Getting Started with UIKit Dynamics

Session 206

Olivier Gutknecht
iOS Applications & Frameworks Engineer

Agenda

What we will cover

Agenda

What we will cover

- Core concepts

Agenda

What we will cover

- Core concepts
- Predefined behaviors

Agenda

What we will cover

- Core concepts
- Predefined behaviors
- Best practices

Animations and Interactions on iOS

Animations and Interactions on iOS

- Core Animation

Animations and Interactions on iOS

- Core Animation
- UIView animations

Animations and Interactions on iOS

- Core Animation
- UIView animations
- Motion effects

Animations and Interactions on iOS

- Core Animation
- UIView animations
- Motion effects
- Gesture driven interactions

Animations and Interactions on iOS

- Core Animation
- UIView animations
- Motion effects
- Gesture driven interactions
- CADisplayLink

Animations and Interactions on iOS

- Core Animation
- UIView animations
- Motion effects
- Gesture driven interactions
- CADisplayLink
- All of above

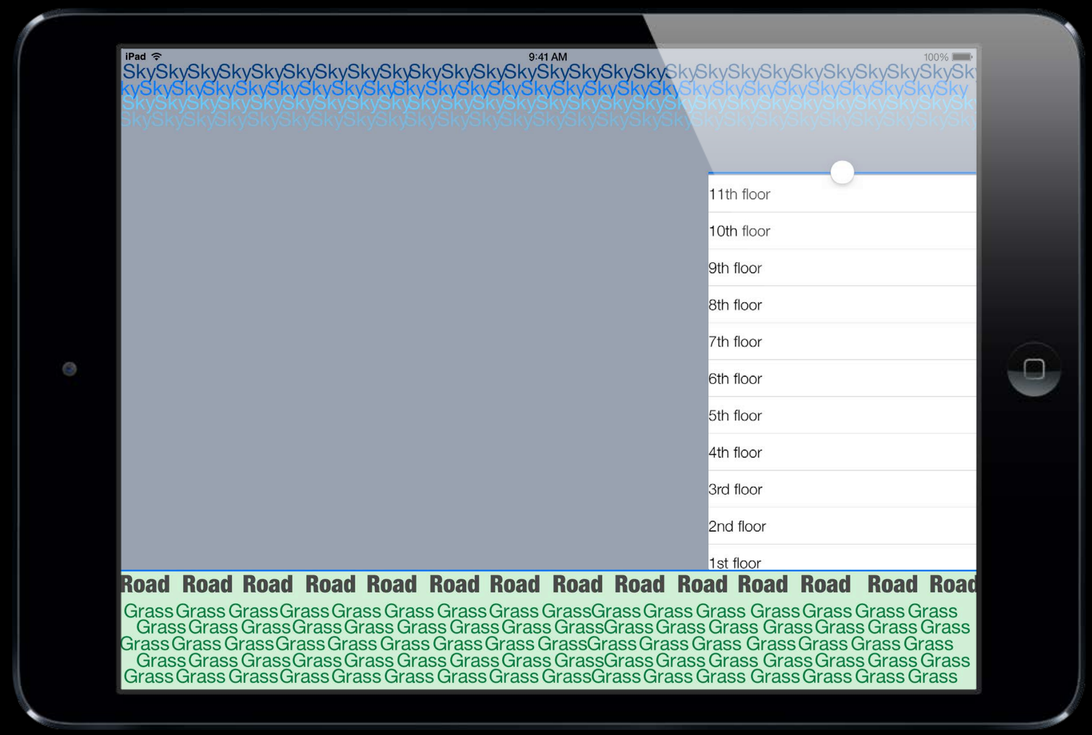
What Is UIKit Dynamics?

What Is UIKit Dynamics?

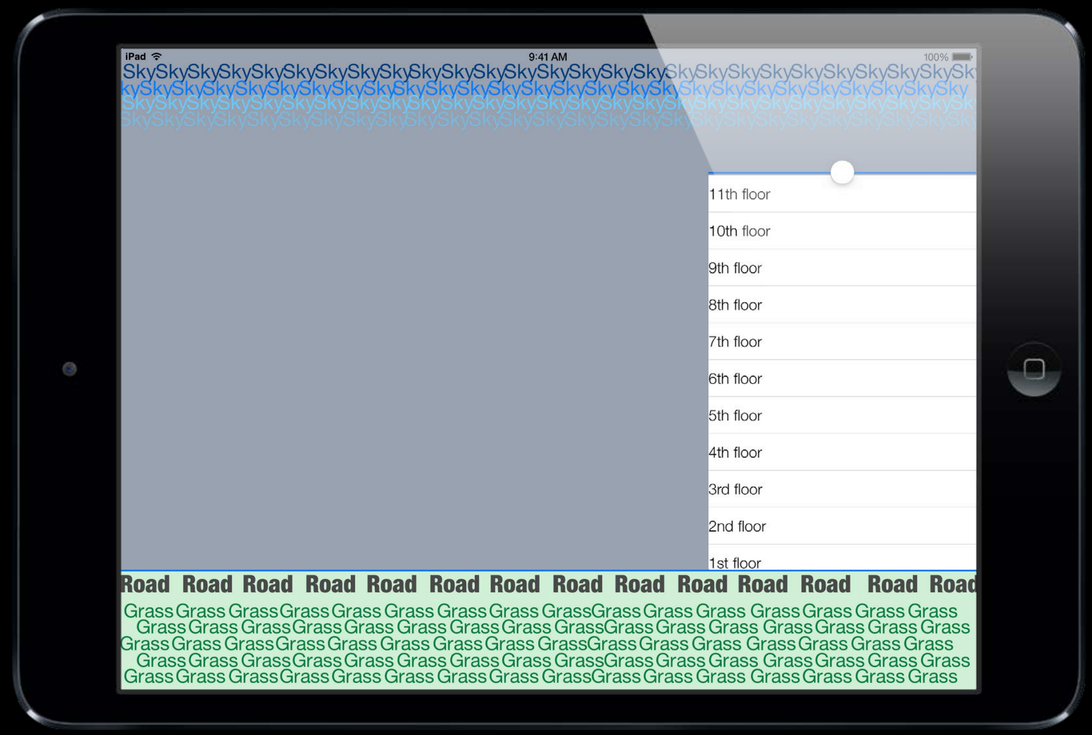
A composable, reusable, declarative,
real-world inspired animation, and interaction system

UIKit Games

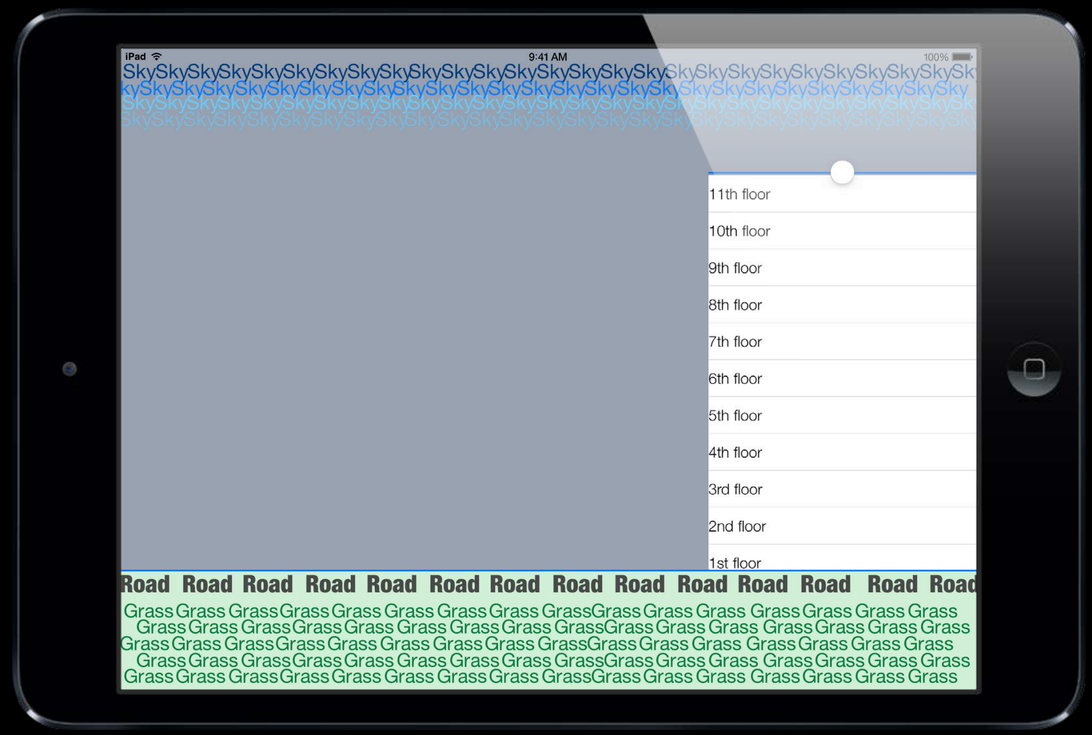
UIKit Games


iPad
9:41 AM
100%
11th floor
10th floor
9th floor
8th floor
7th floor
6th floor
5th floor
4th floor
3rd floor
2nd floor
1st floor
Road
Grass

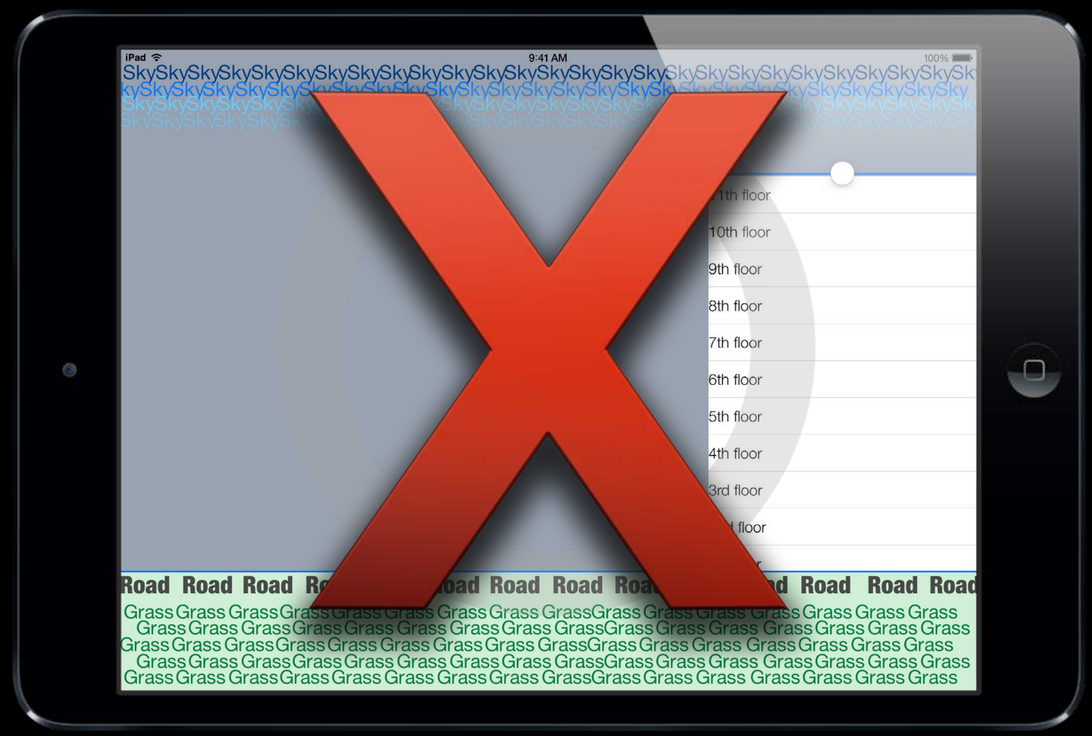
UIKit Games


iPad
9:41 AM
100%
11th floor
10th floor
9th floor
8th floor
7th floor
6th floor
5th floor
4th floor
3rd floor
2nd floor
1st floor
Road
Grass

UIKit Games


iPad
9:41 AM
100%
11th floor
10th floor
9th floor
8th floor
7th floor
6th floor
5th floor
4th floor
3rd floor
2nd floor
1st floor
Road
Grass

UIKit Games


iPad
9:41 AM
100%
11th floor
10th floor
9th floor
8th floor
7th floor
6th floor
5th floor
4th floor
3rd floor
Road
Grass


NEW

Sprite Kit

Introduction to Sprite Kit

Presidio
Wednesday 11:30AM

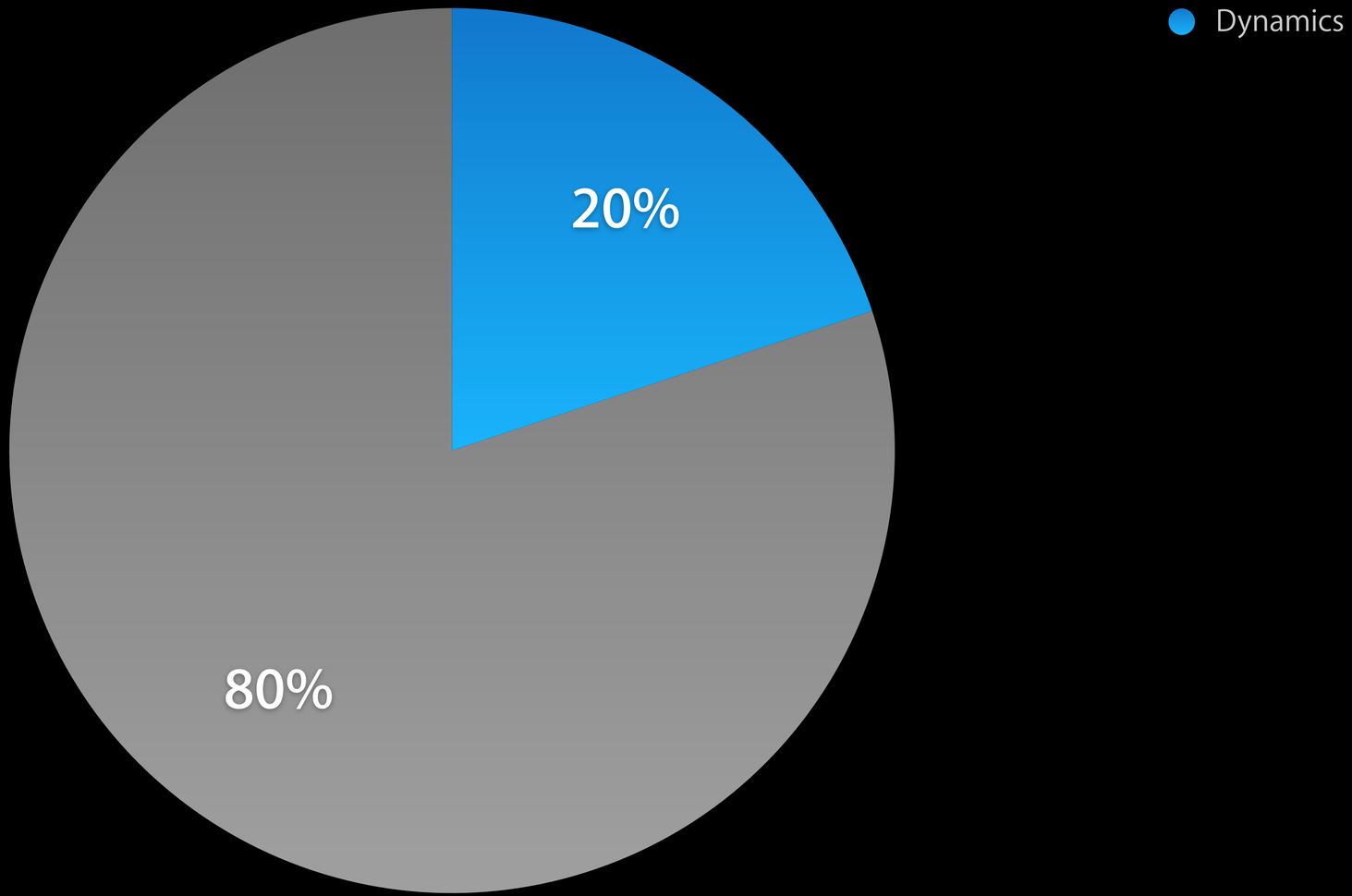
Demo

How Complex Is This?

Entire application: 400 lines of code

How Complex Is This?

Entire application: 400 lines of code



Overview

Why?

Why?

- Real world inspired interactions

Why?

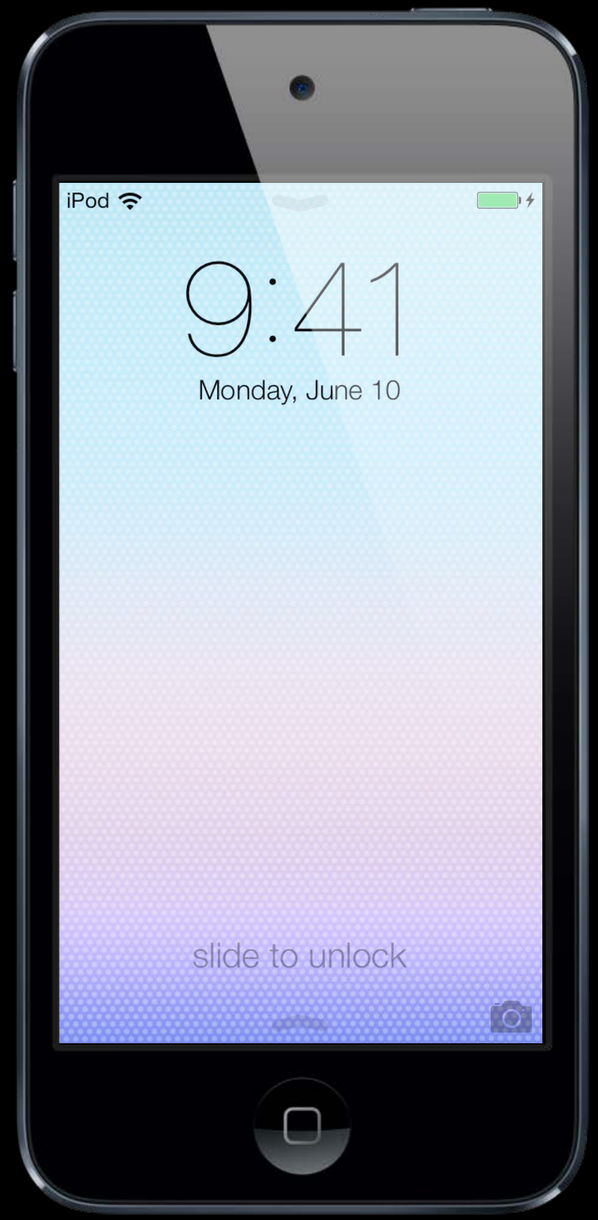
- Real world inspired interactions
- Combining predefined
  and interactive animations

Why?

- Real world inspired interactions
- Combining predefined
  and interactive animations
- Designed for UI

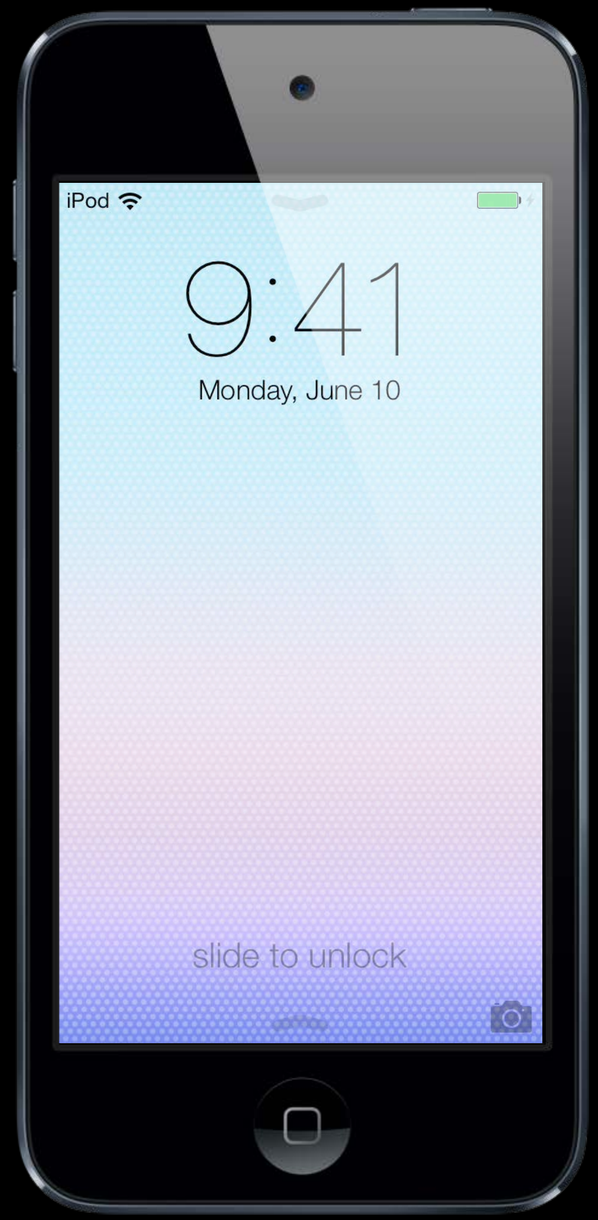
Why?

- Real world inspired interactions
- Combining predefined and interactive animations
- Designed for UI



Why?

- Real world inspired interactions
- Combining predefined and interactive animations
- Designed for UI



How?

How?

- High-level expression

```
[myView setMass:0.42] ?
```

How?

- High-level expression

~~`[myView setMass:0.42] ?`~~

How?

- High-level expression
- Composition of…

How?

- High-level expression
- Composition of…
  - Primitive behaviors

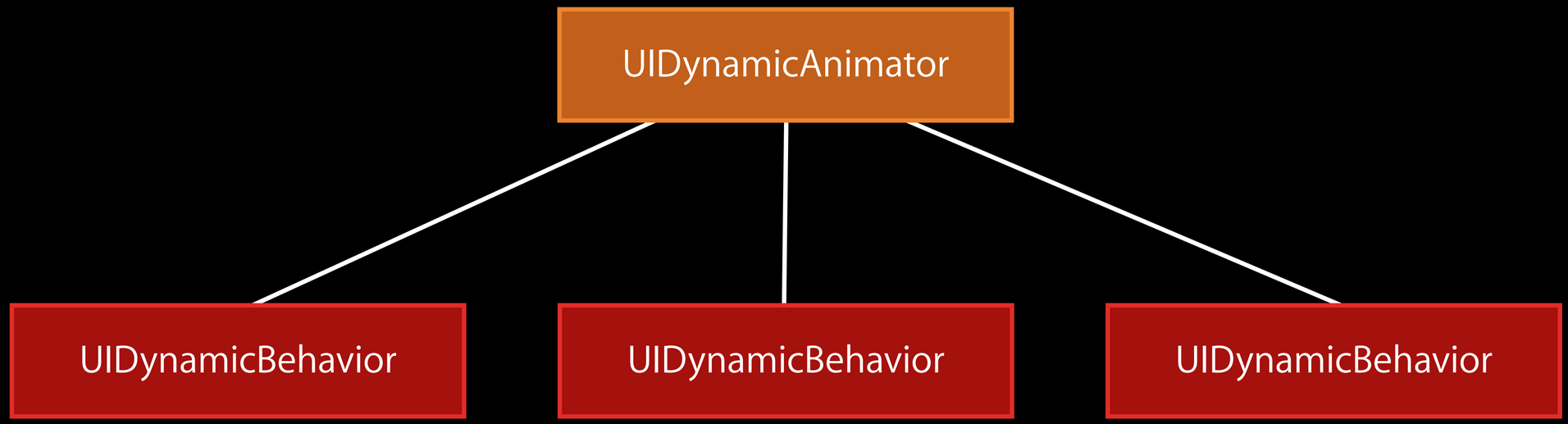
How?

- High-level expression
- Composition of…
  - Primitive behaviors
- Animation context

