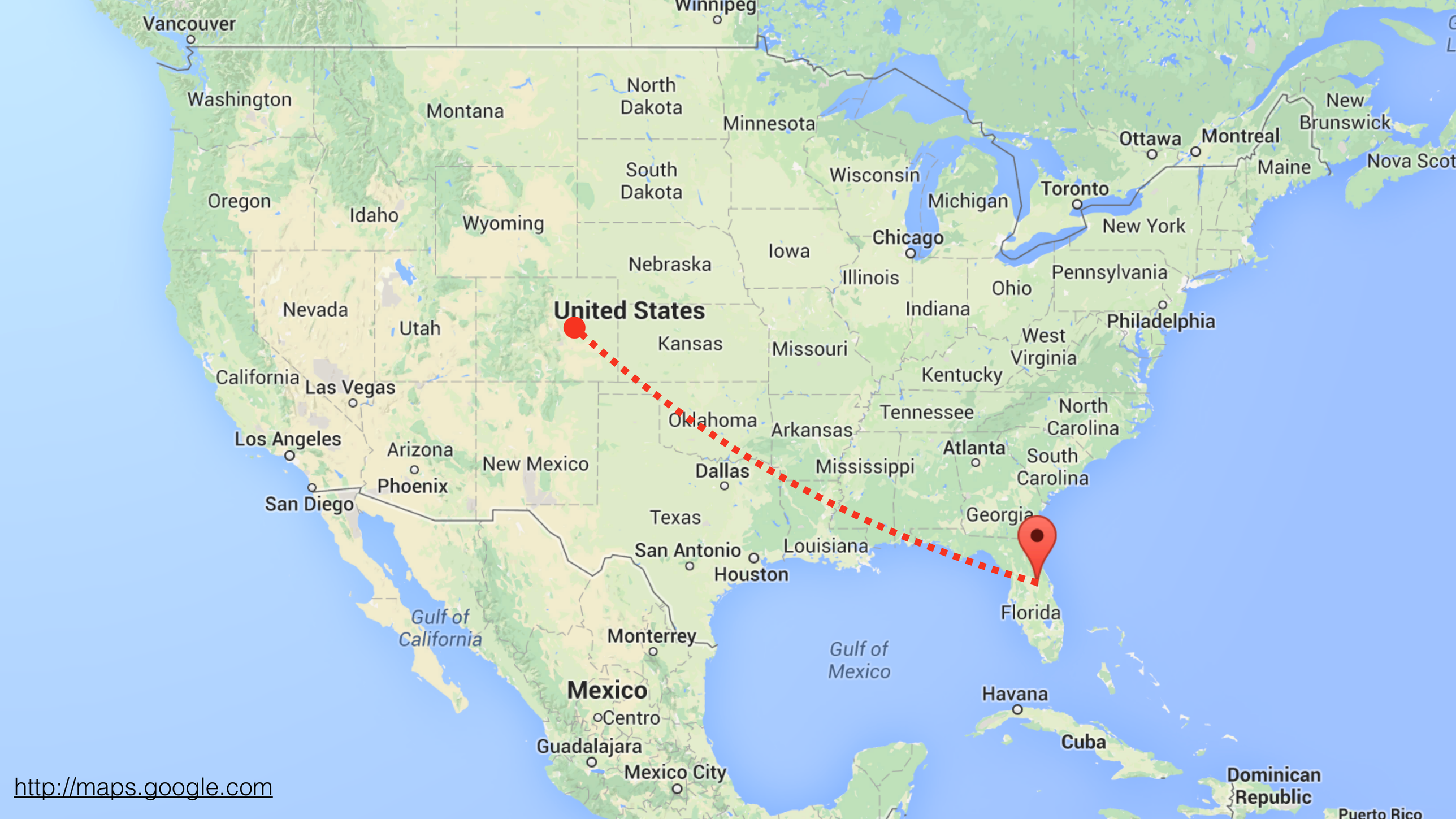
Practical UIKit Dynamics

Jon Friskics
@jonfriskics

360iDev
August 24, 2014



http://maps.google.com

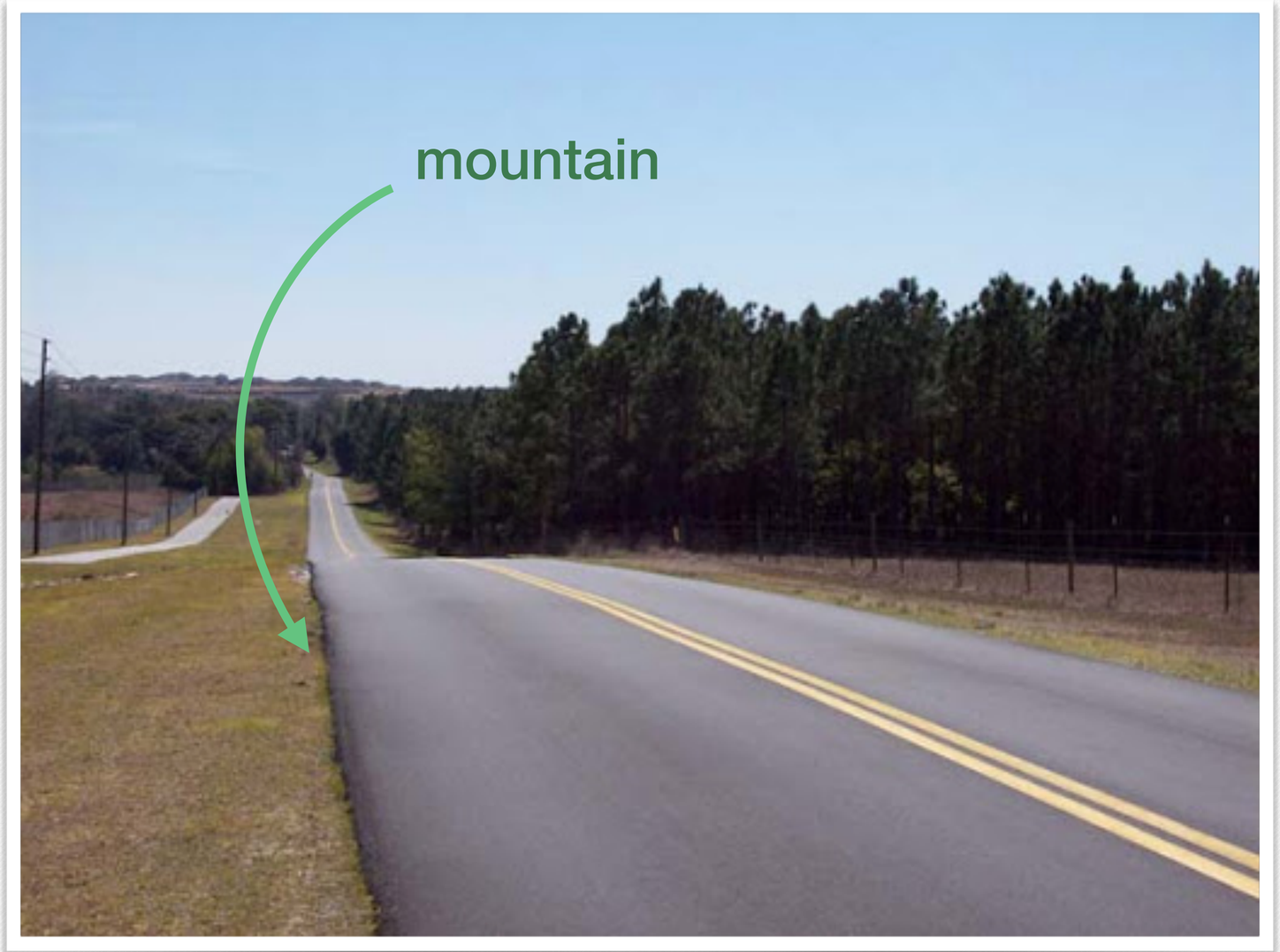
Orlando in the summer

http://www.hilton.com/top-destinations/assets/uploads/hero-images/Destinations_Header_Orlando.jpg

Orlando in the ~~summer~~ winter

http://www.hilton.com/top-destinations/assets/uploads/hero-images/Destinations_Header_Orlando.jpg

Florida mountains



http://lh3.ggpht.com/_SnE5o5tcT9U/S5LqMyRpt6I/AAAAAAAAAO4/NIGxaSf4r1s/Clermont%20Hills.jpg

Where I work

code school
COURSES
SCREENCASTS
DISCUSS
ENROLL
CREATE FREE ACCOUNT
SIGN IN
Learn By Doing
No setup. No hassle. Just learning.
Code School teaches web technologies in the comfort of your browser with video lessons, coding challenges, and screencasts.
VIEW OUR COURSES
Learn where to start!
LEARN:
GIT
BACKBONE.JS
SASS
RAILS
JQUERY
Learn where to start and what to take next with Paths

What I do there

A podcast I co-host

http://iosbytes.envylabs.com

Animation options on iOS

UIView Animations

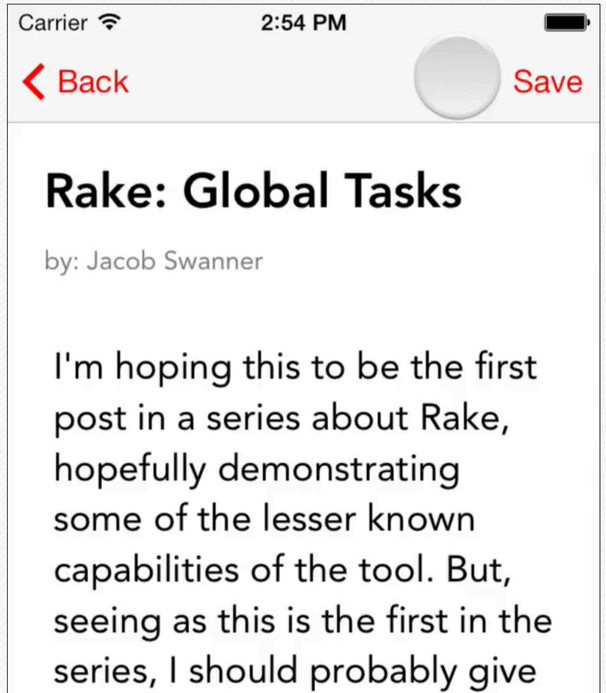
- Curve-based
- Spring
- Keyframe

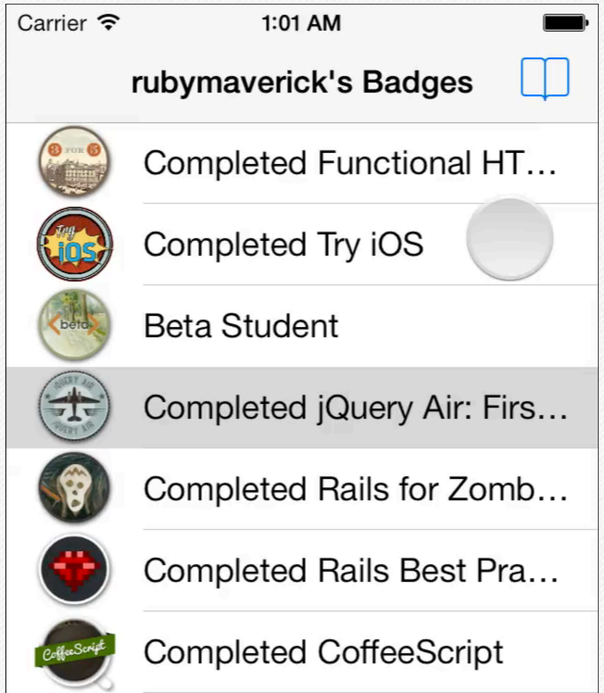
UIKit Dynamics

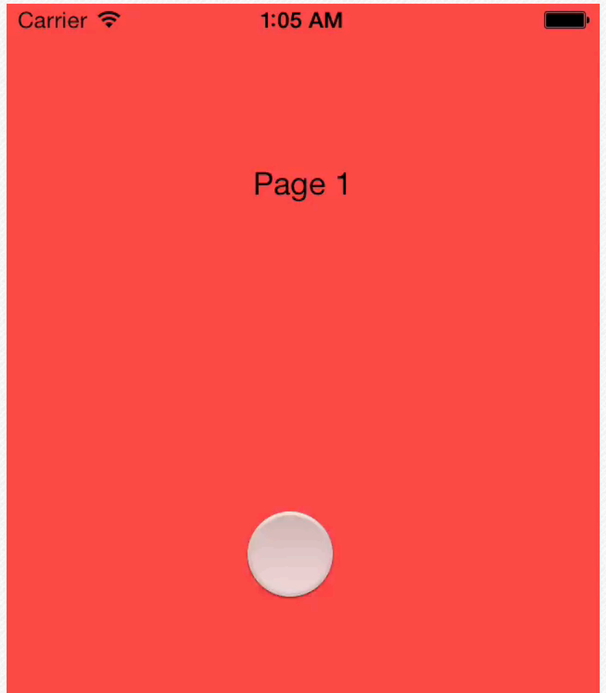
What is UIKit Dynamics?

