Stanford CS193p

Developing Applications for iOS
Fall 2013-14



Today

- What is this class all about?
  Description
  Prerequisites
  Homework / Final Project
- iOS Overview
  What's in iOS?
- MVC
  Object-Oriented Design Concept
- Objective C
  (Time Permitting)
  New language!
  Basic concepts only for today.

What will I learn in this course?

- How to build cool apps
  - Easy to build even very complex applications
  - Result lives in your pocket or backpack!
  - Very easy to distribute your application through the AppStore
  - Vibrant development community
- Real-life Object-Oriented Programming
  - The heart of Cocoa Touch is 100% object-oriented
  - Application of MVC design model
  - Many computer science concepts applied in a commercial development platform:
    - Databases, Graphics, Multimedia, Multithreading, Animation, Networking, and much, much more!
  - Numerous students have gone on to sell products on the AppStore

Prerequisites

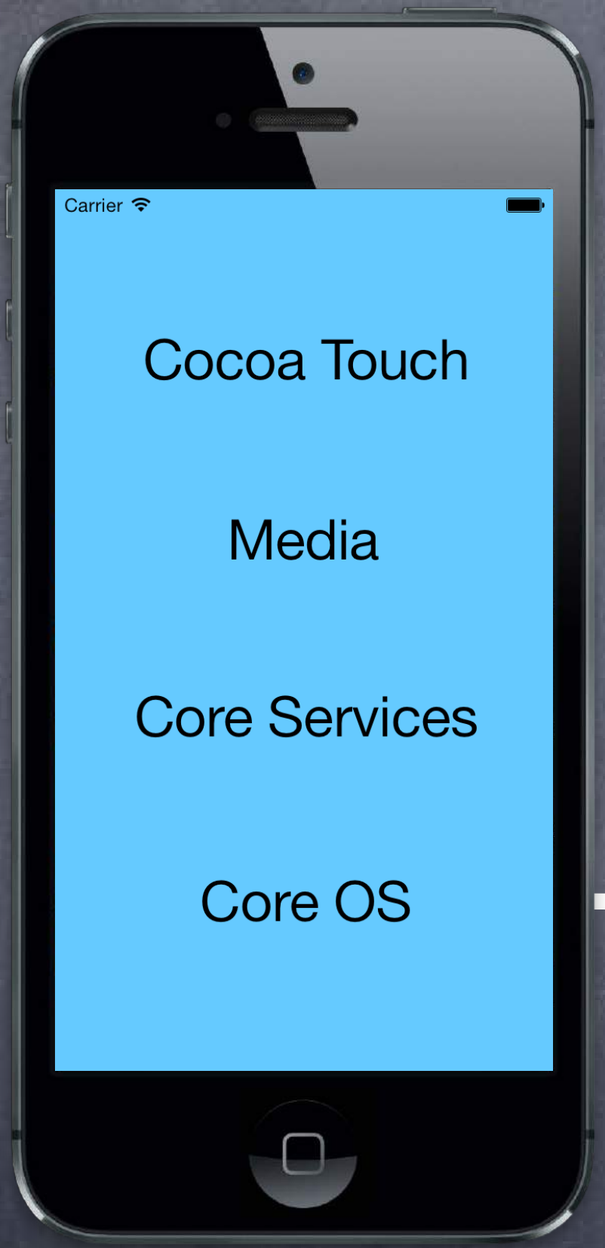
- **Most Important Prereq!**
  Object-Oriented Programming
  CS106A&B (or X) required
  CS107 or CS108 or CS110 required
  (or equivalent for non-Stanford undergrad)
- **Object-Oriented Terms**
  Class (description/template for an object)
  Instance (manifestation of a class)
  Message (sent to object to make it act)
  Method (code invoked by a Message)
  Instance Variable (object-specific storage)
  Superclass/Subclass (Inheritance)
- **You should know these terms!**
  If you are not very comfortable with all of these, this might not be the class for you!
- **Programming Experience**
  This is an upper-level CS course.

  If you have never written a program where you had to design and implement more than a handful of classes, this will be a big step up in difficulty for you.

Assignments

- Weekly Homework
  - 6 weekly (approximately) assignments
  - Individual work only
  - Required Tasks and Evaluation criteria
- Final Project
  - 3 weeks to work on it
  - Individual work only
  - Keynote presentation required (2 mins or so)

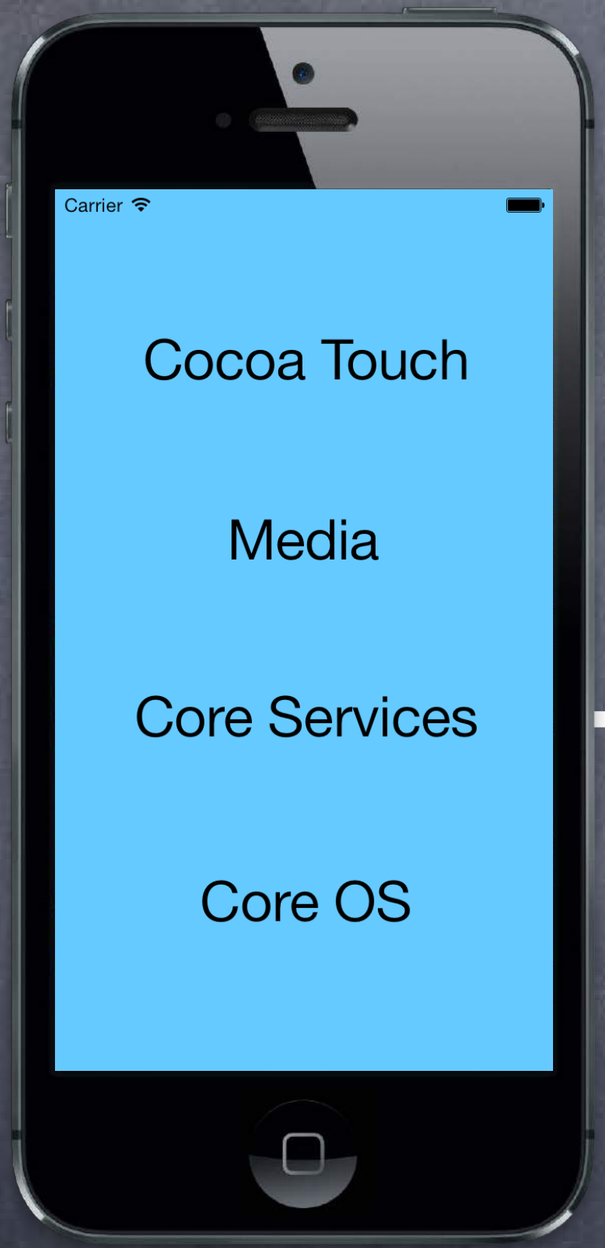
What's in iOS?



Core OS

OSX Kernel	Power Management
Mach 3.0	Keychain Access
BSD	Certificates
Sockets	File System
Security	Bonjour

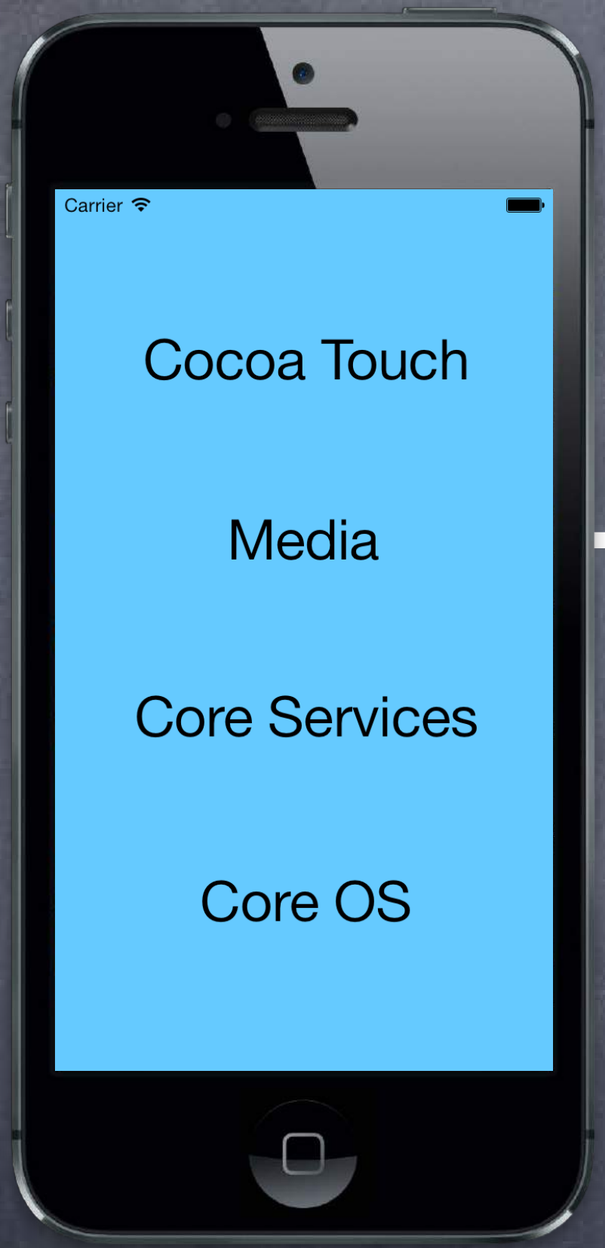
What's in iOS?



Core Services

Collections	Core Location
Address Book	Net Services
Networking	Threading
File Access	Preferences
SQLite	URL Utilities

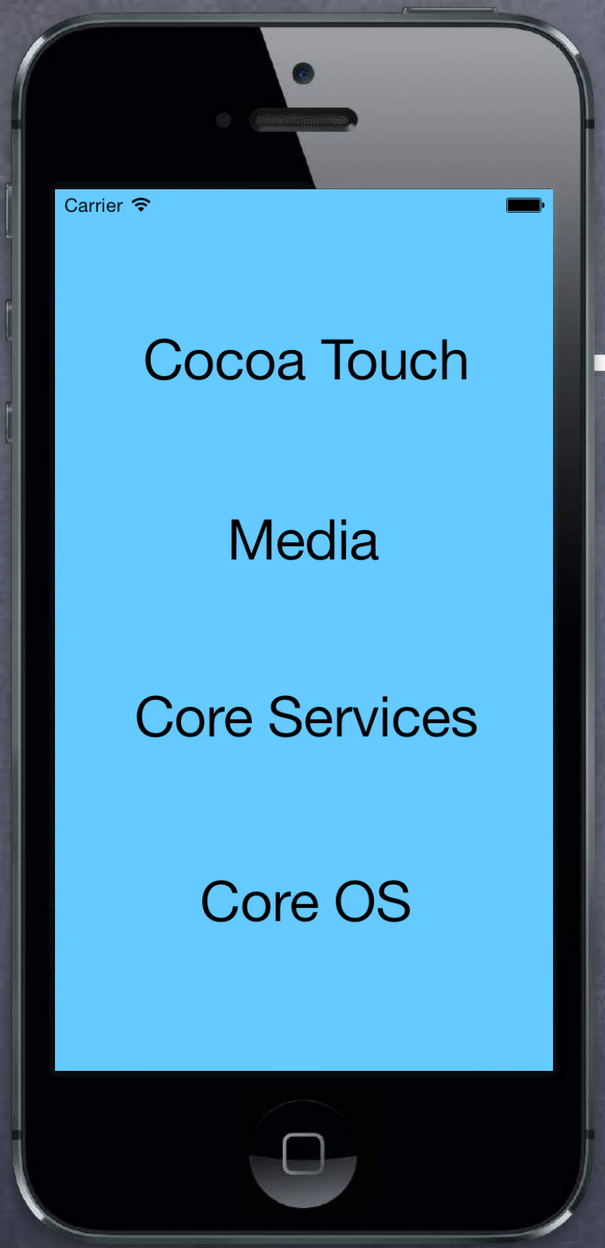
What's in iOS?



Media

Core Audio	JPEG, PNG, TIFF
OpenAL	PDF
Audio Mixing	Quartz (2D)
Audio Recording	Core Animation
Video Playback	OpenGL ES

What's in iOS?



Cocoa Touch

- Multi-Touch
- Core Motion
- View Hierarchy
- Localization
- Controls
- Alerts
- Web View
- Map Kit
- Image Picker
- Camera

Platform Components

- Tools



Xcode 5



Instruments

- Language

```
[display setTextColor:[UIColor blackColor]];
```

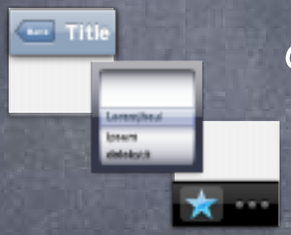
- Frameworks



Foundation

Core Data

Map Kit



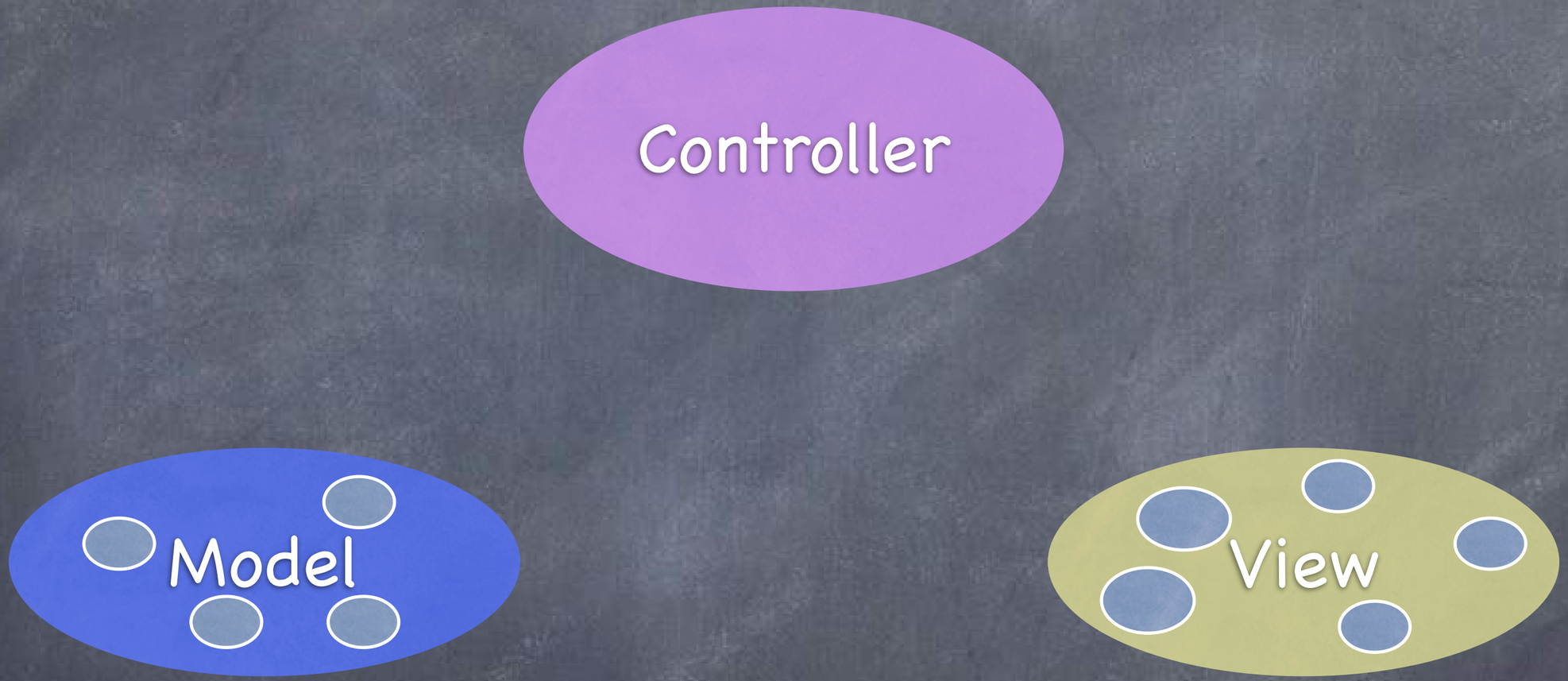
UIKit

Core Motion

- Design Strategies

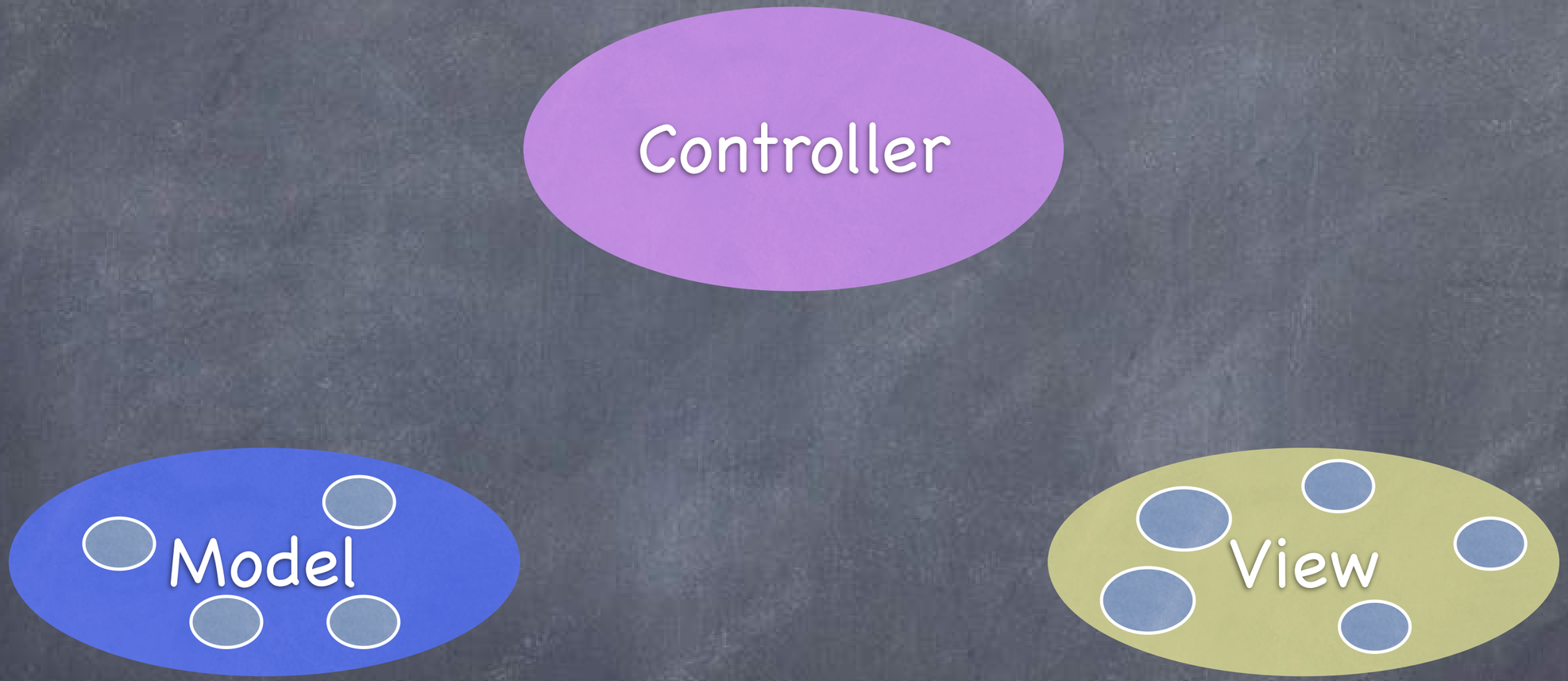
MVC

MVC



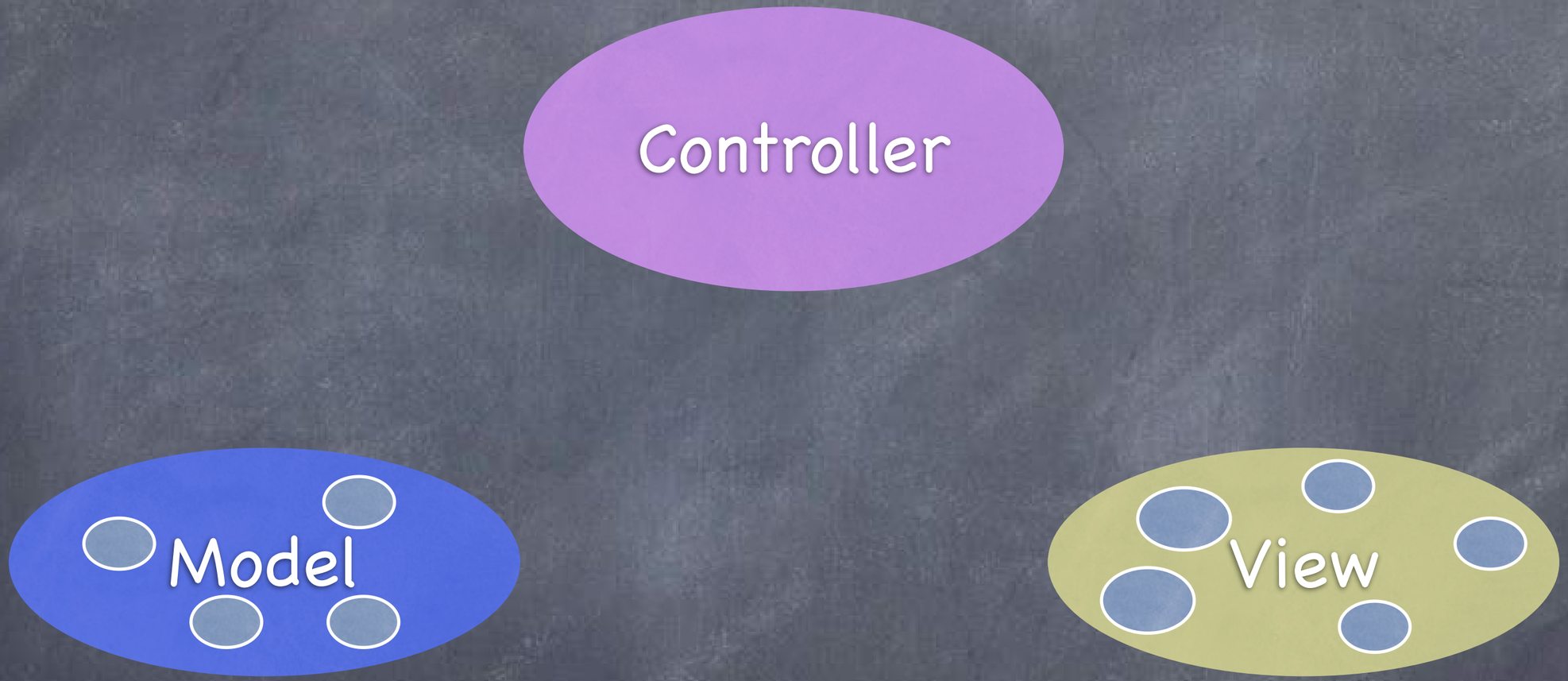
Divide objects in your program into 3 "camps."

MVC



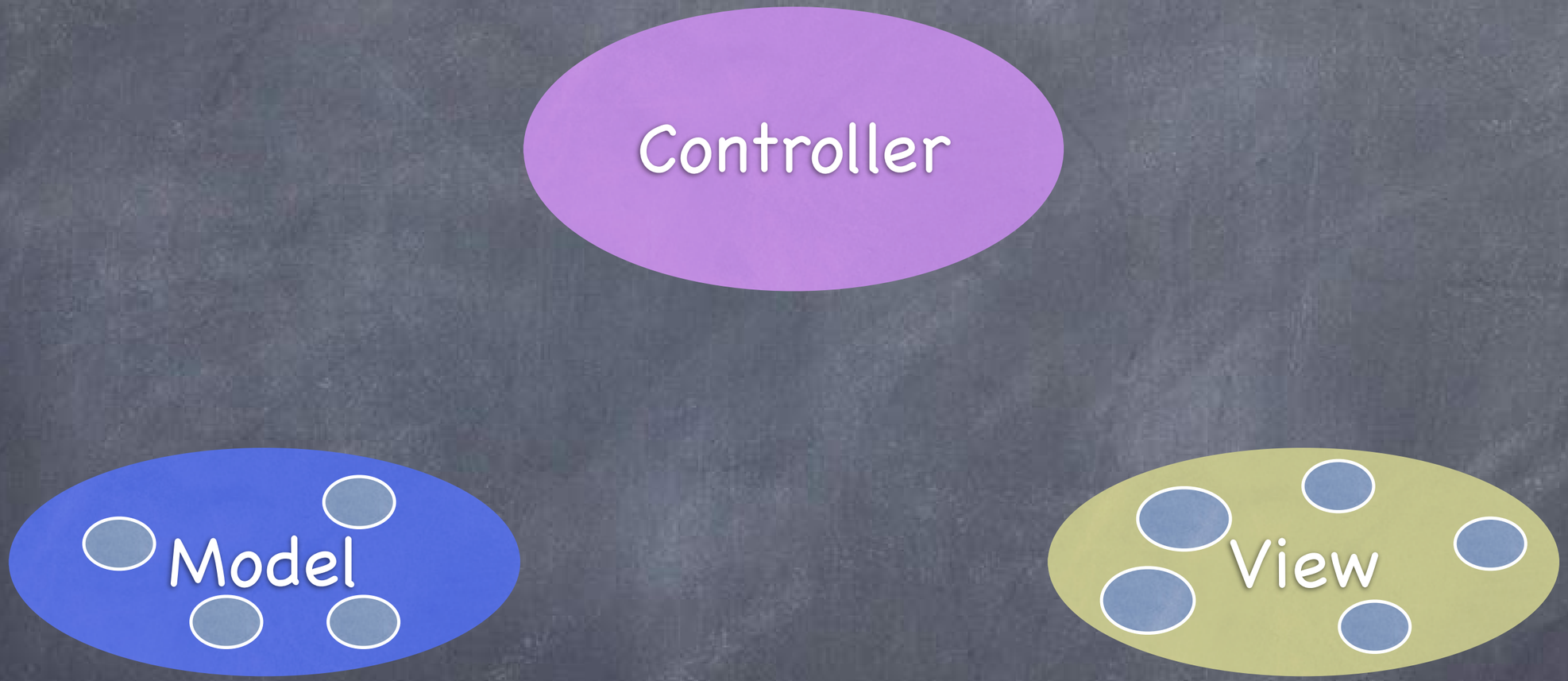
Model = What your application is (but not how it is displayed)

MVC



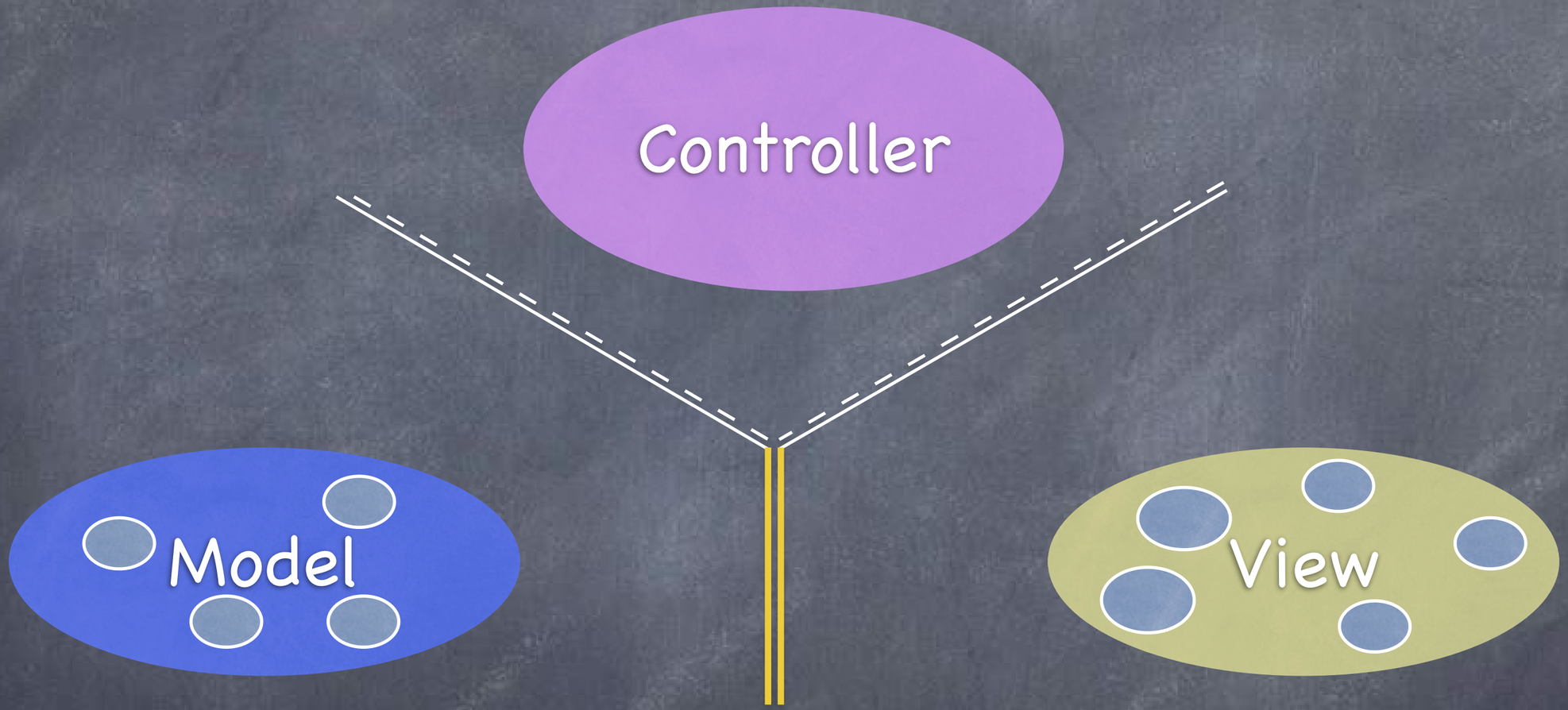
Controller = How your Model is presented to the user (UI logic)

MVC



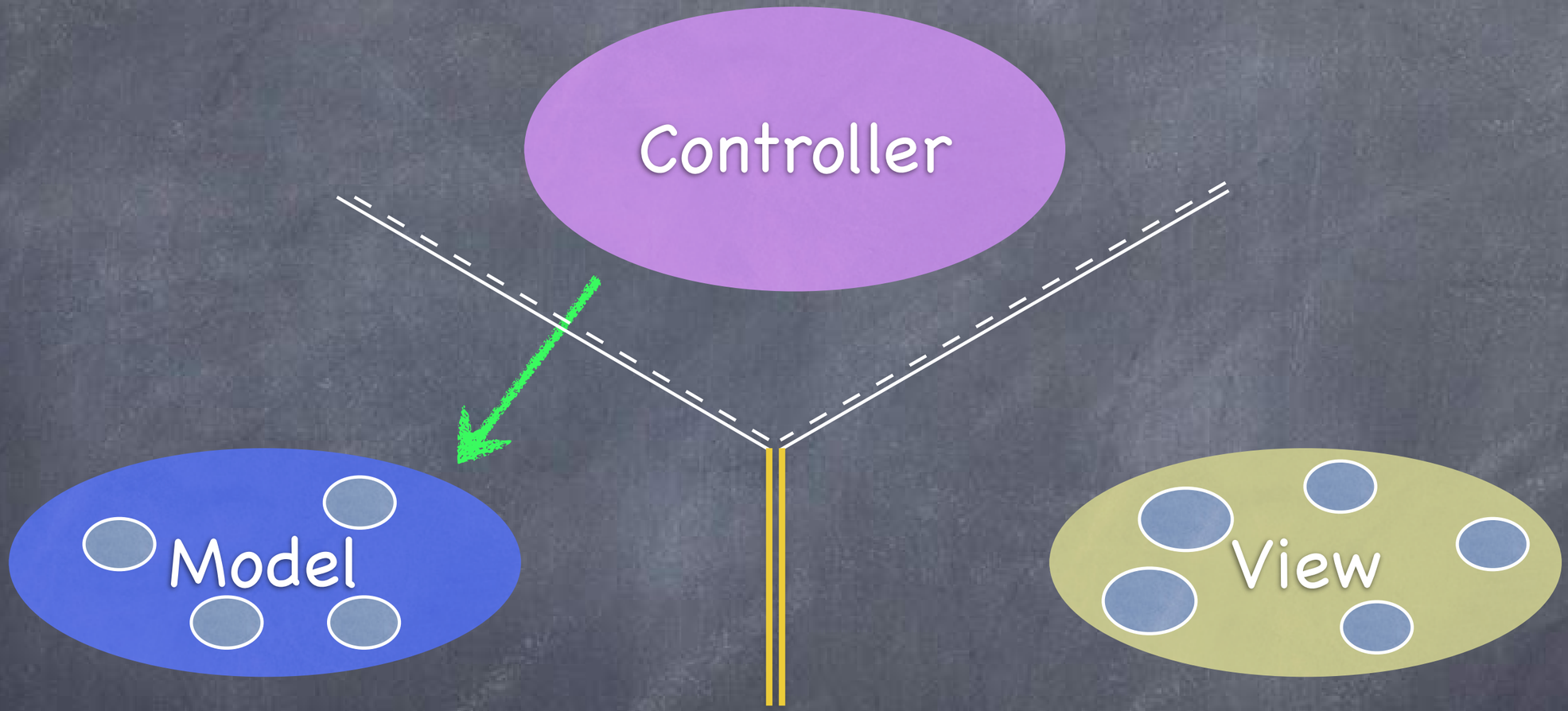
View = Your Controller's minions

MVC



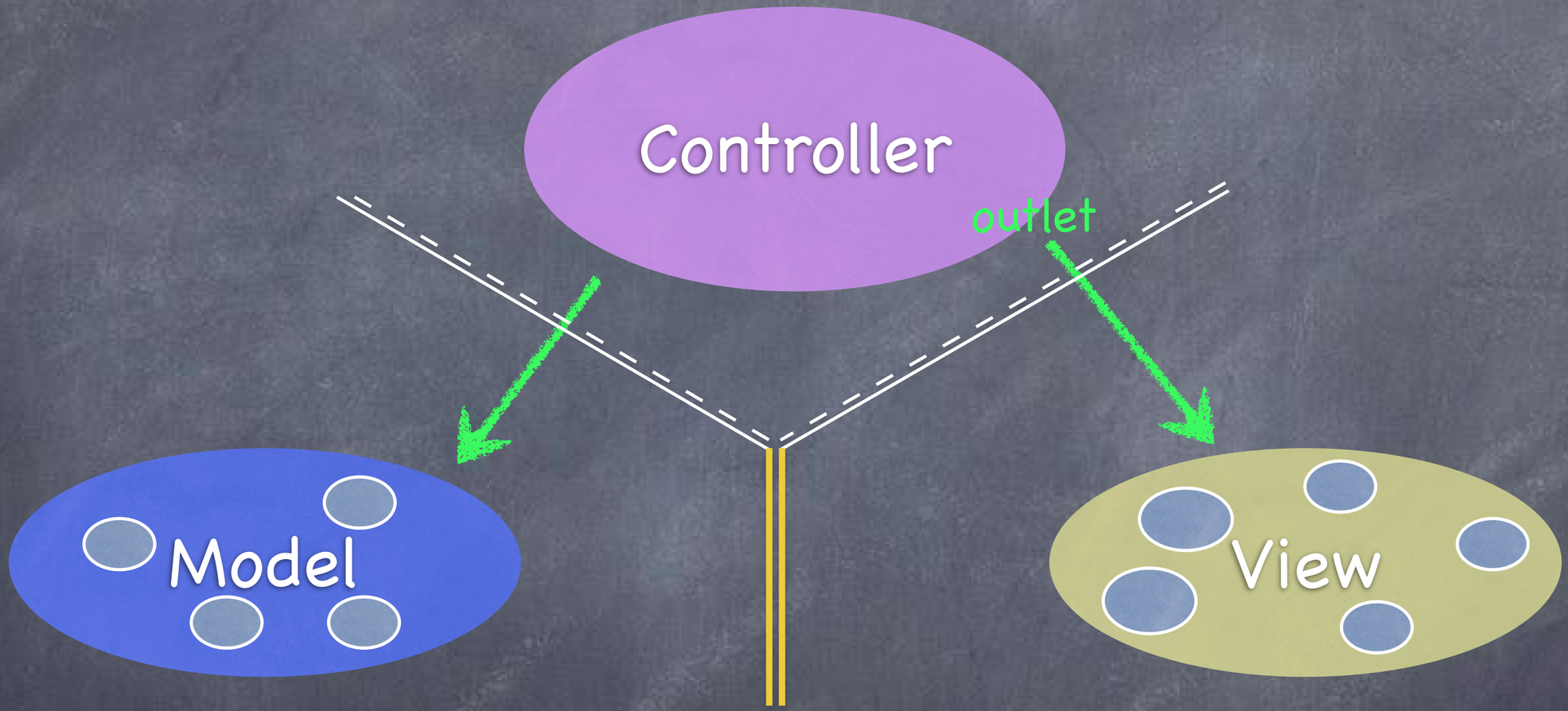
It's all about managing communication between camps

MVC



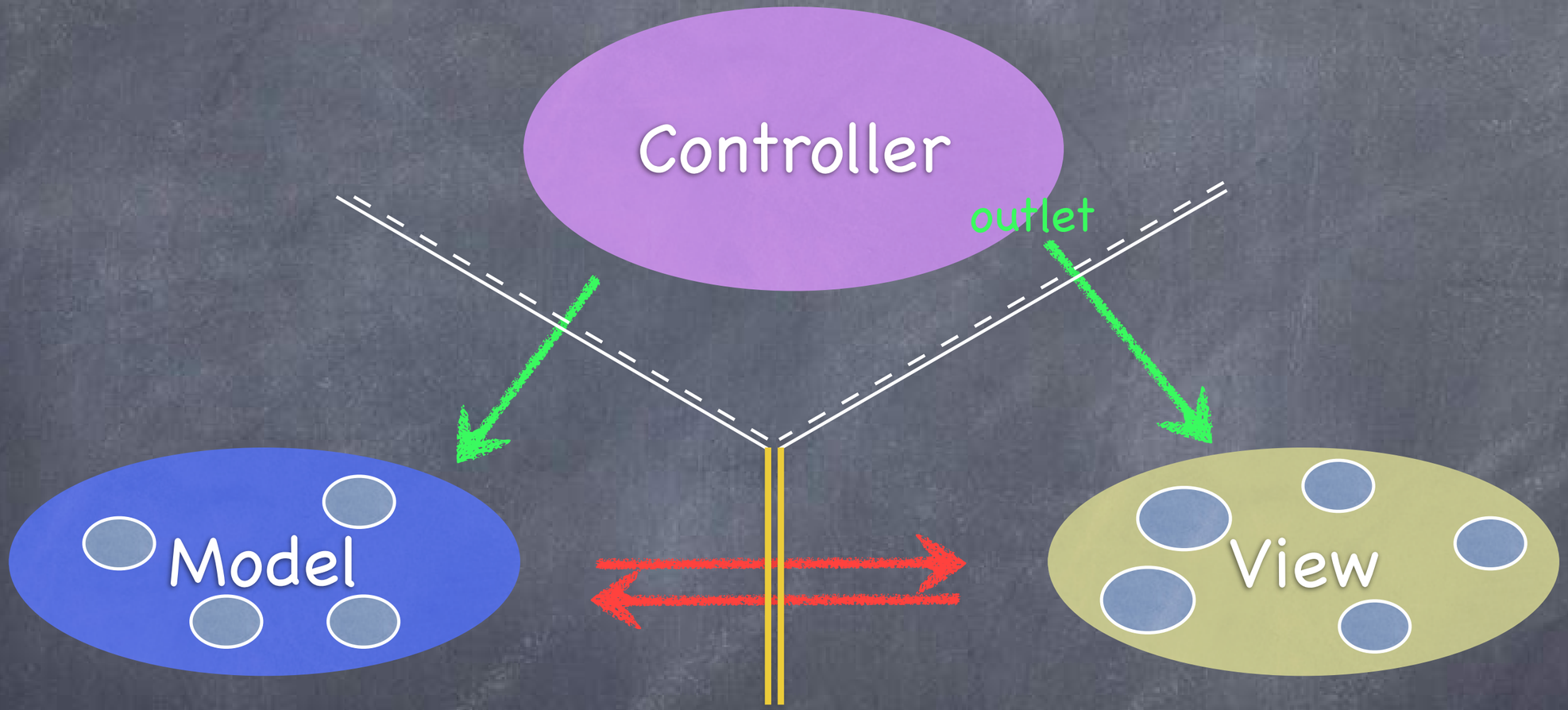
Controllers can always talk directly to their Model.

MVC



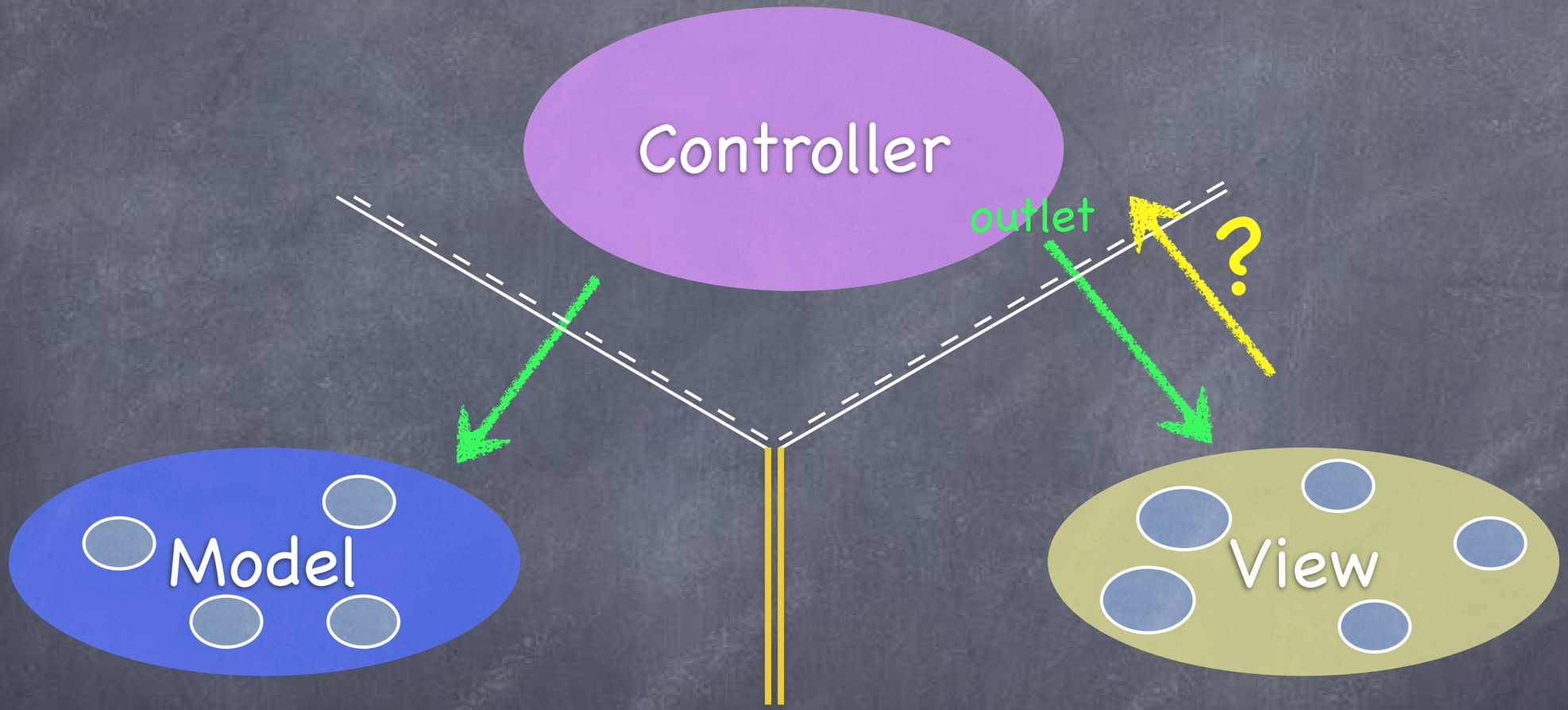
Controllers can also talk directly to their View.

MVC



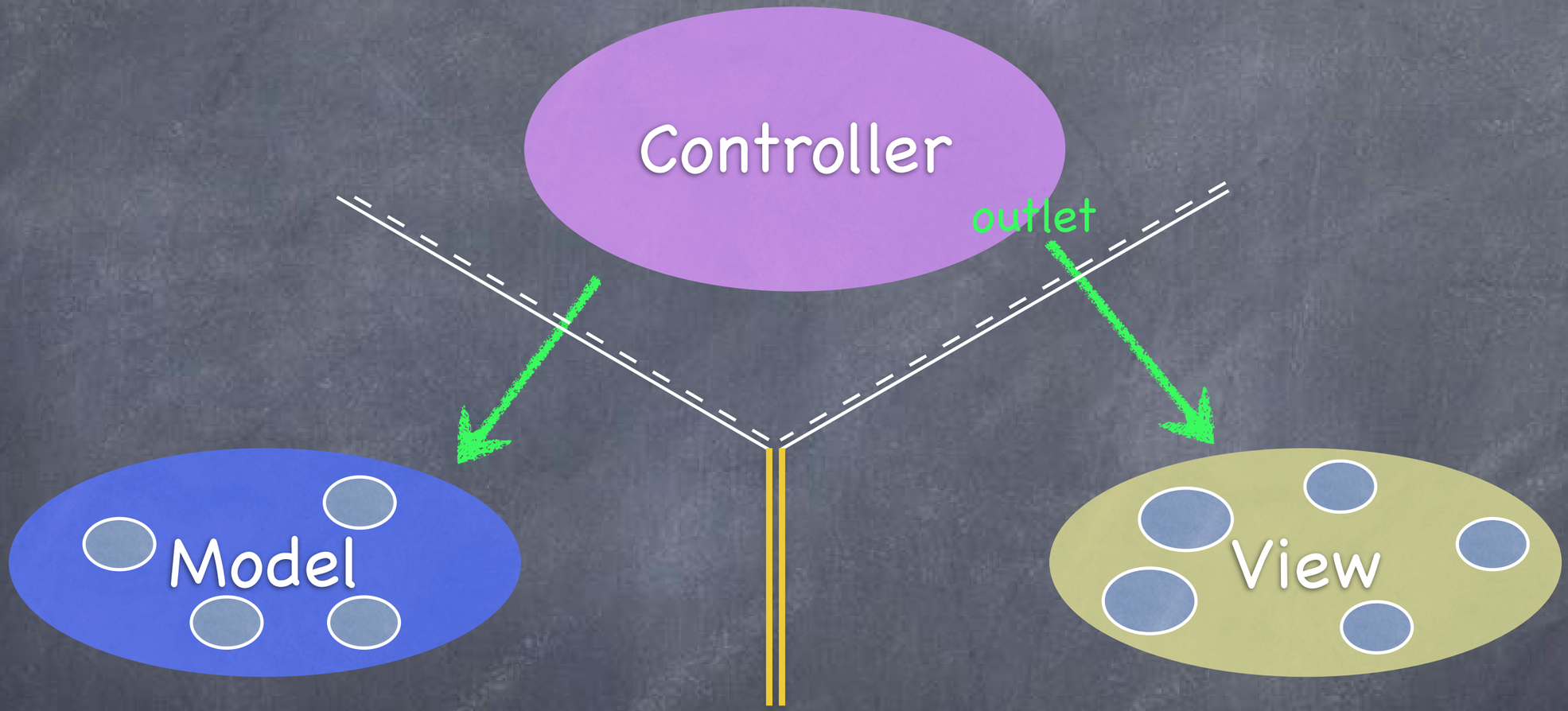
The Model and View should never speak to each other.

MVC



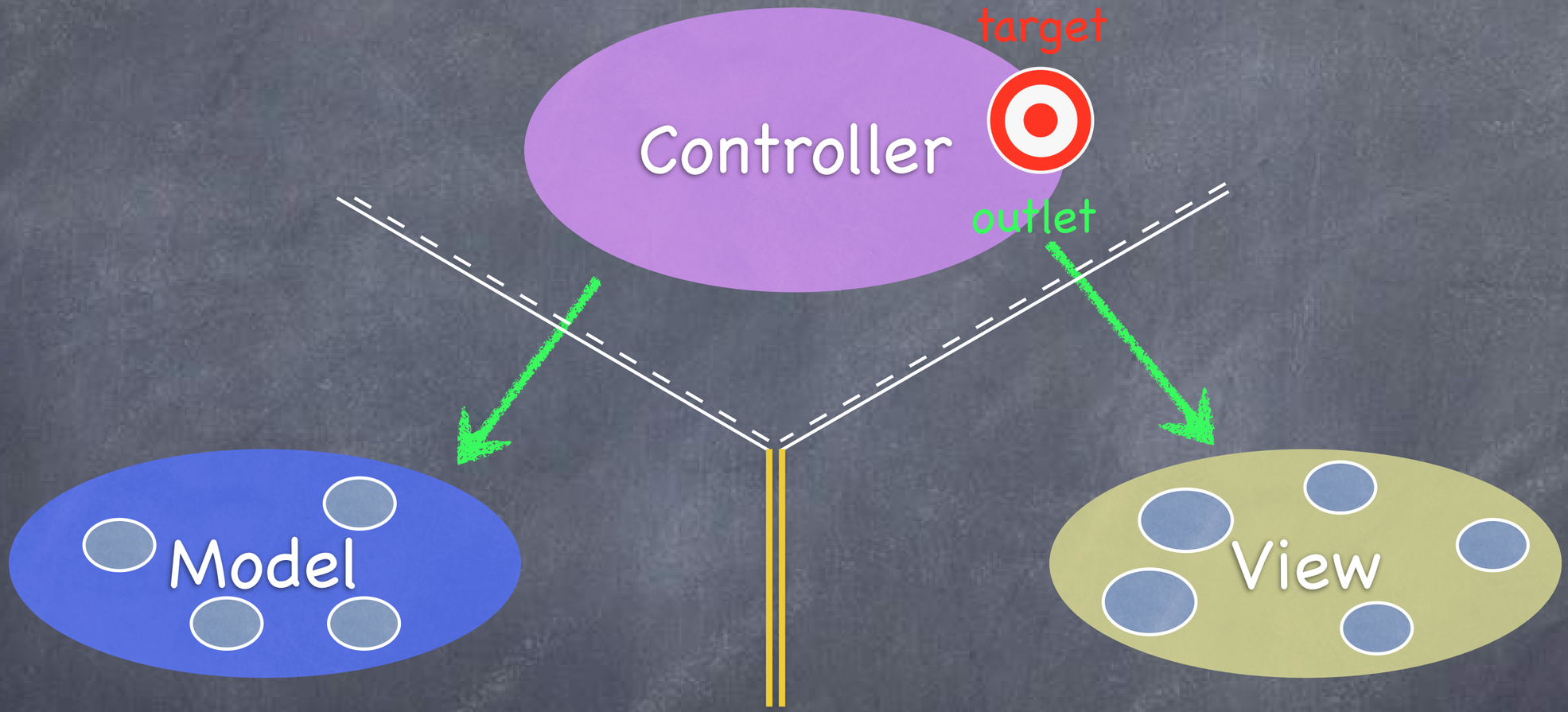
Can the View speak to its Controller?

MVC



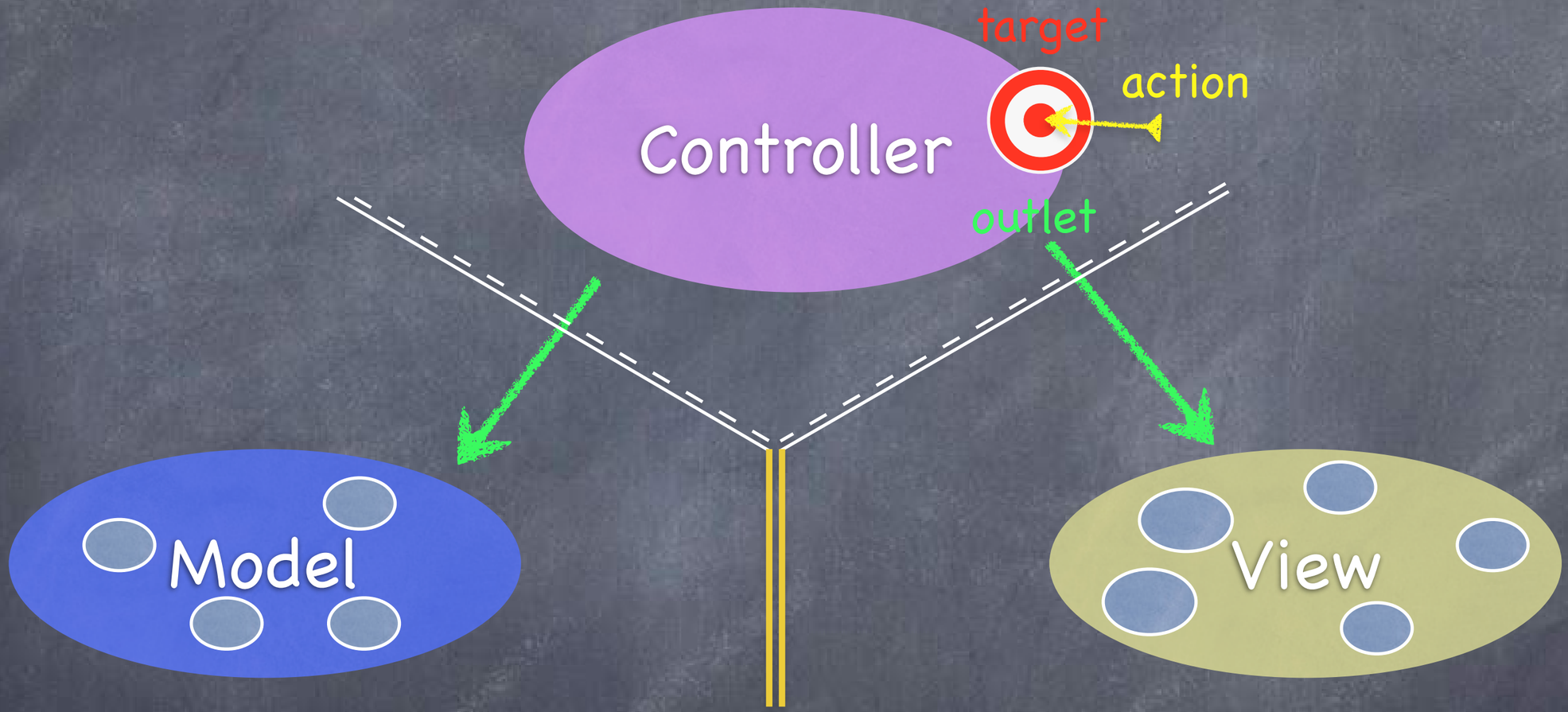
Sort of. Communication is "blind" and structured.

MVC



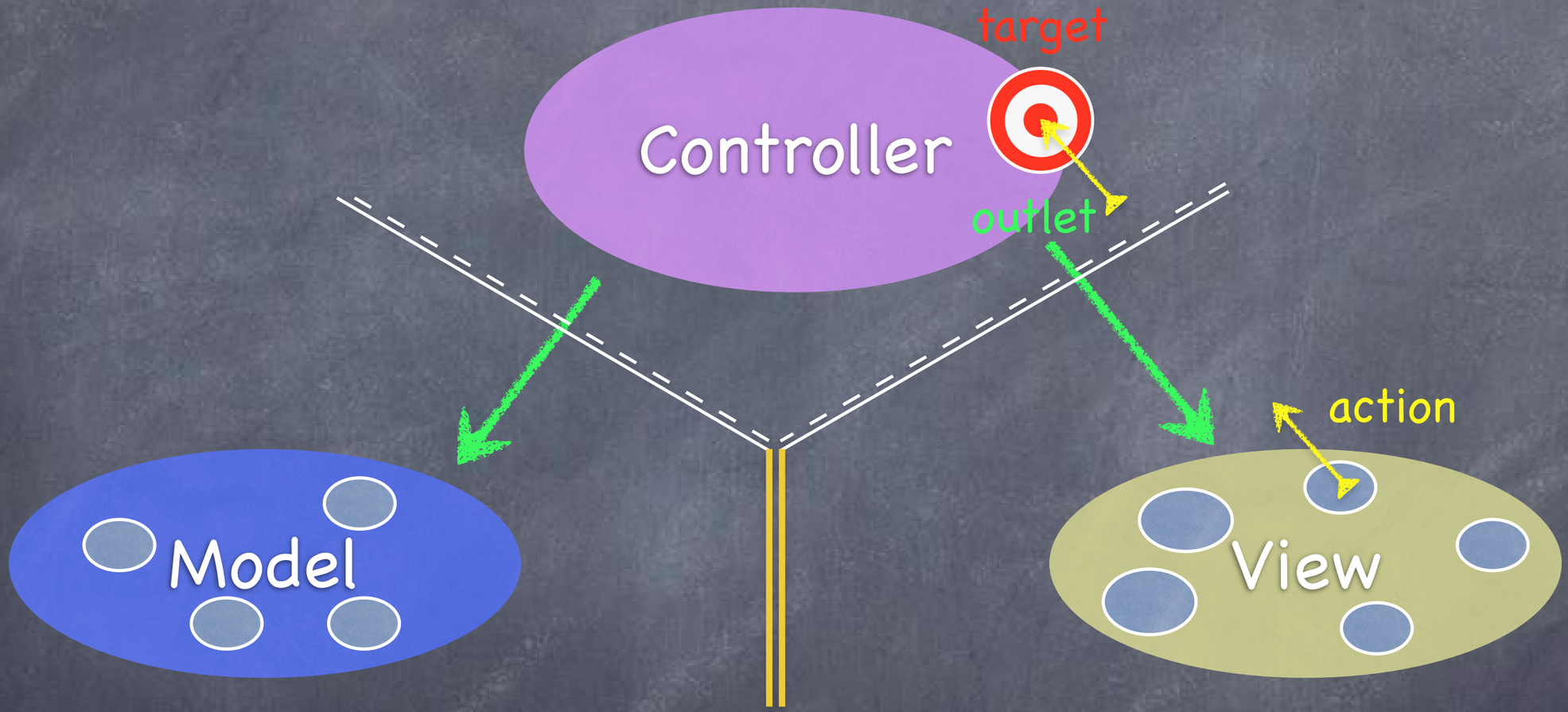
The Controller can drop a target on itself.

MVC



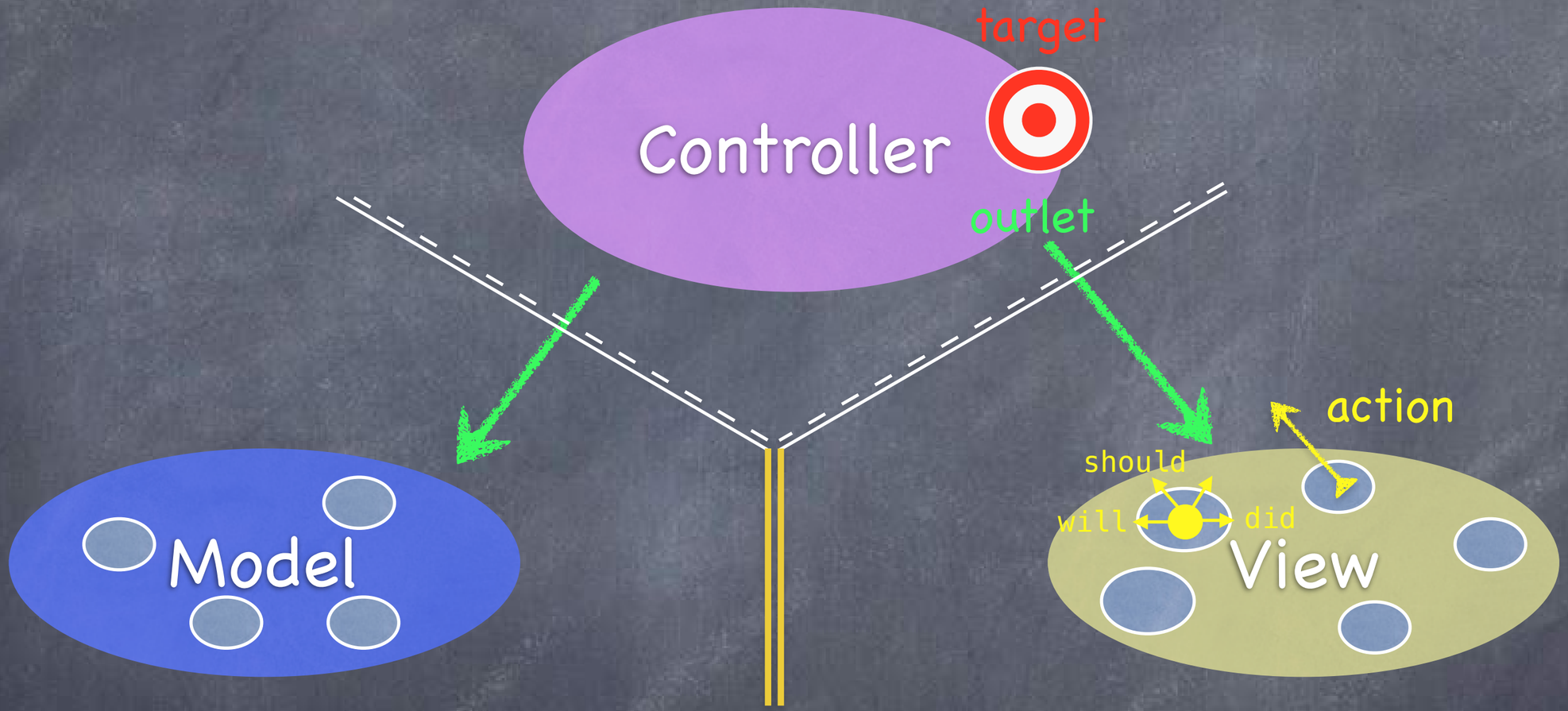
Then hand out an action to the View.

MVC



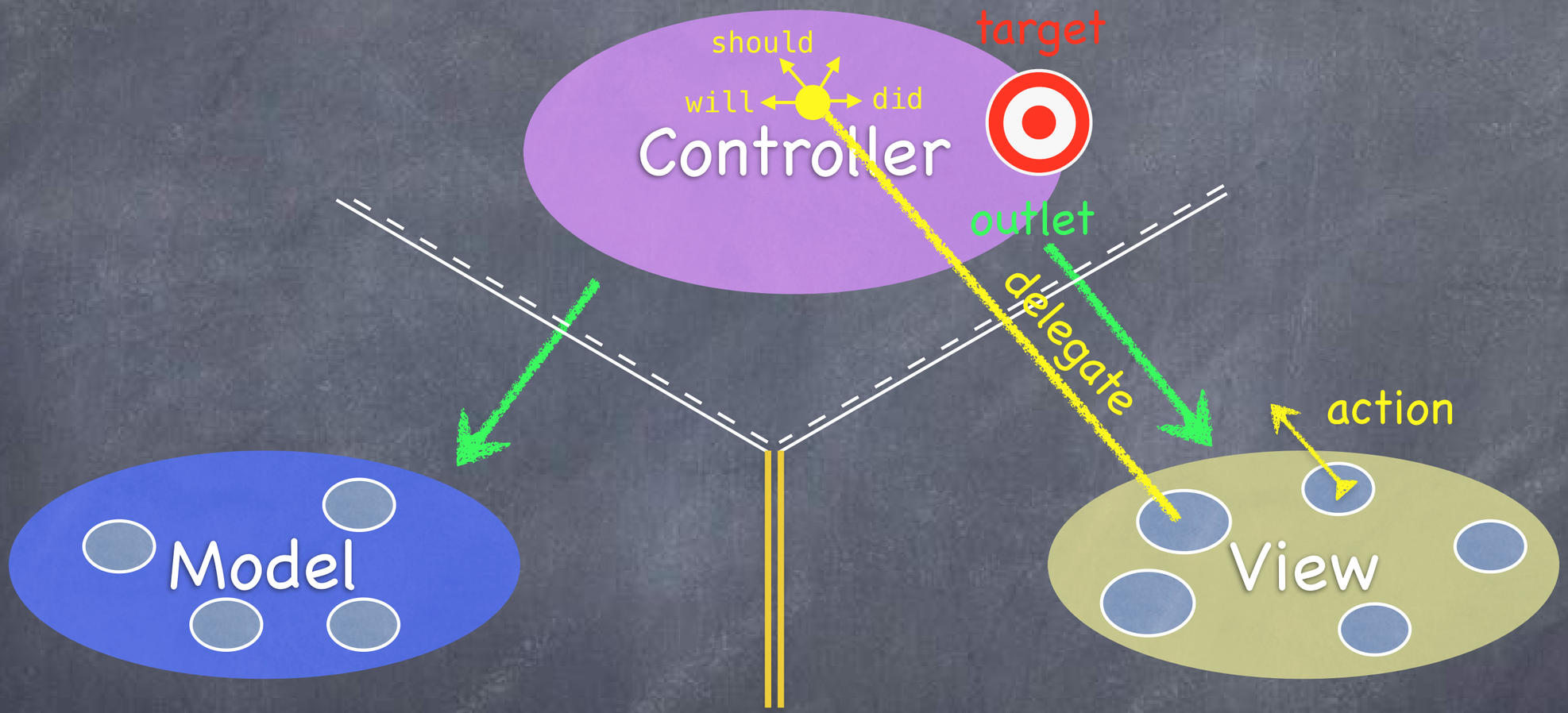
The View sends the action when things happen in the UI.

MVC



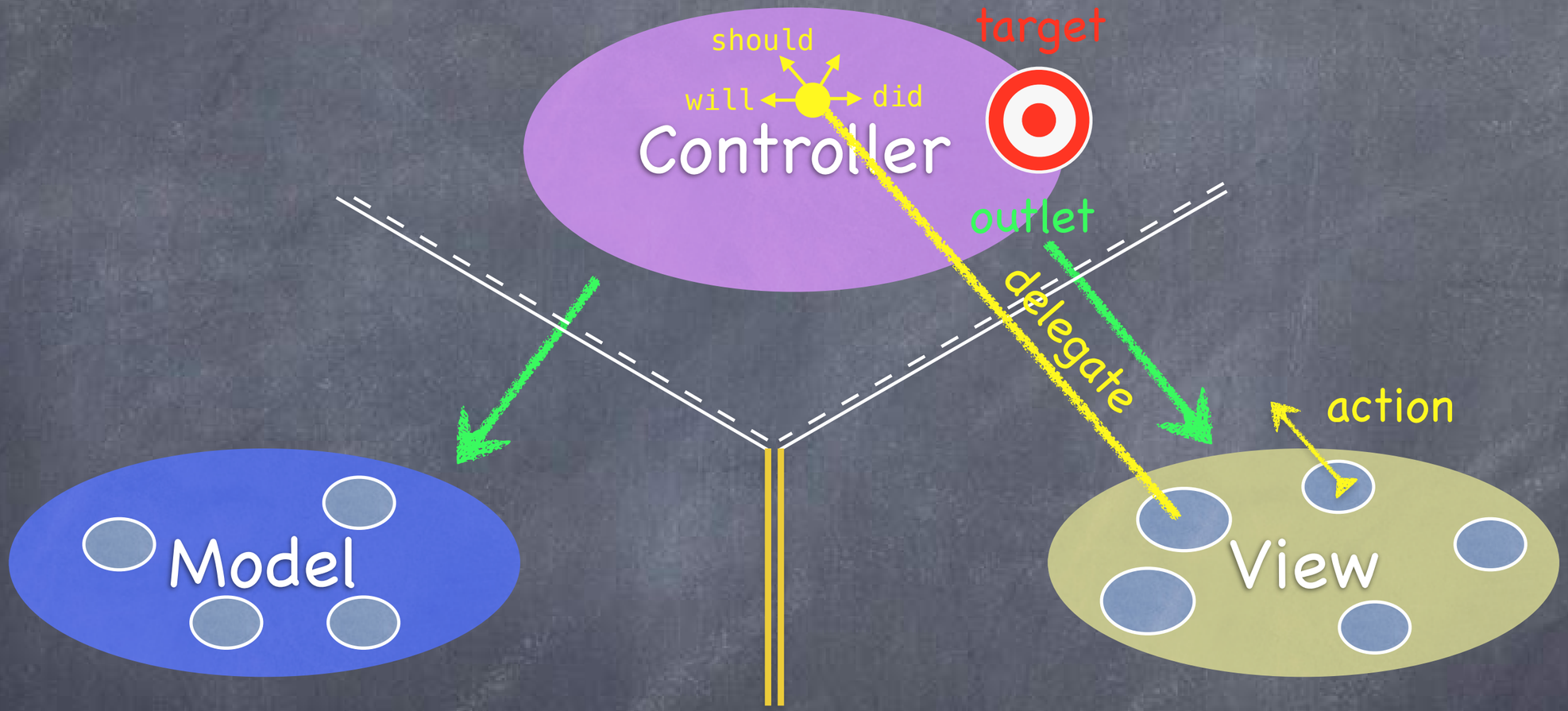
Sometimes the View needs to synchronize with the Controller.

MVC



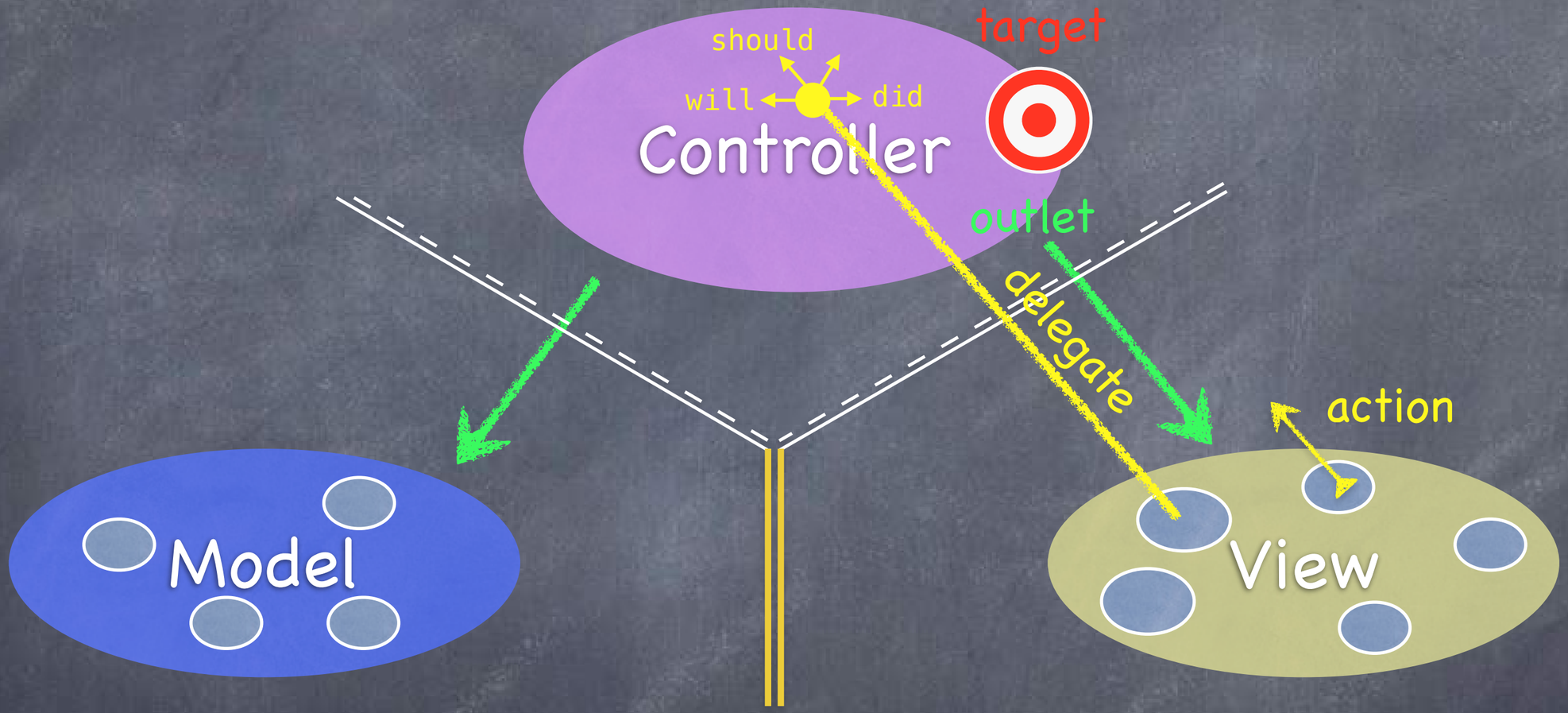
The Controller sets itself as the View's delegate.

MVC



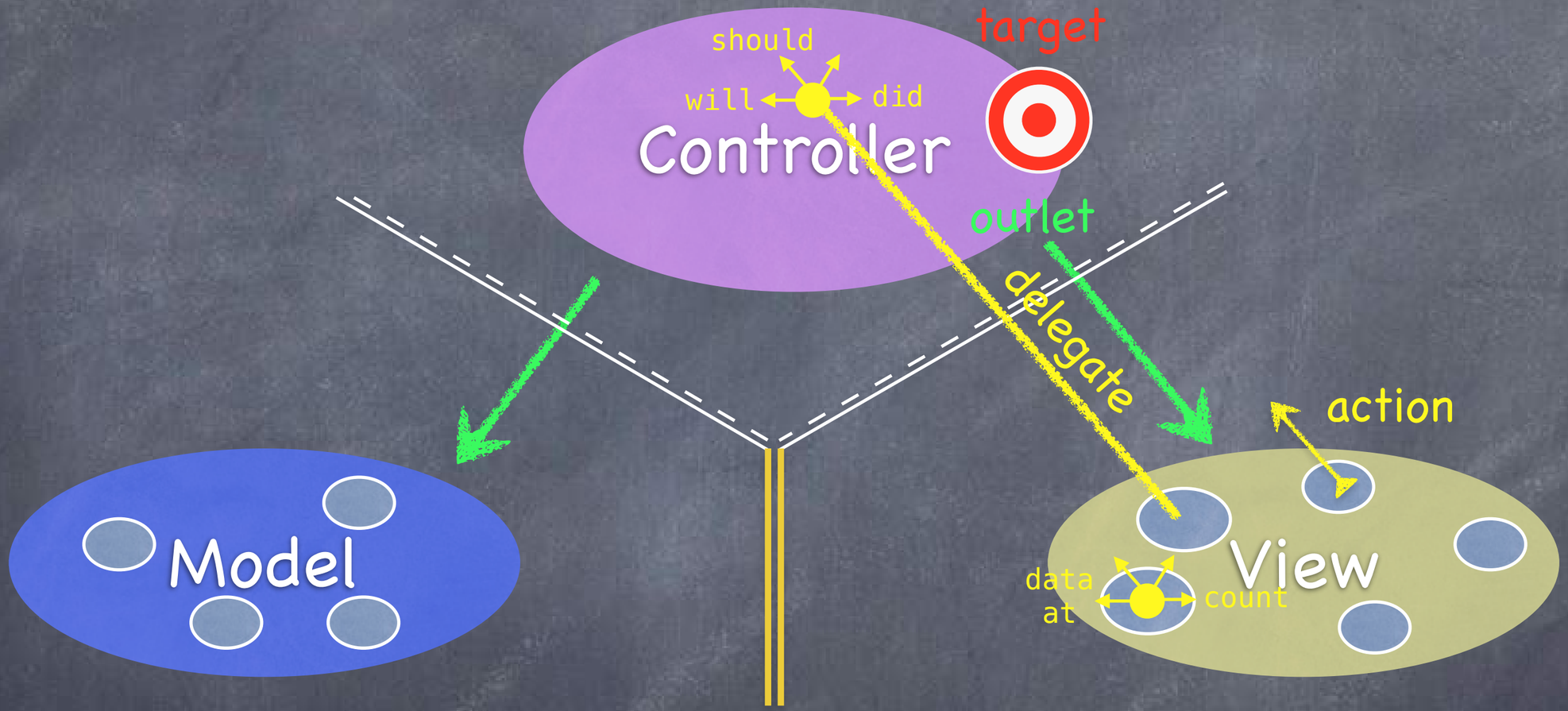
The delegate is set via a protocol (i.e. it's "blind" to class).

MVC



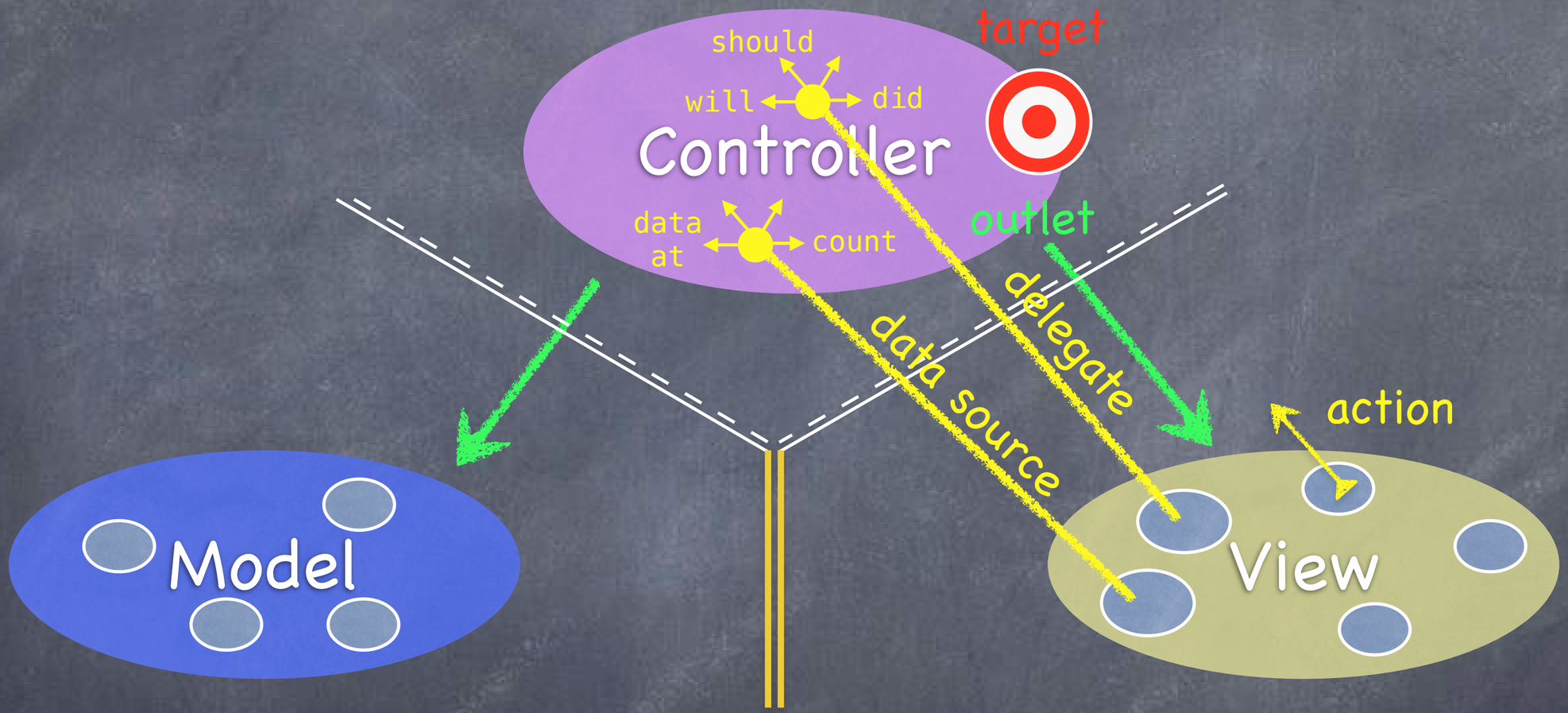
Views do not own the data they display.

MVC



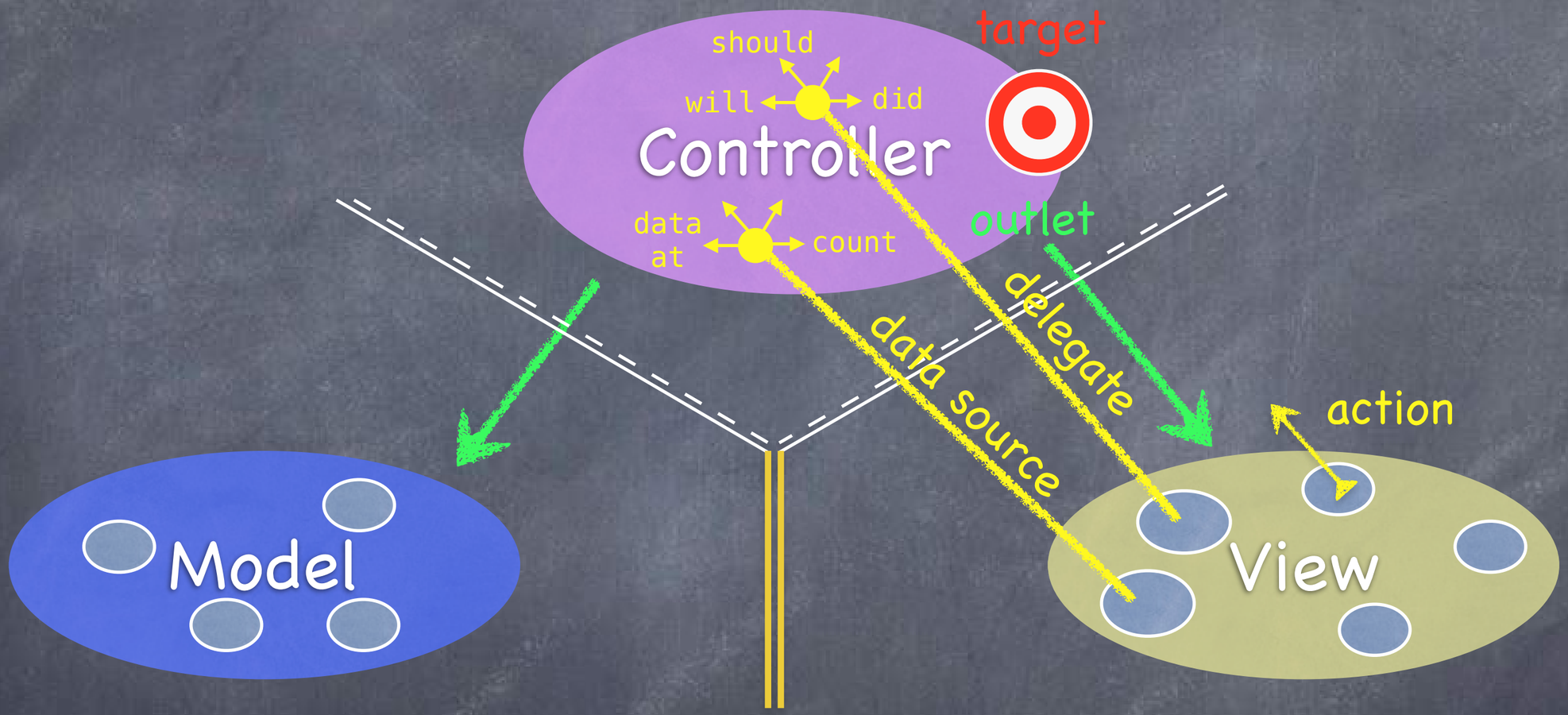
So, if needed, they have a protocol to acquire it.

MVC



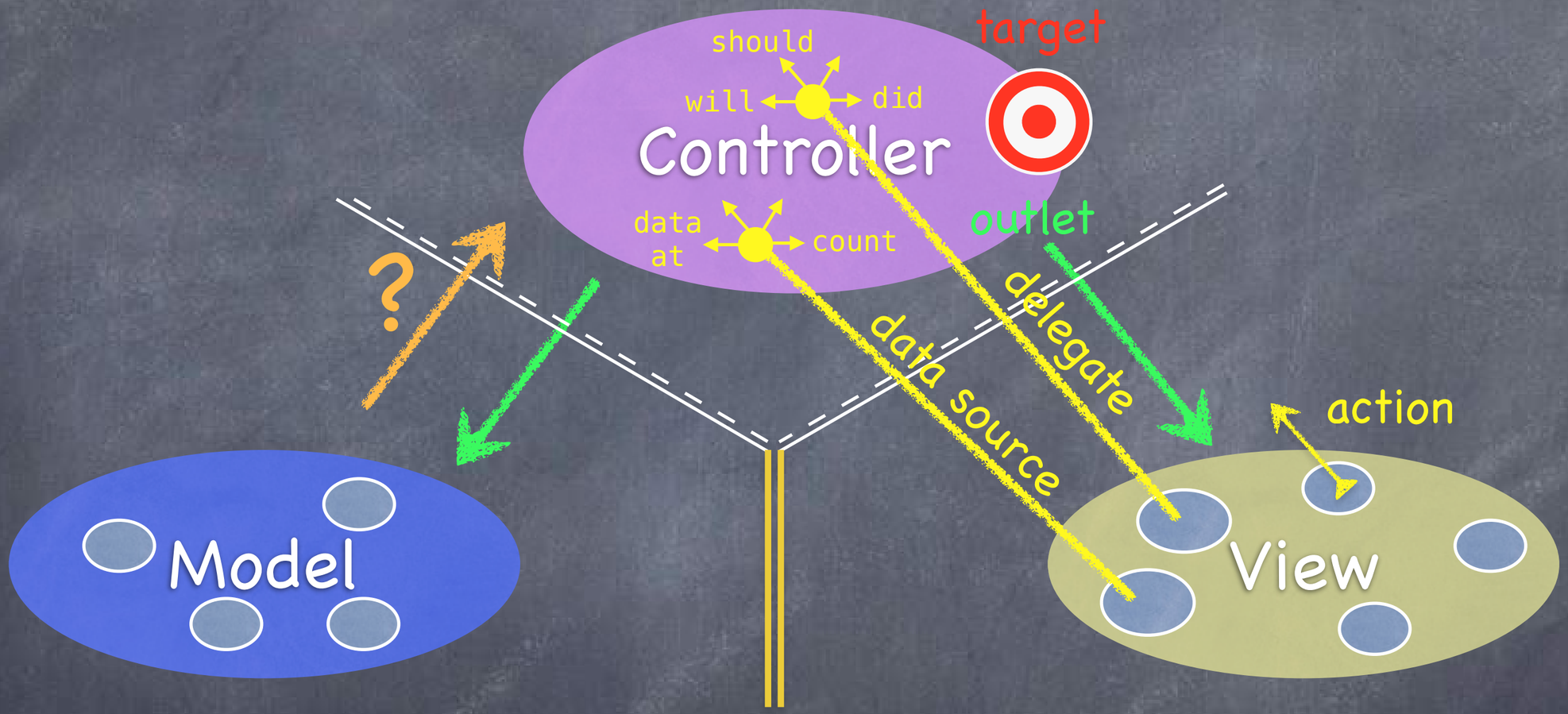
Controllers are almost always that data source (not Model!).

MVC



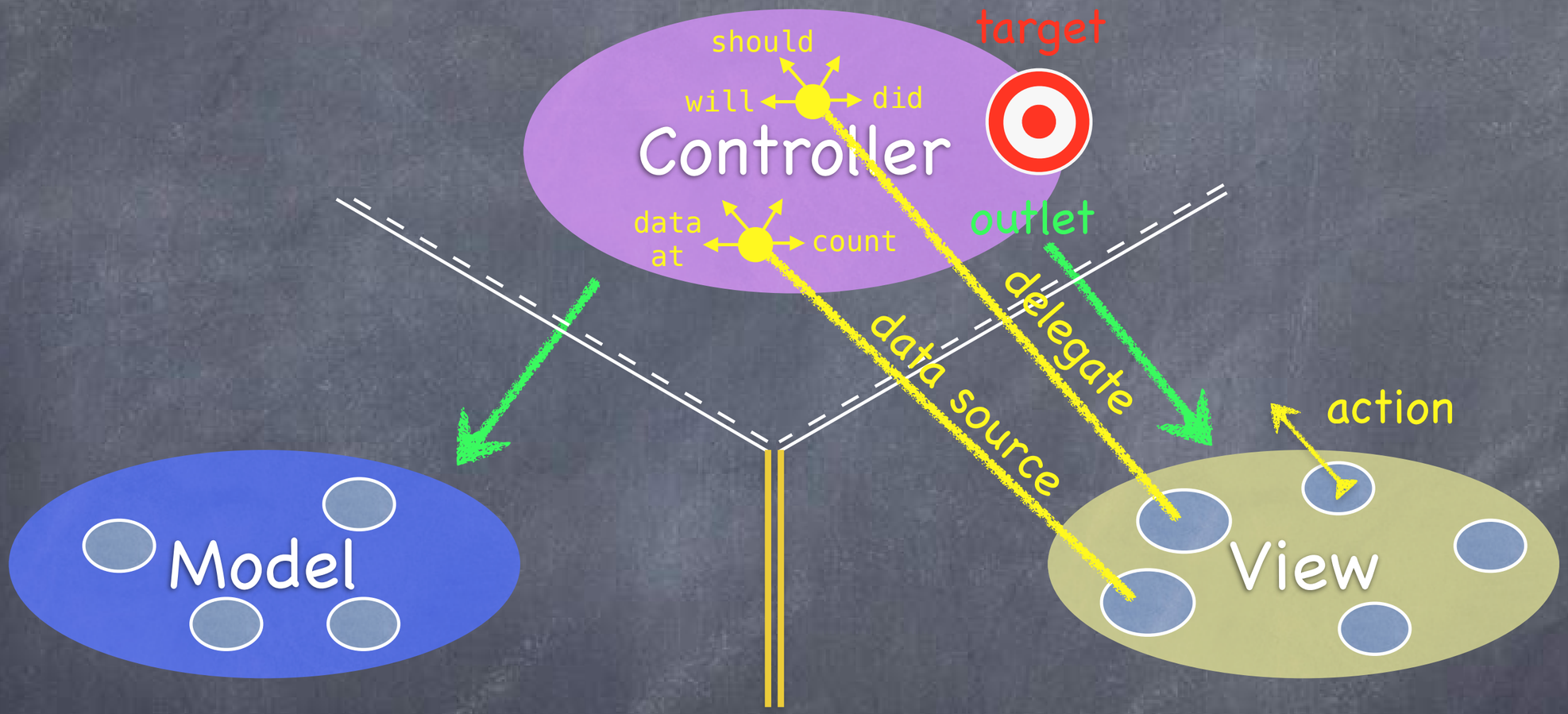
Controllers interpret/format Model information for the View.

MVC



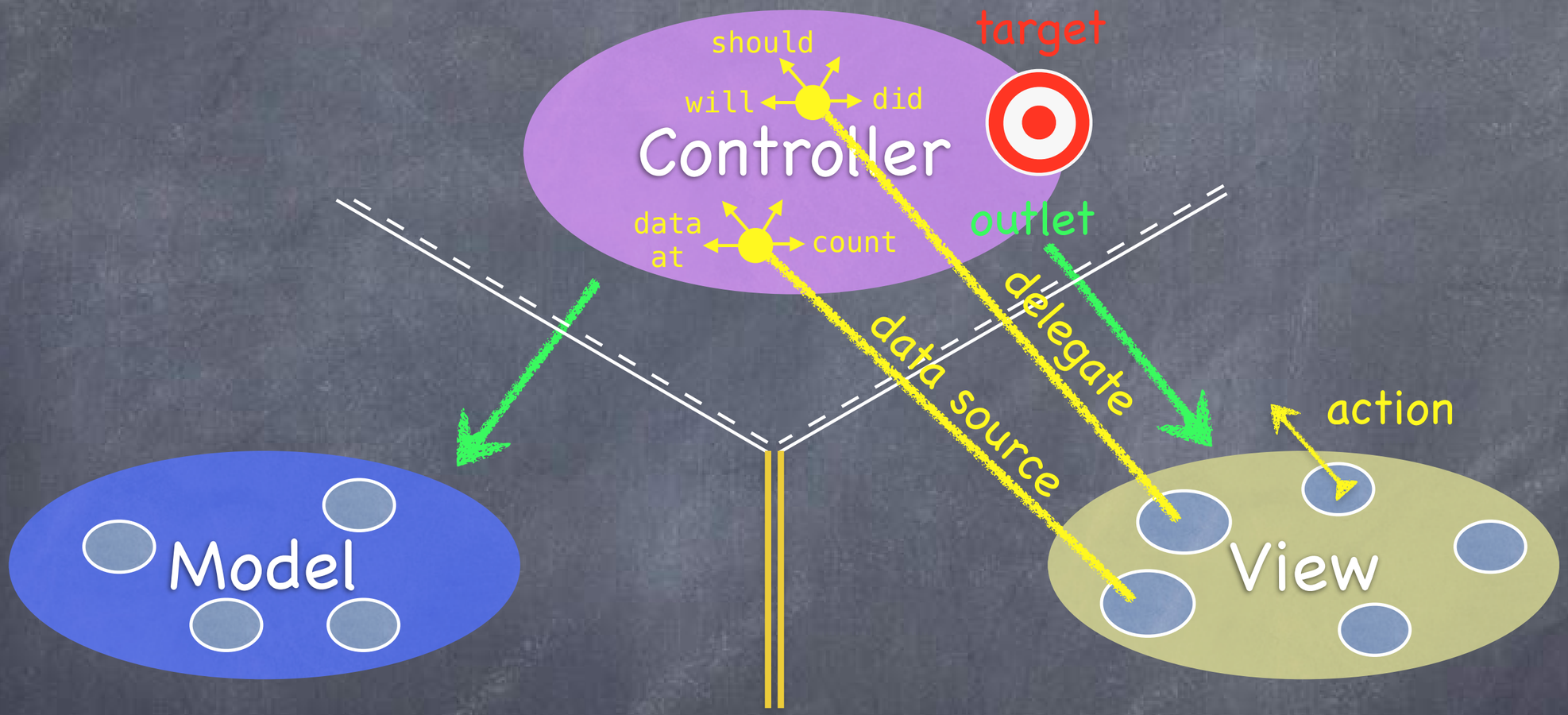
Can the Model talk directly to the Controller?

MVC



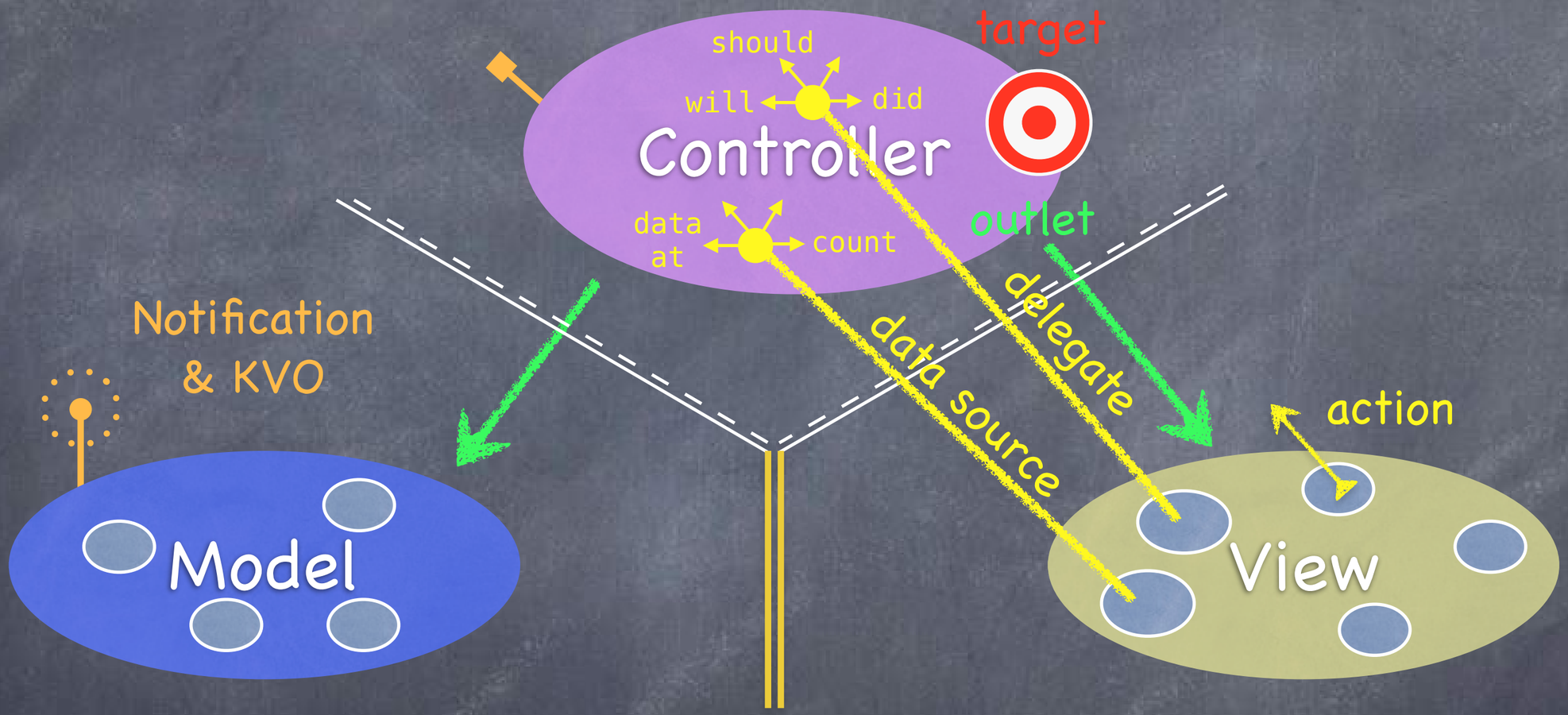
No. The Model is (should be) UI independent.

MVC



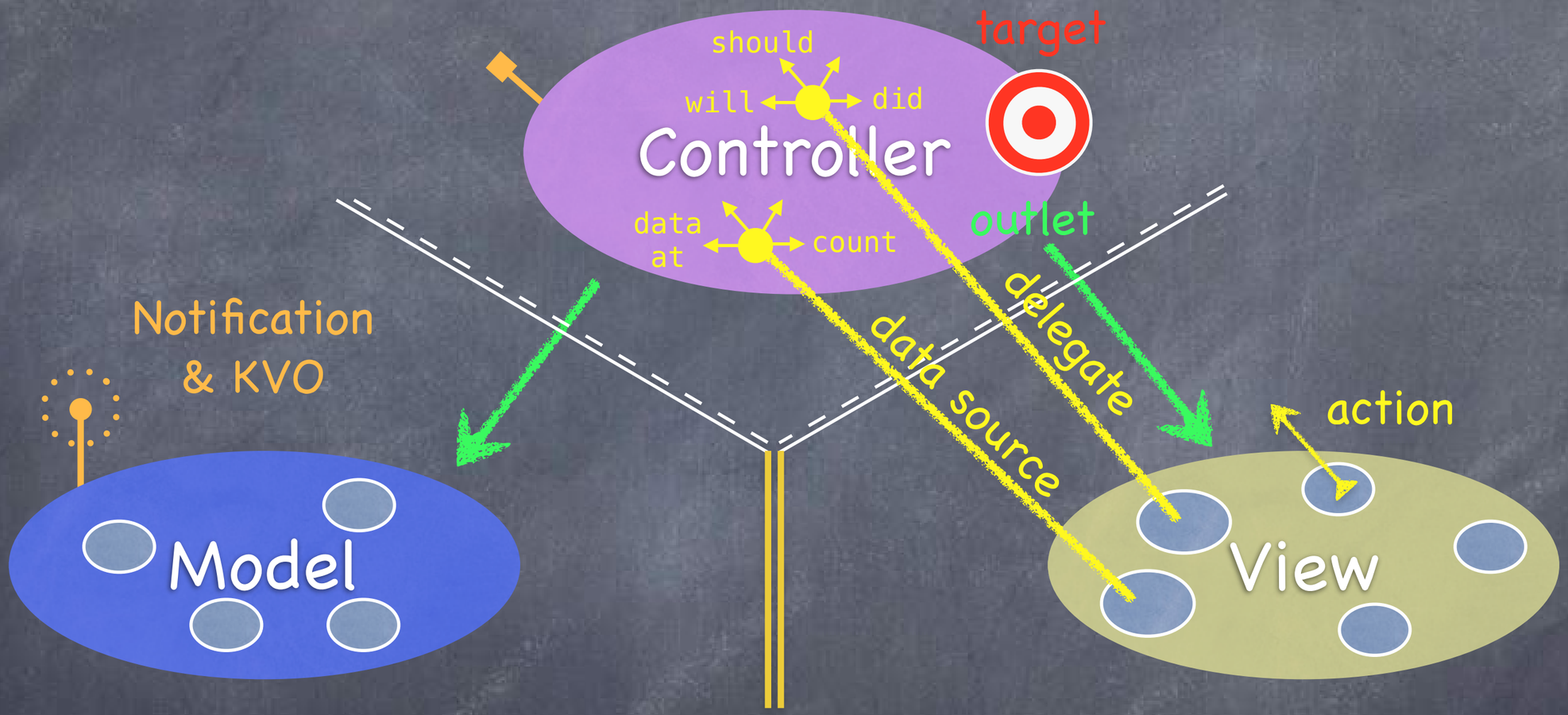
So what if the Model has information to update or something?

MVC



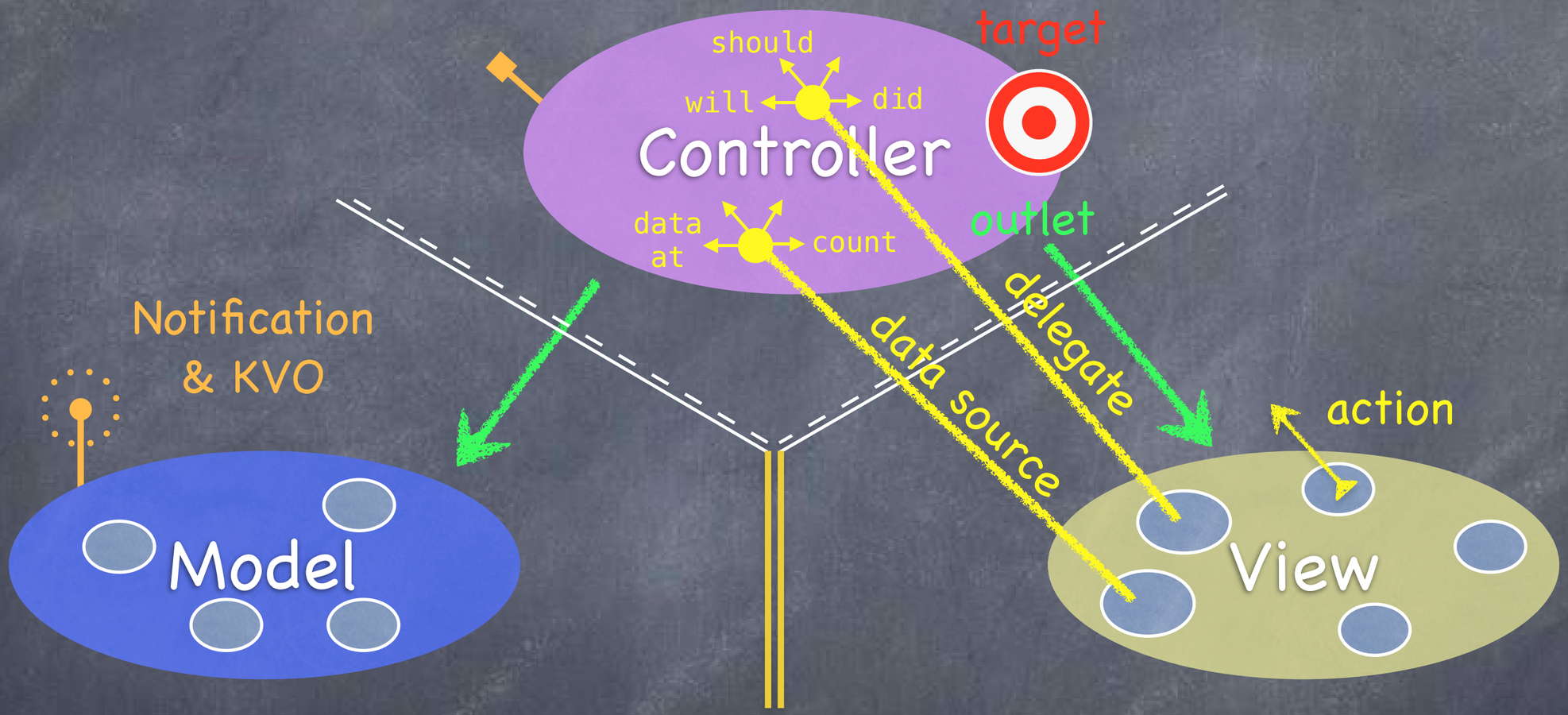
It uses a "radio station"-like broadcast mechanism.

MVC



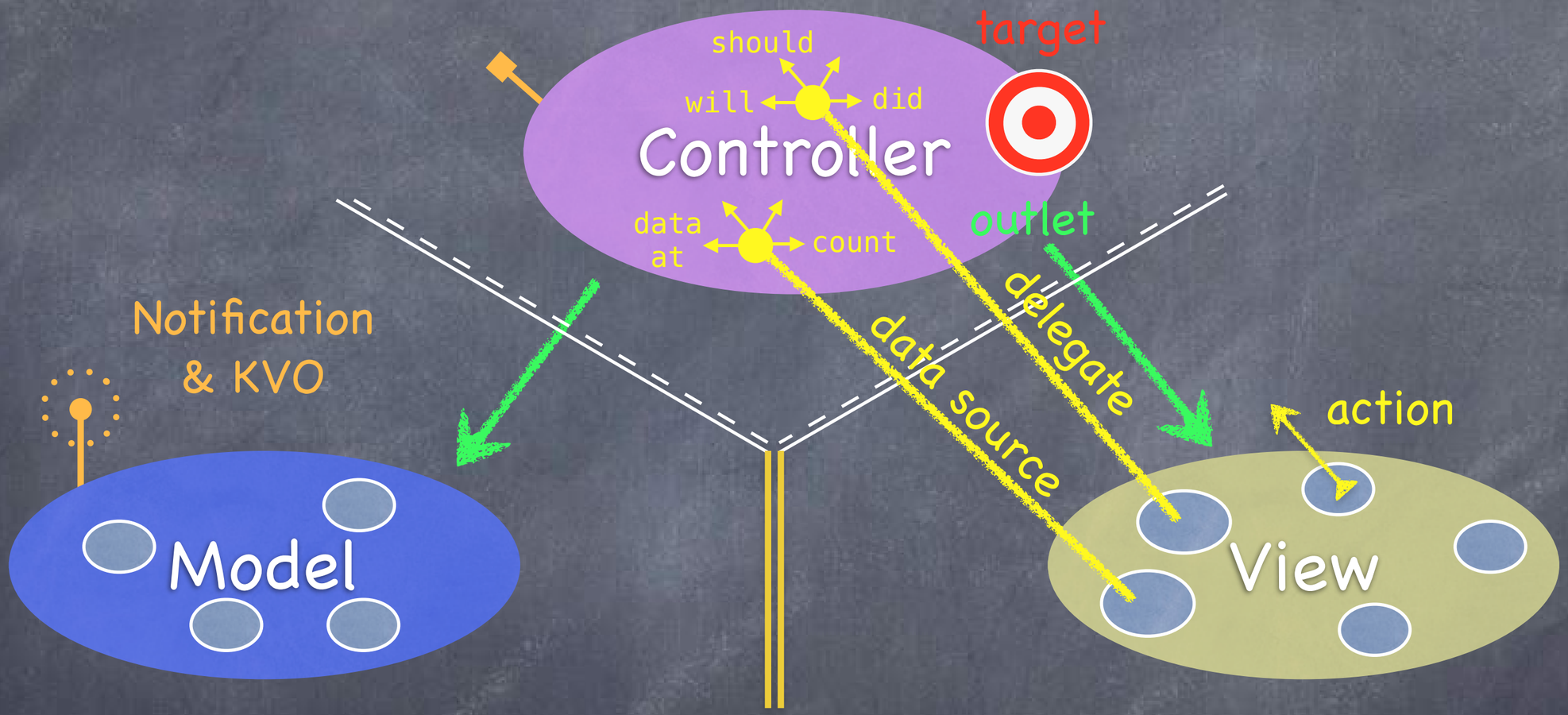
Controllers (or other Model) "tune in" to interesting stuff.

MVC



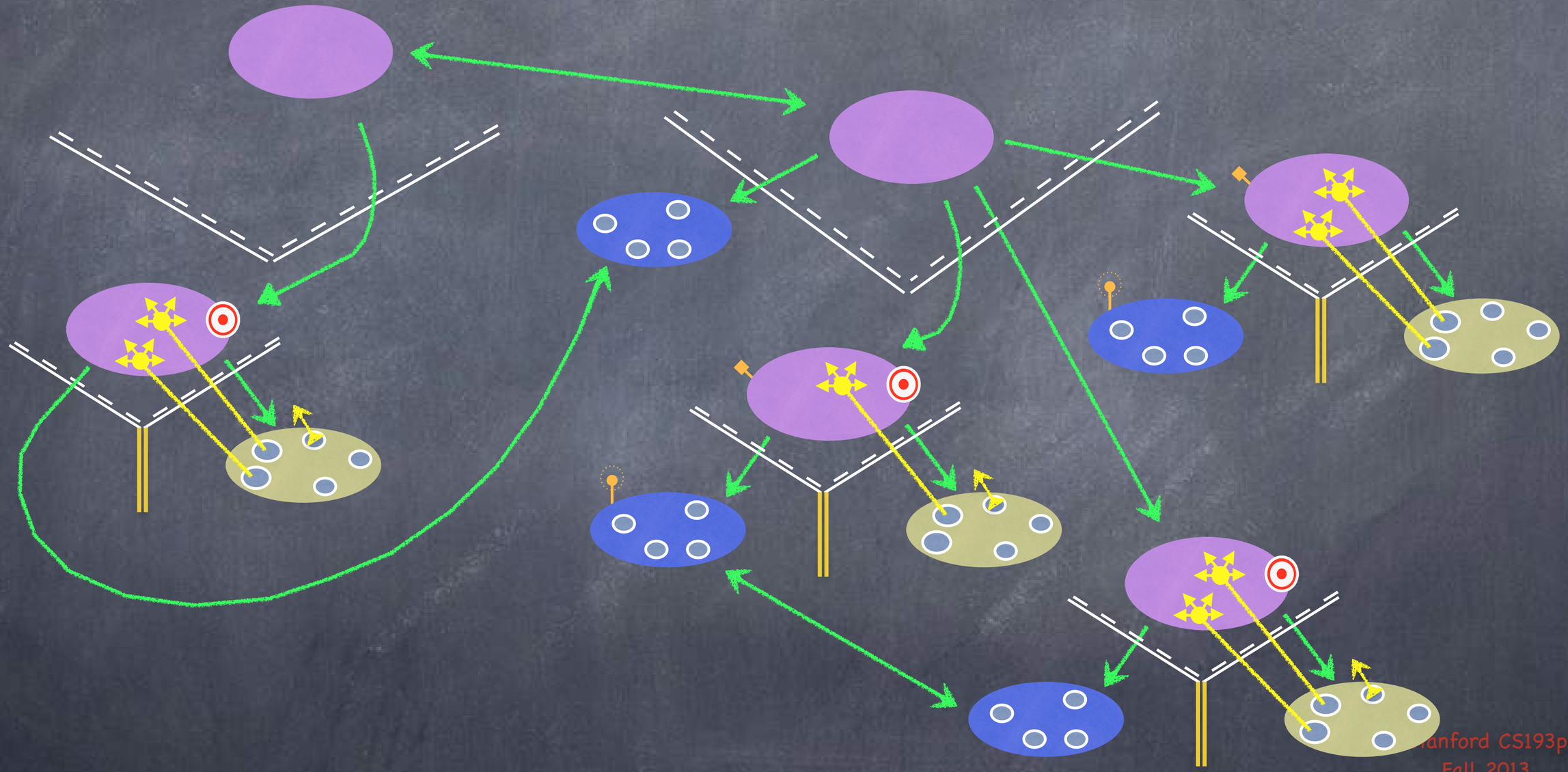
A View might "tune in," but probably not to a Model's "station."

MVC

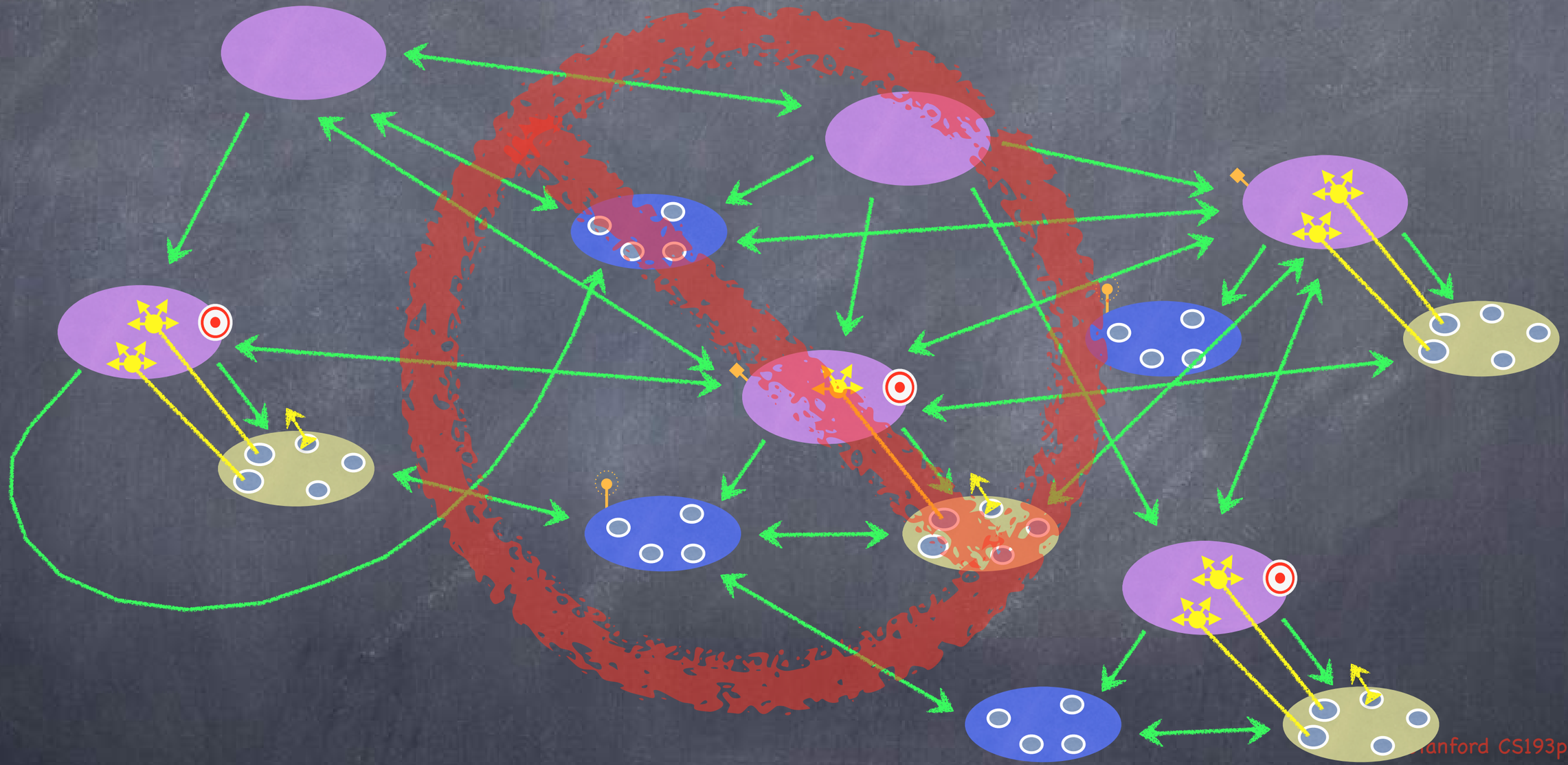


Now combine MVC groups to make complicated programs ...

MVCs working together



MVCs not working together



Objective-C

- New language to learn!
  - Strict superset of C
  - Adds syntax for classes, methods, etc.
  - A few things to "think differently" about (e.g. properties, dynamic binding)
- Most important concept to understand today: Properties
  - Usually we do not access instance variables directly in Objective-C.
  - Instead, we use "properties."
  - A "property" is just the combination of a getter method and a setter method in a class.
  - The getter (usually) has the name of the property (e.g. "myValue")
  - The setter's name is "set" plus capitalized property name (e.g. "setMyValue:")
  - (To make this look nice, we always use a lowercase letter as the first letter of a property name.)
  - We just call the setter to store the value we want and the getter to get it. Simple.
- This is just your first glimpse of this language!
  - We'll go much more into the details next week.
  - Don't get too freaked out by the syntax at this point.

Objective-C

Card.h

Public Declarations

Card.m

Private Implementation

Objective-C

Card.h

Card.m

Its superclass.

```
@interface Card : NSObject
```

The name of this class.

`NSObject` is the root class from which pretty much all iOS classes inherit (including the classes you author yourself).

Don't forget this!

```
@end
```

Objective-C

Card.h

```
@interface Card : NSObject

@end
```

Card.m

```
@implementation Card

@end
```

Note, superclass is not specified here.

Objective-C

Card.h

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

@interface Card : NSObject

@end
```

Superclass's header file.

Card.m

```
@implementation Card

@end
```

Objective-C

Card.h

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



@interface Card : NSObject



@end
```

Card.m

```
@implementation Card

@end
```

If the superclass is in iOS itself, we import the entire "framework" that includes the superclass.
In this case, `Foundation`, which contains basic non-UI objects like `NSObject`.

Objective-C

Card.h

```
@import Foundation;

@interface Card : NSObject

@end
```

In fact, in iOS 7 (only), there is special syntax for importing an entire framework called `@import`.

Card.m

```
@implementation Card

@end
```

Objective-C

Card.h

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



@interface Card : NSObject



@end
```

However, the old framework importing syntax is backwards-compatible in iOS 7.

Card.m

```
@implementation Card



@end
```

Objective-C

Card.h

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



@interface Card : NSObject



@end
```

Card.m

```
#import "Card.h"



@implementation Card



@end
```

Our own header file must be imported into our implementation file.

Objective-C

Card.h

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




@interface Card : NSObject









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









@end
```

Private declarations can go here.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong) NSString *contents;




@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card




@end
```

In iOS, we don't access instance variables directly.
Instead, we use an `@property` which declares two methods: a "setter" and a "getter".
It is with those two methods that the `@property`'s instance variable is accessed
(both publicly and privately).

This particular `@property` is a pointer.
Specifically, a pointer to an object whose class is (or inherits from) `NSString`.

ALL objects live in the heap (i.e. are pointed to) in Objective-C!
Thus you would never have a property of type "`NSString`" (rather, "`NSString *`").

Because this `@property` is in this class's header file, it is public.
Its setter and getter can be called from outside this class's `@implementation` block.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong) NSString *contents;


@end
```

`strong` means:
"keep the object that this property points to
in memory until I set this property to `nil` (zero)
(and it will stay in memory until everyone who has a `strong`
pointer to it sets their property to `nil` too)"

`weak` would mean:
"if no one else has a `strong` pointer to this object,
then you can throw it out of memory
and set this property to `nil`
(this can happen at any time)"

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card


@end
```

Objective-C

Card.h

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



@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;



@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card



@end
```

nonatomic means:
"access to this property is not thread-safe".
We will always specify this for object pointers in this course.
If you do not, then the compiler will generate locking code that will complicate your code elsewhere.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;


@end
```

This is the @property implementation that the compiler generates automatically for you (behind the scenes).
You are welcome to write the setter or getter yourself, but this would only be necessary if you needed to do something in addition to simply setting or getting the value of the property.

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card

@synthesize contents = _contents;

- (NSString *)contents
{
    return _contents;
}

- (void)setContents:(NSString *)contents
{
    _contents = contents;
}


@end
```

This @synthesize is the line of code that actually creates the backing instance variable that is set and gotten.
Notice that by default the backing variable's name is the same as the property's name but with an underbar in front.

Objective-C

Card.h

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



@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









@end
```

Because the compiler takes care of everything you need to implement a property, it's usually only one line of code (the `@property` declaration) to add one to your class.

Objective-C

Card.h

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


@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

@property (nonatomic) BOOL chosen;
@property (nonatomic) BOOL matched;


@end
```

Notice no strong or weak here.
Primitive types are not stored in the heap, so there's no need to specify how the storage for them in the heap is treated.

Let's look at some more properties.
These are not pointers.
They are simple BOOLs.

Properties can be any C type.
That includes int, float, etc., even C structs.

C does not define a "boolean" type.
This BOOL is an Objective-C typedef.
It's values are YES or NO.

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card


@end
```

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

@property (nonatomic) BOOL chosen;
@property (nonatomic) BOOL matched;









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card

@synthesize chosen = _chosen;
@synthesize matched = _matched;

- (BOOL)chosen
{
    return _chosen;
}
- (void)setChosen:(BOOL)chosen
{
    _chosen = chosen;
}

- (BOOL)matched
{
    return _matched;
}
- (void)setMatched:(BOOL)matched
{
    _matched = matched;
}

@end
```

Here's what the compiler is doing behind the scenes for these two properties.

Objective-C

Card.h

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



@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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



@end
```

It is actually possible to change the name of the getter that is generated. The only time you'll ever see that done in this class (or anywhere probably) is boolean getters.

This is done simply to make the code "read" a little bit nicer. You'll see this in action later.

Card.m

```
#import "Card.h"

@interface Card()
@end

@implementation Card

@synthesize chosen = _chosen;
@synthesize matched = _matched;

- (BOOL)isChosen
{
    return _chosen;
}
- (void)setChosen:(BOOL)chosen
{
    _chosen = chosen;
}

- (BOOL)isMatched
{
    return _matched;
}
- (void)setMatched:(BOOL)matched
{
    _matched = matched;
}

@end
```

Note change in getter method.

Note change in getter method.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









@end
```

Remember, unless you need to do something besides setting or getting when a property is being set or gotten, the implementation side of this will all happen automatically for you.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

- (int)match:(Card *)card;




@end
```

Enough properties for now.
Let's take a look at defining methods.

Here's the declaration of a public method called `match:` which takes one argument (a pointer to a `Card`) and returns an `integer`.

What makes this method public?
Because we've declared it in the header file.

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card




@end
```

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

- (int)match:(Card *)card;

@end
```

Here's the declaration of a public method called `match:` which takes one argument (a pointer to a `Card`) and returns an `integer`.

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card




- (int)match:(Card *)card
{
    int score = 0;

    return score;
}

@end
```

`match:` is going to return a "score" which says how good a match the passed `card` is to the `Card` that is receiving this message. 0 means "no match", higher numbers mean a better match.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

- (int)match:(Card *)card;









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









- (int)match:(Card *)card
{
    int score = 0;


    if ([card.contents isEqualToString:self.contents]) {
        score = 1;
    }


    return score;
}




@end
```

There's a lot going on here!
For the first time, we are seeing the "calling" side of properties (and methods).

For this example, we'll return `1` if the passed `card` has the same `contents` as we do or `0` otherwise (you could imagine more complex scoring).

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

- (int)match:(Card *)card;









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









- (int)match:(Card *)card
{
    int score = 0;


    if ([card.contents isEqualToString:self.contents]) {
        score = 1;
    }


    return score;
}




@end
```

Notice that we are calling the "getter" for the contents `@property` (both on our `self` and on the passed `card`). This calling syntax is called "dot notation." It's only for setters and getters.

Objective-C

Card.h

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



@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

- (int)match:(Card *)card;









@end
```

Recall that the contents property is an NSString.

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card









- (int)match:(Card *)card
{
    int score = 0;


    if ([card.contents isEqualToString:self.contents]) {
        score = 1;
    }


    return score;
}


@end
```

isEqualToString: is an NSString method which takes another NSString as an argument and returns a BOOL (YES if the 2 strings are the same).

Also, we see the "square bracket" notation we use to send a message to an object.
In this case, the message isEqualToString: is being sent to the NSString returned by the contents getter.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

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

@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@implementation Card




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


    if ([card.contents isEqualToString:self.contents]) {
        score = 1;
    }


    return score;
}

@end
```

We could make `match:` even more powerful by allowing it to match against multiple cards by passing an array of cards using the `NSArray` class in Foundation.