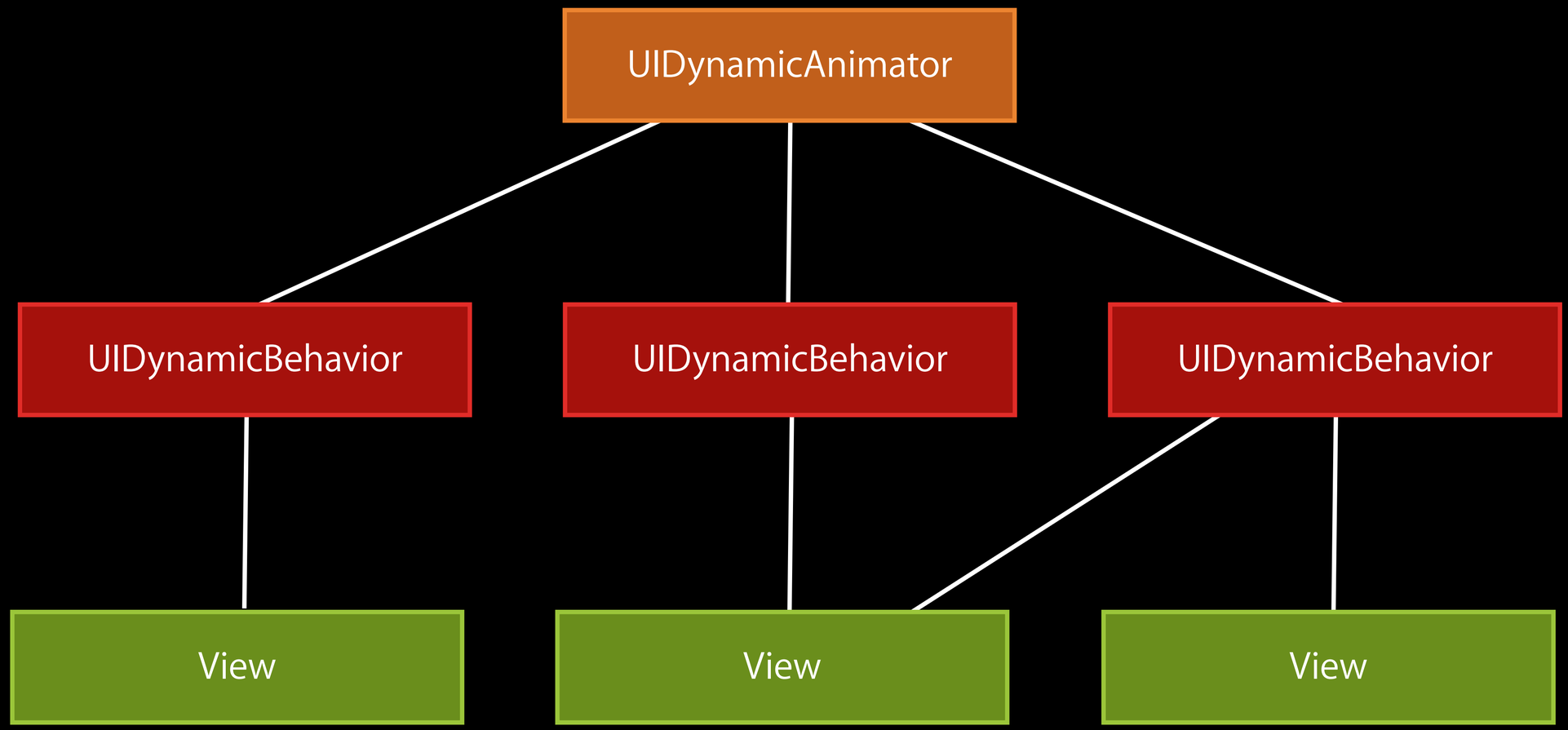
Architecture

UIDynamicAnimator

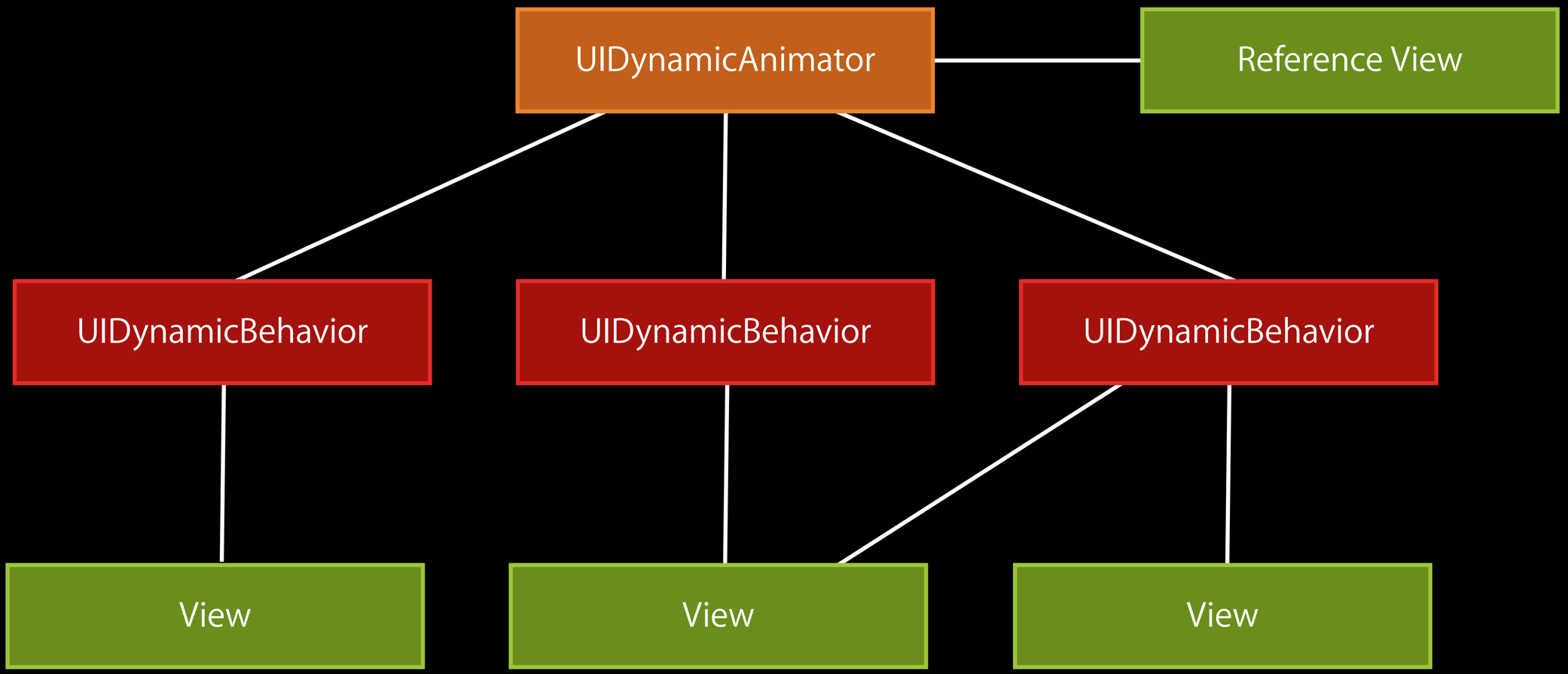
Architecture



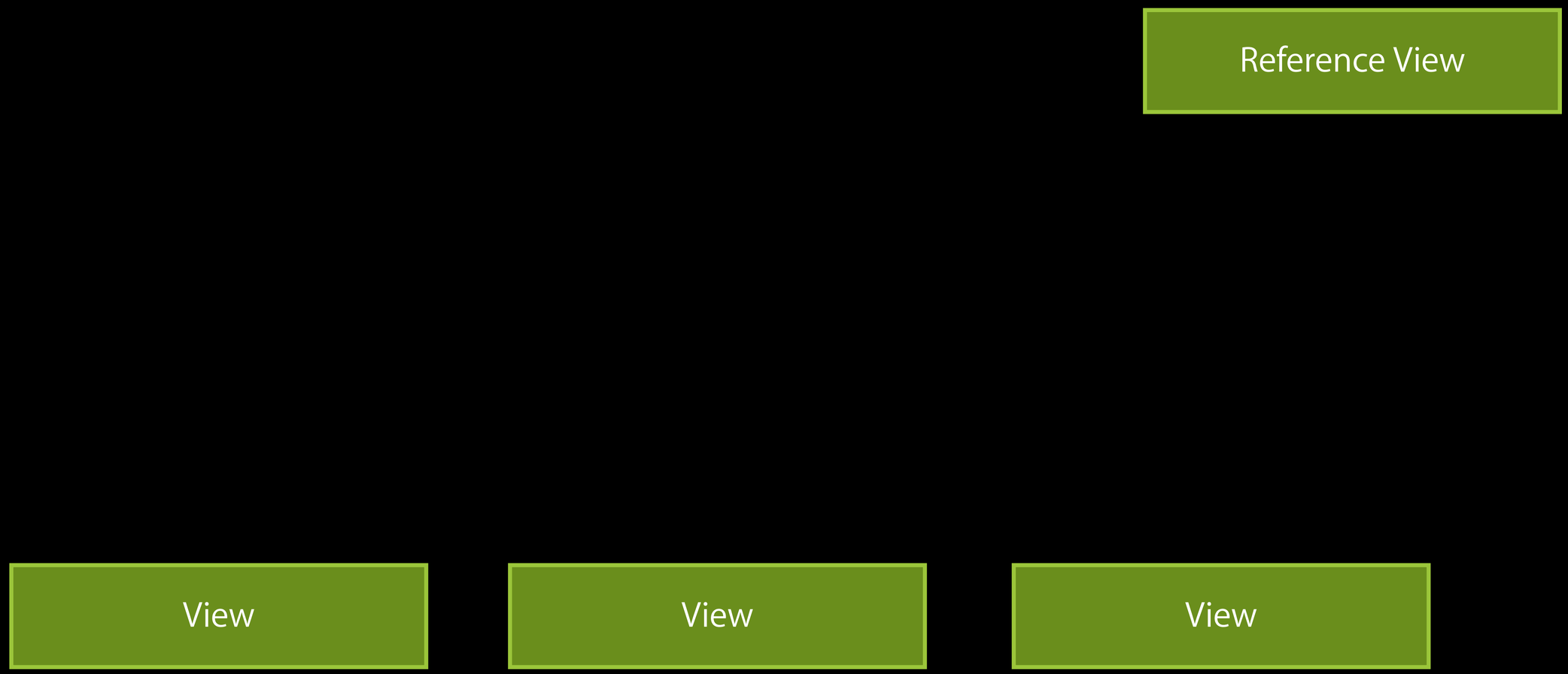
Architecture



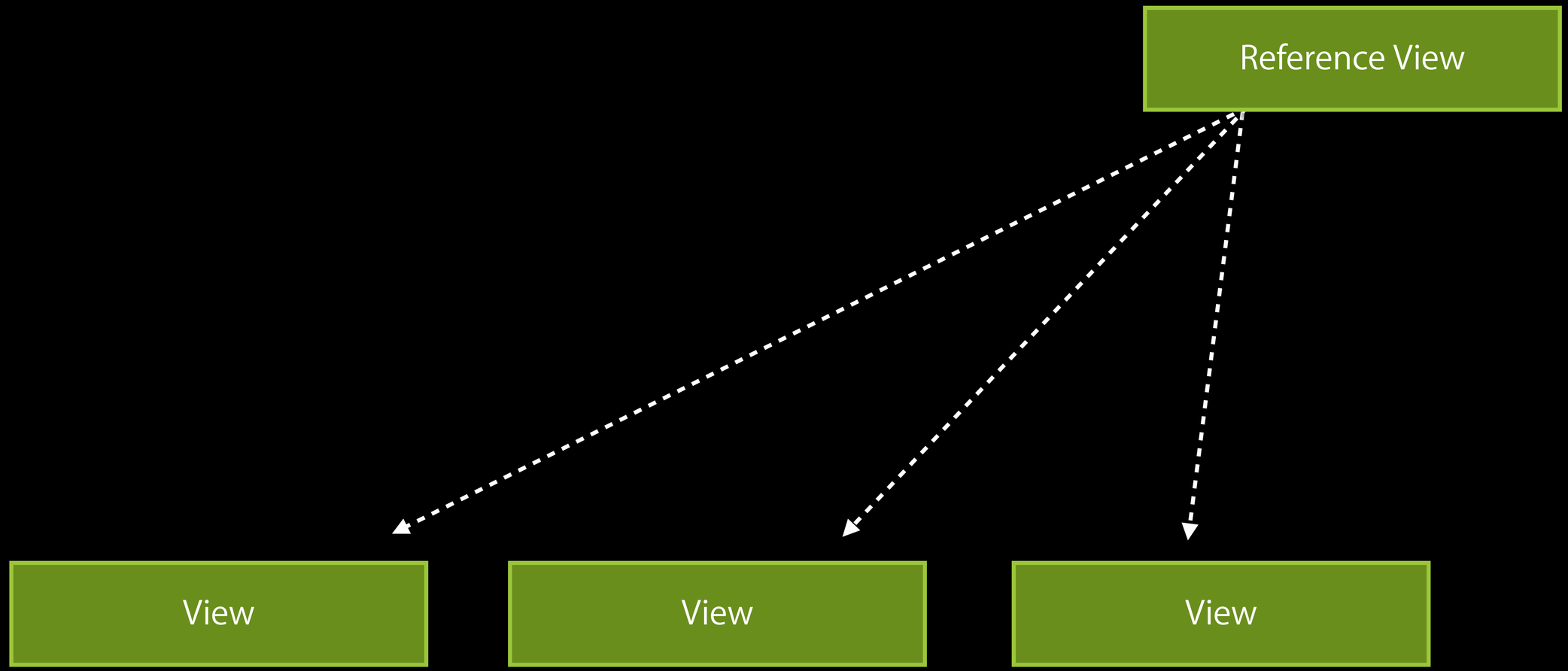
Architecture



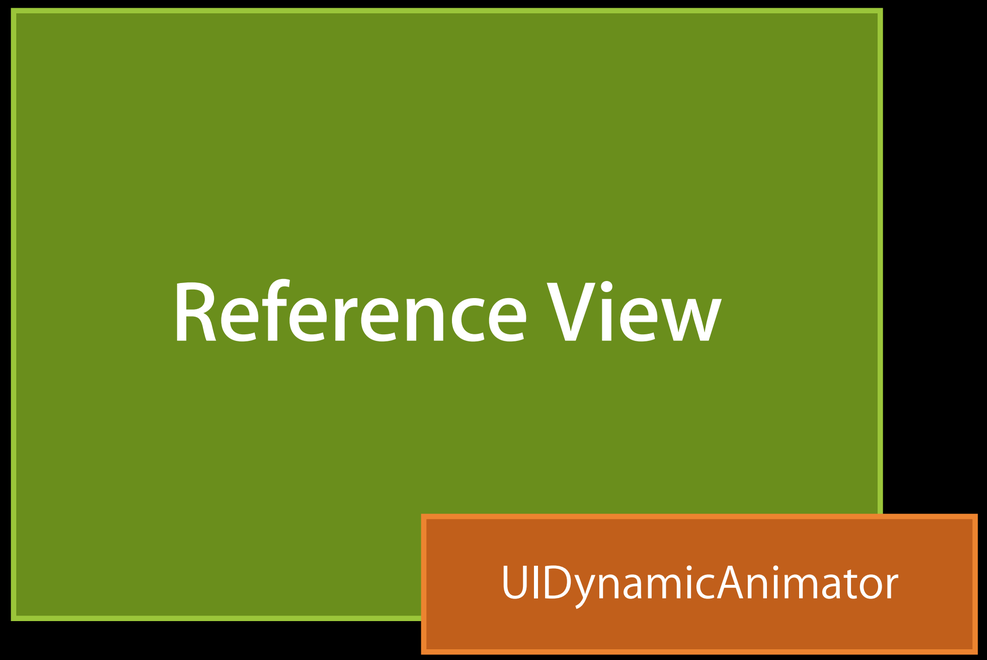
Architecture



Architecture

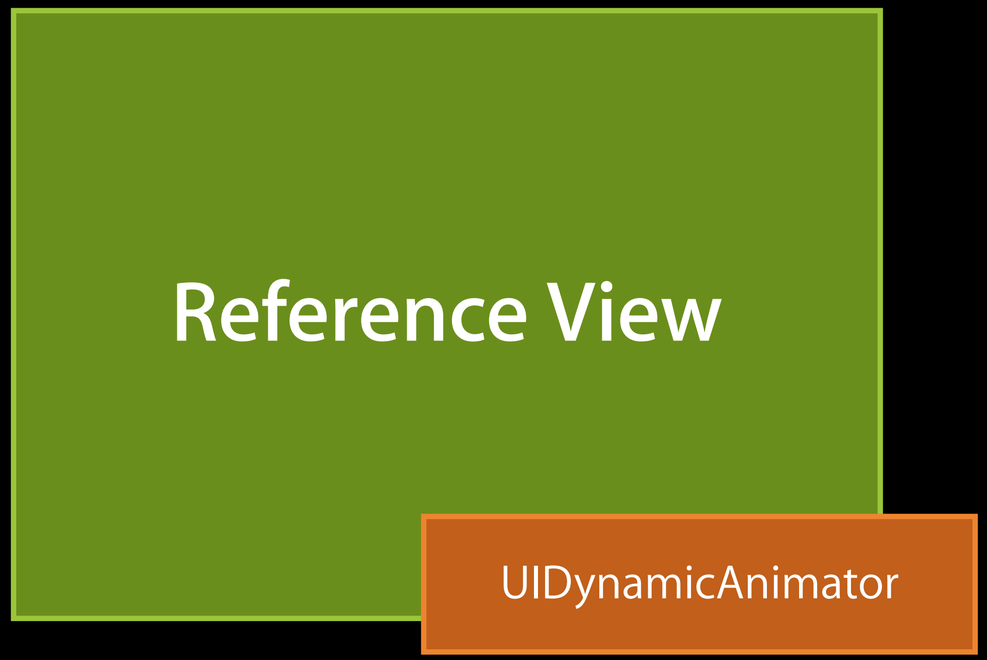


UIDynamicAnimator



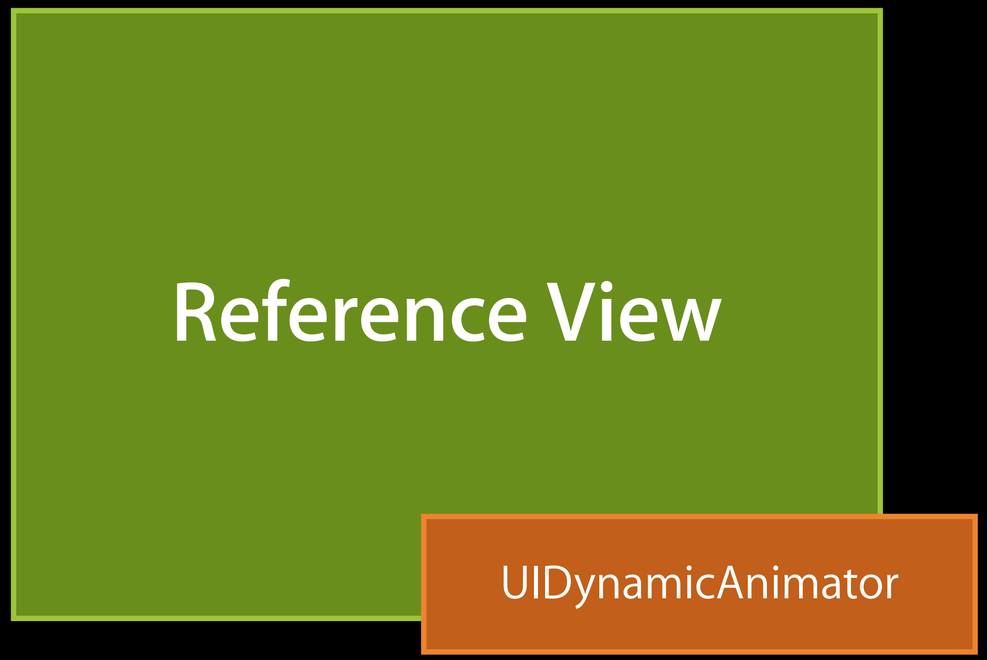
UIDynamicAnimator

- Provide the overall context



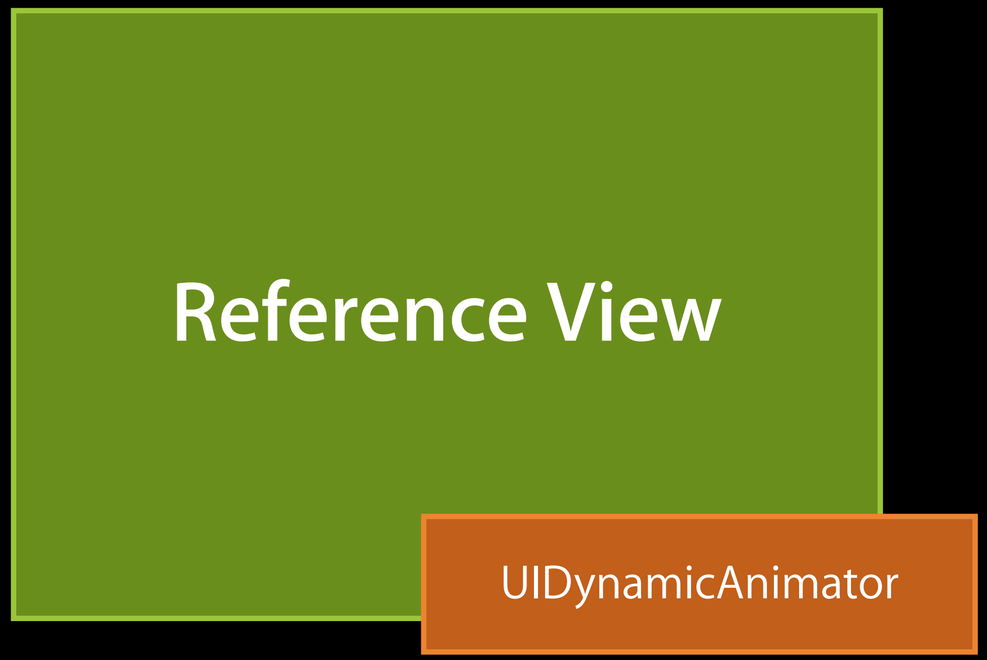
UIDynamicAnimator

- Provide the overall context
- Define the coordinate system



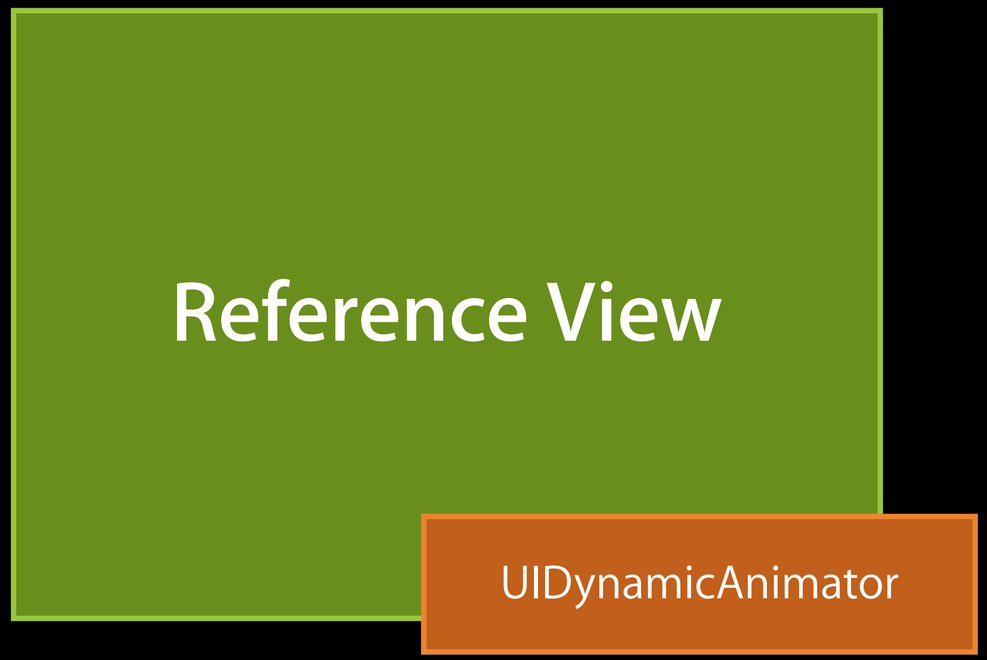
UIDynamicAnimator

- Provide the overall context
- Define the coordinate system
- Control the engine



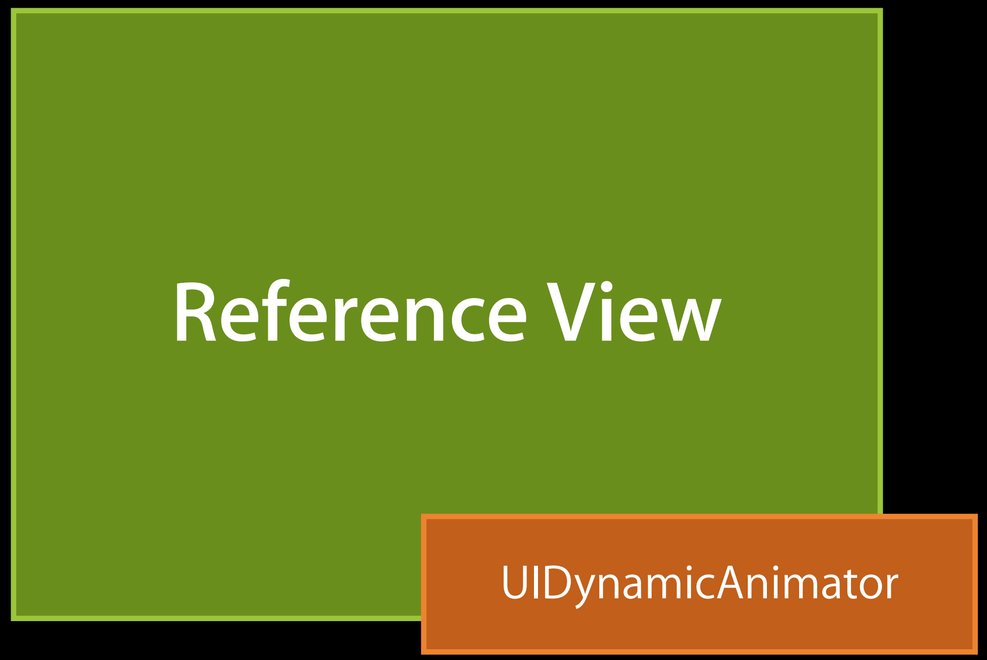
UIDynamicAnimator

- Provide the overall context
- Define the coordinate system
- Control the engine
- Keep track of behaviors



UIDynamicAnimator

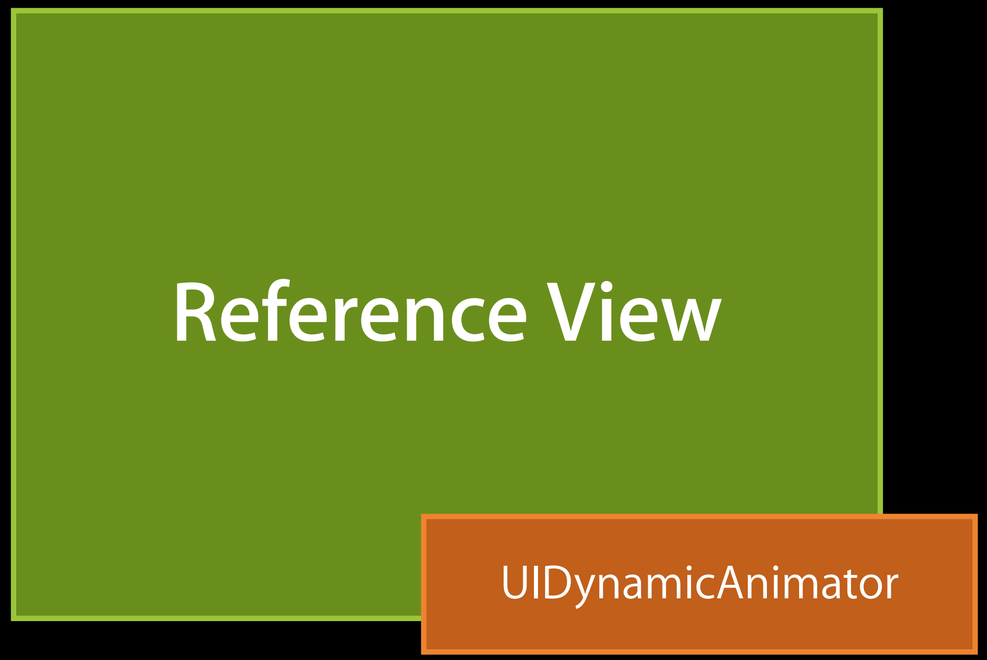
- Provide the overall context
- Define the coordinate system
- Control the engine
- Keep track of behaviors



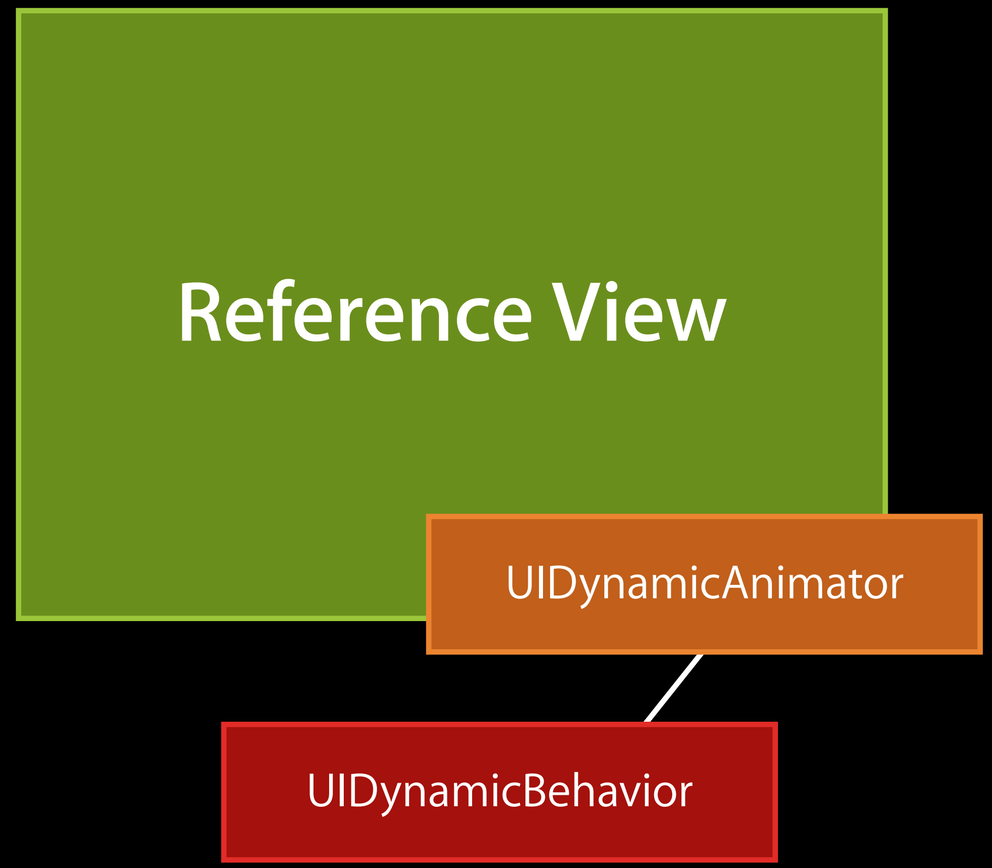

```
animator = [[UIDynamicAnimator alloc] initWithReferenceView:referenceView];

[animator addBehavior:...];
[animator addBehavior:...];
```

