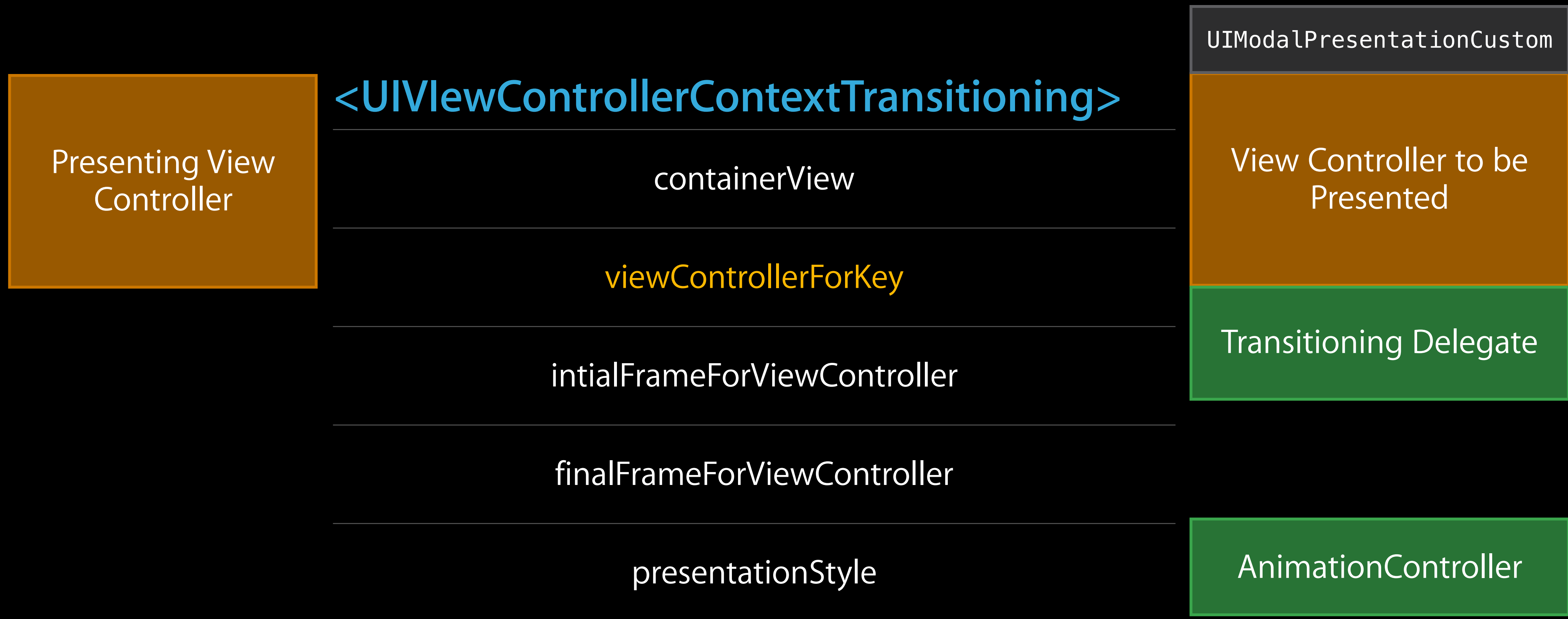
iOS 7 custom presentations





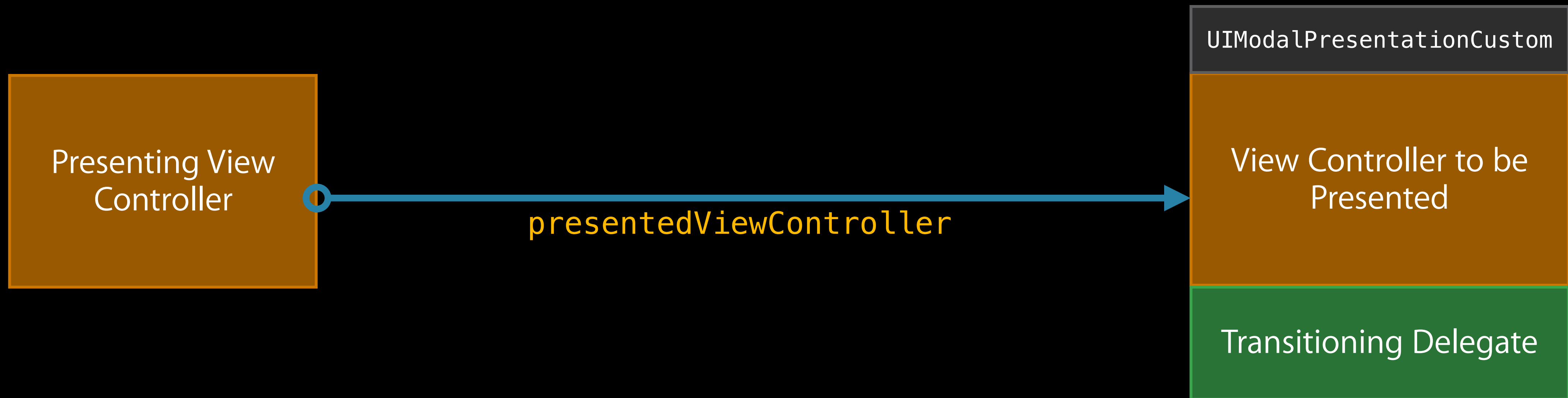
# Presentation Controllers

iOS 7 custom presentations



# Presentation Controllers

iOS 7 custom presentations



# Presentation Controllers

## iOS 7 Custom Presentation Limitations

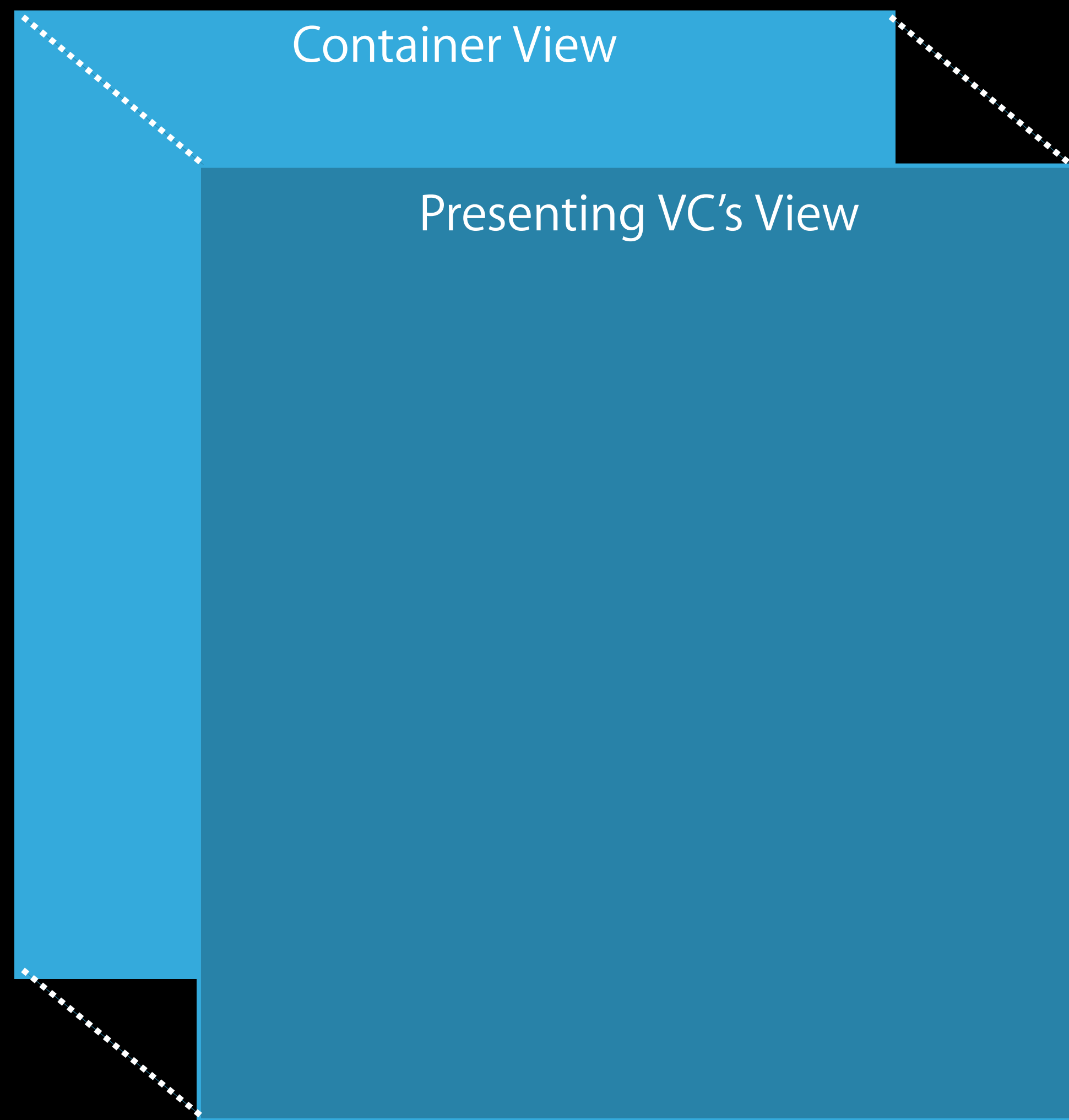


Presenting VC's View



# Presentation Controllers

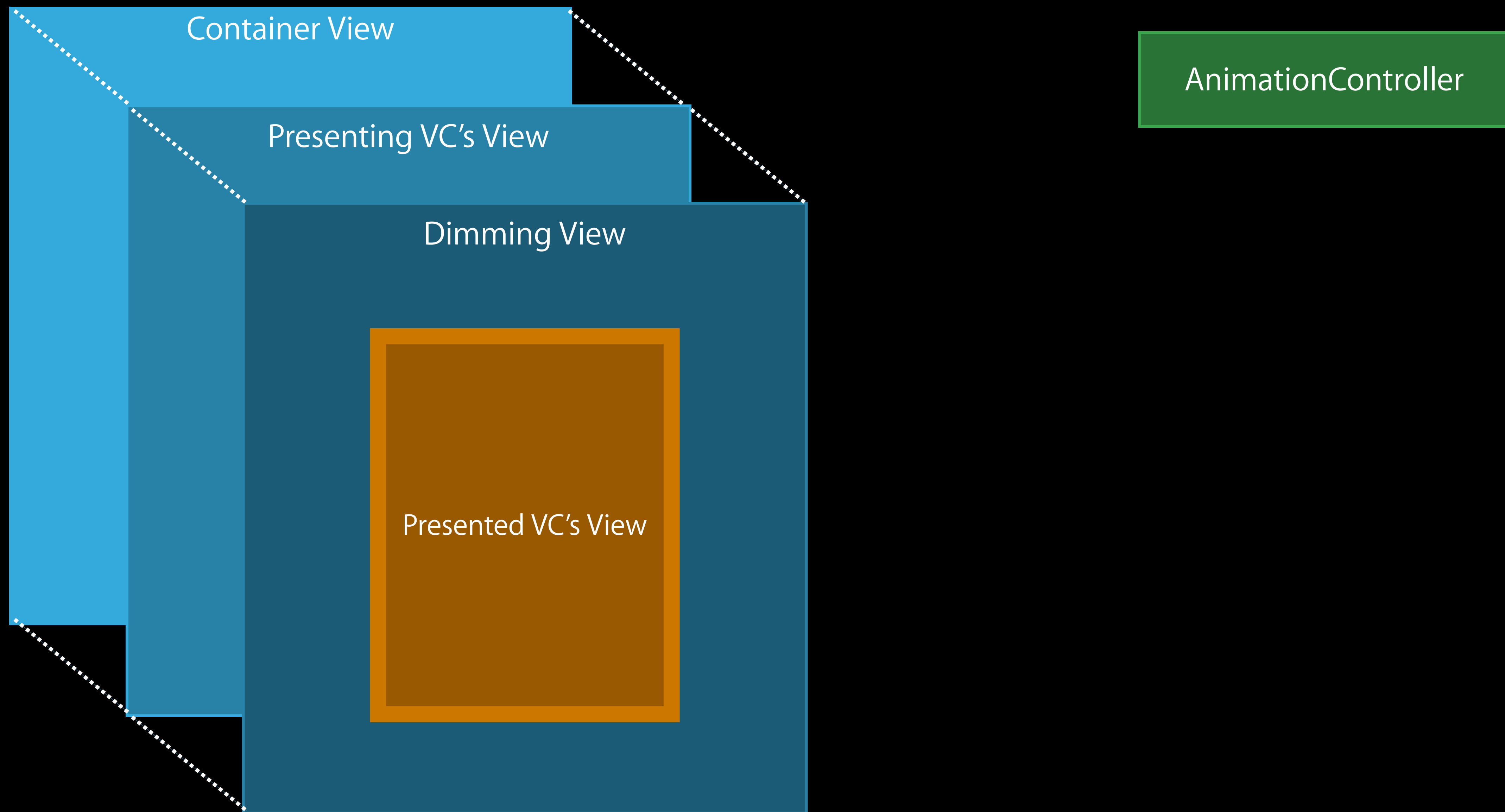
## iOS 7 Custom Presentation Limitations