A framework for adding real-world-like physics to view objects

Describes the types of physical behaviors a view should adopt and lets the physics system control how they animate

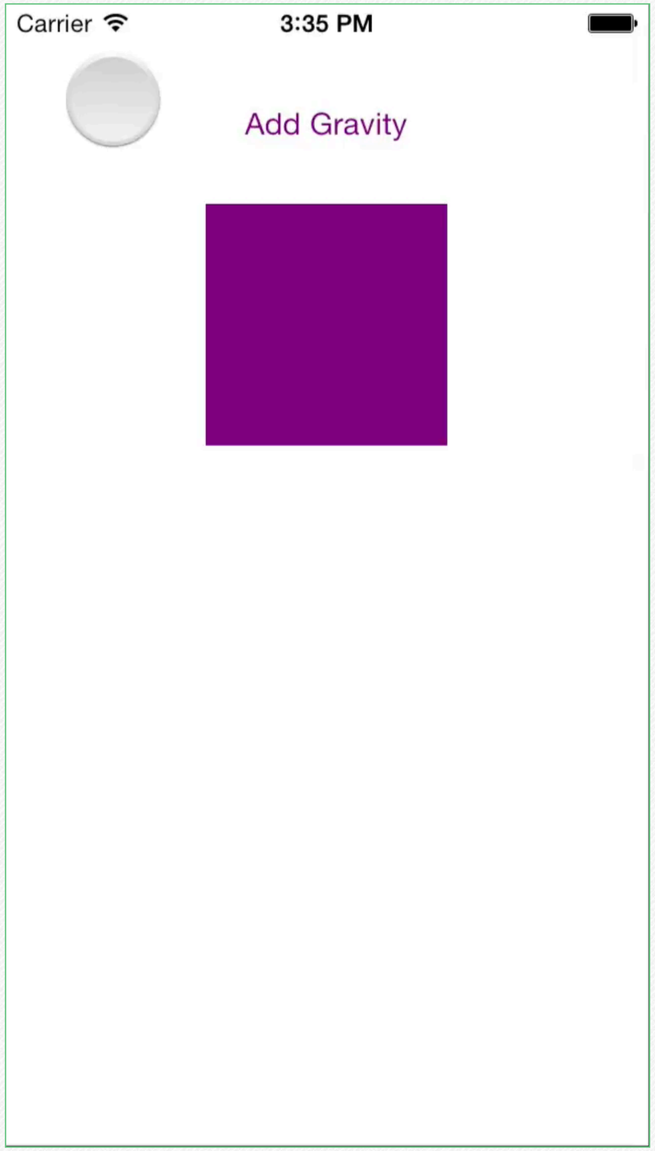






Gravity

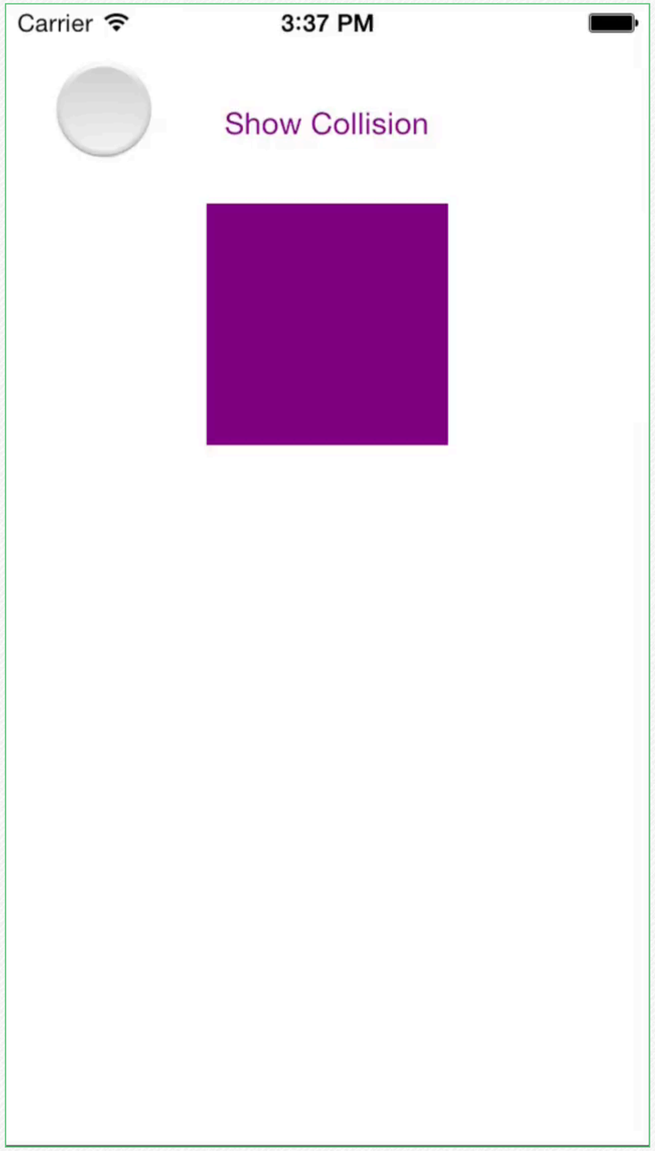
Set a gravity direction and your views will automatically animate towards it



`UIGravityBehavior`

Collisions

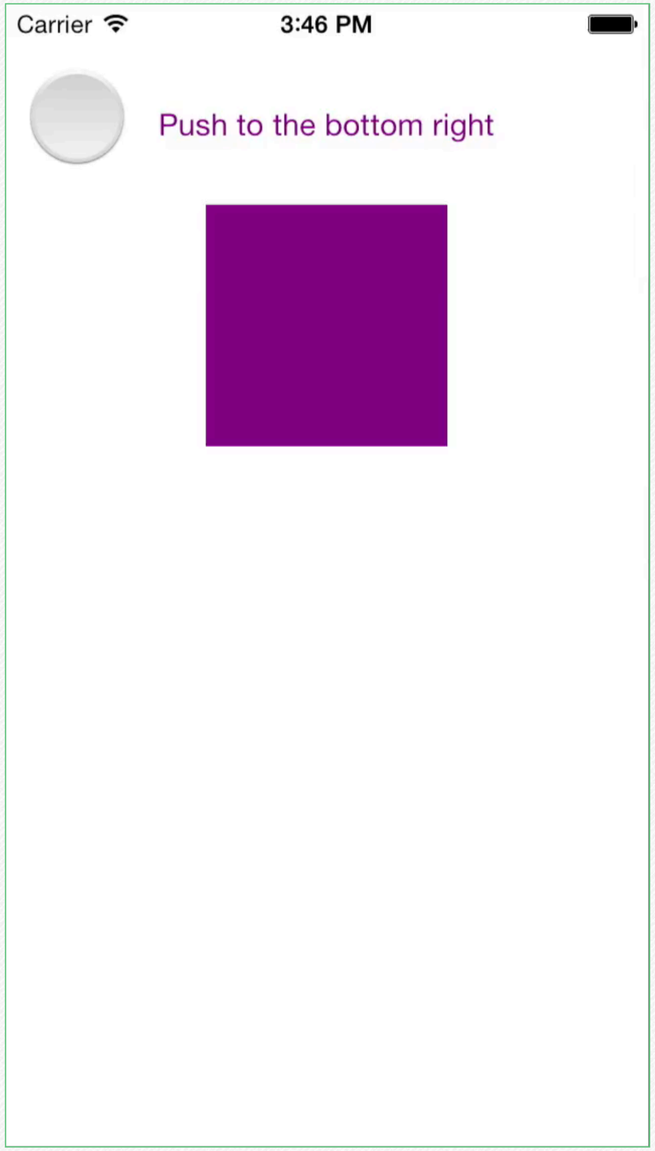
Create boundaries that stop motion once a view tries to cross them




```
UICollisionBehavior
```

Pushing

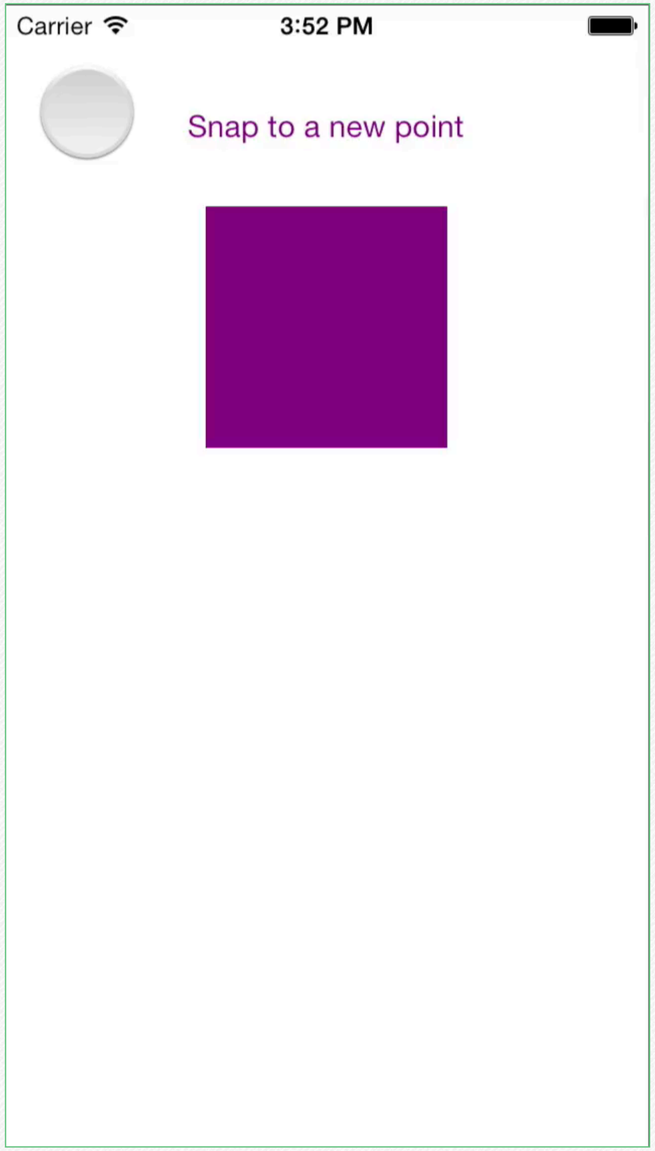
Apply a force to a view to move it in a specific direction



`UIPushBehavior`

Snapping

Set an arbitrary point where a view should move and it will snap into place



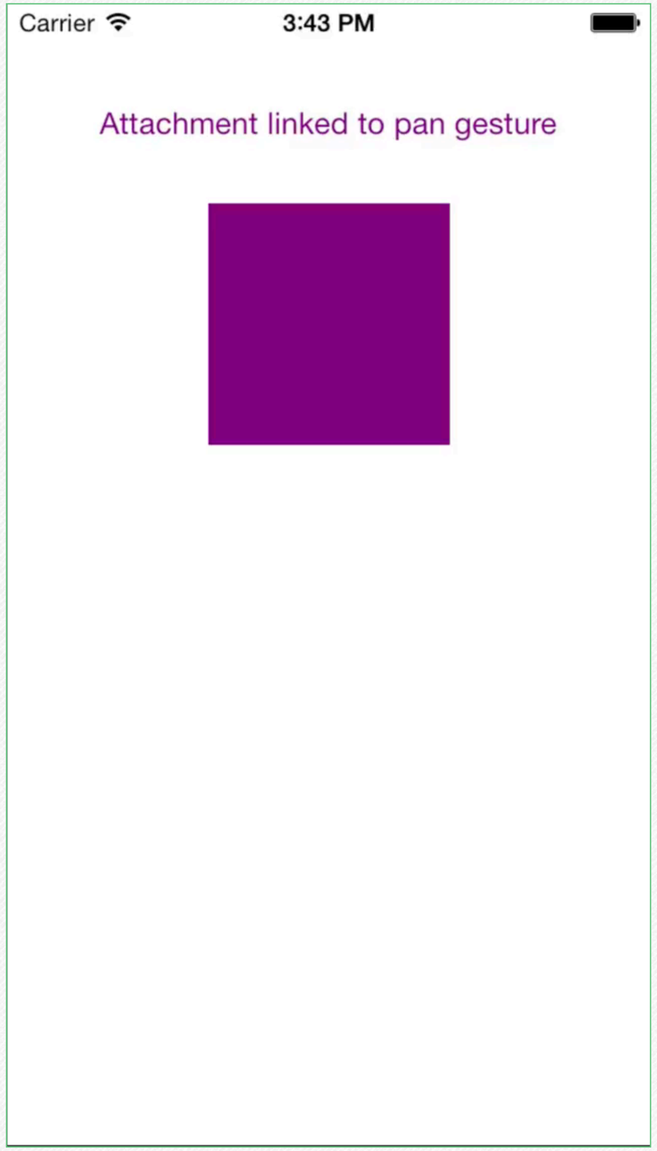
`UISnapBehavior`

Attachment

Attach one view to an arbitrary point or other view

Can be linked to the tap position during a gesture

`UIAttachmentBehavior`



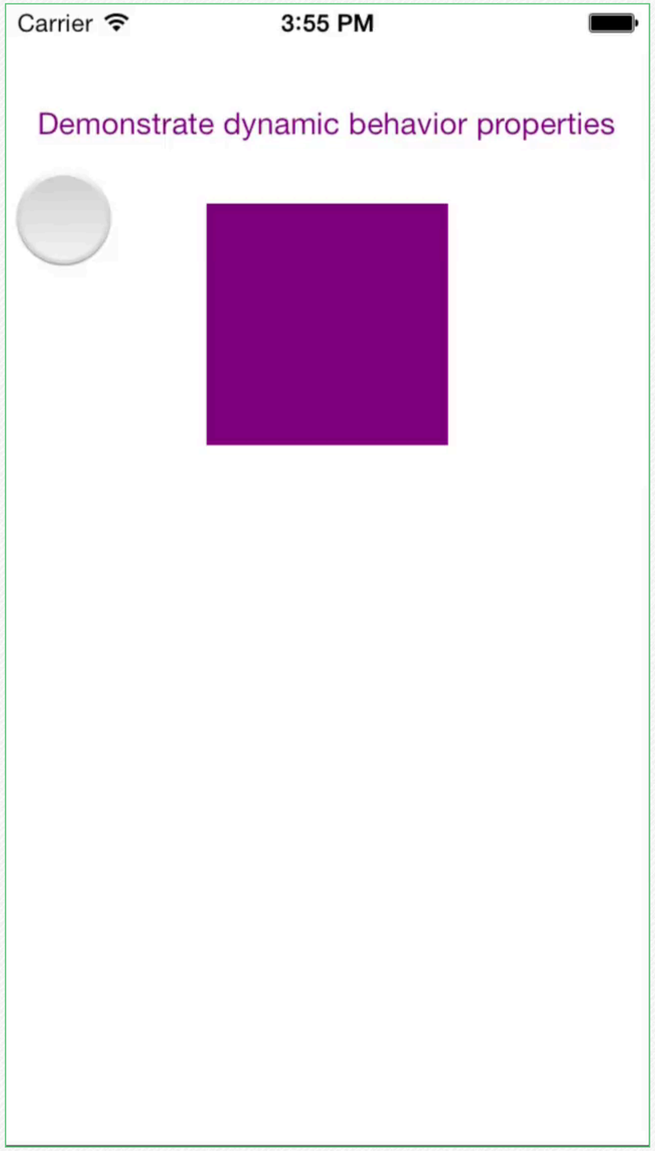
Generic Behaviors

A generic behavior class that controls some of the underlying physical properties of a view

- density
- velocity (linear and angular)
- elasticity
- friction
- resistance

`UIDynamicItemBehavior`

this example shows a gravity/collision behavior with different elasticity and an angular velocity



Start by creating a UIDynamicAnimator object

ViewController.h

```
@property (strong, nonatomic) UIDynamicAnimator *animator;
```

the animator **must** be a <u>strong</u> reference

ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
}
```

reference view == container where subviews will be animated

Adding a gravity behavior to a view

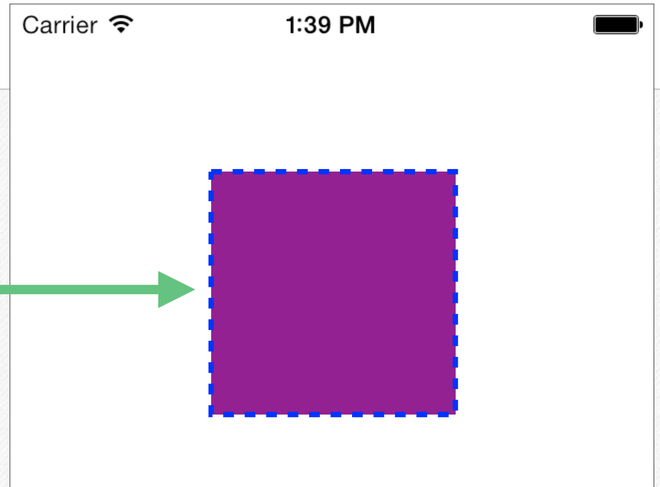
ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];

    UIGravityBehavior *gravity = [[UIGravityBehavior alloc]
                                  initWithItems:@[self.boxView]];
    [self.animator addBehavior:gravity];
}
```

the view the behavior is being applied to

self.view == the reference view

self.boxView == the dynamic item that will be moving



Setting the direction that gravity pulls

ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];