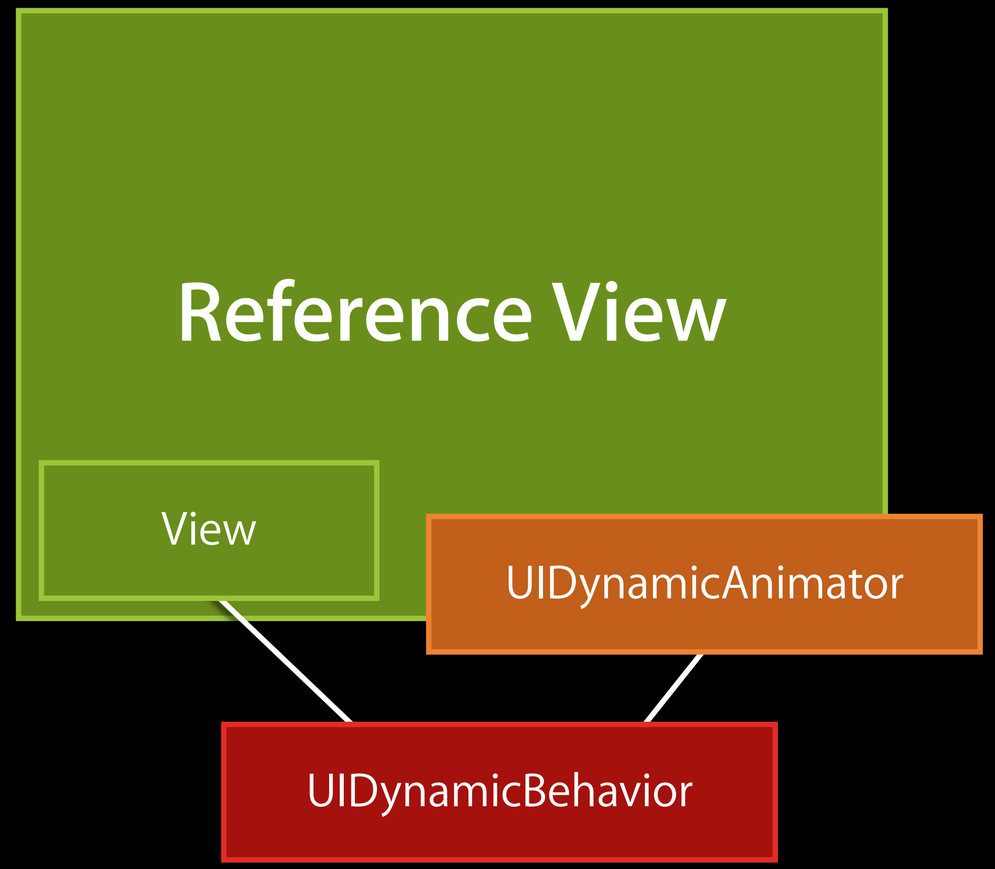
UIDynamicBehavior



UIDynamicBehavior

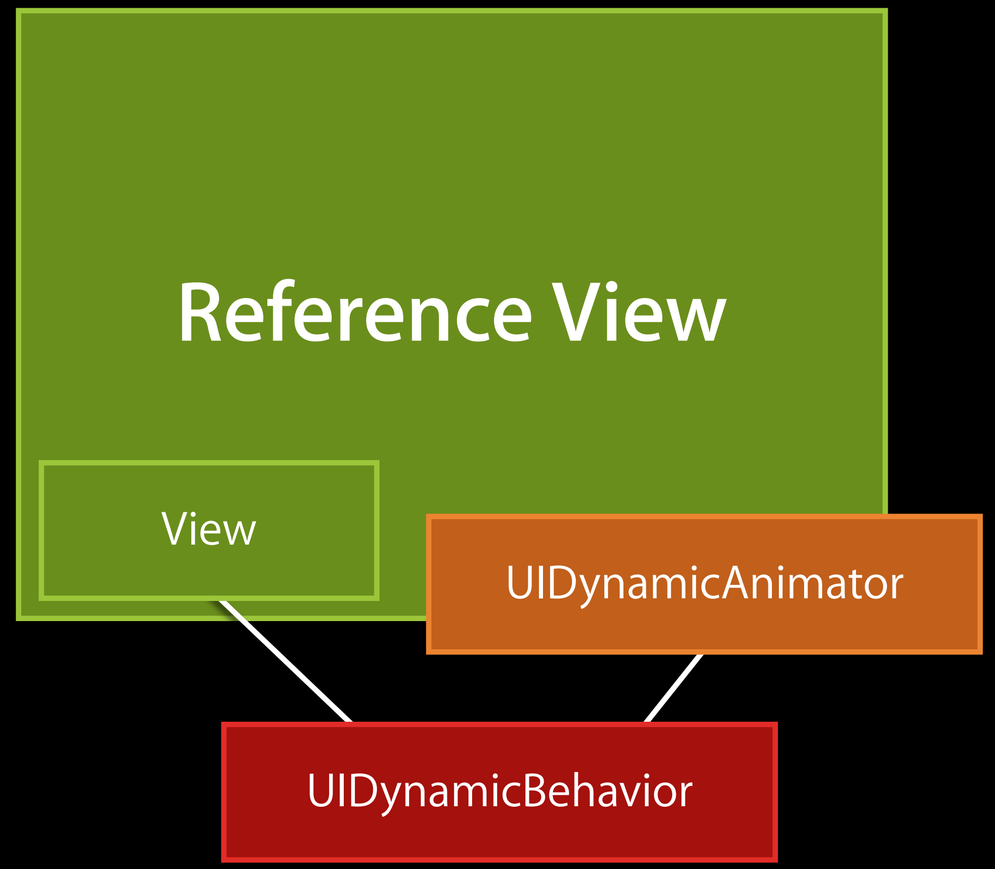


UIDynamicBehavior



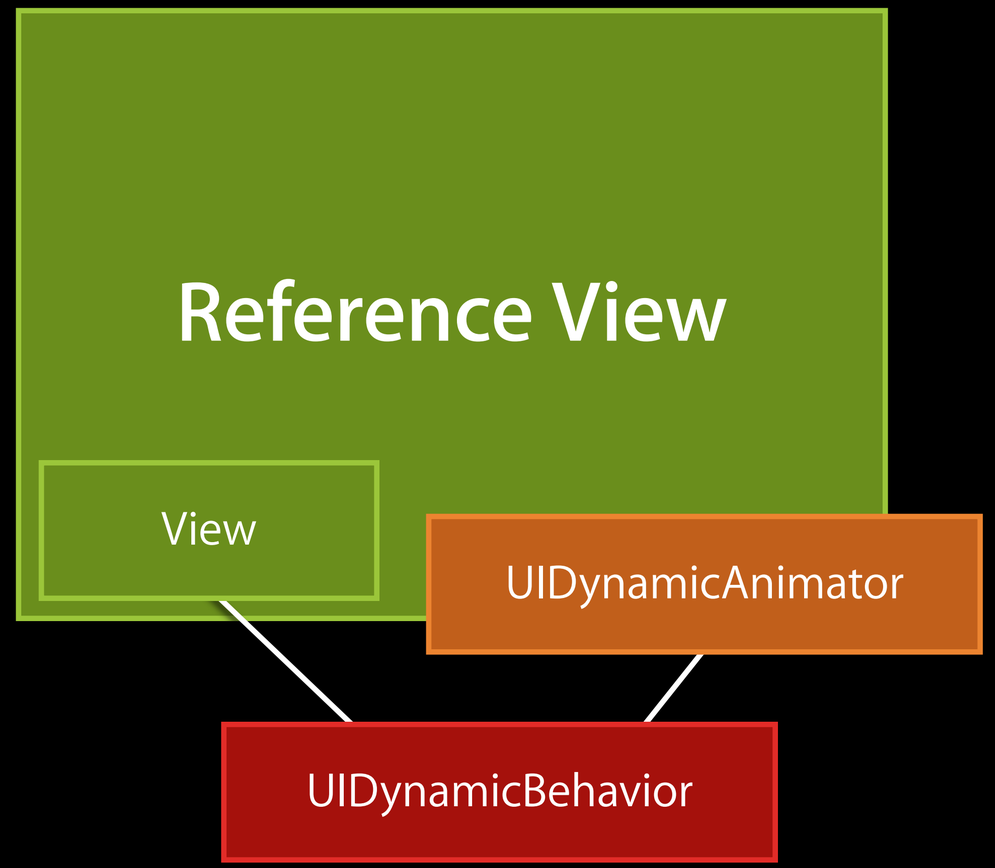
UIDynamicBehavior

- Declarative



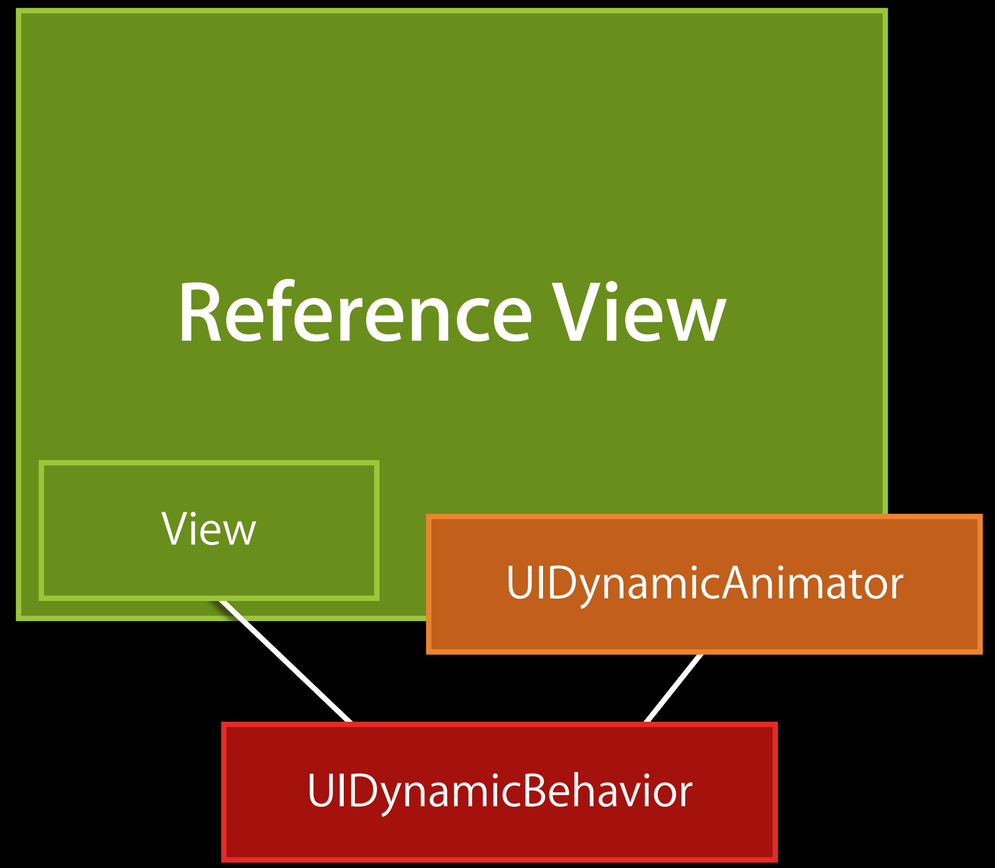
UIDynamicBehavior

- Declarative
- Describe “influences” on views



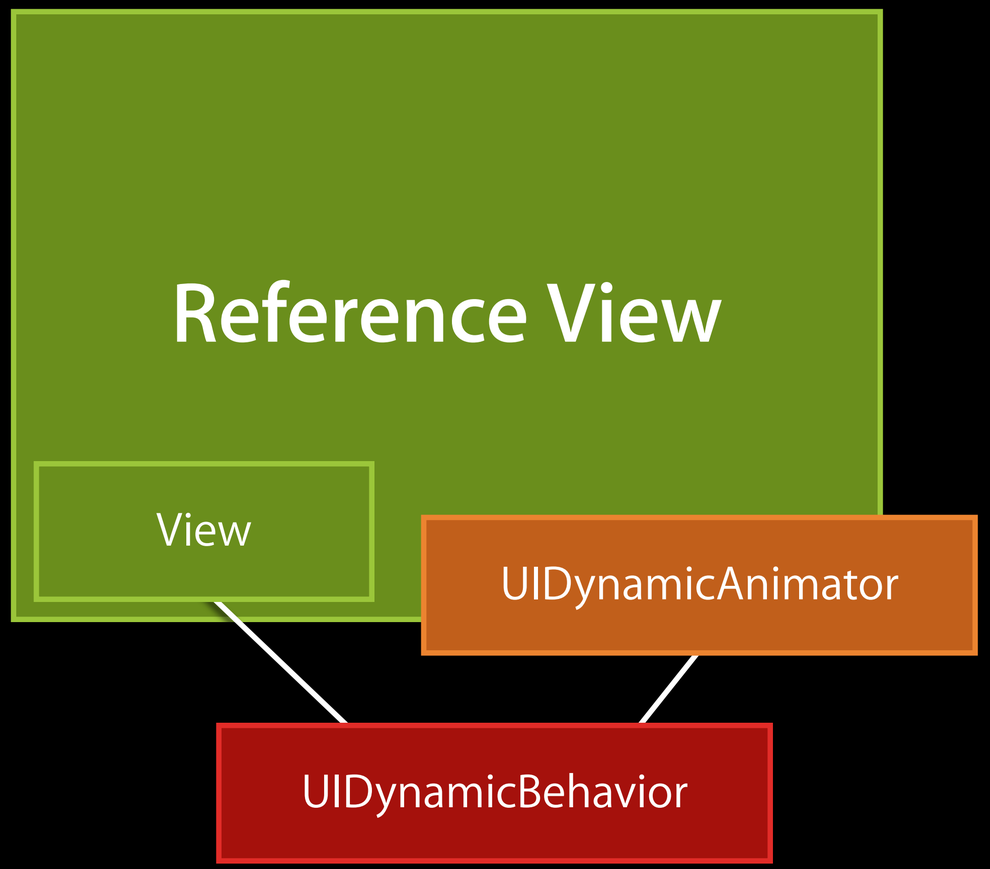
UIDynamicBehavior

- Declarative
- Describe “influences” on views
- Added and removed at any time



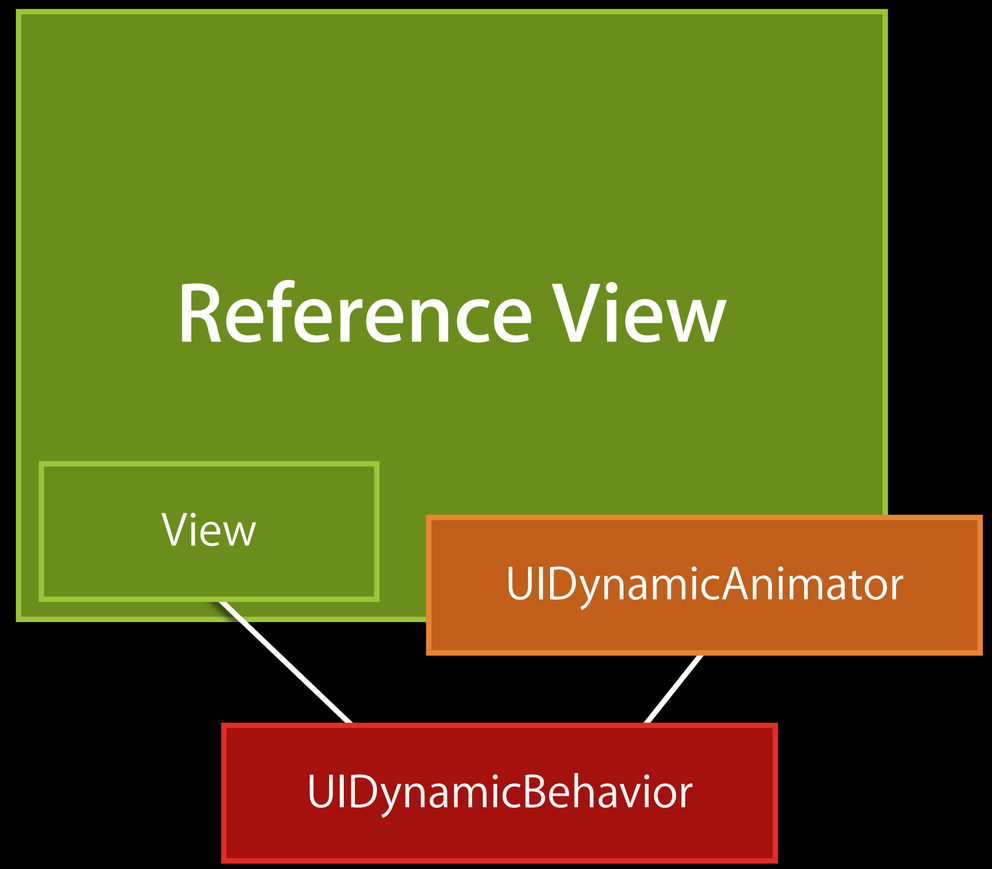
UIDynamicBehavior

- Declarative
- Describe “influences” on views
- Added and removed at any time
- Composable



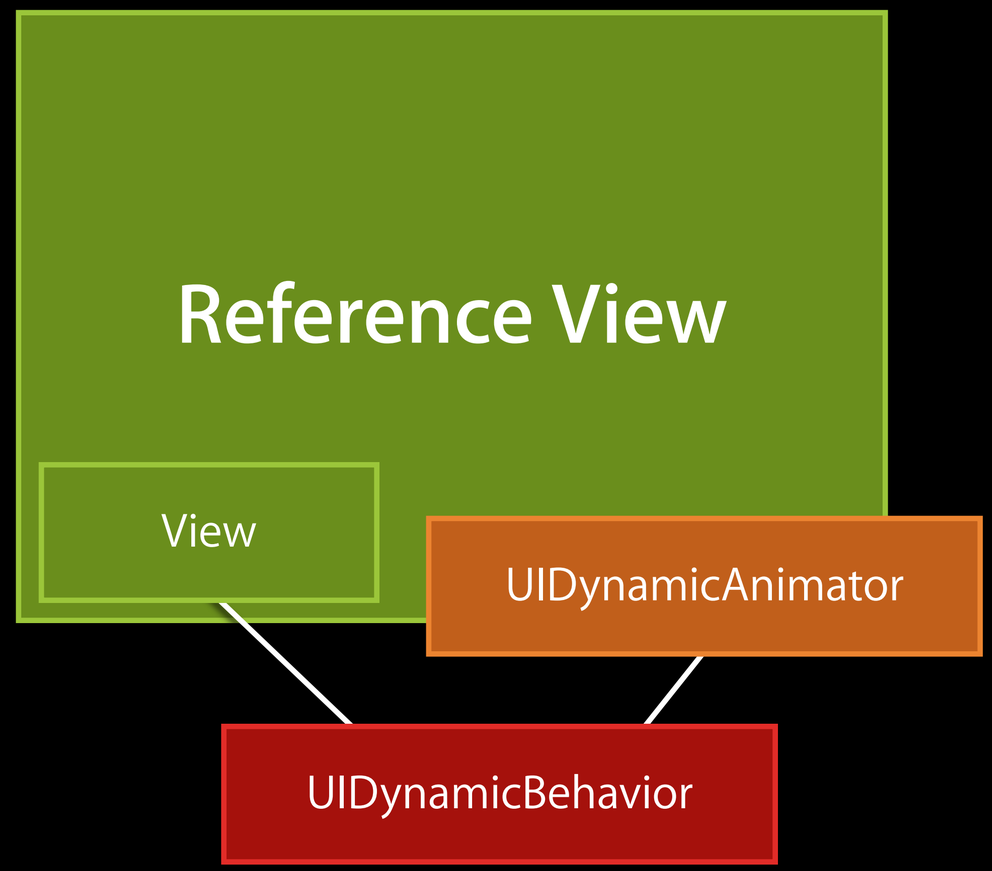
UIDynamicBehavior

- Declarative
- Describe "influences" on views
- Added and removed at any time
- Composable
- Subclassable



UIDynamicBehavior

- Declarative
- Describe "influences" on views
- Added and removed at any time
- Composable
- Subclassable




```
myBehavior = [[MyBehavior alloc] initWith...];

[animator addBehavior:myBehavior];
```

Primitive Behaviors

Common Traits

- Configured with items to animate

Common Traits

- Configured with items to animate
- Most primitive behaviors support adding and removing items

Common Traits

- Configured with items to animate
- Most primitive behaviors support adding and removing items
- Can be parametrized before adding to an animator

Common Traits

- Configured with items to animate
- Most primitive behaviors support adding and removing items
- Can be parametrized before adding to an animator
- The influence stops when the behavior is removed

Predefined Behaviors

A rich set of composable classes

Predefined Behaviors

A rich set of composable classes

- Gravity

Predefined Behaviors

A rich set of composable classes

- Gravity
- Collision

Predefined Behaviors

A rich set of composable classes

- Gravity
- Collision
- Attachments

Predefined Behaviors

A rich set of composable classes

- Gravity
- Collision
- Attachments
- Snap

Predefined Behaviors

A rich set of composable classes

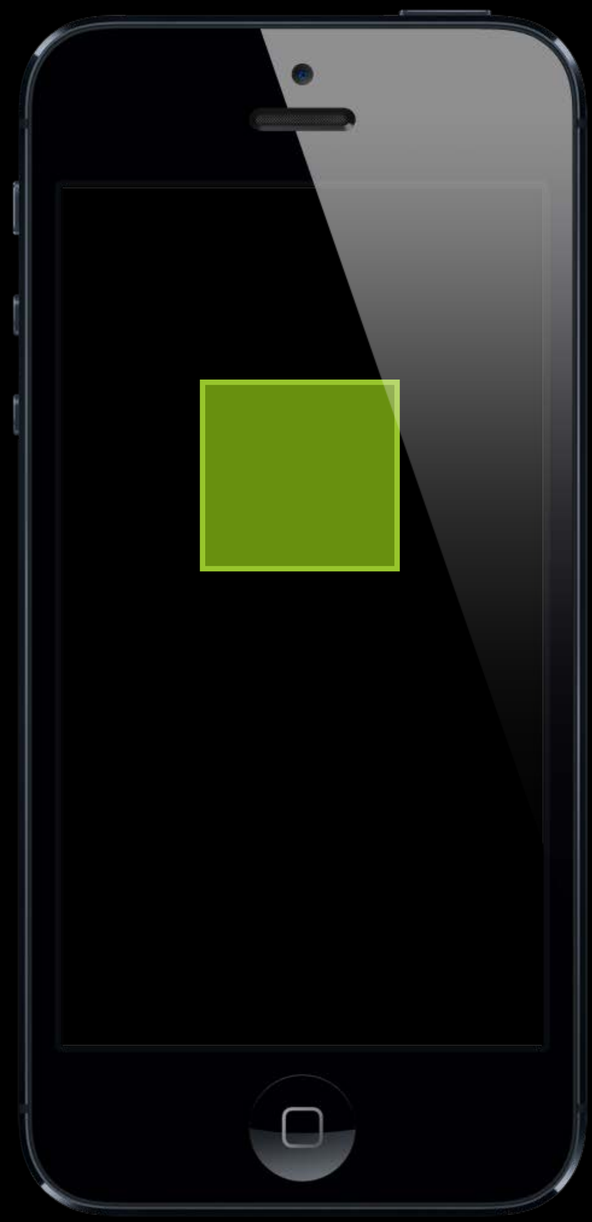
- Gravity
- Collision
- Attachments
- Snap
- Forces

Predefined Behaviors

A rich set of composable classes

- Gravity
- Collision
- Attachments
- Snap
- Forces
- Item properties

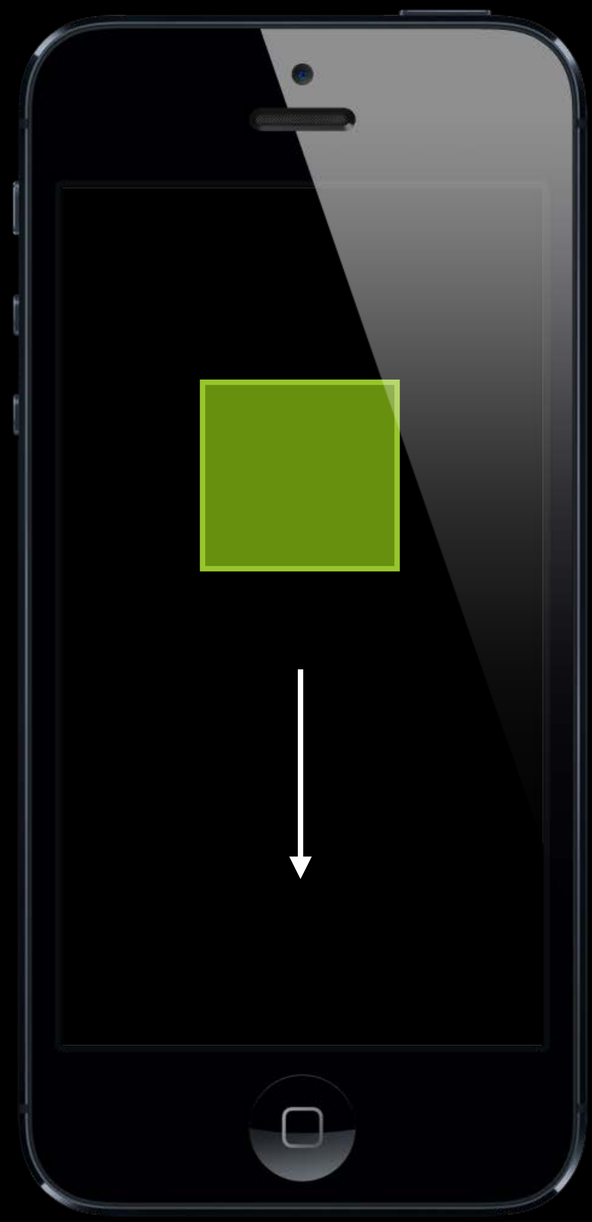
UIGravityBehavior



UIGravityBehavior

UIGravityBehavior

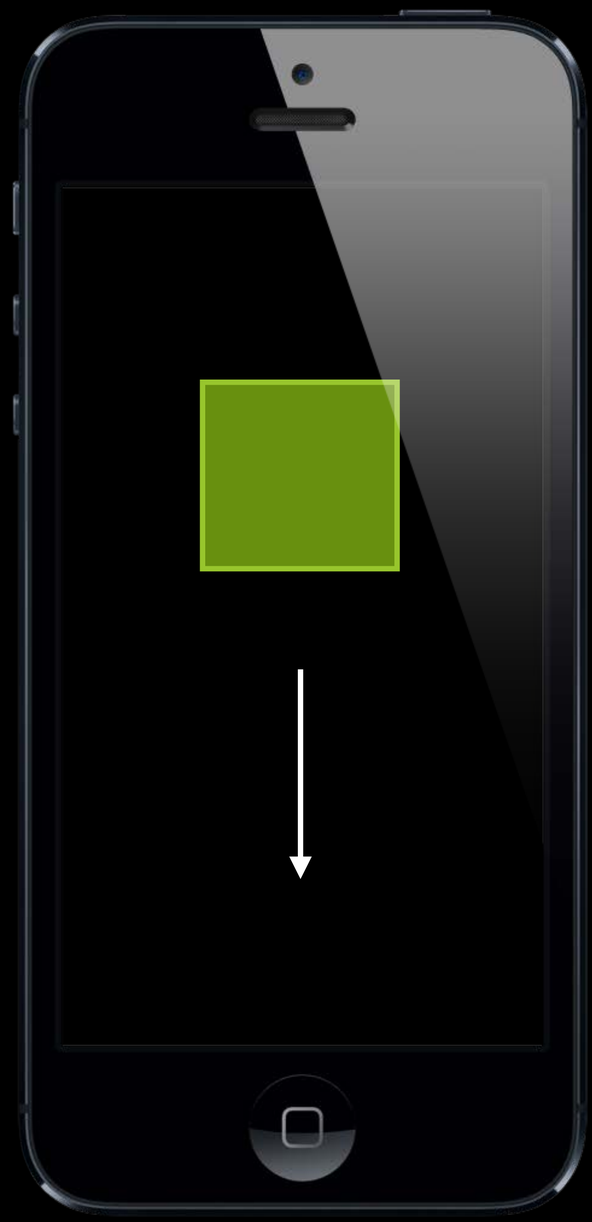
- A simple gravity vector



UIGravityBehavior

- A simple gravity vector

```
@property (readwrite,nonatomic)
CGFloat xComponent;
@property (readwrite,nonatomic)
CGFloat yComponent;
```

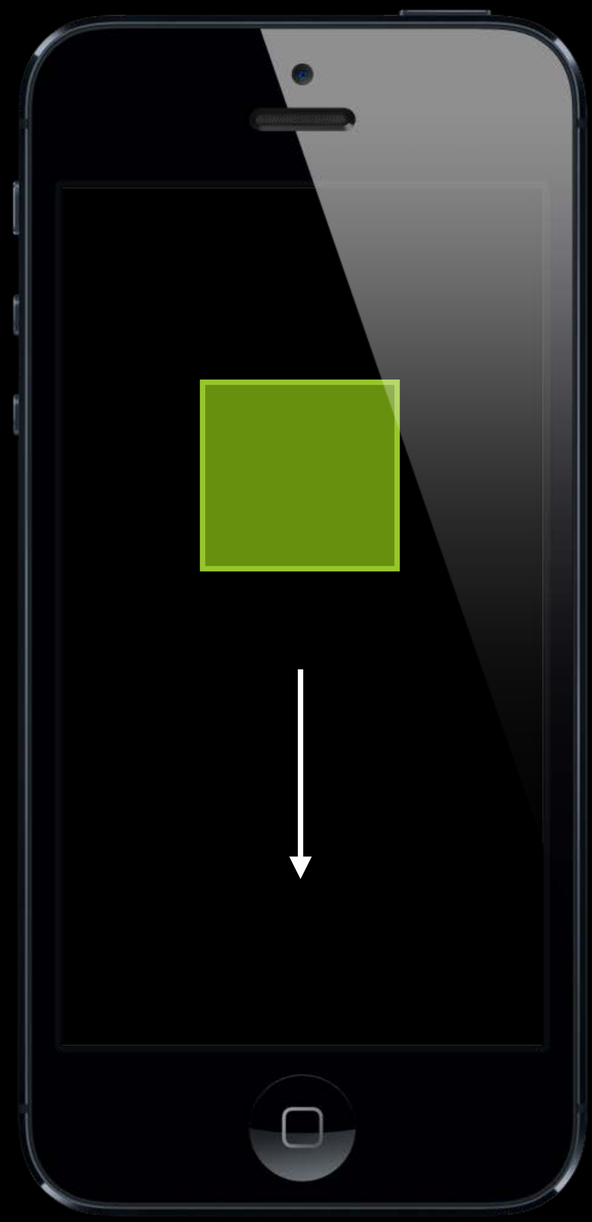


UIGravityBehavior

- A simple gravity vector

```
@property (readwrite,nonatomic)
CGFloat xComponent;
@property (readwrite,nonatomic)
CGFloat yComponent;
```

