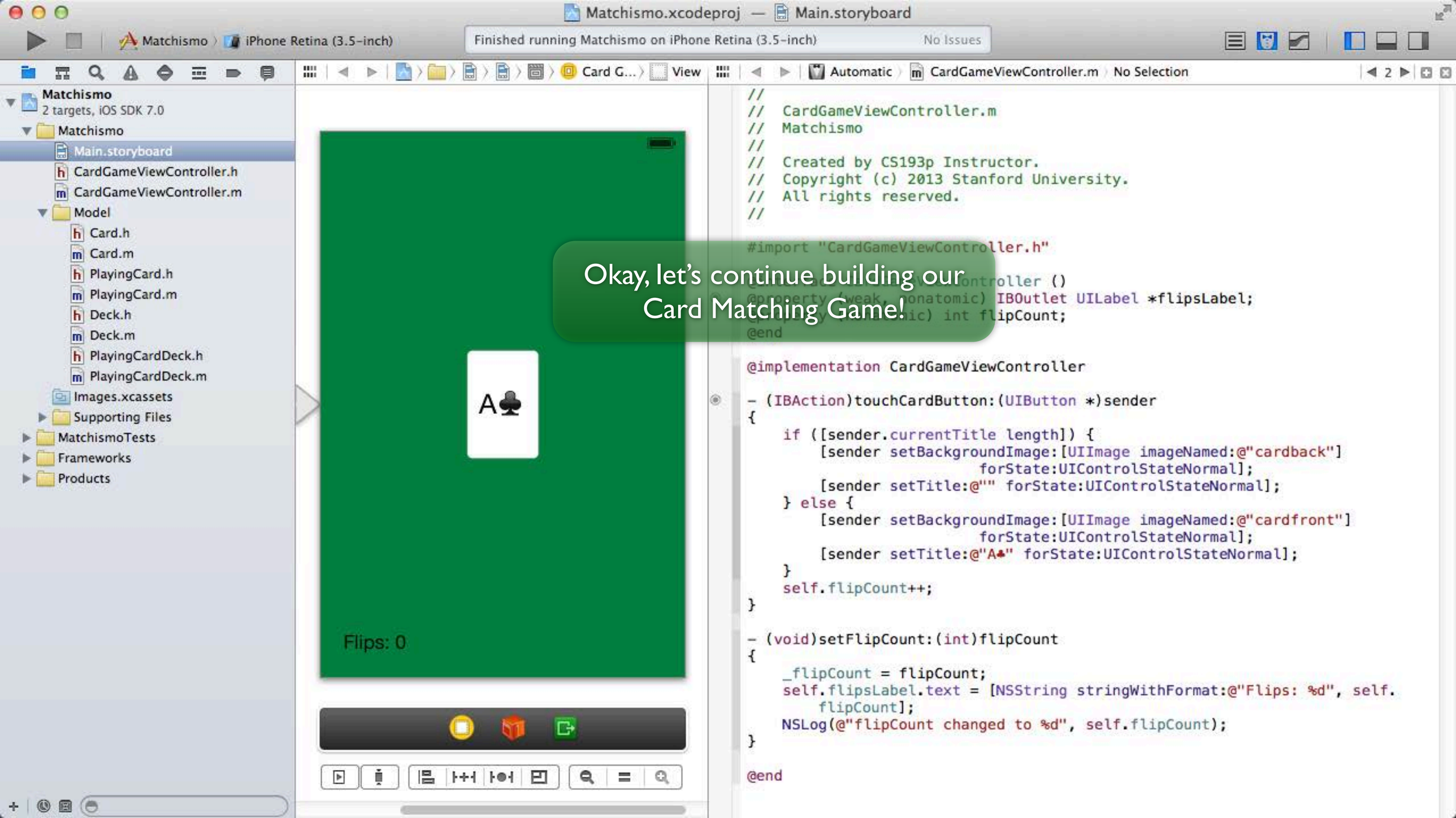
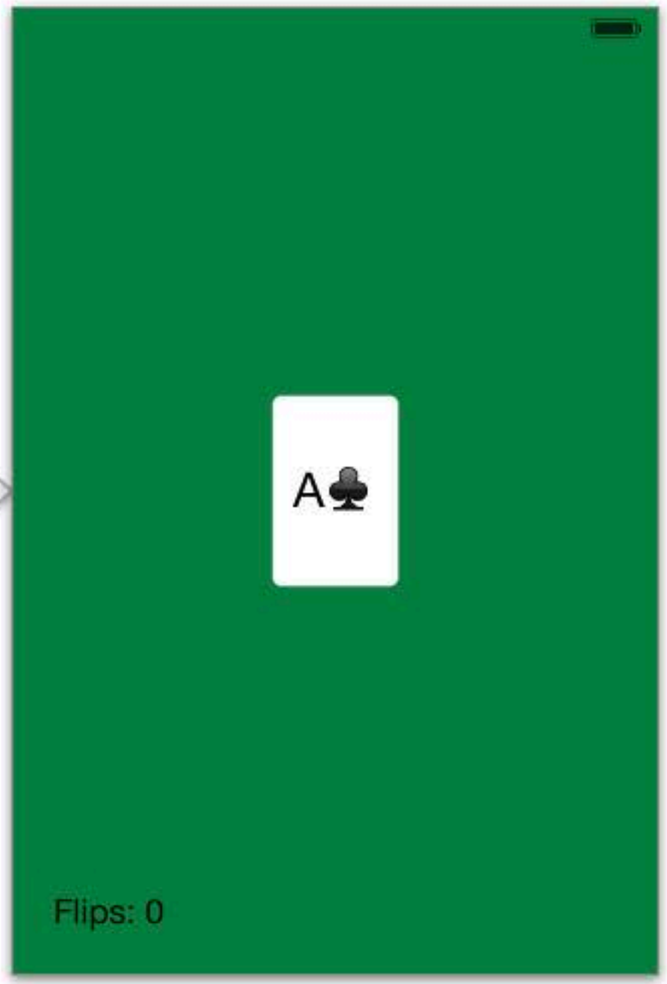
iOS Mobile Development

Okay, let's continue building our Card Matching Game!





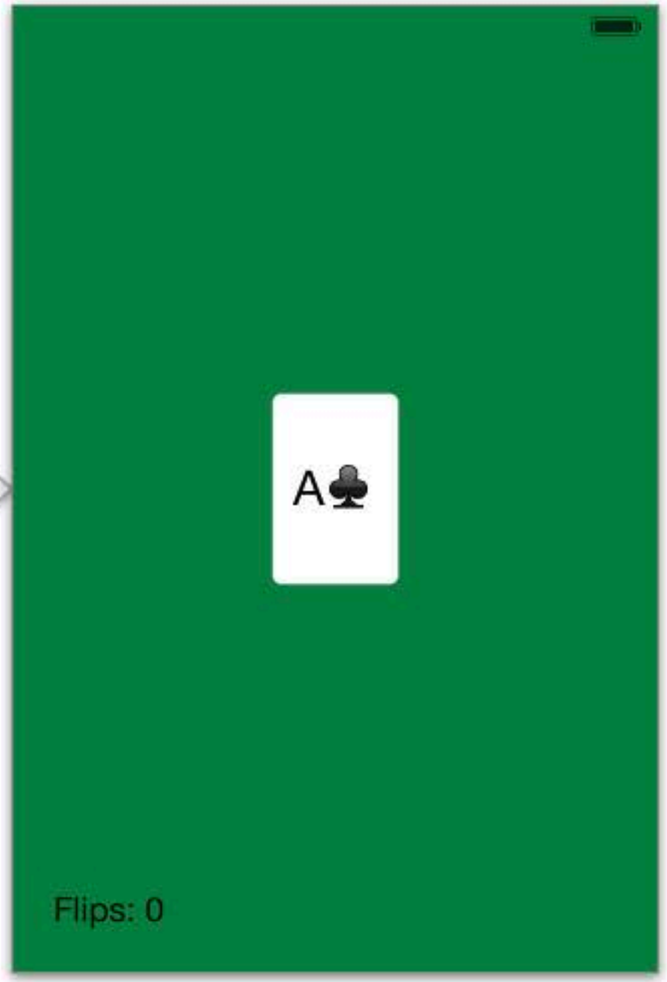
Close Navigator.

```
//
//  CardGameViewController.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardGameViewController.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:@"A♣" forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › No Selection

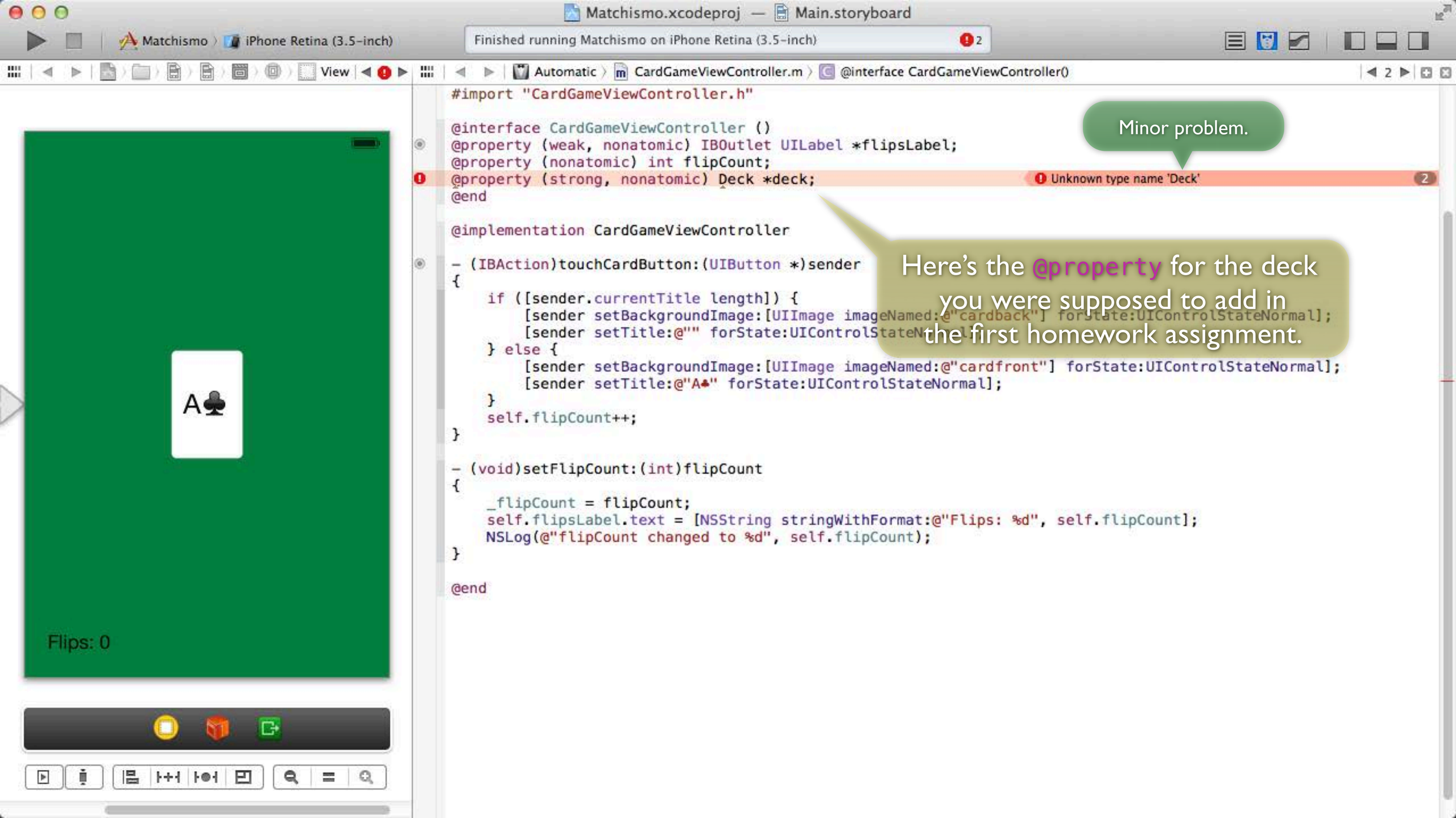


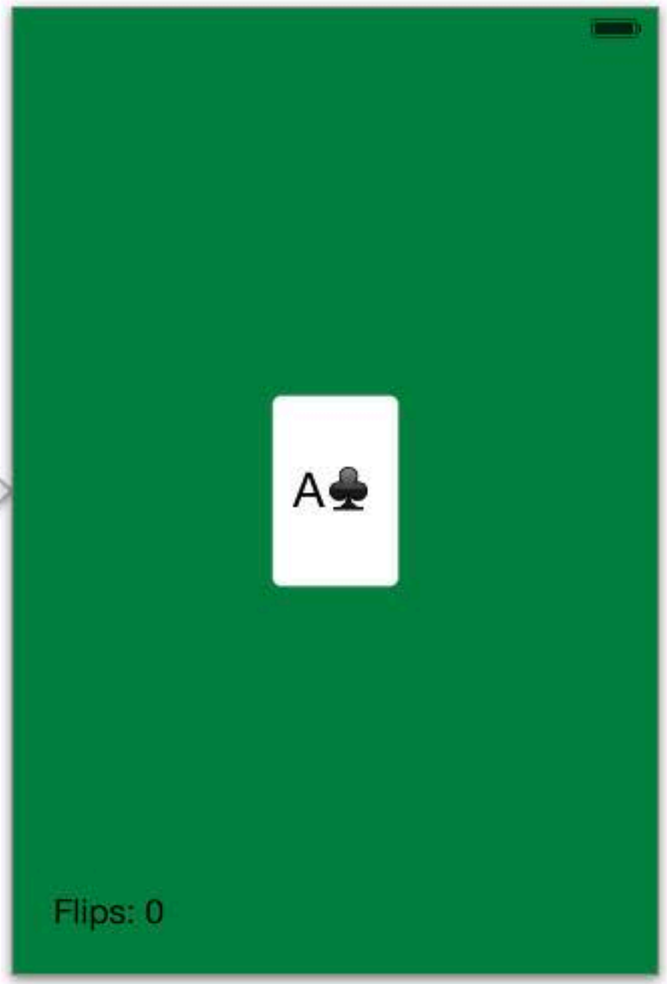

```objc
#import "CardGameViewController.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:@"A♣" forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Scrolled down.

The first thing we're going to do is go over last week homework assignment.

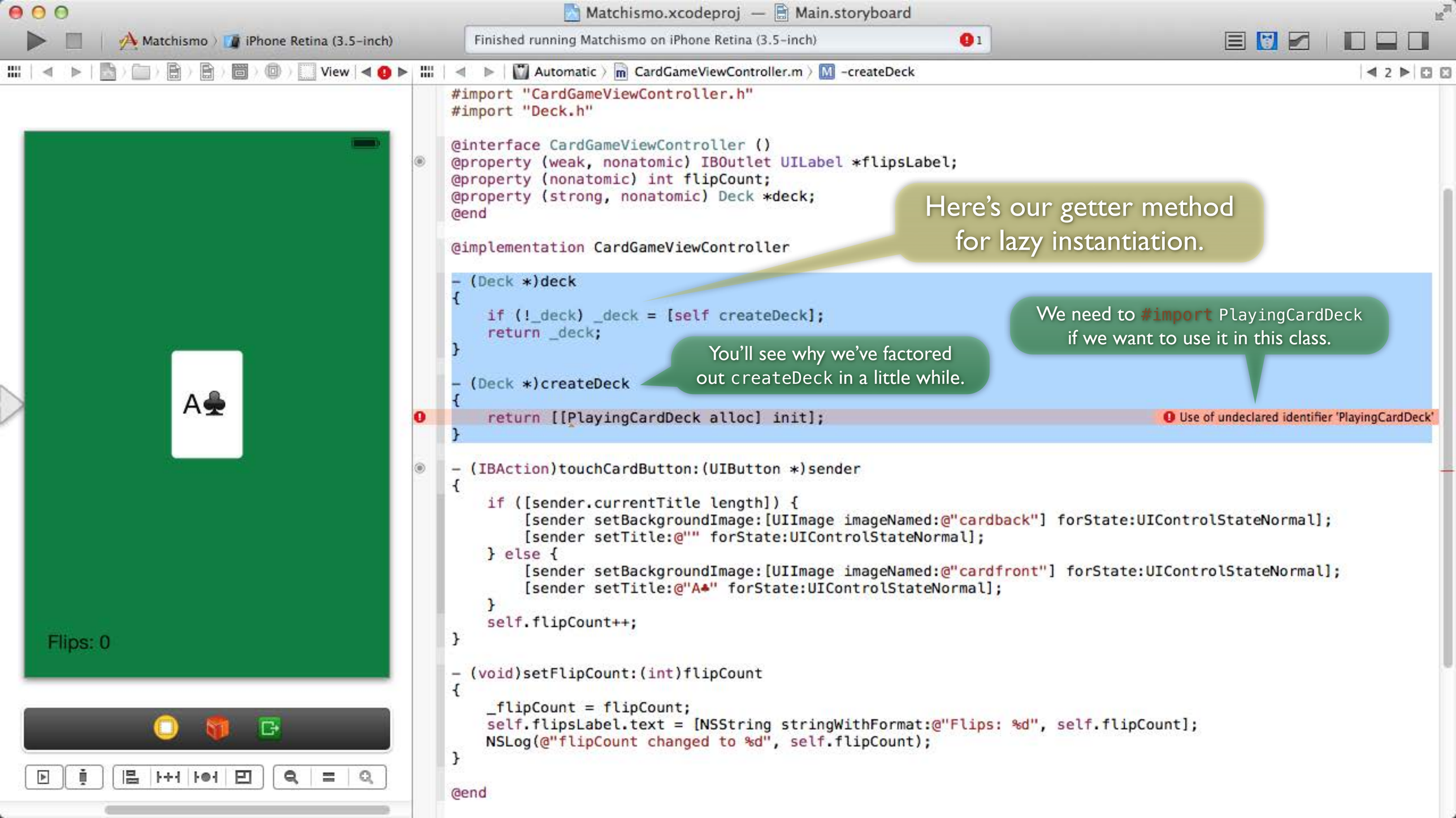
This is the only time we will go over the homework assignment in lecture.

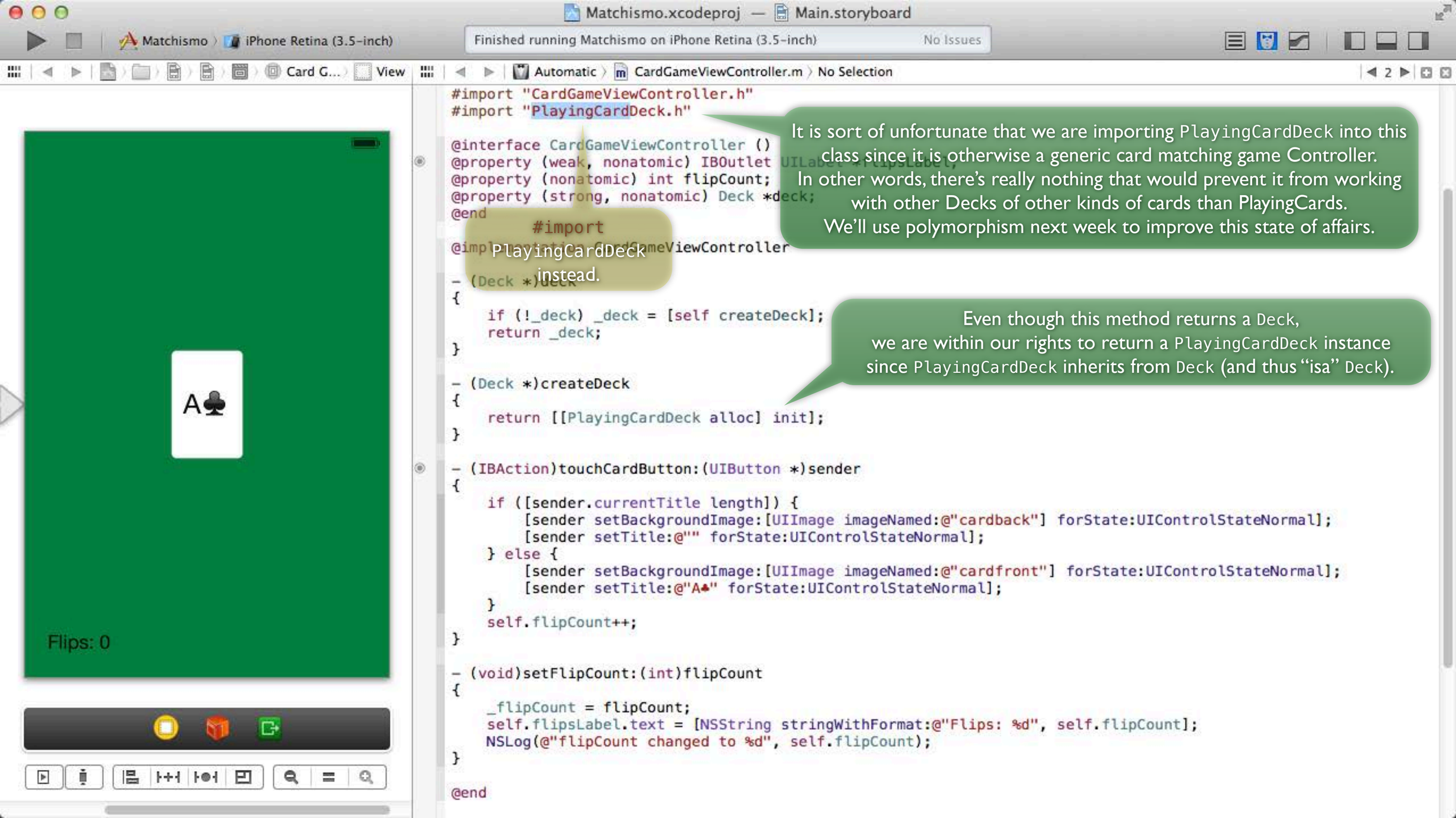

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
2
View
Automatic 〉 CardGameViewController.m 〉 @interface CardGameViewController()
2
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
Unknown type name 'Deck'
2
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:@"A♣" forState:UIControlStateNormal];
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
Minor problem.
Here's the @property for the deck you were supposed to add in the first homework assignment.
A♣
Flips: 0

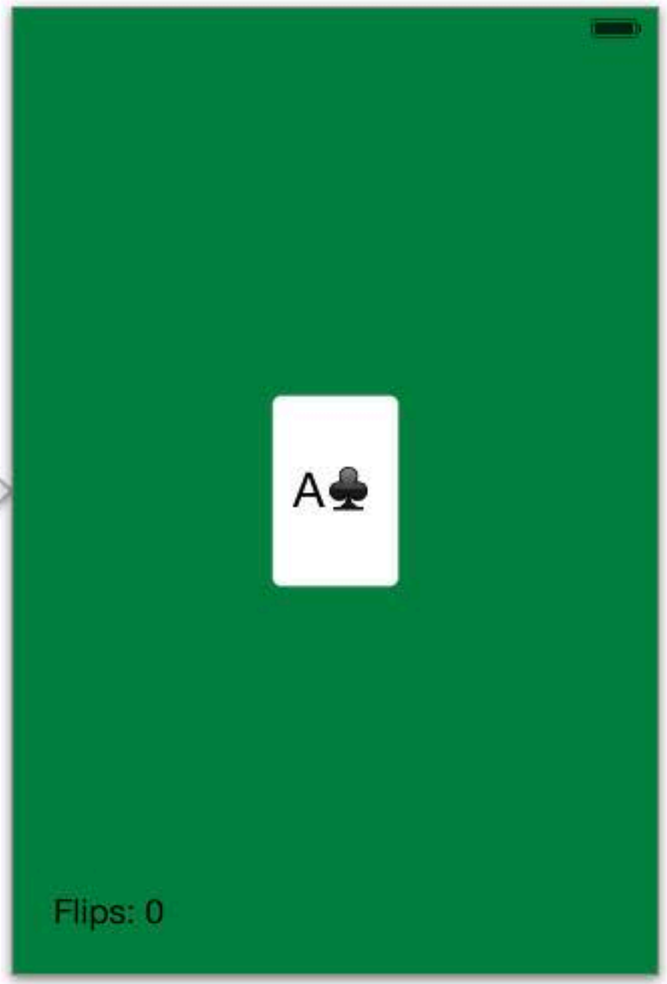



```
#import "CardGameViewController.h"
#import "Deck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:@"A♣" forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

We need to `#import Deck` if we want to use it in this class.

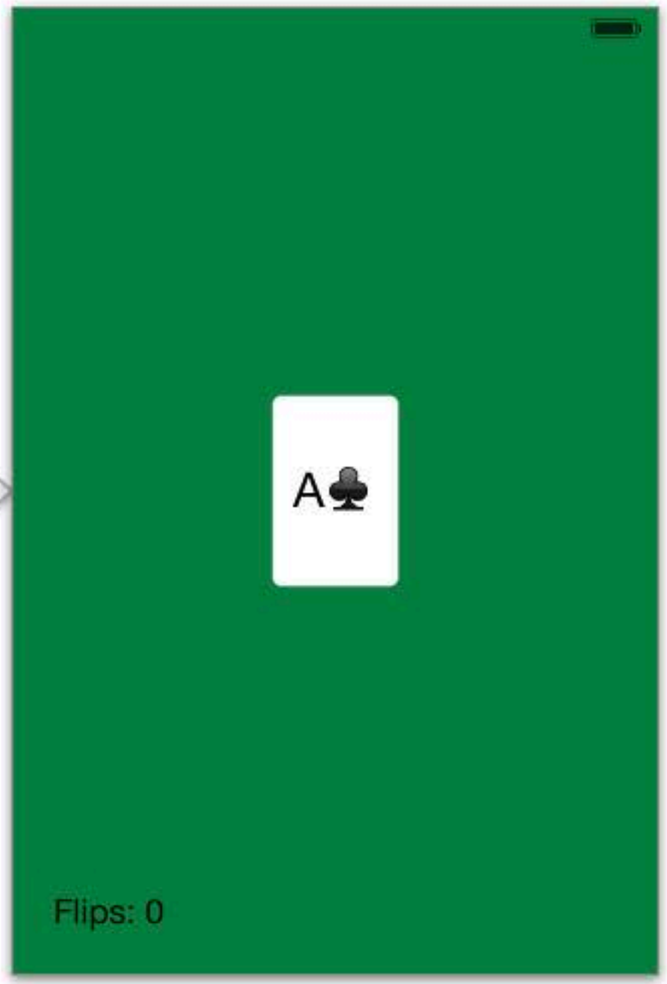

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
1
View
Automatic › CardGameViewController.m › -createDeck
2
A♣
Flips: 0
#import "CardGameViewController.h"
#import "Deck.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end
@implementation CardGameViewController
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
Use of undeclared identifier 'PlayingCardDeck'
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:@"A♣" forState:UIControlStateNormal];
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
Here's our getter method for lazy instantiation.
You'll see why we've factored out createDeck in a little while.
We need to #import PlayingCardDeck if we want to use it in this class.


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › No Selection
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end
@implementation CardGameViewController
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:@"A♣" forState:UIControlStateNormal];
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
#import PlayingCardDeck instead.
It is sort of unfortunate that we are importing PlayingCardDeck into this class since it is otherwise a generic card matching game Controller.
In other words, there's really nothing that would prevent it from working with other Decks of other kinds of cards than PlayingCards.
We'll use polymorphism next week to improve this state of affairs.
Even though this method returns a Deck,
we are within our rights to return a PlayingCardDeck instance
since PlayingCardDeck inherits from Deck (and thus "isa" Deck).
A♣
Flips: 0



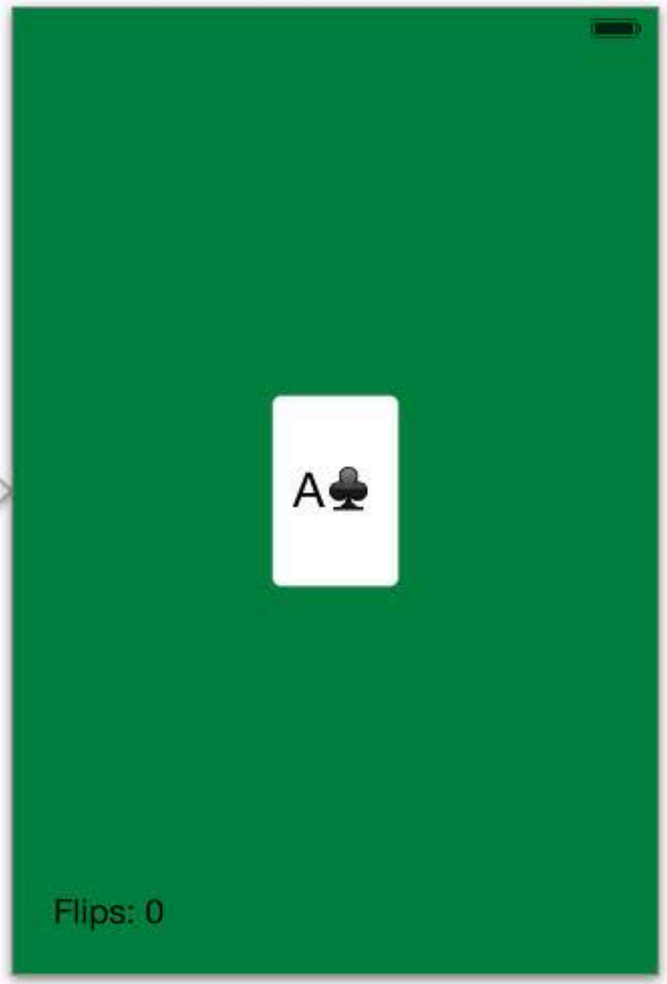

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:@"A♣" forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Simply draw a random card ...




```objc
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
        [sender setTitle:randomCard.contents forState:UIControlStateNormal];
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

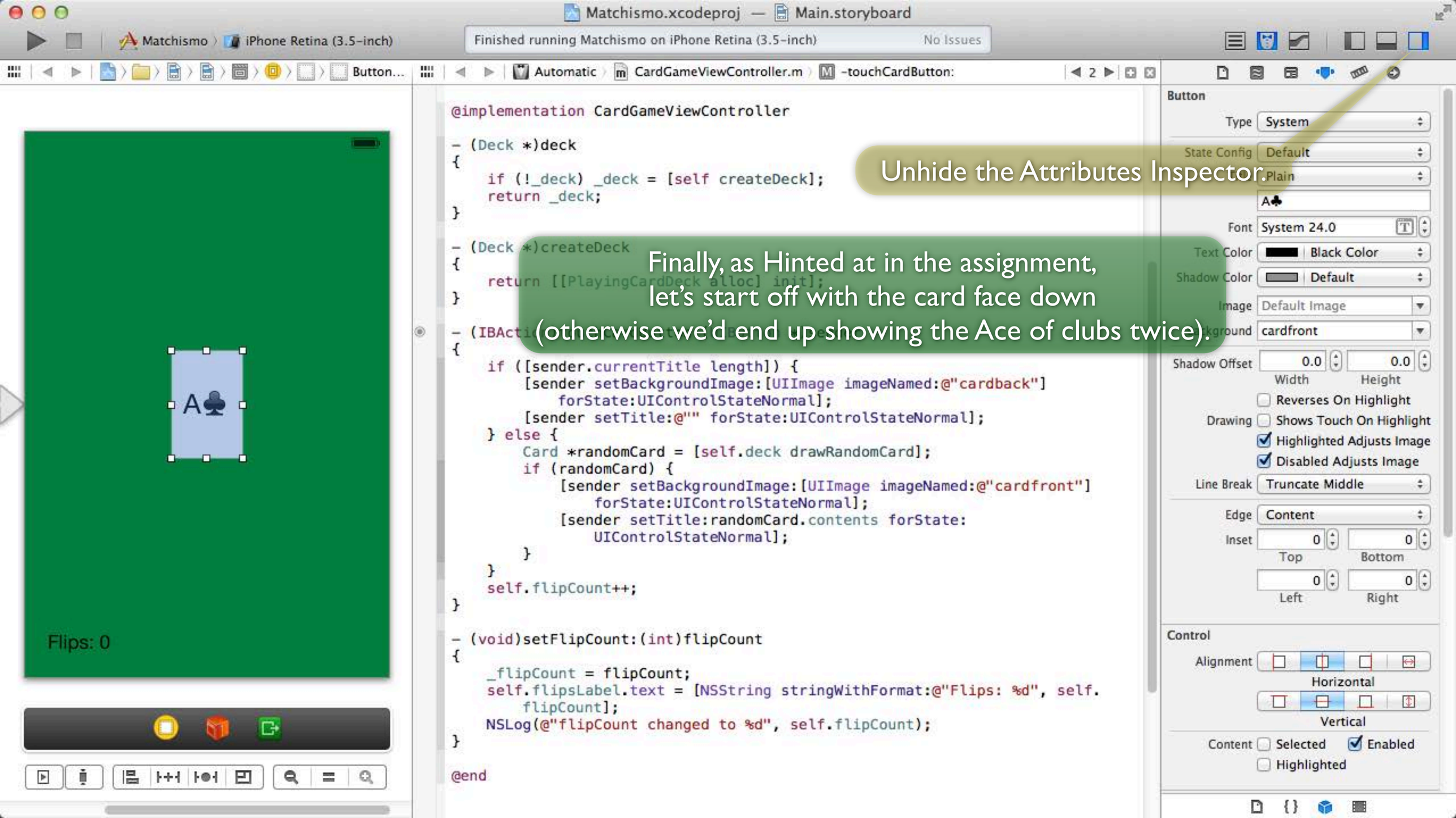
... and put its contents on the button.




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

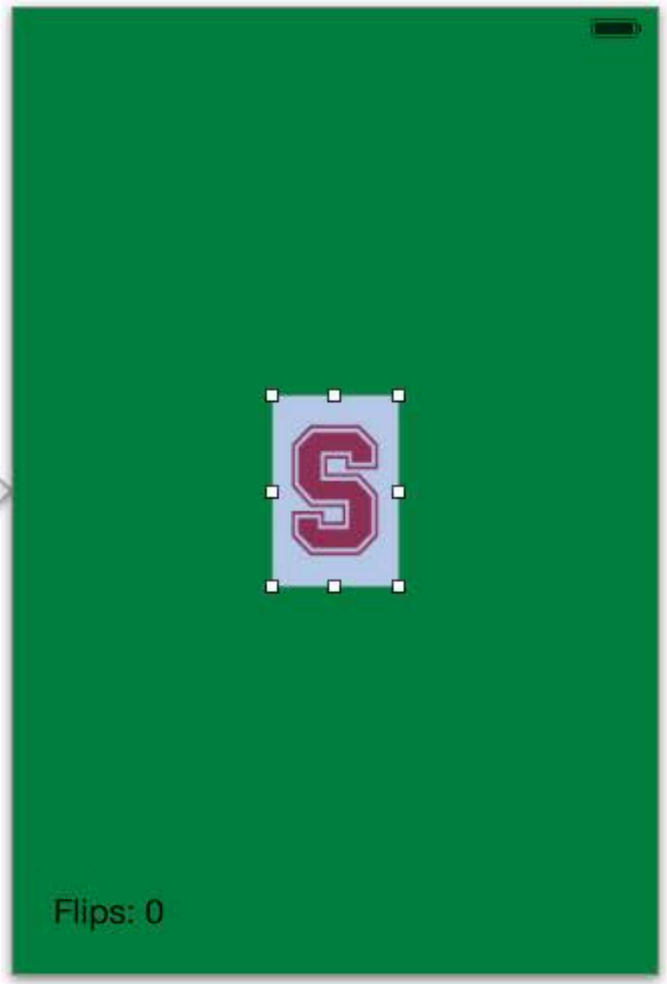
But how do we handle running out of cards?
Easy.
Just don't flip to the front if the deck is empty
(which we know if `drawRandomCard` returns `nil`).

Protect against an empty deck.