    UIGravityBehavior *gravity = [[UIGravityBehavior alloc]
                                  initWithItems:@[self.boxView]];
    gravity.gravityDirection = CGVectorMake(0, 1);
    [self.animator addBehavior:gravity];
}
```

x direction

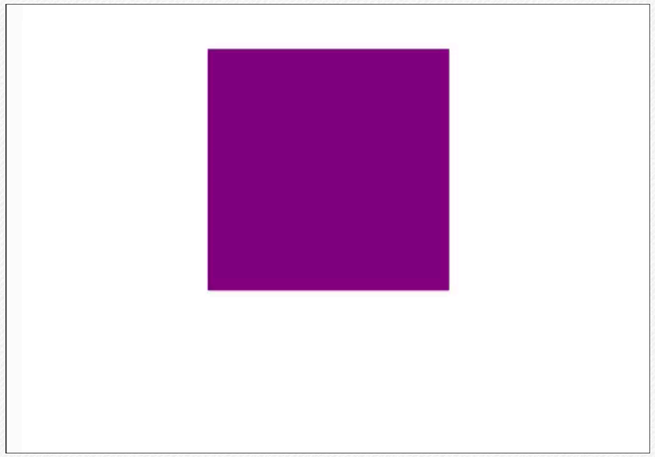
y direction

Control the gravity direction with vectors

each vector is a combination of **angle** and **magnitude**

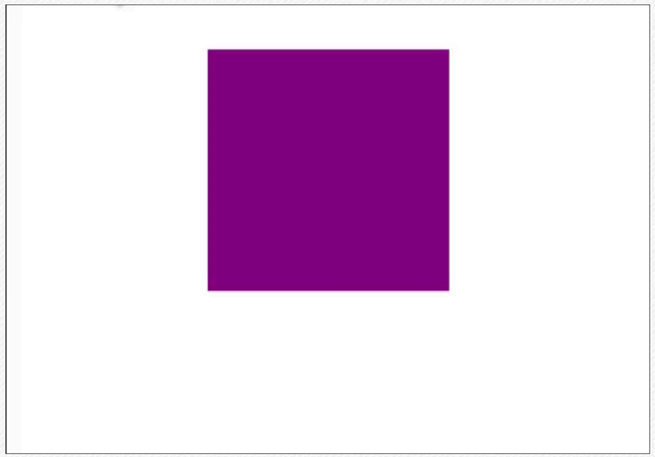
```
gravity.gravityDirection = CGVectorMake(0, 1);
```

angle: straight down
magnitude: 1 times normal rate



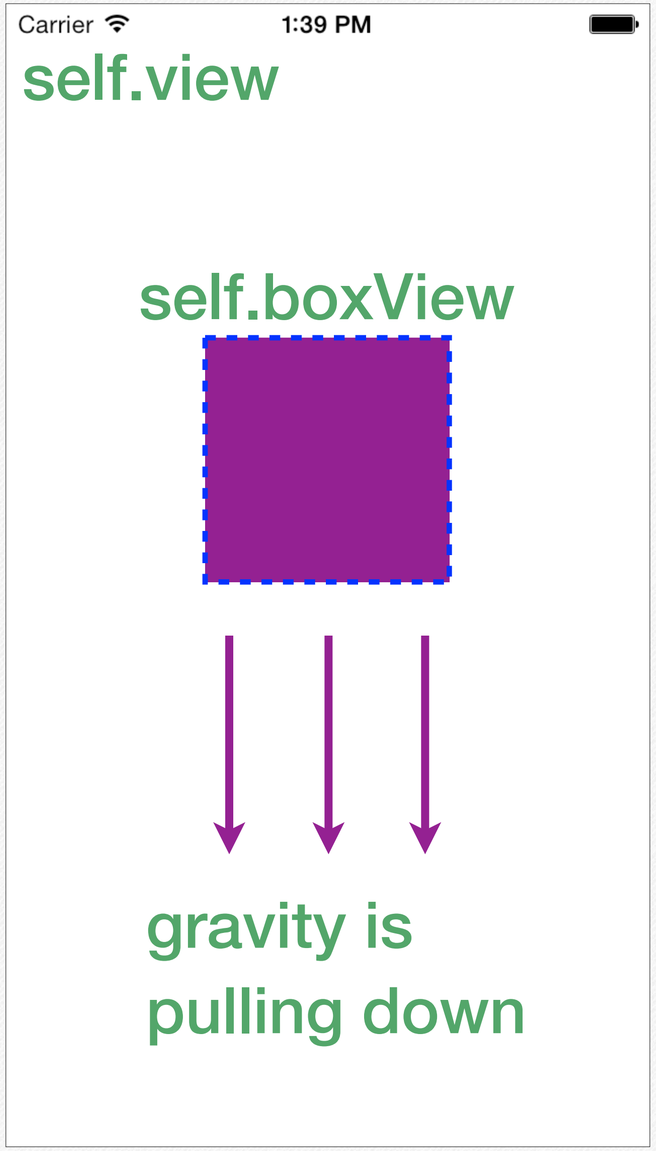
```
gravity.gravityDirection = CGVectorMake(0, 3);
```

angle: straight down
magnitude: 3 times normal rate

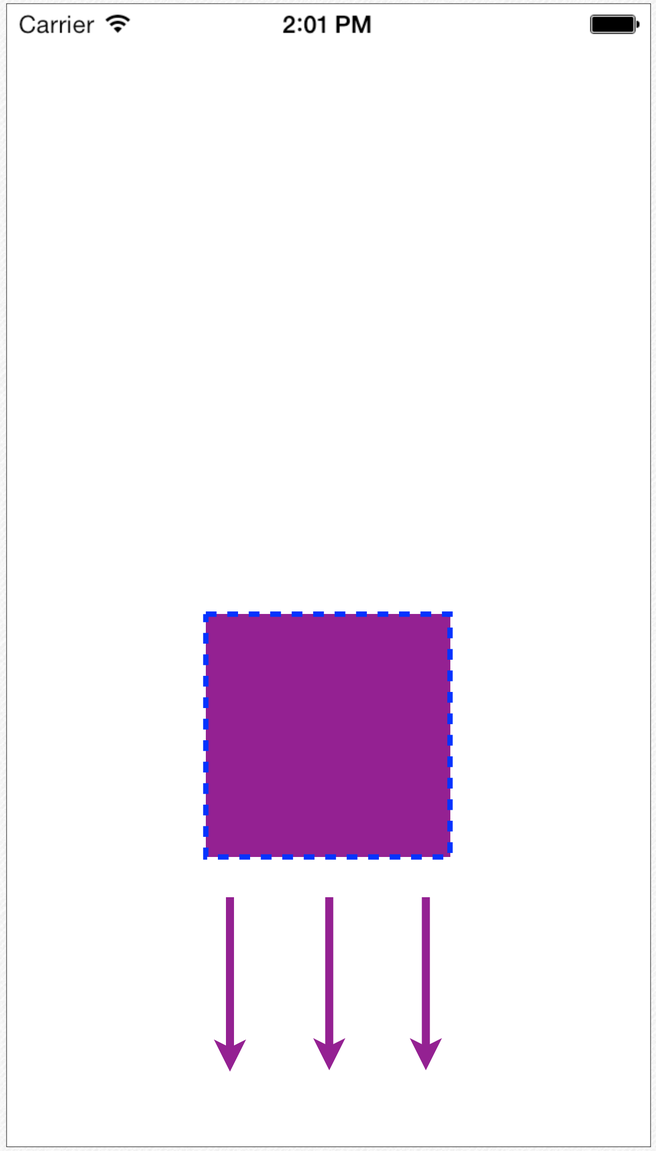


Gravity without collisions

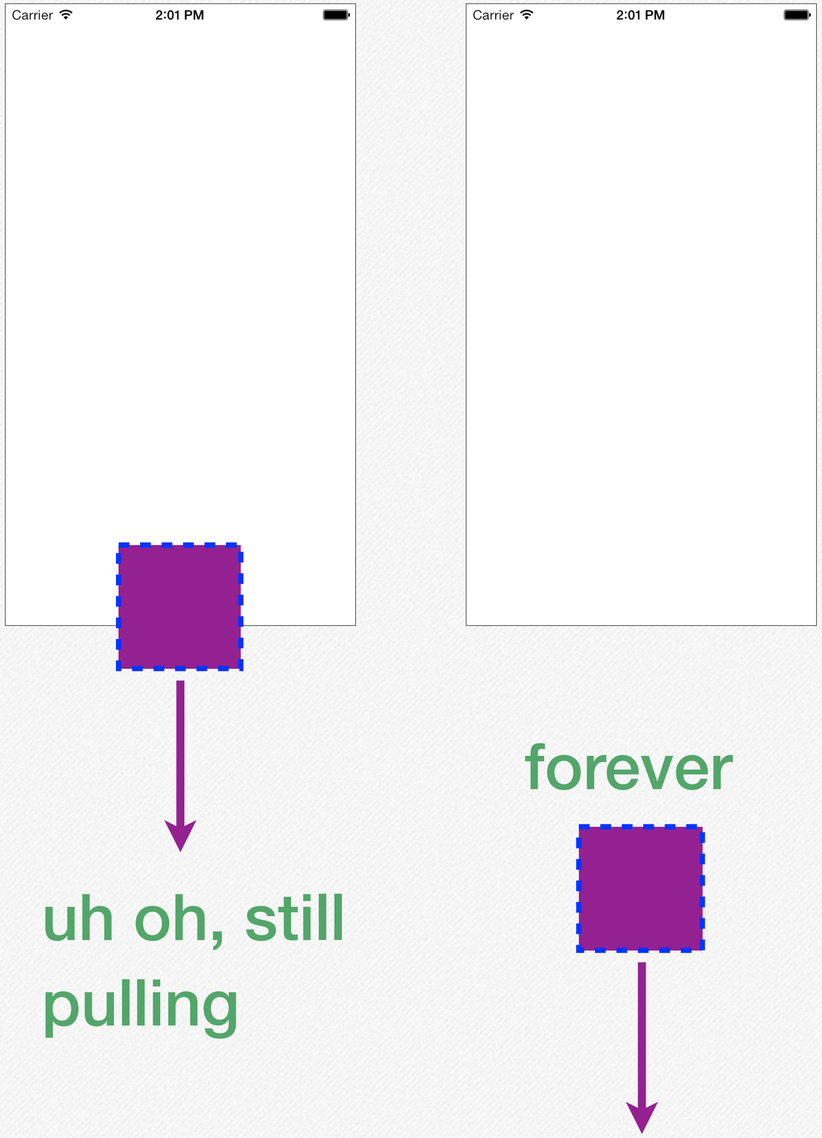
before motion



motion starts



motion continues



Setting collision boundaries

ViewController.m

```
self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
...

UICollisionBehavior *viewBoundaries = [[UICollisionBehavior alloc]
                                        initWithItems:@[self.boxView]];

[viewBoundaries addBoundaryWithIdentifier:@"bottom-of-container-boundary"
                                fromPoint:lineStartPoint
                                  toPoint:lineEndPoint];

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

Setting collision boundaries

ViewController.m

bottom-left of window

```
CGPoint lineStartPoint = CGPointMake(CGRectGetMinX(self.view.bounds),
                                     CGRectGetMaxY(self.view.bounds));
```

bottom-right of window

```
CGPoint lineEndPoint   = CGPointMake(CGRectGetMaxX(self.view.bounds),
                                     CGRectGetMaxY(self.view.bounds));
```

```
[viewBoundaries addBoundaryWithIdentifier:@"bottom-of-container-boundary"
                                fromPoint:lineStartPoint
                                  toPoint:lineEndPoint];
```

A shortcut for collision boundaries

ViewController.m

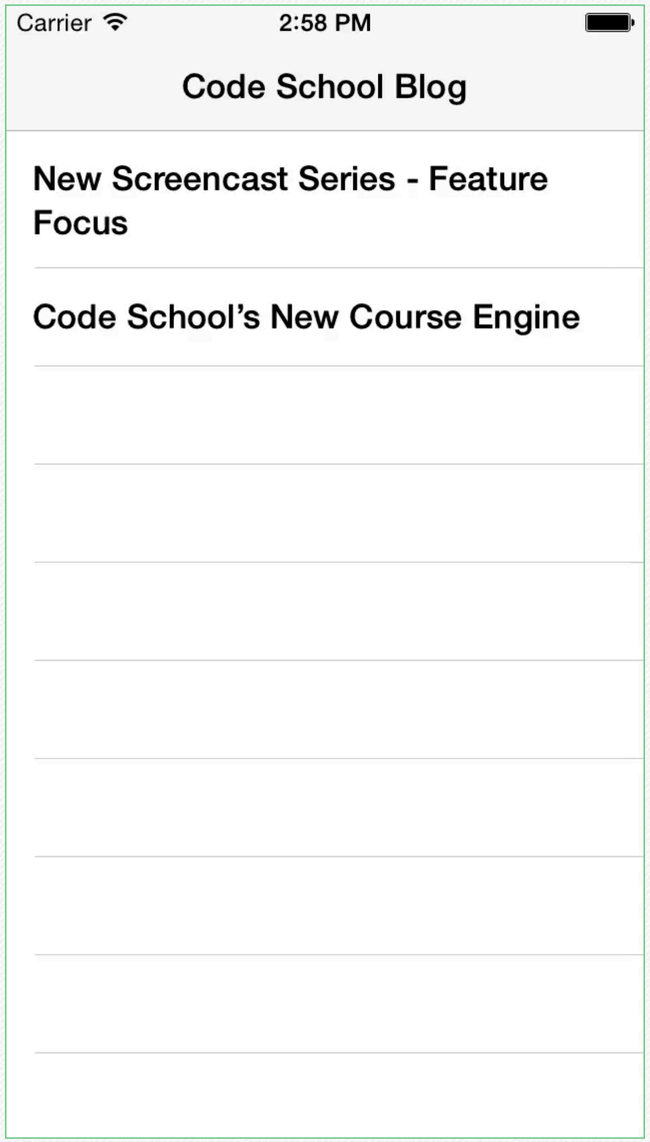
```
self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
...

UICollisionBehavior *viewBoundaries = [[UICollisionBehavior alloc]
                                        initWithItems:@[self.boxView]];

viewBoundaries.translatesReferenceBoundsIntoBoundary = YES;

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

sets four boundaries equal to
the “walls” of the container view

Let’s make some stuff!

https://github.com/jonfriskics/360iDevWorkshop

Project #1 - A custom alert view-style pop-up



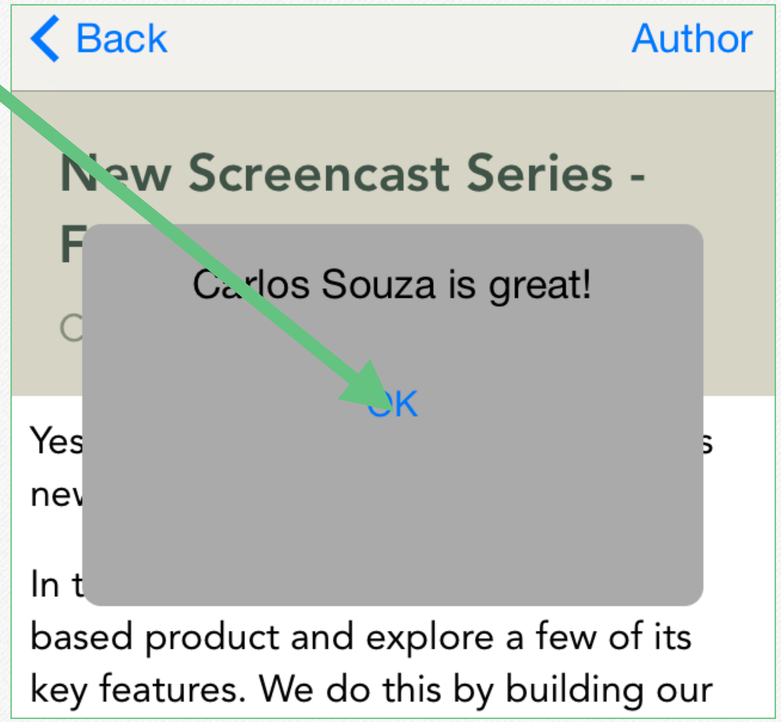
Project #1 - A custom alert view-style pop-up

1. Create a new view and add it in response to some event
2. Position the view off-screen
3. Snap the view into position
4. Drop, rotate, and fade the view out in response to some other event

UISnapBehavior snapToPoint:

The snap point is the center of where the snapped view will end up