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

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









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@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
```

We'll implement a very simple match scoring system here which is to score 1 point if ANY of the passed `otherCards`' `contents` match the receiving `Card`'s `contents`.
(You could imagine giving more points if multiple cards match.)

Note the `for-in` looping syntax here.
This is called "fast enumeration."
It works on arrays, dictionaries, etc.

Coming Up

- Next Lecture
  More of our Card Game Model
  Xcode 5 Demonstration (start building our Card Game application)

- Next Week
  Finish off our Card Game application
  Objective-C in more detail
  Foundation (array, dictionary, etc.)
  And much much more!

Stanford CS193p

Developing Applications for iOS
Fall 2013-14



Today

- Introduction to Objective-C (con't)
  Continue showing Card Game Model with `Deck`, `PlayingCard`, `PlayingCardDeck`
- Xcode 5 Demonstration
  Start building the simple Card Game

Objective-C

Card.h

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




@interface Card : NSObject

@property (strong, nonatomic) NSString *contents;

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

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









@end
```

Card.m

```
#import "Card.h"

@interface Card()

@end

@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
```

Objective-C

Deck.h

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

@interface Deck : NSObject




@end
```

Let's look at another class.
This one represents a deck of cards.

Deck.m

```
#import "Deck.h"

@interface Deck()

@end

@implementation Deck




@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;

- (Card *)drawRandomCard;

@end
```

Note that this method has 2 arguments (and returns nothing). It's called "addCard:atTop:".

And this one takes no arguments and returns a Card (i.e. a pointer to an instance of a Card in the heap).

Deck.m

```
#import "Deck.h"

@interface Deck()

@end

@implementation Deck



@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;


- (Card *)drawRandomCard;

@end
```

We must #import the header file for any class we use in this file (e.g. Card).

Deck.m

```
#import "Deck.h"

@interface Deck()

@end

@implementation Deck




@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;


- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()

@end

@implementation Deck









- (void)addCard:(Card *)card atTop:(BOOL)atTop
{




}




- (Card *)drawRandomCard { }

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;


- (Card *)drawRandomCard;

@end
```

Arguments to methods (like the atTop: argument) are never "optional."

Deck.m

```
#import "Deck.h"

@implementation Deck




- (void)addCard:(Card *)card atTop:(BOOL)atTop
{




}




- (Card *)drawRandomCard { }

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Arguments to methods (like the `atTop:` argument) are never "optional."

However, if we want an `addCard:` method without `atTop:`, we can define it separately.

Deck.m

```
#import "Deck.h"

@implementation Deck

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{



}

- (Card *)drawRandomCard { }

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@implementation Deck

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{




}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Arguments to methods (like the `atTop:` argument) are never "optional."

However, if we want an `addCard:` method without `atTop:`, we can define it separately.

And then simply implement it in terms of the the other method.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

A deck of cards obviously needs some storage to keep the cards in.
We need an `@property` for that.
But we don't want it to be public (since it's part of our private, internal implementation).

Deck.m

```
#import "Deck.h"

@interface Deck()

@end

@implementation Deck









- (void)addCard:(Card *)card atTop:(BOOL)atTop
{




}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

A deck of cards obviously needs some storage to keep the cards in.
We need an `@property` for that.
But we don't want it to be public (since it's part of our private, internal implementation).

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{

}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

So we put the `@property` declaration we need here in our `@implementation`.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Now that we have a property to store our cards in, let's take a look at a sample implementation of the addCard:atTop: method.

self.cards is an NSMutableArray ...

... and these are NSMutableArray methods. (insertObject:atIndex: and addObject:).

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

But there's a problem here.
When does the object pointed to by the pointer returned by `self.cards` ever get created?

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Declaring a `@property` makes space in the instance for the pointer itself, but not does not allocate space in the heap for the object the pointer points to.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

The place to put this needed heap allocation is in the getter for the cards @property.

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

- (NSMutableArray *)cards
{
    return _cards;
}


- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

The place to put this needed heap allocation is in the getter for the cards @property.

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

All properties start out with a value of 0 (called nil for pointers to objects).
So all we need to do is allocate and initialize the object if the pointer to it is nil.
This is called "lazy instantiation".
Now you can start to see the usefulness of a @property.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

We'll talk about allocating and initializing objects more later, but here's a simple way to do it.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

- (void)addCard:(Card *)card
{
    [self addCard:card atTop:NO];
}

- (Card *)drawRandomCard { }

@end
```

Now the cards property will always at least be an empty mutable array, so this code will always do what we want.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Let's collapse the code we've written so far to make some space.

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop { ... }
- (void)addCard:(Card *)card { ... }

- (Card *)drawRandomCard
{









}

@end
```

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop { ... }
- (void)addCard:(Card *)card { ... }

- (Card *)drawRandomCard
{
    Card *randomCard = nil;


    return randomCard;
}

@end
```

drawRandomCard simply grabs a card from a random spot in our `self.cards` array.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop { ... }
- (void)addCard:(Card *)card { ... }

- (Card *)drawRandomCard
{
    Card *randomCard = nil;

    unsigned index = arc4random() % [self.cards count];
    randomCard = self.cards[index];
    [self.cards removeObjectAtIndex:index];

    return randomCard;
}

@end
```

arc4random() returns a random integer.

This is the C modulo operator.

These square brackets actually are the equivalent of sending the message objectAtIndexedSubscript: to the array.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop { ... }
- (void)addCard:(Card *)card { ... }

- (Card *)drawRandomCard
{
    Card *randomCard = nil;

    if ([self.cards count]) {
        unsigned index = arc4random() % [self.cards count];
        randomCard = self.cards[index];
        [self.cards removeObjectAtIndex:index];
    }

    return randomCard;
}

@end
```

Calling `objectAtIndexedSubscript:` with an argument of zero on an empty array will **crash** (array index out of bounds)!

So let's protect against that case.

Objective-C

Deck.h

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Deck.m

```
#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop { ... }
- (void)addCard:(Card *)card { ... }

- (Card *)drawRandomCard
{
    Card *randomCard = nil;

    if ([self.cards count]) {
        unsigned index = arc4random() % [self.cards count];
        randomCard = self.cards[index];
        [self.cards removeObjectAtIndex:index];
    }

    return randomCard;
}

@end
```

Objective-C

PlayingCard.h

PlayingCard.m

Let's see what it's like to make a subclass of one of our own classes.
In this example, a subclass of `Card` specific to a playing card (e.g. A♠).

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card



@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard



@end
```

Of course we must `#import` our superclass.

And `#import` our own header file in our implementation file.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard




@end
```

A PlayingCard has some properties that a vanilla Card doesn't have.
Namely, the PlayingCard's suit and rank.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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



@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard


@end
```

We'll represent the suit as an `NSString` that simply contains a single character corresponding to the suit (i.e. one of these characters: ♠♣♥♦).
If this property is `nil`, it'll mean "suit not set".

We'll represent the rank as an integer from 0 (rank not set) to 13 (a King).

`NSUInteger` is a `typedef` for an unsigned integer.

We could just use the C type `unsigned int` here.
It's mostly a style choice.
Many people like to use `NSUInteger` and `NSInteger` in public API and `unsigned int` and `int` inside implementation.
But be careful, `int` is 32 bits, `NSInteger` might be 64 bits.
If you have an `NSInteger` that is really big (i.e. > 32 bits worth) it could get truncated if you assign it to an `int`.
Probably safer to use one or the other everywhere.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{

    return [NSString stringWithFormat:@"%d%@", self.rank, self.suit];
}




@end
```

Users of our `PlayingCard` class might well simply access `suit` and `rank` properties directly.
But we can also support our superclass's `contents` property by overriding the getter to return a suitable (no pun intended) `NSString`.

Even though we are overriding the implementation of the `contents` method, we are not re-declaring the `contents` property in our header file. We'll just inherit that declaration from our superclass.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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



@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{

    return [NSString stringWithFormat:@"%d%@", self.rank, self.suit];
}



@end
```

The method `stringWithFormat:` is an `NSString` method that's sort of like using the C function `printf` to create the string.

Note we are creating an `NSString` here in a different way than `alloc/init`. We'll see more about "class methods" like `stringWithFormat:` a little later.

Users of our `PlayingCard` class might well simply access `suit` and `rank` properties directly. But we can also support our superclass's `contents` property by overriding the getter to return a suitable (no pun intended) `NSString`.

Even though we are overriding the implementation of the `contents` method, we are not re-declaring the `contents` property in our header file. We'll just inherit that declaration from our superclass.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{

    return [NSString stringWithFormat:@"%d%@", self.rank, self.suit];
}



@end
```

Calling the getters of our two properties (rank and suit) on ourself.

But this is a pretty bad representation of the card (e.g., it would say 11♣ instead of J♣ and 1♥ instead of A♥).

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}
```

We'll create an `NSArray` of `NSString`s, each of which corresponds to a given rank.
Again, `0` will be "rank not set" (so we'll use ?).
11, 12 and 13 will be `J Q K` and 1 will be `A`.

Then we'll create our "J♠" string by appending (with the `stringByAppendingString:` method) the suit onto the end of the string we get by looking in the array.

```
@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}




@end
```

Notice the @[ ] notation to create an array.

Here's the array-accessing [] notation again (like we used with `self.cards[index]` earlier).

Also note the @" " notation to create a (constant) NSString.

All of these notations are converted into normal message-sends by the compiler.
For example, @[...] is [[NSArray alloc] initWithObjects:...].
rankStrings[self.rank] is [rankStrings objectAtIndexedSubscript:self.rank].

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}
```

This is nice because a "not yet set" rank shows up as ?.

But what about a "not yet set" `suit`?
Let's override the getter for `suit` to make a `suit` of `nil` return ?.

Yet another nice use for properties versus direct instance variables.

```
- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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



@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}



- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Let's take this a little further and override the setter for `suit` to have it check to be sure no one tries to set a `suit` to something invalid.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}









- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Notice that we can embed the array creation as the target of this message send. We're simply sending `containsObject:` to the array created by the `@[ ]`.

`containsObject:` is an `NSArray` method.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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



@end
```

But there's a problem here now.
A compiler warning will be generated if we do this.
Why?
Because if you implement BOTH the setter and the getter for a property, then you have to create the instance variable for the property yourself.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}









- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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



@end
```

But there's a problem here now.
A compiler warning will be generated if we do this.
Why?
Because if you implement BOTH the setter and the getter for a property, then you have to create the instance variable for the property yourself.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter
```

Luckily, the compiler can help with this using the `@synthesize` directive.

If you implement only the setter OR the getter (or neither), the compiler adds this `@synthesize` for you.

```
- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter
```

Name of the property we're creating an instance variable for.

Name of the instance variable to associate with the property.

We almost always pick an instance variable name that is underbar followed by the name of the property.

```
- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter



- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

You should only ever access the instance variable directly ...

... in the property's setter ...

... in its getter ...

... or in an initializer (more on this later).

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

All of the methods we've seen so far are "instance methods".
They are methods sent to instances of a class.
But it is also possible to create methods that are sent to the class itself.
Usually these are either creation methods (like `alloc` or `stringWithFormat:`) or utility methods.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter




- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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


@end
```

Class methods start with +
Instance methods start with −

Here's an example of a class utility method which returns an NSArray of the NSStrings which are valid suits (e.g. ♠, ♣, ♥, and ♦).

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter

+ (NSArray *)validSuits
{
    return
}

- (void)setSuit:(NSString *)suit
{
    if ([@[@"♥",@"♦",@"♠",@"♣"] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Since a class method is not sent to an instance, we cannot reference our properties in here (since properties represent per-instance storage).

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

Here's an example of a class utility method which returns an NSArray of the NSStrings which are valid suits (e.g. ♠, ♣, ♥, and ♦).

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter

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

- (void)setSuit:(NSString *)suit
{
    if ([                              containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

We actually already have the array of valid suits, so let's just move that up into our new class method.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter

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

- (void)setSuit:(NSString *)suit
{
    if ([[PlayingCard validSuits] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Now let's invoke our new class method here.

See how the name of the class appears in the place you'd normally see a pointer to an instance of an object?

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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




@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter

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

- (void)setSuit:(NSString *)suit
{
    if ([[PlayingCard validSuits] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Now let's invoke our new class method here.

See how the name of the class appears in the place you'd normally see a pointer to an instance of an object?

It'd probably be instructive to go back and look at the invocation of the `NSString` class method `stringWithFormat:` a few slides ago.

Also, make sure you understand that stringByAppendingString: above is not a class method, it is an instance method.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

The `validSuits` class method might be useful to users of our `PlayingCard` class, so let's make it public.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter

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

- (void)setSuit:(NSString *)suit
{
    if ([[PlayingCard validSuits] containsObject:suit]) {
        _suit = suit;
    }
}

- (NSString *)suit
{
    return _suit ? _suit : @"?";
}

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

- (NSString *)contents
{
    NSArray *rankStrings = @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
    return [rankStrings[self.rank] stringByAppendingString:self.suit];
}

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

Let's move our other array (the strings of the ranks) into a class method too.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

+ (NSArray *)rankStrings
{
    return @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
}


@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

We'll leave this one private because the public API for the rank is purely numeric.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

+ (NSArray *)rankStrings
{
    return @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
}




@end
```

And now let's call that class method.

Note that we are not required to declare this earlier in the file than we use it.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

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

@end
```

So we'll add it to the header file.

But here's another class method that might be good to make public.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

+ (NSArray *)rankStrings
{
    return @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
}

+ (NSUInteger)maxRank { return [[self rankStrings] count]-1; }

@end
```

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

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

@end
```

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

+ (NSArray *)rankStrings
{
    return @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
}

+ (NSUInteger)maxRank { return [[self rankStrings] count]-1; }

- (void)setRank:(NSUInteger)rank
{
    if (rank <= [PlayingCard maxRank]) {
        _rank = rank;
    }
}

@end
```

And, finally, let's use maxRank inside the setter for the rank @property to make sure the rank is never set to an improper value.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

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

@end
```

That's it for our PlayingCard. It's a good example of array notation, @synthesize, class methods, and using getters and setters for validation.

PlayingCard.m

```
#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit; // because we provide setter AND getter
+ (NSArray *)validSuits { ... }
- (void)setSuit:(NSString *)suit { ... }
- (NSString *)suit { ... }

+ (NSArray *)rankStrings
{
    return @[@"?",@"A",@"2",@"3",...,@"10",@"J",@"Q",@"K"];
}

+ (NSUInteger)maxRank { return [[self rankStrings] count]-1; }

- (void)setRank:(NSUInteger)rank
{
    if (rank <= [PlayingCard maxRank]) {
        _rank = rank;
    }
}

@end
```

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

Let's look at one last class.
This one is a subclass of Deck and represents a full 52-card deck of PlayingCards.

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck









@end
```

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

It appears to have no public API, but it is going to override a method that `Deck` inherits from `NSObject` called `init`.

`init` will contain everything necessary to initialize a `PlayingCardDeck`.

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck









@end
```

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{

}

@end
```

Initialization in Objective-C happens immediately after allocation.
We always nest a call to `init` around a call to `alloc`.
e.g. `Deck *myDeck = [[PlayingCardDeck alloc] init]`
or `NSMutableArray *cards = [[NSMutableArray alloc] init]`

Classes can have more complicated initializers than just plain "`init`"
(e.g. `initWithCapacity:` or some such).
We'll talk more about that next week as well.

Only call an `init` method immediately after calling `alloc` to make space in the heap for that new object.
And never call `alloc` without immediately calling some `init` method on the newly allocated object.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{









}

@end
```

Notice this weird "return type" of `instancetype`.
It basically tells the compiler that this method returns an object which will be the same type as the object that this message was sent to.
We will pretty much only use it for `init` methods.
Don't worry about it too much for now.
But always use this return type for your `init` methods.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {

    }

    return self;
}

@end
```

This sequence of code might also seem weird.
Especially an assignment to `self`!
This is the ONLY time you would ever assign something to `self`.
The idea here is to return `nil` if you cannot initialize this object.
But we have to check to see if our `super`class can `init`ialize itself.
The assignment to `self` is a bit of protection against our trying to continue to initialize ourselves if our `super`class couldn't initialize.
Just always do this and don't worry about it too much.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {









    }

    return self;
}

@end
```

Sending a message to `super` is how we send a message to ourselves, but use our superclass's implementation instead of our own.
Standard object-oriented stuff.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {




            }
        }
    }

    return self;
}

@end
```

The implementation of `init` is quite simple. We'll just iterate through all the `suits` and then through all the `ranks` in that `suit` ...

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;

            }
        }
    }

    return self;
}

@end
```

Then we will allocate and initialize a `PlayingCard` and then set its `suit` and `rank`.

We never implemented an `init` method in `PlayingCard`, so it just inherits the one from `NSObject`. Even so, we must always call an `init` method after `alloc`.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"
#import "PlayingCard.h"

@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;

            }
        }
    }

    return self;
}

@end
```

We will need to `#import` `PlayingCard`'s header file since we are referencing it now in our implementation.

Then we will allocate and initialize a `PlayingCard` and then set its `suit` and `rank`.

We never implemented an `init` method in `PlayingCard`, so it just inherits the one from `NSObject`. Even so, we must always call an `init` method after `alloc`.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"
#import "PlayingCard.h"

@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;
                [self addCard:card];
            }
        }
    }

    return self;
}

@end
```

Finally we just add each PlayingCard we create to ourself (we are a Deck, remember).

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"
#import "PlayingCard.h"

@implementation PlayingCardDeck

- (instancetype)init
{
    self = [super init];

    if (self) {
        for (NSString *suit in [PlayingCard validSuits]) {
            for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
                PlayingCard *card = [[PlayingCard alloc] init];
                card.rank = rank;
                card.suit = suit;
                [self addCard:card];
            }
        }
    }

    return self;
}

@end
```

And that's it!
We inherit everything else we need to be a Deck of cards
(like the ability to drawRandomCard) from our superclass.

Demo

- Let's start building a Card Game out of these classes
  Today we'll just have a single card that we can flip over to reveal the Ace of clubs.

The following slides are a walkthrough of the demonstration done in class.
You will need this walkthrough to do your first homework assignment.

Yellow Bubbles
mean "do something."

Green Bubbles
are just for
"information."

Red Bubbles
mean "important!"

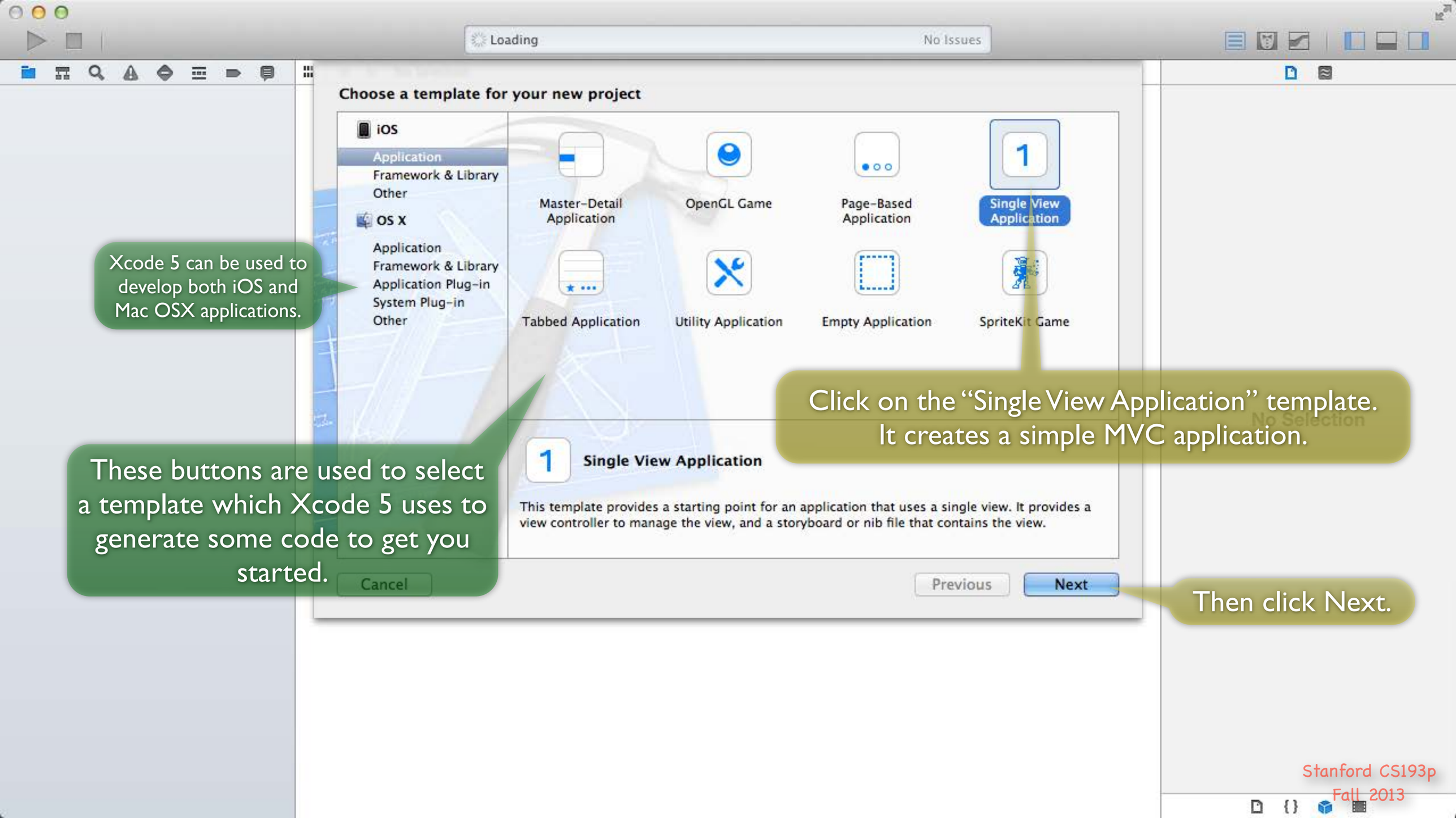
Green Bubbles
with small text is for
"minor notes."



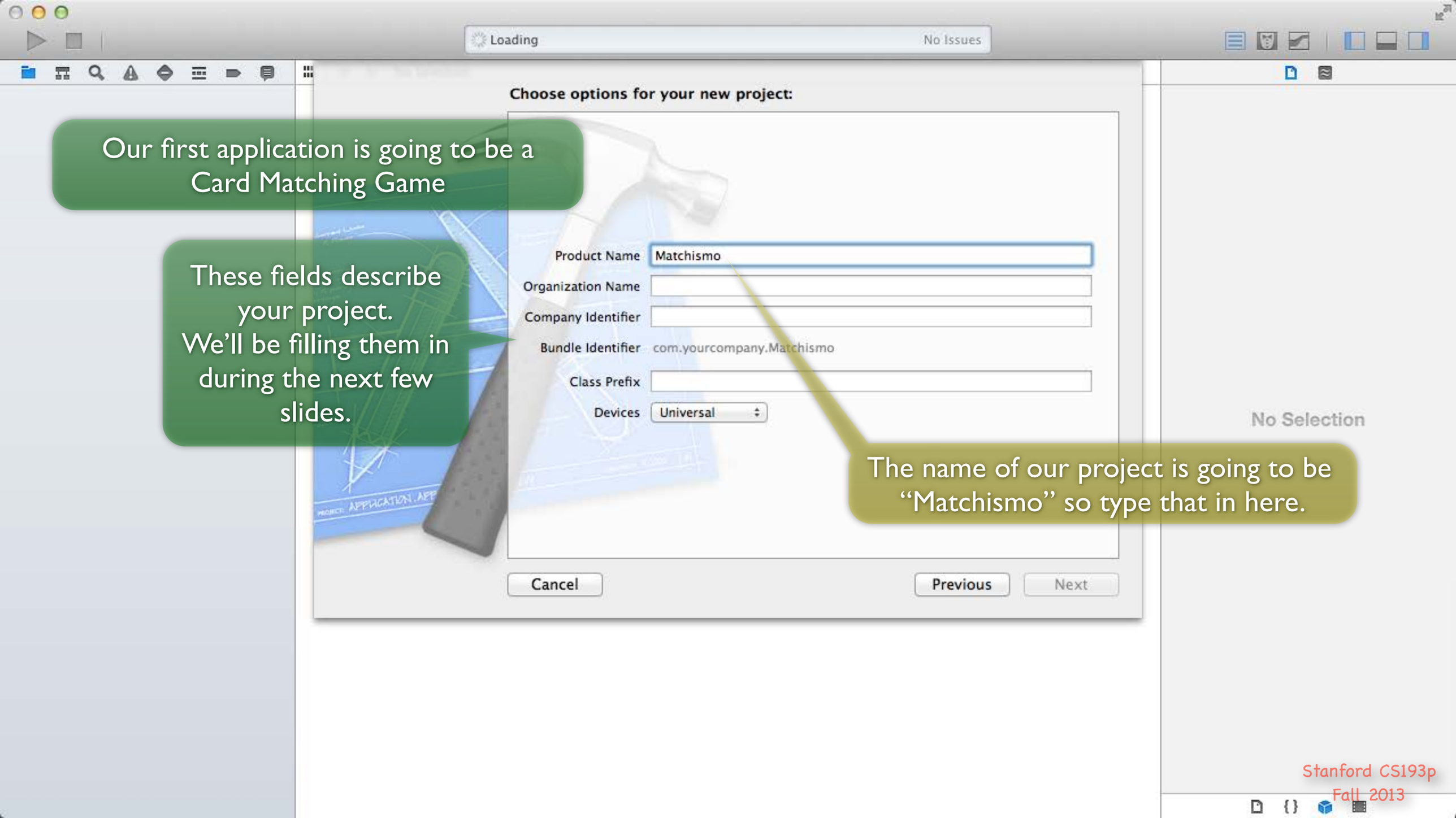
Xcode 5
Splash Screen

Launch Xcode 5 and click here to create a new project.

As you create projects, they will appear here.



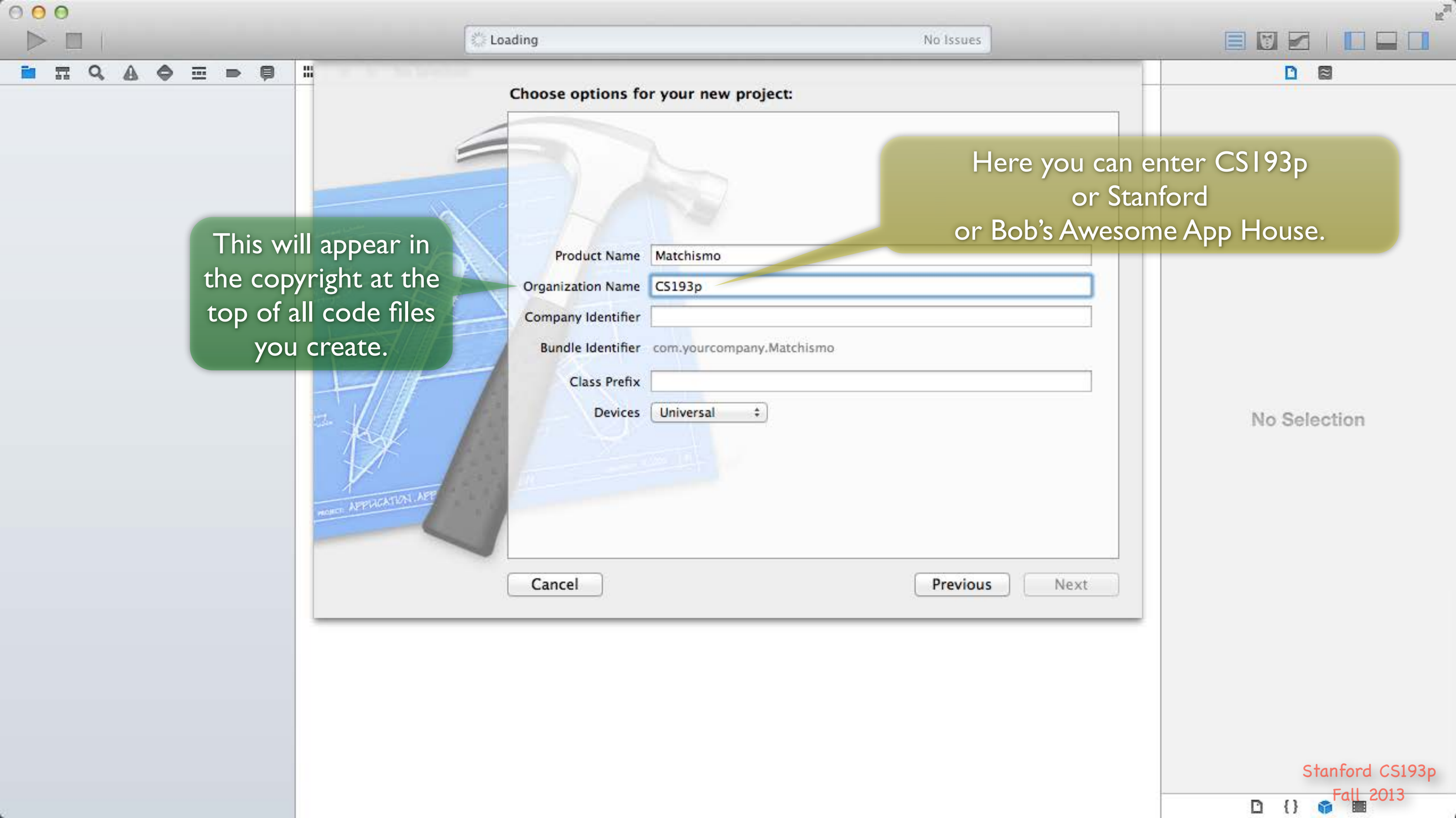
Stanford CS193p
Fall 2013



Our first application is going to be a Card Matching Game

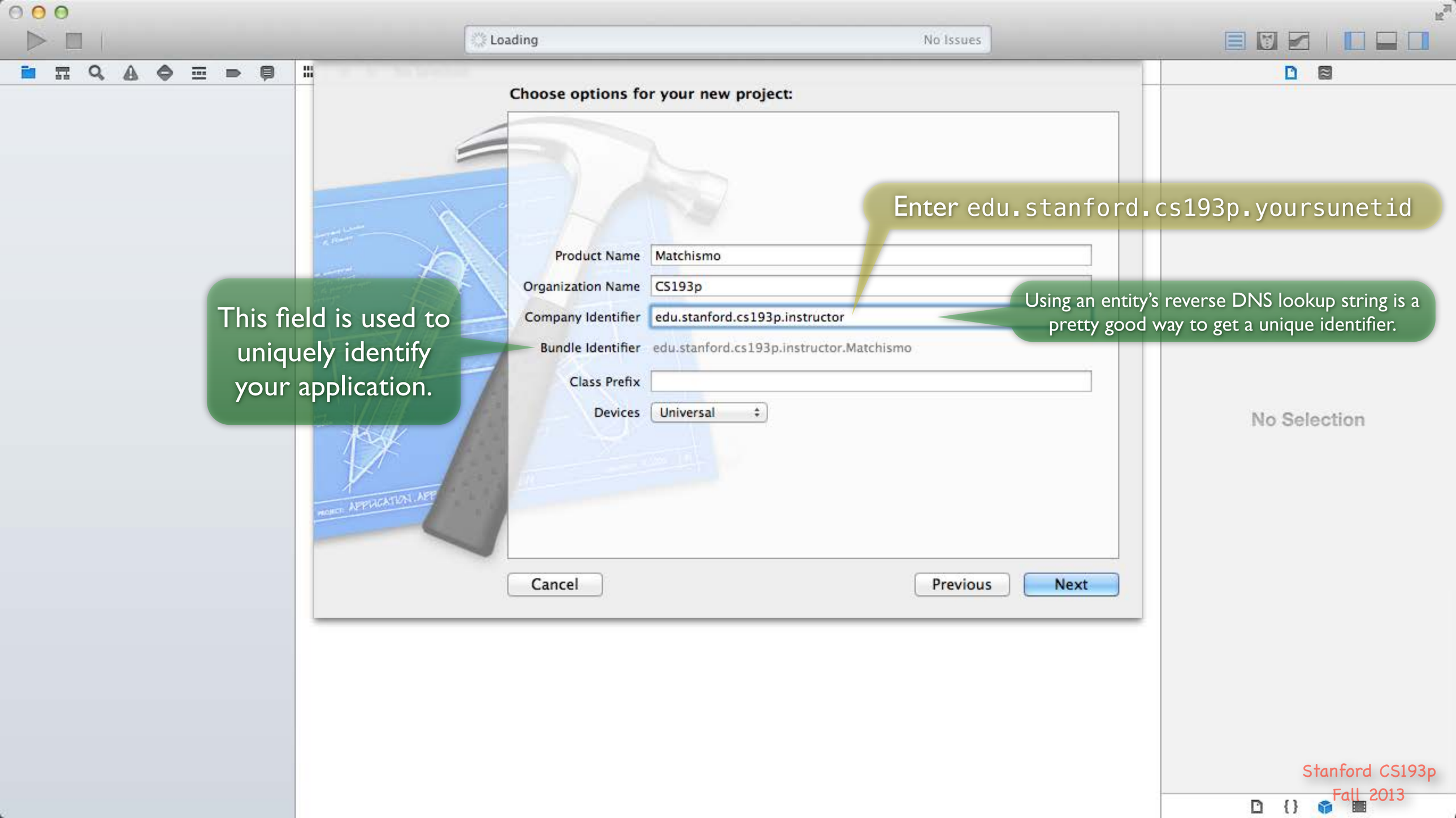
These fields describe your project. We'll be filling them in during the next few slides.

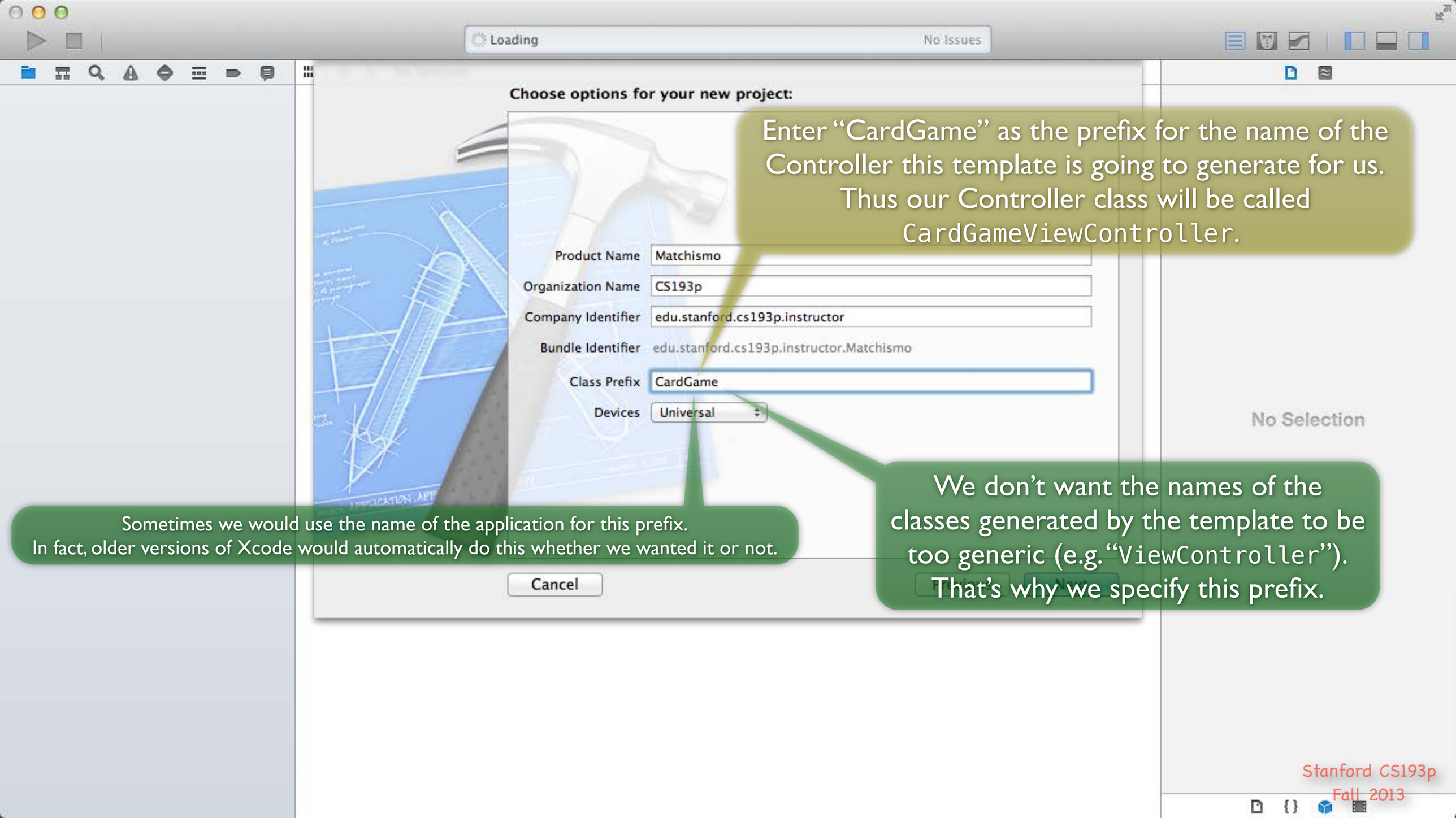
The name of our project is going to be "Matchismo" so type that in here.

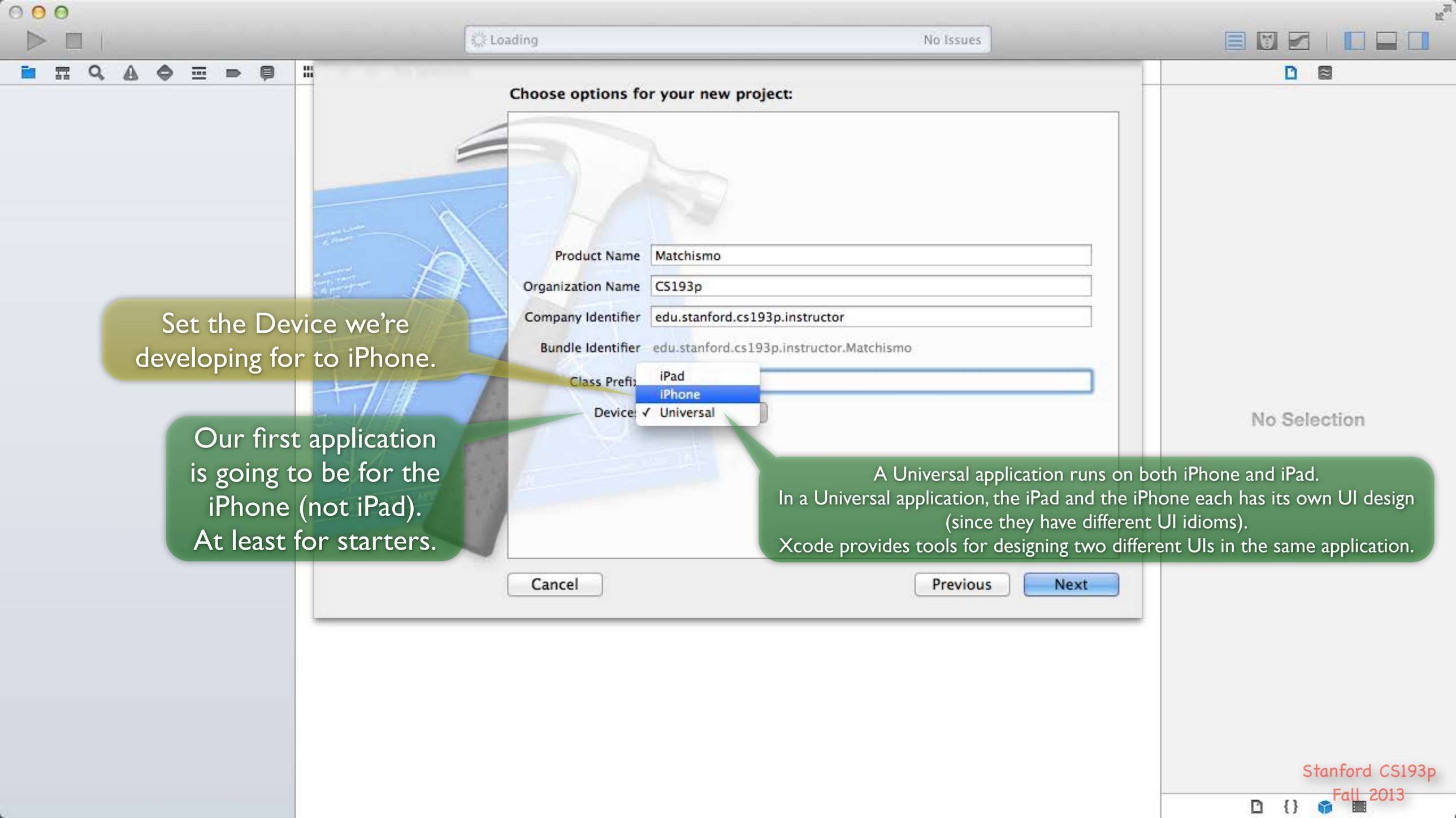


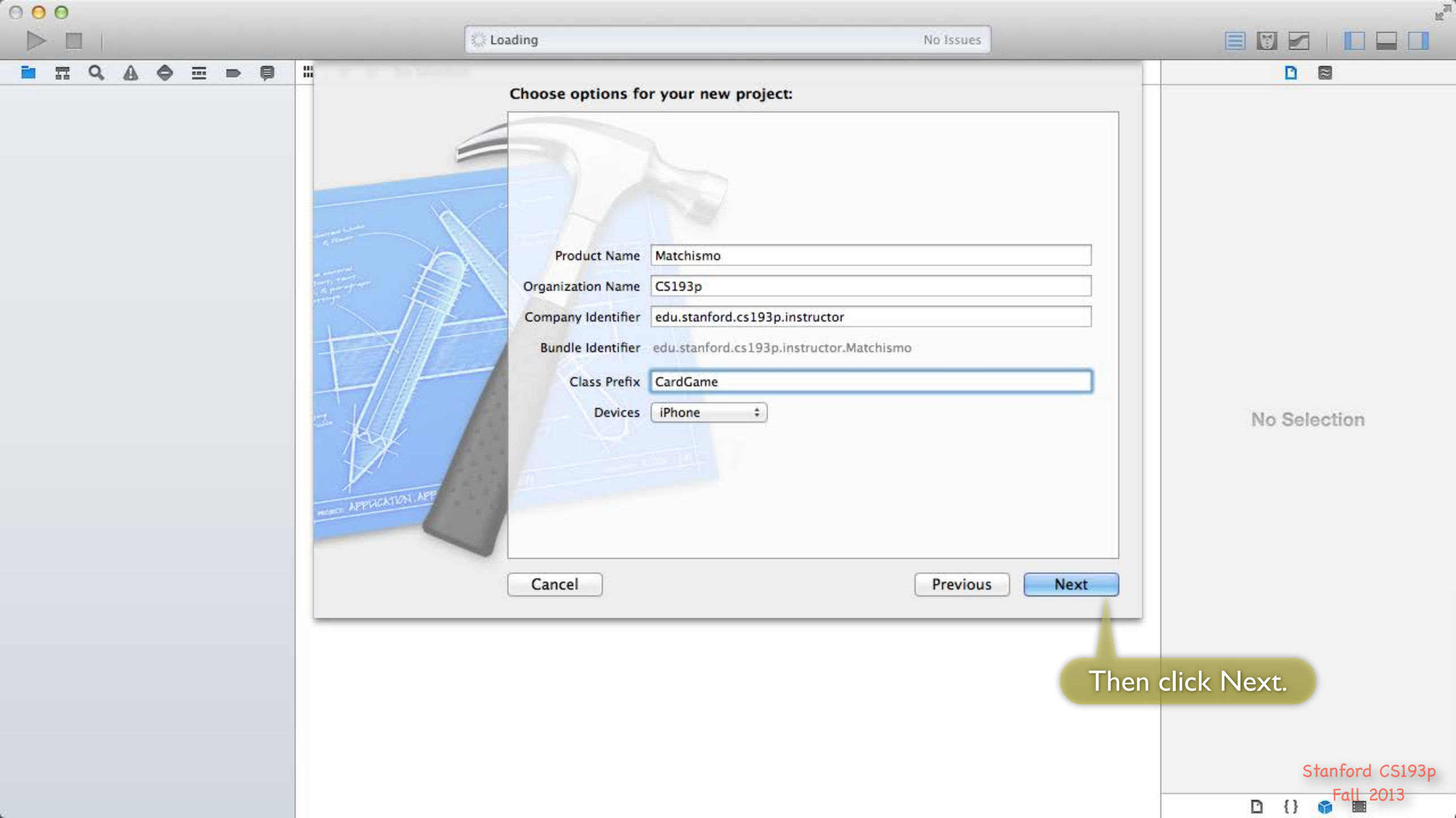
This will appear in the copyright at the top of all code files you create.

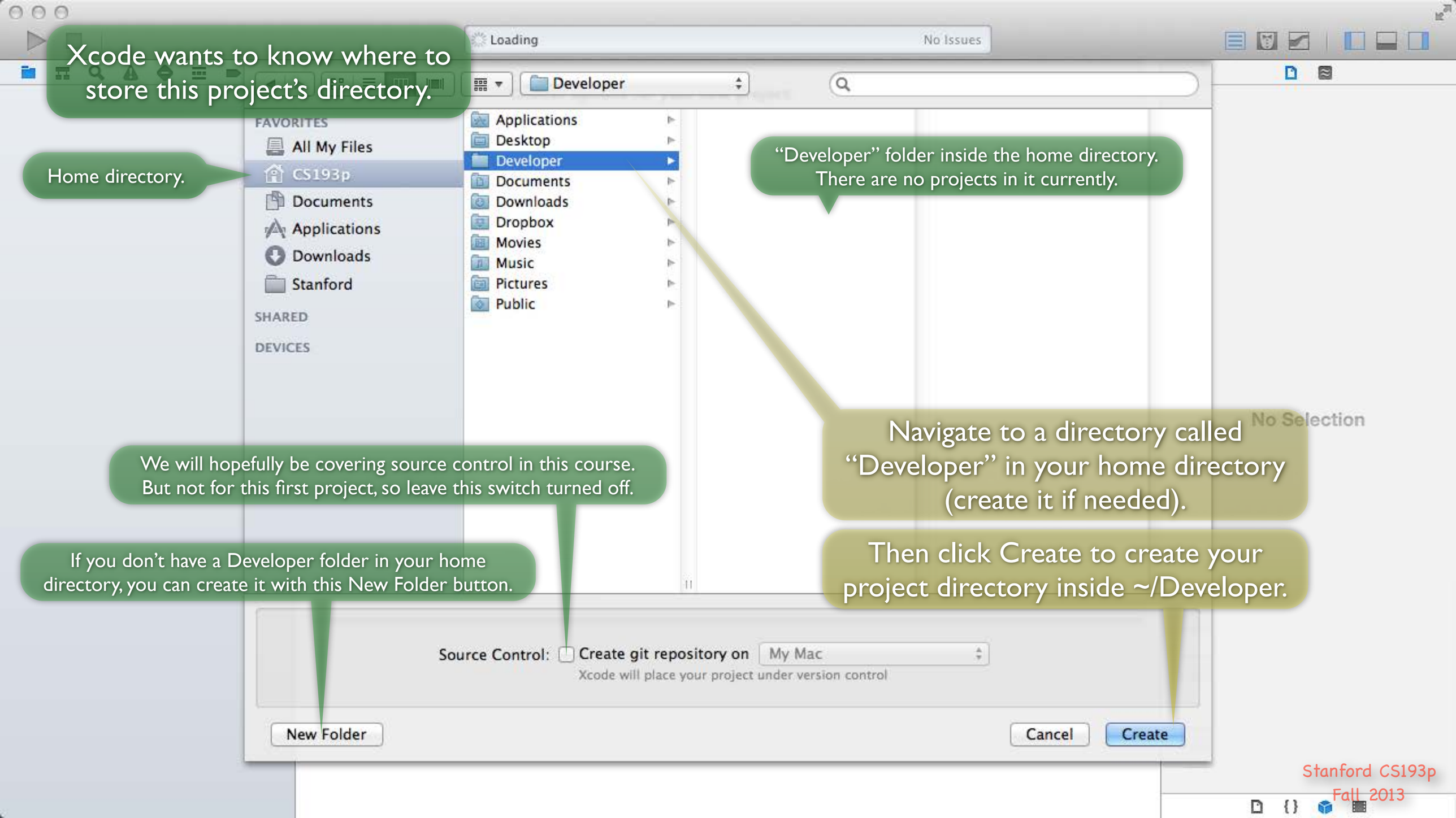
Here you can enter CS193p or Stanford or Bob's Awesome App House.

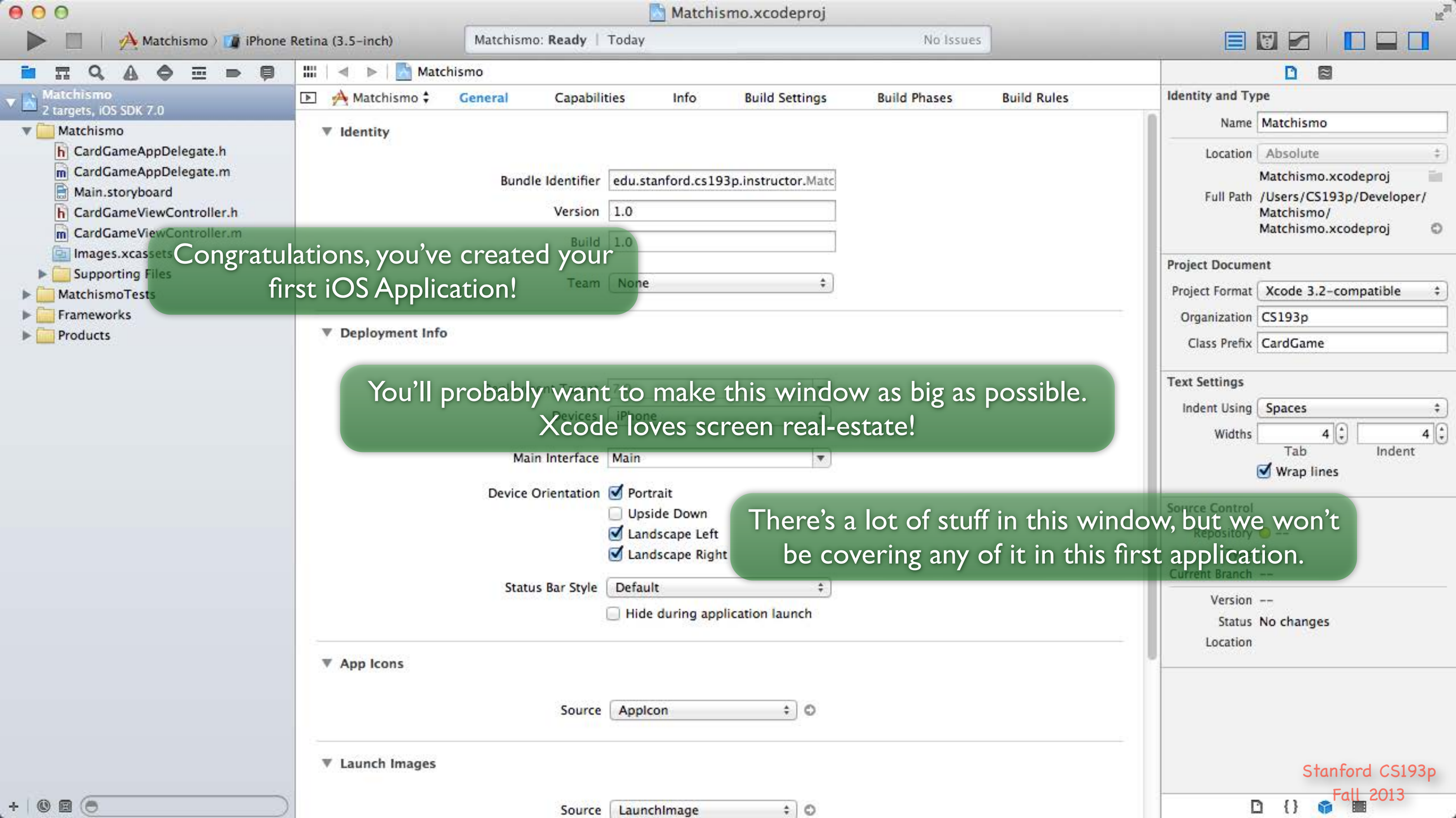

Loading
No Issues
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
Devices
Universal
Cancel
Previous
Next
Enter edu.stanford.cs193p.yoursunetid
This field is used to uniquely identify your application.
Using an entity's reverse DNS lookup string is a pretty good way to get a unique identifier.
No Selection

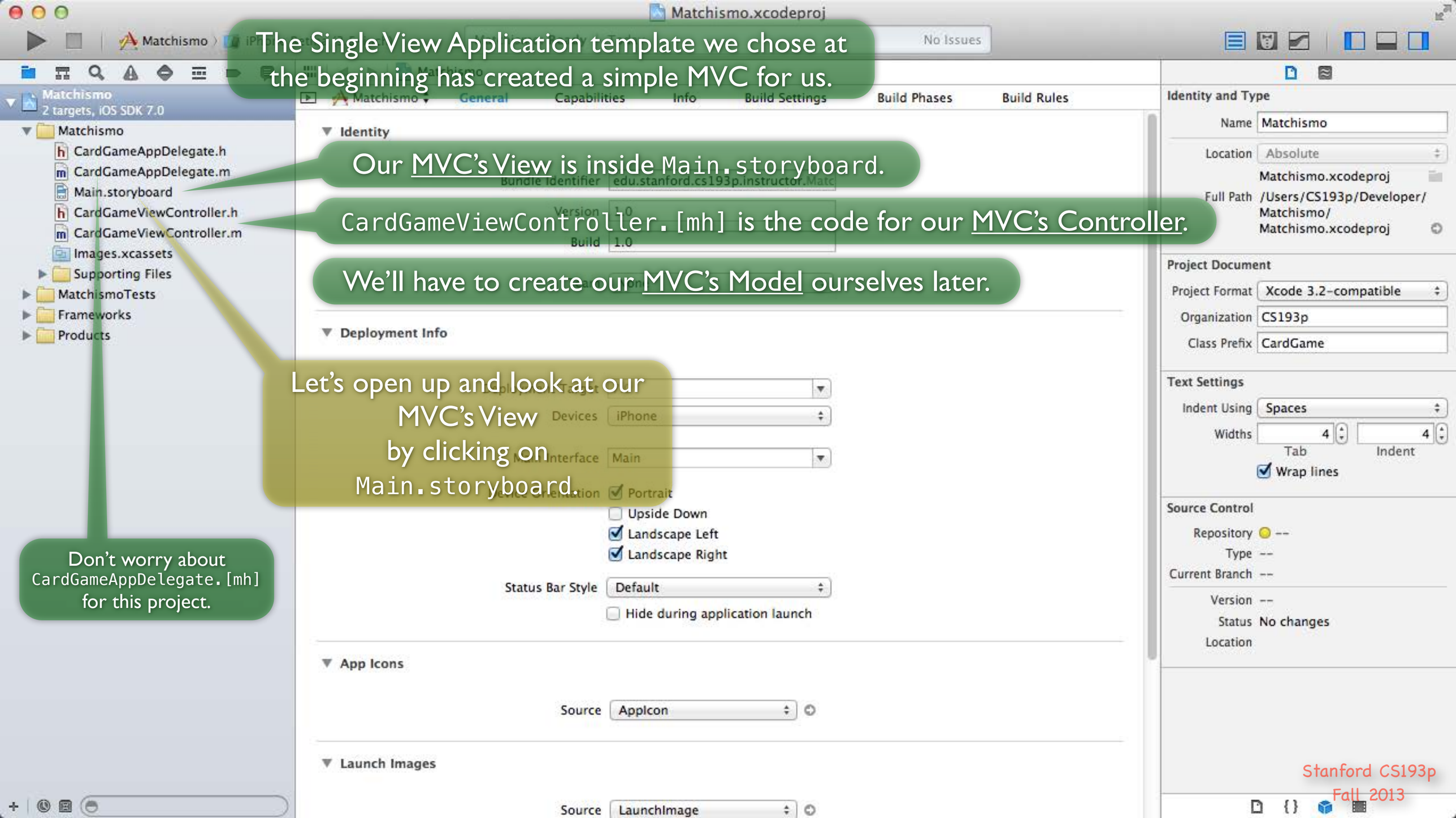

Loading
No Issues
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
CardGame
Devices
Universal
Cancel
No Selection
Enter "CardGame" as the prefix for the name of the Controller this template is going to generate for us. Thus our Controller class will be called CardGameViewController.
Sometimes we would use the name of the application for this prefix.
In fact, older versions of Xcode would automatically do this whether we wanted it or not.
We don't want the names of the classes generated by the template to be too generic (e.g. "ViewController"). That's why we specify this prefix.

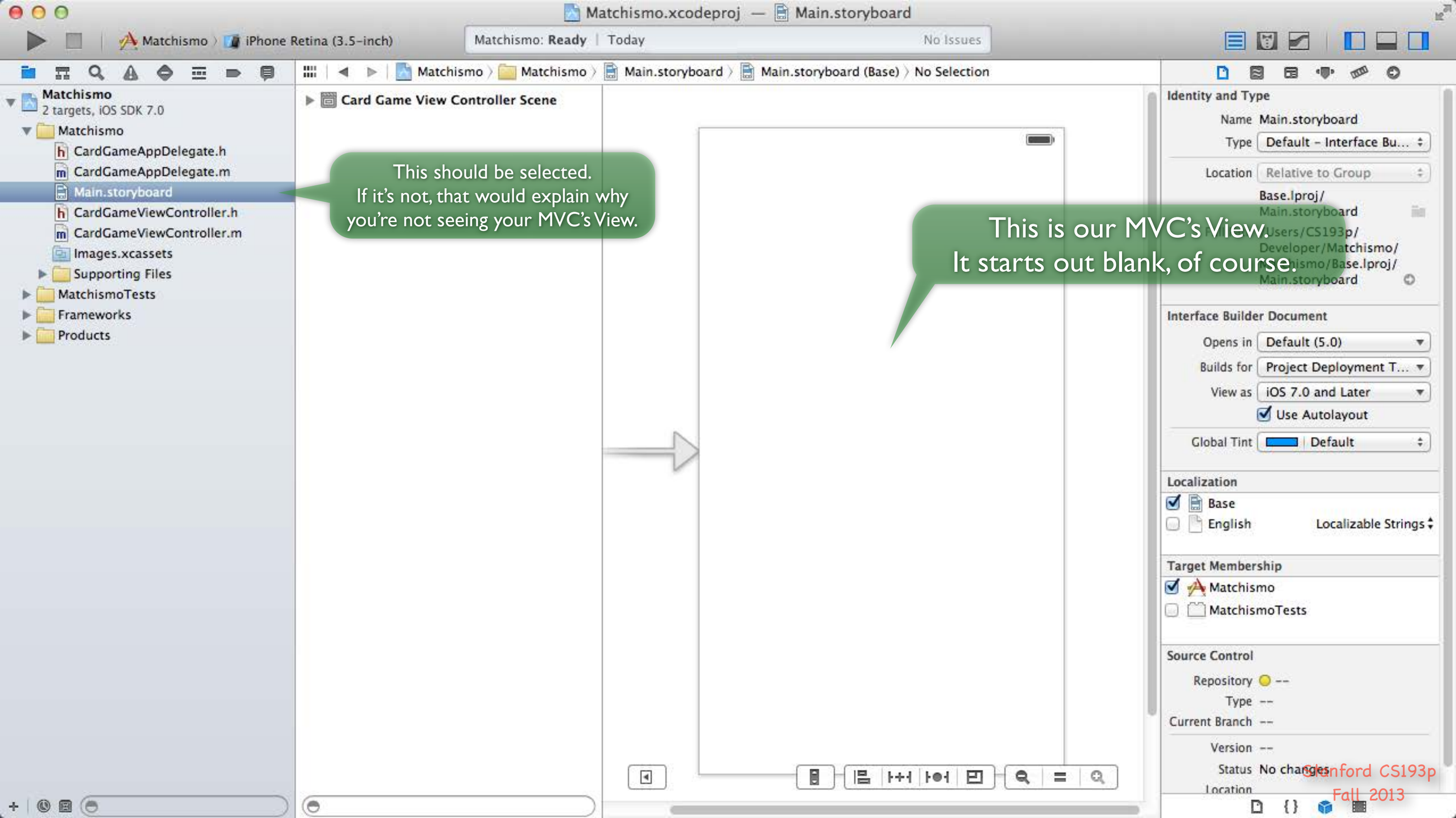

Loading
No Issues
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
Device
iPad
iPhone
✓ Universal
Cancel
Previous
Next
No Selection
Set the Device we're developing for to iPhone.
Our first application is going to be for the iPhone (not iPad). At least for starters.
A Universal application runs on both iPhone and iPad.
In a Universal application, the iPad and the iPhone each has its own UI design (since they have different UI idioms).
Xcode provides tools for designing two different UIs in the same application.

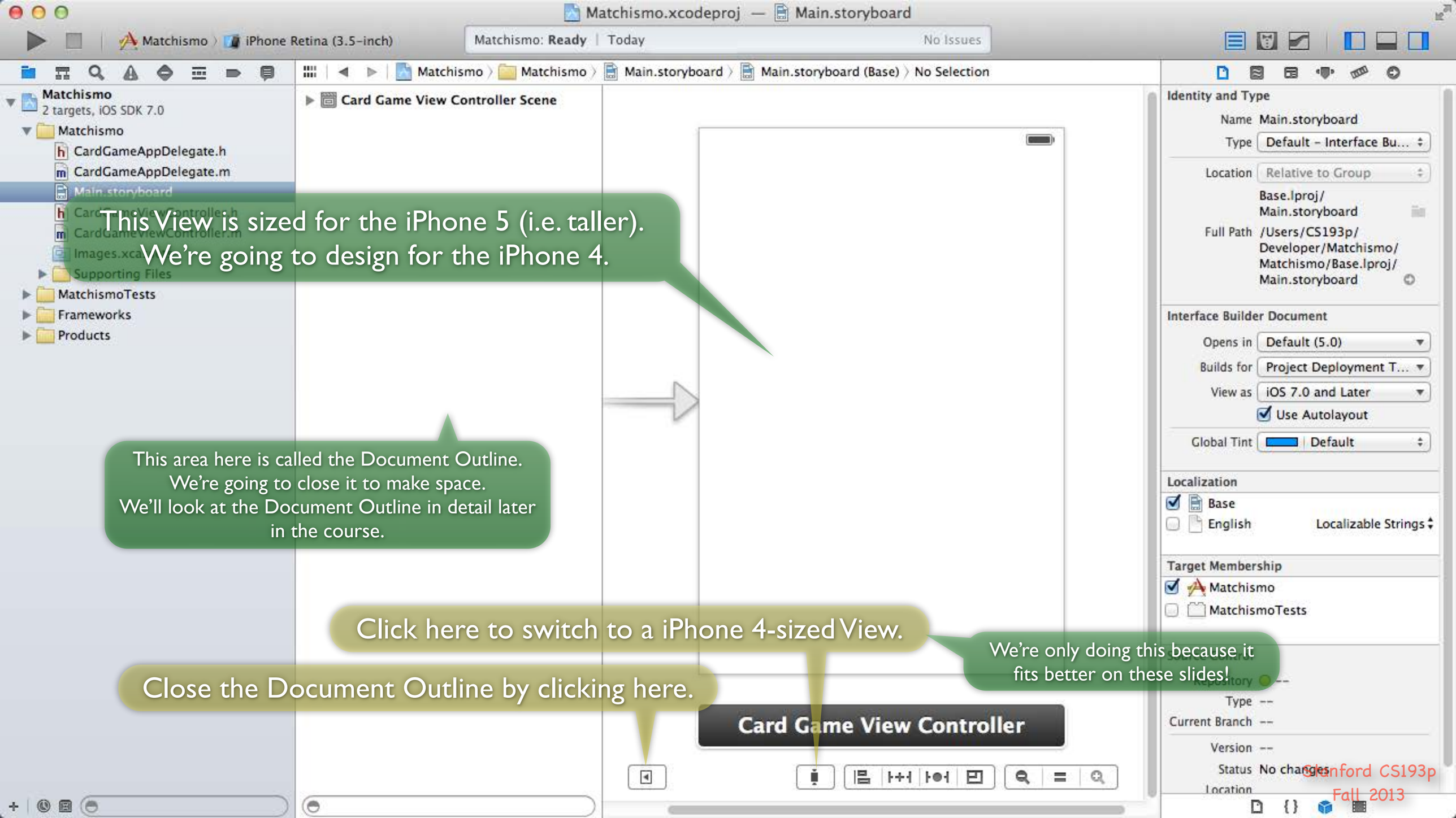

Loading
No Issues
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
CardGame
Devices
iPhone
Cancel
Previous
Next
No Selection
Then click Next.
Stanford CS193p
Fall 2013

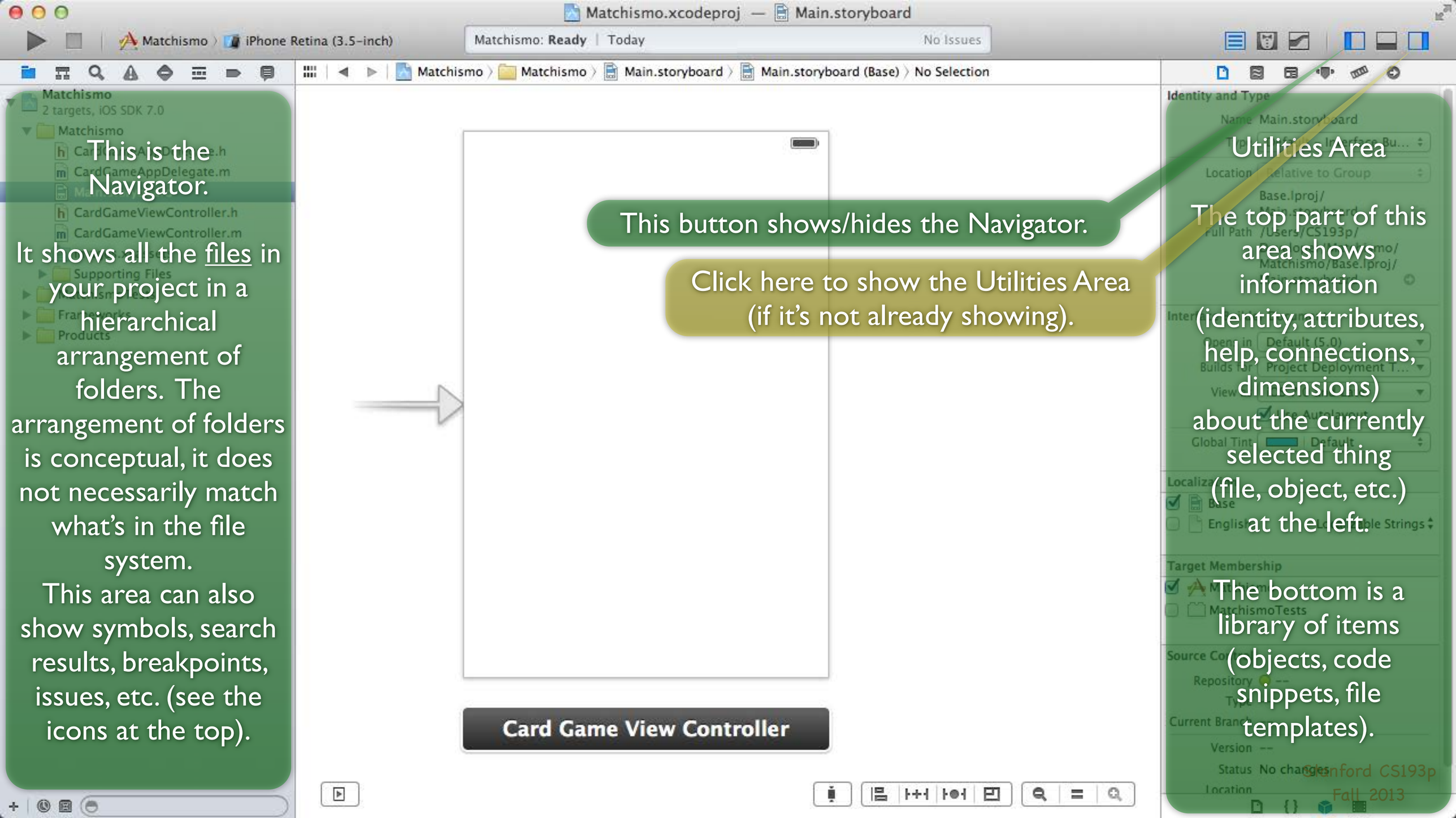

Xcode wants to know where to store this project's directory.
Loading
No Issues
Developer
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
SHARED
DEVICES
Applications
Desktop
Developer
Documents
Downloads
Dropbox
Movies
Music
Pictures
Public
Home directory.
"Developer" folder inside the home directory. There are no projects in it currently.
Navigate to a directory called "Developer" in your home directory (create it if needed).
No Selection
We will hopefully be covering source control in this course. But not for this first project, so leave this switch turned off.
If you don't have a Developer folder in your home directory, you can create it with this New Folder button.
Then click Create to create your project directory inside ~/Developer.
Source Control:
Create git repository on
My Mac
Xcode will place your project under version control
New Folder
Cancel
Create

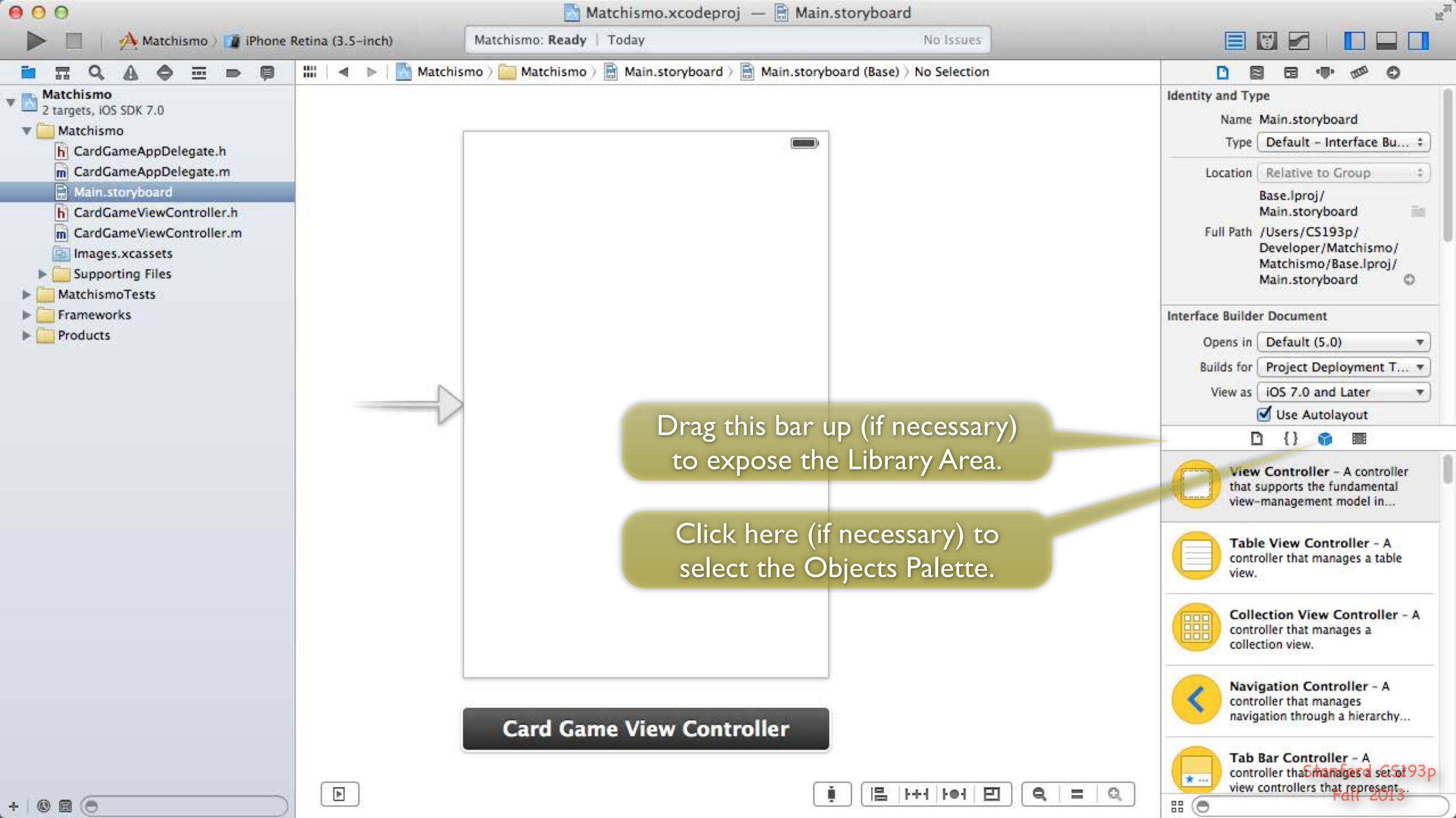

Matchismo.xcodeproj
Matchismo
iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Bundle Identifier
edu.stanford.cs193p.instructor.Matc
Version
1.0
Build
1.0
Team
None
Congratulations, you've created your first iOS Application!
Deployment Info
You'll probably want to make this window as big as possible. Xcode loves screen real-estate!
Devices
iPhone
Main Interface
Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
There's a lot of stuff in this window, but we won't be covering any of it in this first application.
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Launch Images
Source
LaunchImage
Identity and Type
Name
Matchismo
Location
Absolute
Matchismo.xcodeproj
Full Path
/Users/CS193p/Developer/Matchismo/Matchismo.xcodeproj
Project Document
Project Format
Xcode 3.2-compatible
Organization
CS193p
Class Prefix
CardGame
Text Settings
Indent Using
Spaces
Widths
4
4
Tab
Indent
Wrap lines
Source Control
Repository
--
Current Branch
--
Version
--
Status
No changes
Location
Stanford CS193p
Fall 2013

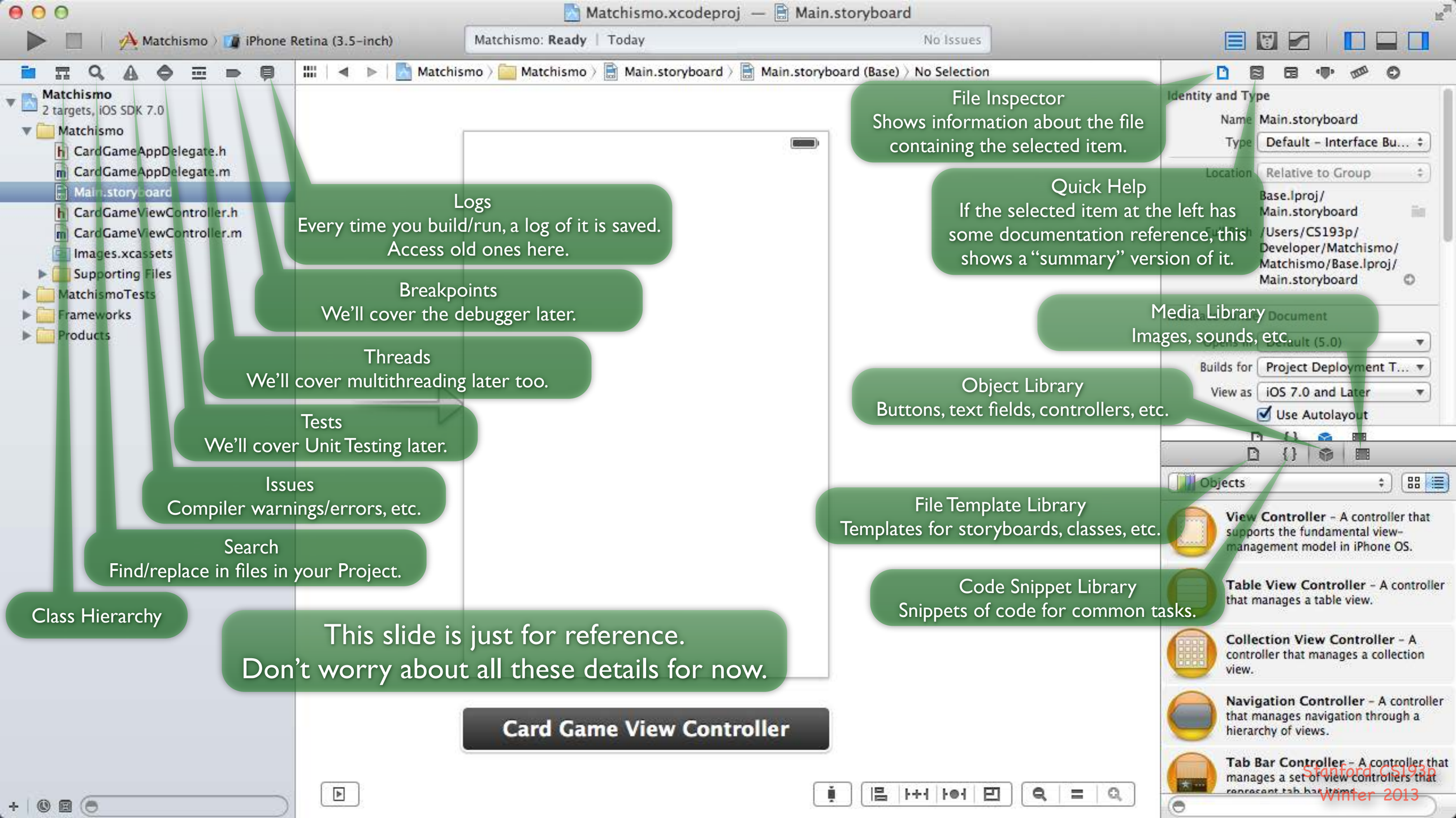

Matchismo.xcodeproj
The Single View Application template we chose at the beginning has created a simple MVC for us.
Our MVC's View is inside Main.storyboard.
CardGameViewController.[mh] is the code for our MVC's Controller.
We'll have to create our MVC's Model ourselves later.
Let's open up and look at our MVC's View by clicking on Main.storyboard.
Don't worry about CardGameAppDelegate.[mh] for this project.
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Deployment Info
Devices
iPhone
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Launch Images
Source
LaunchImage
Identity and Type
Name
Matchismo
Location
Absolute
Full Path
/Users/CS193p/Developer/Matchismo/Matchismo.xcodeproj
Project Document
Project Format
Xcode 3.2-compatible
Organization
CS193p
Class Prefix
CardGame
Text Settings
Indent Using
Spaces
Widths
Tab
Indent
Wrap lines
Source Control
Repository
Type
Current Branch
Version
Status
No changes
Location
Stanford CS193p
Fall 2013

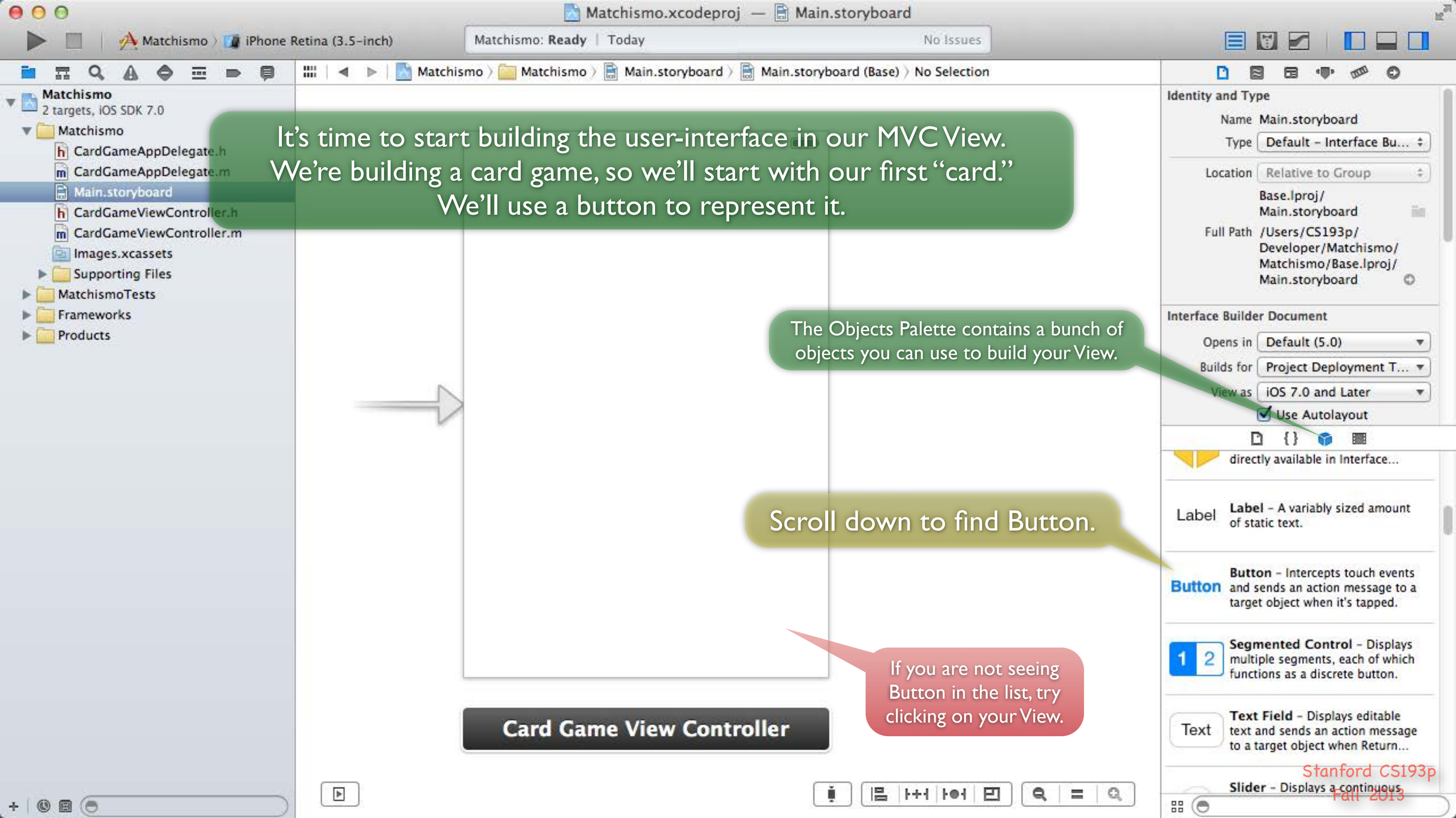

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller Scene
This should be selected.
If it's not, that would explain why
you're not seeing your MVC's View.
This is our MVC's View.
It starts out blank, of course.
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
Global Tint Default
Localization
Base
English
Localizable Strings
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Stanford CS193p
Fall 2013

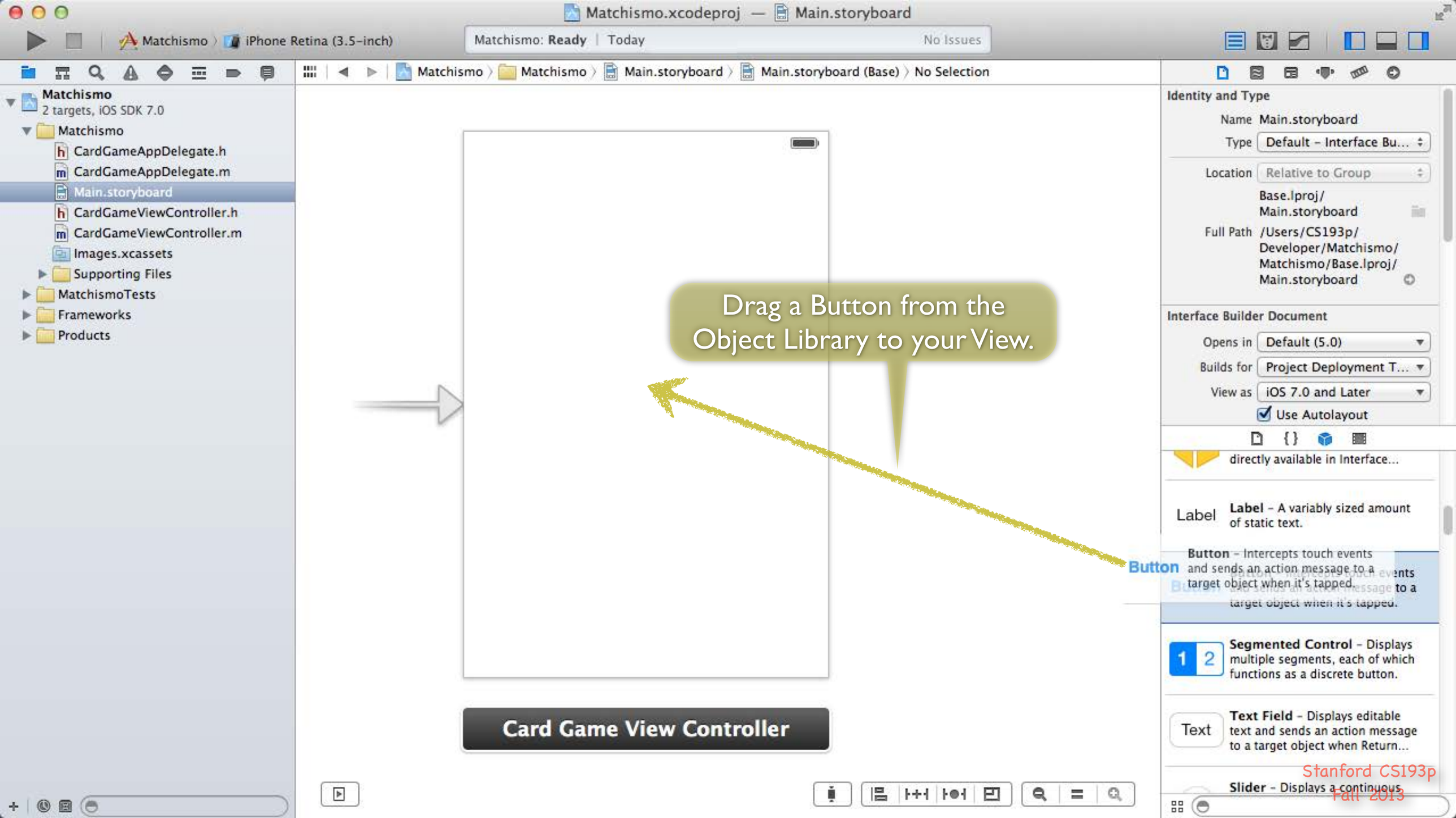

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller Scene
Identity and Type
Name Main.storyboard
Type Default - Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo/Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
Global Tint Default
Localization
Base
English
Localizable Strings
Target Membership
Matchismo
MatchismoTests
Type --
Current Branch --
Version --
Status No changes
Location
Card Game View Controller
This View is sized for the iPhone 5 (i.e. taller).
We're going to design for the iPhone 4.
This area here is called the Document Outline.
We're going to close it to make space.
We'll look at the Document Outline in detail later in the course.
Click here to switch to a iPhone 4-sized View.
We're only doing this because it fits better on these slides!
Close the Document Outline by clicking here.
Stanford CS193p
Fall 2013

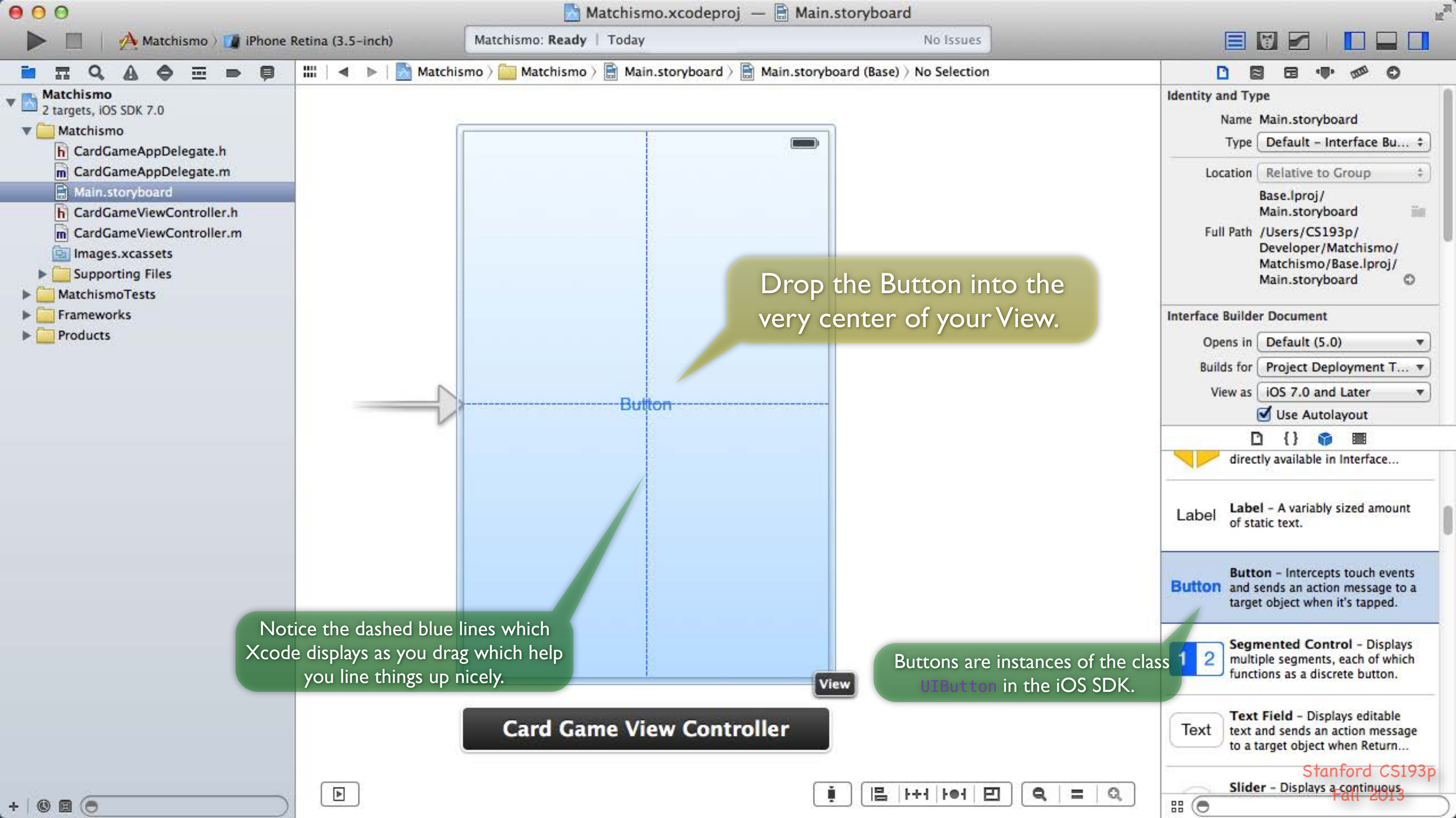

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
This is the Navigator.
It shows all the files in your project in a hierarchical arrangement of folders. The arrangement of folders is conceptual, it does not necessarily match what's in the file system.
This area can also show symbols, search results, breakpoints, issues, etc. (see the icons at the top).
This button shows/hides the Navigator.
Click here to show the Utilities Area (if it's not already showing).
Utilities Area
The top part of this area shows information (identity, attributes, help, connections, dimensions) about the currently selected thing (file, object, etc.) at the left.
The bottom is a library of items (objects, code snippets, file templates).
Card Game View Controller
Stanford CS193p
Fall 2013

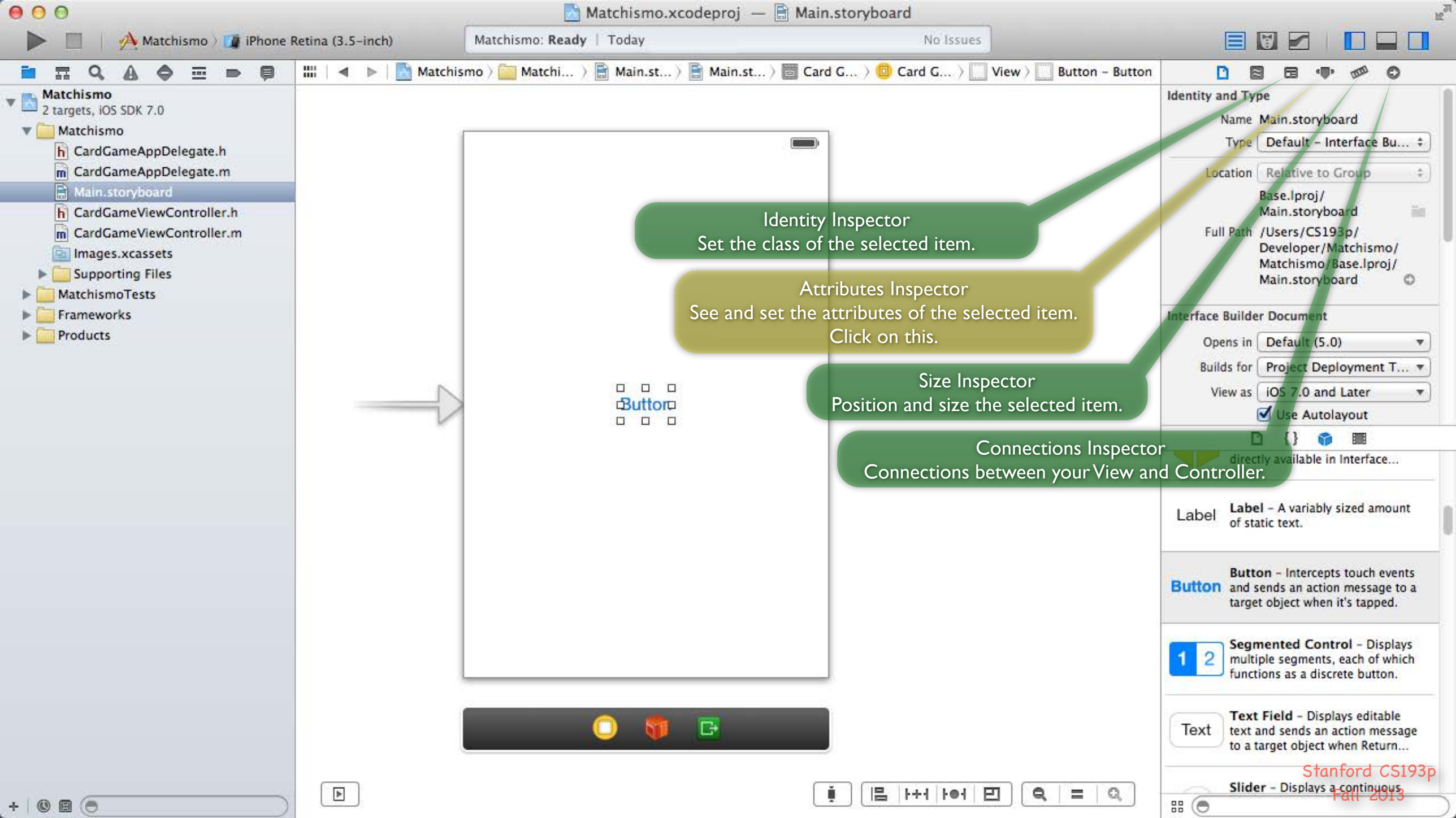

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo/Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
Drag this bar up (if necessary) to expose the Library Area.
Click here (if necessary) to select the Objects Palette.
Card Game View Controller
View Controller – A controller that supports the fundamental view-management model in...
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of view controllers that represent...
Stanford CS193p
Fall 2013

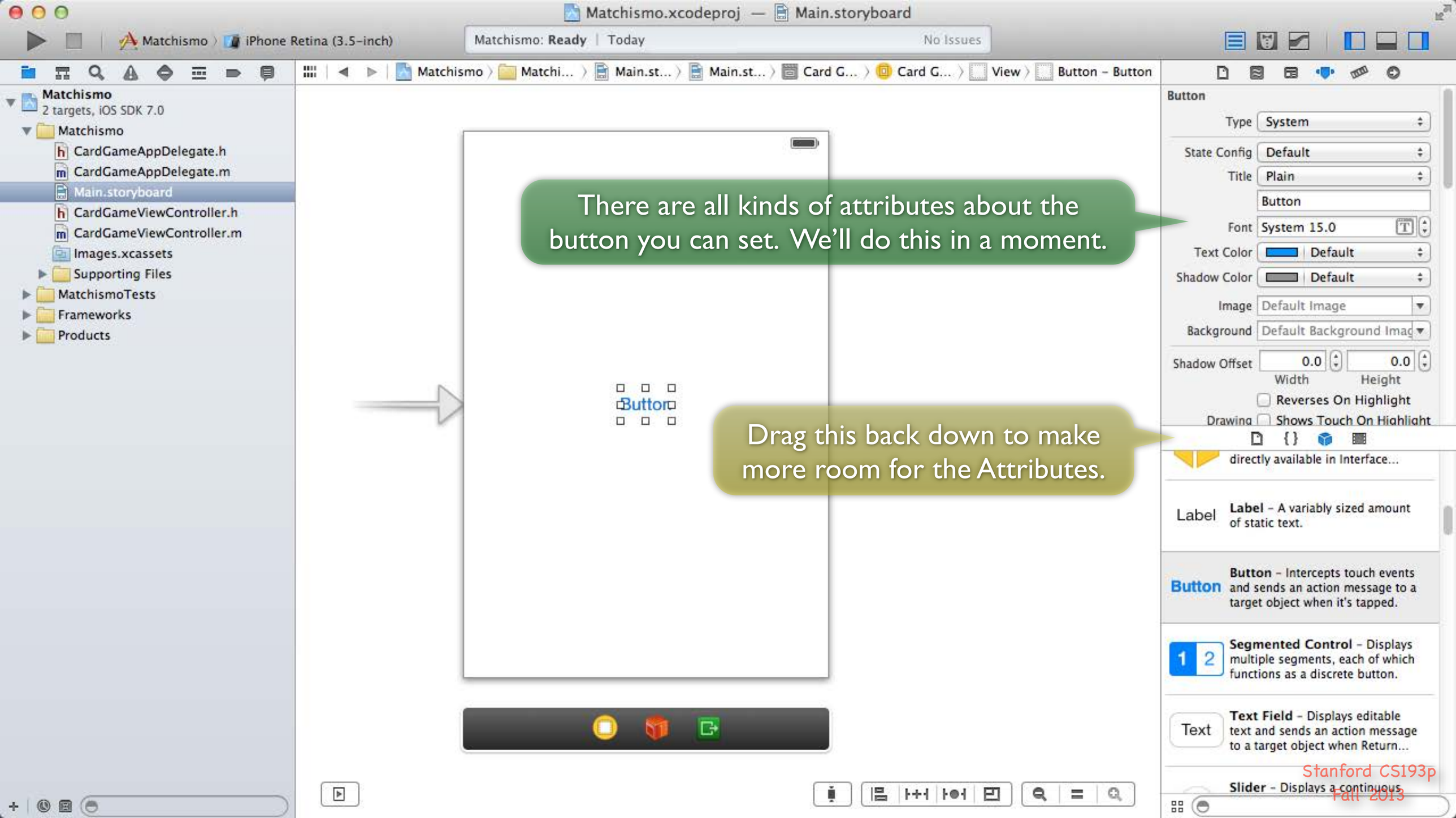

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Logs
Every time you build/run, a log of it is saved.
Access old ones here.
Breakpoints
We'll cover the debugger later.
Threads
We'll cover multithreading later too.
Tests
We'll cover Unit Testing later.
Issues
Compiler warnings/errors, etc.
Search
Find/replace in files in your Project.
Class Hierarchy
This slide is just for reference.
Don't worry about all these details for now.
Card Game View Controller
File Inspector
Shows information about the file containing the selected item.
Quick Help
If the selected item at the left has some documentation reference, this shows a "summary" version of it.
Media Library
Images, sounds, etc.
Object Library
Buttons, text fields, controllers, etc.
File Template Library
Templates for storyboards, classes, etc.
Code Snippet Library
Snippets of code for common tasks.
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
Objects
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that
Stanford CS193p
Winter 2013

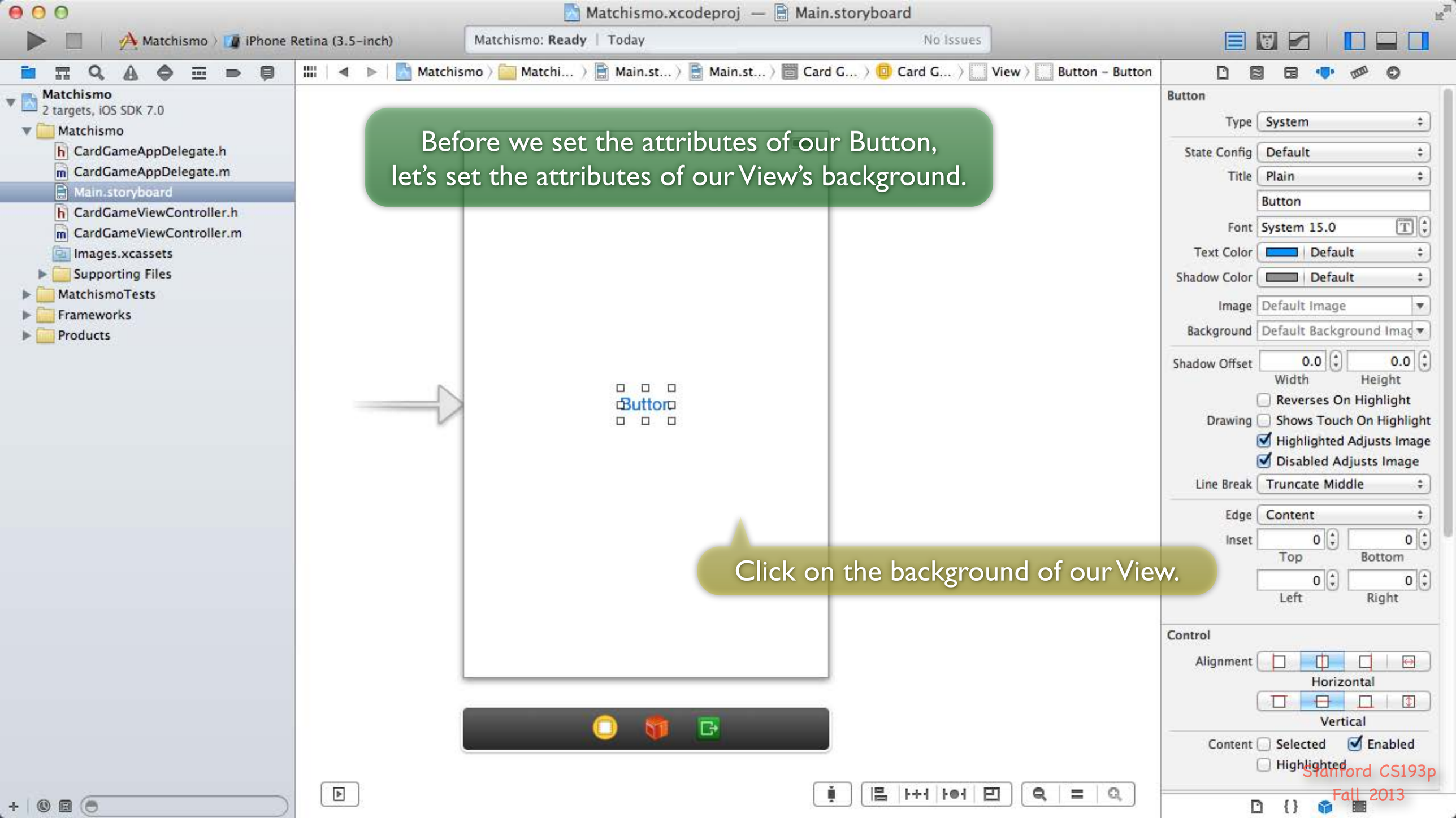

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
It's time to start building the user-interface in our MVC View.
We're building a card game, so we'll start with our first "card."
We'll use a button to represent it.
The Objects Palette contains a bunch of objects you can use to build your View.
Scroll down to find Button.
If you are not seeing Button in the list, try clicking on your View.
Card Game View Controller
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo/Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
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 text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

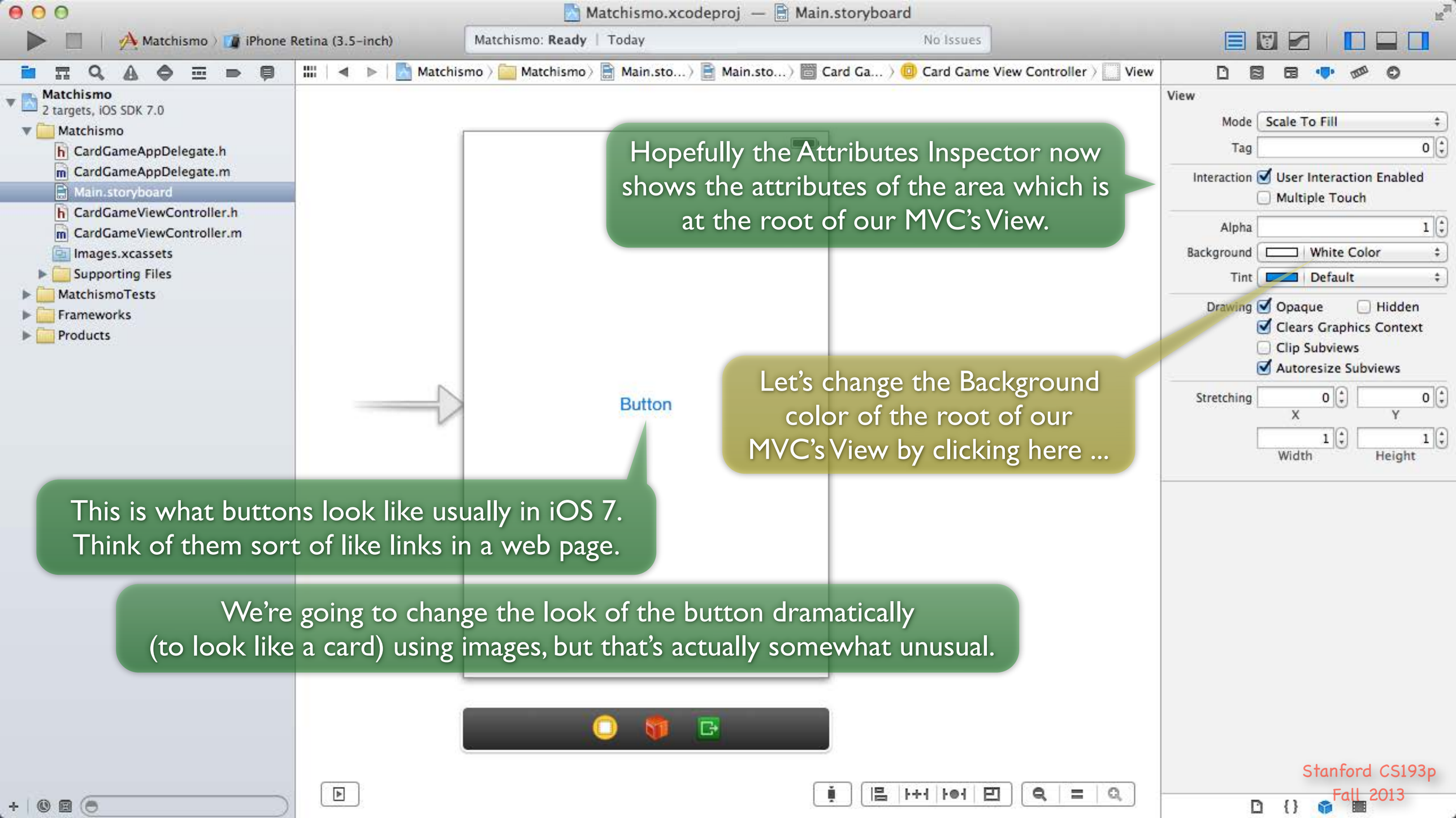

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo › Matchismo › Main.storyboard › Main.storyboard (Base) › No Selection
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Drag a Button from the Object Library to your View.
Card Game View Controller
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo/Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
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
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

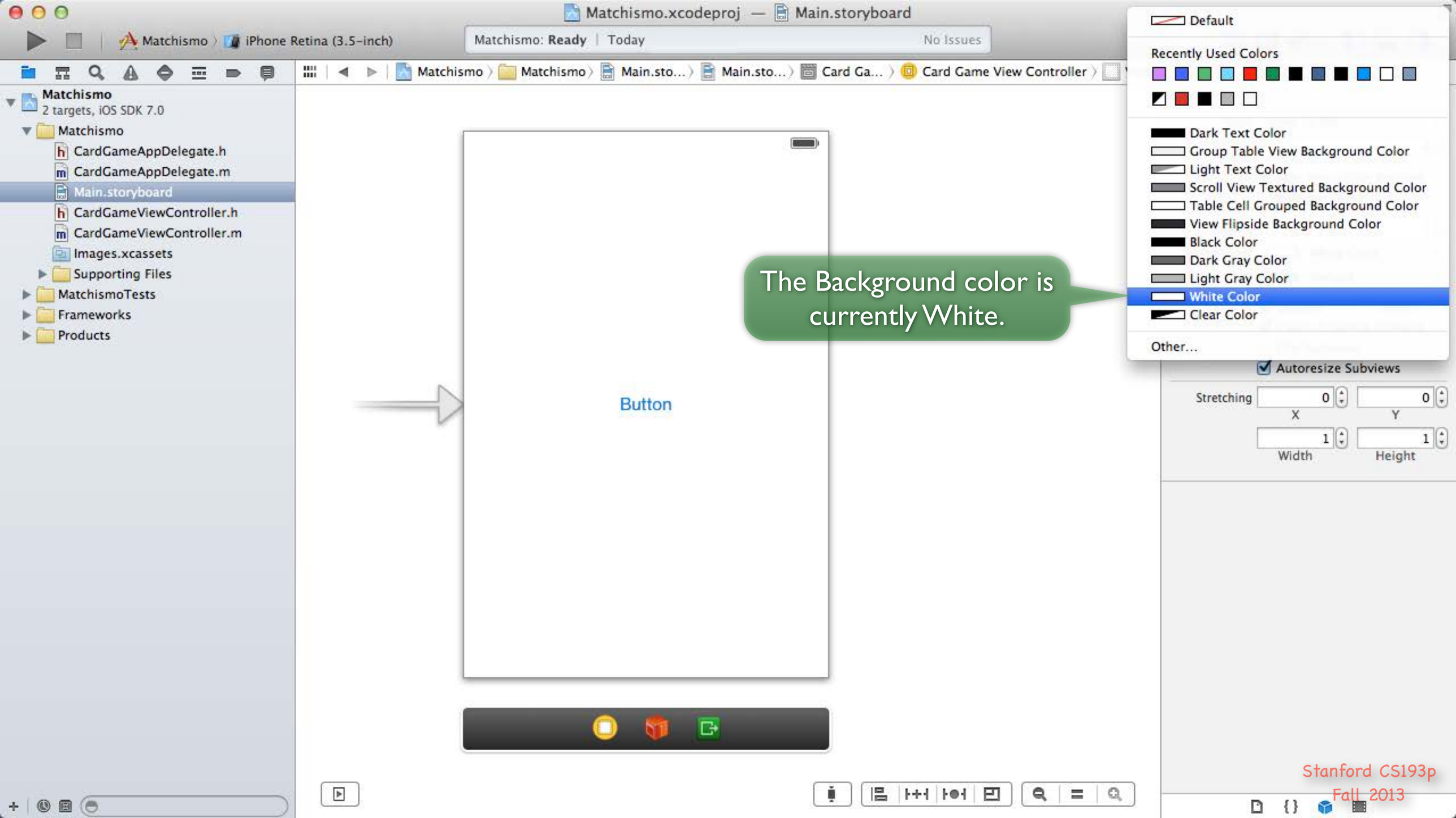

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo 2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/ Main.storyboard
Full Path /Users/CS193p/ Developer/Matchismo/ Matchismo/Base.lproj/ Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
Drop the Button into the very center of your View.
Button
Notice the dashed blue lines which Xcode displays as you drag which help you line things up nicely.
View
Buttons are instances of the class UIButton in the iOS SDK.
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 text and sends an action message to a target object when Return...
Slider – Displays a continuous
Card Game View Controller
Stanford CS193p
Fall 2013

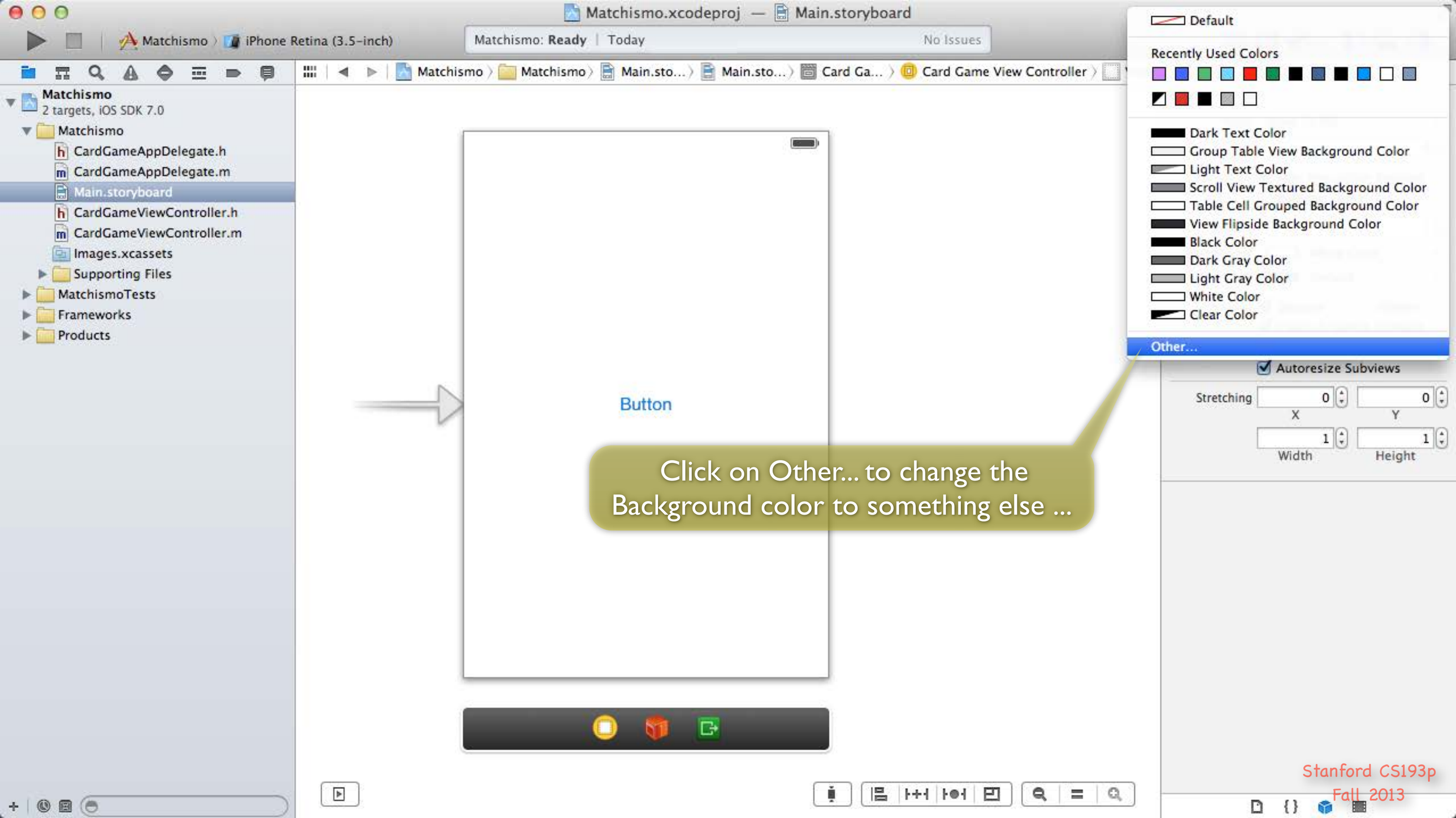

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchi... › Main.st... › Main.st... › Card G... › Card G... › View › Button – Button
Button
Identity Inspector
Set the class of the selected item.
Attributes Inspector
See and set the attributes of the selected item.
Click on this.
Size Inspector
Position and size the selected item.
Connections Inspector
Connections between your View and Controller.
Identity and Type
Name Main.storyboard
Type Default – Interface Bu...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo/Base.lproj/
Main.storyboard
Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Autolayout
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
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

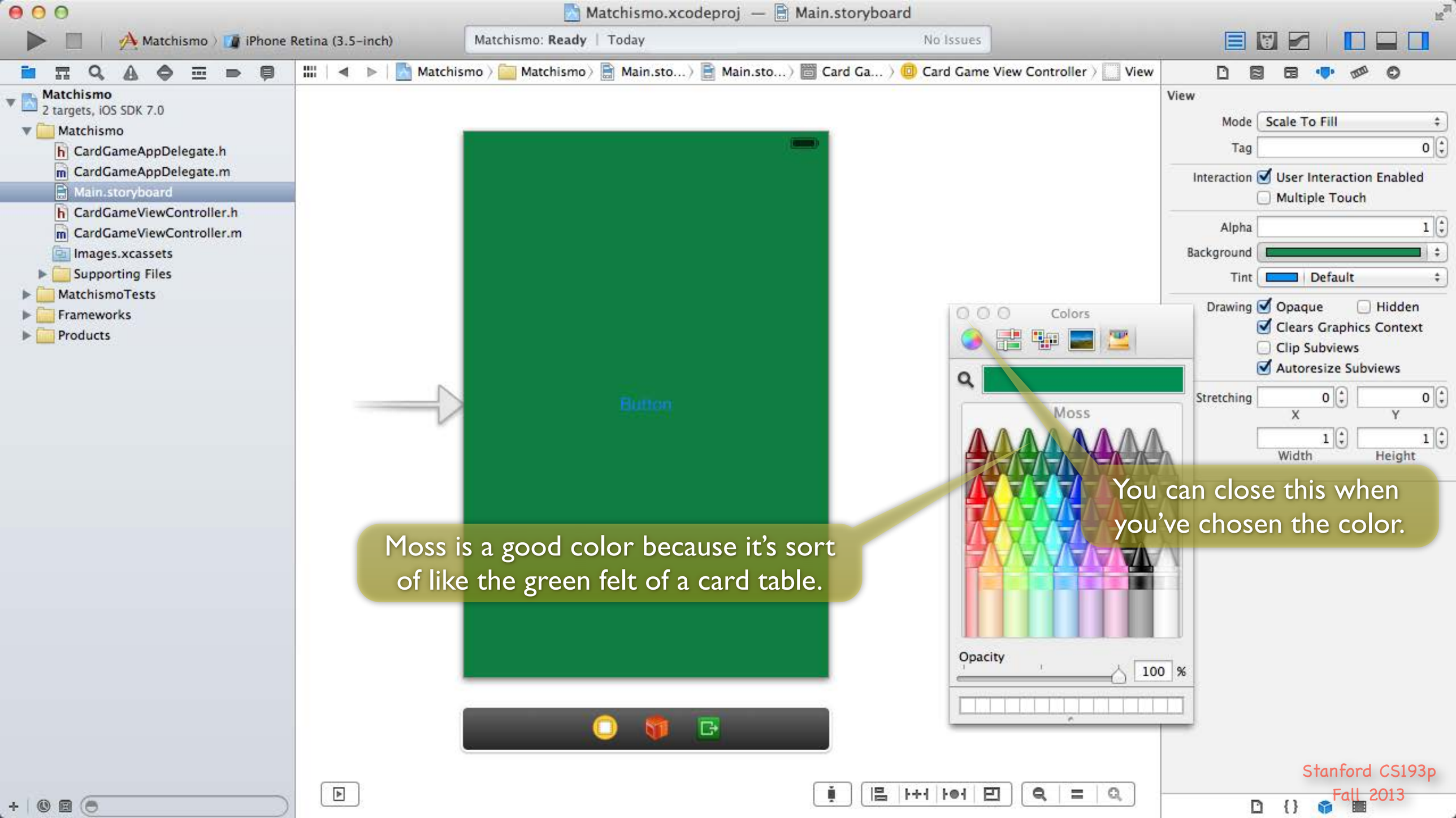

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Button
Type
System
State Config
Default
Title
Plain
Font
System 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Shadow Offset
Width
Height
Reverses On Highlight
Shows Touch On Highlight
There are all kinds of attributes about the button you can set. We'll do this in a moment.
Drag this back down to make more room for the Attributes.
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 text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

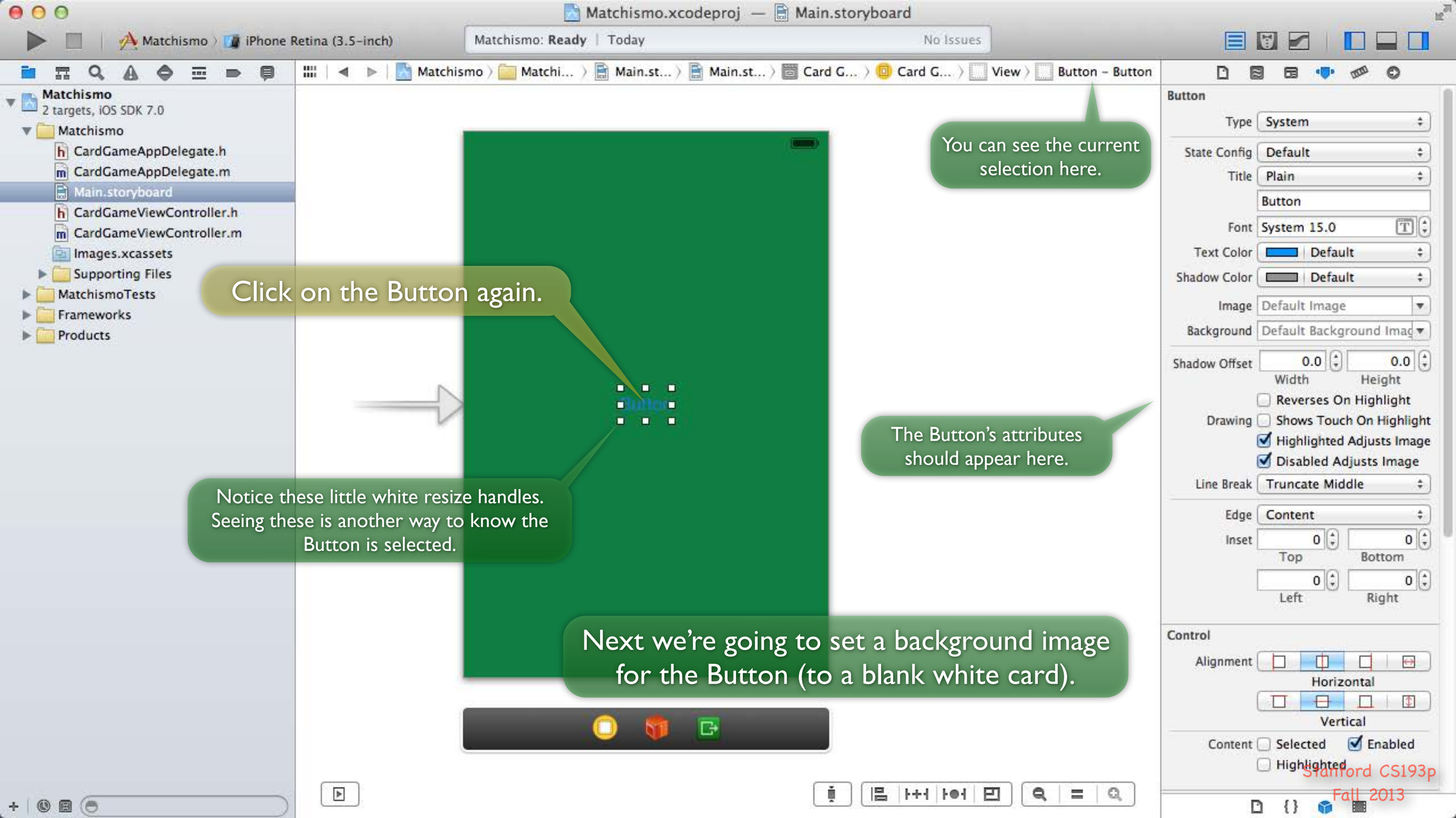

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Button – Button
Before we set the attributes of our Button, let's set the attributes of our View's background.
Button
Click on the background of our View.
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Imag
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
Stanford CS193p
Fall 2013

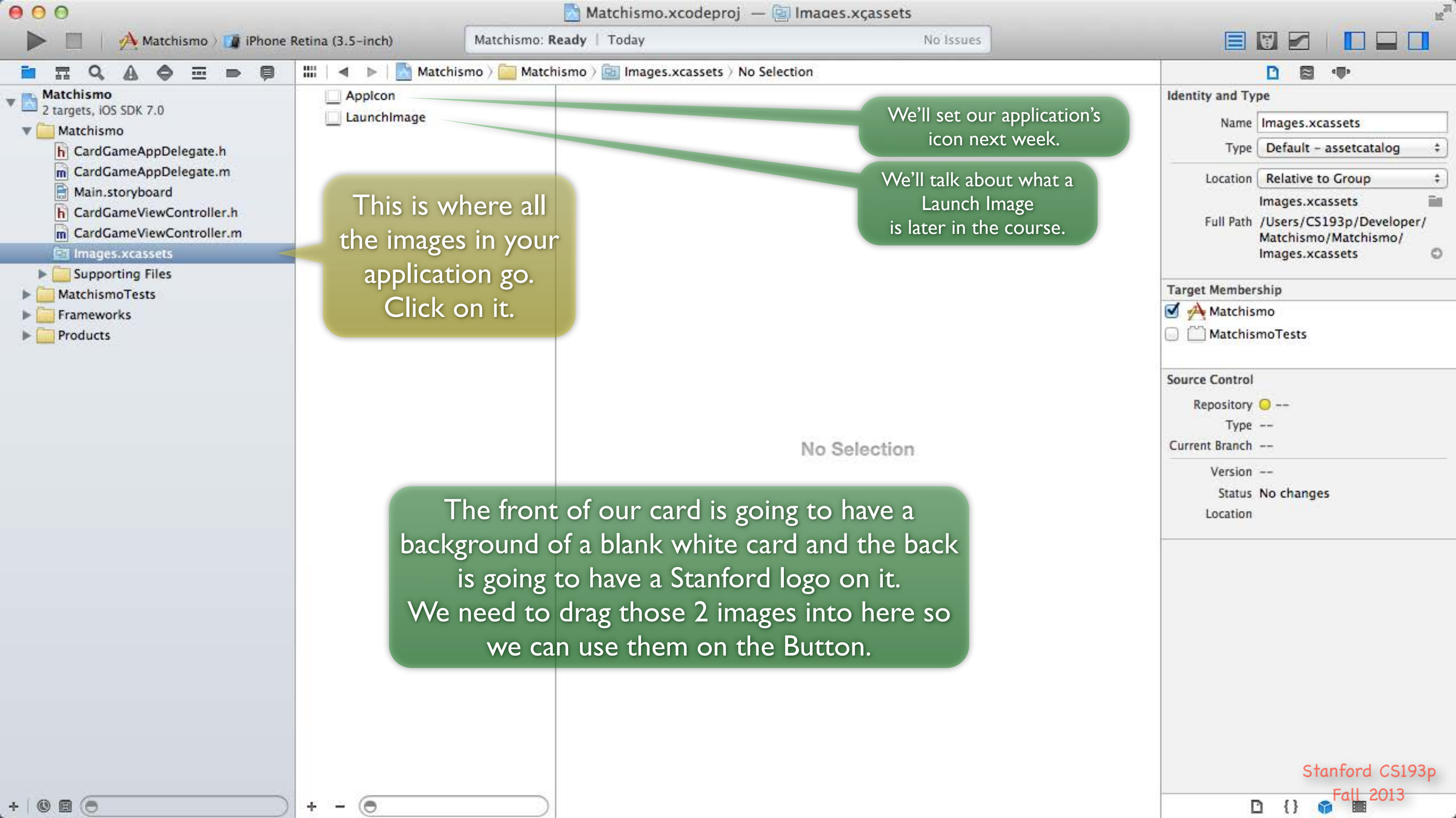

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller
View
Button
Hopefully the Attributes Inspector now shows the attributes of the area which is at the root of our MVC's View.
Let's change the Background color of the root of our MVC's View by clicking here ...
This is what buttons look like usually in iOS 7. Think of them sort of like links in a web page.
We're going to change the look of the button dramatically (to look like a card) using images, but that's actually somewhat unusual.
Mode
Scale To Fill
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
White Color
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Width
Height
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller
Button
Default
Recently Used Colors
Dark Text Color
Group Table View Background Color
Light Text Color
Scroll View Textured Background Color
Table Cell Grouped Background Color
View Flipside Background Color
Black Color
Dark Gray Color
Light Gray Color
White Color
Clear Color
Other...
The Background color is currently White.
Autoresize Subviews
Stretching
X
Y
Width
Height
Stanford CS193p
Fall 2013

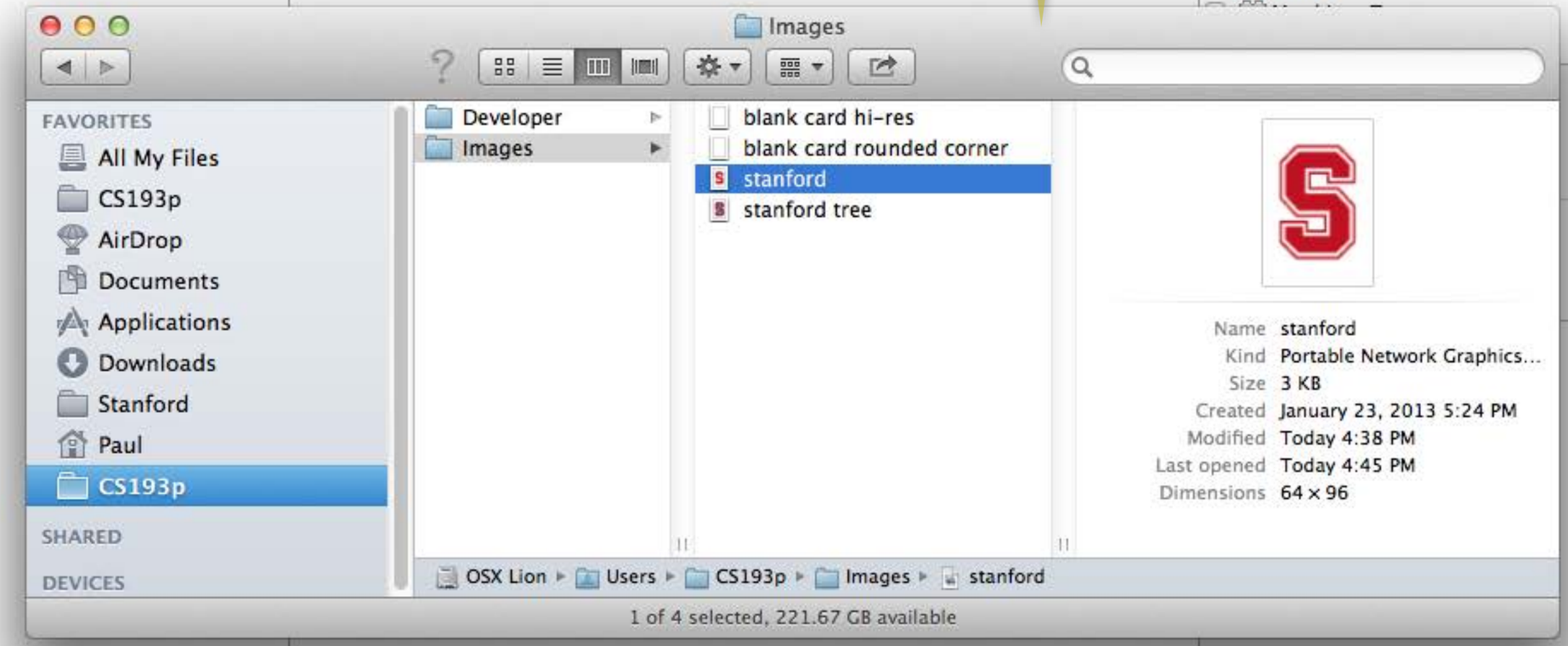

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.sto... 〉 Main.sto... 〉 Card Ga... 〉 Card Game View Controller
Button
Default
Recently Used Colors
Dark Text Color
Group Table View Background Color
Light Text Color
Scroll View Textured Background Color
Table Cell Grouped Background Color
View Flipside Background Color
Black Color
Dark Gray Color
Light Gray Color
White Color
Clear Color
Other...
Autoresize Subviews
Stretching
X
Y
Width
Height
Click on Other... to change the Background color to something else ...
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller
View
Button
Colors
Moss
Opacity
100 %
Mode
Scale To Fill
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Width
Height
Moss is a good color because it's sort of like the green felt of a card table.
You can close this when you've chosen the color.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
You can see the current selection here.
Click on the Button again.
Notice these little white resize handles. Seeing these is another way to know the Button is selected.
The Button's attributes should appear here.
Next we're going to set a background image for the Button (to a blank white card).
Button
Type System
State Config Default
Title Plain
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Imag
Shadow Offset
Reverses On Highlight
Drawing
Shows Touch On Highlight
Highlighted Adjusts Image
Disabled Adjusts Image
Line Break Truncate Middle
Edge Content
Inset
Control
Alignment
Horizontal
Vertical
Content
Selected
Enabled
Highlighted
Stanford CS193p
Fall 2013

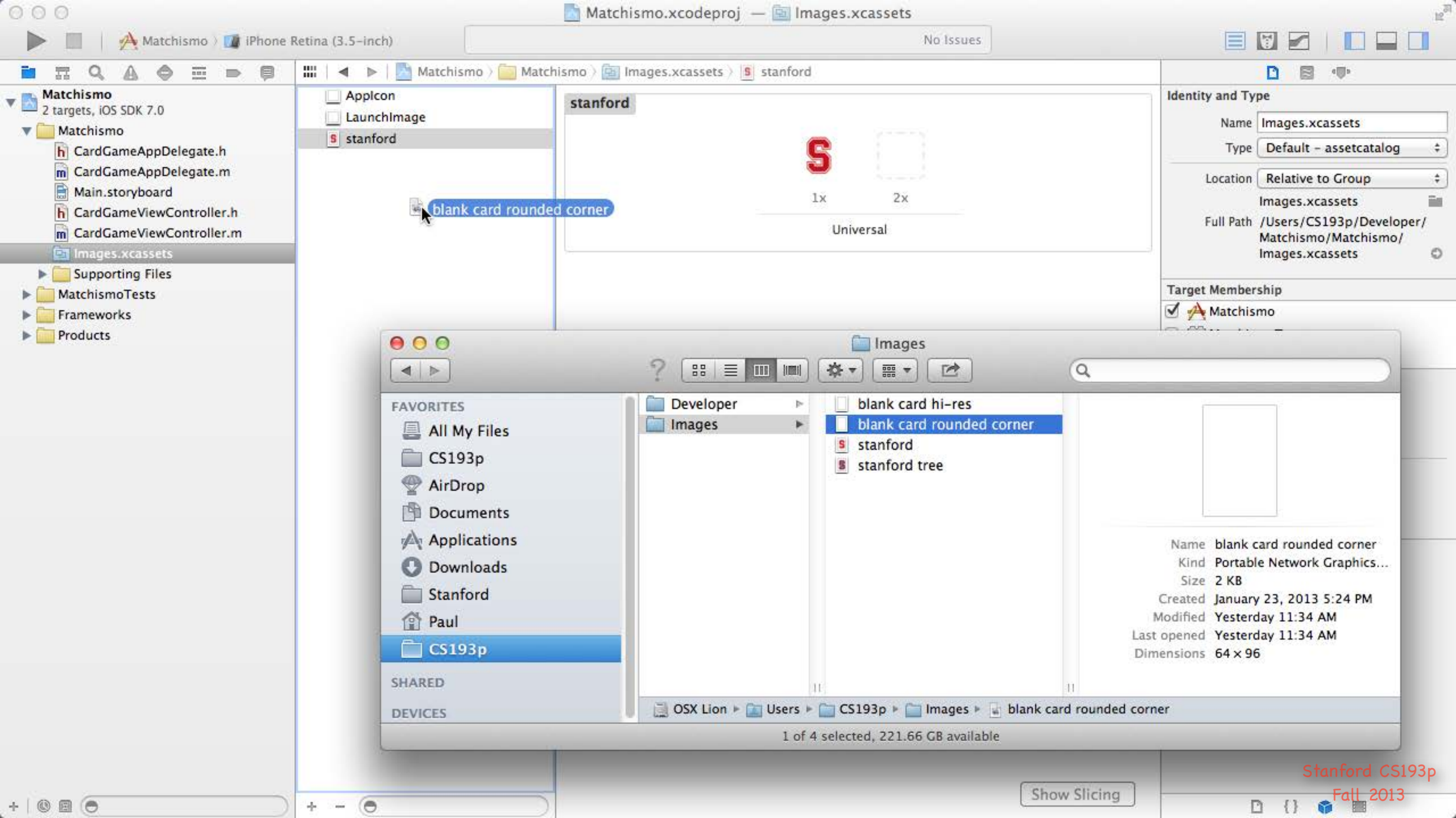

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 No Selection
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
LaunchImage
We'll set our application's icon next week.
We'll talk about what a Launch Image is later in the course.
This is where all the images in your application go. Click on it.
No Selection
The front of our card is going to have a background of a blank white card and the back is going to have a Stanford logo on it. We need to drag those 2 images into here so we can use them on the Button.
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Stanford CS193p
Fall 2013

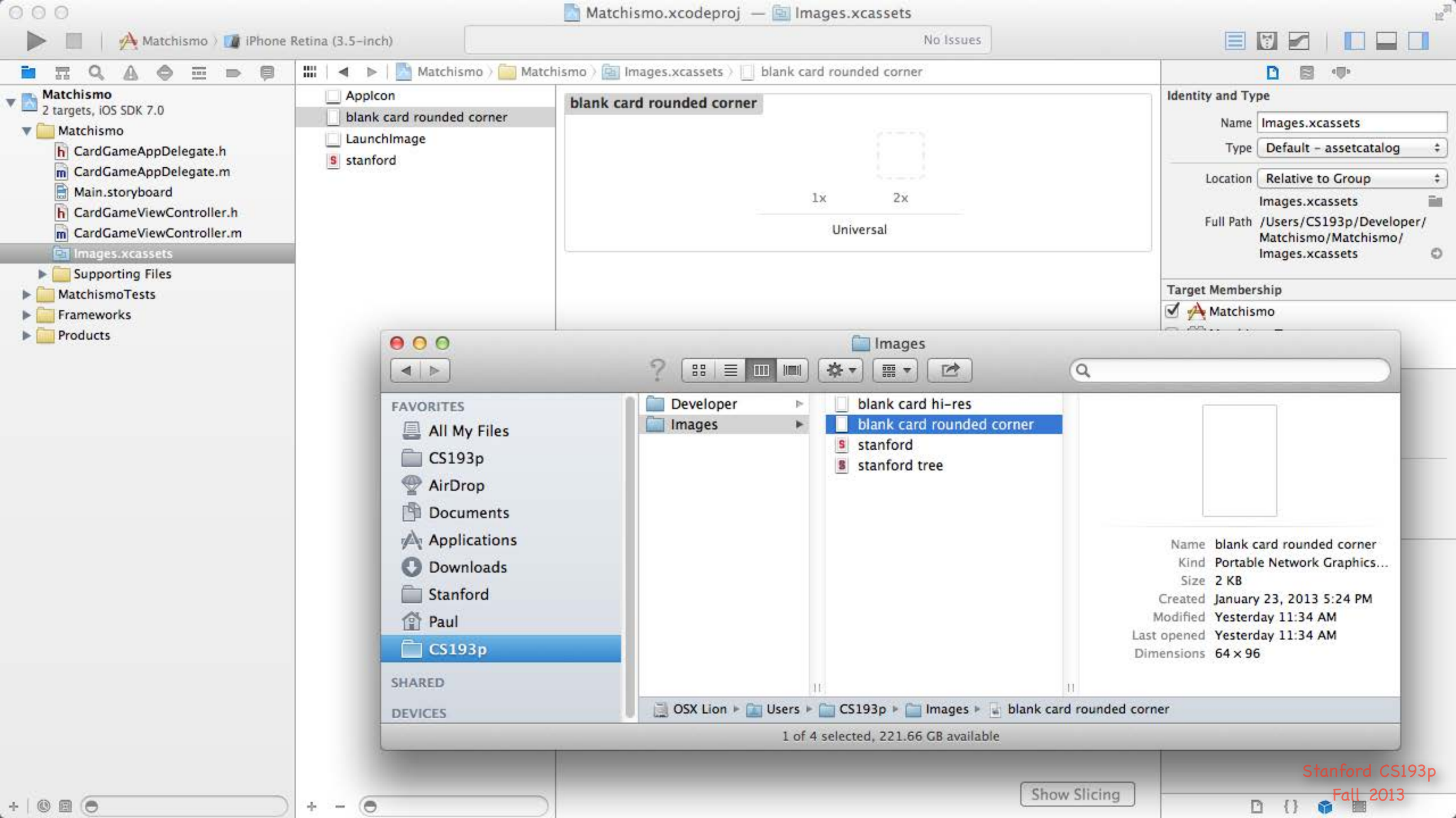
Bring up the Finder with the images we want to use (linked in your homework assignment).