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Automatic › CardGameViewController.m › -touchCardButton:
@implementation CardGameViewController
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAct
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:
UIControlStateNormal];
}
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.
flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
Unhide the Attributes Inspector.
Finally, as Hinted at in the assignment,
let's start off with the card face down
(otherwise we'd end up showing the Ace of clubs twice).
Button
Type System
State Config Default
Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
cardfront
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Drawing Shows Touch On Highlight
Highlighted Adjusts Image
Disabled Adjusts Image
Line Break Truncate Middle
Edge Content
Inset 0 0
Top Bottom
0 0
Left Right
Control
Alignment
Horizontal
Vertical
Content Selected Enabled
Highlighted
A♣
Flips: 0

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues




```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
            forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
                forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:
                UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.
        flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Just clear out the title ...

... and set the image to the back of the card.

Button

Type: System

State Config: Default

Title: Plain

Default Title

Font: System 24.0

Text Color: Black Color

Shadow Color: Default

Image: Default Image

Background: cardback

Shadow Offset: 0.0 Width, 0.0 Height

Drawing: Reverses On Highlight; Shows Touch On Highlight; Highlighted Adjusts Image; Disabled Adjusts Image

Line Break: Truncate Middle

Edge: Content

Inset: 0 Top, 0 Bottom, 0 Left, 0 Right

Control

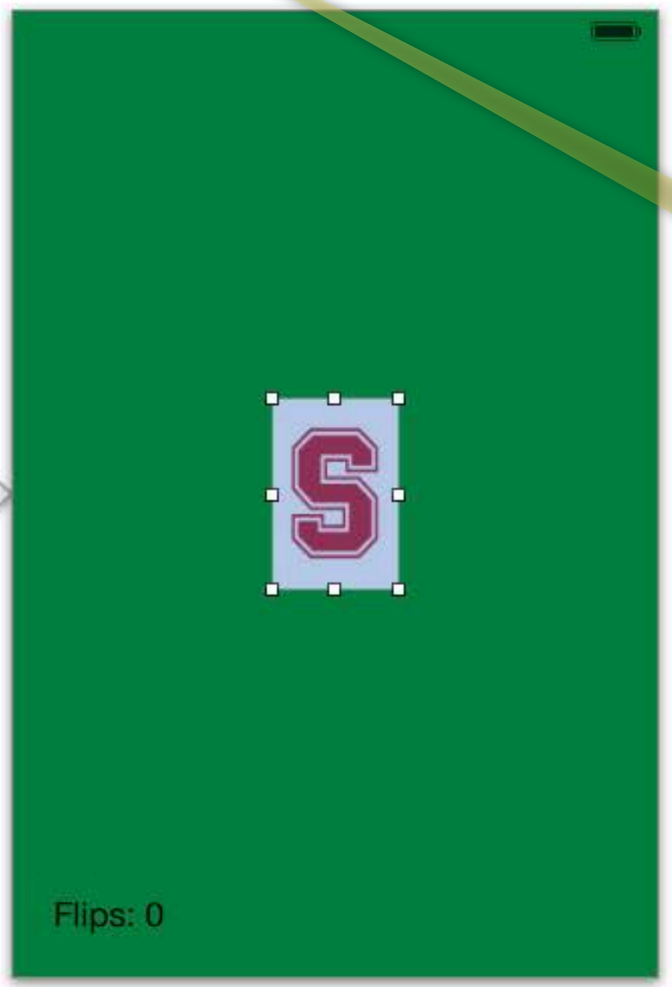
Alignment: Horizontal, Vertical

Content: Selected, Enabled, Highlighted

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

V. › Button | Automatic › CardGameViewController.m › -touchCardButton: | 2



That's all there is to it.

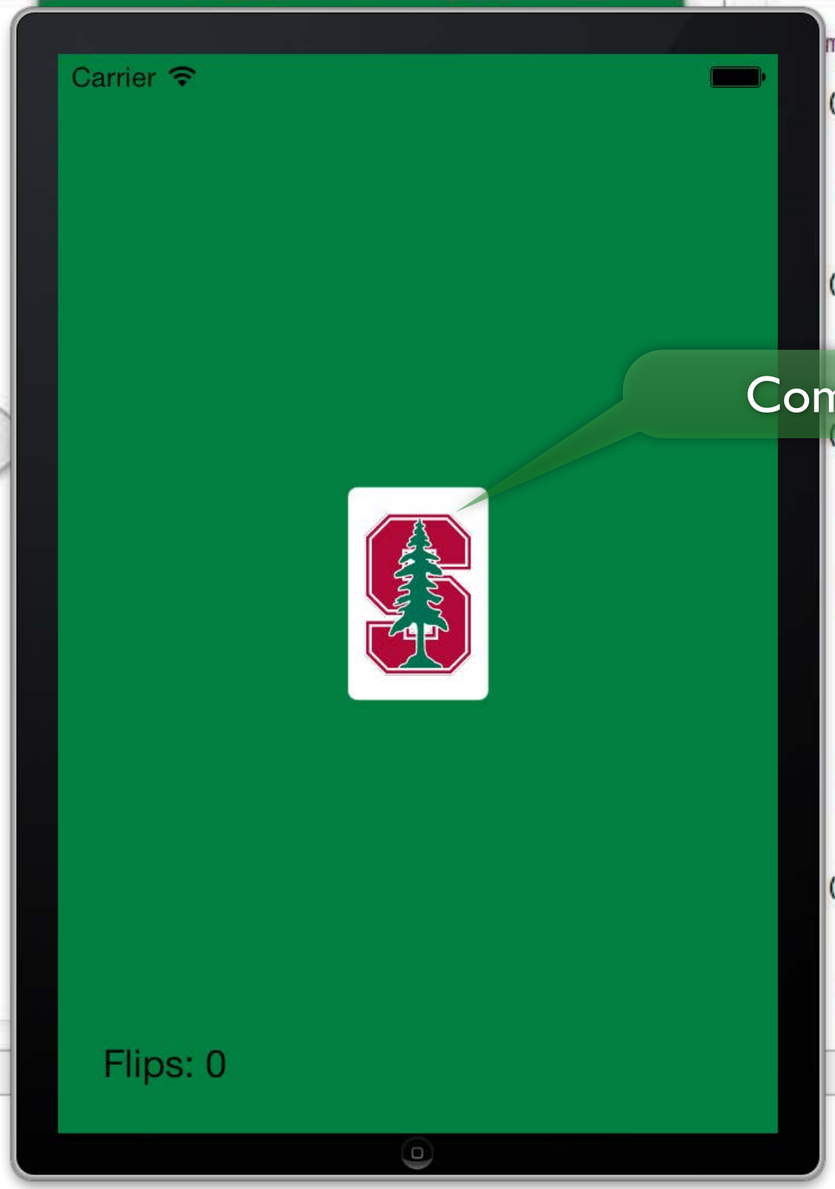
Let's run to see if it works.

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

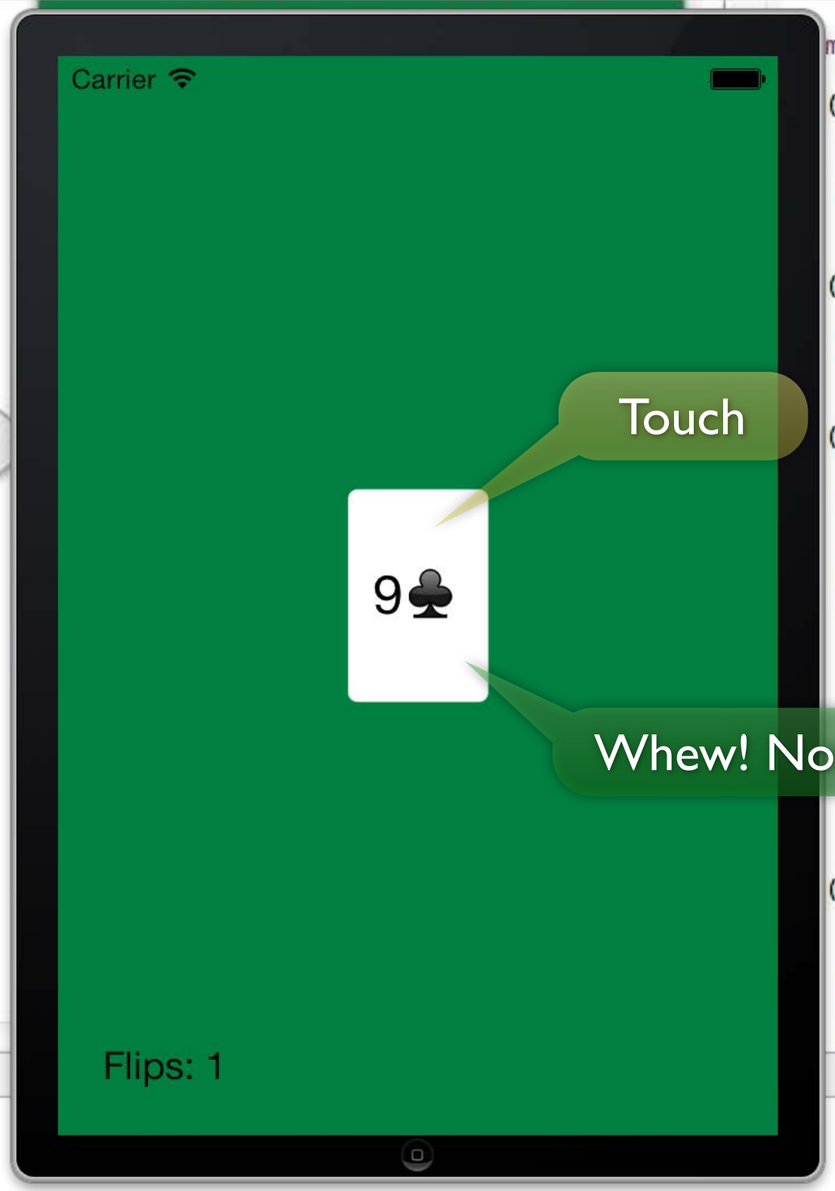
V. › Button | Automatic › CardGameViewController.m › -touchCardButton:



Comes up face down.

```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];

(IBAction)touchCardButton:(UIButton *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;


(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

Auto | All Output



Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

V. › Button | Automatic › CardGameViewController.m › -touchCardButton:

```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;


(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

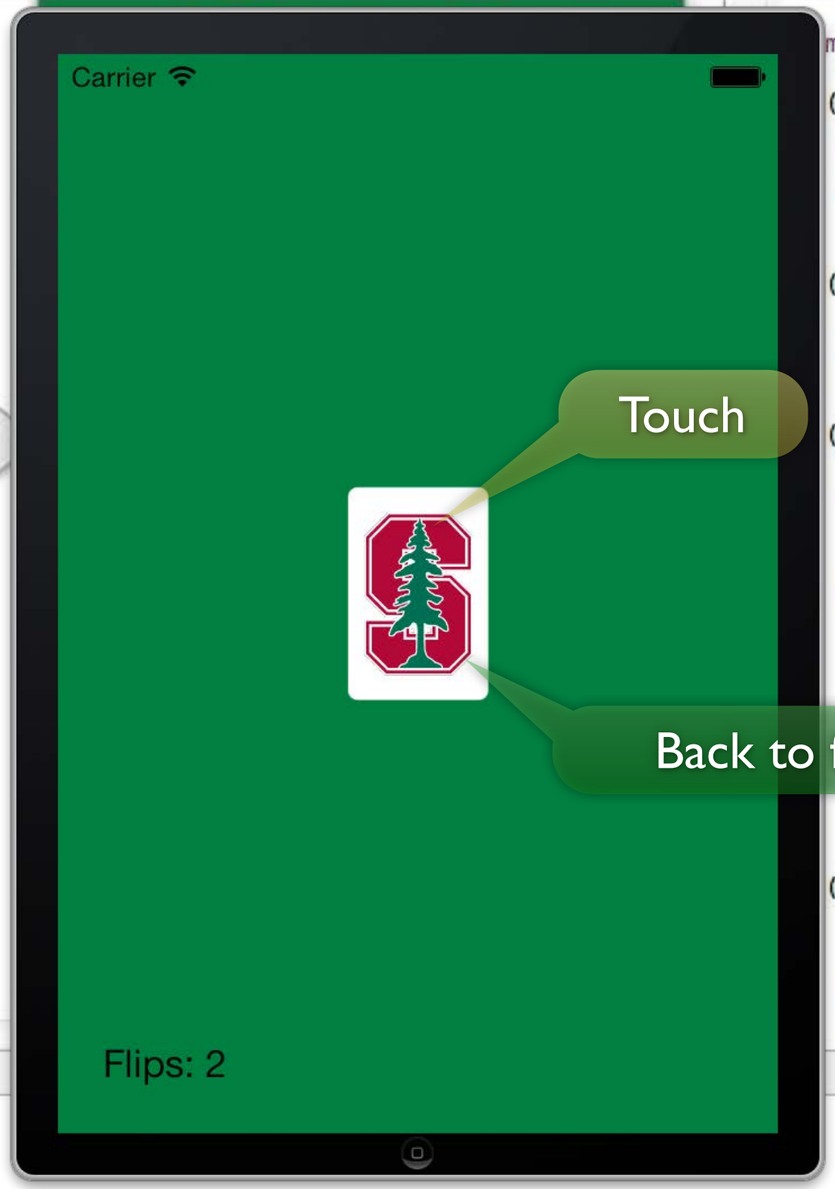
```
2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
```

Auto | All Output

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

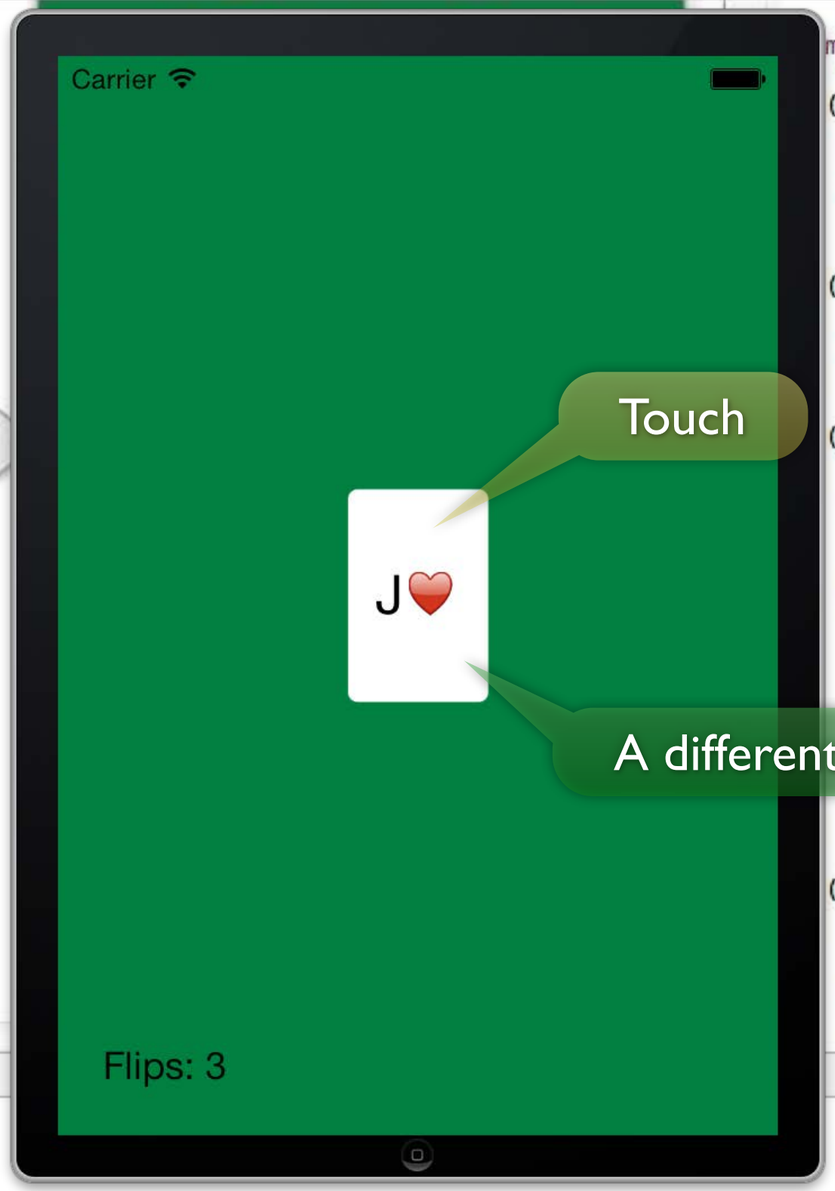
V. › Button | Automatic › CardGameViewController.m › -touchCardButton:




```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;


(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

```
2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
2013-00-00 00:00:00.318 Matchismo[11937:a0b] flipCount changed to 2
```

Auto | All Output



Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

V. › Button | Automatic › CardGameViewController.m › -touchCardButton:

```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;


(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

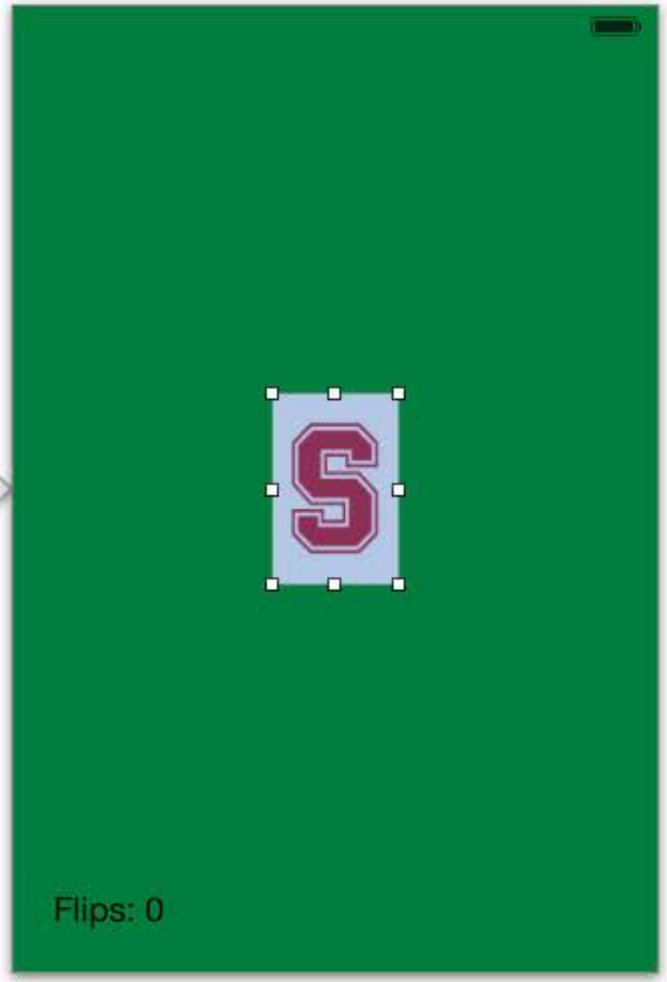
```
2013-00-00 00:00:00.154 Matchismo[11937:a0b] flipCount changed to 1
2013-00-00 00:00:00.318 Matchismo[11937:a0b] flipCount changed to 2
2013-00-00 00:00:00.865 Matchismo[11937:a0b] flipCount changed to 3
```

Auto | All Output

Matchismo.xcodeproj — Main.storyboard

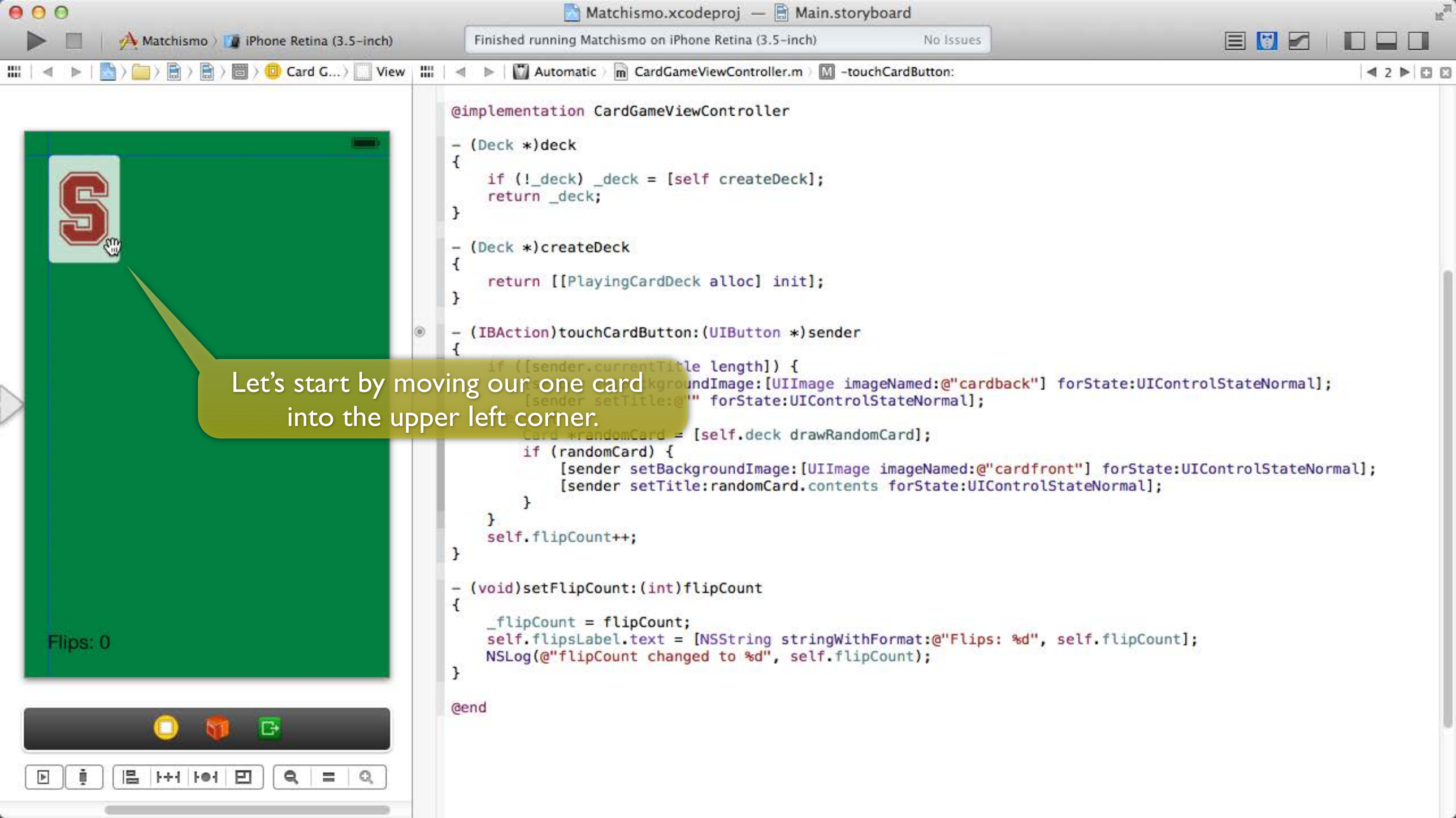
Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

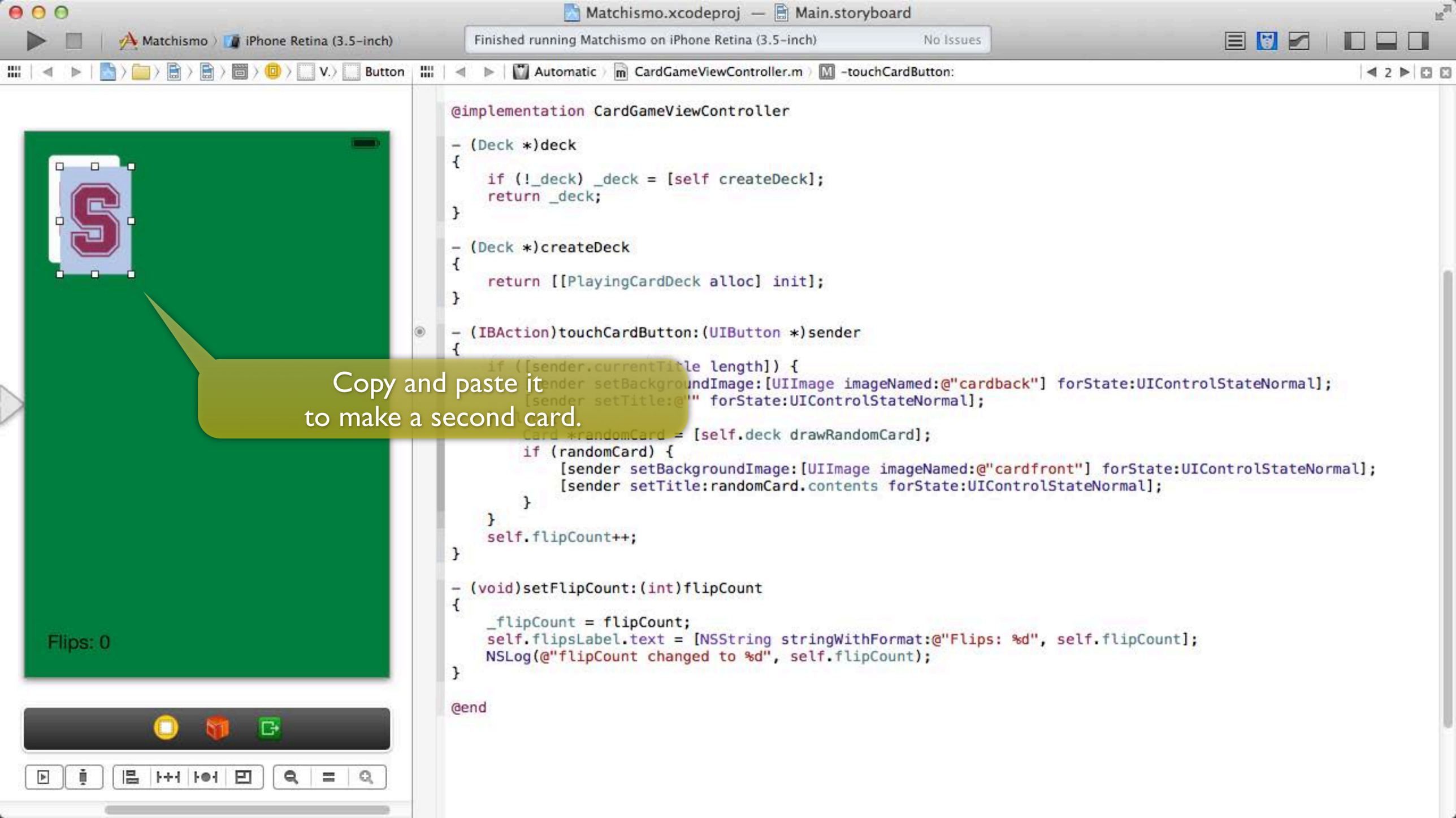
V. › Button | Automatic › CardGameViewController.m › -touchCardButton:




```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

This game would be a lot more interesting with more than just one card!

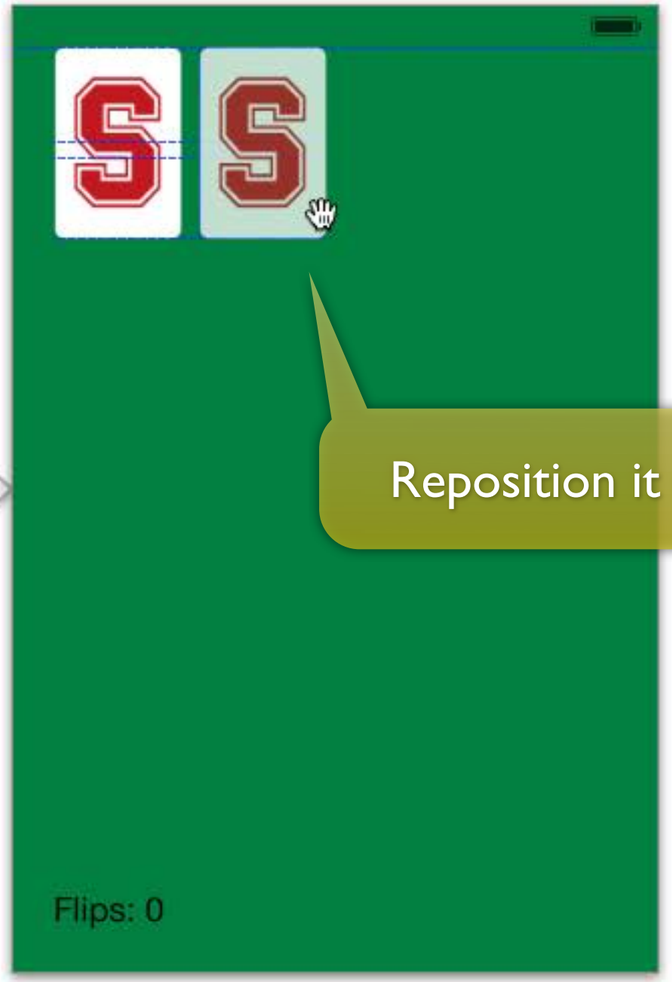

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › -touchCardButton:
@implementation CardGameViewController
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
ndImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
Let's start by moving our one card into the upper left corner.
Flips: 0


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Automatic › CardGameViewController.m › -touchCardButton:
Copy and paste it
to make a second card.
Flips: 0
@implementation CardGameViewController
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › -touchCardButton:



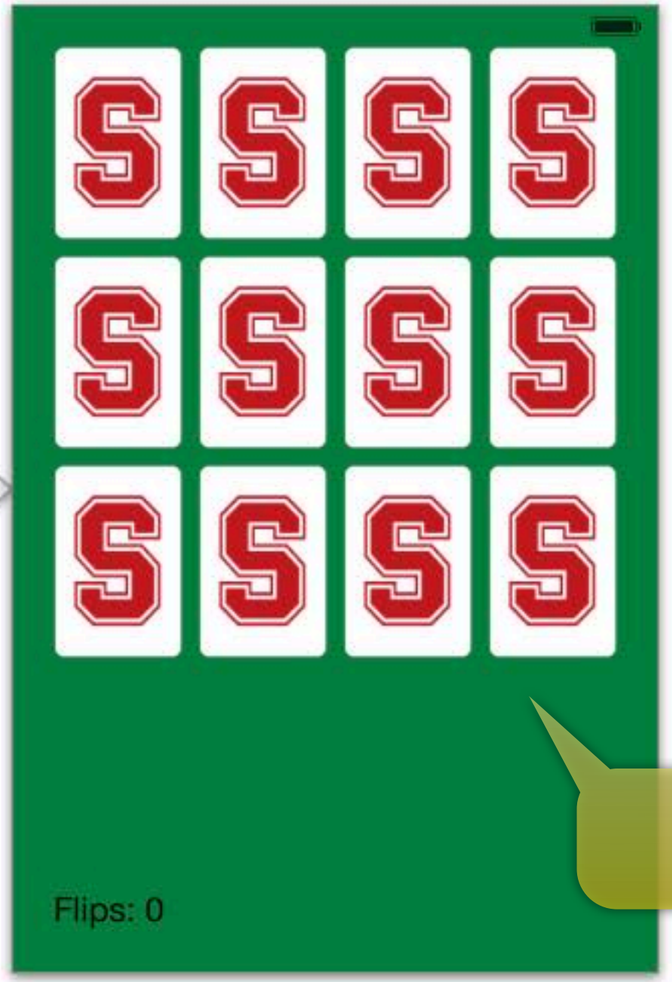
Reposition it using blue guidelines.

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › -touchCardButton:



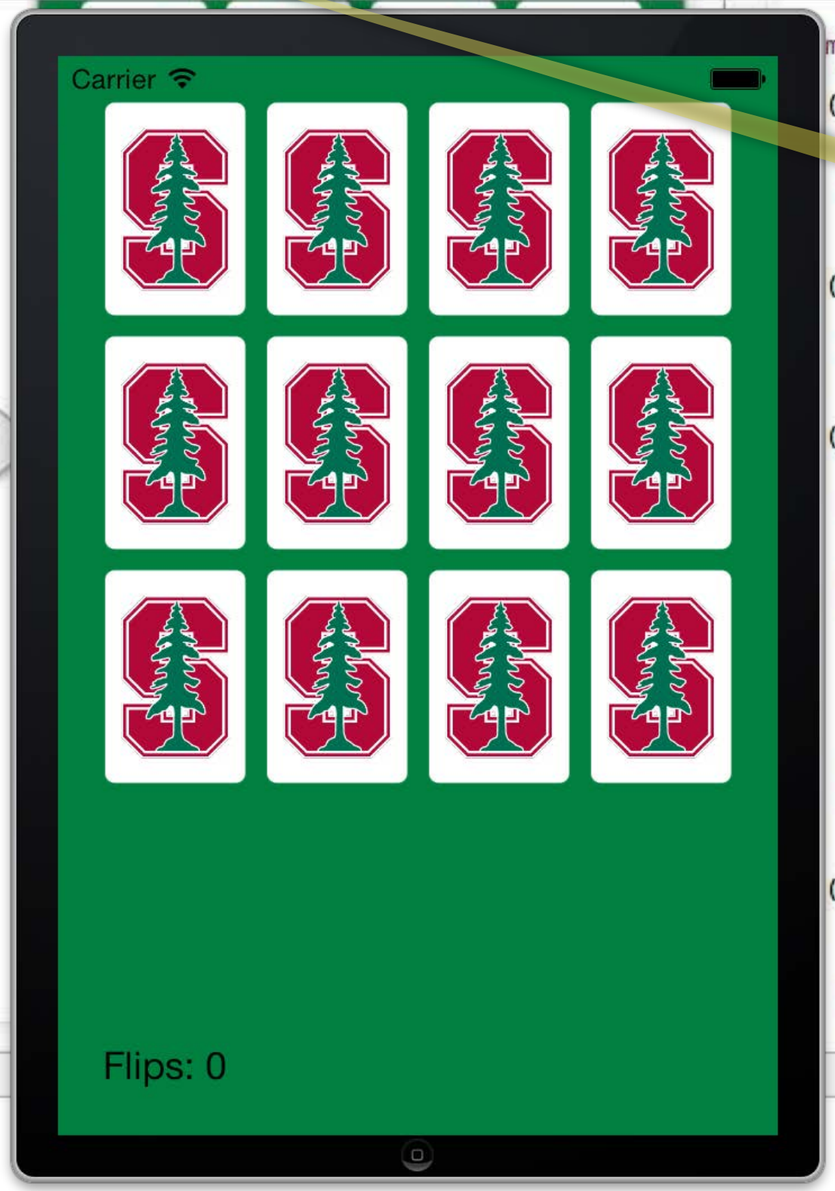

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```

Make a total of 12 cards.