AnimationController

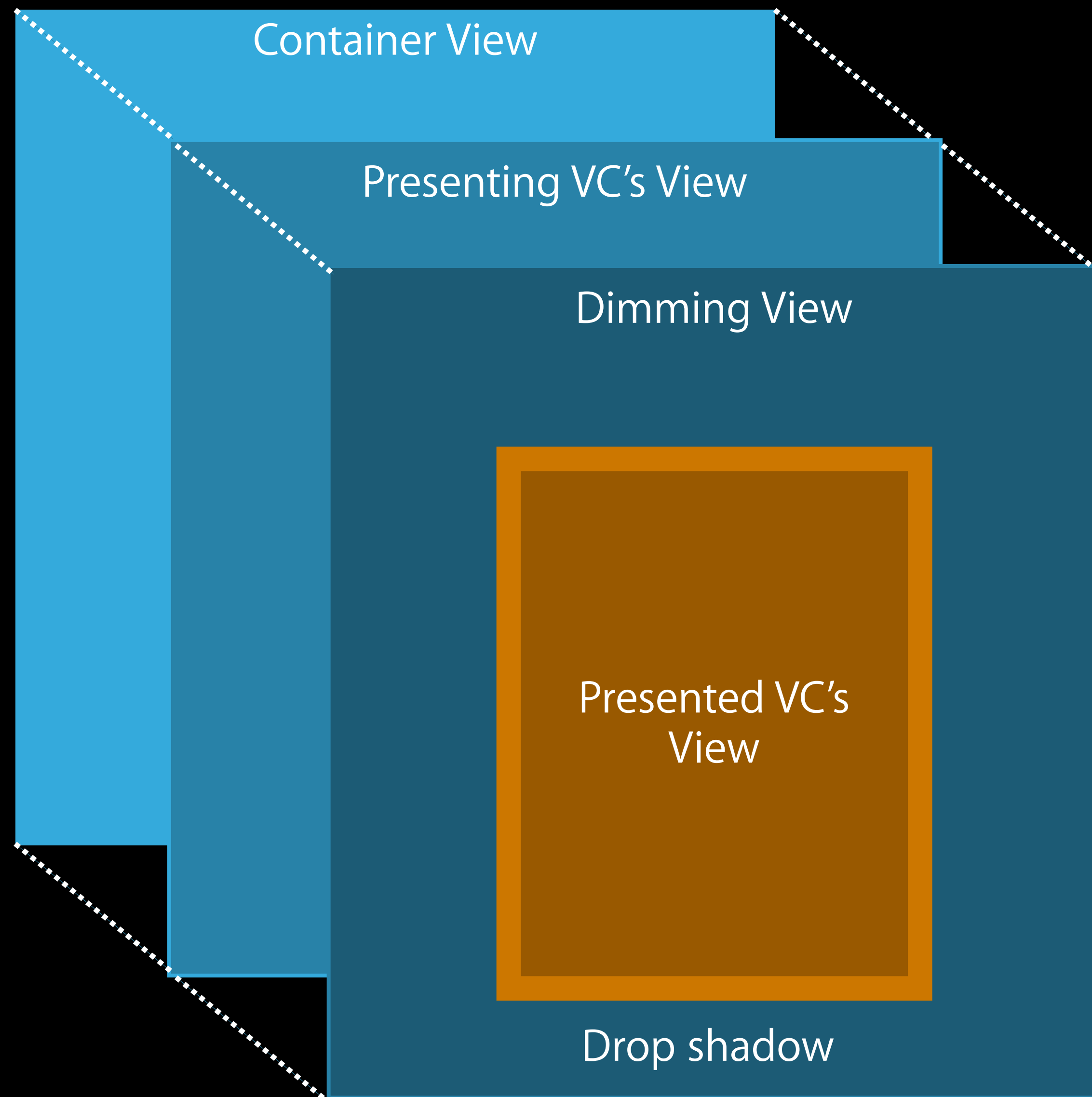
# Presentation Controllers

## iOS 7 Custom Presentation Limitations



# Presentation Controllers

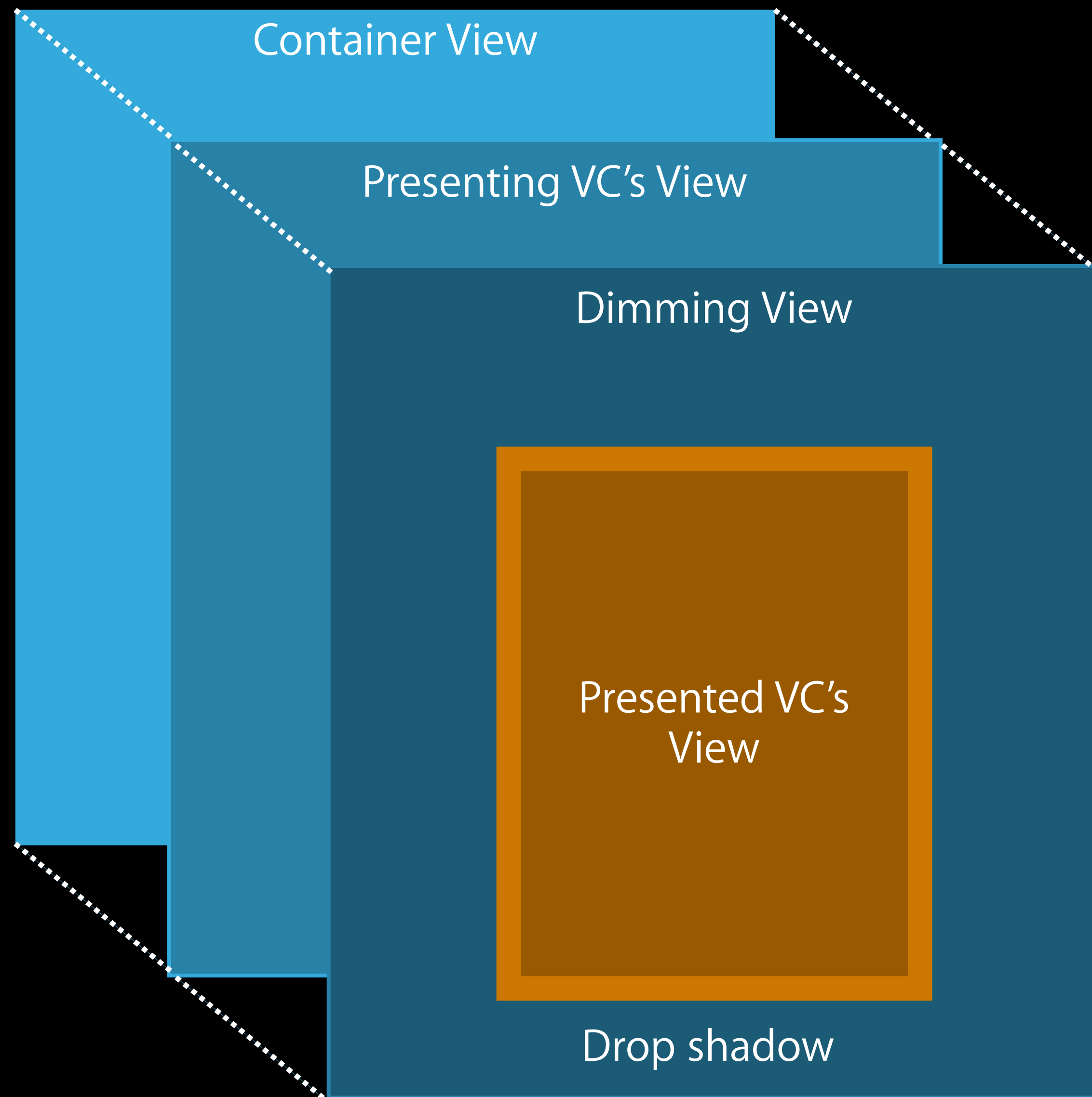
iOS 7 custom presentation limitations





# Presentation Controllers

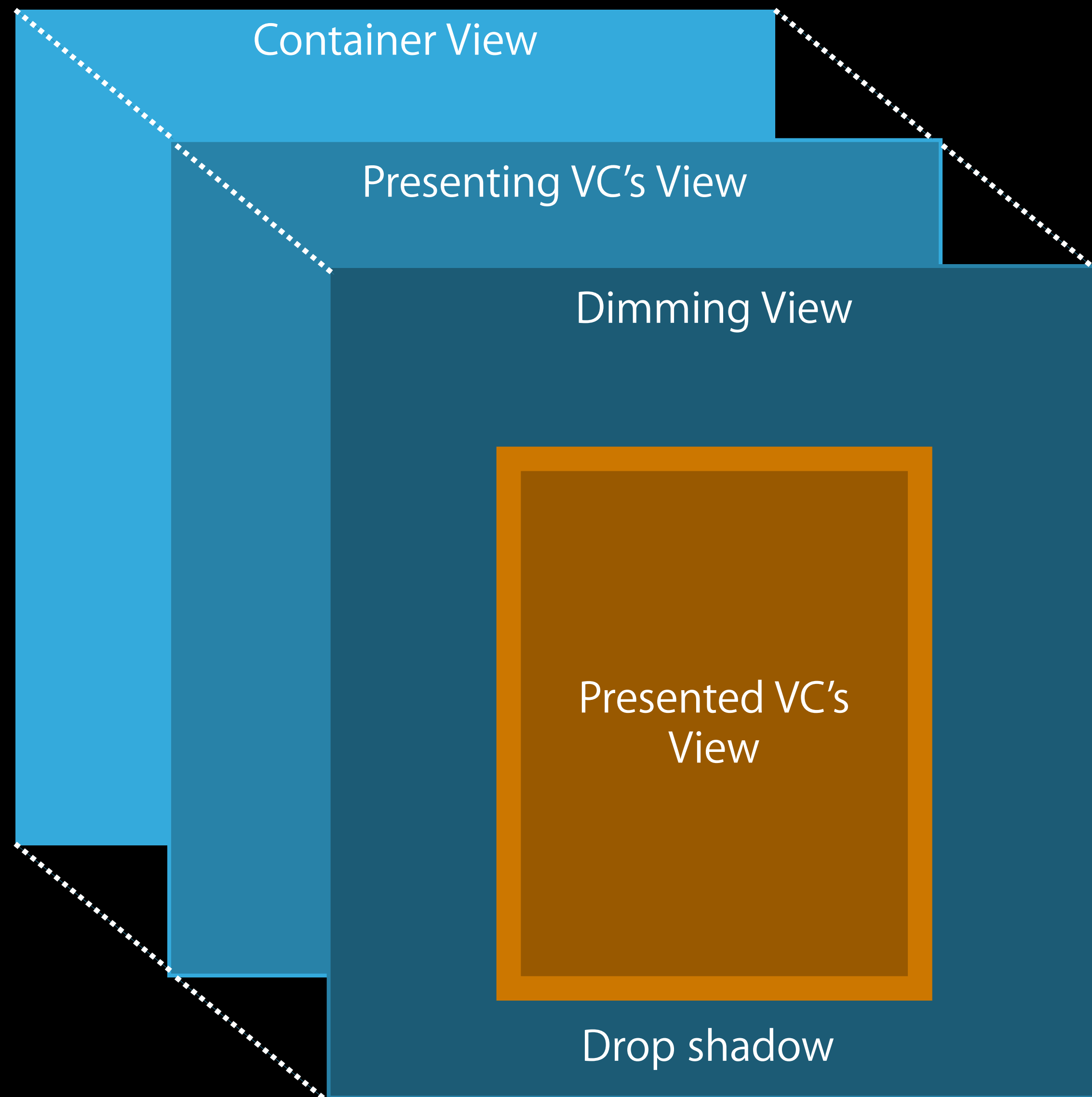
iOS 7 custom presentation limitations



What object owns the chrome?

# Presentation Controllers

iOS 7 custom presentation limitations

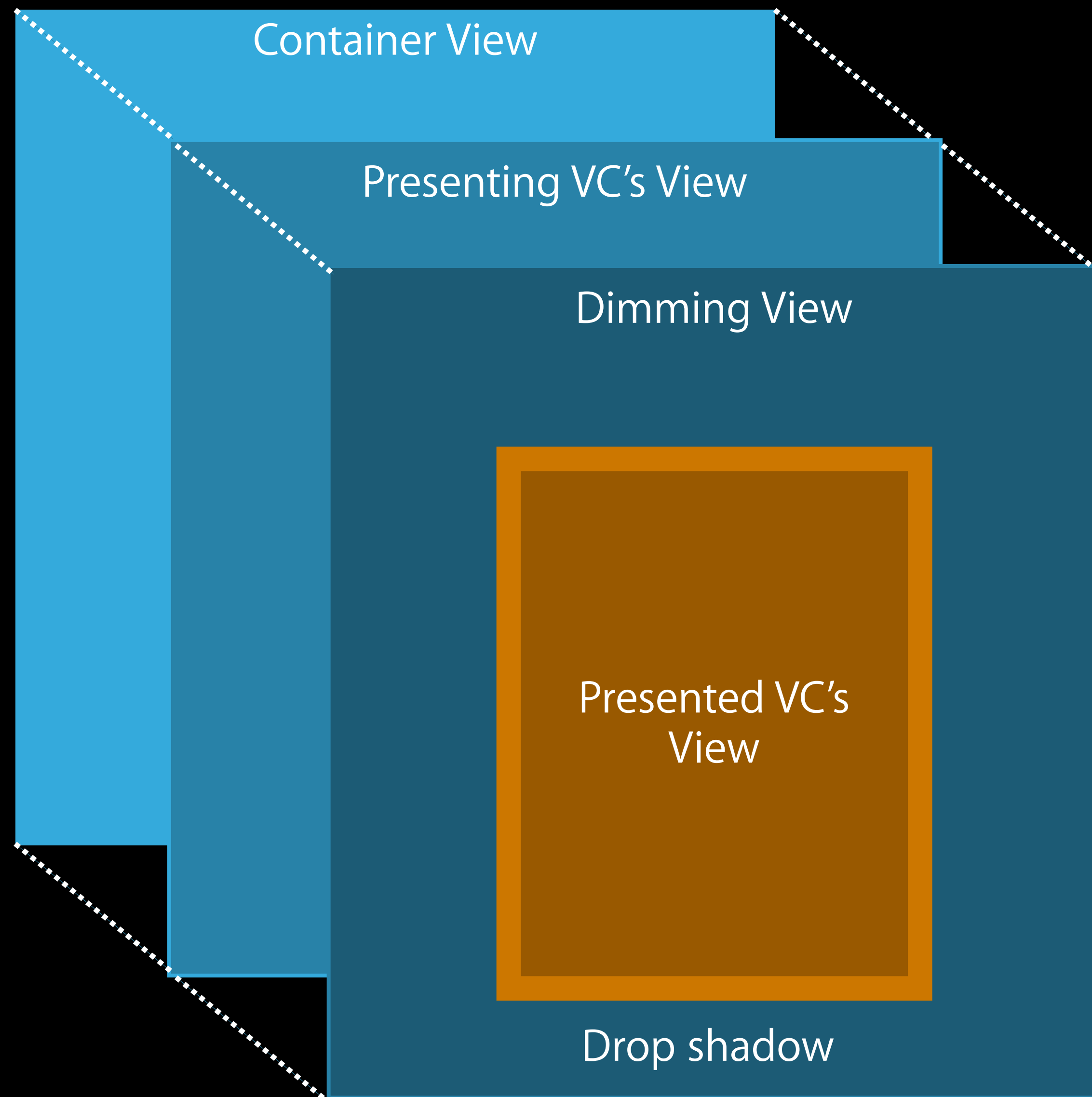


What object owns the chrome?

Tight coupling between presentation animator and dismissing animator

# Presentation Controllers

iOS 7 custom presentation limitations



What object owns the chrome?

Tight coupling between presentation animator and dismissing animator

Back to back presentations become problematic



# Presentation Controllers

iOS 8 custom presentations

Presenting View  
Controller

The diagram consists of two orange rectangular boxes on a black background. The box on the left contains the text 'Presenting View Controller'. The box on the right contains the text 'View Controller to be Presented'. There are no lines or arrows connecting the two boxes.