```
self.snapBehavior = [[UISnapBehavior alloc]
                initWithItem:self.authorView
                 snapToPoint:CGPointMake(CGRectGetMidX(self.view.frame), 200)];
```



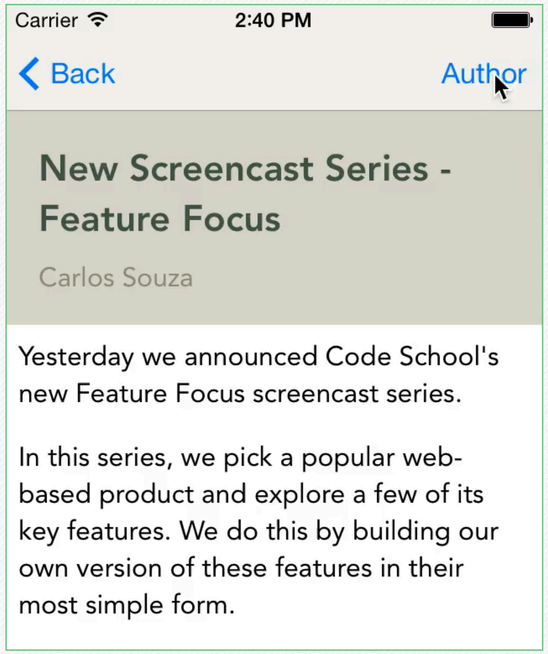
UISnapBehavior Damping

Damping helps allow more or less oscillation while snapping

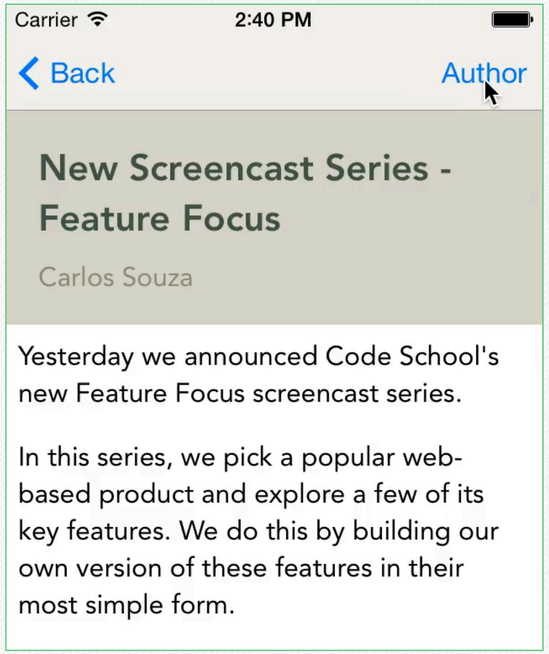
the animation doesn't look as **bouncy**

```
snapBehavior.damping = 1.0;
```

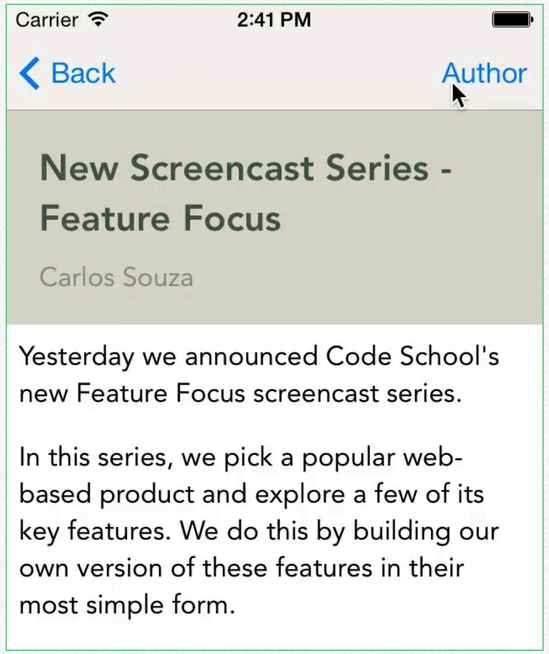
0.1



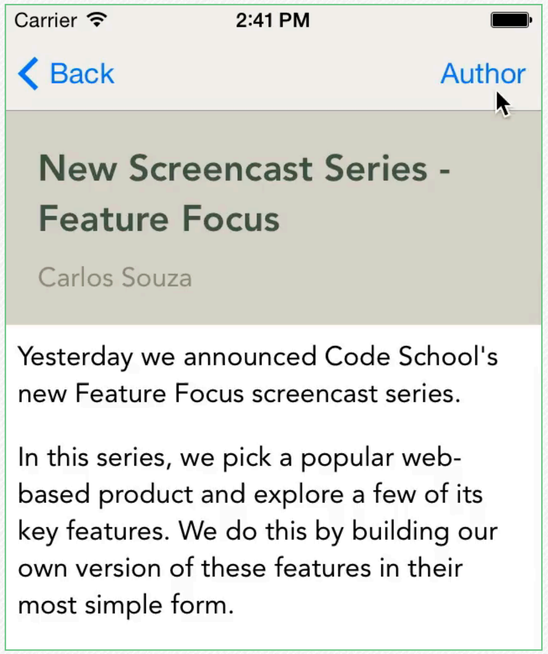
0.8



1.3



1.9



Block interaction until view is in place

If you don’t block interaction, the button on the custom alert view can be hit while it is animating in

The Plan

1. Disable user interaction on the author view when it is first created
2. Detect when the animation stops and the view is in place
3. Enable user interaction

Finding out when animation stops

UIDynamicAnimatorDelegate has a method for this

```
- (void)viewDidLoad
{
    self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    self.animator.delegate = self;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    [self.animator removeBehavior:self.snapBehavior];
}
```

fires when the snap animation is finished

the snap behavior needs to be accessible from the delegate method

Combining Gravity and Dynamic Item Behaviors

the item affected by gravity will be the author view

CSStoryViewController.m

```
- (void)hideAuthorView:(id)sender
{
  self.gravityBehavior = [[UIGravityBehavior alloc] initWithItems:@[self.authorView]];
  self.gravityBehavior.gravityDirection = CGVectorMake(0, 7);
  [self.animator addBehavior:self.gravityBehavior];
}
```

gravity will pull the targeted item down

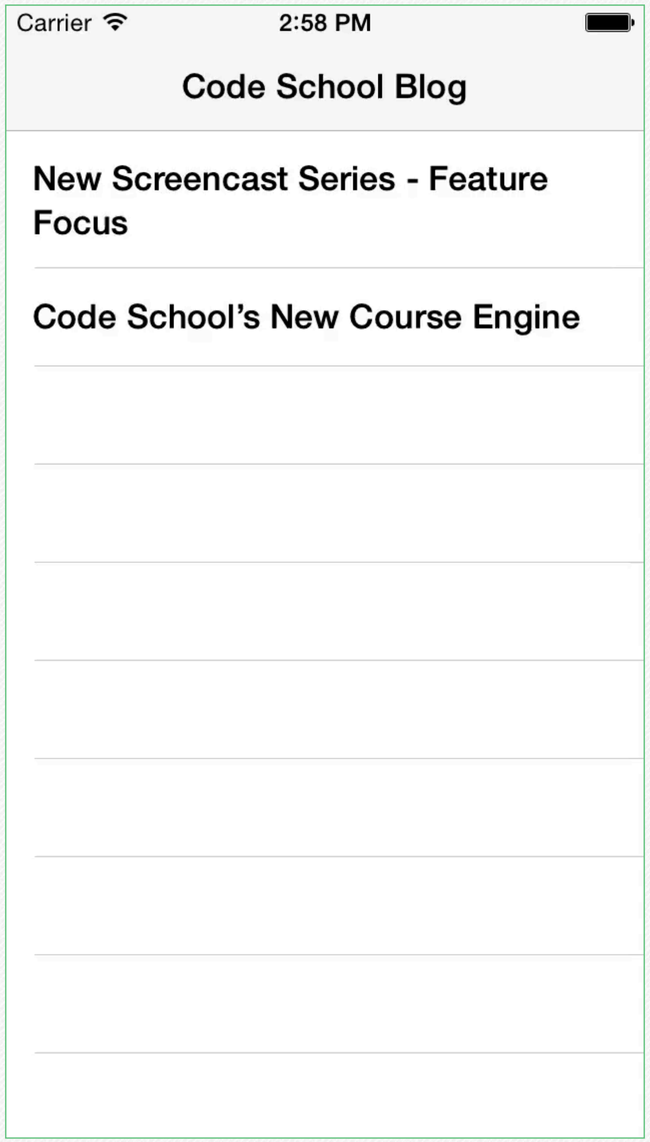
Combining Gravity and Dynamic Item Behaviors

Allow rotation and add an angular velocity
to tilt the view at an angle as it falls

CSStoryViewController.m

```
- (void)hideAuthorView:(id)sender
{
  ...
  [self.animator addBehavior:self.gravityBehavior];

  self.dynamicItemBehavior = [[UIDynamicItemBehavior alloc] initWithItems:@[self.authorView]];
  self.dynamicItemBehavior.allowsRotation = YES;
  [self.dynamicItemBehavior addAngularVelocity:(M_PI / 2)
                                       forItem:self.authorView];
  [self.animator addBehavior:self.dynamicItemBehavior];
}
```

Demo of gravity and rotation during dismiss



Still one problem…

Removing the view and behavior

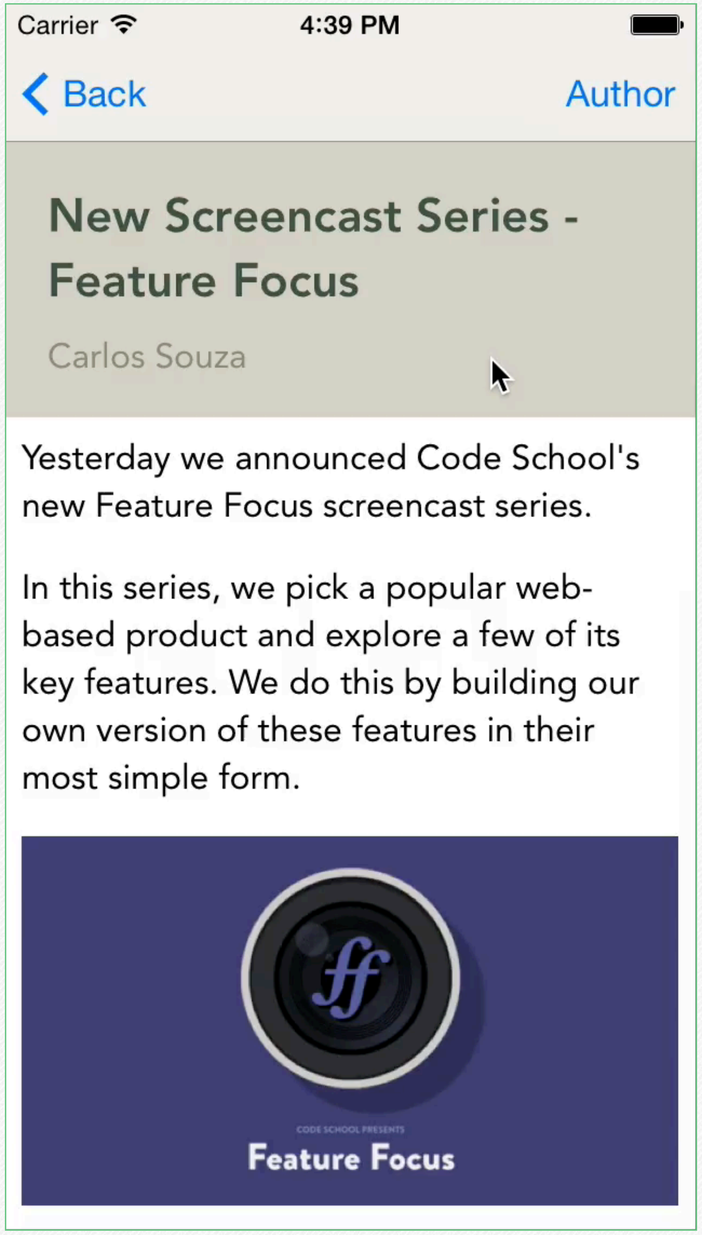
CSStoryViewController.m

```
- (void)hideAuthorView:(id)sender
{
  ...
    __weak CSStoryViewController *weakSelf = self;
    self.gravityBehavior.action = ^{
        if(!CGRectIntersectsRect(weakSelf.view.frame, weakSelf.authorView.frame)) {
            [weakSelf.authorView removeFromSuperview];
            [weakSelf.animator removeBehavior:weakSelf.gravityBehavior];
            [weakSelf.animator removeBehavior:weakSelf.dynamicItemBehavior];
        }
    };
}
```

this will be true as long as authorView is partially onscreen

we want to check when this is **NOT** true, so we can remove the view behaviors

Project #2 - A custom menu



Project #2 - A custom menu

1. Create a new view that will hold a few menu option buttons in response to some event
2. Position the view off-screen
3. Fade in the view, and snap in the menu option buttons
4. When dismissing the menu view, push the menu option buttons away
5. As soon as the menu buttons have moved out of frame, dismiss the entire menu view

Pushing the menu options away when dismissing

CSStoryViewController.m

```
- (void)removeMenu:(id)sender
{
    [self.animator removeAllBehaviors];

    UIPushBehavior *pushBehavior = [[UIPushBehavior alloc]
                                    initWithItems:@[self.mailButton] mode:UIPushBehaviorModeContinuous];
    pushBehavior.pushDirection = CGVectorMake(-9,0);