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

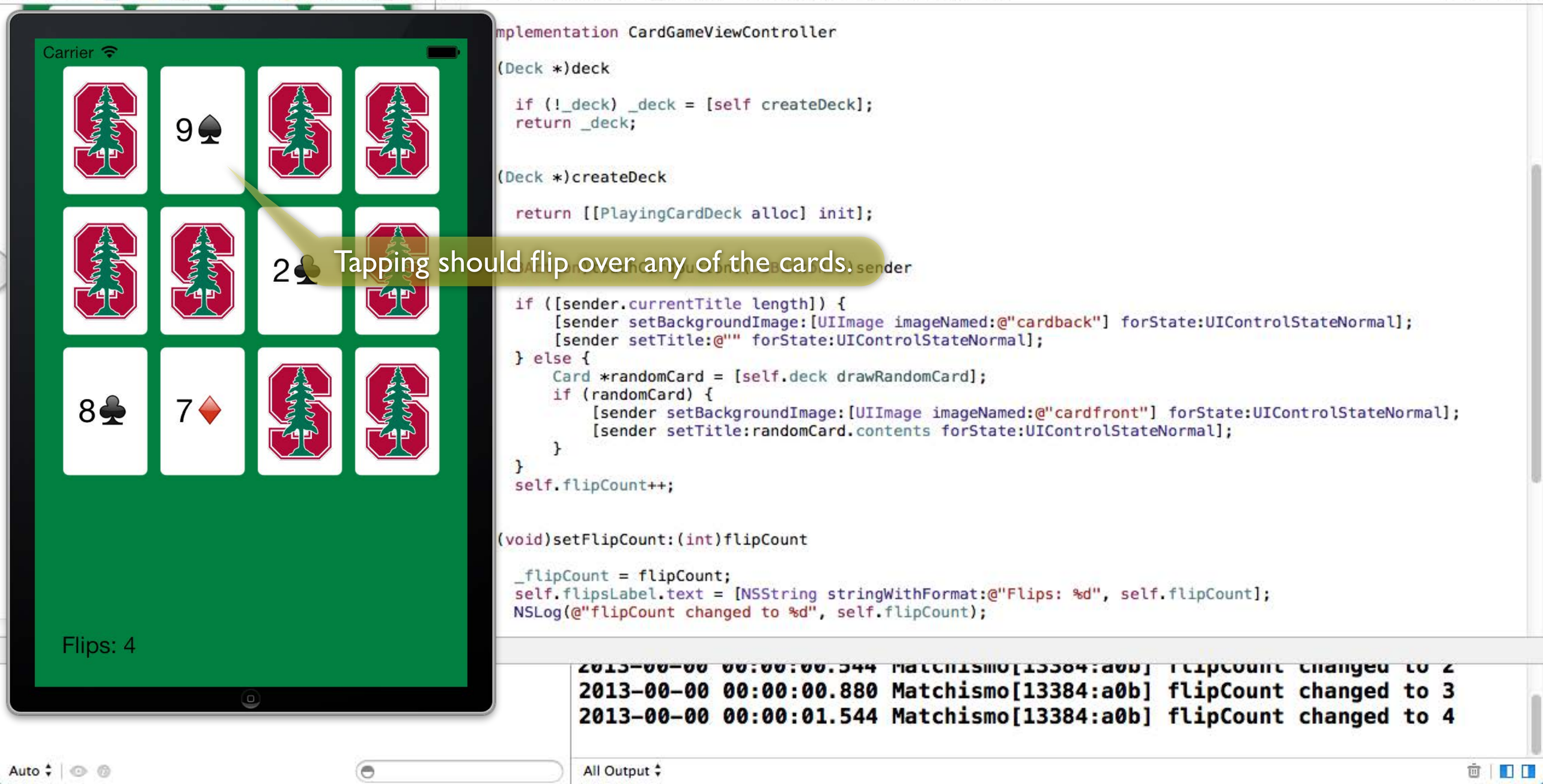
Card G... › View | Automatic › CardGameViewController.m › -touchCardButton:

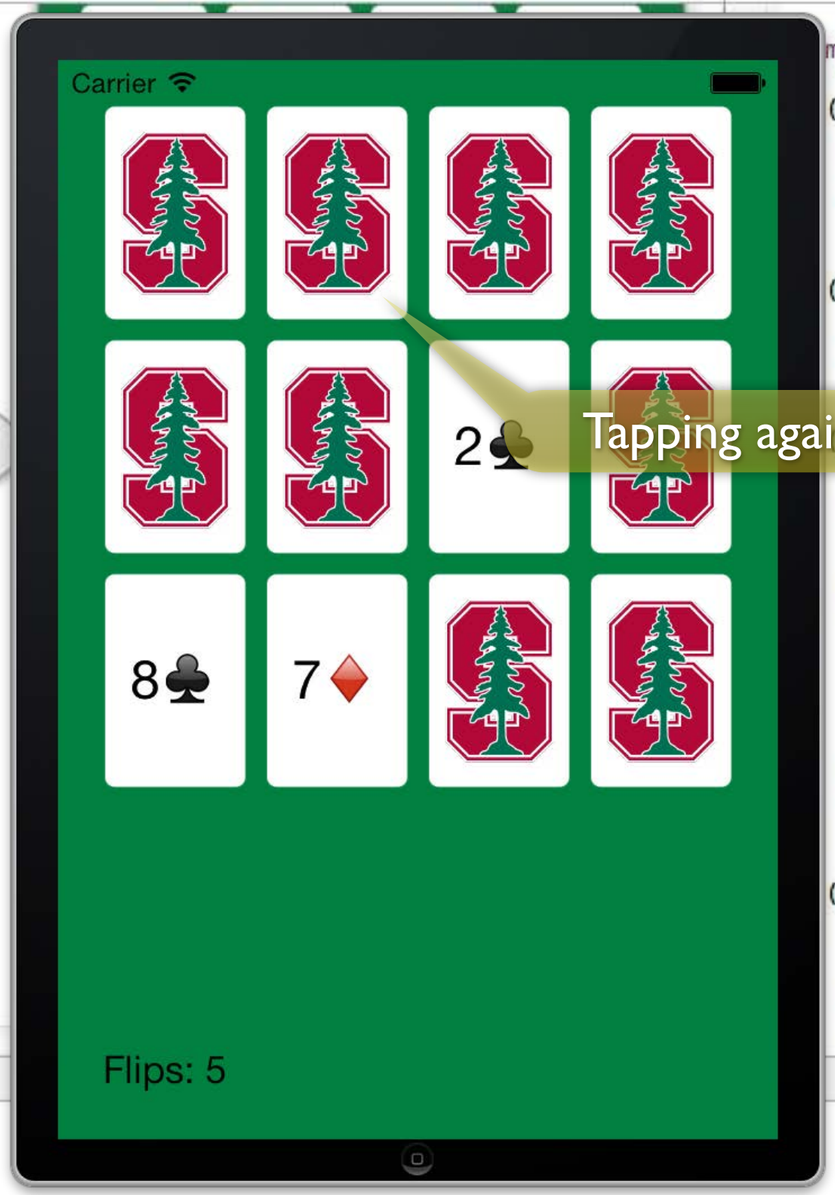


Let's Run and see what we've got so far.

```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;

(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];

(IBAction)touchCardButton:(UIButton *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;

(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

Auto | All Output

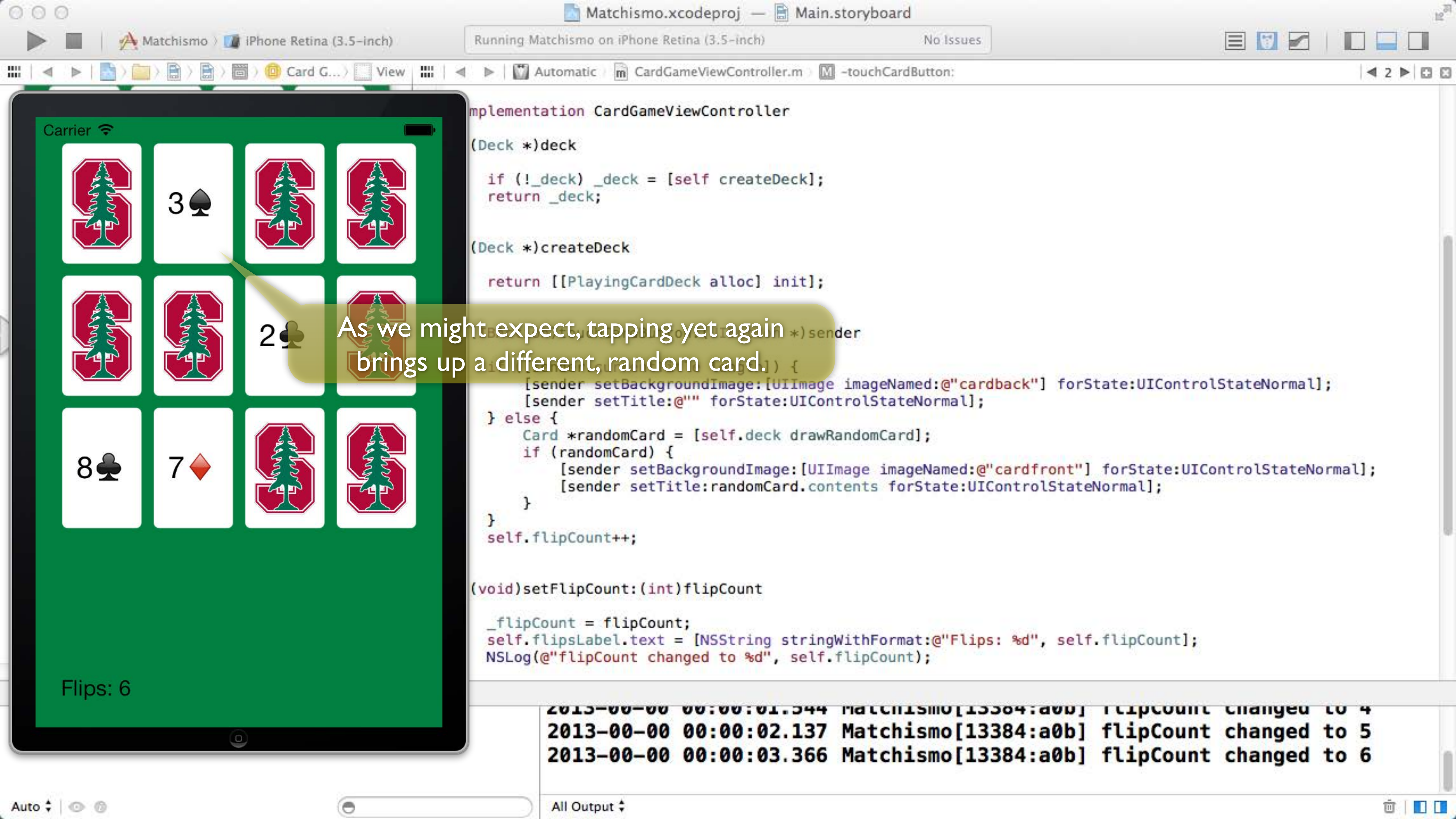

Matchismo.xcodeproj — Main.storyboard
Matchismo
iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G...
View
Automatic
CardGameViewController.m
-touchCardButton:
Carrier
9♠
2♣
8♣
7♦
Flips: 4
Tapping should flip over any of the cards.
mplementation CardGameViewController
(Deck *)deck
if (!_deck) _deck = [self createDeck];
return _deck;
(Deck *)createDeck
return [[PlayingCardDeck alloc] init];
sender
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
self.flipCount++;
(void)setFlipCount:(int)flipCount
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
2013-00-00 00:00:00.880 Matchismo[13384:a0b] flipCount changed to 3
2013-00-00 00:00:01.544 Matchismo[13384:a0b] flipCount changed to 4
Auto
All Output



Tapping again should flip it back down.

```
mplementation CardGameViewController

(Deck *)deck

  if (!_deck) _deck = [self createDeck];
  return _deck;


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];

                                                    *)sender

  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
          [sender setTitle:randomCard.contents forState:UIControlStateNormal];
      }
  }
  self.flipCount++;


(void)setFlipCount:(int)flipCount

  _flipCount = flipCount;
  self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
  NSLog(@"flipCount changed to %d", self.flipCount);
```

```
2013-00-00 00:00:01.544 Matchismo[13384:a0b] flipCount changed to 4
2013-00-00 00:00:02.137 Matchismo[13384:a0b] flipCount changed to 5
```


Matchismo.xcodeproj — Main.storyboard
Matchismo
iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G...
View
Automatic
CardGameViewController.m
-touchCardButton:
Carrier
3♠
2♣
8♣
7♦
Flips: 6
As we might expect, tapping yet again brings up a different, random card.
mplementation CardGameViewController
(Deck *)deck
if (!_deck) _deck = [self createDeck];
return _deck;
(Deck *)createDeck
return [[PlayingCardDeck alloc] init];
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
self.flipCount++;
(void)setFlipCount:(int)flipCount
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
2013-00-00 00:00:02.137 Matchismo[13384:a0b] flipCount changed to 5
2013-00-00 00:00:03.366 Matchismo[13384:a0b] flipCount changed to 6
Auto
All Output

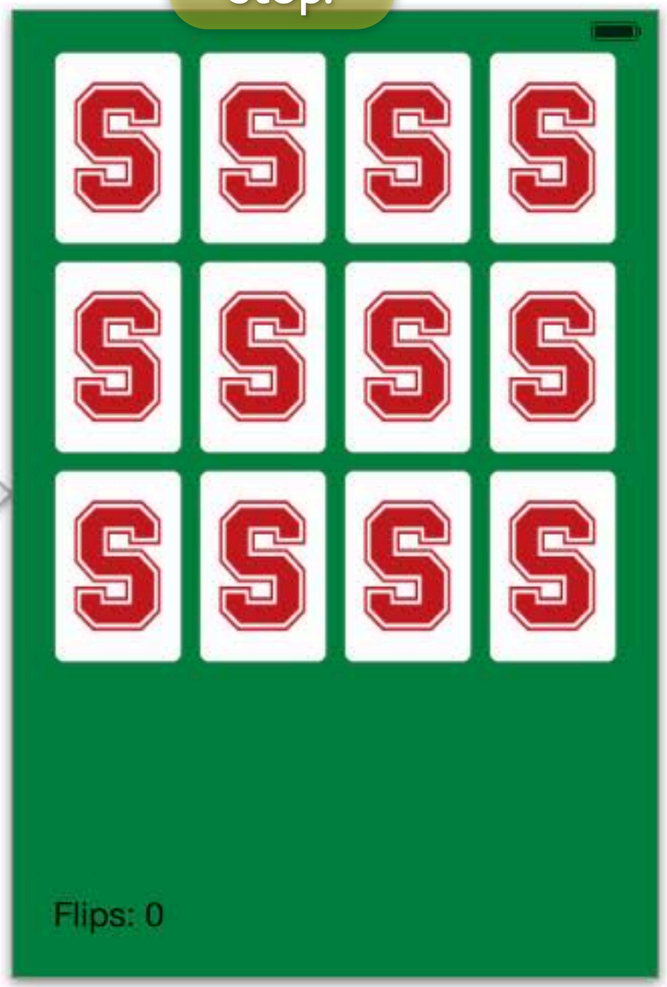
Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Stop.

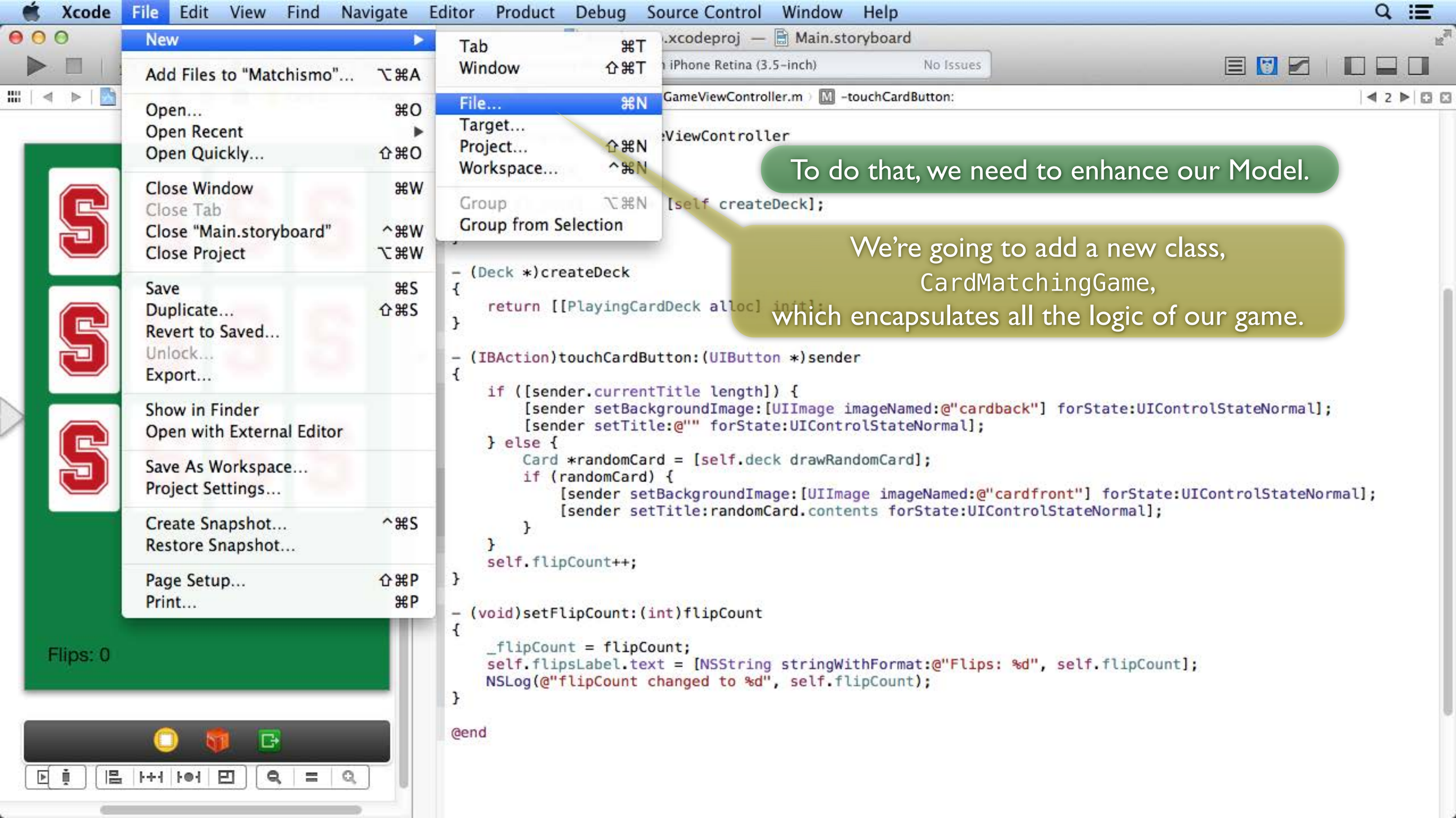
Card G... › View

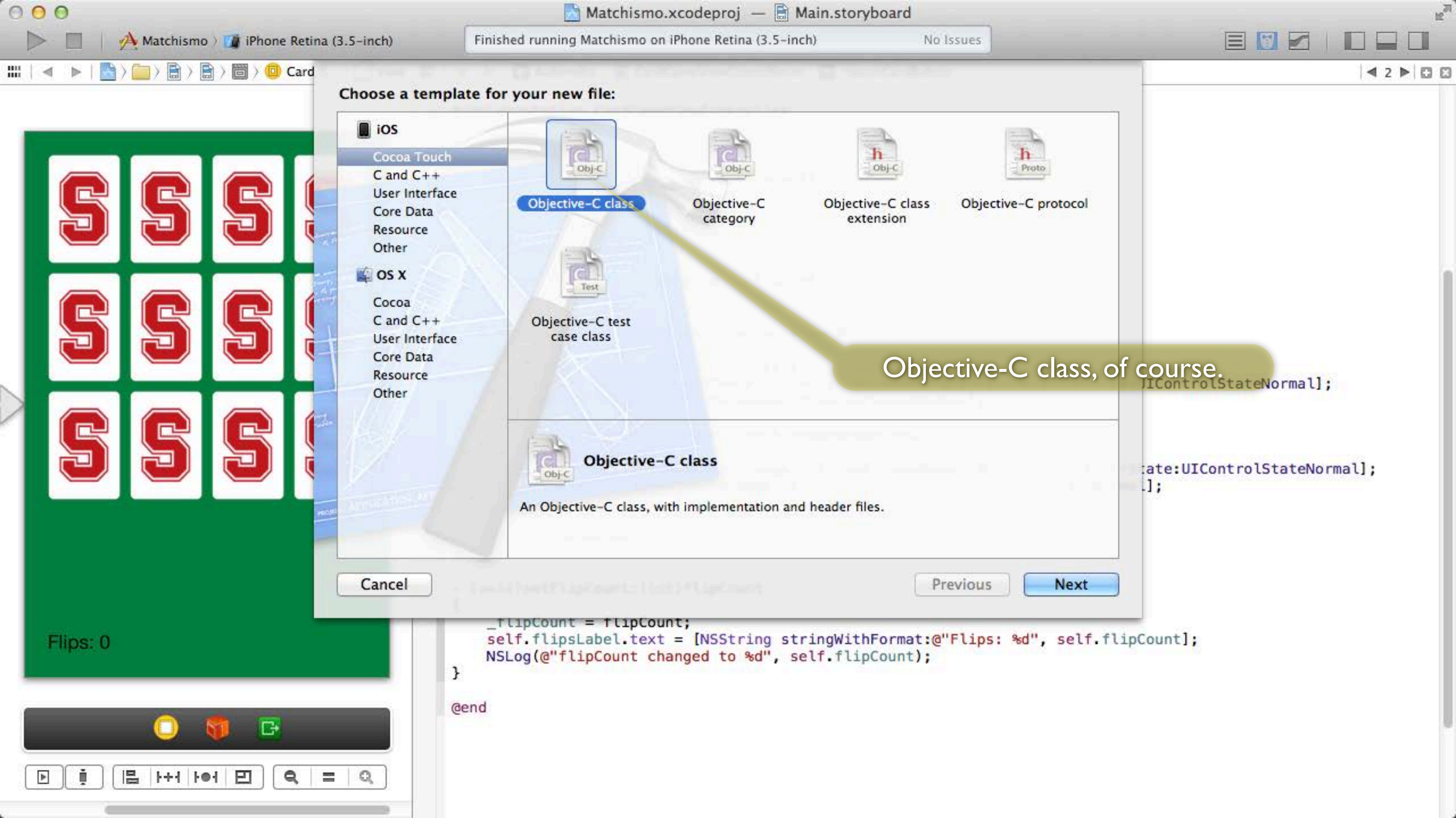
Automatic › CardGameViewController.m › -touchCardButton:

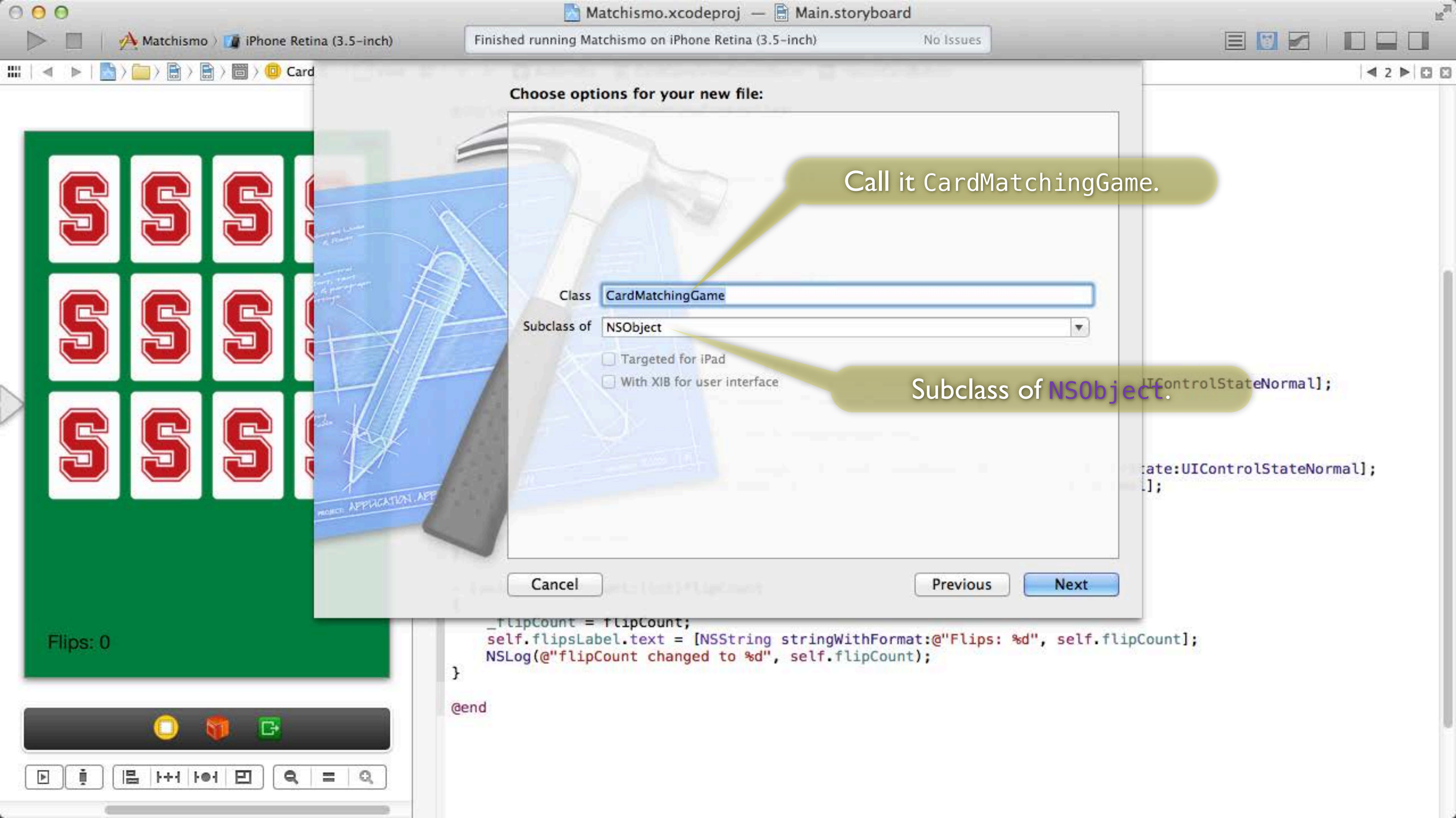


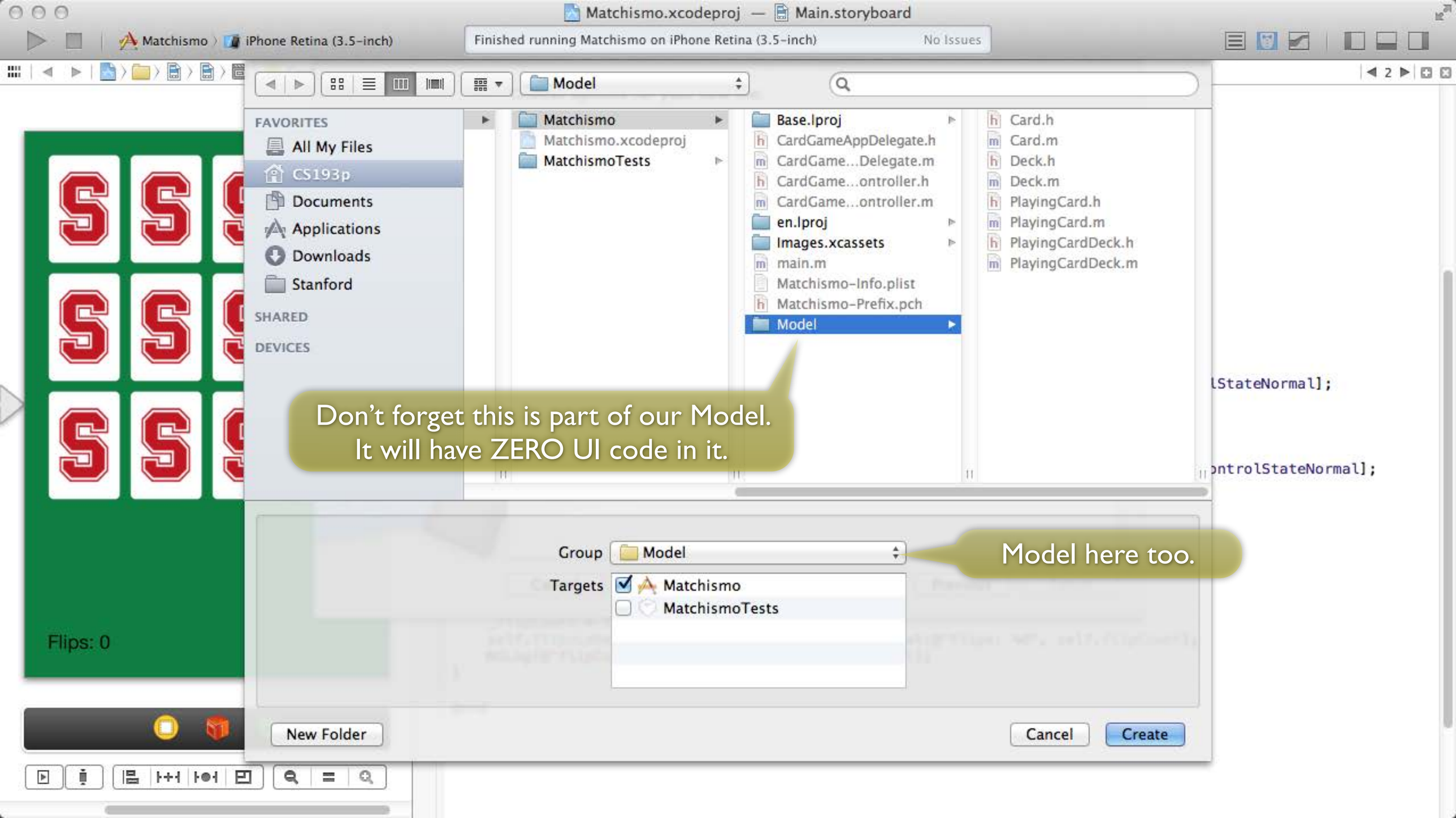
Now it's time to make a card matching game that actually matches cards!

```
@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)cre
{
    return [[PlayingC
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@end
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
New
Add Files to "Matchismo"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close "Main.storyboard" ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Create Snapshot... ^⌘S
Restore Snapshot...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
.xcodeproj — Main.storyboard
iPhone Retina (3.5-inch) No Issues
GameViewController.m -touchCardButton:
Flips: 0
To do that, we need to enhance our Model.
We're going to add a new class, CardMatchingGame, which encapsulates all the logic of our game.
ViewController
[self createDeck];
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
self.flipCount++;
}
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Choose a template for your new file:
iOS
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
Objective-C class
Objective-C category
Objective-C class extension
Objective-C protocol
Objective-C test case class
Objective-C class
An Objective-C class, with implementation and header files.
Cancel
Previous
Next
Objective-C class, of course.
Flips: 0
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Choose options for your new file:
Call it CardMatchingGame.
Class
CardMatchingGame
Subclass of
NSObject
Targeted for iPad
With XIB for user interface
Subclass of NSObject.
Cancel
Previous
Next
Flips: 0
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Model
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
SHARED
DEVICES
Matchismo
Matchismo.xcodeproj
MatchismoTests
Base.lproj
CardGameAppDelegate.h
CardGame...Delegate.m
CardGame...ontroller.h
CardGame...ontroller.m
en.lproj
Images.xcassets
main.m
Matchismo-Info.plist
Matchismo-Prefix.pch
Model
Card.h
Card.m
Deck.h
Deck.m
PlayingCard.h
PlayingCard.m
PlayingCardDeck.h
PlayingCardDeck.m
Don't forget this is part of our Model.
It will have ZERO UI code in it.
Group
Model
Model here too.
Targets
Matchismo
MatchismoTests
New Folder
Cancel
Create
Flips: 0
lStateNormal];
ontrolStateNormal];

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMat... › @interface CardMatchingGame

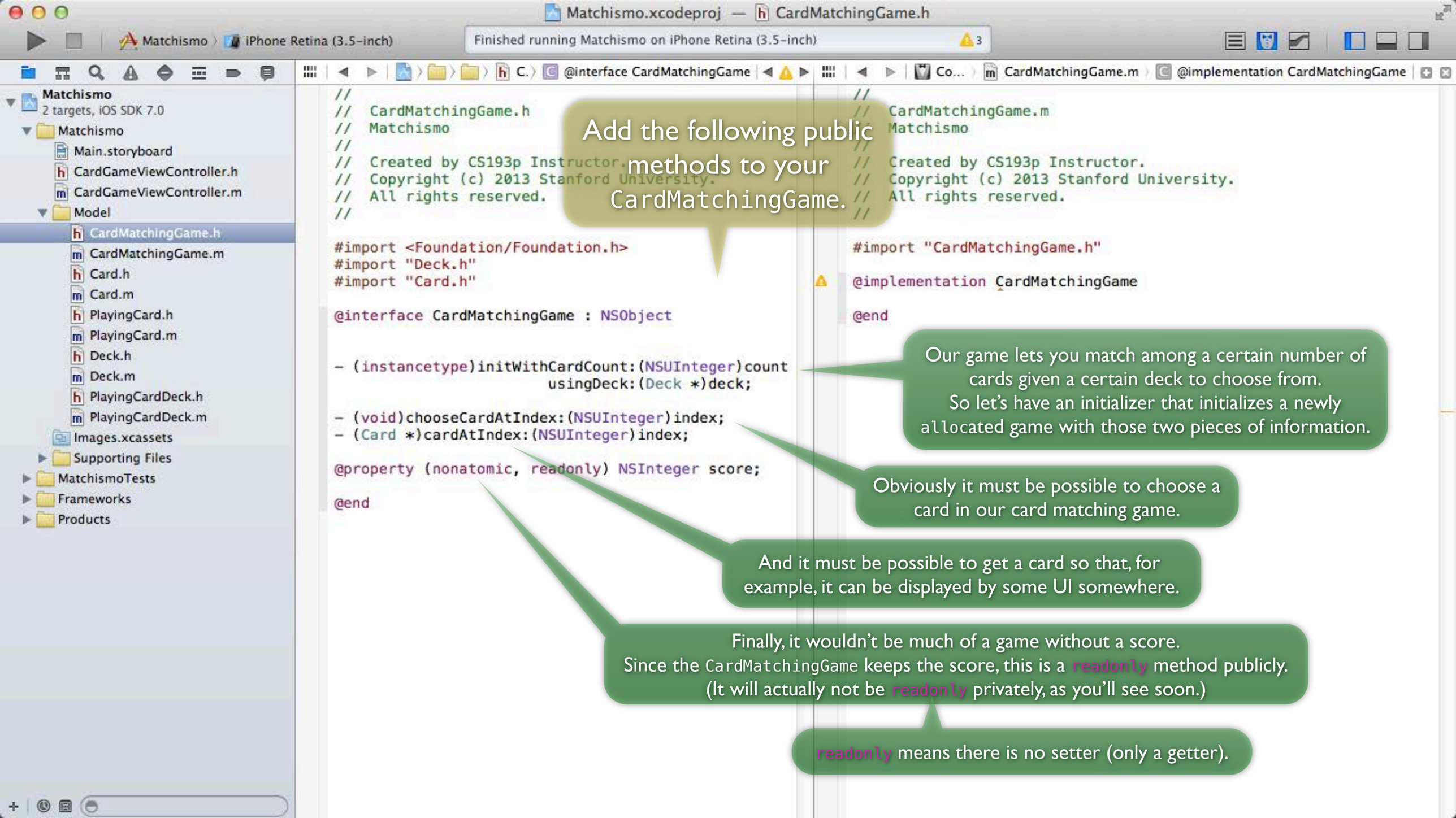
```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>

@interface CardMatchingGame : NSObject

@end
```

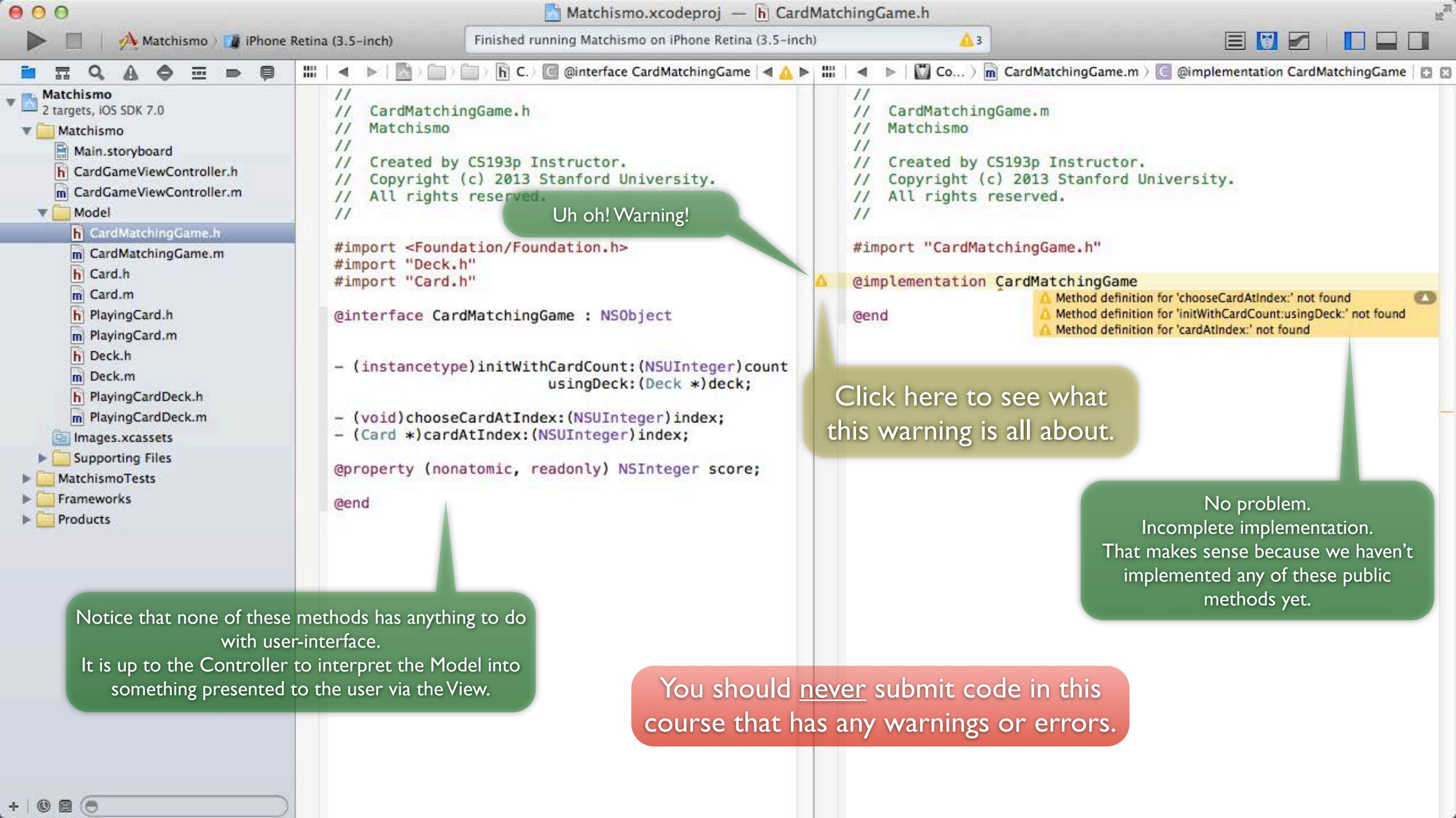
Co... › CardMatchingGame.m › @implementation CardMatchingGame

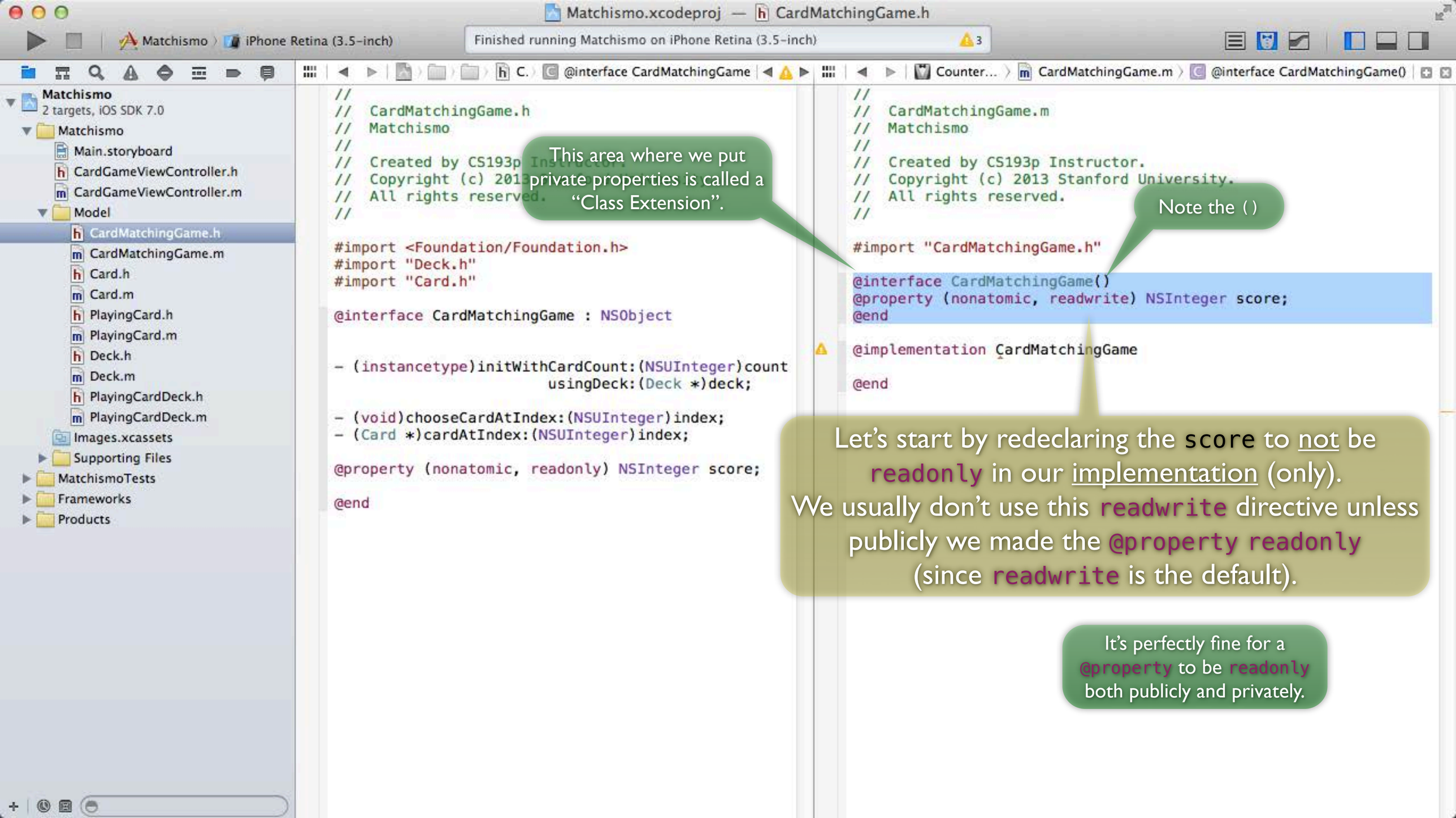
```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@implementation CardMatchingGame

@end
```

Hopefully your `CardMatchingGame` appeared here in the Model group. If not, you can simply drag it there.