View Controller to be  
Presented

# Presentation Controllers

iOS 8 custom presentations

Presenting View  
Controller

UIModalPresentationCustom

View Controller to be  
Presented

# Presentation Controllers

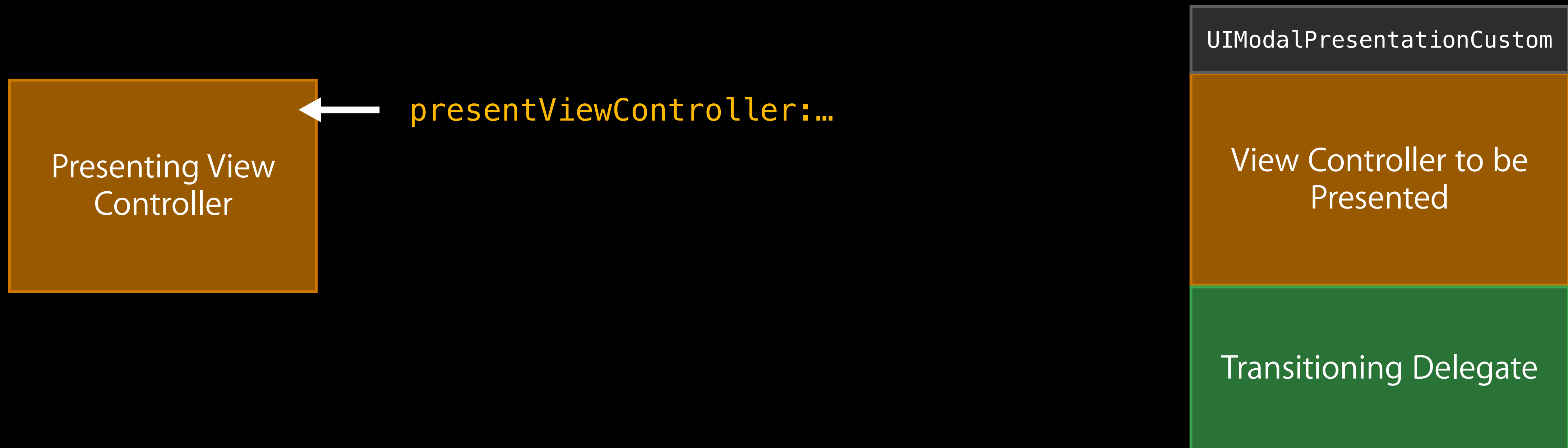
iOS 8 custom presentations





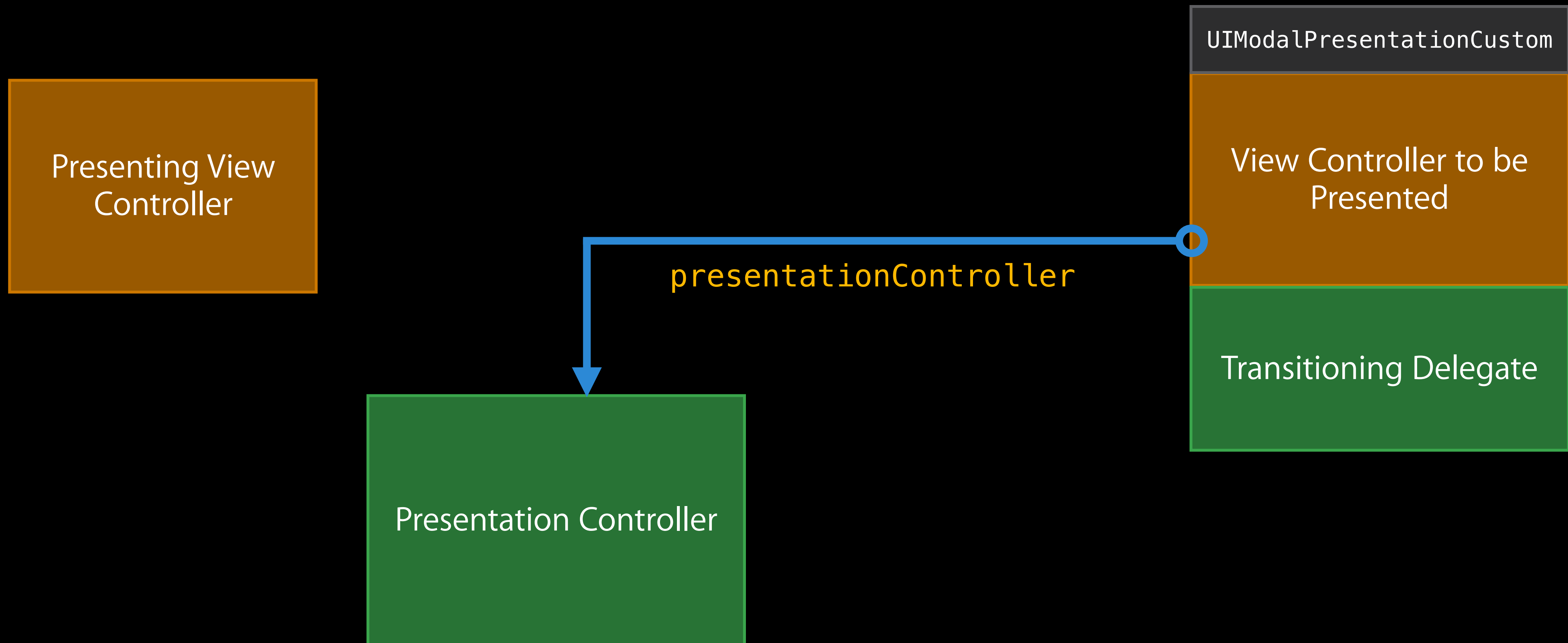
# Presentation Controllers

iOS 8 custom presentations



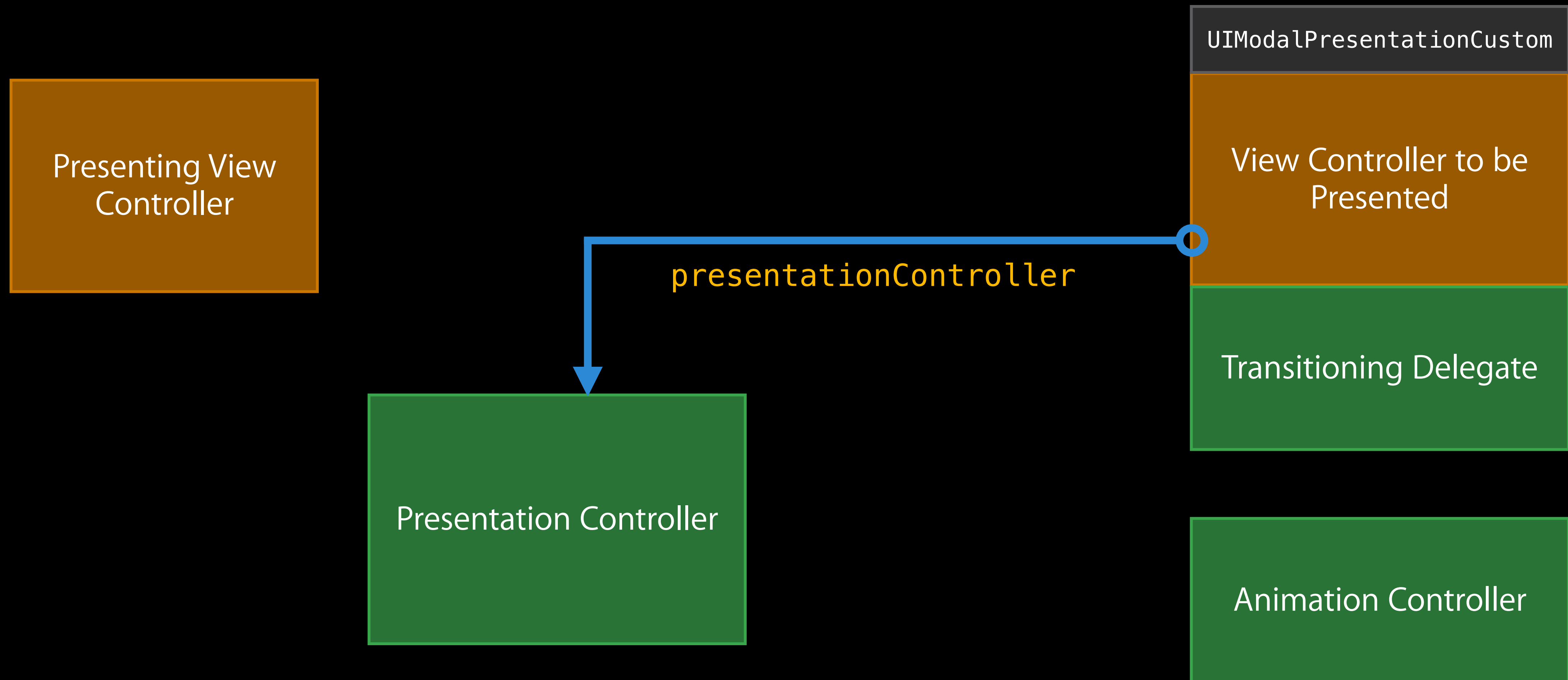
# Presentation Controllers

iOS 8 custom presentations



# Presentation Controllers

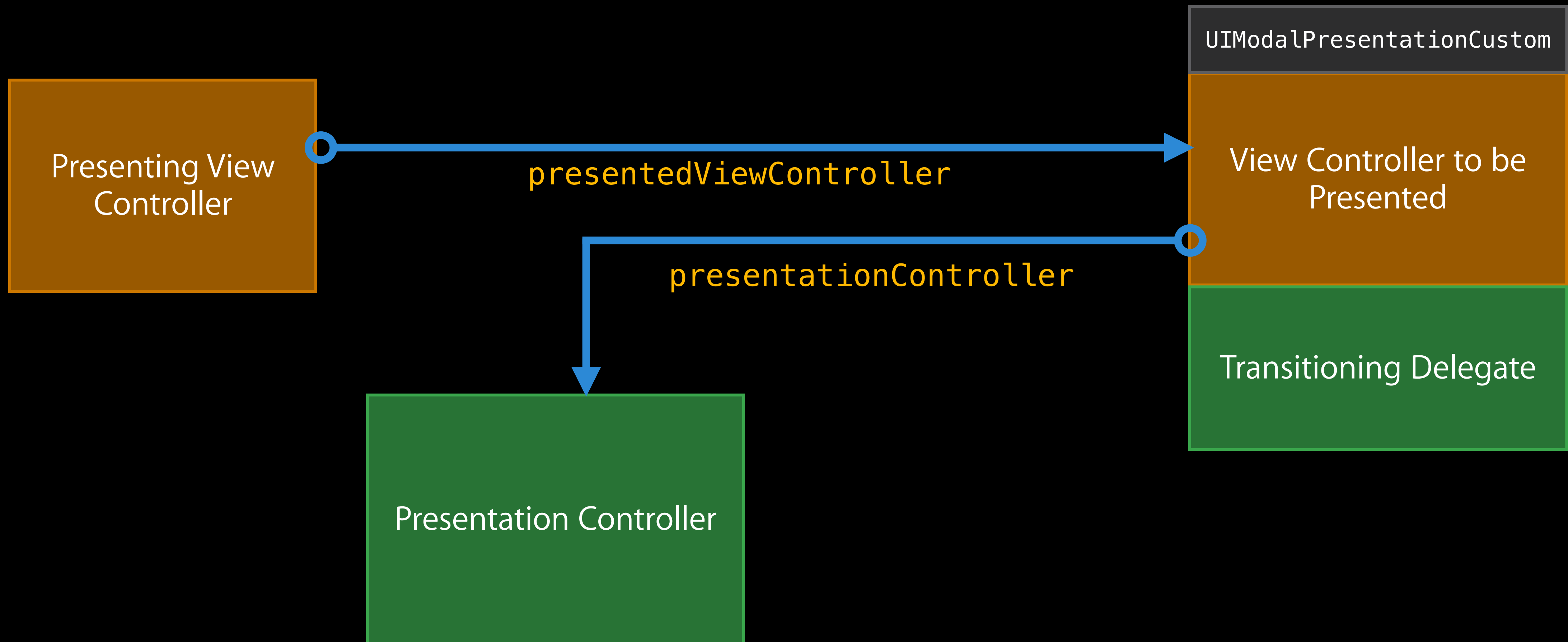
iOS 8 custom presentations



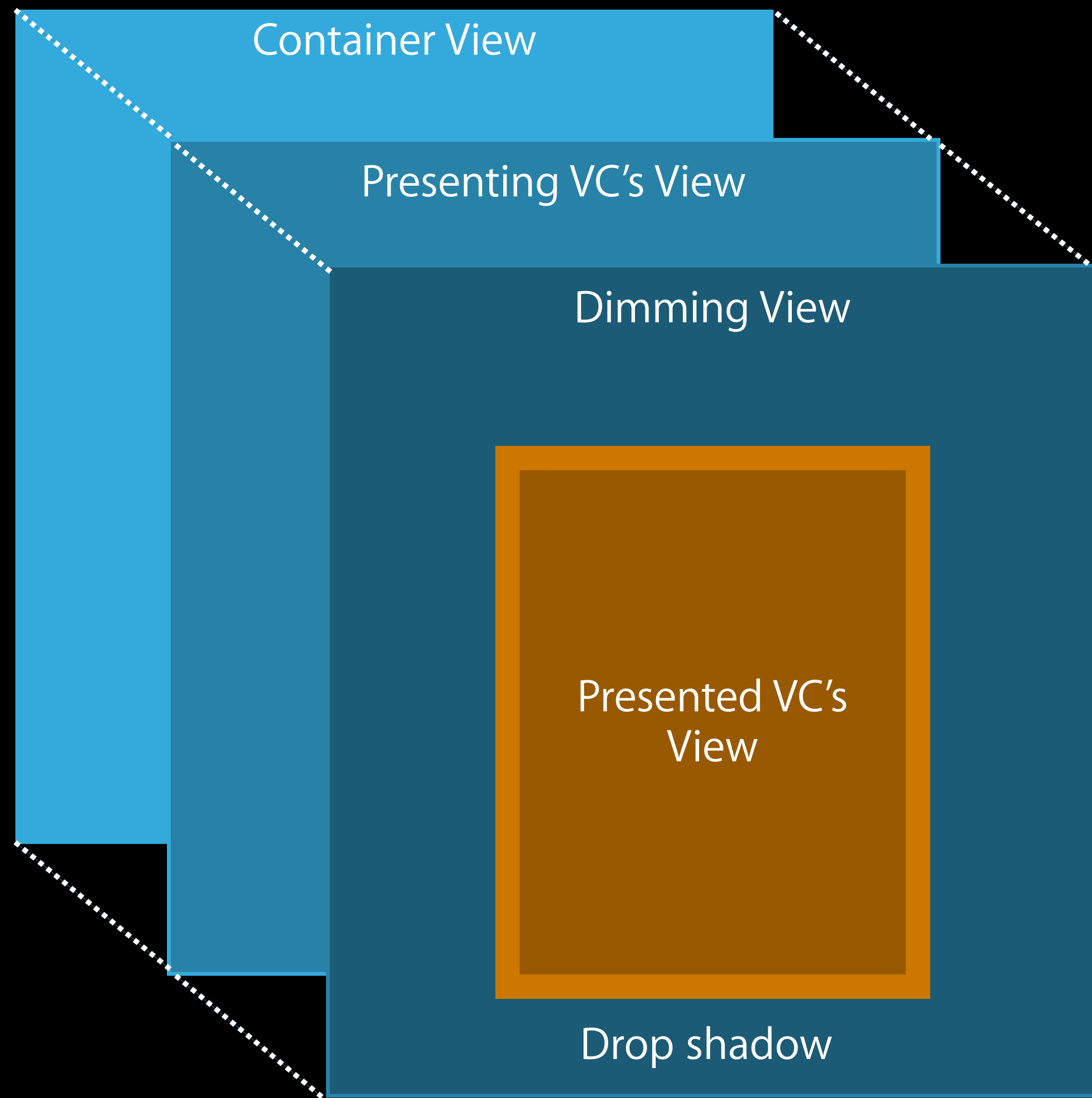


# Presentation Controllers

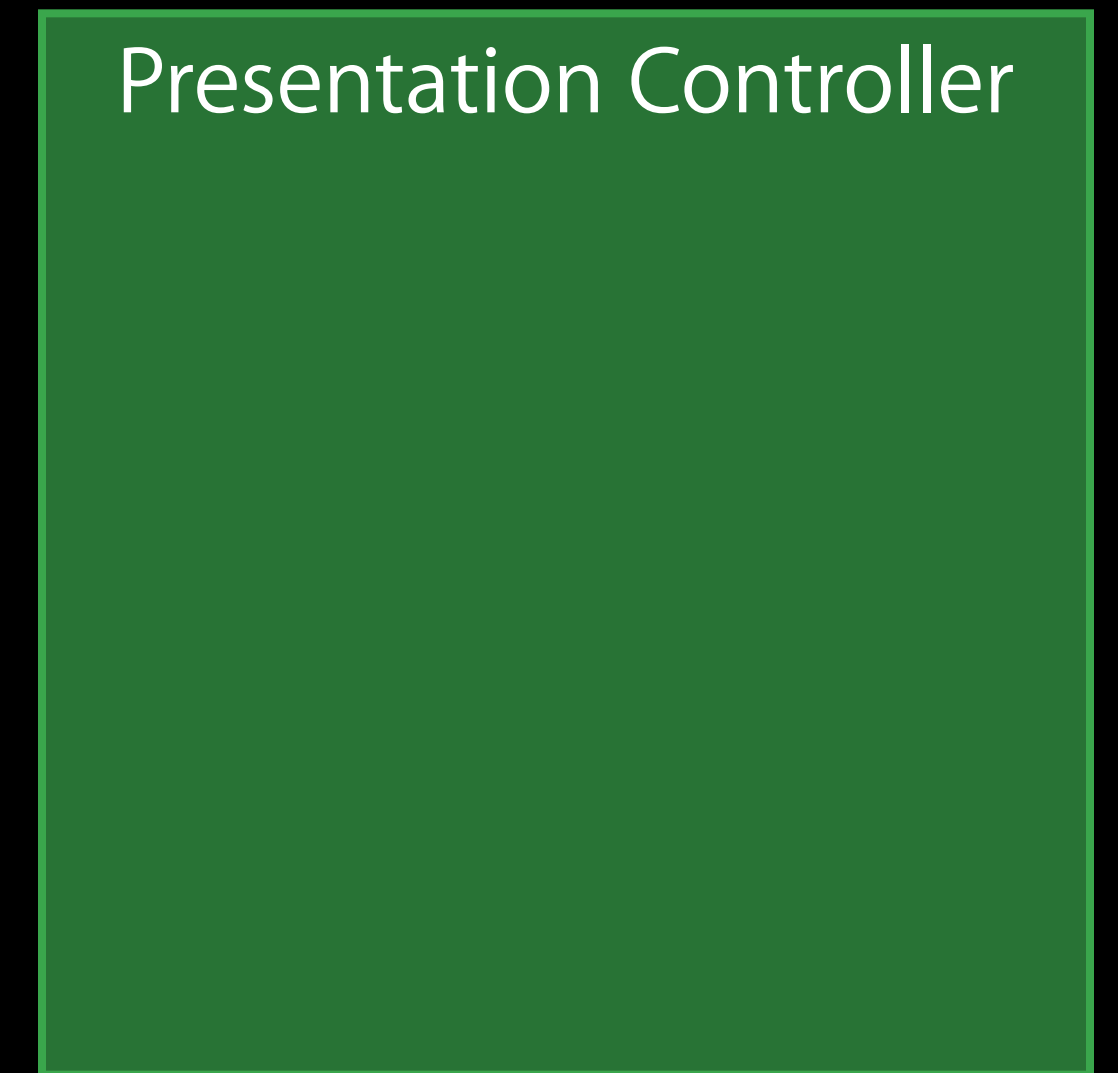
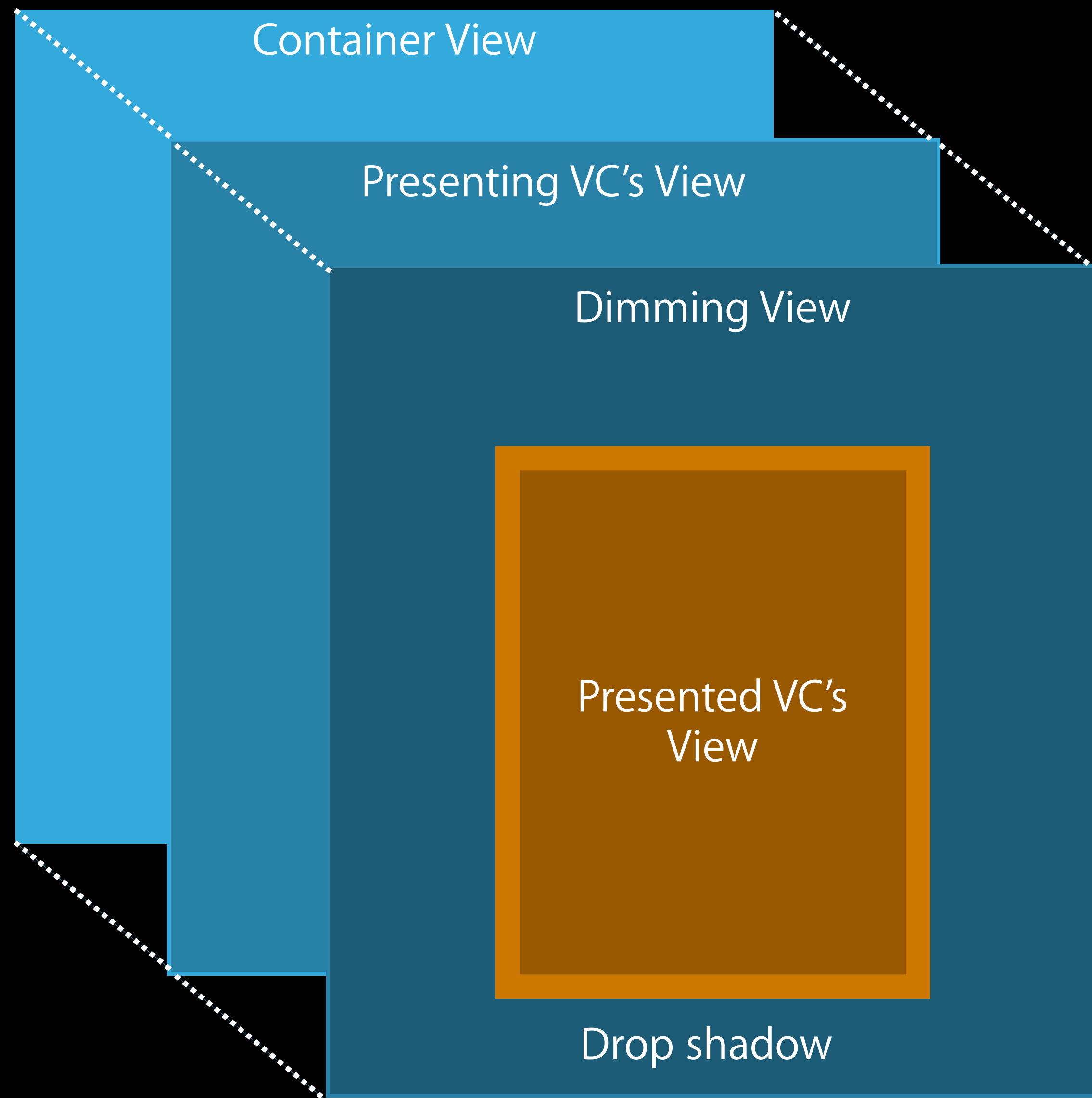
iOS 8 custom presentations