    [self.animator addBehavior:pushBehavior];
}
```

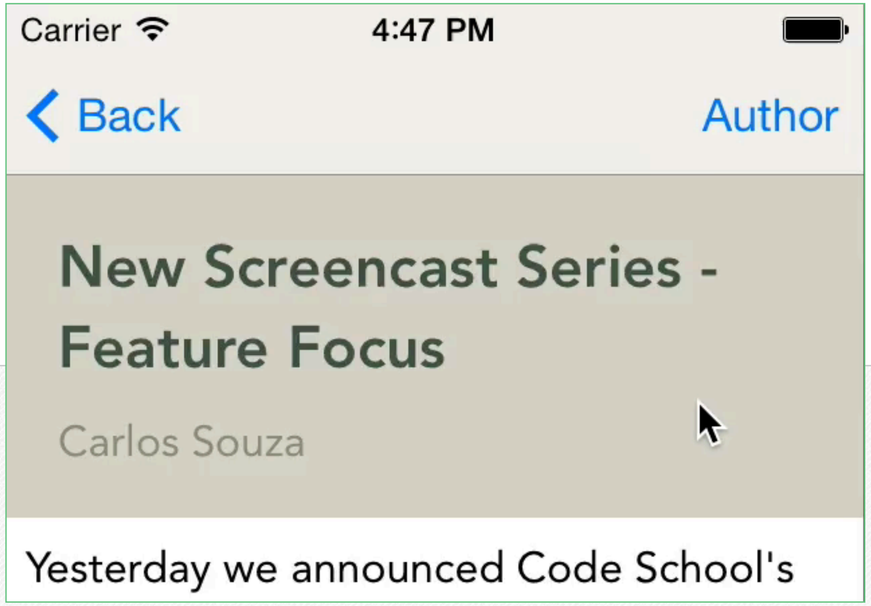
pushing to the left

Removing the push behavior after it has started

CSStoryViewController.m

```
- (void)removeMenu:(id)sender
{

    ...
    [self.animator addBehavior:pushBehavior];

    __weak CSStoryViewController *weakSelf = self;
    pushBehavior.action = ^{
        if(!CGRectIntersectsRect(weakSelf.overlayView.frame, weakSelf.mailButton.frame)) {
            [weakSelf.animator removeAllBehaviors];
            [weakSelf.overlayView removeFromSuperview];
            weakSelf.overlayView = nil;
        }
    };
}
```

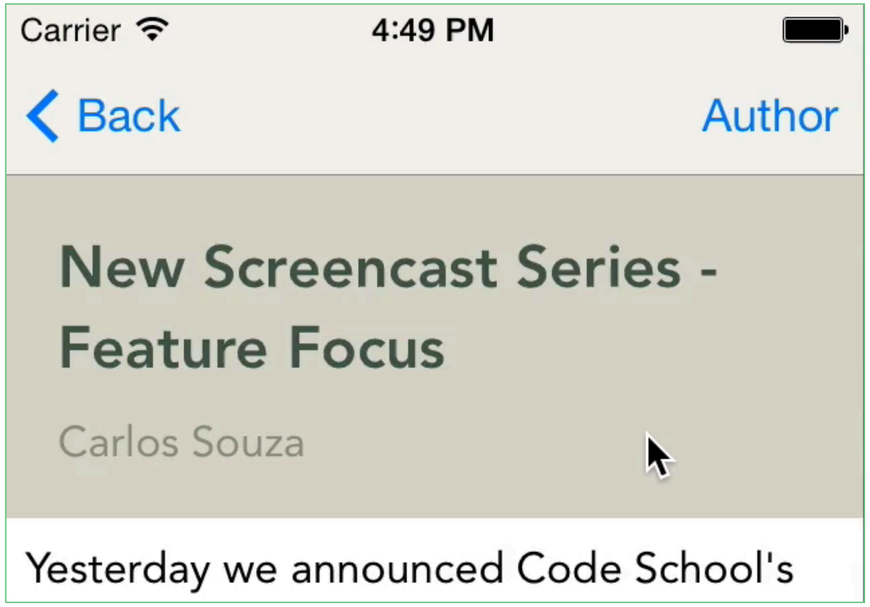
why not animate the alpha here?



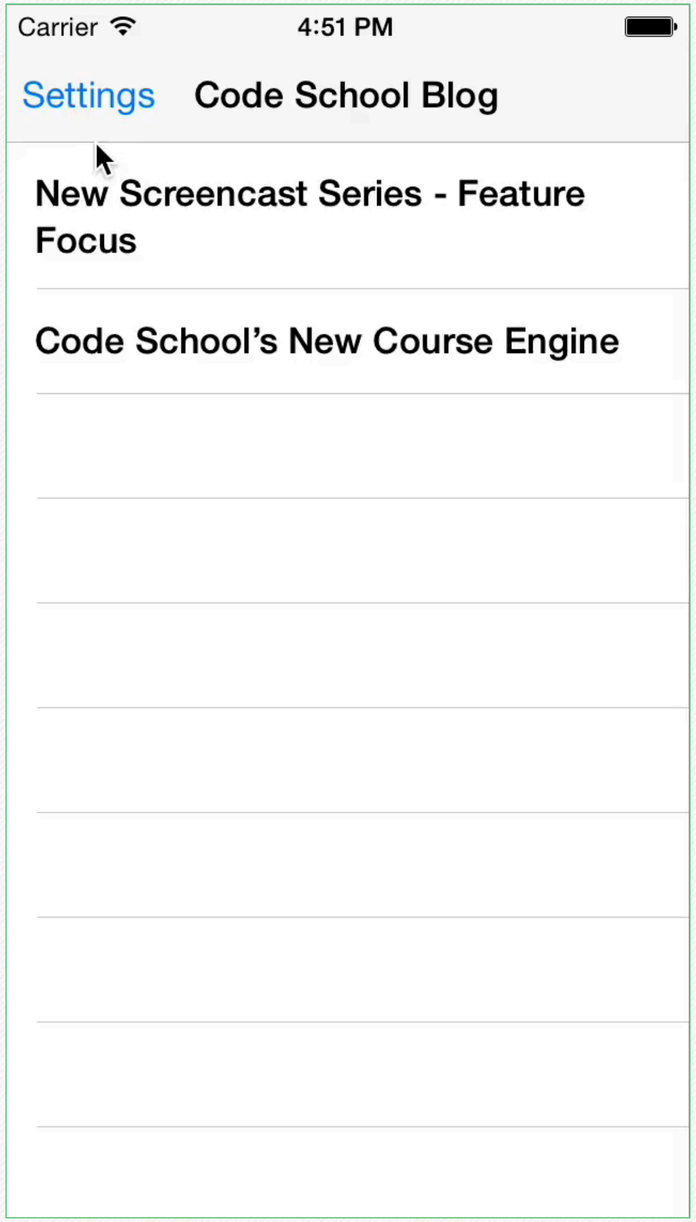
Combining UIView animations and UIKit Dynamics

CSStoryViewController.m

```
- (void)removeMenu:(id)sender
{

    ...
    [self.animator addBehavior:pushBehavior];

    __weak CSStoryViewController *weakSelf = self;
    pushBehavior.action = ^{
        if(!CGRectIntersectsRect(weakSelf.overlayView.frame, weakSelf.mailButton.frame)) {
            [weakSelf.animator removeAllBehaviors];
            [UIView animateWithDuration:0.3 animations:^{
                weakSelf.overlayView.alpha = 0.0;
            } completion:^(BOOL finished) {
                [weakSelf.overlayView removeFromSuperview];
                weakSelf.overlayView = nil;
            }];
        }
    };
}
```



Project #3 - A weighted settings screen



Preparing for an animated transition

CSStoriesTableViewController.m

```
- (void)settingsTapped:(id)sender
{
    CSSettingsViewController *settingsVC = [[CSSettingsViewController alloc] init];
    settingsVC.delegate = self;
    settingsVC.modalPresentationStyle = UIModalPresentationCustom;
    settingsVC.transitioningDelegate = self;

    [self presentViewController:settingsVC animated:YES completion:^{
    }];
}
```

this means we also need to adopt UIViewControllerTransitioningDelegate

Preparing for an animated transition

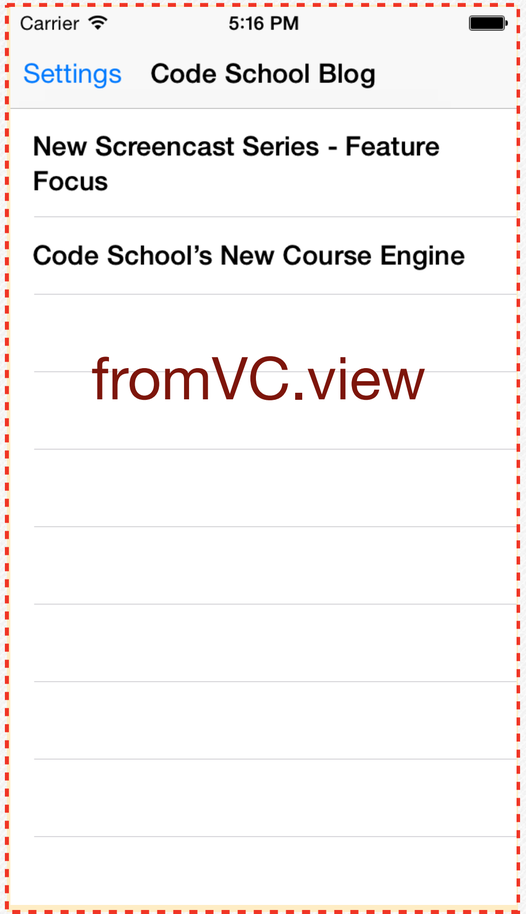
CSStoriesTableViewController.m

```
#pragma mark - Transition delegate methods

- (id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedController:
(UIViewController *)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return [[CSPresentSettingsAnimationController alloc] init];
}

- (id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedController:
(UIViewController *)dismissed
{
    return [[CSDismissSettingsAnimatorController alloc] init];
}
```

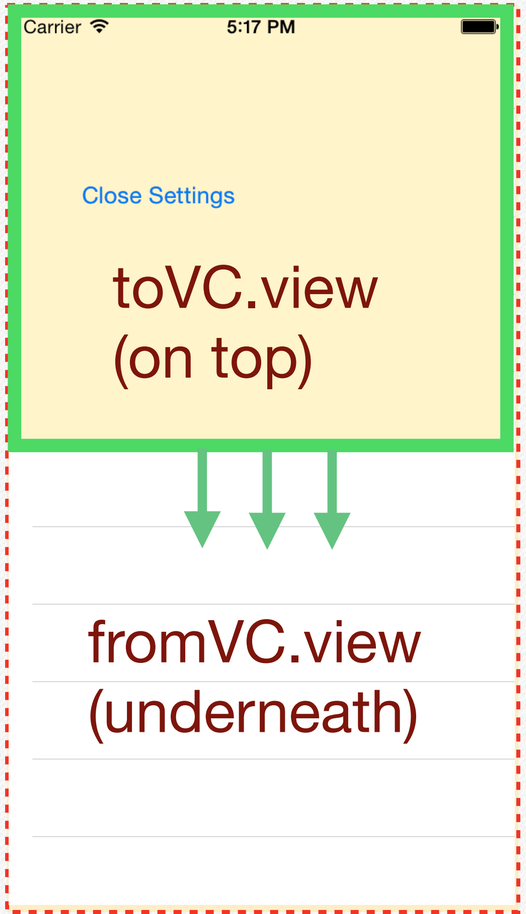
Visualizing the transition

add gravity behavior to start moving in the toVC.view

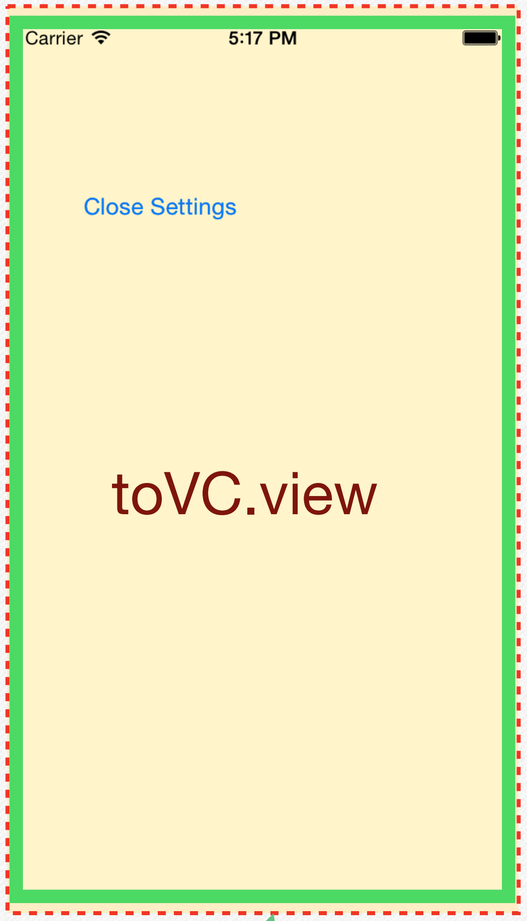
containerView



containerView



containerView



add a collision boundary on the bottom edge so toVC.view doesn't continue forever

Setting the start frame

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    self.transitionContext = transitionContext;

    UIViewController *fromVC = [transitionContext
                                viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
                                viewControllerForKey:UITransitionContextToViewControllerKey];

    CGRect toVCStartFrame = CGRectMake(0, -CGRectGetHeight(fromVCStartFrame),
                                       CGRectGetWidth(fromVCStartFrame),
                                       CGRectGetHeight(fromVCStartFrame));
    toVC.view.frame = toVCStartFrame;
    [[transitionContext containerView] addSubview:toVC.view];

    [self.transitionContext completeTransition:YES];

}
```

start the toVC with a frame outside of the visible window

Adding the animator to the present animation controller

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    ...
    [[transitionContext containerView] addSubview:toVC.view];

    self.animator = [[UIDynamicAnimator alloc]
                     initWithReferenceView:[transitionContext containerView]];
    self.animator.delegate = self;

    [self.transitionContext completeTransition:YES];

}
```

the reference view is the container view

NOT self.view

Adding gravity behavior to the presenting animation

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

    [[transitionContext containerView] addSubview:toVC.view];

    self.animator = [[UIDynamicAnimator alloc]
                     initWithReferenceView:[transitionContext containerView]];
    self.animator.delegate = self;

    UIGravityBehavior *gravity = [[UIGravityBehavior alloc] initWithItems:@[toVC.view]];
    gravity.gravityDirection = CGVectorMake(0, 3);
    [self.animator addBehavior:gravity];

    [self.transitionContext completeTransition:YES];

}
```

gravity will pull the toVC downward

Adding a collision behavior to the presenting animation

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

    [[transitionContext containerView] addSubview:toVC.view];

    self.animator = [[UIDynamicAnimator alloc]
                     initWithReferenceView:[transitionContext containerView]];
    self.animator.delegate = self;