- UIKit coordinate system

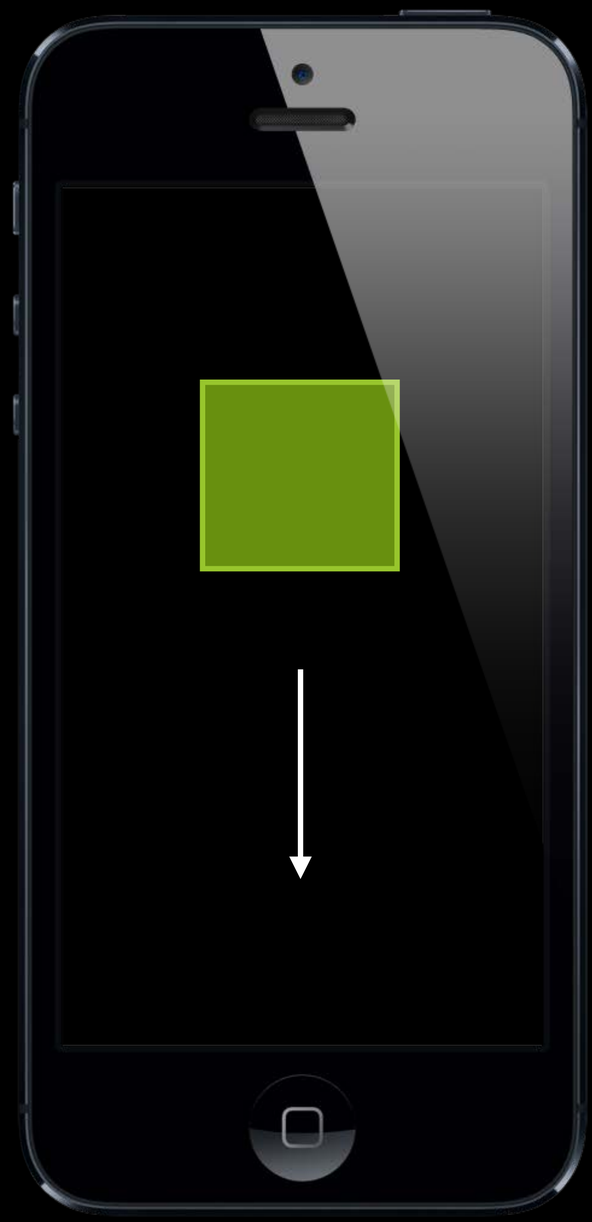


UIGravityBehavior

- A simple gravity vector

```
@property (readwrite,nonatomic)
CGFloat xComponent;
@property (readwrite,nonatomic)
CGFloat yComponent;
```

- UIKit coordinate system
  - (0,1) by default

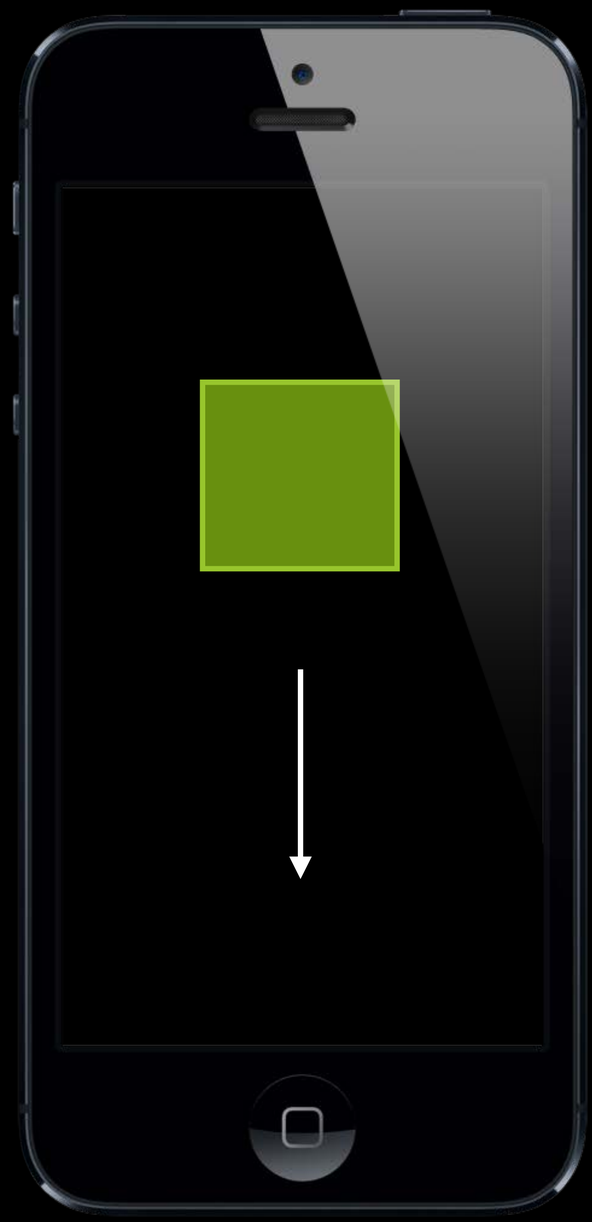


UIGravityBehavior

- A simple gravity vector

```
@property (readwrite,nonatomic)
CGFloat xComponent;
@property (readwrite,nonatomic)
CGFloat yComponent;
```

- UIKit coordinate system
  - (0,1) by default
- Items can be added and removed at any time



UIGravityBehavior

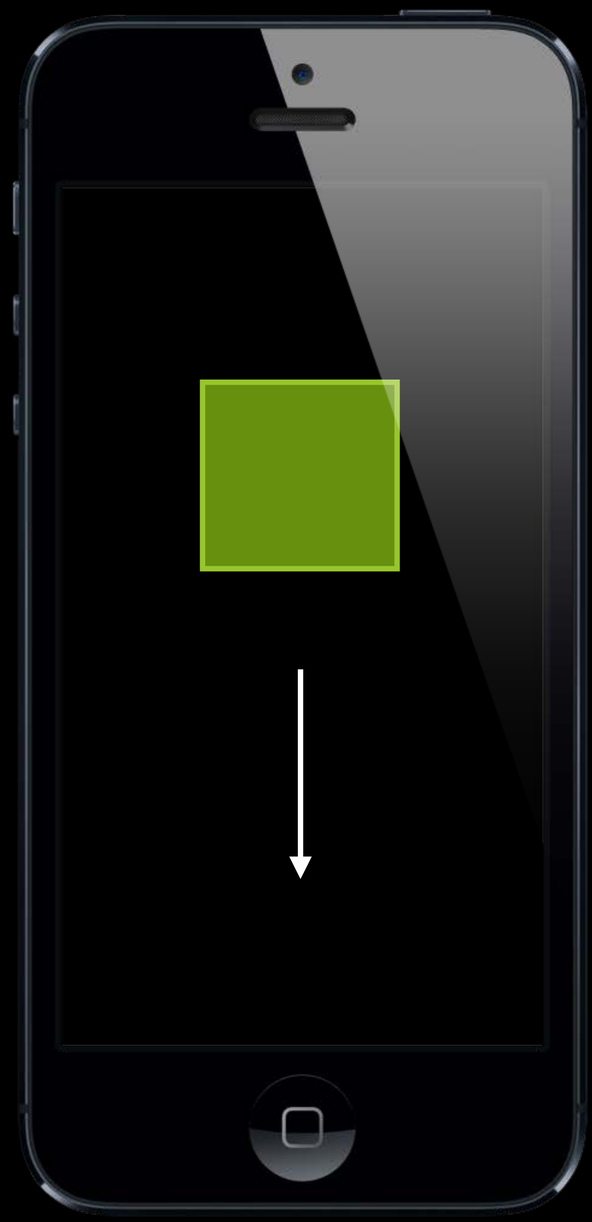
- A simple gravity vector

```
@property (readwrite,nonatomic)
CGFloat xComponent;
@property (readwrite,nonatomic)
CGFloat yComponent;
```

- UIKit coordinate system
  - (0,1) by default
- Items can be added and removed at any time

```
g = [[UIGravityBehavior alloc] initWithItems:@[v]];

[animator addBehavior:g];
```



A Well-known Constant…

Earth Gravity

A Well-known Constant…

Earth Gravity
$9.80665\ m/s^2$

Introducing…

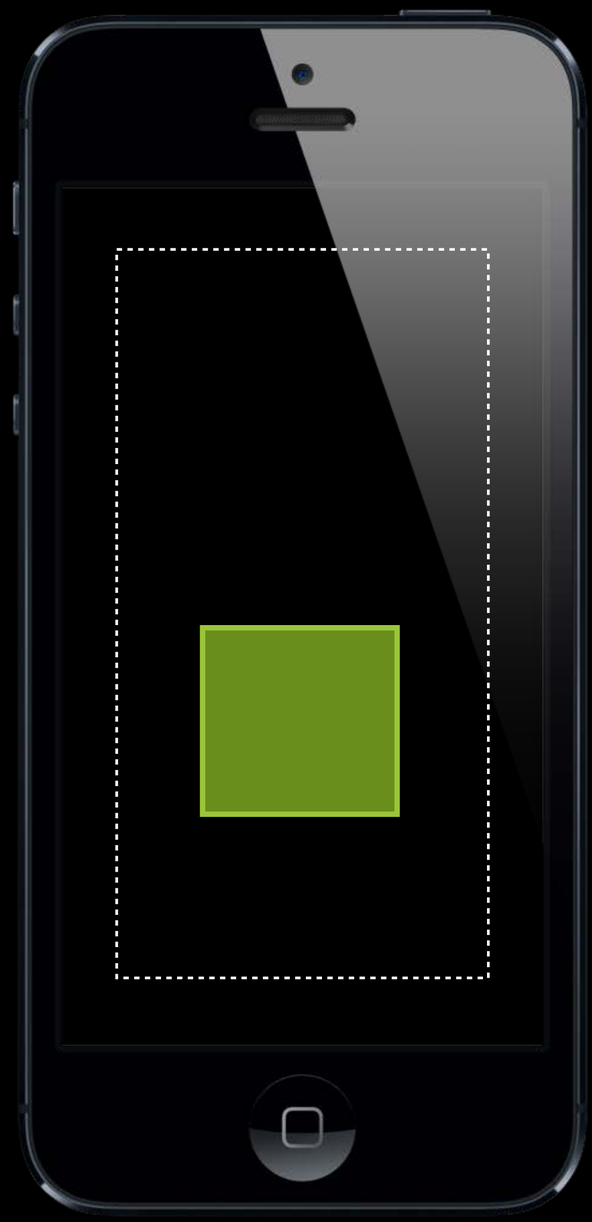
UIKit Gravity

Introducing…

UIKit Gravity

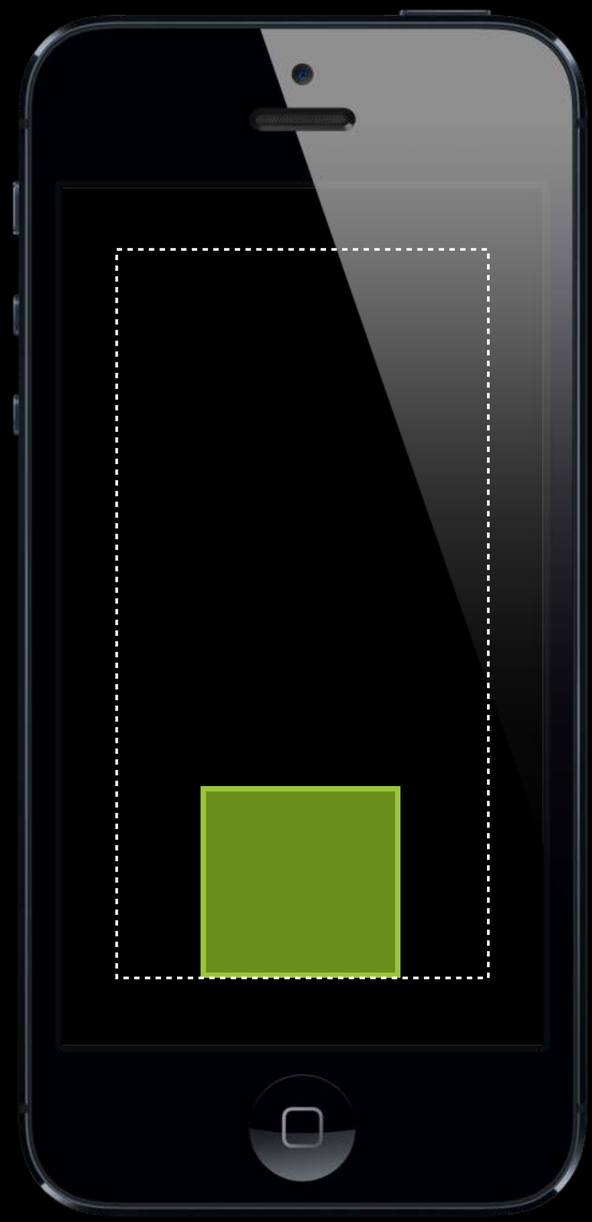
1000 p/s²

UICollisionBehavior



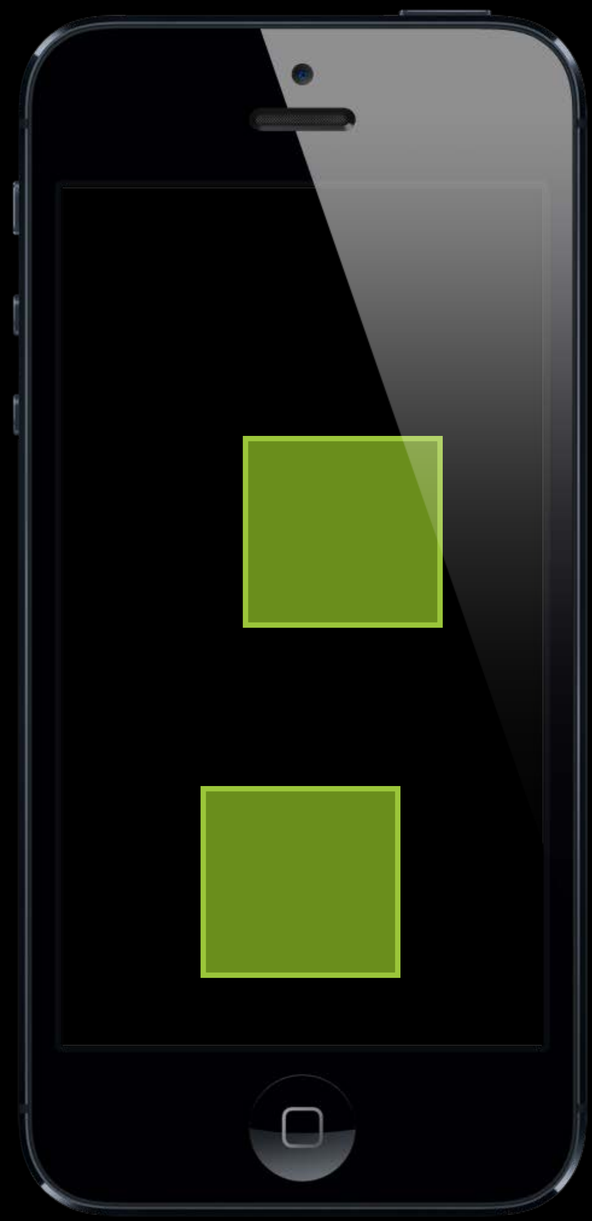
UICollisionBehavior

- Between a view and a boundary



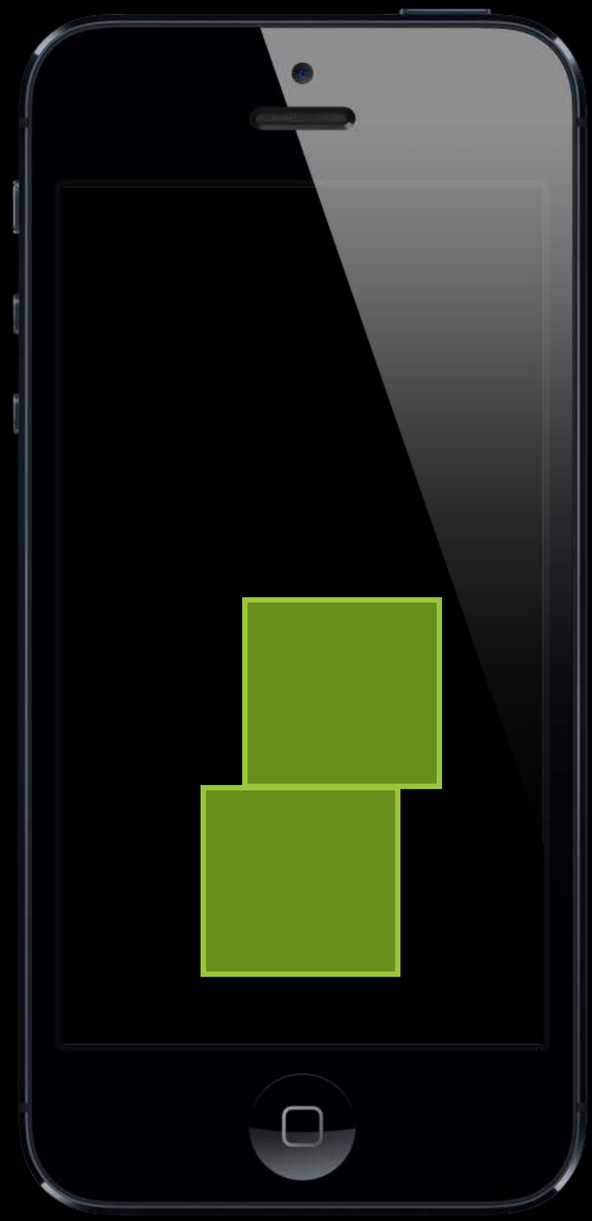
UICollisionBehavior

- Between a view and a boundary



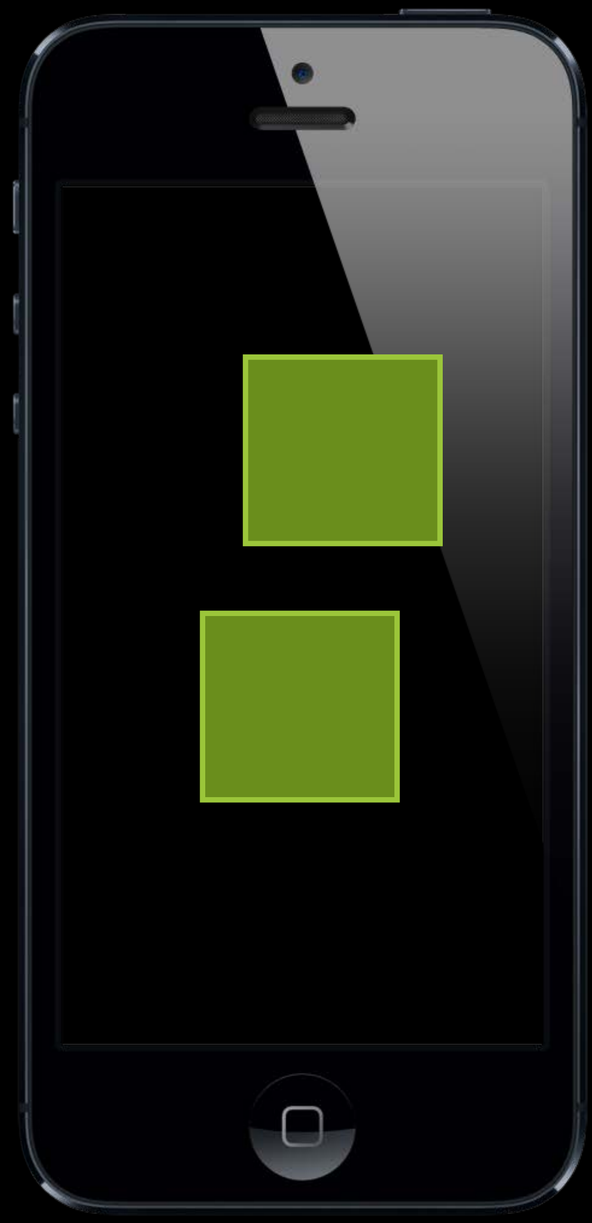
UICollisionBehavior

- Between a view and a boundary
- Or between views associated to the same behavior



UICollisionBehavior

- Between a view and a boundary
- Or between views associated to the same behavior

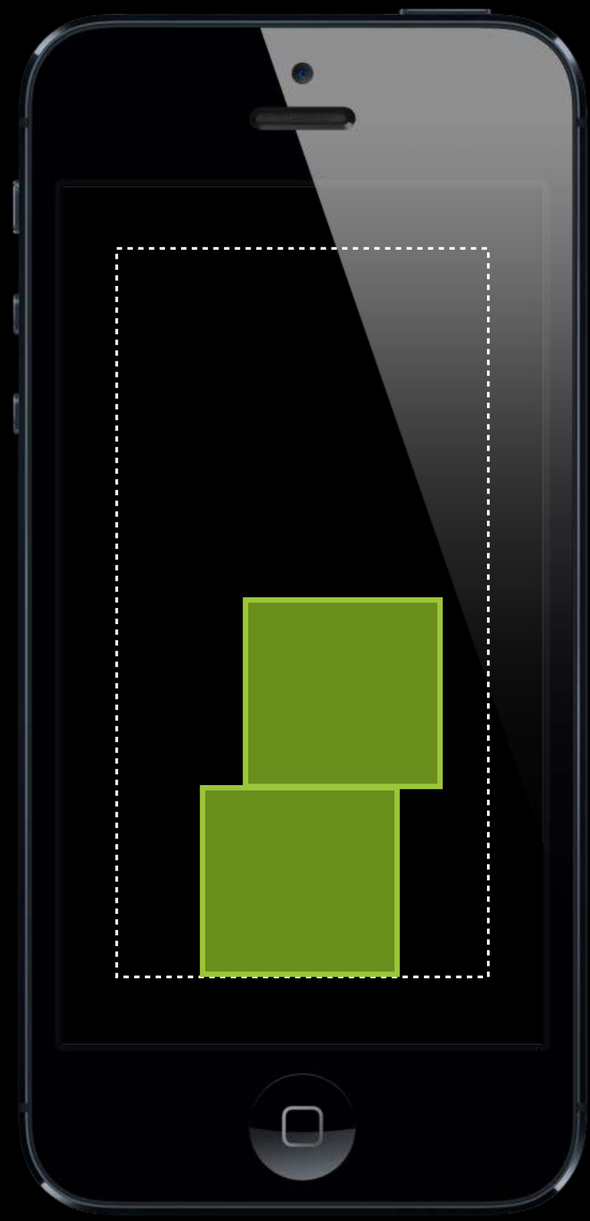


UICollisionBehavior

- Between a view and a boundary
- Or between views associated to the same behavior
- Or both, by default

UICollisionBehavior

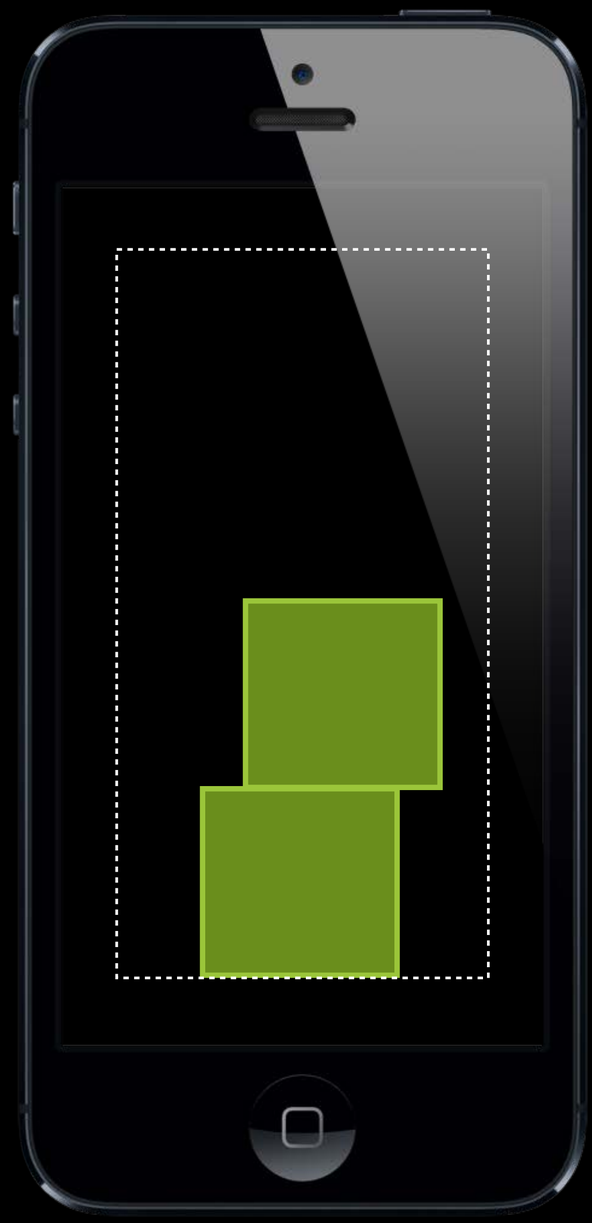
- Between a view and a boundary
- Or between views associated to the same behavior
- Or both, by default



UICollisionBehavior

- Collision mode

```
@property (nonatomic, readwrite)
UICollisionBehaviorMode collisionMode;
    UICollisionBehaviorModeItems
    UICollisionBehaviorModeBoundaries
    UICollisionBehaviorModeEverything
```

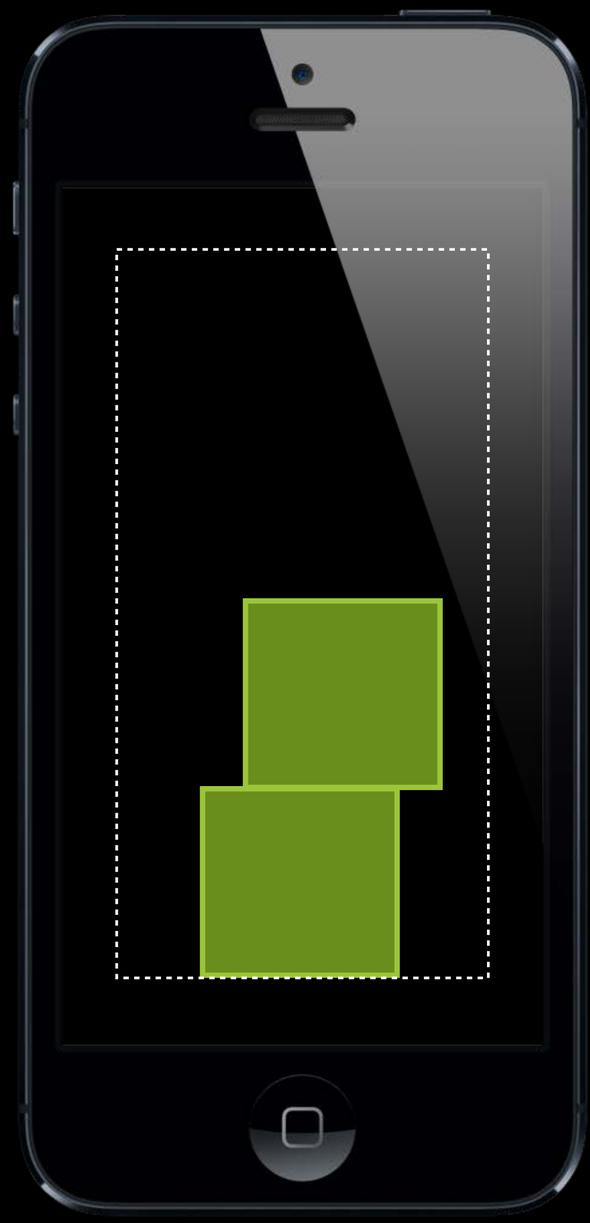


UICollisionBehavior

- Collision mode

```
@property (nonatomic, readwrite)
UICollisionBehaviorMode collisionMode;
    UICollisionBehaviorModeItems
    UICollisionBehaviorModeBoundaries
    UICollisionBehaviorModeEverything
```