Often the best way to start designing a new class is to fill out its `@interface` since that delineates what it is this class is responsible for doing.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
CardMatchingGame.h
Matchismo
Created by CS193p Instructor.
Copyright (c) 2013 Stanford University.
All rights reserved.
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
Co... › CardMatchingGame.m › @implementation CardMatchingGame
CardMatchingGame.m
Matchismo
Created by CS193p Instructor.
Copyright (c) 2013 Stanford University.
All rights reserved.
#import "CardMatchingGame.h"
@implementation CardMatchingGame
@end
Add the following public methods to your CardMatchingGame.
Our game lets you match among a certain number of cards given a certain deck to choose from.
So let's have an initializer that initializes a newly allocated game with those two pieces of information.
Obviously it must be possible to choose a card in our card matching game.
And it must be possible to get a card so that, for example, it can be displayed by some UI somewhere.
Finally, it wouldn't be much of a game without a score.
Since the CardMatchingGame keeps the score, this is a readonly method publicly.
(It will actually not be readonly privately, as you'll see soon.)
readonly means there is no setter (only a getter).


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
Uh oh! Warning!
Co... › CardMatchingGame.m › @implementation CardMatchingGame
//
// CardMatchingGame.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardMatchingGame.h"
@implementation CardMatchingGame
Method definition for 'chooseCardAtIndex:' not found
Method definition for 'initWithCardCount:usingDeck:' not found
Method definition for 'cardAtIndex:' not found
@end
Click here to see what this warning is all about.
No problem.
Incomplete implementation.
That makes sense because we haven't implemented any of these public methods yet.
Notice that none of these methods has anything to do with user-interface.
It is up to the Controller to interpret the Model into something presented to the user via the View.
You should never submit code in this course that has any warnings or errors.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
This area where we put private properties is called a "Class Extension".
Counter... › CardMatchingGame.m › @interface CardMatchingGame()
//
// CardMatchingGame.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardMatchingGame.h"
@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@end
@implementation CardMatchingGame
@end
Note the ()
Let's start by redeclaring the score to not be readonly in our implementation (only).
We usually don't use this readwrite directive unless publicly we made the @property readonly (since readwrite is the default).
It's perfectly fine for a @property to be readonly both publicly and privately.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 3

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

C. › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject


- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counter... › CardMatchingGame.m › @interface CardMatchingGame()

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

@end
```

Our game needs to keep track of the cards, so we need a private `@property` to do that.

Indeed there is no way to express in Objective-C that this array should only have `Card` objects in it. One might argue that that is a shortcoming. All we can do is be sure to comment what we intend.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 3

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

C. › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject


- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -cards

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

@end
```

Lazy instantiation!
Hopefully this is quite familiar to you by now.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 2

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

C › -initWithCardCount:usingDeck:

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject


- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterp... › CardMatchingGame.m › -initWithCardCount:usingDeck:

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck
{
    self = [super init];

    if (self) {

    }

    return self;
}

@end
```

Start off our initializer by letting our superclass have a chance to initialize itself (and checking for failure return of nil).

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 2

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

C. › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterp... › CardMatchingGame.m › -initWithCardCount:usingDeck:

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer

    if (self) {

    }

    return self;
}

@end
```

You should always comment which initializer is your designated initializer if it is different from your superclass's.

This is our class's designated initializer.
That means that it is not legally initialized unless this gets called at some point.
We must always call our superclass's designated initializer from our designated initializer
(if this were just a convenience initializer, we'd have to call our own designated initializer from it).

Classes can have multiple initializers,
but obviously only one designated initializer.

NSObject's designated initializer is `init`.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 2

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMatchingGame.h

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

CardMatchingGame.m

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer

    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            [self.cards addObject:card];
        }
    }

    return self;
}

@end
```

All we need to do to initialize our game is to iterate through the passed `count` of cards, `drawRandomCard` from the passed `deck`, then `addObject:` to our `NSMutableArray` of `cards` each time.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 2

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

C. › @interface CardMatchingGame

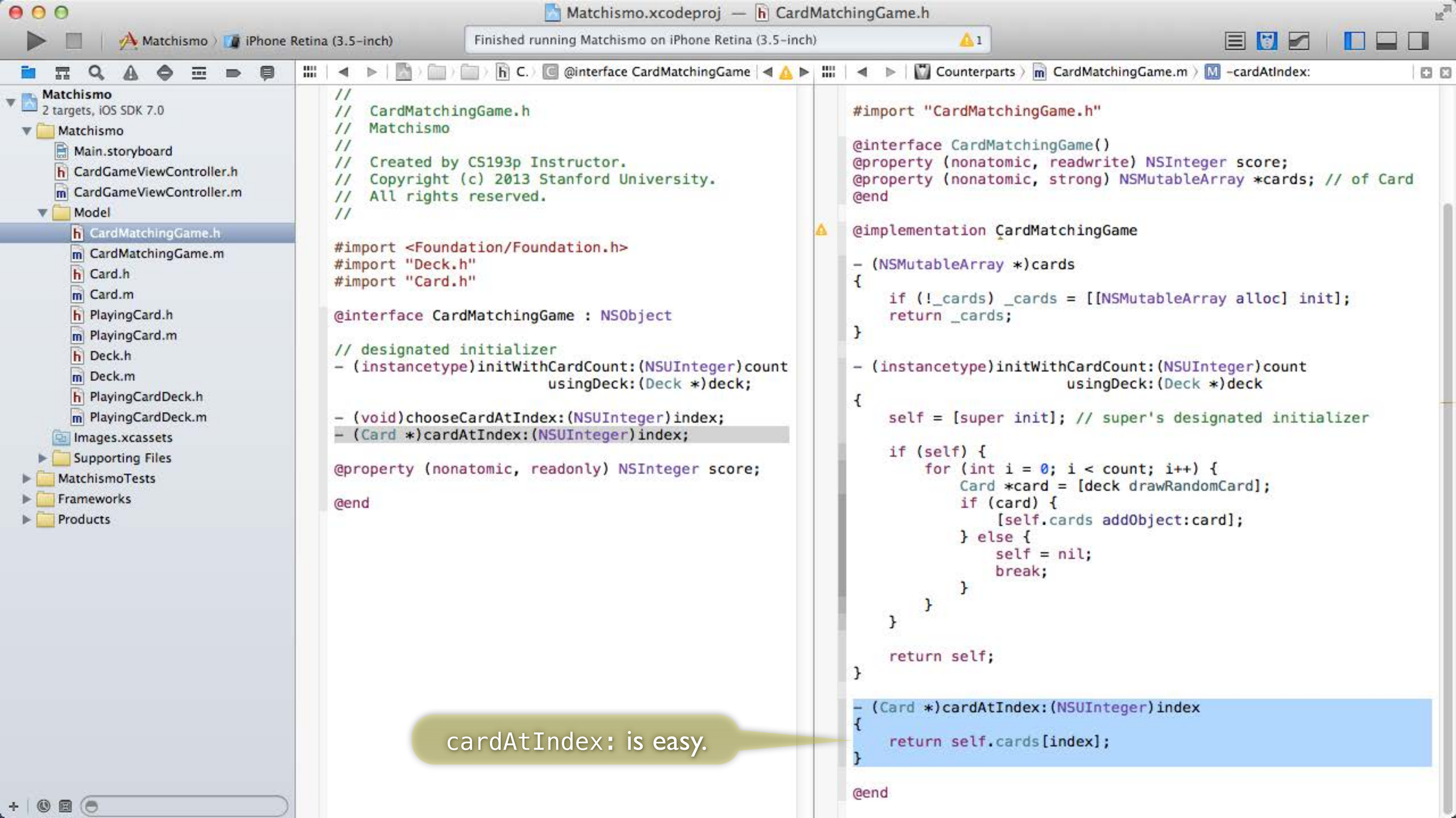
```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

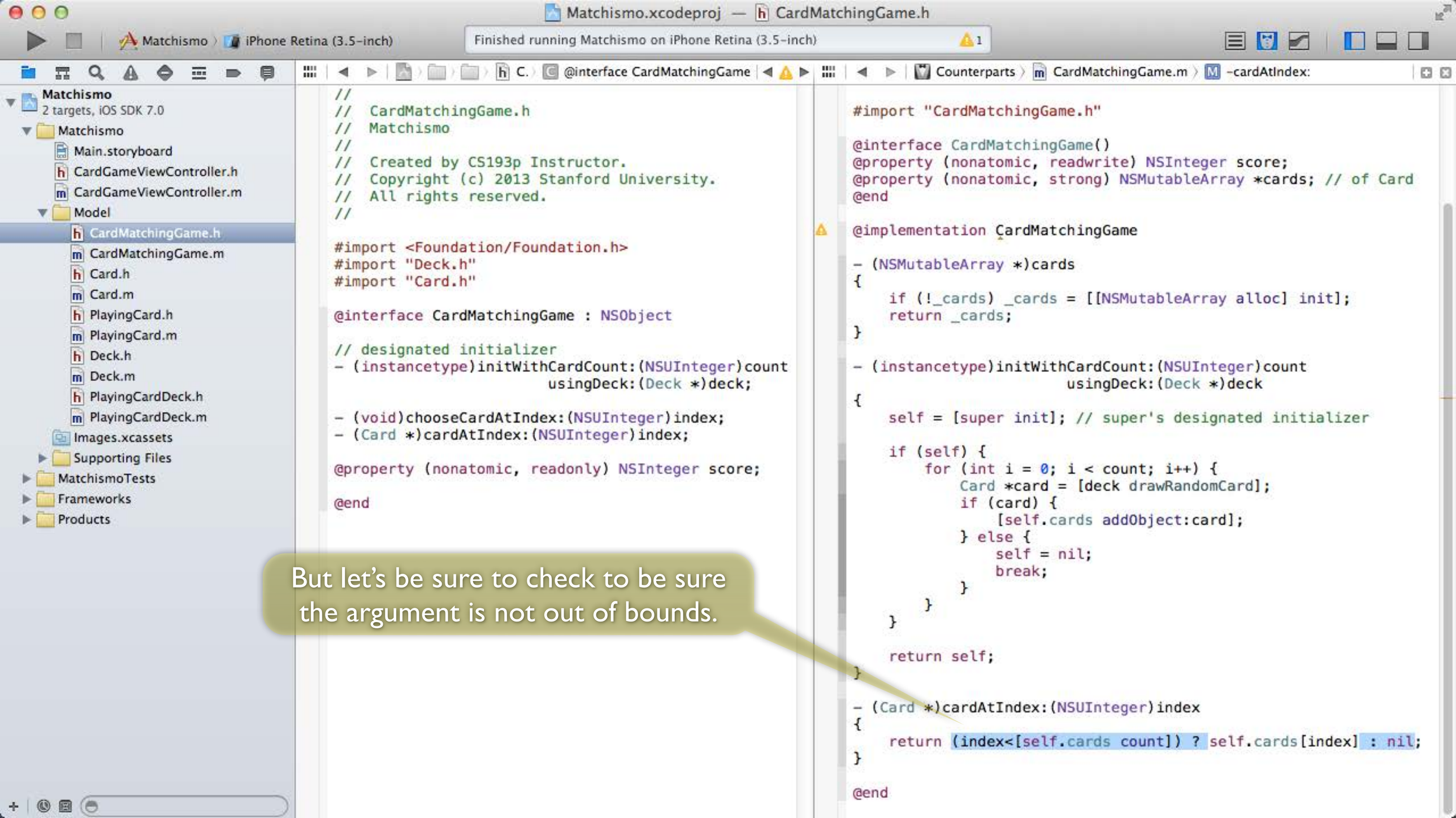
Counterp... › CardMatchingGame.m › -initWithCardCount:usingDeck:

```
//
//  CardMatchingGame.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardMatchingGame.h"

@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end

@implementation CardMatchingGame

- (NSMutableArray *)cards
{
    if (!_cards) _cards = [[NSMutableArray alloc] init];
    return _cards;
}

- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck
{
    self = [super init]; // super's designated initializer

    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            if (card) {
                [self.cards addObject:card];
            } else {
                self = nil;
                break;
            }
        }
    }

    return self;
}
```

Adding `nil` to an `NSMutableArray` will **crash** your program. Let's protect against this!

Note that we will return `nil` if we cannot initialize properly given the arguments passed.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
Counterparts › CardMatchingGame.m › -cardAtIndex:
#import "CardMatchingGame.h"
@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end
@implementation CardMatchingGame
- (NSMutableArray *)cards
{
if (!_cards) _cards = [[NSMutableArray alloc] init];
return _cards;
}
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck
{
self = [super init]; // super's designated initializer
if (self) {
for (int i = 0; i < count; i++) {
Card *card = [deck drawRandomCard];
if (card) {
[self.cards addObject:card];
} else {
self = nil;
break;
}
}
}
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return self.cards[index];
}
@end
cardAtIndex: is easy.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
But let's be sure to check to be sure the argument is not out of bounds.
Counterparts › CardMatchingGame.m › -cardAtIndex:
#import "CardMatchingGame.h"
@interface CardMatchingGame()
@property (nonatomic, readwrite) NSInteger score;
@property (nonatomic, strong) NSMutableArray *cards; // of Card
@end
@implementation CardMatchingGame
- (NSMutableArray *)cards
{
if (!_cards) _cards = [[NSMutableArray alloc] init];
return _cards;
}
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck
{
self = [super init]; // super's designated initializer
if (self) {
for (int i = 0; i < count; i++) {
Card *card = [deck drawRandomCard];
if (card) {
[self.cards addObject:card];
} else {
self = nil;
break;
}
}
}
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return (index<[self.cards count]) ? self.cards[index] : nil;
}
@end

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

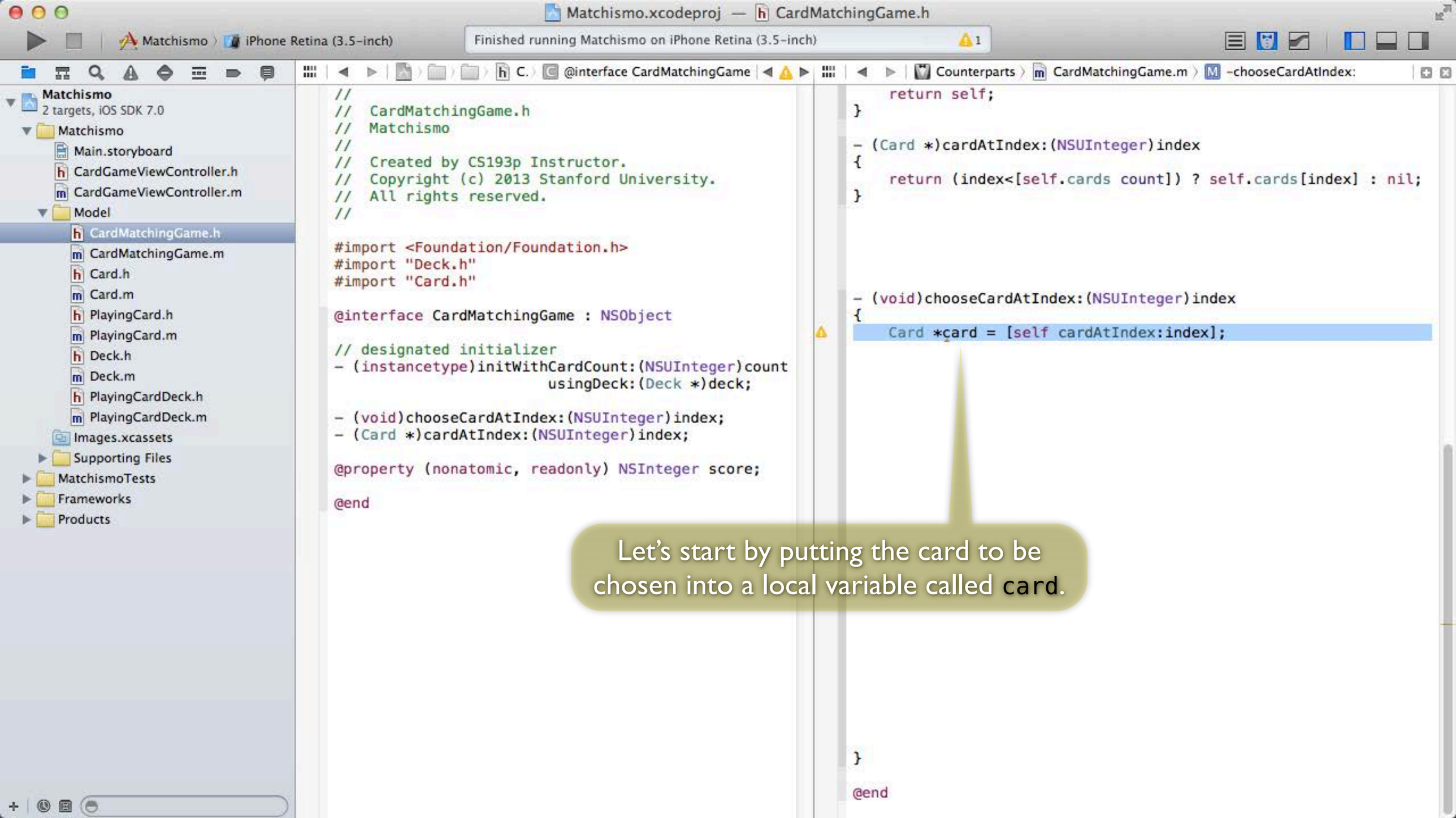
CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

- (void)chooseCardAtIndex:(NSUInteger)index
{

}

@end
```

Next comes `chooseCardAtIndex:`.
This contains the primary logic of this class.
We'll make some space to work here.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
Counterparts › CardMatchingGame.m › -chooseCardAtIndex:
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return (index<[self.cards count]) ? self.cards[index] : nil;
}
- (void)chooseCardAtIndex:(NSUInteger)index
{
Card *card = [self cardAtIndex:index];
}
@end
Let's start by putting the card to be chosen into a local variable called card.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}



- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {

    }
}

@end
```

We will only allow unmatched cards to be chosen (i.e. once a card is matched, it's "out of the game").

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}



- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards



            card.chosen = YES;
        }
    }
}

@end
```

If the card is already chosen, we'll "unchoose" it (so really this method is more like "toggle chosen state of card").

Otherwise, it is being chosen and so we need to match it against other chosen cards.

And then we'll mark it as chosen.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

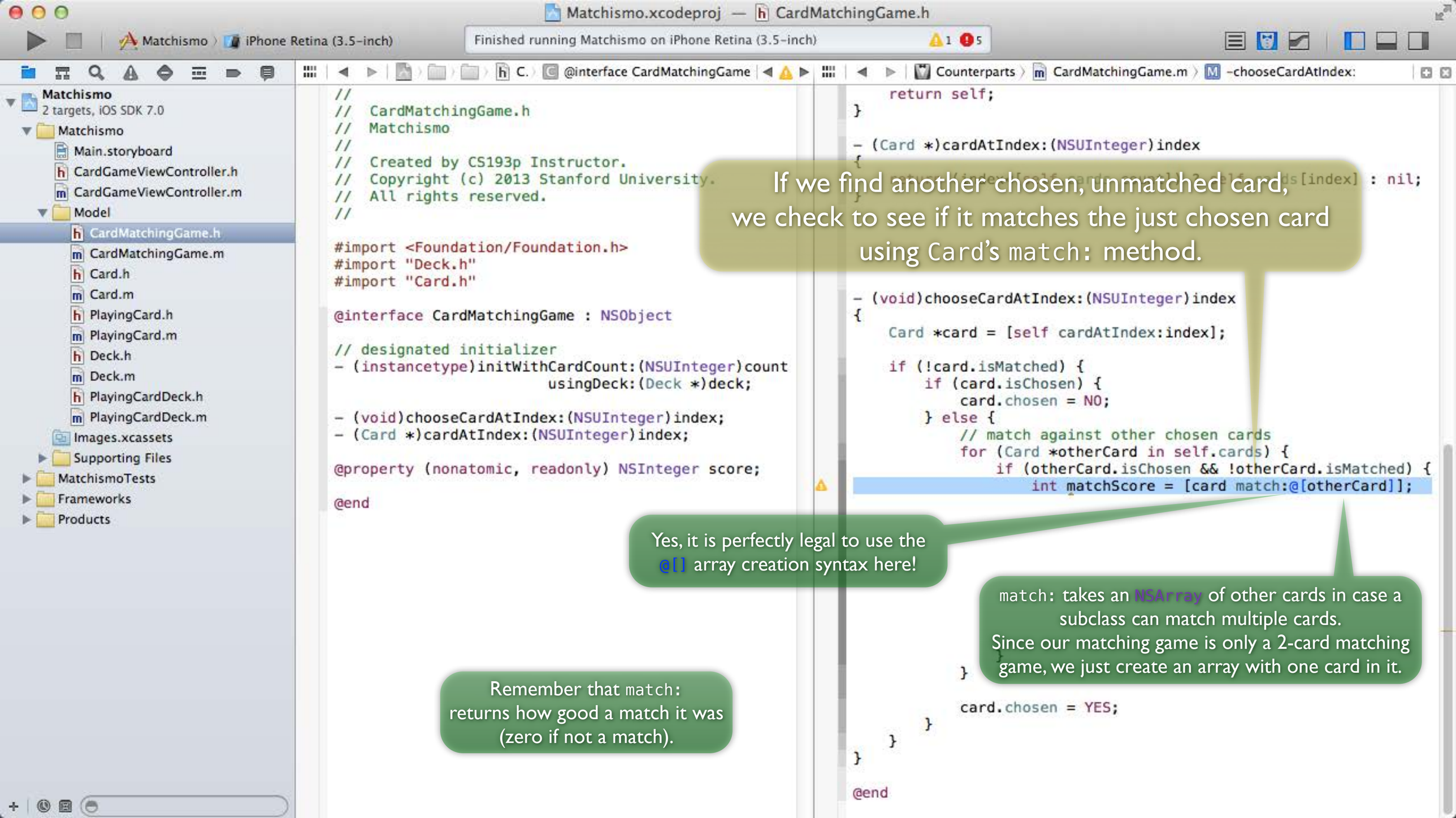
CardMat… › @interface CardMatchingGame

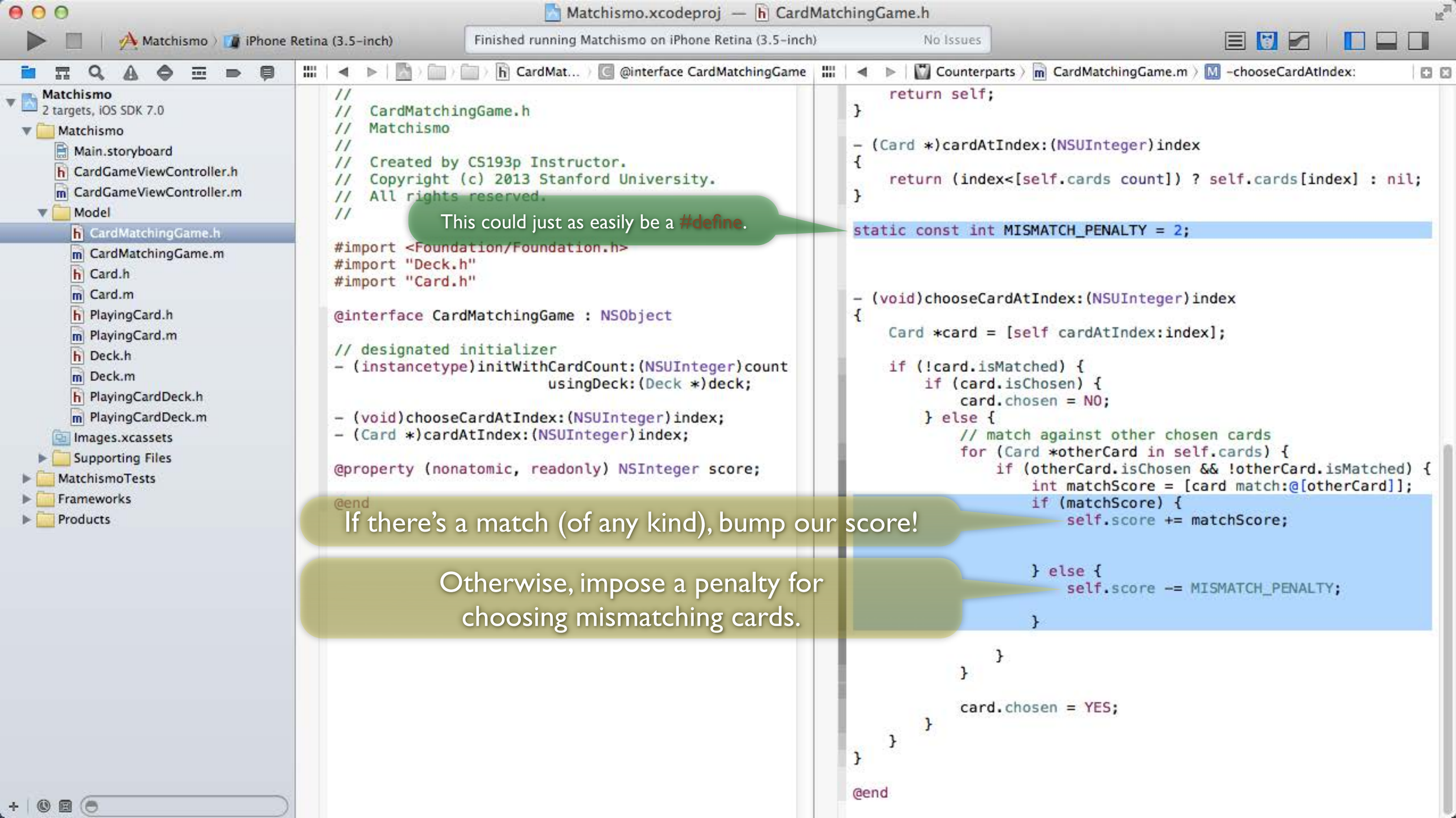
```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}



- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {

                }
            }

            card.chosen = YES;
        }
    }
}

@end
```

So we'll just iterate through all the cards in the game, looking for ones that are unmatched and already chosen.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
C. › @interface CardMatchingGame
Counterparts › CardMatchingGame.m › -chooseCardAtIndex:
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
}
- (void)chooseCardAtIndex:(NSUInteger)index
{
Card *card = [self cardAtIndex:index];
if (!card.isMatched) {
if (card.isChosen) {
card.chosen = NO;
} else {
// match against other chosen cards
for (Card *otherCard in self.cards) {
if (otherCard.isChosen && !otherCard.isMatched) {
int matchScore = [card match:@[otherCard]];
}
card.chosen = YES;
}
}
}
@end
If we find another chosen, unmatched card, we check to see if it matches the just chosen card using Card's match: method.
Yes, it is perfectly legal to use the @[] array creation syntax here!
match: takes an NSArray of other cards in case a subclass can match multiple cards.
Since our matching game is only a 2-card matching game, we just create an array with one card in it.
Remember that match: returns how good a match it was (zero if not a match).


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardMat... › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
This could just as easily be a #define.
If there's a match (of any kind), bump our score!
Otherwise, impose a penalty for choosing mismatching cards.
Counterparts › CardMatchingGame.m › -chooseCardAtIndex:
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return (index<[self.cards count]) ? self.cards[index] : nil;
}
static const int MISMATCH_PENALTY = 2;
- (void)chooseCardAtIndex:(NSUInteger)index
{
Card *card = [self cardAtIndex:index];
if (!card.isMatched) {
if (card.isChosen) {
card.chosen = NO;
} else {
// match against other chosen cards
for (Card *otherCard in self.cards) {
if (otherCard.isChosen && !otherCard.isMatched) {
int matchScore = [card match:@[otherCard]];
if (matchScore) {
self.score += matchScore;
} else {
self.score -= MISMATCH_PENALTY;
}
}
}
card.chosen = YES;
}
}
}
@end

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

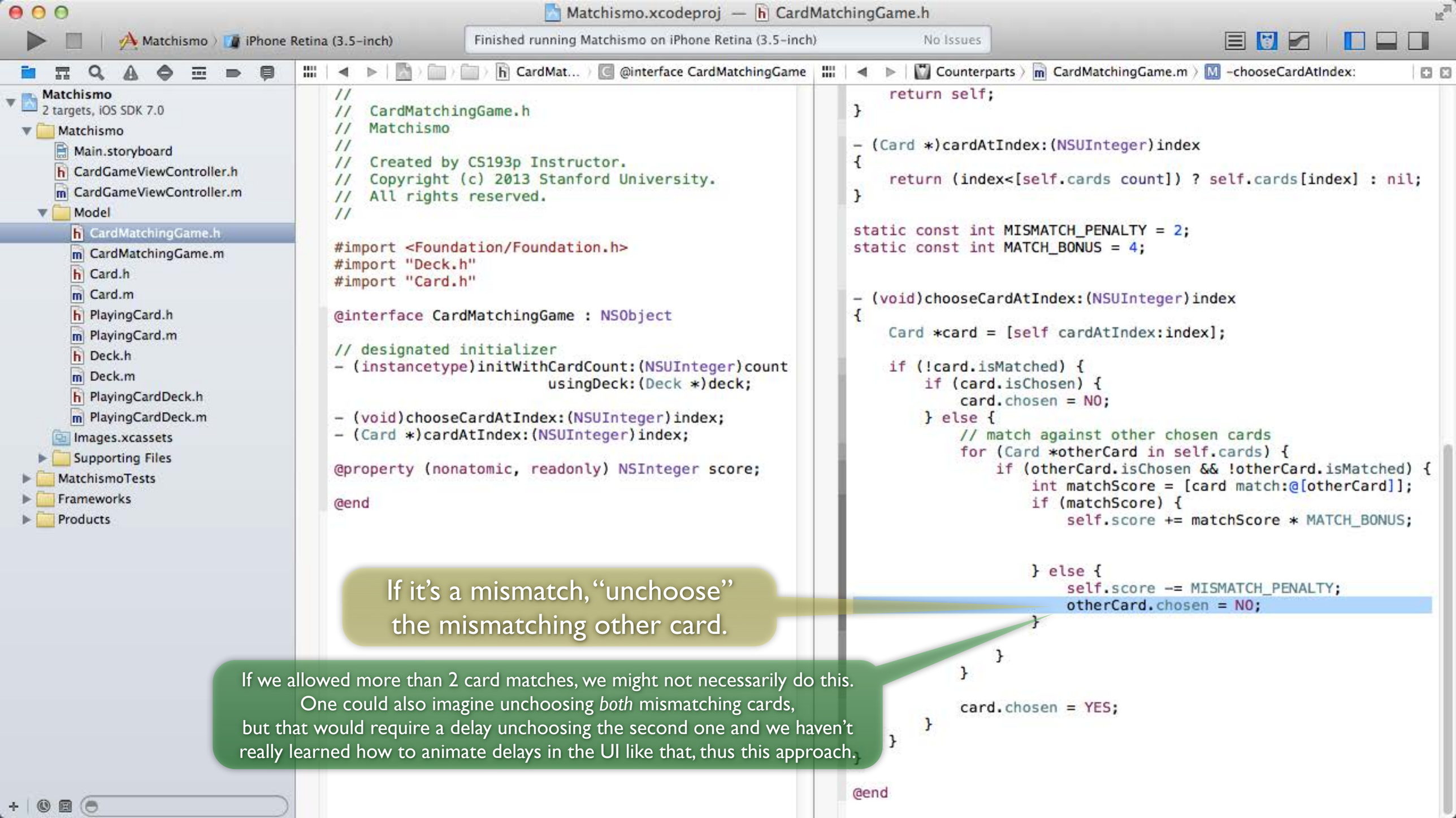
CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;


- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;

                    } else {
                        self.score -= MISMATCH_PENALTY;

                    }

                }
            }

            card.chosen = YES;
        }
    }
}

@end
```

We can even give a bonus to matches if we want.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardMat... › @interface CardMatchingGame
Counterparts › CardMatchingGame.m › -chooseCardAtIndex:
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return (index<[self.cards count]) ? self.cards[index] : nil;
}
static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
- (void)chooseCardAtIndex:(NSUInteger)index
{
Card *card = [self cardAtIndex:index];
if (!card.isMatched) {
if (card.isChosen) {
card.chosen = NO;
} else {
// match against other chosen cards
for (Card *otherCard in self.cards) {
if (otherCard.isChosen && !otherCard.isMatched) {
int matchScore = [card match:@[otherCard]];
if (matchScore) {
self.score += matchScore * MATCH_BONUS;
} else {
self.score -= MISMATCH_PENALTY;
otherCard.chosen = NO;
}
}
}
card.chosen = YES;
}
}
@end
If it's a mismatch, "unchoose" the mismatching other card.
If we allowed more than 2 card matches, we might not necessarily do this.
One could also imagine unchoosing both mismatching cards,
but that would require a delay unchoosing the second one and we haven't
really learned how to animate delays in the UI like that, thus this approach.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

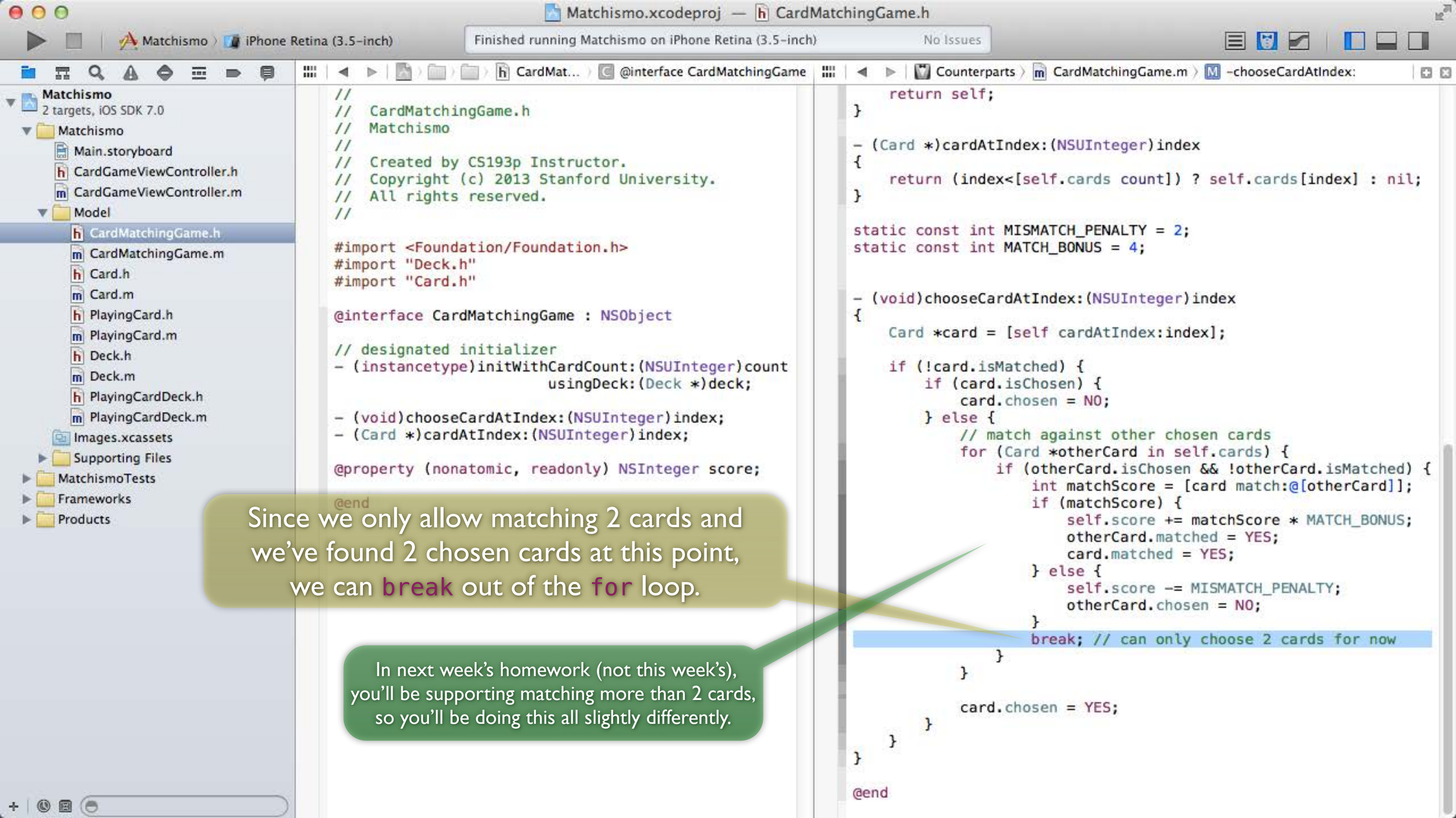
CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;


- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }

                }
            }

            card.chosen = YES;
        }
    }
}

@end
```

If it's a match, mark both matching cards as matched.


Matchismo.xcodeproj — CardMatchingGame.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardMat... › @interface CardMatchingGame
//
// CardMatchingGame.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"
@interface CardMatchingGame : NSObject
// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
usingDeck:(Deck *)deck;
- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;
@property (nonatomic, readonly) NSInteger score;
@end
Since we only allow matching 2 cards and we've found 2 chosen cards at this point, we can break out of the for loop.
In next week's homework (not this week's), you'll be supporting matching more than 2 cards, so you'll be doing this all slightly differently.
Counterparts › CardMatchingGame.m › -chooseCardAtIndex:
return self;
}
- (Card *)cardAtIndex:(NSUInteger)index
{
return (index<[self.cards count]) ? self.cards[index] : nil;
}
static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
- (void)chooseCardAtIndex:(NSUInteger)index
{
Card *card = [self cardAtIndex:index];
if (!card.isMatched) {
if (card.isChosen) {
card.chosen = NO;
} else {
// match against other chosen cards
for (Card *otherCard in self.cards) {
if (otherCard.isChosen && !otherCard.isMatched) {
int matchScore = [card match:@[otherCard]];
if (matchScore) {
self.score += matchScore * MATCH_BONUS;
otherCard.matched = YES;
card.matched = YES;
} else {
self.score -= MISMATCH_PENALTY;
otherCard.chosen = NO;
}
break; // can only choose 2 cards for now
}
}
card.chosen = YES;
}
}
}
@end

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
static const int COST_TO_CHOOSE = 1;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }
                    break; // can only choose 2 cards for now
                }
            }
            self.score -= COST_TO_CHOOSE;
            card.chosen = YES;
        }
    }
}

@end
```

Let's making choosing cards not be "free" by imposing a cost to choose.