    UIGravityBehavior *gravity = [[UIGravityBehavior alloc] initWithItems:@[toVC.view]];
    gravity.gravityDirection = CGVectorMake(0, 3);
    [self.animator addBehavior:gravity];

    UICollisionBehavior *collision = [[UICollisionBehavior alloc]
                                      initWithItems:@[toVC.view]];

    [self.transitionContext completeTransition:YES];

}
```

the toVC will collide with something to stop falling

Setting a collision boundary

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    ...
    UICollisionBehavior *collision = [[UICollisionBehavior alloc]
                                      initWithItems:@[toVC.view]];

    [collision addBoundaryWithIdentifier:@"bottomEdge"
                                 fromPoint:CGPointMake(0,
                                                       CGRectGetMaxY(fromVC.view.frame)
                                                       )
                                   toPoint:CGPointMake(CGRectGetMaxX(fromVC.view.frame),
                                                       CGRectGetMaxY(fromVC.view.frame)
                                                       )];

    [self.animator addBehavior:collision];

    [self.transitionContext completeTransition:YES];

}
```

the bottom left of fromVC

the bottom right of fromVC

Complete the transition after the dynamic animator stops

CSPresentSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    ...
    [self.animator addBehavior:collision];

    [self.transitionContext completeTransition:YES];

}




- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    [self.transitionContext completeTransition:YES];
}
```

don’t complete the transition until the animator stops

Dismissing the Settings view

CSDismissSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];

    self.animator = [[UIDynamicAnimator alloc]
                         initWithReferenceView:[transitionContext containerView]];

    UIGravityBehavior *gravity = [[UIGravityBehavior alloc]
                                  initWithItems:@[fromVC.view]];
    gravity.gravityDirection = CGVectorMake(0, -12);
    [self.animator addBehavior:gravity];
}
```

the settings view is the fromVC during dismissal

negative y-direction gravity means the settings view will “fall up”

Canceling the gravity once the view disappears

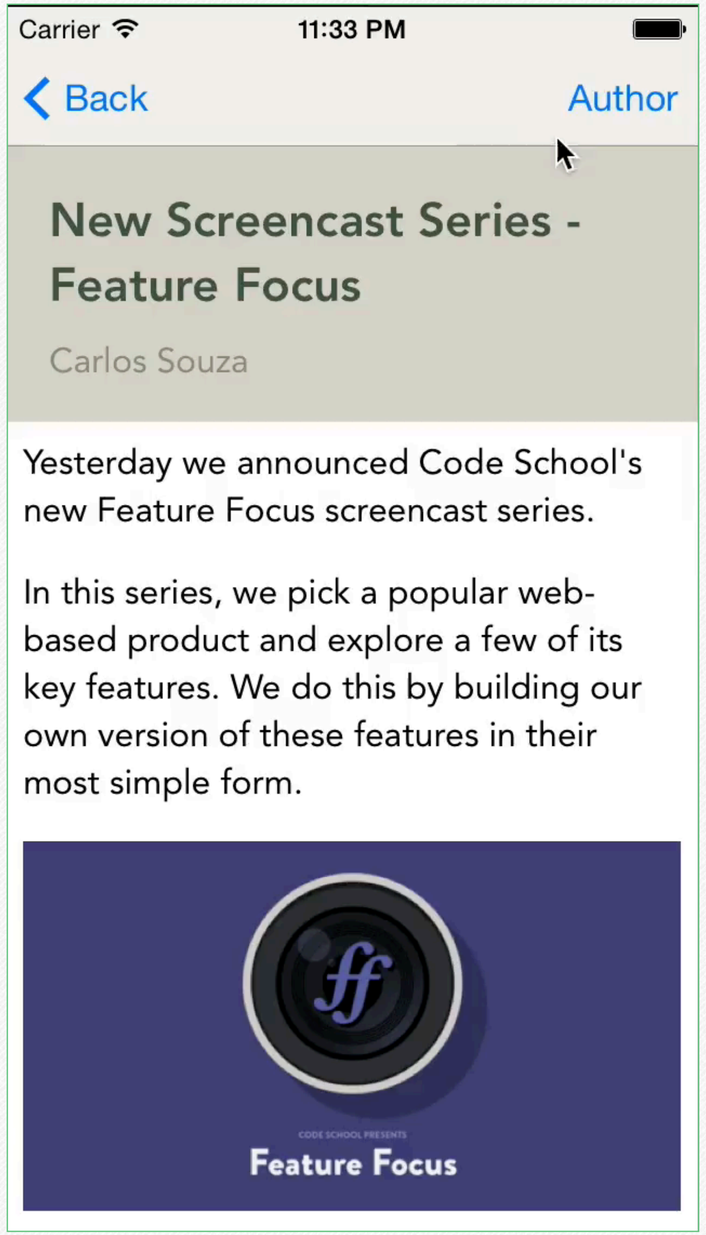
CSDismissSettingsAnimationController.m

```
- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    ...
    [self.animator addBehavior:gravity];

    gravity.action = ^{
        if(!CGRectIntersectsRect(fromVC.view.frame,
                                 [[transitionContext containerView] frame])) {
            [self.animator removeAllBehaviors];
            [transitionContext completeTransition:YES];
        }
    };
```

don't forget to complete the transition!

Since there's no collision to stop this gravity, remove it when the fromVC is not visible anymore

Project #4 - Drag



Adding the pan gesture after the author view appears

CSStoryViewController.m

```
- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    [self.animator removeBehavior:self.snapBehavior];

    UIPanGestureRecognizer *gesture = [[UIPanGestureRecognizer alloc]
                                       initWithTarget:self
                                               action:@selector(handlePan:)];
    [self.authorView addGestureRecognizer:gesture];
}
```

Getting the handler ready

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {

    } else if (gesture.state == UIGestureRecognizerStateChanged) {

    } else if (gesture.state == UIGestureRecognizerStateEnded) {

    }
}
```

Getting the handler ready

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {

        get the point where the authorView was touched

        calculate the offset from the center of the authorView

        calculate the anchor point (where touched in self.view)

        add behaviors

    }
}
```

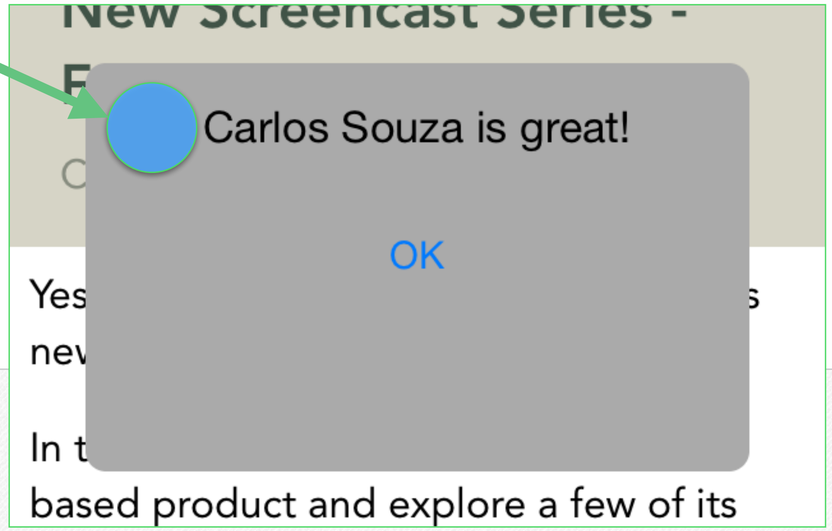
Get the point where the authorView was touched

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {
        CGPoint authorViewTouchPoint = [gesture locationInView:self.authorView];


    }
}
```

authorViewTouchPoint

relative to the top left of authorView



Calculate the offset from the center of the authorView

CSStoryViewController.m

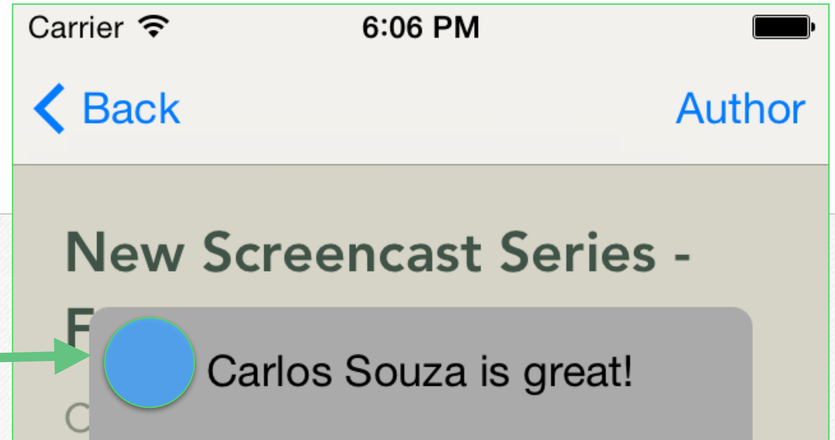
```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {
        CGPoint authorViewTouchPoint = [gesture locationInView:self.authorView];

        CGFloat halfTheWidth = CGRectGetWidth(self.authorView.frame) / 2.0;
        CGFloat halfTheHeight = CGRectGetHeight(self.authorView.frame) / 2.0;

        UIOffset offsetFromCenter = UIOffsetMake(authorViewTouchPoint.x - halfTheWidth,
                                                 authorViewTouchPoint.y - halfTheHeight);

    }
}
```

Calculate the anchor point

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {
        CGPoint authorViewTouchPoint = [gesture locationInView:self.authorView];

        CGFloat halfTheWidth = CGRectGetWidth(self.authorView.frame) / 2.0;
        CGFloat halfTheHeight = CGRectGetHeight(self.authorView.frame) / 2.0;

        UIOffset offsetFromCenter = UIOffsetMake(authorViewTouchPoint.x - halfTheWidth,
                                                 authorViewTouchPoint.y - halfTheHeight);

        CGPoint anchorInSuperview = [gesture locationInView:self.view];

    }
}
```

relative to the top left of self.view

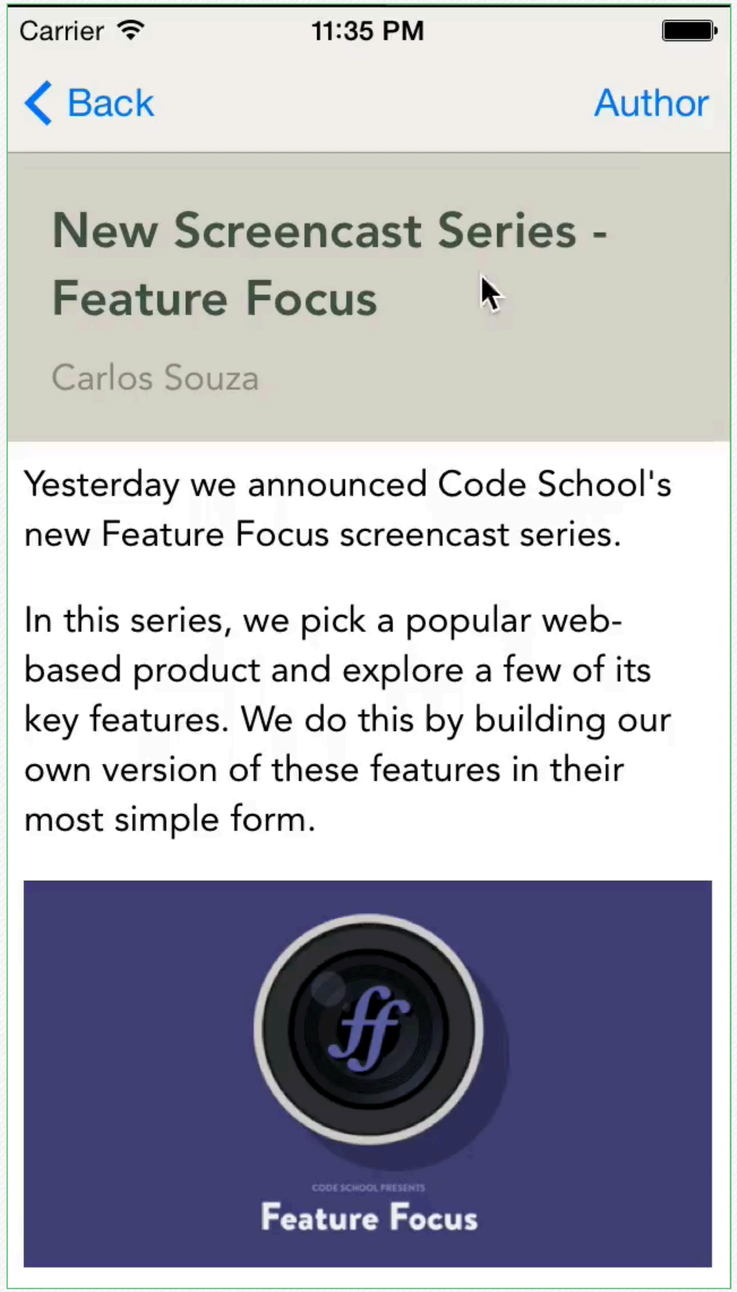


Add behaviors

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateBegan) {
        ...
        self.attachmentBehavior = [[UIAttachmentBehavior alloc]
                                       initWithItem:self.authorView
                                  offsetFromCenter:touchPointOffsetFromCenter
                                  attachedToAnchor:anchorInSuperview];

        [self.animator addBehavior:self.attachmentBehavior];