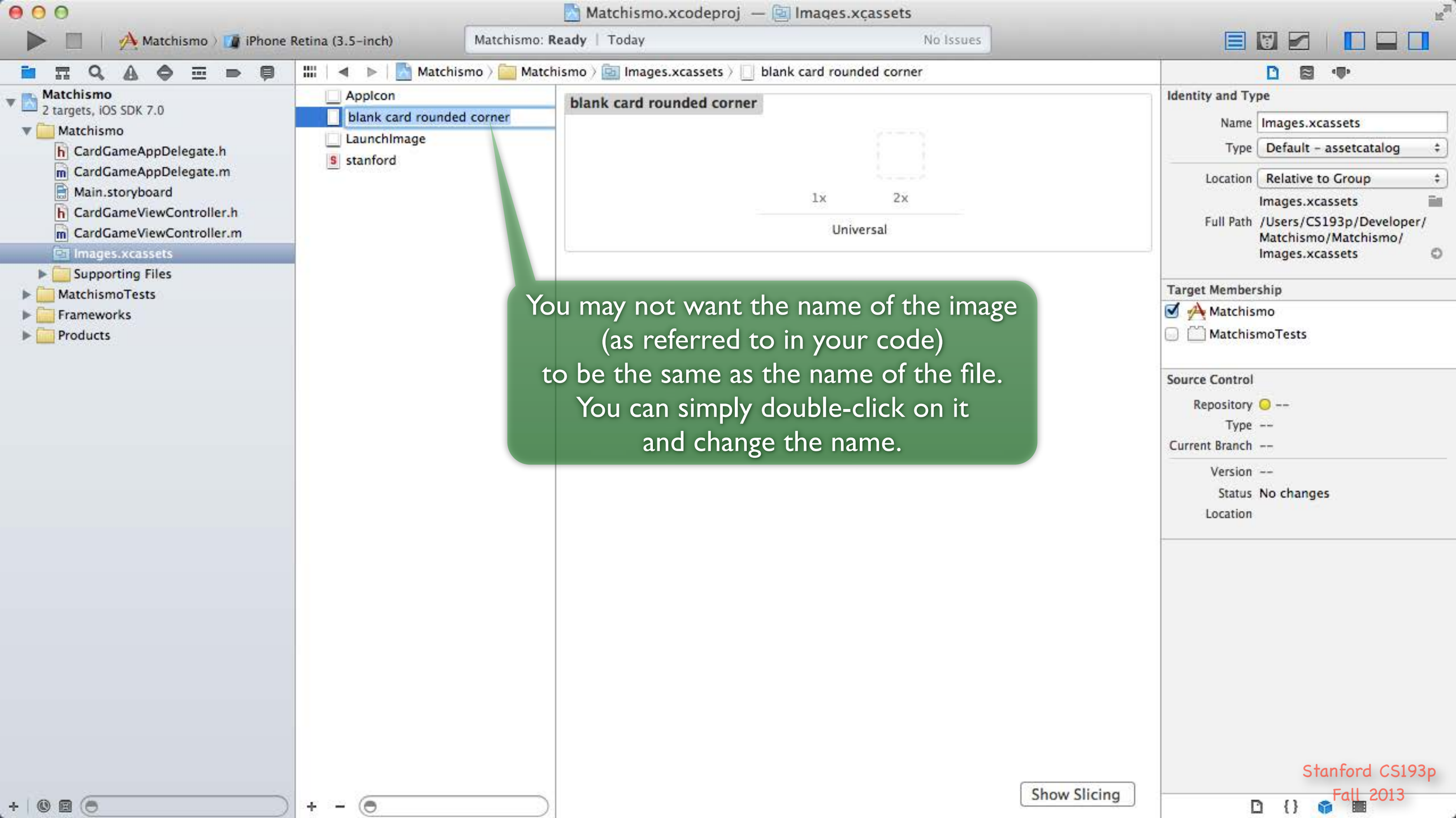


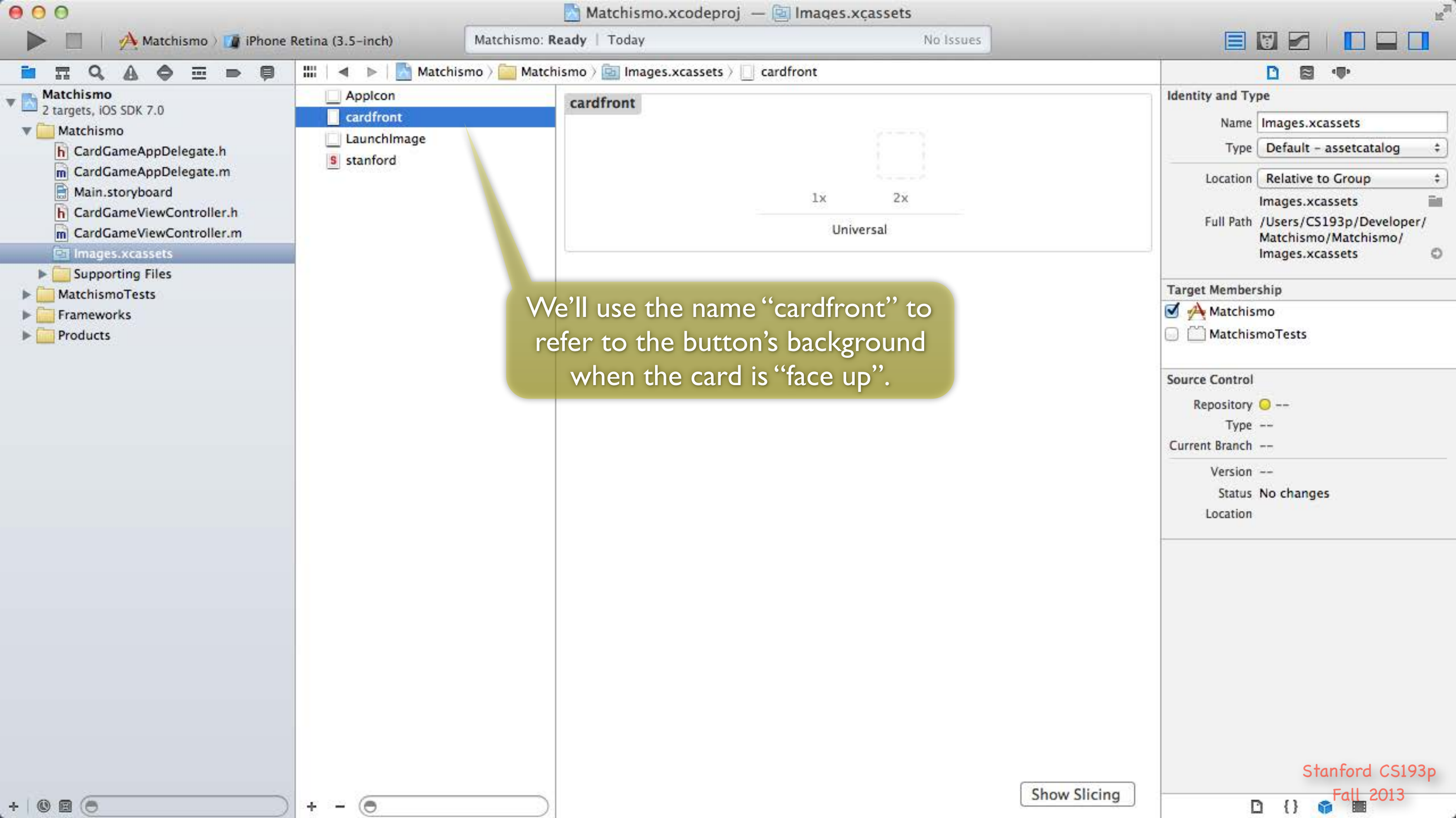
Then drag them
(one by one or all at the same time)
into Xcode.

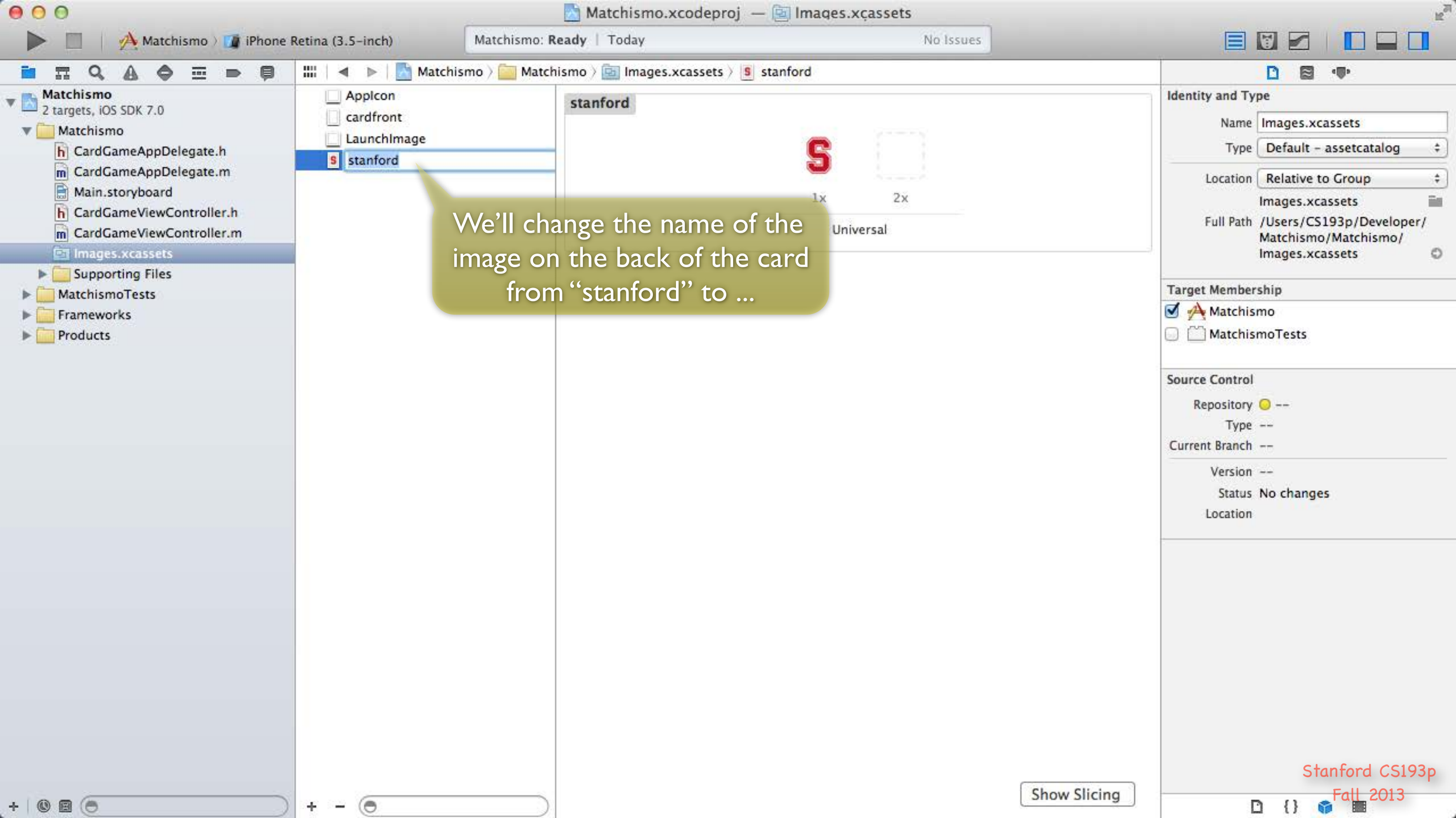


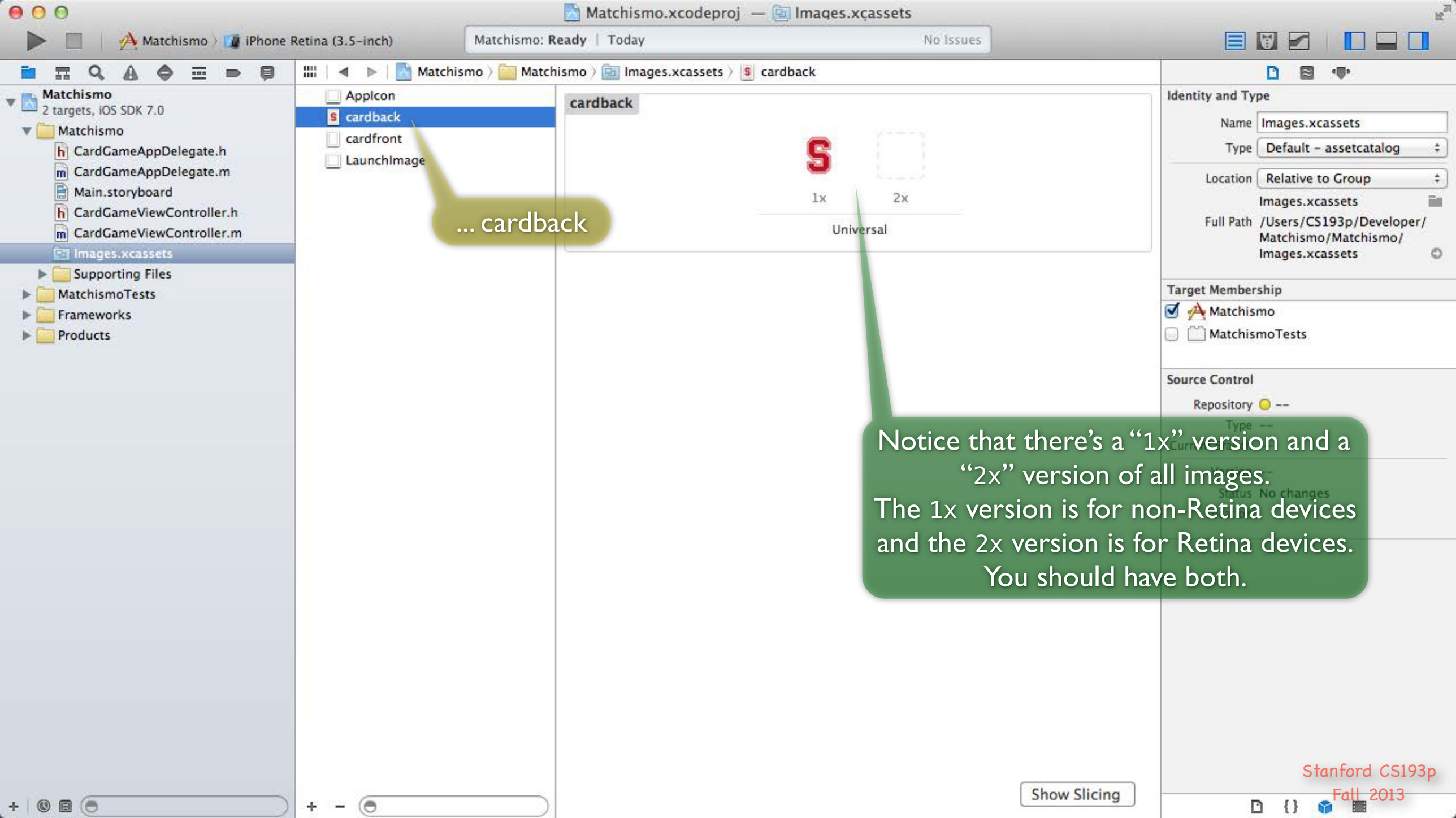

Matchismo.xcodeproj — Images.xcassets
Matchismo › iPhone Retina (3.5-inch)
No Issues
Matchismo › Matchismo › Images.xcassets › stanford
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
LaunchImage
stanford
blank card rounded corner
stanford
1x
2x
Universal
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
Images
FAVORITES
All My Files
CS193p
AirDrop
Documents
Applications
Downloads
Stanford
Paul
CS193p
SHARED
DEVICES
Developer
Images
blank card hi-res
blank card rounded corner
stanford
stanford tree
Name blank card rounded corner
Kind Portable Network Graphics...
Size 2 KB
Created January 23, 2013 5:24 PM
Modified Yesterday 11:34 AM
Last opened Yesterday 11:34 AM
Dimensions 64 × 96
OSX Lion ▸ Users ▸ CS193p ▸ Images ▸ blank card rounded corner
1 of 4 selected, 221.66 GB available
Show Slicing
Stanford CS193p
Fall 2013

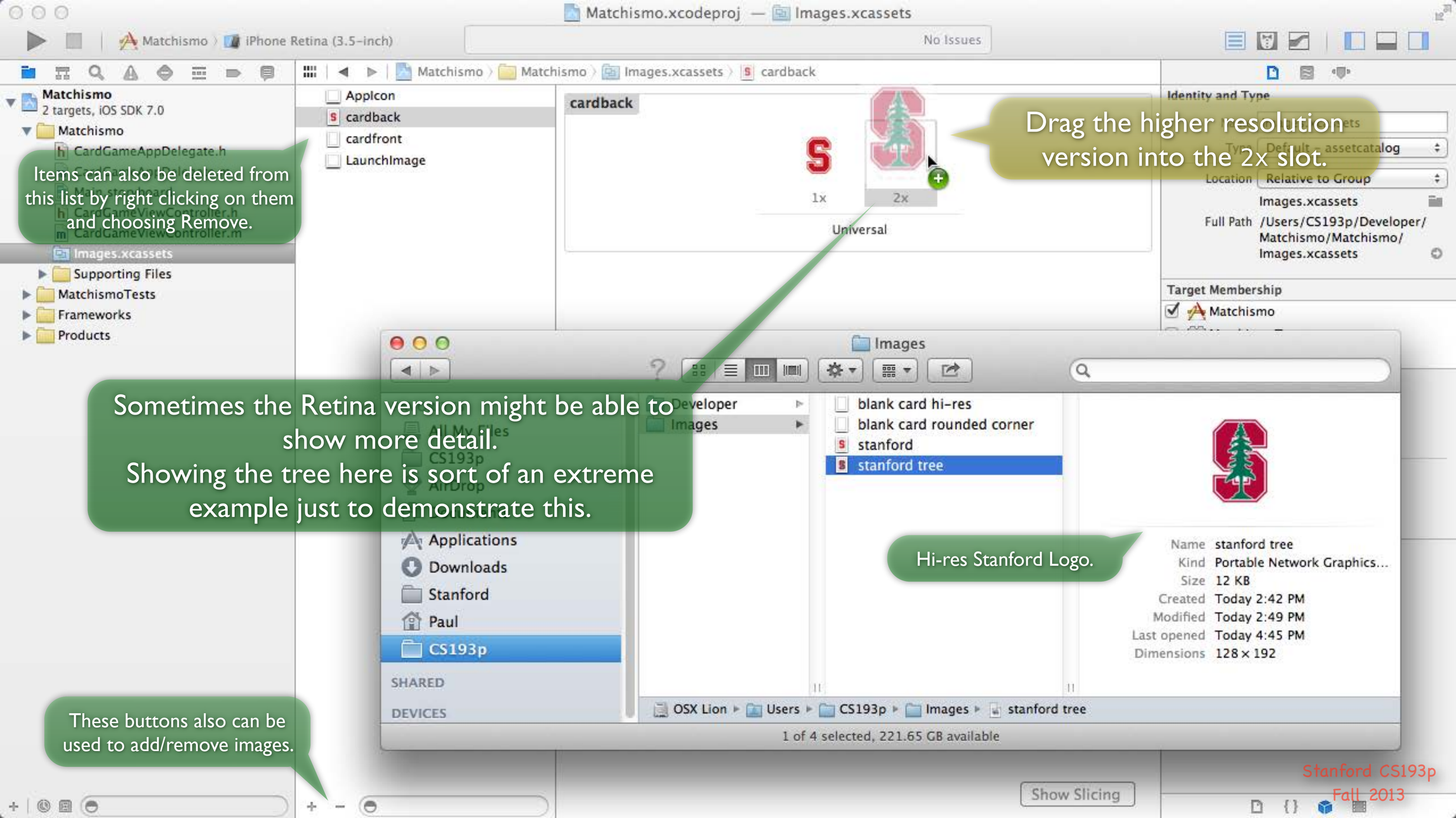

Matchismo.xcodeproj — Images.xcassets
Matchismo
iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Images.xcassets › blank card rounded corner
AppIcon
blank card rounded corner
LaunchImage
stanford
blank card rounded corner
1x
2x
Universal
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
Images
FAVORITES
All My Files
CS193p
AirDrop
Documents
Applications
Downloads
Stanford
Paul
CS193p
SHARED
DEVICES
Developer
Images
blank card hi-res
blank card rounded corner
stanford
stanford tree
Name blank card rounded corner
Kind Portable Network Graphics...
Size 2 KB
Created January 23, 2013 5:24 PM
Modified Yesterday 11:34 AM
Last opened Yesterday 11:34 AM
Dimensions 64 × 96
OSX Lion ▸ Users ▸ CS193p ▸ Images ▸ blank card rounded corner
1 of 4 selected, 221.66 GB available
Show Slicing
Stanford CS193p
Fall 2013

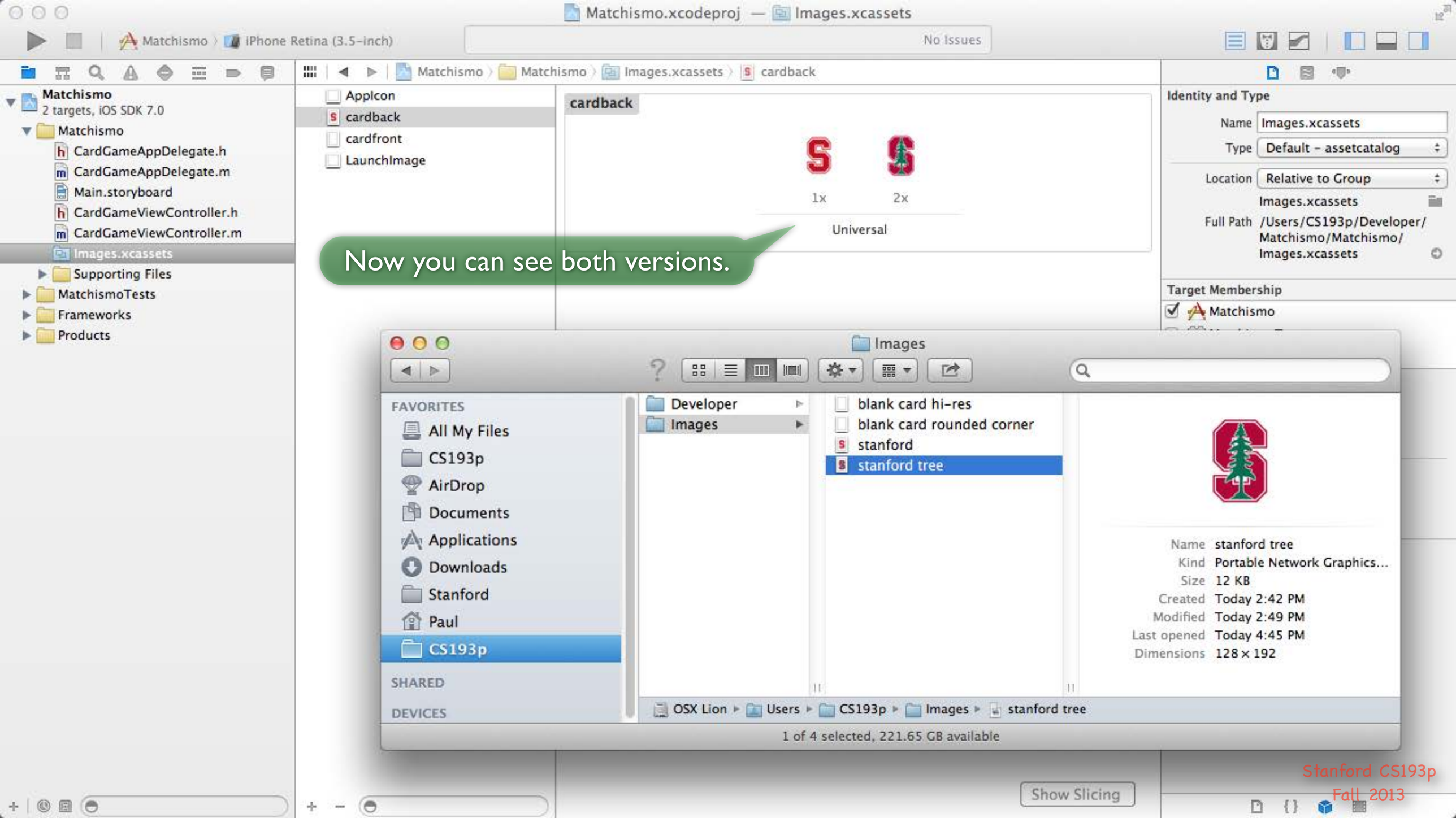

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 blank card rounded corner
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
blank card rounded corner
LaunchImage
stanford
blank card rounded corner
1x
2x
Universal
You may not want the name of the image
(as referred to in your code)
to be the same as the name of the file.
You can simply double-click on it
and change the name.
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Show Slicing
Stanford CS193p
Fall 2013

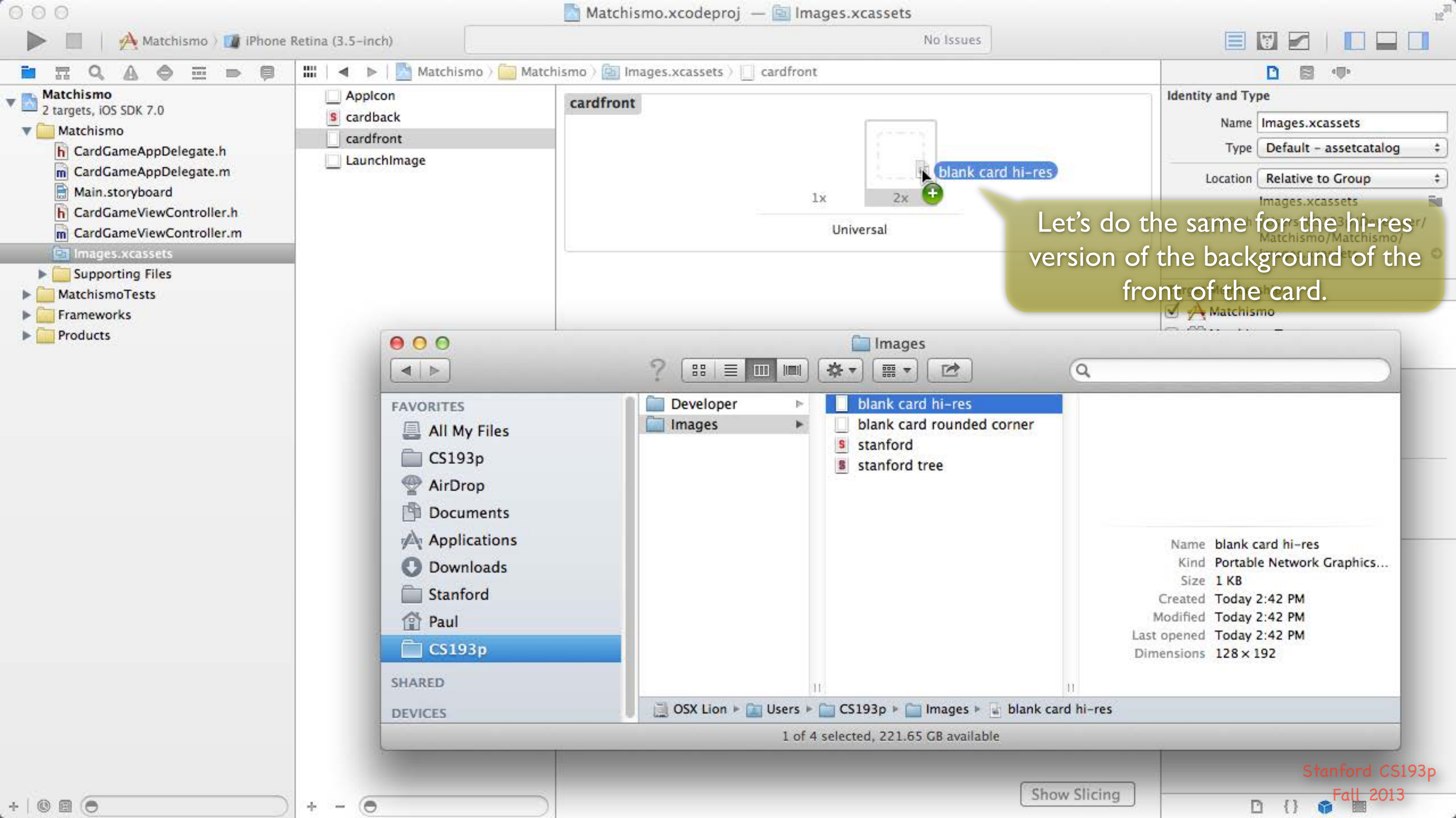

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 cardfront
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
cardfront
LaunchImage
stanford
cardfront
1x
2x
Universal
We'll use the name "cardfront" to refer to the button's background when the card is "face up".
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Show Slicing
Stanford CS193p
Fall 2013

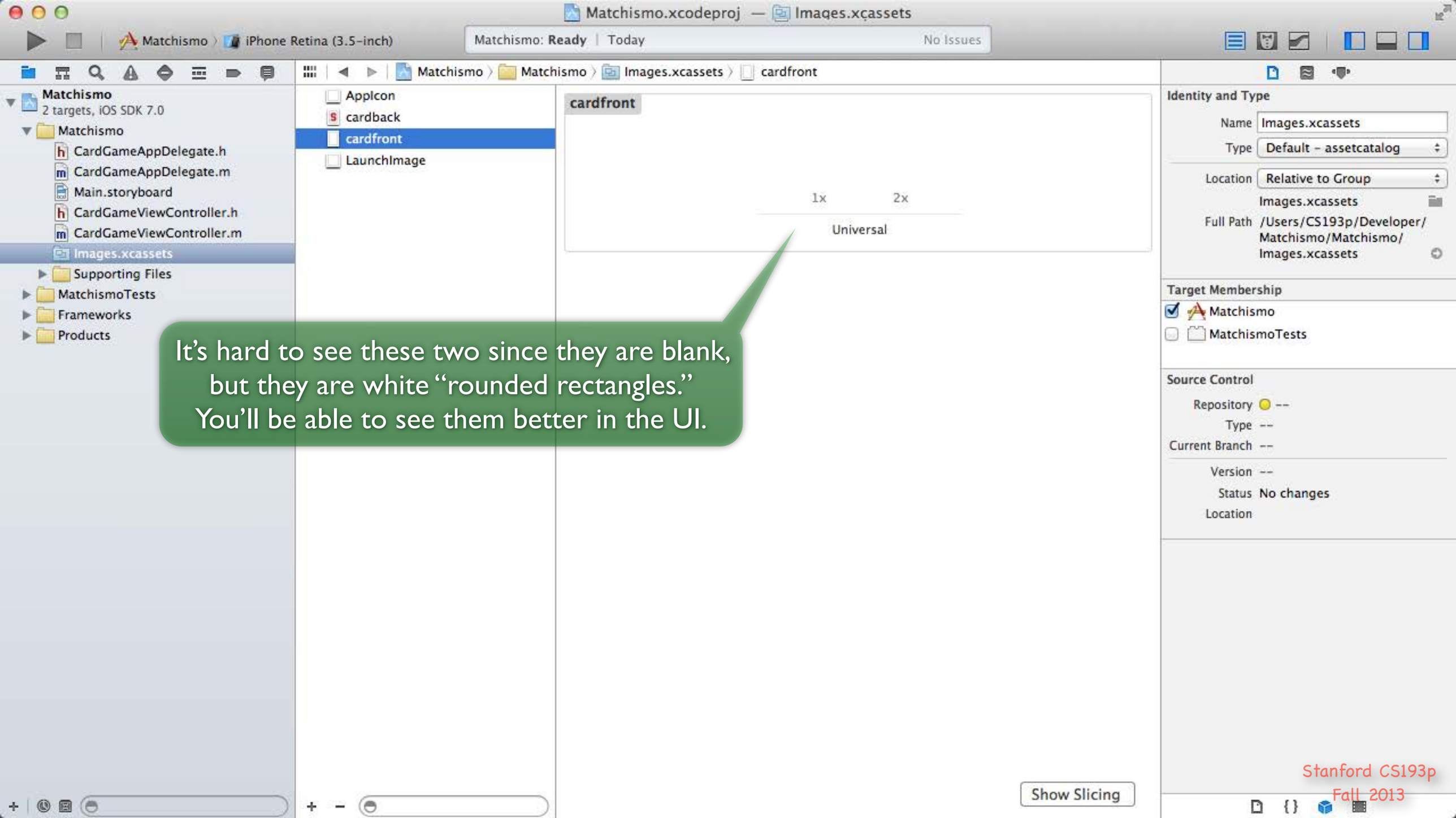

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 stanford
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
cardfront
LaunchImage
stanford
stanford
1x
2x
Universal
We'll change the name of the image on the back of the card from "stanford" to ...
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Show Slicing
Stanford CS193p
Fall 2013

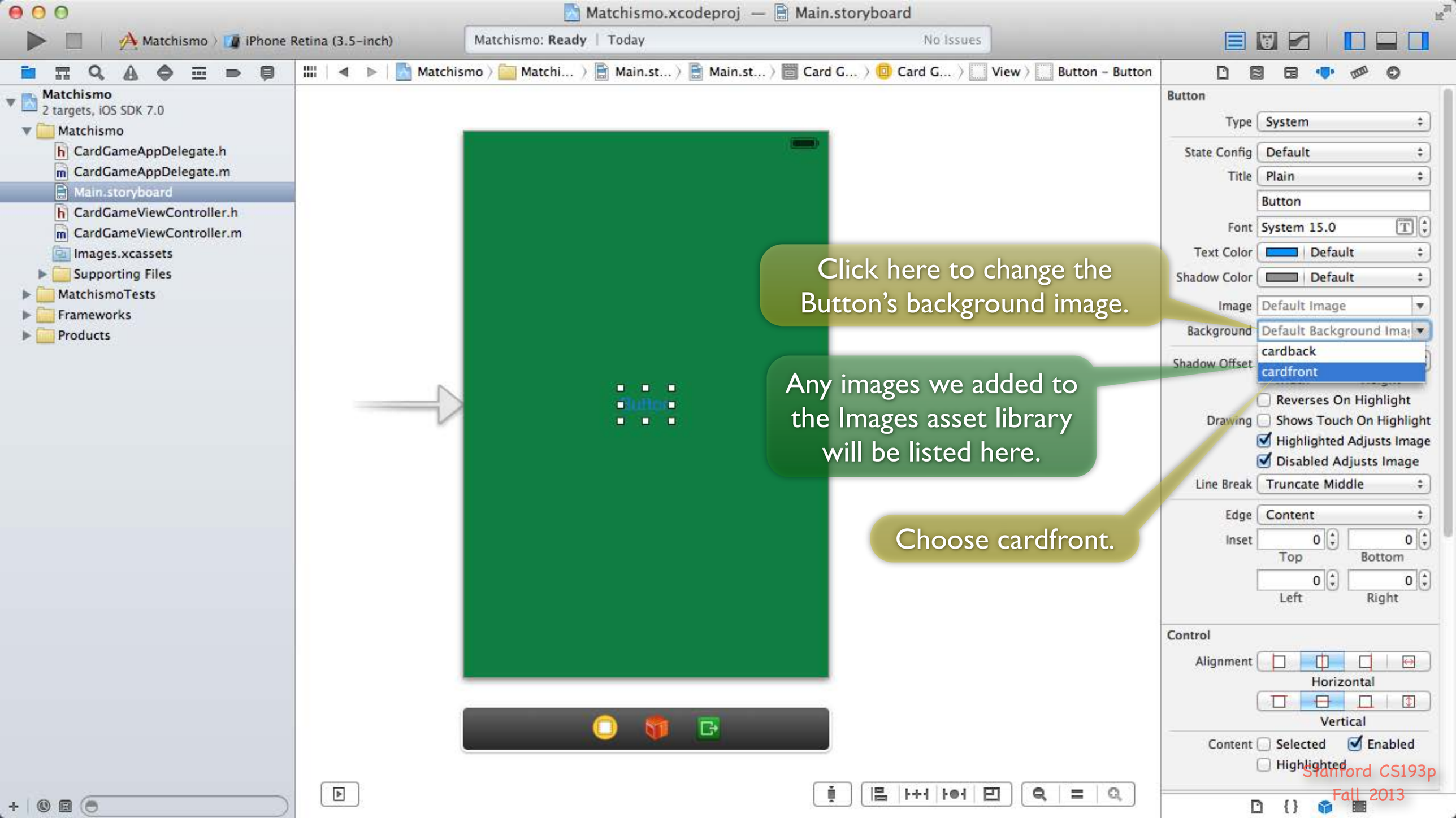

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 cardback
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
cardback
cardfront
LaunchImage
... cardback
cardback
1x
2x
Universal
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Status No changes
Notice that there's a "1x" version and a "2x" version of all images.
The 1x version is for non-Retina devices and the 2x version is for Retina devices.
You should have both.
Show Slicing
Stanford CS193p
Fall 2013

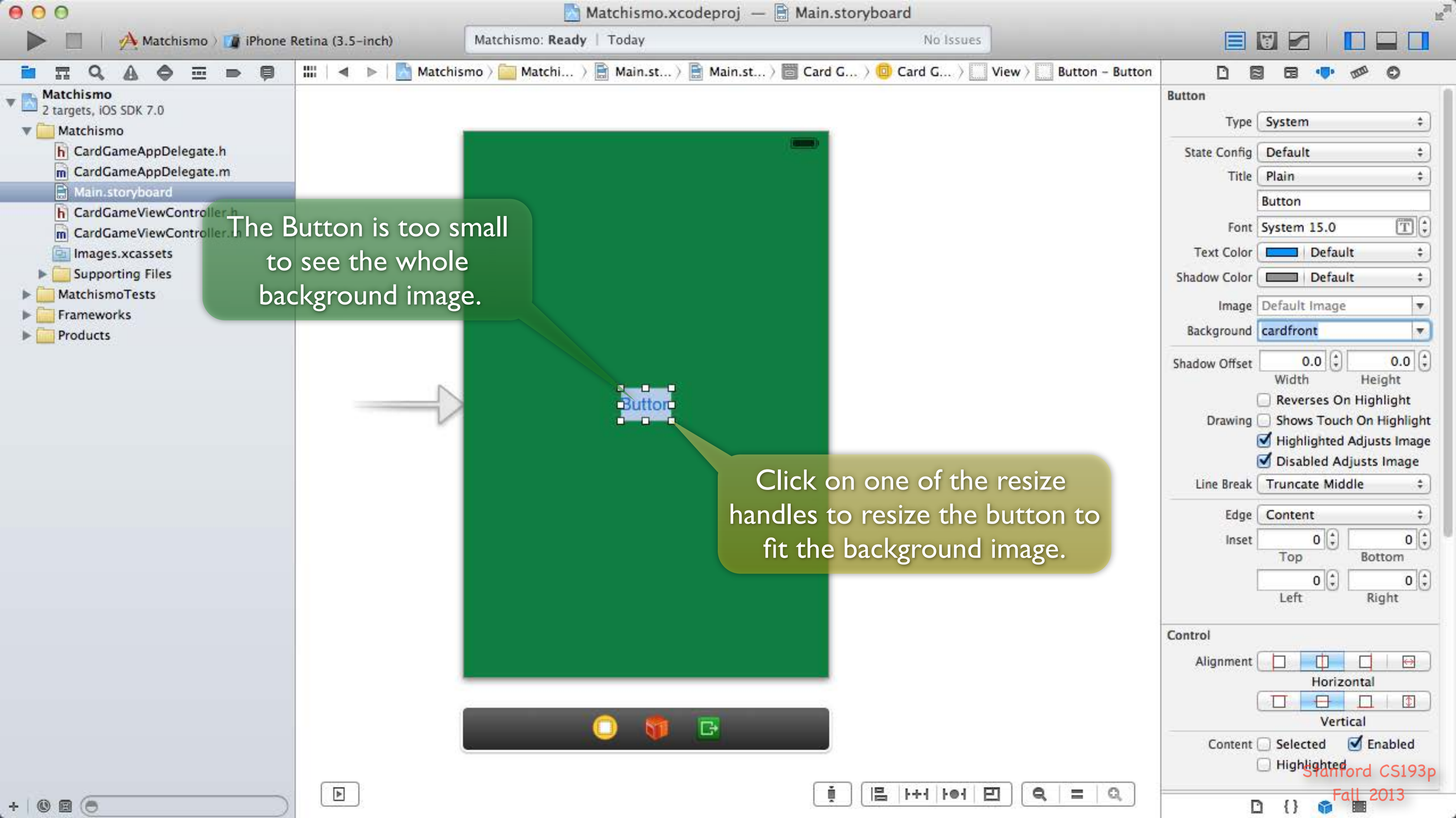

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 cardback
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
cardback
cardfront
LaunchImage
cardback
1x
2x
Universal
Identity and Type
Type Default - assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
Drag the higher resolution version into the 2x slot.
Items can also be deleted from this list by right clicking on them and choosing Remove.
Sometimes the Retina version might be able to show more detail.
Showing the tree here is sort of an extreme example just to demonstrate this.
Images
Developer
Images
blank card hi-res
blank card rounded corner
stanford
stanford tree
Applications
Downloads
Stanford
Paul
CS193p
SHARED
DEVICES
Hi-res Stanford Logo.
Name stanford tree
Kind Portable Network Graphics...
Size 12 KB
Created Today 2:42 PM
Modified Today 2:49 PM
Last opened Today 4:45 PM
Dimensions 128 × 192
OSX Lion ▸ Users ▸ CS193p ▸ Images ▸ stanford tree
1 of 4 selected, 221.65 GB available
These buttons also can be used to add/remove images.
Show Slicing
Stanford CS193p
Fall 2013

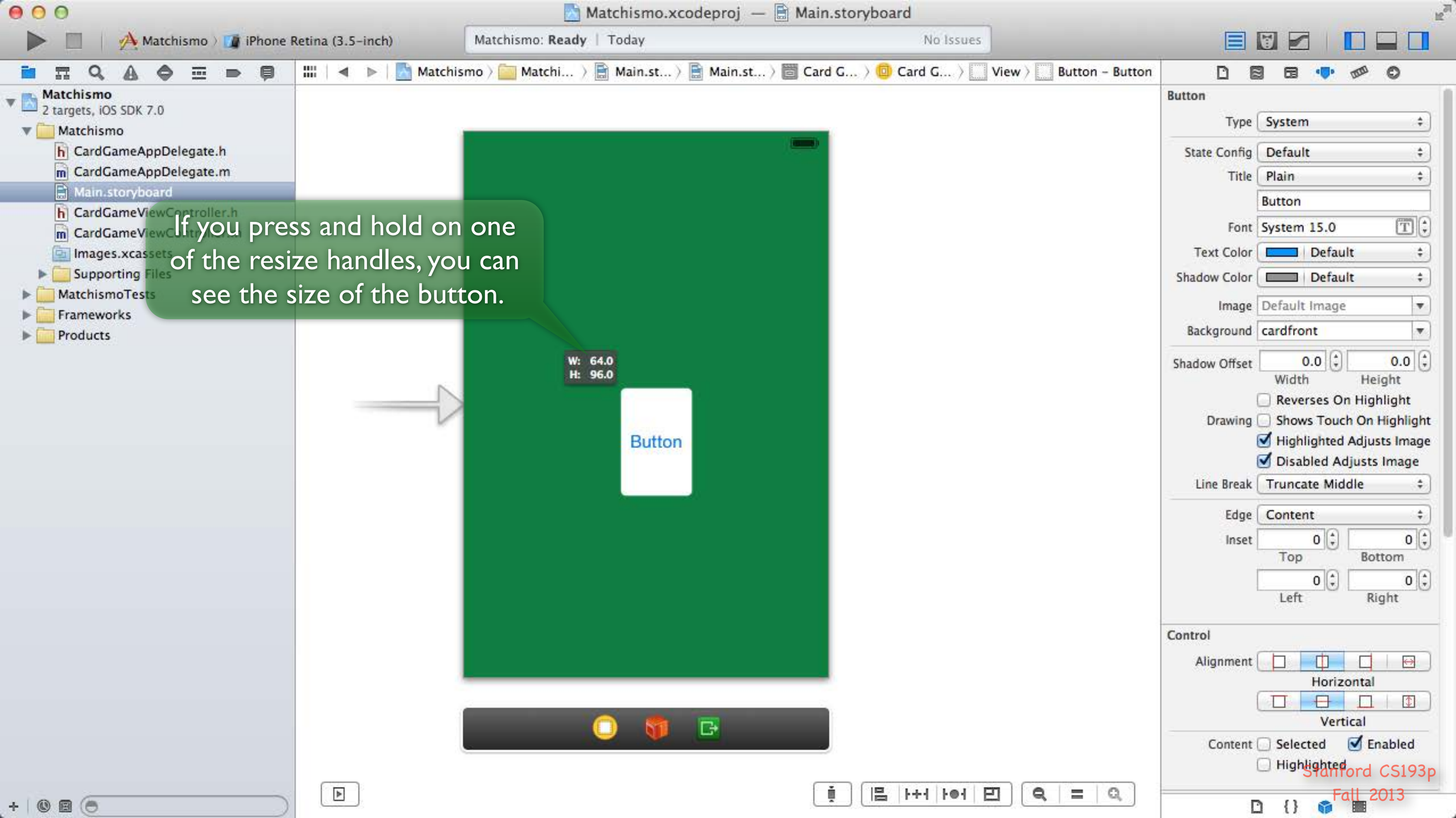

Matchismo.xcodeproj — Images.xcassets
Matchismo
iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Images.xcassets › cardback
AppIcon
cardback
cardfront
LaunchImage
cardback
1x
2x
Universal
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
Now you can see both versions.
Images
FAVORITES
All My Files
CS193p
AirDrop
Documents
Applications
Downloads
Stanford
Paul
CS193p
SHARED
DEVICES
Developer
Images
blank card hi-res
blank card rounded corner
stanford
stanford tree
Name stanford tree
Kind Portable Network Graphics...
Size 12 KB
Created Today 2:42 PM
Modified Today 2:49 PM
Last opened Today 4:45 PM
Dimensions 128 × 192
OSX Lion ▸ Users ▸ CS193p ▸ Images ▸ stanford tree
1 of 4 selected, 221.65 GB available
Show Slicing
Stanford CS193p
Fall 2013

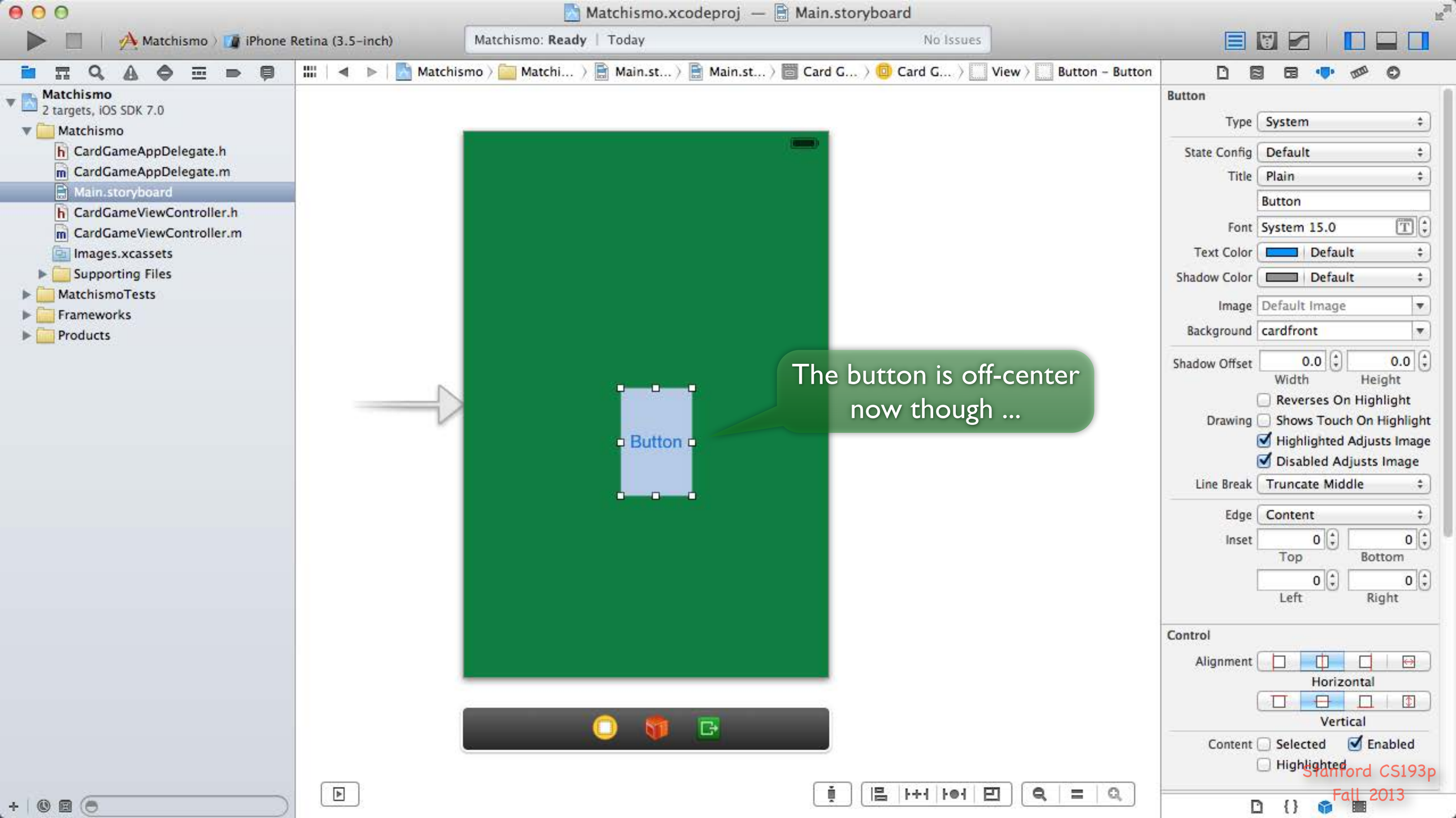

Matchismo.xcodeproj — Images.xcassets
Matchismo
iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Images.xcassets › cardfront
AppIcon
cardback
cardfront
LaunchImage
cardfront
blank card hi-res
1x
2x
Universal
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Let's do the same for the hi-res version of the background of the front of the card.
Images
FAVORITES
All My Files
CS193p
AirDrop
Documents
Applications
Downloads
Stanford
Paul
CS193p
SHARED
DEVICES
Developer
Images
blank card hi-res
blank card rounded corner
stanford
stanford tree
Name blank card hi-res
Kind Portable Network Graphics...
Size 1 KB
Created Today 2:42 PM
Modified Today 2:42 PM
Last opened Today 2:42 PM
Dimensions 128 × 192
OSX Lion ▸ Users ▸ CS193p ▸ Images ▸ blank card hi-res
1 of 4 selected, 221.65 GB available
Show Slicing
Stanford CS193p
Fall 2013

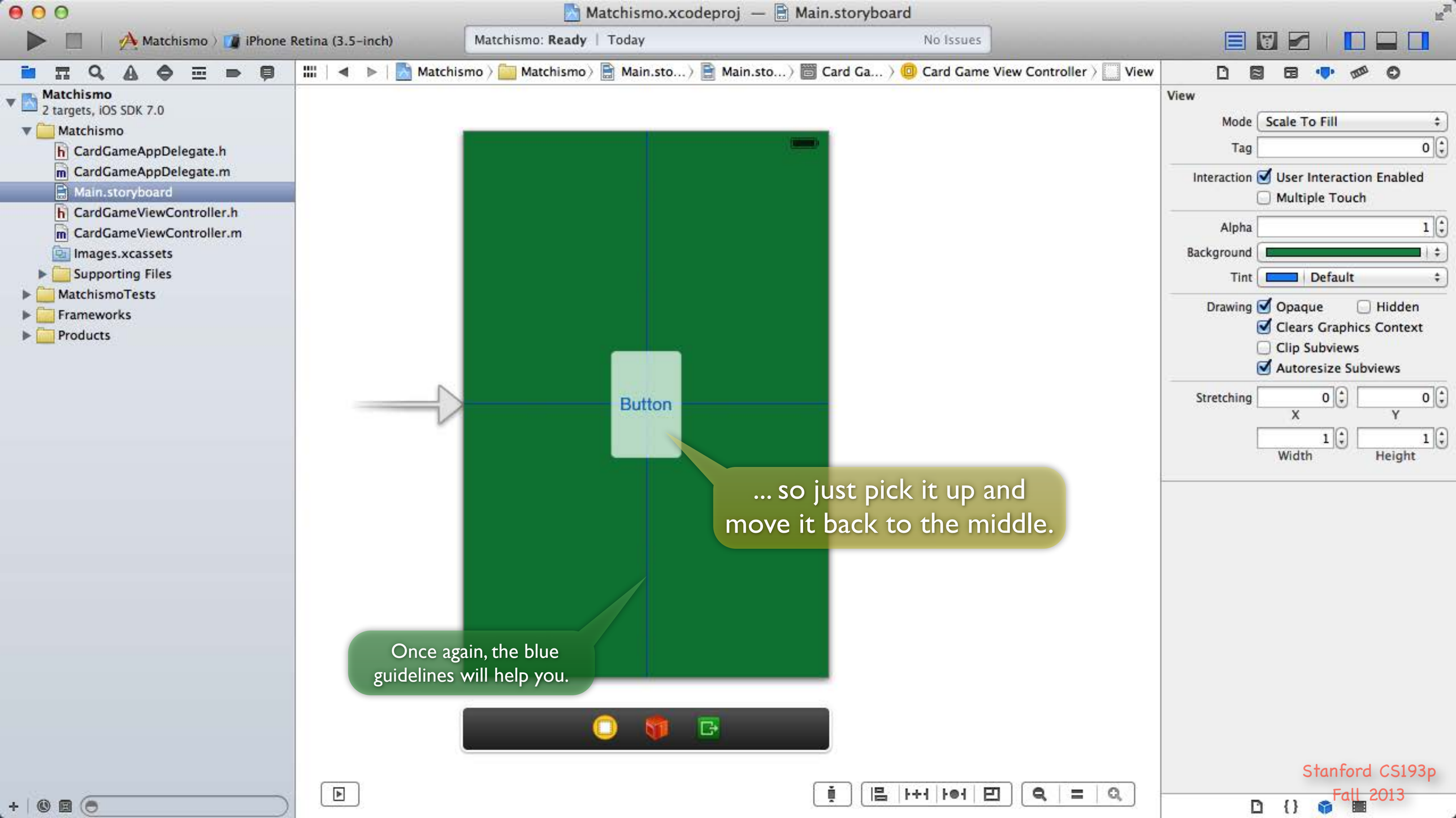

Matchismo.xcodeproj — Images.xcassets
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchismo 〉 Images.xcassets 〉 cardfront
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
AppIcon
cardback
cardfront
LaunchImage
cardfront
1x
2x
Universal
It's hard to see these two since they are blank, but they are white "rounded rectangles." You'll be able to see them better in the UI.
Identity and Type
Name Images.xcassets
Type Default – assetcatalog
Location Relative to Group
Images.xcassets
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/Images.xcassets
Target Membership
Matchismo
MatchismoTests
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Show Slicing
Stanford CS193p
Fall 2013

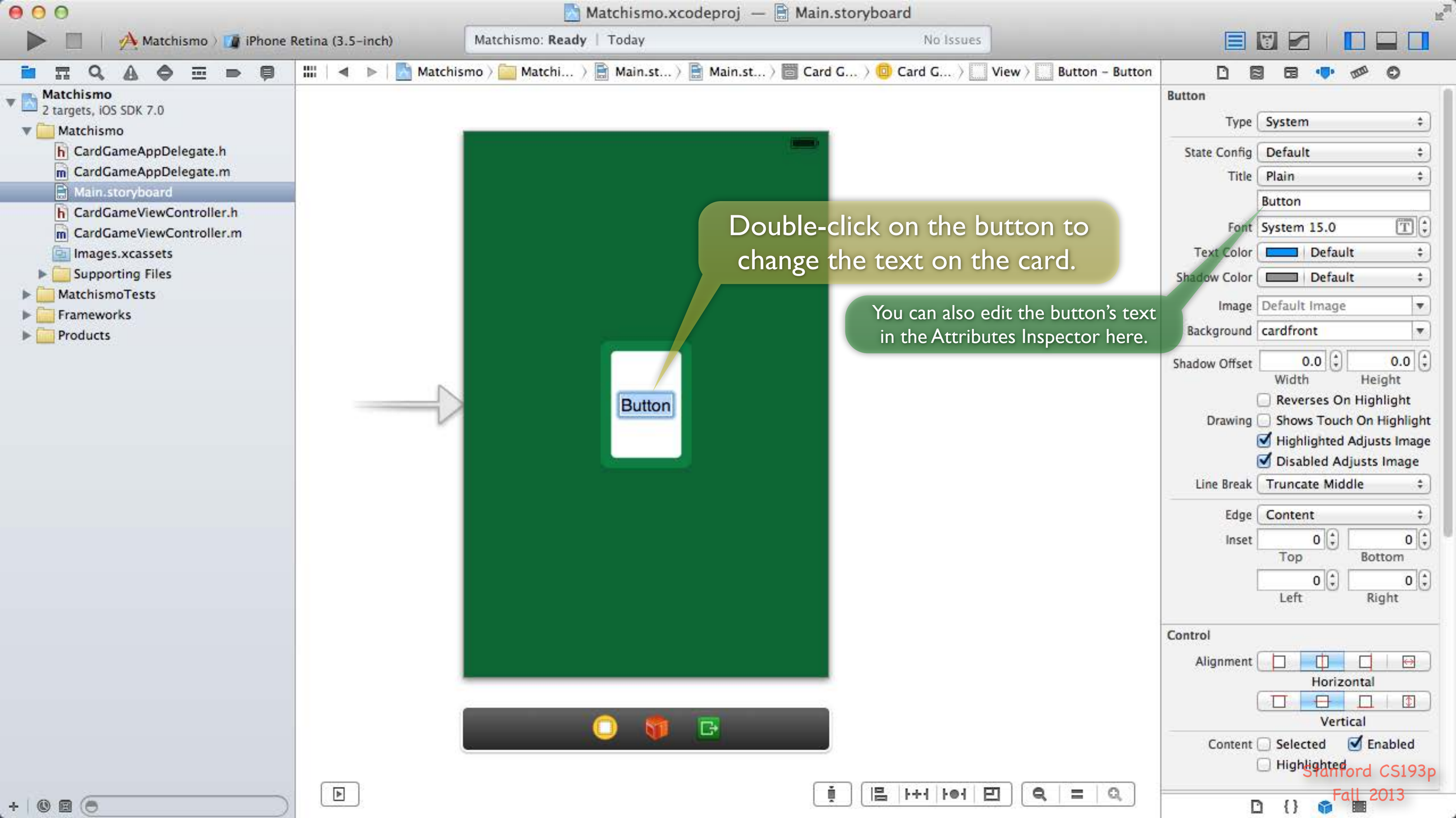

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchi... 〉 Main.st... 〉 Main.st... 〉 Card G... 〉 Card G... 〉 View 〉 Button – Button
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Ima
cardback
cardfront
Shadow Offset
Reverses On Highlight
Drawing 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
Click here to change the Button's background image.
Any images we added to the Images asset library will be listed here.
Choose cardfront.
Stanford CS193p
Fall 2013

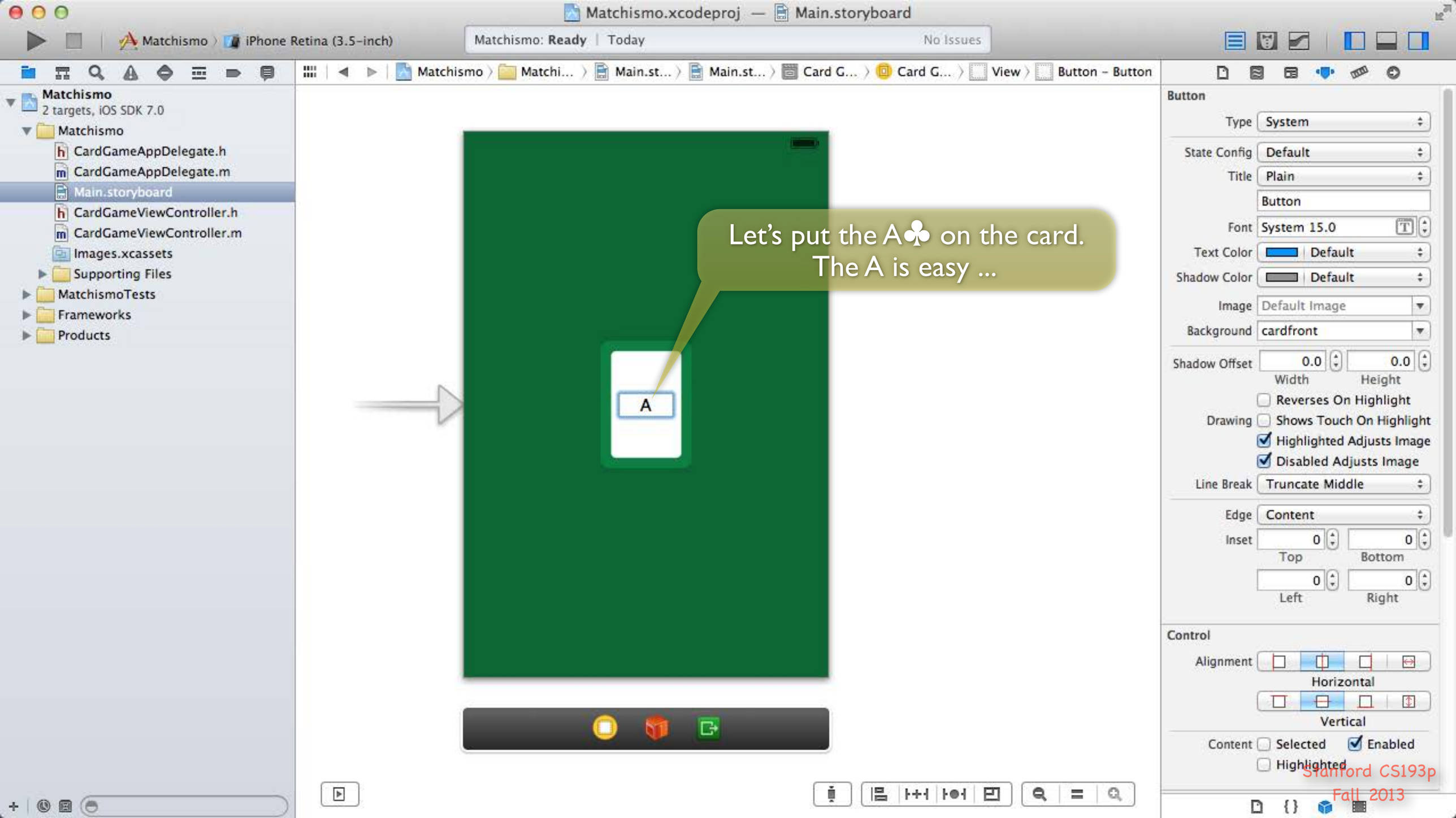

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchi... 〉 Main.st... 〉 Main.st... 〉 Card G... 〉 Card G... 〉 View 〉 Button – Button
The Button is too small to see the whole background image.
Button
Click on one of the resize handles to resize the button to fit the background image.
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

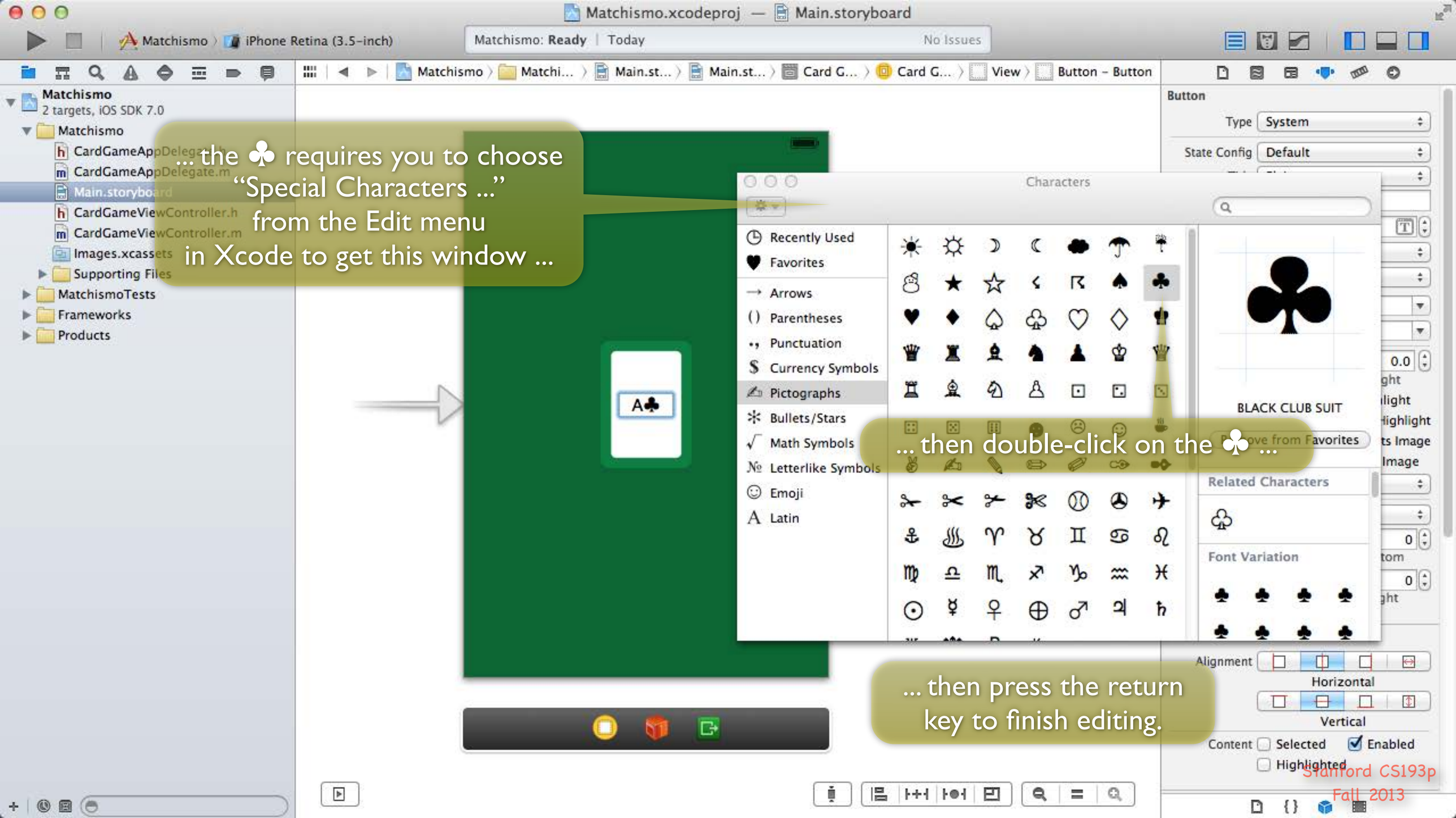

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
If you press and hold on one of the resize handles, you can see the size of the button.
W: 64.0
H: 96.0
Button
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

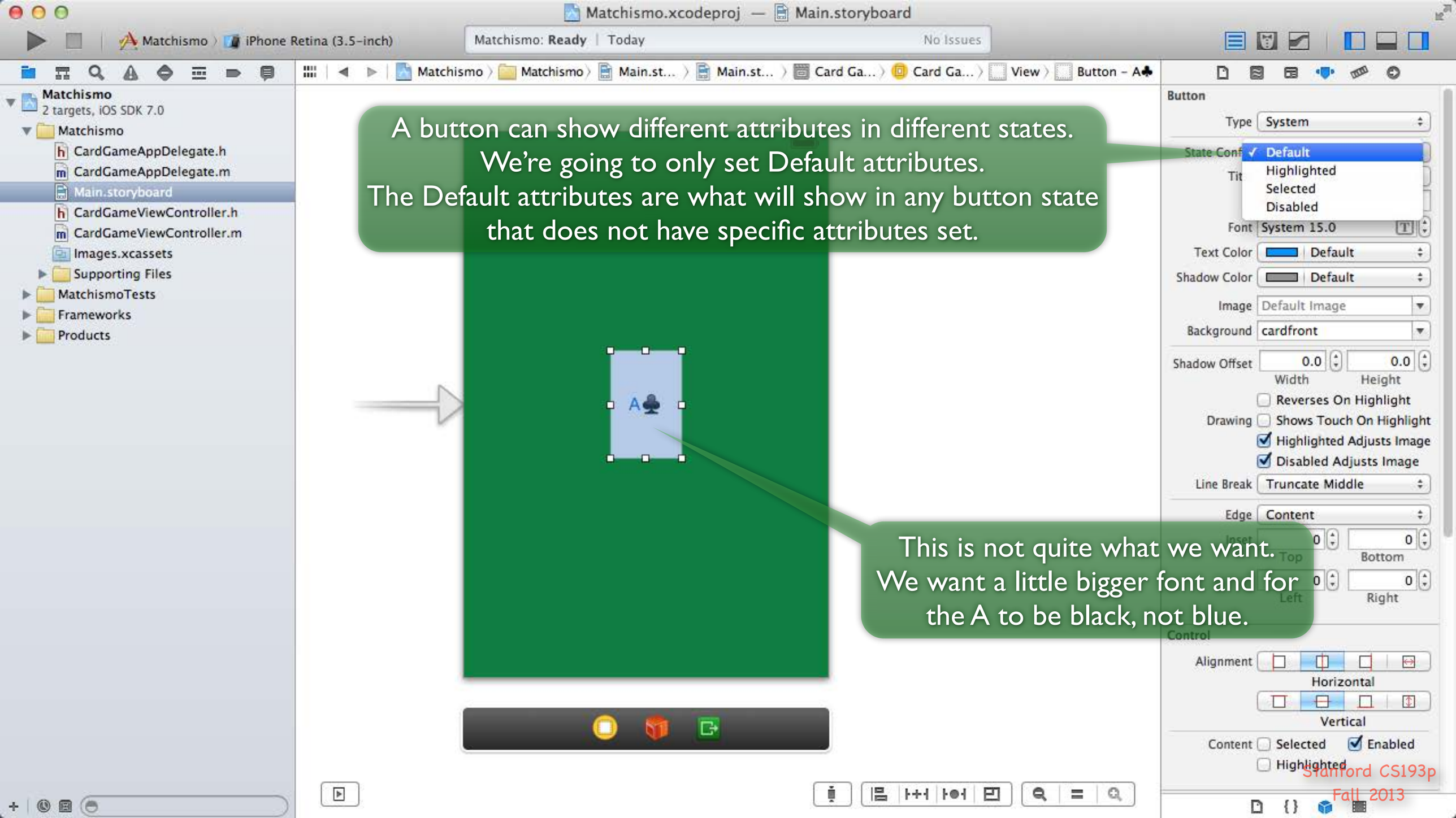

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchi... 〉 Main.st... 〉 Main.st... 〉 Card G... 〉 Card G... 〉 View 〉 Button – Button
Button
The button is off-center now though ...
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

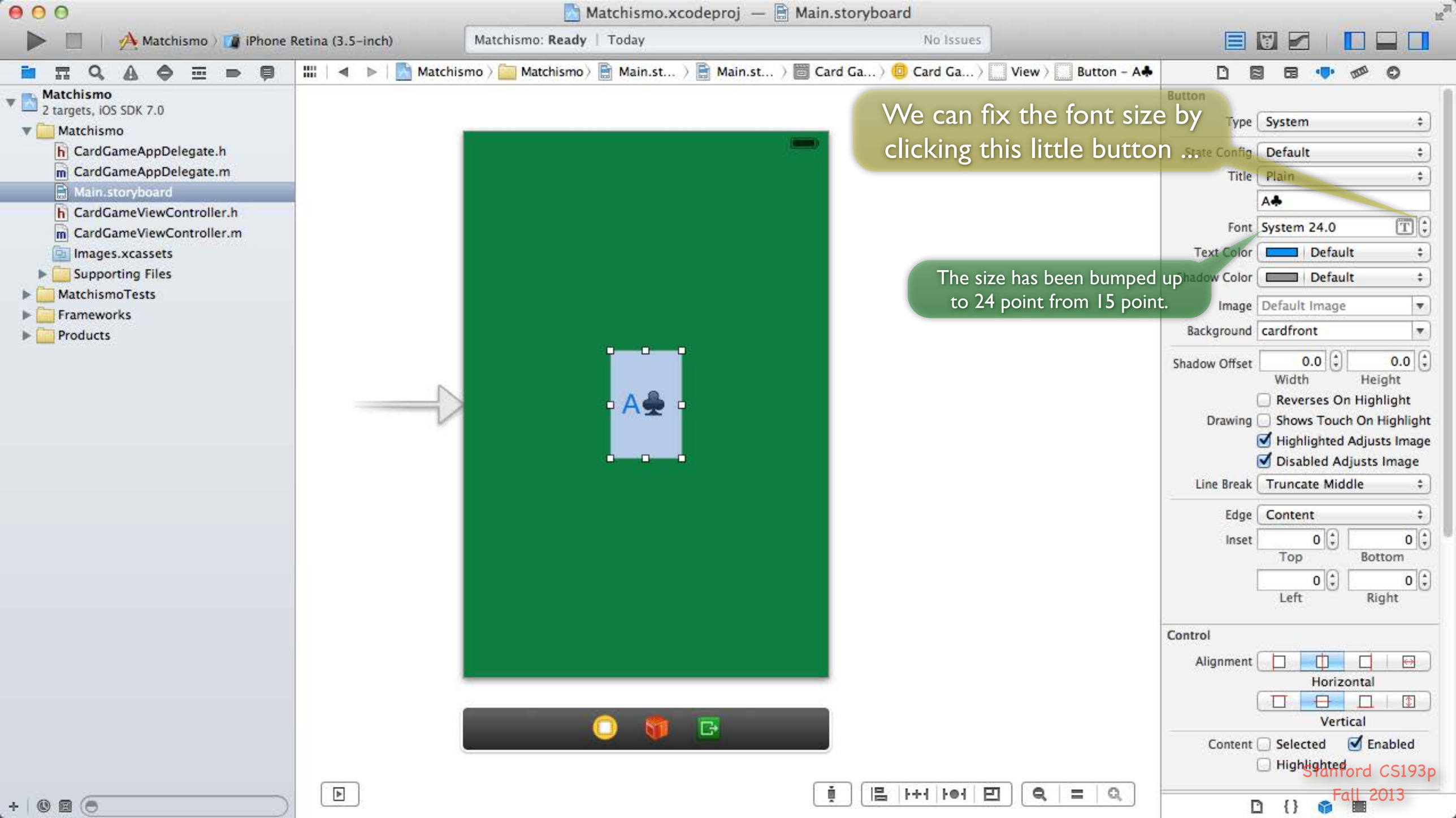

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Card Game View Controller
View
Button
... so just pick it up and move it back to the middle.
Once again, the blue guidelines will help you.
Mode
Scale To Fill
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Width
Height
Stanford CS193p
Fall 2013

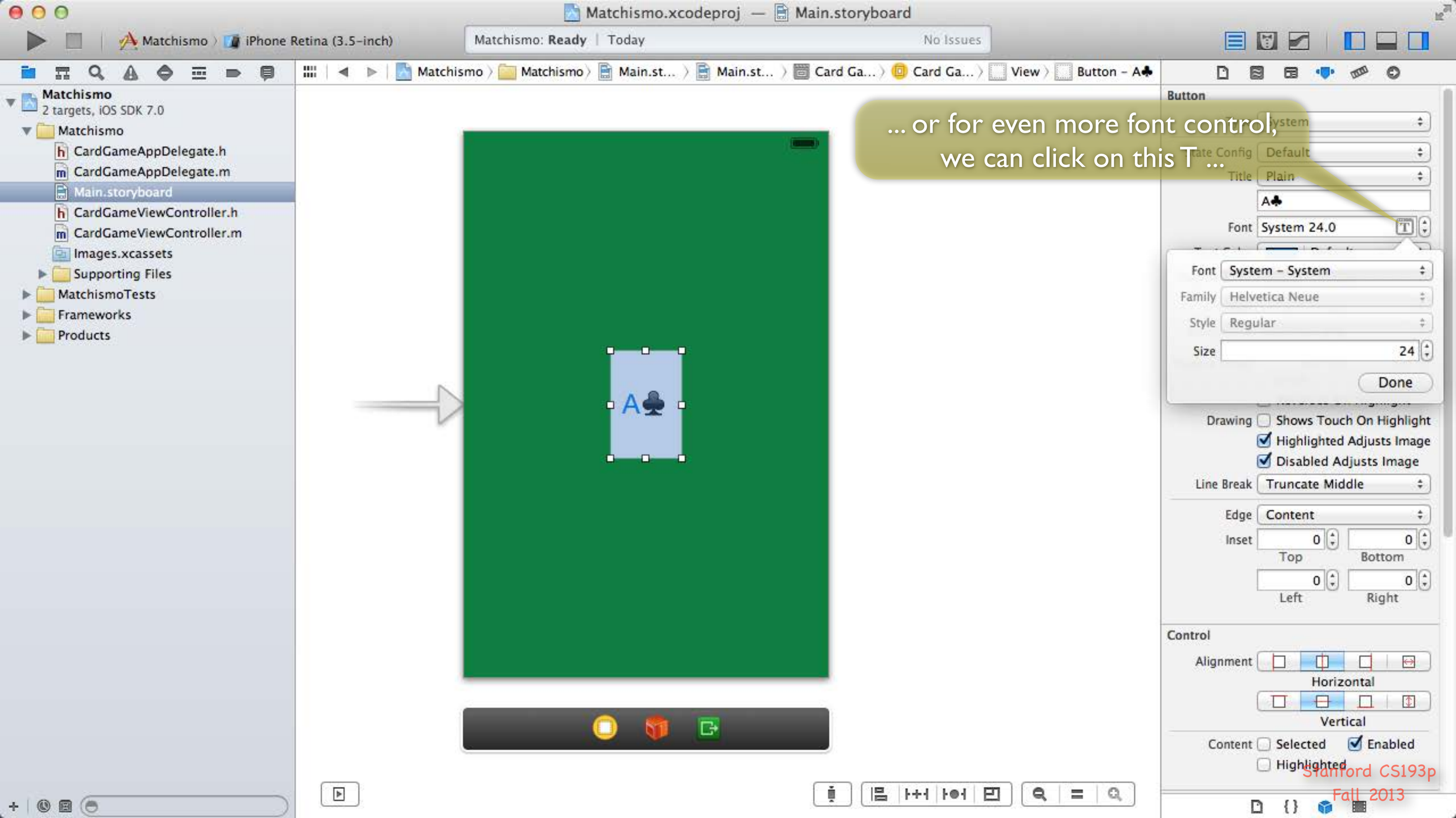

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchi... 〉 Main.st... 〉 Main.st... 〉 Card G... 〉 Card G... 〉 View 〉 Button – Button
Button
Double-click on the button to change the text on the card.
You can also edit the button's text in the Attributes Inspector here.
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

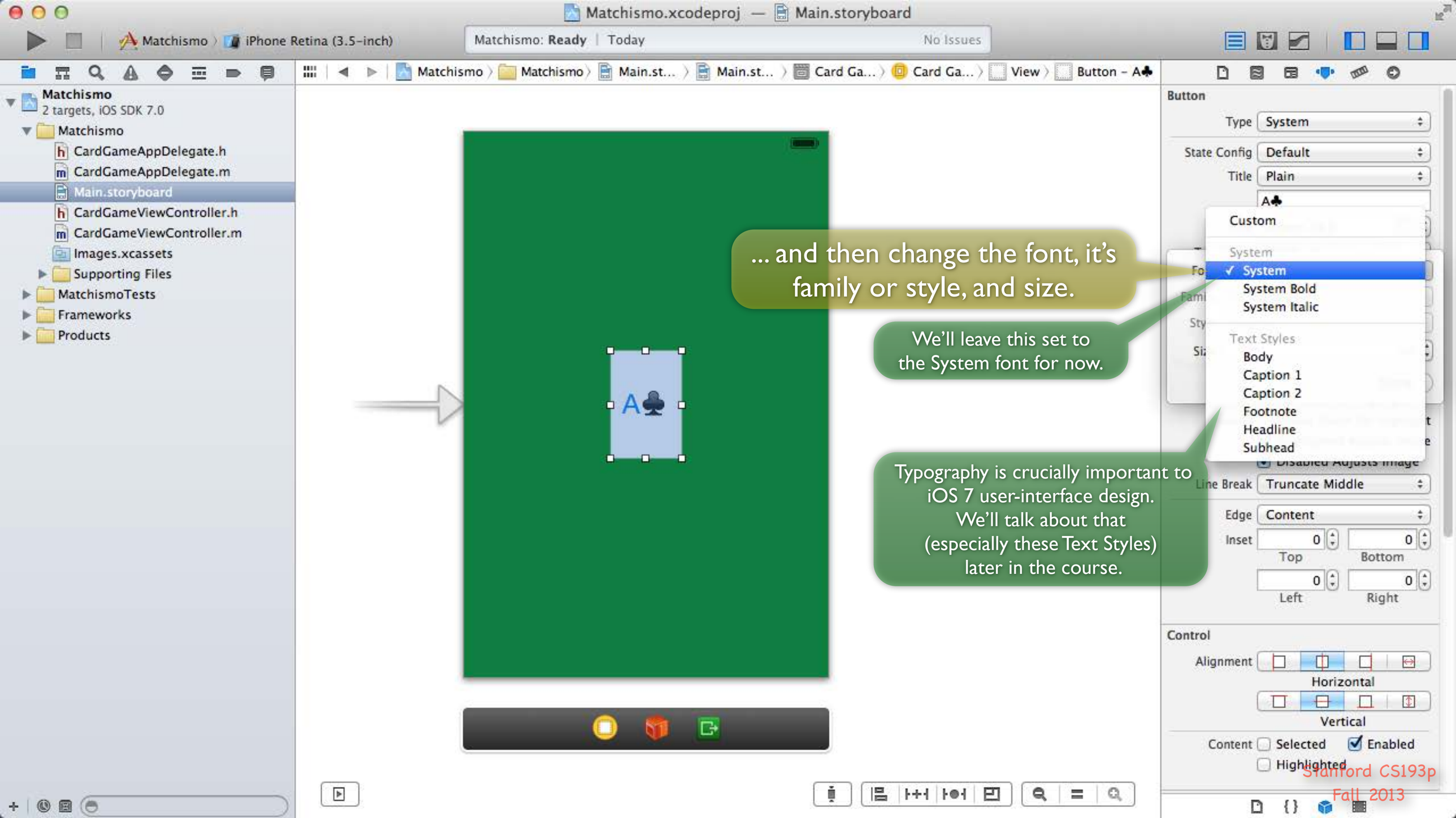

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo 〉 Matchi... 〉 Main.st... 〉 Main.st... 〉 Card G... 〉 Card G... 〉 View 〉 Button – Button
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Let's put the A♣ on the card.
The A is easy ...
A
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

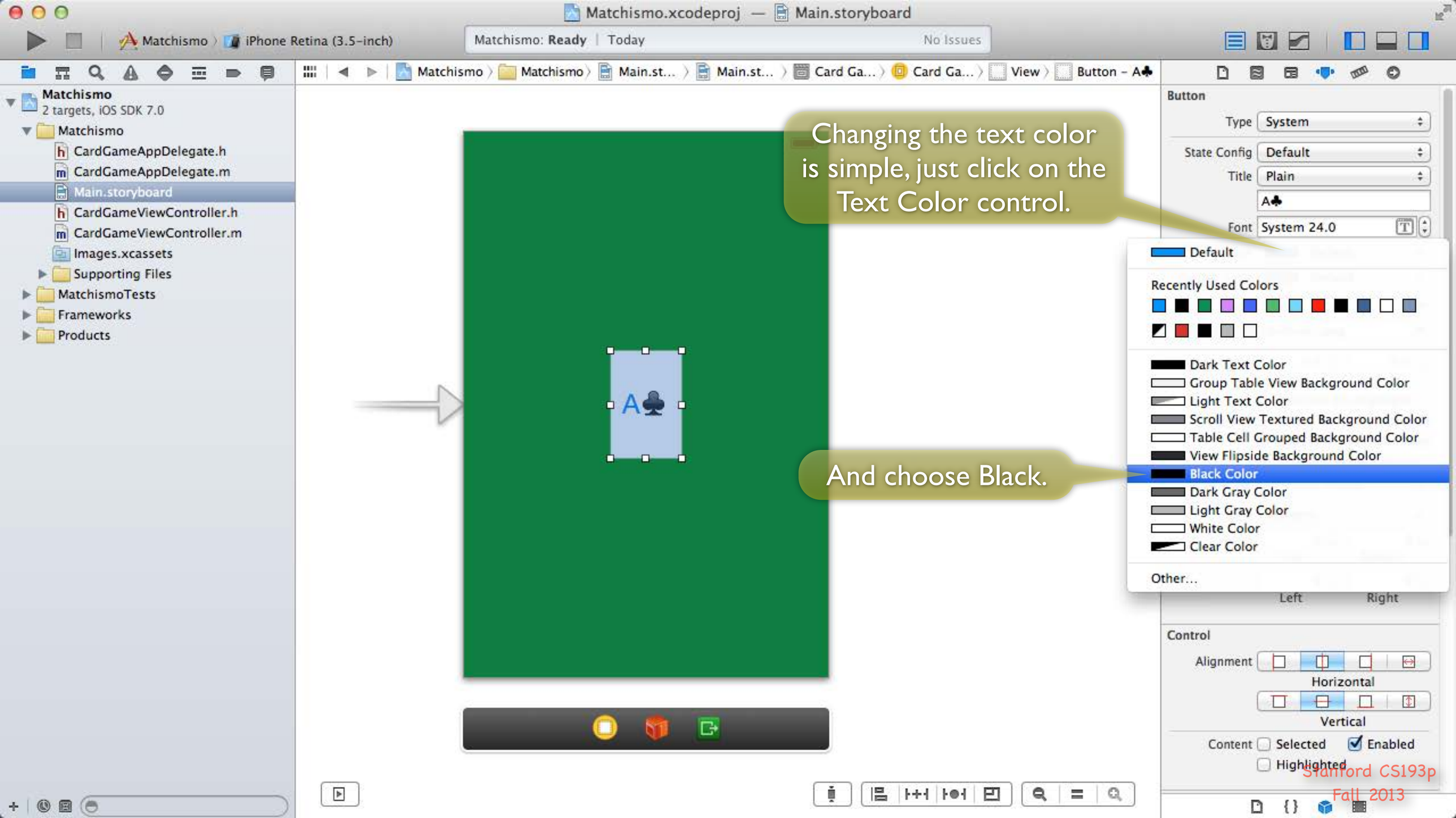

Matchismo.xcodeproj — Main.storyboard
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
... the ♣ requires you to choose "Special Characters ..." from the Edit menu in Xcode to get this window ...
Characters
Recently Used
Favorites
Arrows
Parentheses
Punctuation
Currency Symbols
Pictographs
Bullets/Stars
Math Symbols
Letterlike Symbols
Emoji
Latin
BLACK CLUB SUIT
Related Characters
Font Variation
... then double-click on the ♣ ...
... then press the return key to finish editing.
Button
Type System
State Config Default
Alignment
Horizontal
Vertical
Content
Selected
Enabled
Highlighted
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.st... 〉 Main.st... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – A♣
A button can show different attributes in different states.
We're going to only set Default attributes.
The Default attributes are what will show in any button state
that does not have specific attributes set.
A♣
This is not quite what we want.
We want a little bigger font and for
the A to be black, not blue.
Button
Type System
State Conf
Default
Highlighted
Selected
Disabled
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Top Bottom
Left Right
Control
Alignment
Horizontal
Vertical
Content Selected Enabled
Highlighted
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.st... 〉 Main.st... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – A♣
We can fix the font size by clicking this little button ...
The size has been bumped up to 24 point from 15 point.
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Default
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.st... 〉 Main.st... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – A♣
A♣
... or for even more font control, we can click on this T ...
Button
Default
Plain
A♣
Font System 24.0
Font System – System
Family Helvetica Neue
Style Regular
Size 24
Done
Drawing 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
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.st... 〉 Main.st... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – A♣
A♣
... and then change the font, it's family or style, and size.
We'll leave this set to the System font for now.
Typography is crucially important to iOS 7 user-interface design. We'll talk about that (especially these Text Styles) later in the course.
Button
Type System
State Config Default
Title Plain
A♣
Custom
System
System
System Bold
System Italic
Text Styles
Body
Caption 1
Caption 2
Footnote
Headline
Subhead
Line Break Truncate Middle
Edge Content
Inset 0 Top 0 Bottom 0 Left 0 Right
Control
Alignment
Horizontal
Vertical
Content Selected Enabled Highlighted
Stanford CS193p
Fall 2013

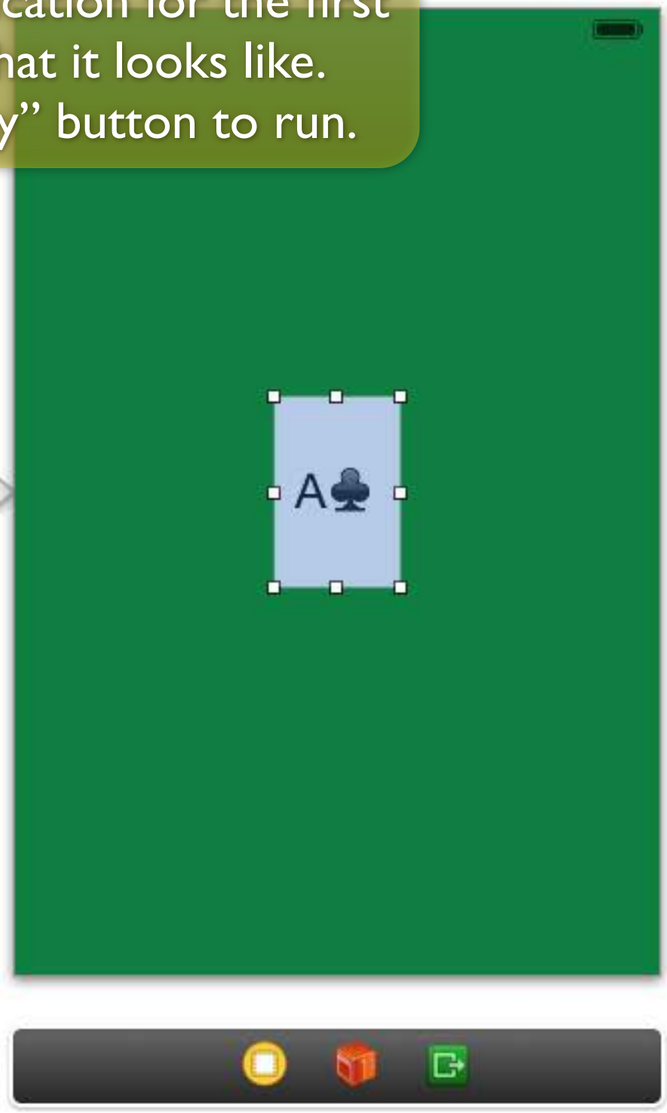

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Matchismo: Ready | Today
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Changing the text color is simple, just click on the Text Color control.
And choose Black.
Button
Type System
State Config Default
Title Plain
Font System 24.0
Default
Recently Used Colors
Dark Text Color
Group Table View Background Color
Light Text Color
Scroll View Textured Background Color
Table Cell Grouped Background Color
View Flipside Background Color
Black Color
Dark Gray Color
Light Gray Color
White Color
Clear Color
Other...
Control
Alignment
Horizontal
Vertical
Content Selected Enabled
Highlighted
Stanford CS193p
Fall 2013

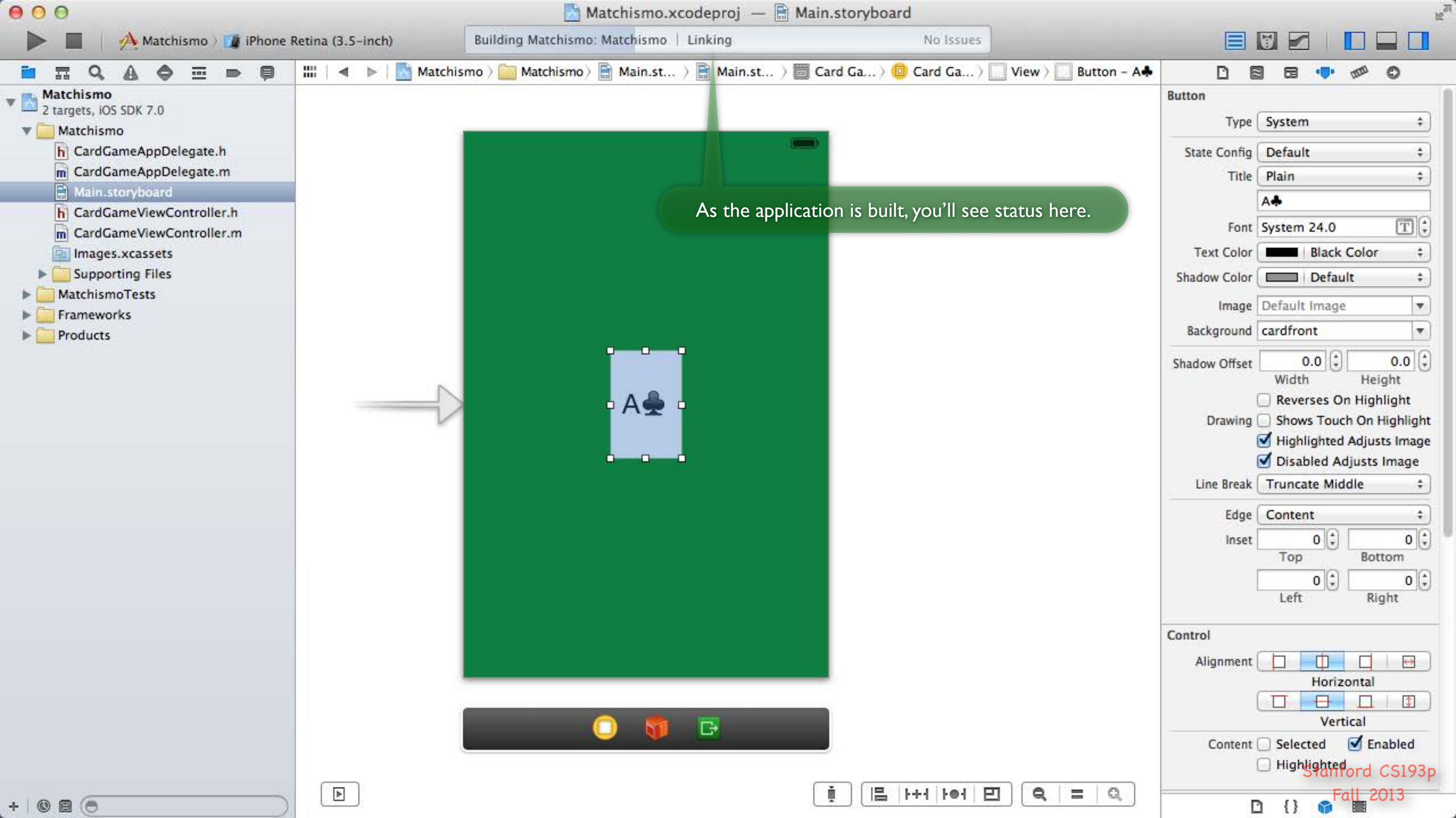
Our UI is built for 3.5-inch Retina iPhone, so you'll probably want to make sure that's set in this popup.

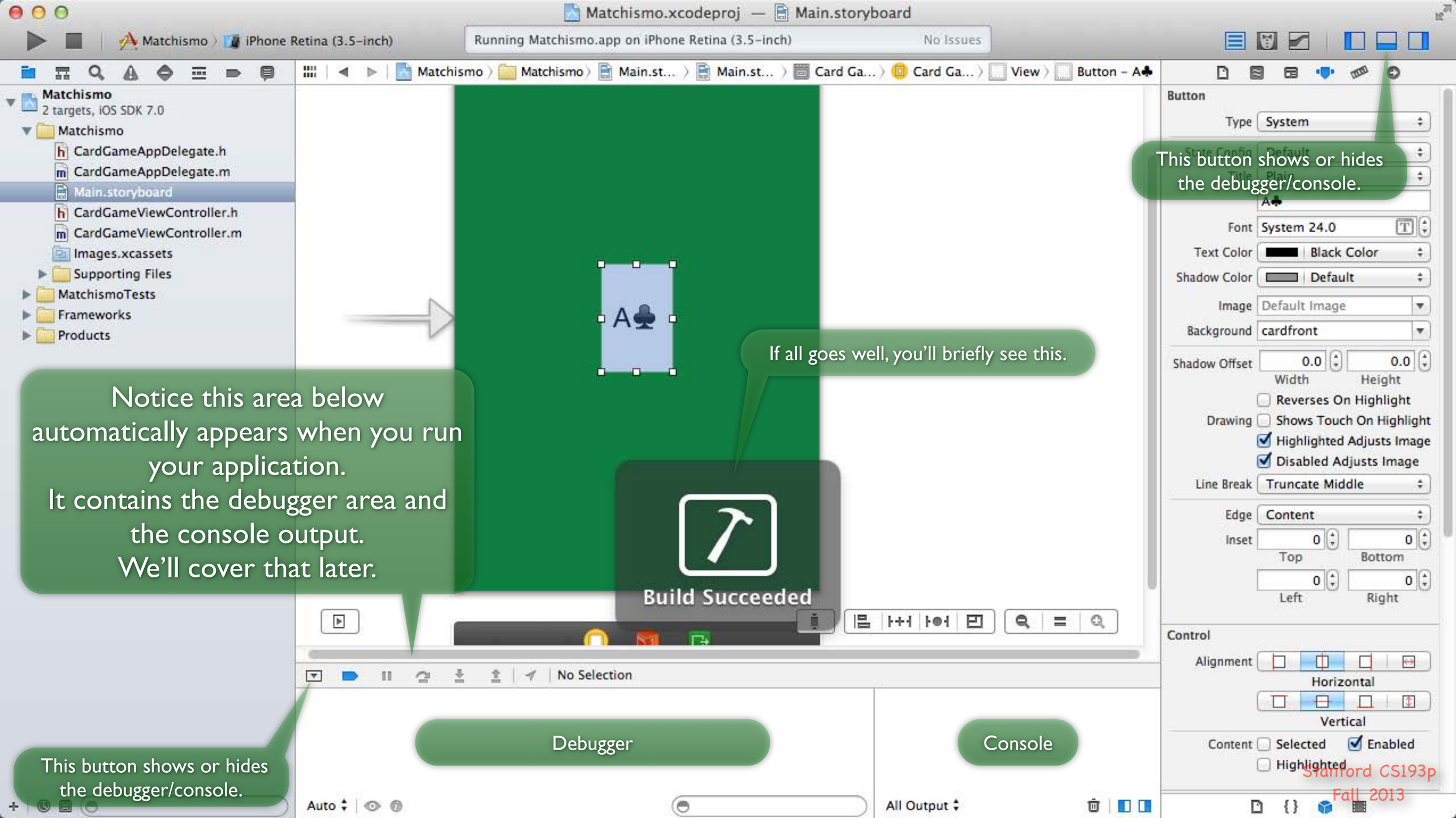
Let's Run our application for the first time and see what it looks like. Click on this "Play" button to run.

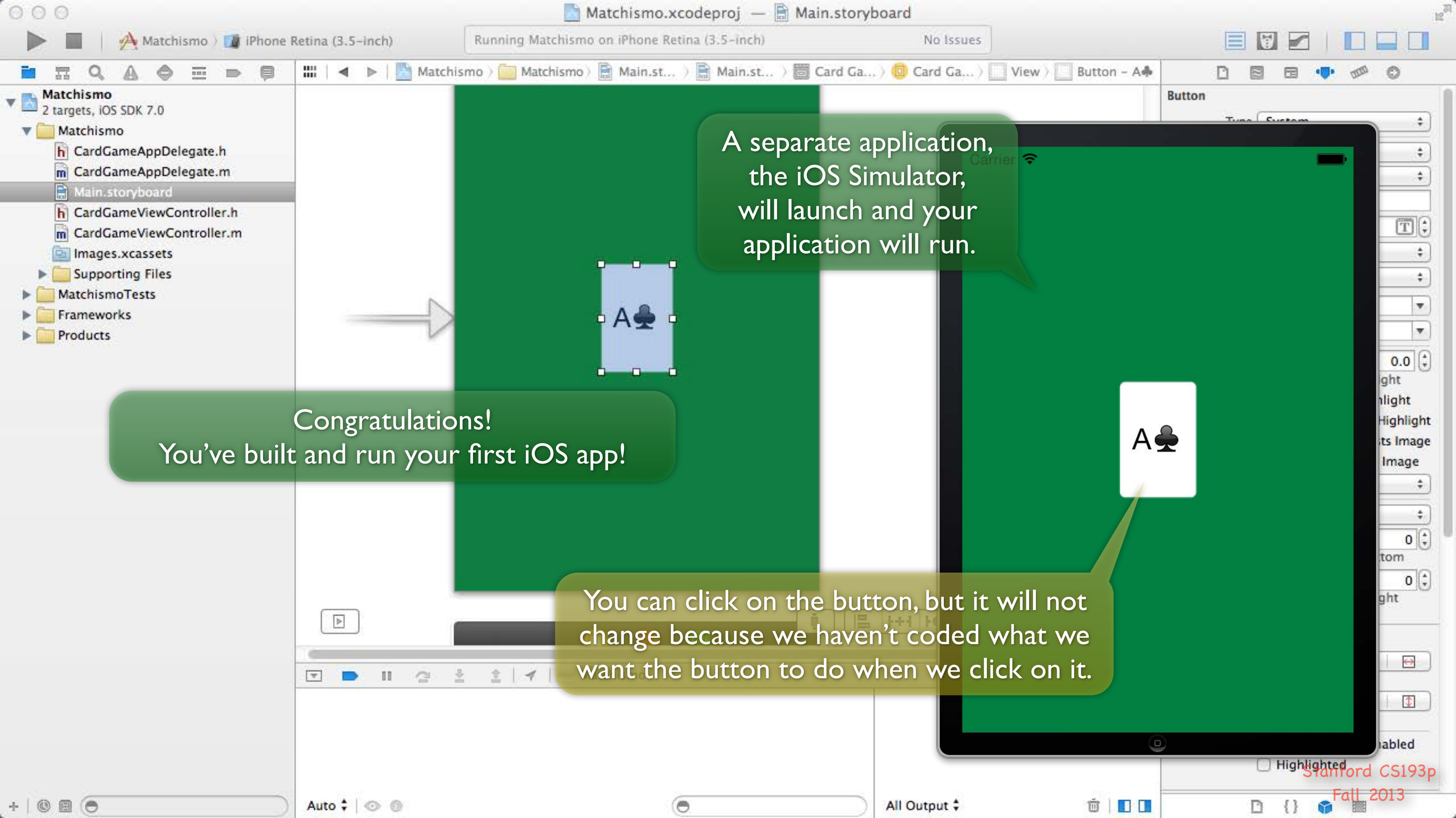
If you press and hold this Run button, other run options will be available, but we're just using plain "Run" for now.

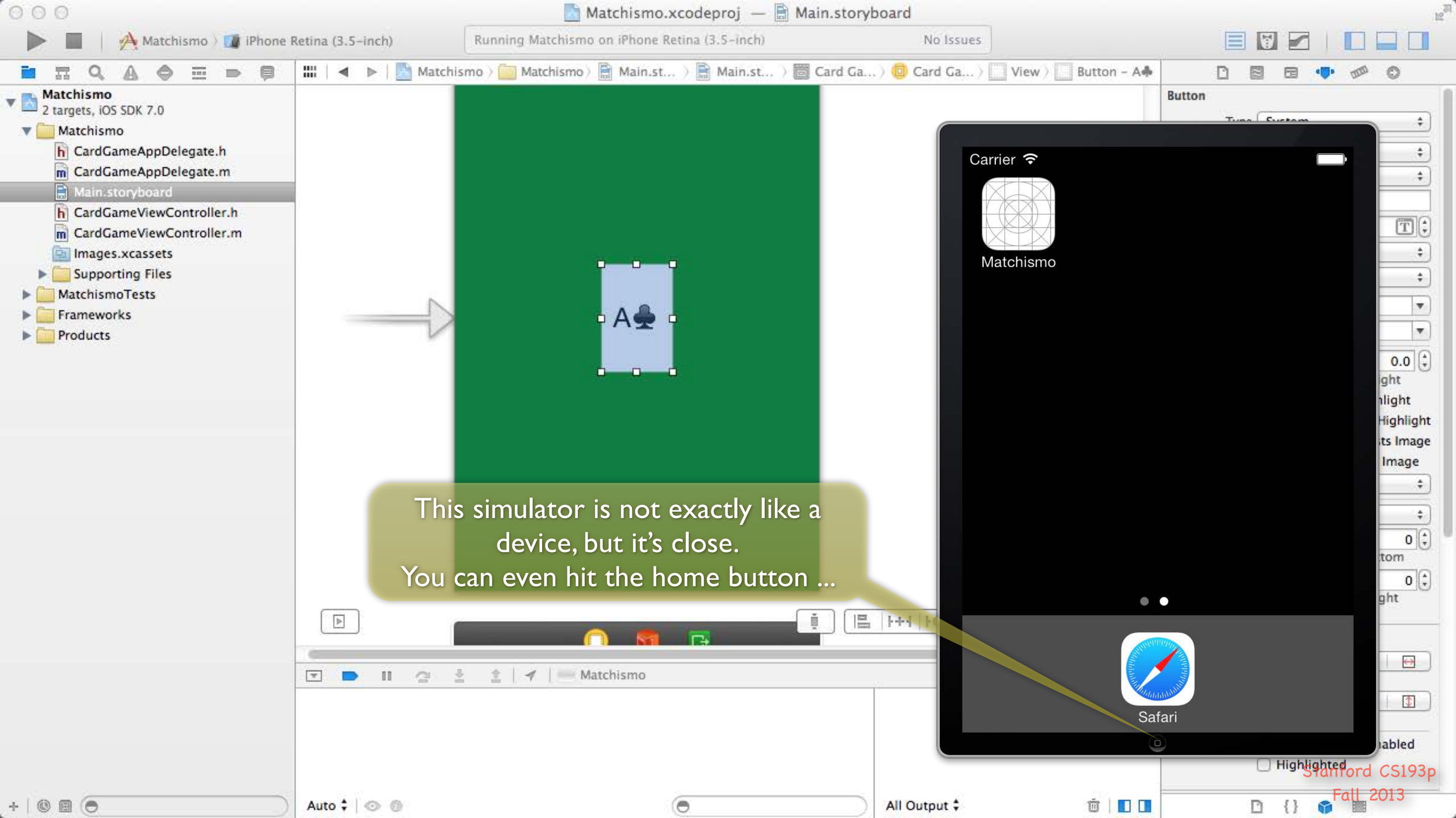
Color is now black.