Matchismo.xcodeproj — CardMatchingGame.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

CardMat… › @interface CardMatchingGame

```
//
//  CardMatchingGame.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//
#import <Foundation/Foundation.h>
#import "Deck.h"
#import "Card.h"

@interface CardMatchingGame : NSObject

// designated initializer
- (instancetype)initWithCardCount:(NSUInteger)count
                        usingDeck:(Deck *)deck;

- (void)chooseCardAtIndex:(NSUInteger)index;
- (Card *)cardAtIndex:(NSUInteger)index;

@property (nonatomic, readonly) NSInteger score;

@end
```

Counterparts › CardMatchingGame.m › -chooseCardAtIndex:

```
    return self;
}

- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
static const int COST_TO_CHOOSE = 1;

- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {
                    int matchScore = [card match:@[otherCard]];
                    if (matchScore) {
                        self.score += matchScore * MATCH_BONUS;
                        otherCard.matched = YES;
                        card.matched = YES;
                    } else {
                        self.score -= MISMATCH_PENALTY;
                        otherCard.chosen = NO;
                    }
                    break; // can only choose 2 cards for now
                }
            }
            self.score -= COST_TO_CHOOSE;
            card.chosen = YES;
        }
    }
}

@end
```

That's it!
Pretty simple really.
But there's one more change we'll want to make to our Model.
Specifically, `PlayingCard`'s `match:` algorithm.

Matchismo.xcodeproj — Card.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › Matc... › Model › Card.h › -match:

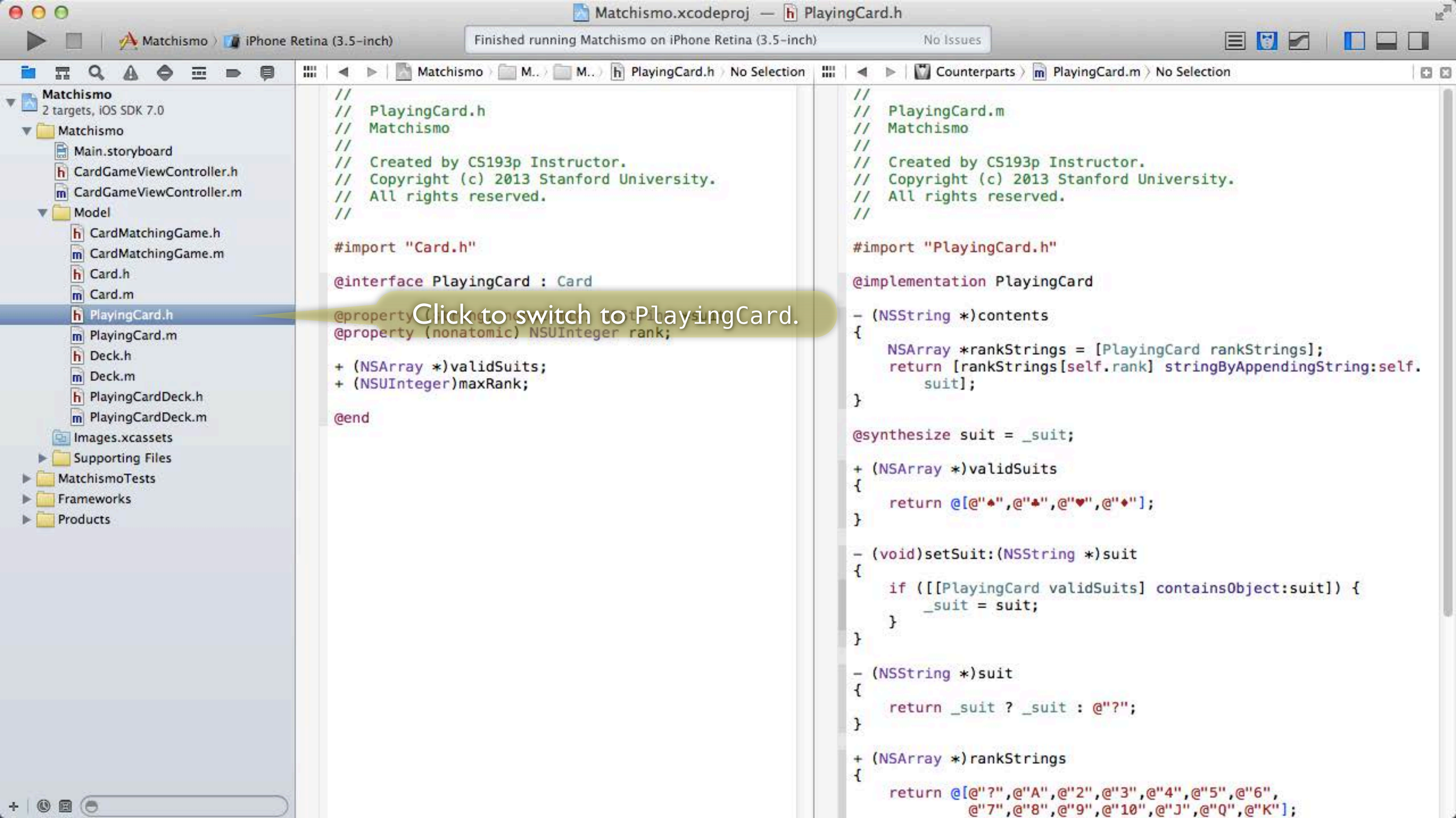
```
//
//  Card.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import ...

@int...

@property (s...tents;

@property (nonatomic, getter=isChosen) BOOL chosen;
@property (nonatomic, getter=isMatched) BOOL
    matched;

- (int)match:(NSArray *)otherCards;

@end
```

Counterparts › Card.m › -match:

```
//
//  Card.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@implementation Card

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    for (Card *card in otherCards) {
        if ([card.contents isEqualToString:self.contents]) {
            score = 1;
        }
    }

    return score;
}

@end
```

Click on `Card` in the Navigator so that we remind ourselves what its `match:` algorithm is.

`Card` matches only if the cards are exactly the same (that is to say, their `contents` `@property` values are equal). `PlayingCards` should match if the suit and/or rank is the same. Let's go to `PlayingCard` and override `Card`'s implementation of `match:` to make this so.


Matchismo.xcodeproj — PlayingCard.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › M.. › M.. › PlayingCard.h › No Selection
//
// PlayingCard.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@interface PlayingCard : Card
@property (nonatomic) NSUInteger rank;
+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;
@end
Click to switch to PlayingCard.
Counterparts › PlayingCard.m › No Selection
//
// PlayingCard.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "PlayingCard.h"
@implementation PlayingCard
- (NSString *)contents
{
NSArray *rankStrings = [PlayingCard rankStrings];
return [rankStrings[self.rank] stringByAppendingString:self.
suit];
}
@synthesize suit = _suit;
+ (NSArray *)validSuits
{
return @[@"♠",@"♣",@"♥",@"♦"];
}
- (void)setSuit:(NSString *)suit
{
if ([[PlayingCard validSuits] containsObject:suit]) {
_suit = suit;
}
}
- (NSString *)suit
{
return _suit ? _suit : @"?";
}
+ (NSArray *)rankStrings
{
return @[@"?",@"A",@"2",@"3",@"4",@"5",@"6",
@"7",@"8",@"9",@"10",@"J",@"Q",@"K"];

Matchismo.xcodeproj — PlayingCard.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › M.. › M.. › PlayingCard.h › No Selection

```
//
//  PlayingCard.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

Note that even though `PlayingCard` is overriding its superclass's implementation of a method (`match:`), it is not required to redeclare `match:` in its header file. Generally we do not redeclare overridden methods.

Counterparts › PlayingCard.m › -match:

Add an implementation for `match:`.

```
//
//  PlayingCard.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.
        suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠",@"♣",@"♥",@"♦"];
}
```

Often a subclass's implementation of a method will call its superclass's implementation by invoking `super` (e.g. `[super match:...]`), but `PlayingCard` has its own, standalone implementation of this method and thus does not need to call `super`'s implementation.

Matchismo.xcodeproj — PlayingCard.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › M.. › M.. › PlayingCard.h › No Selection

```
//
//  PlayingCard.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

Counterparts › PlayingCard.m › -match:

```
//
//  PlayingCard.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    if ([otherCards count] == 1) {



    }

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.
        suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠",@"♣",@"♥",@"♦"];
}
```

First, we will only match a single other card (next week's homework assignment will have to do better than this).

Matchismo.xcodeproj — PlayingCard.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 1

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchi… › PlayingCard.h › No Selection

```
//
//  PlayingCard.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

Counterparts › PlayingCard.m › -match:

```
//
//  PlayingCard.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    if ([otherCards count] == 1) {
        PlayingCard *otherCard = [otherCards firstObject];



    }

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.
        suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠",@"♣",@"♥",@"♦"];
}
```

Let's get the card in the array
(there will only be one card in the array
if we got this far).

`firstObject` is an `NSArray` method. It is just like `[array objectAtIndex:0]`
except that it will not crash if the array is empty
(it will just return `nil`).
Convenient.

Matchismo.xcodeproj — PlayingCard.h

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

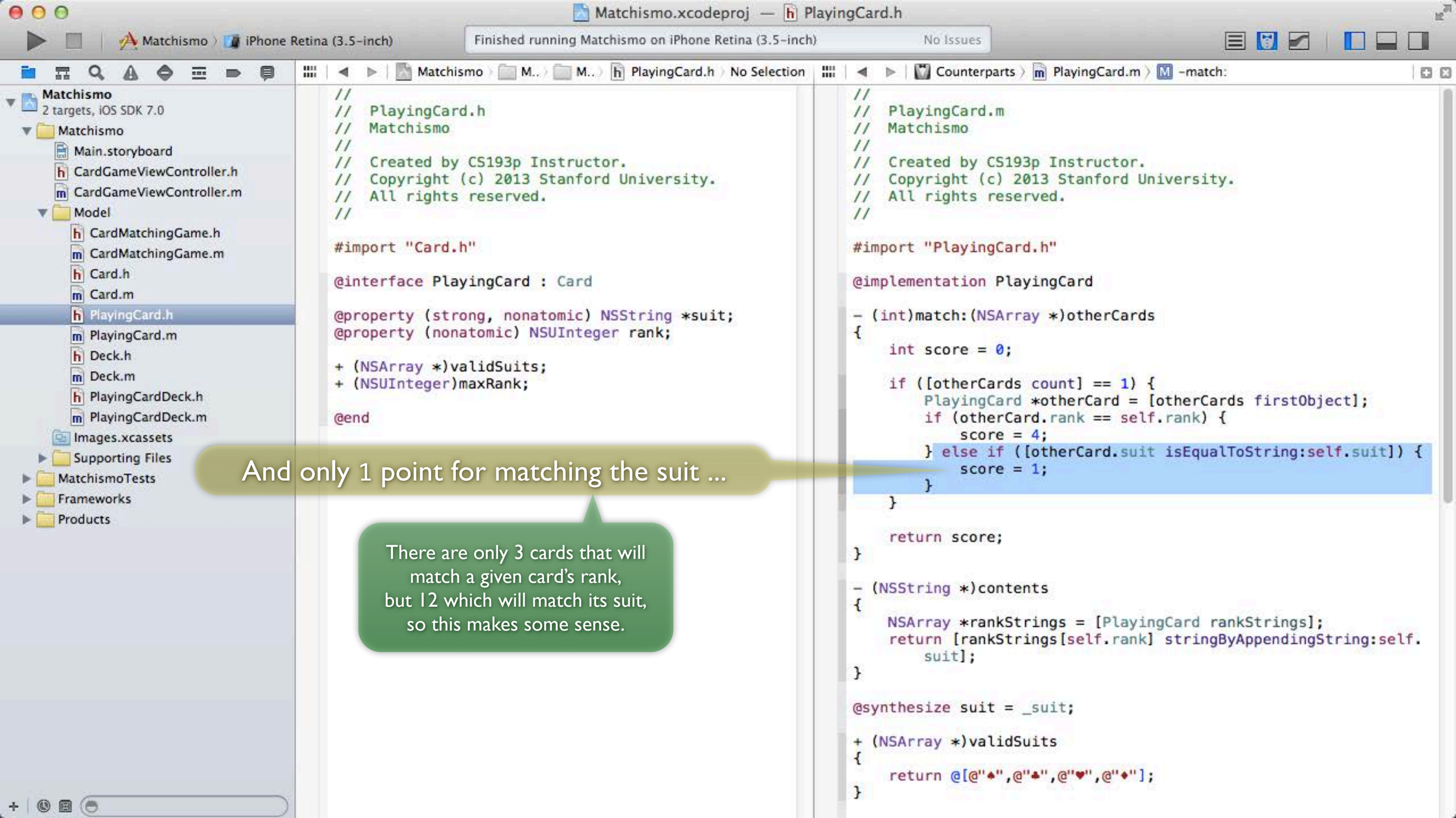
Matchismo › M.. › M.. › PlayingCard.h › No Selection

```
//
//  PlayingCard.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@interface PlayingCard : Card

@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;

+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;

@end
```

Give 4 points for matching the rank ...

Counterparts › PlayingCard.m › -match:

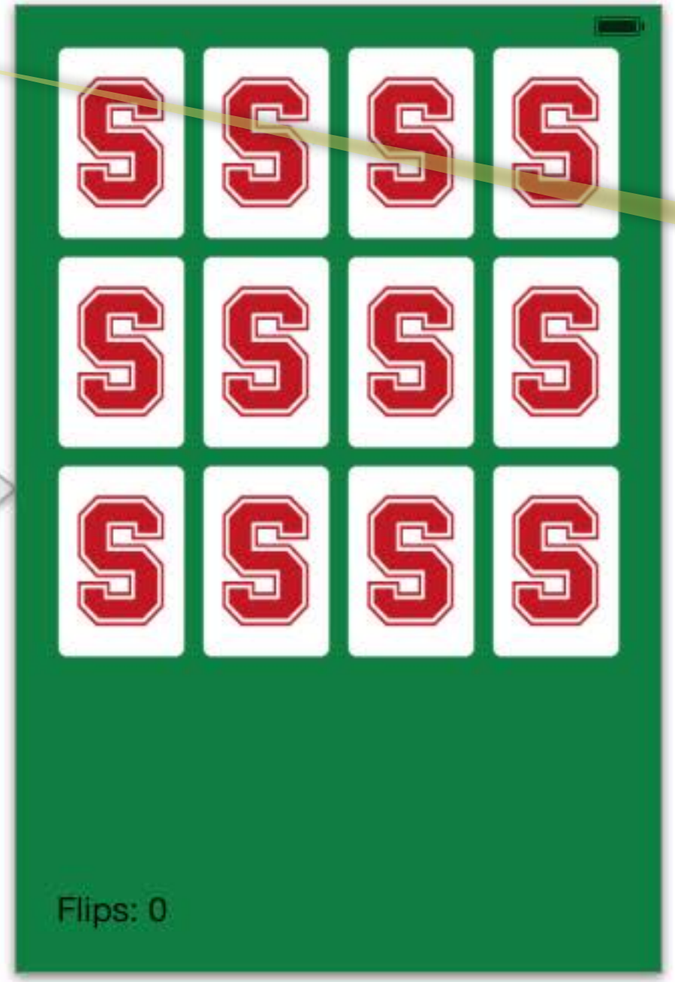
```
//
//  PlayingCard.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "PlayingCard.h"

@implementation PlayingCard

- (int)match:(NSArray *)otherCards
{
    int score = 0;

    if ([otherCards count] == 1) {
        PlayingCard *otherCard = [otherCards firstObject];
        if (otherCard.rank == self.rank) {
            score = 4;


        }
    }

    return score;
}

- (NSString *)contents
{
    NSArray *rankStrings = [PlayingCard rankStrings];
    return [rankStrings[self.rank] stringByAppendingString:self.
        suit];
}

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    return @[@"♠",@"♣",@"♥",@"♦"];
}
```


Matchismo.xcodeproj — PlayingCard.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
CardMatchingGame.h
CardMatchingGame.m
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › M.. › M.. › PlayingCard.h › No Selection
//
// PlayingCard.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@interface PlayingCard : Card
@property (strong, nonatomic) NSString *suit;
@property (nonatomic) NSUInteger rank;
+ (NSArray *)validSuits;
+ (NSUInteger)maxRank;
@end
Counterparts › PlayingCard.m › -match:
//
// PlayingCard.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "PlayingCard.h"
@implementation PlayingCard
- (int)match:(NSArray *)otherCards
{
int score = 0;
if ([otherCards count] == 1) {
PlayingCard *otherCard = [otherCards firstObject];
if (otherCard.rank == self.rank) {
score = 4;
} else if ([otherCard.suit isEqualToString:self.suit]) {
score = 1;
}
}
return score;
}
- (NSString *)contents
{
NSArray *rankStrings = [PlayingCard rankStrings];
return [rankStrings[self.rank] stringByAppendingString:self.
suit];
}
@synthesize suit = _suit;
+ (NSArray *)validSuits
{
return @[@"♠",@"♣",@"♥",@"♦"];
}
And only 1 point for matching the suit ...
There are only 3 cards that will match a given card's rank, but 12 which will match its suit, so this makes some sense.

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - CardMatchingGame.h
    - CardMatchingGame.m
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
            forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
                forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:
                UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
```

That's it for our Model.
Back to our View and Controller.

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

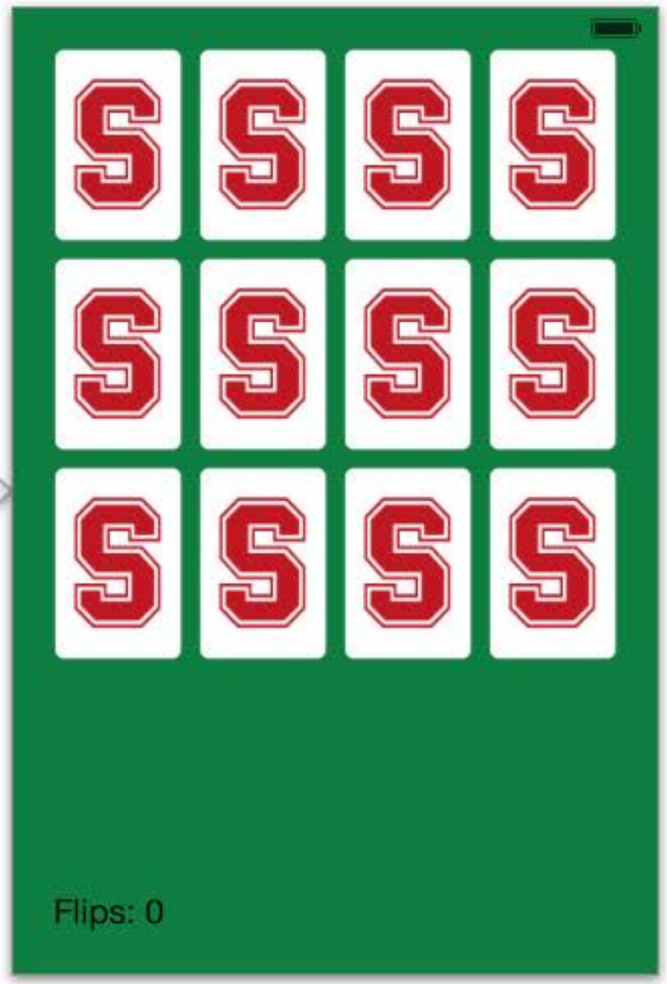
Card G... › View | Automatic › CardGameViewController.m › -touchCardButton:


```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}
```

Close the Navigator.

Almost done.

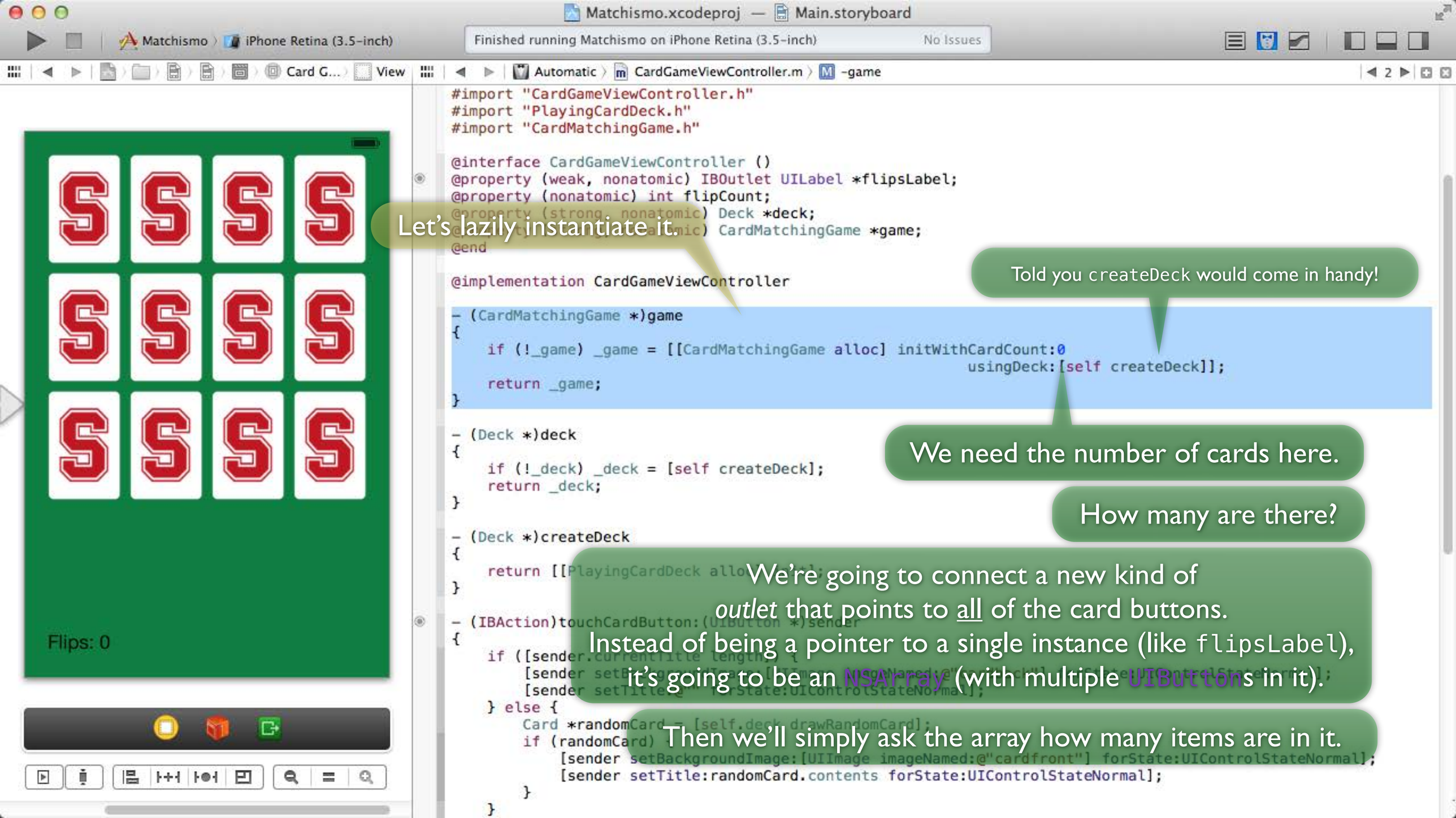
All we have left to do is use our new `CardMatchingGame` Model in our Controller and add some UI to show the score.



Don't forget the `#import` of the `CardMatchingGame`.

We need a `@property` for our game Model.

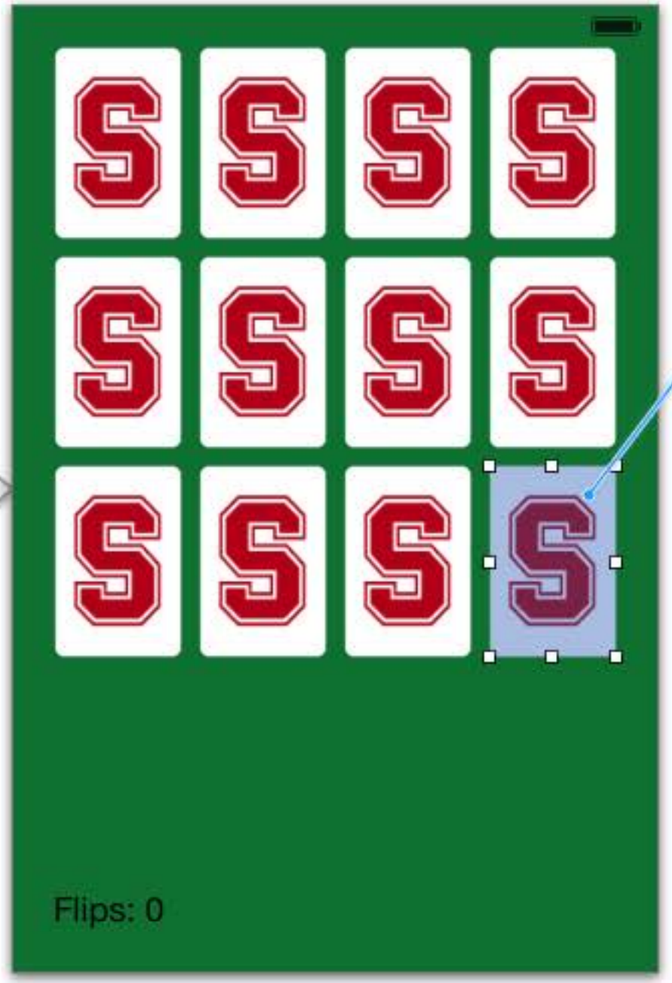
```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

@implementation CardGameViewController

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
```


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › -game
Flips: 0
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
usingDeck:[self createDeck]];
return _game;
}
- (Deck *)deck
{
if (!_deck) _deck = [self createDeck];
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
Let's lazily instantiate it.
Told you createDeck would come in handy!
We need the number of cards here.
How many are there?
We're going to connect a new kind of
outlet that points to all of the card buttons.
Instead of being a pointer to a single instance (like flipsLabel),
it's going to be an NSArray (with multiple UIButtons in it).
Then we'll simply ask the array how many items are in it.

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

V. › Button | Automatic › CardGameViewController.m › @interface CardGameViewController() | 2




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
```

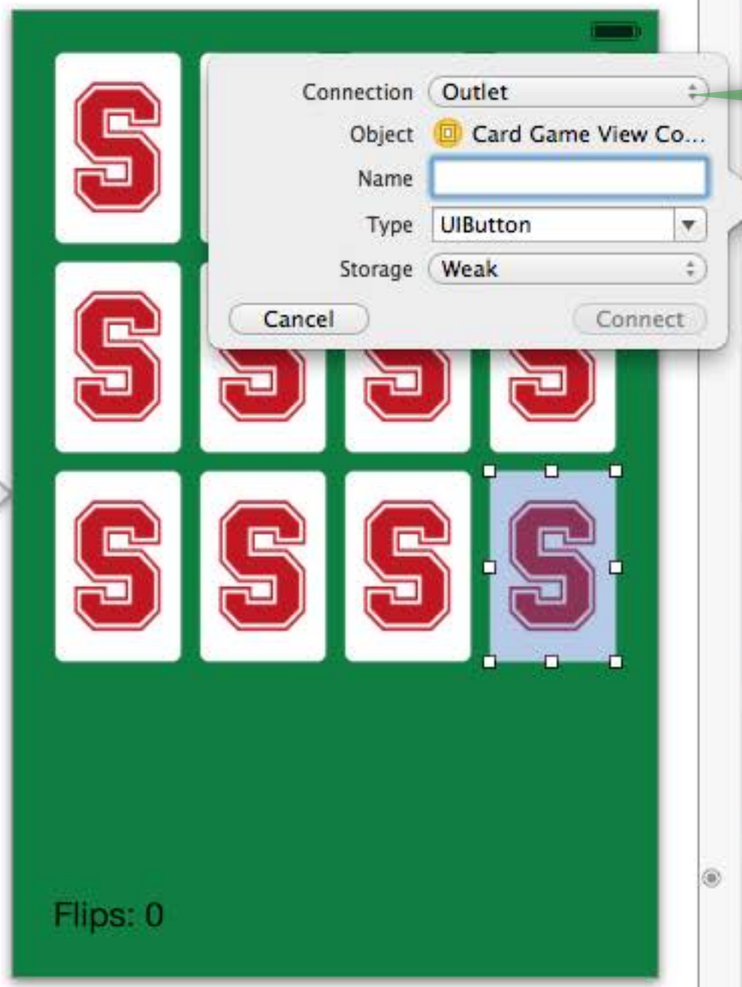
Insert Outlet, Action, or Outlet Collection

Ctrl-drag (as usual) from any one of the buttons into the `@interface`-`@end` area (because that's how we create an outlet, as you'll remember from the `flipsLabel`).

Matchismo.xcodeproj — Main.storyboard

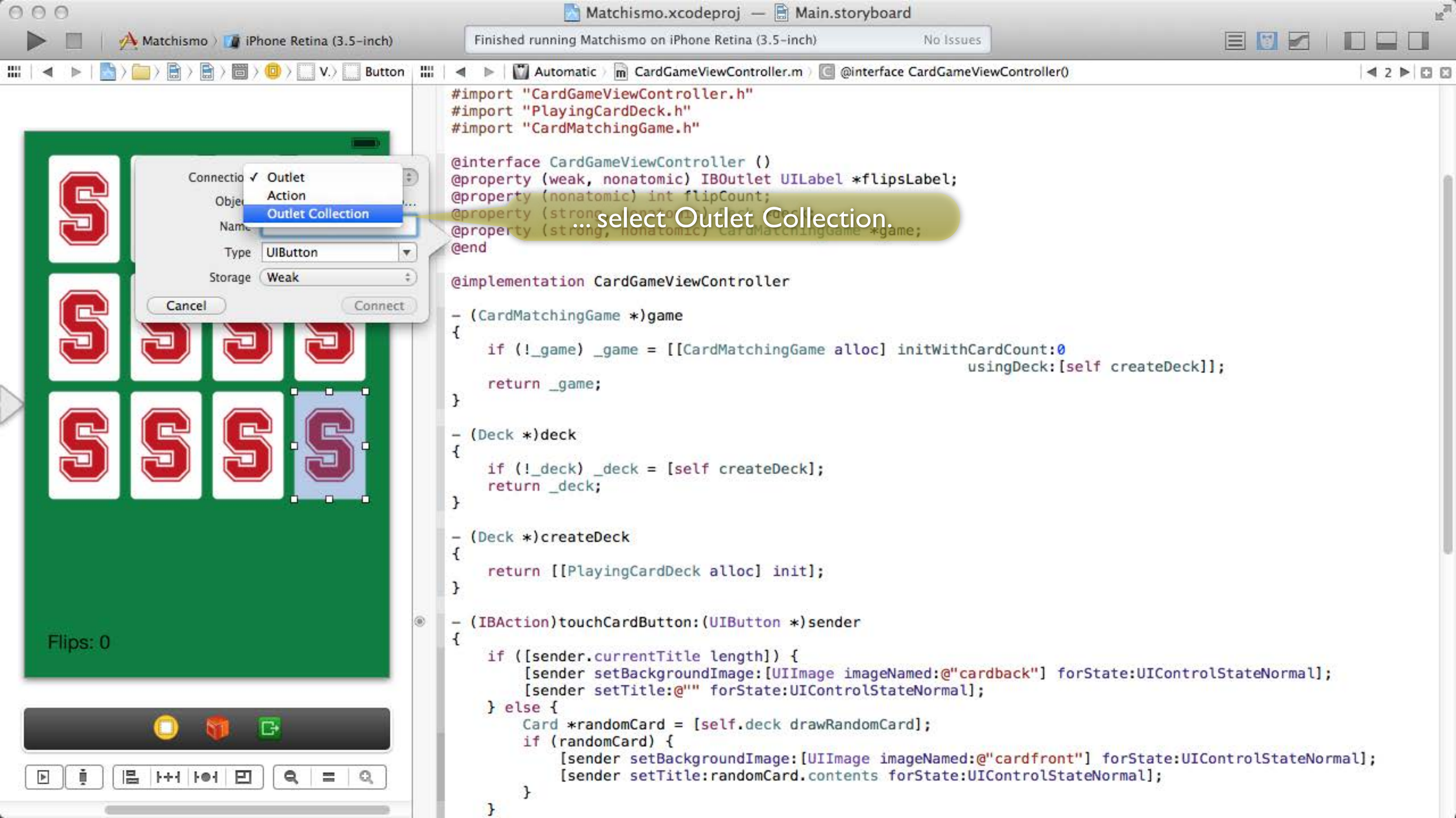
Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

V. › Button | Automatic › CardGameViewController.m › @interface CardGameViewController() | 2

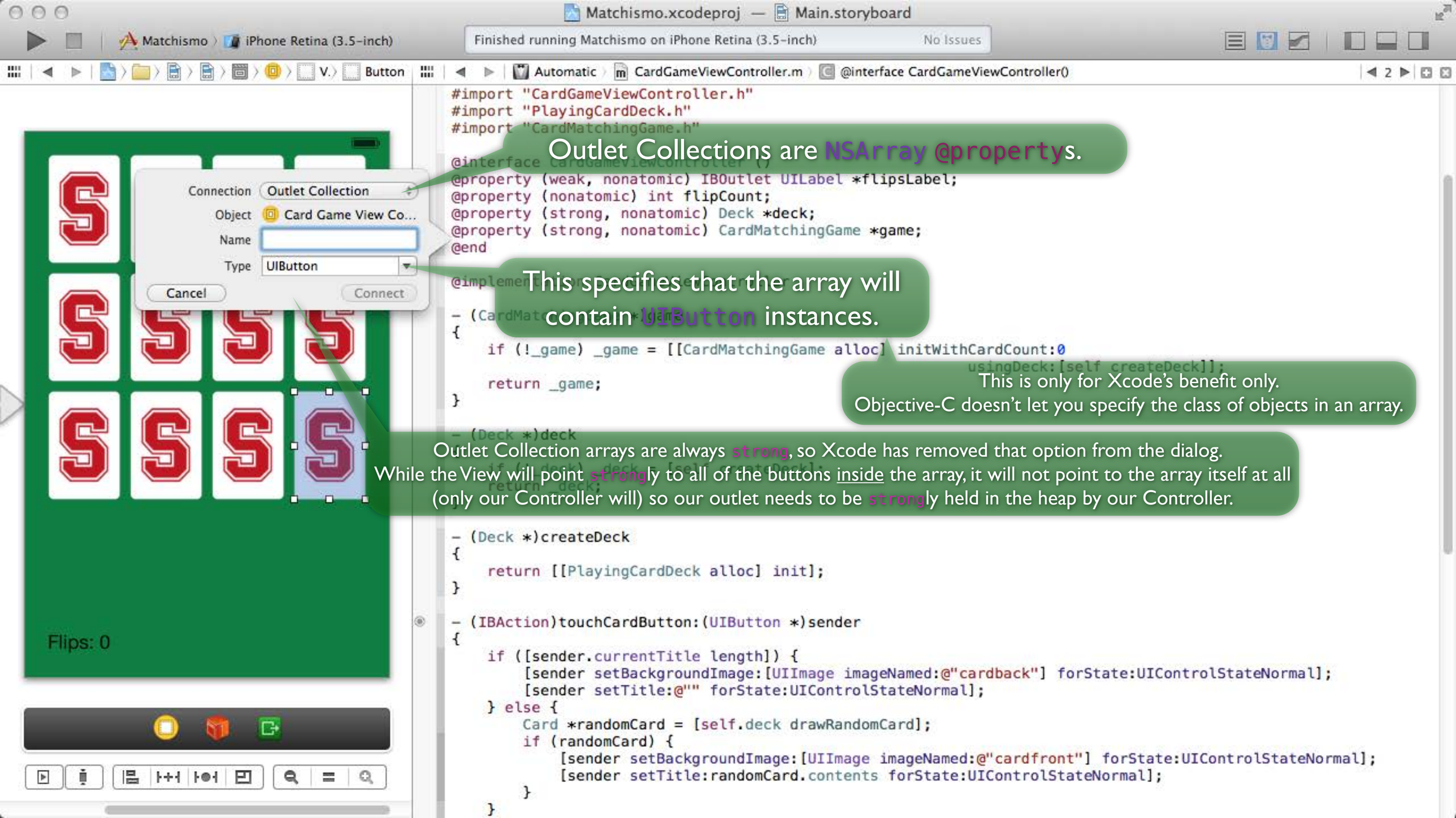


Instead of a normal Outlet ...

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (w                    IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
```