        // and gravity behavior

        // and dynamic item behavior
    }
}
```

Demo video

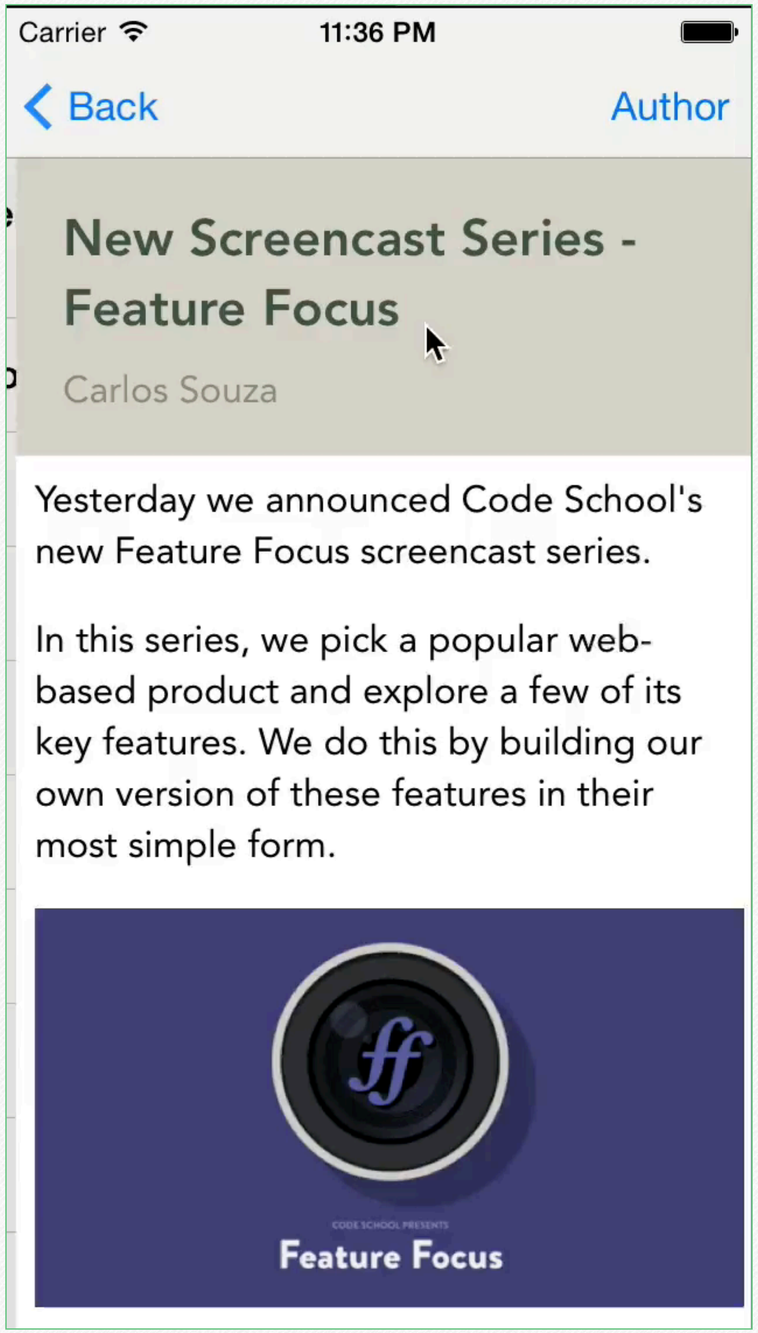

Carrier
11:35 PM
Back
Author
New Screencast Series - Feature Focus
Carlos Souza
Yesterday we announced Code School's new Feature Focus screencast series.
In this series, we pick a popular web-based product and explore a few of its key features. We do this by building our own version of these features in their most simple form.
Feature Focus

Adjust the anchor point

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateChanged) {
        CGPoint newAnchor = [gesture locationInView:self.view];
        self.attachmentBehavior.anchorPoint = newAnchor;
    }
}
```

Demo video

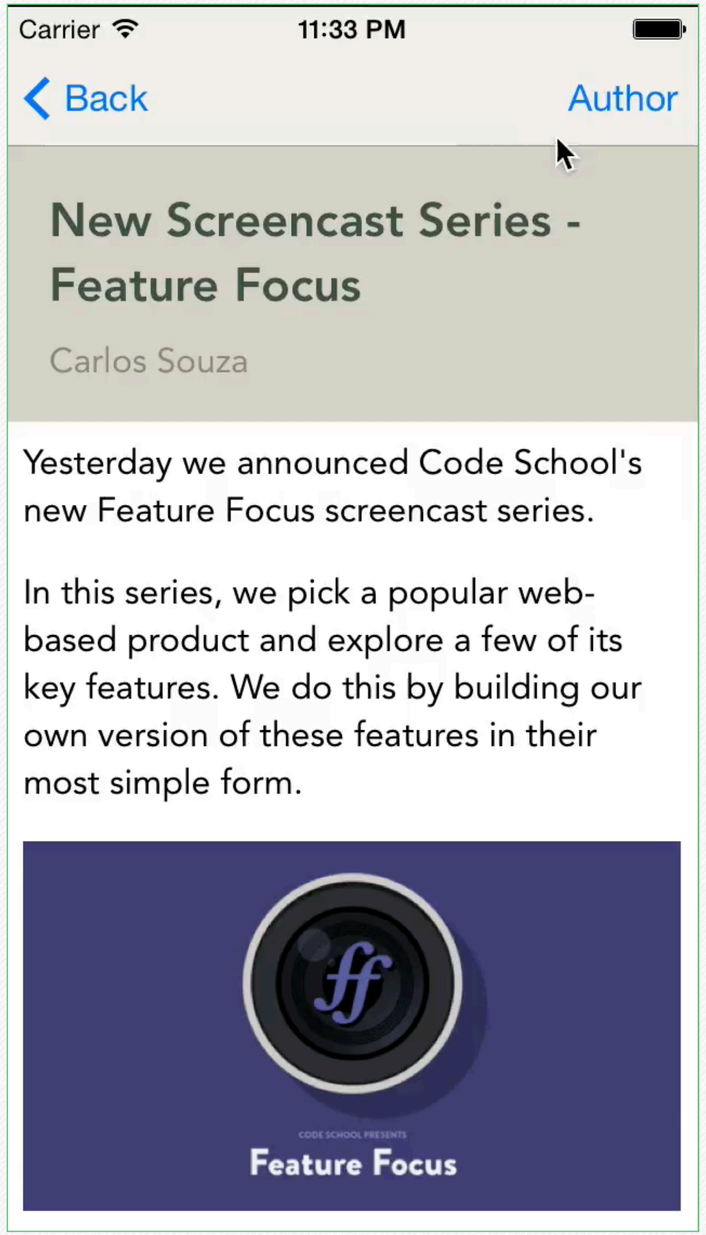

Carrier
11:36 PM
Back
Author
New Screencast Series - Feature Focus
Carlos Souza
Yesterday we announced Code School's new Feature Focus screencast series.
In this series, we pick a popular web-based product and explore a few of its key features. We do this by building our own version of these features in their most simple form.
Feature Focus

Handling the ended state

CSStoryViewController.m

```
- (void)handlePan:(UIPanGestureRecognizer *)gesture
{
    if (gesture.state == UIGestureRecognizerStateEnded) {

        remove existing behaviors

        add a downward gravity behavior

        remove the view and gravity behavior when
        the view disappears from the screen

    }
}
```

Drag working video



Composed behaviors

Behaviors can be grouped together into a single class

Makes code easier to read and more organized

`UIDynamicBehavior` (not UIDynamicItemBehavior)

Setting up the behavior subclass

```
#import <Foundation/Foundation.h>

@interface CSAuthorViewBehavior : UIDynamicBehavior

- (instancetype)initWithItem:(UIView *)item referenceView:(UIView *)referenceView;

@end
```

```
#import "CSAuthorViewBehavior.h"

@implementation CSAuthorViewBehavior

- (id)initWithItem:(UIView *)item referenceView:(UIView *)referenceView {
    self = [super init];
    if(self) {
    }
    return self;
}
```

the view being affected by the behavior

Setting up the behavior subclass

```
#import "CSAuthorViewBehavior.h"

@implementation CSAuthorViewBehavior

- (id)initWithItem:(UIView *)item referenceView:(UIView *)referenceView {
    self = [super init];
    if(self) {
        UISnapBehavior *snapBehavior = [[UISnapBehavior alloc] initWithItem:item
                                        snapToPoint:CGPointMake(CGRectGetMidX(referenceView.frame), 200)];
        snapBehavior.damping = 0.4;
        [self addChildBehavior:snapBehavior];
    }
    return self;
}
```

self is a UIDynamicBehavior

add child behaviors to self

Using the behavior subclass

CSStoryViewController.m

```
- (void)showAuthorView:(id)sender
{
    ...
    self.authorViewBehavior = [[CSAuthorViewBehavior alloc] initWithItem:self.authorView
                                                referenceView:self.view];

    [self.animator addBehavior:self.authorViewBehavior];
}
```

the code you call is now reduced to this

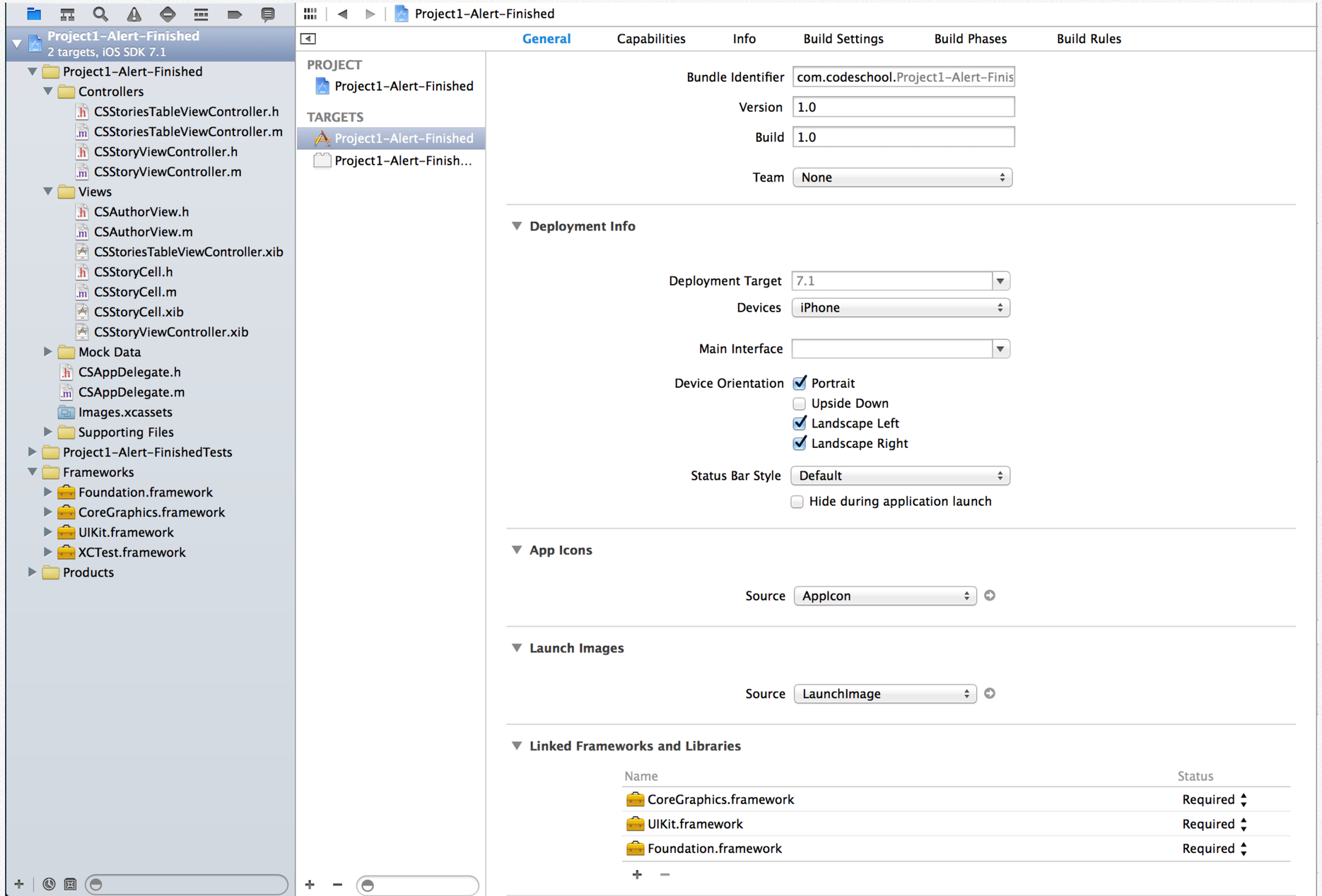
DynamicXray

UIKit Dynamics run-time inspection library created by Chris Miles

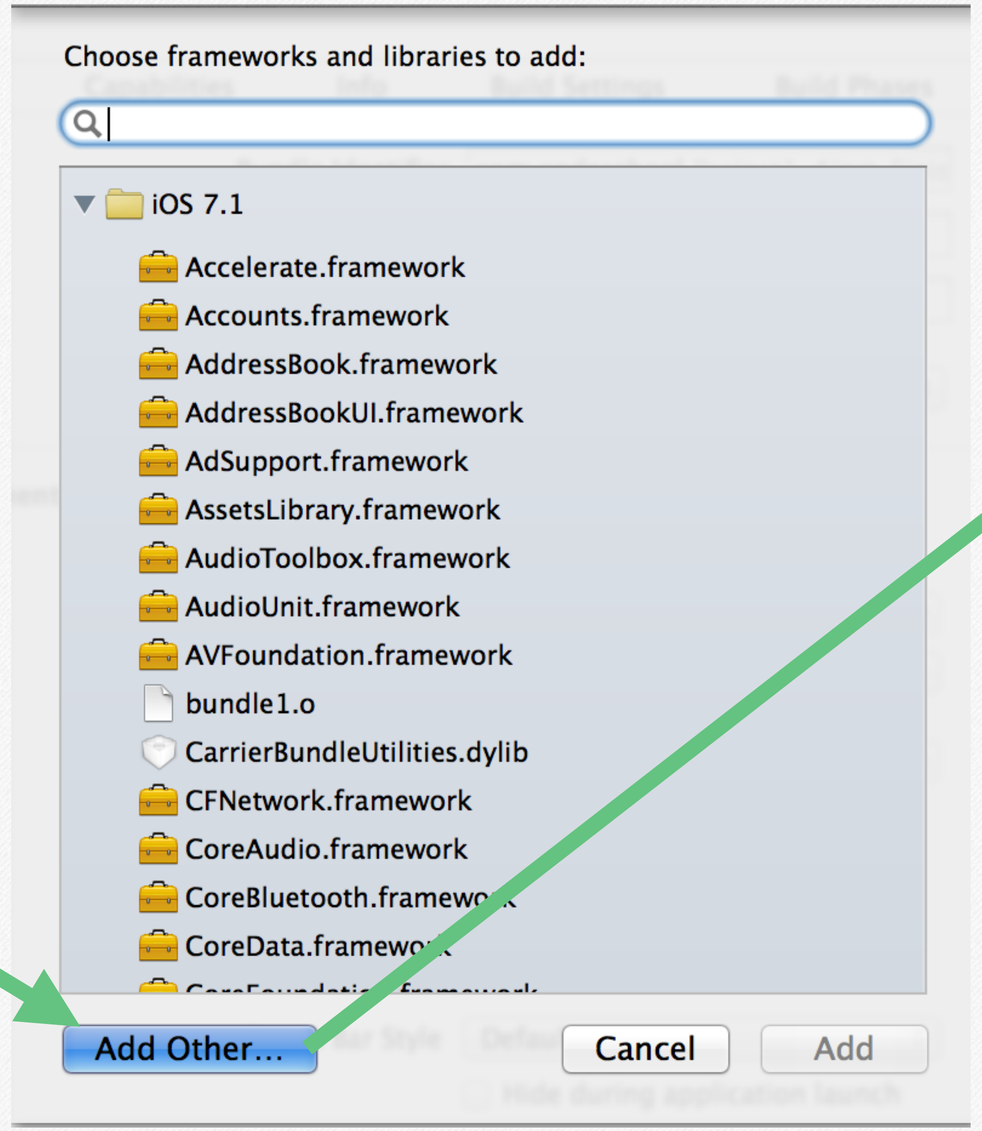
http://dynamicxray.net/

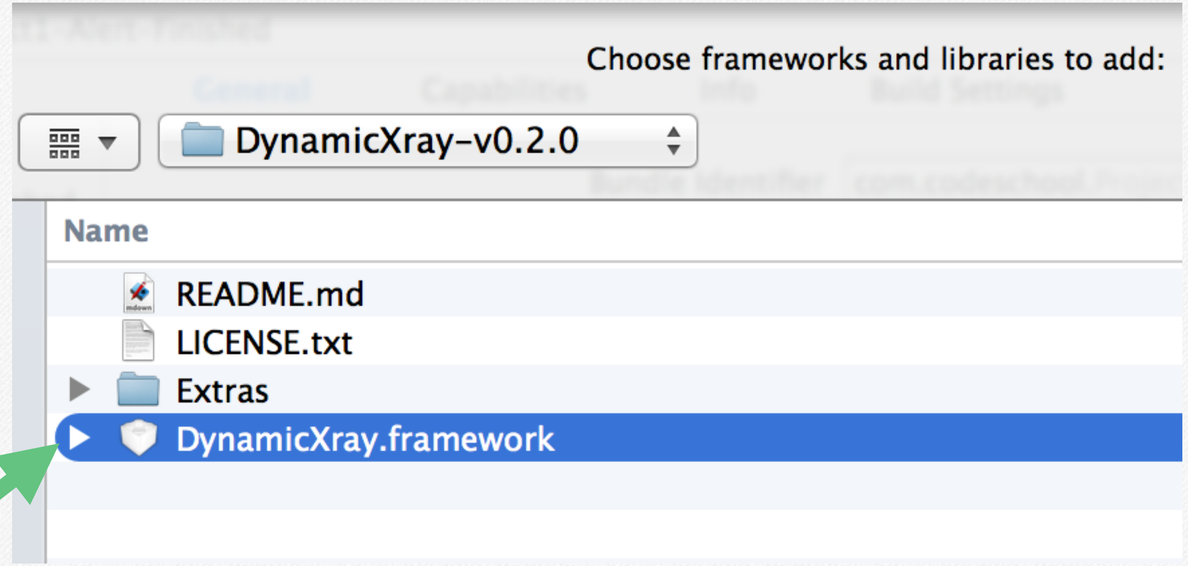
.com is a Turkish X-ray machine distributor