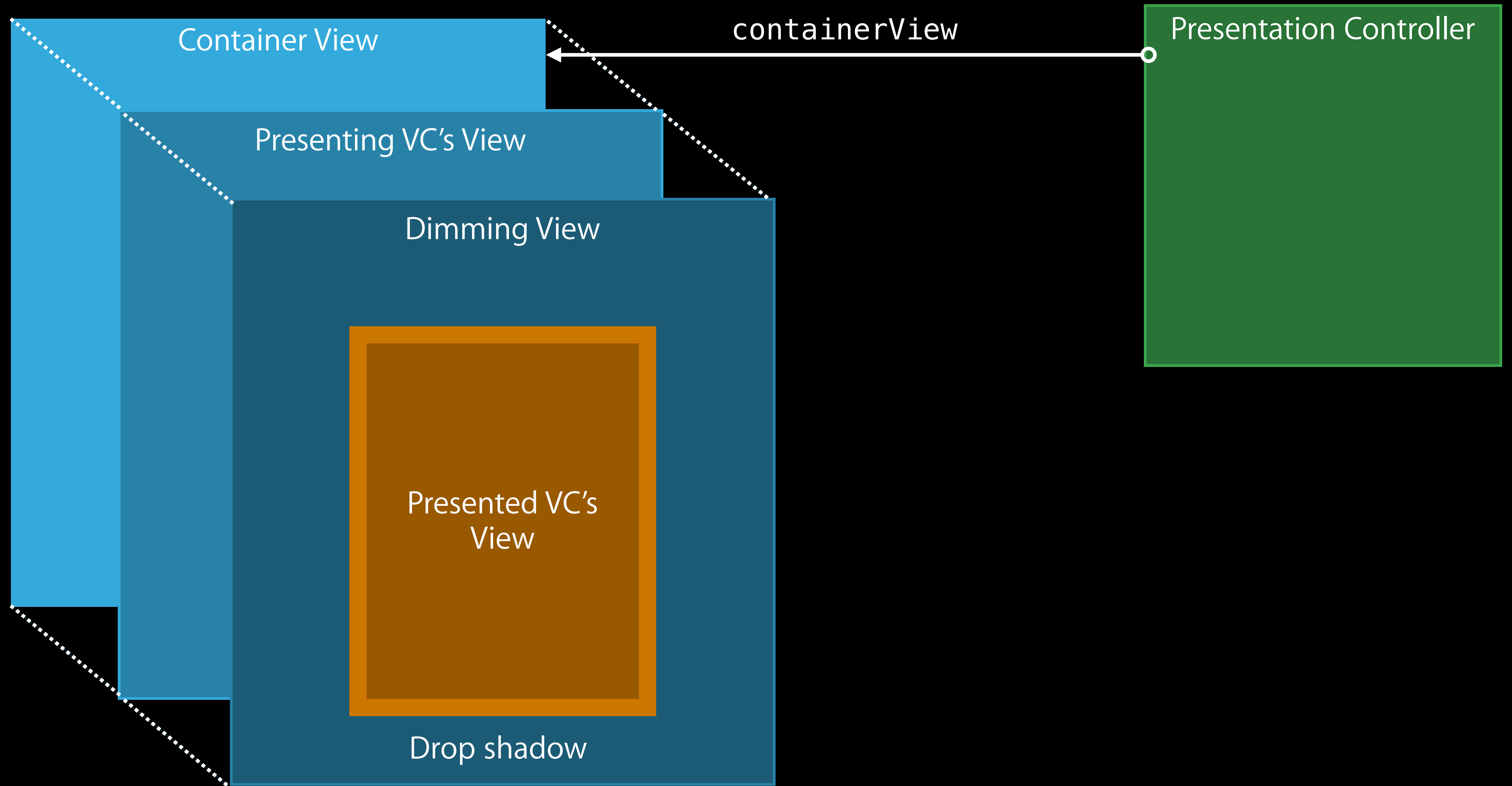
# Presentation Controllers



# Presentation Controllers

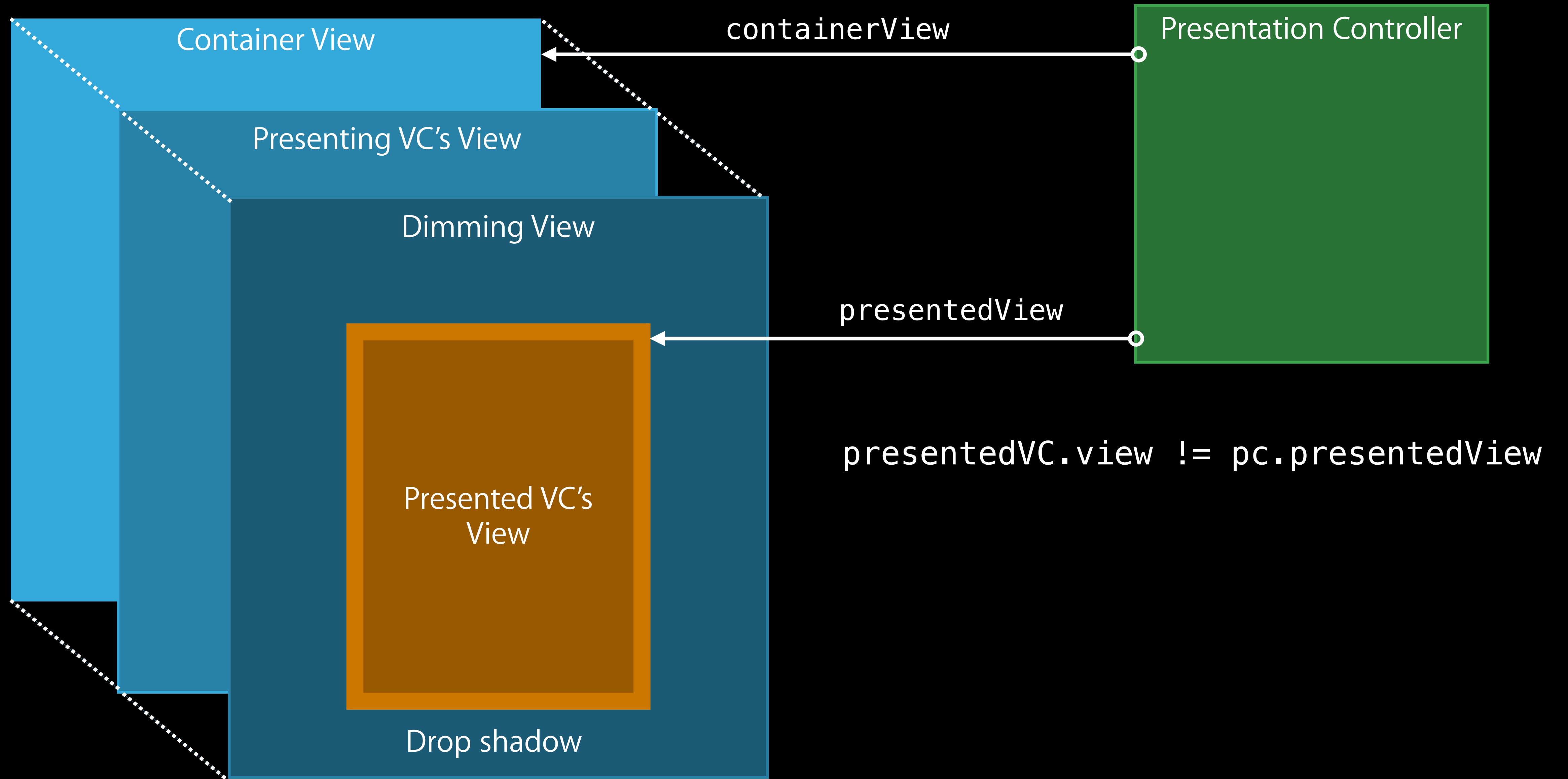


# Presentation Controllers





# Presentation Controllers



# Presentation Controllers

## Custom presentation controllers

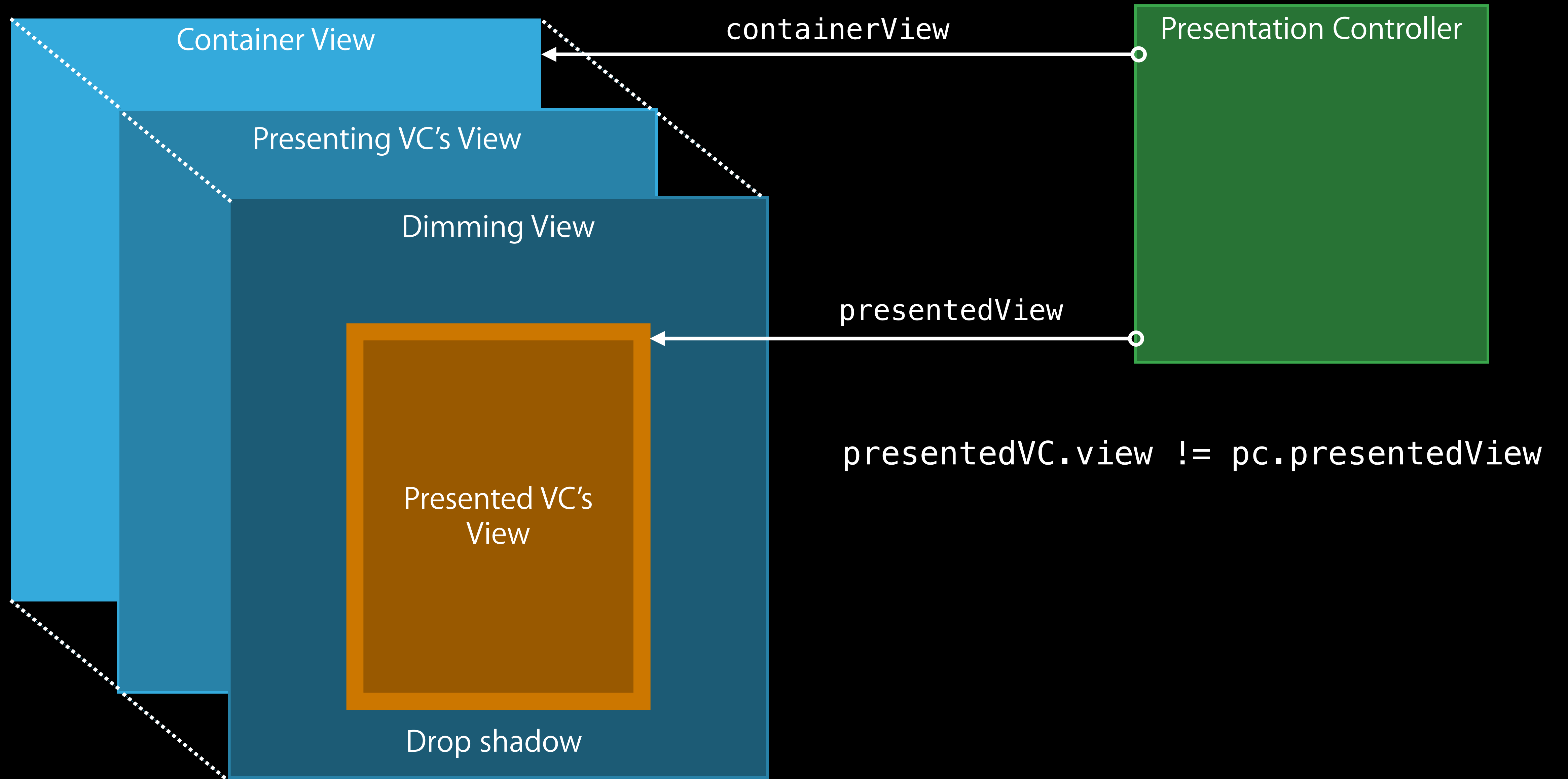


```
@protocol UIViewControllerContextTransitioning
```

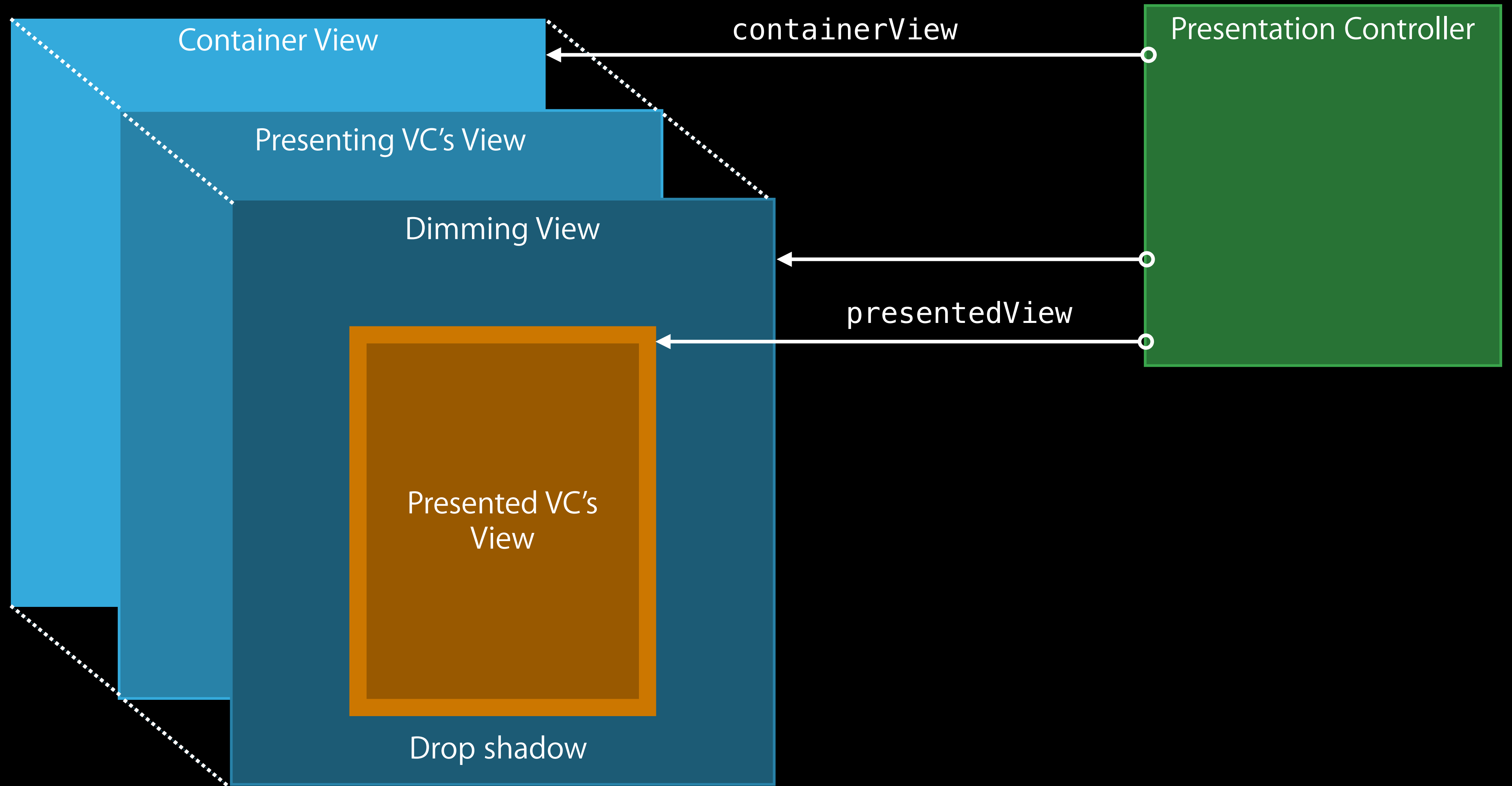
```
- (UIView *)viewForKey:(NSString *)key AVAILABLE_IOS(8_0);
```

```
@end
```