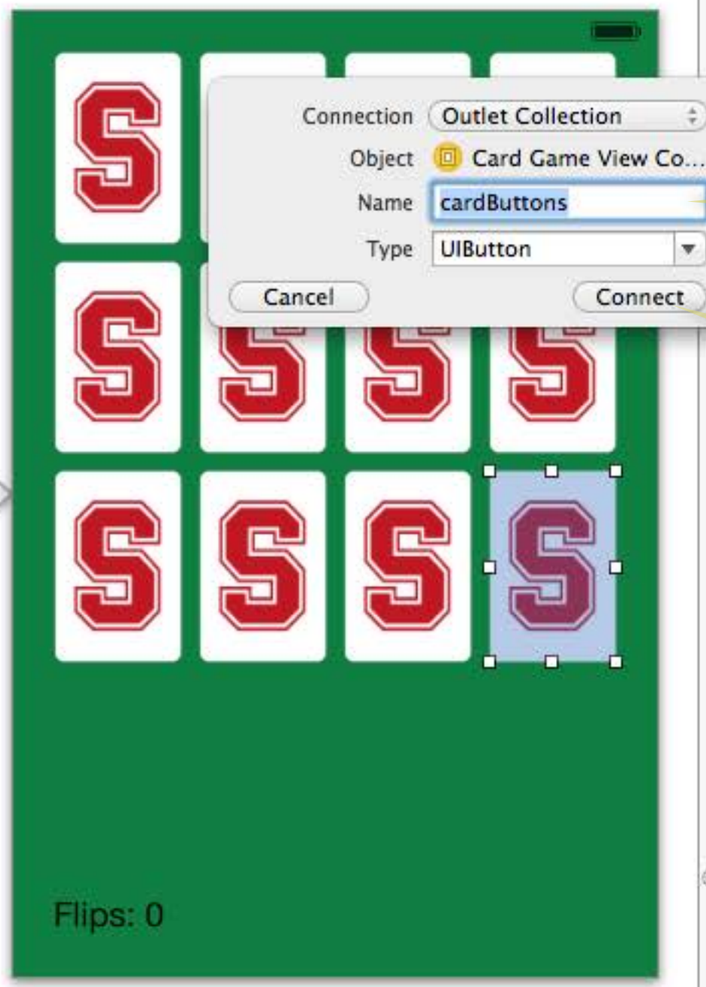
... select Outlet Collection.


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button
Automatic › CardGameViewController.m › @interface CardGameViewController()
Connection Outlet Collection
Object Card Game View Co...
Name
Type UIButton
Cancel
Connect
Flips: 0
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
usingDeck:[self createDeck]];
return _game;
}
- (Deck *)deck
{
return _deck;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
Card *randomCard = [self.deck drawRandomCard];
if (randomCard) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
[sender setTitle:randomCard.contents forState:UIControlStateNormal];
}
}
Outlet Collections are NSArray @propertys.
This specifies that the array will contain UIButton instances.
This is only for Xcode's benefit only.
Objective-C doesn't let you specify the class of objects in an array.
Outlet Collection arrays are always strong, so Xcode has removed that option from the dialog.
While the View will point strongly to all of the buttons inside the array, it will not point to the array itself at all
(only our Controller will) so our outlet needs to be strongly held in the heap by our Controller.

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

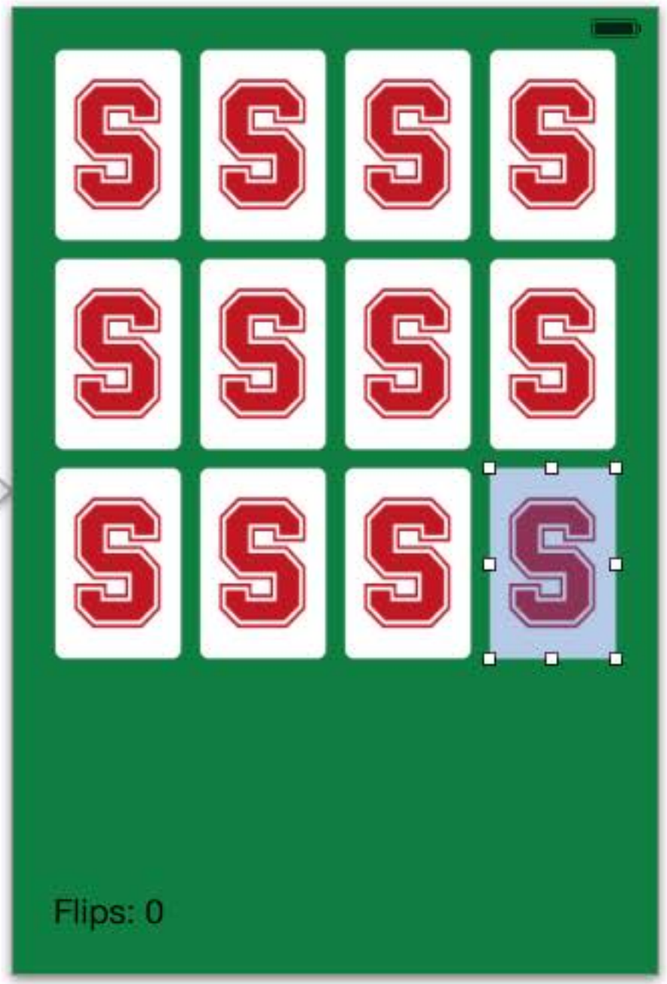
V. › Button | Automatic › CardGameViewController.m › @interface CardGameViewController() | 2



Enter cardButtons as the name of our outlet collection property.

And click Connect.

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
    }
```

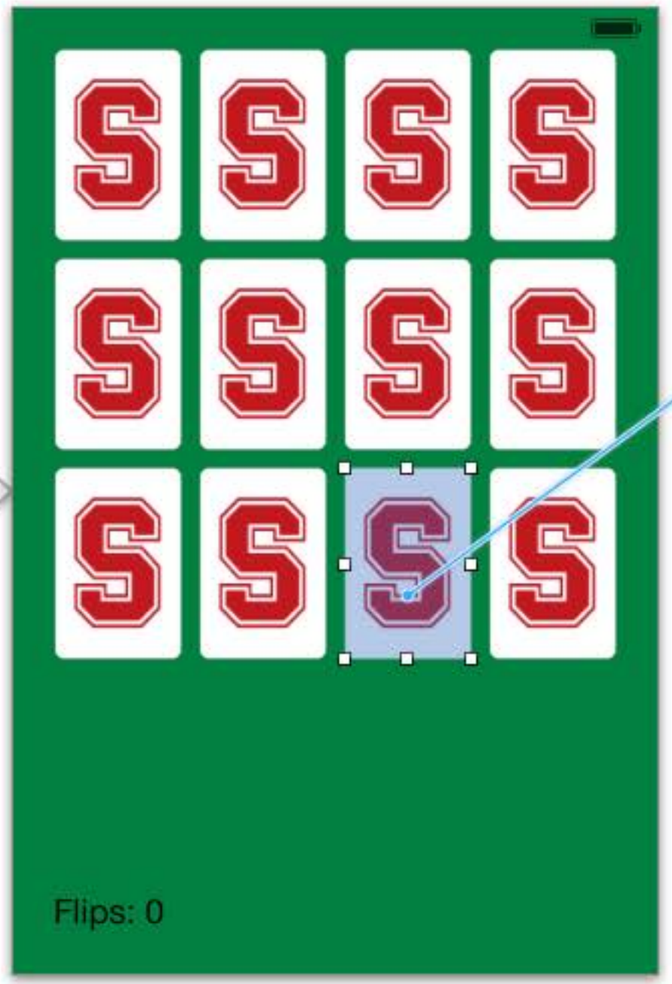



```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

See? strong.

As expected, this outlet is an NSArray.

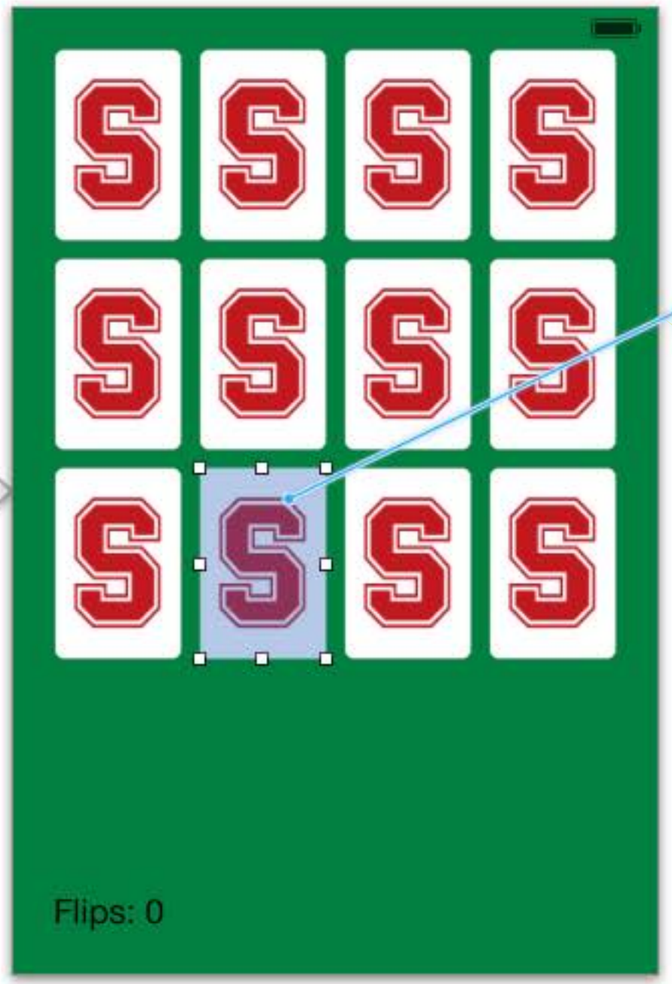
This IBOutletConnection(UIButton) is again just something Xcode puts in there to remember that this is an outlet not just a random NSArray. The compiler ignores this.




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

Connect Outlet Collection

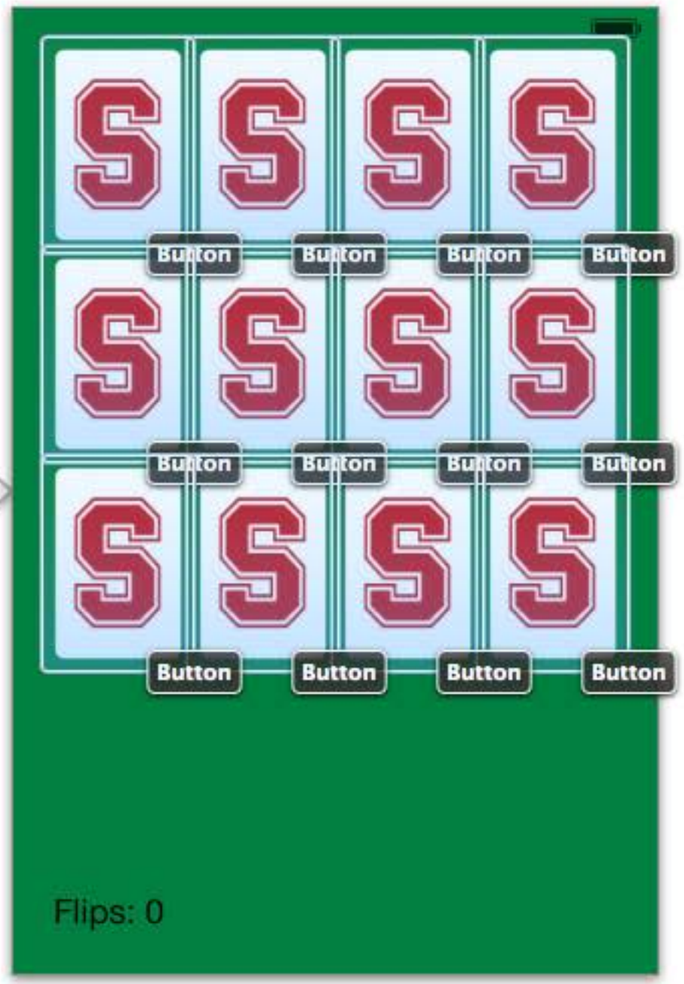
To connect a second button to this outlet collection array, we simply ctrl-drag from the second button to the @property line of code.




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

Connect Outlet Collection

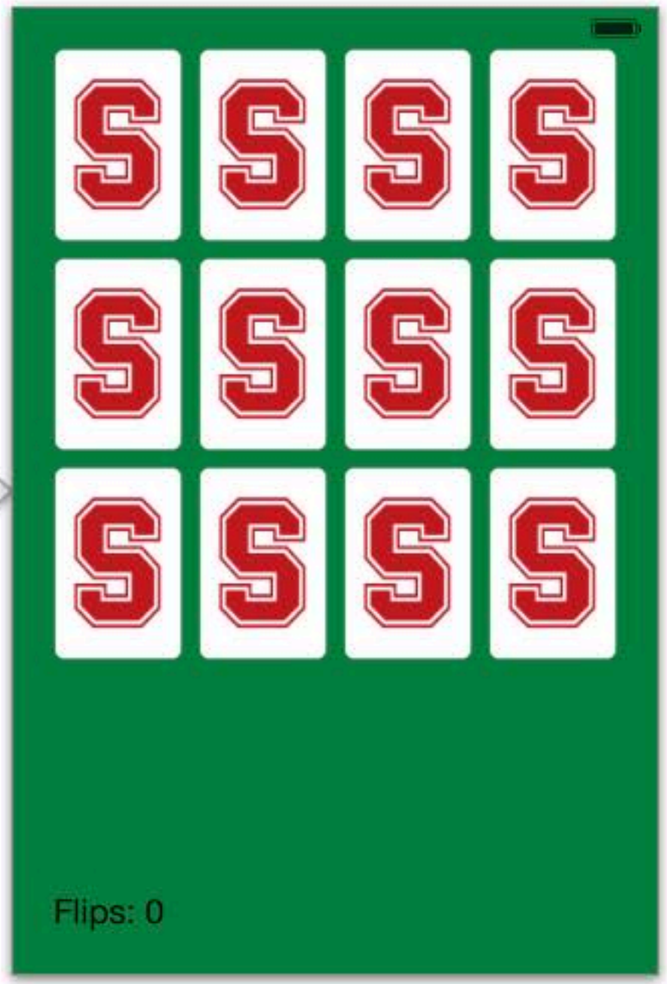
Ditto a third.




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (! game) _game = [[CardMatchingGame alloc] initWithCardCount:0
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

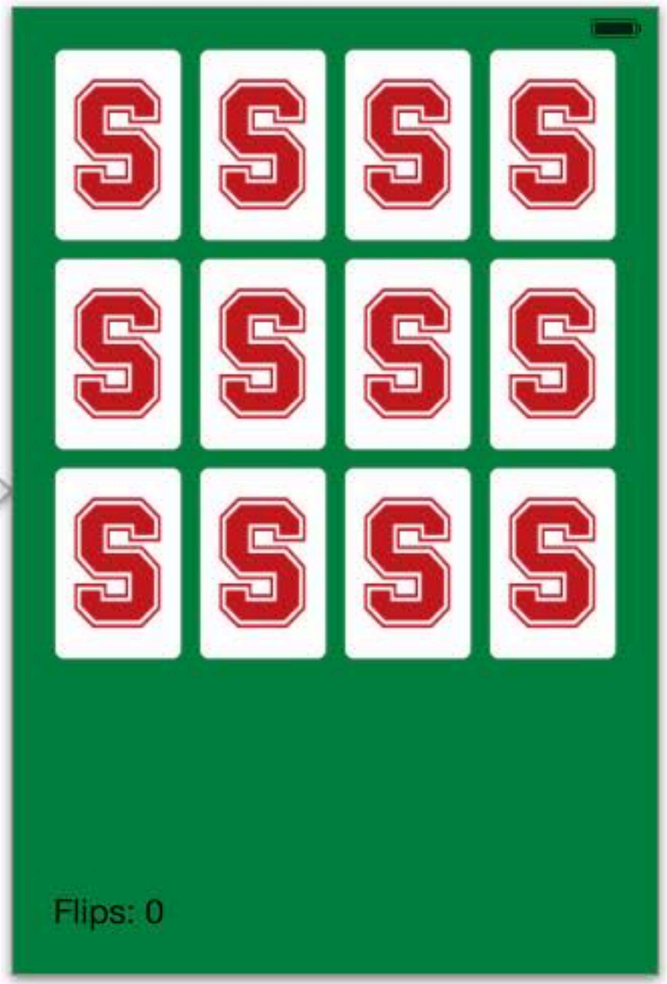
Mousing over this icon shows you what this Outlet Collection is connected to (as usual).
Verify that all the buttons are in the collection.

This is now an NSArray which will contain all the UIButtons you connected in random order.
This randomness is okay for our purposes because the order of these cards will mean nothing to our matching game.
It is something to consider though if you are developing an app where the order does matter (in that case, an outlet collection may not be what you want).



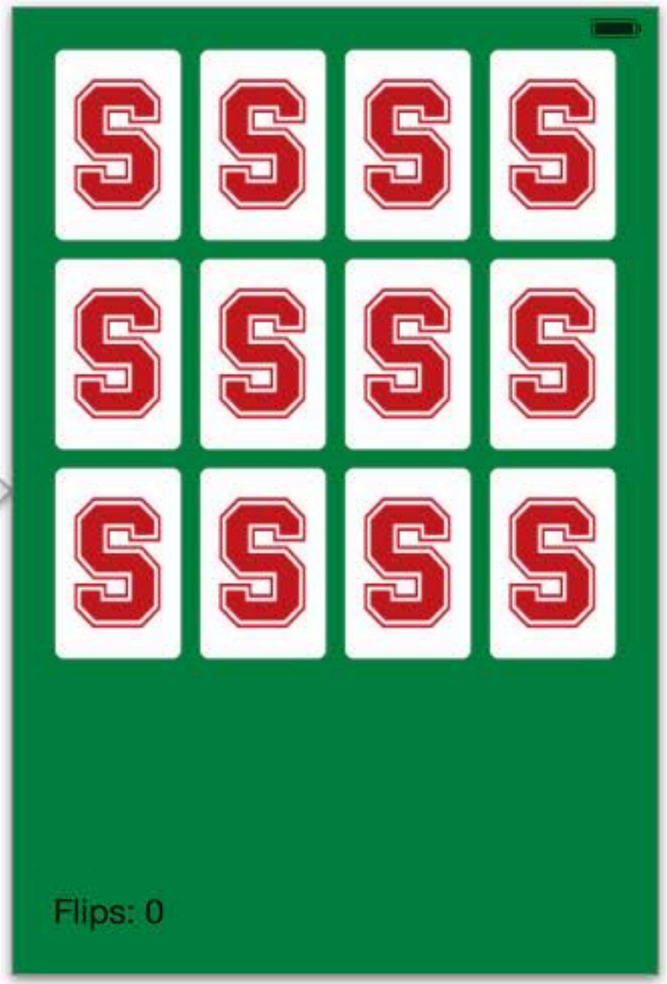

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

Now we know how many cards there are in the UI.




```objc
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck drawRandomCard];
        if (randomCard) {
            [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"] forState:UIControlStateNormal];
            [sender setTitle:randomCard.contents forState:UIControlStateNormal];
        }
```

We don't need any of this anymore because our Model is going to handle it.




```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{

    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
```

So delete it.

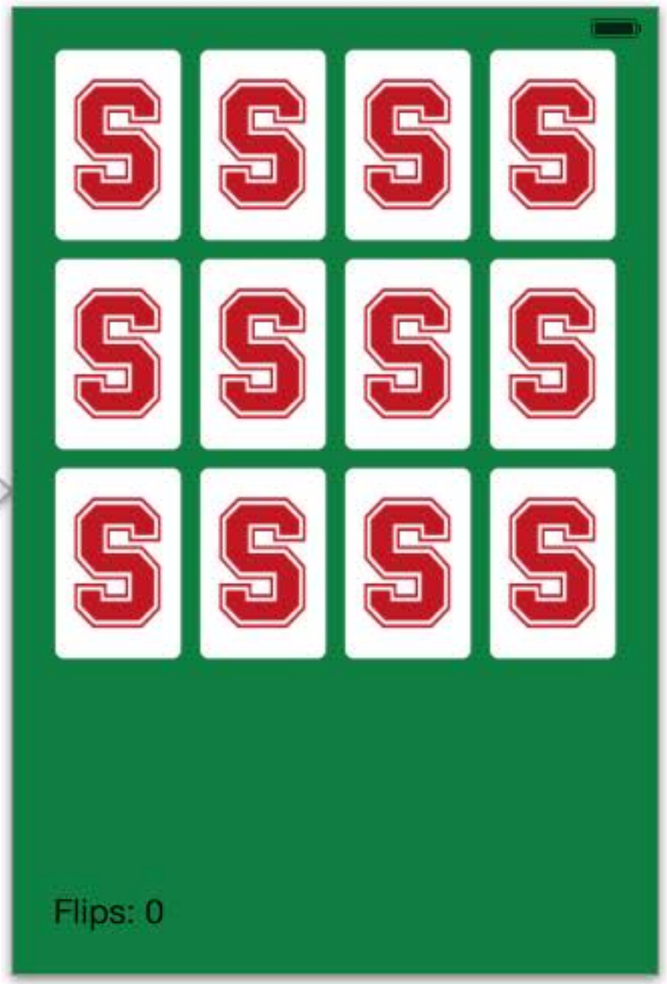

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];

    self.flipCount++;
}

- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
```

We're using our Model here.

CardMatchingGame will now handle all the effects of choosing a card.

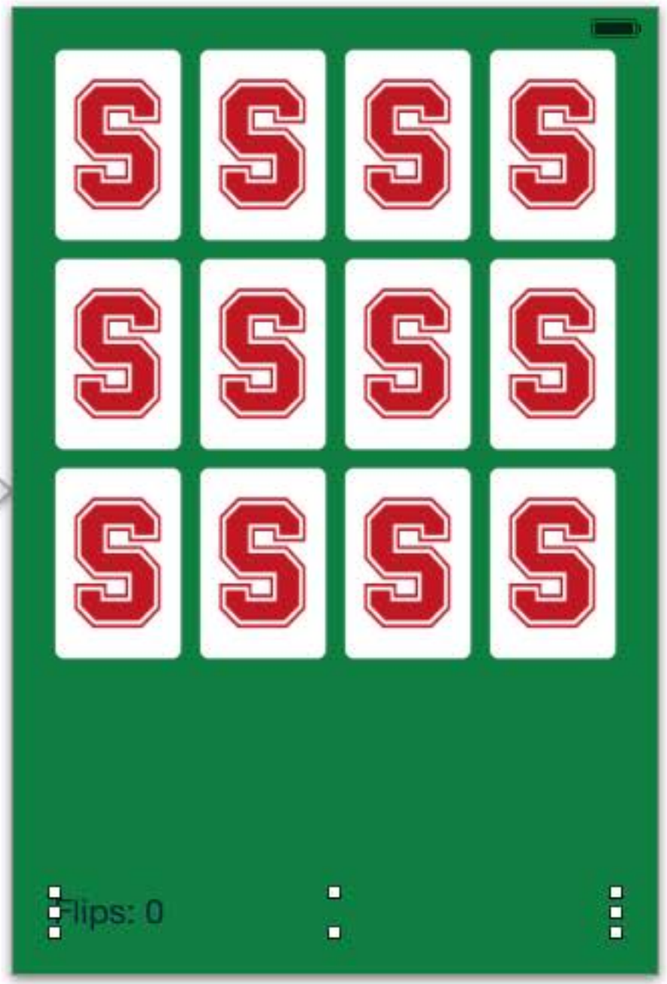
indexOfObject: is an NSArray method.

It does exactly what you would expect (it tells you where the passed object is in the array).



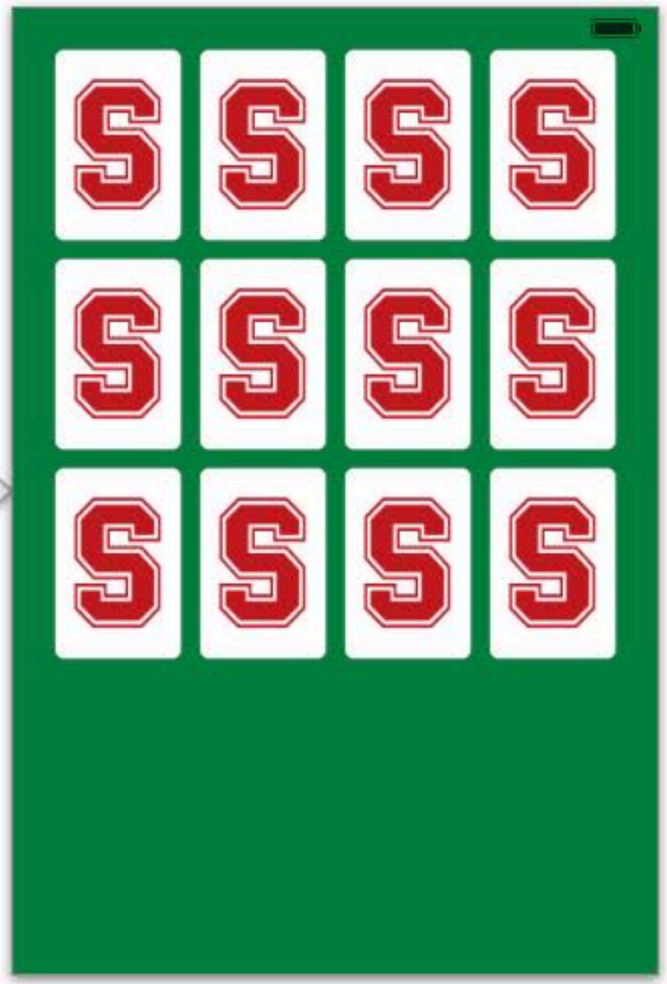

```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
    self.flipCount++;
}

- (void)updateUI { }

- (void)setFlipCount:(int)flipCount
```

However, our Controller must still do its job of interpreting the Model into the View. We'll implement `updateUI` in a moment.




```objc
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
    self.flipCount++;
}

- (void)updateUI { }

- (void)setFlipCount:(int)flipCount
```

Let's get rid of all the old stuff from our simpler version.
We don't need the `Deck`, nor the `flip` stuff.

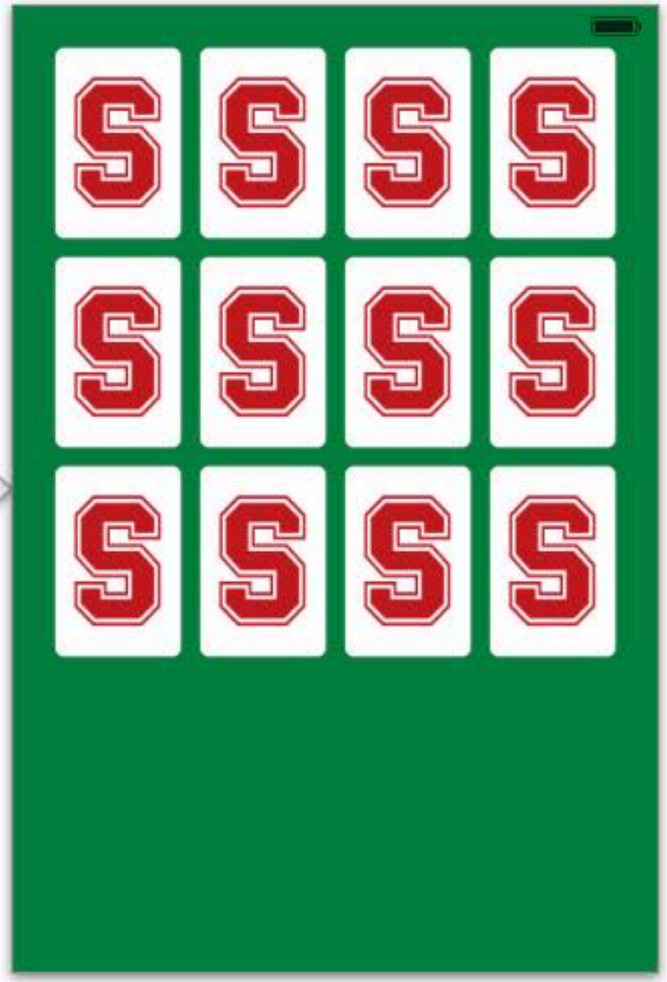


```
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"

@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{




}

@end
```

Matchismo.xcodeproj — Main.storyboard

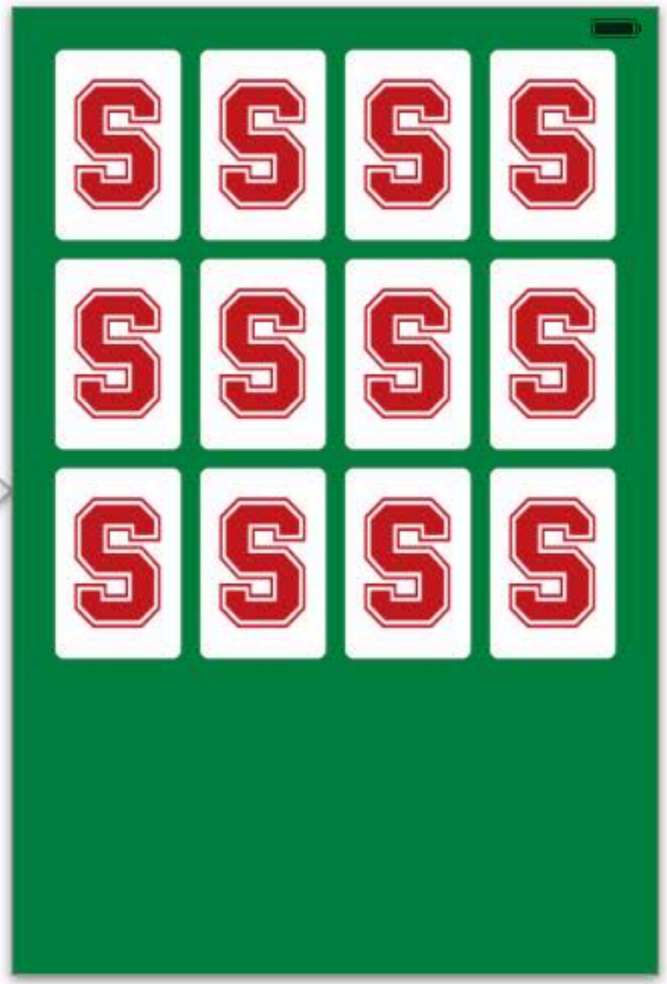
Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › @implementation CardGameViewController



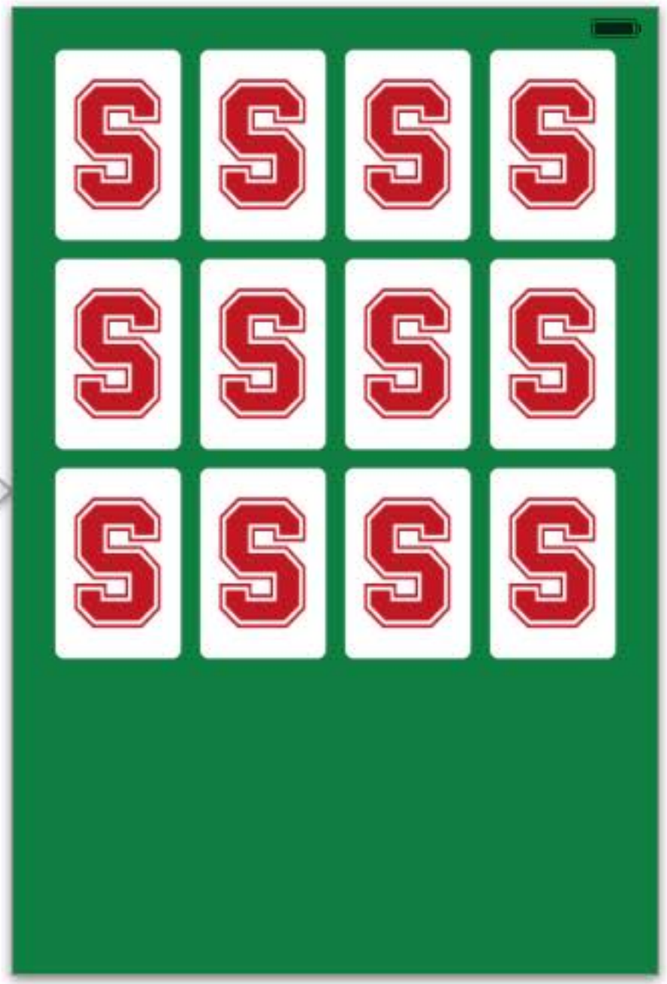
```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{



}



@end
```

Scrolling down to make space ...