- Items can be added or removed at any time

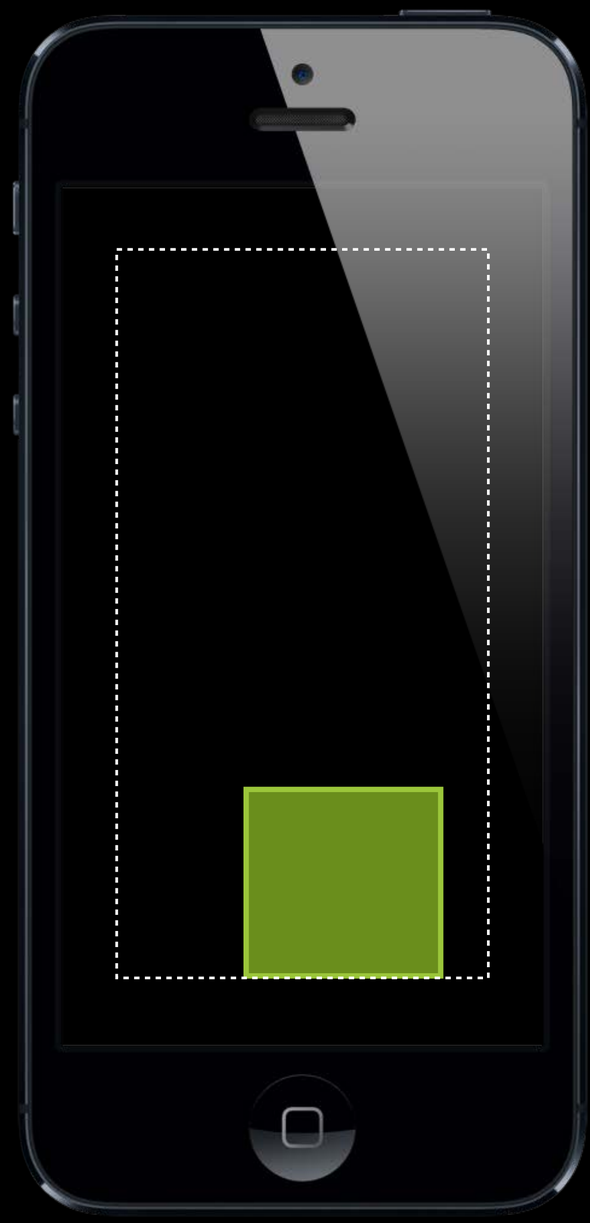


UICollisionBehavior

- Collision mode

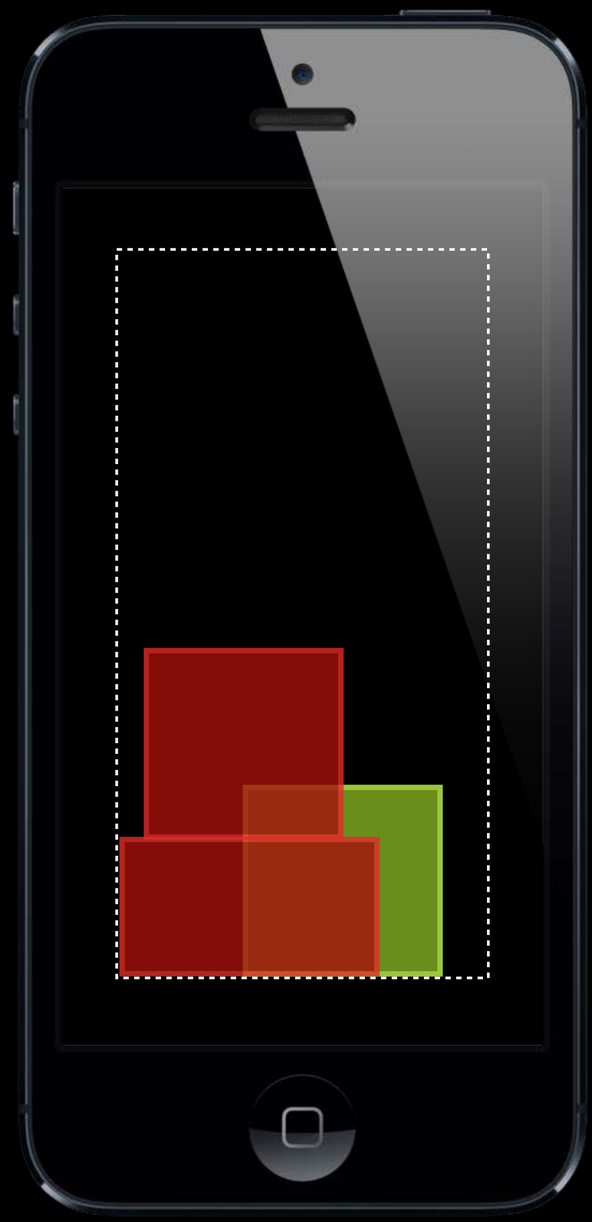
```
@property (nonatomic, readwrite)
UICollisionBehaviorMode collisionMode;
    UICollisionBehaviorModeItems
    UICollisionBehaviorModeBoundaries
    UICollisionBehaviorModeEverything
```

- Items can be added or removed at any time



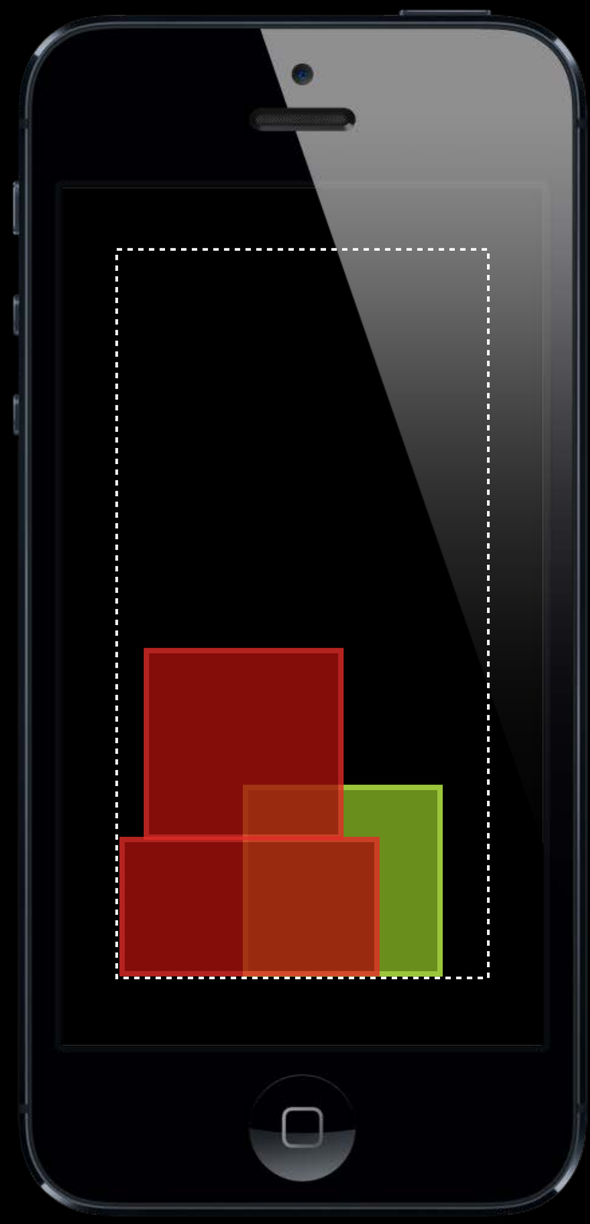
UICollisionBehavior

- You can create multiple collision behaviors
  - “red views collide with red views, green views with green views”



UICollisionBehavior

- You can create multiple collision behaviors
  - "red views collide with red views, green views with green views"
- A word of warning: collisions have a CPU cost

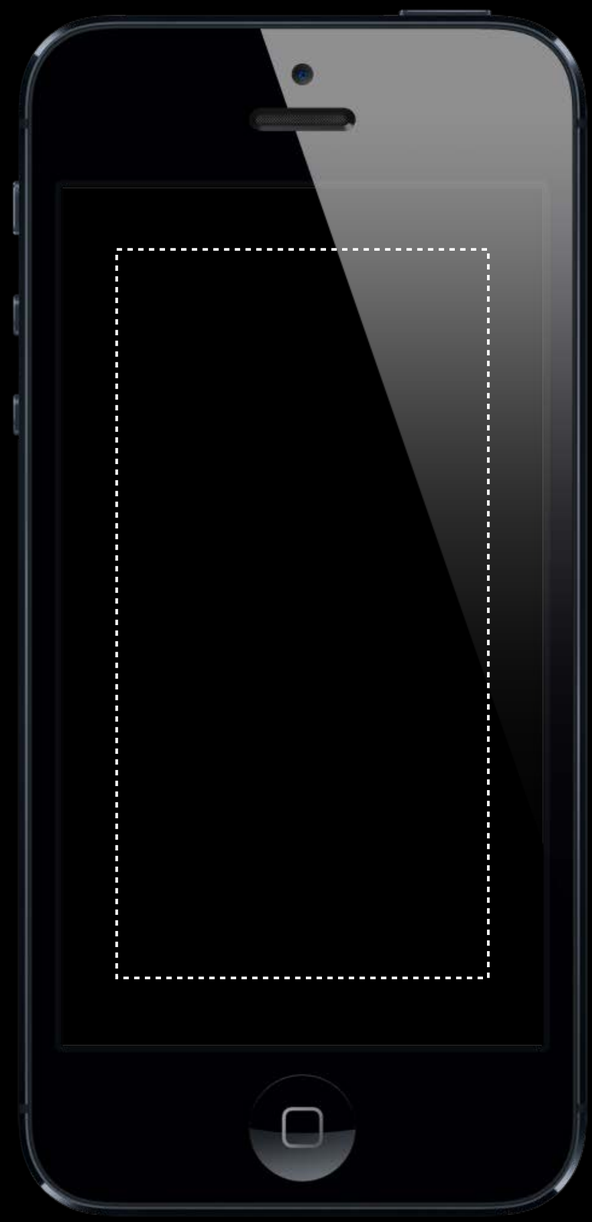


UICollisionBehavior

Boundaries

- Easy setup using the reference view

```
@property (nonatomic, readwrite) BOOL
translatesReferenceBoundsIntoBoundary;
```



UICollisionBehavior

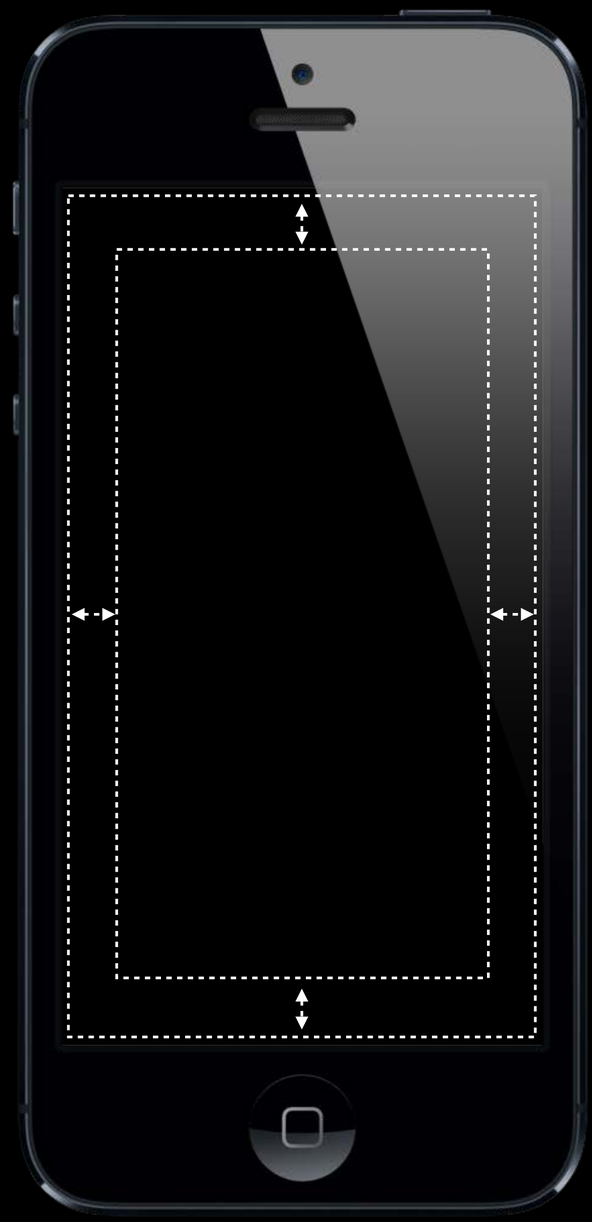
Boundaries

- Easy setup using the reference view

```
@property (nonatomic, readwrite) BOOL
translatesReferenceBoundsIntoBoundary;
```

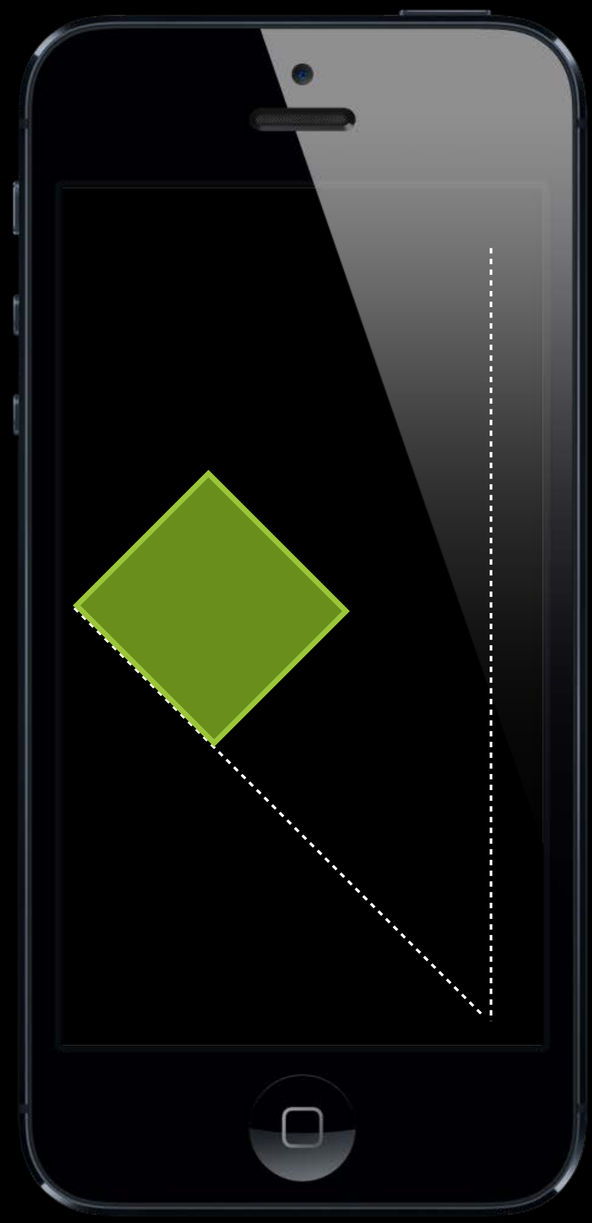
- Or with insets

```
-(void)setTranslatesReferenceBoundsIntoBoundaryWithInsets:
(UIEdgeInsets)insets;
```



UICollisionBehavior

Boundaries

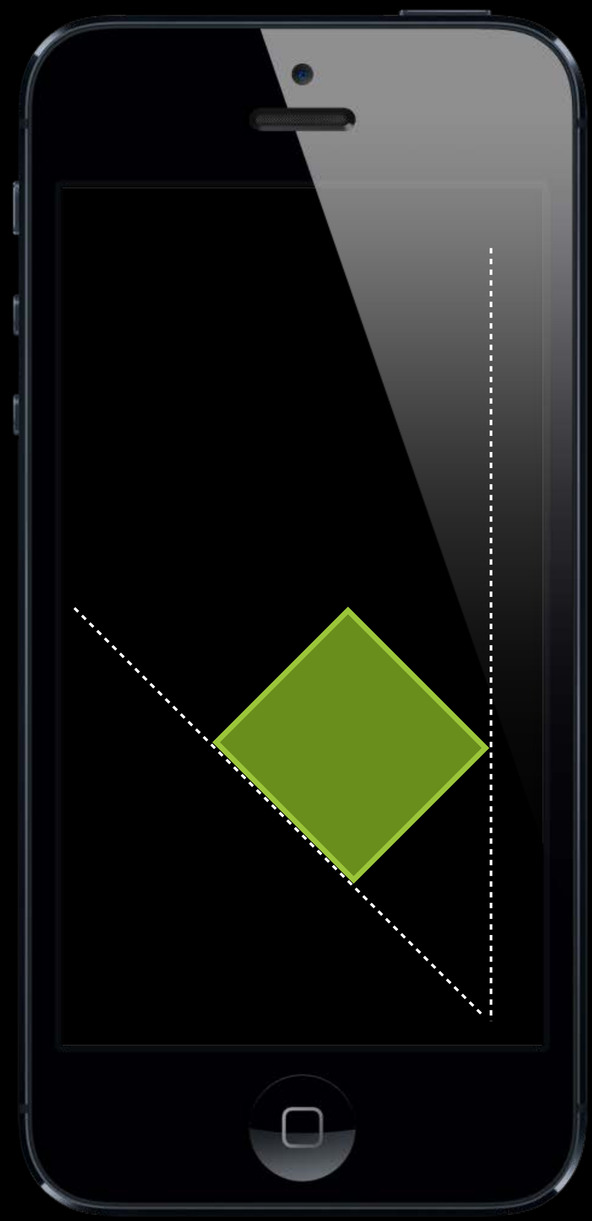


UICollisionBehavior

Boundaries

- Explicitly with segments

```
– (void)addBoundaryWithIdentifier:(id)identifier
fromPoint:(CGPoint)p1 toPoint:(CGPoint)p2;
```



UICollisionBehavior

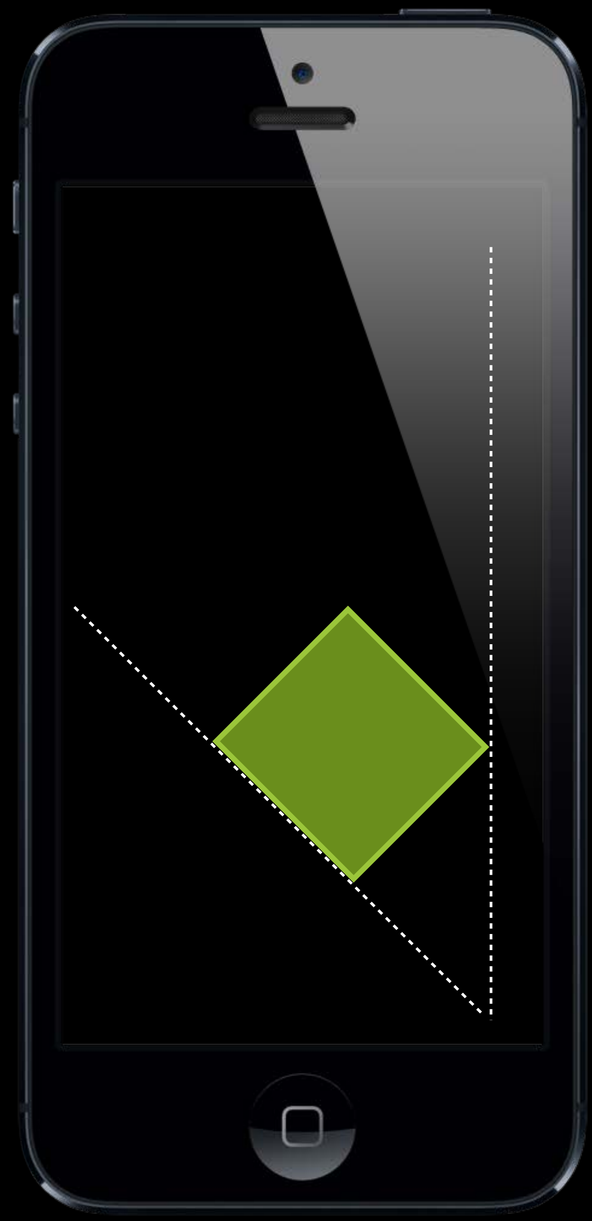
Boundaries

- Explicitly with segments

```
- (void)addBoundaryWithIdentifier:(id)identifier
fromPoint:(CGPoint)p1 toPoint:(CGPoint)p2;
```

- Or paths (approximated)

```
- (void)addBoundaryWithIdentifier:(id)identifier
forPath:(UIBezierPath*)p;
```



UICollisionBehavior

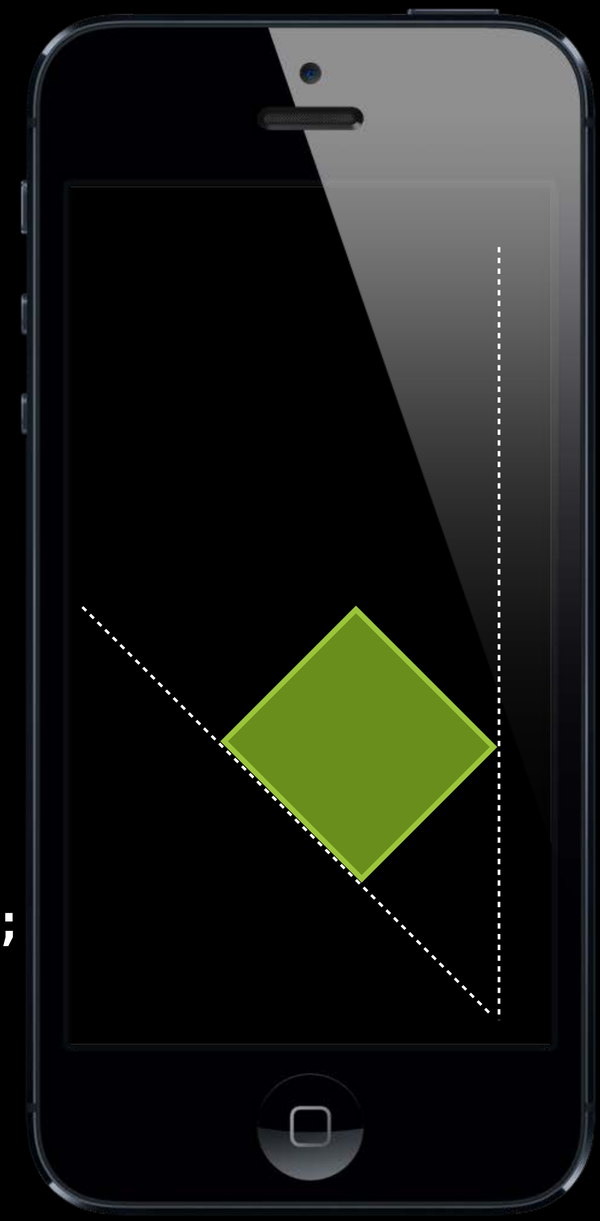
Boundaries

- Explicitly with segments

```
– (void)addBoundaryWithIdentifier:(id)identifier
fromPoint:(CGPoint)p1 toPoint:(CGPoint)p2;
```

- Or paths (approximated)

```
– (void)addBoundaryWithIdentifier:(id)identifier
forPath:(UIBezierPath*)p;
```



```
c = [[UICollisionBehavior alloc] initWithItems:@[view];

[c addBoundaryWithIdentifier:@”Wall1”
                   fromPoint:p1 toPoint:p2];
[c addBoundaryWithIdentifier:@”Wall2”
                   fromPoint:p3 toPoint:p4];
```

UICollisionBehaviorDelegate

- Callback on begin and end of contact

UICollisionBehaviorDelegate

- Callback on begin and end of contact
- Between views
  - `collisionBehavior: beganContactForItem: withItem: atPoint:`
  - `collisionBehavior: endedContactForItem: withItem:`

UICollisionBehaviorDelegate

- Callback on begin and end of contact
- Between views
  - `collisionBehavior: beganContactForItem: withItem: atPoint:`
  - `collisionBehavior: endedContactForItem: withItem:`
- Or boundaries
  - `collisionBehavior: beganContactForItem: withBoundaryIdentifier: atPoint:`
  - `collisionBehavior: endedContactForItem: withBoundaryIdentifier:`

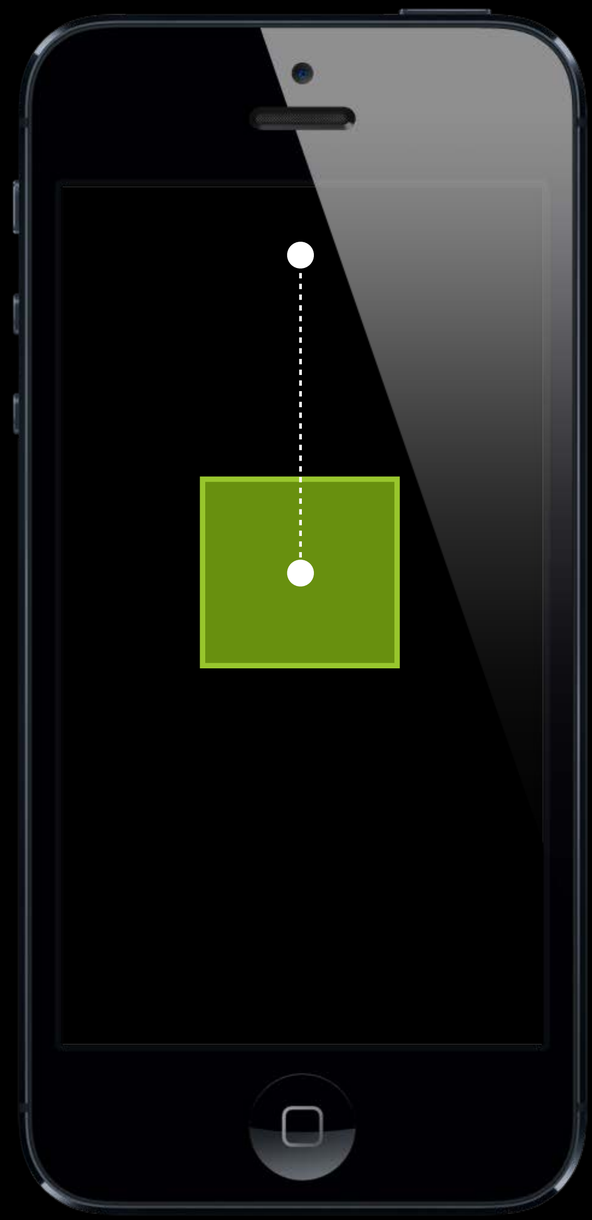
UICollisionBehaviorDelegate

- Callback on begin and end of contact
- Between views
  - `collisionBehavior: beganContactForItem: withItem: atPoint:`
  - `collisionBehavior: endedContactForItem: withItem:`
- Or boundaries
  - `collisionBehavior: beganContactForItem: withBoundaryIdentifier: atPoint:`
  - `collisionBehavior: endedContactForItem: withBoundaryIdentifier:`
- The reference boundary identifier is always nil

UIAttachmentBehavior

UIAttachmentBehavior

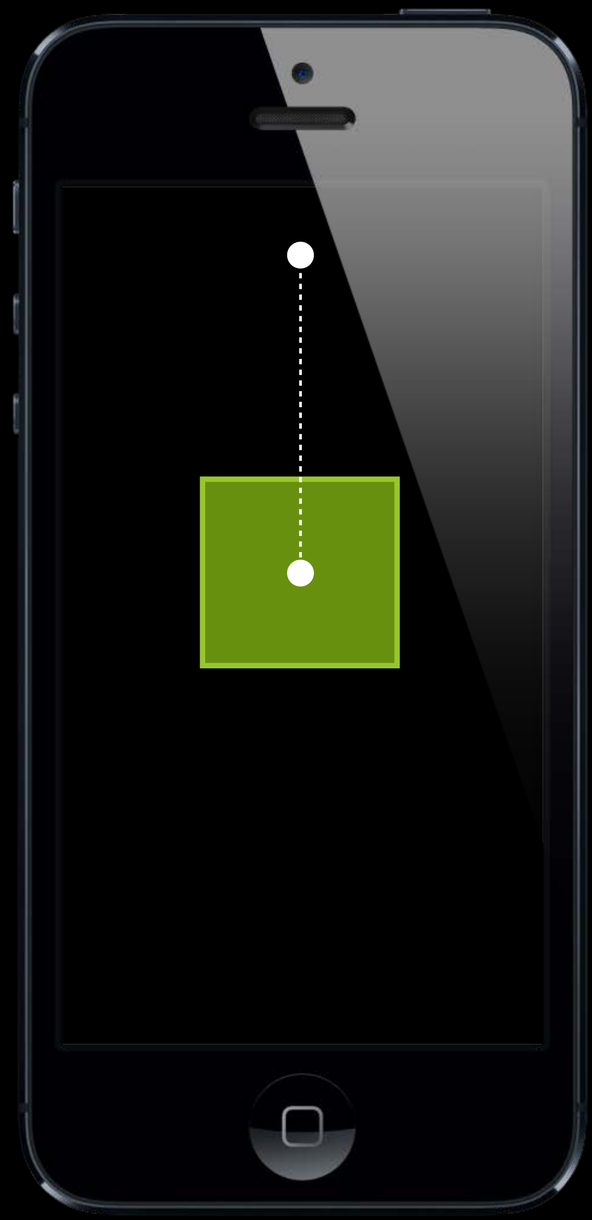
- Between a view and an anchor point



UIAttachmentBehavior

- Between a view and an anchor point

```
a1 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 attachedToAnchor:ap];
```