# Presentation Controllers

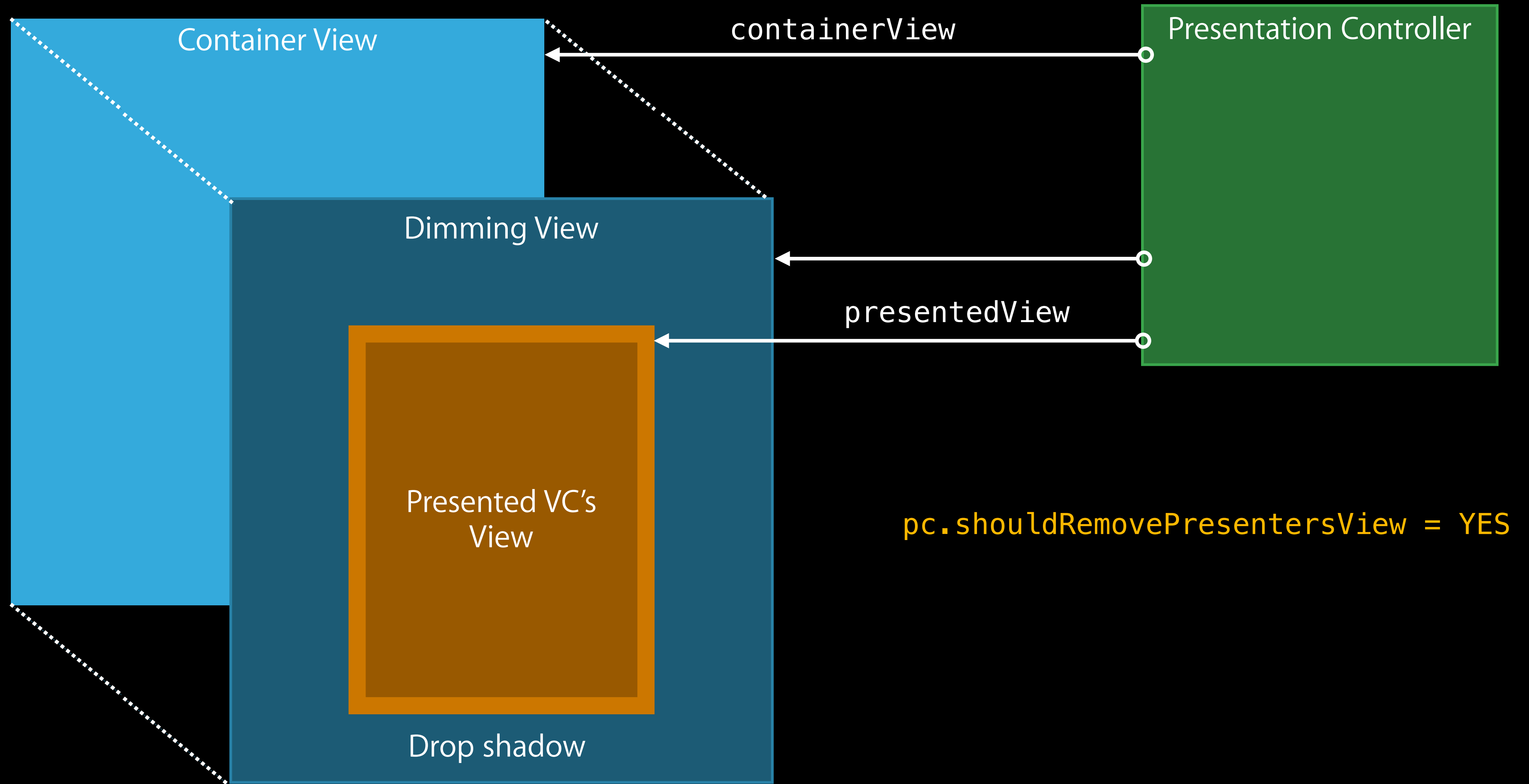


# Presentation Controllers

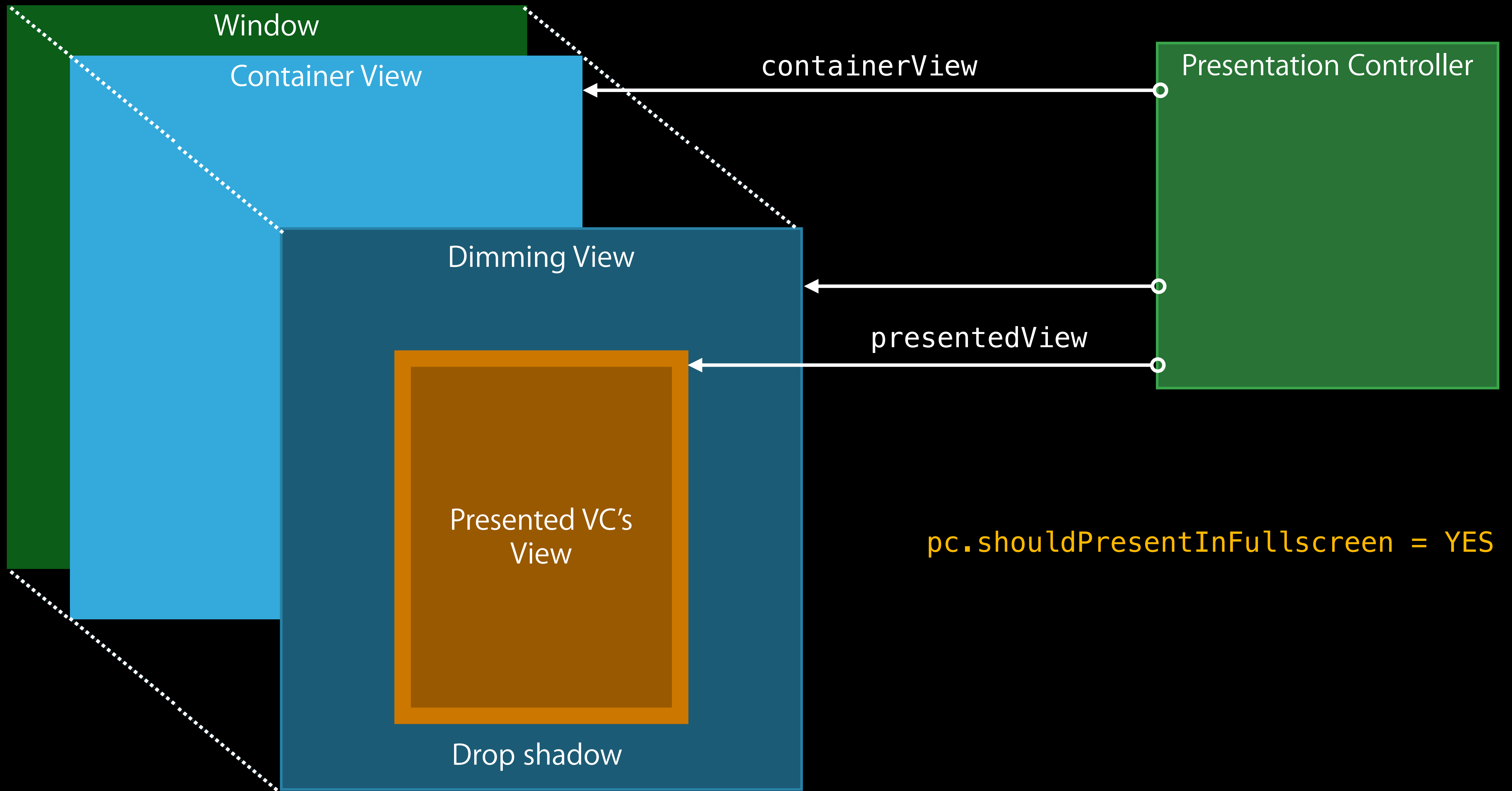




# Presentation Controllers



# Presentation Controllers



# Presentation Controllers

## Custom presentation controllers



```
@interface UIPresentationController : NSObject
    <UIAppearanceContainer, UITraitEnvironment, UIContentContainer>

@property(n nonatomic, readonly) UIView *containerView;
- (UIView *)presentedView;

- (BOOL)shouldRemovePresentersView;
- (BOOL)shouldPresentInFullscreen;

@end
```

# Presentation Controllers

Presentation styles



# Presentation Controllers

## Presentation styles

Previous iPad-only styles are available on the iPhone

- (But they adapt to full screen presentations)

# Presentation Controllers

## Presentation styles

Previous iPad-only styles are available on the iPhone

- (But they adapt to full screen presentations)

## New Presentation Styles

```
UIModalPresentationOverFullscreen;  
UIModalPresentationOverCurrentContext;  
UIModalPresentationPopover
```

# Presentation Controllers

## Presentation styles

Previous iPad-only styles are available on the iPhone

- (But they adapt to full screen presentations)

## New Presentation Styles

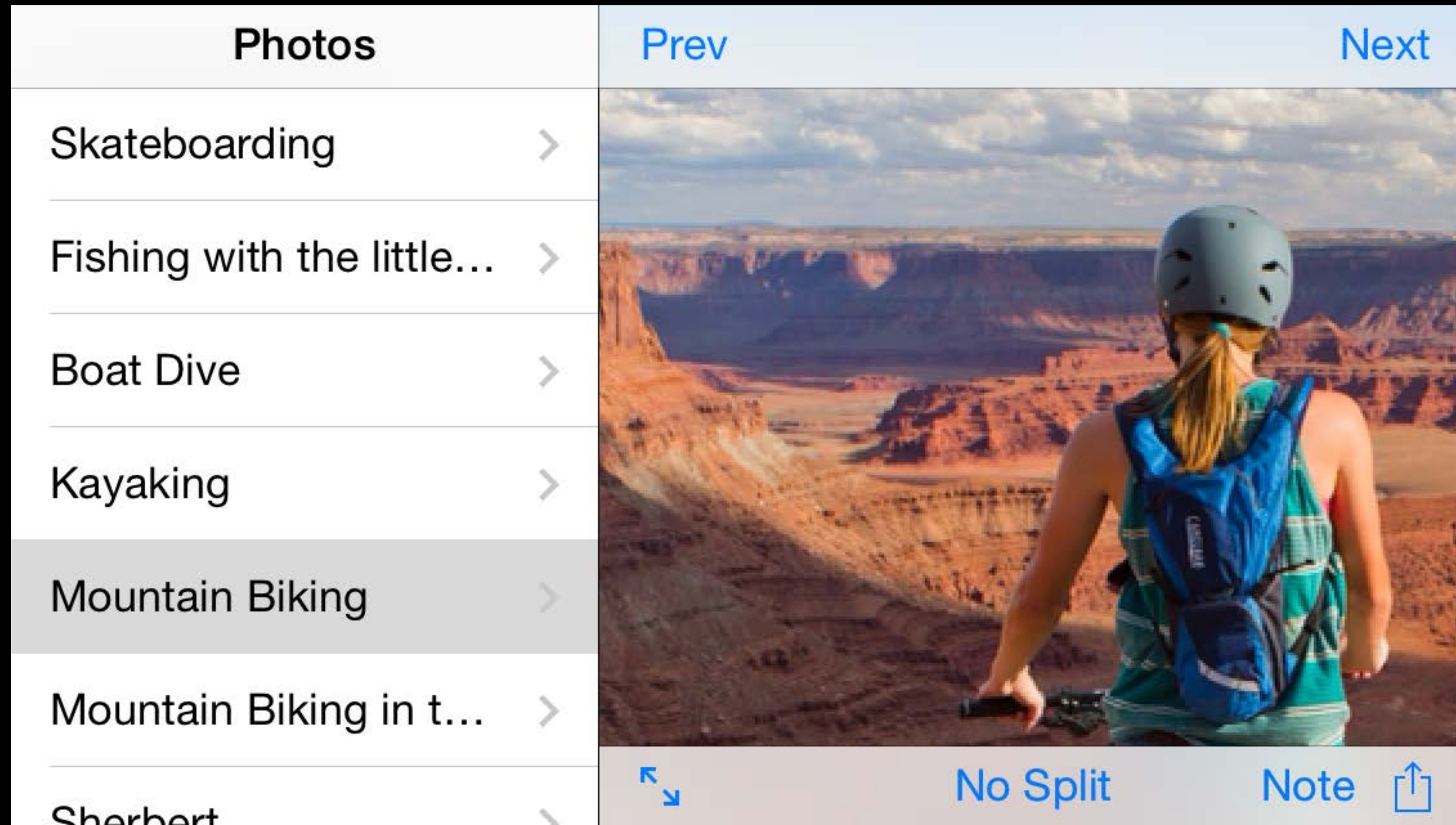
```
UIModalPresentationOverFullscreen;  
UIModalPresentationOverCurrentContext;  
UIModalPresentationPopover
```

All presentation styles have an associated presentation controller

```
-[UIViewController presentationController]  
-[UIViewController popoverPresentationController]
```