```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];



    }
}

@end
```

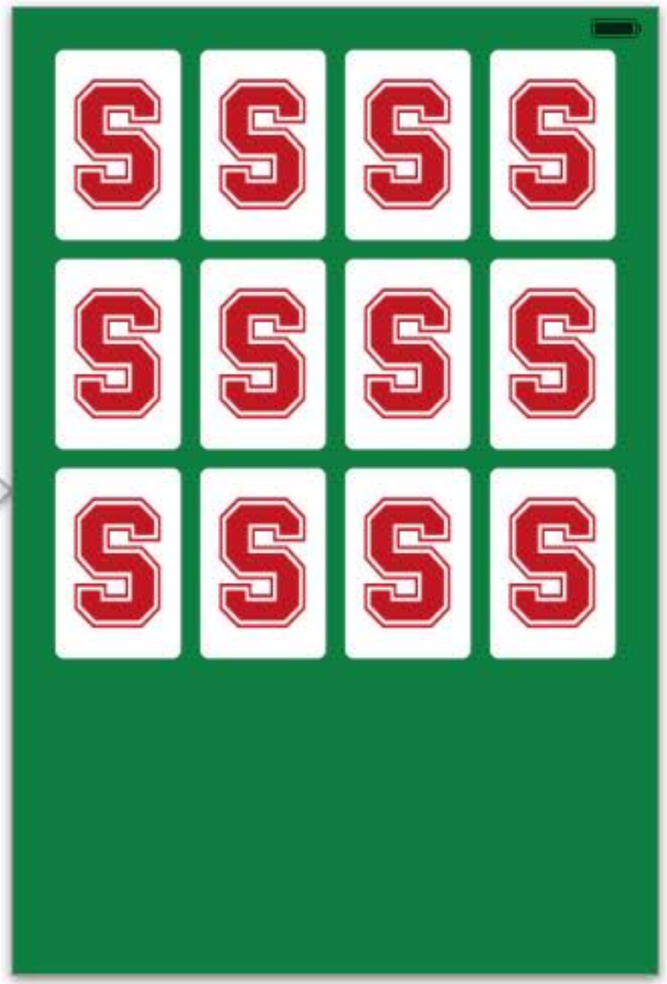
Updating the UI is quite straightforward.
We are just going to cycle through all the `cardButtons` and, based on the corresponding `card` in our Model ...



```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:(NSString *) forState:UIControlStateNormal];
        [cardButton setBackgroundImage:(UIImage *) forState:UIControlStateNormal];

    }
}

@end
```

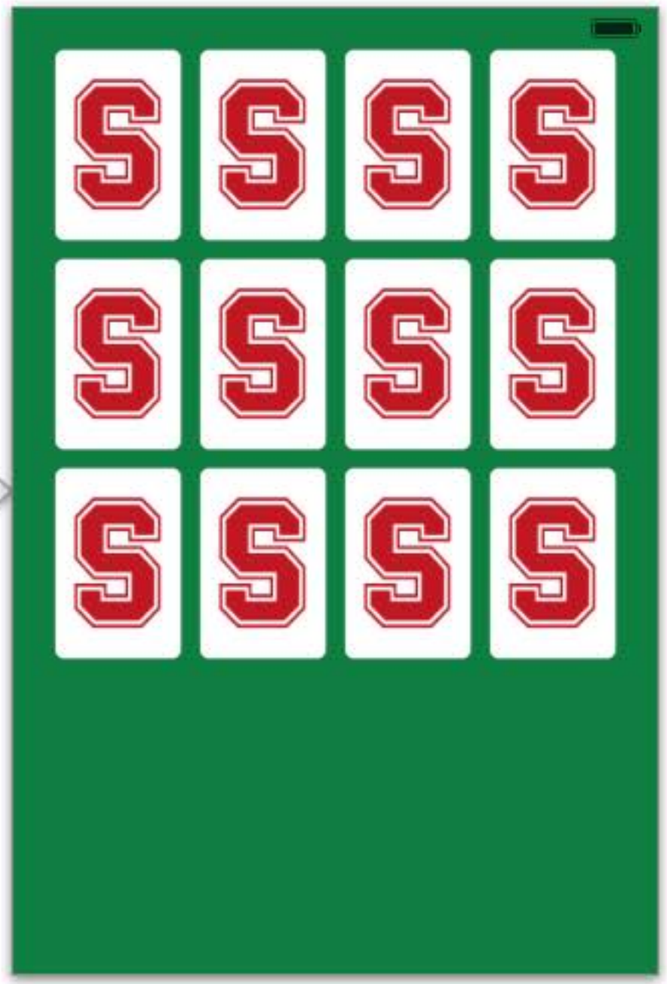
... we'll set the title and background image of the `cardButton`.

We'll create some helper methods to calculate the title and image based on the card.



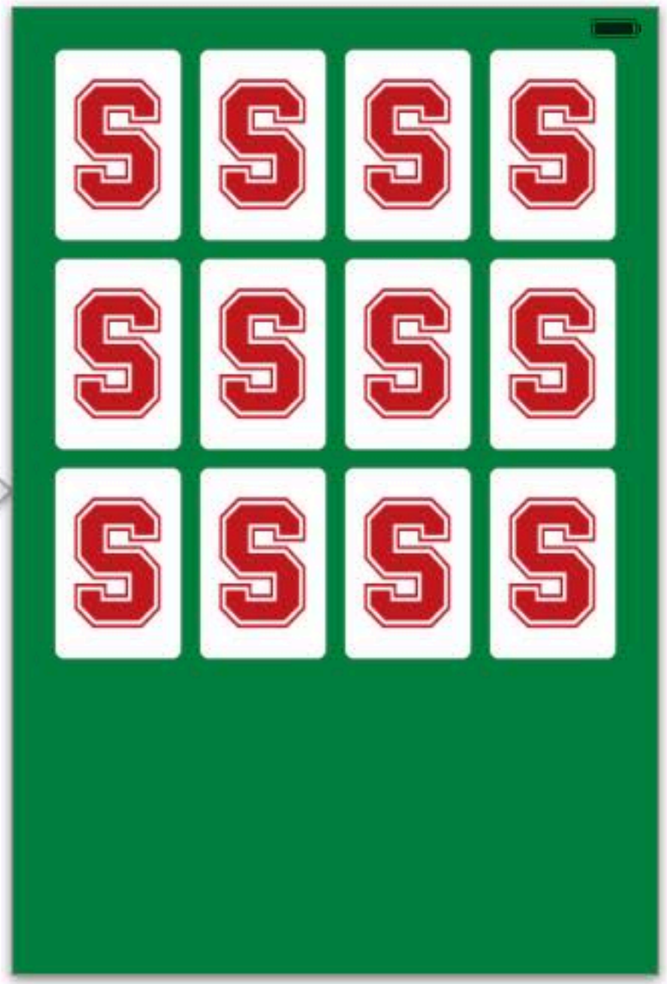
```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:(NSString *) forState:UIControlStateNormal];
        [cardButton setBackgroundImage:(UIImage *) forState:UIControlStateNormal];

    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

The title and background image just depend on whether the card is "chosen" (chosen is face up, not chosen is face down).



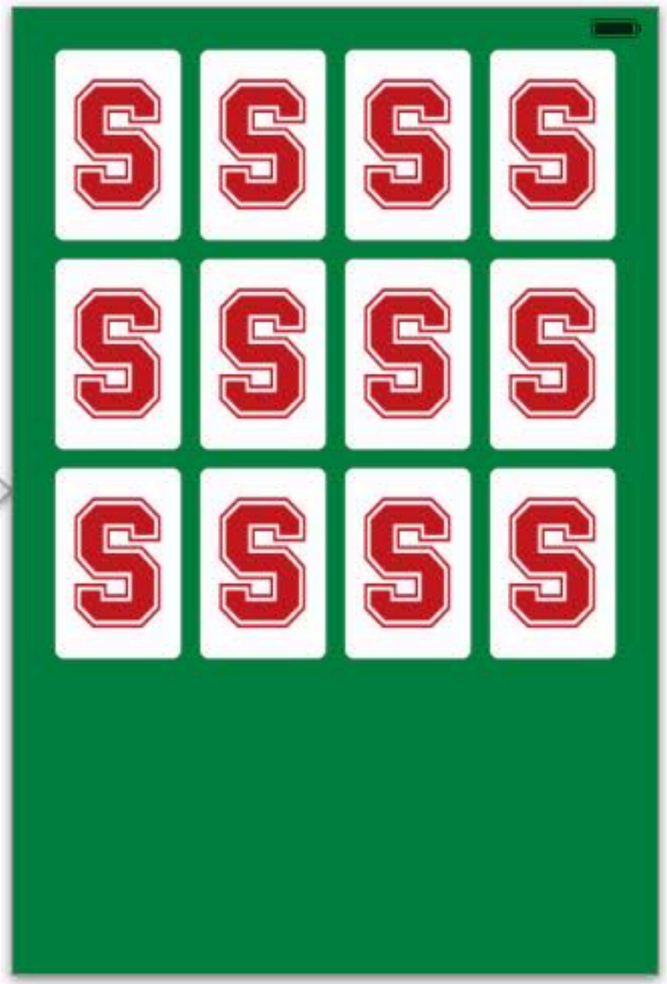
```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];

    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

Now let's use those helper methods.



```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

If a `card.isMatched`, we can disable the corresponding `cardButton`.



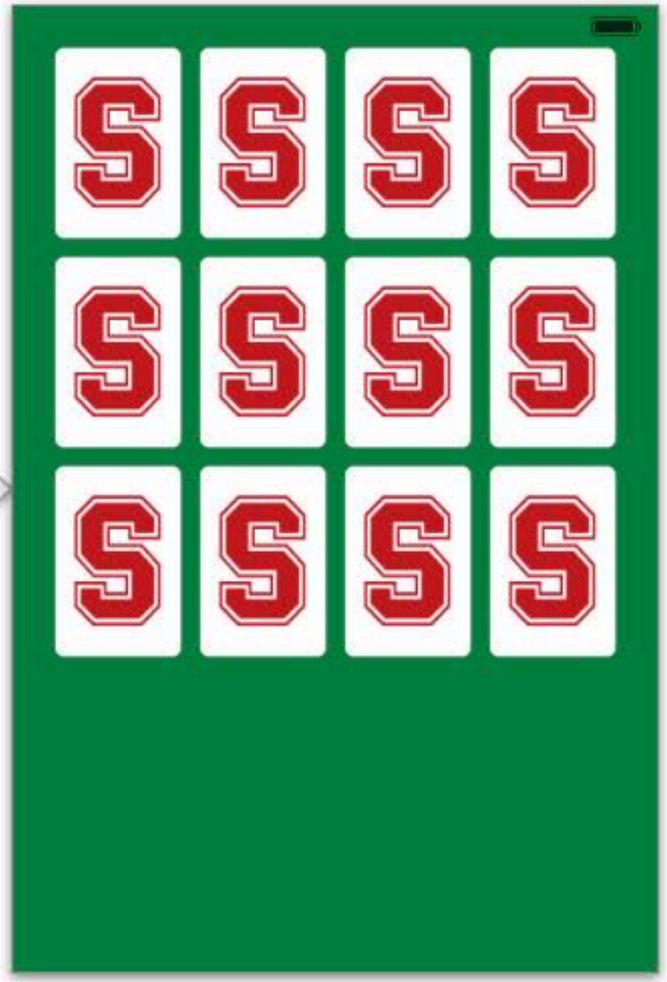
We really need to see the score!

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
    return [UIImage imageNamed:card.isChosen ? @"cardfront" : @"cardback"];
}

@end
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View



Automatic › CardGameViewController.m › -updateUI

```
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
        cardButtons count]
                                                          usingDeck:[self
                                                        createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:
            UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card]
            forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
{
```

View

Mode: Scale To Fill

Tag: 0

Interaction: User Interaction Enabled

Multiple Touch

Alpha: 1

Background

Opaque Hidden

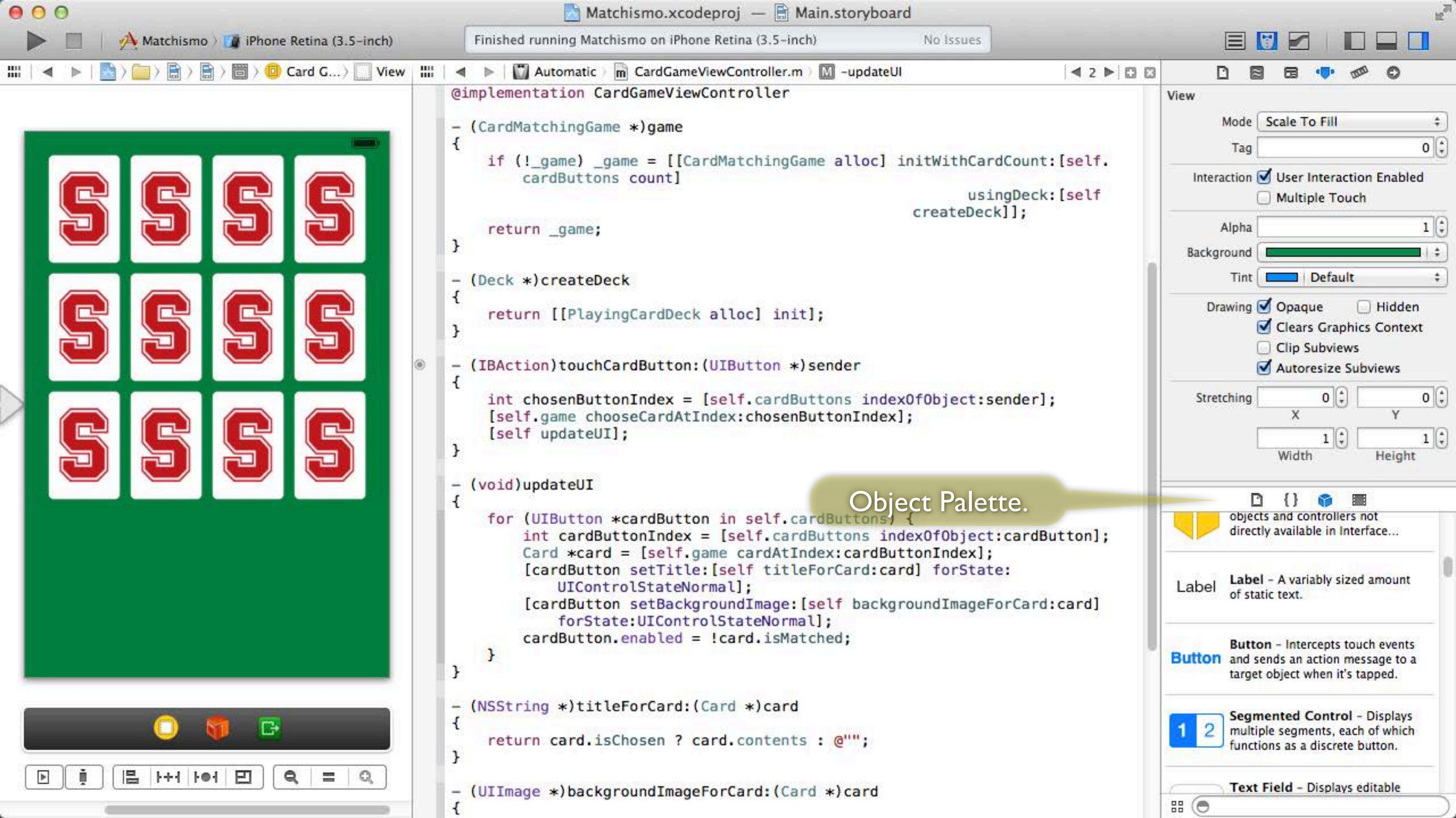
Clears Graphics Context

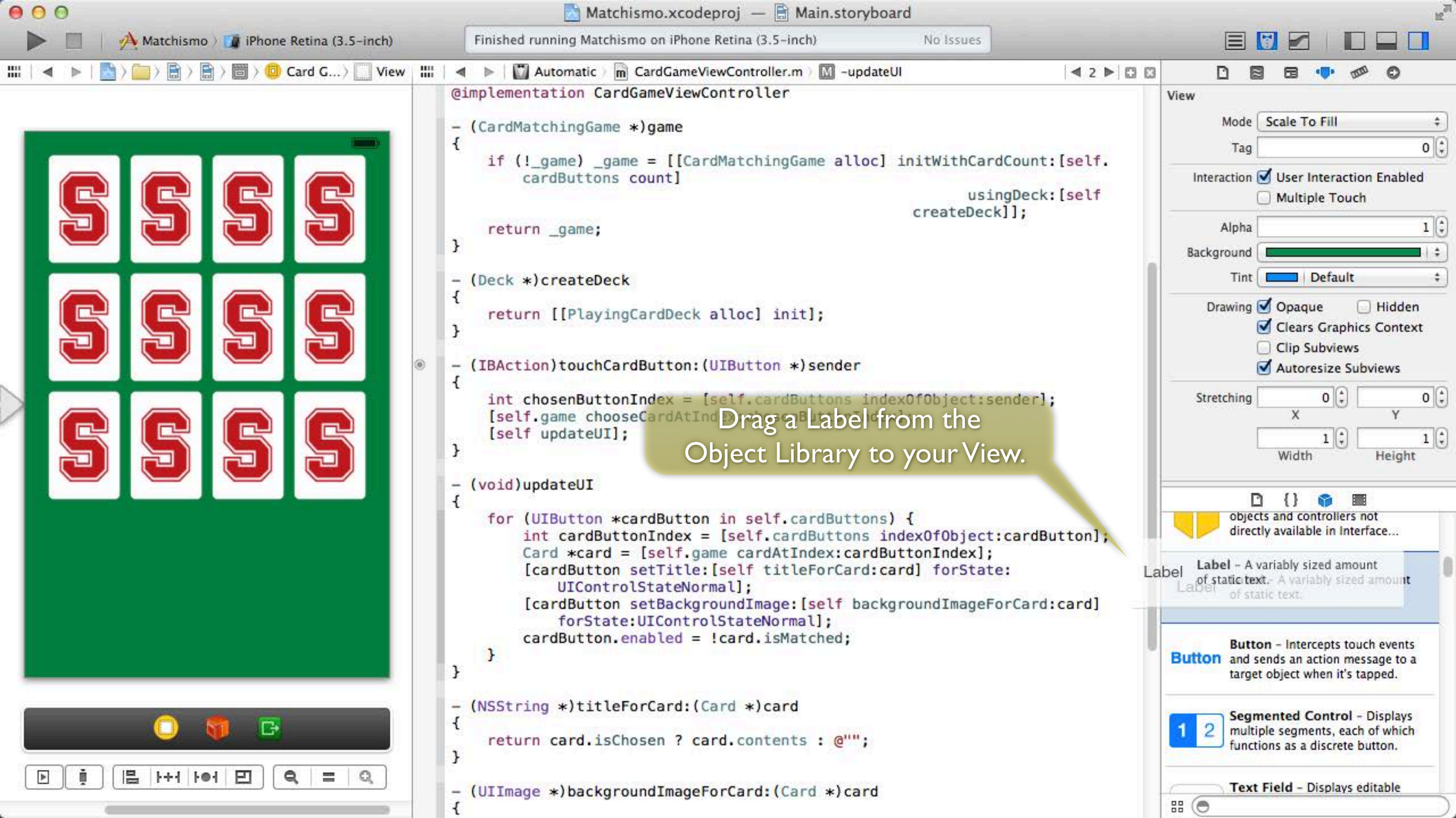
Autoresize Subviews

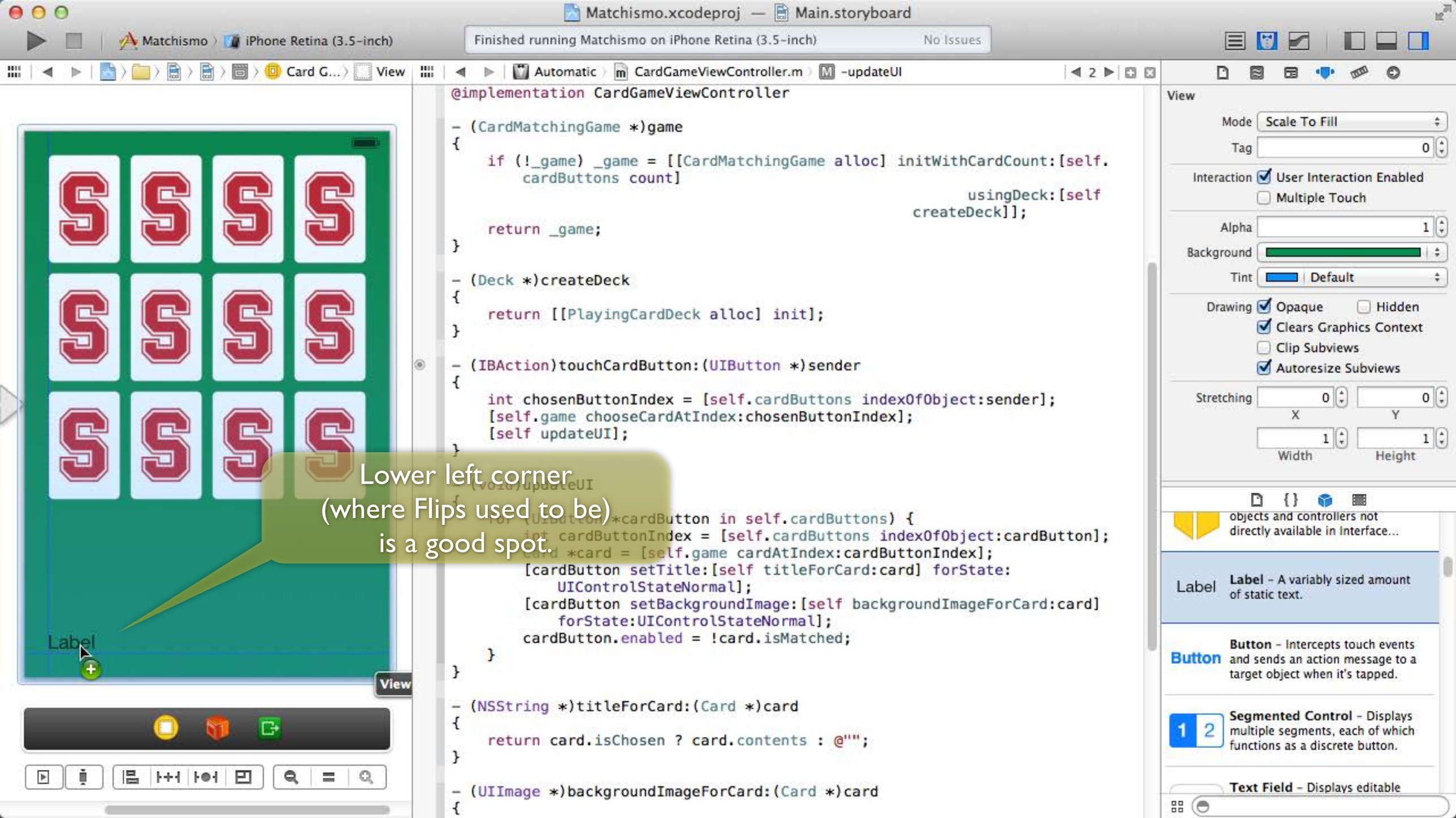
Stretching: X 0 Y 0

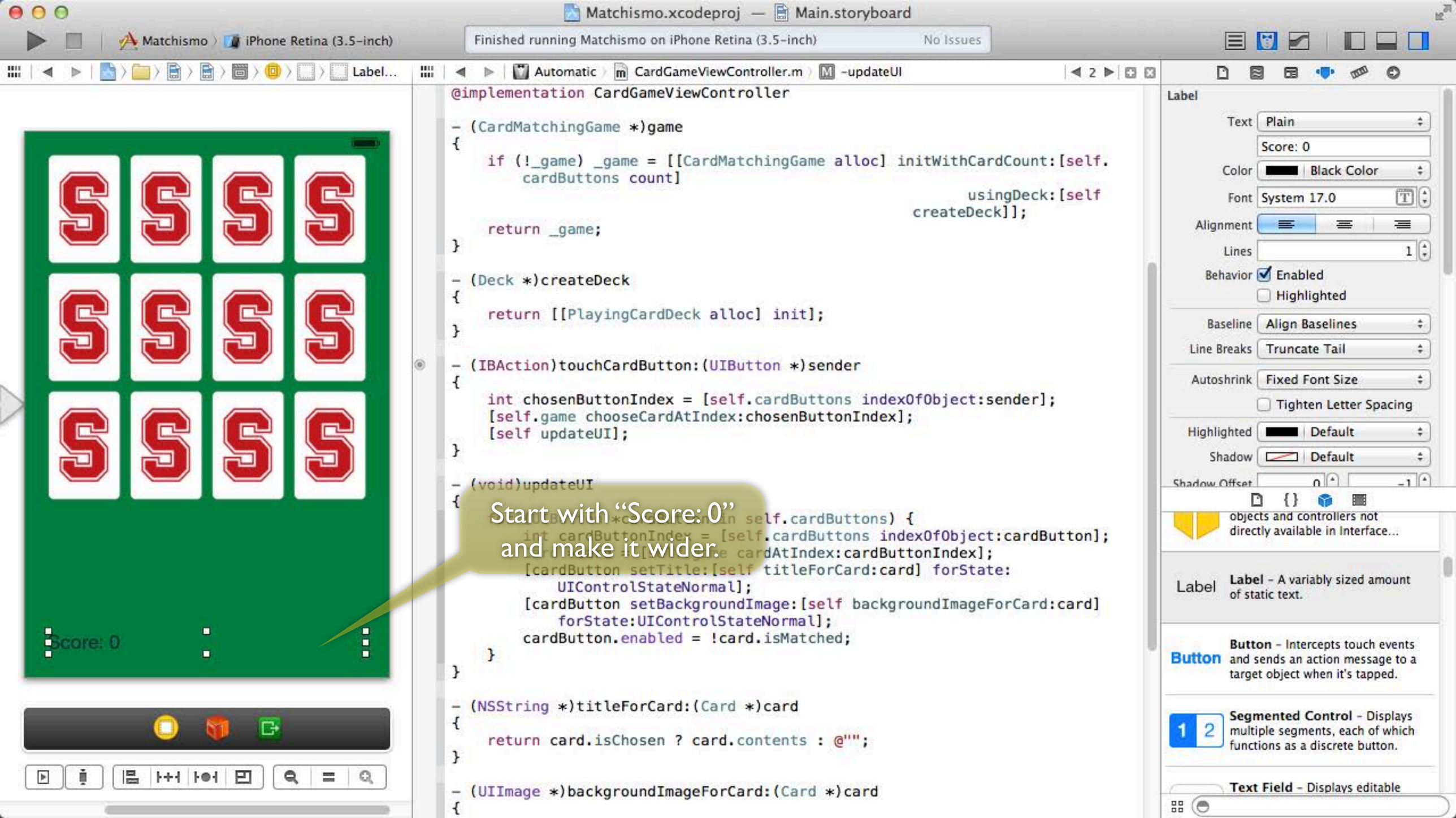
Width 1 Height 1

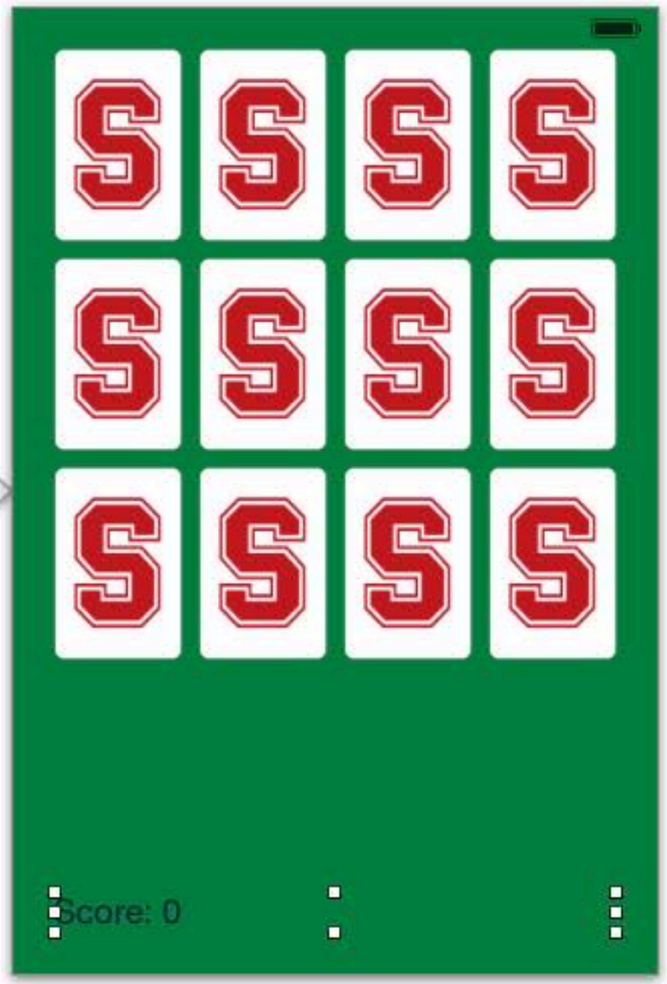
We need a Label for the score, so bring back the Utilities Area so we can drag one out of the palette.


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › -updateUI
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
cardButtons count]
usingDeck:[self
createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:
UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card]
forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card
{
Object Palette.
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 X 0 Y
1 Width 1 Height
objects and controllers not directly available in Interface...
Label Label – A variably sized amount of static text.
Button Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2 Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text Field – Displays editable


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... 〉 View
Automatic 〉 CardGameViewController.m 〉 -updateUI
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
cardButtons count]
usingDeck:[self
createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:
UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card]
forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card
{
Drag a Label from the
Object Library to your View.
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 X 0 Y
1 Width 1 Height
objects and controllers not
directly available in Interface...
Label
Label – A variably sized amount
of static text.
Button
Button – Intercepts touch events
and sends an action message to a
target object when it's tapped.
1 2
Segmented Control – Displays
multiple segments, each of which
functions as a discrete button.
Text Field – Displays editable

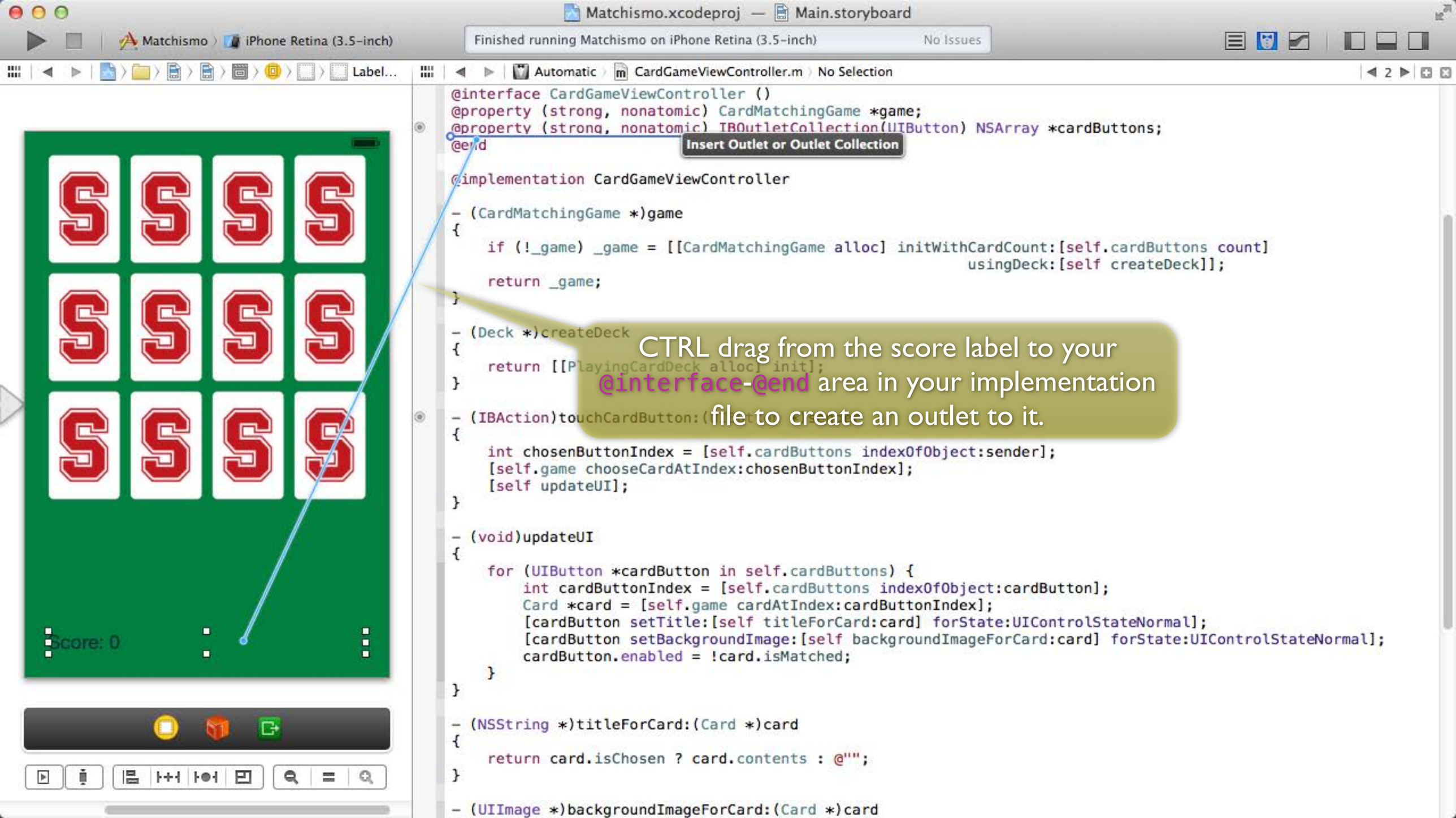

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › -updateUI
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
cardButtons count]
usingDeck:[self
createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:
UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card]
forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card
{
Lower left corner
(where Flips used to be)
is a good spot.
Label
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 0
X Y
1 1
Width Height
objects and controllers not directly available in Interface...
Label – A variably sized amount of static text.
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text Field – Displays editable

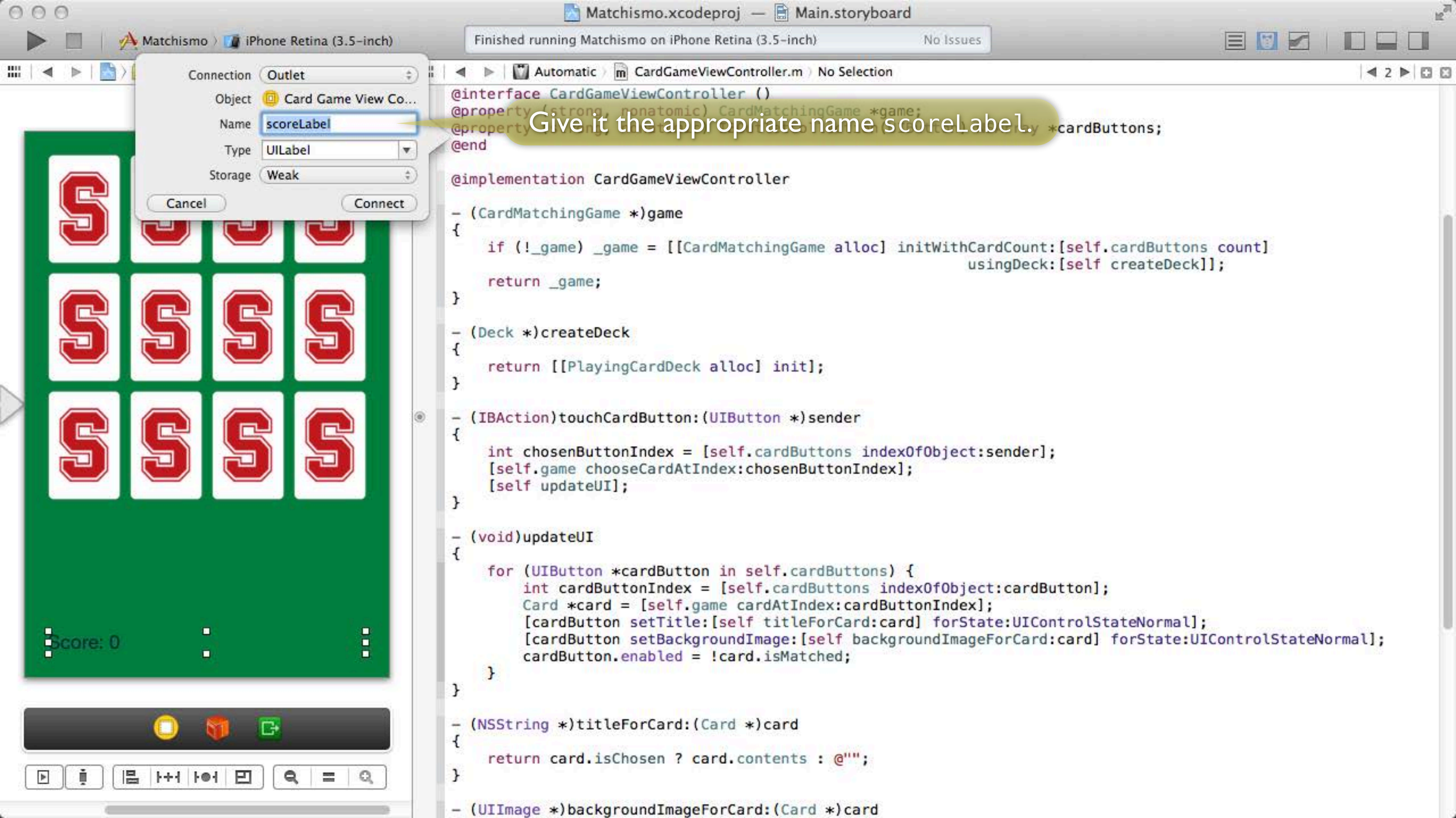

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Label...
Automatic › CardGameViewController.m › -updateUI
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.
cardButtons count]
usingDeck:[self
createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:
UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card]
forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card
{
Score: 0
Start with “Score: 0”
and make it wider.
Label
Text Plain
Score: 0
Color Black Color
Font System 17.0
Alignment
Lines 1
Behavior Enabled
Highlighted
Baseline Align Baselines
Line Breaks Truncate Tail
Autoshrink Fixed Font Size
Tighten Letter Spacing
Highlighted Default
Shadow Default
Shadow Offset
objects and controllers not
directly available in Interface...
Label – A variably sized amount
of static text.
Button – Intercepts touch events
and sends an action message to a
target object when it's tapped.
Segmented Control – Displays
multiple segments, each of which
functions as a discrete button.
Text Field – Displays editable

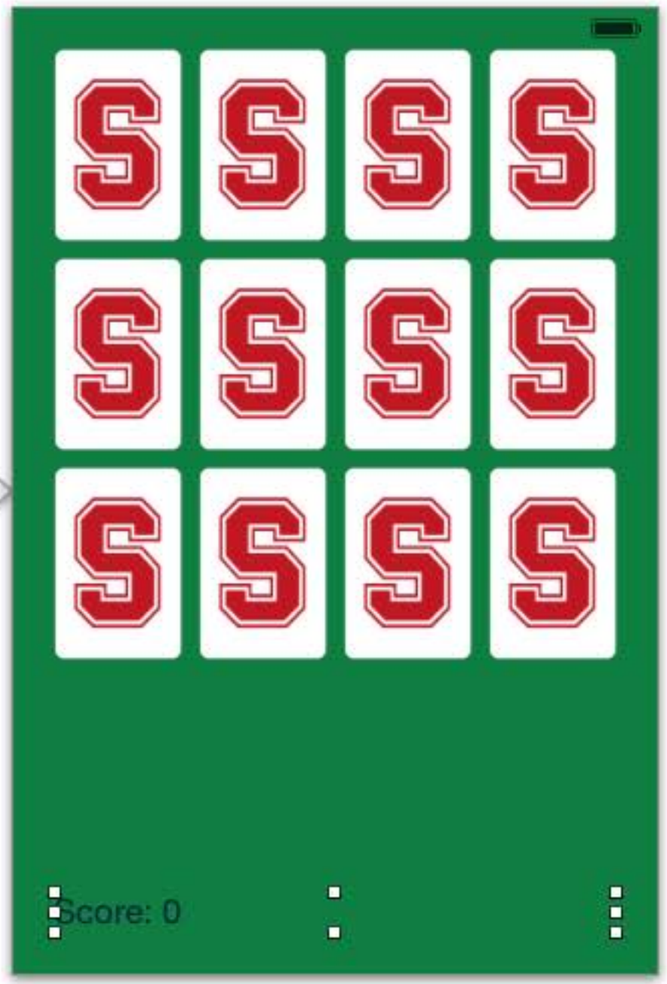


Close Utilities again.