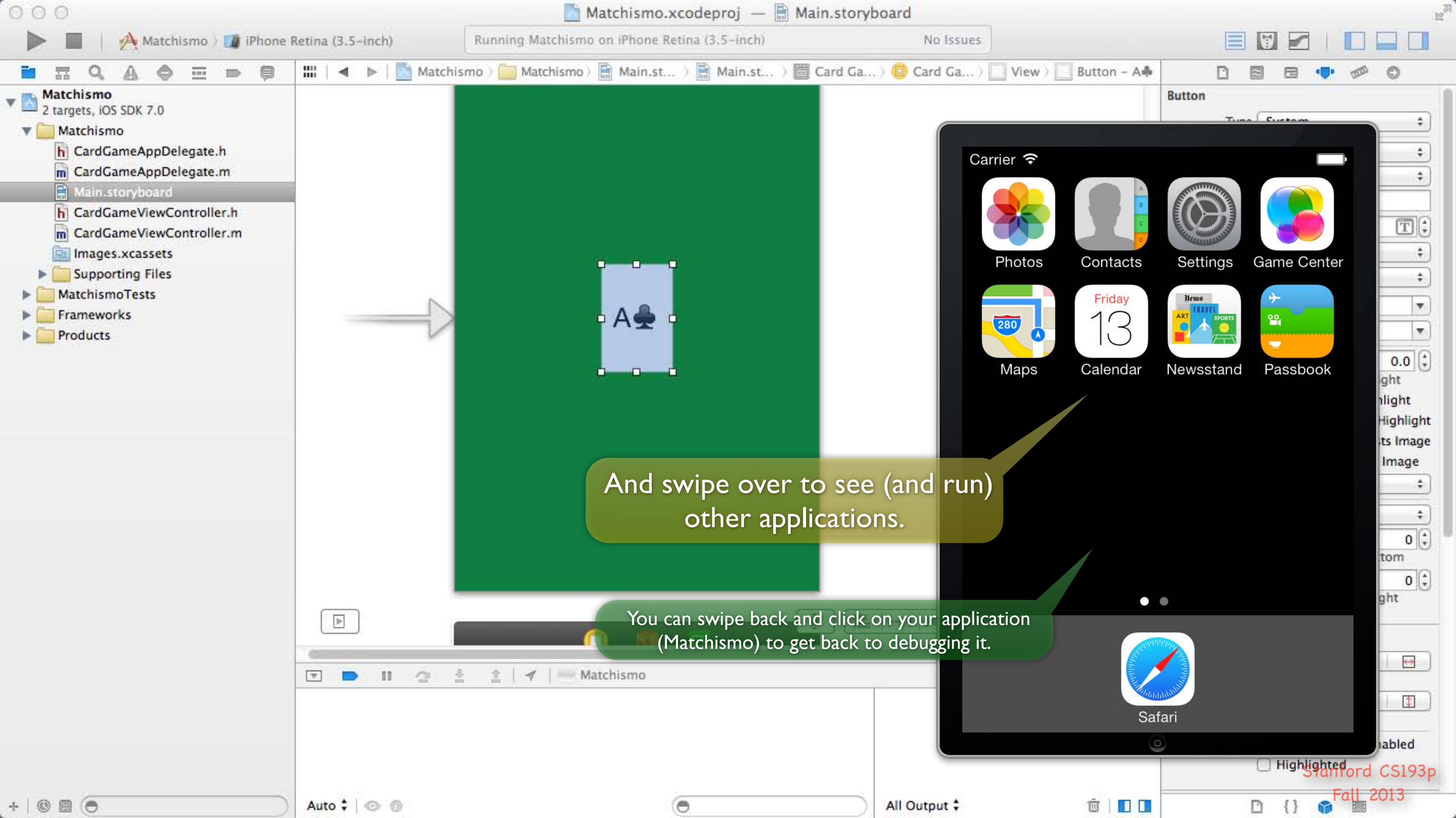


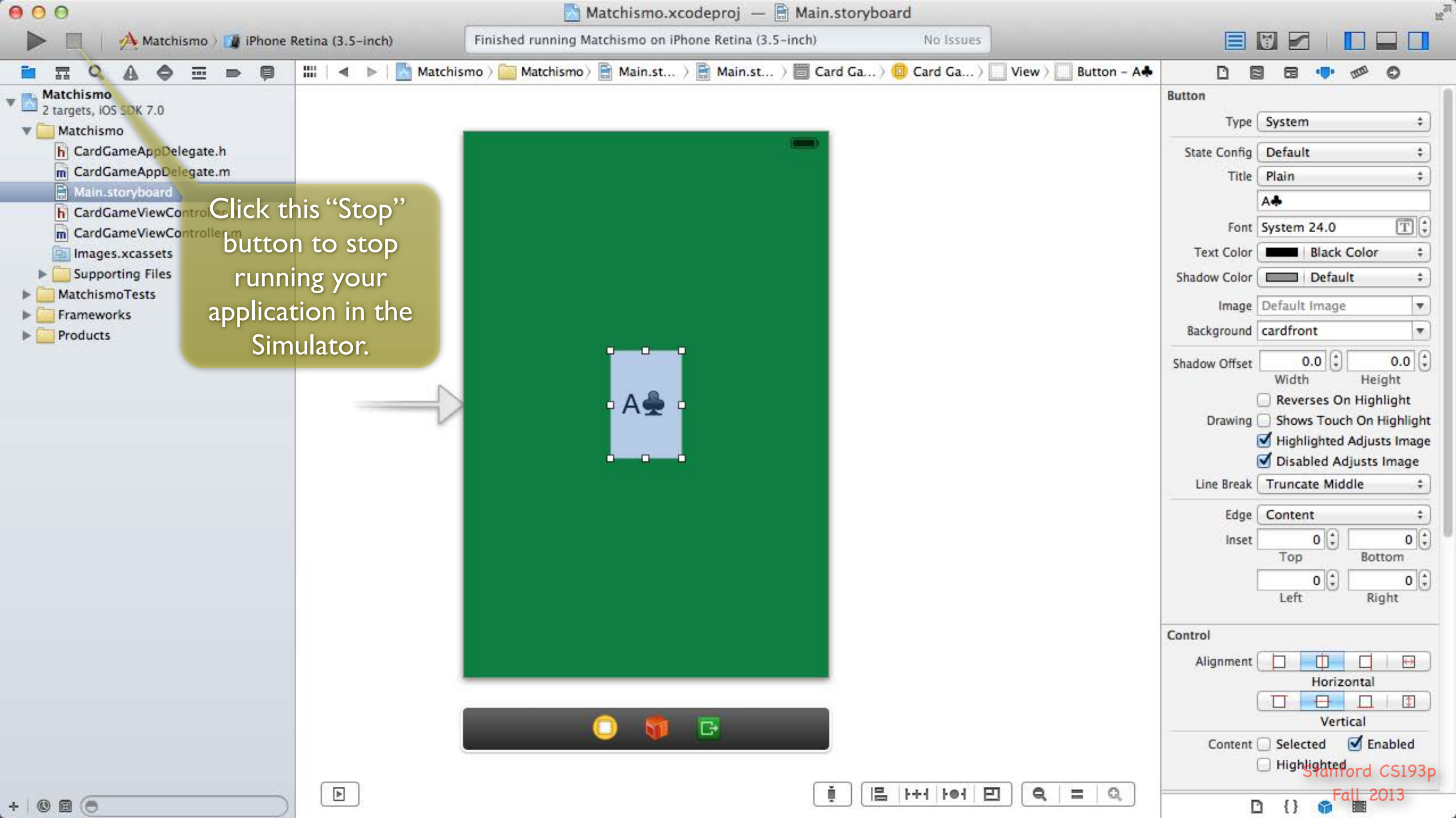

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Building Matchismo: Matchismo | Linking
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Main.st... › Main.st... › Card Ga... › Card Ga... › View › Button – A♣
As the application is built, you'll see status here.
A♣
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

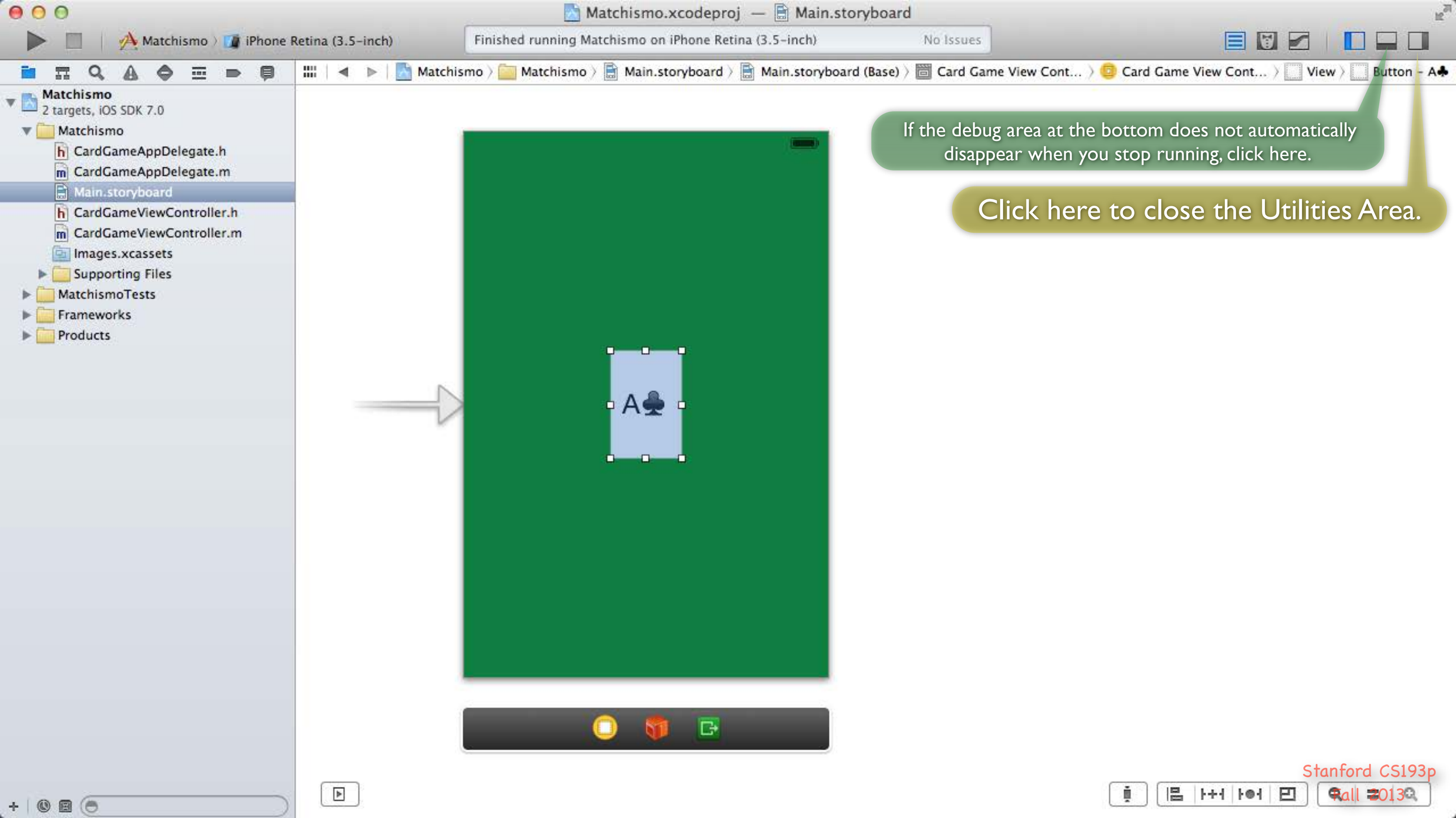

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Running Matchismo.app on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Main.st... 〉 Main.st... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – A♣
A♣
Build Succeeded
No Selection
Auto
All Output
Button
Type System
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
This button shows or hides the debugger/console.
If all goes well, you'll briefly see this.
Notice this area below automatically appears when you run your application.
It contains the debugger area and the console output.
We'll cover that later.
Debugger
Console
This button shows or hides the debugger/console.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo
iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
A♣
A separate application, the iOS Simulator, will launch and your application will run.
Congratulations!
You've built and run your first iOS app!
You can click on the button, but it will not change because we haven't coded what we want the button to do when we click on it.
Carrier
Button
Highlighted
Auto
All Output
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo
iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Button
Carrier
Matchismo
Safari
This simulator is not exactly like a device, but it's close.
You can even hit the home button ...
Highlighted
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Button
Carrier
Photos
Contacts
Settings
Game Center
Maps
Calendar
Friday
13
Newsstand
Passbook
Safari
And swipe over to see (and run) other applications.
You can swipe back and click on your application (Matchismo) to get back to debugging it.
Highlighted
Auto
All Output
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewControl
CardGameViewControlle m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Click this "Stop" button to stop running your application in the Simulator.
Matchismo › Matchismo › Main.st... › Main.st... › Card Ga... › Card Ga... › View › Button – A♣
A♣
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background cardfront
Shadow Offset 0.0 Width 0.0 Height
Reverses On Highlight
Drawing 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
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Main.storyboard › Main.storyboard (Base) › Card Game View Cont… › Card Game View Cont… › View › Button – A♣
A♣
If the debug area at the bottom does not automatically disappear when you stop running, click here.
Click here to close the Utilities Area.
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Click here to show the Assistant Editor.

When an MVC View is showing, the Assistant Editor will bring up the code for the corresponding MVC Controller.
That's exactly what we want.

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

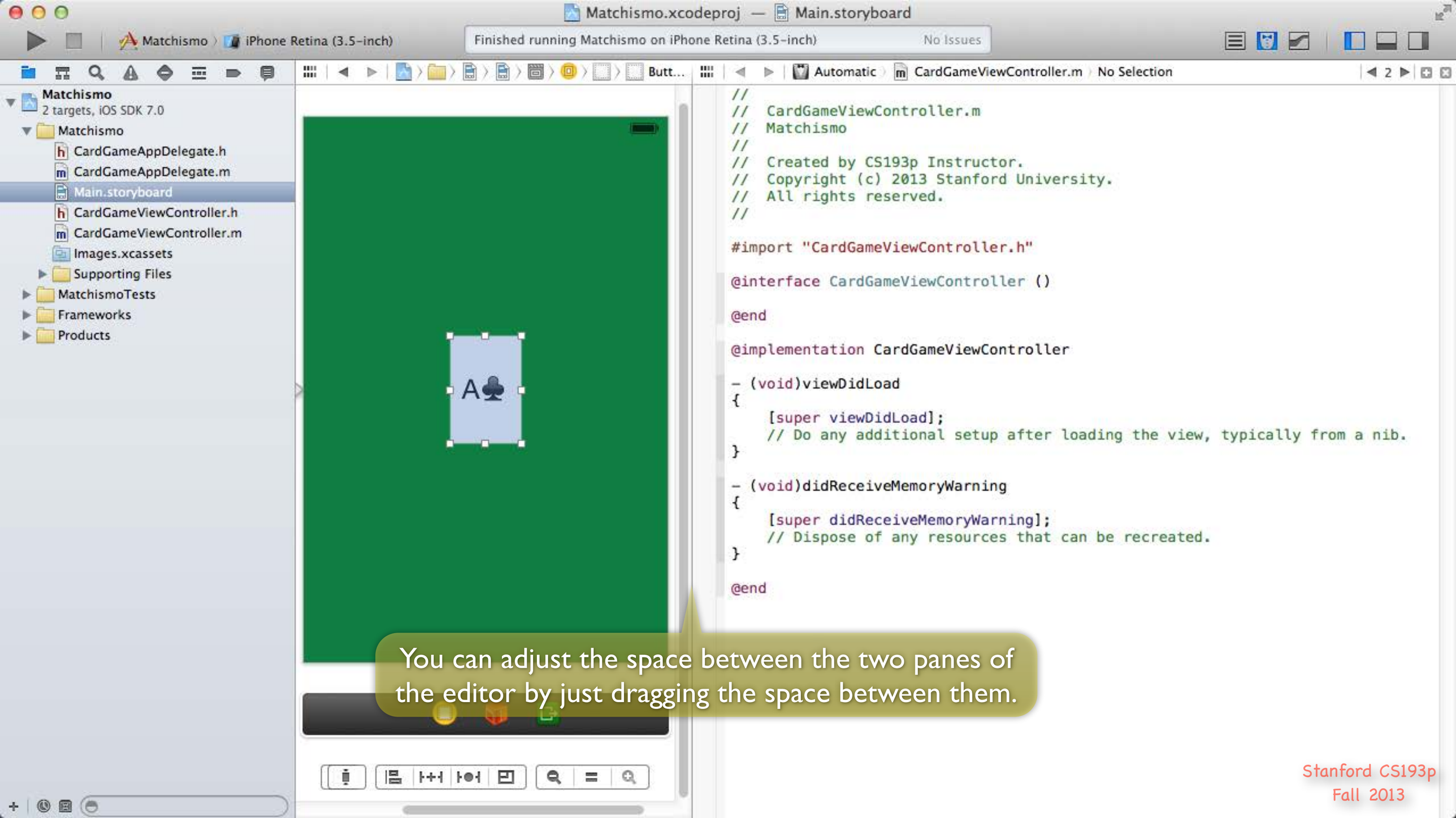
@implementation CardGameViewController

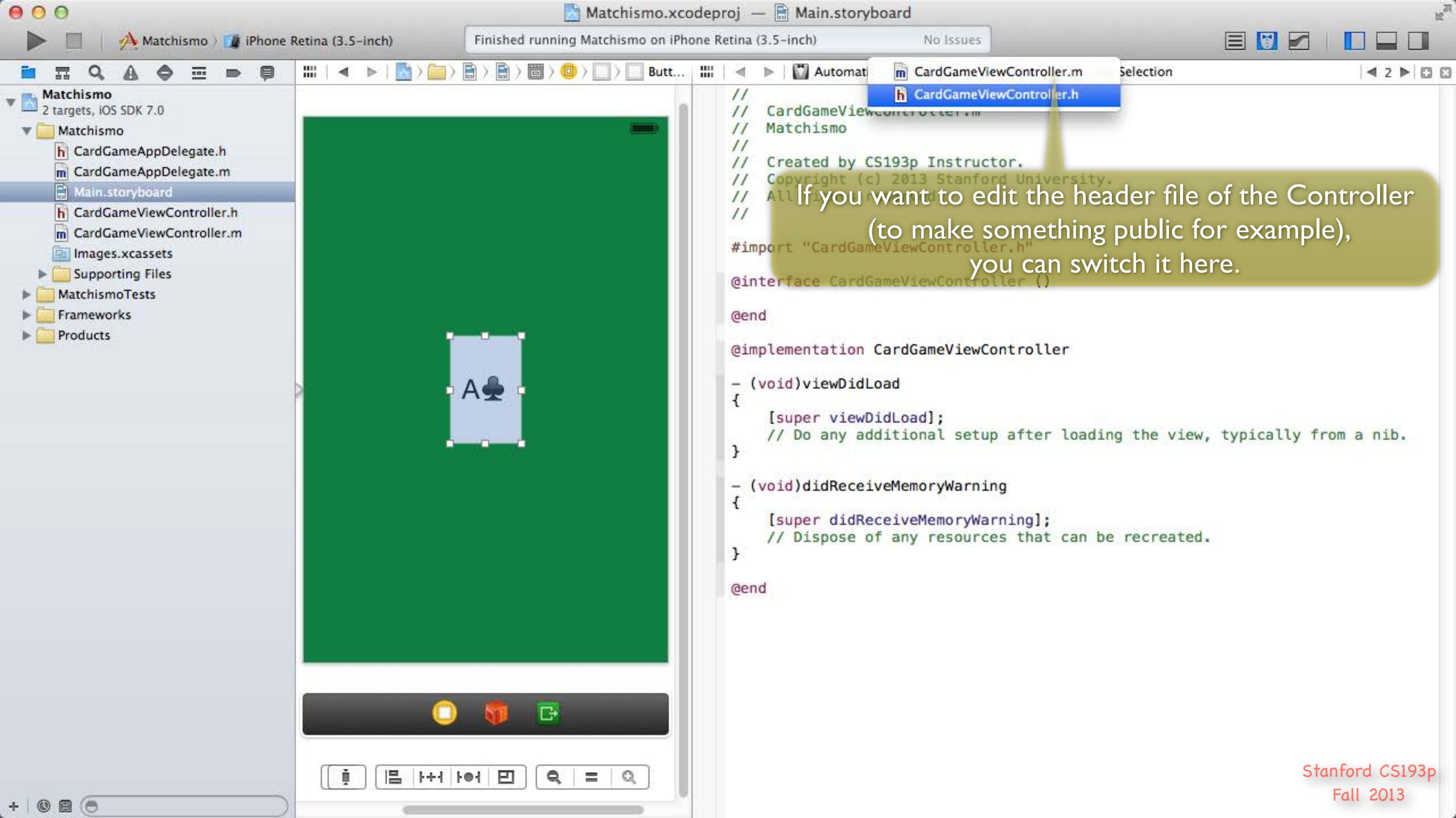
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view,
        typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

This is the code for our MVC's Controller.
So far it's just some stubs created by Xcode based on the template we chose at the start.

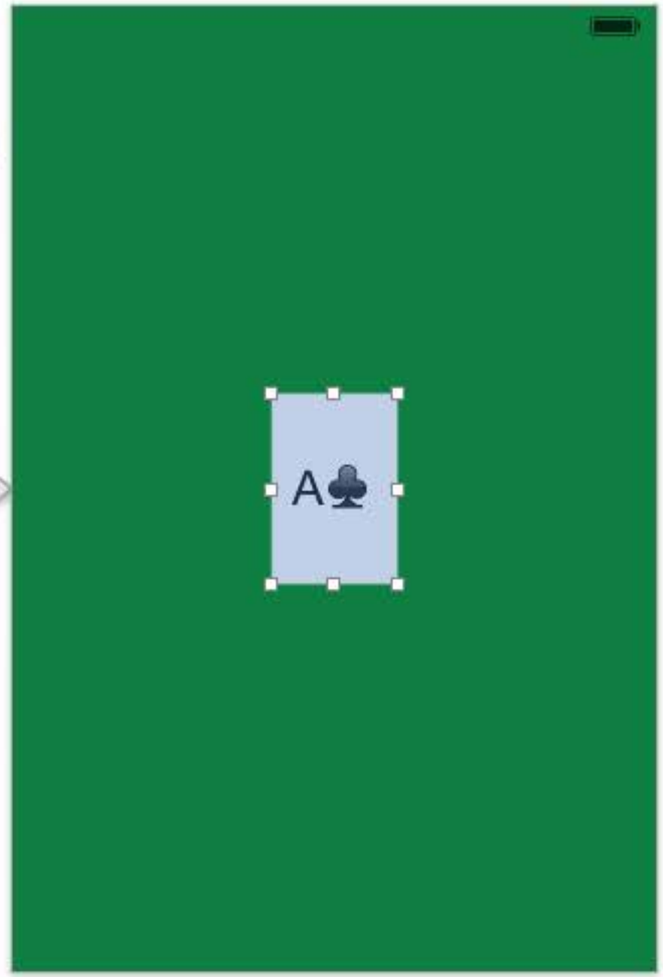

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › No Selection
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}
@end
You can adjust the space between the two panes of the editor by just dragging the space between them.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardGameViewController.m
CardGameViewController.h
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}
@end
If you want to edit the header file of the Controller
(to make something public for example),
you can switch it here.
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.h › No Selection

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

#import <UIKit/UIKit.h>

@interface CardGameViewController : UIViewController

@end
```

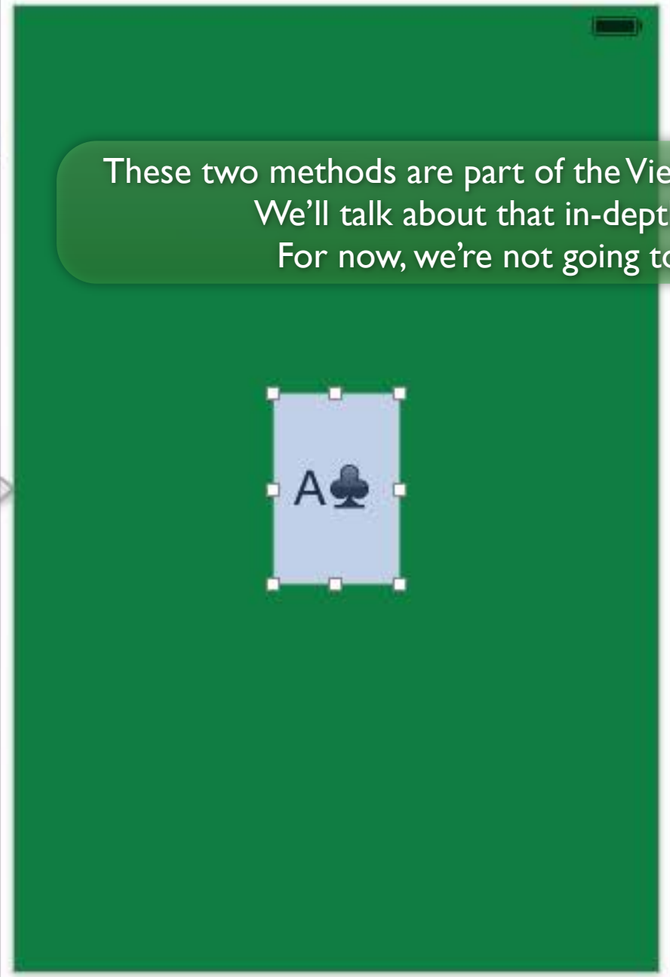
All Controllers inherit from `UIViewController`.

Switch back to the Controller's implementation.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



These two methods are part of the View Controller Lifecycle.
We'll talk about that in-depth next week.
For now, we're not going to use them.

Automatic › CardGameViewController.m › -viewDidLoad

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

@interface CardGameViewController ()

@end

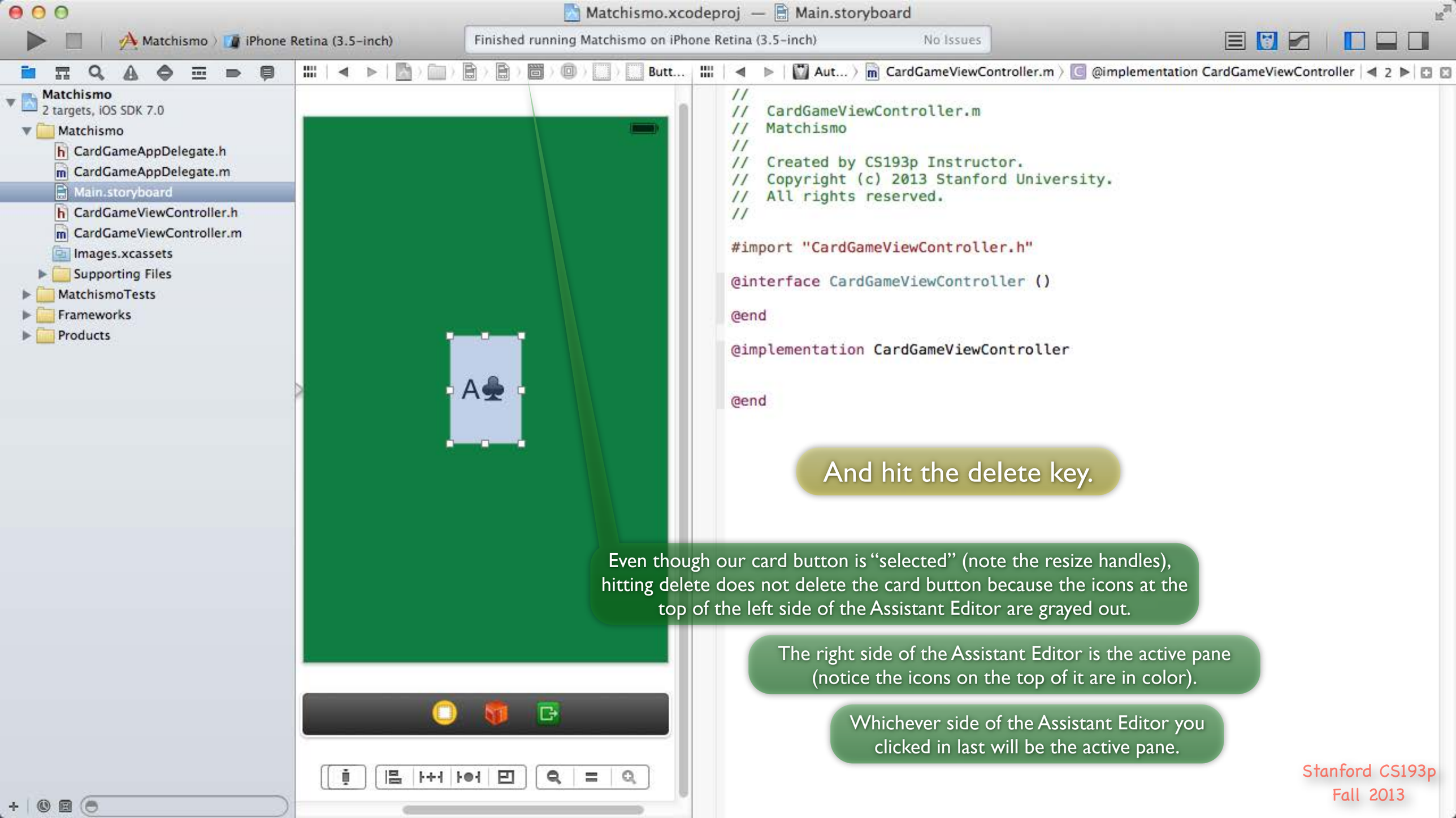
@implementation CardGameViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

Select all of the code inside the `@implementation` block ...


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
A♣
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
And hit the delete key.
Even though our card button is “selected” (note the resize handles), hitting delete does not delete the card button because the icons at the top of the left side of the Assistant Editor are grayed out.
The right side of the Assistant Editor is the active pane (notice the icons on the top of it are in color).
Whichever side of the Assistant Editor you clicked in last will be the active pane.
Stanford CS193p
Fall 2013

Now it's time to connect the button to our Controller so that touching the button "flips the card over."

Believe it or not, you connect your View to your Controller by directly dragging from objects in your View into your source code. Sounds crazy, I know ...

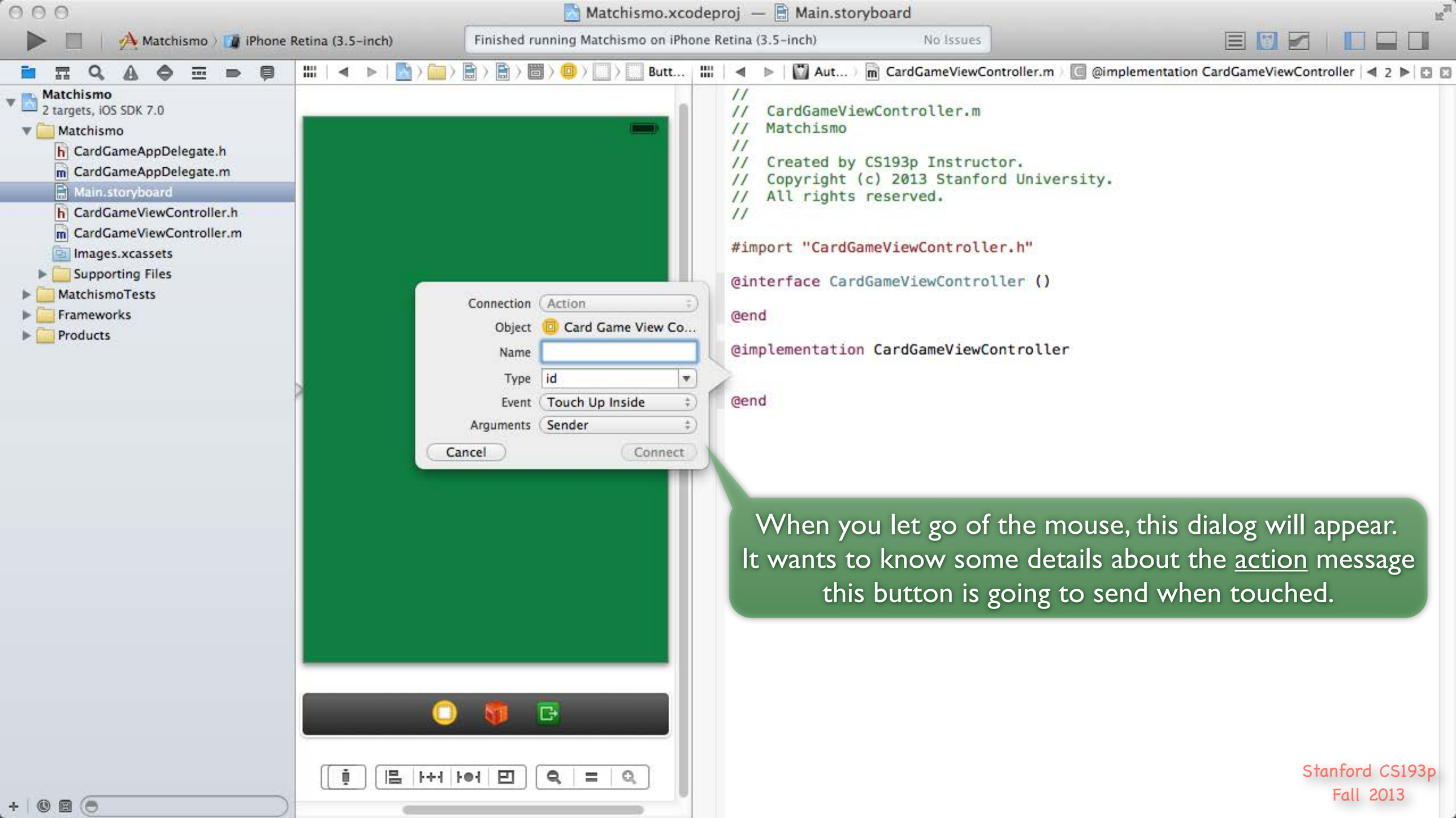


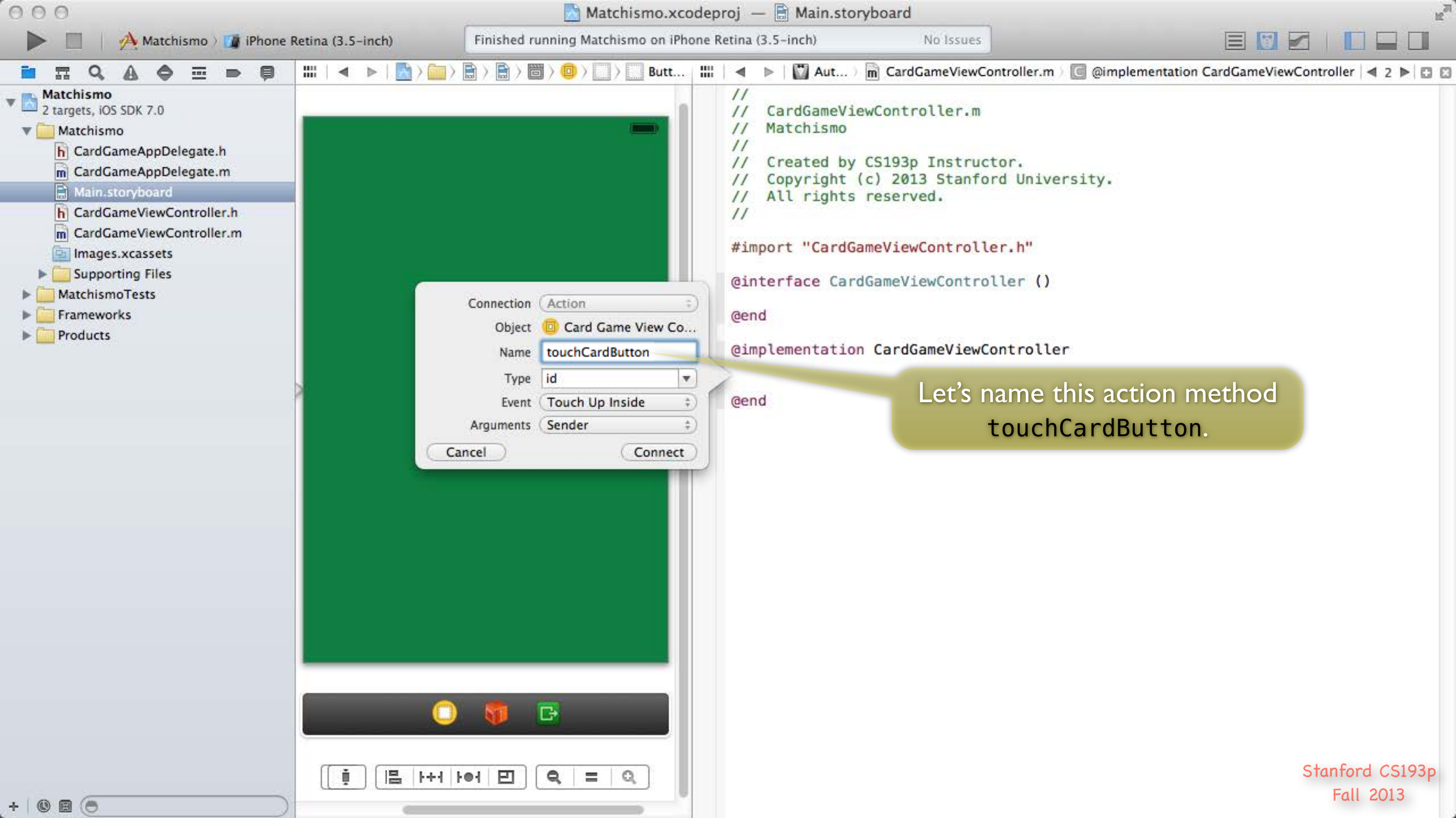
Hold down the ctrl key while dragging from the card button into the `@implementation` block somewhere.

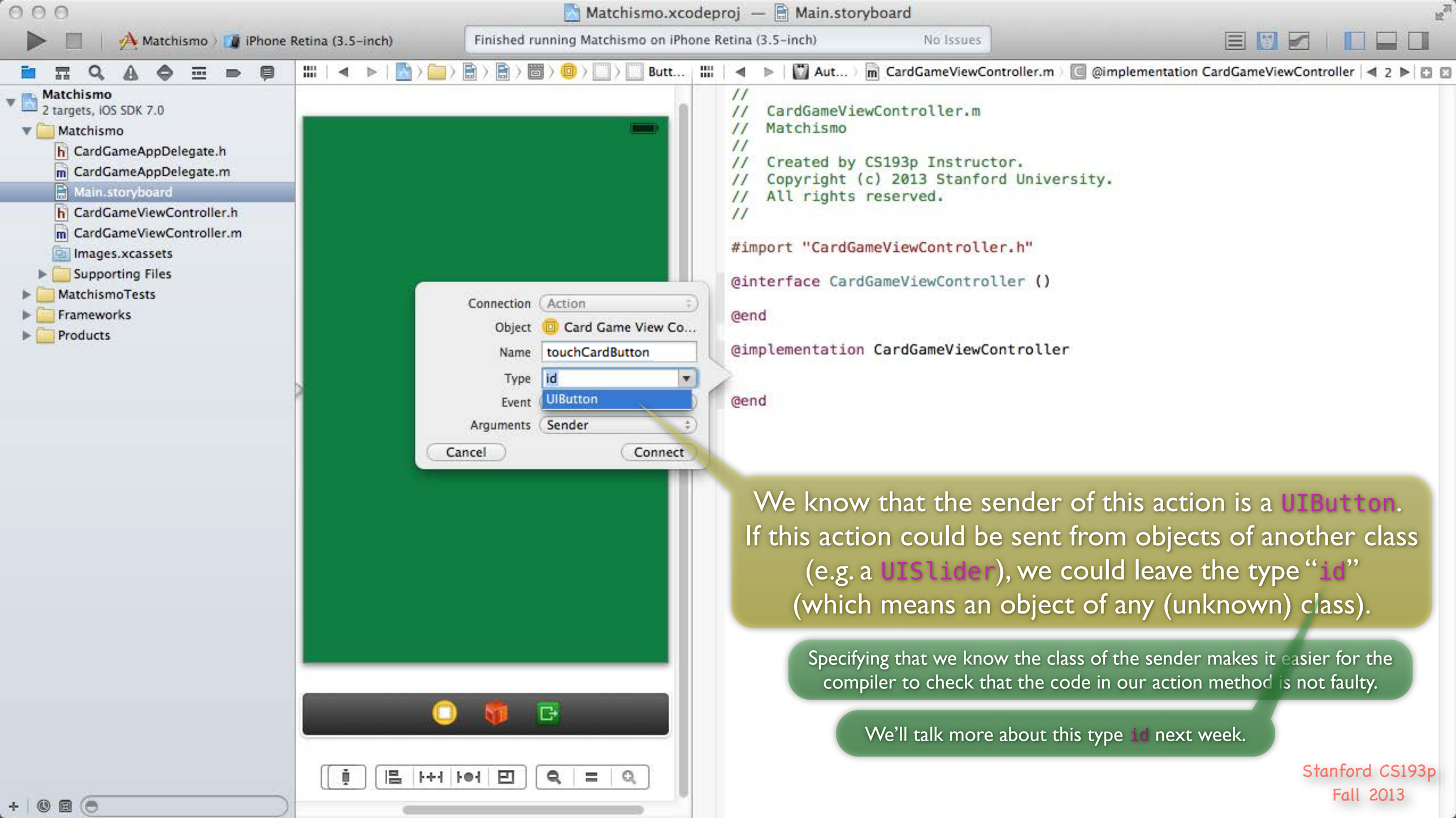
If you do not hold down the ctrl key, this will simply drag the button around.

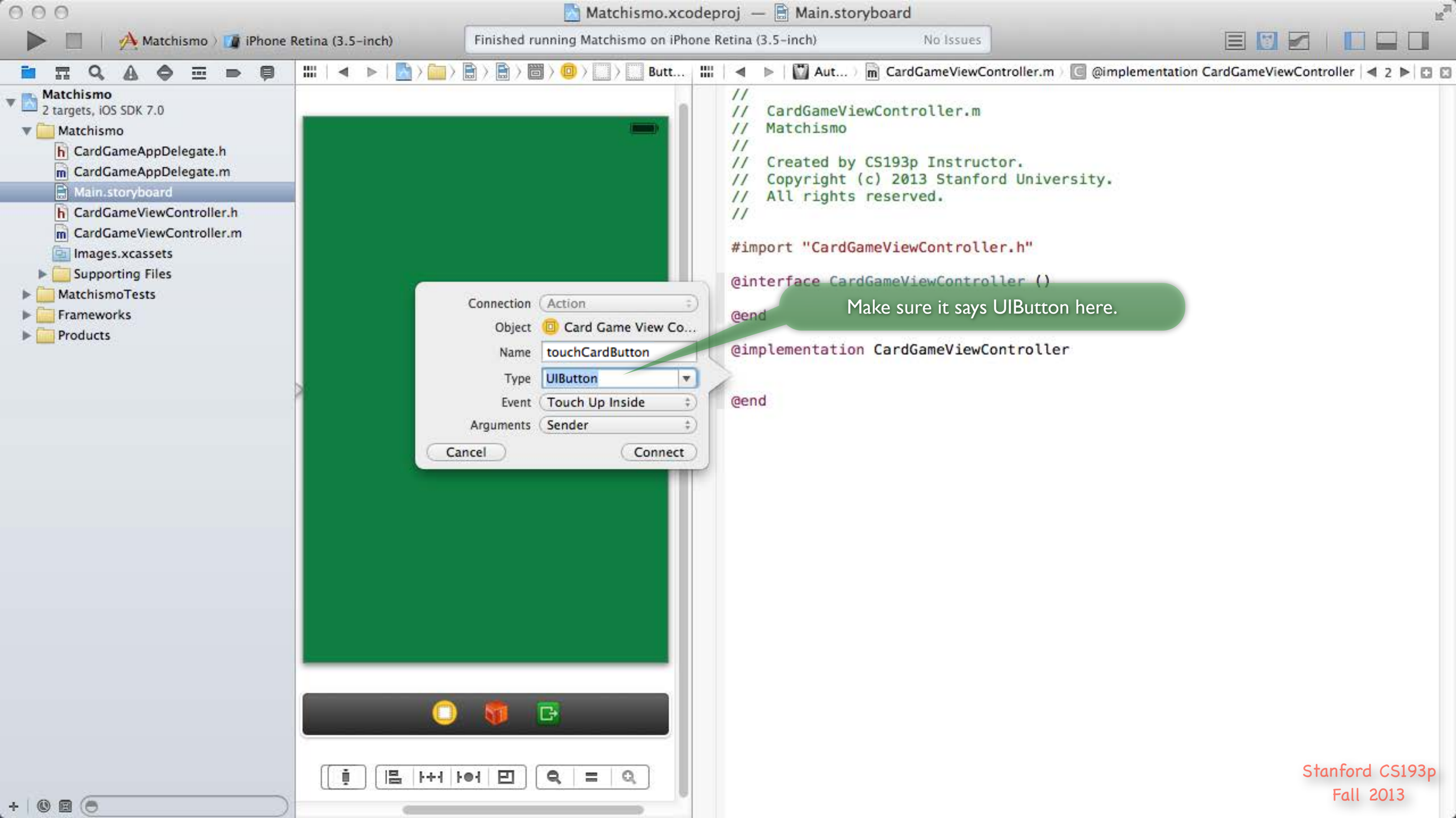
Be sure to drag somewhere between the `@implementation` and the `@end`.

You can drag from your View to the header (.h) file of your Controller if you want to make a public connection (rare).


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Connection Action
Object Card Game View Co...
Name
Type id
Event Touch Up Inside
Arguments Sender
Cancel
Connect
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
When you let go of the mouse, this dialog will appear.
It wants to know some details about the action message
this button is going to send when touched.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Connection Action
Object Card Game View Co...
Name touchCardButton
Type id
Event Touch Up Inside
Arguments Sender
Cancel
Connect
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
Let's name this action method touchCardButton.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Connection Action
Object Card Game View Co...
Name touchCardButton
Type id
UIButton
Event
Arguments Sender
Cancel
Connect
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
We know that the sender of this action is a UIButton.
If this action could be sent from objects of another class
(e.g. a UISlider), we could leave the type "id"
(which means an object of any (unknown) class).
Specifying that we know the class of the sender makes it easier for the compiler to check that the code in our action method is not faulty.
We'll talk more about this type id next week.
Stanford CS193p
Fall 2013

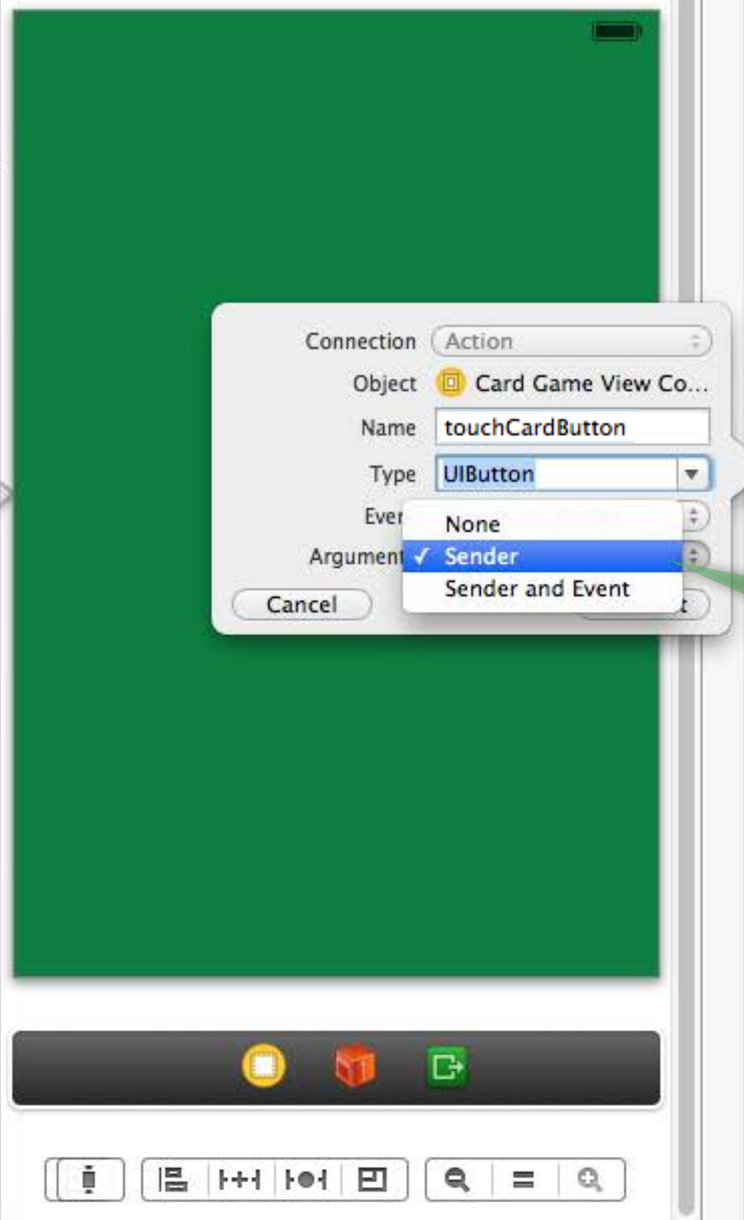


Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products




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

#import "CardGameViewController.h"

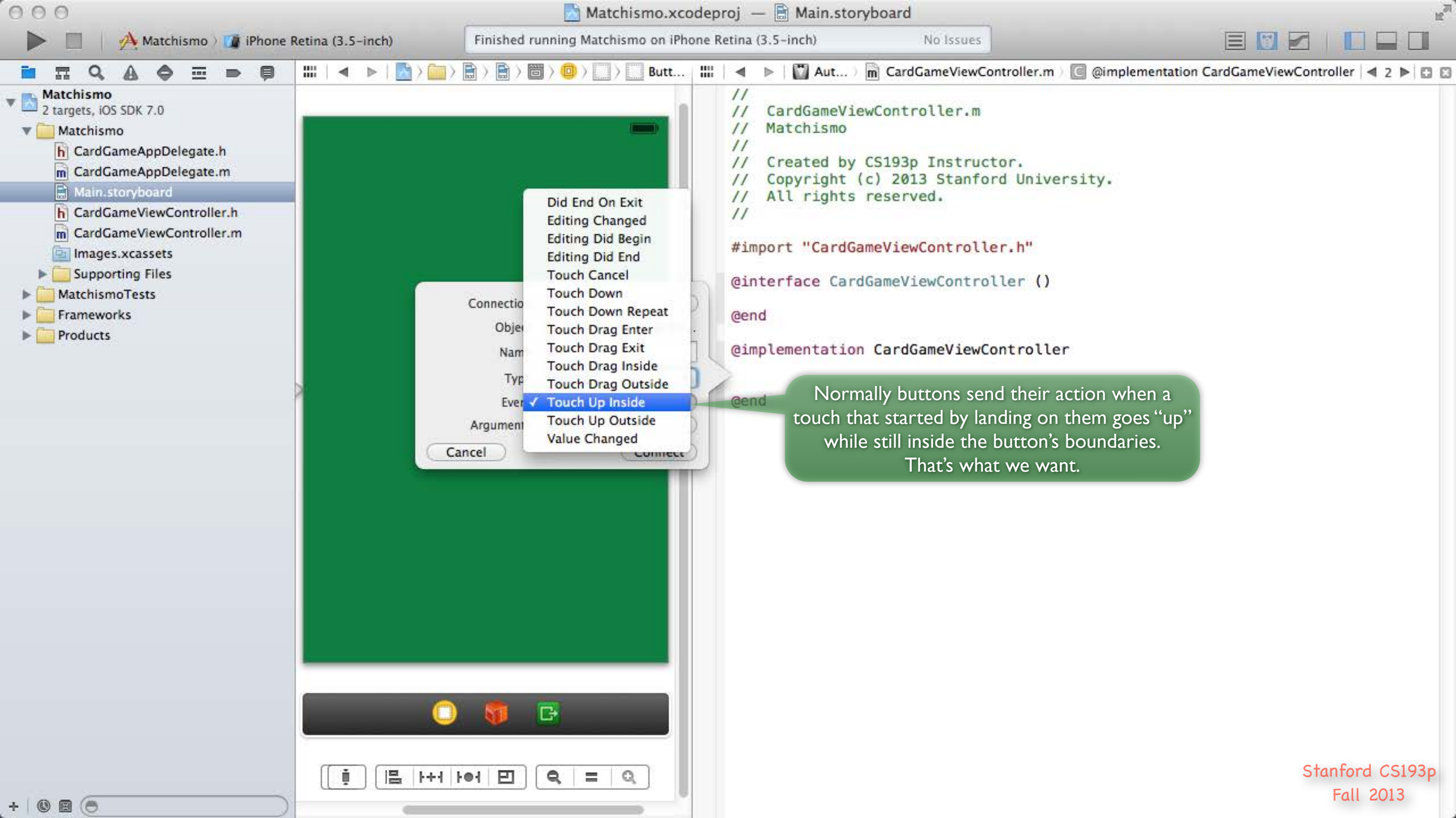
@interface CardGameViewController ()

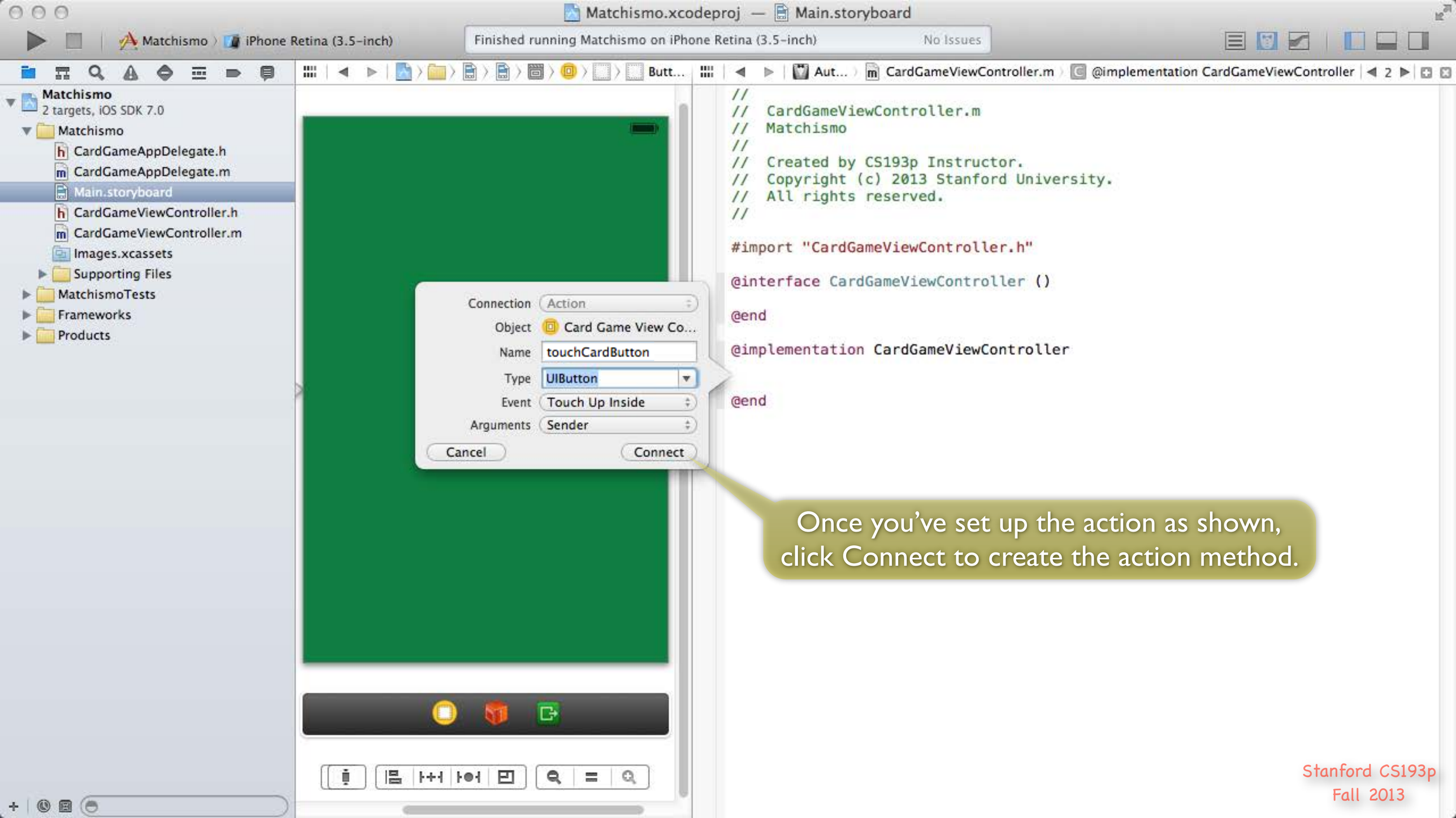
@end

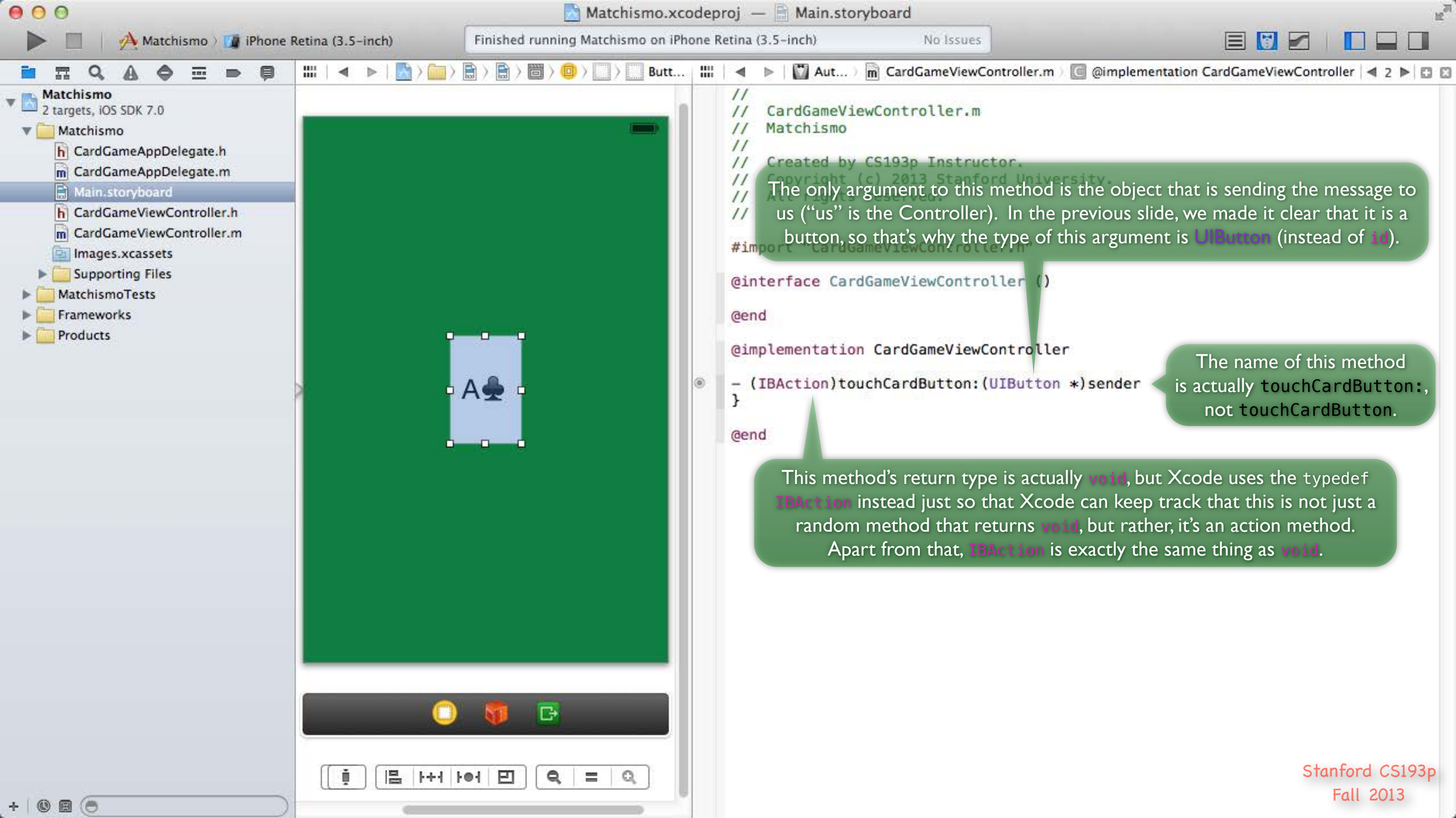
@implementation CardGameViewController

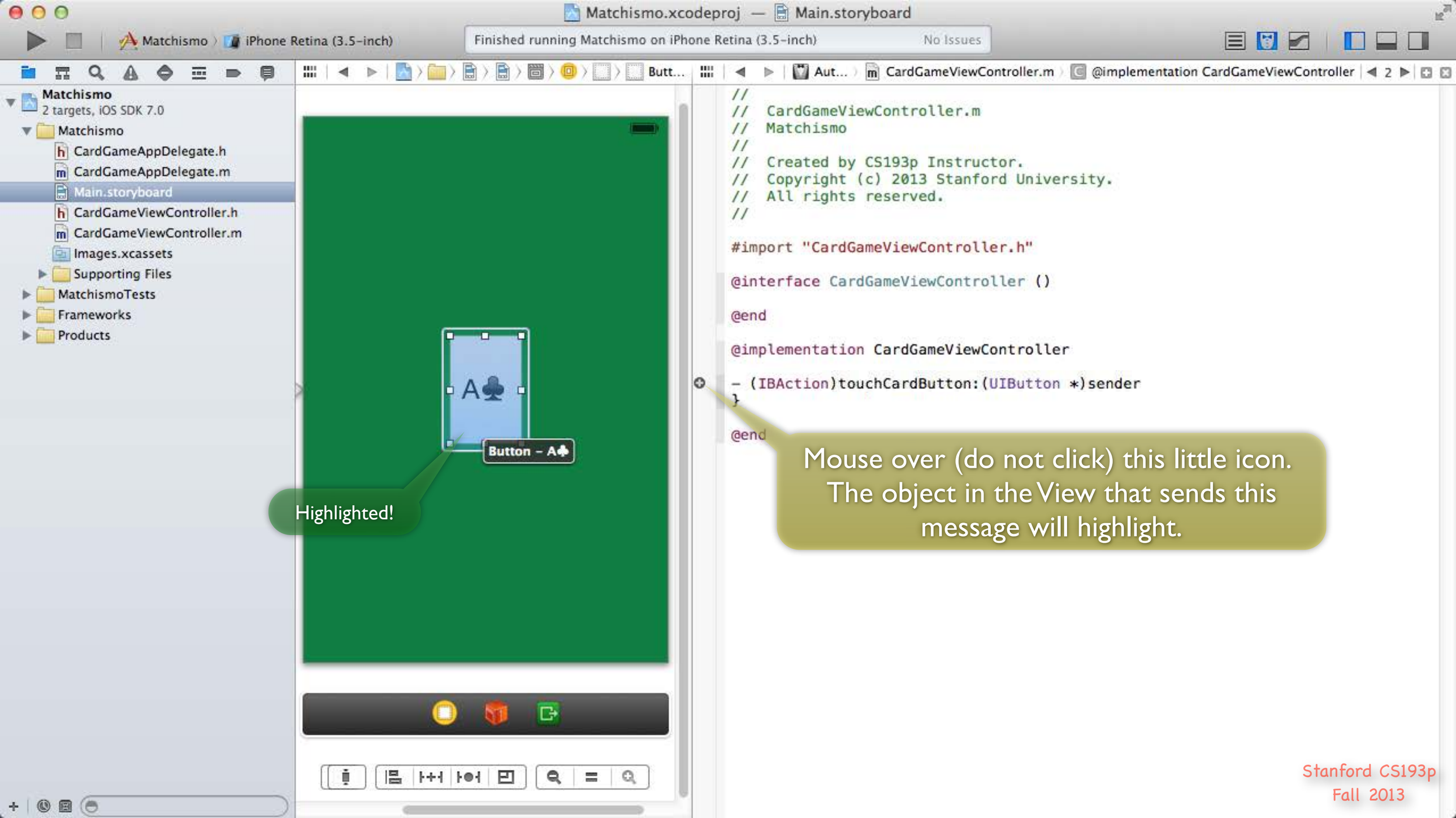

@end
```

Action methods are allowed to have either no arguments,
one argument (the object sending the message),
or even two arguments
(the sender and the touch event provoking the action).
In this case, though, what we want is just the sending button
so that we can change it when it is touched.


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Touch Up Outside
Value Changed
Cancel
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
Normally buttons send their action when a touch that started by landing on them goes "up" while still inside the button's boundaries. That's what we want.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Connection Action
Object Card Game View Co...
Name touchCardButton
Type UIButton
Event Touch Up Inside
Arguments Sender
Cancel
Connect
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
Once you've set up the action as shown, click Connect to create the action method.


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
}
@end
The only argument to this method is the object that is sending the message to us ("us" is the Controller). In the previous slide, we made it clear that it is a button, so that's why the type of this argument is UIButton (instead of id).
The name of this method is actually touchCardButton:, not touchCardButton.
This method's return type is actually void, but Xcode uses the typedef IBAction instead just so that Xcode can keep track that this is not just a random method that returns void, but rather, it's an action method. Apart from that, IBAction is exactly the same thing as void.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Button – A♣
Highlighted!
CardGameViewController.m
@implementation CardGameViewController
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
}
@end
Mouse over (do not click) this little icon. The object in the View that sends this message will highlight.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Our implementation of this method is quite simple.
We're just going to change the text on the button to be blank
and change the background image to be our card back
(the Stanford logo), thus "flipping the card over to its back."



Automatic › CardGameViewController.m › -touchCardButton:

```
//
//  CardGameViewController.m
//
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    UIImage *cardImage =
}

@end
```

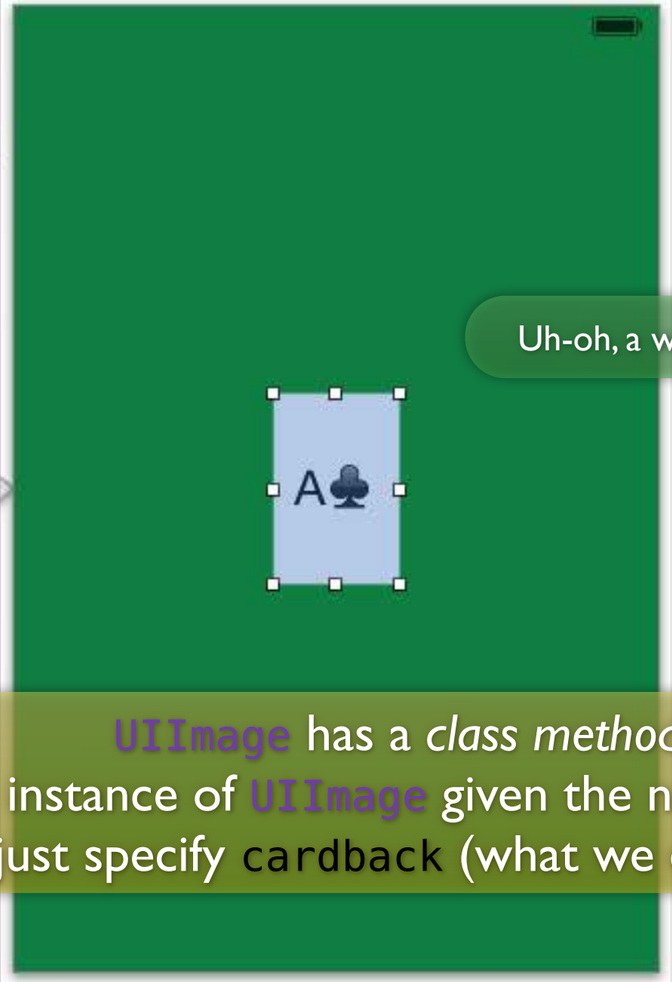
Let's start by declaring a *local variable* called `cardImage` to hold the `cardback` image (the Stanford logo we imported).
The local variable is a pointer to an instance of the class `UIImage` which represents a JPEG, PNG, TIFF or other image.

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton: | 2

- Matchismo
  - 2 targets, iOS SDK 7.0
  - Matchismo
    - CardGameAppDelegate.h
    - CardGameAppDelegate.m
    - Main.storyboard
    - CardGameViewController.h
    - CardGameViewController.m
    - Images.xcassets
    - Supporting Files
  - MatchismoTests
  - Frameworks
  - Products




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    UIImage *cardImage = [UIImage imageNamed:@"cardback"];
}

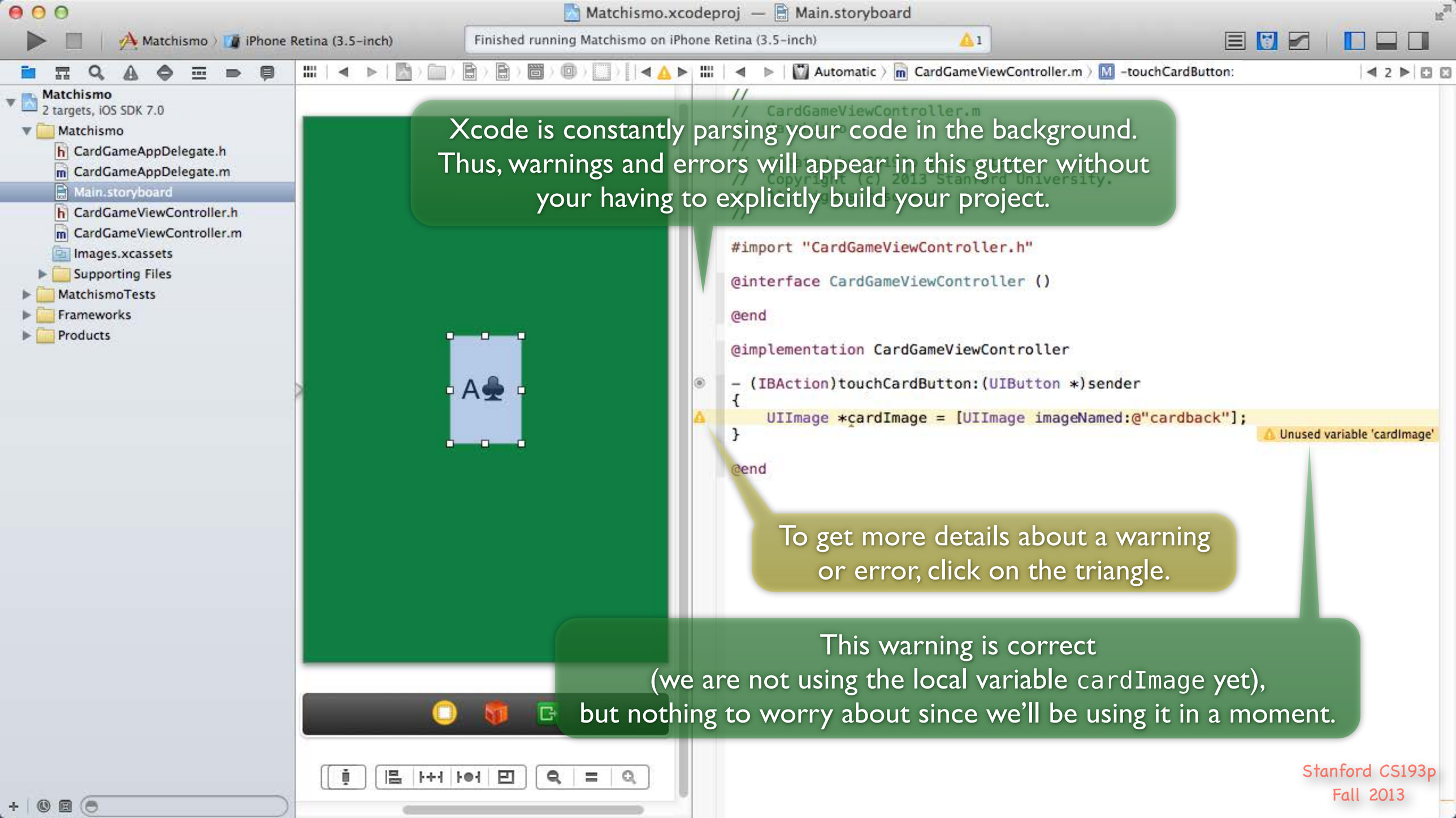
@end
```

Uh-oh, a warning!

`UIImage` has a *class method* called `imageNamed:` which creates an instance of `UIImage` given the name of an image in the image assets library. We just specify `cardback` (what we called the Stanford logo in the assets library).

Remember that `@""` notation just creates an `NSString` object.

Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
Xcode is constantly parsing your code in the background. Thus, warnings and errors will appear in this gutter without your having to explicitly build your project.
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
}
Unused variable 'cardImage'
@end
To get more details about a warning or error, click on the triangle.
This warning is correct (we are not using the local variable cardImage yet), but nothing to worry about since we'll be using it in a moment.
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

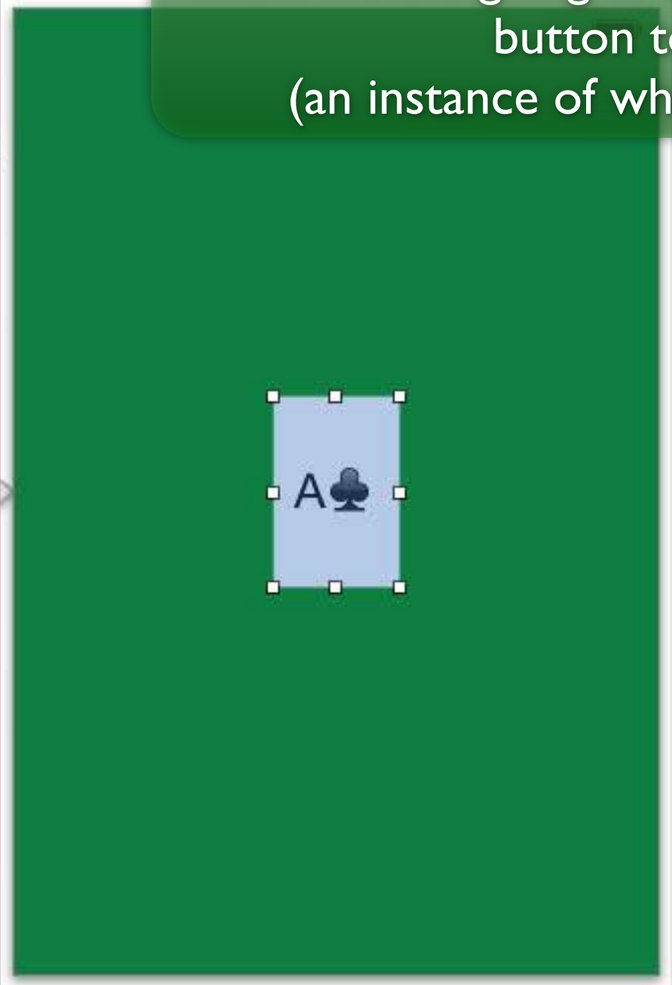
Matchismo › iPhone Retina (3.5-inch) | Finished running Matchismo on iPhone Retina (3.5-inch) | 1

Automatic › CardGameViewController.m › -touchCardButton: | 2

Matchismo
2 targets, iOS SDK 7.0

- Matchismo
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Next we are going to set the background image of the card button to be that Stanford logo (an instance of which is now stored in `cardImage`).



To do that, we need to send a message to the `UIButton` that sent this `touchCardButton:` message to us.

```
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//
#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

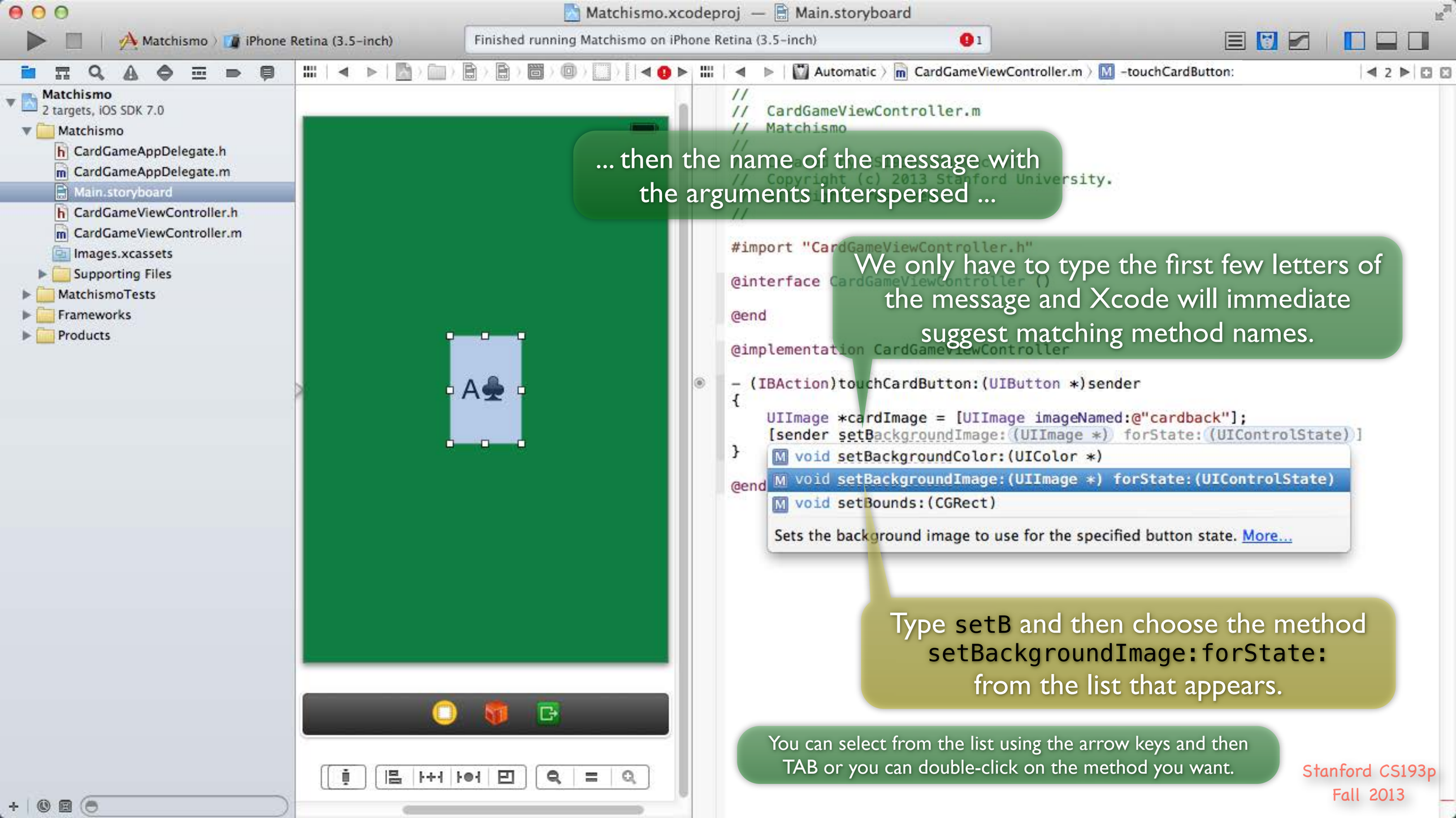
@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    UIImage *cardImage = [UIImage imageNamed:@"cardback"];
    [sender ]
}

@end
```

The argument `sender` is the `UIButton` sending this message.

The syntax for sending a message in Objective-C starts off with `[`, then a pointer to the instance of the object to send the message to ...

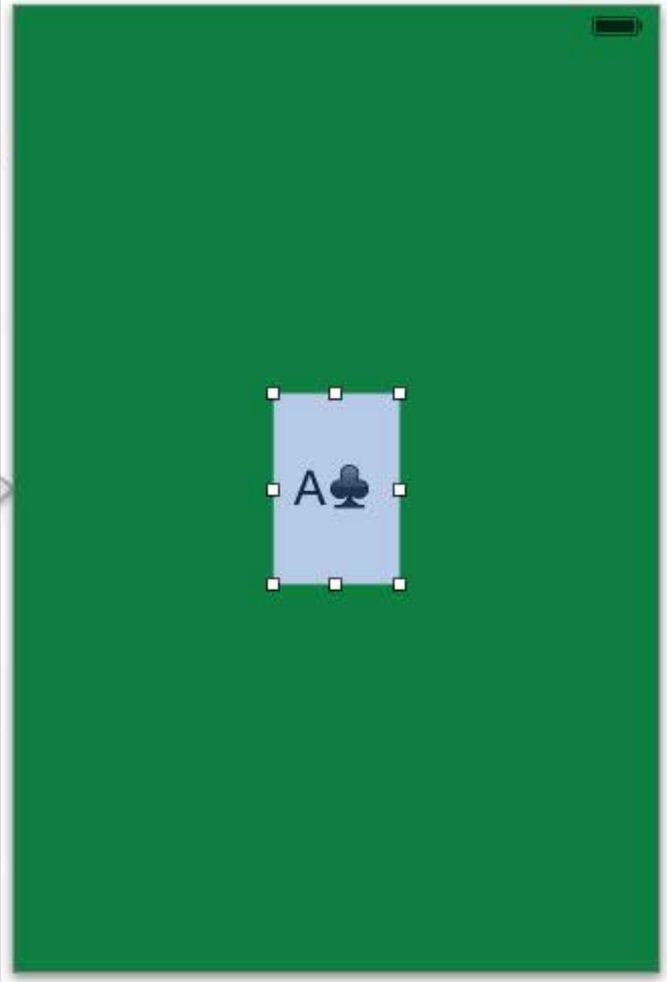

Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic 〉 CardGameViewController.m 〉 -touchCardButton:
// CardGameViewController.m
// Matchismo
// Copyright (c) 2013 Stanford University.
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setB
}
@end
void setBackgroundColor:(UIColor *)
void setBackgroundImage:(UIImage *) forState:(UIControlState)
void setBounds:(CGRect)
Sets the background image to use for the specified button state. More...
... then the name of the message with the arguments interspersed ...
We only have to type the first few letters of the message and Xcode will immediate suggest matching method names.
Type setB and then choose the method setBackgroundImage:forState: from the list that appears.
You can select from the list using the arrow keys and then TAB or you can double-click on the method you want.
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton: | 2

- Matchismo
  - 2 targets, iOS SDK 7.0
  - Matchismo
    - CardGameAppDelegate.h
    - CardGameAppDelegate.m
    - Main.storyboard
    - CardGameViewController.h
    - CardGameViewController.m
    - Images.xcassets
    - Supporting Files
  - MatchismoTests
  - Frameworks
  - Products




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

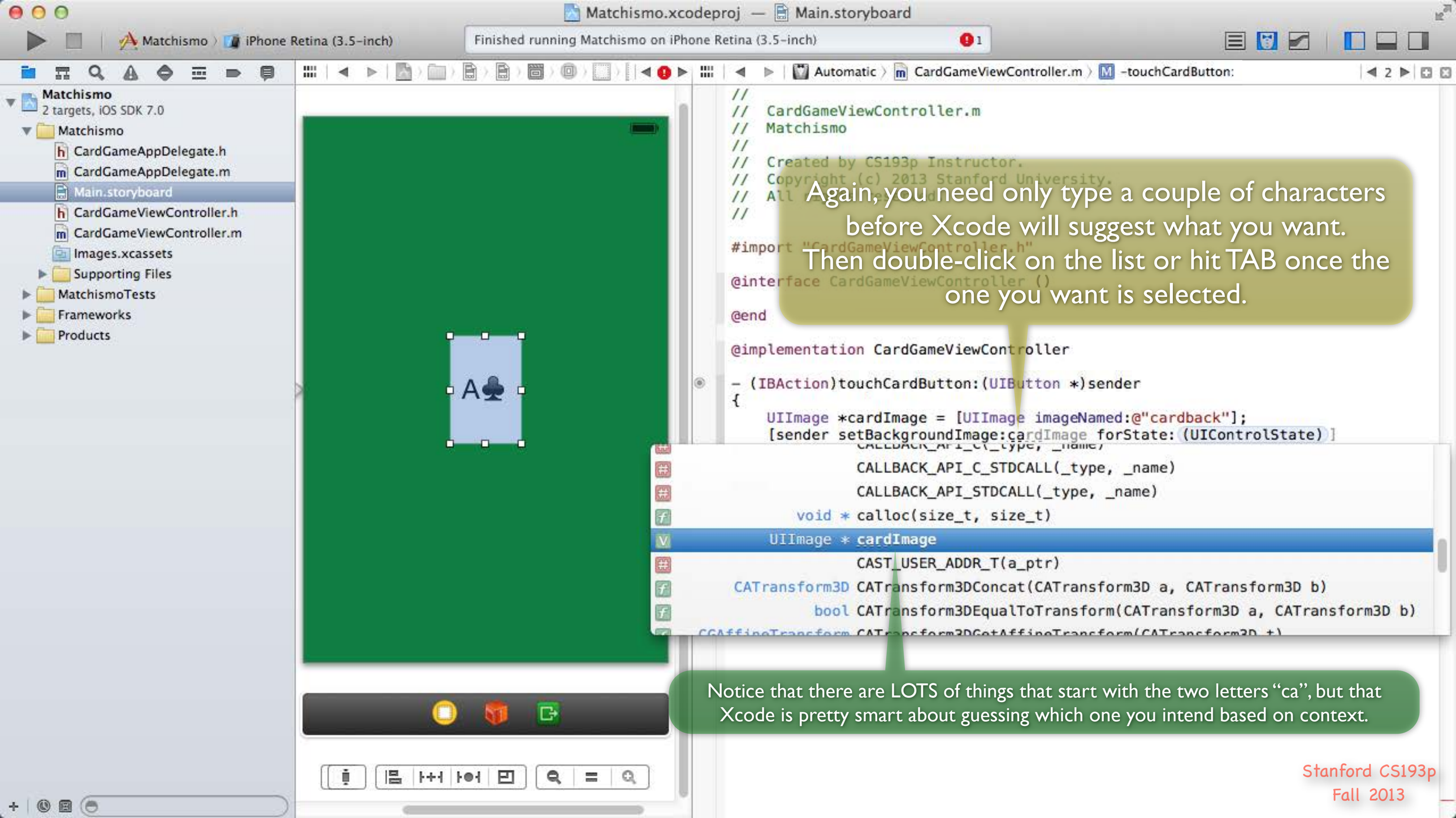
@implementation CardGameViewController

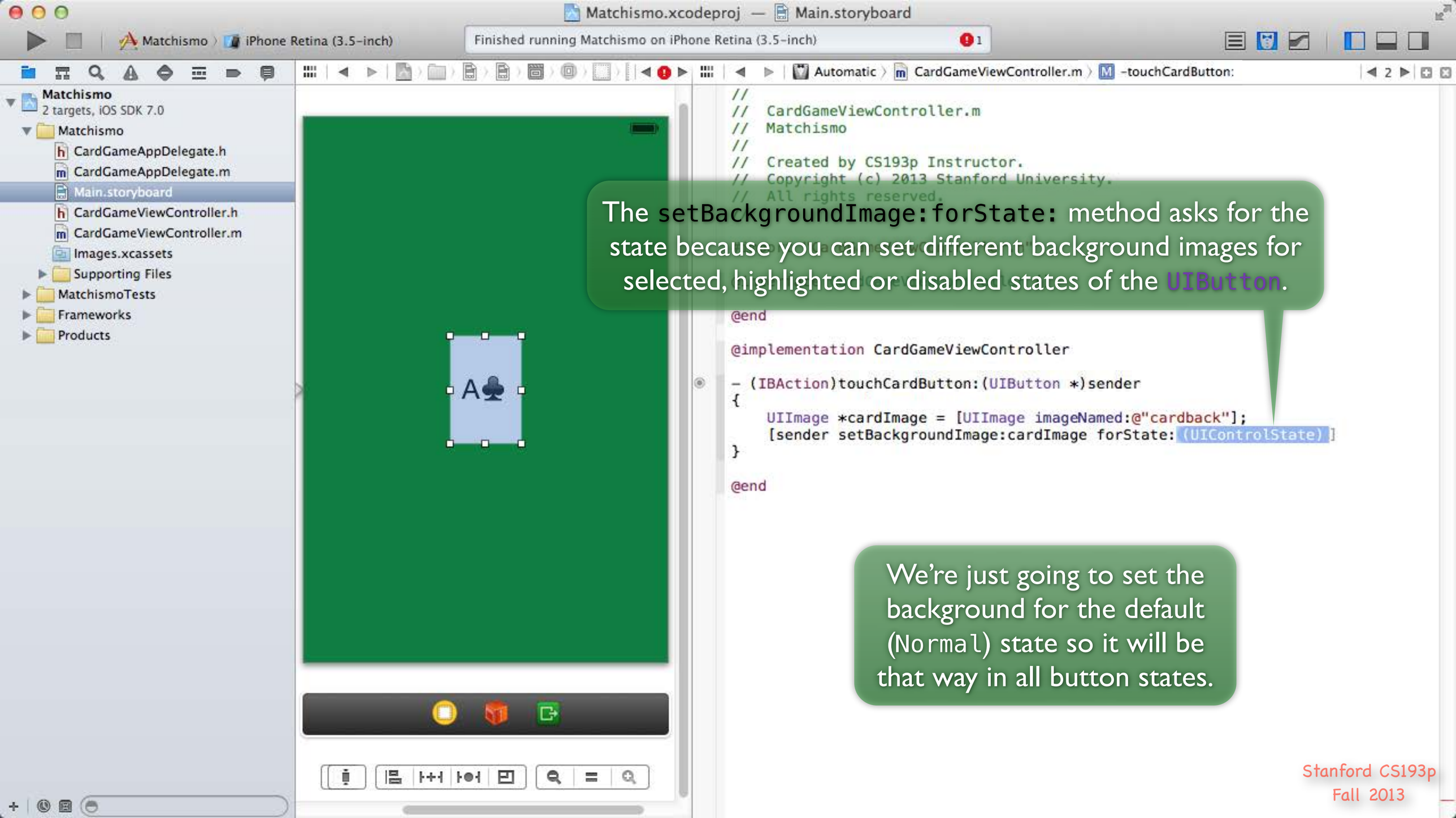
- (IBAction)touchCardButton:(UIButton *)sender
{
    UIImage *cardImage = [UIImage imageNamed:@"cardback"];
    [sender setBackgroundImage:(UIImage *) forState:(UIControlState)]
}

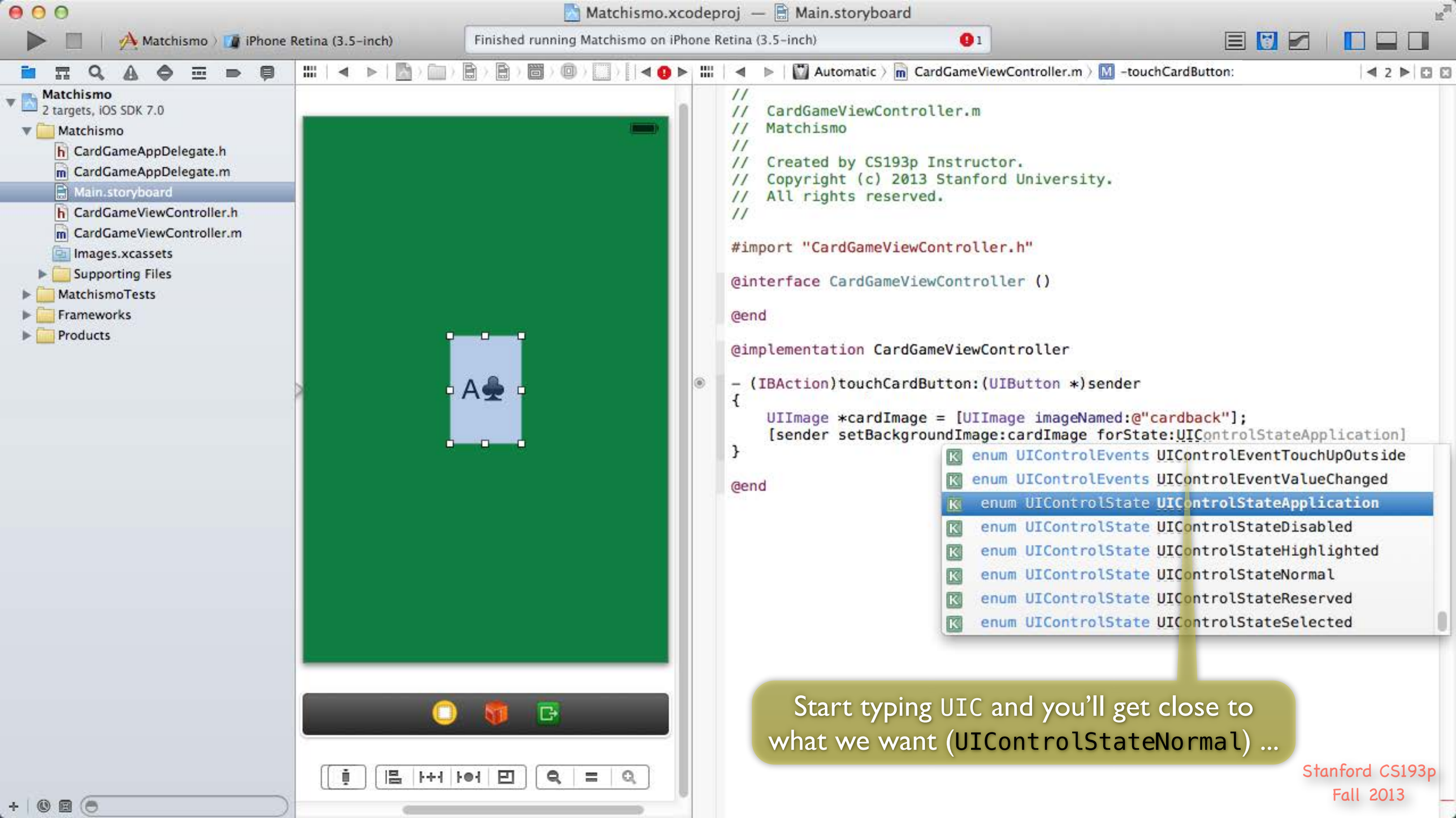
@end
```

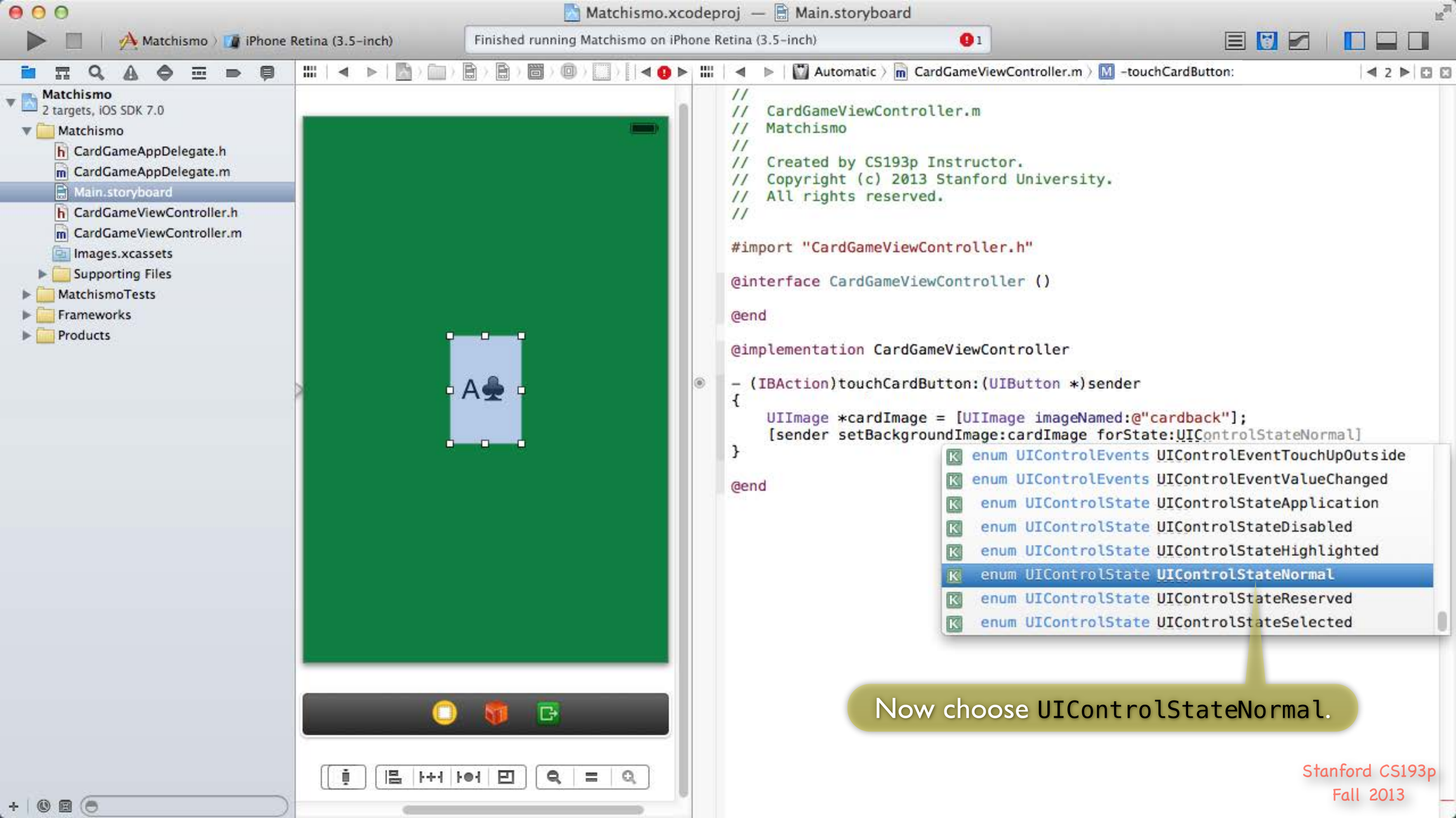
After choosing the method you want, it should fill that method in and highlight the first argument. This argument is obviously the image to set as the background of the `UIButton`.

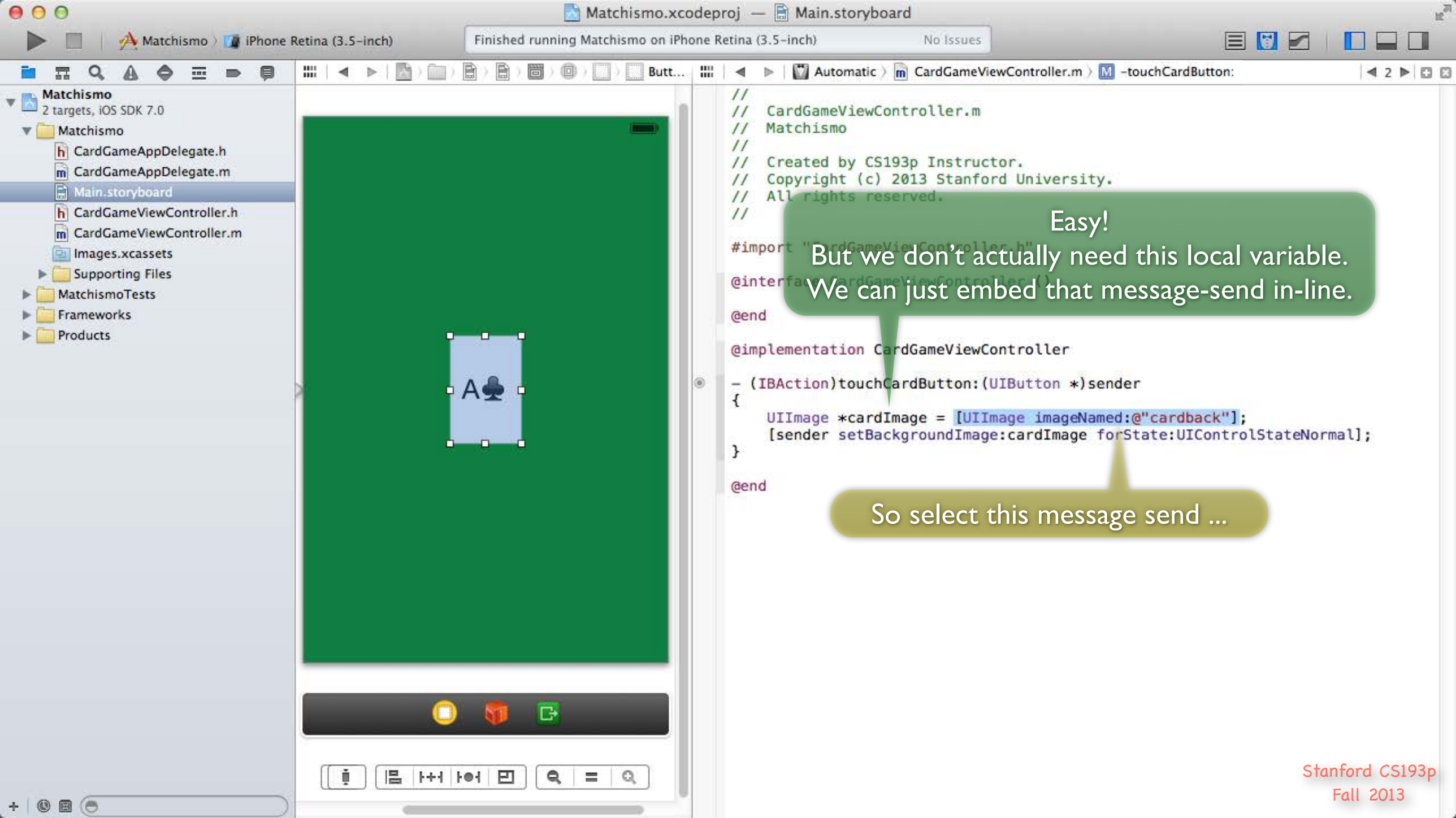
For us, that's the `cardImage` local variable.

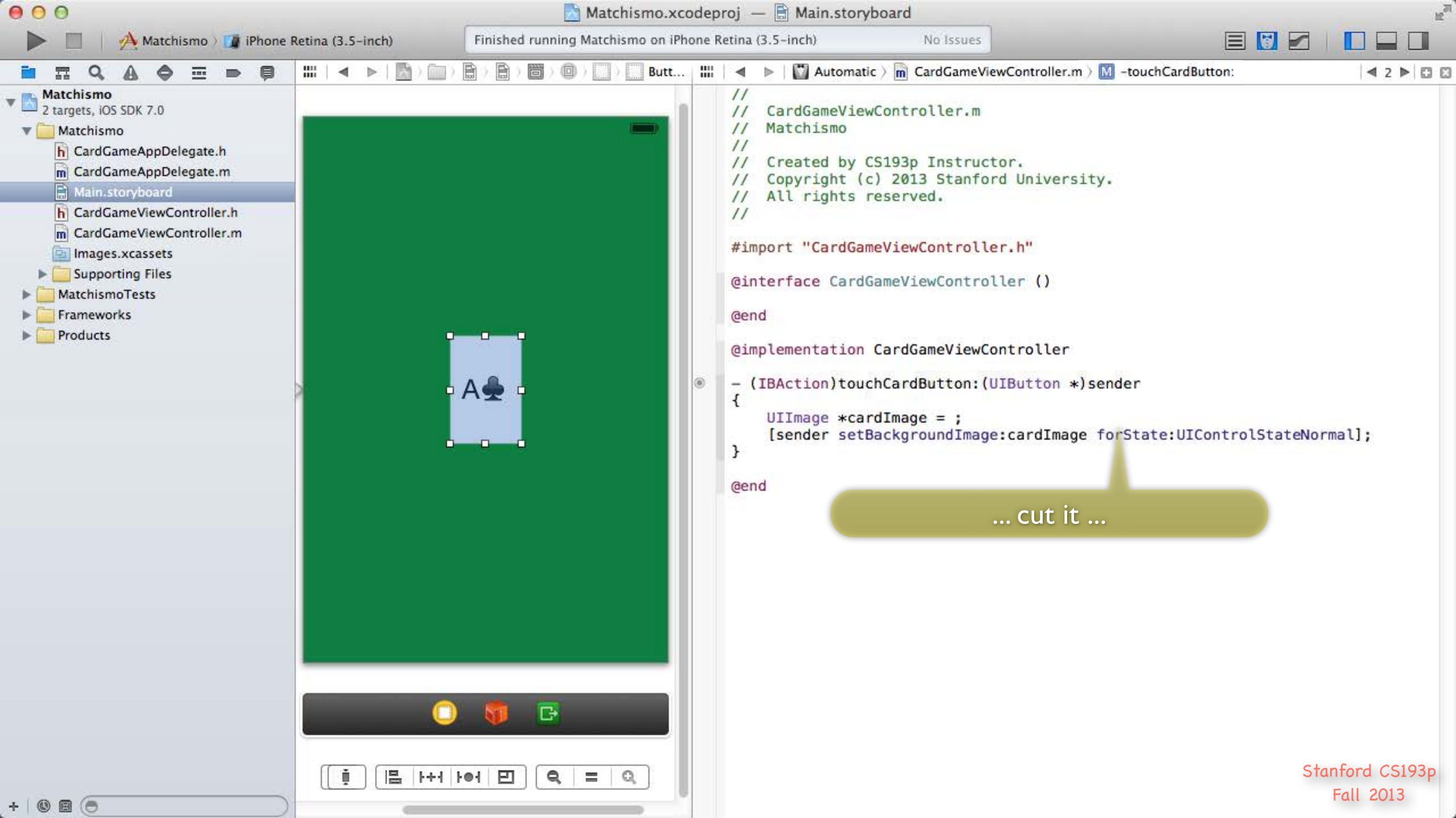

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setBackgroundImage:cardImage forState:(UIControlState)]
CALLBACK_API_C_STDCALL(_type, _name)
CALLBACK_API_STDCALL(_type, _name)
void * calloc(size_t, size_t)
UIImage * cardImage
CAST_USER_ADDR_T(a_ptr)
CATransform3D CATransform3DConcat(CATransform3D a, CATransform3D b)
bool CATransform3DEqualToTransform(CATransform3D a, CATransform3D b)
Again, you need only type a couple of characters before Xcode will suggest what you want. Then double-click on the list or hit TAB once the one you want is selected.
Notice that there are LOTS of things that start with the two letters "ca", but that Xcode is pretty smart about guessing which one you intend based on context.
Stanford CS193p
Fall 2013

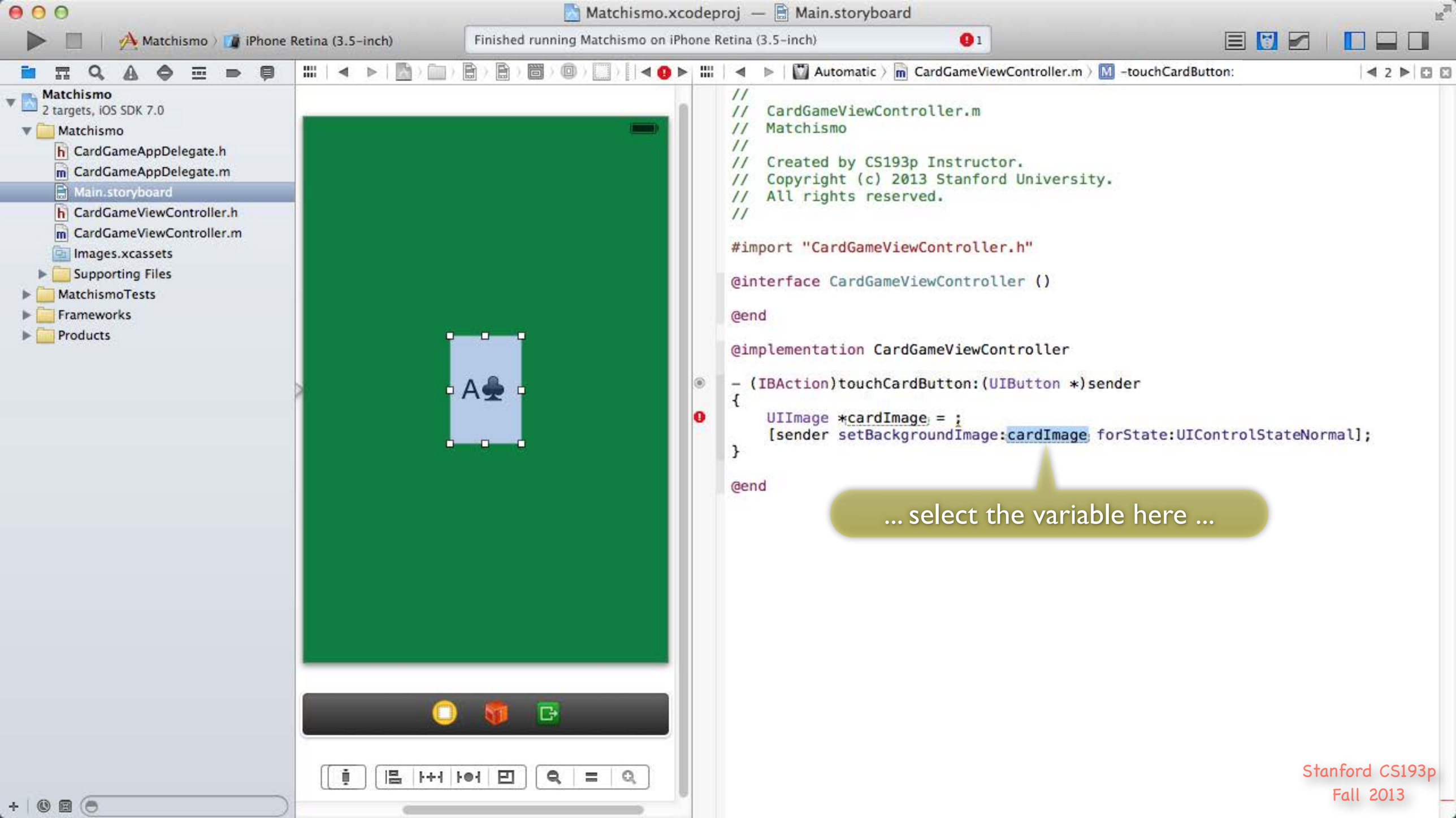

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setBackgroundImage:cardImage forState:(UIControlState)]
}
@end
The setBackgroundImage:forState: method asks for the state because you can set different background images for selected, highlighted or disabled states of the UIButton.
We're just going to set the background for the default (Normal) state so it will be that way in all button states.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setBackgroundImage:cardImage forState:UIControlStateApplication]
}
@end
enum UIControlEvents UIControlEventTouchUpOutside
enum UIControlEvents UIControlEventValueChanged
enum UIControlState UIControlStateApplication
enum UIControlState UIControlStateDisabled
enum UIControlState UIControlStateHighlighted
enum UIControlState UIControlStateNormal
enum UIControlState UIControlStateReserved
enum UIControlState UIControlStateSelected
Start typing UIC and you'll get close to what we want (UIControlStateNormal) ...
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setBackgroundImage:cardImage forState:UIControlStateNormal]
}
@end
enum UIControlEvents UIControlEventTouchUpOutside
enum UIControlEvents UIControlEventValueChanged
enum UIControlState UIControlStateApplication
enum UIControlState UIControlStateDisabled
enum UIControlState UIControlStateHighlighted
enum UIControlState UIControlStateNormal
enum UIControlState UIControlStateReserved
enum UIControlState UIControlStateSelected
Now choose UIControlStateNormal.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = [UIImage imageNamed:@"cardback"];
[sender setBackgroundImage:cardImage forState:UIControlStateNormal];
}
@end
Easy!
But we don't actually need this local variable.
We can just embed that message-send in-line.
So select this message send ...
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic 〉 CardGameViewController.m 〉 -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = ;
[sender setBackgroundImage:cardImage forState:UIControlStateNormal];
}
@end
... cut it ...
Stanford CS193p
Fall 2013

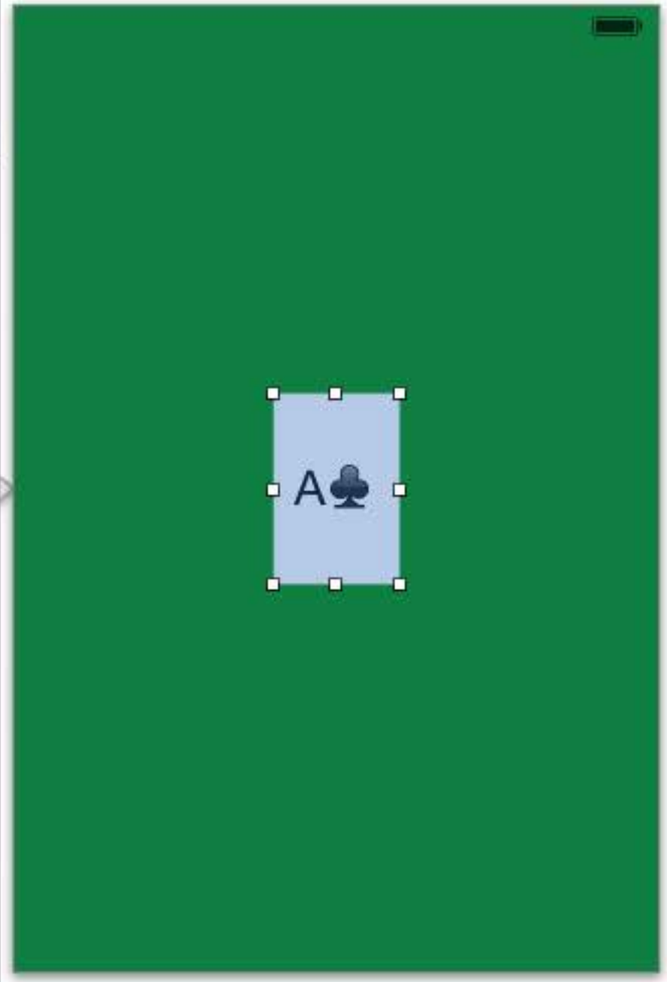

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = ;
[sender setBackgroundImage:cardImage forState:UIControlStateNormal];
}
@end
... select the variable here ...
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton: | 2

- Matchismo
  - 2 targets, iOS SDK 7.0
  - Matchismo
    - CardGameAppDelegate.h
    - CardGameAppDelegate.m
    - Main.storyboard
    - CardGameViewController.h
    - CardGameViewController.m
    - Images.xcassets
    - Supporting Files
  - MatchismoTests
  - Frameworks
  - Products




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

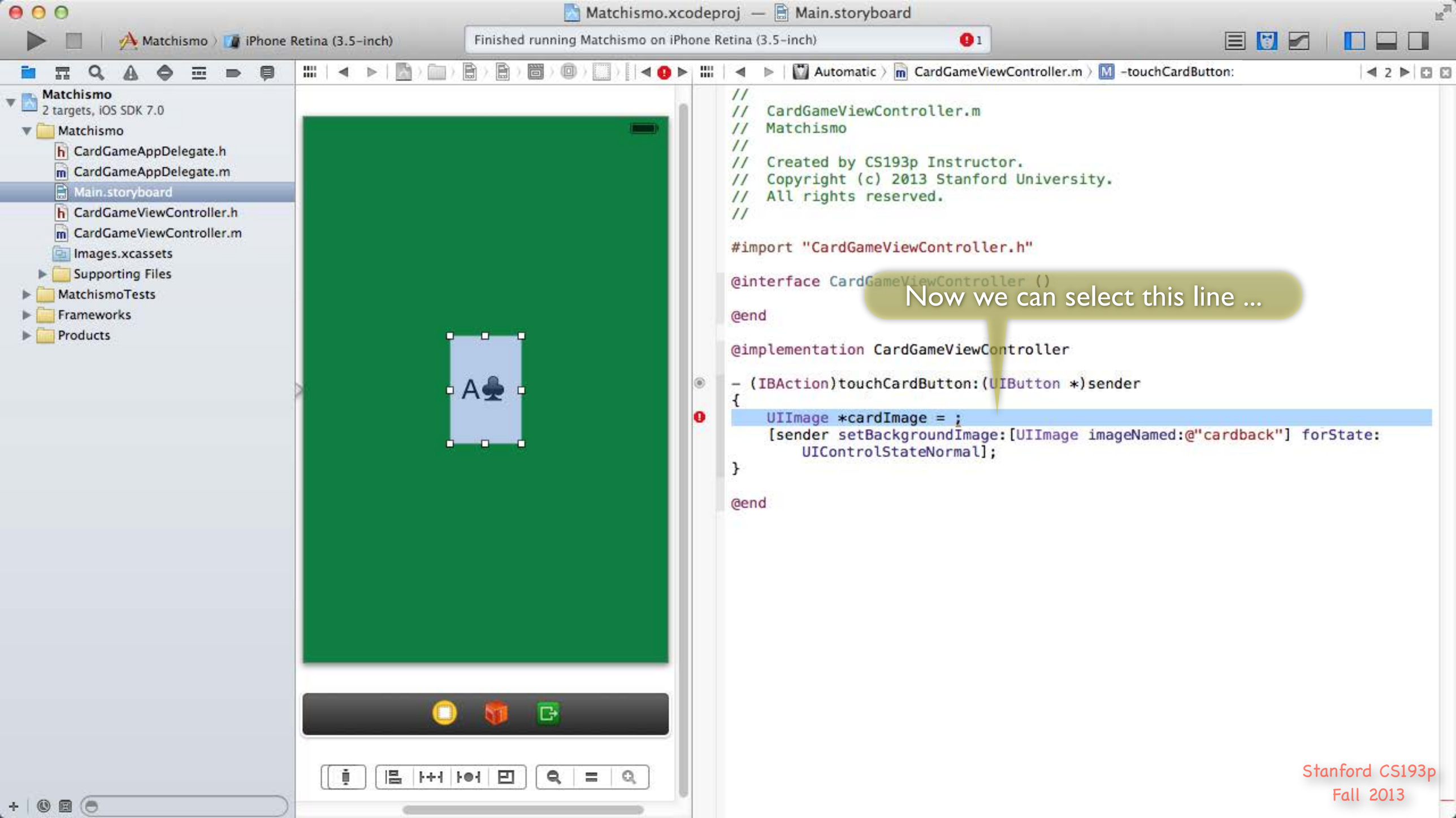
@end

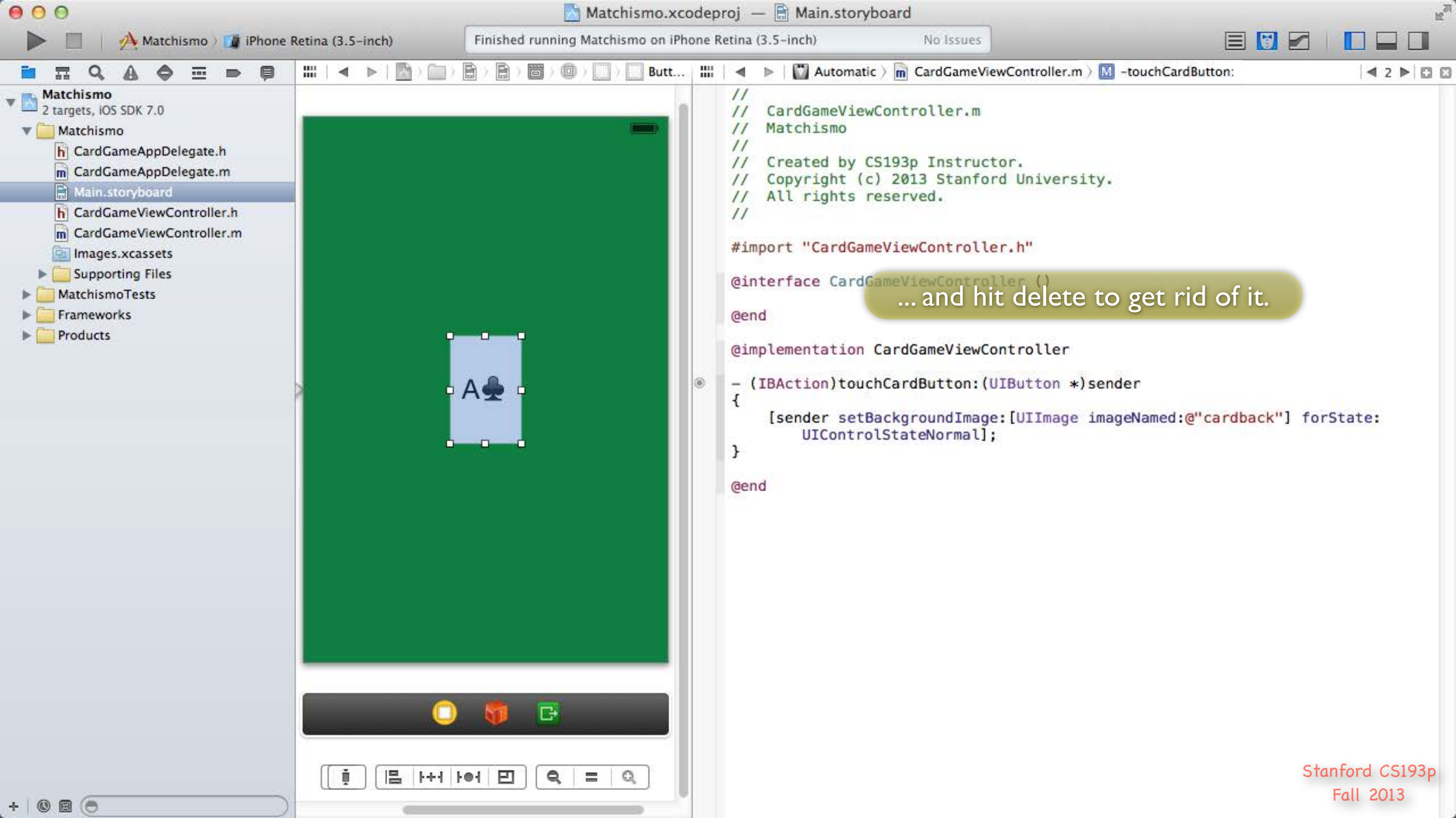
@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    UIImage *cardImage = ;
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:
        UIControlStateNormal];
}

@end
```

... and paste.

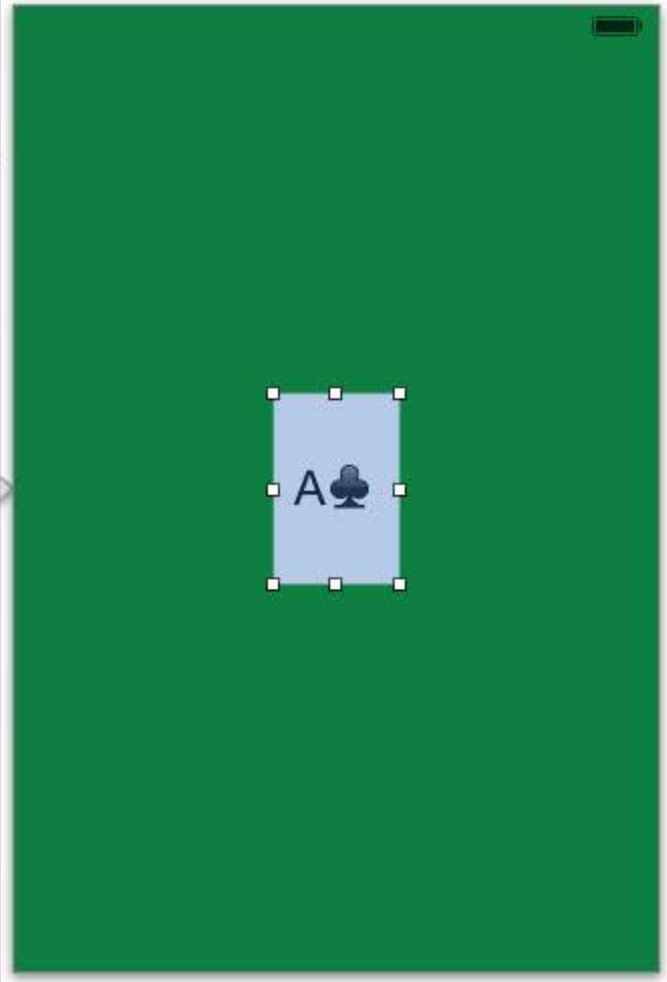

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
UIImage *cardImage = ;
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:
UIControlStateNormal];
}
@end
Now we can select this line ...
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:
UIControlStateNormal];
}
@end
... and hit delete to get rid of it.
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

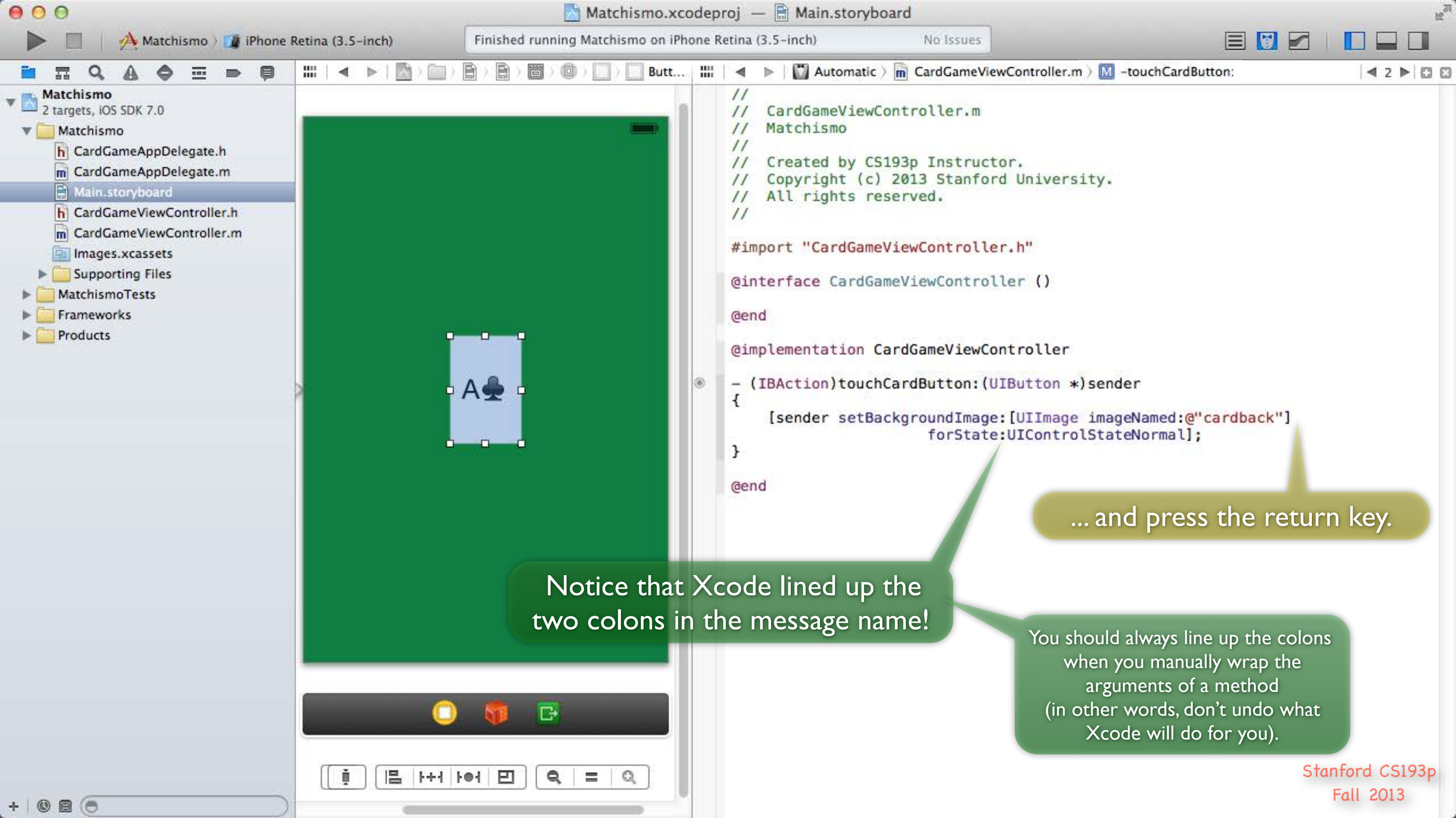
- (IBAction)touchCardButton:(UIButton *)sender
{
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:
        UIControlStateNormal];
}

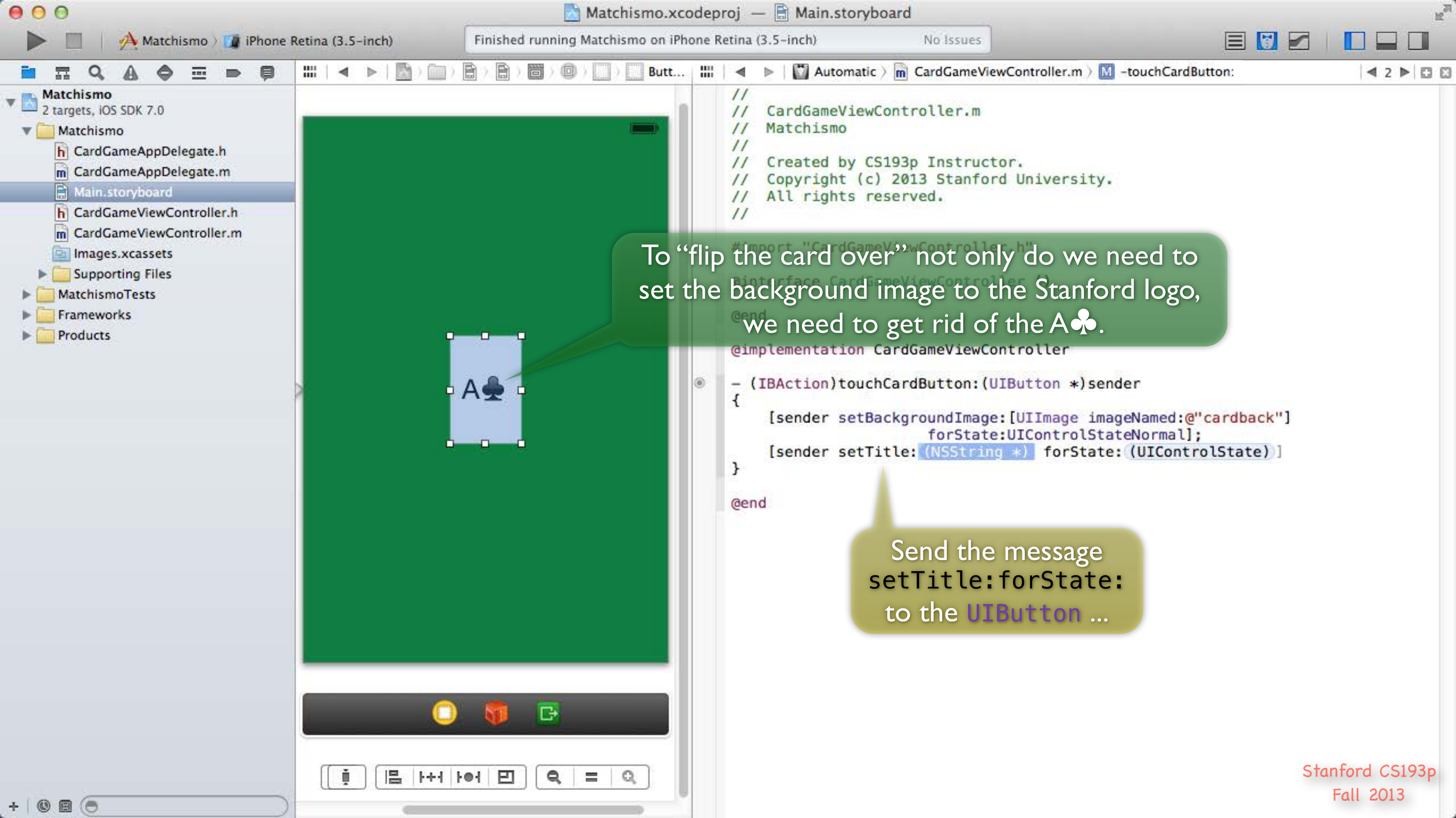
@end
```

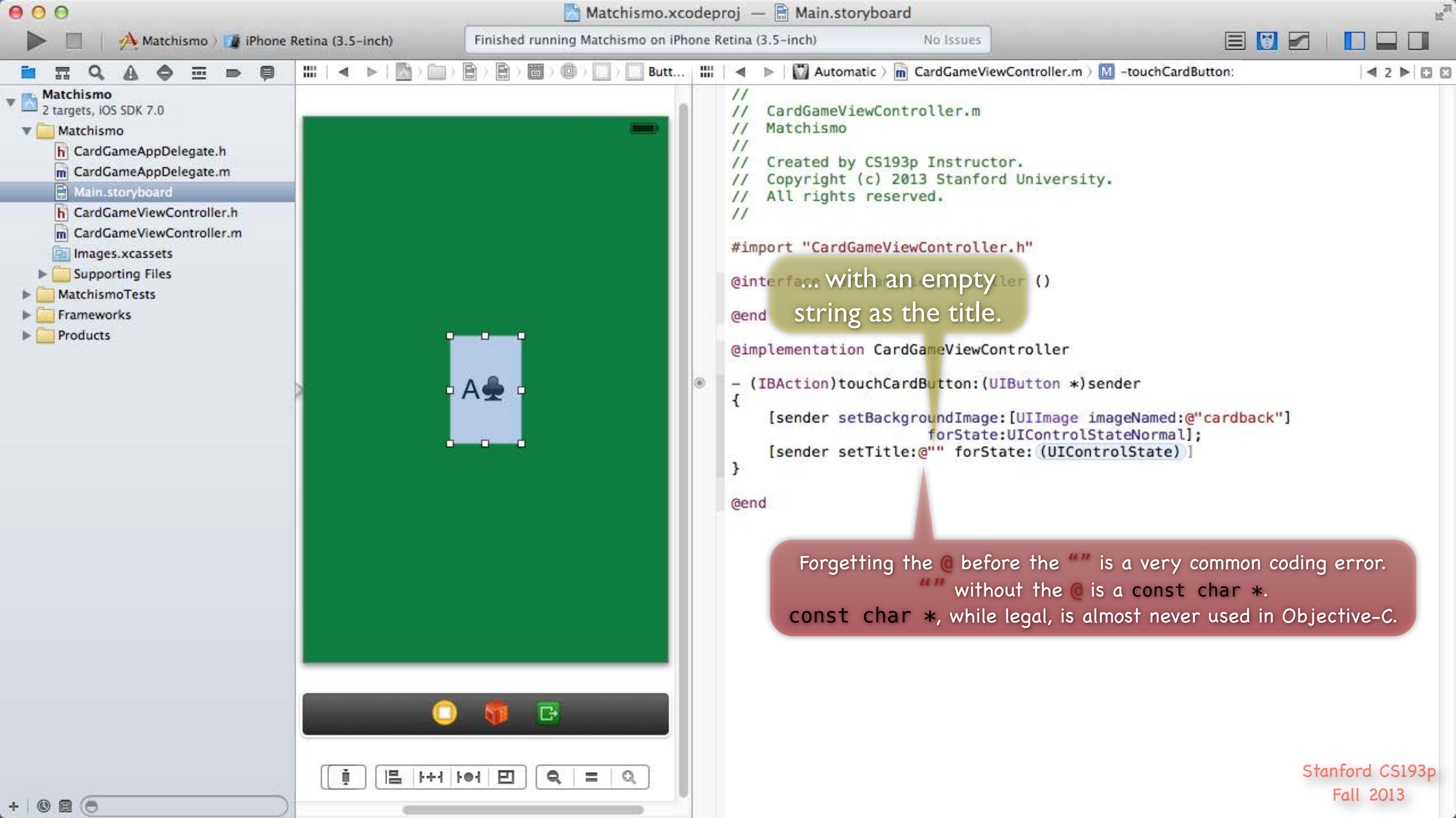
This line of code is so long that it is wrapping now. And not really in a great spot.