# Presentation Controllers

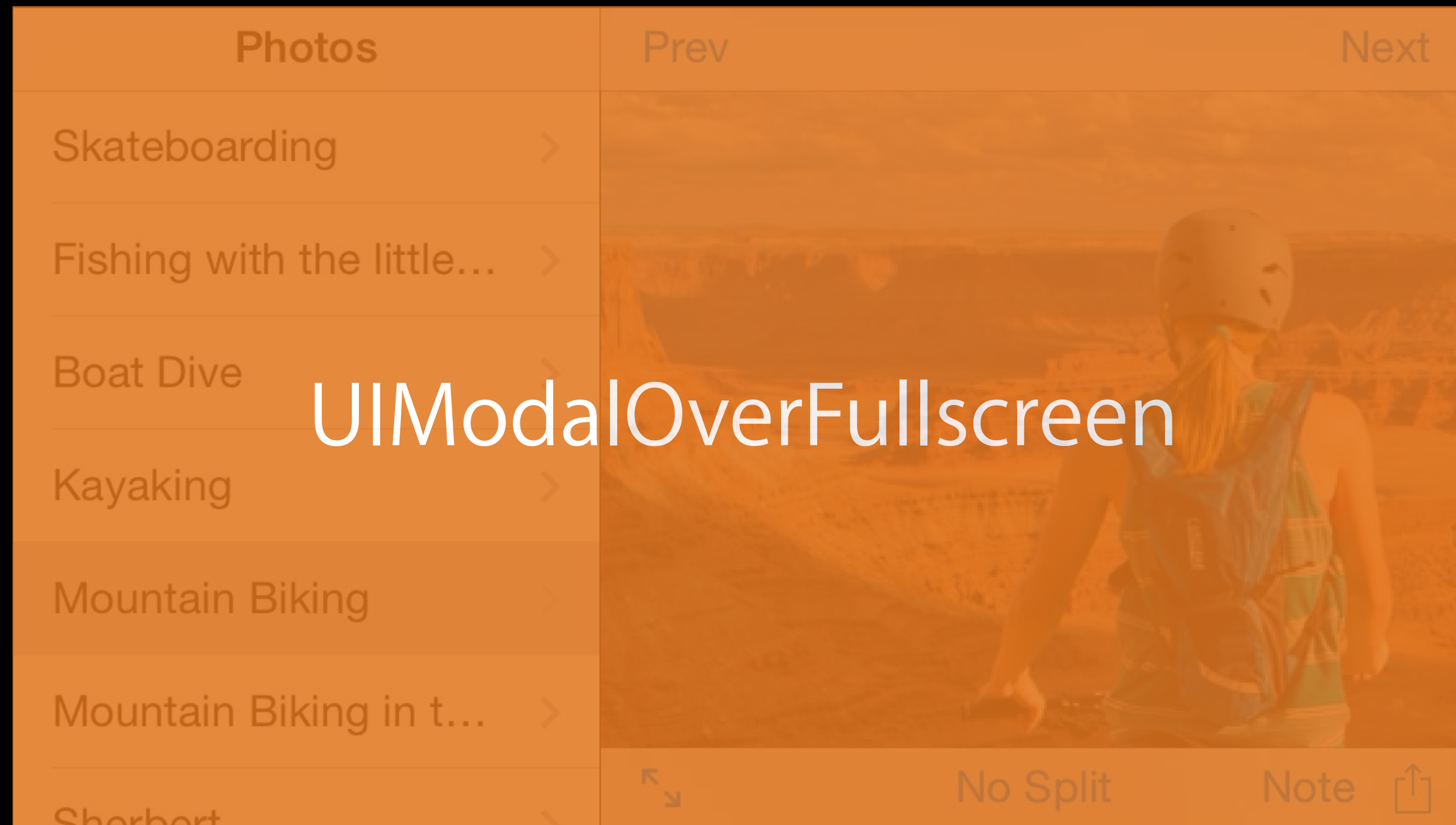
UIModalPresentationOverFullScreen





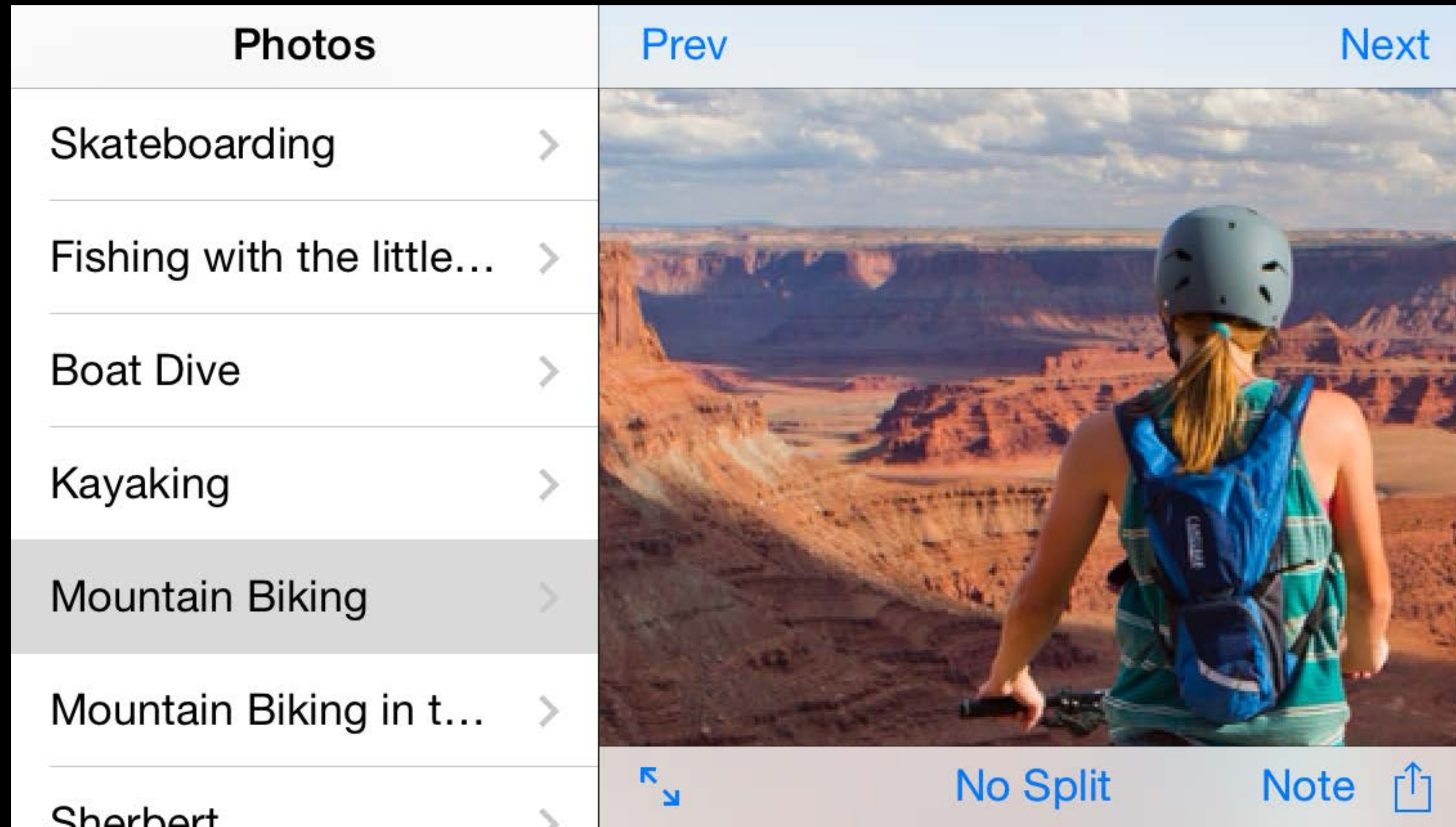
# Presentation Controllers

UIModalPresentationOverFullScreen



# Presentation Controllers

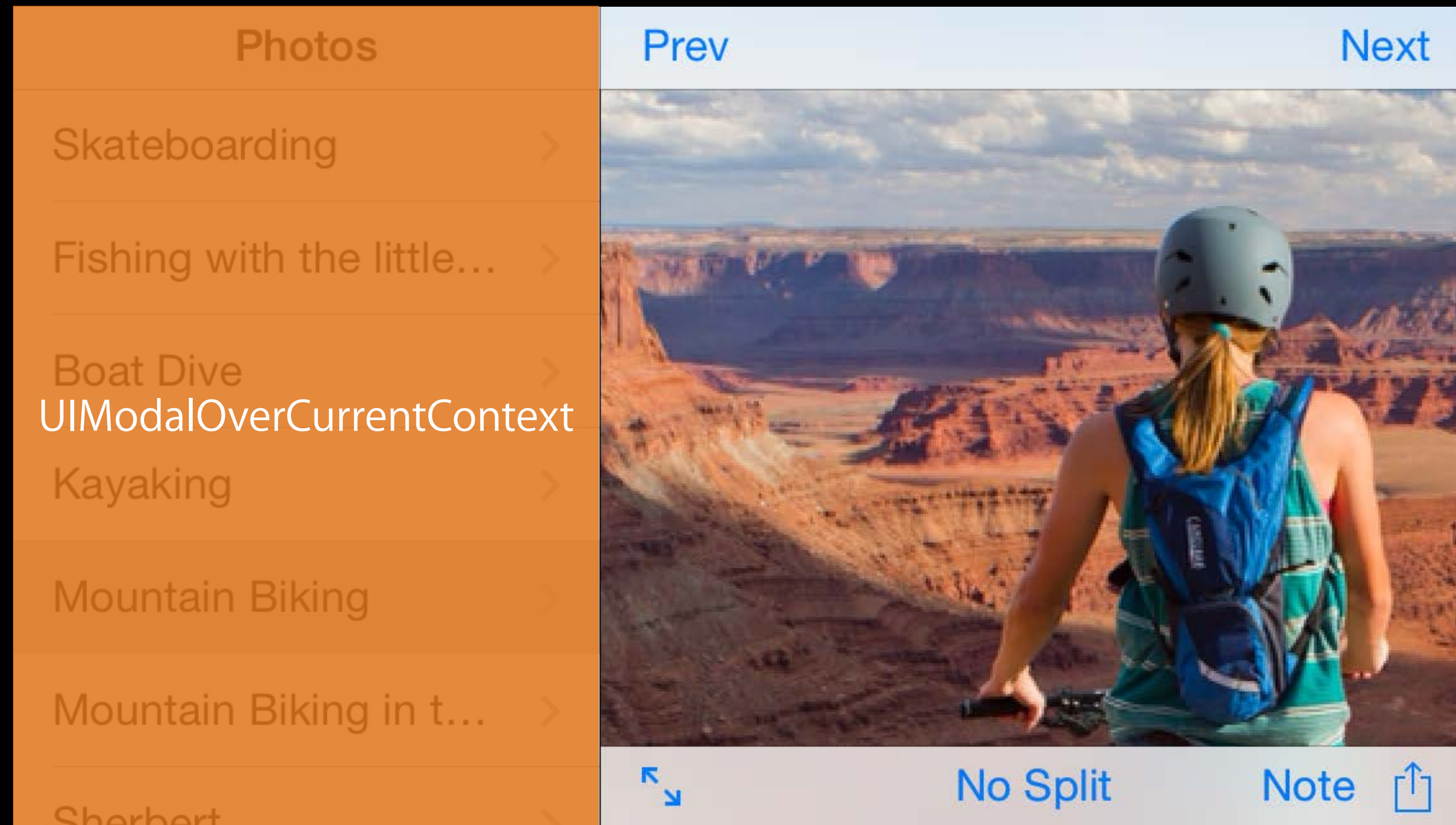
UIModalPresentationOverCurrentContext





# Presentation Controllers

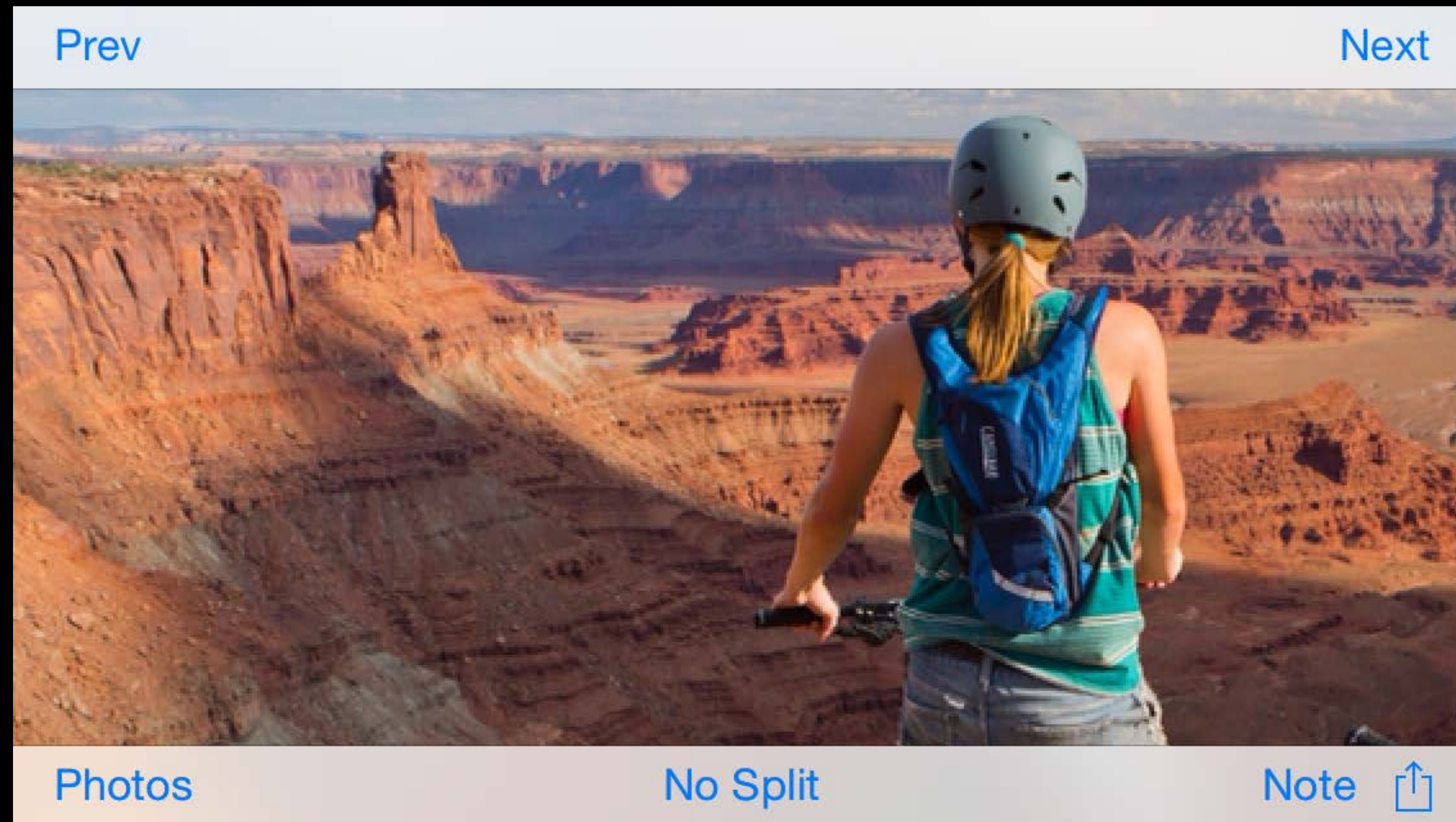
UIModalPresentationOverCurrentContext





# Presentation Controllers

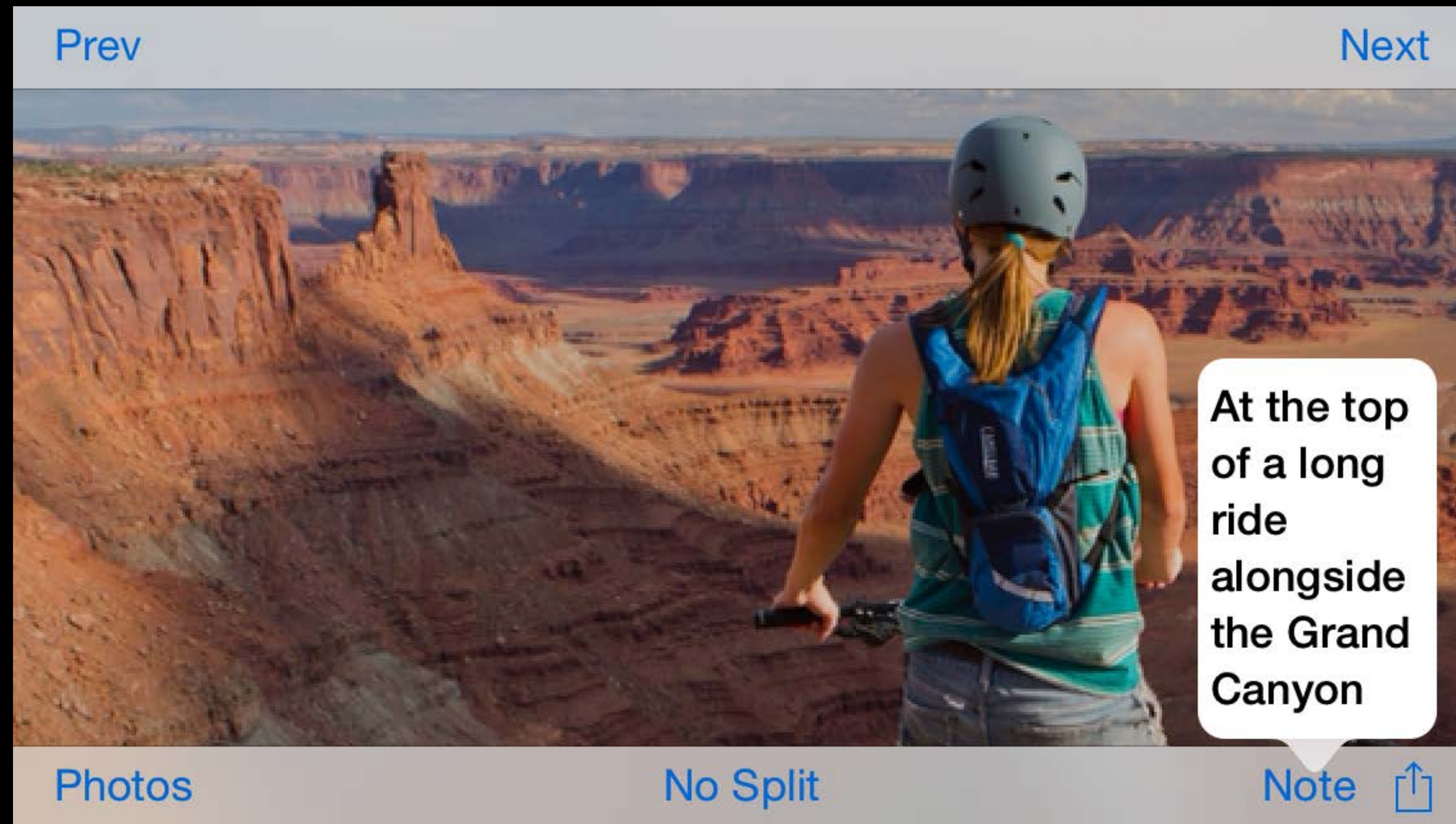
## UIModalPresentationPopover





# Presentation Controllers

## UIModalPresentationPopover



# Presentation Controllers

Adaptive Behavior

# Presentation Controllers

## Adaptive Behavior

Previous iPad-only styles adapt their style when in horizontally compact containers

- `UIModalPresentationFormSheet`
- `UIModalPresentationPageSheet`
- `UIModalPresentationPopover`
- `UIModalPresentationCustom`



# Presentation Controllers

## Adaptive Behavior

Previous iPad-only styles adapt their style when in horizontally compact containers

- `UIModalPresentationFormSheet`
- `UIModalPresentationPageSheet`
- `UIModalPresentationPopover`
- `UIModalPresentationCustom`

Supported adaptive presentation styles include

- `UIModalPresentationFullscreen`
- `UIModalPresentationOverFullscreen`
- `UIModalPresentationNone`



# Presentation Controllers

## Adaptive Behavior

Previous iPad-only styles adapt their style when in horizontally compact containers

- `UIModalPresentationFormSheet`
- `UIModalPresentationPageSheet`
- `UIModalPresentationPopover`
- `UIModalPresentationCustom`

Supported adaptive presentation styles include

- `UIModalPresentationFullscreen`
- `UIModalPresentationOverFullscreen`
- `UIModalPresentationNone`

Use the presentation controller's delegate to specify an adapted presentation style

# Presentation Controllers

## Adaptive presentations



```
@protocol UIAdaptivePresentationControllerDelegate <NSObject>
```

```
@optional
```

```
- (UIModalPresentationStyle)adaptivePresentationStyleForPresentationController:
```

```
- (UINavigationController *)presentationController:viewControllerForAdaptivePresentationStyle:
```

```
@end
```

```
@protocol UIPopoverPresentationControllerDelegate <UIAPCD>
```

```
@end
```

# Presentation Controllers

## Adaptive presentations



```
@protocol UIAdaptivePresentationControllerDelegate <NSObject>
```

```
@optional
```

```
- (UIModalPresentationStyle)adaptivePresentationStyleForPresentationController:
```

```
- (UINavigationController *)presentationController:viewControllerForAdaptivePresentationStyle:
```

```
@end
```

```
@protocol UIPopoverPresentationControllerDelegate <UIAPCD>
```

```
@end
```

# Presentation Controllers

## Adaptive presentations



```
@protocol UIAdaptivePresentationControllerDelegate <NSObject>
```

```
@optional
```

```
- (UIModalPresentationStyle)adaptivePresentationStyleForPresentationController:
```

```
- (UINavigationController *)presentationController:viewControllerForAdaptivePresentationStyle:
```

```
@end
```

```
@protocol UIPopoverPresentationControllerDelegate <UIAPCD>
```

```
@end
```



# Presentation Controllers

## Adaptive popovers

How do you present a popover?

# Presentation Controllers

## Adaptive popovers

How do you present a popover?

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.barButtonItem = self.commentButton;  
pc.permittedArrowDirections = UIPopoverArrowDirectionAny;  
vc.preferredContentSize = CGSizeMake(...);  
[self presentViewController: vc animated:YES completion:nil];
```

# Presentation Controllers

## Adaptive popovers

How do you present a popover?

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.barButtonItem = self.commentButton;  
pc.permittedArrowDirections = UIPopoverArrowDirectionAny;  
vc.preferredContentSize = CGSizeMake(...);  
[self presentViewController: vc animated:YES completion:nil];
```

# Presentation Controllers

## Adaptive popovers

How do you present a popover?

```
vc.modalPresentationStyle = UIModalPresentationPopover;
UIPopoverPresentationController *pc = [vc popoverPresentationController];
pc.barButtonItem = self.commentButton;
pc.permittedArrowDirections = UIPopoverArrowDirectionAny;
vc.preferredContentSize = CGSizeMake(...);
[self presentViewController: vc animated:YES completion:nil];
```



# Presentation Controllers

## Adaptive popovers

How do you present a popover?

```
vc.modalPresentationStyle = UIModalPresentationPopover;
UIPopoverPresentationController *pc = [vc popoverPresentationController];
pc.barButtonItem = self.commentButton;
pc.permittedArrowDirections = UIPopoverArrowDirectionAny;
vc.preferredContentSize = CGSizeMake(...);
[self presentViewController: vc animated:YES completion:nil];
```

# Presentation Controllers

## Adaptive popovers

How do you present a popover?

```
vc.modalPresentationStyle = UIModalPresentationPopover;
UIPopoverPresentationController *pc = [vc popoverPresentationController];
pc.barButtonItem = self.commentButton;
pc.permittedArrowDirections = UIPopoverArrowDirectionAny;
vc.preferredContentSize = CGSizeMake(...);
[self presentViewController: vc animated:YES completion:nil];
```

# Presentation Controllers

Adaptive popovers



# Presentation Controllers

Adaptive popovers

This is a popover presentation!





There are a few problems with this

# Presentation Controllers

Adaptive popovers



# Presentation Controllers

Adaptive popovers



# Presentation Controllers

## Adaptive popovers

- Underlaps the status bar





# Presentation Controllers

## Adaptive popovers

- Underlaps the status bar
- Looks real bad



# Presentation Controllers

## Adaptive popovers

- Underlaps the status bar
- Looks real bad
- There is no way to dismiss the popover!



# Presentation Controllers

Adaptive popovers

# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```



# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```

- (UIModalPresentationStyle) **adaptivePresentationStyleForPresentationController:**
- Have it return UIModalPresentationOverFullscreen
- Use UIViewEffectView in the presented view controller
- Adjust the content position by accessing the presentedController