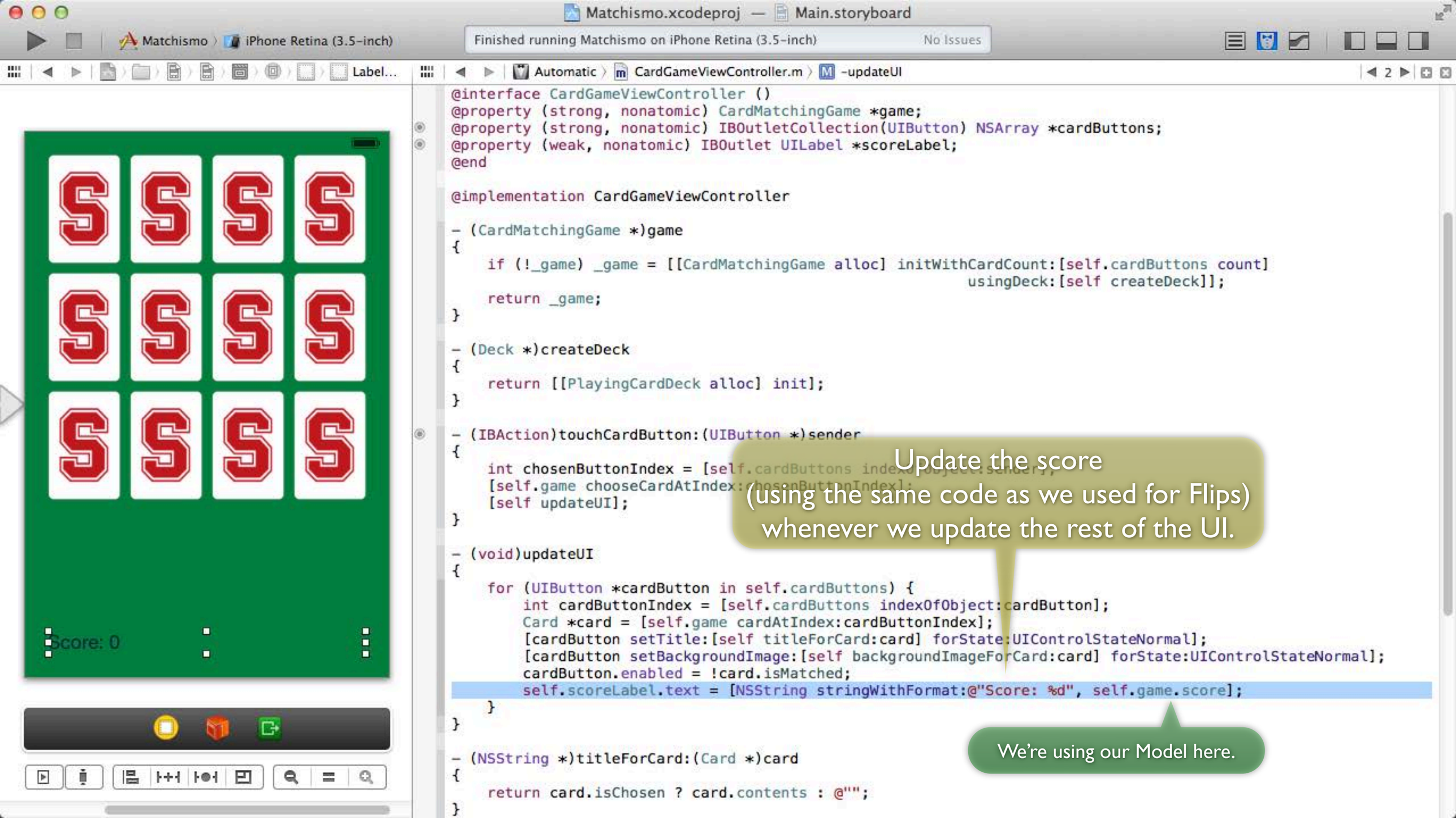
```
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

- (UIImage *)backgroundImageForCard:(Card *)card
```


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Automatic › CardGameViewController.m › No Selection
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end
Insert Outlet or Outlet Collection
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
usingDeck:[self createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card
Score: 0
CTRL drag from the score label to your @interface-@end area in your implementation file to create an outlet to it.


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Connection Outlet
Object Card Game View Co...
Name scoreLabel
Type UILabel
Storage Weak
Cancel
Connect
Give it the appropriate name scoreLabel.
Automatic › CardGameViewController.m › No Selection
Score: 0
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutlet NSArray *cardButtons;
@end
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
usingDeck:[self createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
- (UIImage *)backgroundImageForCard:(Card *)card




```
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@property (weak, nonatomic) IBOutlet UILabel *scoreLabel;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}
```


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Label...
Automatic 〉 CardGameViewController.m 〉 -updateUI
Score: 0
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@property (weak, nonatomic) IBOutlet UILabel *scoreLabel;
@end
@implementation CardGameViewController
- (CardMatchingGame *)game
{
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
usingDeck:[self createDeck]];
return _game;
}
- (Deck *)createDeck
{
return [[PlayingCardDeck alloc] init];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];
}
- (void)updateUI
{
for (UIButton *cardButton in self.cardButtons) {
int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
Card *card = [self.game cardAtIndex:cardButtonIndex];
[cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
[cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
cardButton.enabled = !card.isMatched;
self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}
}
- (NSString *)titleForCard:(Card *)card
{
return card.isChosen ? card.contents : @"";
}
Update the score
(using the same code as we used for Flips)
whenever we update the rest of the UI.
We're using our Model here.

Matchismo.xcodeproj — Main.storyboard

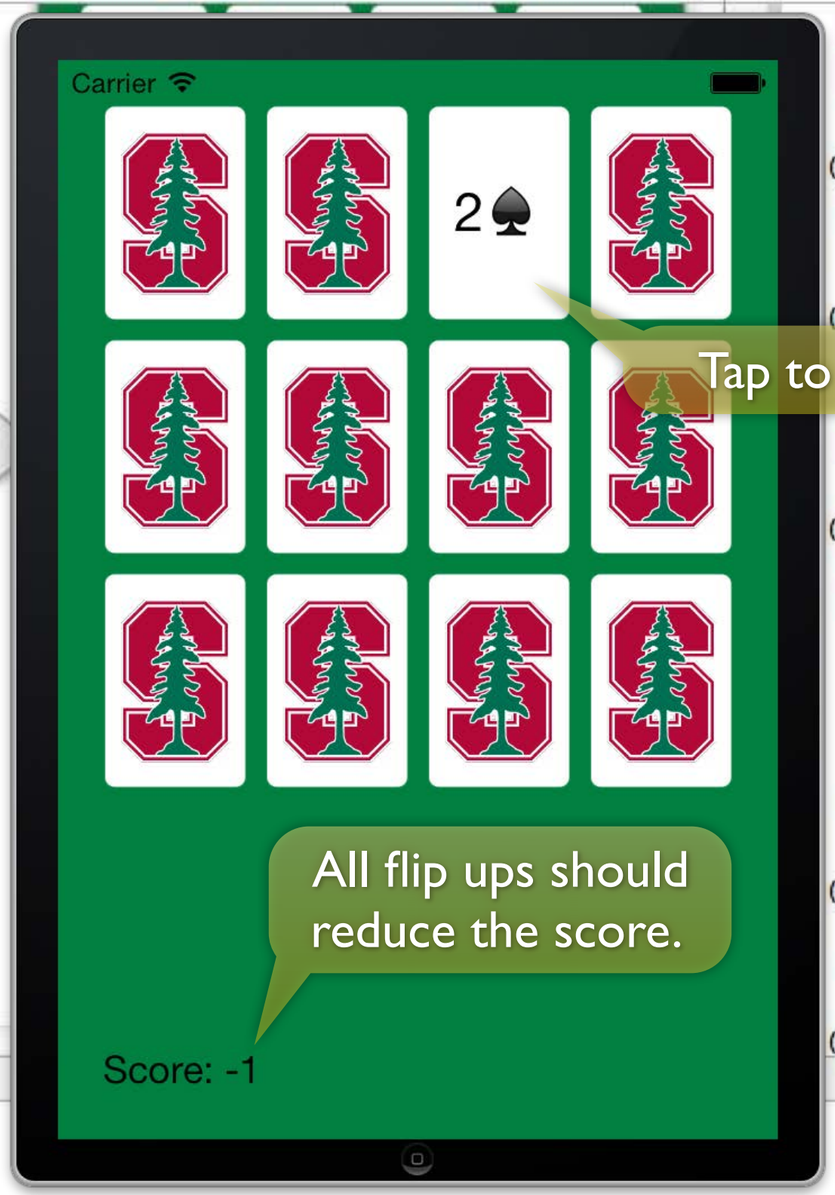
Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › @implementation CardGameViewController | 2

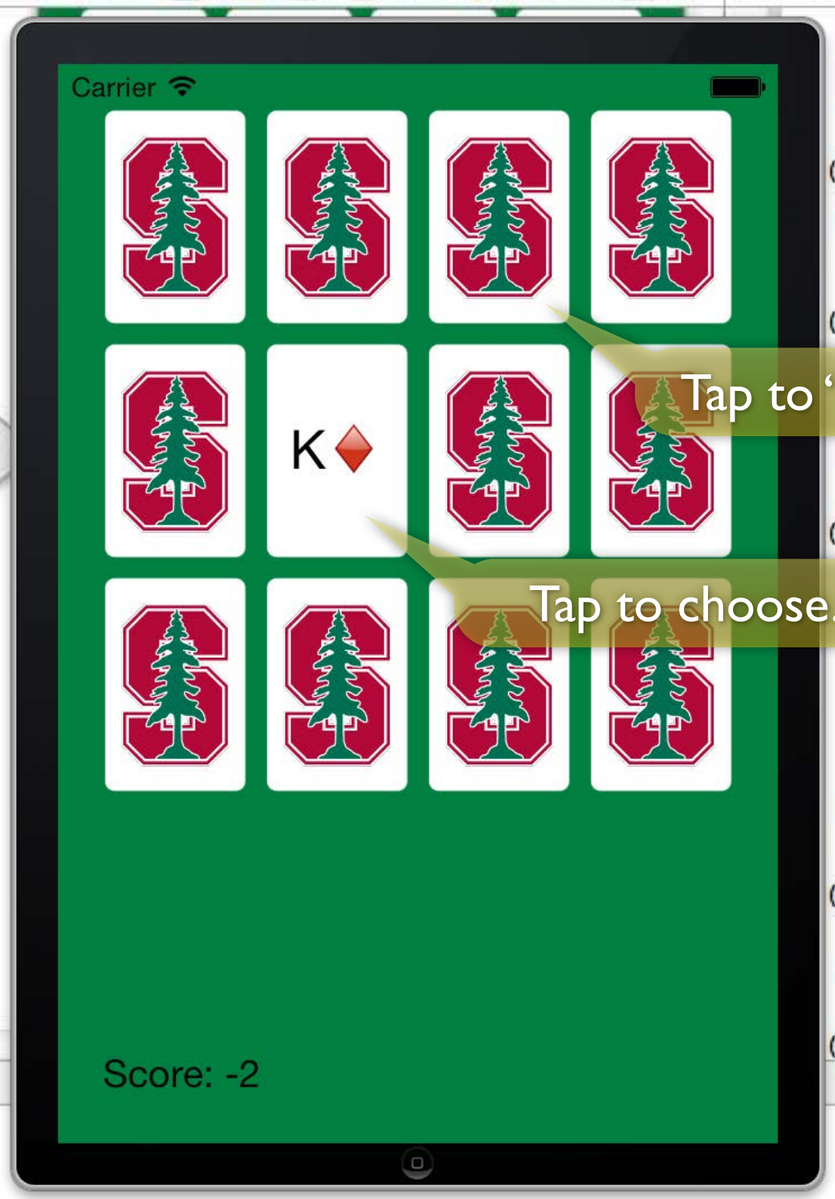
Done! Final run!

```
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;


(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];


(void)updateUI

    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }


(NSString *)titleForCard:(Card *)card

    return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```

Auto | All Output




```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
return _game;


(Deck *)createDeck

return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

nButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];


(void)updateUI

for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}


(NSString *)titleForCard:(Card *)card

return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```



Tap to "unchoose" (should not cost a point).

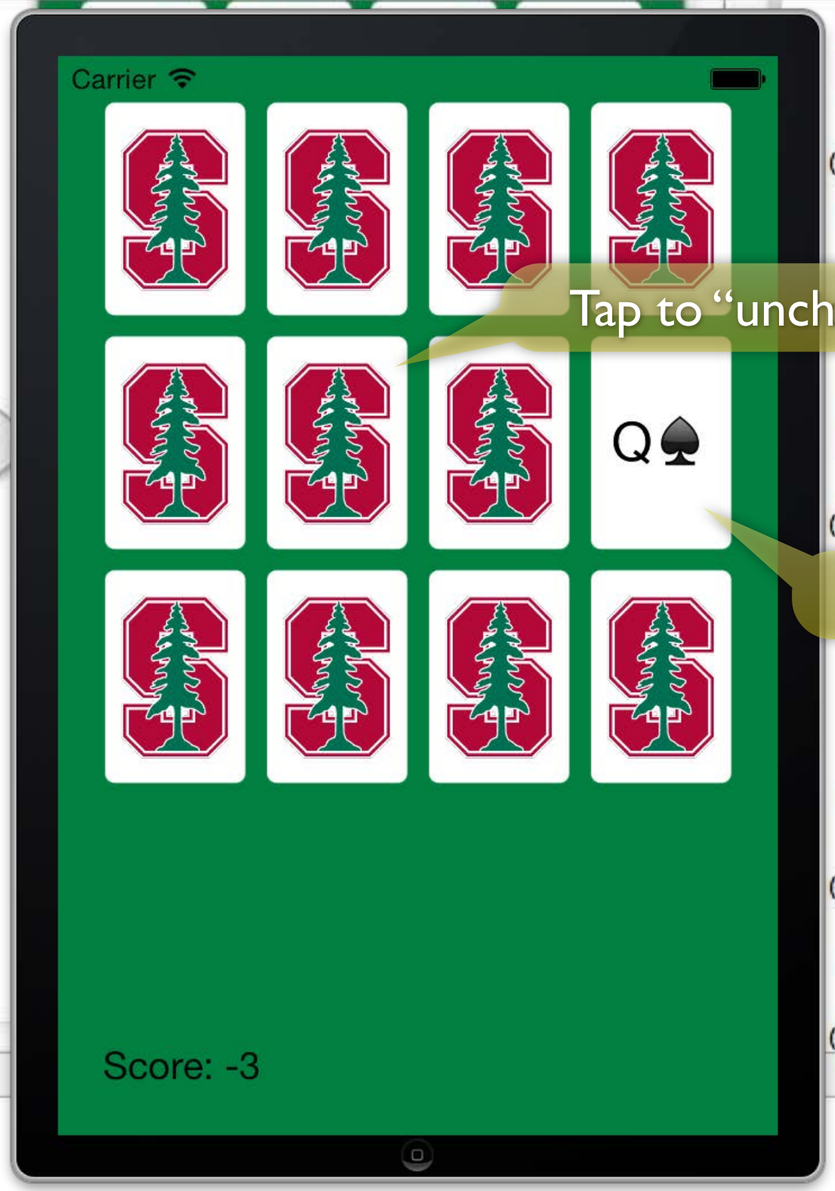
Tap to choose.

```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
return _game;

(Deck *)createDeck

return [[PlayingCardDeck alloc] init];

(IBAction)touchCardButton:(UIButton *)sender

int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];

(void)updateUI

for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}

(NSString *)titleForCard:(Card *)card

return card.isChosen ? card.contents : @"";

(UIImage *)backgroundImageForCard:(Card *)card
```

Matchismo.xcodeproj — Main.storyboard

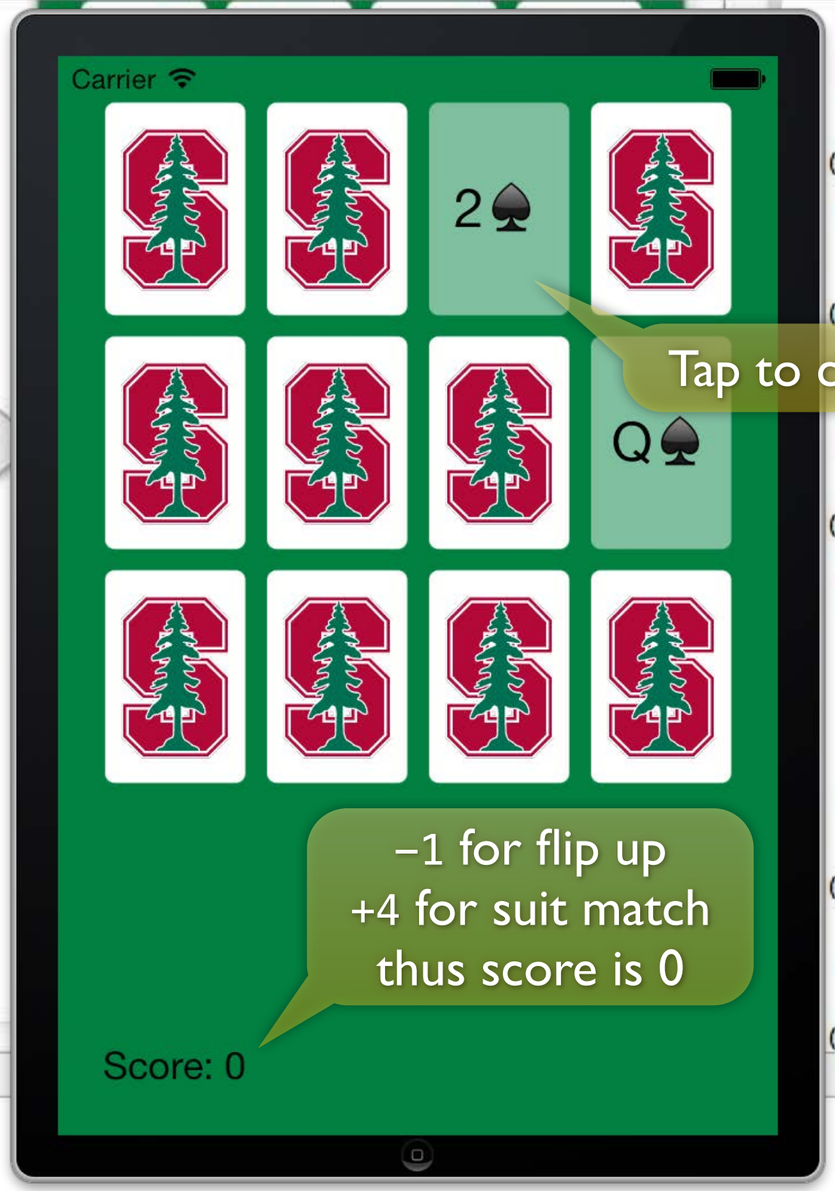
Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › @implementation CardGameViewController




```
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;

}

- (Deck *)createDeck

    return [[PlayingCardDeck alloc] init];

}

- (IBAction)touchCardButton:(UIButton *)sender

    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];

}

- (void)updateUI

    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }

- (NSString *)titleForCard:(Card *)card

    return card.isChosen ? card.contents : @"";

- (UIImage *)backgroundImageForCard:(Card *)card
```

Auto | All Output




```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
return _game;


(Deck *)createDeck

return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];


(void)updateUI

    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }


(NSString *)titleForCard:(Card *)card

    return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```


```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
return _game;


(Deck *)createDeck

return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];


(void)updateUI

for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
    self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
}


(NSString *)titleForCard:(Card *)card

return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › @implementation CardGameViewController | 2

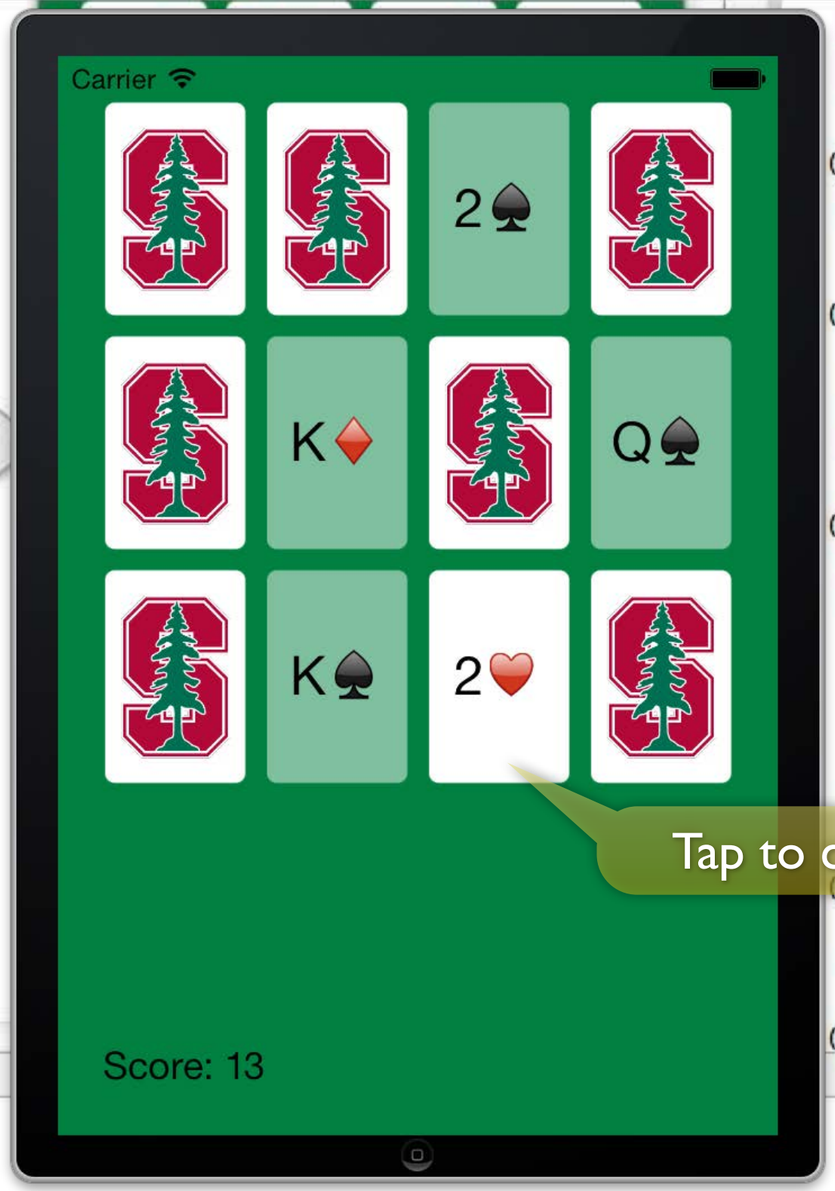

```
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;


(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];


(void)updateUI

    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }


(NSString *)titleForCard:(Card *)card

    return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```

Auto | All Output

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch) | No Issues

Card G... › View | Automatic › CardGameViewController.m › @implementation CardGameViewController




```
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;


(Deck *)createDeck

    return [[PlayingCardDeck alloc] init];


(IBAction)touchCardButton:(UIButton *)sender

    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];


(void)updateUI

    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }

(NSString *)titleForCard:(Card *)card

    return card.isChosen ? card.contents : @"";


(UIImage *)backgroundImageForCard:(Card *)card
```

Auto | All Output

Tap to choose (and cause mismatch!)

Automatically "unchosen" on mismatch.

Be sure to check for rank and suit matches and also for mismatches.

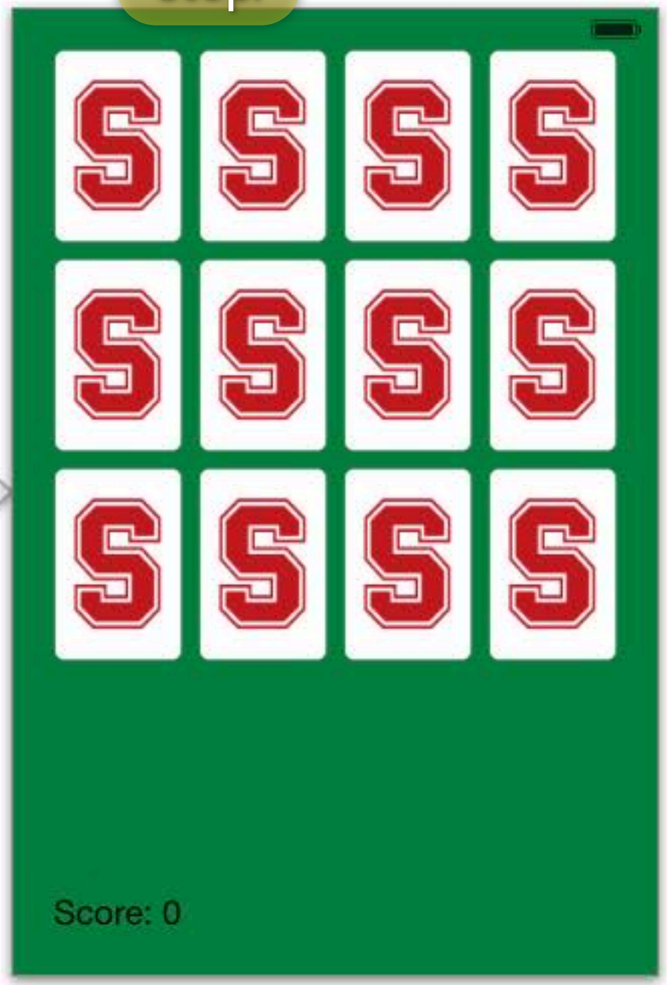
-1 for flip
-2 for mismatch!

```
if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                       usingDeck:[self createDeck]];
return _game;

(Deck *)createDeck

return [[PlayingCardDeck alloc] init];

(IBAction)touchCardButton:(UIButton *)sender

int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
[self.game chooseCardAtIndex:chosenButtonIndex];
[self updateUI];

for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
    cardButton.enabled = !card.isMatched;
}

(NSString *)titleForCard:(Card *)card

return card.isChosen ? card.contents : @"";

(UIImage *)backgroundImageForCard:(Card *)card
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | No Issues

Stop.



Card G... › View | Automatic › CardGameViewController.m › No Selection

```
@interface CardGameViewController ()
@property (strong, nonatomic) CardMatchingGame *game;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@property (weak, nonatomic) IBOutlet UILabel *scoreLabel;
@end

@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self backgroundImageForCard:card] forState:UIControlStateNormal];
        cardButton.enabled = !card.isMatched;
        self.scoreLabel.text = [NSString stringWithFormat:@"Score: %d", self.game.score];
    }
}

- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}
```

That's all folks!

Review

- Things you should know by now ...
  MVC
  Xcode
  Basic Objective-C

Review (MVC)

- Model is UI-independent
  Cards and Decks, not UIButtons and UILabels
- View is (so far) completely generic UI elements
  UIButton
  UILabel
- Controller interprets Model for View (and vice-versa)
  Example: converting isChosen to selected state of a button
  Example: converting isMatched to enabled state of a button
  Example: taking a button touch and turning it into a chooseCardAtIndex: in the Model
- Target/Action and Outlets (so far)

Review (Xcode)

- ## Create a Project and maneuver through Xcode's UI
  Hide/Show Navigator, Utilities, Assistant Editor, etc., etc. Also how to run in the Simulator.
- ## Edit
  Not just code, but also your storyboard, use Attributes Inspector to edit buttons, labels, et. al. Ctrl-drag to make connections (actions and outlets).
  Right click on buttons, etc., to find out about and disconnect connections.
  Look at warnings and errors (and get rid of them hopefully!). Debugger on Friday this week.
- ## Add classes to your project
  e.g. you added the `Card`, etc., Model classes in your Homework assignment.
- ## Use the documentation
  Many ways to get to documentation, but ALT-clicking on a keyword is one of the coolest. Once there, search and click on links to find what you want.
  Crucial to being a good iOS programming to become familiar with all the documentation.

Review (Basic Objective-C)

Classes

Header .h (public) versus Implementation .m (private)

`@interface MyClass : MySuperclass ... @end` (only in header file) `@interface MyClass() ... @end` (only in implementation file)

`@implementation ... @end` (only in implementation file)

`#import`

Properties

`@property (nonatomic)` <type> <property name> (always `nonatomic` in this course)

It's just setter and getter methods. Default ones automatically generated for you by compiler.

Better than instance variables alone (lazy instantiation, consistency checking, UI updating, etc.).

`@property (strong` or `weak)` <type which is a pointer to an object> <property name>

`@property (getter=`<getter name>`) ...`

`@property (readonly) ...` & `@property (readwrite) ...`

Invoking setter and getter using dot notation, e.g., `self.cards = ...` or `if (rank > self.rank) ...` `@synthesize` <prop name> = _<prop name> (only if you implement both setter and getter)

Review (Basic Objective-C)

Types and Memory

Types: `MyClass *`, `BOOL` (`YES` or `NO`), `int`, `NSUInteger`, etc. (`id` not fully explained yet.)

All objects live in the heap (i.e. we only have pointers to them).

Object storage in the heap is managed automatically (guided by `strong` and `weak` declarations). Lazy instantiation (using a `@property`'s getter to `allocate` and `initialize` the object that the @property points to in an "on demand" fashion). Not everything is lazily instantiated, btw. :) If a pointer has the value nil (i.e. 0), it means the pointer does not point to anything.

Methods

Declaring and defining instance methods, e.g., `- (int)match:(NSArray *)otherCards`

Declaring and defining class methods, e.g., `+ (NSArray *)validSuits`

Invoking instance methods, e.g., `[myArray addObject:anObject]`

Invoking class methods, e.g., `unsigned int rank = [PlayingCard maxRank]`

Method's name and its parameters are interspersed, e.g., `[deck addCard:aCard atTop:YES]`

Review (Basic Objective-C)

NSString

Immutable and usually created by manipulating other strings or @"" notation or class methods. e.g. `NSString *myString = @"hello"`

e.g. `NSString *myString = [otherString stringByAppendingString:yetAnotherString]`

e.g. `NSString *myString = [NSString stringWithFormat:@"%d%@", myInt, myObj]`

There is an `NSMutableString` subclass but we almost never use it.

Instead, we create new strings by asking existing ones to create a modified version of themselves.

Review (Basic Objective-C)

NSArray

Immutable and usually created by manipulating other arrays (not seen yet) or with @[] notation. `@[@"a",@"b"]` is the same as `[[NSArray alloc] initWithObjects:@"a",@"b",nil]`.

Access the array using `[]` notation (like a normal C array), e.g., `myArray[index]`.

`myArray[index]` works the same as `[myArray objectAtIndex:index]`.

The method `count` (which returns `NSUInteger`) will tell you how many items in the array.

(We accidentally used dot notation to call this method in Lecture 2!)

Be careful not to access array index out of bounds (crashes). Only `last/firstObject` immune. Can contain any mix of objects of any class) No syntax to say which it contains.

Use `NSMutableArray` subclass if mutability is needed. Then you get ...

```
- (void)addObject:(id)anObject;
- (void)insertObject:(id)anObject atIndex:(int)index;
- (void)removeObjectAtIndex:(int)index;
```

Usually created with `[[NSMutableArray alloc] init]`

Review (Basic Objective-C)

Creating Objects in the Heap

Allocation (NSObject's `alloc`) and initialization (with an `init`... method) always happen together!

e.g. `[[NSMutableArray alloc] init]`

e.g. `[[CardMatchingGame alloc] initWithCardCount:c usingDeck:d]`

Writing initializers for your own classes ...

Two kinds of initializers: designated (one per class) and convenience (zero or more per class). Only denoted by comments (not enforced by the syntax of the language in any way).

Must call your `super`'s designated initializer (from your designated initializer) or your own designated initializer (from your own convenience initializers).

This whole concept takes some getting used to.

Luckily, because of lazy instantiation, et. al., we don't need initializers that much in Objective-C. And calling initializers is easy (it's just `alloc` plus whatever `init`ializer you can find that you like).

Review (Basic Objective-C)

- Other

Fast enumeration: `for (MyClass *myObject in arrayOfMyObjects) { }`.

`#define`

`NSLog(@“show this object %@ in the console”, anObject)`

- Quiz

What does this do?

```
cardA.contents = @[cardB.contents,cardC.contents][[cardB match:@[cardC]] ? 1 : 0]
```

This line has a setter, getters, method invocation, array creation and array accessing all in one. And lots of square brackets.

Coming Up

- **Next Lecture**
  - More detail about Objective-C
  - More Foundation classes (besides strings and arrays)
  - Attributed strings
- **Next week ...**
  - Multiple MVCs in your storyboard
  - View Controller Lifecycle