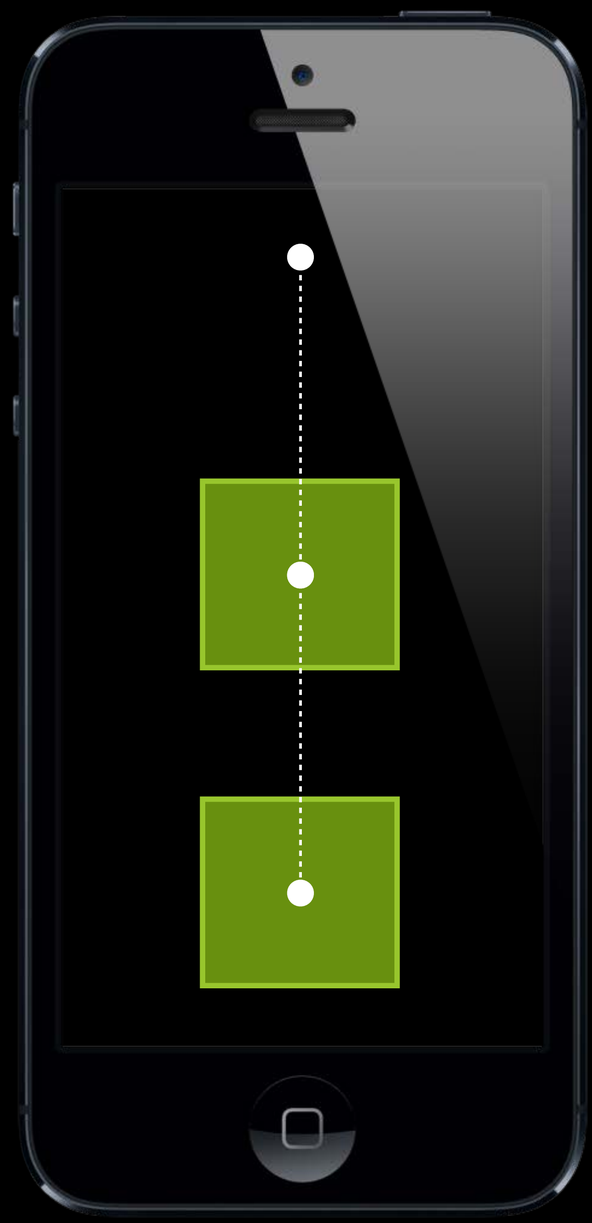


UIAttachmentBehavior

- Between a view and an anchor point

```
a1 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 attachedToAnchor:ap];
```

- Between two views



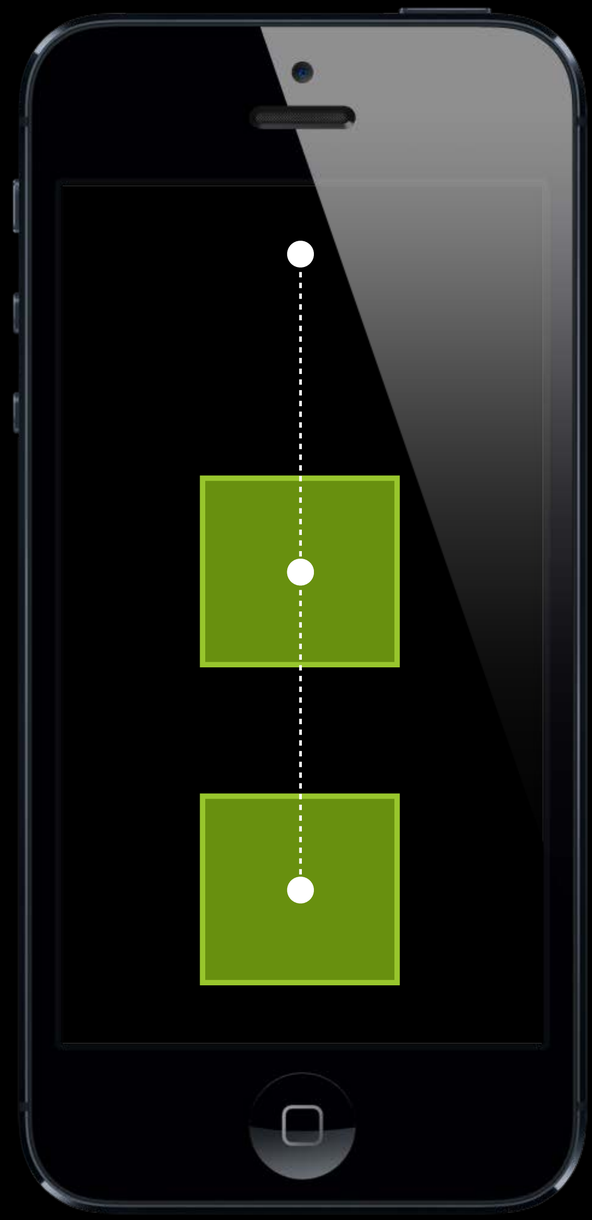
UIAttachmentBehavior

- Between a view and an anchor point

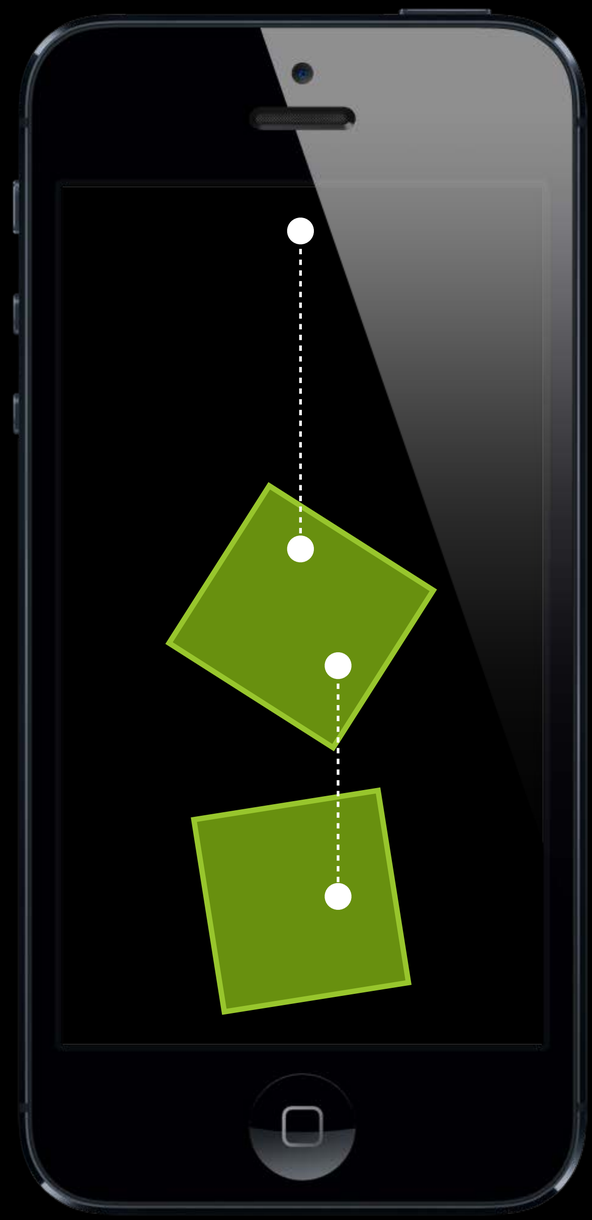
```
a1 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 attachedToAnchor:ap];
```

- Between two views

```
a2 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 attachedToItem:v2];
```



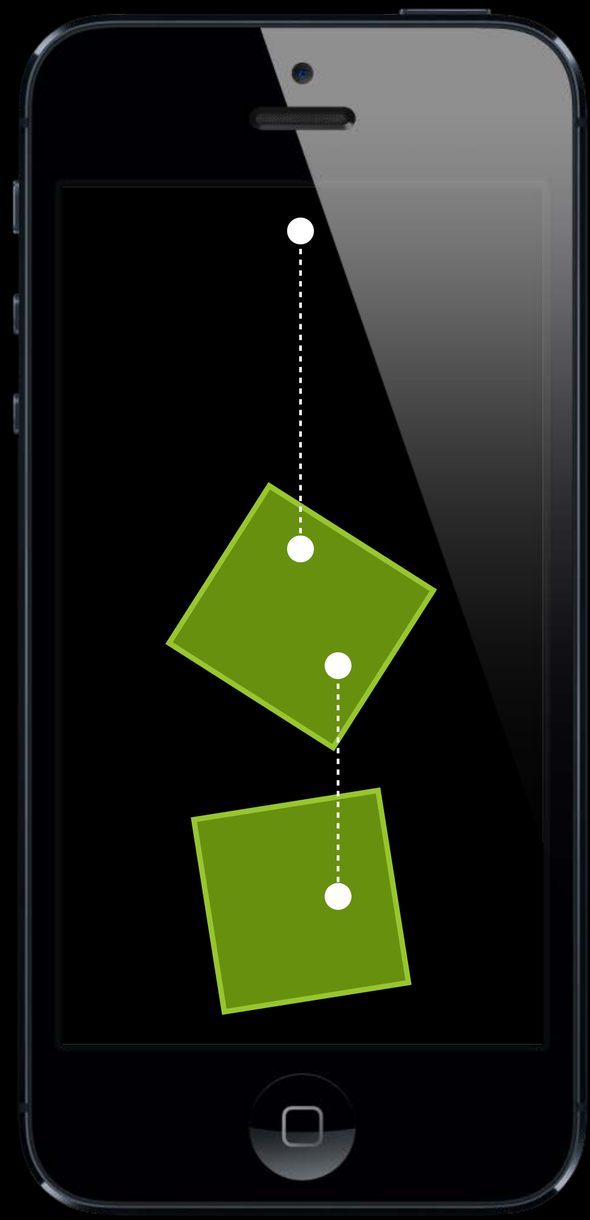
UIAttachmentBehavior



UIAttachmentBehavior

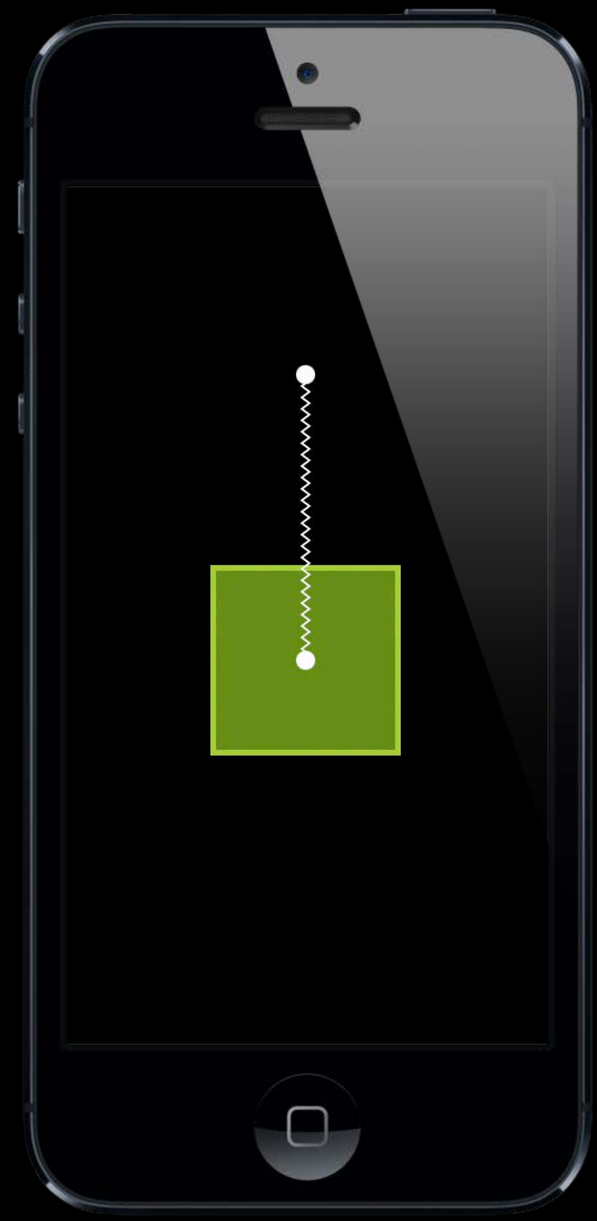
- The view attachment point can be an offset from the center

```
a1 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 point:p1
                attachedToAnchor:ap];
a2 = [[UIAttachmentBehavior alloc]
                initWithItem:v1 point:p2
                attachedToItem:v2];
```



UIAttachmentBehavior

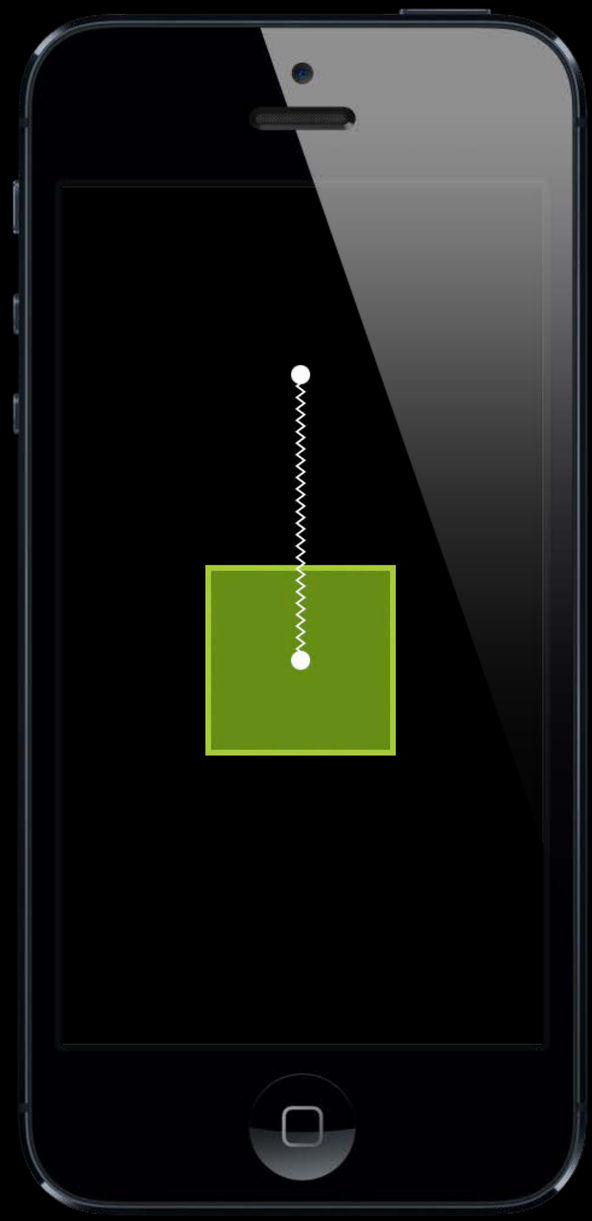
UIAttachmentBehavior



UIAttachmentBehavior

- An attachment can act as a spring

```
[a setFrequency:4.0];
[a setDamping:0.5];
```

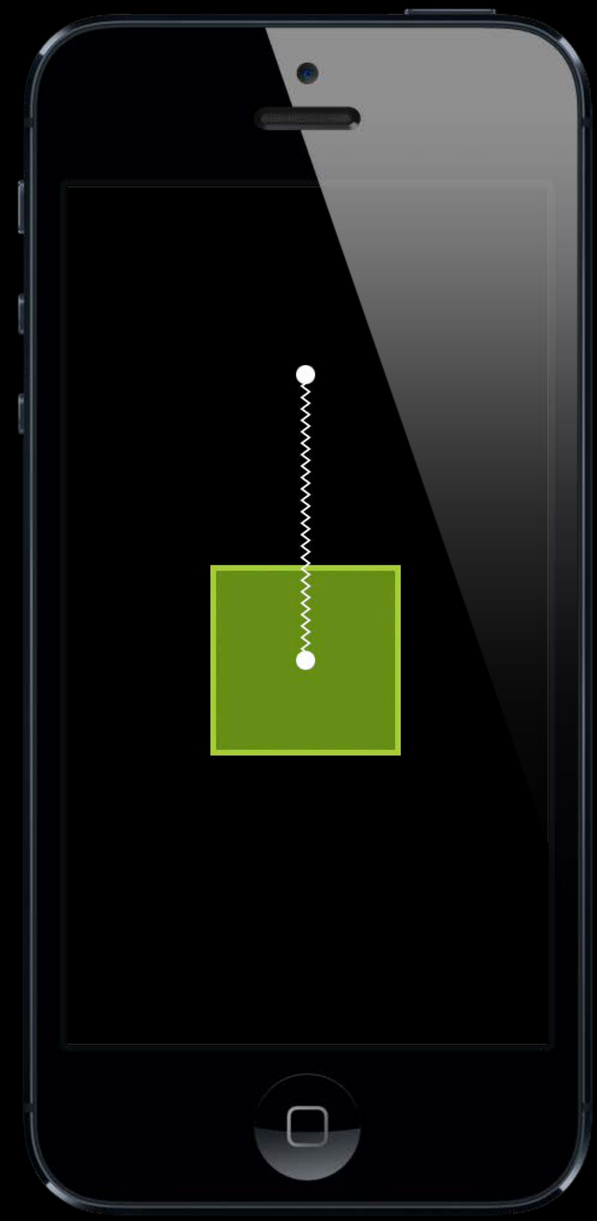


UIAttachmentBehavior

- An attachment can act as a spring

```
[a setFrequency:4.0];
[a setDamping:0.5];
```

- An anchor point can be modified later

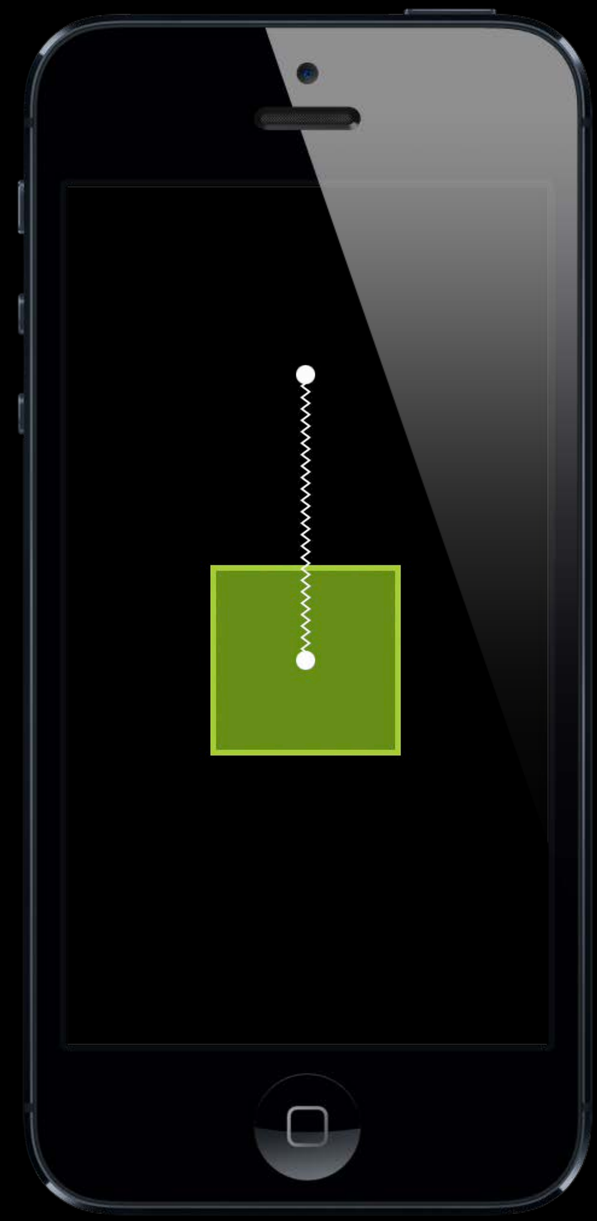


UIAttachmentBehavior

- An attachment can act as a spring

```
[a setFrequency:4.0];
[a setDamping:0.5];
```

- An anchor point can be modified later
- Only use `length` if to change the distance after setup

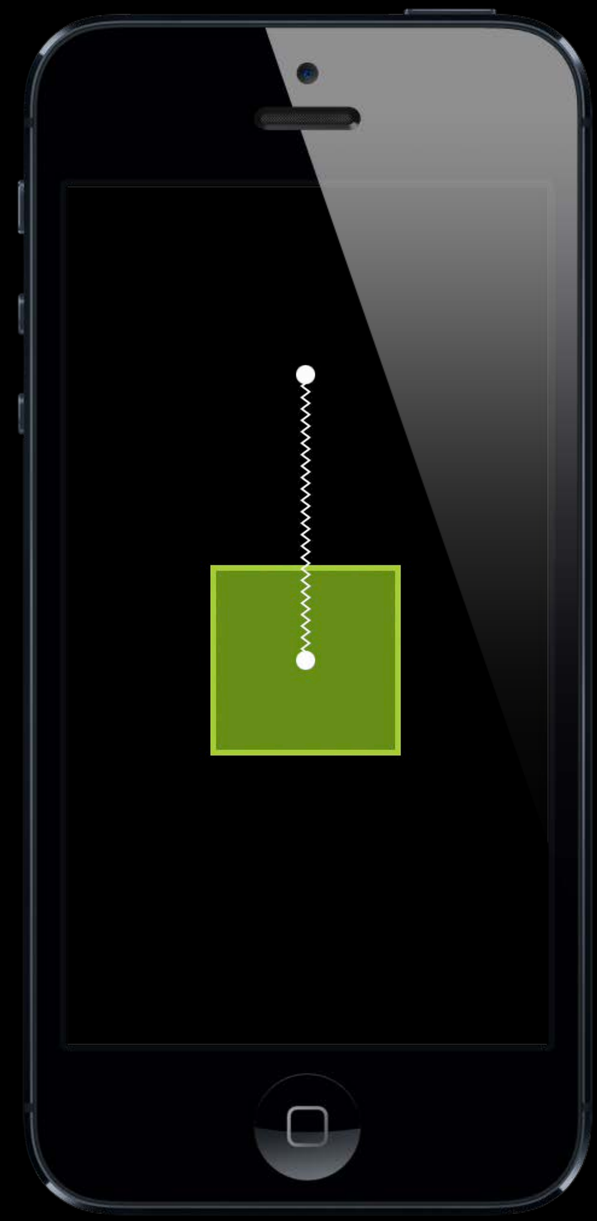


UIAttachmentBehavior

- An attachment can act as a spring

```
[a setFrequency:4.0];
[a setDamping:0.5];
```

- An anchor point can be modified later
- Only use `length` if to change the distance after setup
- Attachments are invisible!