Put your cursor here ...

Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
}
@end
... and press the return key.
Notice that Xcode lined up the two colons in the message name!
You should always line up the colons when you manually wrap the arguments of a method (in other words, don't undo what Xcode will do for you).
Stanford CS193p
Fall 2013

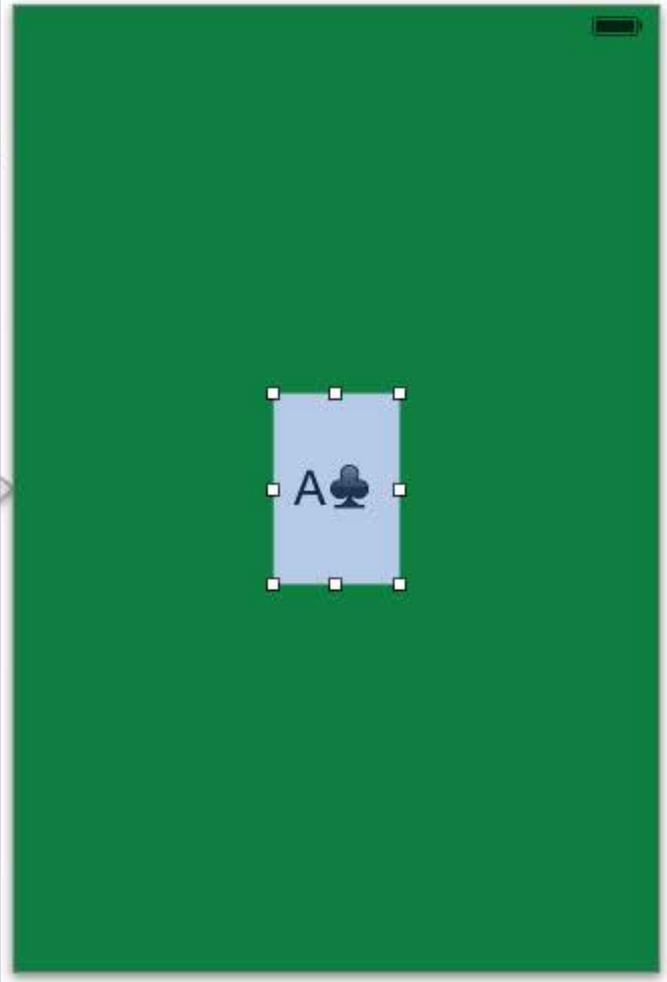

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
[sender setTitle:(NSString *) forState:(UIControlState)]
}
@end
To “flip the card over” not only do we need to set the background image to the Stanford logo, we need to get rid of the A♣.
Send the message setTitle:forState: to the UIButton ...
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
[sender setTitle:@"" forState:(UIControlState)];
}
@end
... with an empty string as the title.
Forgetting the @ before the "" is a very common coding error.
"" without the @ is a const char *.
const char *, while legal, is almost never used in Objective-C.
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

@end
```

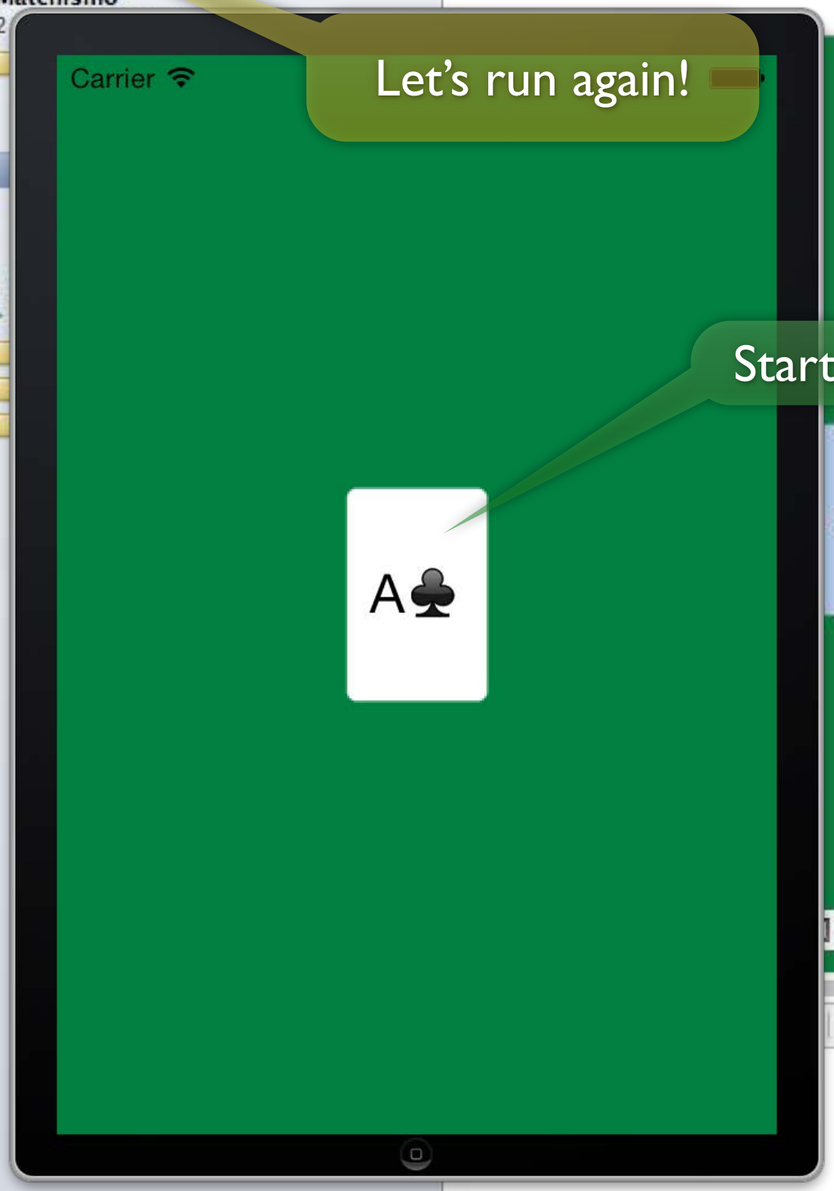
Again, we specify that we are setting this title for the default (Normal) button state.

Matchismo.xcodeproj — Main.storyboard

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

Let's run again!

Starts out the same as before ...



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()
@end

@implementation CardGameViewController

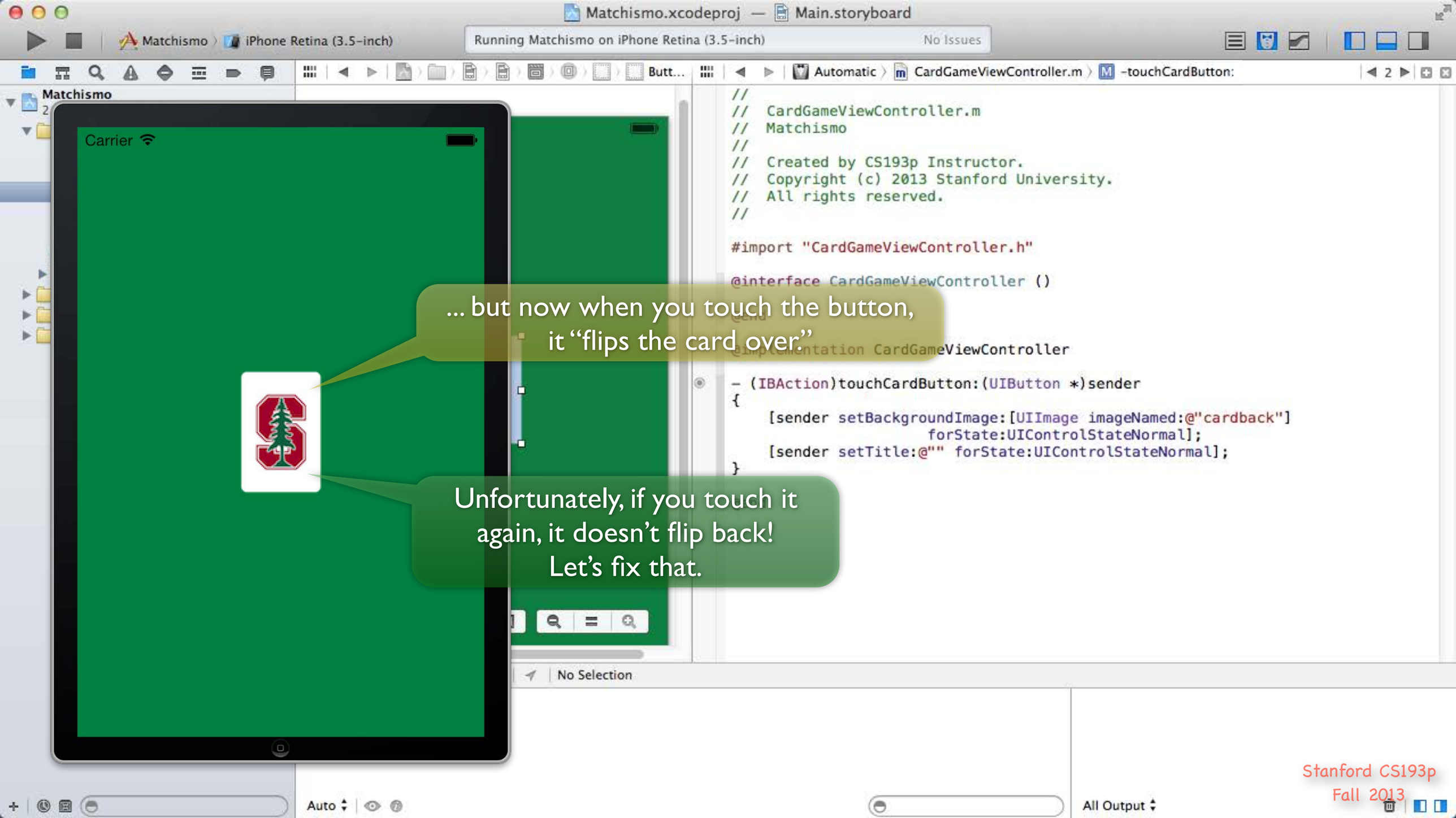
- (IBAction)touchCardButton:(UIButton *)sender
{
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

@end
```

No Selection

Auto | All Output

Stanford CS193p
Fall 2013

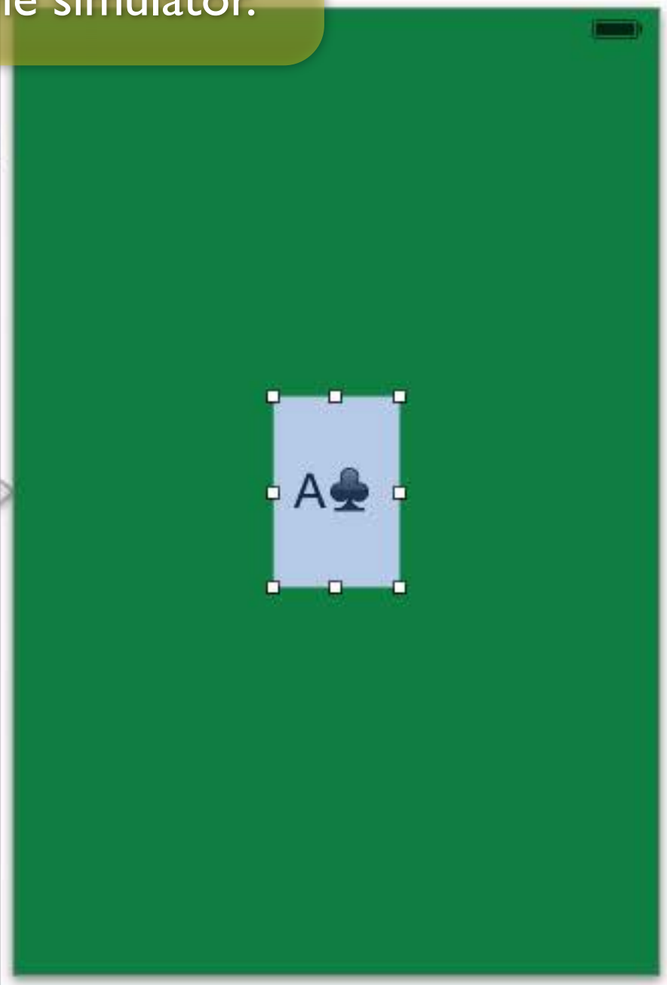

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
Carrier
... but now when you touch the button, it "flips the card over."
Unfortunately, if you touch it again, it doesn't flip back! Let's fix that.
No Selection
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
}
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 the simulator.

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

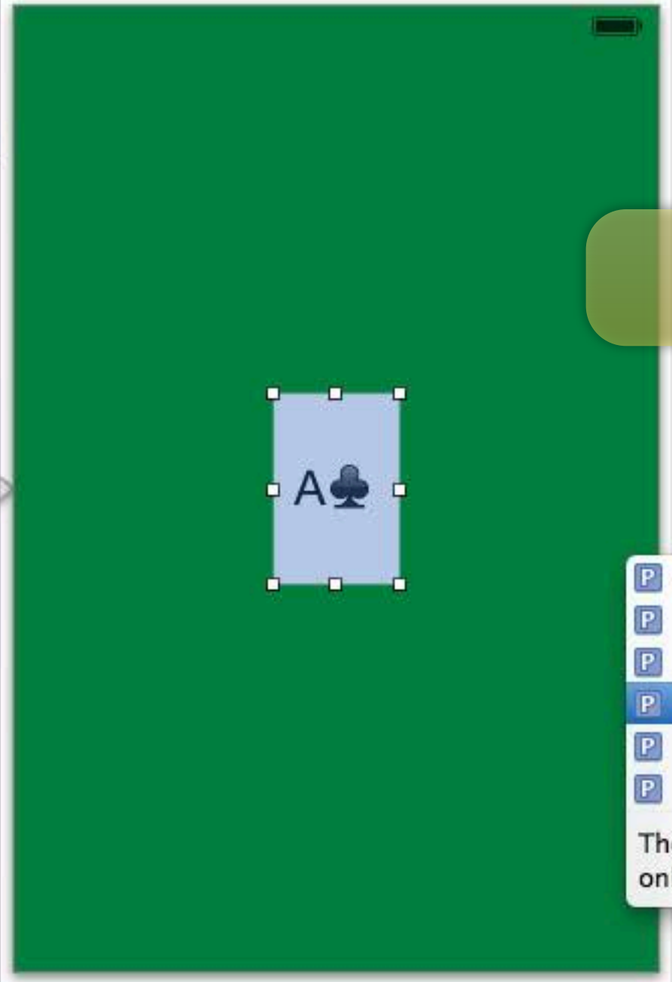
- (IBAction)touchCardButton:(UIButton *)sender
{
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

@end
```

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if (sender.currentTitle)
```

```
P NSAttributedString * currentAttributedTitle
P          UIImage * currentBackgroundImage
P          UIImage * currentImage
P         NSString * currentTitle
P          UIColor * currentTitleColor
P          UIColor * currentTitleShadowColor
```

The current title that is displayed on the button. (read-only) More...

If the sending button's currentTitle's ...

Notice that currentTitle is a @property of the button, so we are using dot notation here.

imageNamed:@"cardback"]
lStateNormal];
trolStateNormal];

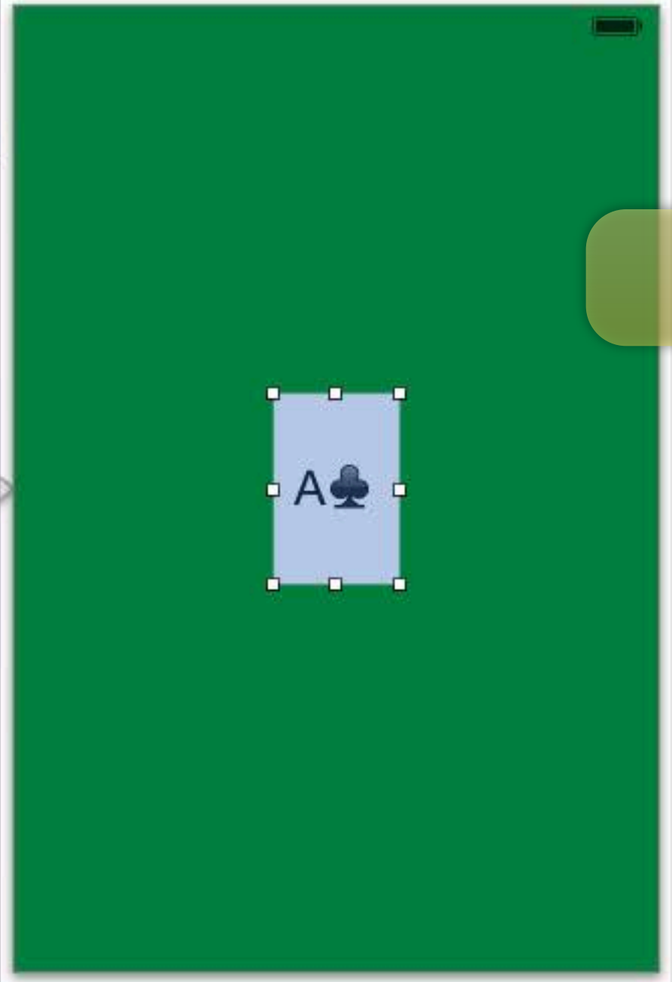
Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton: | 2

Matchismo
2 targets, iOS SDK 7.0

- Matchismo
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if (sender.currentTitle length)
    [sender se

    [sender se
}

@end
```

…length…

M NSUInteger length
M NSUInteger lengthOfBytesUsingEncoding:(NSStringEncoding)

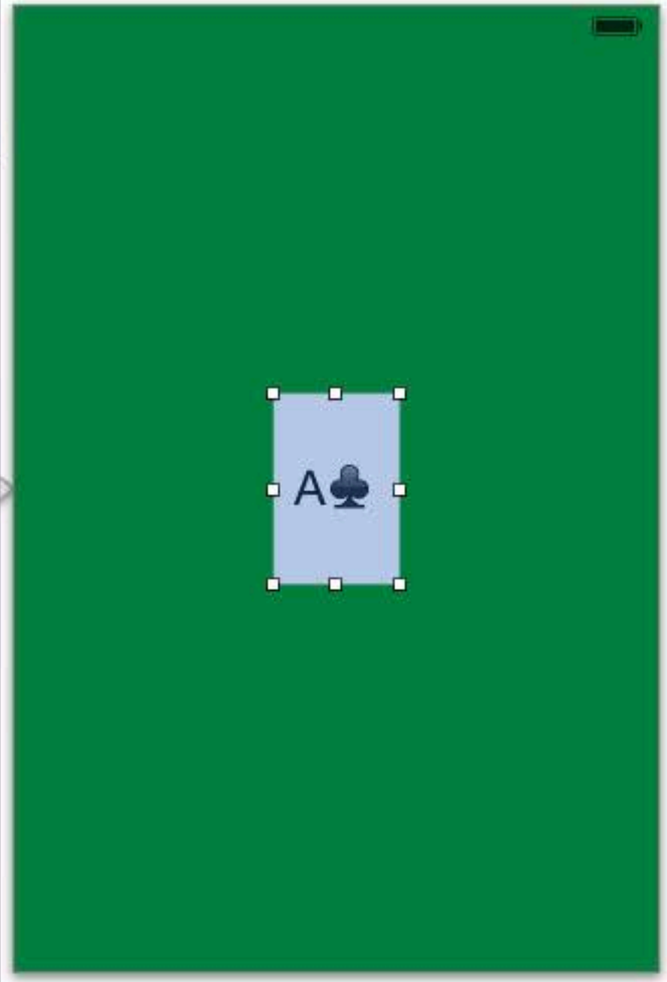
Returns the number of Unicode characters in the receiver. More…

Checking the `length` of an `NSString` to see if it is blank is cool because it works whether the string is the empty string (i.e. `@""`) or is `nil`.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length])
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

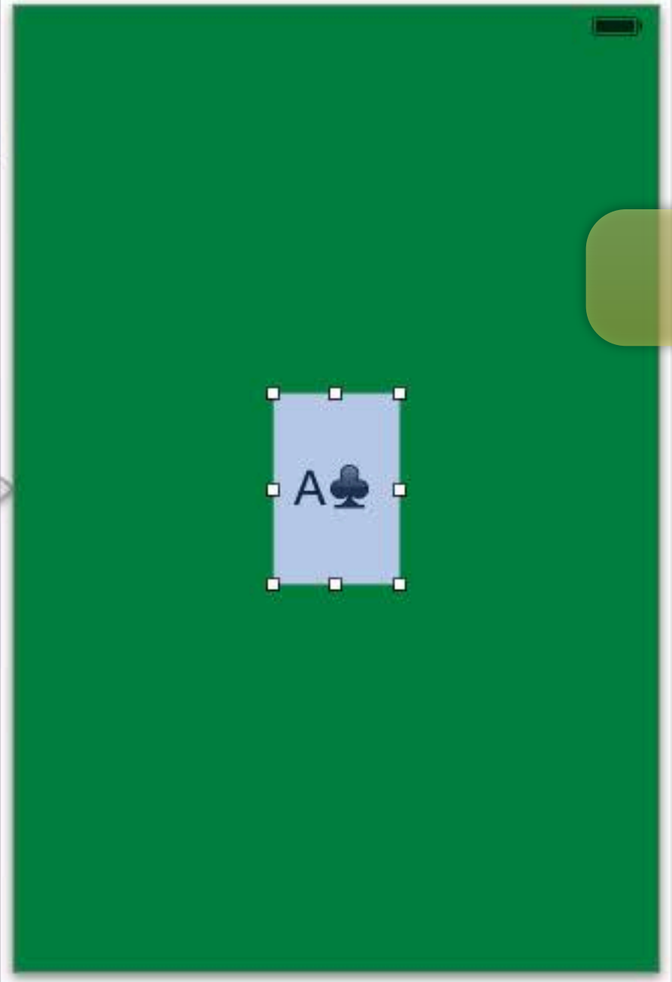
@end
```

... (note that we don't have to type the [,
since Xcode automatically adds it when we type]) ...

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {

    }
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

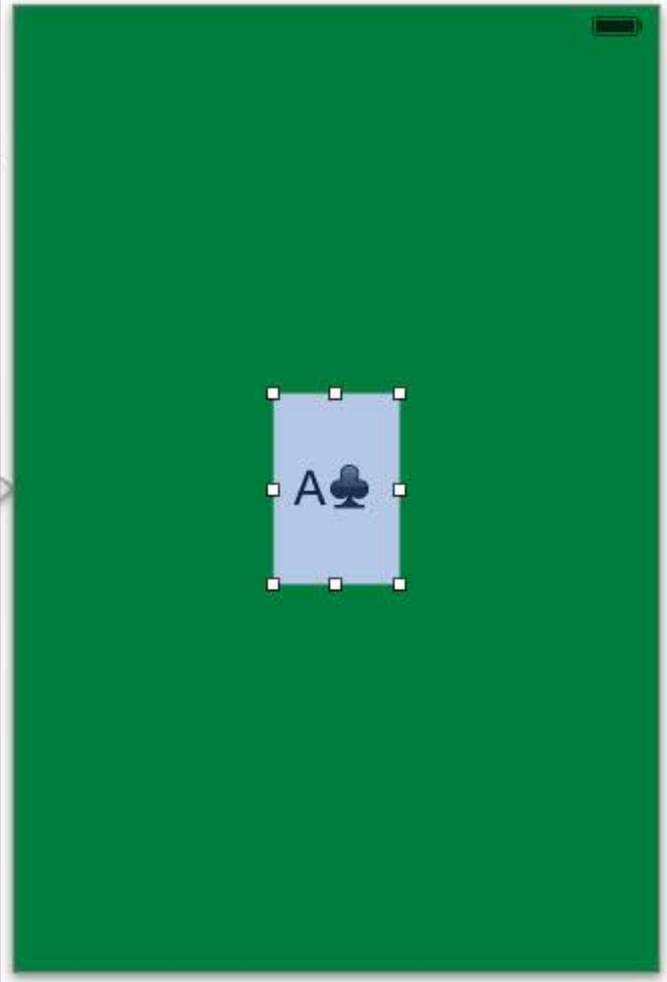
@end
```

... is non-zero (front is showing), then ...

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {

    }
    [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                      forState:UIControlStateNormal];
    [sender setTitle:@"" forState:UIControlStateNormal];
}

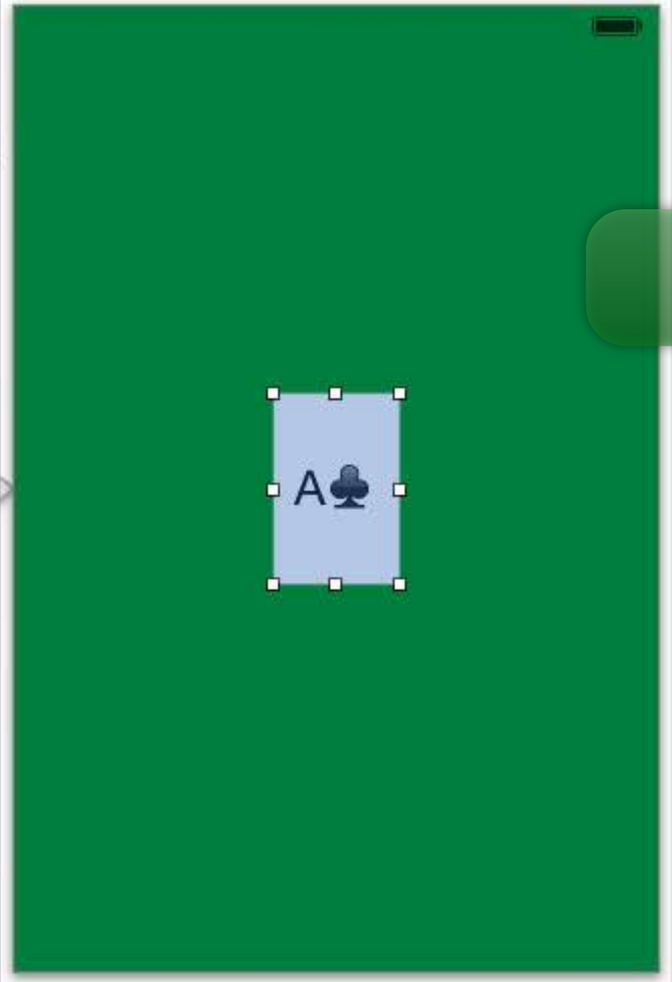
@end
```

Move this code inside the curly braces.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                          forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    }
}

@end
```

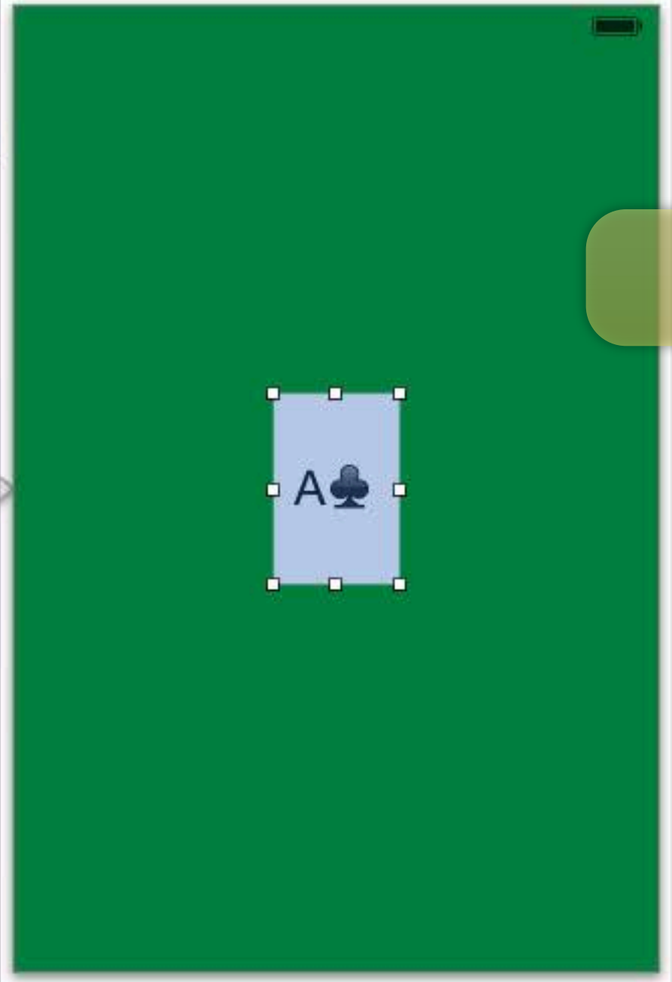
... flip the card over to show it's back ...

Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

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

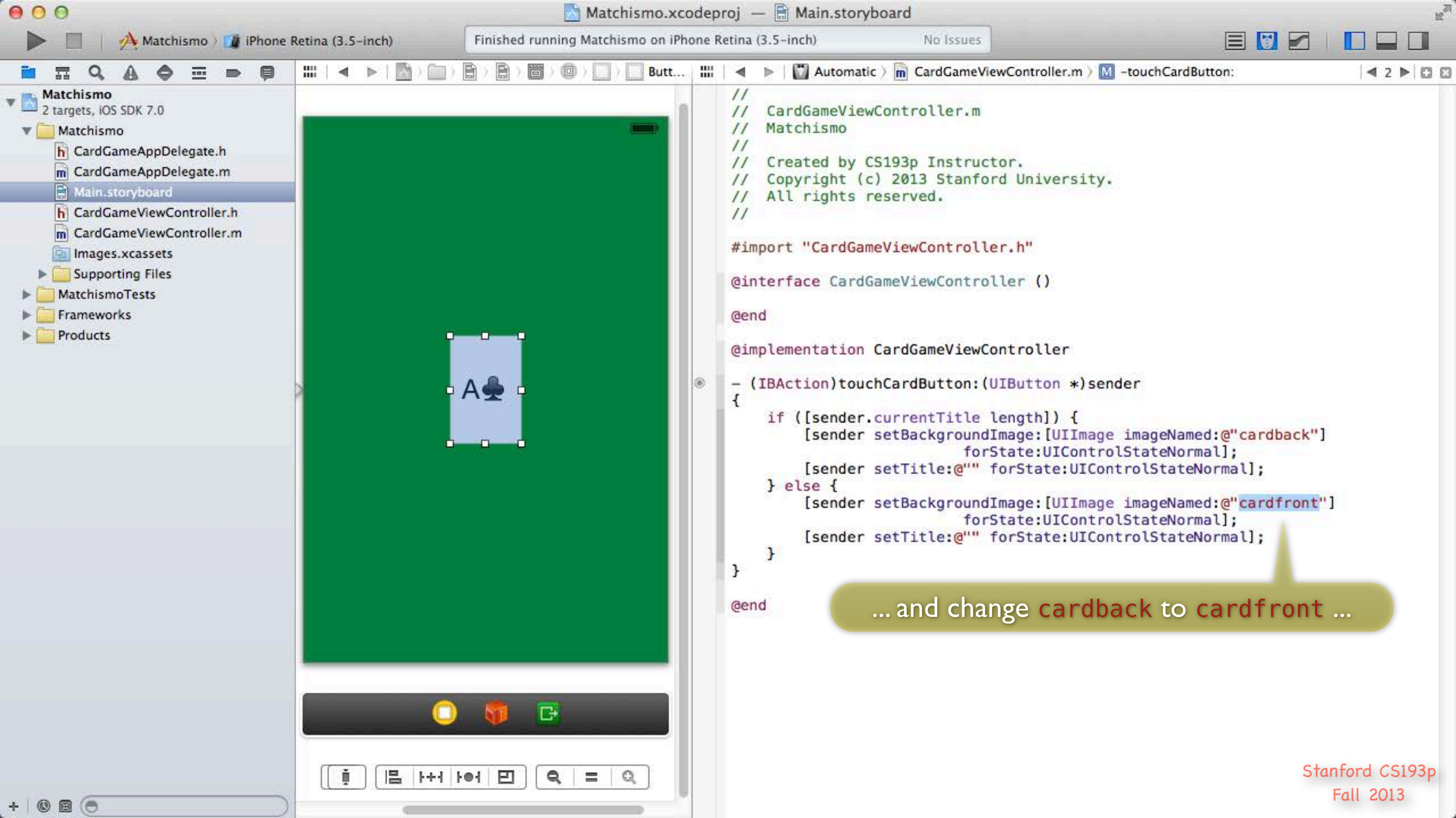
    }
}

@end
```

… else …

Copy/paste this code into the else …

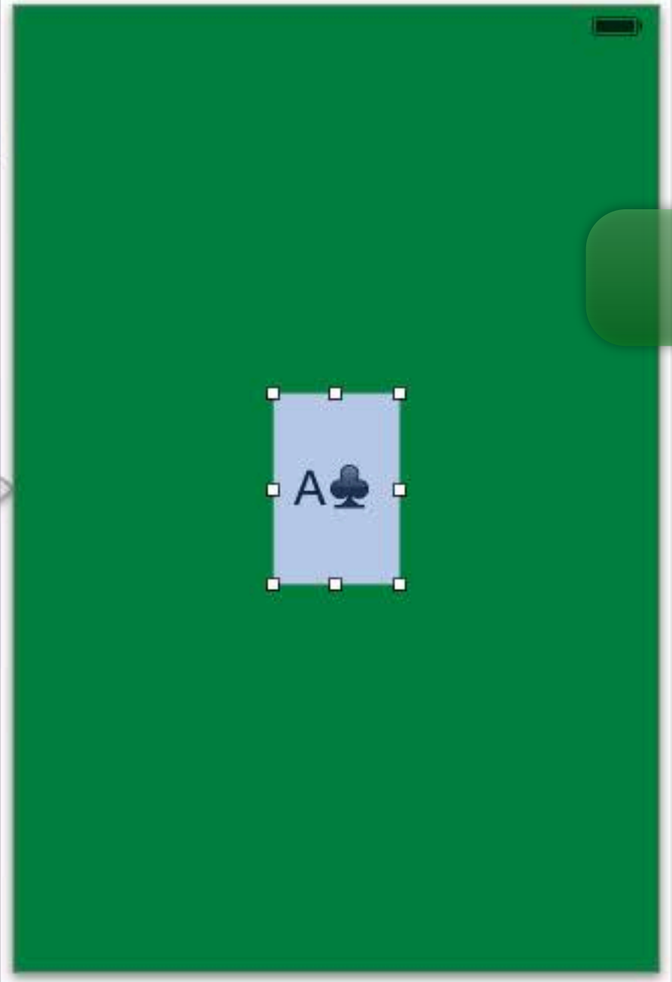
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Automatic › CardGameViewController.m › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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:@"" forState:UIControlStateNormal];
}
}
@end
... and change cardback to cardfront ...
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@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];
    }
}

@end
```

... flip back over to the A♣ (the front).

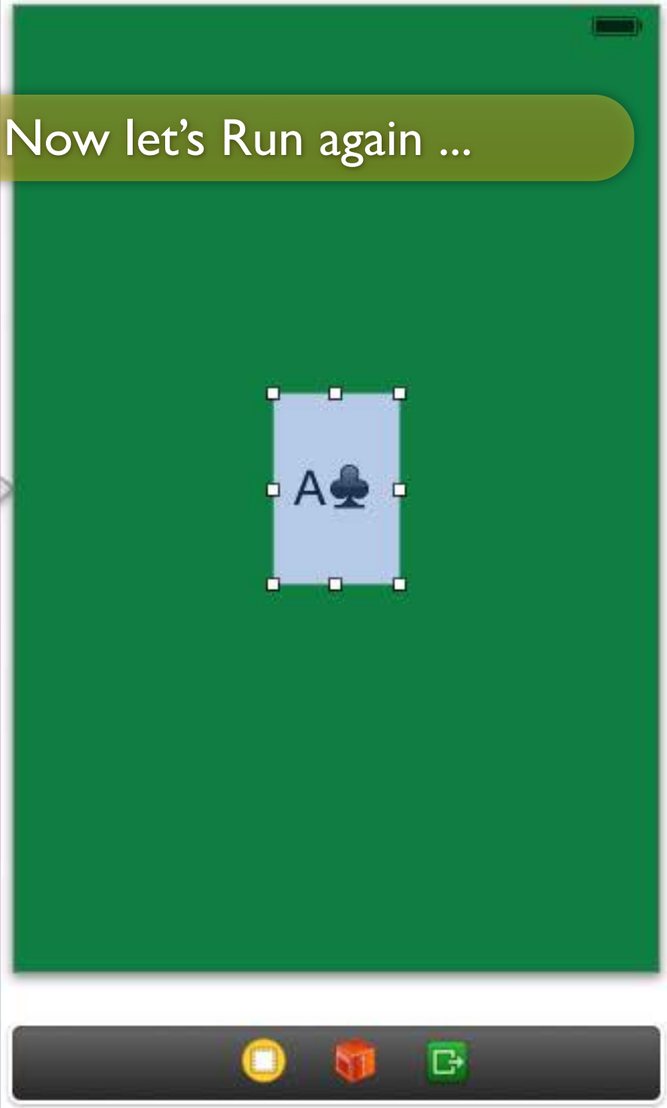
... and change @“” to @“A♣”.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Now let's Run again ...



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

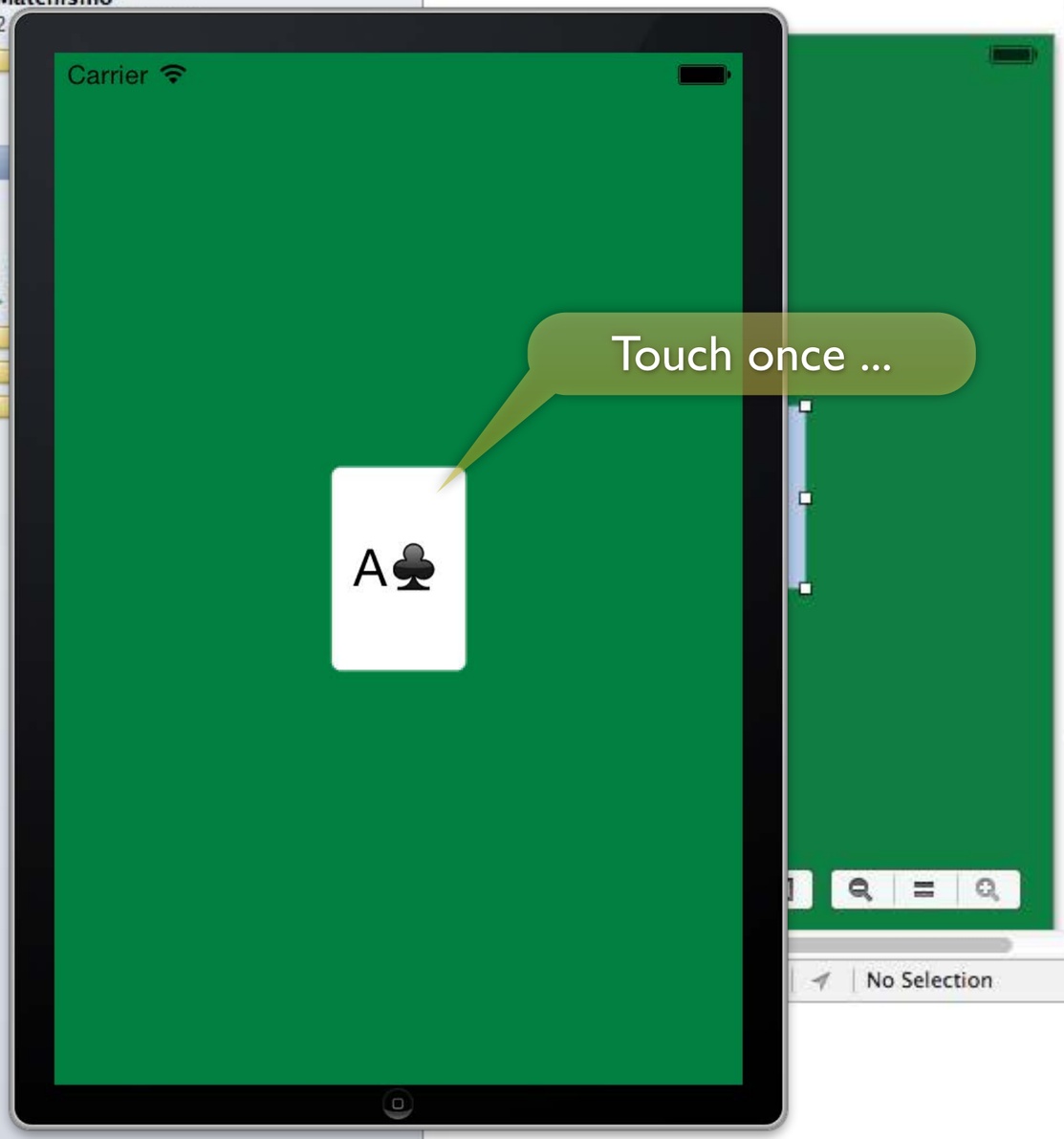
@interface CardGameViewController ()

@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];
    }
}

@end
```



No Selection

Auto

All Output

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

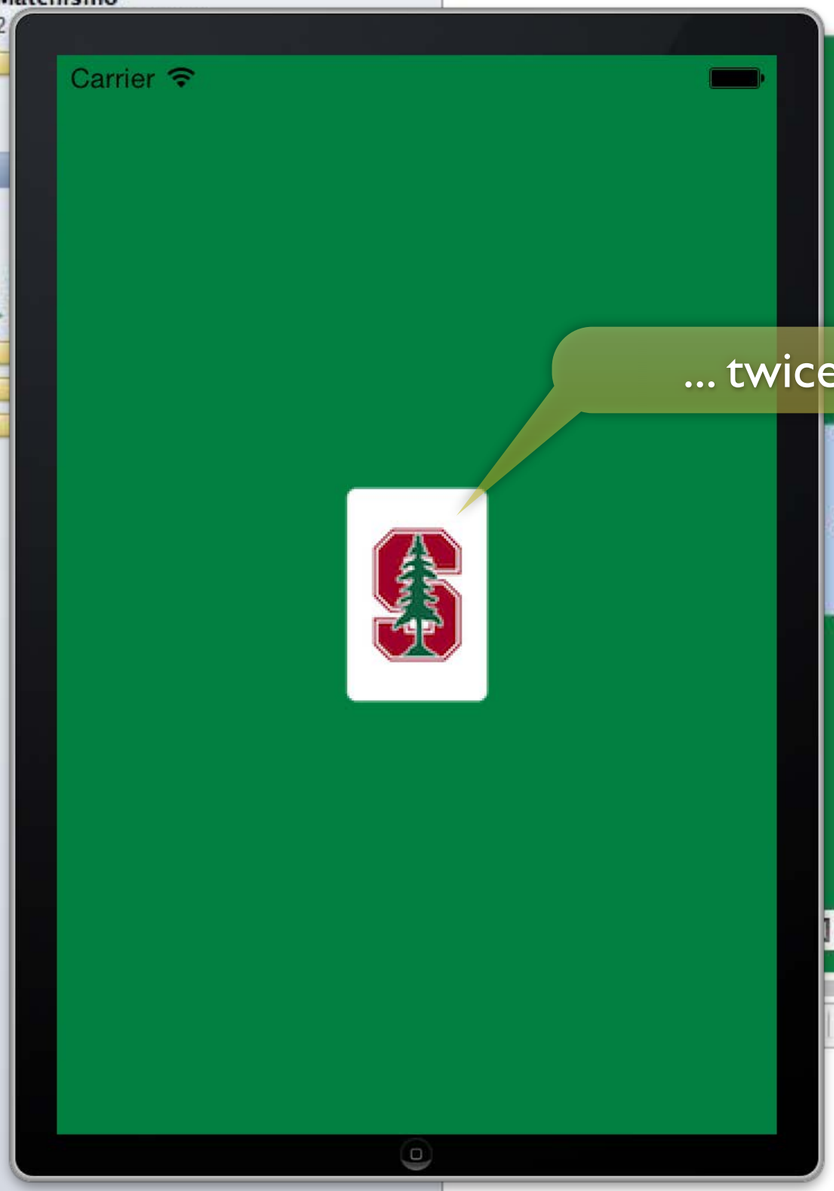
@interface CardGameViewController ()

@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];
    }
}

@end
```



No Selection

Auto | All Output

Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

Carrier

A♣

... three times.

Hopefully your application is working!

No Selection

Auto | All Output

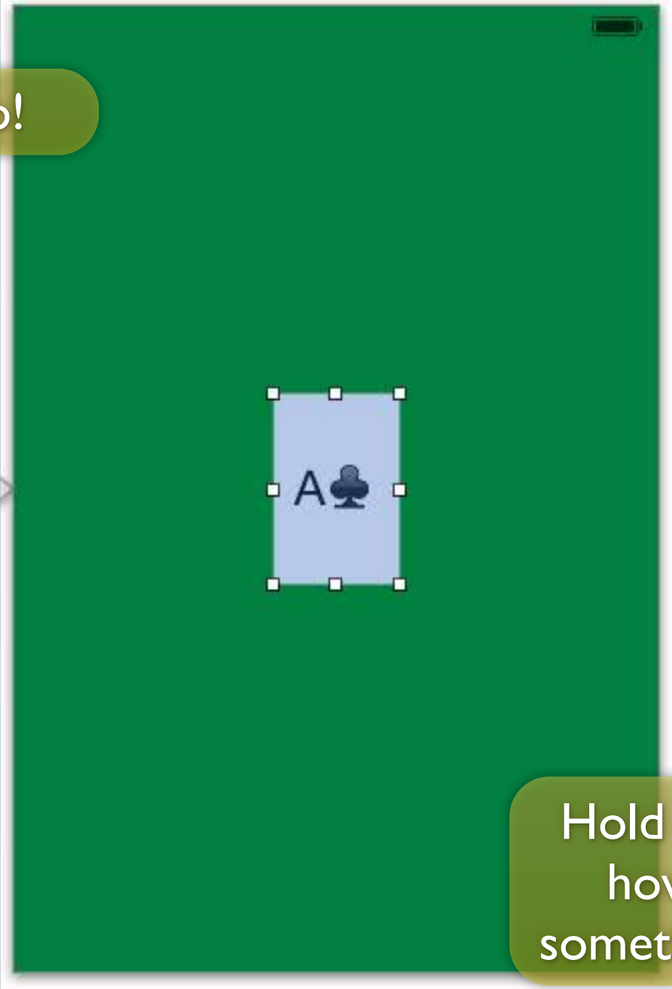
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Stop!



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}
@end
```

Let's take a little timeout to talk about documentation.
There are numerous ways to transition to the documentation, but an easy one is to use the ALT key.

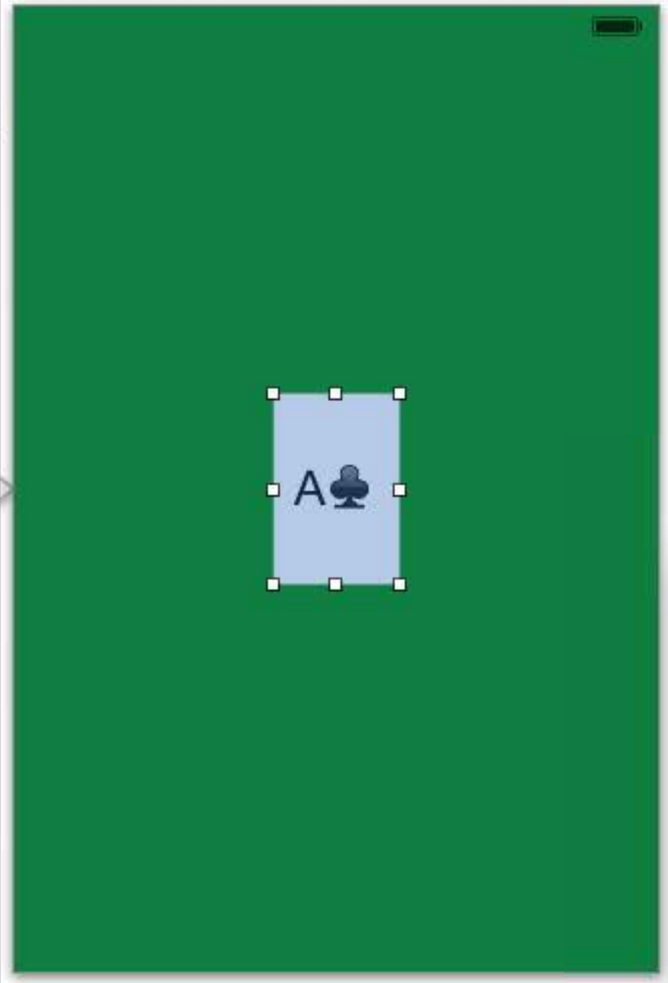
Hold down the ALT key and hover your mouse over something like `currentTitle`.

A dashed line should appear underneath and the cursor should be a question mark.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

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

ALT-clicking on `currentTitle` will bring up this "mini-documentation" in-line.

Declaration `@property(nonatomic, readonly, retain) NSString * currentTitle`

Description The current title that is displayed on the button. (read-only)

The value for this property is set automatically whenever the button state changes. For states that do not have a custom title string associated with them, this method returns the title that is currently displayed, which is typically the one associated with the UIControlStateNormal state. The value may be nil.

Availability iOS (2.0 and later)

Declared In UIButton.h

Reference UIButton Class Reference

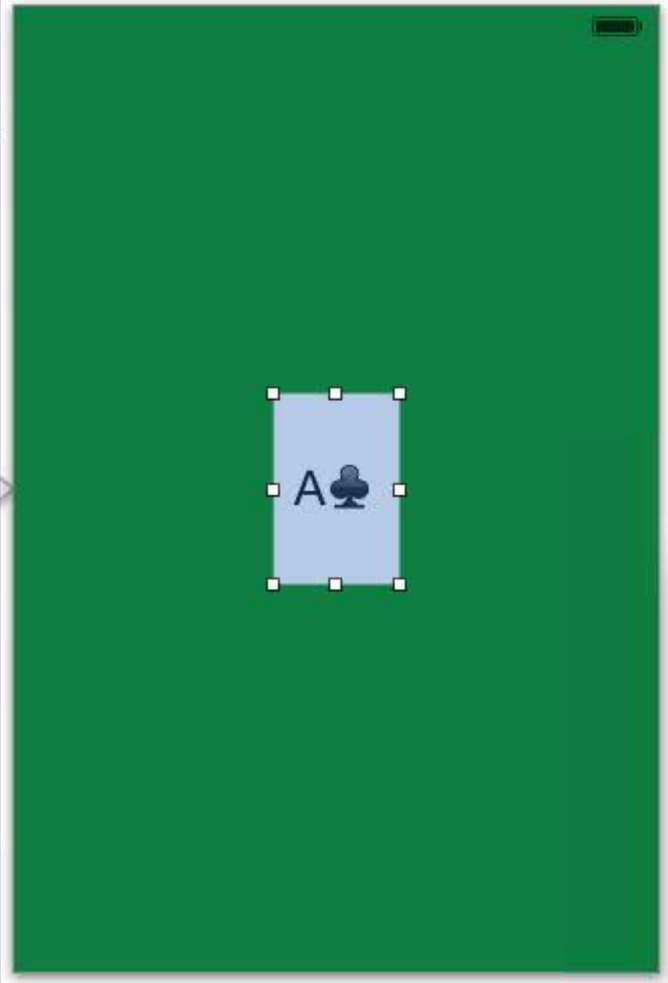
`retain` is the same as `strong`.
We'll cover the `readonly` directive next lecture.

Matchismo.xcodeproj — Main.storyboard

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

TIP: If you ever accidentally navigate away from your source code, you can click on this back button to get back.

Matchismo
2 targets, iOS SDK 7.0
- Matchismo
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



Automatic › CardGameViewController.m › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                          forState:UIControlStateNormal];
        sender.currentTitle = @"";
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
                          forState:UIControlStateNormal];
        sender.currentTitle = [self.game cardAtIndex:[self.cardButtons indexOfObject:sender]].contents;
    }
}
```

Click on the UIButton Class Reference link in this little "mini-documentation" window to get more detailed documentation.

You can also hold down the ALT key and double-click on a term go directly to the documentation.

Declaration: @property(nonatomic, readonly, retain) NSString * currentTitle

Description: The current title that is displayed on the button. (read-only)

The value for this property is set automatically whenever the button state changes. For states that do not have a custom title string associated with them, this method returns the title that is currently displayed, which is typically the one associated with the UIControlStateNormal state. The value may be nil.

Availability: iOS (2.0 and later)

Declared In: UIButton.h

Reference: UIButton Class Reference

Documentation — UIButton Class Reference

Search documentation

This is the Documentation window.

Overview

Important: This is a preliminary document for an API or technology in development. Although this document has been reviewed for technical accuracy, it is not final. This Apple confidential information is for use only by registered members of the applicable Apple Developer program. Apple is supplying this confidential information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future seeds of the API or technology.

An instance of the `UIButton` class implements a button on the touch screen. A button intercepts touch events and sends an action message to a target object when tapped. Methods for setting the target and action are inherited from `UIControl`. This class provides methods for setting the title, image, and other appearance properties of a button. By using these accessors, you can specify a different appearance for each button state.

For information about basic view behaviors, see *View Programming Guide for iOS*.

For more information about appearance and behavior configuration, see "Buttons".

Tasks

Creating Buttons

+ `buttonWithType:`

Configuring the Button Title

`titleLabel` *property*
`reversesTitleShadowWhenHighlighted` *property*
- `setTitle:forState:`
- `setAttributedTitle:forState:`
- `setTitleColor:forState:`
- `setTitleShadowColor:forState:`
- `titleColorForState:`
- `titleForState:`
- `attributedTitleForState:`
- `titleShadowColorForState:`

Configuring Button Presentation

You should explore what is here.
It is substantial.
Being able to maneuver through the documentation is critical to success in iOS Development.

Provide Feedback

- UIButton Class Reference
  - Overview
  - Tasks
    - Creating Buttons
    - Configuring the Button Title
    - Configuring Button Presentation
    - Configuring Edge Insets
    - Getting the Current State
    - Getting Dimensions
    - Deprecated Properties
  - Properties
    - adjustsImageWhenDisabled
    - adjustsImageWhenHighlighted
    - buttonType
    - contentEdgeInsets
    - currentAttributedTitle
    - currentBackgroundImage
    - currentImage
    - currentTitle
    - currentTitleColor
    - currentTitleShadowColor
    - imageEdgeInsets
    - imageView
    - reversesTitleShadowWhenHighlighted
    - showsTouchWhenHighlighted
    - tintColor
    - titleEdgeInsets
    - titleLabel
  - Class Methods
    - buttonWithType:
  - Instance Methods
    - attributedTitleForState:
    - backgroundImageForState:
    - backgroundRectForBounds:
    - contentRectForBounds:
    - imageForState:

Available in iOS 6.0 and later.

Declared In

`UIButton.h`

For example, scroll down to `setBackgroundImage:forState:`.

setBackgroundImage:forState:

Sets the background image to use for the specified button state.

```
- (void)setBackgroundImage:(UIImage *)image forState:(UIControlState)state
```

Parameters

image

The background image to use for the specified state.

state

The state that uses the specified image. The values are described in `UIControlState`.

Discussion

In general, if a property is not specified for a state, the default is to use the `UIControlStateNormal` value. If the `UIControlStateNormal` value is not set, then the property defaults to a system value. Therefore, at a minimum, you should set the value for the normal state.

Availability

Available in iOS 2.0 and later.

See Also

– `backgroundImageForState:`

Related Sample Code

Accessory
AddMusic
UICatalog

Declared In

`UIButton.h`

You can click on the many links here, like `UIImage` ...

setImage:forState:

Sets the image to use for the specified state.

```
- (void)setImage:(UIImage *)image forState:(UIControlState)state
```

Parameters

UIButton Class Reference
Overview
Tasks
Creating Buttons
Configuring the Button Title
Configuring Button Presentation
Configuring Edge Insets
Getting the Current State
Getting Dimensions
Deprecated Properties
Properties
adjustsImageWhenDisabled
adjustsImageWhenHighlighted
buttonType
contentEdgeInsets
currentAttributedTitle
currentBackgroundImage
currentImage
currentTitle
currentTitleColor
currentTitleShadowColor
imageEdgeInsets
imageView
reversesTitleShadowWhenHighlighted
showsTouchWhenHighlighted
tintColor
titleEdgeInsets
titleLabel
Class Methods
buttonWithType:
Instance Methods
attributedTitleForState:
backgroundImageForState:
backgroundRectForBounds:
contentRectForBounds:
imageForState:

Provide Feedback

Next

UIImage Class Reference

And get detailed class overviews.

Overview

Important: This is a preliminary document for an API or technology in development. Although this document has been reviewed for technical accuracy, it is not final. This Apple confidential information is for use only by registered members of the applicable Apple Developer program. Apple is supplying this confidential information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future seeds of the API or technology.

A `UIImage` object is a high-level way to display image data. You can create images from files, from Quartz image objects, or from raw image data you receive. The `UIImage` class also offers several options for drawing images to the current graphics context using different blend modes and opacity values.

Image objects are immutable, so you cannot change their properties after creation. This means that you generally specify an image's properties at initialization time or rely on the image's metadata to provide the property value. It also means that image objects are themselves safe to use from any thread. The way you change the properties of an existing image object is to use one of the available convenience methods to create a copy of the image but with the custom value you want.

Because image objects are immutable, they also do not provide direct access to their underlying image data. However, you can get an `NSData` object containing either a PNG or JPEG representation of the image data using the `UIImagePNGRepresentation` and `UIImageJPEGRepresentation` functions.

The system uses image objects to represent still pictures taken with the camera on supported devices. To take a picture, use the `UIImagePickerController` class. To save a picture to the Saved Photos album, use the `UIImageWriteToSavedPhotosAlbum` function.

Images and Memory Management

In low-memory situations, image data may be purged from a `UIImage` object to free up memory on the system. This purging behavior affects only the image data stored internally by the `UIImage` object and not the object itself. When you attempt to draw an image whose data has been purged, the image object automatically reloads the data from its original file. This extra load step, however, may incur a small performance penalty.

You should avoid creating `UIImage` objects that are greater than 1024 x 1024 in size. Besides the large amount of memory such an image would consume, you may run into problems when using the image as a texture in OpenGL ES or when drawing the image to a view or layer. This size restriction does not apply if you are performing code-based manipulations, such as resizing an image larger than 1024 x 1024 pixels by d
to a bitmap-backed graphics context. In fact, you may need to resize an image in this manner (or break it into several smaller images) in order to

Provide Feedback

UIImage Class Reference
- Overview
  - Images and Memory Management
  - Supported Image Formats
- Tasks
  - Cached Image Loading Routines
  - Creating New Images
  - Initializing Images
  - Image Attributes
  - Drawing Images
- Properties
  - alignmentRectInsets
  - capInsets
  - CGImage
  - CIImage
  - duration
  - imageOrientation
  - images
  - renderingMode
  - resizingMode
  - scale
  - size
- Class Methods
  - animatedImageNamed:duration:
  - animatedImageWithImages:duration:
  - animatedResizableImageNamed:ca...
  - animatedResizableImageNamed:ca...
  - imageNamed:
  - imageWithCGImage:
  - imageWithCGImage:scale:orientation:
  - imageWithCIImage:
  - imageWithCIImage:scale:orientation:
  - imageWithContentsOfFile:
  - imageWithData:
  - imageWithData:scale:
- Instance Methods

NSString

Top Hit
- NSString

API Reference
- Appendix A: Deprecated NSString Methods — NSString Class Reference
- Appendix A: Deprecated NSString UIKit Additions Methods — NSString UIKit Additions Reference
- CFStringConvertEncodingToNSStringEncoding
- CFStringConvertNSStringEncodingToEncoding
- NSString

SDK Guides
- NSString — Appendix A: Old-Style ASCII Property...
- Strings Are Represented b...ances of the NSString Class — Objects Can Represent Primitive Values
- NSString from C Strings and Data — Creating Strings
- Technical Q&A QA1235 — Converting to Precomposed Unicode

Sample Code
- Birthdays — IDEs
- LLDBCustomDataFormatter — Languages & Utilities
- Show All Results

You can also search for classes, methods, or just general topics of interest.

Supported

Table 1 lists th

Table 1 Suppo

Format	
Tagged Image	
Joint Photogra	
Graphic Interc	
Portable Netw	
Windows Bitm	
Windows Icon	
Windows Cursor	.cur
XWindow bitmap	.xbm

Note: Windows Bitmap Format (BMP) files that are formatted as RGB-565 are converted to ARGB-1555 when they are loaded.

Tasks

Cached Image Loading Routines

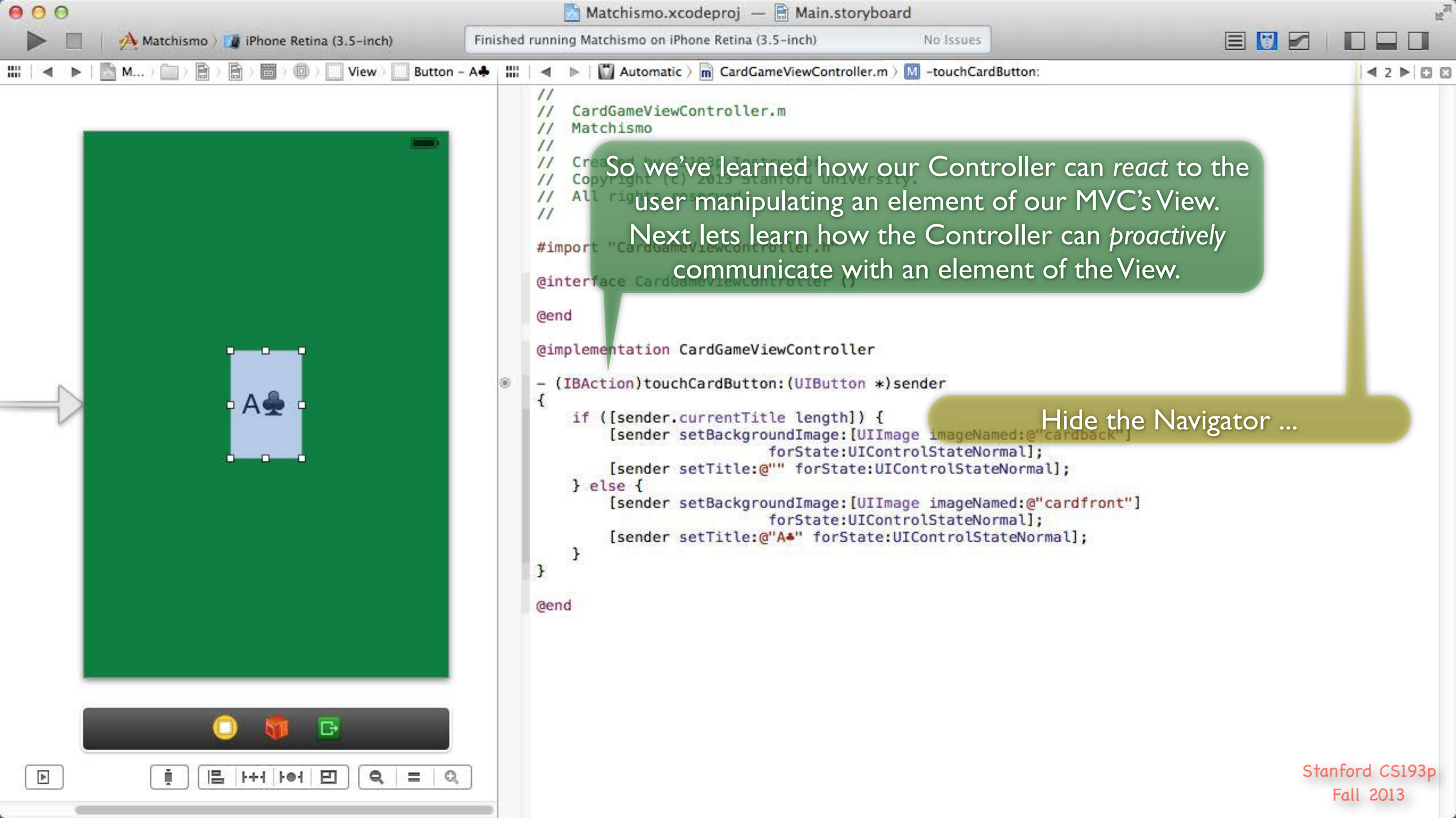
+ imageNamed:

Creating New Images

+ imageWithContentsOfFile:

- UIImage Class Reference
  - Overview
    - Images and Memory Management
    - Supported Image Formats
  - Tasks
    - Cached Image Loading Routines
    - Creating New Images
    - Initializing Images
    - Image Attributes
    - Drawing Images
  - Properties
    - alignmentRectInsets
    - capInsets
    - CGImage
    - CIImage
    - duration
    - imageOrientation
    - images
    - renderingMode
    - resizingMode
    - scale
    - size
  - Class Methods
    - animatedImageNamed:duration:
    - animatedImageWithImages:duration:
    - animatedResizableImageNamed:ca...
    - animatedResizableImageNamed:ca...
    - imageNamed:
    - imageWithCGImage:
    - imageWithCGImage:scale:orientation:
    - imageWithCIImage:
    - imageWithCIImage:scale:orientation:
    - imageWithContentsOfFile:
    - imageWithData:
    - imageWithData:scale:
  - Instance Methods

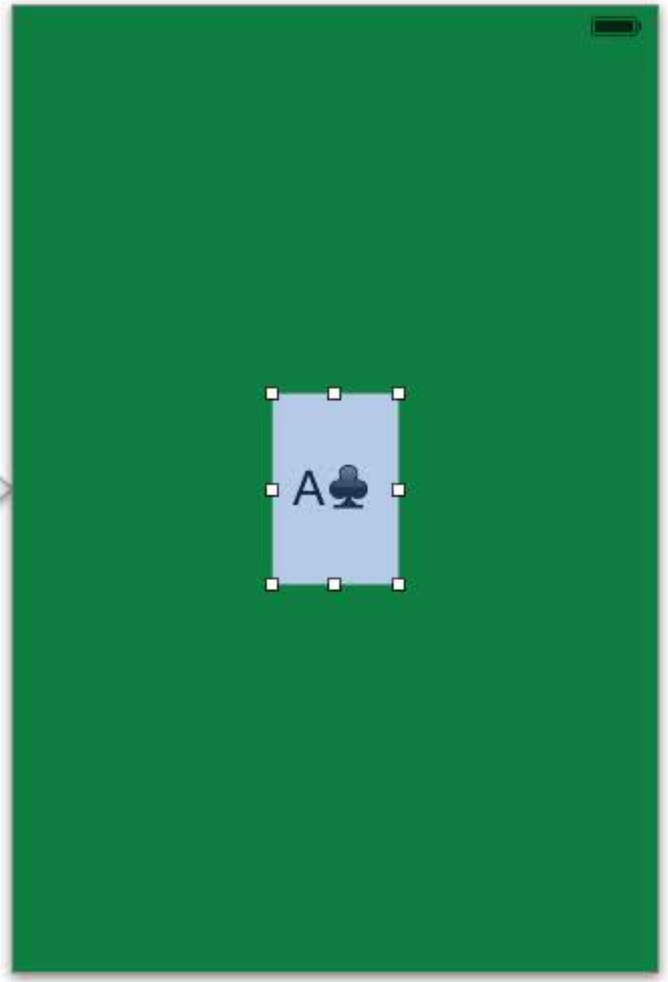
Provide Feedback


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
View › Button – A♣
Automatic › CardGameViewController.m › -touchCardButton:
A♣
//
// CardGameViewController.m
// Matchismo
//
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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];
}
}
@end
So we've learned how our Controller can react to the user manipulating an element of our MVC's View.
Next lets learn how the Controller can proactively communicate with an element of the View.
Hide the Navigator ...

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: | 2




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

Button

Type: System

State Config: Default

Title: Plain

A♣

Font: System 24.0

Text Color: Black Color

Shadow Color: Default

Image: Default Image

Background: cardfront

Shadow Offset: 0.0 Width, 0.0 Height

Reverses On Highlight

Highlighted Adjusts Image

Disabled Adjusts Image

Line Break: Truncate Middle

Edge: Content

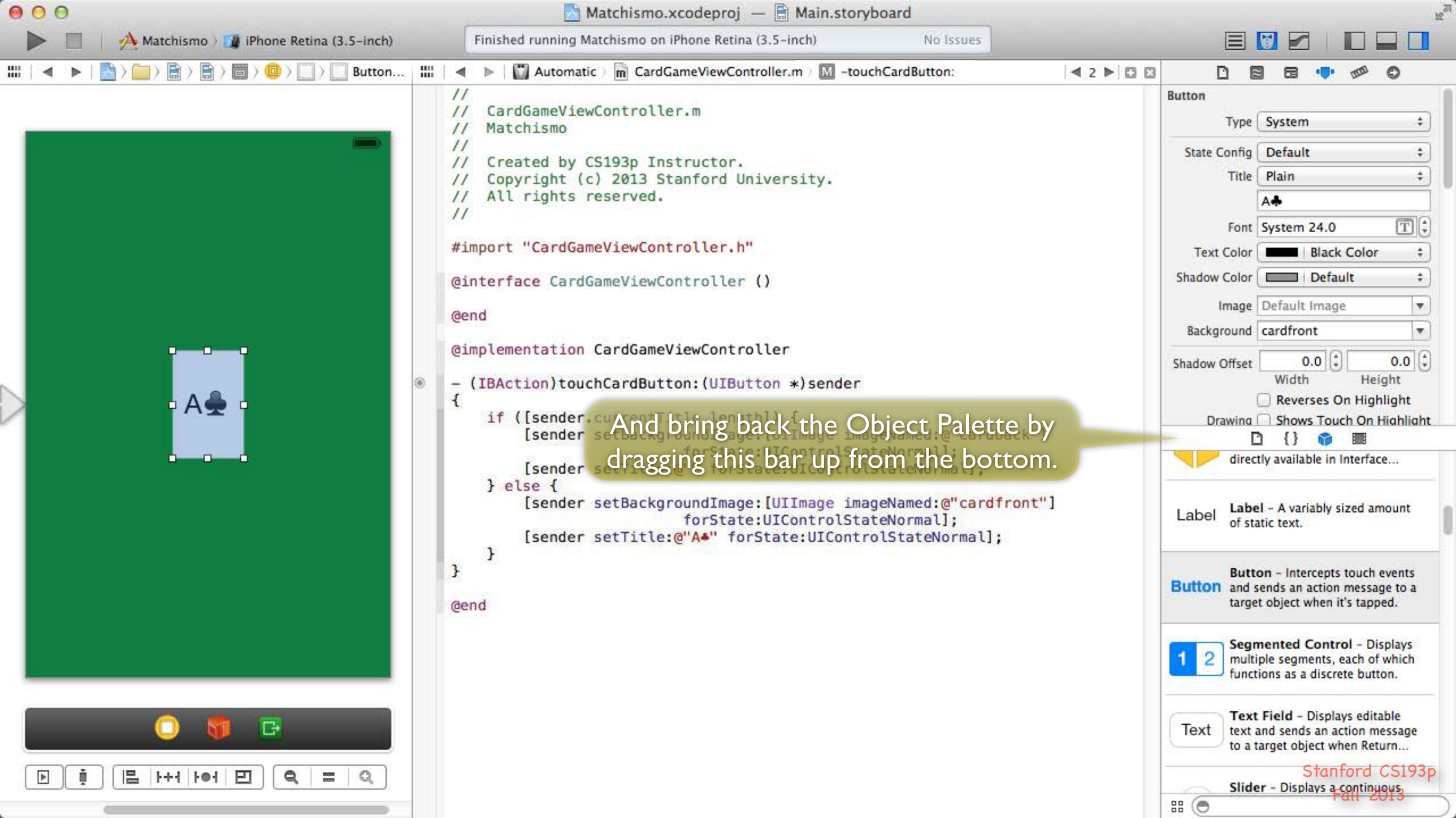
Inset: 0 Top, 0 Bottom, 0 Left, 0 Right

... and bring back the Utilities area.

Control

Alignment: Horizontal, Vertical

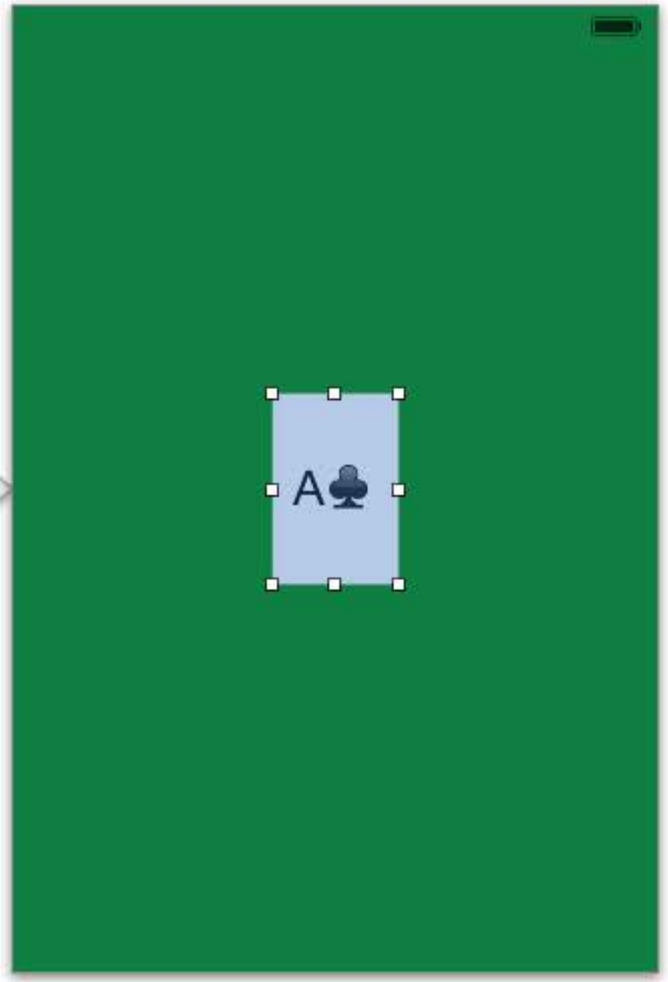
Content: Selected, Enabled, Highlighted


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Automatic › CardGameViewController.m › -touchCardButton:
A♣
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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];
}
}
@end
And bring back the Object Palette by dragging this bar up from the bottom.
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background cardfront
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Shows Touch On Highlight
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
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

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: | 2




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

@interface CardGameViewController ()

@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];
    }
}

@end
```

We're going to have a bit of text in our UI which shows how many times we've flipped the card.
A `UILabel` is the UIKit class we want
(it displays small bits of uneditable text).

Find `UILabel` in the Object Palette
(it's right above `UIButton`).

Button

Type: System
State Config: Default
Title: Plain
A♣
Font: System 24.0
Text Color: Black Color
Shadow Color: Default
Image: Default Image
Background: cardfront
Shadow Offset: 0.0 Width, 0.0 Height
Reverses On Highlight
Drawing: Shows Touch On Highlight

directly available in Interface…

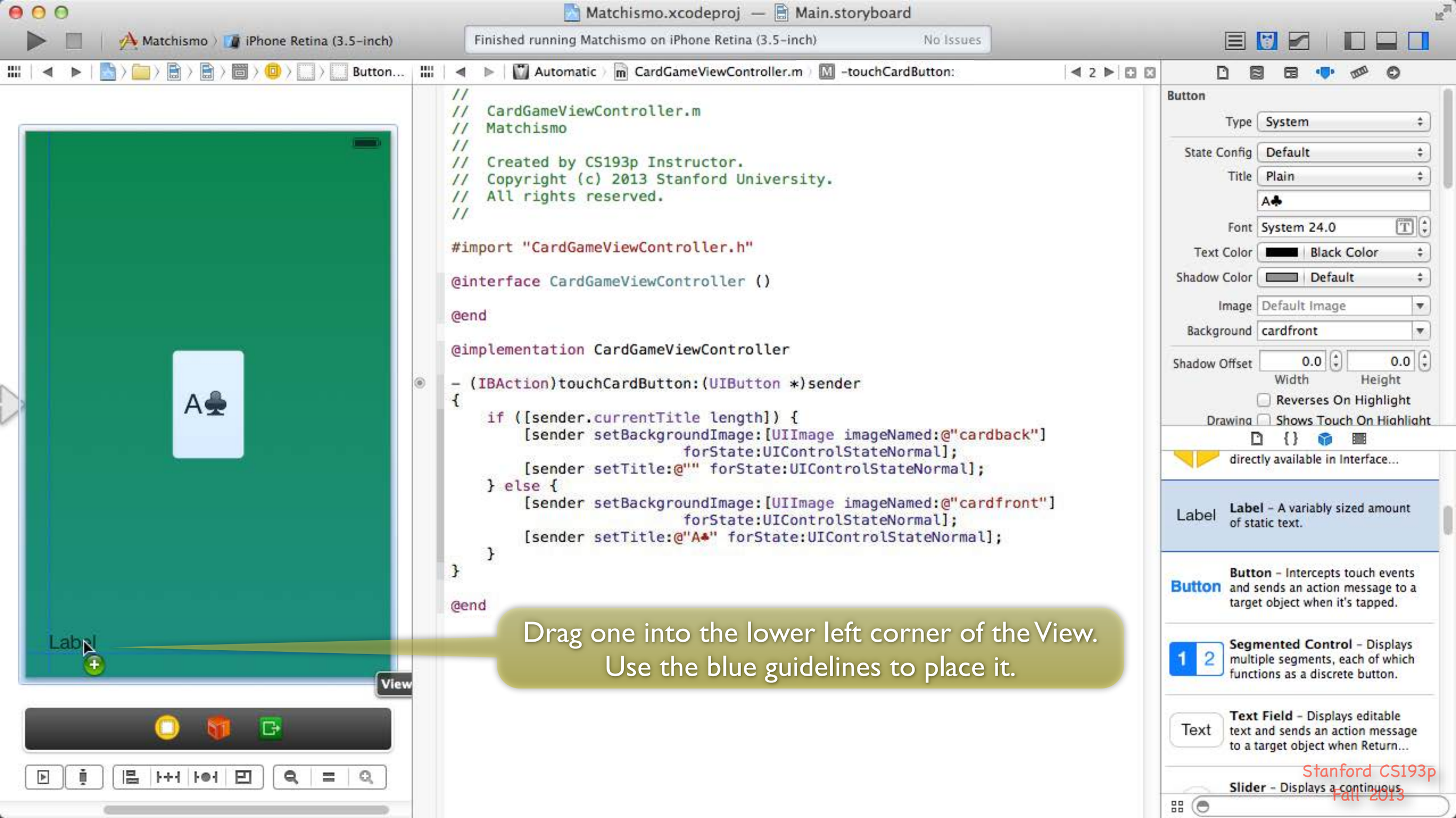
Label – A variably sized amount of static text.

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 text and sends an action message to a target object when Return…

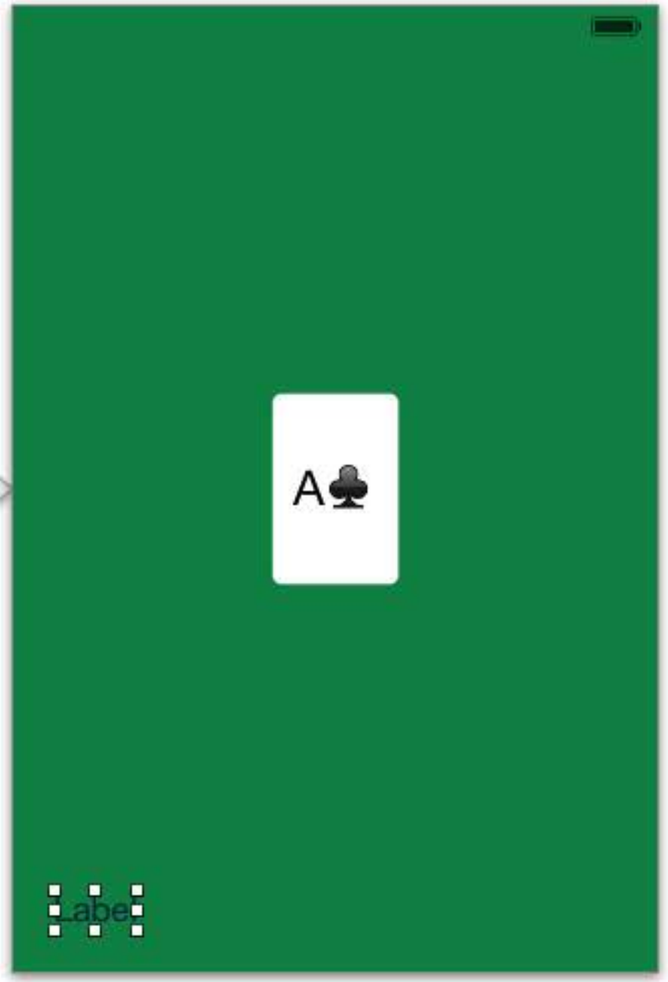
Slider – Displays a continuous


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:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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];
}
}
@end
A♣
Label
View
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background cardfront
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Drawing Shows Touch On Highlight
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 Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Drag one into the lower left corner of the View.
Use the blue guidelines to place it.
Stanford CS193p
Fall 2013

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 › -touchCardButton:

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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

The Attributes Inspector has now changed to show attributes of the Label.

Put this away by dragging it down.

Label

Text: Plain
Label
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

directly available in Interface...

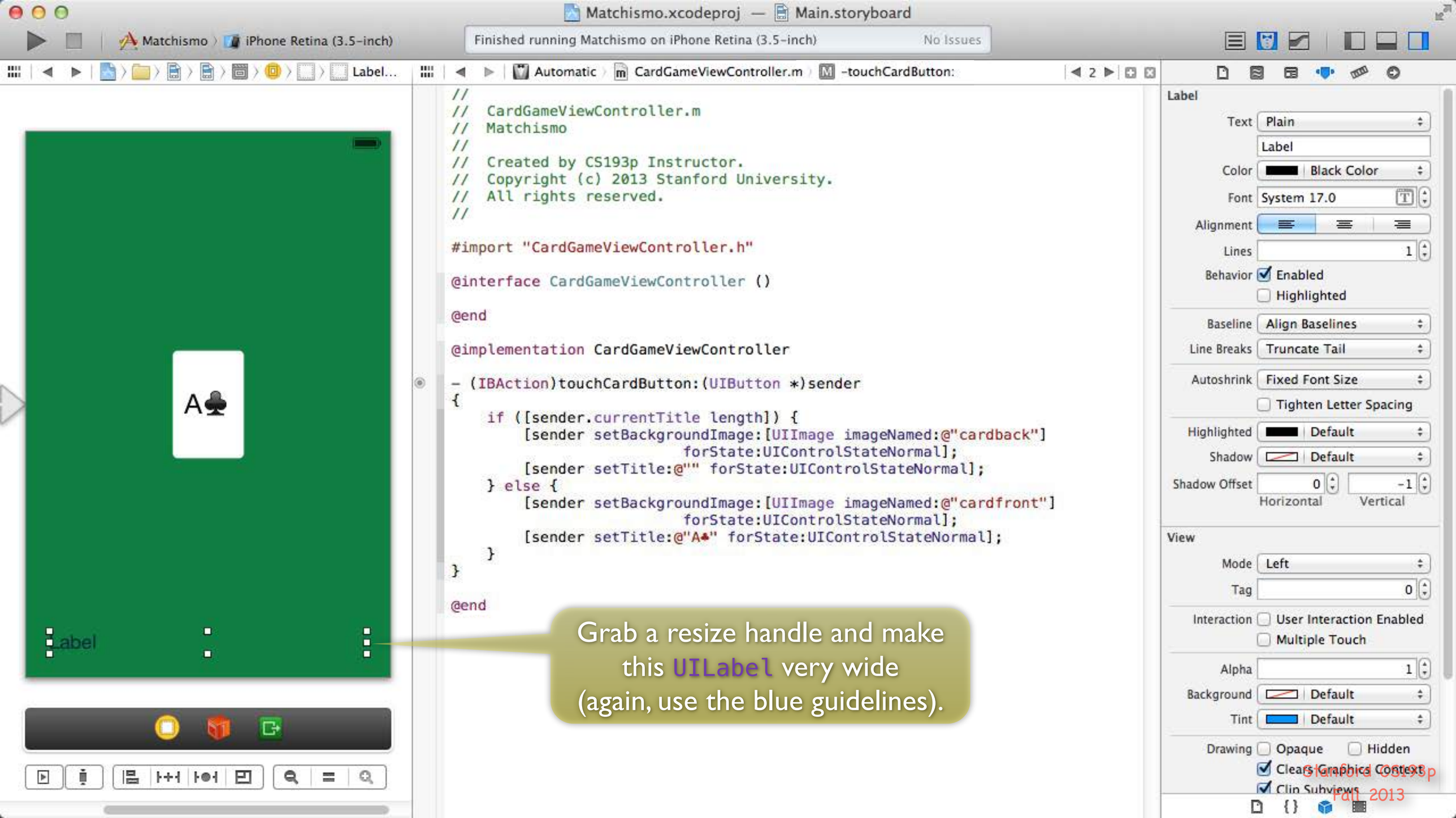
Label – **Label** – A variably sized amount of static text.

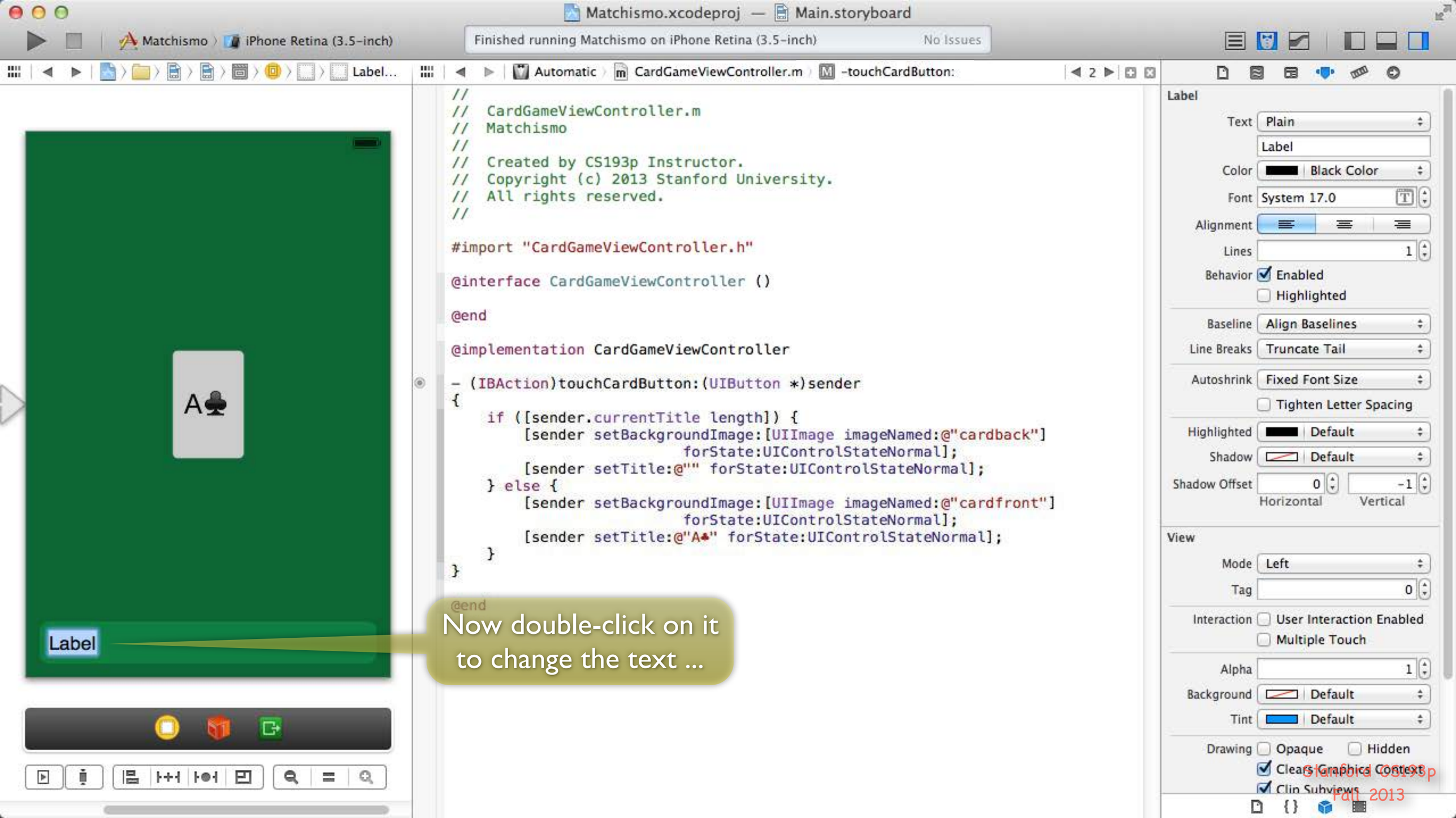
Button – **Button** – Intercepts touch events and sends an action message to a target object when it's tapped.

1 2 – **Segmented Control** – Displays multiple segments, each of which functions as a discrete button.

Text – **Text Field** – Displays editable text and sends an action message to a target object when Return...

Slider – Displays a continuous

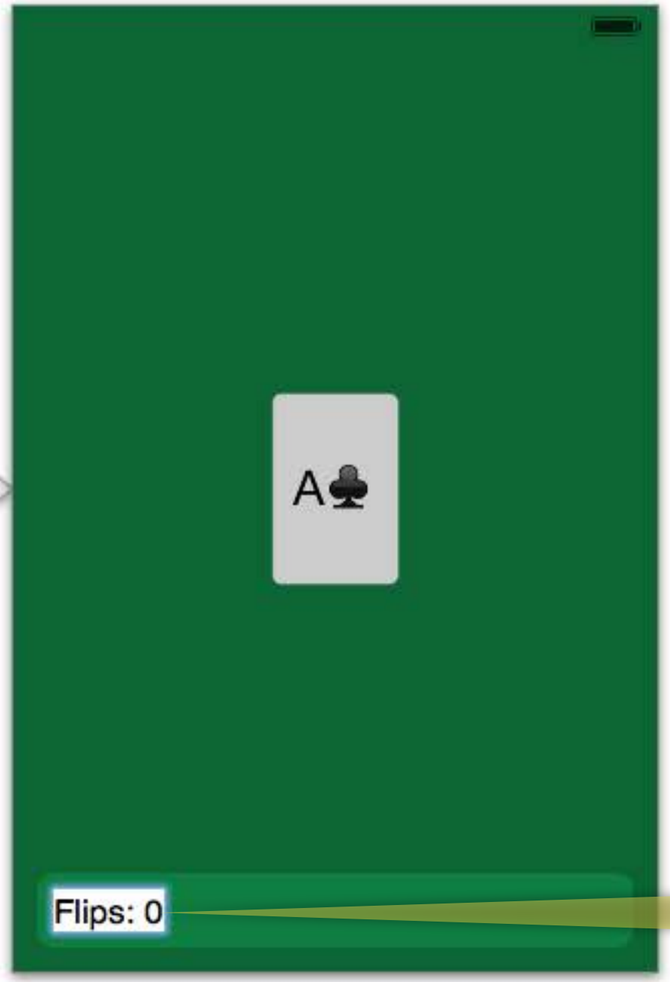

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Label...
Automatic
CardGameViewController.m
-touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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];
}
}
@end
A♣
Label
Label
Text Plain
Label
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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Grab a resize handle and make this UILabel very wide (again, use the blue guidelines).
Stanford CS193p
Fall 2013


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
-touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@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];
}
}
@end
A♣
Label
Now double-click on it to change the text ...
Label
Text Plain
Label
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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Stanford CS193p
Fall 2013

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 › -touchCardButton: | 2




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

... to "Flips: 0".

We're actually going to set this text entirely from our code, but "Flips: 0" is what is going to appear when our application first launches.

Label

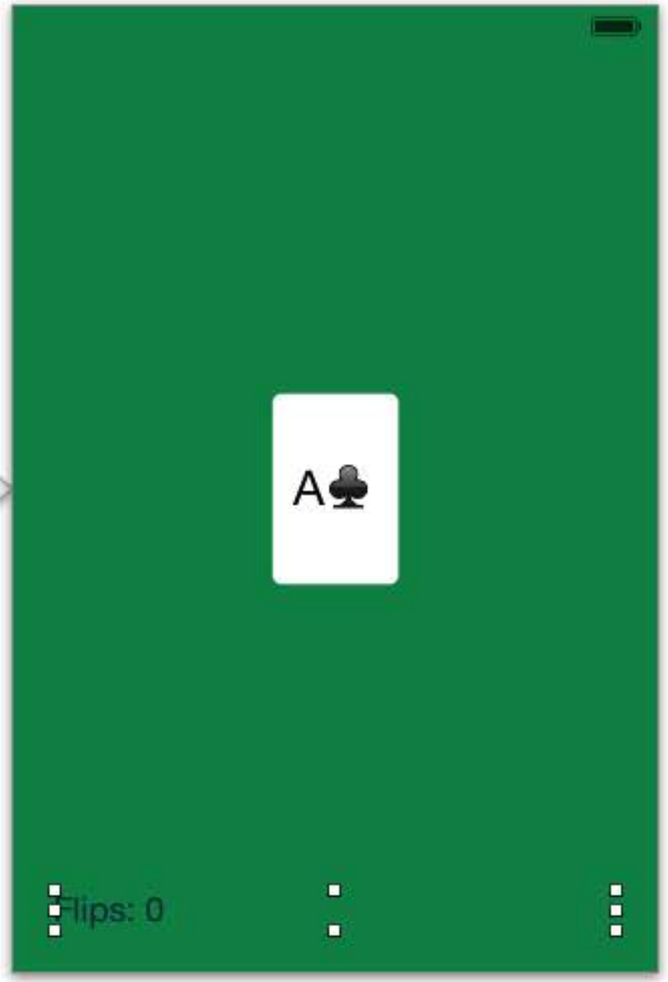
Text: Plain
Label
Color: Black Color
Font: System 17.0
Alignment
Lines: 1
Behavior: Enabled, Highlighted
Baseline: Align Baselines
Line Breaks: Truncate Tail
Autoshrink: Fixed Font Size
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Horizontal, -1 Vertical

View

Mode: Left
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews

Matchismo.xcodeproj — Main.storyboard

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




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

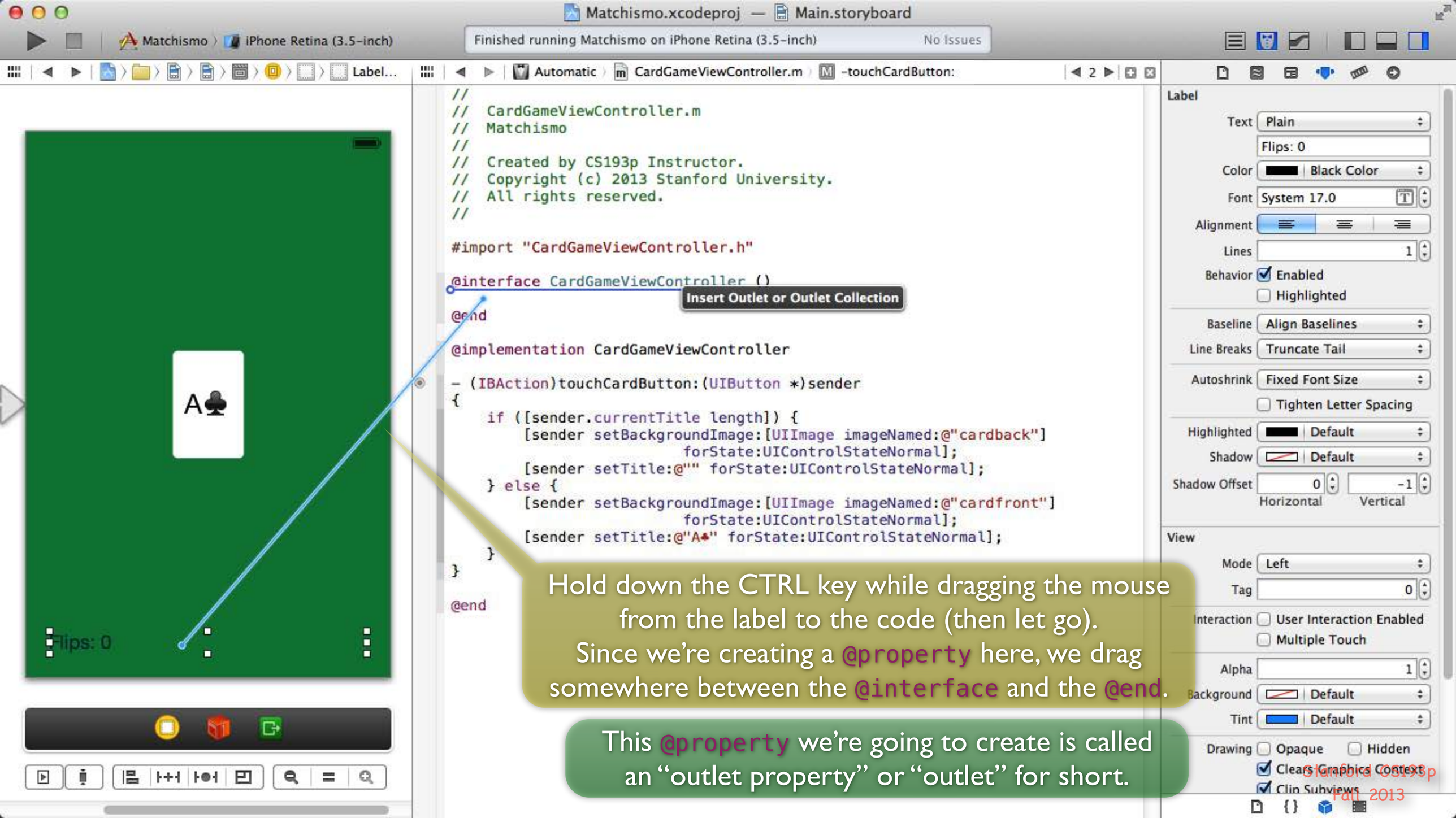
Label

Text: Plain
Flips: 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: 0 Horizontal, -1 Vertical

View

Mode: Left
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews

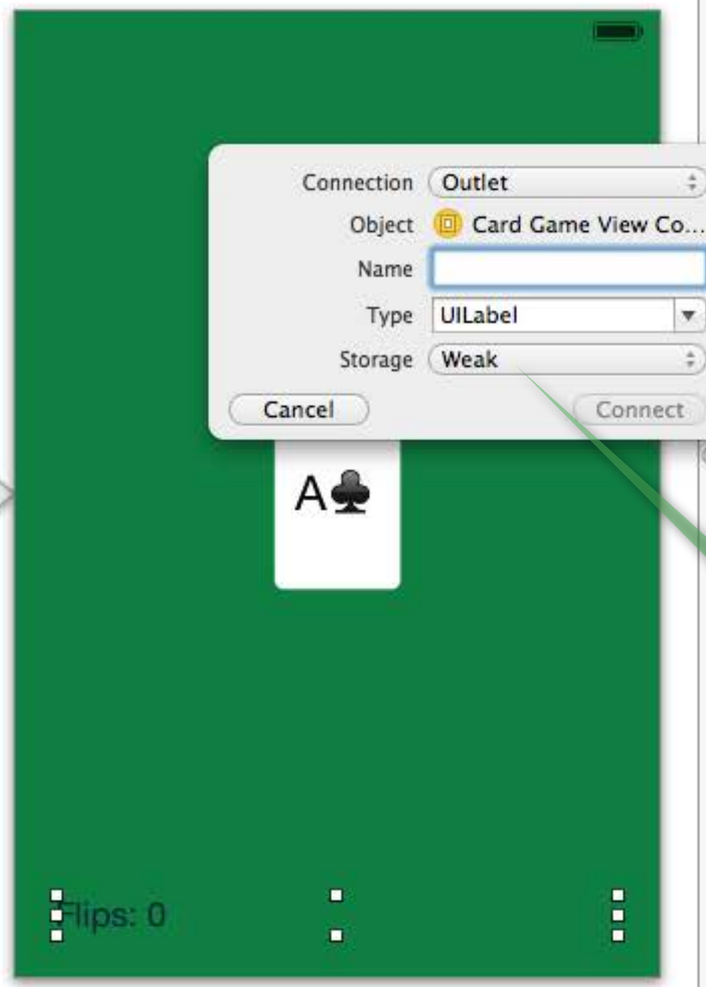
Now we have to connect this label to our Controller.
We do this by dragging to our code again
(but to the `@interface` instead of the `@implementation`).