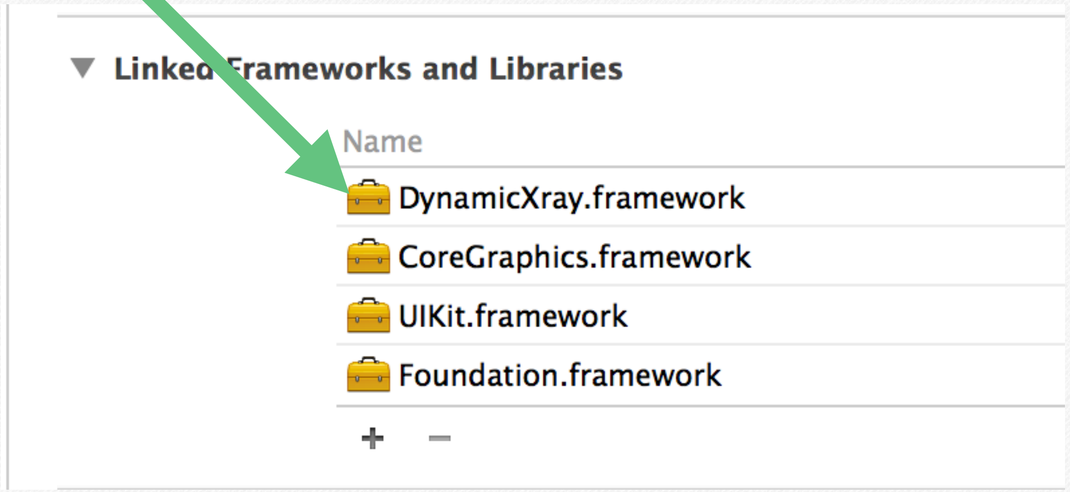
Setting up DynamicXray - Link the framework



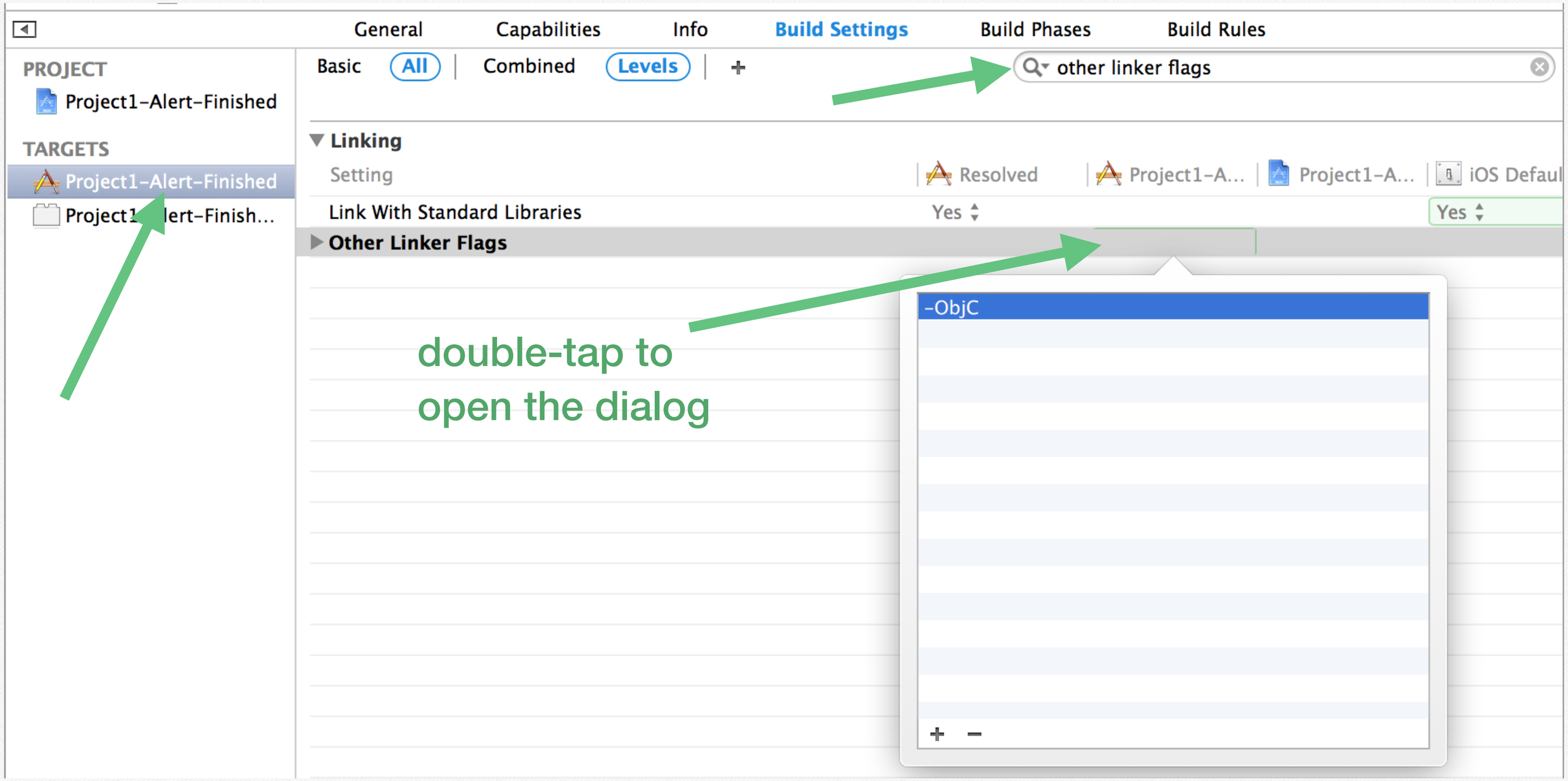
Setting up DynamicXray - Link the framework







Setting up DynamicXray - Add -ObjC linker flag



Setting up DynamicXray - Import the framework in a class

CSStoryViewController.m

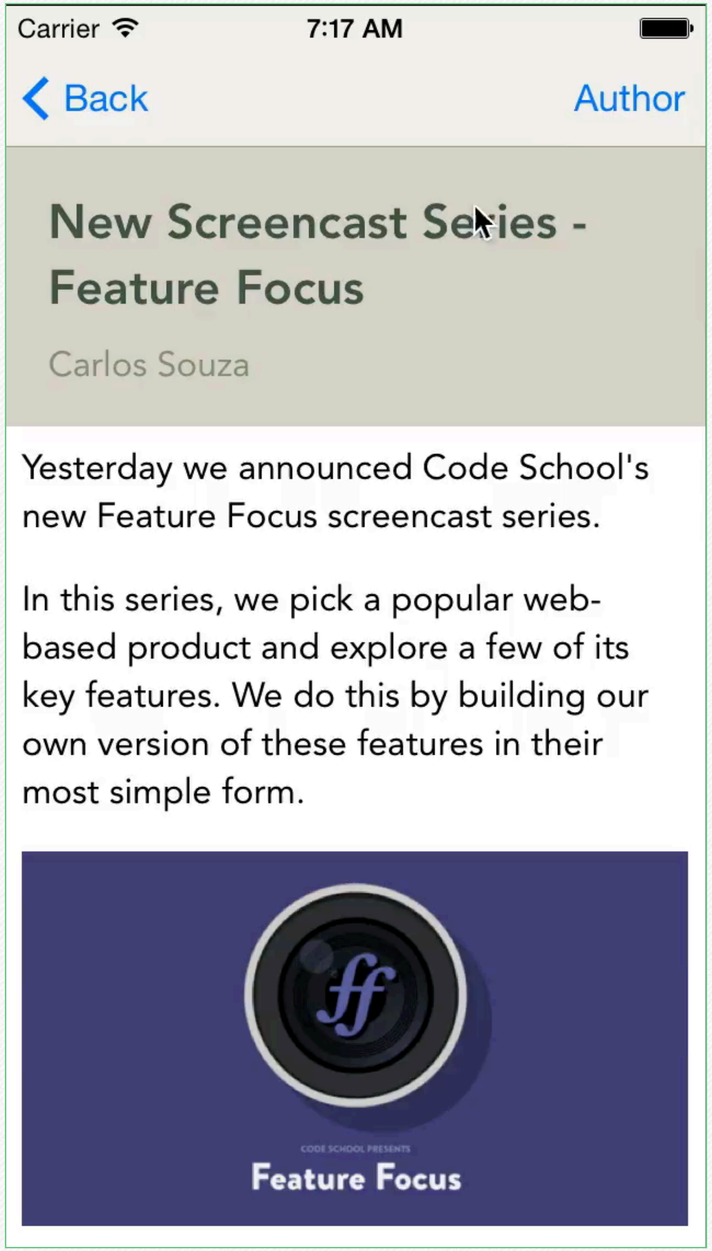
```
#import "CSStoryViewController.h"
#import "CSAuthorView.h"
#import <DynamicXray/DynamicXray.h>

@interface CSStoryViewController () <UIWebViewDelegate, UIDynamicAnimatorDelegate>

...
```

Setting up DynamicXray - Create and add an instance

CSStoryViewController.m

```
- (void)showAuthorView:(id)sender
{
    ...

    self.snapBehavior = [[UISnapBehavior alloc] initWithItem:self.authorView
snapToPoint:CGPointMake(CGRectGetMidX(self.view.frame), 200)];
    self.snapBehavior.damping = 0.9;

    [self.animator addBehavior:self.snapBehavior];

    DynamicXray *xray = [[DynamicXray alloc] init];
    [self.animator addBehavior:xray];
}
```

DynamicXray in action

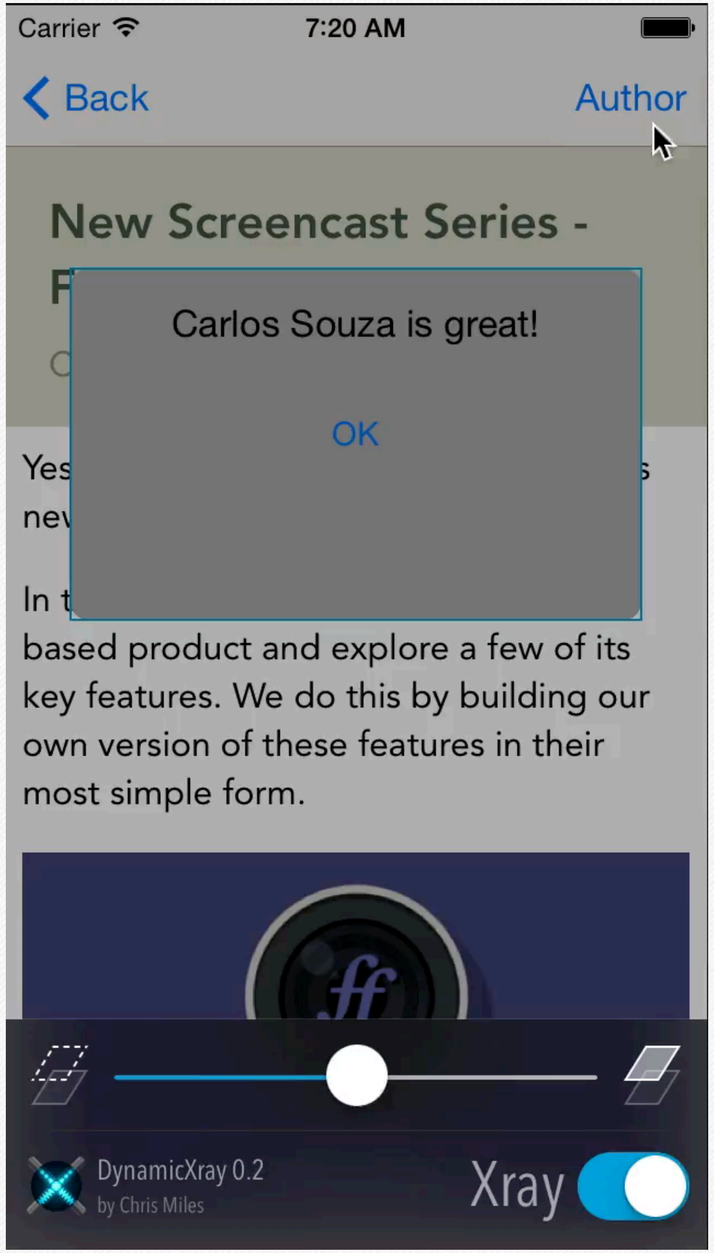


Controlling options in DynamicXray

CSStoryViewController.m

```
- (void)showAuthorView:(id)sender
{
    ...

    self.snapBehavior = [[UISnapBehavior alloc] initWithItem:self.authorView
snapToPoint:CGPointMake(CGRectGetMidX(self.view.frame), 200)];
    self.snapBehavior.damping = 0.9;

    [self.animator addBehavior:self.snapBehavior];

    DynamicXray *xray = [[DynamicXray alloc] init];
    [self.animator addBehavior:xray];
    [xray presentConfigurationViewController];
}
```

DynamicXray in action



Morning Recap

Simple behaviors

Practiced with:

- Custom alert pop-up
- Unique menu options
- Dynamics during transitioning
- Dragging views with gravity

Composed behaviors for greater re-use and readability

DynamicXray for real-time behavior inspection

Thanks!

Jon Friskics

@jonfriskics

https://github.com/jonfriskics/360iDevWorkshop

http://codeschool.com