Demo

UISnapBehavior

UISnapBehavior

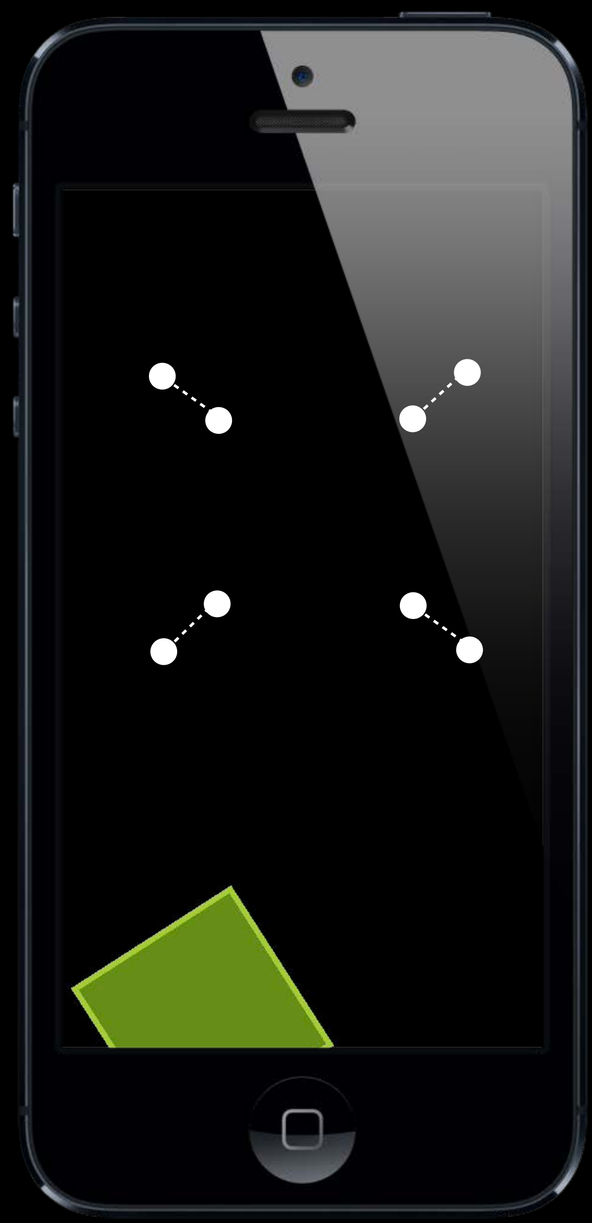
- Snap a view in place

UISnapBehavior

- Snap a view in place

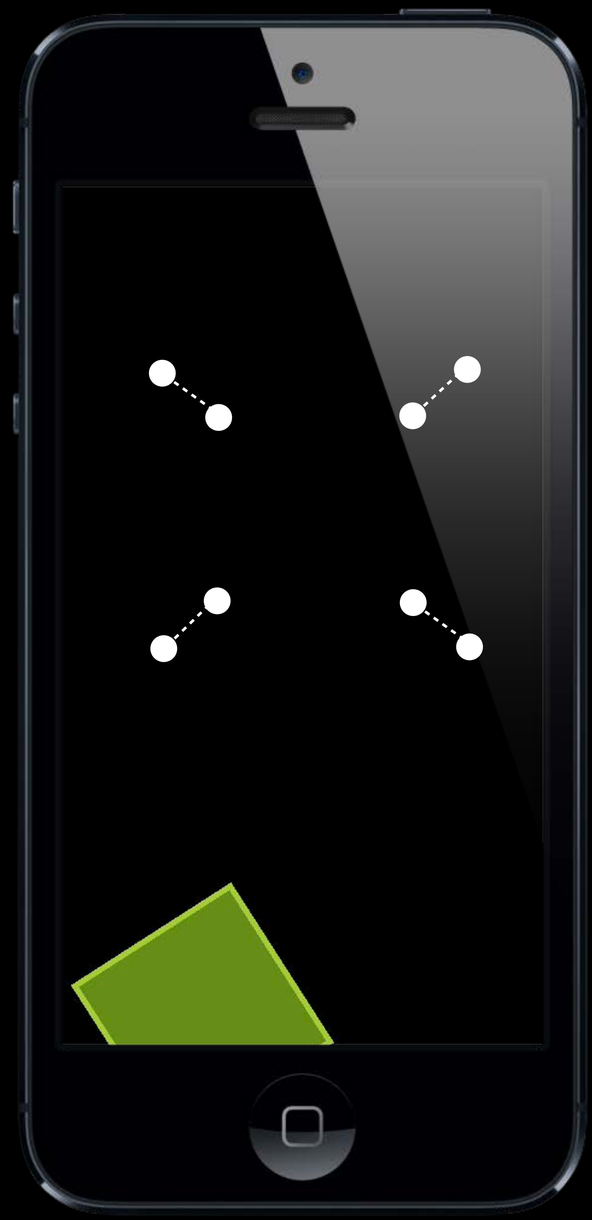
UISnapBehavior

- Snap a view in place
- Ensure position and angle



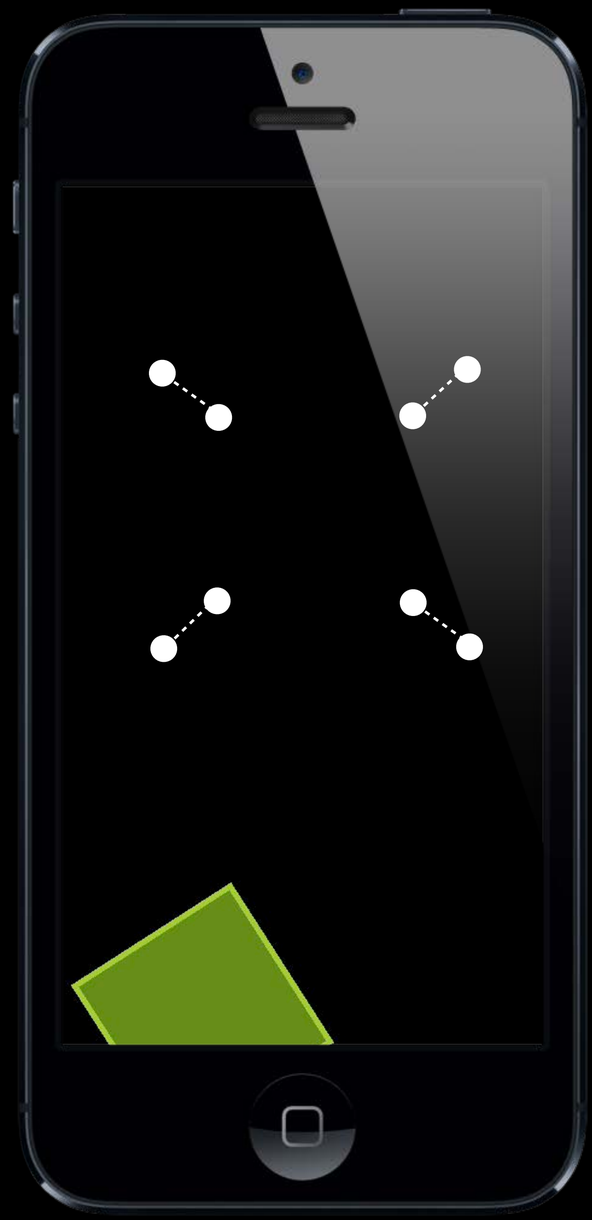
UISnapBehavior

- Snap a view in place
- Ensure position and angle
- Damping is customizable



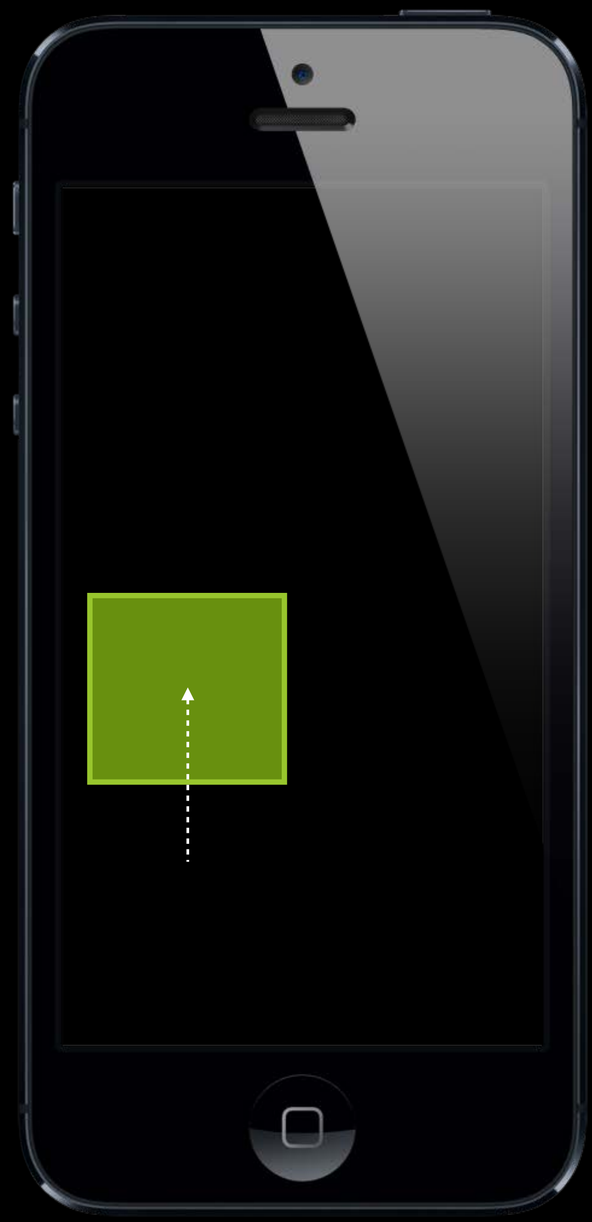
UISnapBehavior

- Snap a view in place
- Ensure position and angle
- Damping is customizable

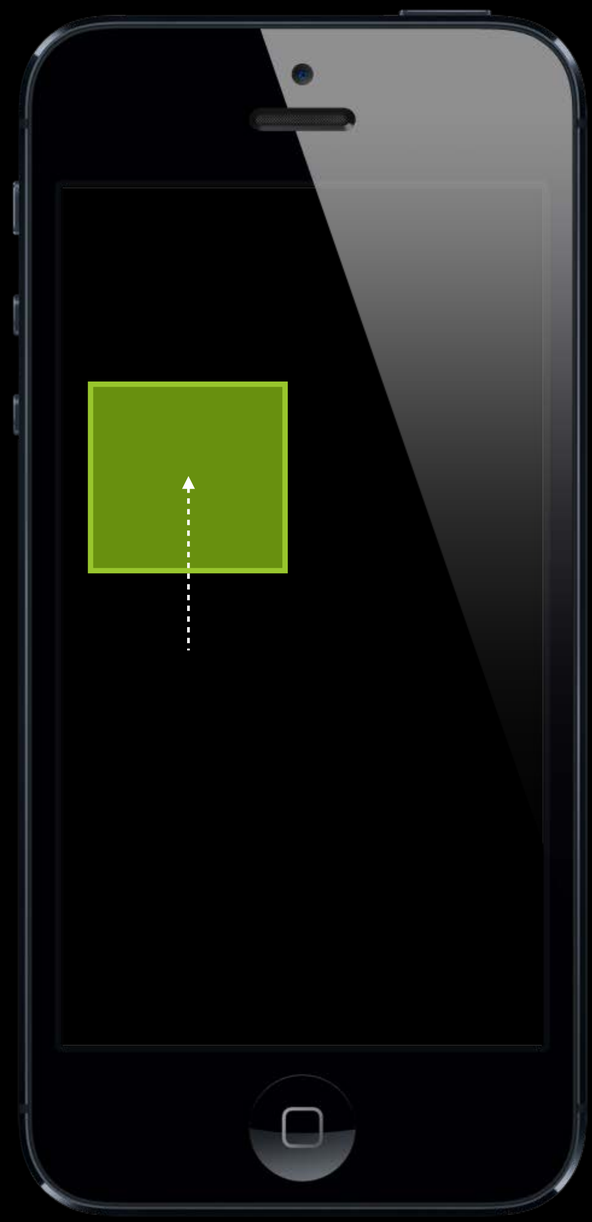


```
s = [[UISnapBehavior alloc] initWithItem:v
                            snapToPoint:p];
[animator addBehavior:s];
```

UIPushBehavior



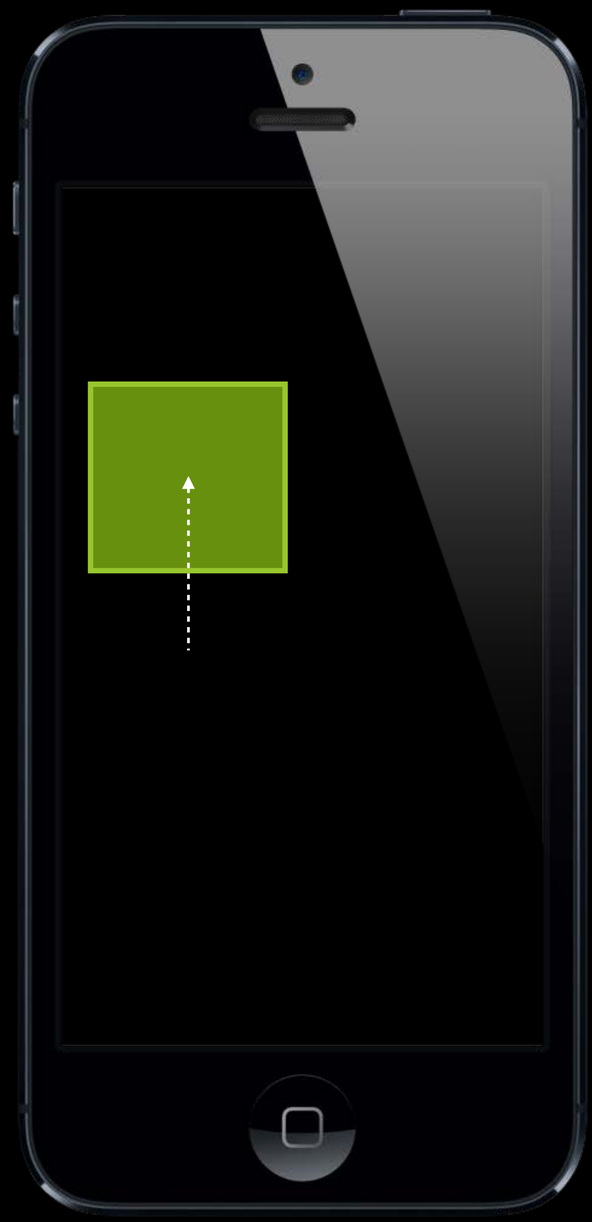
UIPushBehavior



UIPushBehavior

- Apply a force to a view (or views)

```
p = [[UIPushBehavior alloc]
                  initWithItems:@[view]
                  mode:UIPushBehaviorModeContinuous];
```



UIPushBehavior

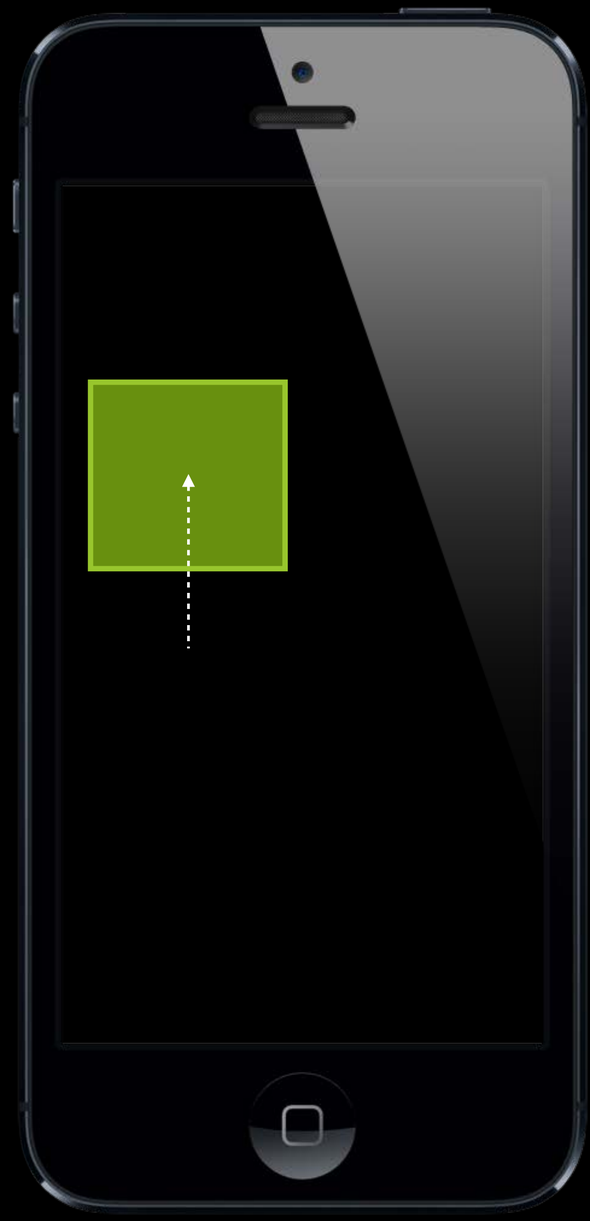
- Apply a force to a view (or views)

```
p = [[UIPushBehavior alloc]
                initWithItems:@[view]
                mode:UIPushBehaviorModeContinuous];
```

- A simple force vector

```
@property (readwrite,nonatomic) CGFloat xComponent;
@property (readwrite,nonatomic) CGFloat yComponent;

@property (readwrite,nonatomic) CGFloat angle;
@property (readwrite,nonatomic) CGFloat magnitude;
```



UIPushBehavior

- Apply a force to a view (or views)

```
p = [[UIPushBehavior alloc]
                initWithItems:@[view]
                mode:UIPushBehaviorModeContinuous];
```

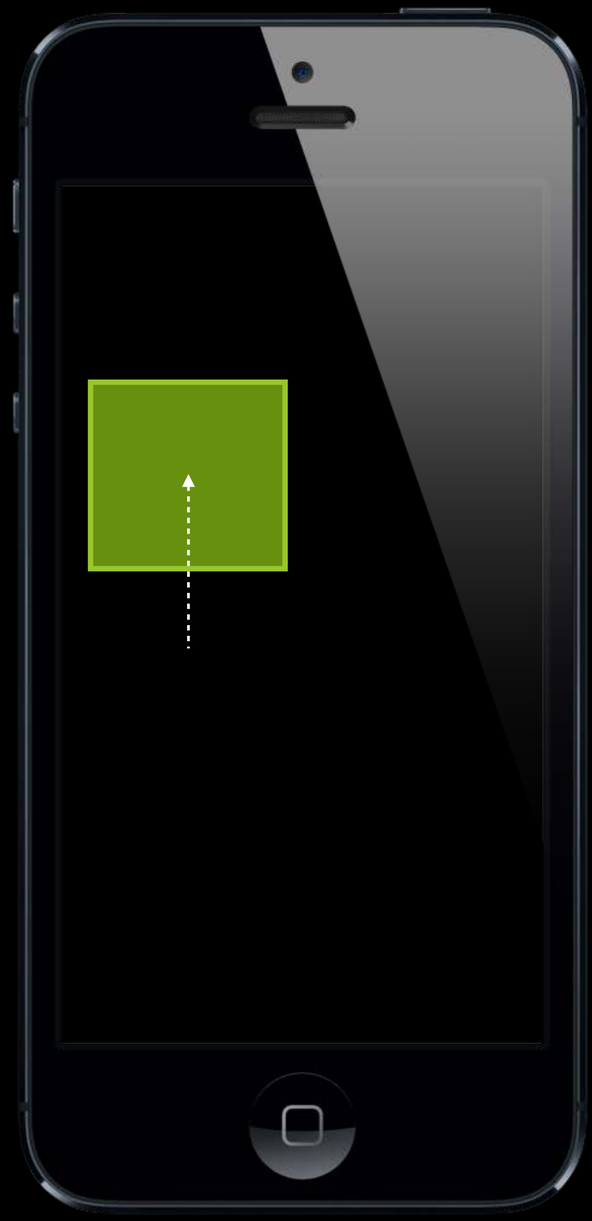
- A simple force vector

```
@property (readwrite,nonatomic) CGFloat xComponent;
@property (readwrite,nonatomic) CGFloat yComponent;

@property (readwrite,nonatomic) CGFloat angle;
@property (readwrite,nonatomic) CGFloat magnitude;
```

- The target point can be customized

```
[p setTargetPoint:x forItem:view];
```



UIPushBehavior

- Apply a force to a view (or views)

```
p = [[UIPushBehavior alloc]
                initWithItems:@[view]
                mode:UIPushBehaviorModeContinuous];
```

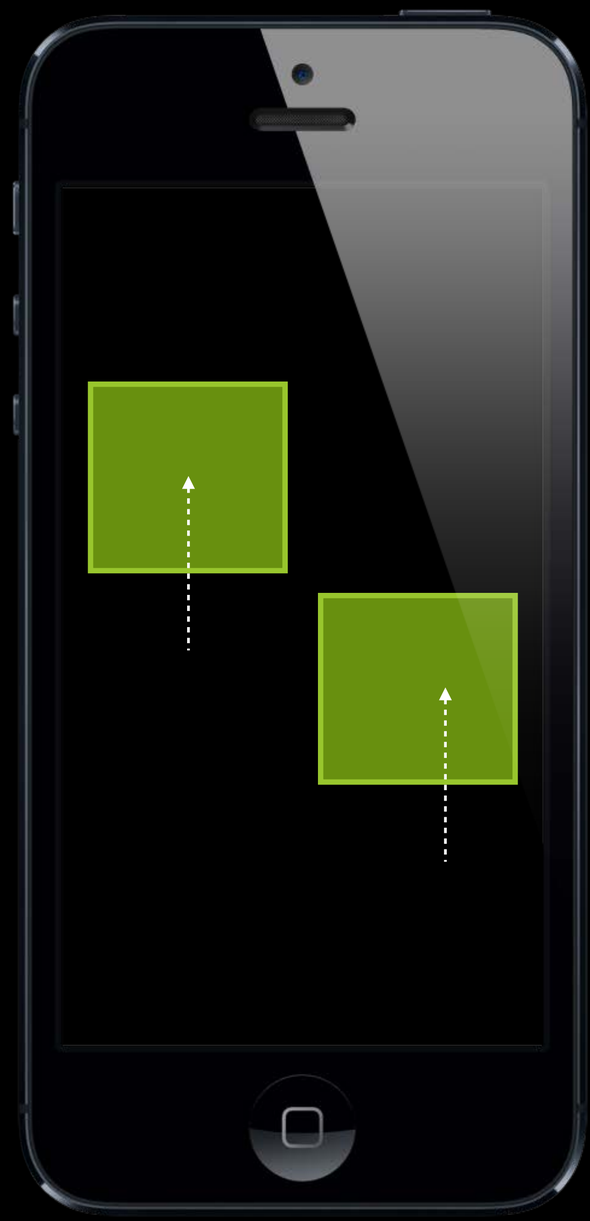
- A simple force vector

```
@property (readwrite,nonatomic) CGFloat xComponent;
@property (readwrite,nonatomic) CGFloat yComponent;

@property (readwrite,nonatomic) CGFloat angle;
@property (readwrite,nonatomic) CGFloat magnitude;
```

- The target point can be customized

```
[p setTargetPoint:x forItem:view];
```



UIPushBehavior

- Apply a force to a view (or views)

```
p = [[UIPushBehavior alloc]
                initWithItems:@[view]
                mode:UIPushBehaviorModeContinuous];
```

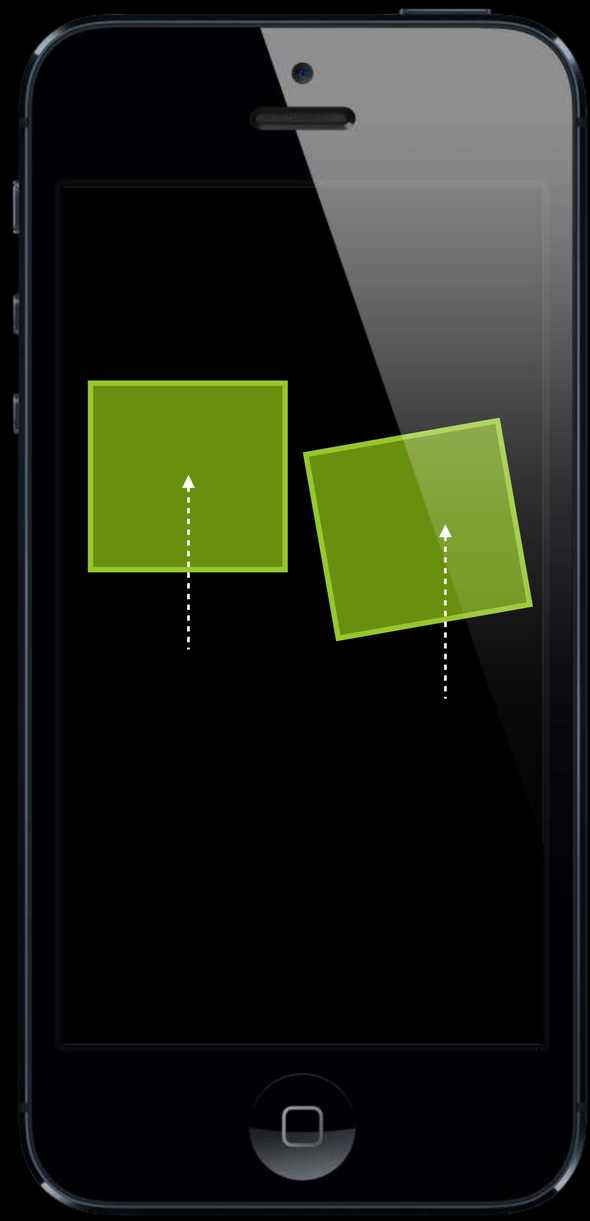
- A simple force vector

```
@property (readwrite,nonatomic) CGFloat xComponent;
@property (readwrite,nonatomic) CGFloat yComponent;

@property (readwrite,nonatomic) CGFloat angle;
@property (readwrite,nonatomic) CGFloat magnitude;
```

- The target point can be customized

```
[p setTargetPoint:x forItem:view];
```



A Well-known Unit…

The Newton

A Well-known Unit...

The Newton
Accelerate 1kg at a rate of 1m/s²

Introducing…

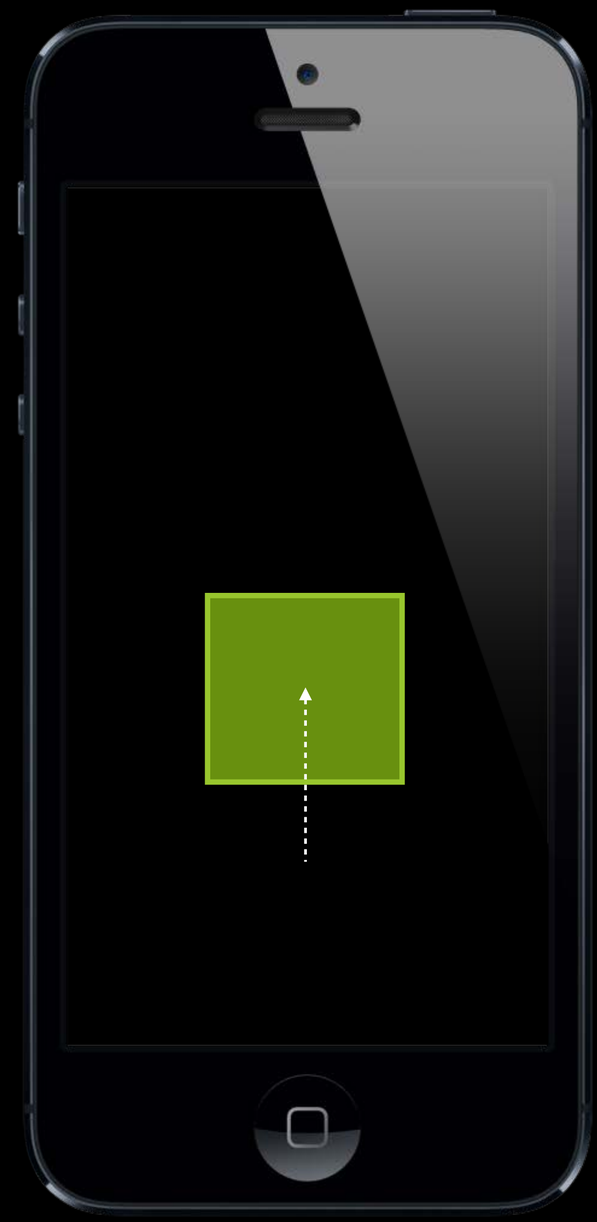
The UIKit Newton

Introducing…

The UIKit Newton

Accelerate a (100,100) view to 100 p/s²

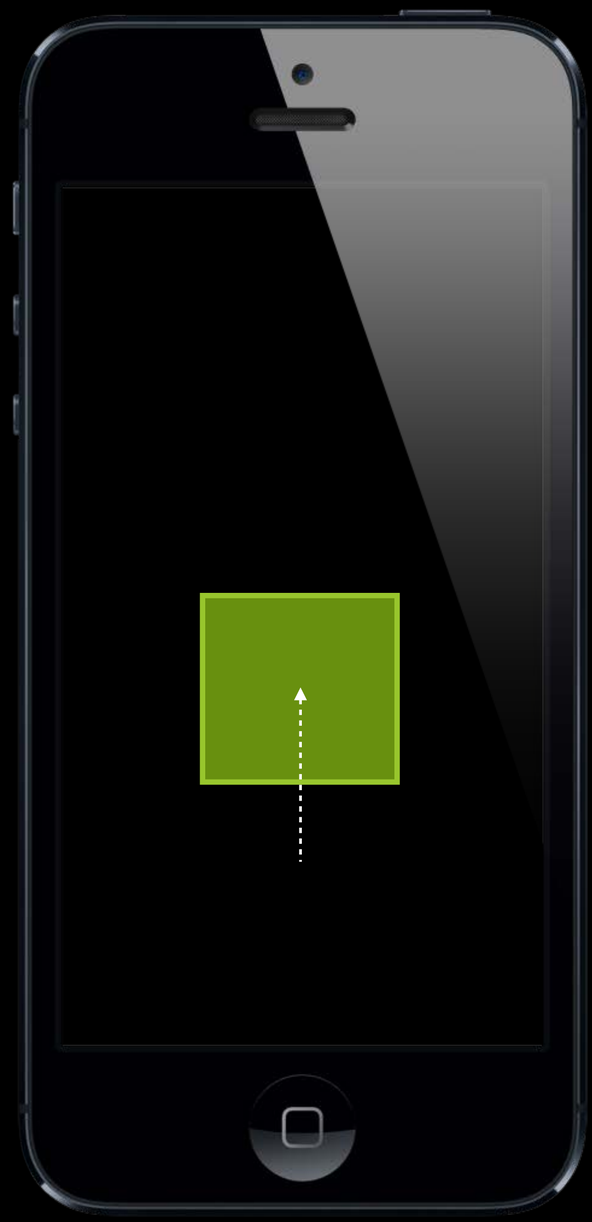
UIPushBehavior



UIPushBehavior

- Instantaneous mode

```
p2 = [[UIPushBehavior alloc]
            initWithItems:@[view]
            mode:UIPushBehaviorModeInstantaneous];
```

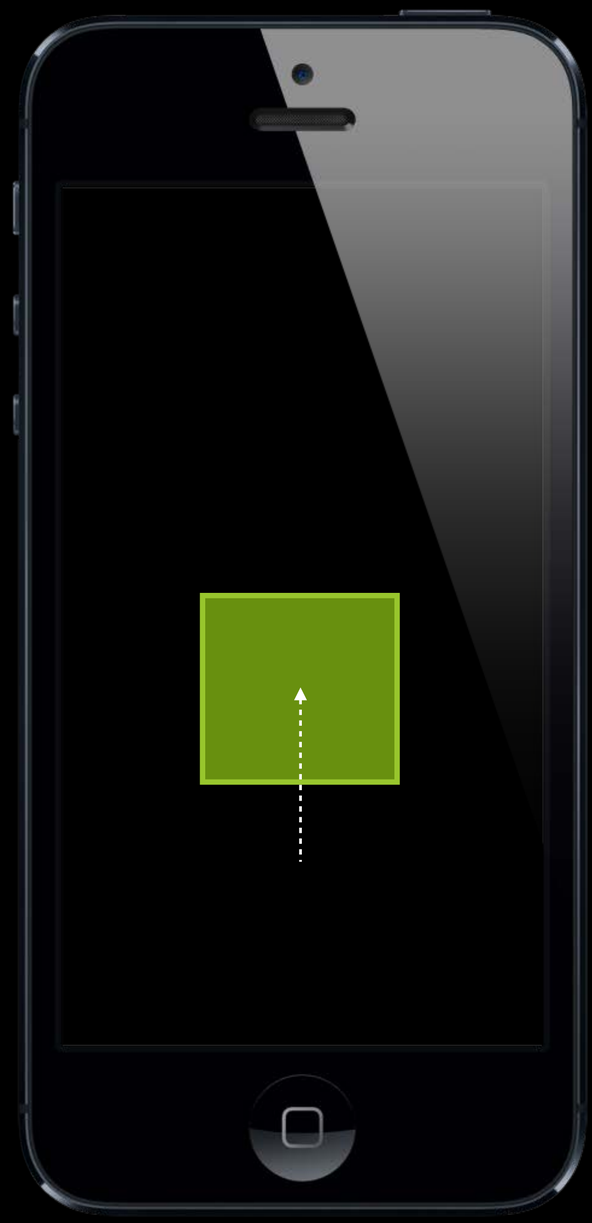


UIPushBehavior

- Instantaneous mode

```
p2 = [[UIPushBehavior alloc]
            initWithItems:@[view]
            mode:UIPushBehaviorModeInstantaneous];
```

- Velocity change is instantaneous

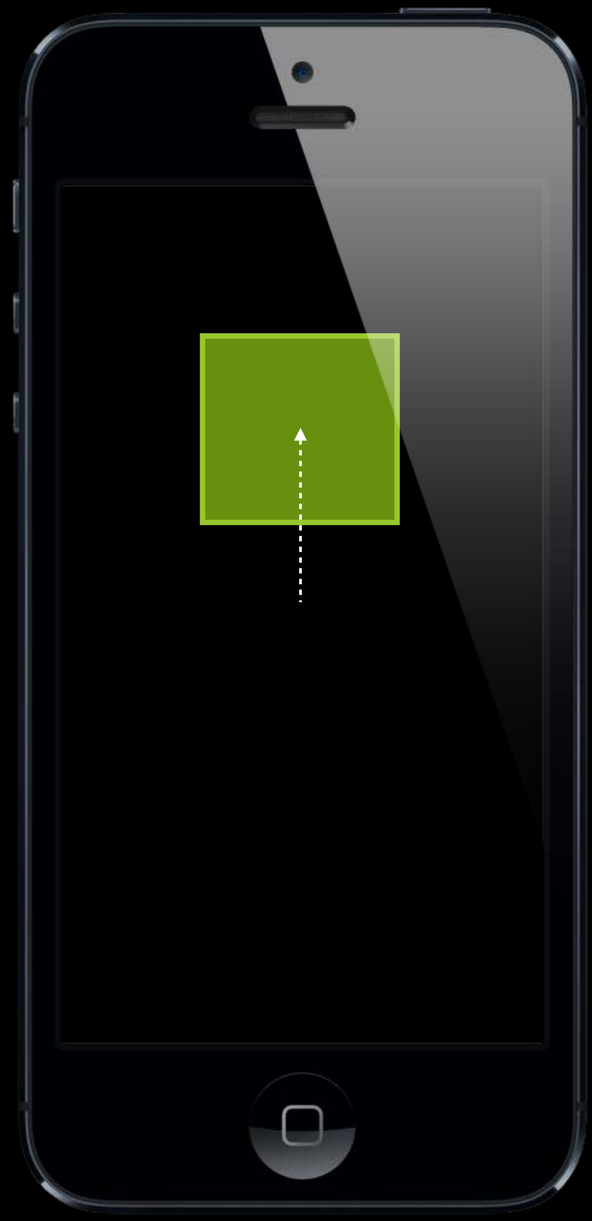


UIPushBehavior

- Instantaneous mode

```
p2 = [[UIPushBehavior alloc]
            initWithItems:@[view]
            mode:UIPushBehaviorModeInstantaneous];
```