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 › -touchCardButton:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
Insert Outlet or Outlet Collection
@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];
}
}
@end
A♣
Flips: 0
Label
Text Plain
Flips: 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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Hold down the CTRL key while dragging the mouse from the label to the code (then let go).
Since we're creating a @property here, we drag somewhere between the @interface and the @end.
This @property we're going to create is called an "outlet property" or "outlet" for short.
Stanford CS193p
Fall 2013

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 › -touchCardButton: | 2




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

#import "CardGameViewController.h"

@interface CardGameViewController ()

@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];
    }
}

@end
```

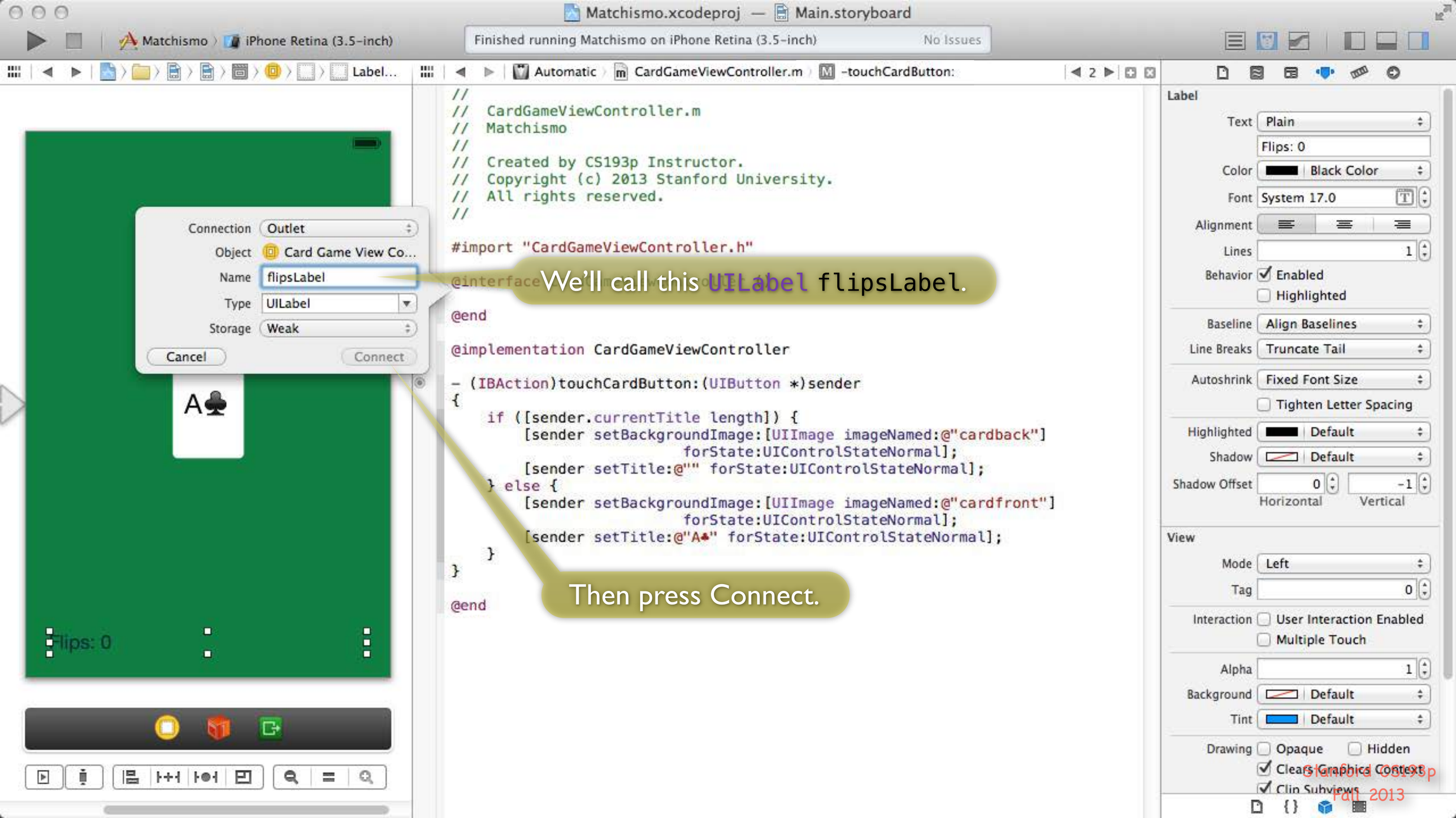
The `@property` created by this process is `weak` because the MVC's View already keeps a `strong` pointer to the `UILabel`, so there's no need for the Controller to do so as well.
And if the `UILabel` ever left the View, the Controller most likely wouldn't want a pointer to it anyway
(but if you did want to keep a pointer to it even if it left the View, you could change this to `strong` (very rare)).

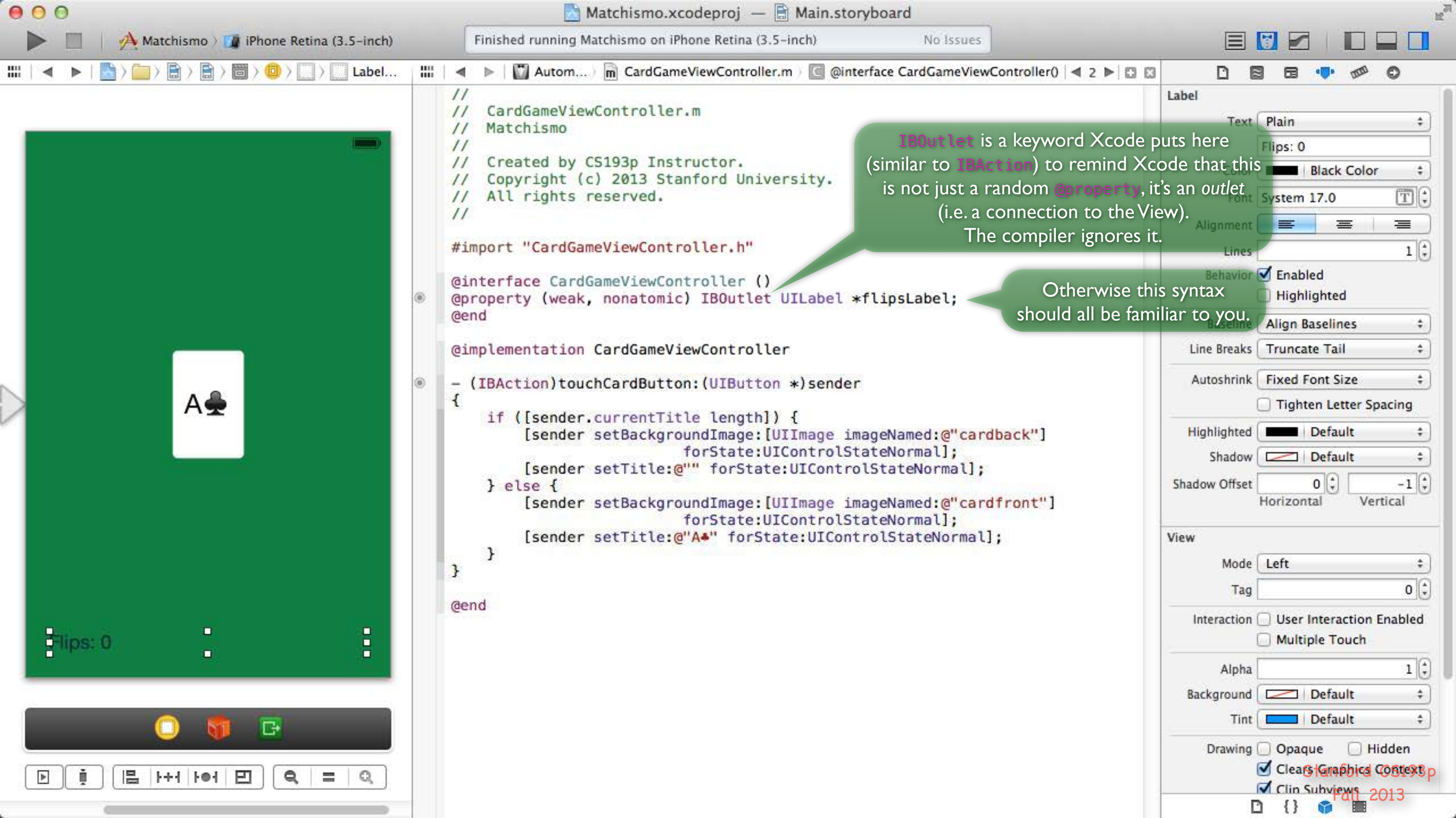
Label

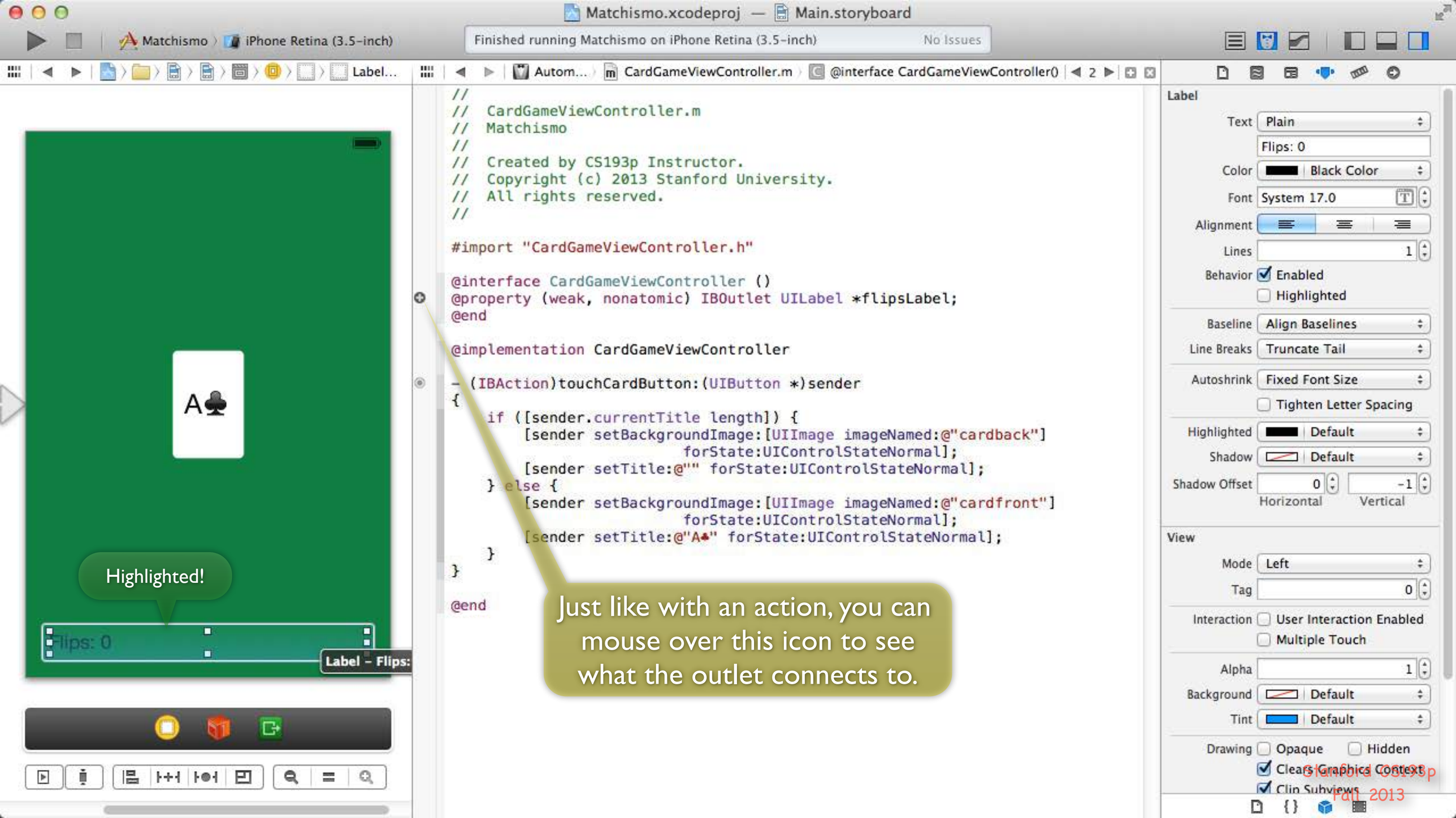
Text: Plain
Flips: 0
Color: Black Color
Font: System 17.0
Alignment
Lines: 1
Behavior: Enabled, Highlighted
Baseline: Align Baselines
Line Breaks: Truncate Tail
Autoshrink: Fixed Font Size
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Horizontal, -1 Vertical

View

Mode: Left
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews


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 › -touchCardButton:
Connection Outlet
Object Card Game View Co...
Name flipsLabel
Type UILabel
Storage Weak
Cancel
Connect
A♣
Flips: 0
We'll call this UILabel flipsLabel.
Then press Connect.
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewController.h"
@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];
}
}
@end
Label
Text Plain
Flips: 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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Label...
Autom...
CardGameViewController.m
@interface CardGameViewController()
2
//
// 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;
@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];
}
}
@end
A♣
Flips: 0
IBOutlet is a keyword Xcode puts here (similar to IBAction) to remind Xcode that this is not just a random @property, it's an outlet (i.e. a connection to the View). The compiler ignores it.
Otherwise this syntax should all be familiar to you.
Label
Text Plain
Flips: 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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Label...
Autom... › CardGameViewController.m › @interface CardGameViewController() 2
A♣
Highlighted!
Flips: 0
Label – Flips:
//
// 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;
@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];
}
}
@end
Just like with an action, you can mouse over this icon to see what the outlet connects to.
Label
Text Plain
Flips: 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 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Label... | Autom... | CardGameViewController.m | @interface CardGameViewController() | 2

```
//
//  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;
@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];
    }
}

@end
```

It is also possible to see the connections between your View and Controller by right-clicking on an element in your View.

For example, right-click on the Flips:0 label.

Here are all the connections to/from this `UILabel`. Notice the outlet `flipsLabel`.

A♣

Flips: 0

Label – Flips: 0

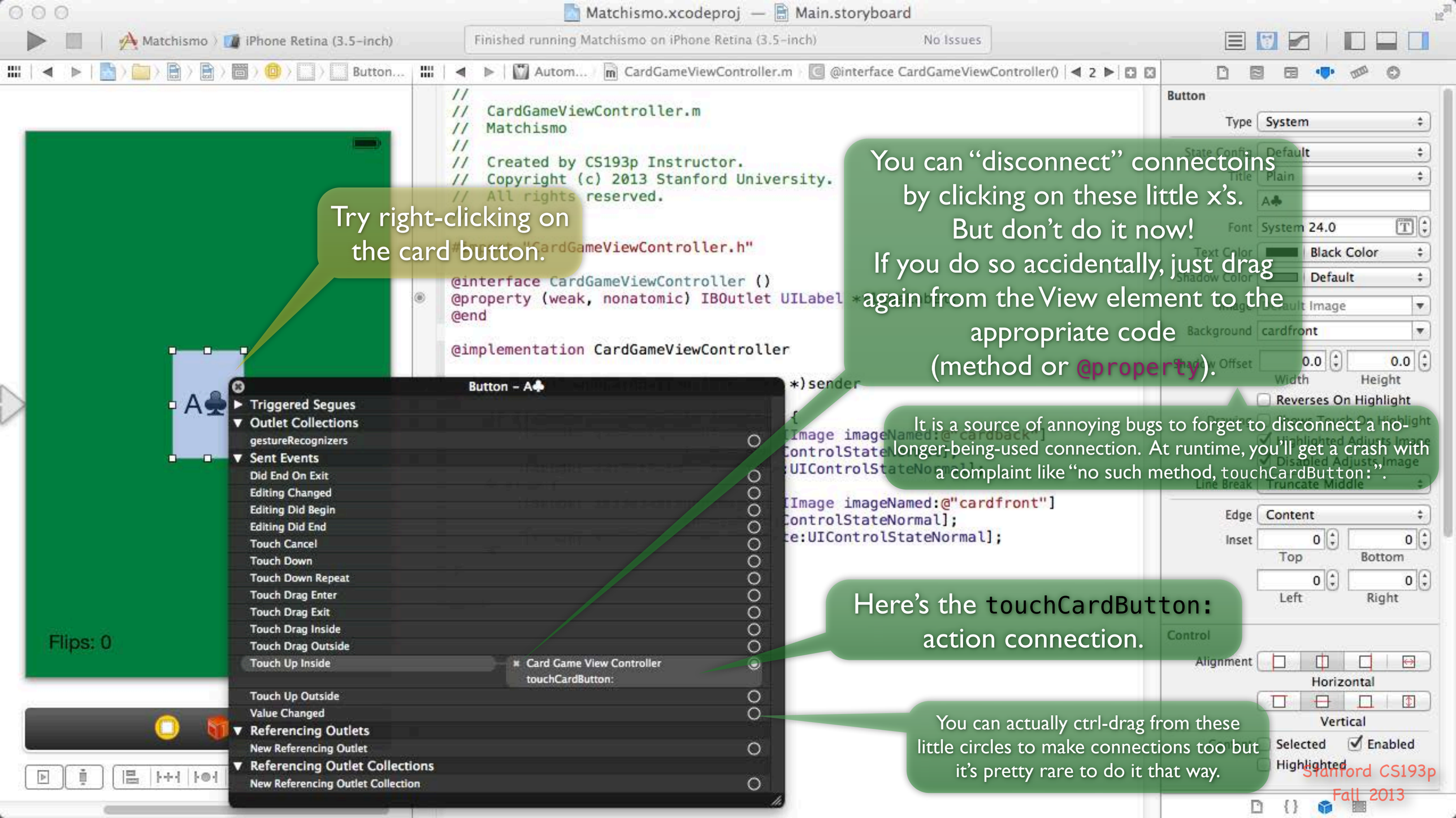
- Outlet Collections
  - gestureRecognizers
- Referencing Outlets
  - flipsLabel → Card Game View Controller
  - New Referencing Outlet
- Referencing Outlet Collections
  - New Referencing Outlet Collection

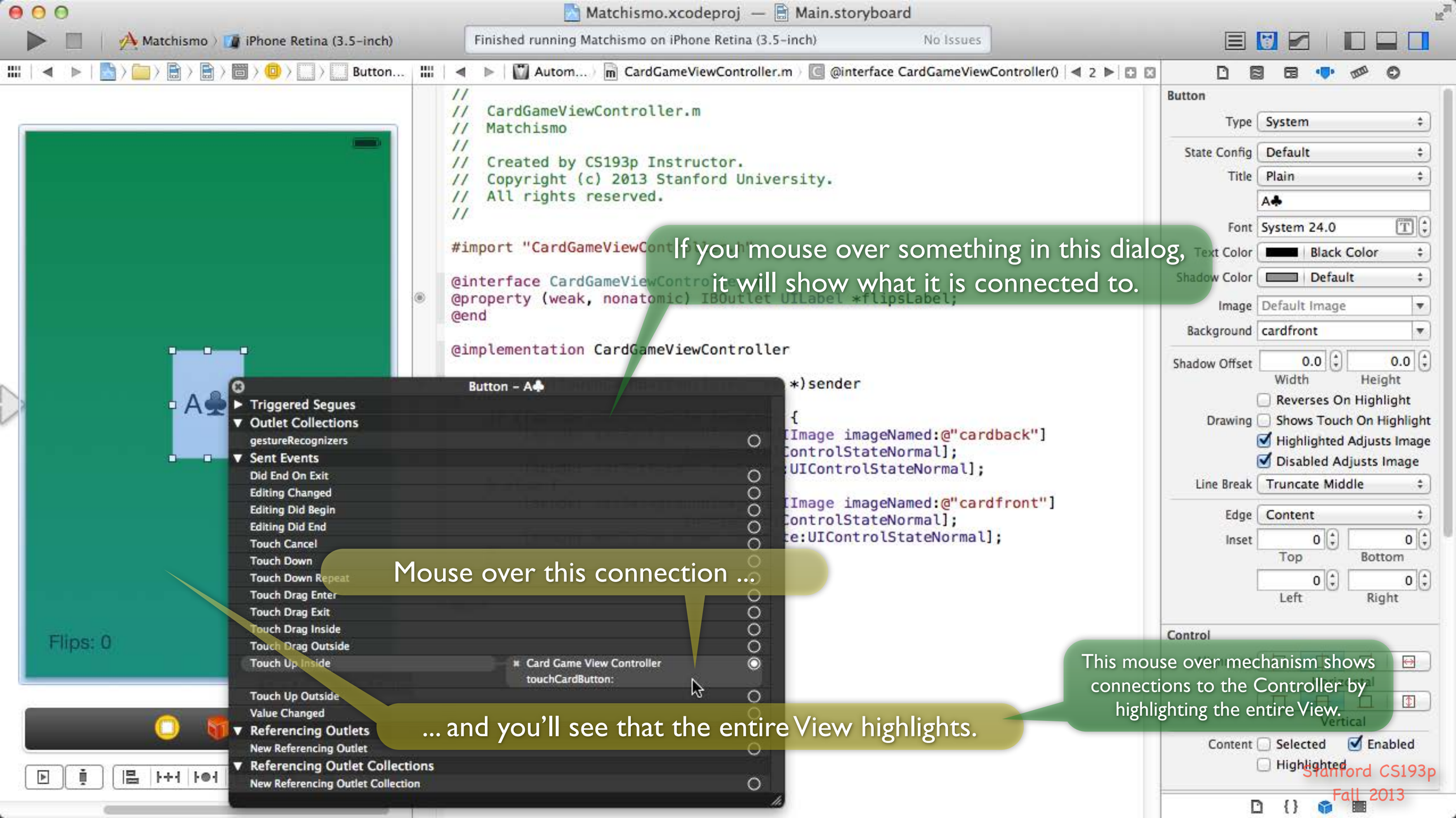
Label

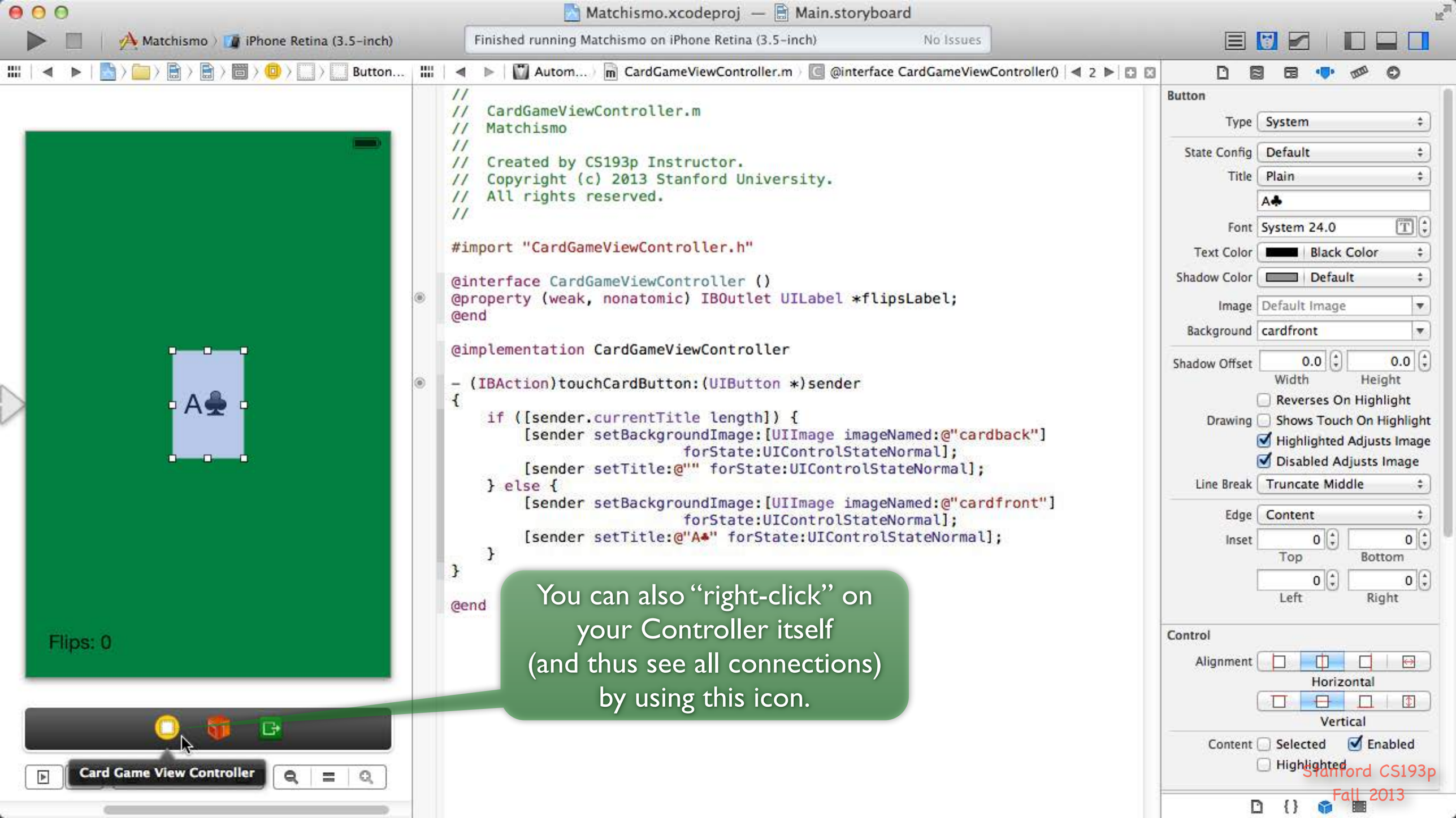
Text: Plain
Flips: 0
Color: Black Color
Font: System 17.0
Alignment
Lines: 1
Behavior: Enabled, Highlighted
Baseline: Align Baselines
Line Breaks: Truncate Tail
Autoshrink: Fixed Font Size
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Horizontal, -1 Vertical

View

Mode: Left
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
CardGameViewController.m
@interface CardGameViewController()
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel
@end
@implementation CardGameViewController
Flips: 0
Button - A♣
Triggered Segues
Outlet Collections
gestureRecognizers
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Card Game View Controller
touchCardButton:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Button
Type System
Font System 24.0
Text Color Black Color
Background cardfront
Edge Content
Inset
Top
Bottom
Left
Right
Control
Alignment
Horizontal
Vertical
Selected
Enabled
Highlighted
Try right-clicking on the card button.
You can "disconnect" connectoins by clicking on these little x's. But don't do it now! If you do so accidentally, just drag again from the View element to the appropriate code (method or @property).
It is a source of annoying bugs to forget to disconnect a no-longer-being-used connection. At runtime, you'll get a crash with a complaint like "no such method, touchCardButton:".
Here's the touchCardButton: action connection.
You can actually ctrl-drag from these little circles to make connections too but it's pretty rare to do it that way.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Autom... › CardGameViewController.m › @interface CardGameViewController() 2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "CardGameViewCont
@interface CardGameViewContro
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
@implementation CardGameViewController
*)sender
{
Image imageNamed:@"cardback"]
ControlStateNormal];
UIControlStateNormal];
Image imageNamed:@"cardfront"]
ControlStateNormal];
e:UIControlStateNormal];
Flips: 0
Button - A♣
Triggered Segues
Outlet Collections
gestureRecognizers
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Card Game View Controller
touchCardButton:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
Vertical
Content Selected Enabled
Highlighted
If you mouse over something in this dialog, it will show what it is connected to.
Mouse over this connection ...
... and you'll see that the entire View highlights.
This mouse over mechanism shows connections to the Controller by highlighting the entire View.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Autom... › CardGameViewController.m › @interface CardGameViewController() 2
//
// 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;
@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];
}
}
@end
A♣
Flips: 0
Card Game View Controller
You can also "right-click" on your Controller itself (and thus see all connections) by using this icon.
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Button... | Autom... | CardGameViewController.m | @interface CardGameViewController() | 2

```
//
//  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;
@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.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", ++self.flips];
}
```

This title bar says that we are looking at the connections to/from our Controller.

Right-click on this icon.

Flips: 0

Card Game View Controller

- ▶ Triggered Segues
- ▼ Outlets
  - flipsLabel → Label – Flips: 0
  - searchDisplayController
  - view → View
- ▼ Presenting Segues
  - relationship
  - push
  - modal
  - custom
  - embed
- ▼ Referencing Outlets
  - New Referencing Outlet
- ▼ Referencing Outlet Collections
  - New Referencing Outlet Collection
- ▼ Received Actions
  - touchCardButton: → Button – A♣ Touch Up Inside

Button

Type: System

State Config: Default

Title: Plain

A♣

Font: System 24.0

Text Color: Black Color

Shadow Color: Default

Image: Default Image

Background: cardfront

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

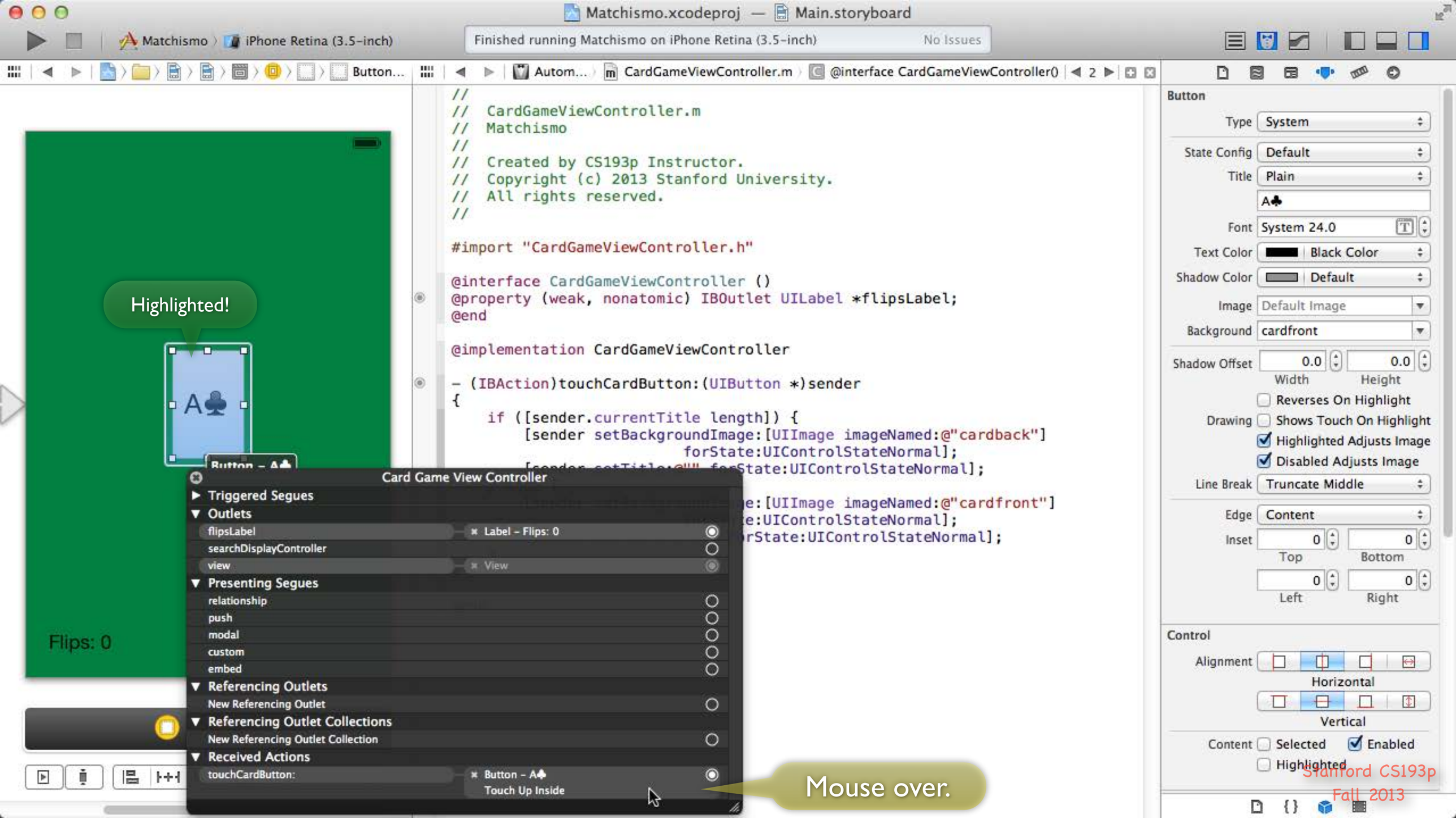
Edge: Content

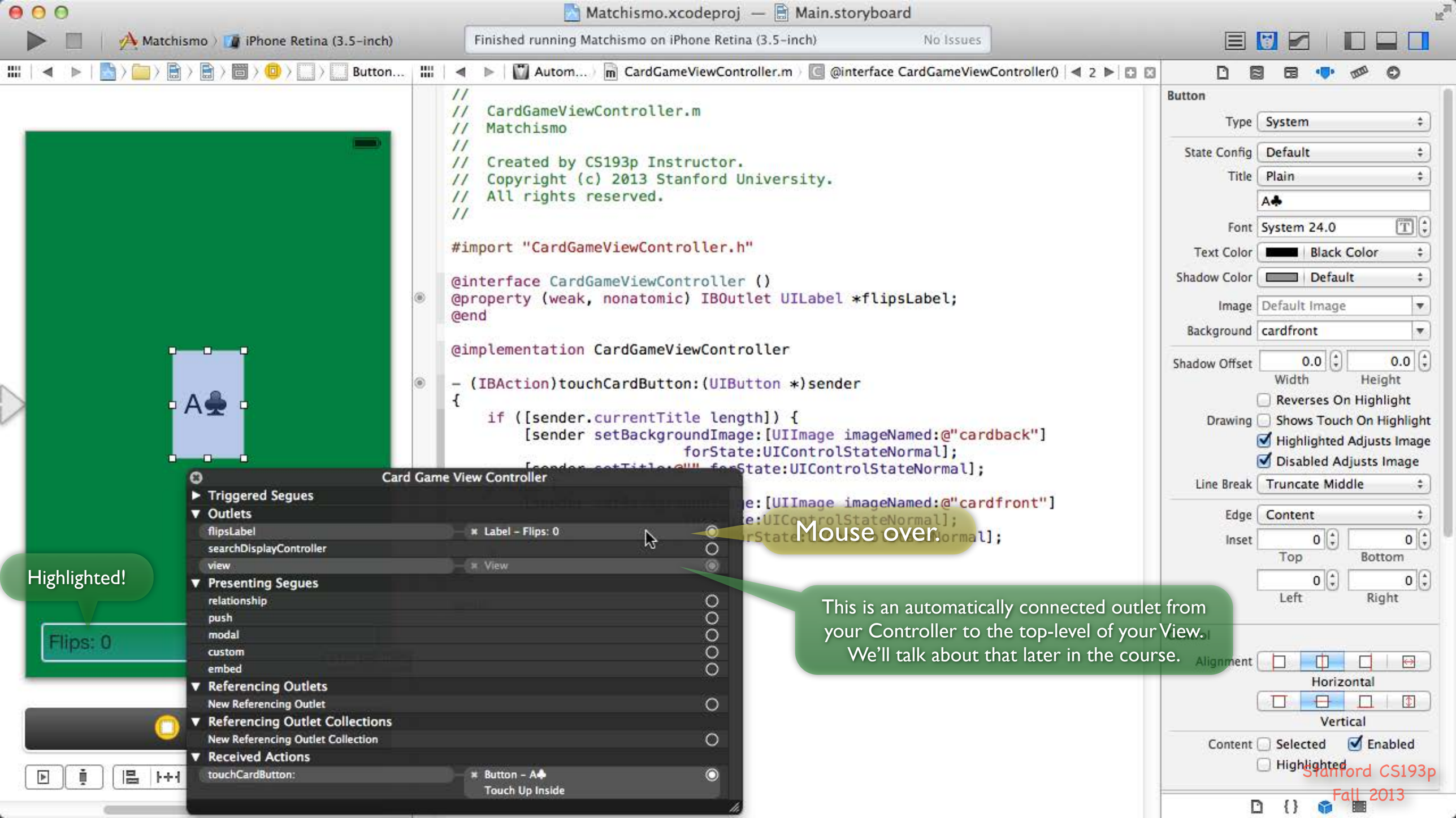
Inset: 0 Top, 0 Bottom, 0 Left, 0 Right

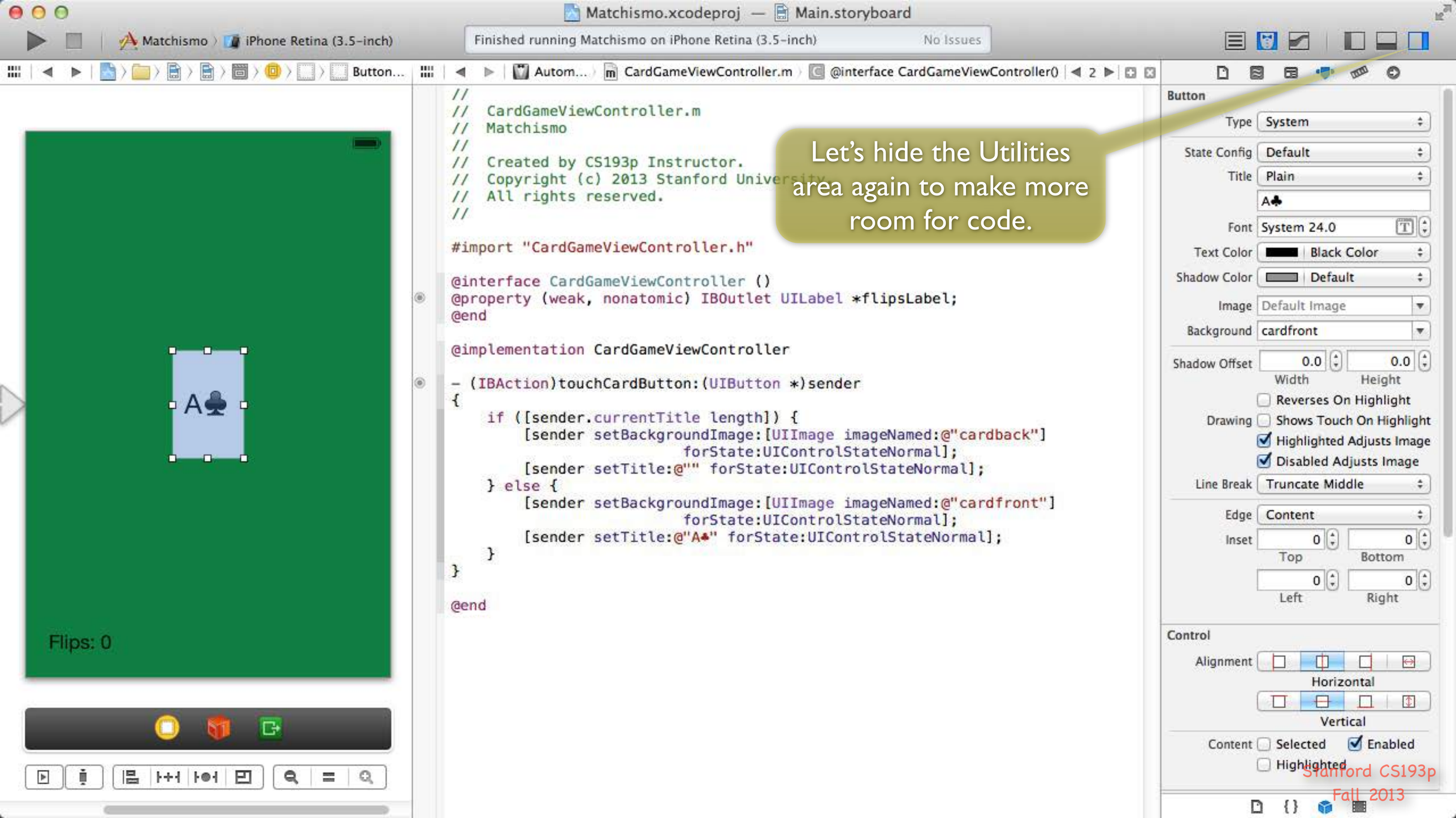
Control

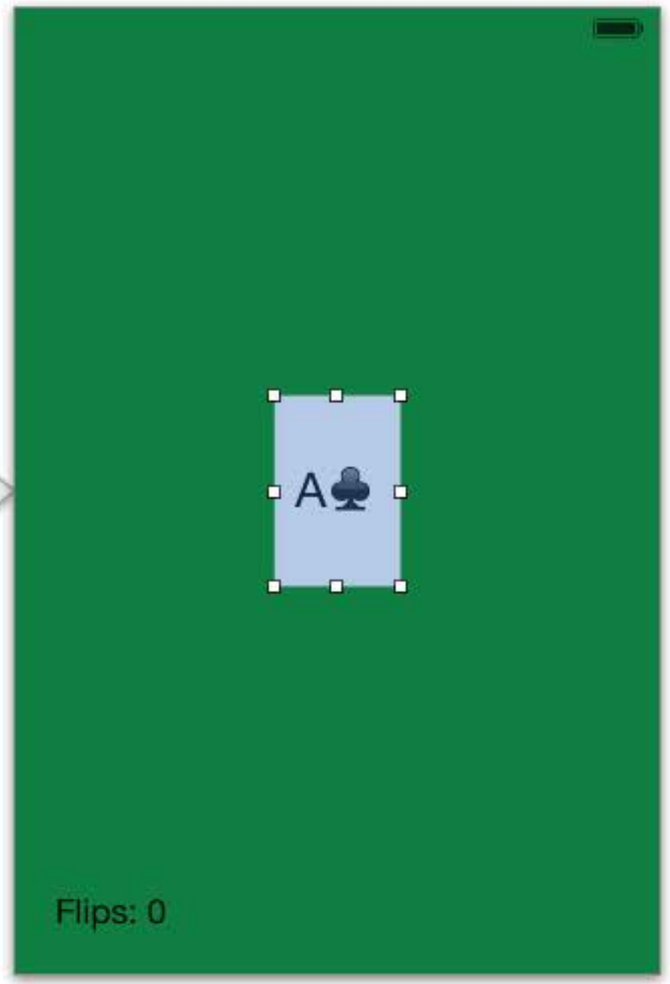
Alignment: Horizontal, Vertical

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
Button...
Autom...
CardGameViewController.m
@interface CardGameViewController()
//
// 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;
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
Highlighted!
A♣
Flips: 0
Card Game View Controller
Triggered Segues
Outlets
flipsLabel
Label - Flips: 0
searchDisplayController
view
View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
touchCardButton:
Button - A♣
Touch Up Inside
Mouse over.
Button
Type
System
State Config
Default
Title
Plain
A♣
Font
System 24.0
Text Color
Black Color
Shadow Color
Default
Image
Default Image
Background
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
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Autom... › CardGameViewController.m › @interface CardGameViewController() 2
//
// 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;
@end
@implementation CardGameViewController
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
forState:UIControlStateNormal];
A♣
Flips: 0
Highlighted!
Card Game View Controller
Triggered Segues
Outlets
flipsLabel
Label – Flips: 0
searchDisplayController
view
View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
touchCardButton:
Button – A♣
Touch Up Inside
Mouse over.
This is an automatically connected outlet from your Controller to the top-level of your View. We'll talk about that later in the course.
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
Alignment
Horizontal
Vertical
Content Selected Enabled
Highlighted
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Button...
Autom...
CardGameViewController.m
@interface CardGameViewController()
2
A♣
Flips: 0
//
// 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;
@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];
}
}
@end
Let's hide the Utilities area again to make more room for code.
Button
Type System
State Config Default
Title Plain
A♣
Font System 24.0
Text Color Black Color
Shadow Color Default
Image Default Image
Background 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
Stanford CS193p
Fall 2013




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

#import "CardGameViewController.h"

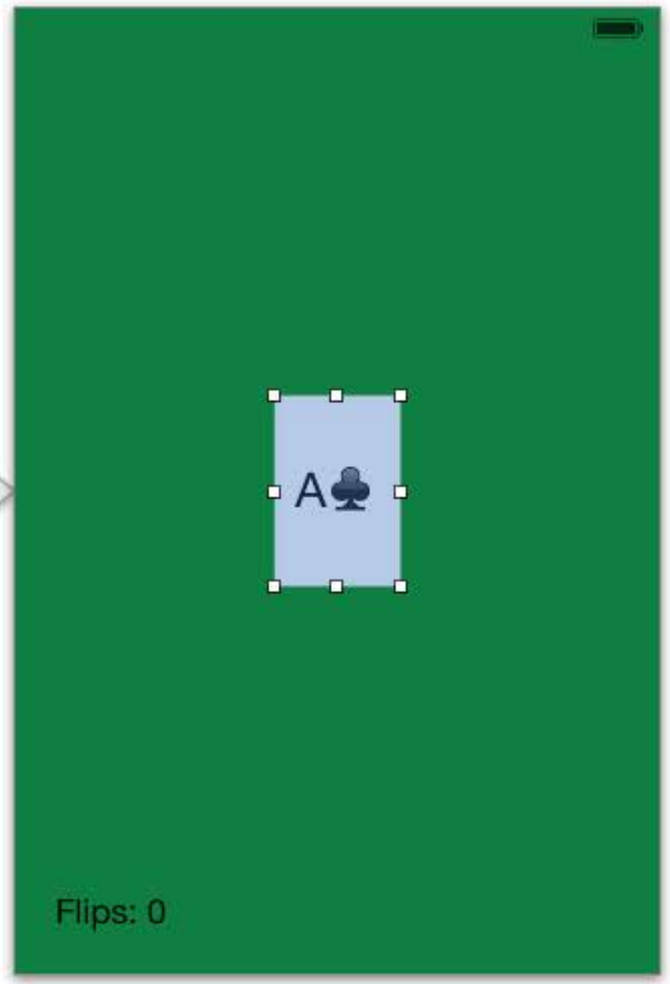
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end

@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];
    }
}

@end
```

We are going to keep track of the number of flips of the card using a new `@property`.




```
//
//  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

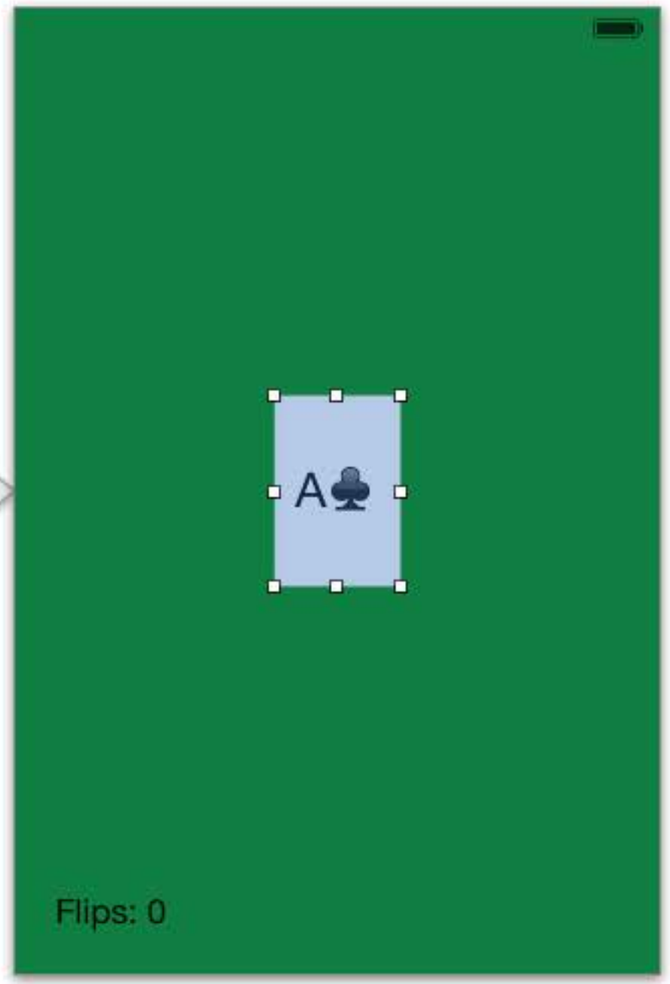
@implementation CardGameViewController

- (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];
    }
}

@end
```

We'll call this `@property` `flipCount`.

Nothing special about this `@property`, it's just an integer.
We could use `NSInteger` or `NSUInteger` here,
but we're using `int`, just to show doing so.




```
//
//  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

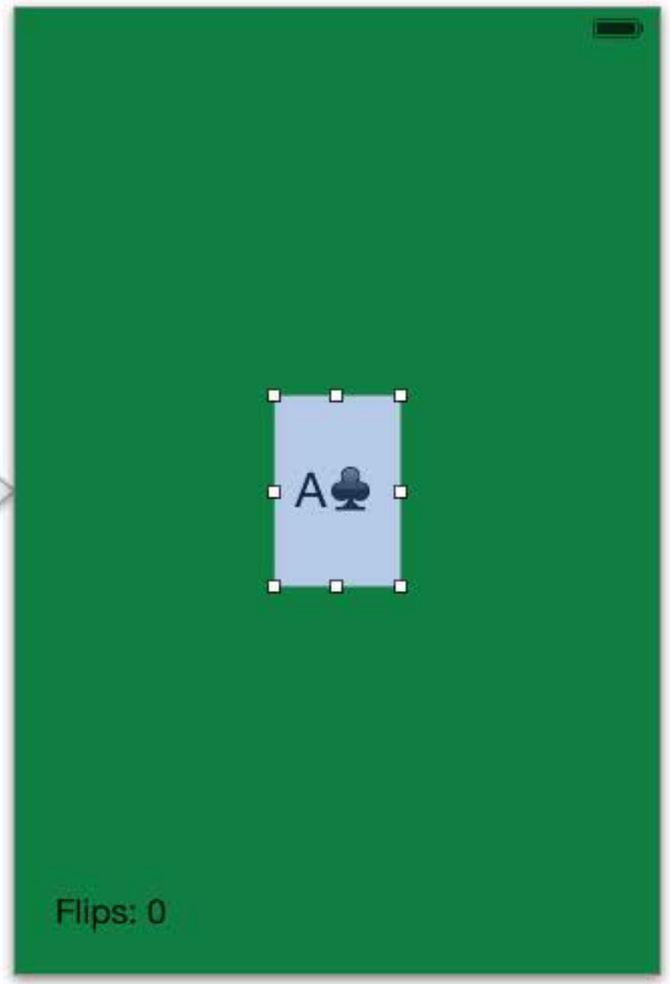
@implementation CardGameViewController

- (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++;
}

@end
```

And we'll just increment it each time we flip the card.

Notice that we can use ++ notation just like with a variable. This is the same as `self.flipCount = self.flipCount+ 1`. In other words, `self.flipCount++` invokes both the getter and the setter for the `flipCount` `@property`.




```
//
//  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

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

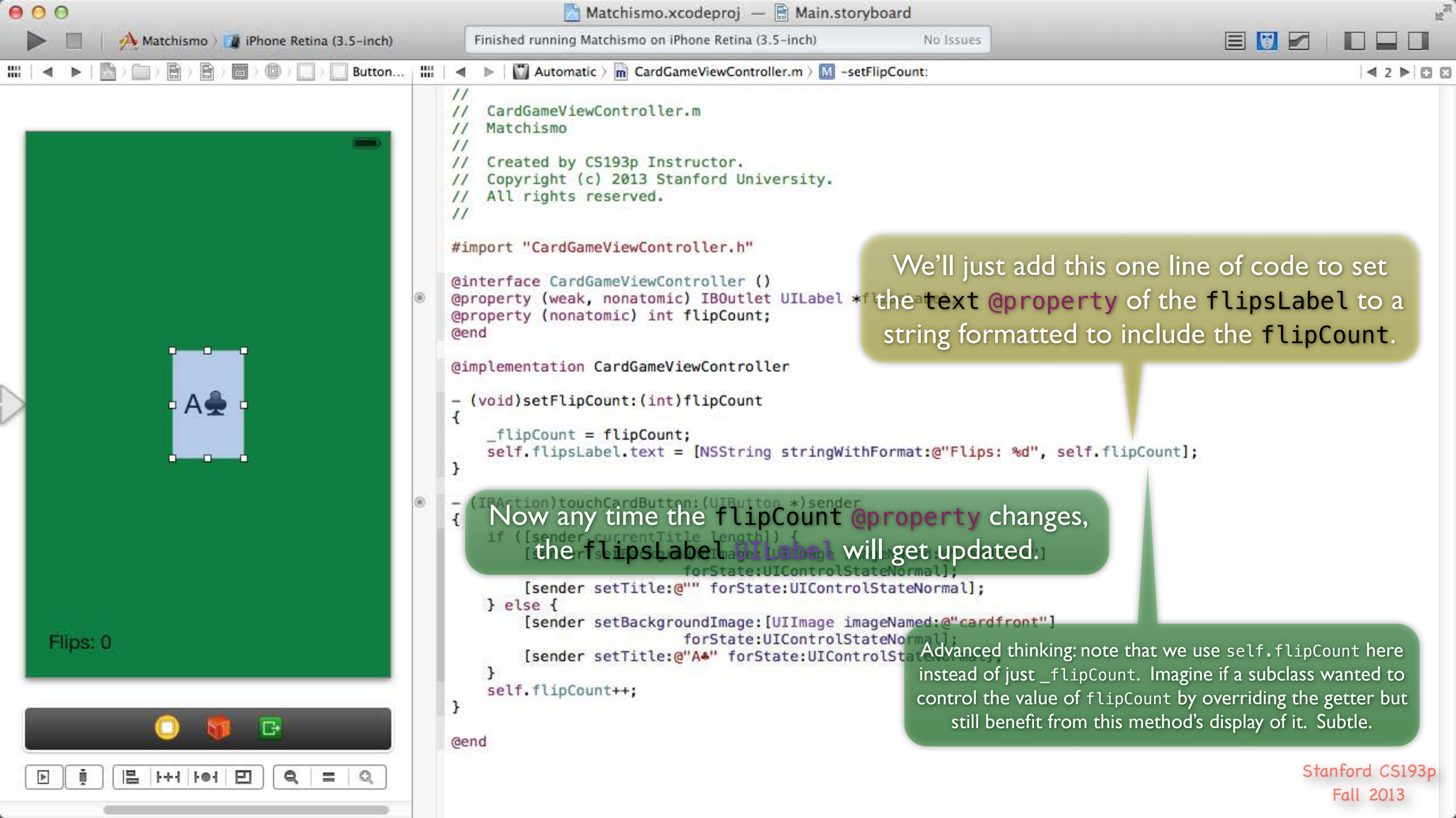
- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"]
                          forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        [sender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

@end
```

But how can we coordinate the `flipCount` `@property` with the `flipsLabel` `UILabel`?

Easy! We'll use the *setter* of the `flipCount` `@property`.
This is yet another advantage of using `@property` instead of accessing instance variables directly.

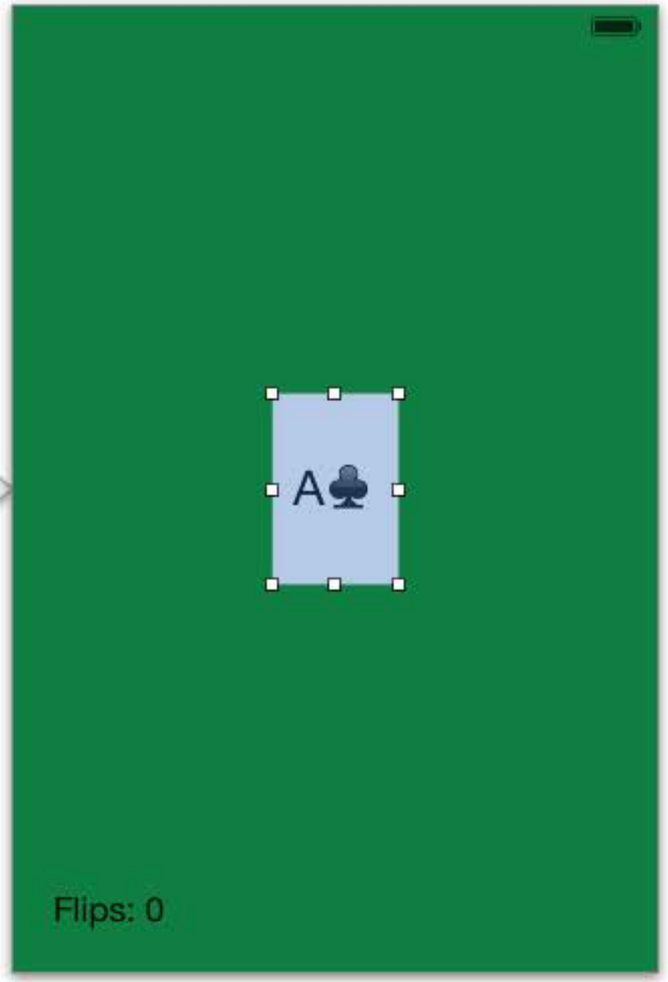
This is what the setter for `flipCount` would normally look like.


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 › -setFlipCount:
A♣
Flips: 0
//
// 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
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
}
- (IBAction)touchCardButton:(UIButton *)sender
{
if ([sender currentTitle].length) {
forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal];
} else {
[sender setBackgroundImage:[UIImage imageNamed:@"cardfront"]
forState:UIControlStateNormal];
[sender setTitle:@"A♣" forState:UIControlStateNormal];
}
self.flipCount++;
}
@end
We'll just add this one line of code to set the text @property of the flipsLabel to a string formatted to include the flipCount.
Now any time the flipCount @property changes, the flipsLabel UILabel will get updated.
Advanced thinking: note that we use self.flipCount here instead of just _flipCount. Imagine if a subclass wanted to control the value of flipCount by overriding the getter but still benefit from this method's display of it. Subtle.
Stanford CS193p
Fall 2013

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 › -setFlipCount: | 2




```
//
//  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

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

- (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++;
}
@end
```

While we're here, let's take another aside to look at a debugging technique. We can output something to the console any time we want.

We do it using the C function NSLog(). The first argument to NSLog() is a @"" printf-like format string specifying what to output. The rest of the arguments are the values matching up with the %'s in the format string.

The first argument to NSLog() must always be an @"". Not any other kind of NSString.

Remember, the console will automatically appear at the bottom of the screen when you run.

Okay, let's run again!

Note this starts out with whatever we typed in our View.

```
//
//  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

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

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"c
                          forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"c
                          forState:UIControlStateNormal];
```

Matchismo.xcodeproj — Main.storyboard

Matchismo › iPhone Retina (3.5-inch) | Running Matchismo on iPhone Retina (3.5-inch)

Button… | Automatic › CardGameViewController.m › -setFlipCount:

```
//
//  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

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

- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"c
                          forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal
    } else {
        [sender setBackgroundImage:[UIImage imageNamed:@"c
                          forState:UIControlStateNormal];
```

Flips: 0

Carrier

Touch

This should change.

Flips: 1

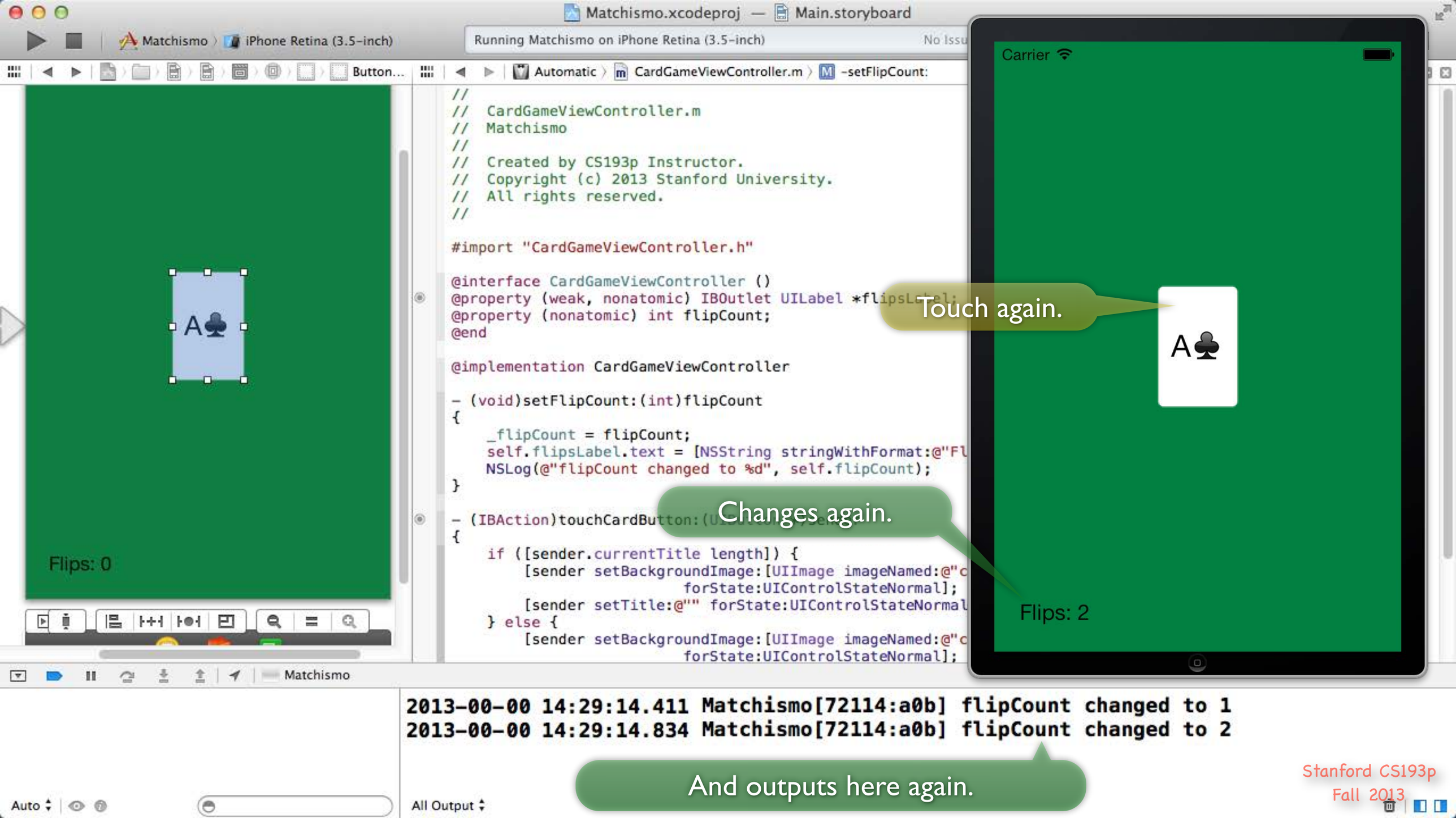
Matchismo

```
2013-00-00 14:29:14.411 Matchismo[72114:a0b] flipCount changed to 1
```

And here's the output of the NSLog().

Auto | All Output

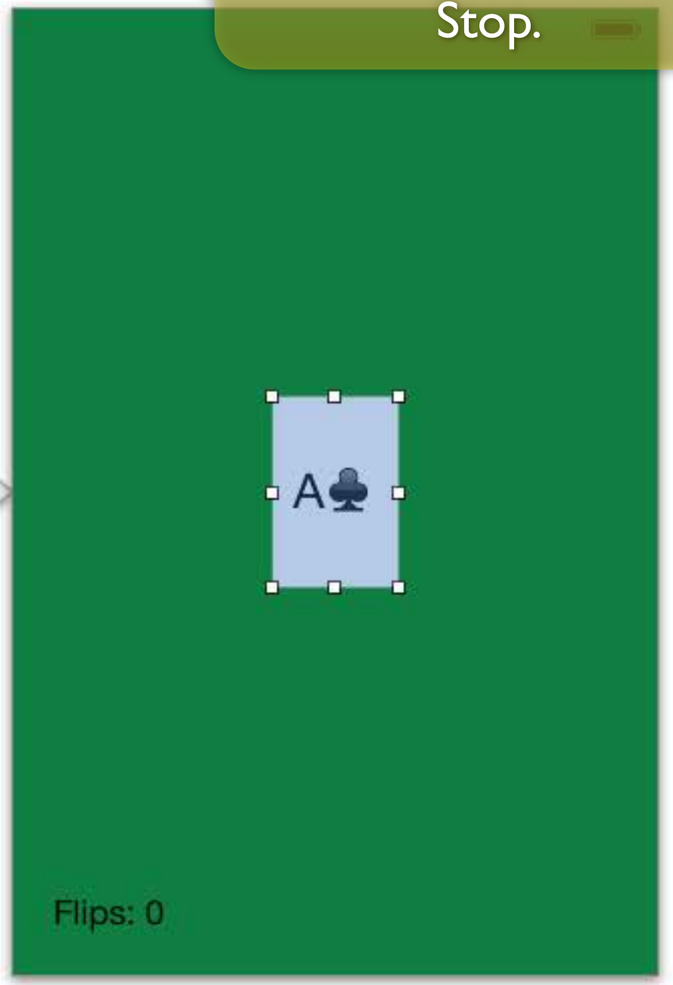
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Running Matchismo on iPhone Retina (3.5-inch)
Automatic › CardGameViewController.m › -setFlipCount:
//
// 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
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Fl
NSLog(@"flipCount changed to %d", self.flipCount);
}
- (IBAction)touchCardButton:
{
if ([sender.currentTitle length]) {
[sender setBackgroundImage:[UIImage imageNamed:@"c
forState:UIControlStateNormal];
[sender setTitle:@"" forState:UIControlStateNormal
} else {
[sender setBackgroundImage:[UIImage imageNamed:@"c
forState:UIControlStateNormal];
Flips: 0
Carrier
Touch again.
Changes again.
Flips: 2
Matchismo
2013-00-00 14:29:14.411 Matchismo[72114:a0b] flipCount changed to 1
2013-00-00 14:29:14.834 Matchismo[72114:a0b] flipCount changed to 2
And outputs here again.
Auto
All Output
Stanford CS193p
Fall 2013

Matchismo.xcodeproj — Main.storyboard

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

Stop.



Button...

Automatic › CardGameViewController.m › -setFlipCount:

```
//
//  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

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

- (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++;
}

@end
```

Well this is all wonderful, but it's sort of boring since it only shows the A♣ all the time.

If only we had a `Deck` of `PlayingCards` to `drawRandomCard` from, we could make each flip show a different card.

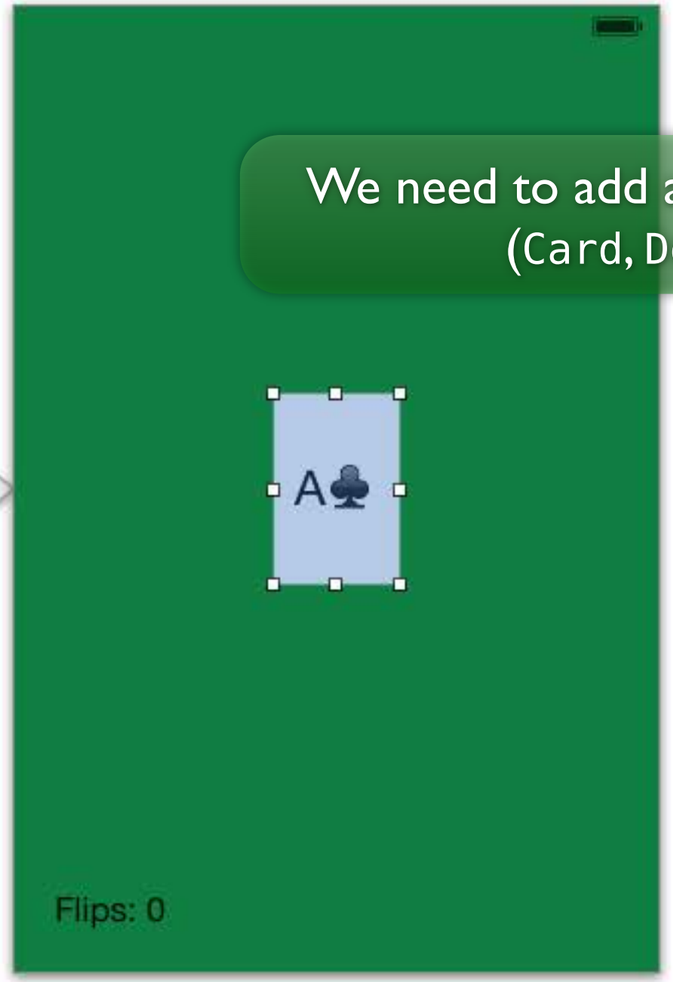
Hmmm ...

Let's start by revealing the Navigator again.

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products



We need to add all the classes from the Model we created in lecture (Card, Deck, PlayingCard and PlayingCardDeck).

```
//
//  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

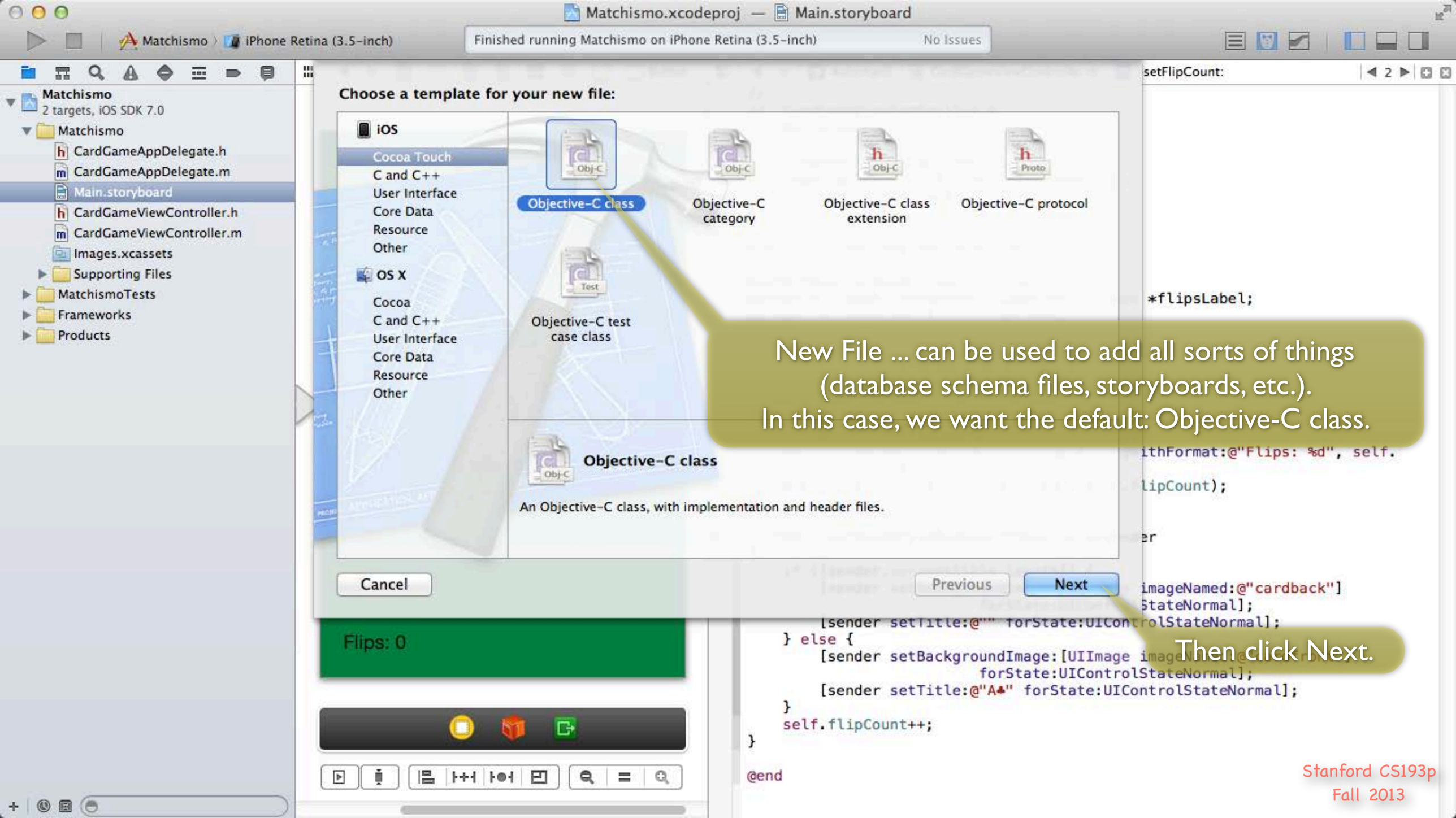
@implementation CardGameViewController

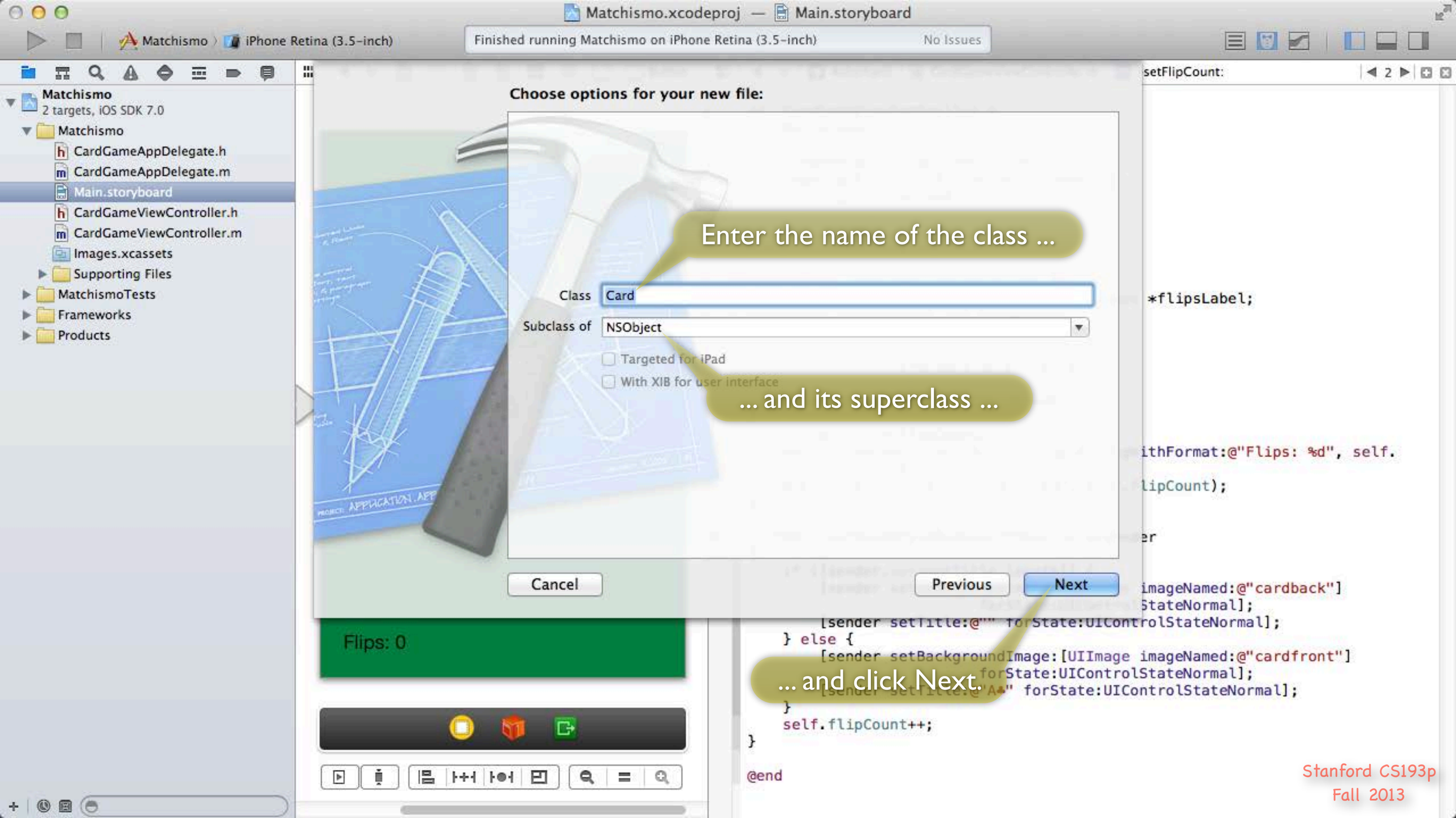
- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.
        flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

- (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++;
}

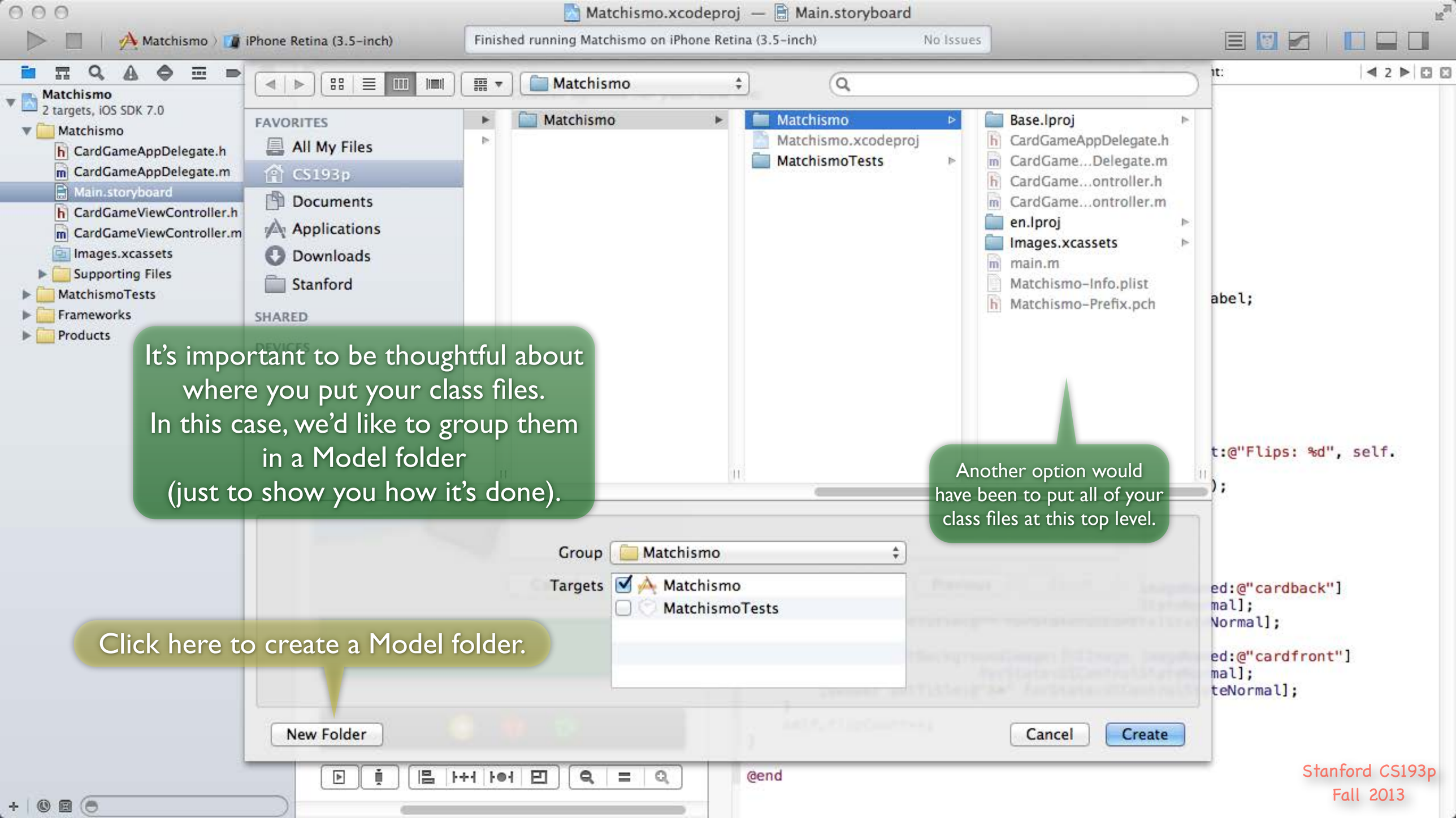
@end
```

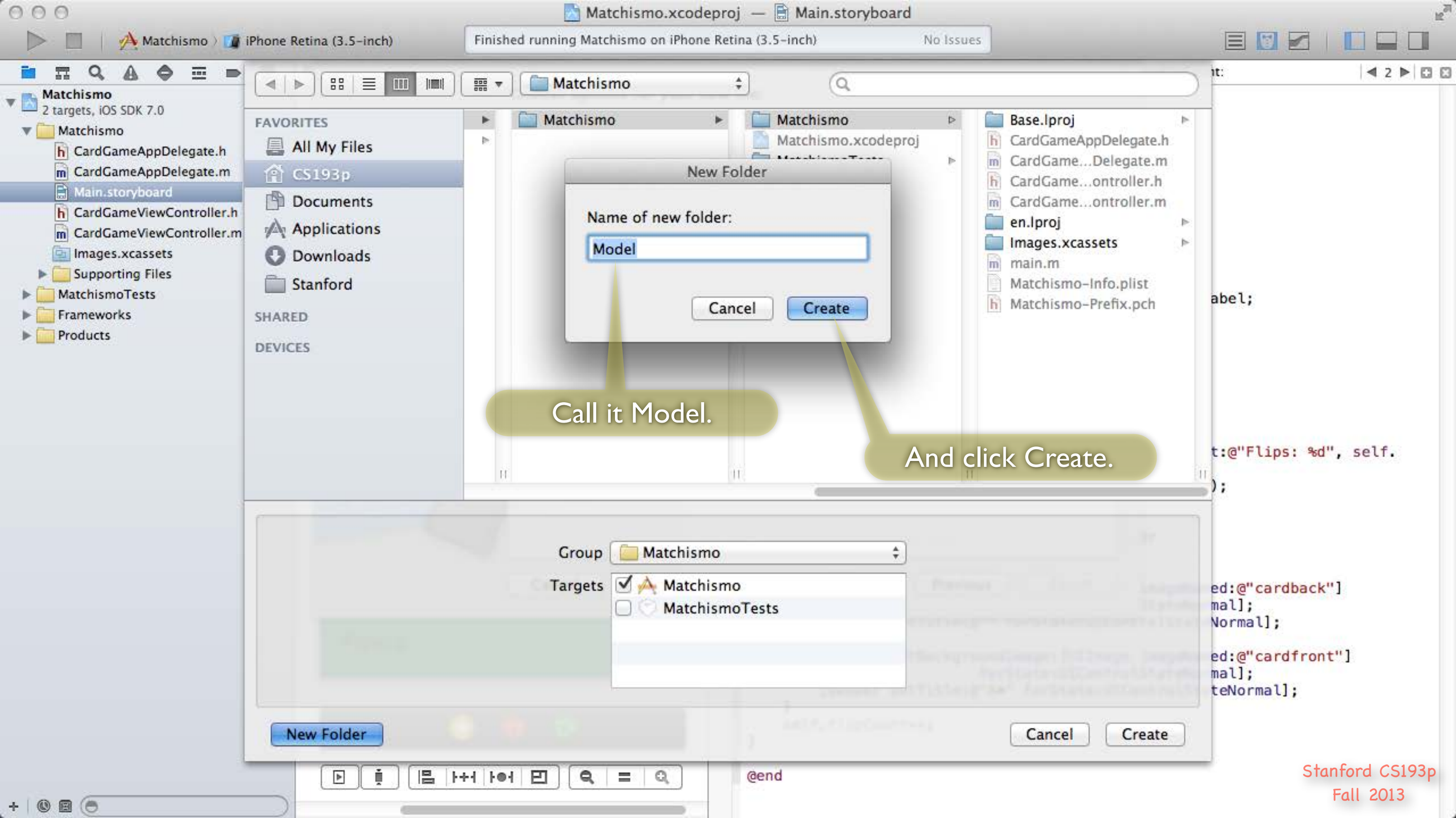
Whenever we want to add a file
to our project in Xcode (of any kind),
we use the `File > New > File ...` menu item.

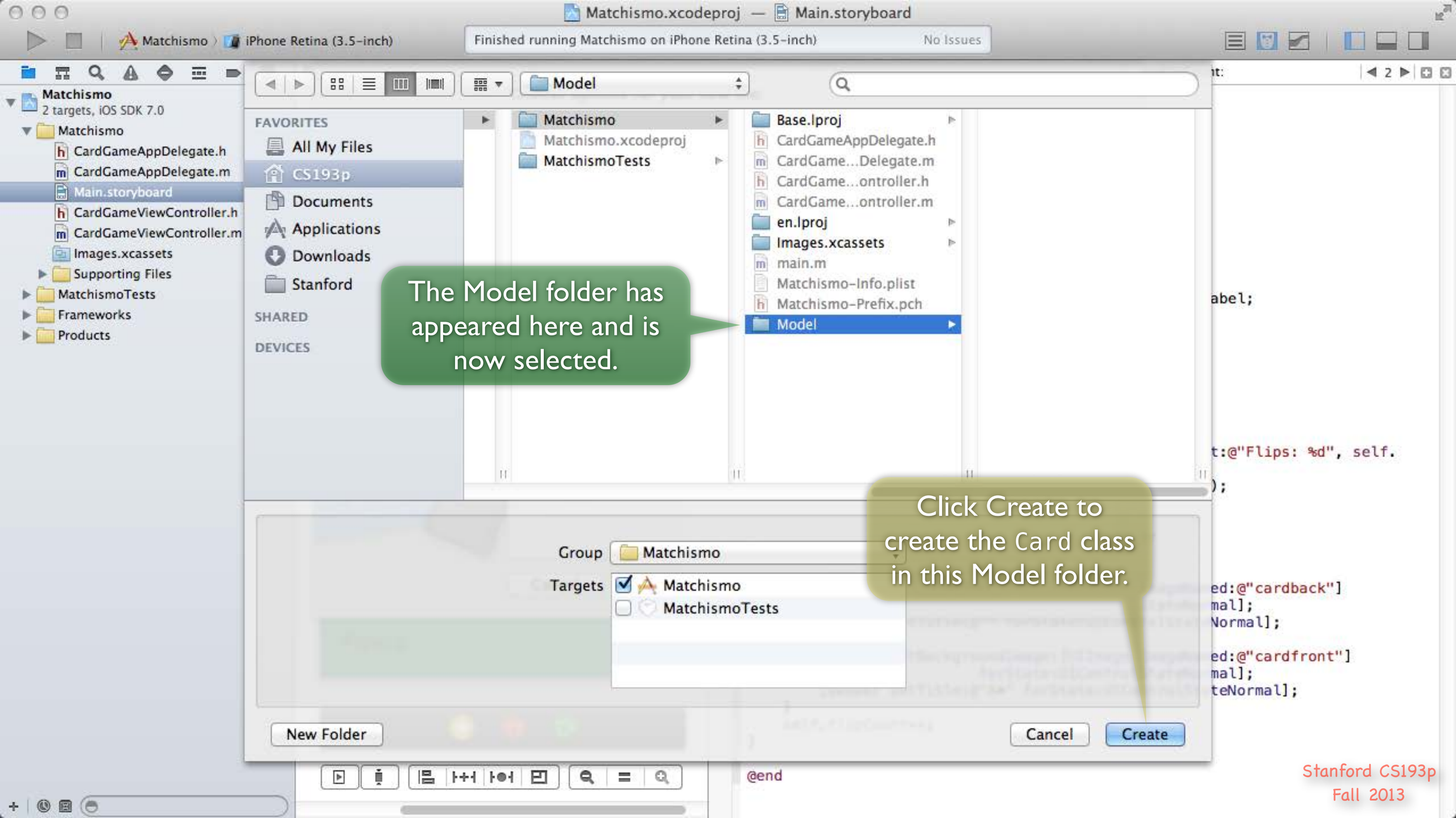

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
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
New File ... can be used to add all sorts of things
(database schema files, storyboards, etc.).
In this case, we want the default: Objective-C class.
Then click Next.
Flips: 0
setFlipCount:
*flipsLabel;
ithFormat:@"Flips: %d", self.
lipCount);
er
imageNamed:@"cardback"]
StateNormal];
rolStateNormal];
} else {
[sender setBackgroundImage:[UIImage
forState:UIControlStateNormal];
[sender setTitle:@"A♣" forState:UIControlStateNormal];
}
self.flipCount++;
}
@end
Stanford CS193p
Fall 2013



Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
SHARED
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
It's important to be thoughtful about where you put your class files. In this case, we'd like to group them in a Model folder (just to show you how it's done).
Another option would have been to put all of your class files at this top level.
Group
Matchismo
Targets
Matchismo
MatchismoTests
Click here to create a Model folder.
New Folder
Cancel
Create
@end
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
SHARED
DEVICES
Matchismo
Matchismo.xcodeproj
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
New Folder
Name of new folder:
Model
Cancel
Create
Call it Model.
And click Create.
Group
Matchismo
Targets
Matchismo
MatchismoTests
New Folder
Cancel
Create
@end
Stanford CS193p
Fall 2013


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
The Model folder has appeared here and is now selected.
Group
Matchismo
Targets
Matchismo
MatchismoTests
Click Create to create the Card class in this Model folder.
New Folder
Cancel
Create
Matchismo
2 targets, iOS SDK 7.0
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
@end
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - Card.h
  - Card.m
  - CardGameViewController.h
  - CardGameViewController.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › Matchismo › Card.h › No Selection

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

#import <Foundation/Foundation.h>

@interface Card : NSObject

@end
```

Counterparts › Card.m › No Selection

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

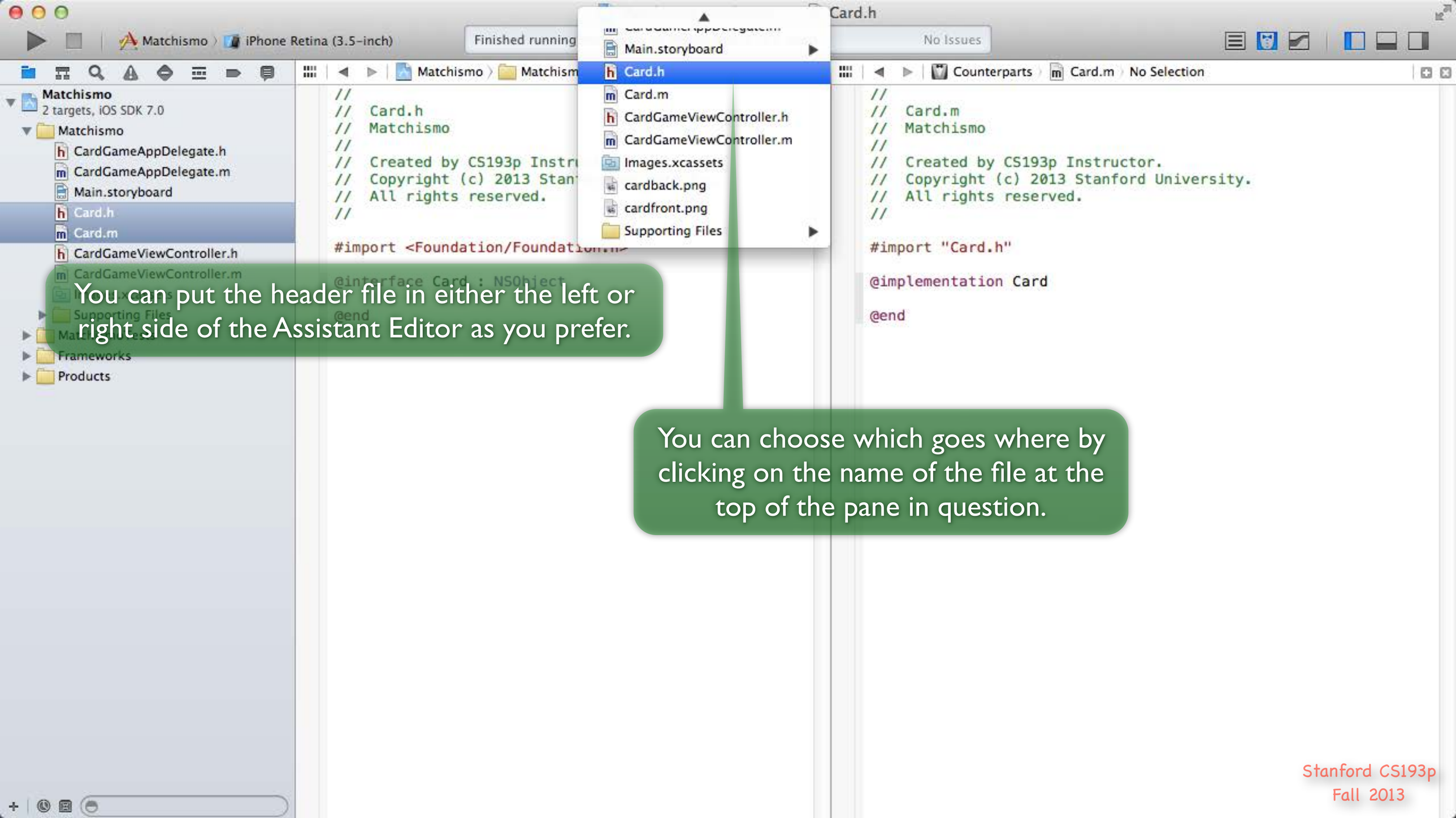
#import "Card.h"

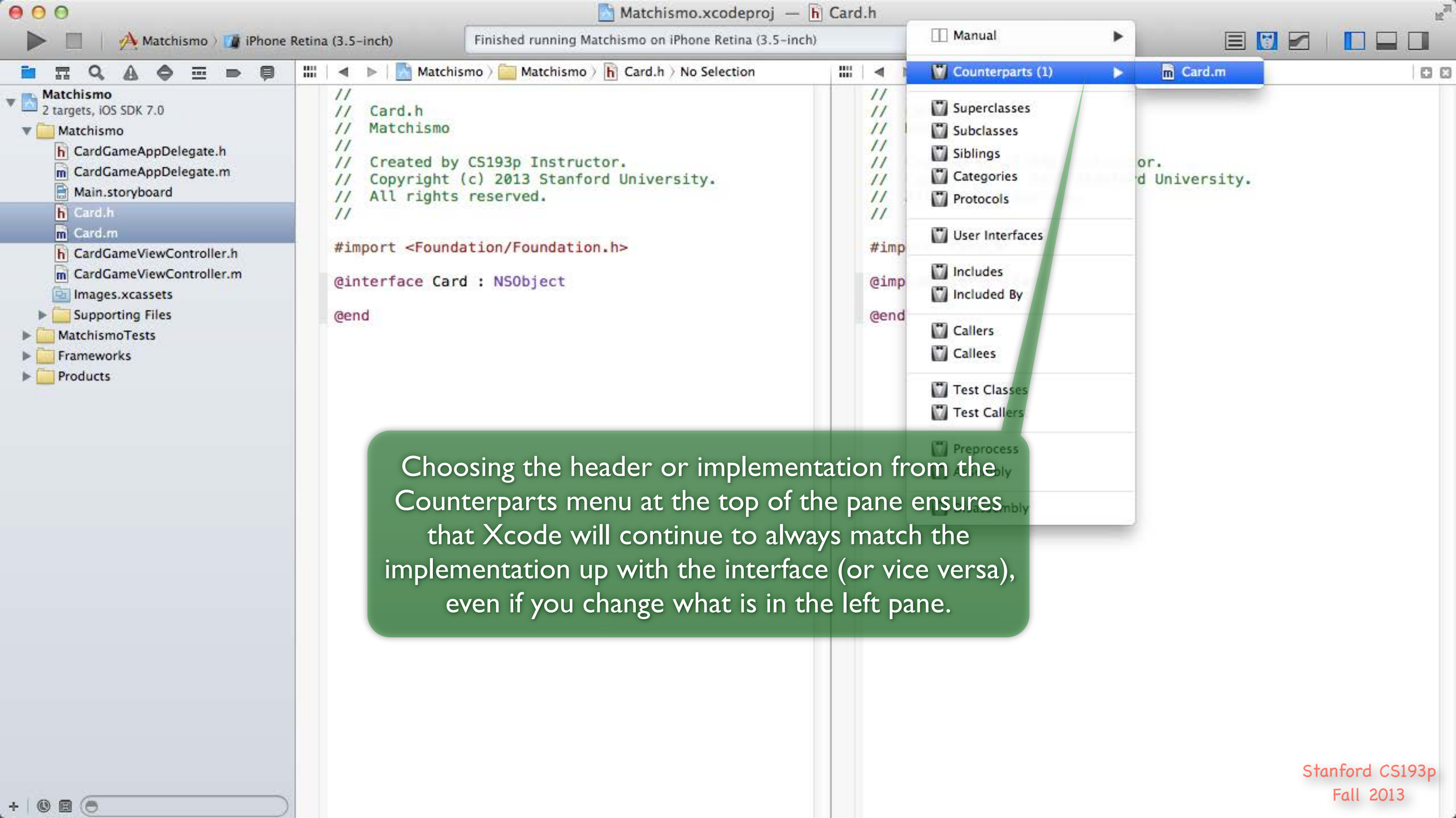
@implementation Card

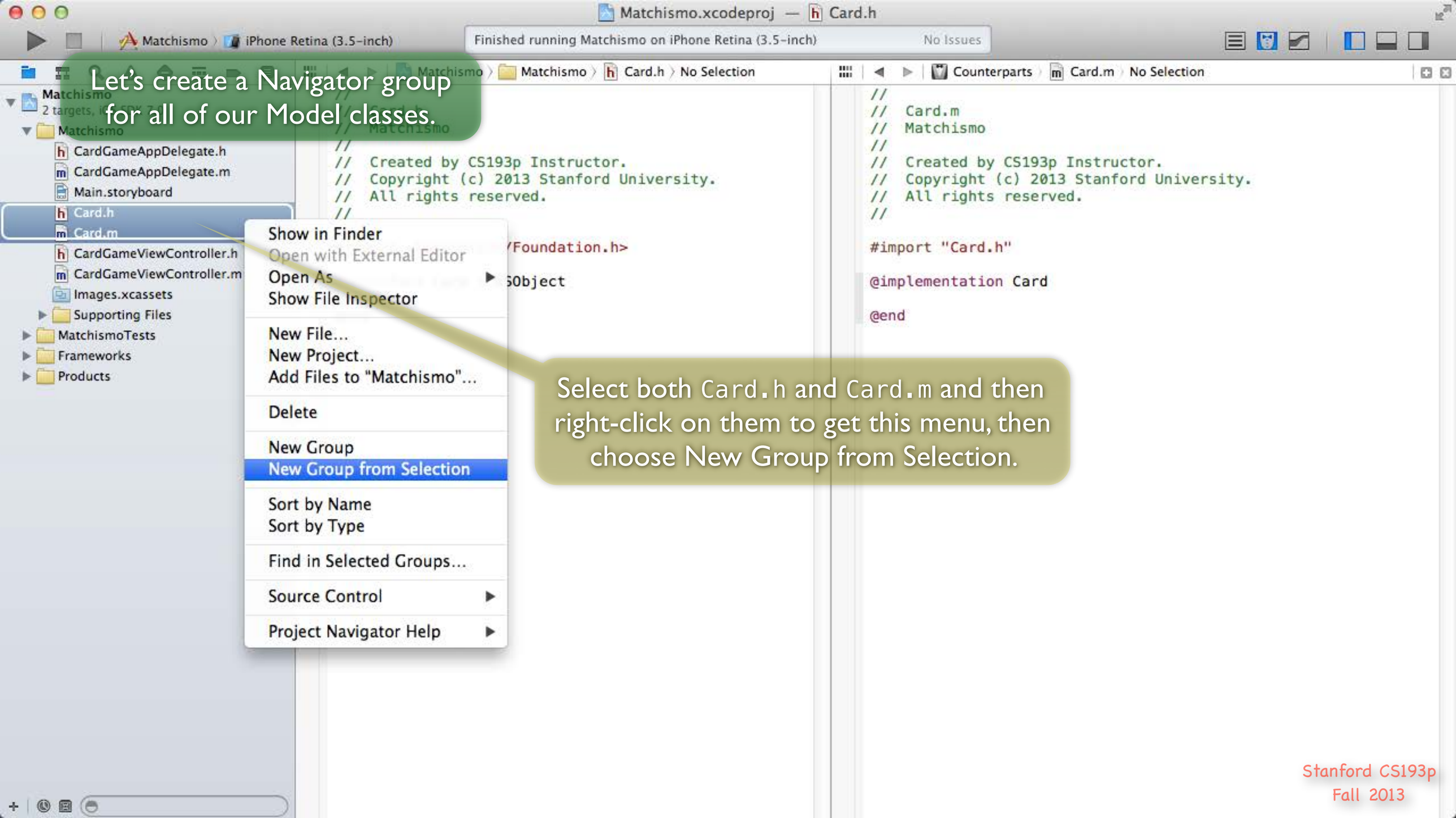
@end
```

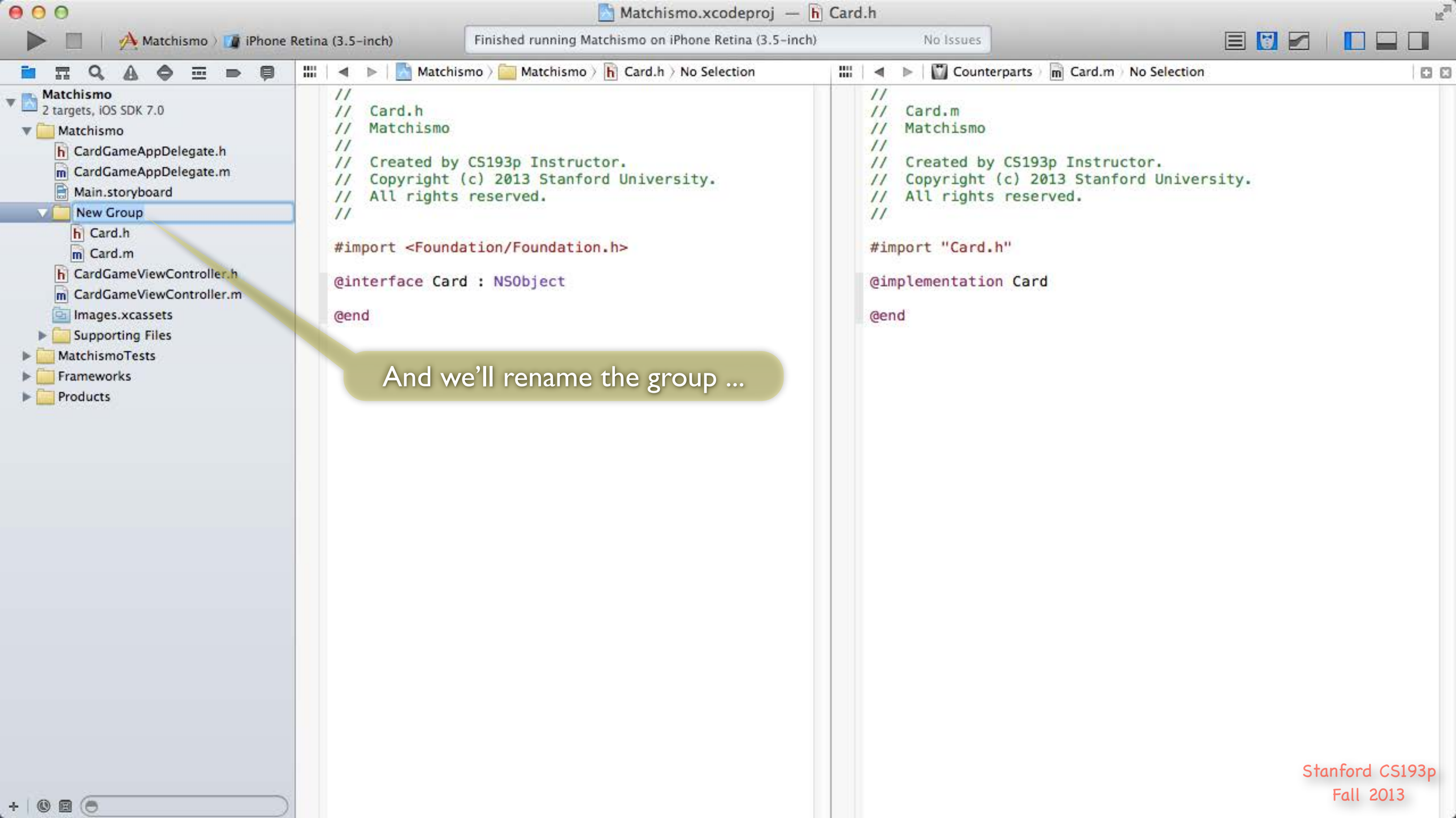
Here is a (blank) `Card` class.
You will have to go through the slides from earlier and type in the implementation of `Card`.

It is important to *type the code in*
(not copy/paste it from somewhere)
so that you gain experience with entering code in Xcode.


Card.h
Matchismo
iPhone Retina (3.5-inch)
Finished running
No Issues
Main.storyboard
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
cardback.png
cardfront.png
Supporting Files
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Counterparts
Card.m
No Selection
// Card.h
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
#import "Card.h"
@implementation Card
@end
// Card.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
#import "Card.h"
@implementation Card
@end
You can put the header file in either the left or right side of the Assistant Editor as you prefer.
You can choose which goes where by clicking on the name of the file at the top of the pane in question.


Matchismo.xcodeproj — Card.h
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchismo 〉 Card.h 〉 No Selection
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
Manual
Counterparts (1)
Card.m
Superclasses
Subclasses
Siblings
Categories
Protocols
User Interfaces
Includes
Included By
Callers
Callees
Test Classes
Test Callers
Preprocess
Choosing the header or implementation from the Counterparts menu at the top of the pane ensures that Xcode will continue to always match the implementation up with the interface (or vice versa), even if you change what is in the left pane.


Matchismo.xcodeproj — Card.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Let's create a Navigator group for all of our Model classes.
Matchismo
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Card.h › No Selection
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
Counterparts › Card.m › No Selection
// Card.m
// Matchismo
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
#import "Card.h"
@implementation Card
@end
Show in Finder
Open with External Editor
Open As
Show File Inspector
New File...
New Project...
Add Files to "Matchismo"...
Delete
New Group
New Group from Selection
Sort by Name
Sort by Type
Find in Selected Groups...
Source Control
Project Navigator Help
Select both Card.h and Card.m and then right-click on them to get this menu, then choose New Group from Selection.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
New Group
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › Matchismo › Card.h › No Selection
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
Counterparts › Card.m › No Selection
//
// Card.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@implementation Card
@end
And we'll rename the group ...

Matchismo.xcodeproj — Card.h

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

Matchismo › Matchis... › Model › Card.h › No Selection

Counterparts › Card.m › No Selection

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

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

#import <Foundation/Foundation.h>

@interface Card : NSObject

@end
```

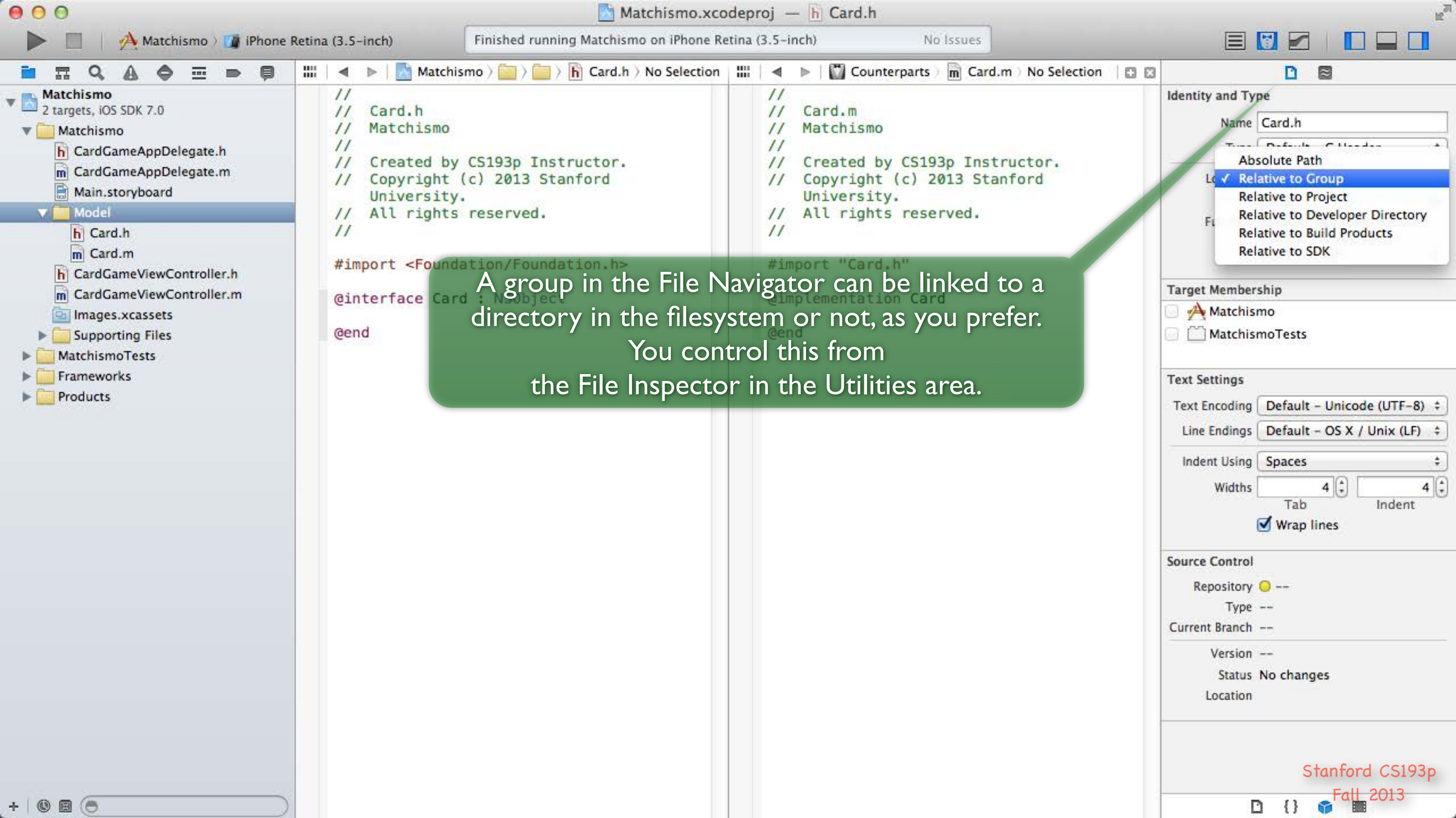
```
//
//  Card.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

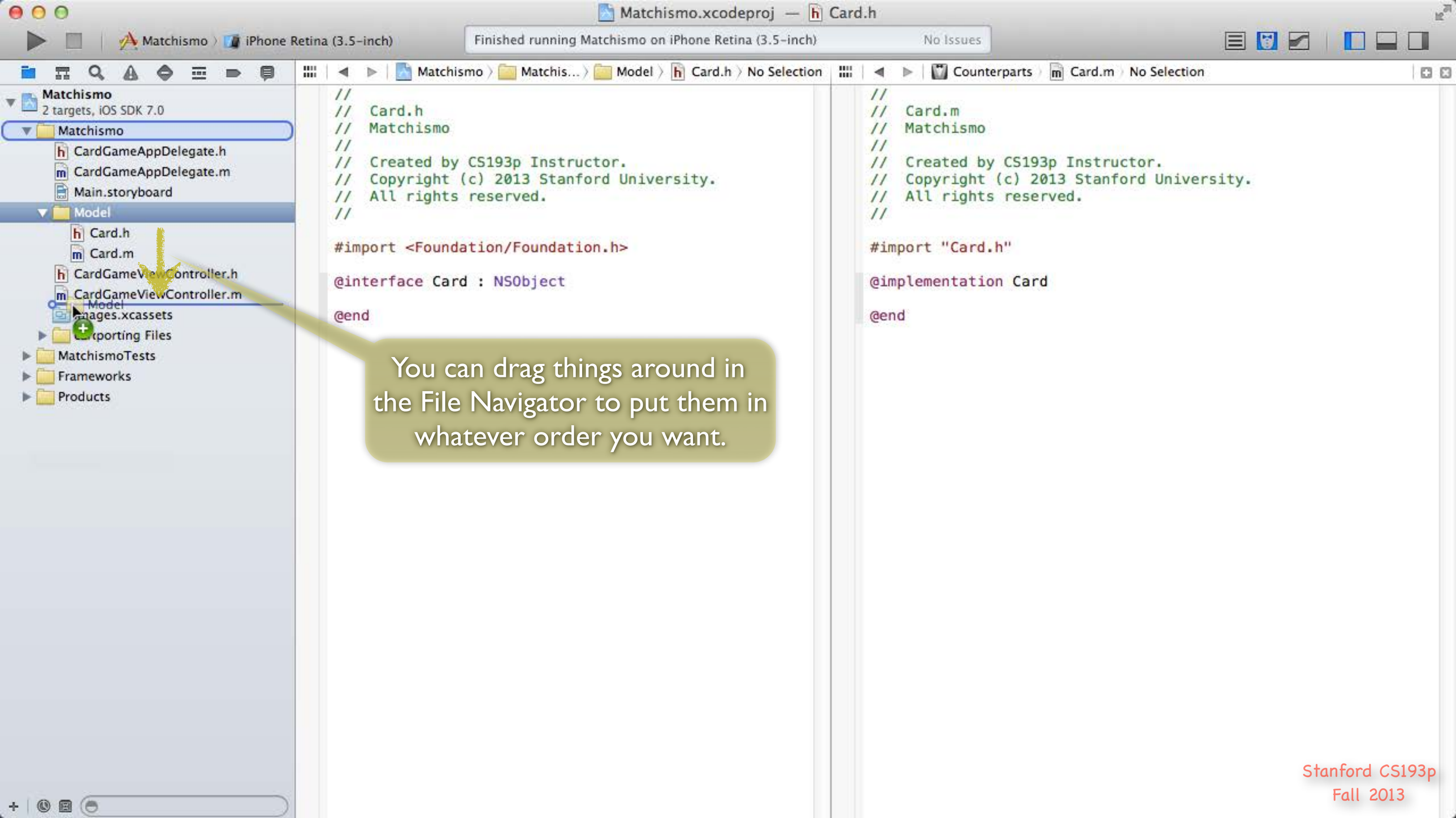
#import "Card.h"

@implementation Card

@end
```

... to Model.


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Model
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Card.h 〉 No Selection
Counterparts 〉 Card.m 〉 No Selection
// Card.h // Matchismo // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
// Card.m // Matchismo // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import "Card.h"
@implementation Card
@end
Identity and Type
Name Card.h
Absolute Path
Relative to Group
Relative to Project
Relative to Developer Directory
Relative to Build Products
Relative to SDK
Target Membership
Matchismo
MatchismoTests
Text Settings
Text Encoding Default - Unicode (UTF-8)
Line Endings Default - OS X / Unix (LF)
Indent Using Spaces
Widths 4 4
Tab
Indent
Wrap lines
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
A group in the File Navigator can be linked to a directory in the filesystem or not, as you prefer.
You control this from
the File Inspector in the Utilities area.
Stanford CS193p
Fall 2013


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
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
Model
Card.h
Card.m
CardGameViewController.h
CardGameViewController.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchis... 〉 Model 〉 Card.h 〉 No Selection
// Card.h // Matchismo // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
Counterparts 〉 Card.m 〉 No Selection
// Card.m // Matchismo // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import "Card.h"
@implementation Card
@end
You can drag things around in the File Navigator to put them in whatever order you want.
Stanford CS193p
Fall 2013

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
  - CardGameAppDelegate.h
  - CardGameAppDelegate.m
  - Main.storyboard
  - CardGameViewController.h
  - CardGameViewController.m
  - Model
    - Card.h
    - Card.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › Matchis... › Model › Card.h › No Selection

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

#import <Foundation/Foundation.h>

@interface Card : NSObject

@end
```

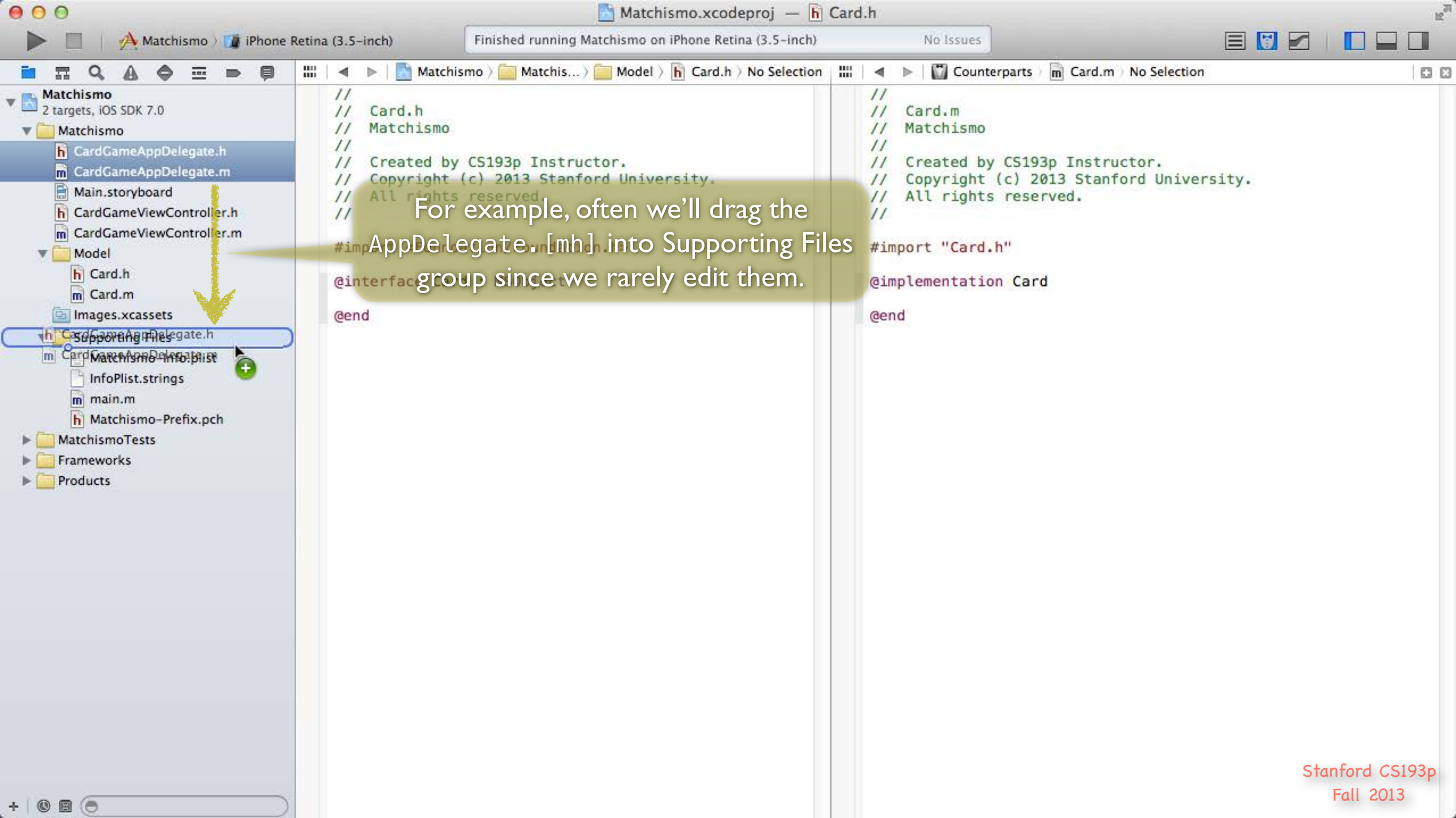
Counterparts › Card.m › No Selection

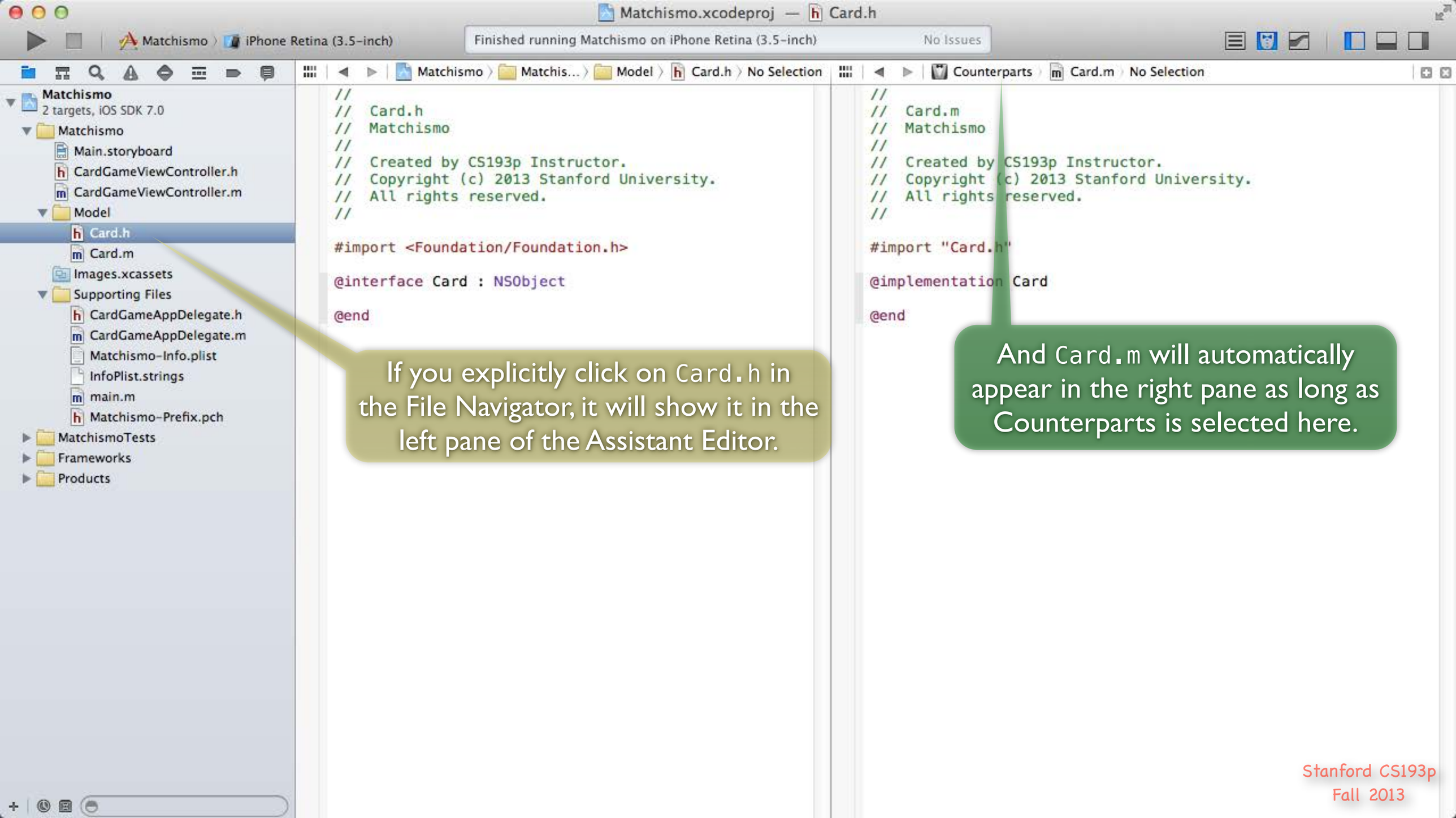
```
//
//  Card.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

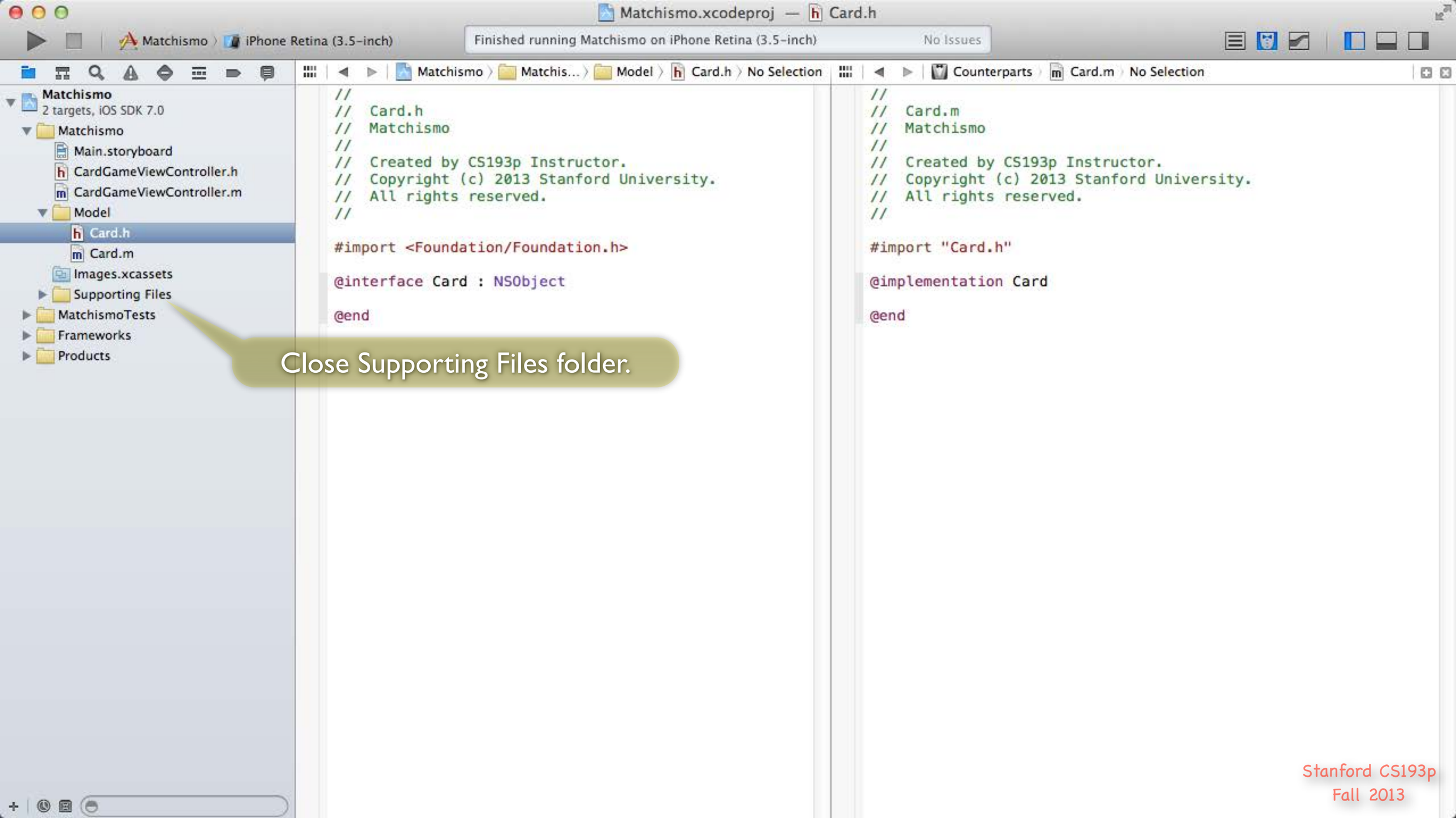
#import "Card.h"

@implementation Card

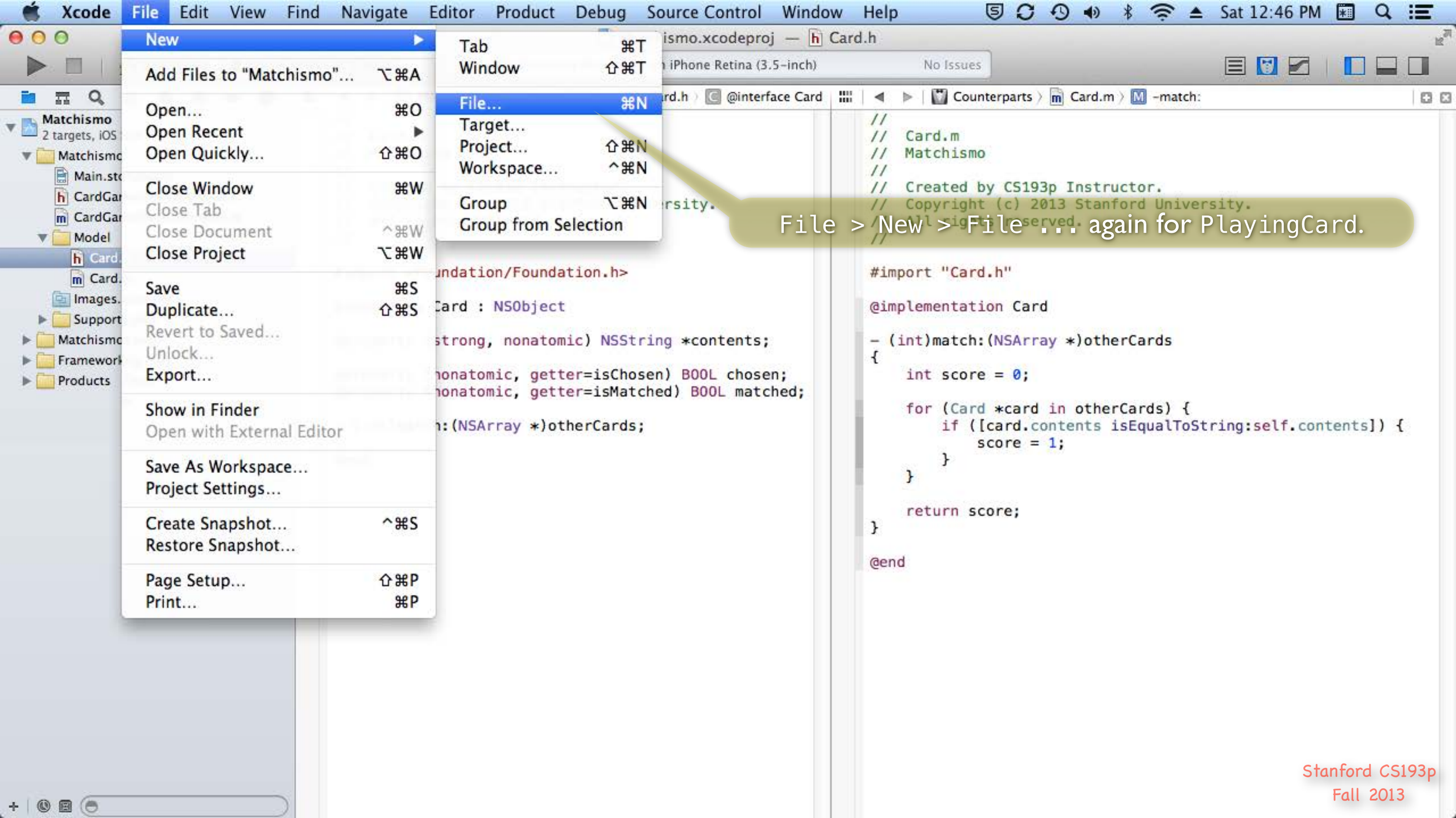
@end
```

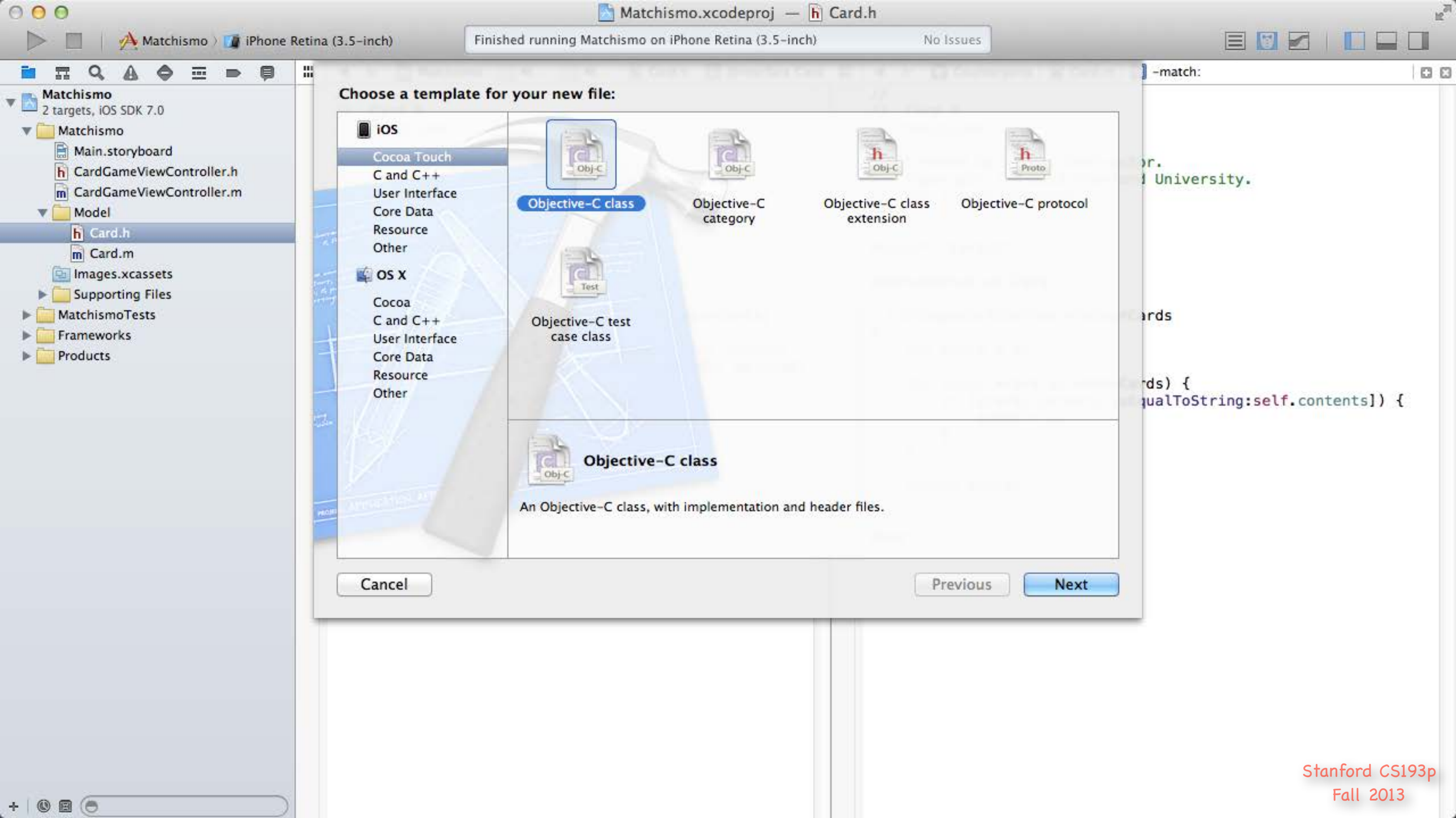

Matchismo.xcodeproj — Card.h
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
Card.h
Card.m
Images.xcassets
Supporting Files
Matchismo-Info.plist
InfoPlist.strings
main.m
Matchismo-Prefix.pch
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchis... 〉 Model 〉 Card.h 〉 No Selection
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
@end
Counterparts 〉 Card.m 〉 No Selection
//
// Card.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@implementation Card
@end
For example, often we'll drag the AppDelegate.[mh] into Supporting Files group since we rarely edit them.
Stanford CS193p
Fall 2013

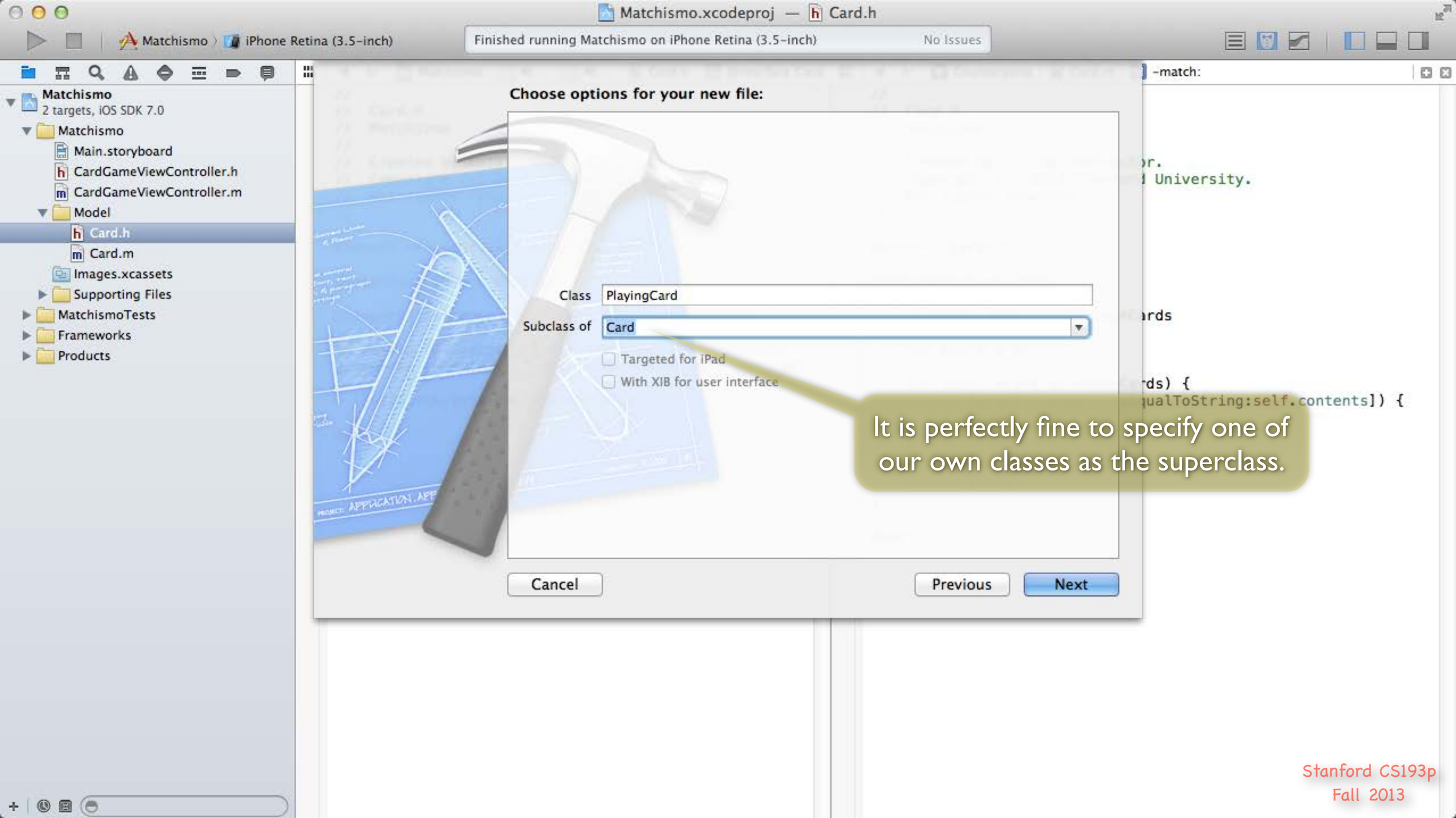

Matchismo.xcodeproj — Card.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
Card.h
Card.m
Images.xcassets
Supporting Files
CardGameAppDelegate.h
CardGameAppDelegate.m
Matchismo-Info.plist
InfoPlist.strings
main.m
Matchismo-Prefix.pch
MatchismoTests
Frameworks
Products
Matchismo › Matchis... › Model › Card.h › No Selection
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
If you explicitly click on Card.h in the File Navigator, it will show it in the left pane of the Assistant Editor.
Counterparts › Card.m › No Selection
//
// Card.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@implementation Card
@end
And Card.m will automatically appear in the right pane as long as Counterparts is selected here.
Stanford CS193p
Fall 2013

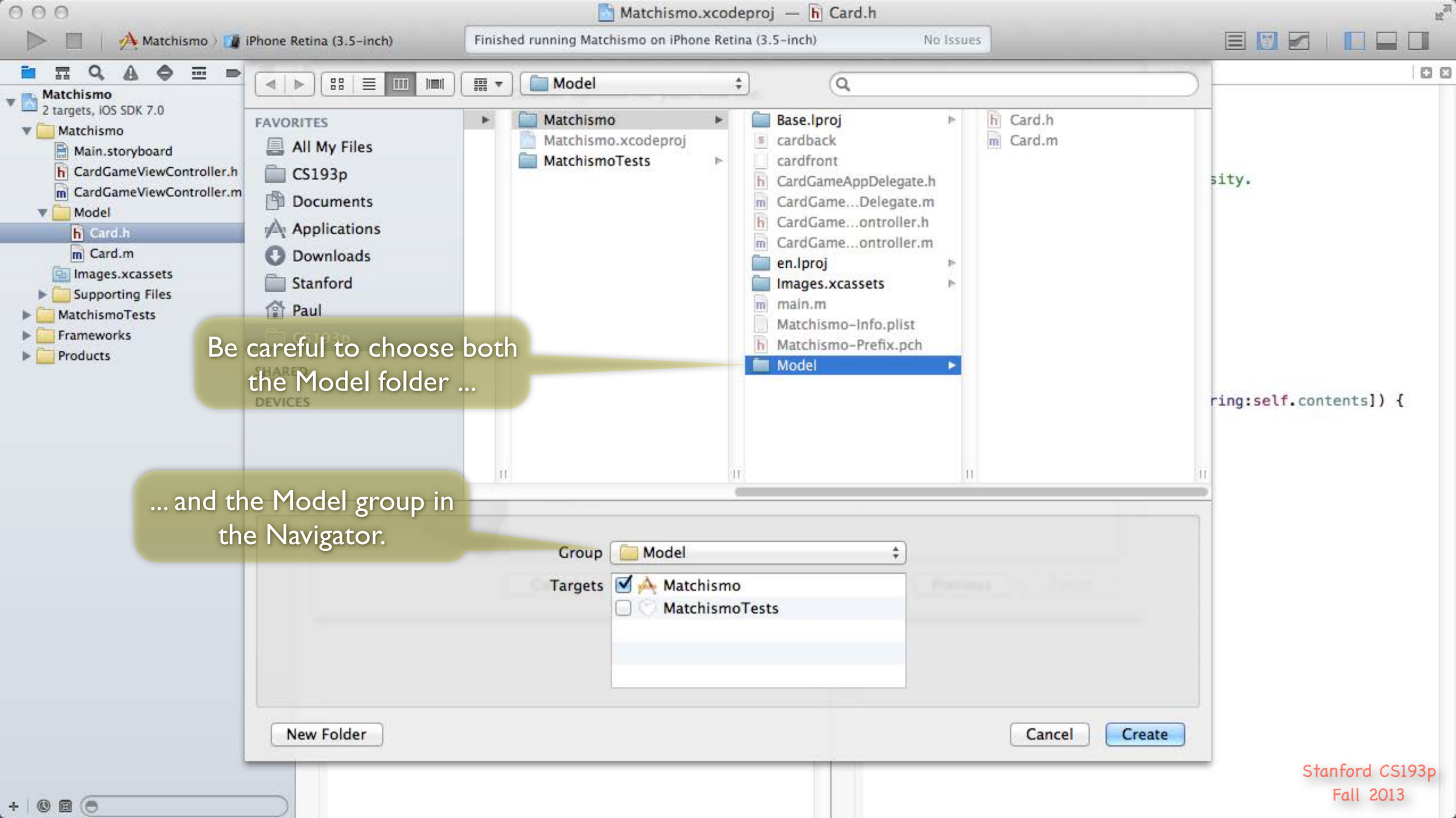

Matchismo.xcodeproj — Card.h
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
Card.h
Card.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo 〉 Matchis... 〉 Model 〉 Card.h 〉 No Selection
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
Counterparts 〉 Card.m 〉 No Selection
//
// Card.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Card.h"
@implementation Card
@end
Close Supporting Files folder.