# Presentation Controllers

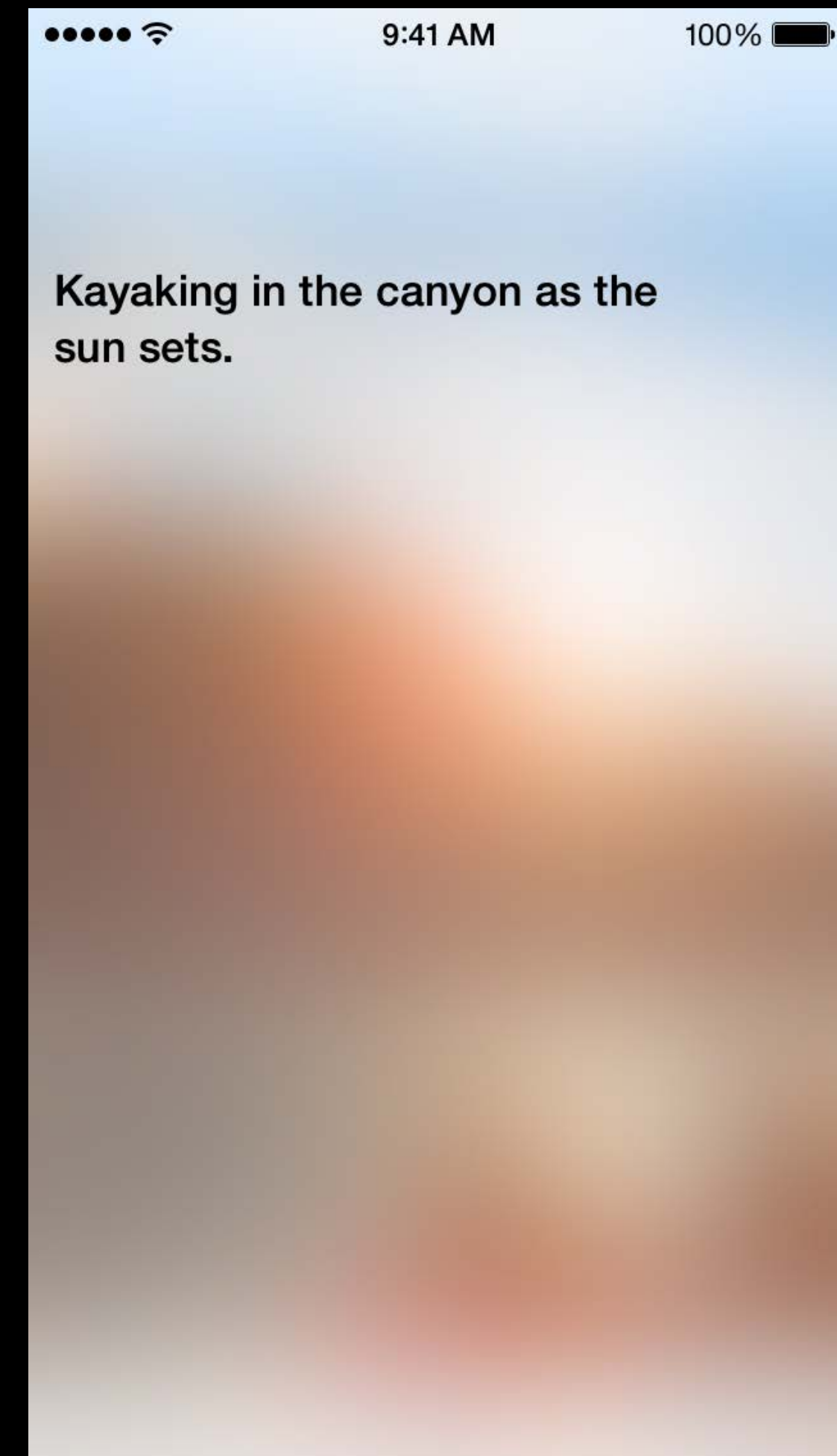
Adaptive popovers



# Presentation Controllers

## Adaptive popovers

This is also a popover presentation  
with a different adaptation

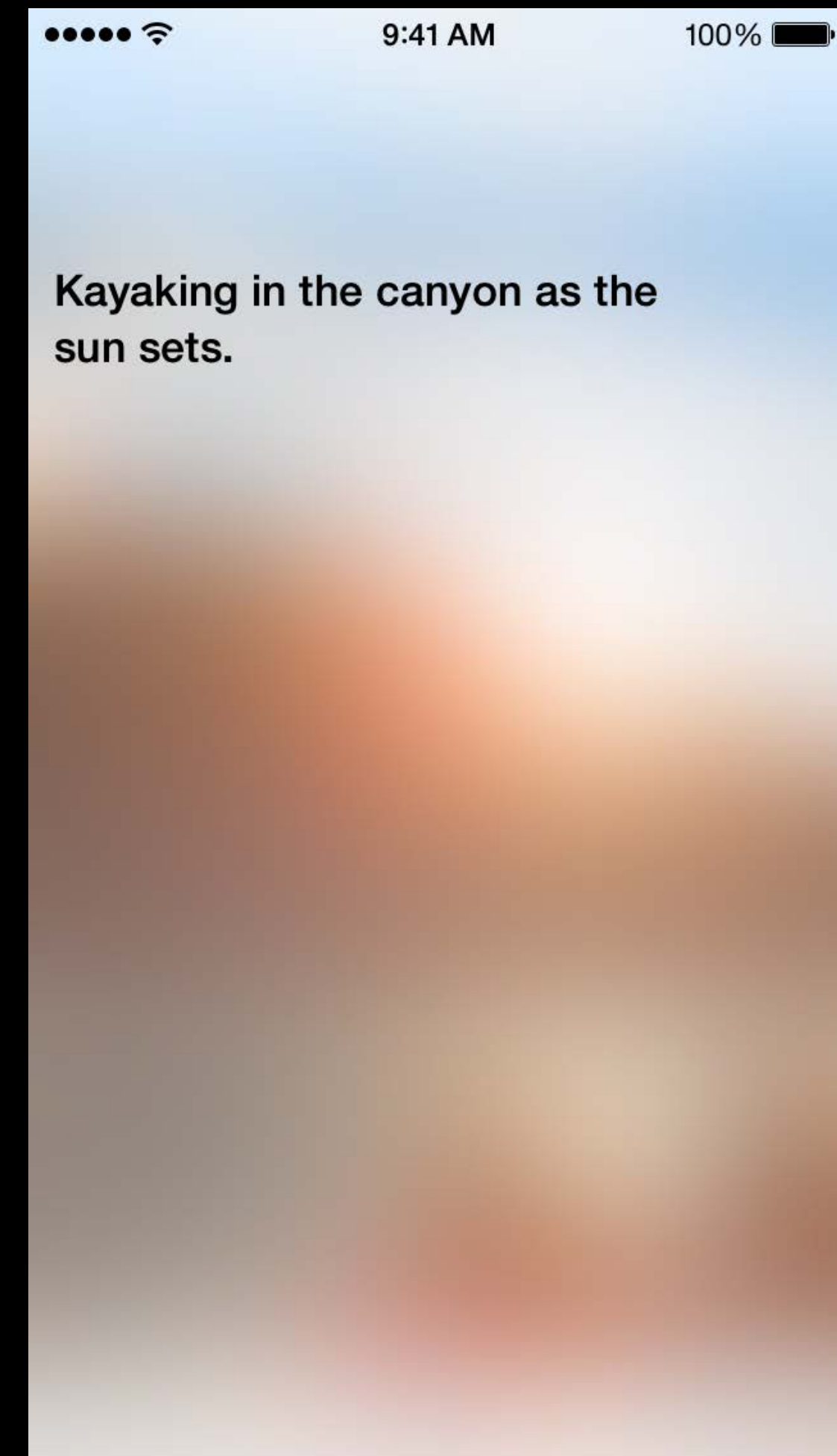


# Presentation Controllers

## Adaptive popovers

This is also a popover presentation with a different adaptation

- But still no way to dismiss the popover!





# Presentation Controllers

Adaptive popovers

# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```

# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```

- (UINavigationController \*)`presentationController:viewControllerForAdaptivePresentationStyle:`
  - Have it return a UINavigationController whose root VC is the presentedController
  - Add a dismiss button to the navigation bar.

# Presentation Controllers

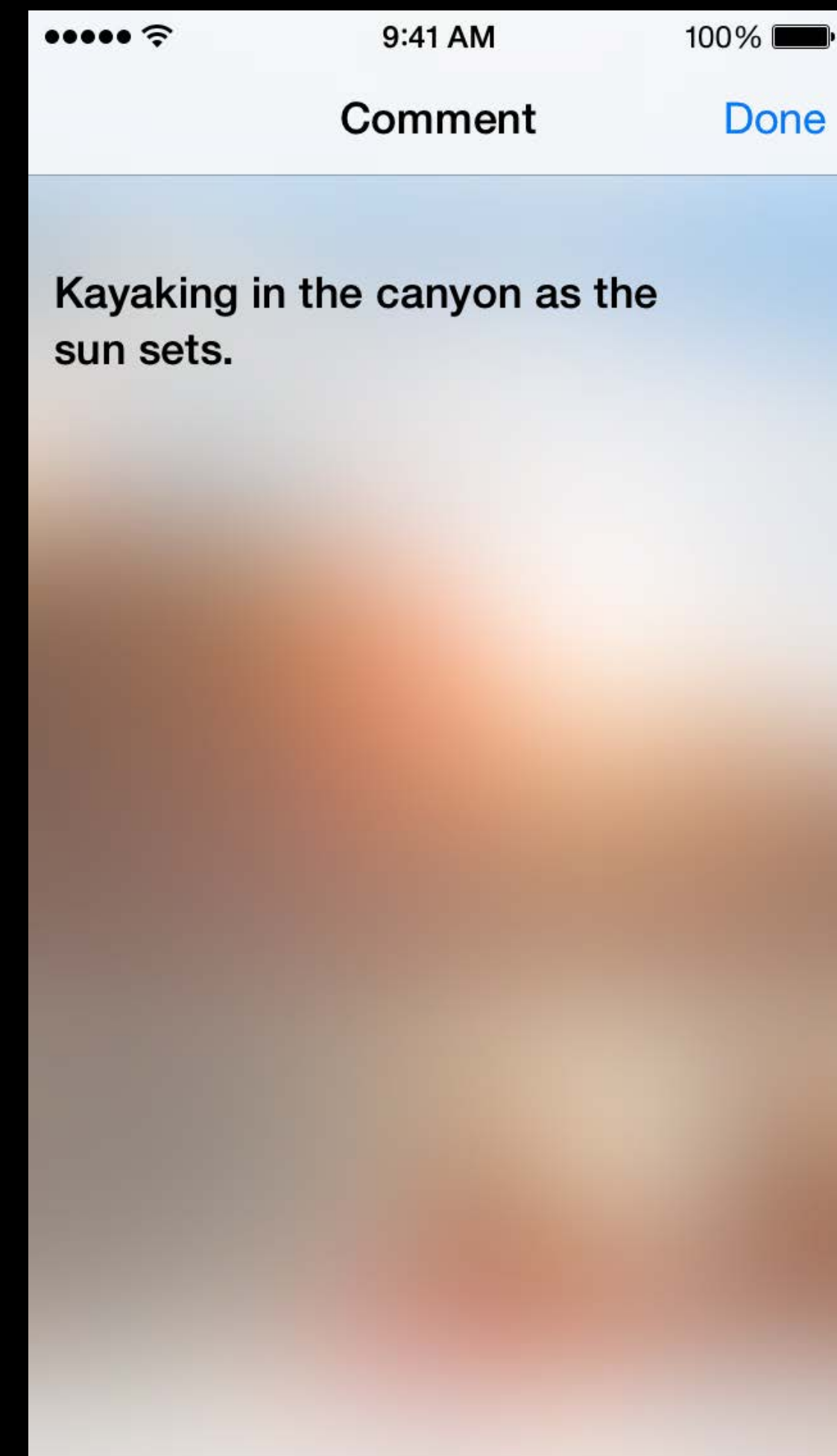
Adaptive popovers





# Presentation Controllers

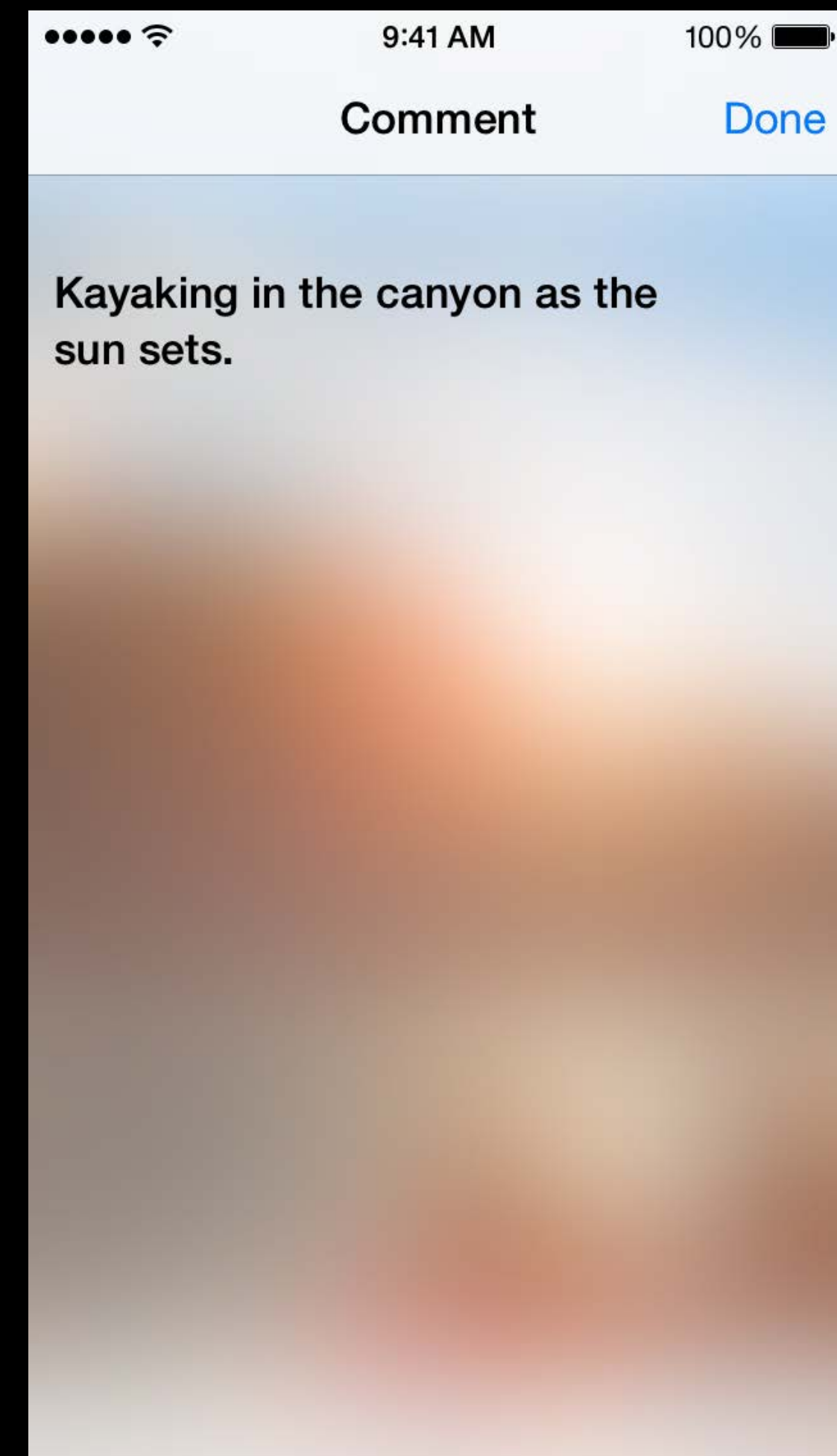
Adaptive popovers



# Presentation Controllers

## Adaptive popovers

Looks great. But what if I really want a popover?



# Presentation Controllers

Adaptive popovers

# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```



# Presentation Controllers

## Adaptive popovers

Set the delegate on the presentation controller

```
vc.modalPresentationStyle = UIModalPresentationPopover;  
UIPopoverPresentationController *pc = [vc popoverPresentationController];  
pc.delegate = self;  
...
```

- (UIModalPresentationStyle)adaptivePresentationStyleForPresentationController:
- Have it return UIModalPresentationNone

# Presentation Controllers

Adaptive popovers



# Presentation Controllers

Adaptive popovers



# Presentation Controllers

What did we learn?



# Presentation Controllers

What did we learn?

Presentation controllers enhance the existing API for creating custom presentations

# Presentation Controllers

What did we learn?

Presentation controllers enhance the existing API for creating custom presentations

View controller presentations are associated with a `UIPresentationController` object

- `-[UIViewController presentationController]`
- `-[UIViewController popoverPresentationController]`

# Presentation Controllers

What did we learn?

Presentation controllers enhance the existing API for creating custom presentations

View controller presentations are associated with a UIPresentationController object

- [UIViewController presentationController]
- [UIViewController popoverPresentationController]

Presentation controllers can easily adapt to horizontal size class changes

- <UIAdaptivePresentationControllerDelegate>
- <UIPopoverPresentationControllerDelegate>

# Presentation Controllers

What did we learn?

Presentation controllers enhance the existing API for creating custom presentations

View controller presentations are associated with a UIPresentationController object

- [UIViewController presentationController]
- [UIViewController popoverPresentationController]

Presentation controllers can easily adapt to horizontal size class changes

- <UIAdaptivePresentationControllerDelegate>
- <UIPopoverPresentationControllerDelegate>

You can create your own presentation controller's which will adapt if `shouldPresentInFullscreen` returns YES.



# Transition Coordinators

What's a "transition coordinator"?

What's a "transition coordinator"?

<UIViewControllerTransitionCoordinator>  
(UIVCTC)

# Transition Coordinators



# Transition Coordinators

Introduced in iOS 7 for custom transitions

```
[UIViewController transitionCoordinator]
```

```
- (BOOL)animateAlongsideTransition: (void (^)(id <UIVCTCContext>context))a  
                                completion: (void (^)(id <UIVCTCContext>context))c;
```

They are part of the UIViewController adaptive UI story

# Transition Coordinators



viewWillTransitionToSize:withTransitionCoordinator:

```
@protocol UIContentContainer <NSObject>
```

```
    @property (nonatomic, assign) CGSize preferredContentSize;
```

```
    - (void)preferredContentSizeDidChangeForChildContentContainer:
```

```
    - (void)systemLayoutFittingSizeDidChangeForChildContentContainer:
```

```
    - (void)sizeForChildContentContainer:withParentContainerSize:
```

```
    - (void)willTransitionToTraitCollection:withTransitionCoordinator:
```

```
    - (void)viewWillTransitionToSize:withTransitionCoordinator:
```

```
@end
```

# Transition Coordinators



viewWillTransitionToSize:withTransitionCoordinator:

```
@protocol UIContentContainer <NSObject>
```

```
    @property (nonatomic, assign) CGSize preferredContentSize;
```

```
    - (void)preferredContentSizeDidChangeForChildContentContainer:
```

```
    - (void)systemLayoutFittingSizeDidChangeForChildContentContainer:
```

```
    - (void)sizeForChildContentContainer:withParentContainerSize:
```

```
    - (void)willTransitionToTraitCollection:withTransitionCoordinator:
```

```
    - (void)viewWillTransitionToSize:withTransitionCoordinator:
```

```
@end
```

# Transition Coordinators



viewWillTransitionToSize:withTransitionCoordinator:

```
@protocol UIContentContainer <NSObject>
```

```
@property (nonatomic, assign) CGSize preferredContentSize;  
- (void)preferredContentSizeDidChangeForChildContentContainer:  
- (void)systemLayoutFittingSizeDidChangeForChildContentContainer:  
  
- (void)sizeForChildContentContainer:withParentContainerSize:  
- (void)willTransitionToTraitCollection:withTransitionCoordinator:  
- (void)viewWillTransitionToSize:withTransitionCoordinator:
```

```
@end
```



“For the self aware App, a device rotation is only an animated bounds change.”