- Velocity change is instantaneous

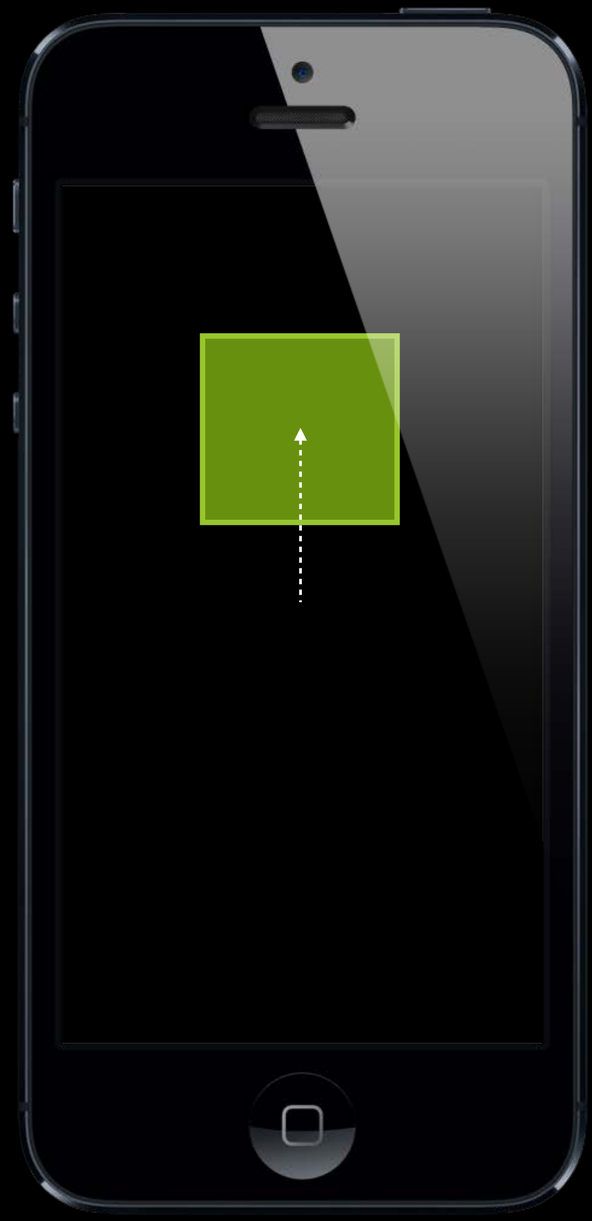


UIPushBehavior

- Instantaneous mode

```
p2 = [[UIPushBehavior alloc]
            initWithItems:@[view]
            mode:UIPushBehaviorModeInstantaneous];
```

- Velocity change is instantaneous
- Automatically disables itself after
  - Reenable with `[p setActive:TRUE]`

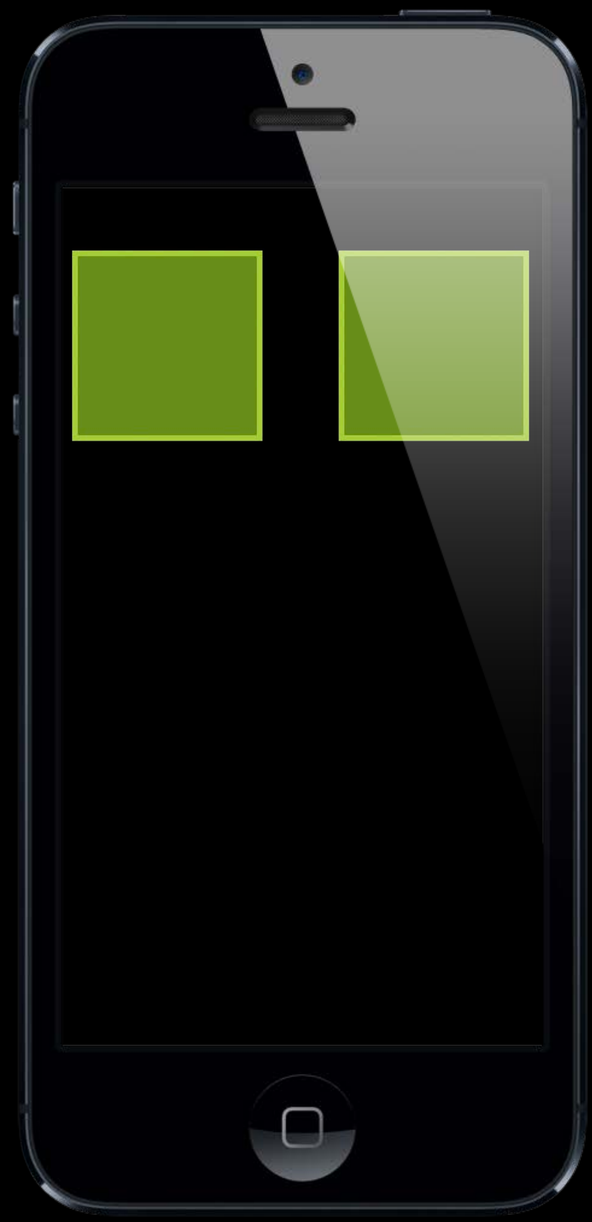


Demo

UIDynamicItemBehavior

UIDynamicItemBehavior

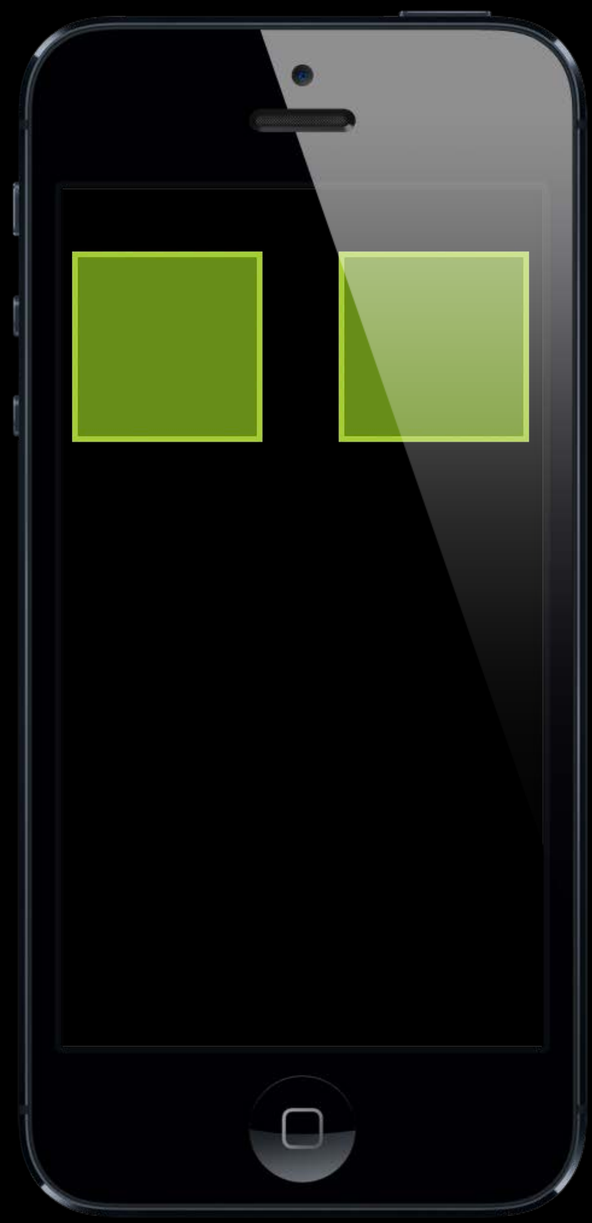
- Applied to one or many items



UIDynamicItemBehavior

- Applied to one or many items
- Change item-level properties

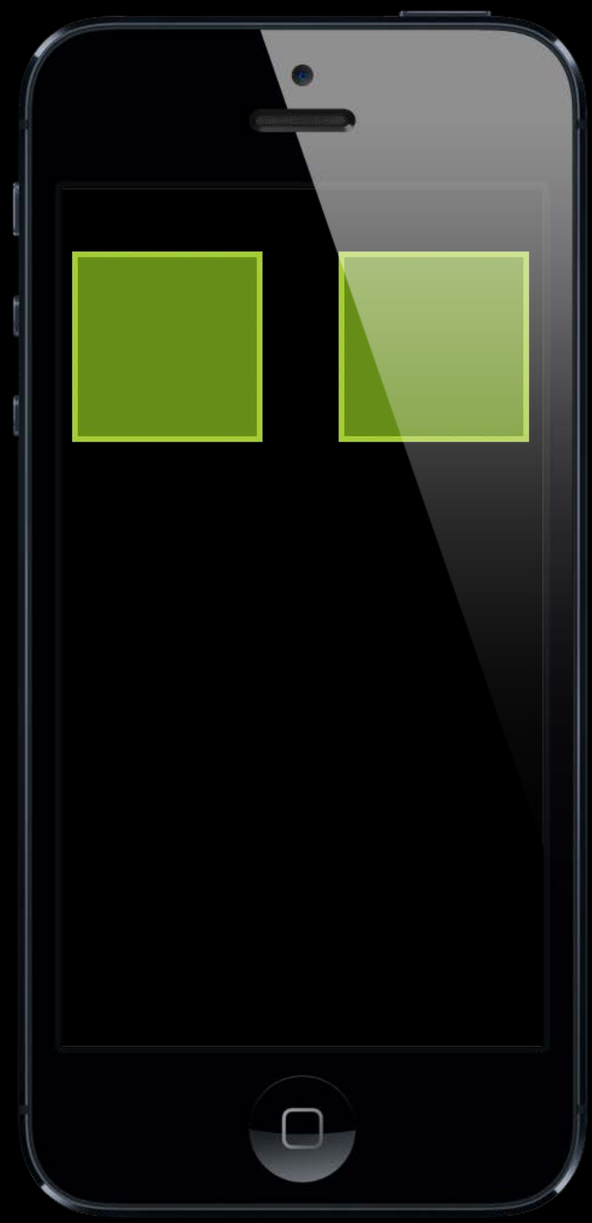
```
friction
resistance
angularResistance
elasticity
density
allowsRotation
```



UIDynamicItemBehavior

- Applied to one or many items
- Change item-level properties

```
friction
resistance
angularResistance
elasticity
density
allowsRotation
```

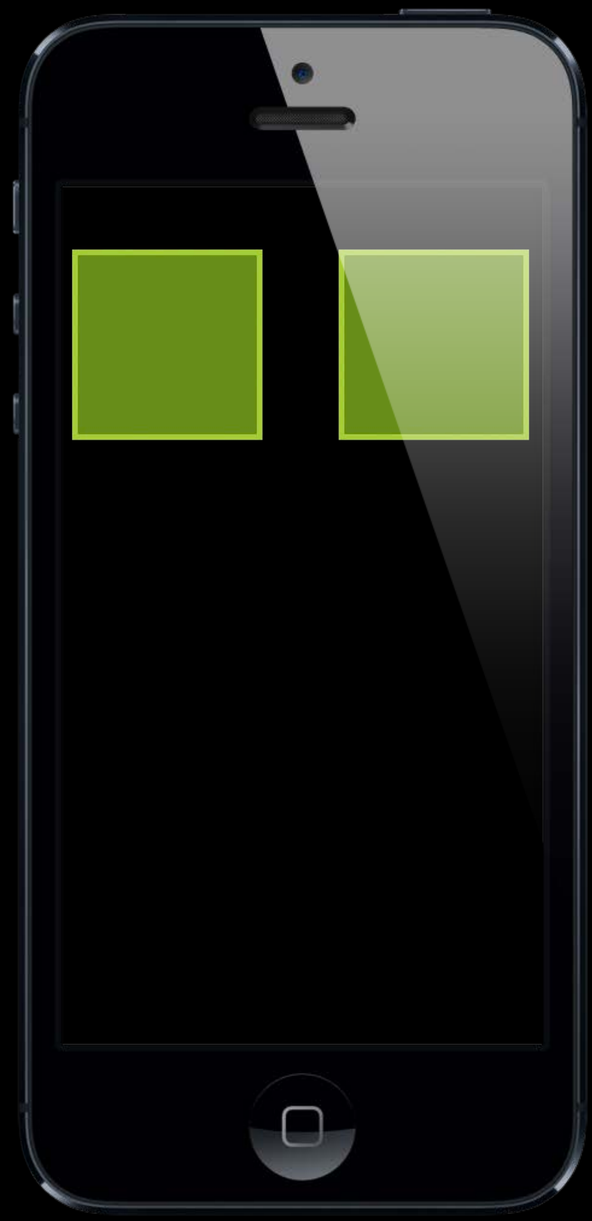


UIDynamicItemBehavior

- Applied to one or many items
- Change item-level properties

```
friction
resistance
angularResistance
elasticity
density
allowsRotation
```

- Directly add angular or linear velocities
  - i.e. map with a previous gesture



Applying Dynamics

Applying Dynamics

- Add and remove views to behaviors

Applying Dynamics

- Add and remove views to behaviors
- Configure, add, and remove behaviors to an animator

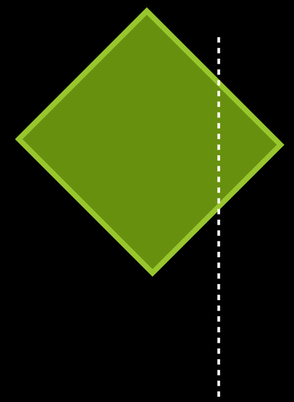
Applying Dynamics

- Add and remove views to behaviors
- Configure, add, and remove behaviors to an animator
- There is no step 3

Don’t Expect the Impossible

- You can create setups which don’t have solutions
- Build your system iteratively
- Not a physics-accurate tool

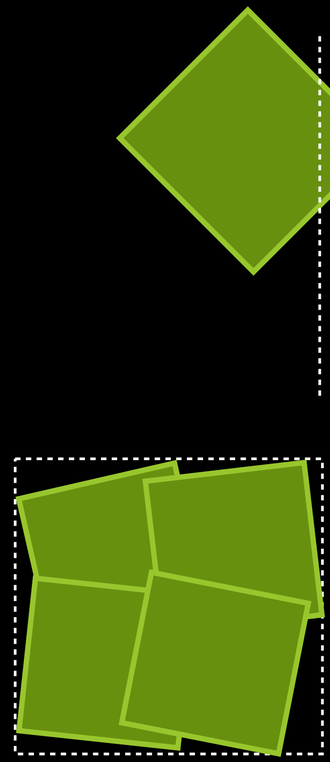
Don’t Expect the Impossible



- You can create setups which don’t have solutions
- Build your system iteratively
- Not a physics-accurate tool

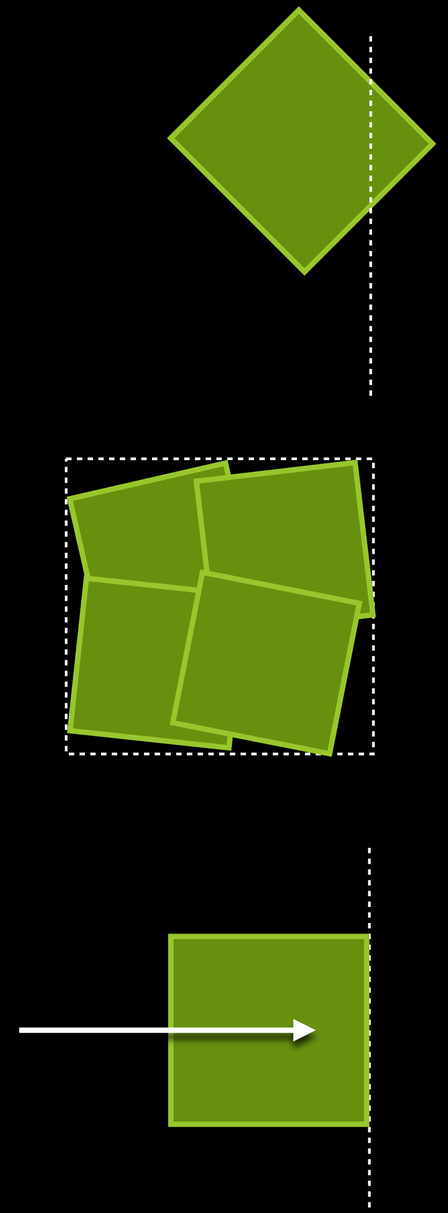
Don’t Expect the Impossible

- You can create setups which don’t have solutions
- Build your system iteratively
- Not a physics-accurate tool



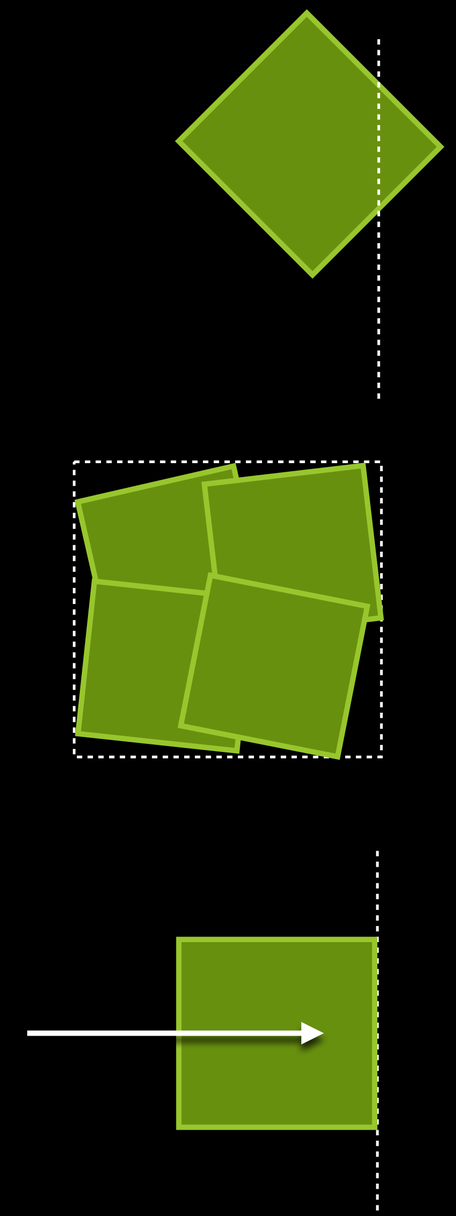
Don’t Expect the Impossible

- You can create setups which don’t have solutions
- Build your system iteratively
- Not a physics-accurate tool



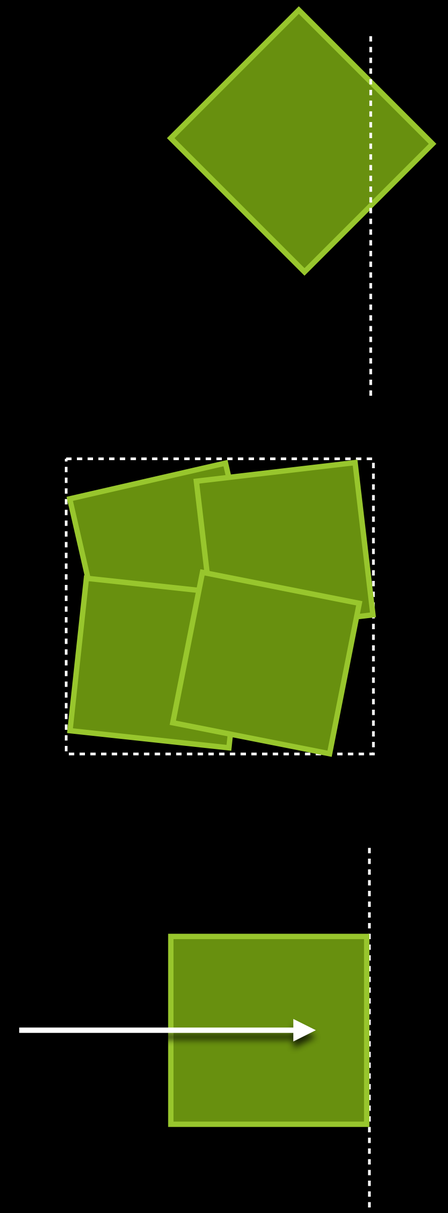
Don’t Expect the Impossible

- You can create setups which don’t have solutions
- Build your system iteratively



Don’t Expect the Impossible

- You can create setups which don’t have solutions
- Build your system iteratively
- Not a physics-accurate tool



Dynamic Items

UIDynamicItem Protocol

UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors

UIDynamicItem Protocol

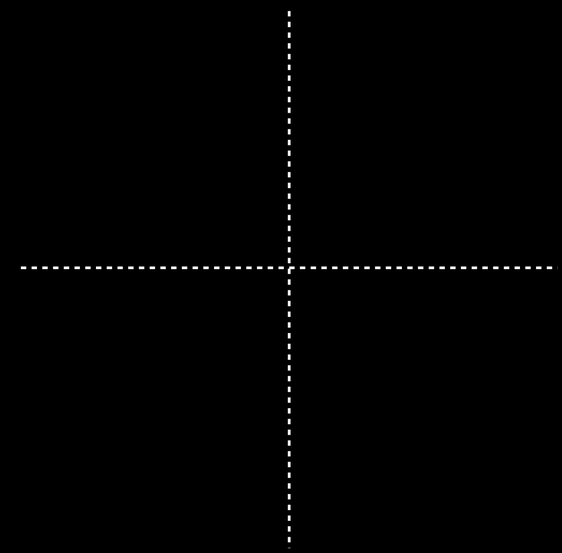
- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item

UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item

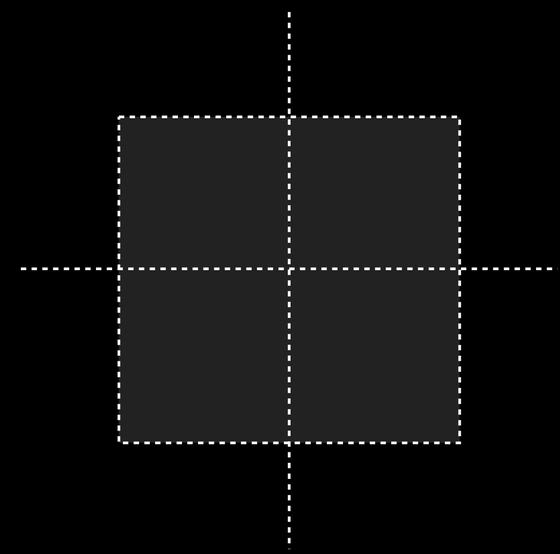
UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item



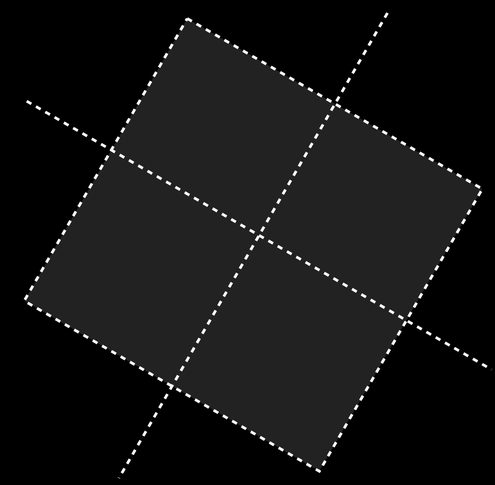
UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item



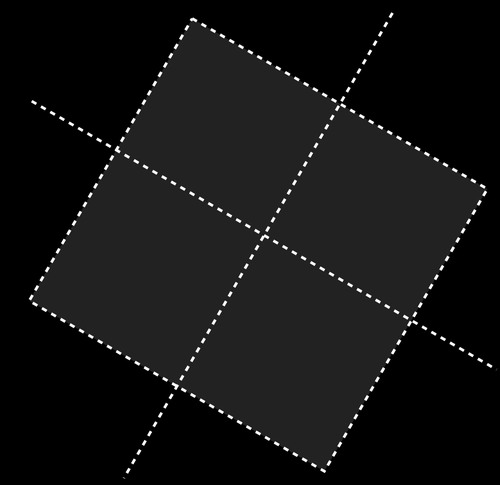
UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item



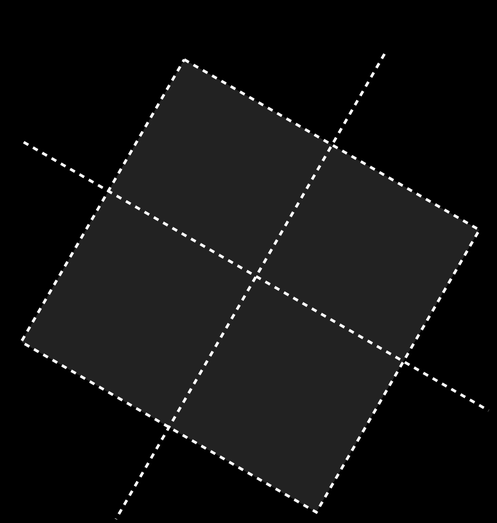
UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item
- UIView implements it



UIDynamicItem Protocol

- A protocol for items associated to predefined behaviors
- Describe what UIKit needs to animate an item
- UIView implements it
- You can implement it



UIDynamicItem

```
@protocol UIDynamicItem <NSObject>

@property (nonatomic, readwrite) CGPoint center;
@property (nonatomic, readonly)  CGRect bounds;
@property (nonatomic, readwrite) CGAffineTransform transform;

@end
```

Collection View

Layout Attributes

Dynamics and UICollectionView

Dynamics and UICollectionView

- `UICollectionViewLayoutAttributes` conforms to `UIDynamicItem`

Dynamics and UICollectionView

- `UICollectionViewLayoutAttributes` conforms to `UIDynamicItem`
- You can initialize an animator with a layout

```
animator = [[UIDynamicAnimator alloc] initWithCollectionViewLayout:myLayout]
```

Dynamics and UICollectionView

- `UICollectionViewLayoutAttributes` conforms to `UIDynamicItem`
- You can initialize an animator with a layout

```
animator = [[UIDynamicAnimator alloc] initWithCollectionViewLayout:myLayout]
```

- Just pass `UICollectionViewLayoutAttributes` to your behaviors

Dynamics and UICollectionView

- `UICollectionViewLayoutAttributes` conforms to `UIDynamicItem`
- You can initialize an animator with a layout

```
animator = [[UIDynamicAnimator alloc] initWithCollectionViewLayout:myLayout]
```

- Just pass `UICollectionViewLayoutAttributes` to your behaviors
- UIKit will invalidate the layout as needed

Demo

Summary

Summary

- An interaction-oriented animation system

Summary

- An interaction-oriented animation system
- Animate key elements

Summary

- An interaction-oriented animation system
- Animate key elements
- Focus on the user experience

More Information

Jake Behrens
UI Frameworks Evangelist
behrens@apple.com

Documentation
UIKit Framework Reference
http://developer.apple.com/library/ios

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

Related Sessions

Advanced Techniques with UIKit Dynamics	Presidio Thursday 3:15PM
Custom Transitions Using View Controllers	Pacific Heights Thursday 11:30AM
Introduction to Sprite Kit	Presidio Wednesday 11:30AM
Designing Games with Sprite Kit	Mission Wednesday 2:00PM

Labs

UIKit Dynamics Lab	Frameworks Lab A Wednesday 4:30PM	
Scroll View, Collection View, and Table View on iOS Lab	Frameworks Lab B Wednesday 2:00PM	
Scroll View, Collection View, and Table View on iOS Lab	Frameworks Lab B Thursday 11:30AM	
Cocoa Touch Animation Lab	Frameworks Lab B Thursday 2:00PM	
Cocoa Touch Lab	Frameworks Lab A Wednesday 9:00AM	
Cocoa Touch Lab	Frameworks Lab B Friday 9:00AM	
Sprite Kit Lab	Graphics and Games Lab B Wednesday 3:15PM	

WWDC2013