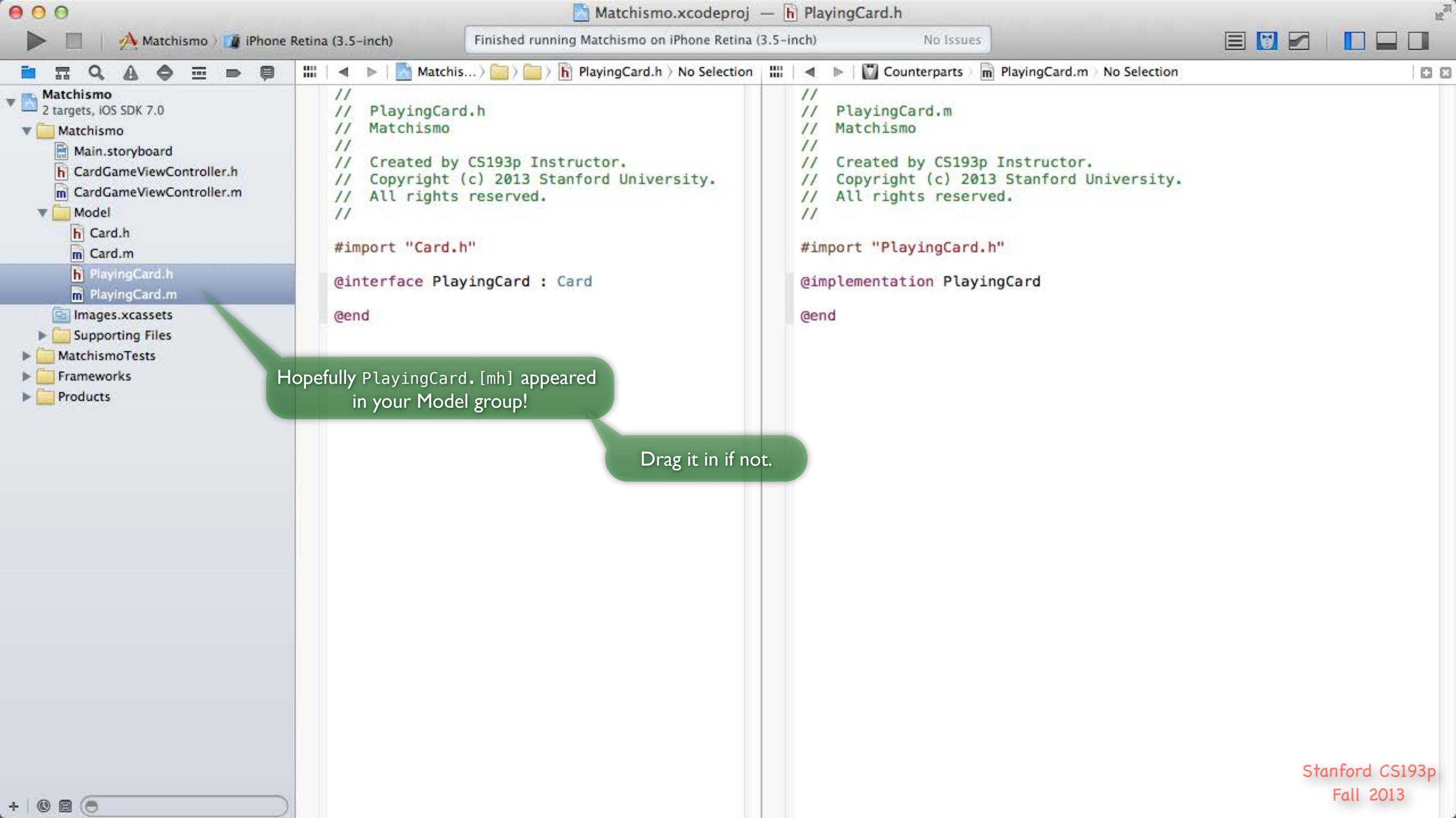

Matchismo.xcodeproj — Card.h
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
Card.h
Card.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchismo › M... › M... › Card.h › @interface Card
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@property (strong, nonatomic) NSString *contents;
@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
Type in the code for Card.[mh].
Now let's move on to creating templates for the Deck, PlayingCard and PlayingCardDeck classes.
Stanford CS193p
Fall 2013


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Sat 12:46 PM
New
Add Files to "Matchismo"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close Document ^⌘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
File > New > File ... again for PlayingCard.
Matchismo
2 targets, iOS
Model
Card.h
Card.m
Images.
Support
Products
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
Stanford CS193p
Fall 2013


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
Card.h
Card.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
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
-match:
r.
University.
rds
rds) {
qualToString:self.contents]) {
Stanford CS193p
Fall 2013


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
Card.h
Card.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Choose options for your new file:
Class
PlayingCard
Subclass of
Card
Targeted for iPad
With XIB for user interface
Cancel
Previous
Next
It is perfectly fine to specify one of our own classes as the superclass.
-match:
University.
rds
rds) {
qualToString:self.contents]) {
Stanford CS193p
Fall 2013


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
Card.h
Card.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Model
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
Paul
SHARED
DEVICES
Matchismo
Matchismo.xcodeproj
MatchismoTests
Base.lproj
cardback
cardfront
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
Be careful to choose both the Model folder ...
... and the Model group in the Navigator.
Group
Model
Targets
Matchismo
MatchismoTests
New Folder
Cancel
Create
sity.
ring:self.contents]) {
Stanford CS193p
Fall 2013


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
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
Matchis… 〉 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
@end
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
@end
Hopefully PlayingCard.[mh] appeared in your Model group!
Drag it in if not.

Matchismo.xcodeproj — PlayingCard.h

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

Playing... › @interface PlayingCard | Counterparts › PlayingCard.m › @implementation PlayingCard

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

PlayingCard.h

```
//
//  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
```

PlayingCard.m

```
//
//  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"];
}
```

Type in the code for `PlayingCard.[mh]`.

All the code doesn't fit here, so use the other lecture slides to enter this code.

Matchismo.xcodeproj — Deck.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
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Matchismo › Deck.h › @interface Deck

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

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

@interface Deck : NSObject

- (void)addCard:(Card *)card atTop:(BOOL)atTop;
- (void)addCard:(Card *)card;

- (Card *)drawRandomCard;

@end
```

Counterparts › Deck.m › @implementation Deck

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

#import "Deck.h"

@interface Deck()
@property (strong, nonatomic) NSMutableArray *cards; // of Card
@end

@implementation Deck

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

- (void)addCard:(Card *)card atTop:(BOOL)atTop
{
    if (atTop) {
        [self.cards insertObject:card atIndex:0];
    } else {
        [self.cards addObject:card];
    }
}

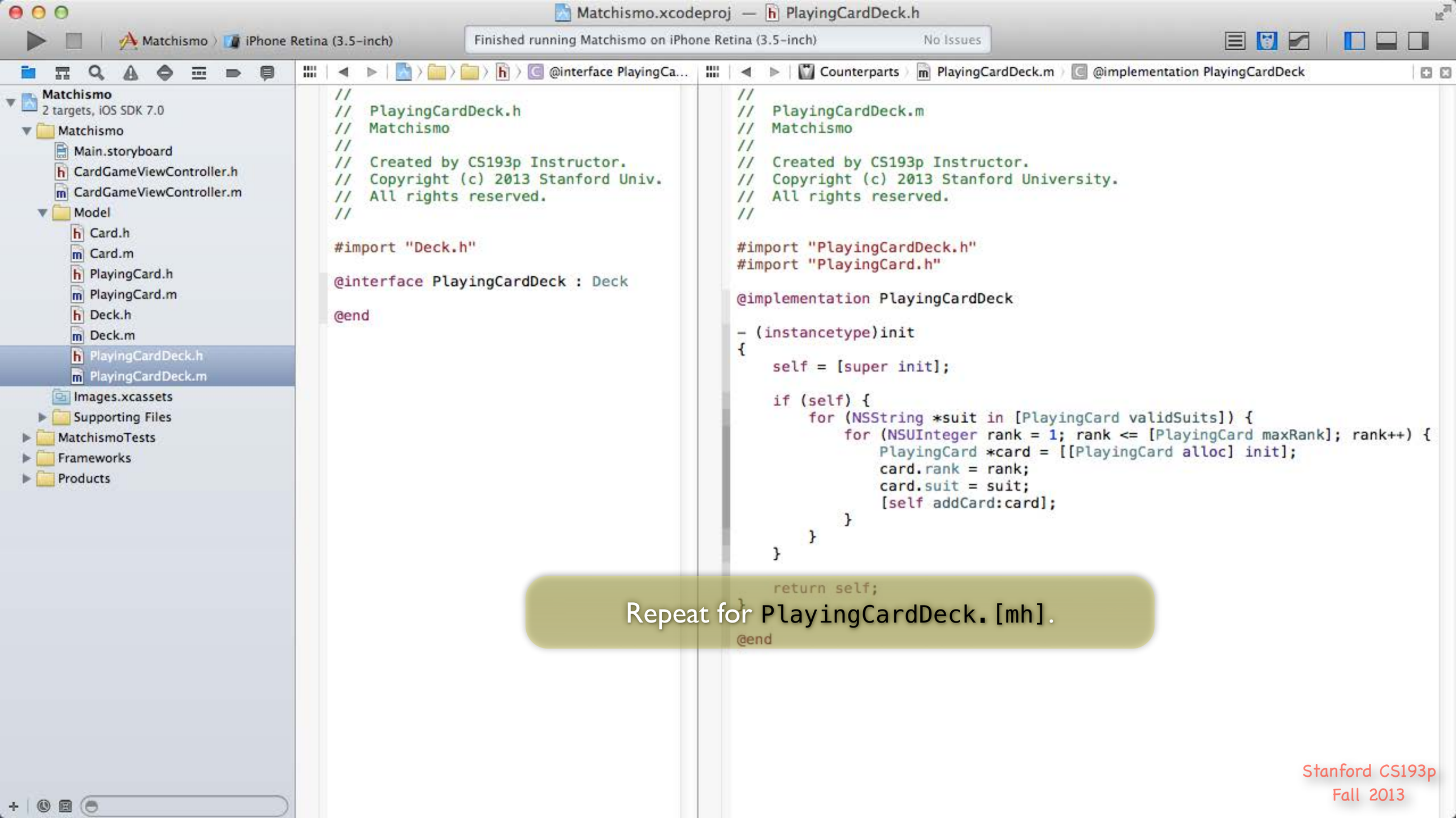
- (void)addCard:(Card *)card
{...}

- (Card *)drawRandomCard
{
    Card *randomCard = nil;

    if ([self.cards count]) {
        unsigned index = arc4random() % [self.cards count];
        randomCard = self.cards[index];
        [self.cards removeObjectAtIndex:index];
```

Repeat for `Deck.[mh]`.

All the code doesn't fit here, so use the other lecture slides to enter this code.

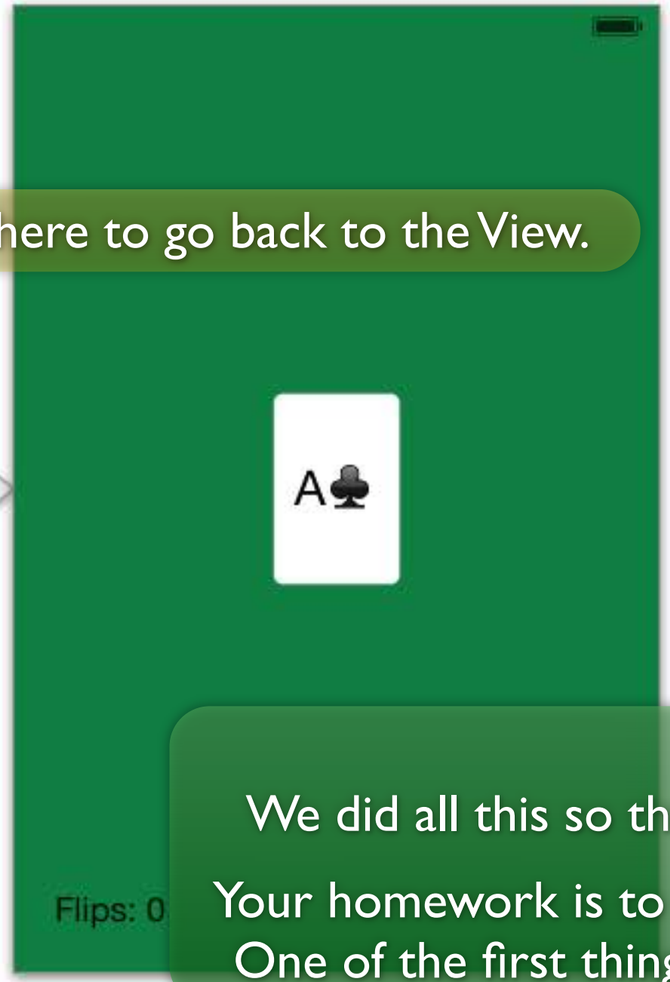

Matchismo.xcodeproj — PlayingCardDeck.h
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Matchismo
2 targets, iOS SDK 7.0
Matchismo
Main.storyboard
CardGameViewController.h
CardGameViewController.m
Model
Card.h
Card.m
PlayingCard.h
PlayingCard.m
Deck.h
Deck.m
PlayingCardDeck.h
PlayingCardDeck.m
Images.xcassets
Supporting Files
MatchismoTests
Frameworks
Products
@interface PlayingCa...
//
// PlayingCardDeck.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford Univ.
// All rights reserved.
//
#import "Deck.h"
@interface PlayingCardDeck : Deck
@end
Counterparts
PlayingCardDeck.m
@implementation PlayingCardDeck
//
// PlayingCardDeck.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "PlayingCardDeck.h"
#import "PlayingCard.h"
@implementation PlayingCardDeck
- (instancetype)init
{
self = [super init];
if (self) {
for (NSString *suit in [PlayingCard validSuits]) {
for (NSUInteger rank = 1; rank <= [PlayingCard maxRank]; rank++) {
PlayingCard *card = [[PlayingCard alloc] init];
card.rank = rank;
card.suit = suit;
[self addCard:card];
}
}
}
return self;
}
@end
Repeat for PlayingCardDeck.[mh].
Stanford CS193p
Fall 2013

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
    - Card.h
    - Card.m
    - PlayingCard.h
    - PlayingCard.m
    - Deck.h
    - Deck.m
    - PlayingCardDeck.h
    - PlayingCardDeck.m
  - Images.xcassets
  - Supporting Files
- MatchismoTests
- Frameworks
- Products

Click here to go back to the View.




```
//
//  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

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

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

@end
```

Whew!
We did all this so that we could have each card not be A♣.

Your homework is to make each flip draw a new random card.
One of the first things you'll want is a @property for a Deck.

Good luck!

Stanford CS193p
Fall 2013

Coming Up

- Needs more Card Game!

  Your homework will be to have that single card flip through an entire `Deck` of `PlayingCards`.
  Next week we'll make multiple cards and put in logic to match them against each other.

- Also next week …

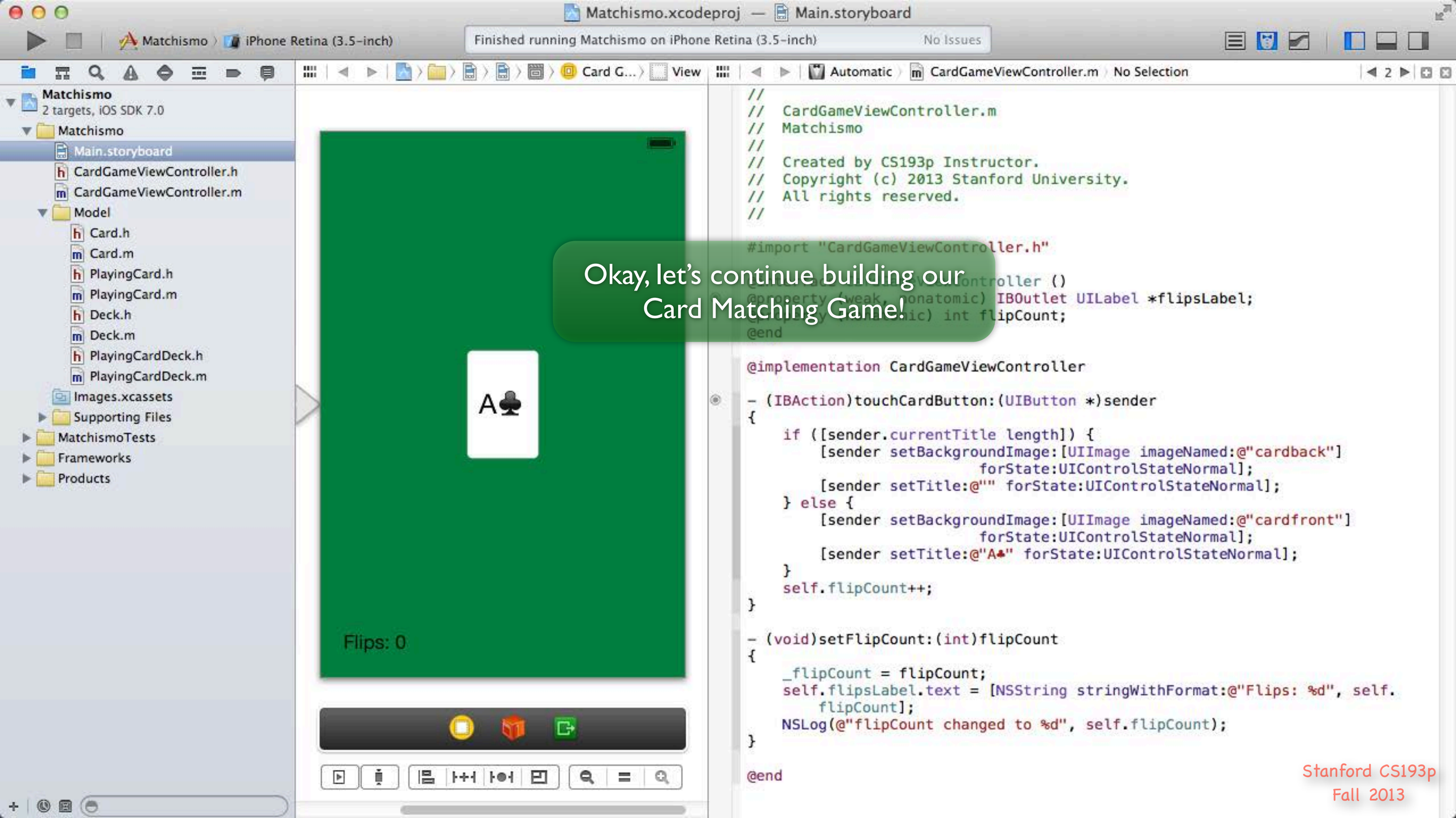
  Objective-C language in depth
  Foundation classes: arrays, dictionaries, strings, etc.
  Dynamic vs. static typing
  Protocols, categories and much, much more!

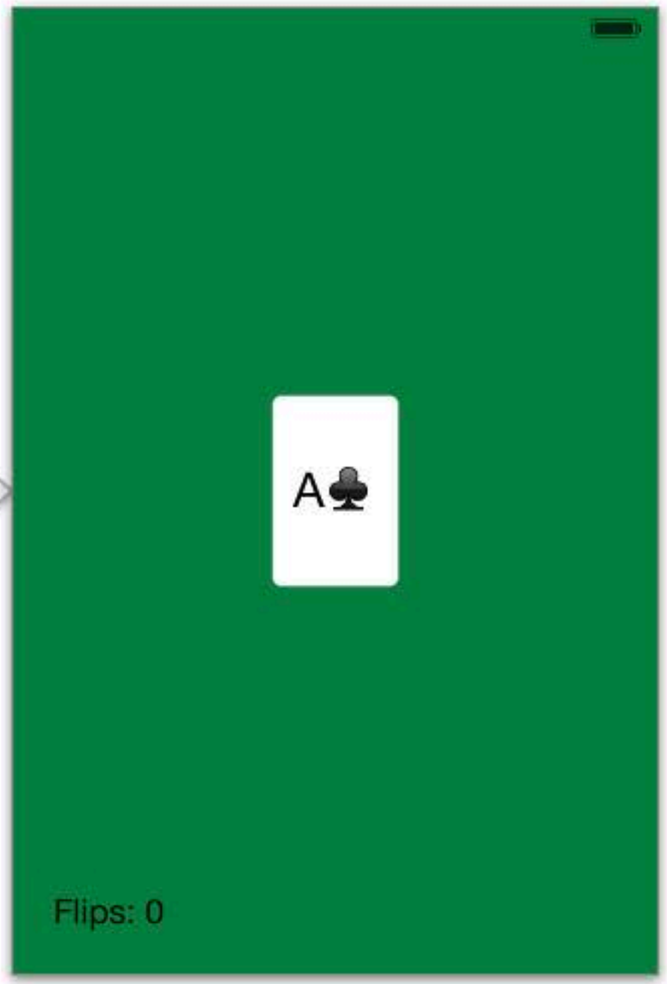
Stanford CS193p

Developing Applications for iOS
Fall 2013-14



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);
}

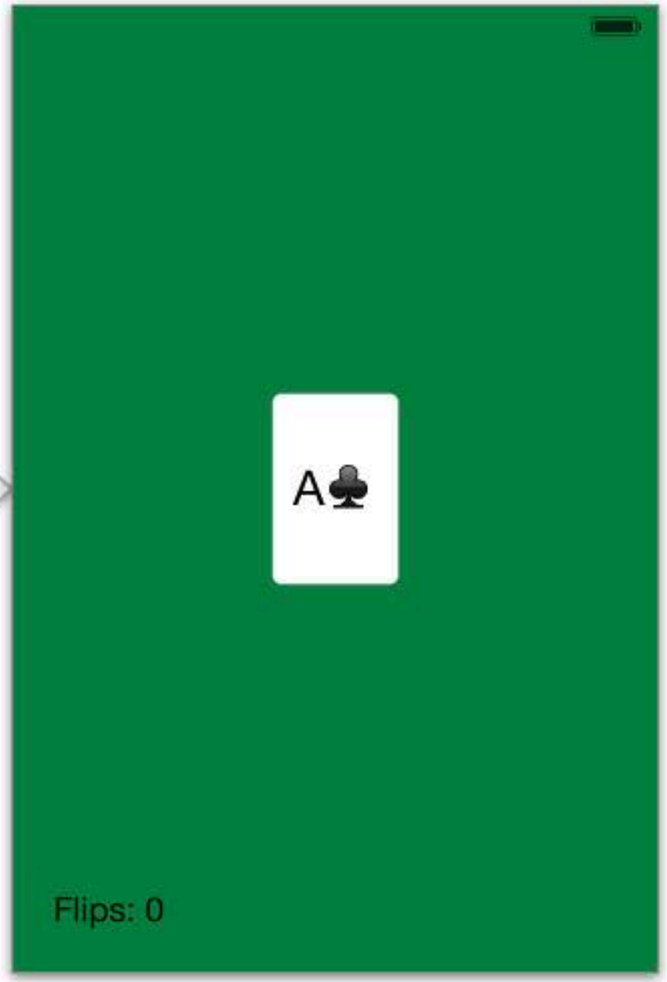
@end
```

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




```
#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);
}

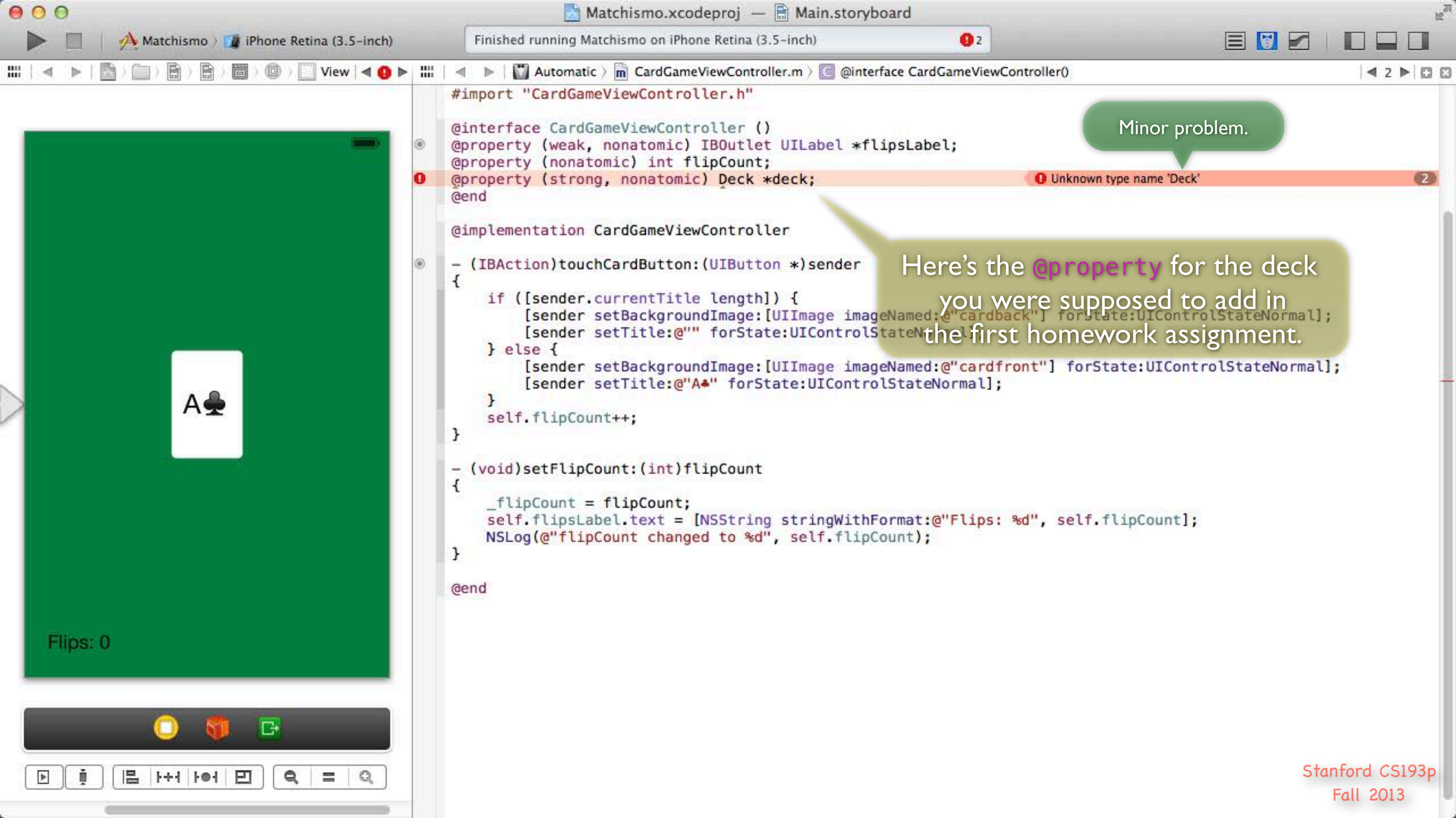
@end
```

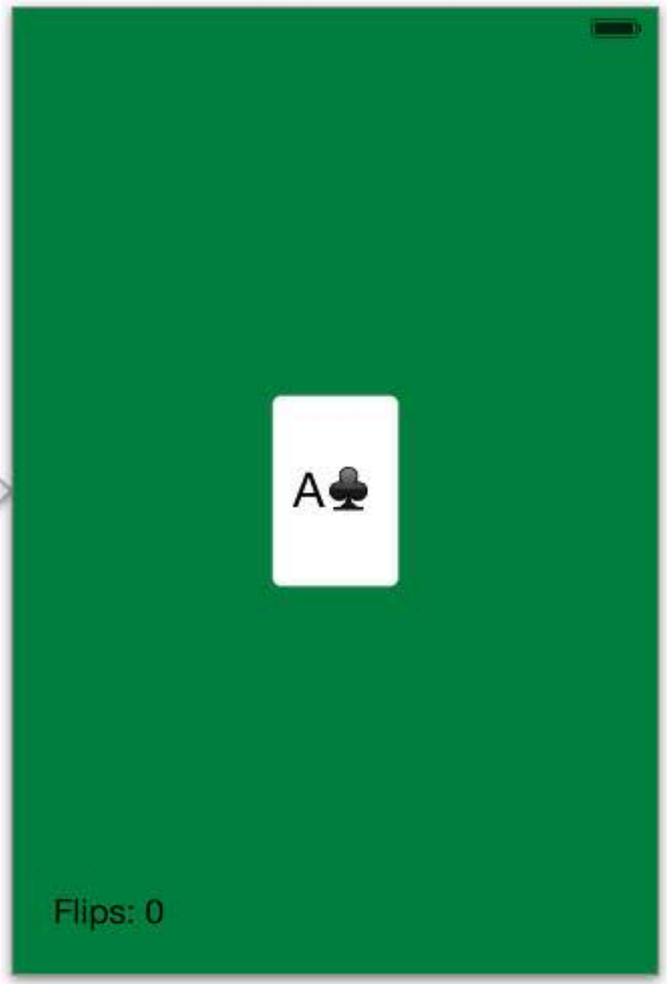
Scrolled down.

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.

Stanford CS193p
Fall 2013


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
A♣
Flips: 0
Minor problem.
Here's the @property for the deck you were supposed to add in the first homework assignment.
Stanford CS193p
Fall 2013




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

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

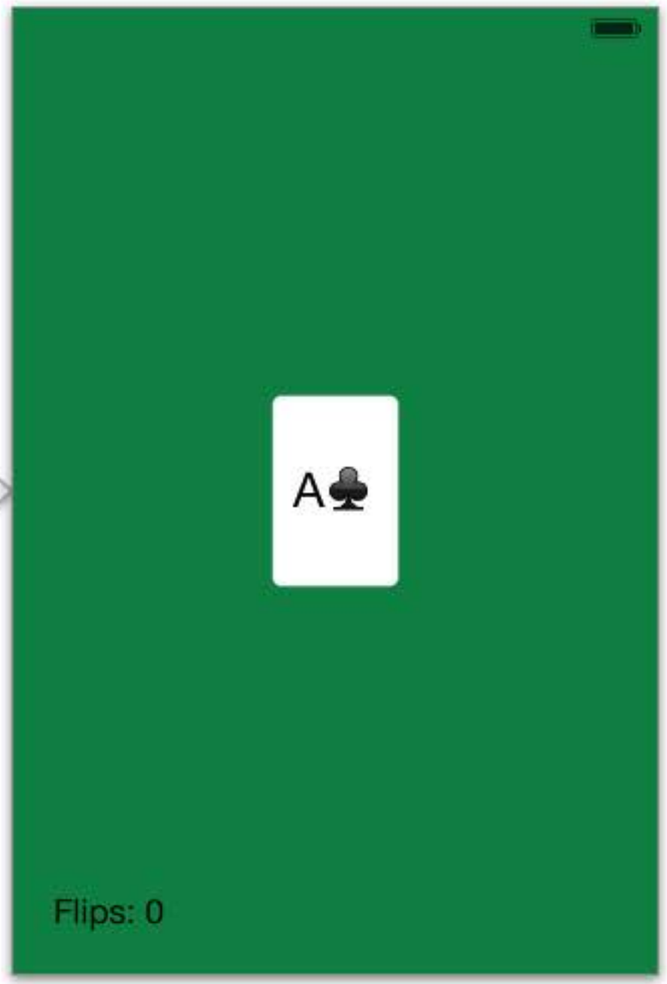
@implementation CardGameViewController

- (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.




```
#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];
}

- (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);
}

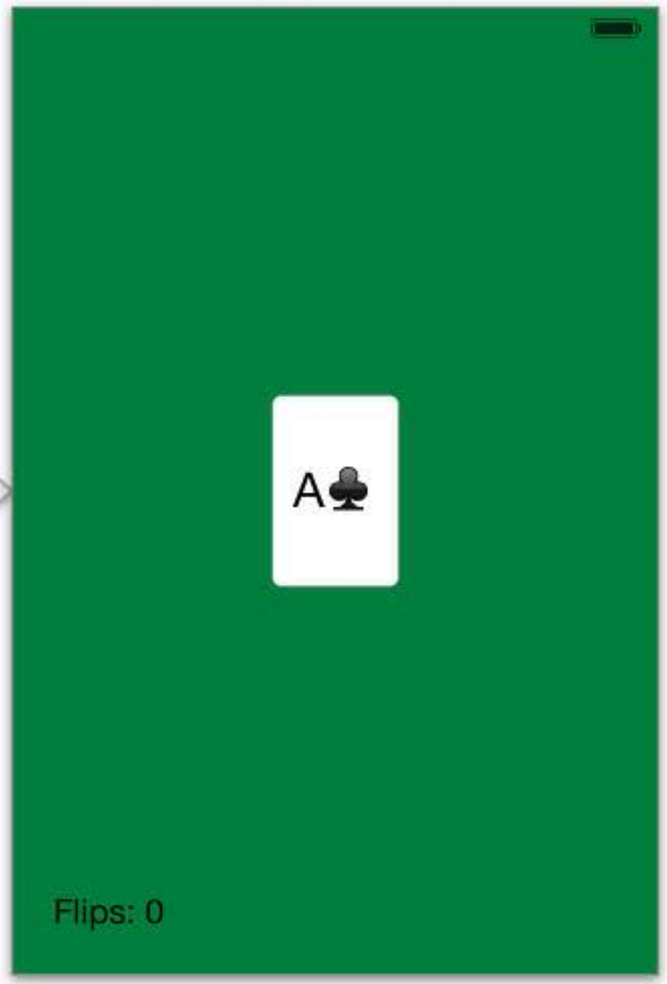
@end
```

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.

Use of undeclared identifier 'PlayingCardDeck'




```
#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++;
}

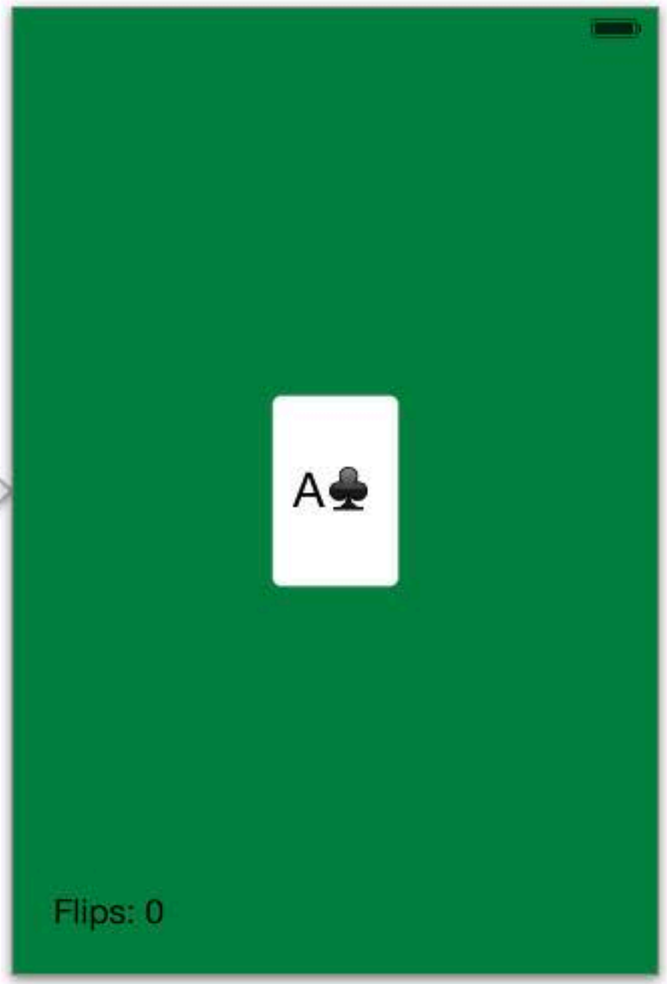
- (void)setFlipCount:(int)flipCount
{
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

@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).




```
#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;
}

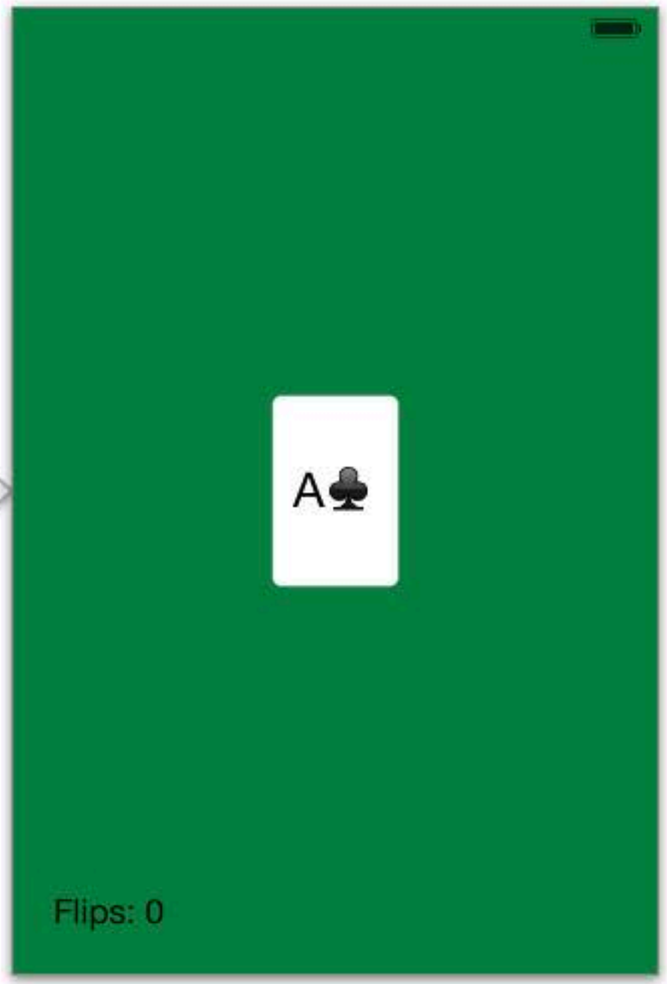
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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 ...




```
#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;
}

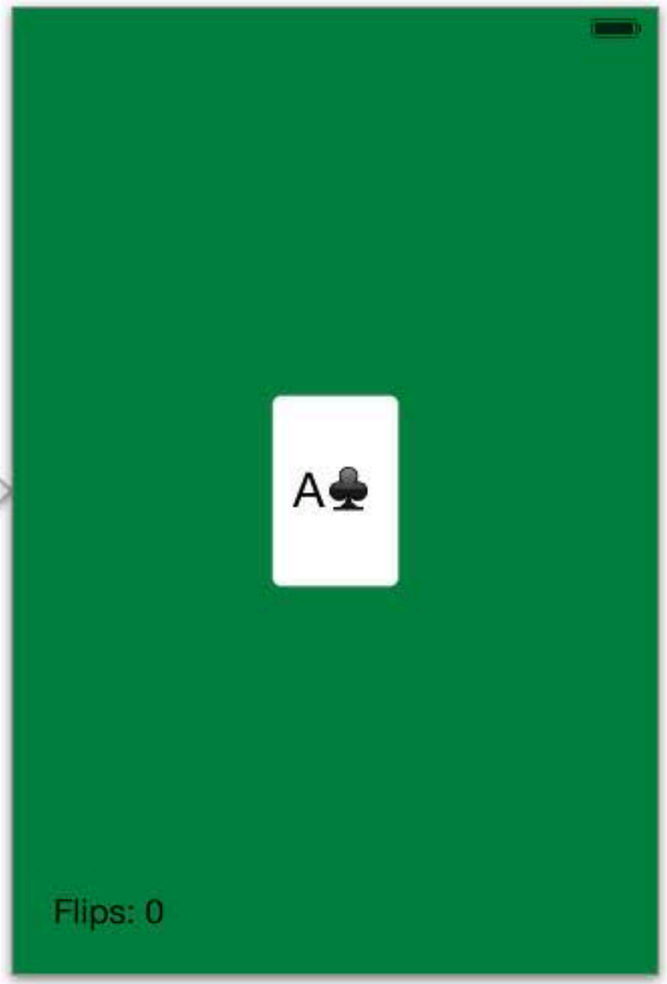
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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);
}
```

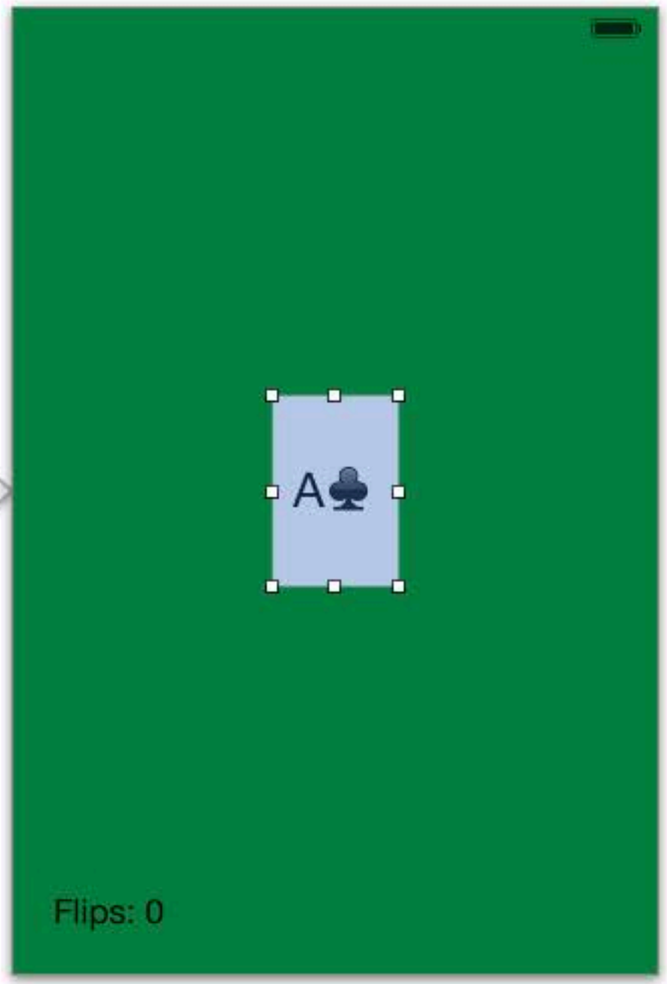
But how do we handle running out of cards?
Easy.
Just don't flip to the front if the deck is empty
(which we know if `drawRandomCard` returns `nil`).

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: | 2




```
@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
kground: cardfront
Shadow Offset: 0.0 (Width) 0.0 (Height)
Reverses On Highlight
Drawing: 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



Flips: 0

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
```

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

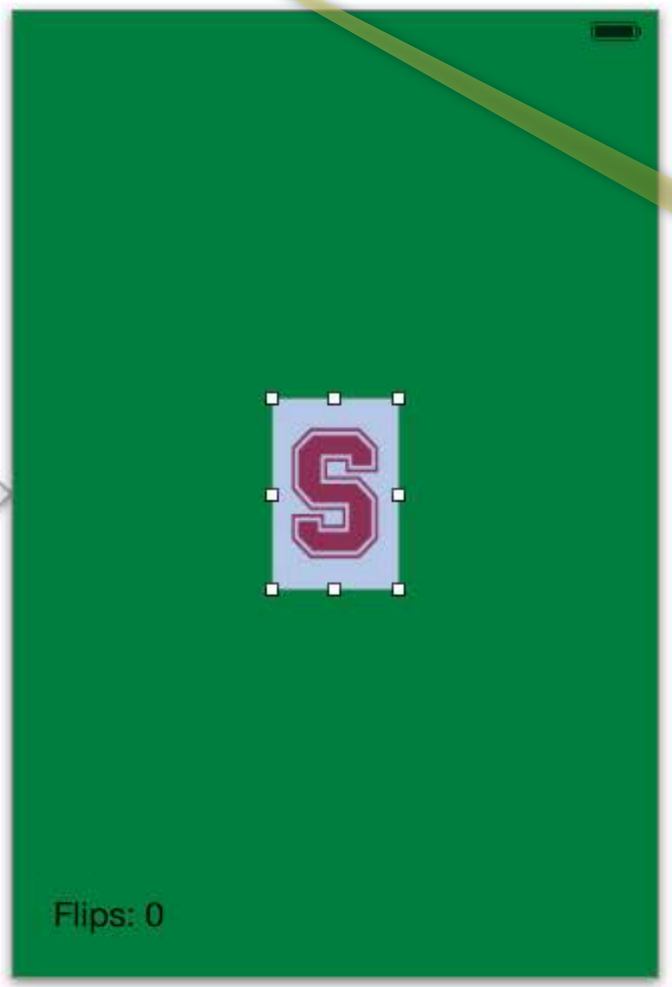
Alignment: Horizontal, Vertical

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);
}

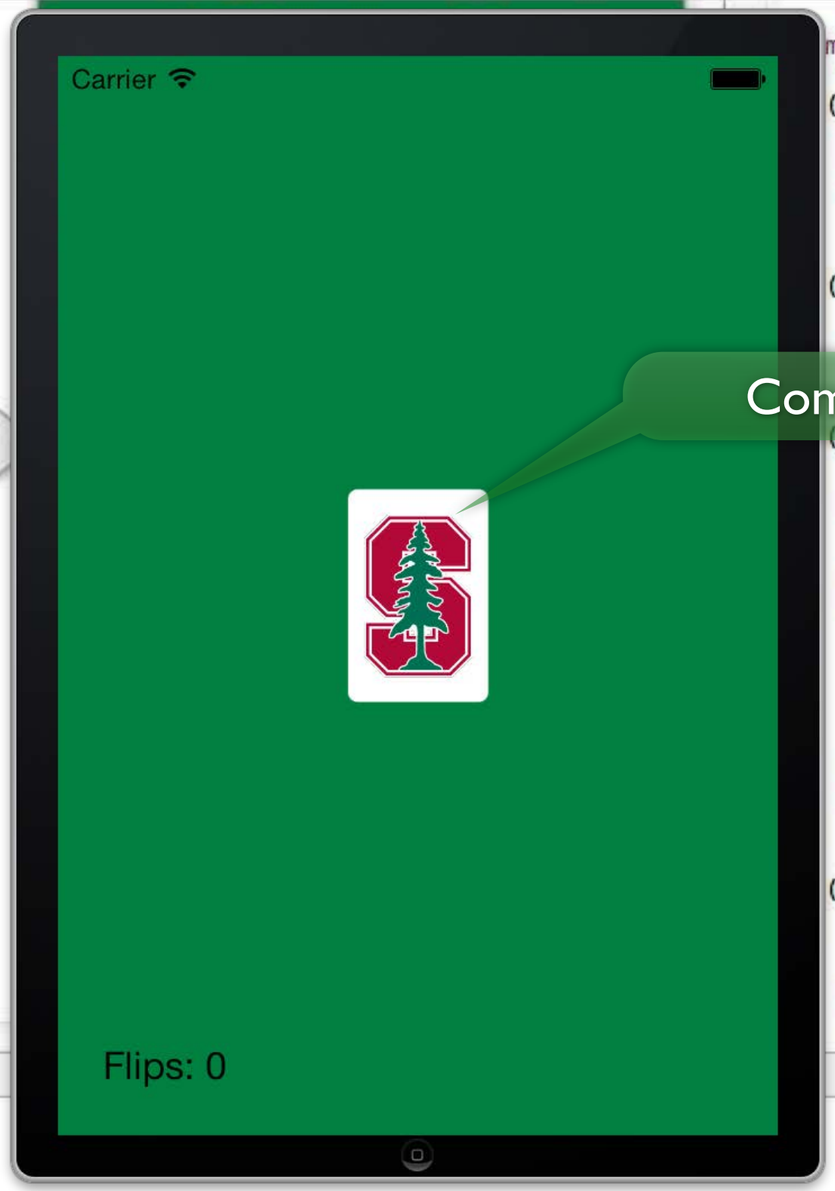
@end
```

Stanford CS193p
Fall 2013

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: | 2



Comes up face down.

```
mplementation CardGameViewController

(Deck *)deck

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


(Deck *)createDeck

  return [[PlayingCardDeck alloc] init];

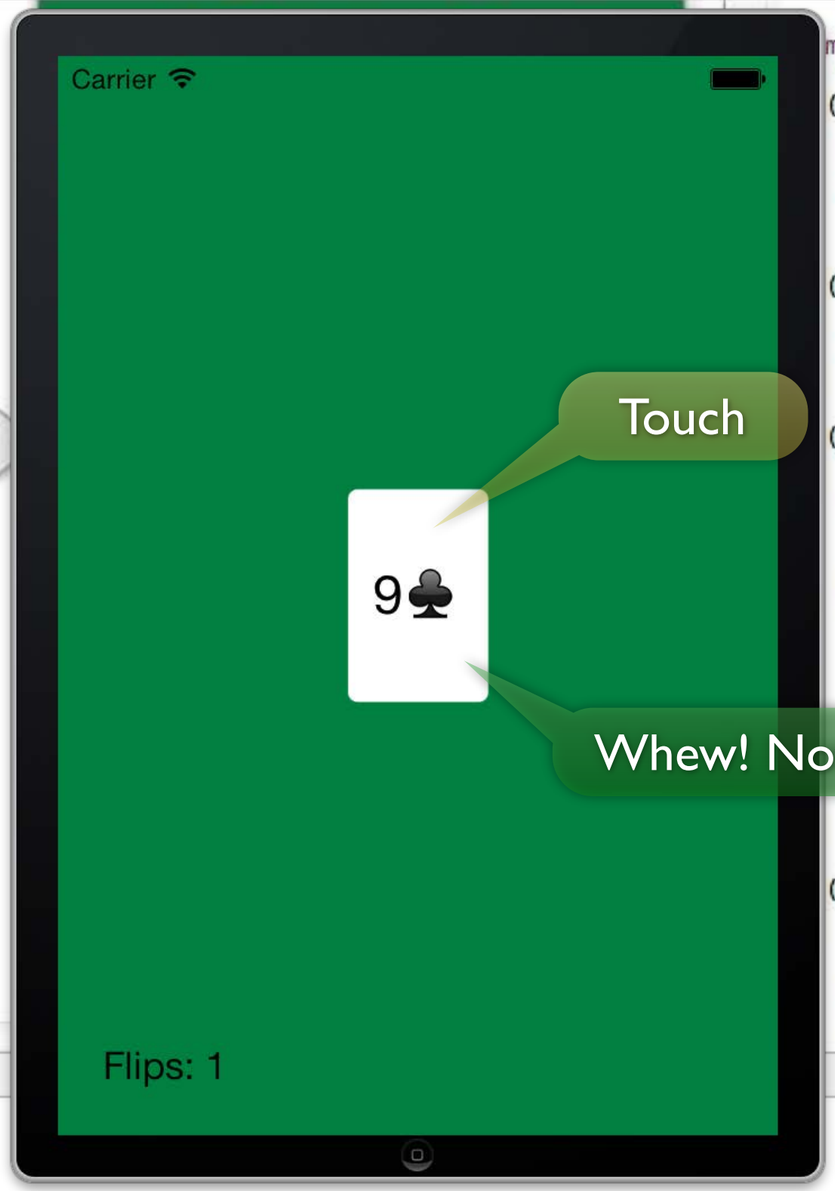
(IBAction)touchCardButton:(UIButton *)sender

  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
```

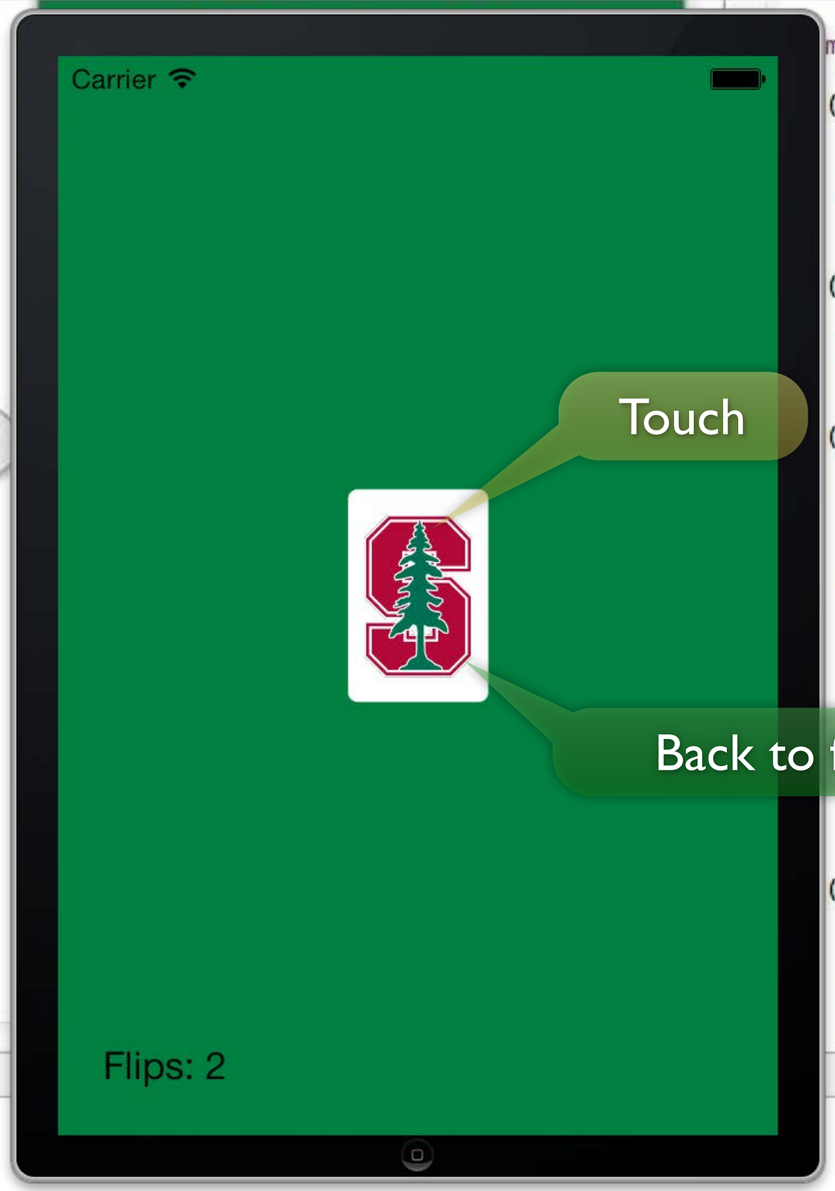
Auto | All Output

Stanford CS193p
Fall 2013

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++;

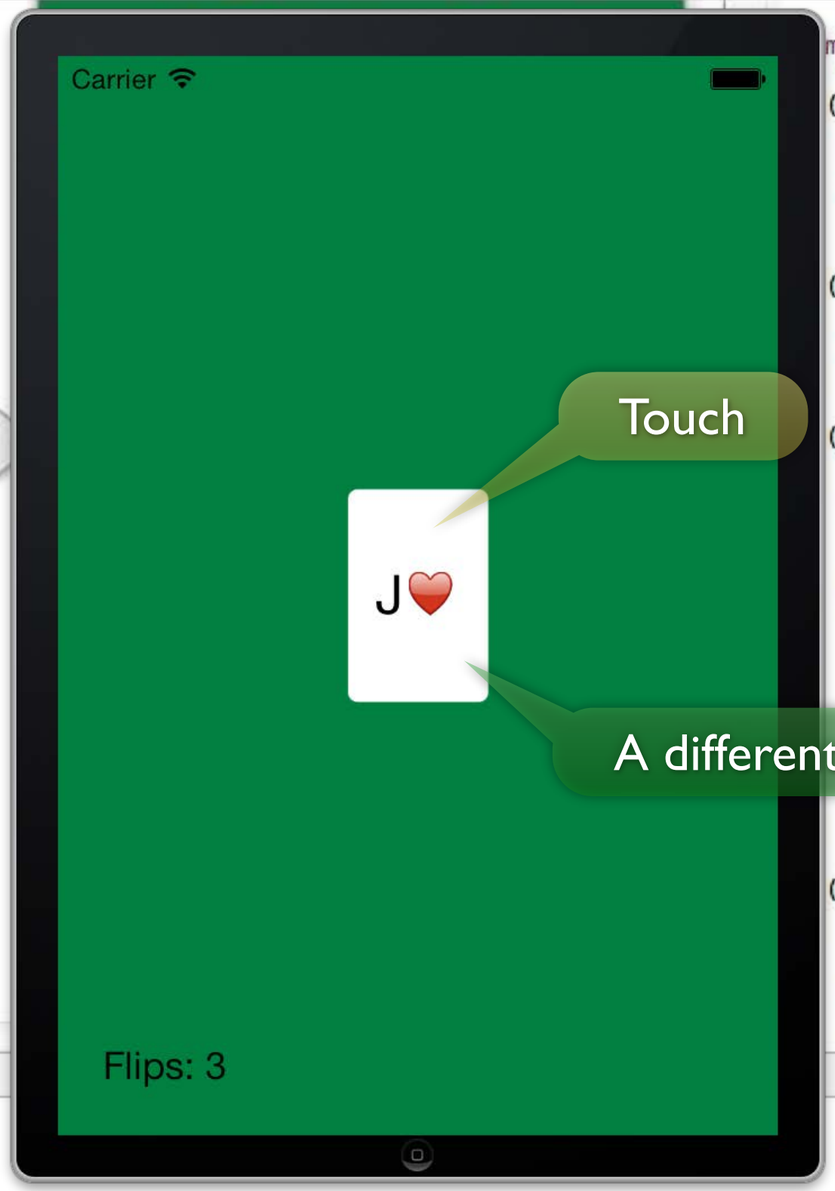

(void)setFlipCount:(int)flipCount

  _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

Stanford CS193p
Fall 2013



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);
```

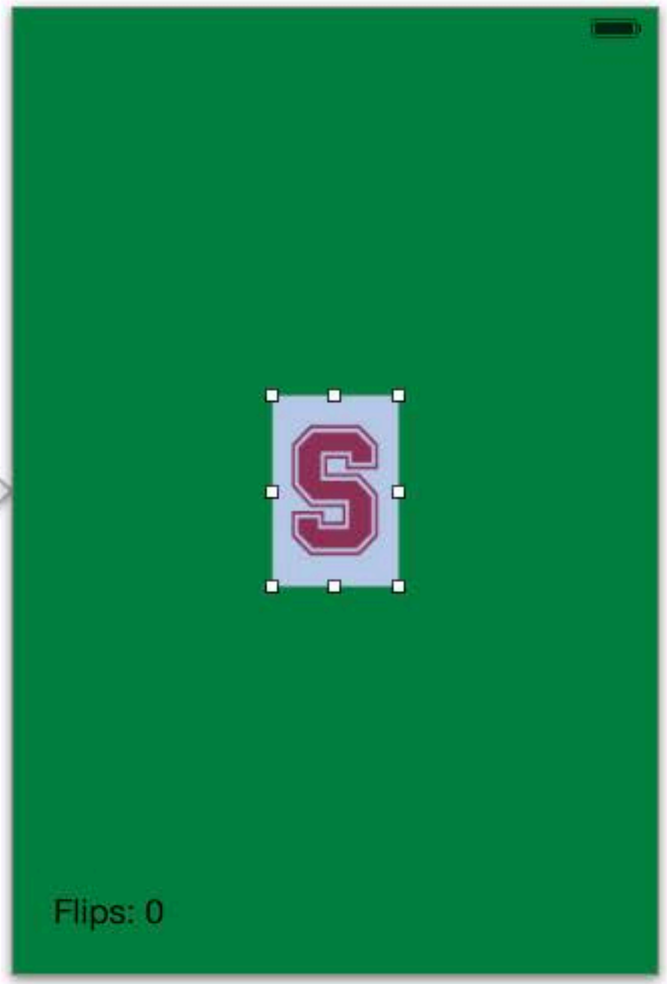
```
2013-00-00 00:00:00.318 Matchismo[11937:a0b] flipCount changed to 2
2013-00-00 00:00:00.865 Matchismo[11937:a0b] flipCount changed to 3
```

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;
}

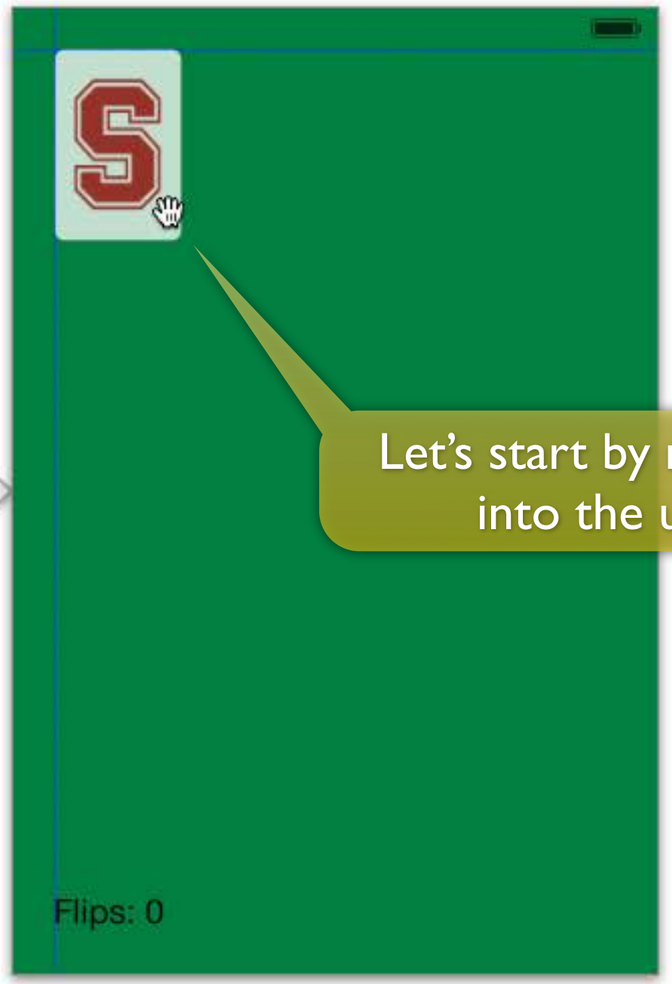
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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!



Let's start by moving our one card into the upper left corner.

```
@implementation CardGameViewController

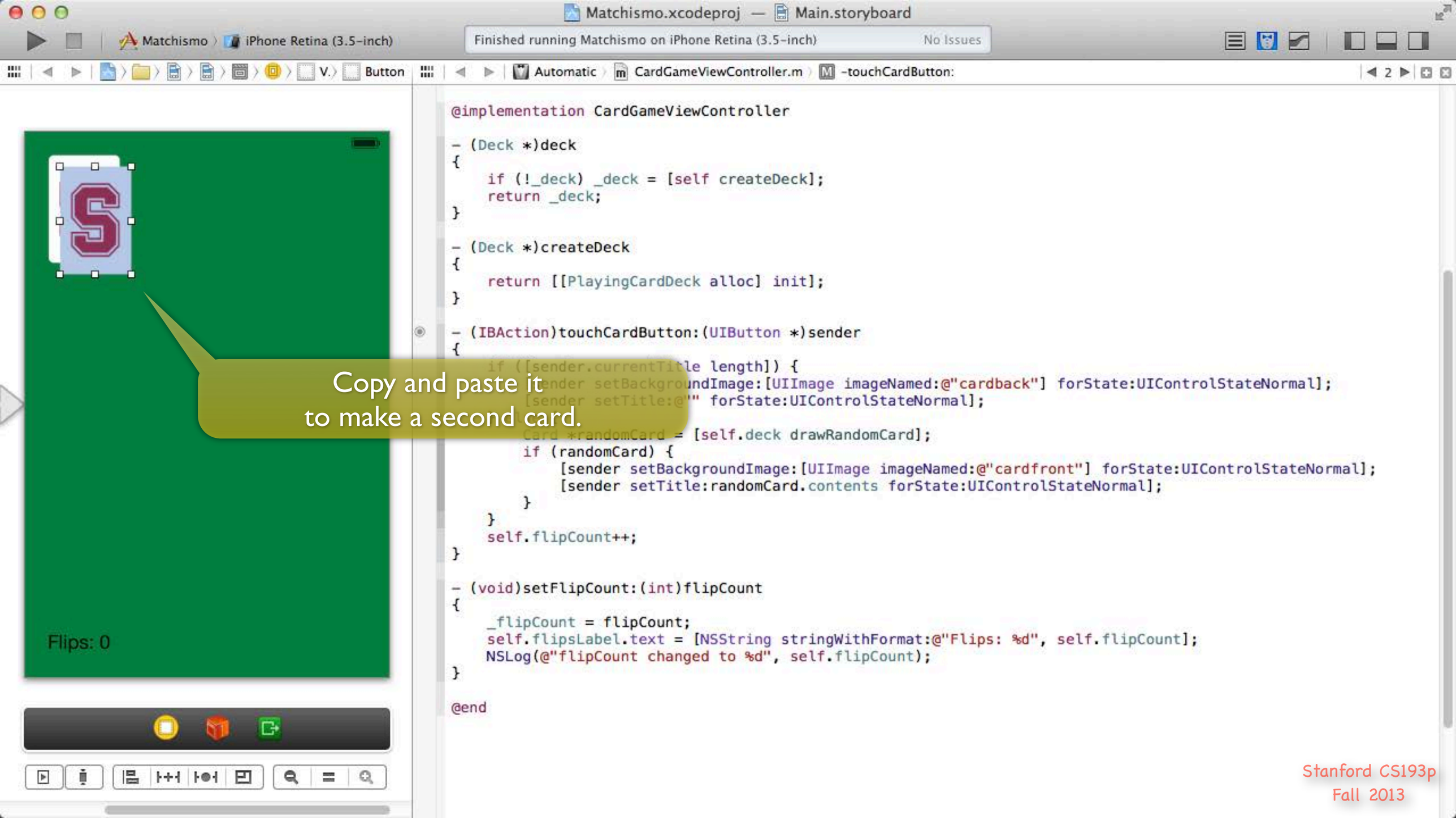
- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

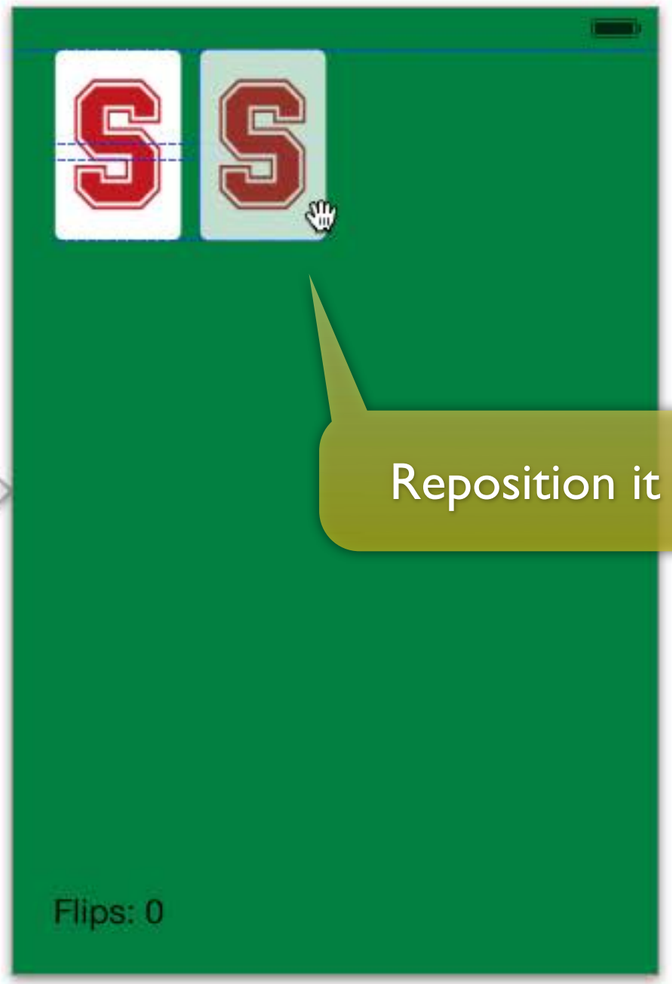
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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
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];
" 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
Flips: 0
Copy and paste it
to make a second card.
Stanford CS193p
Fall 2013



Reposition it using blue guidelines.

```
@implementation CardGameViewController

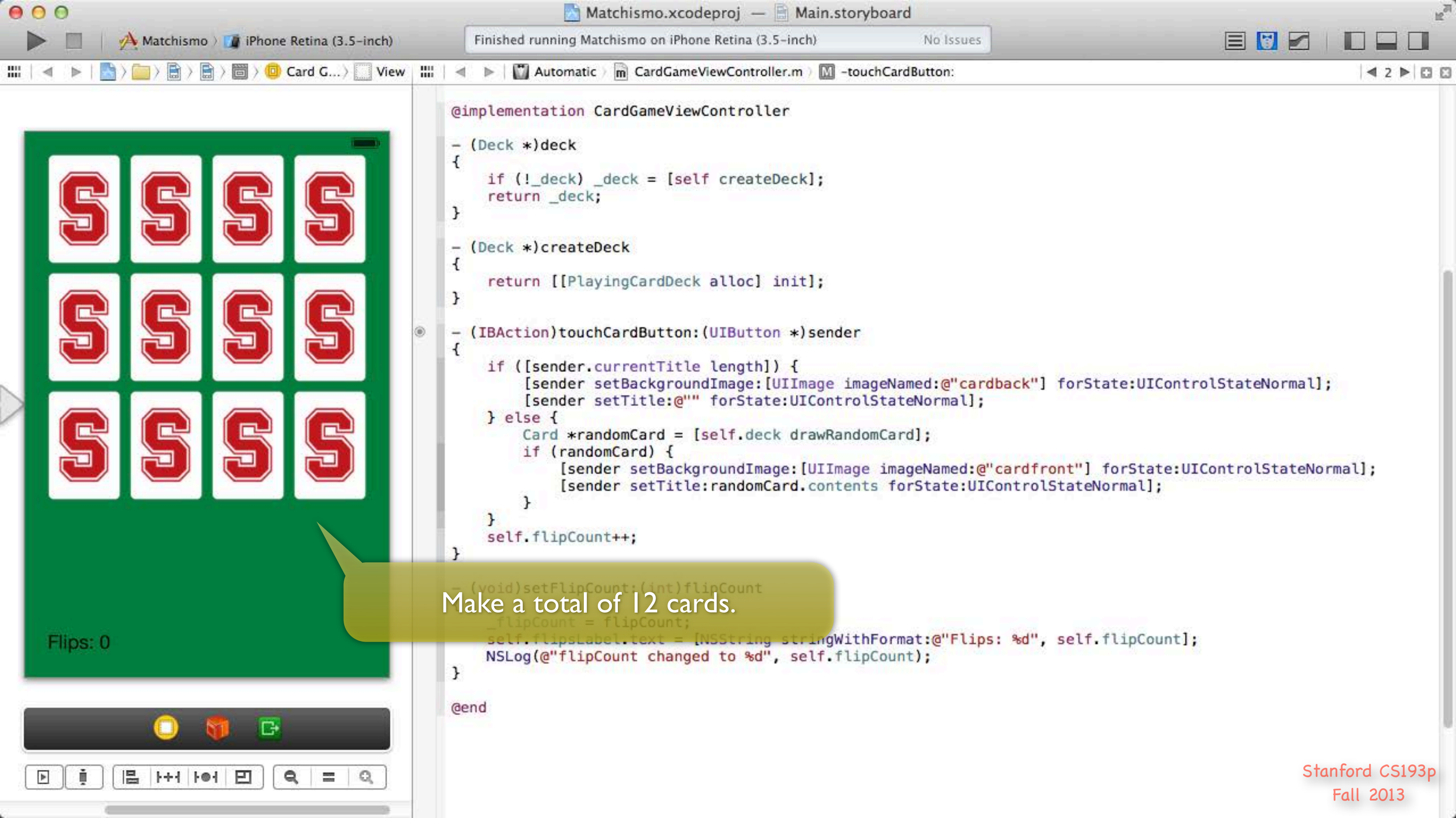
- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (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);
}

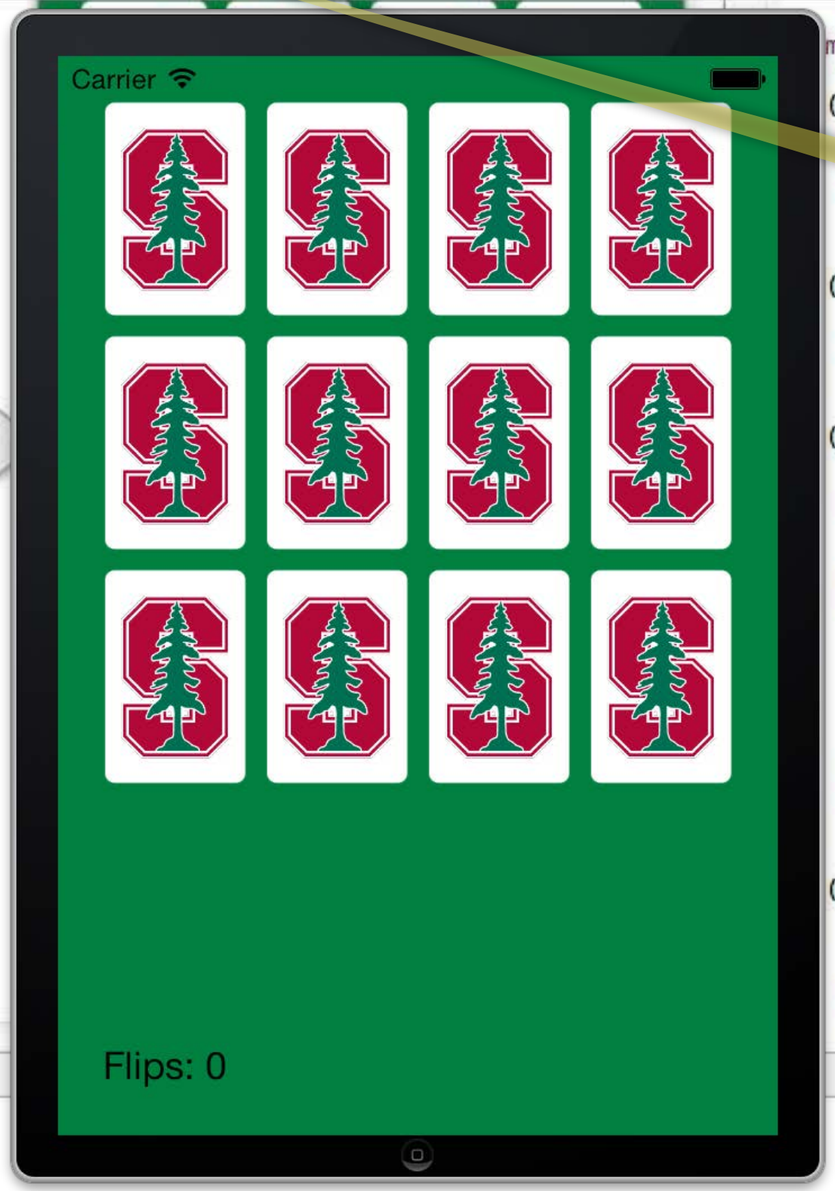
@end
```


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++;
}
NSLog(@"flipCount changed to %d", self.flipCount);
}
@end
Flips: 0
Make a total of 12 cards.
Stanford CS193p
Fall 2013

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:




```
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++;

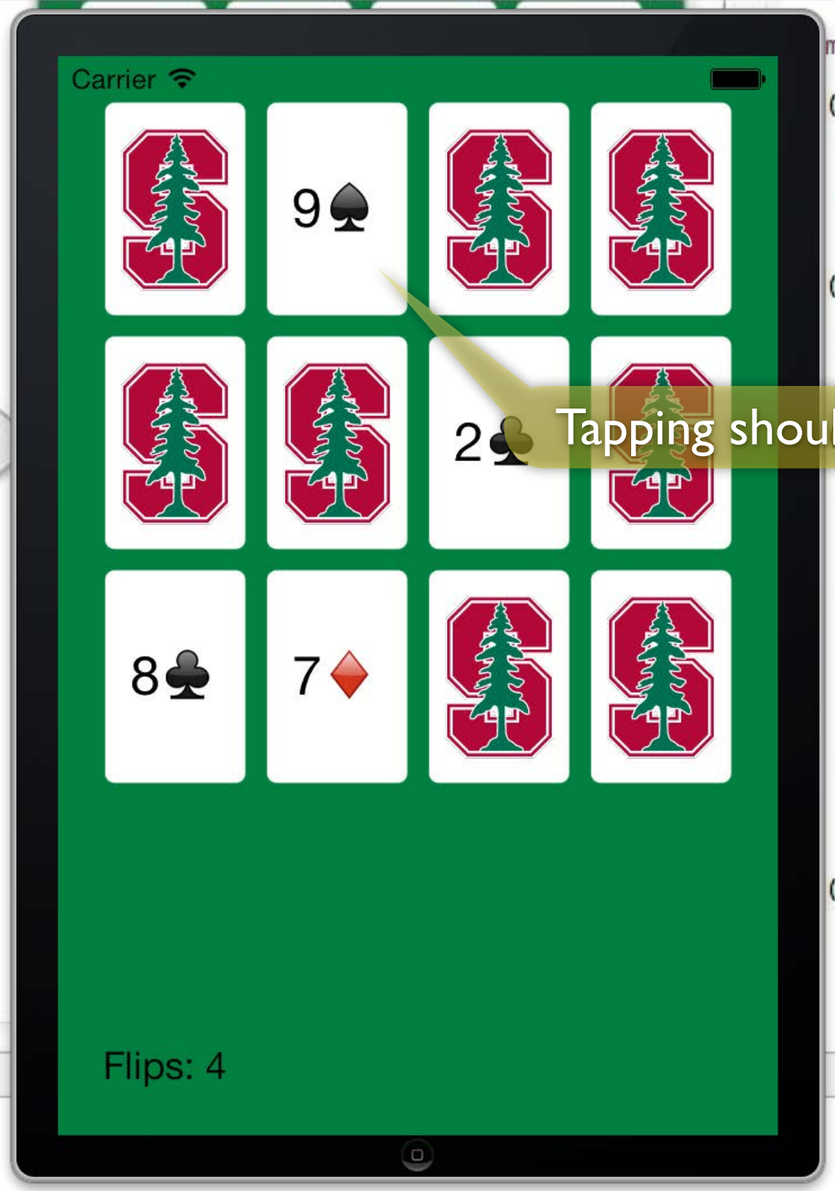
(void)setFlipCount:(int)flipCount

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

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

Auto | All Output

Stanford CS193p
Fall 2013



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

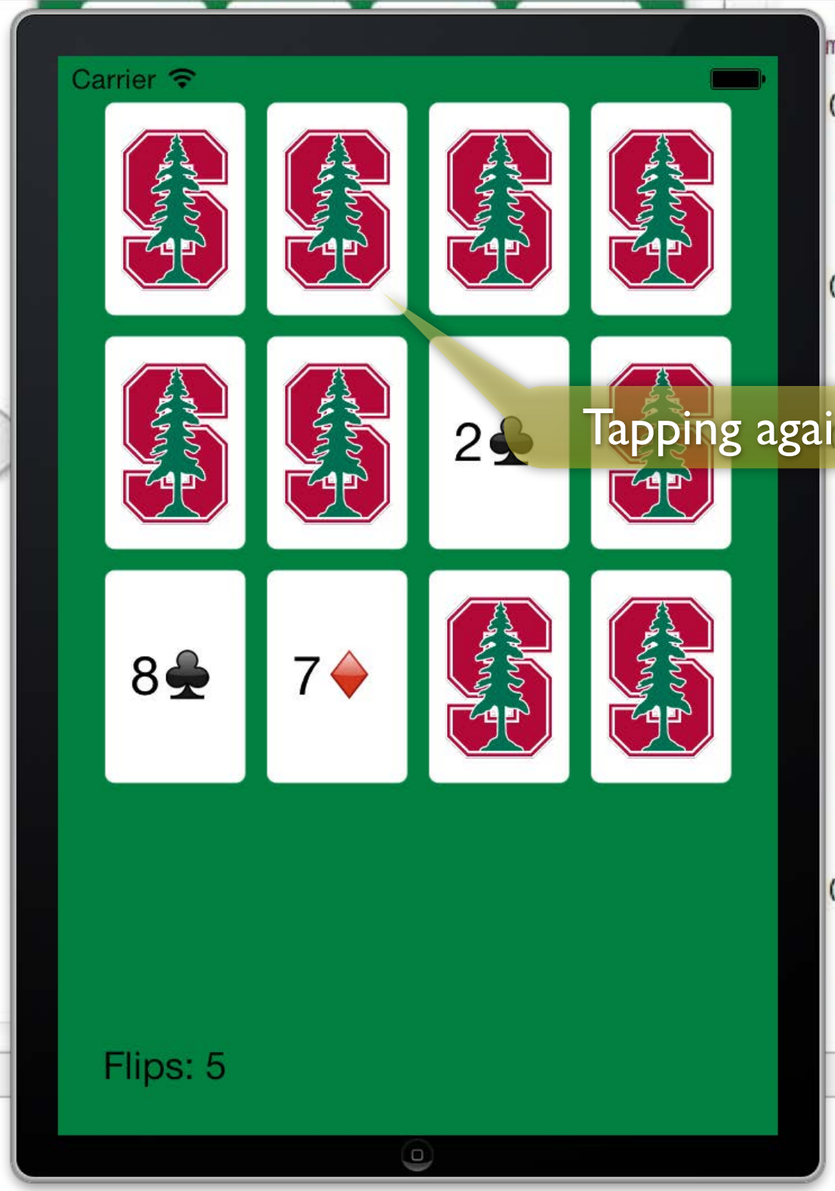
  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage 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 00:00:00.544 Matchismo[13384:a0b] flipCount changed to 2
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
```

Stanford CS193p
Fall 2013



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

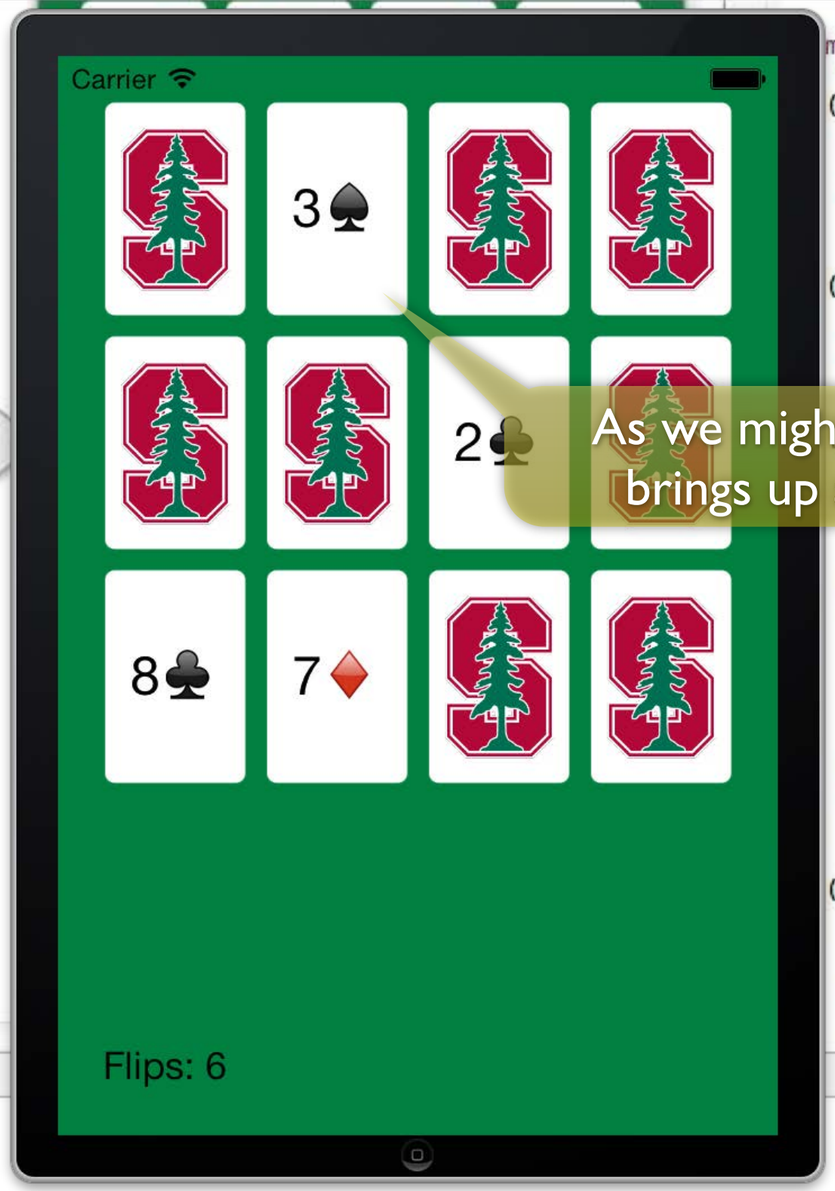
  if ([sender.currentTitle length]) {
      [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
      [sender setTitle:@"" forState:UIControlStateNormal];
  } else {
      Card *randomCard = [self.deck drawRandomCard];
      if (randomCard) {
          [sender setBackgroundImage:[UIImage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

Stanford CS193p
Fall 2013



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
                                ) {
      [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
2013-00-00 00:00:03.366 Matchismo[13384:a0b] flipCount changed to 6
```

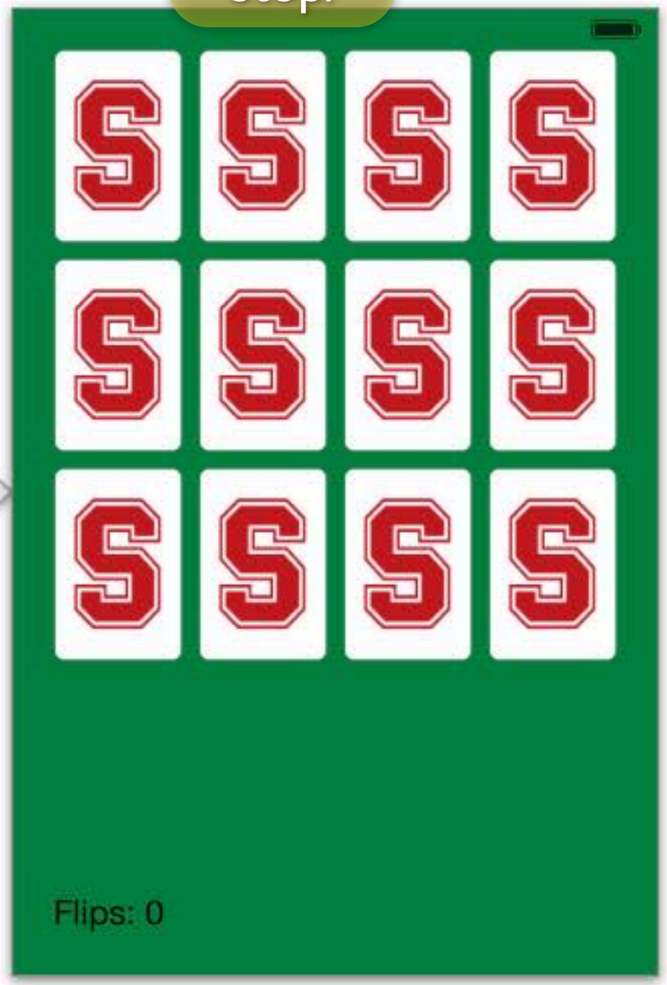
Stanford CS193p
Fall 2013

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:

Stop.



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;
}

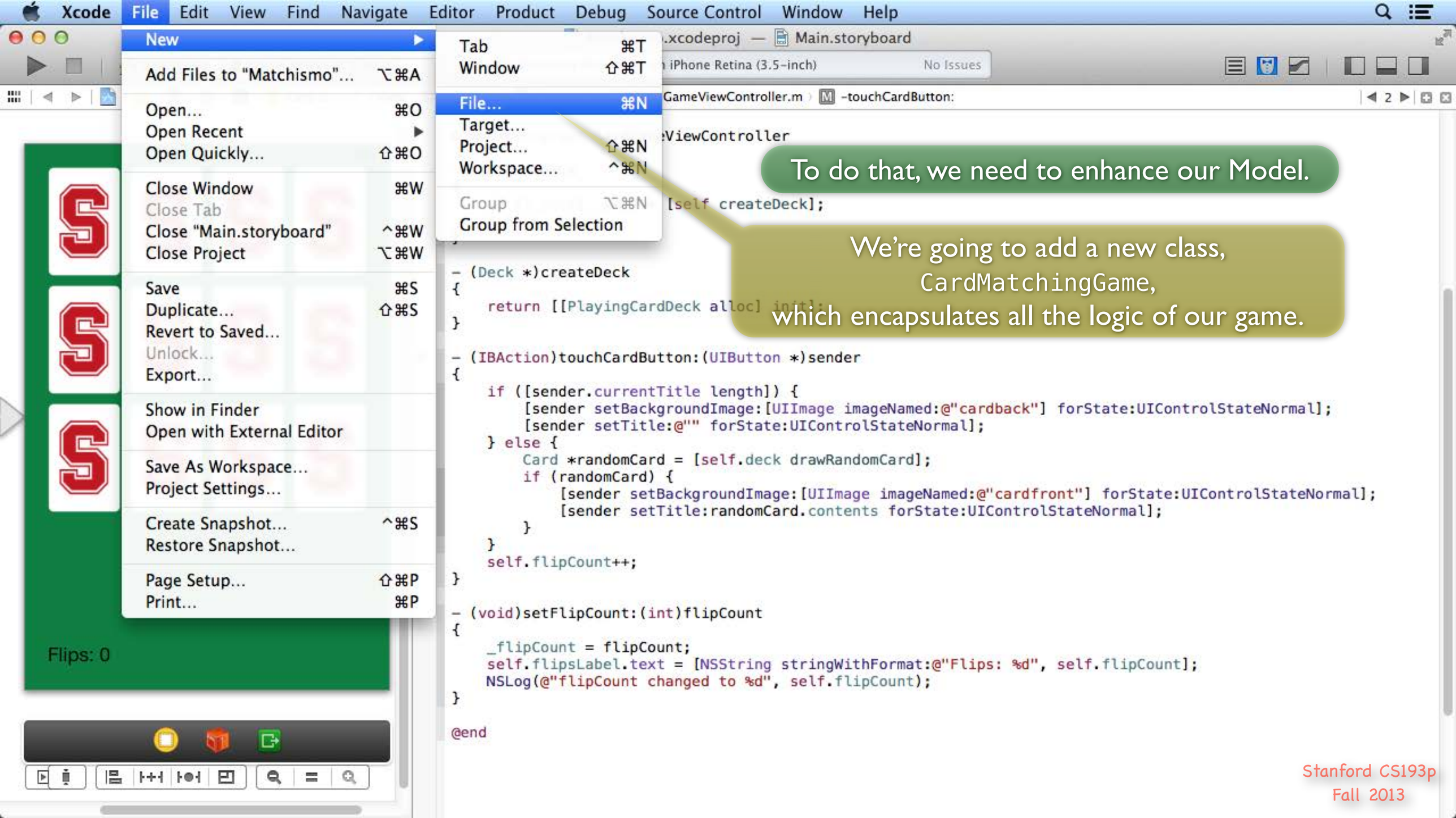
- (Deck *)cre
{
    return [[PlayingC
}

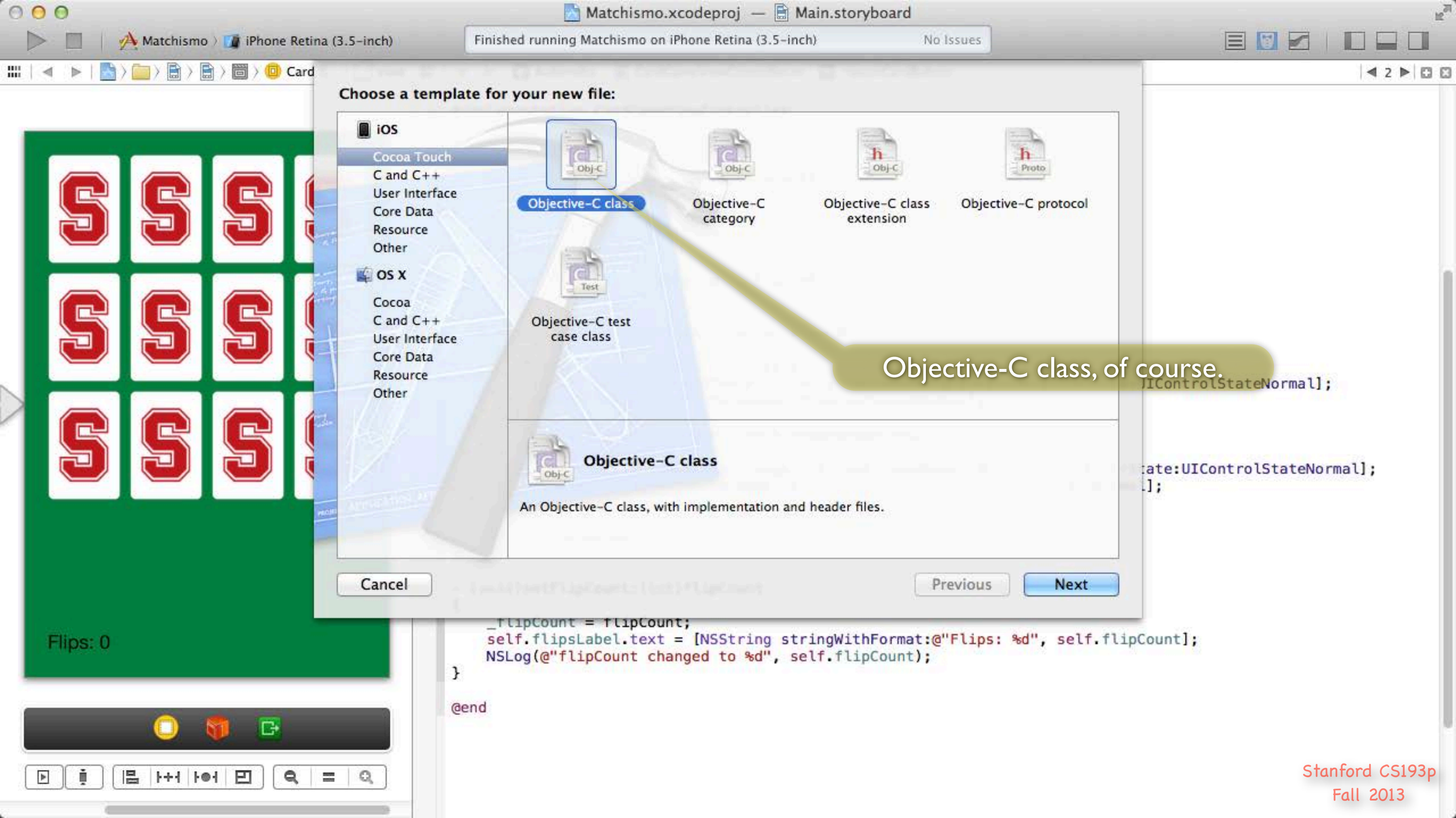
- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck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

Stanford CS193p
Fall 2013


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.
- (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
Stanford CS193p
Fall 2013



Stanford CS193p
Fall 2013

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

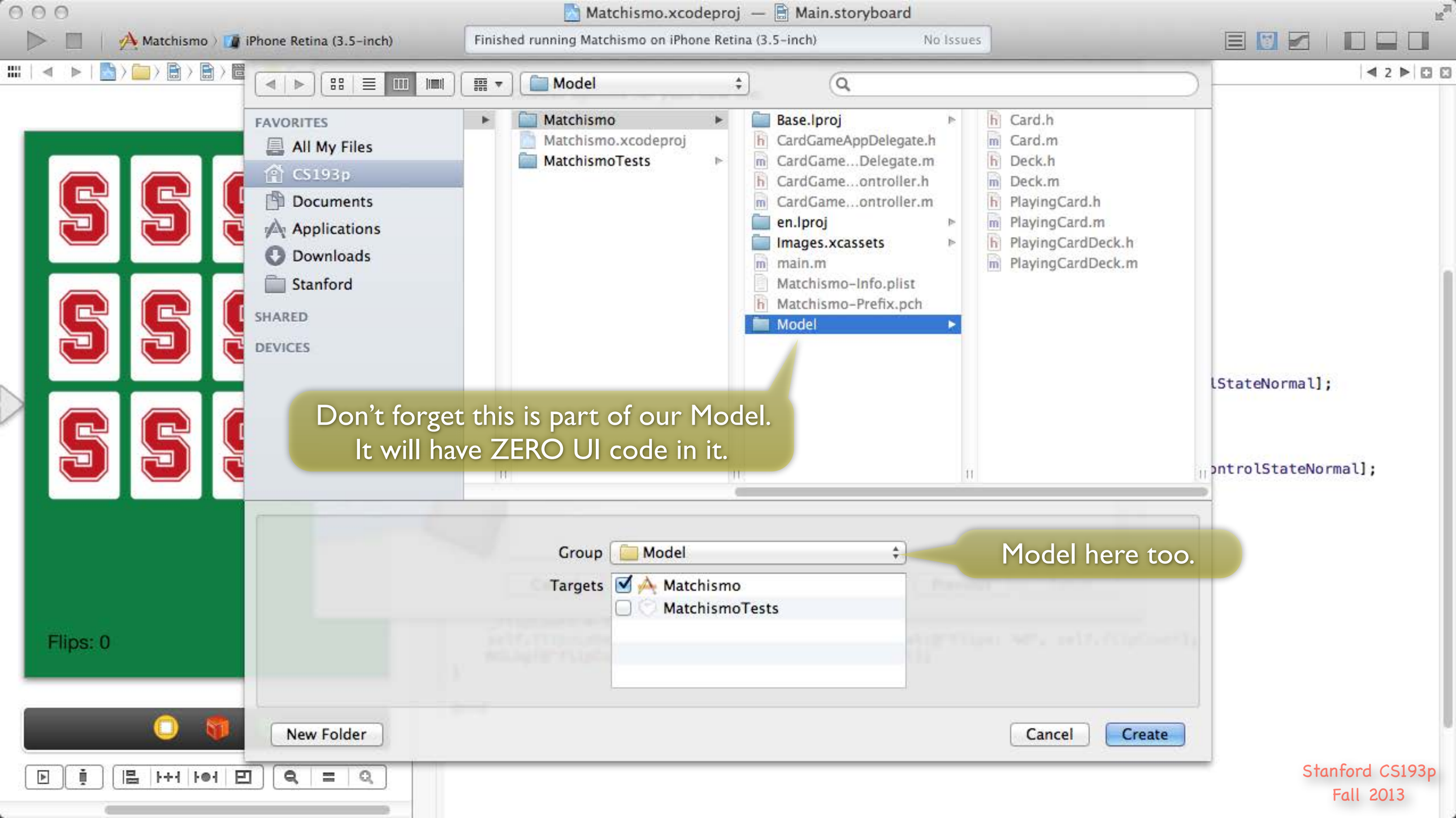
Subclass of NSObject.

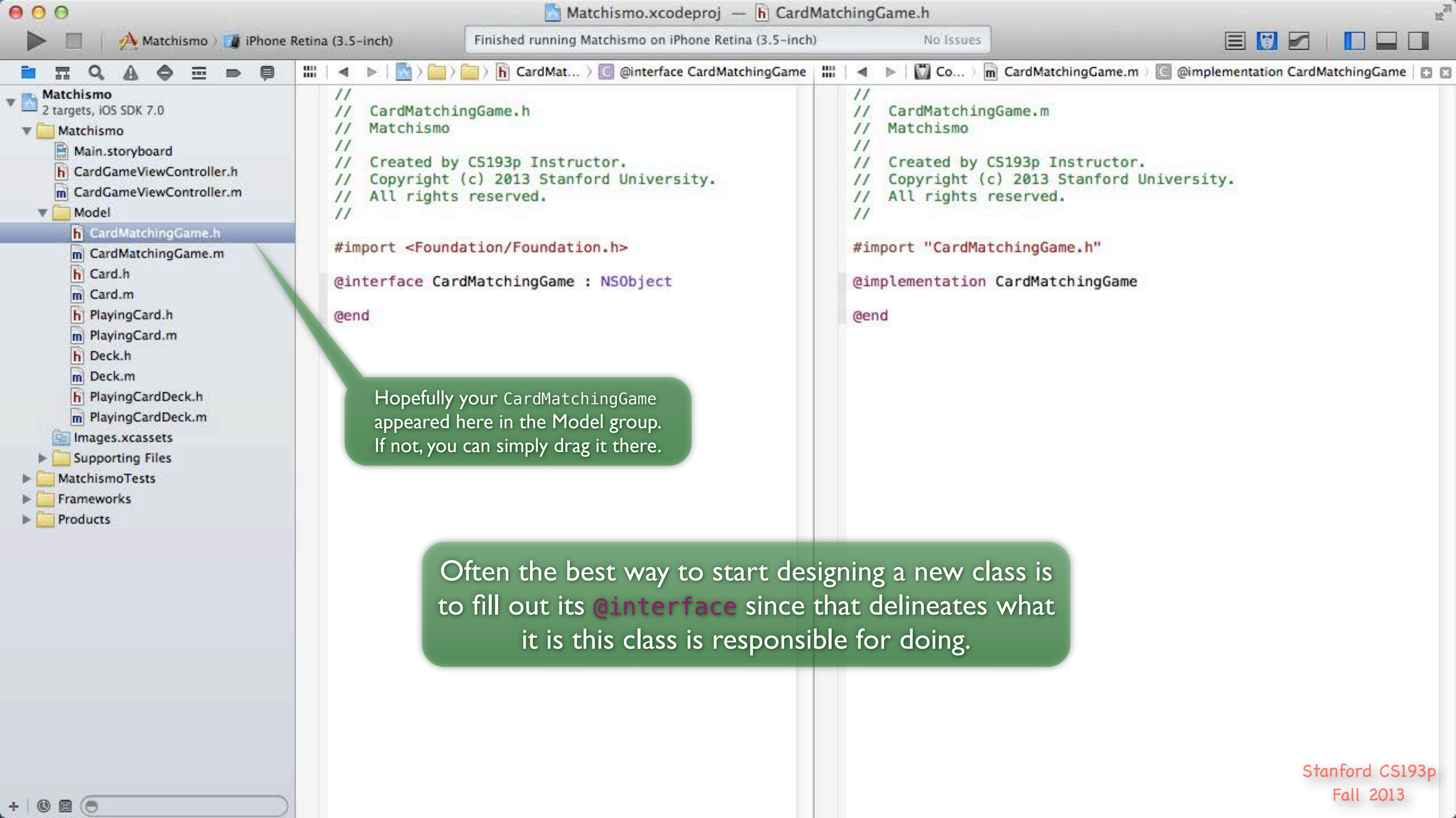
Cancel | Previous | Next

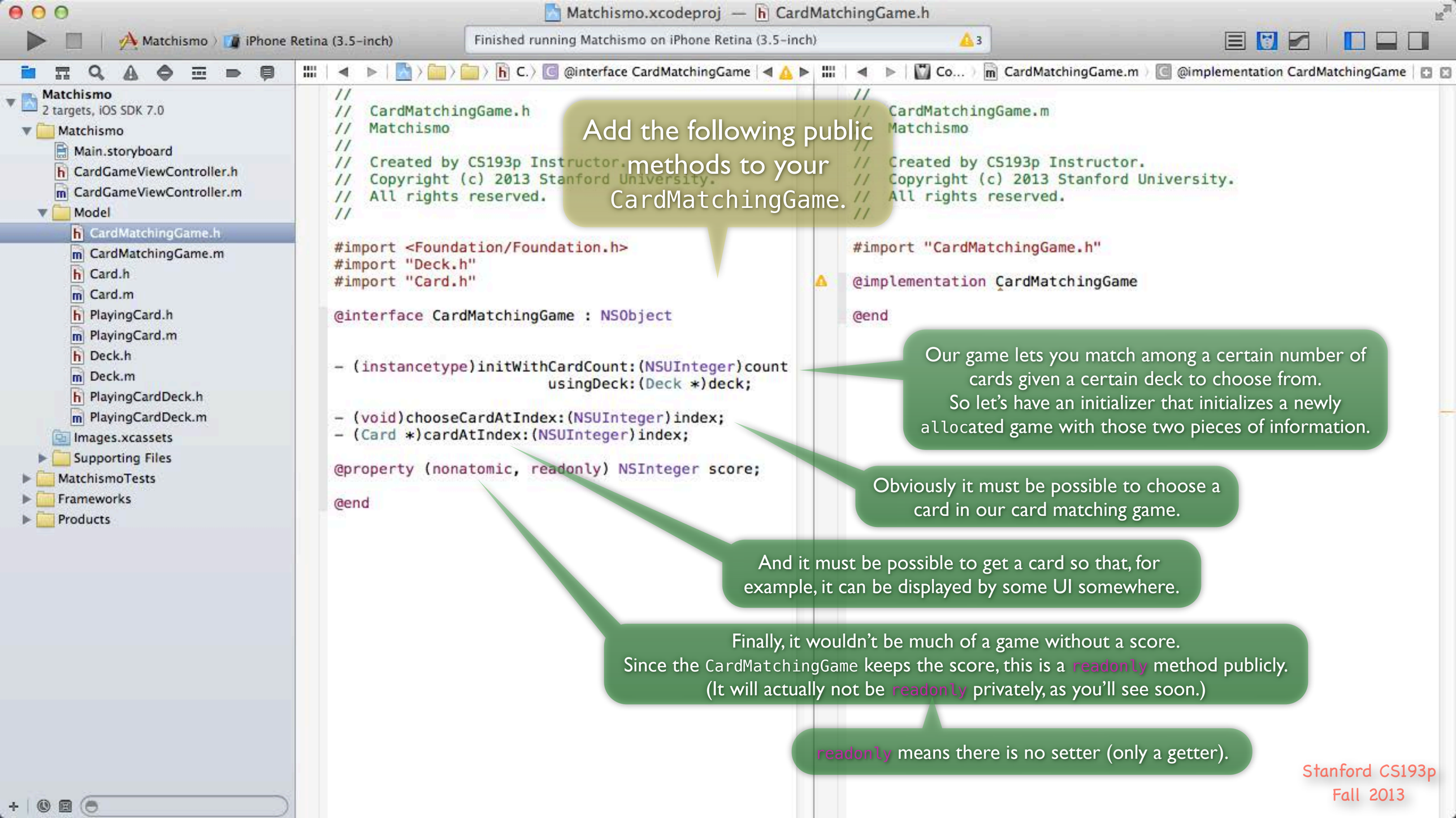
Flips: 0

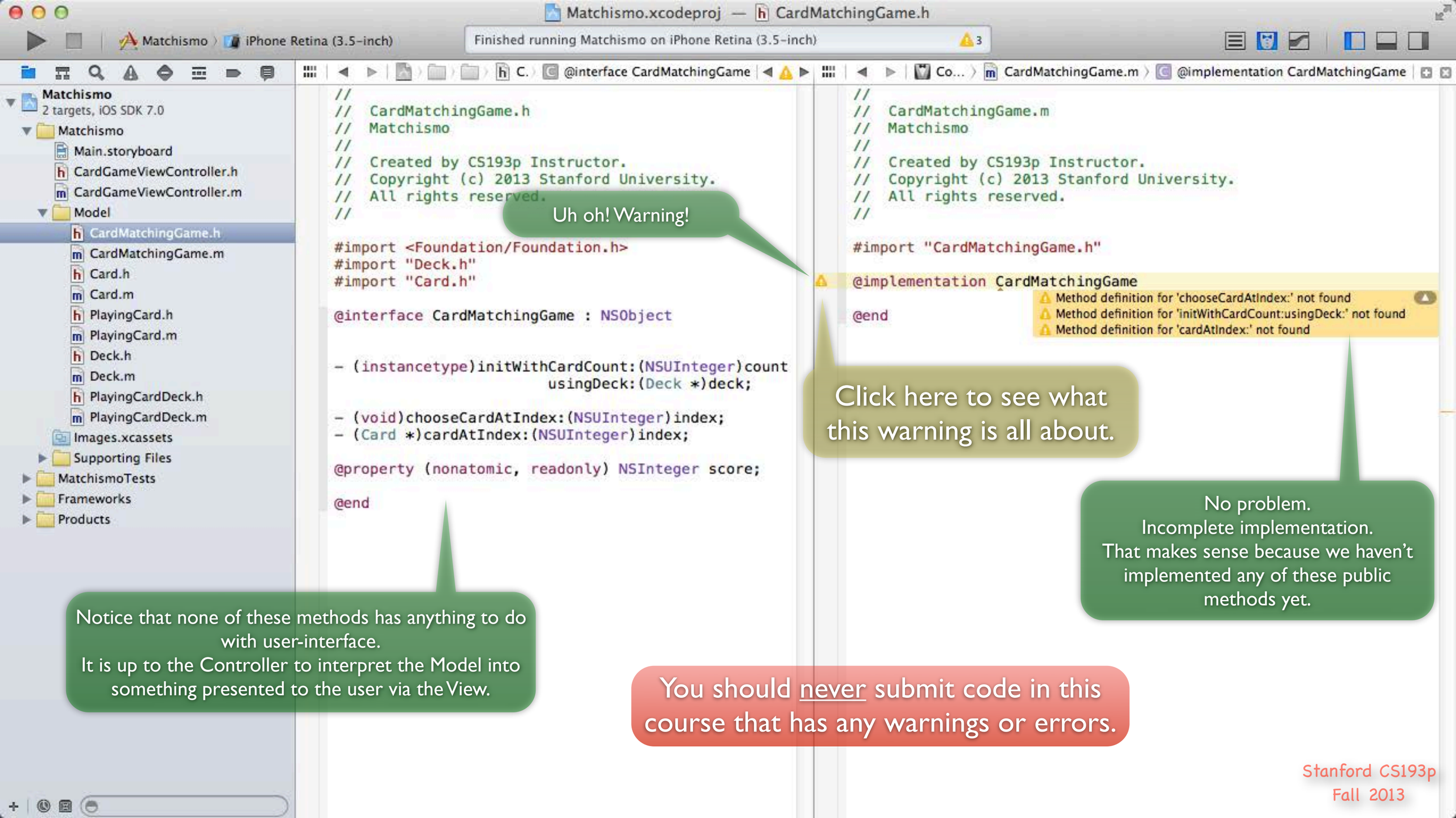
```
    _flipCount = flipCount;
    self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
    NSLog(@"flipCount changed to %d", self.flipCount);
}

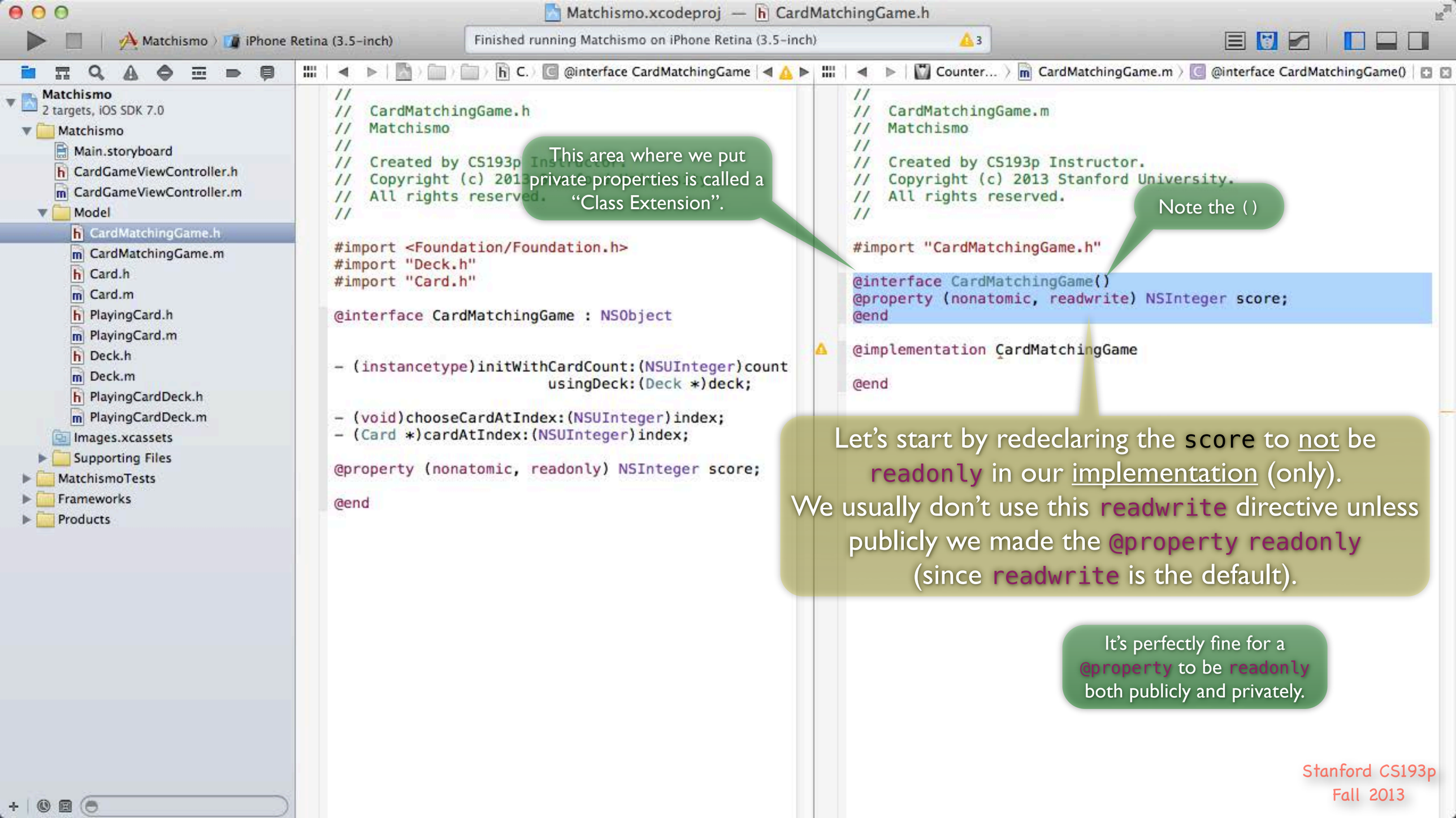
@end
```


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
Targets
Matchismo
MatchismoTests
Model here too.
New Folder
Cancel
Create
Flips: 0
lStateNormal];
ontrolStateNormal];
Stanford CS193p
Fall 2013


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.
Stanford CS193p
Fall 2013


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
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
// 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
// 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).
Stanford CS193p
Fall 2013


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.
Stanford CS193p
Fall 2013


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.
Stanford CS193p
Fall 2013

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

CardMatchingGame.h — 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
```

CardMatchingGame.m — 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`.