Anonymous



9:41 AM

100%

Photos Prev

Next



Split

Note






◀ Photos Prev

Next



Split

Note 

# Transition Coordinators

## Rotation callbacks

`willRotateToInterfaceOrientation:duration:`

`willAnimateRotationToInterfaceOrientation:duration:`

`didRotateFromInterfaceOrientation:`

`interfaceOrientation`



# Transition Coordinators

## Rotation callbacks

```
willRotateToInterfaceOrientation:duration:  
willAnimateRotationToInterfaceOrientation:duration:  
didRotateFromInterfaceOrientation:  
interfaceOrientation
```



# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

Is a replacement for the deprecated rotation callbacks

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

Is a replacement for the deprecated rotation callbacks



# Transition Coordinators

## Rotation callbacks



```
@protocol UIViewControllerTransitionCoordinatorContext
```

```
– (CGAffineTransform)targetTransform;
```

```
– (UIView *)viewForKey:(NSString *)key;
```

```
@end
```

# Transition Coordinators

## Rotation callbacks



```
@protocol UIViewControllerTransitionCoordinatorContext
```

```
– (CGAffineTransform)targetTransform;
```

```
– (UIView *)viewForKey:(NSString *)key;
```

```
@end
```

# Transition Coordinators

## Rotation callbacks

```
- (void) viewWillTransitionToSize:(CGSize)s
    withTransitionCoordinator:(UIVCTC)t {
orientation = [self orientationFromTransform: [t targetTransform]];
oldOrientation = [[UIApplication sharedApplication] statusBarOrientation];

[self myWillRotateToInterfaceOrientation:orientation duration: duration];

[t animateAlongsideTransition:^(id <UIVCTCContext>) {
    [self myWillAnimateRotationToInterfaceOrientation:orientation
                                     duration:duration];
    }
completion:^(id <UIVCTCContext>) {
    [self myDidAnimateFromInterfaceOrientation:oldOrientation];
    }];
}
```

# Transition Coordinators

## Rotation callbacks

```
- (void) viewWillTransitionToSize:(CGSize)s
    withTransitionCoordinator:(UIVCTC)t {
orientation = [self orientationFromTransform: [t targetTransform]];
oldOrientation = [[UIApplication sharedApplication] statusBarOrientation];

[self myWillRotateToInterfaceOrientation:orientation duration: duration];

[t animateAlongsideTransition:^(id <UIVCTCContext>) {
    [self myWillAnimateRotationToInterfaceOrientation:orientation
                                     duration:duration];
    }
completion:^(id <UIVCTCContext>) {
    [self myDidAnimateFromInterfaceOrientation:oldOrientation];
    }];
}
```



# Transition Coordinators

## Rotation callbacks

```
- (void) viewWillTransitionToSize:(CGSize)s
    withTransitionCoordinator:(UIVCTC)t {
orientation = [self orientationFromTransform: [t targetTransform]];
oldOrientation = [[UIApplication sharedApplication] statusBarOrientation];

[self myWillRotateToInterfaceOrientation:orientation duration: duration];

[t animateAlongsideTransition:^(id <UIVCTCContext>) {
    [self myWillAnimateRotationToInterfaceOrientation:orientation
                                     duration:duration];
}
 completion:^(id <UIVCTCContext>) {
    [self myDidAnimateFromInterfaceOrientation:oldOrientation];
}];
}
```

# Transition Coordinators

## Rotation callbacks

```
- (void) viewWillTransitionToSize:(CGSize)s
    withTransitionCoordinator:(UINavigationController) t {
orientation = [self orientationFromTransform: [t targetTransform]];
oldOrientation = [[UIApplication sharedApplication] statusBarOrientation];

[self myWillRotateToInterfaceOrientation:orientation duration: duration];

[t animateAlongsideTransition:^(id <UINavigationControllerContext>) {
    [self myWillAnimateRotationToInterfaceOrientation:orientation
                                     duration:duration];
    }
    completion:^(id <UINavigationControllerContext>) {
        [self myDidAnimateFromInterfaceOrientation:oldOrientation];
    }];
}
```

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

The legacy rotation methods are still available

- Just don't implement the method



# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

The legacy rotation methods are still available

- Just don't implement the method

Most view controller transitions are immediate when called from within the dynamic scope of this method

- (Not in the animate alongside blocks)

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

The legacy rotation methods are still available

- Just don't implement the method

Most view controller transitions are immediate when called from within the dynamic scope of this method

- (Not in the animate alongside blocks)

Call super in order to forward to descendent view controllers

# Transition Coordinators

`viewWillTransitionToSize:withTransitionCoordinator:`

The legacy rotation methods are still available

- Just don't implement the method

Most view controller transitions are immediate when called from within the dynamic scope of this method

- (Not in the animate alongside blocks)

Call super in order to forward to descendent view controllers

Only necessary when you need to do a special size transition





Scroll  
Direction







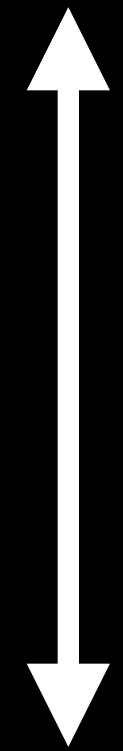
Scroll  
Direction







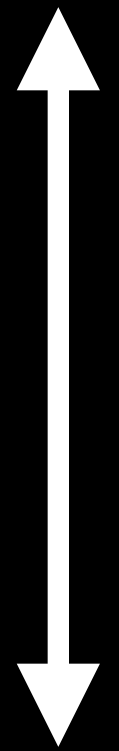
Scroll  
Direction







Scroll  
Direction







Scroll Direction





# Transition Coordinators

viewWillTransitionToSize:withTransitionCoordinator:

**Before `animateAlongside:withTransitionCoordinator:`**

```
CGAffineTransform transform = [coordinator targetTransform];  
CGAffineTransform invertedRotation = CGAffineTransformInvert(transform);  
CGRect currentBounds = self.view.bounds;
```

# Transition Coordinators

viewWillTransitionToSize:withTransitionCoordinator:

**Before animateAlongside:withTransitionCoordinator:**

```
CGAffineTransform transform = [coordinator targetTransform];  
CGAffineTransform invertedRotation = CGAffineTransformInvert(transform);  
CGRect currentBounds = self.view.bounds;
```

**Alongside block**

```
self.view.transform = CGAffineTransformConcat(self.view.transform, invertedRotation);  
_counterRotation = CGAffineTransformConcat(_counterRotation, transform);  
self.view.bounds = currentBounds;
```

# Transition Coordinators

viewWillTransitionToSize:withTransitionCoordinator:

**Before animateAlongside:withTransitionCoordinator:**

```
CGAffineTransform transform = [coordinator targetTransform];
CGAffineTransform invertedRotation = CGAffineTransformInvert(transform);
CGRect currentBounds = self.view.bounds;
```

**Alongside block**

```
self.view.transform = CGAffineTransformConcat(self.view.transform, invertedRotation);
_counterRotation = CGAffineTransformConcat(_counterRotation, transform);
self.view.bounds = currentBounds;
```

**Completion block**

```
[UIView animateWithDuration:.5 animations:^(
    for(PNCollectionViewCell *cell in [self.collectionView visibleCells]){
        cell.contentView.transform = _counterRotation;
    }
}];
```

# Transition Coordinators

What did we learn?



# Transition Coordinators

What did we learn?

Transition coordinators are being used in iOS 8 adaptive APIs

```
viewWillTransitionToSize:transitionCoordinator:
```

```
willTransitionToTraitCollection:transitionCoordinator:
```

# Transition Coordinators

What did we learn?

Transition coordinators are being used in iOS 8 adaptive APIs

```
viewWillTransitionToSize:transitionCoordinator:  
willTransitionToTraitCollection:transitionCoordinator:
```

You may use a transition coordinator in response to

```
preferredContentSizeDidChange:
```

# Transition Coordinators

What did we learn?

Transition coordinators are being used in iOS 8 adaptive APIs

```
viewWillTransitionToSize:transitionCoordinator:  
willTransitionToTraitCollection:transitionCoordinator:
```

You may use a transition coordinator in response to

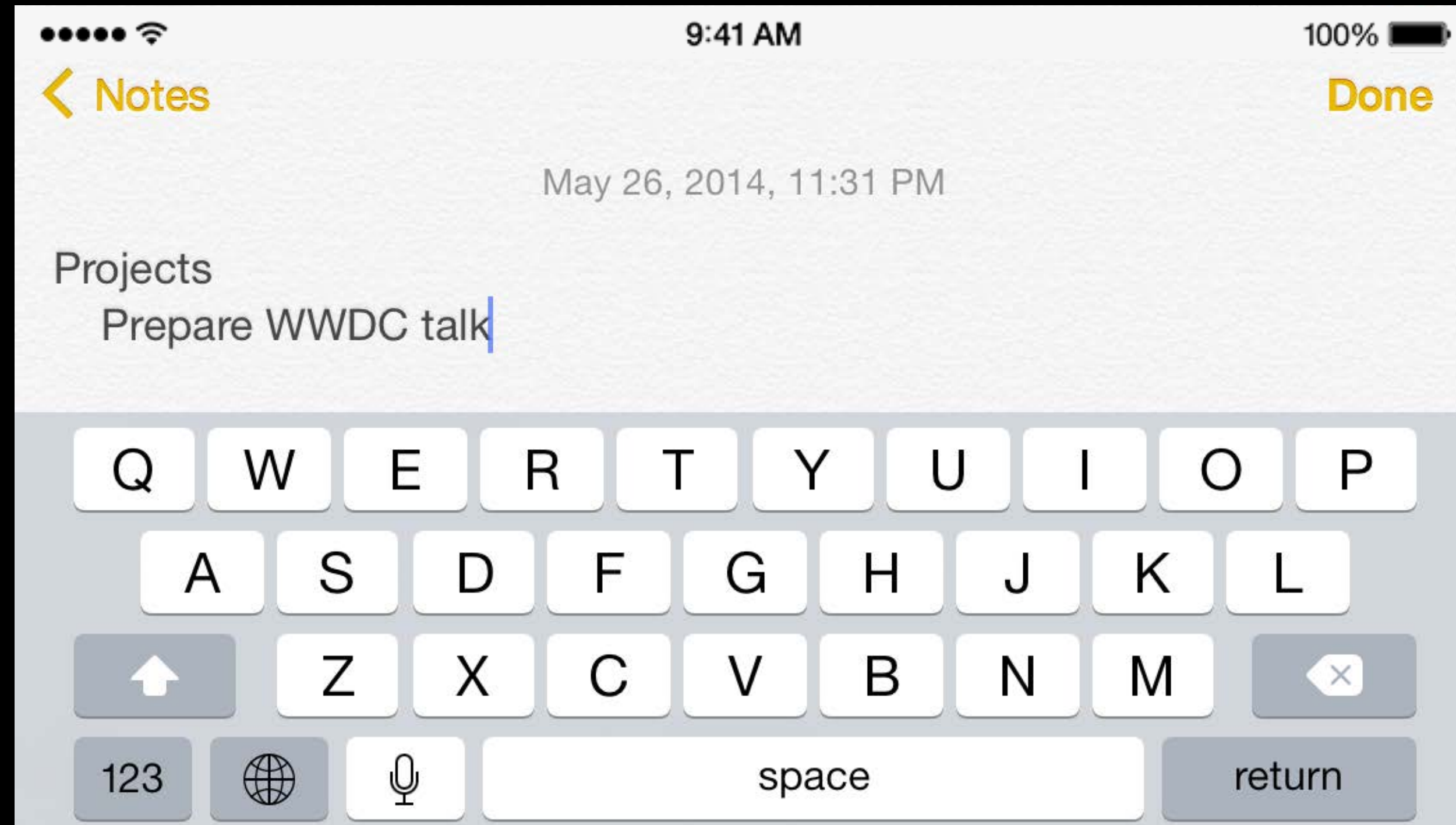
```
preferredContentSizeDidChange:
```

Rotation callbacks are being deprecated

# Screen Coordinates



# Screen Orientation

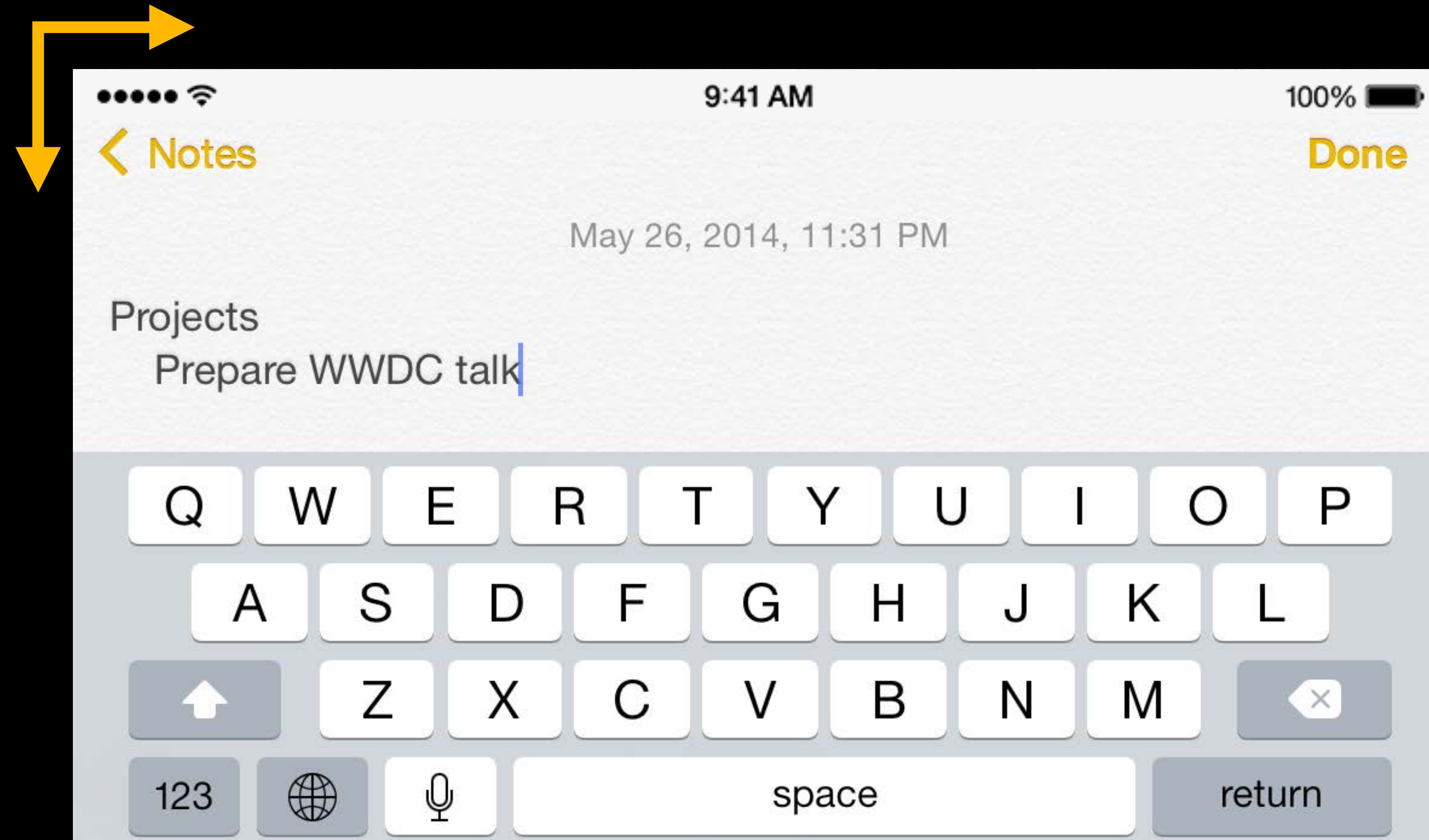


Keyboard Size:  
(162,568)



`[[UIScreen mainScreen] bounds] = {320,568}`

# Interface Orientation



Keyboard Size:  
(568,162)

```
[[UIScreen mainScreen] bounds] = {568,320}
```

# iOS UIViewControllers

## Changes to screen coordinates

UIScreen is now interface oriented

- [UIScreen bounds] now interface-oriented

- [UIScreen applicationFrame] now interface-oriented

Status bar frame notifications are interface-oriented

Keyboard frame notifications are interface-oriented

# iOS UIViewControllers

## Changes to screen coordinates



```
@protocol UICoordinateSpace
```

```
– (CGPoint)convertPoint:(CGPoint)  
    toCoordinateSpace:(id <UICoordinateSpace>)space;
```

```
– (CGRect)convertRect:(CGRect)  
    toCoordinateSpace:(id <UICoordinateSpace>)space;
```

```
@property(readonly, nonatomic) CGRect bounds;
```

```
@end
```



# iOS UIViewControllers

## Changes to screen coordinates



```
UIView <UICoordinateSpace>
```

```
@interface UIScreen
```

```
    @property (readonly) id <UICoordinateSpace> coordinateSpace;
```

```
    @property (readonly) id <UICoordinateSpace> fixedCoordinateSpace;
```

```
@end
```

# iOS UIViewControllers

## Changes to screen coordinates



```
UIView <UICoordinateSpace>
```

```
@interface UIScreen
```

```
    @property (readonly) id <UICoordinateSpace> coordinateSpace;
```

```
    @property (readonly) id <UICoordinateSpace> fixedCoordinateSpace;
```

```
@end
```

```
[view convertPoint:p
```

```
    toCoordinateSpace:view.window.screen.fixedCoordinateSpace];
```

```
[view.window.screen.fixedCoordinateSpace convertPoint:p
```

```
    toCoordinateSpace:view];
```

# What We Discussed Today

# What We Discussed Today

Trait collections and how view controllers may override them



# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

UIPresentationControllers and new presentation styles

# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

UIPresentationControllers and new presentation styles

New API that uses transition coordinators



# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

UIPresentationControllers and new presentation styles

New API that uses transition coordinators

Rotation callbacks are deprecated

# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

UIPresentationControllers and new presentation styles

New API that uses transition coordinators

Rotation callbacks are deprecated

- Use `viewWillTransitionToSize:withTransitionCoordinator:` instead

# What We Discussed Today

Trait collections and how view controllers may override them

Many new UISplitViewController features

Simple properties that condense and hide bars

UIPresentationControllers and new presentation styles

New API that uses transition coordinators

Rotation callbacks are deprecated

- Use `viewWillTransitionToSize:withTransitionCoordinator:` instead

Screen bounds is interface orientated

# Related Sessions

- 
- Building Adaptive Apps with UIKit Mission Wednesday 10:15 AM

---

  - A Look Inside Presentation Controllers Mission Thursday 11:30 AM

---

  - Building Interruptible and Responsive Interactions Presidio Friday 11:30 AM

---

  - Creating Extensions for iOS and OS X, Part 1 Mission Tuesday 2:00 PM

---



# More Information

Jake Behrens

App Frameworks Evangelist

[behrens@apple.com](mailto:behrens@apple.com)

Documentation and Sample Code

iOS Dev Center

<http://developer.apple.com>

Apple Developer Forums

<http://devforums.apple.com>

 WWDC14