Stanford CS193p
Fall 2013

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

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];
            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) | 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

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

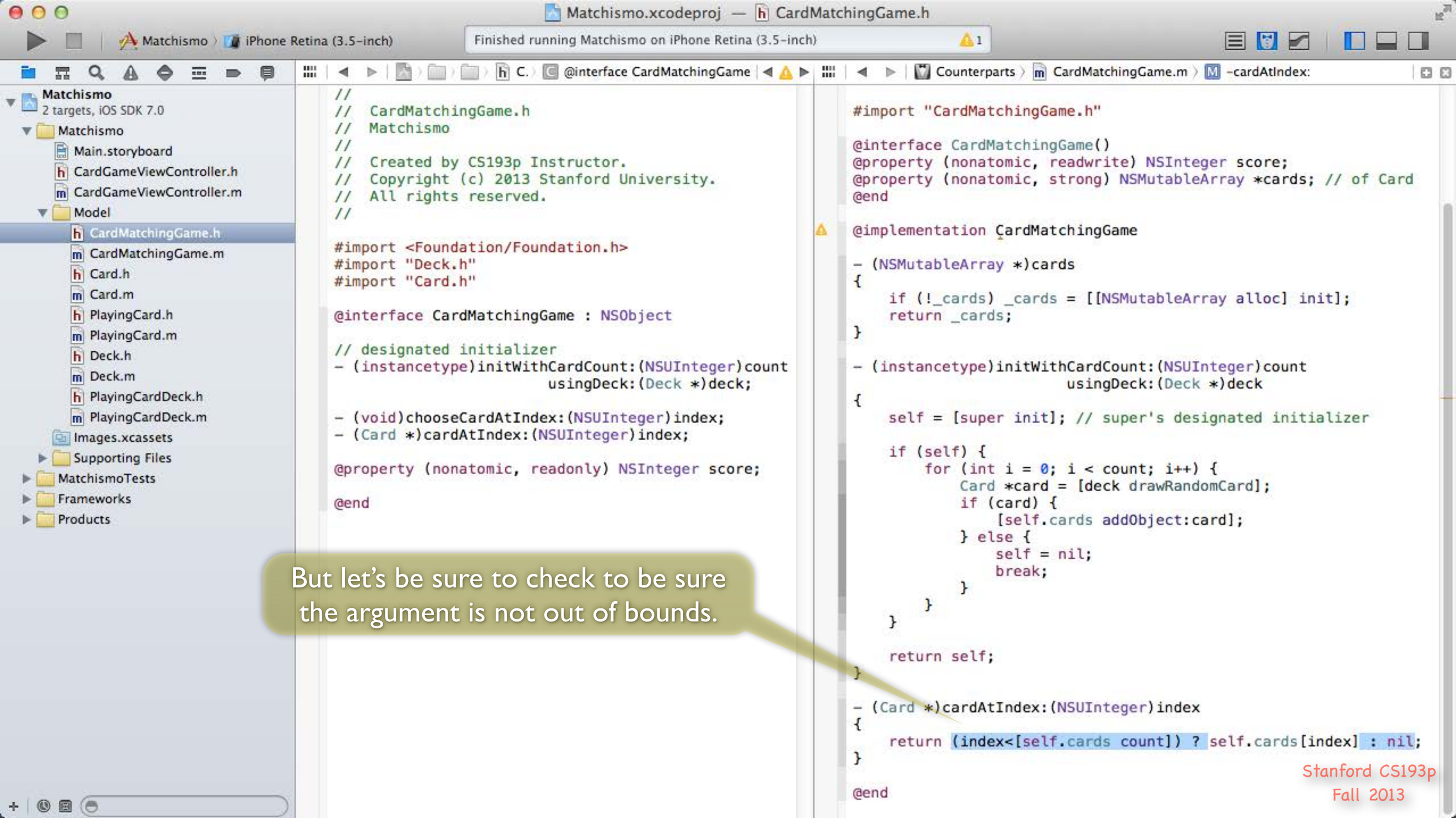
    if (self) {
        for (int i = 0; i < count; i++) {
            Card *card = [deck drawRandomCard];
            if (card) {
                [self.cards addObject:card];
            } else {
                self = nil;
                break;
            }
        }
    }

    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
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
But let's be sure to check to be sure the argument is not out of bounds.
Stanford CS193p
Fall 2013

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) | 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

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;
}



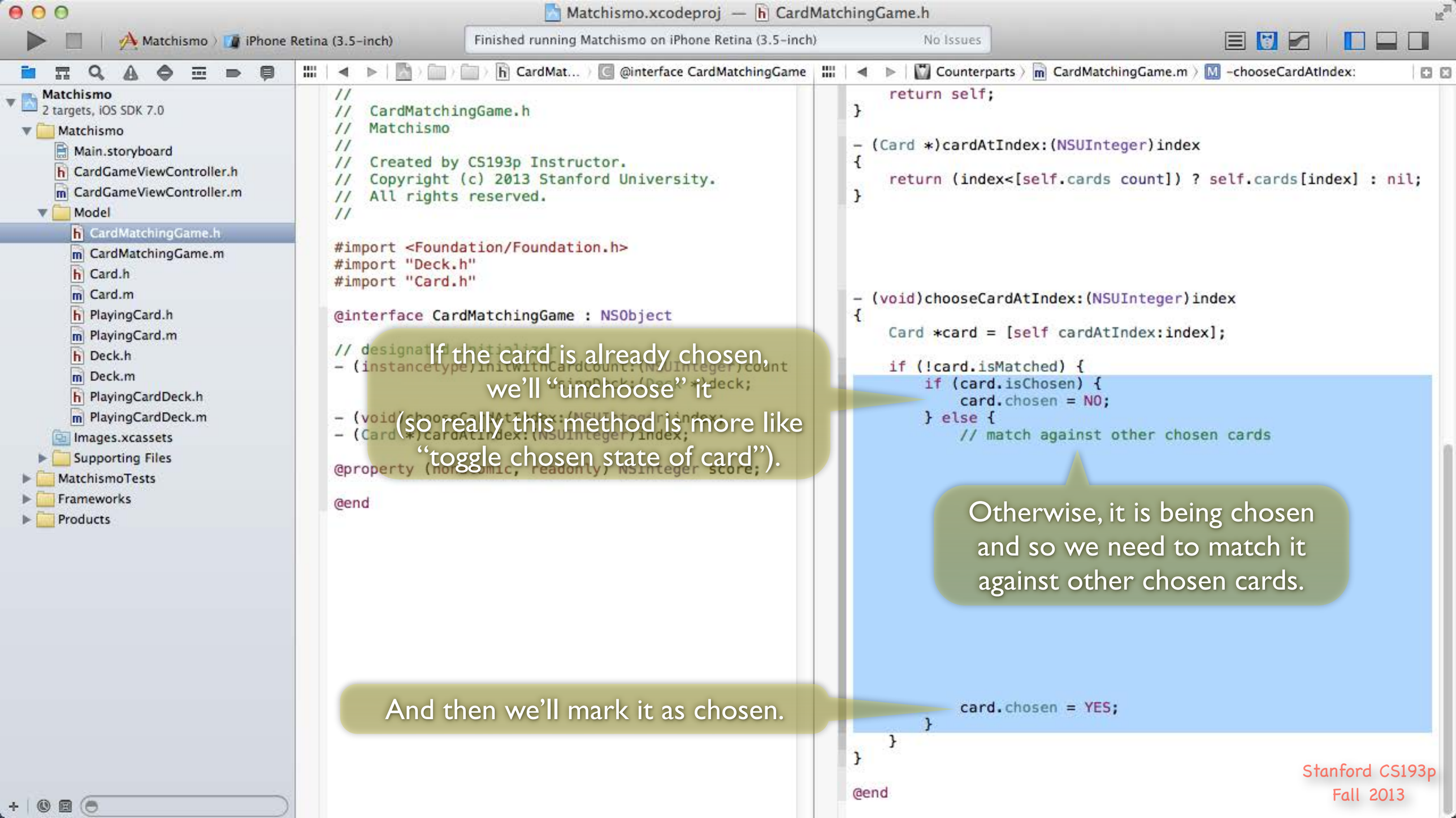
- (void)chooseCardAtIndex:(NSUInteger)index
{
    Card *card = [self cardAtIndex:index];

    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
- (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.
Stanford CS193p
Fall 2013

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];

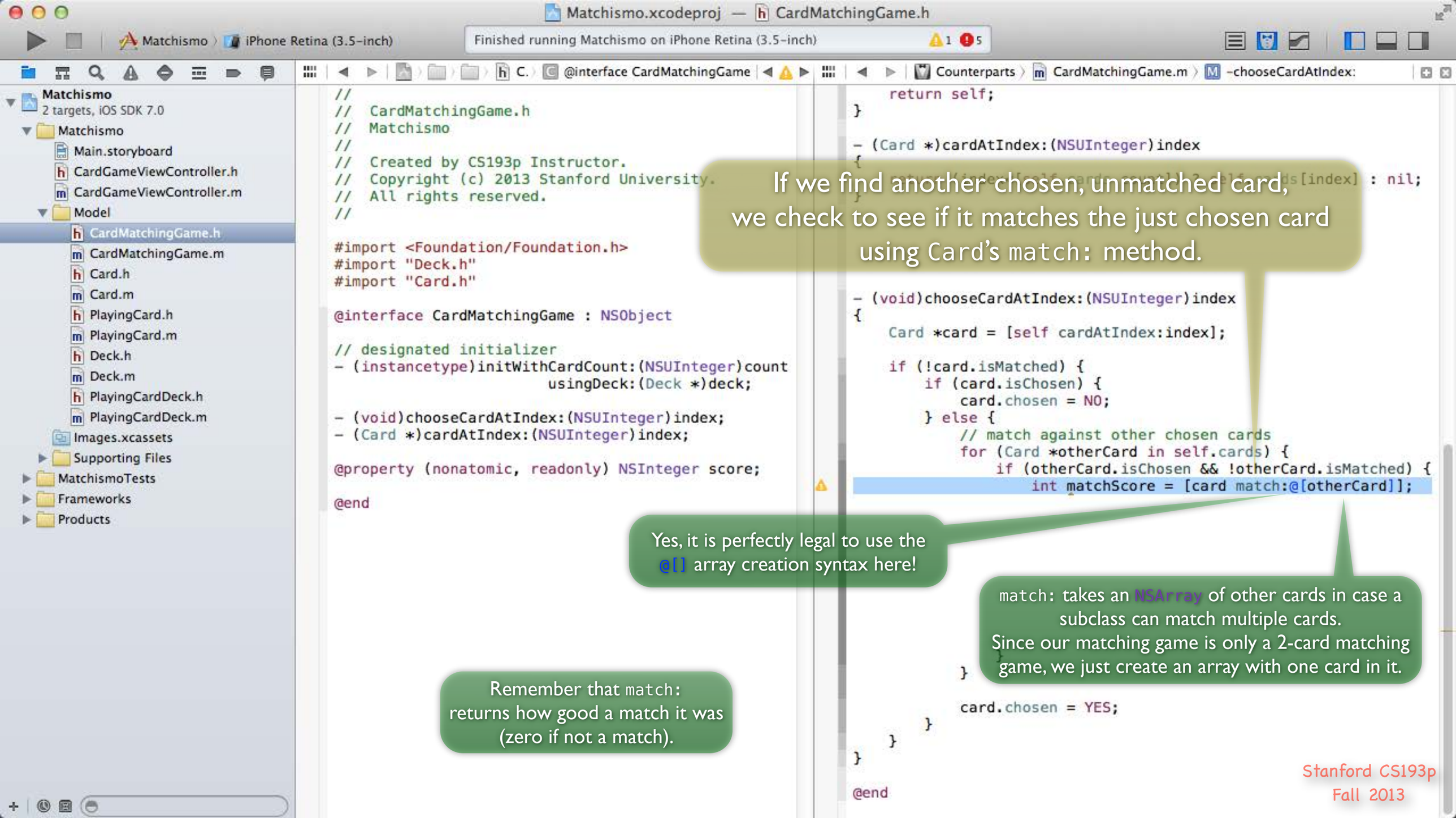
    if (!card.isMatched) {
        if (card.isChosen) {
            card.chosen = NO;
        } else {
            // match against other chosen cards
            for (Card *otherCard in self.cards) {
                if (otherCard.isChosen && !otherCard.isMatched) {

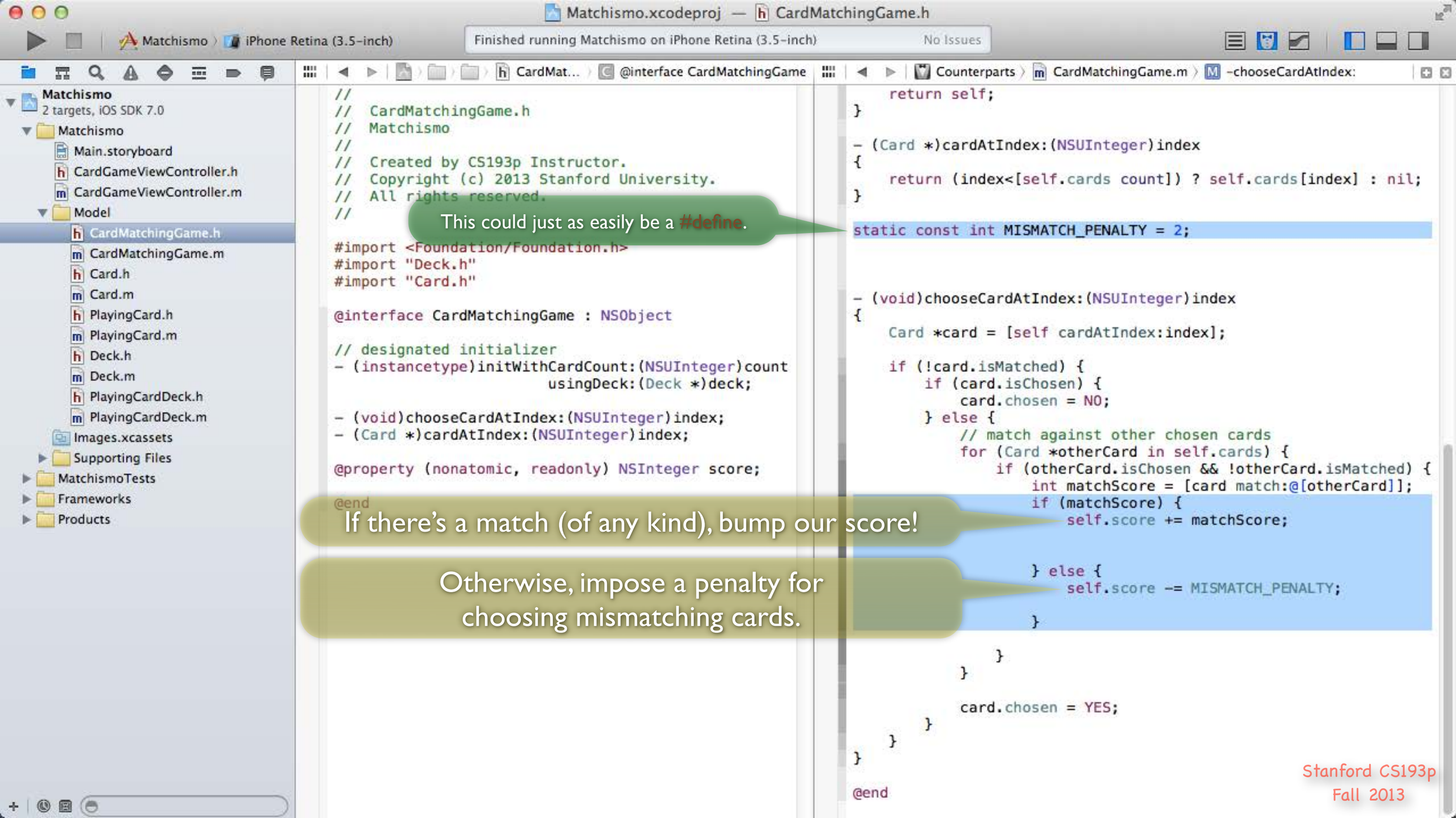
                }
            }

            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).
Stanford CS193p
Fall 2013


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
Stanford CS193p
Fall 2013

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;

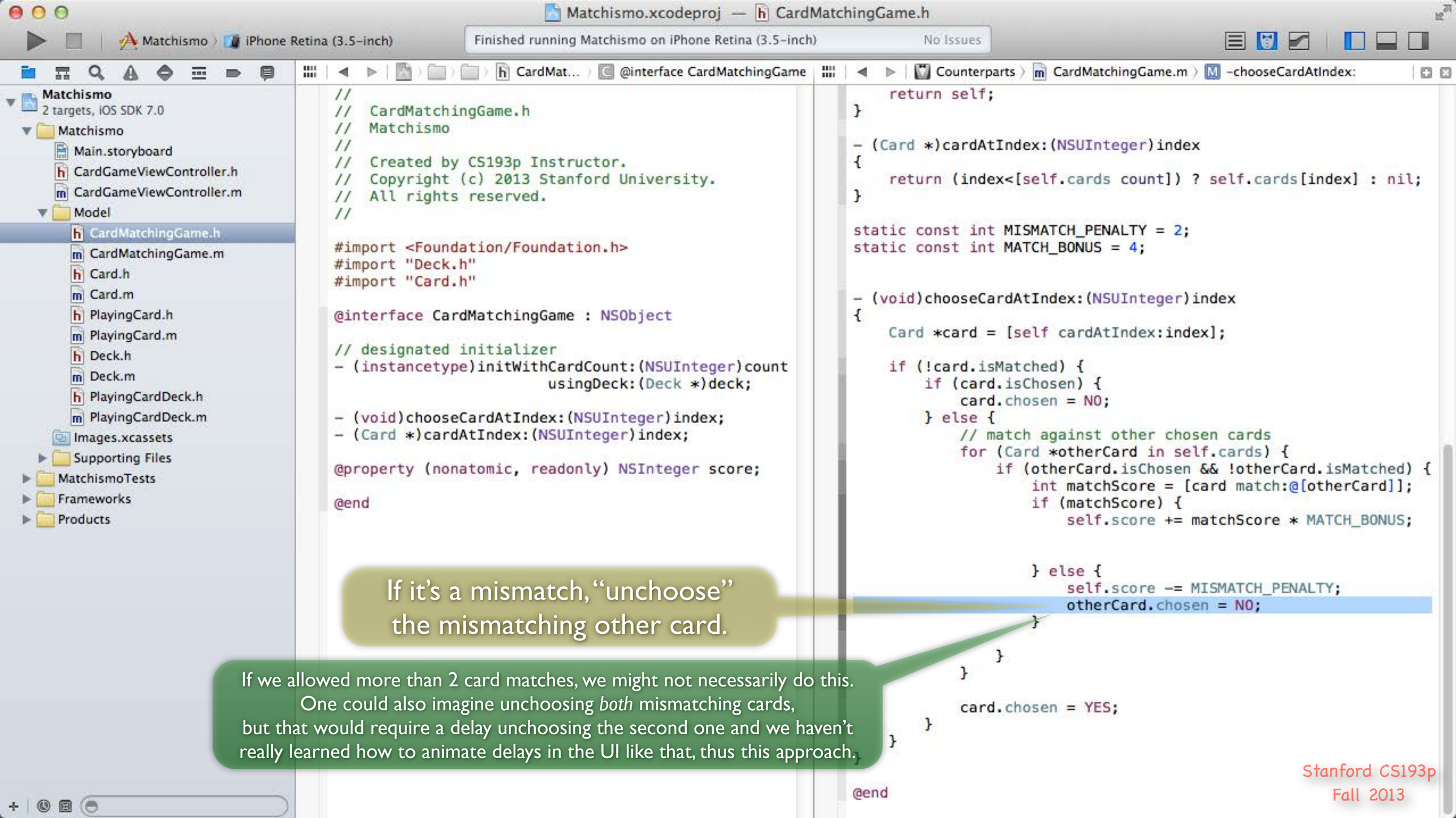
                    }

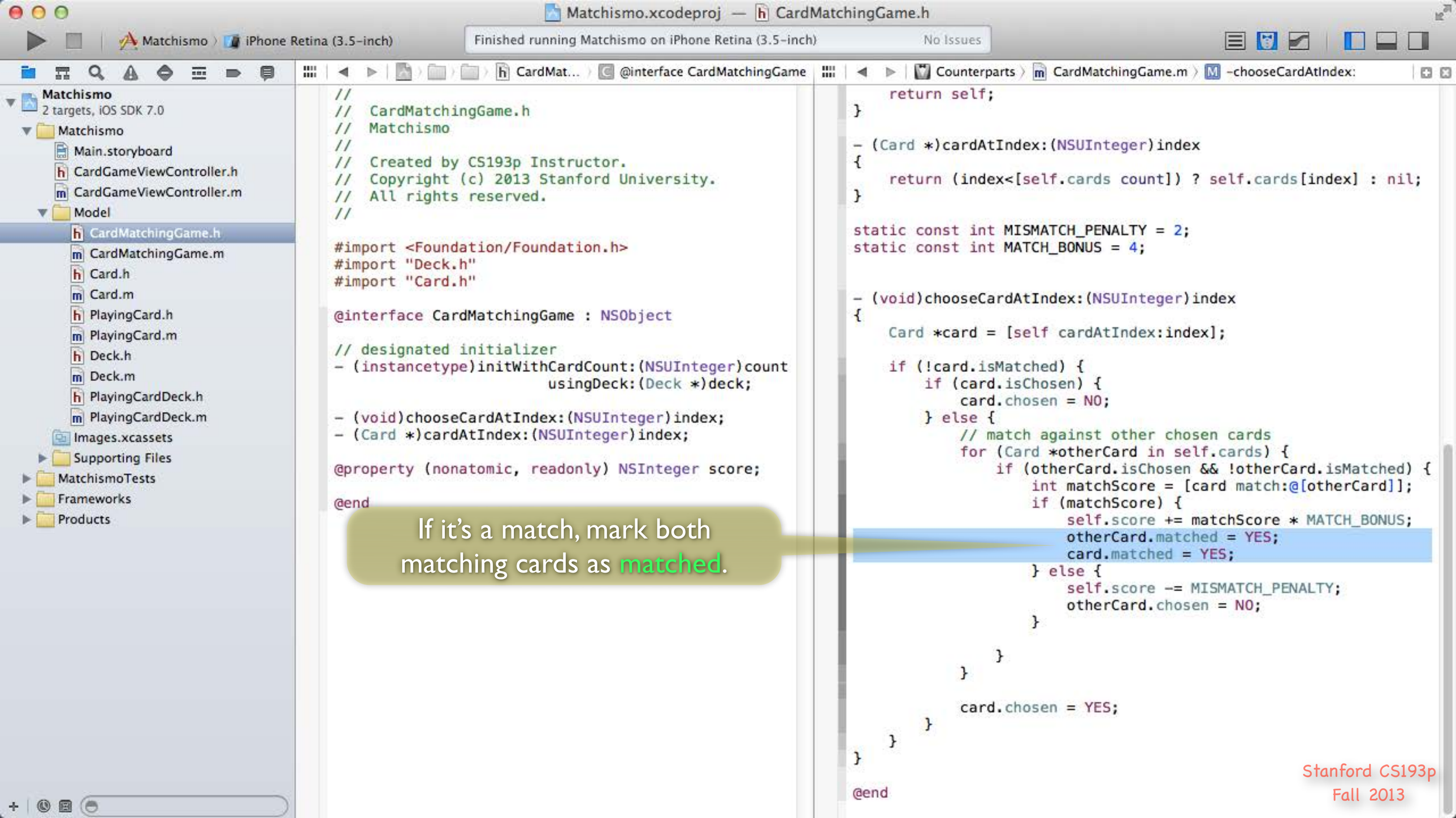
                }
            }

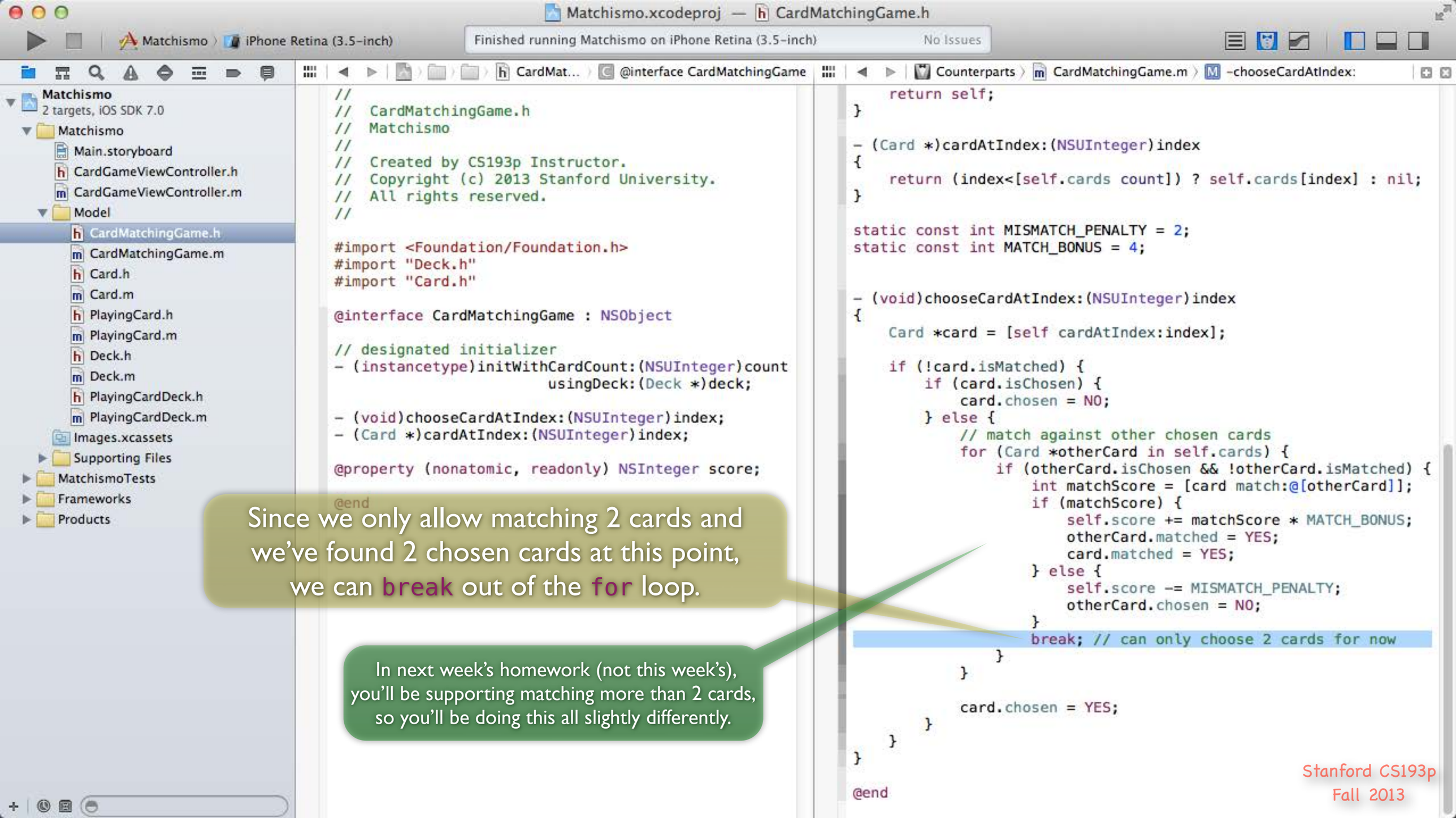
            card.chosen = YES;
        }
    }
}

@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.
Stanford CS193p
Fall 2013


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;
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.
Stanford CS193p
Fall 2013


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
Stanford CS193p
Fall 2013

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;
}

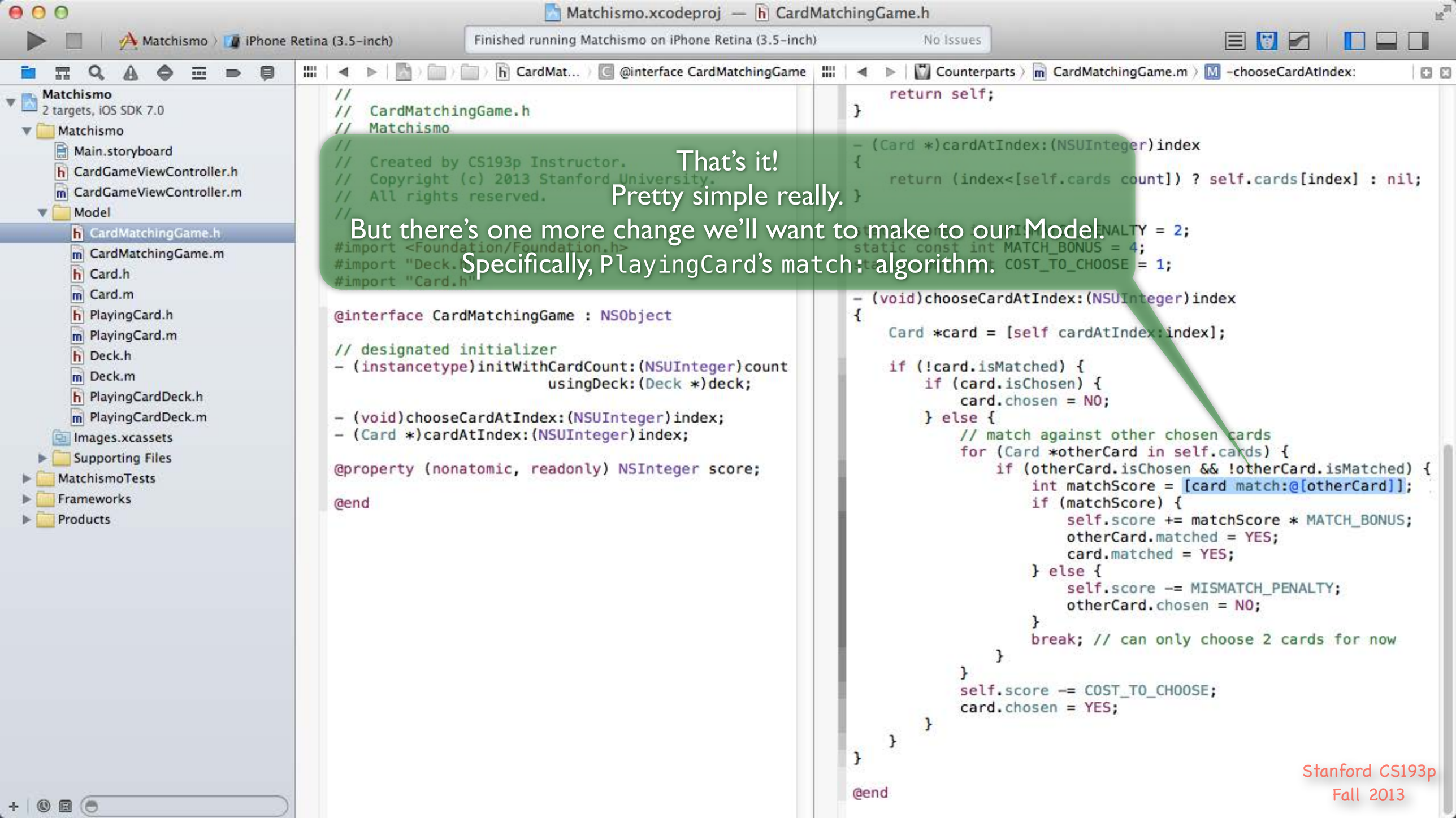
static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
static const int COST_TO_CHOOSE = 1;

- (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.




```
//
//  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;
}

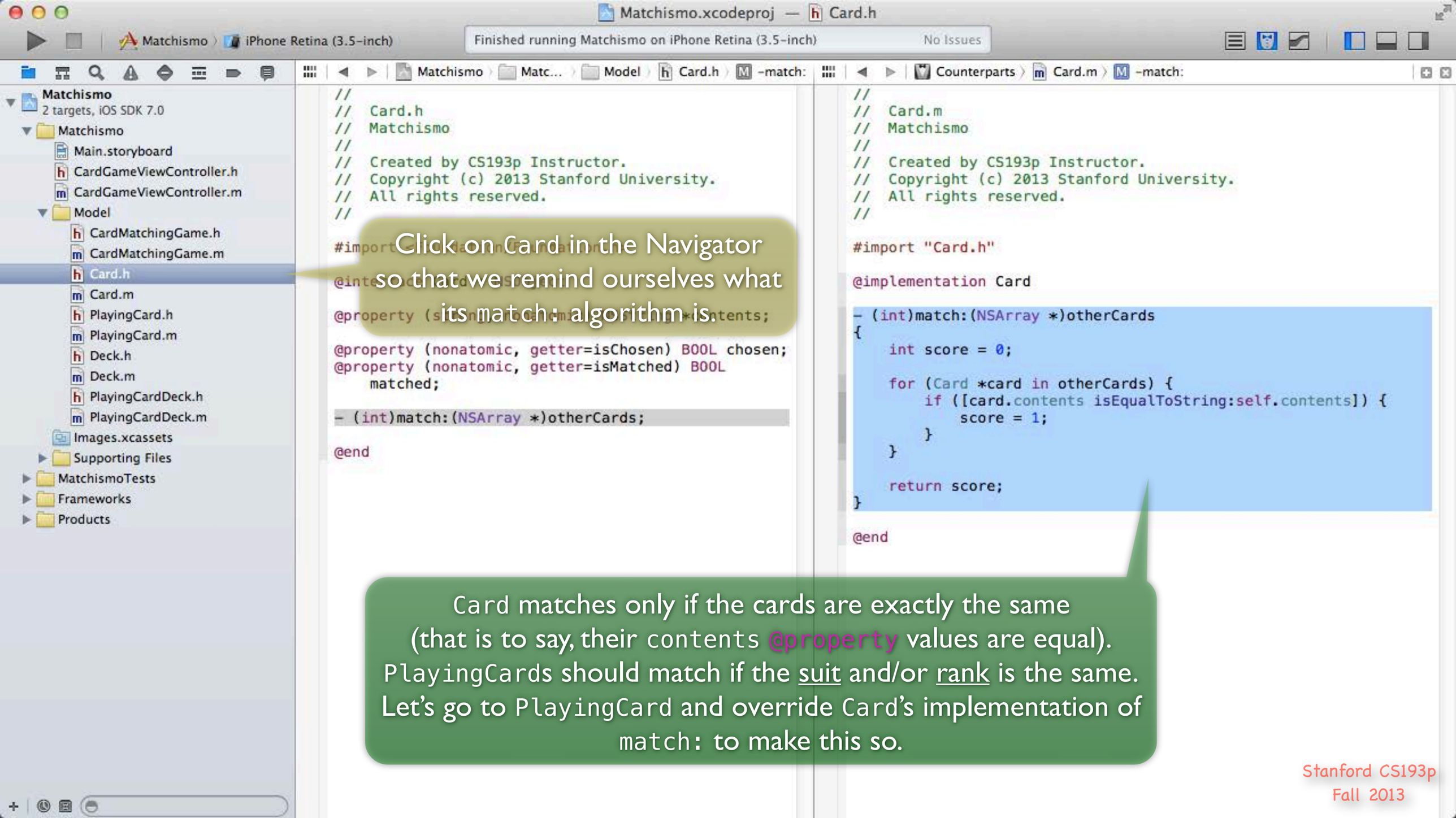
- (Card *)cardAtIndex:(NSUInteger)index
{
    return (index<[self.cards count]) ? self.cards[index] : nil;
}

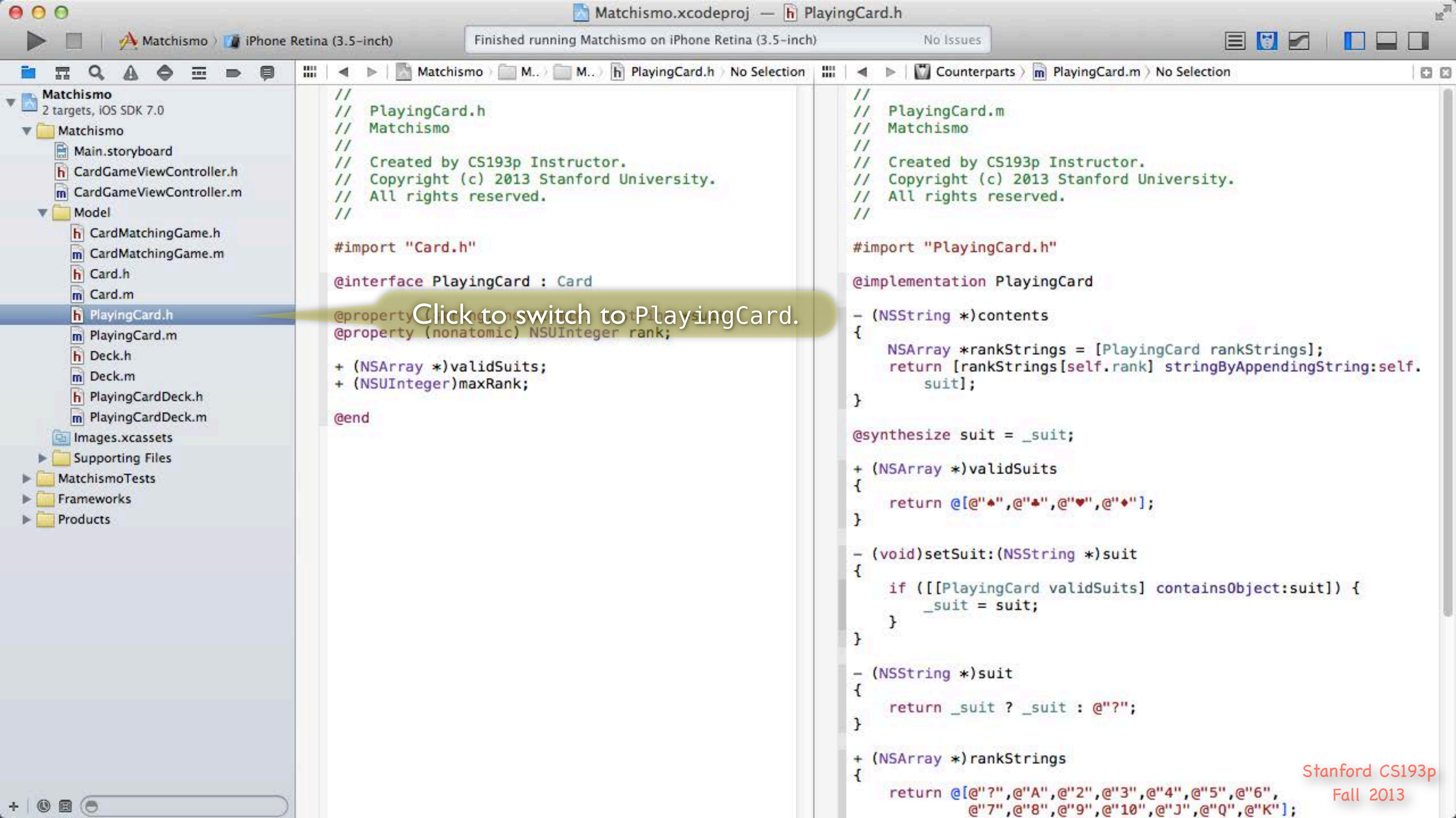
static const int MISMATCH_PENALTY = 2;
static const int MATCH_BONUS = 4;
static const int COST_TO_CHOOSE = 1;

- (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
```


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.
//
@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.
Stanford CS193p
Fall 2013


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"];
Stanford CS193p
Fall 2013

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.

Add an implementation for match:.

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;

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

Counterparts › PlayingCard.m › -match:

- 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

PlayingCard.h

```
//
//  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
```

PlayingCard.m

```
//
//  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) {



    }

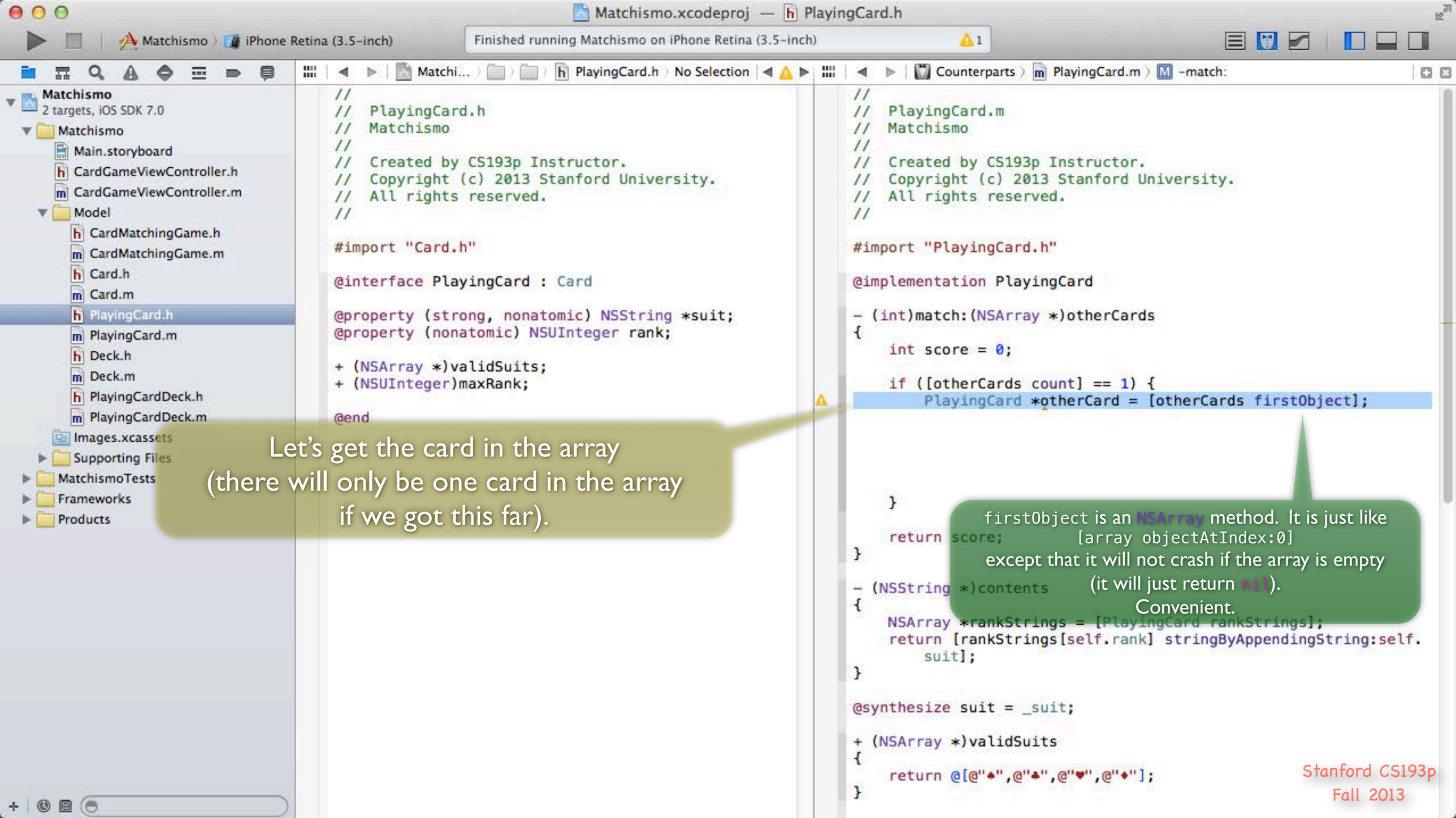
    return score;
}

- (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)
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
Counterparts › PlayingCard.m › -match:
// 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
// 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.
Stanford CS193p Fall 2013

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;

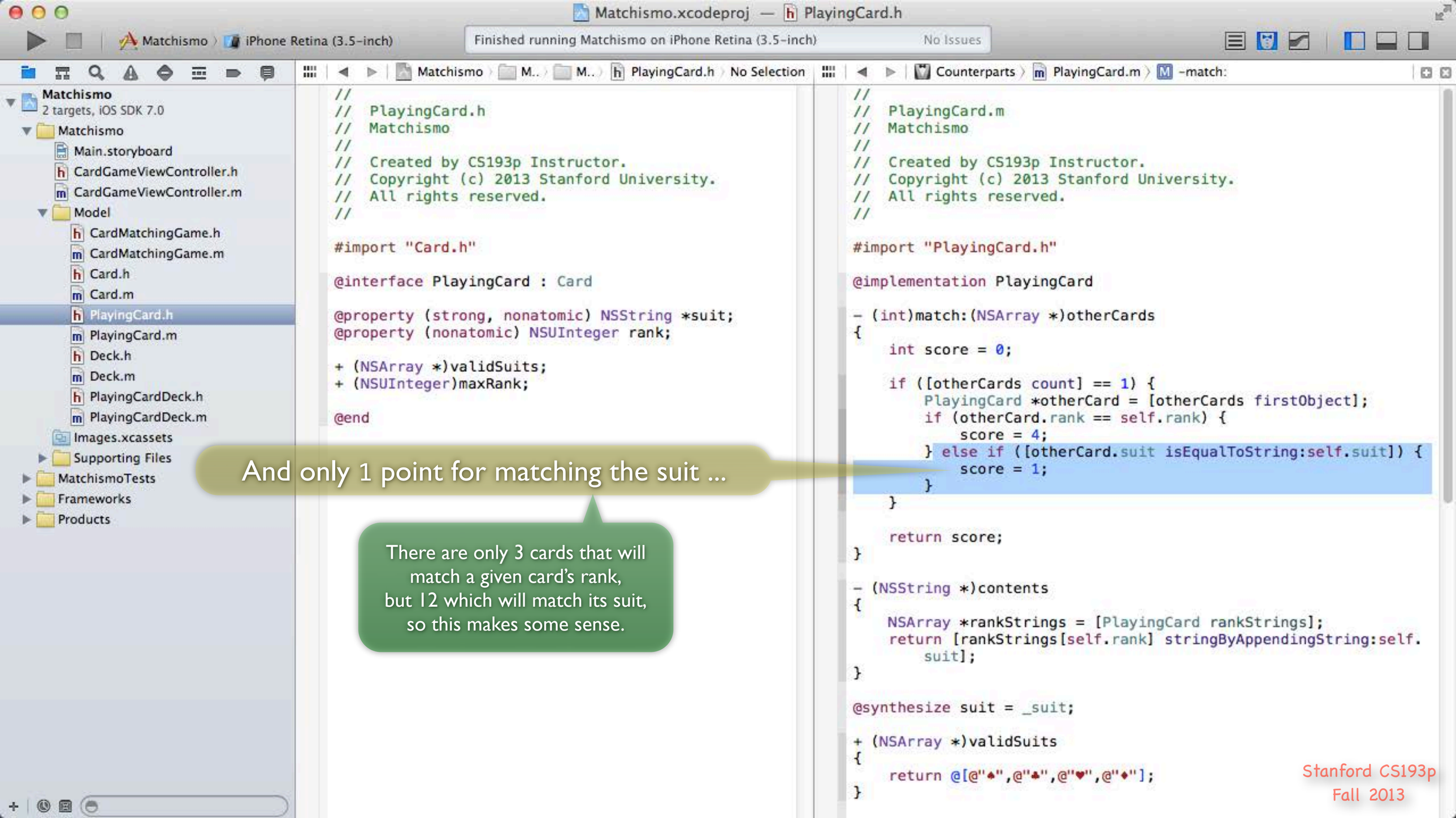

        }
    }

    return score;
}

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

@synthesize suit = _suit;

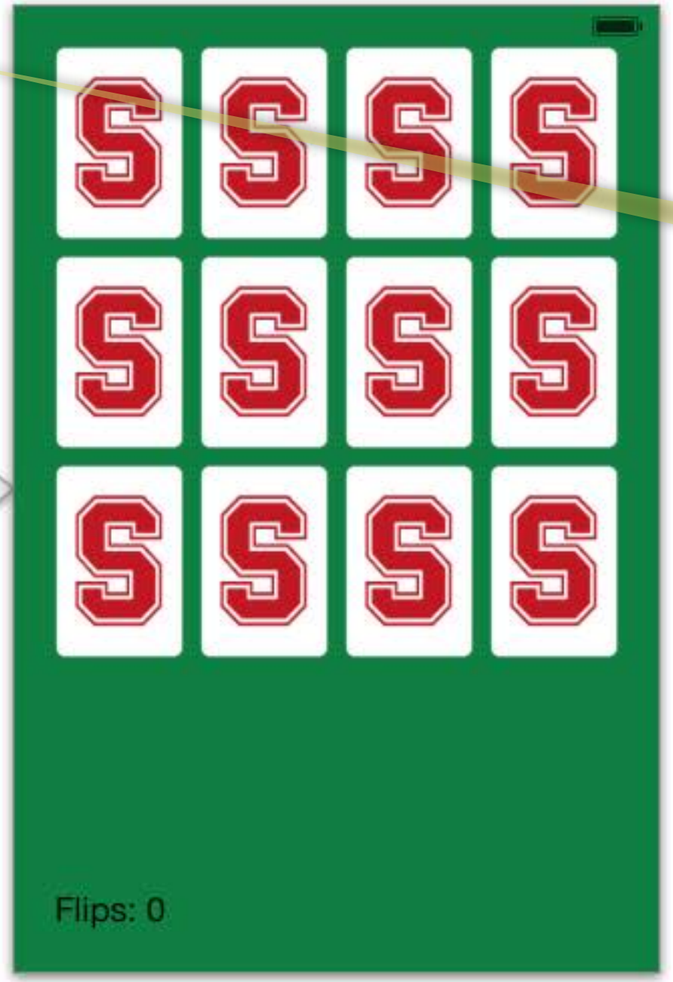
+ (NSArray *)validSuits
{
    return @[@"♠",@"♣",@"♥",@"♦"];
}
```


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.
Stanford CS193p
Fall 2013

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.

Stanford CS193p
Fall 2013

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;
}

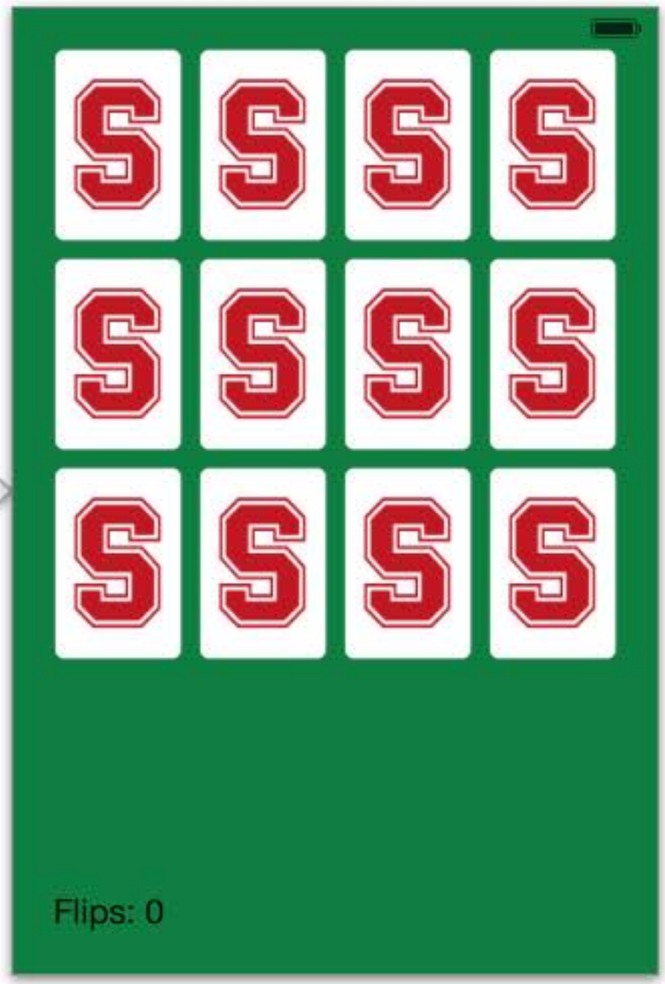
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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.




```
#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;
}

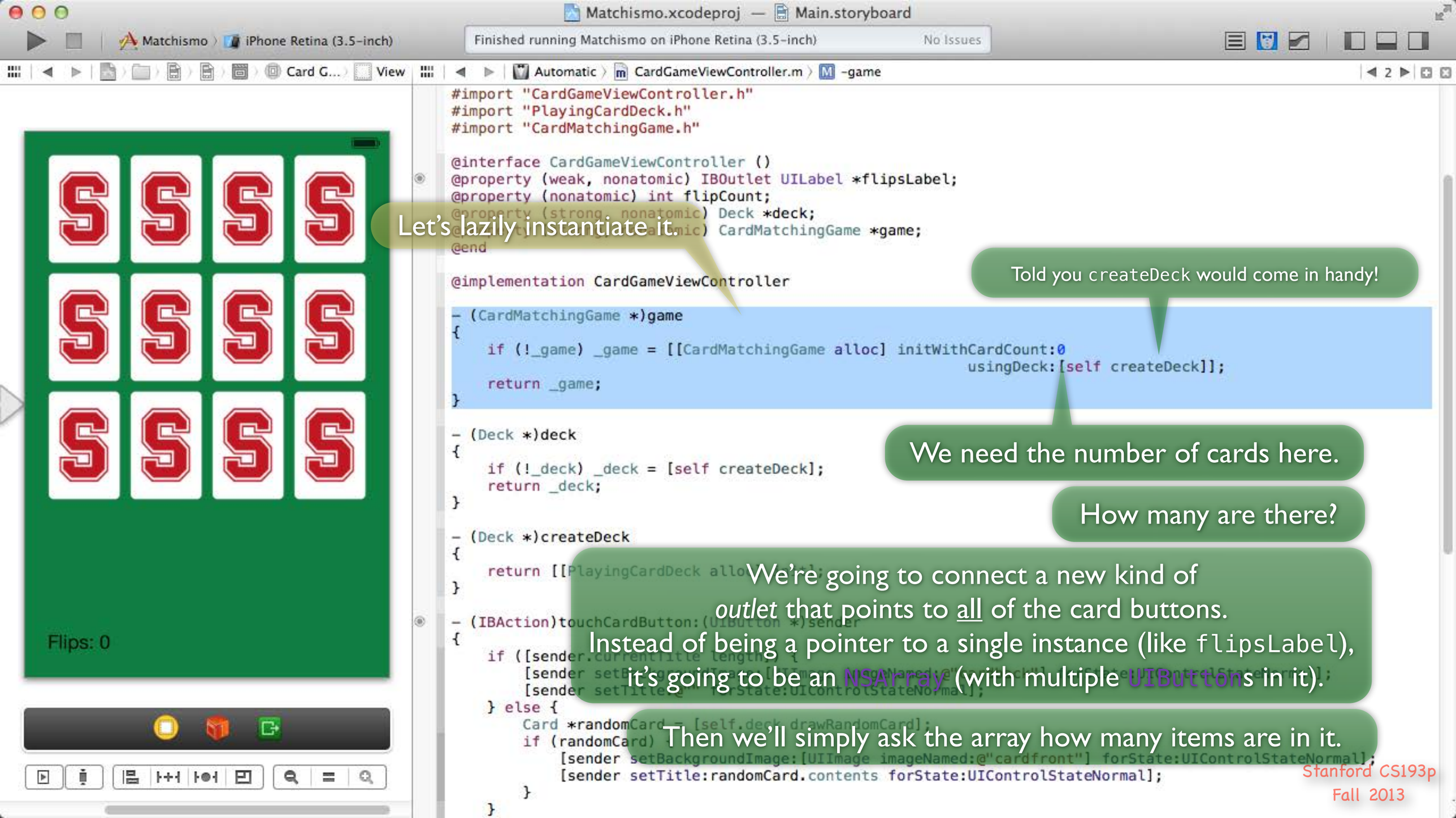
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

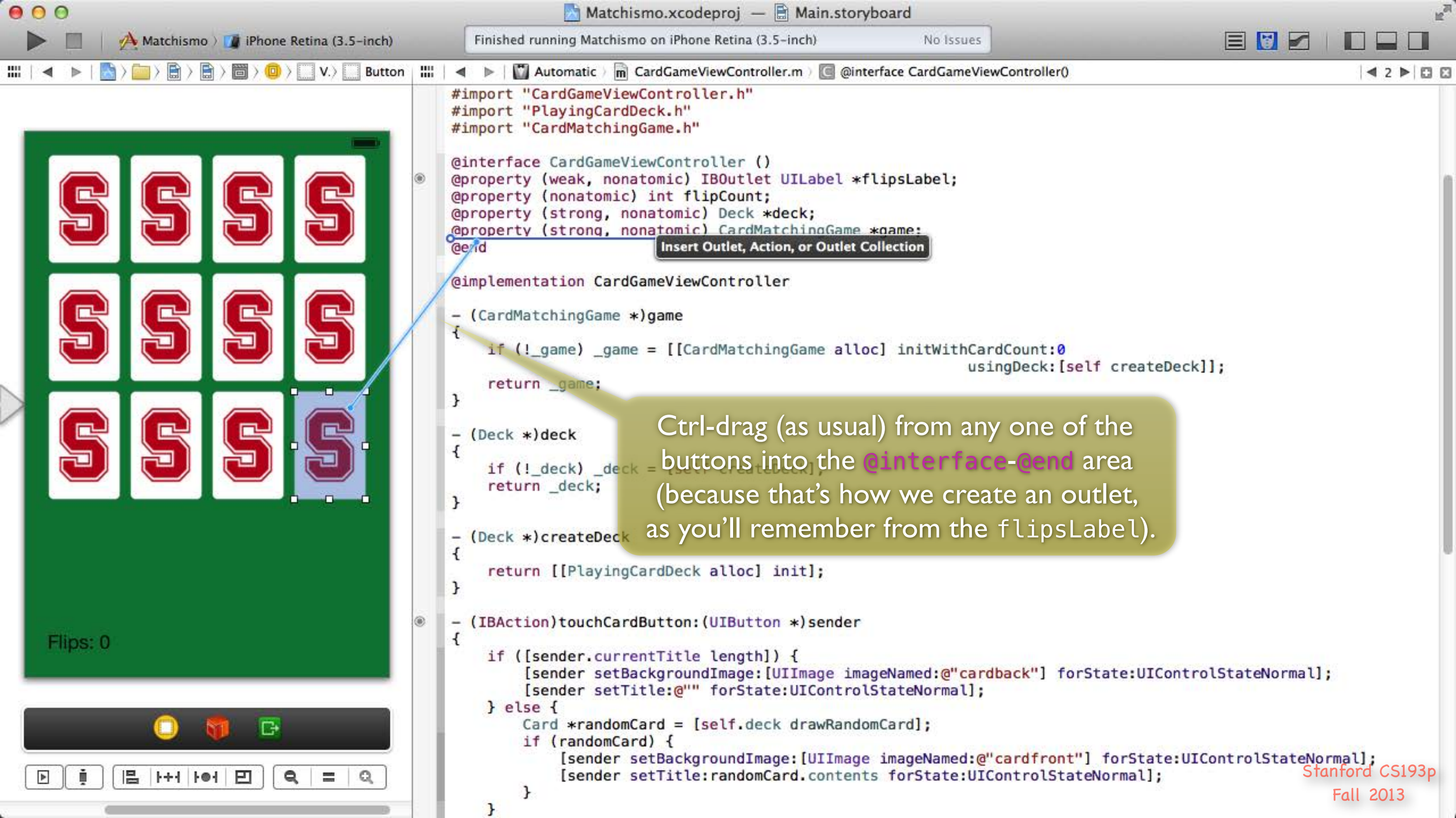
- (IBAction)touchCardButton:(UIButton *)sender
{
    if ([sender.currentTitle length]) {
        [sender setBackgroundImage:[UIImage imageNamed:@"cardback"] forState:UIControlStateNormal];
        [sender setTitle:@"" forState:UIControlStateNormal];
    } else {
        Card *randomCard = [self.deck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

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

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

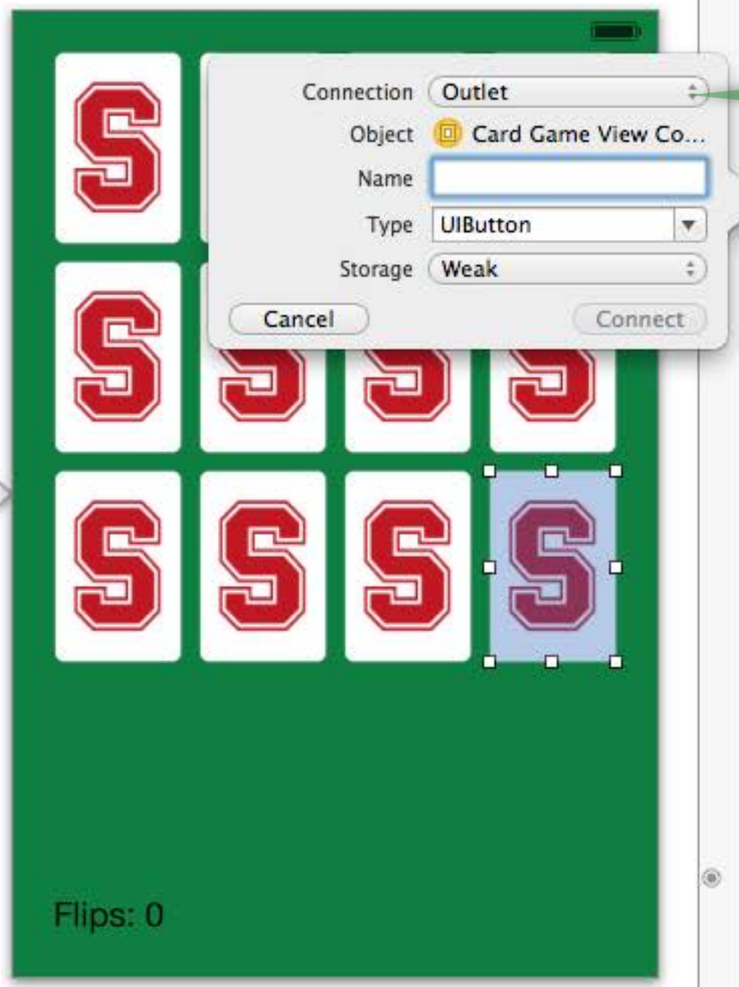

Matchismo.xcodeproj — Main.storyboard
Matchismo › iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Card G... › View
Automatic › CardGameViewController.m › -game
Flips: 0
#import "CardGameViewController.h"
#import "PlayingCardDeck.h"
#import "CardMatchingGame.h"
@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.
Stanford CS193p
Fall 2013


Matchismo.xcodeproj — Main.storyboard
Matchismo 〉 iPhone Retina (3.5-inch)
Finished running Matchismo on iPhone Retina (3.5-inch)
No Issues
Automatic 〉 CardGameViewController.m 〉 @interface CardGameViewController()
#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
Insert Outlet, Action, or Outlet Collection
@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];
}
}
Flips: 0
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).
Stanford CS193p
Fall 2013

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                                    insLabel;
@property (nonatomic) int flipCount;
@property (strong, nonatomic) Deck *deck;
@property (strong, nonatomic) CardMatchingGame *game;
@end

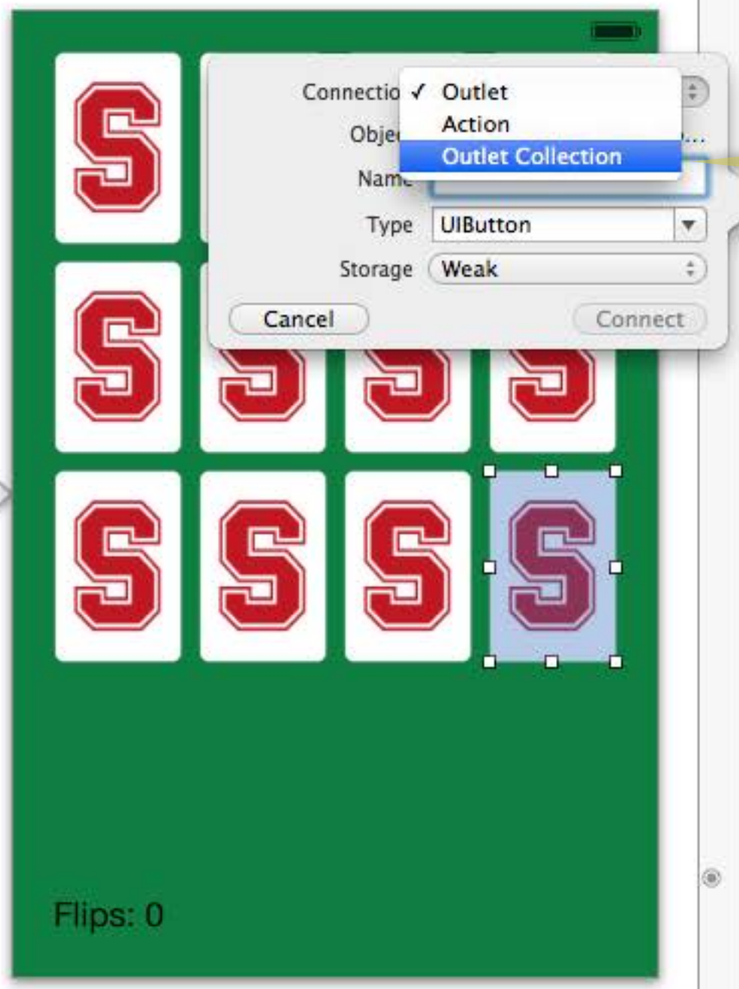
@implementation CardGameViewController

- (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.

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

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



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.

```
#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;
}

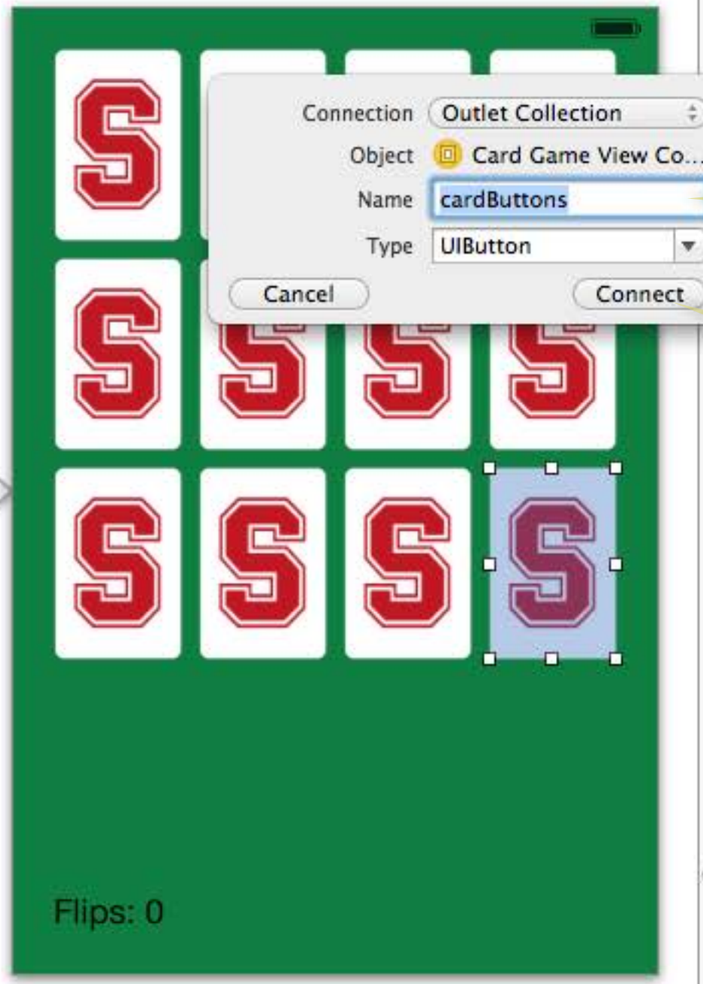
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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];
        }
    }
```

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
@end

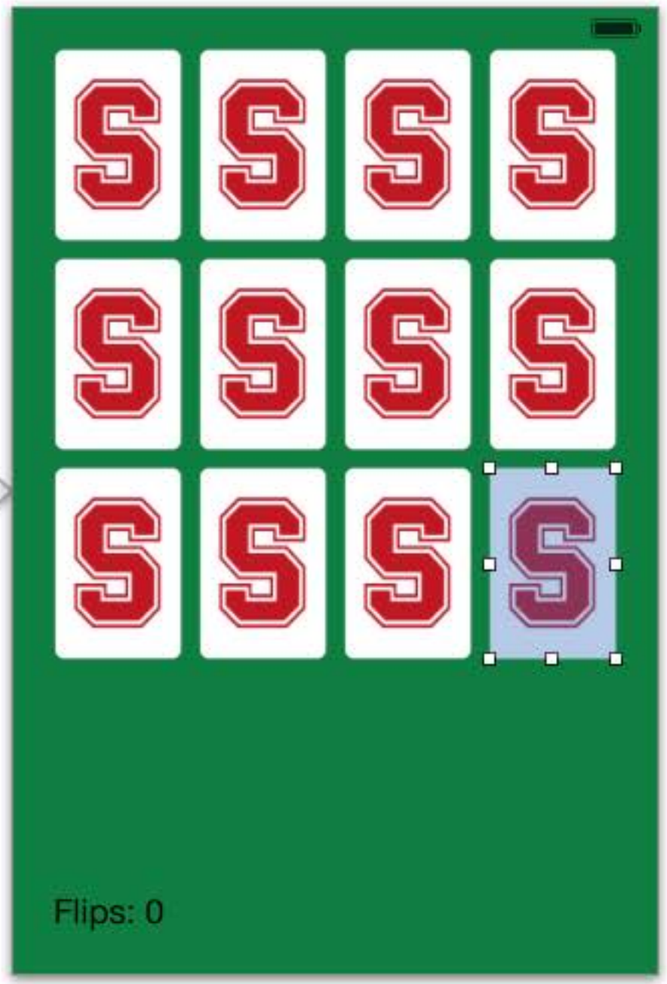
@implementation CardGameViewController

- (CardMatchingGame *)game
{
    if (!_game
                                    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;
}

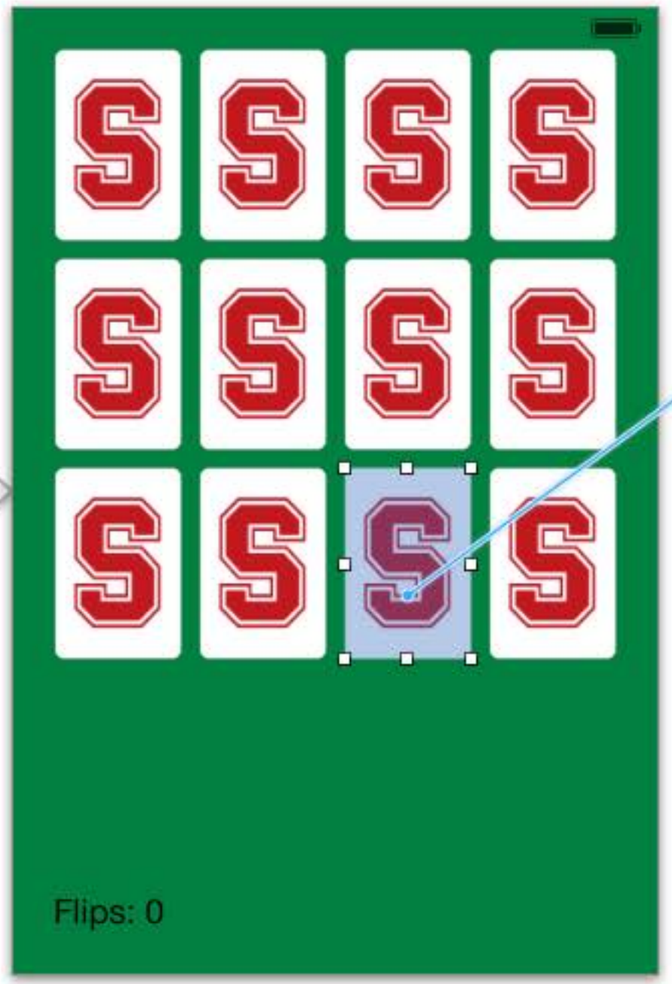
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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;
}

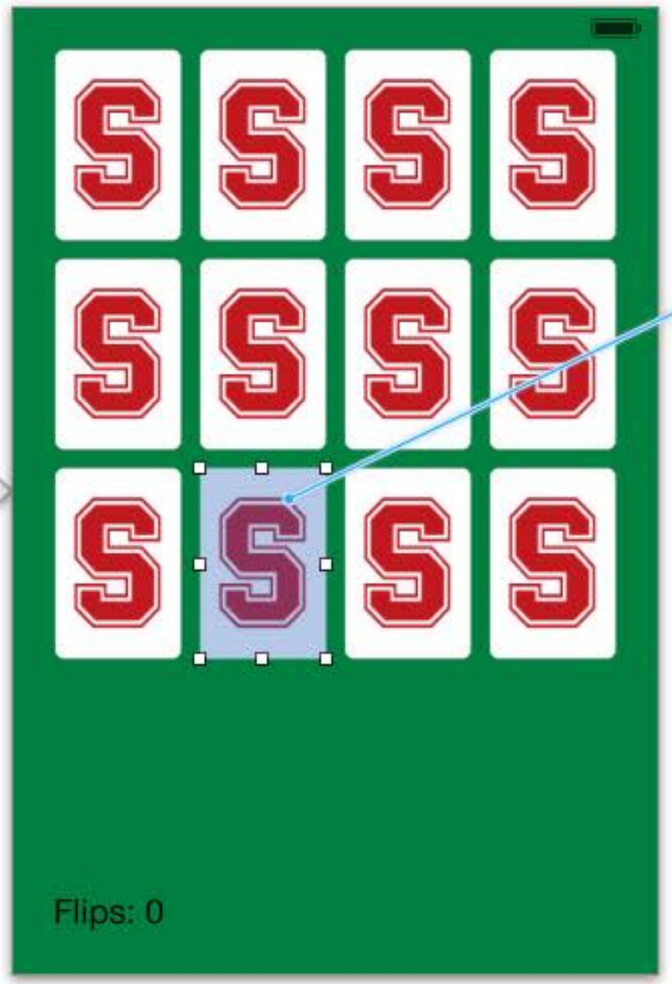
- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (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];
        }
```

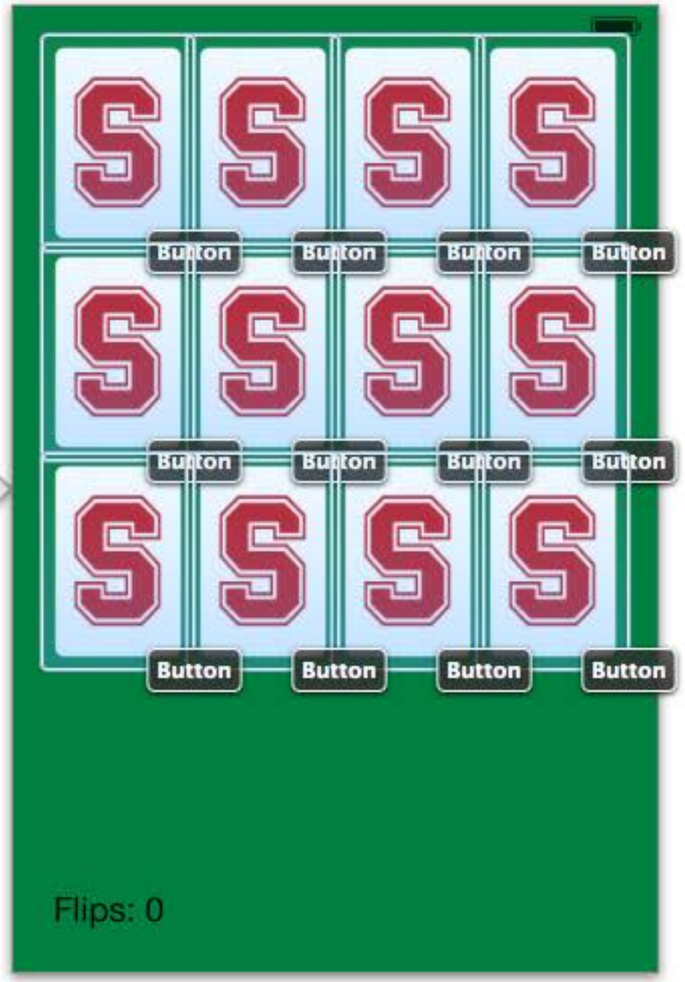
Connect Outlet Collection

Ditto a third.

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 › @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;
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *cardButtons;
@end

@implementation CardGameViewController

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

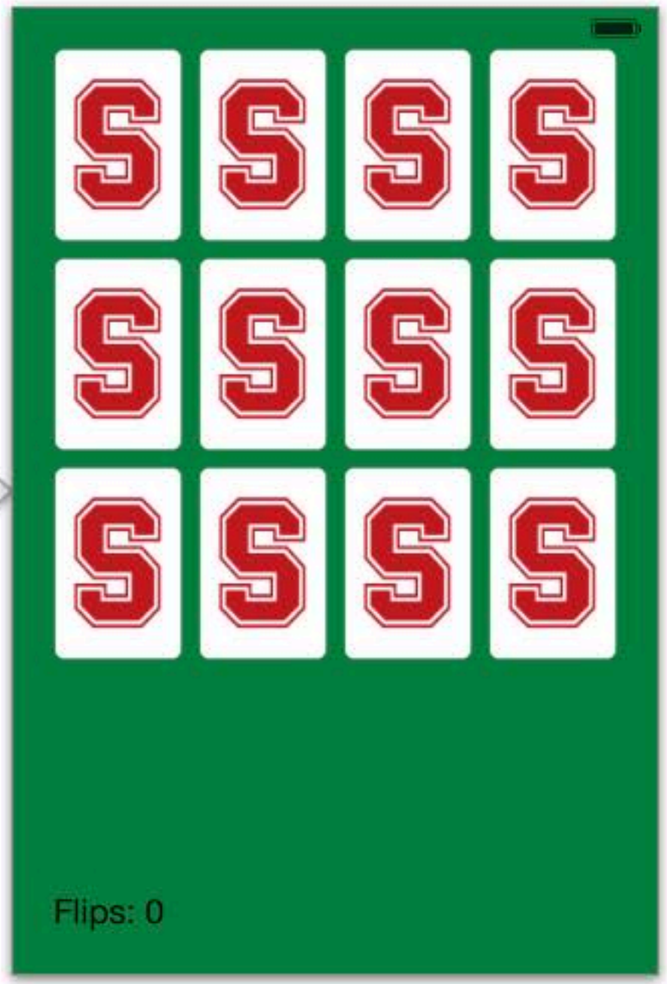
- (Deck *)deck
{
    if (!_deck) _deck = [self createDeck];
    return _deck;
}

- (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).




```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

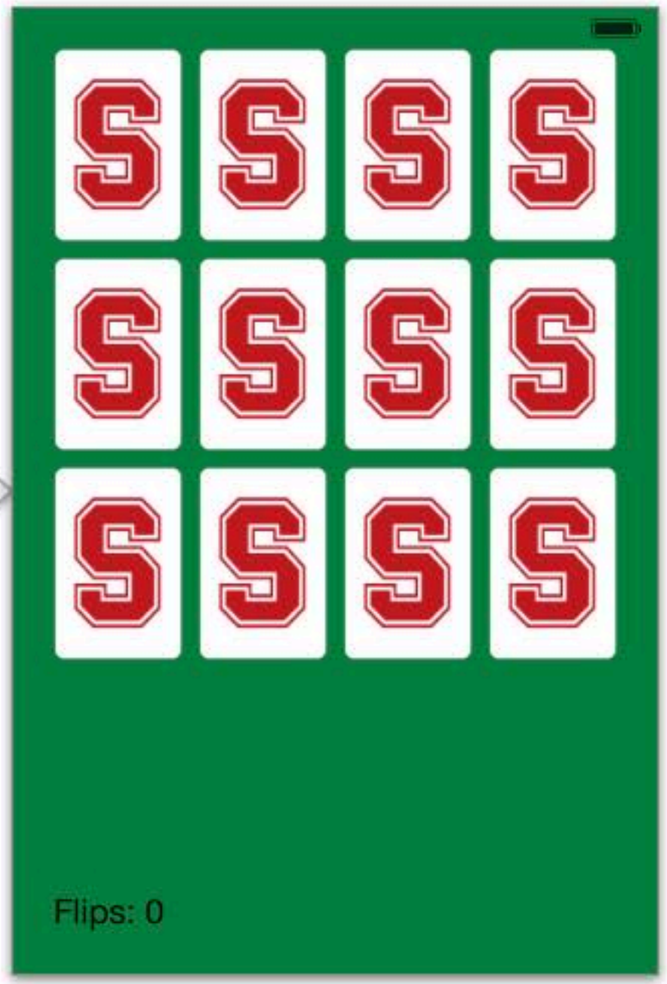
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (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];
        }
```

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




```
#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

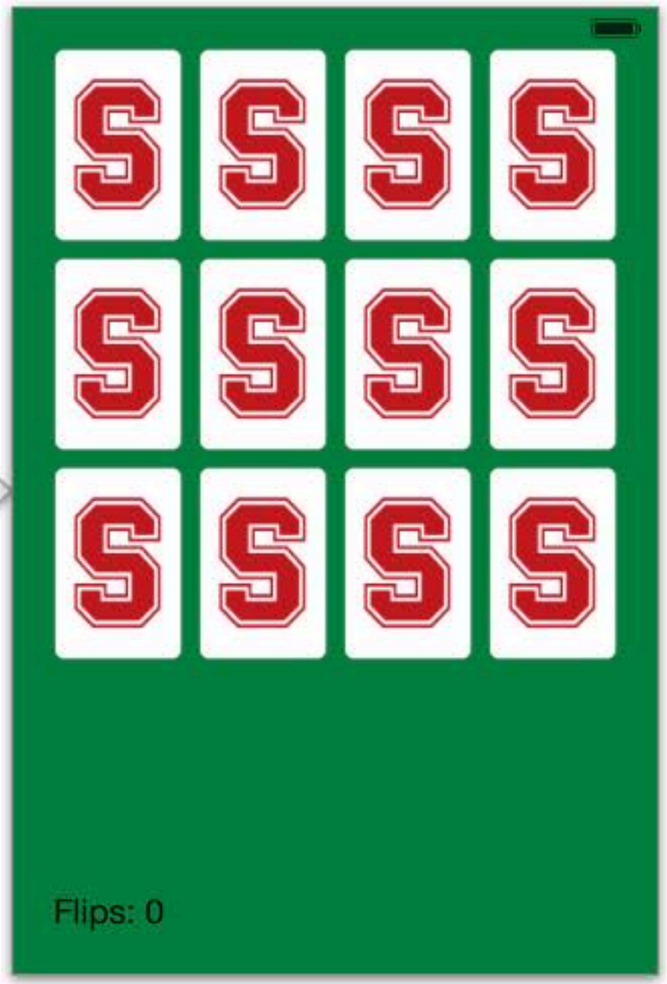
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (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.




```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;
}

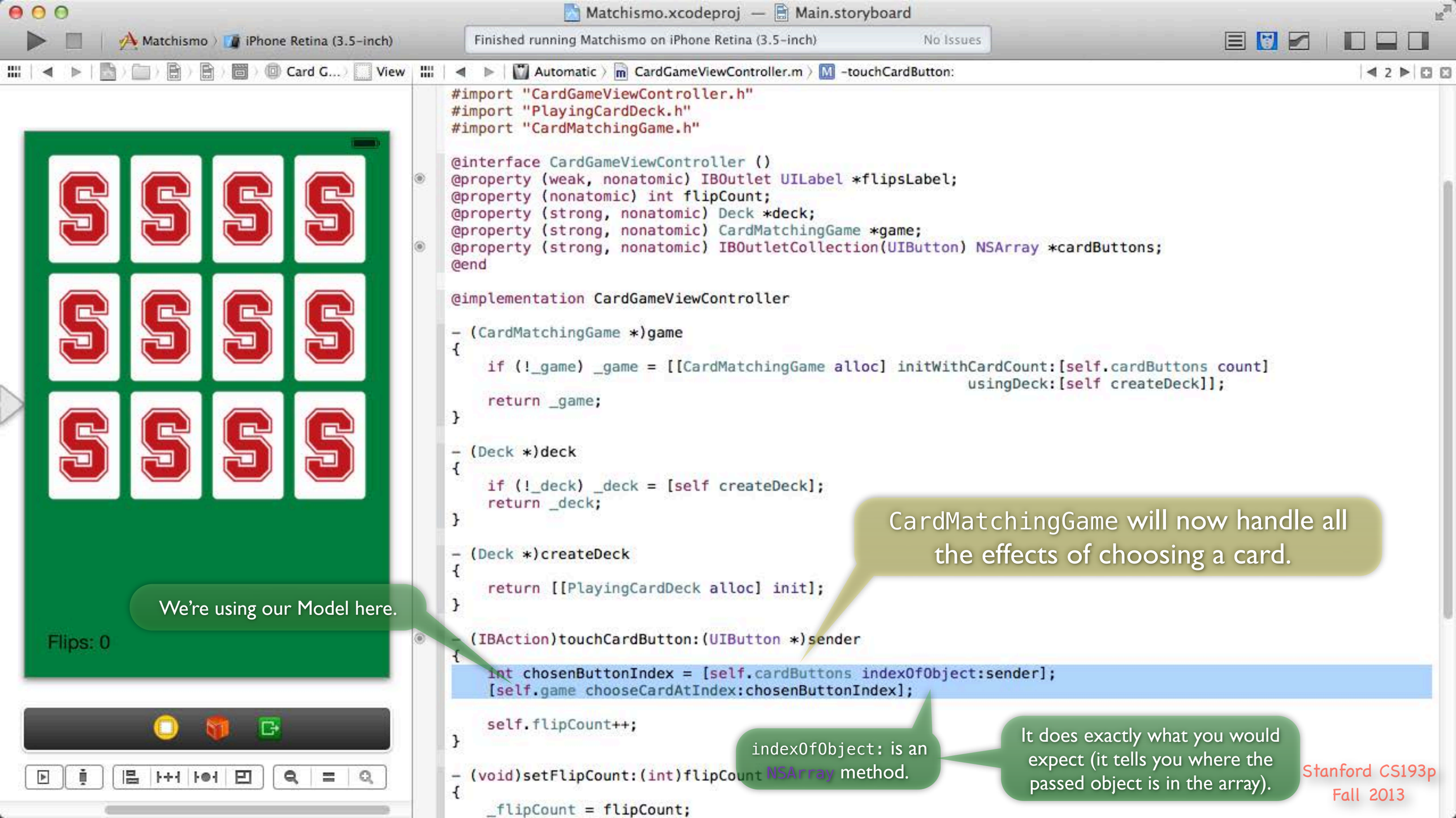
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

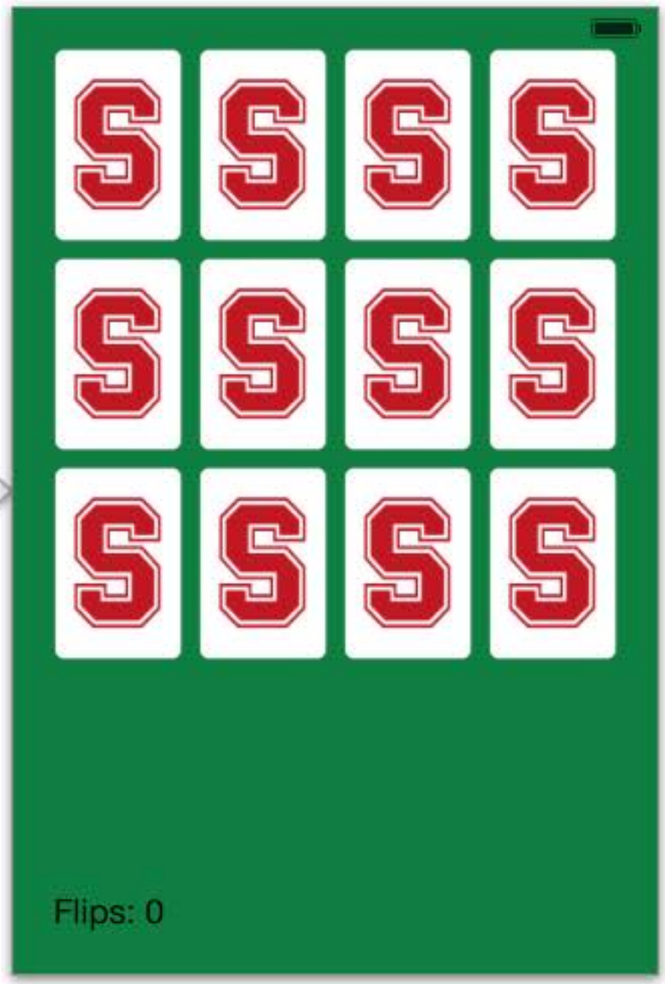
- (IBAction)touchCardButton:(UIButton *)sender
{

    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.


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"
#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;
Flips: 0
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).
Stanford CS193p
Fall 2013




```
#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;
}

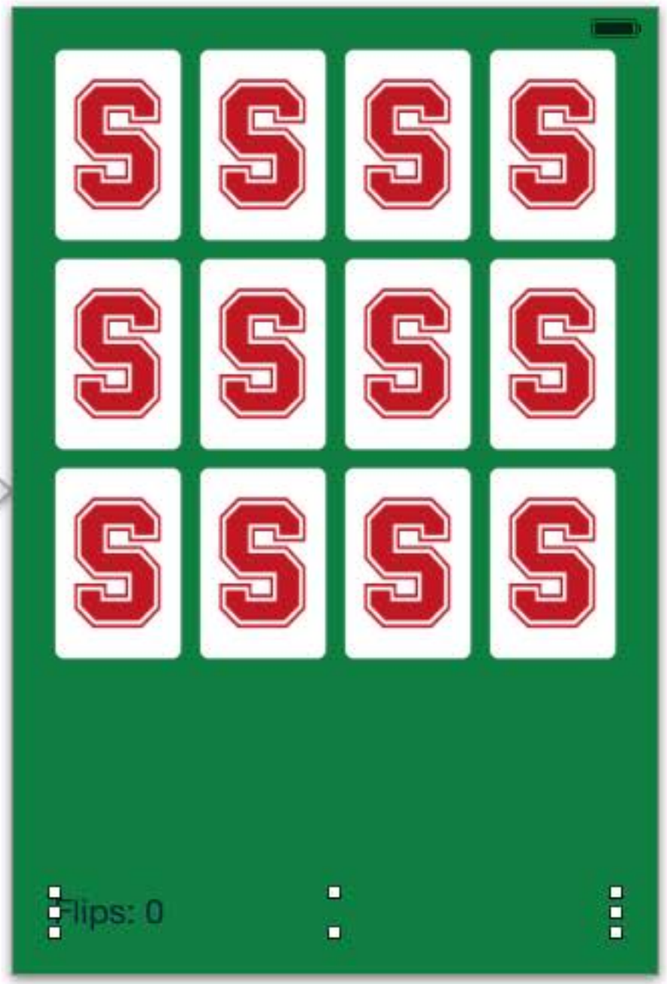
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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.




```
#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;
}

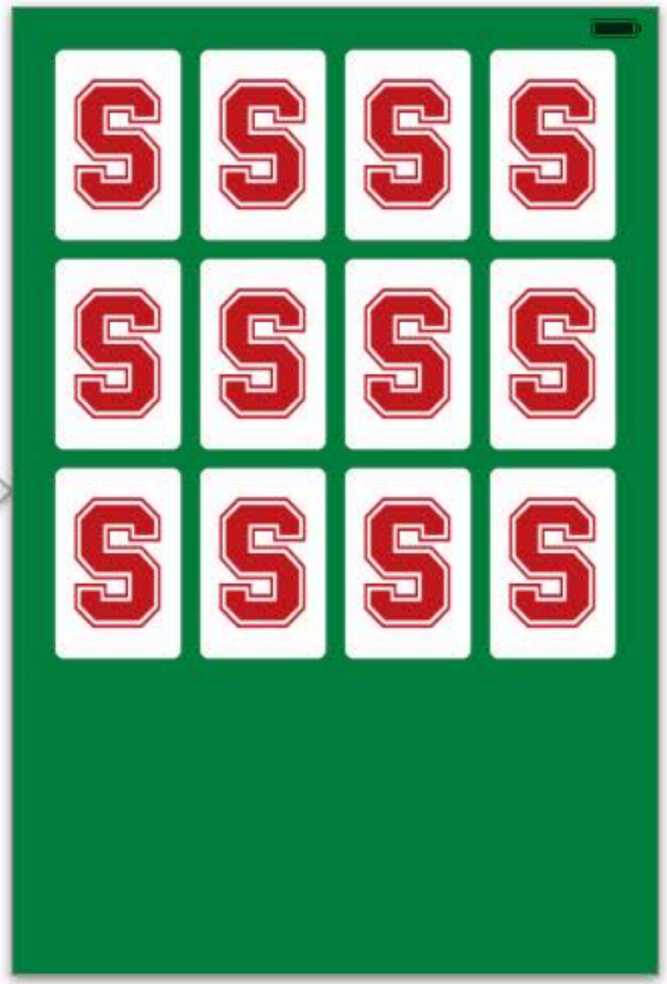
- (Deck *)createDeck
{
    return [[PlayingCardDeck alloc] init];
}

- (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];
}

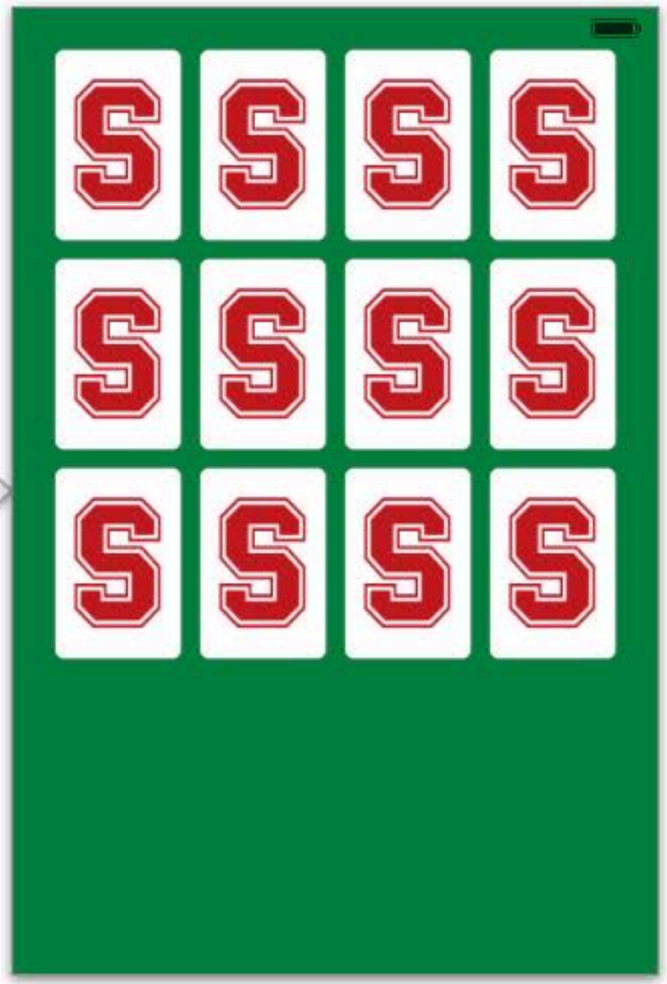
- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (void)updateUI
{




}

@end
```



```
@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
{

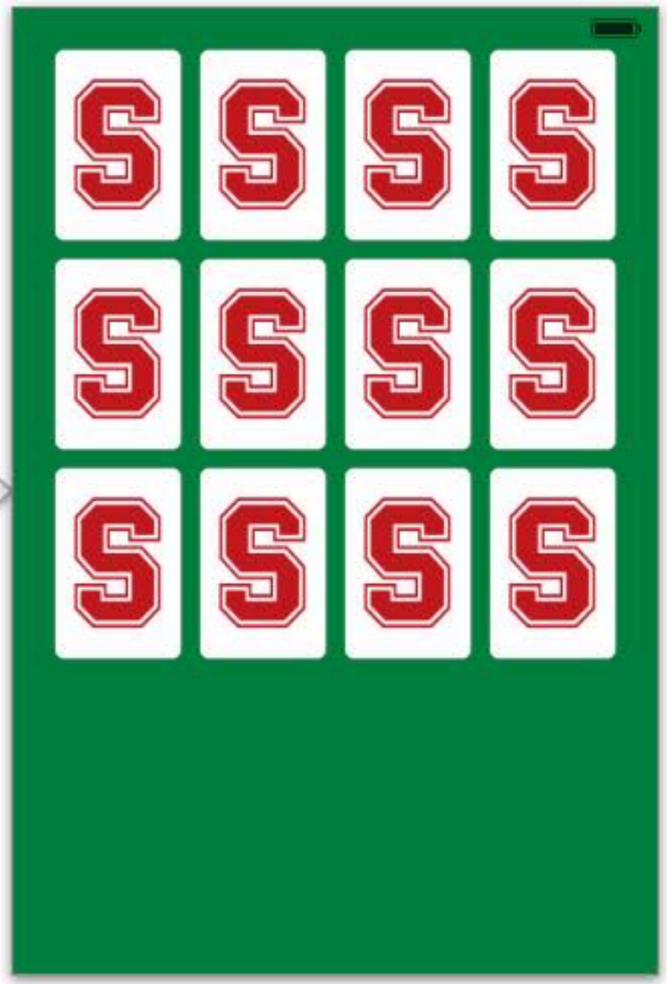


}



@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];
}

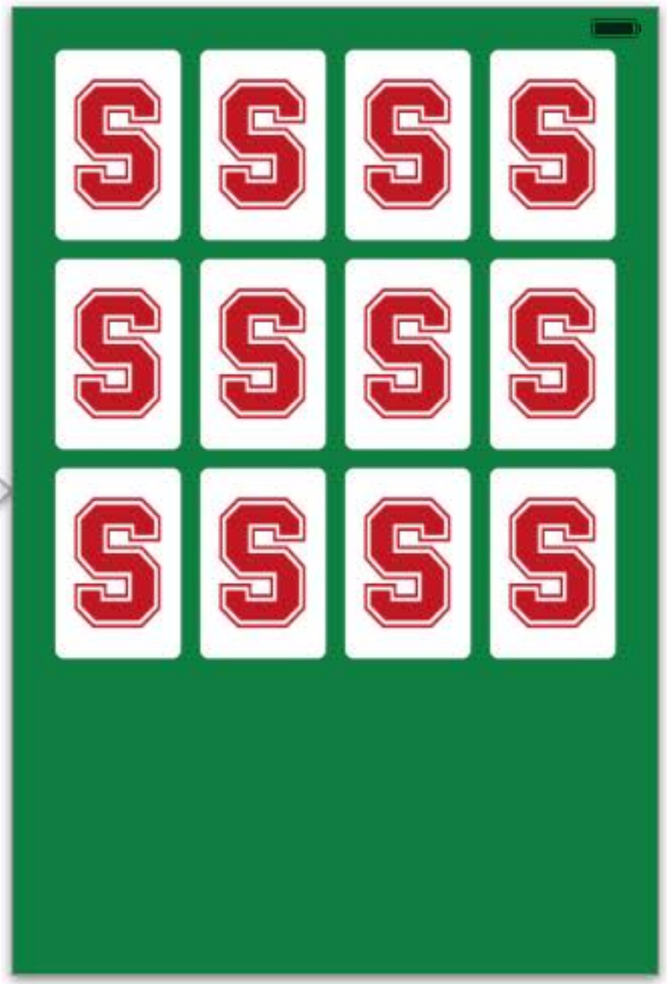
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];



    }
}
```

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 ...

```
@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];
}

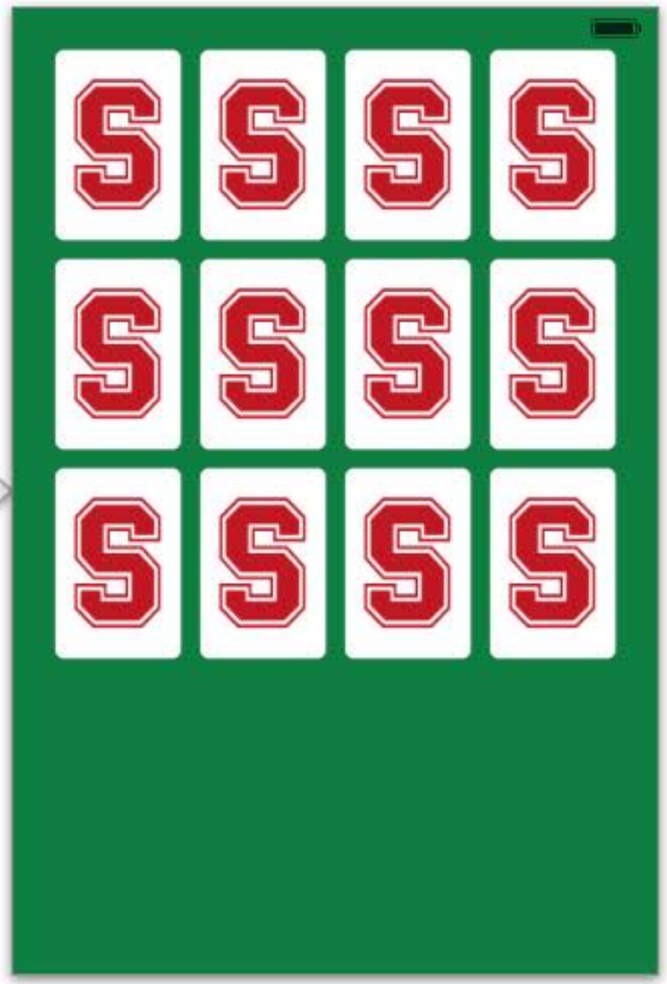
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:(NSString *) forState:UIControlStateNormal];
        [cardButton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

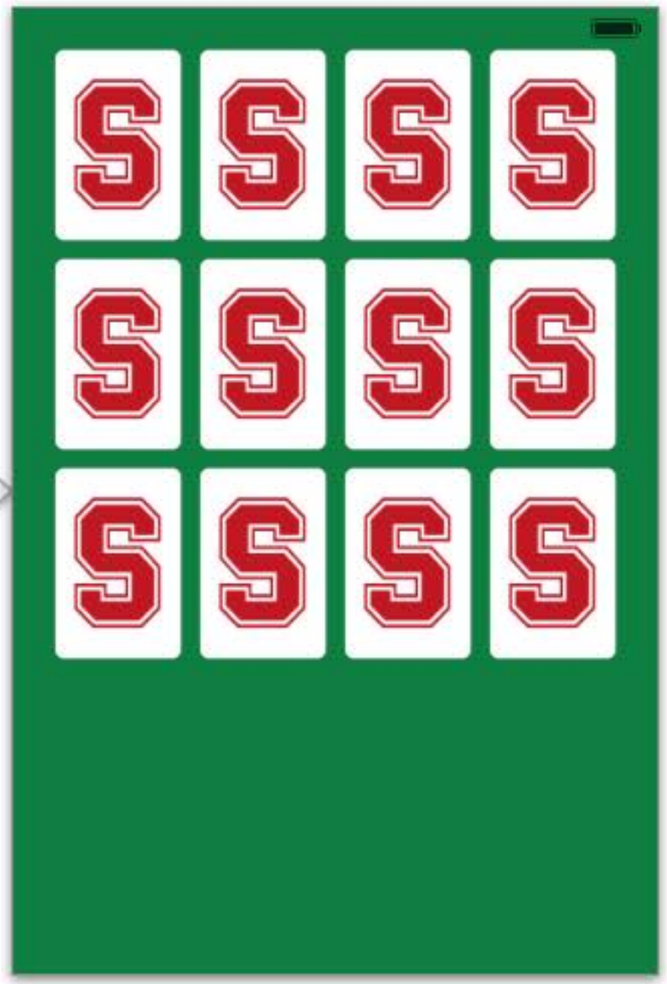
    }
}

- (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];

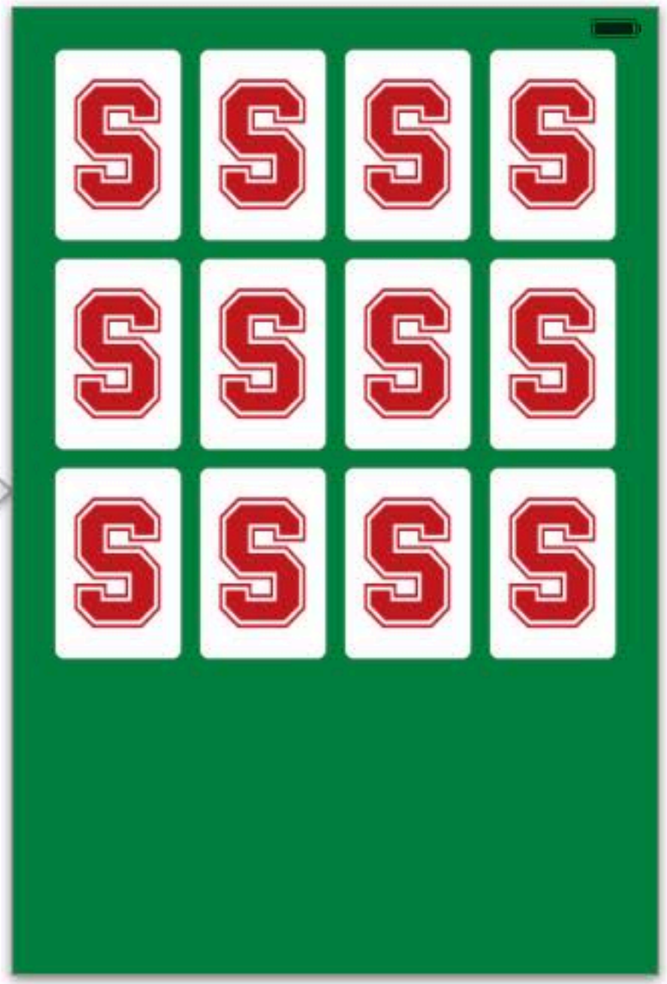
    }
}

- (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];
}

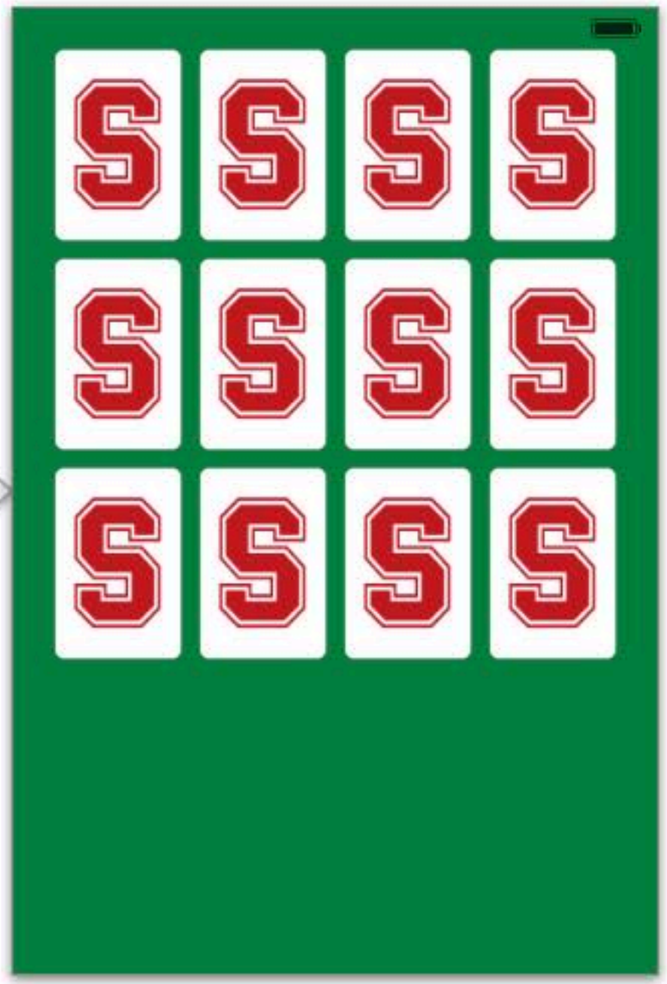
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
     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

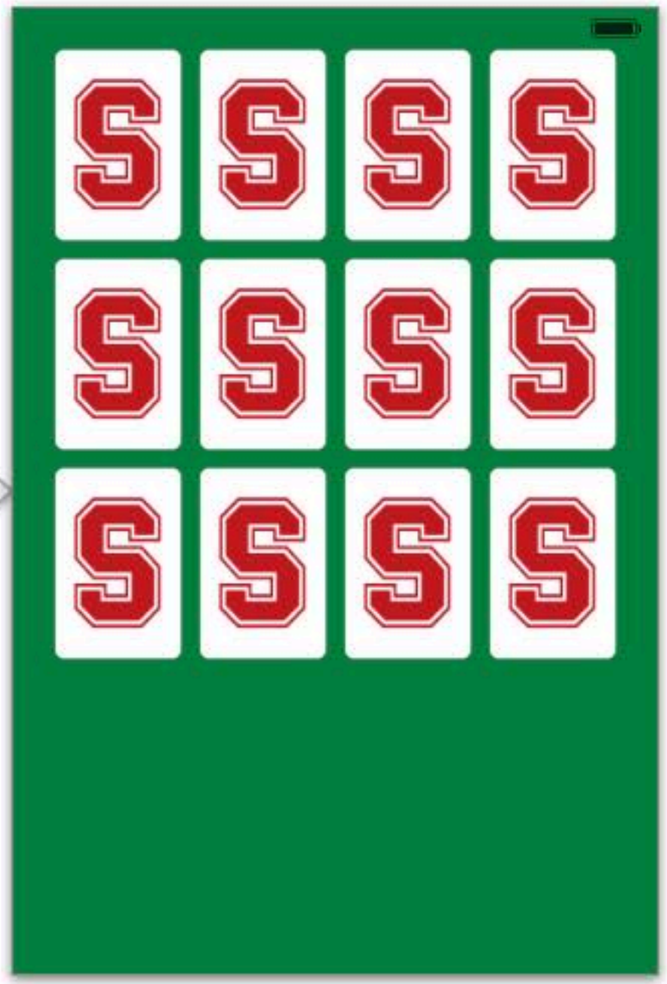
- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

- (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
```



```
@implementation CardGameViewController

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

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

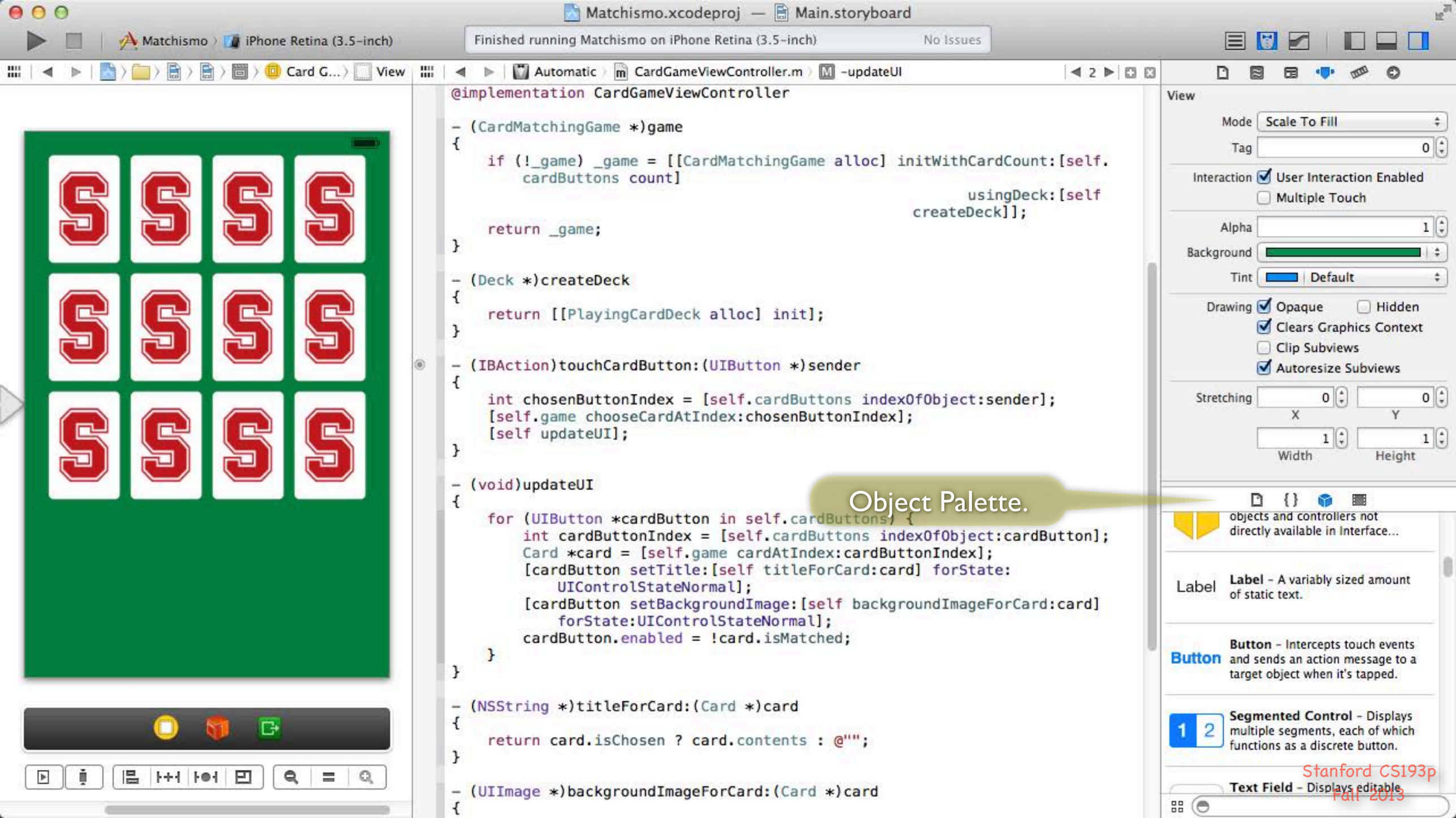
- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

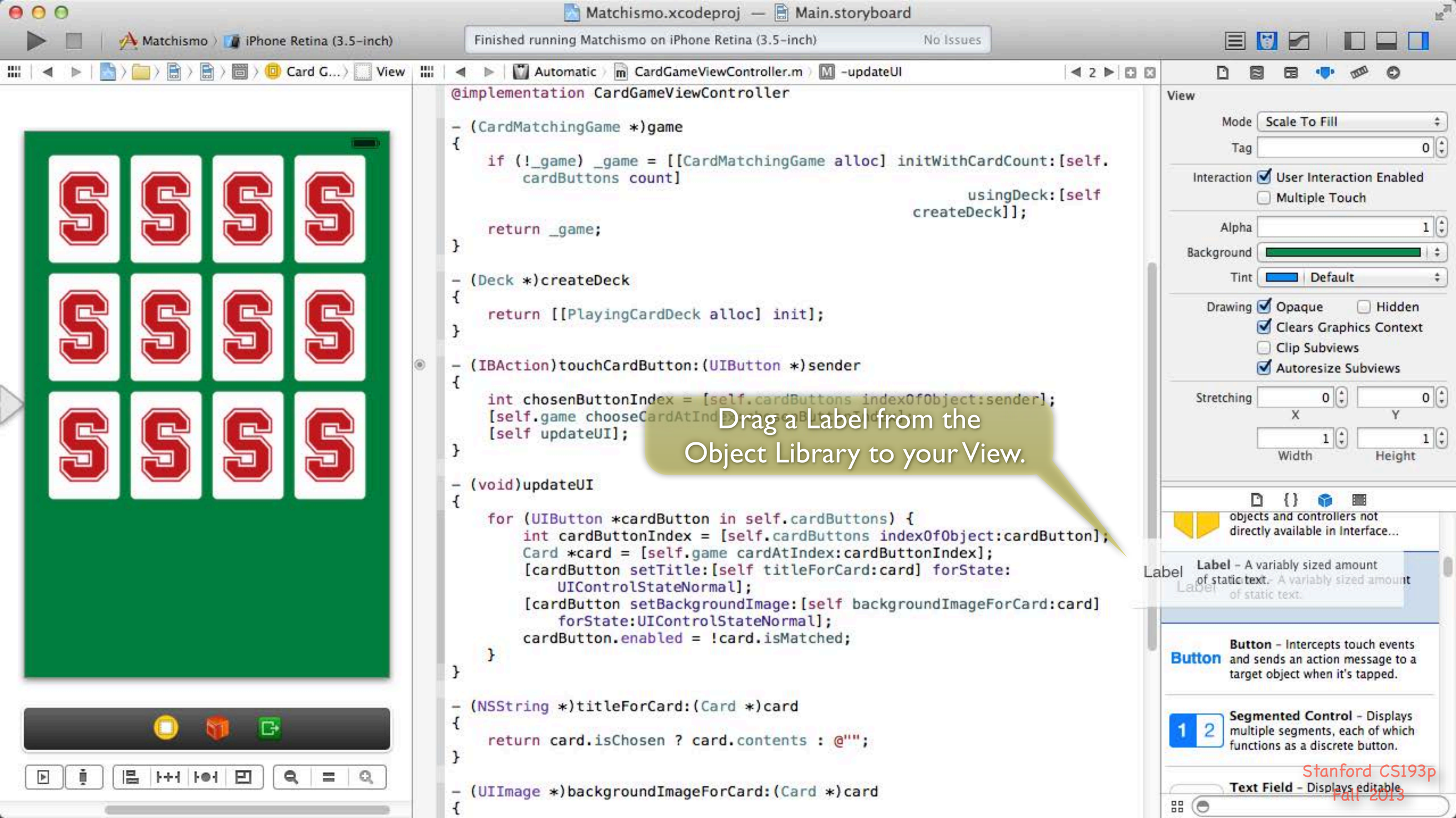
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:
            UIControlStateNormal];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

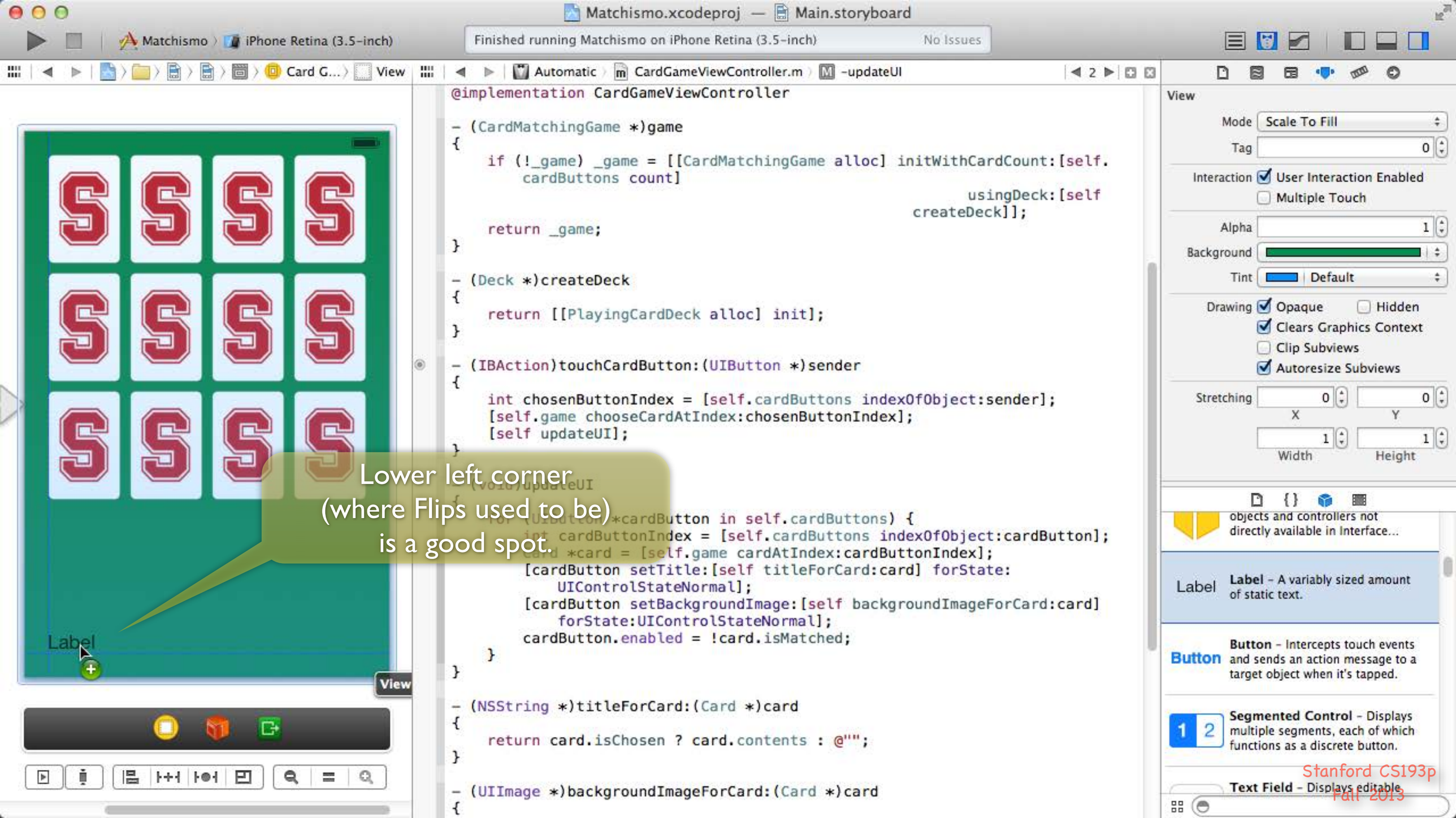
- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

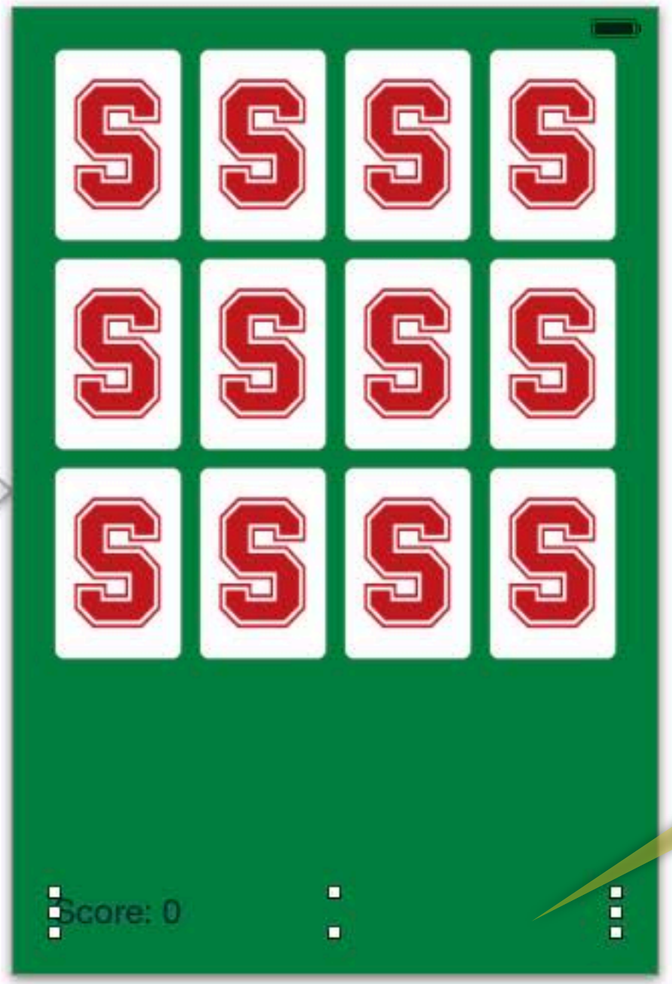
- (UIImage *)backgroundImageForCard:(Card *)card
{
```

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
{
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
Object Palette.
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
Stanford CS193p
Fall 2013


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
Stanford CS193p
Fall 2013


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
{
Label
View
Lower left corner
(where Flips used to be)
is a good spot.
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
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
Stanford CS193p
Fall 2013



Start with "Score: 0" and make it wider.

```
@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
{
```

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...

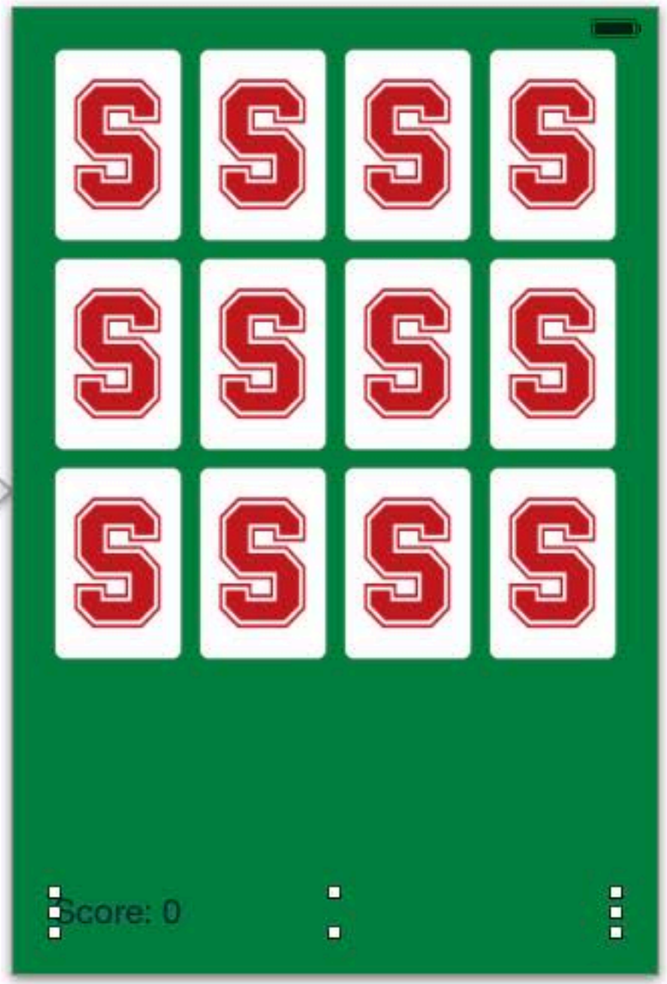
Label – **Label** – A variably sized amount of static text.

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

Stanford CS193p
Fall 2013




```
@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];
}

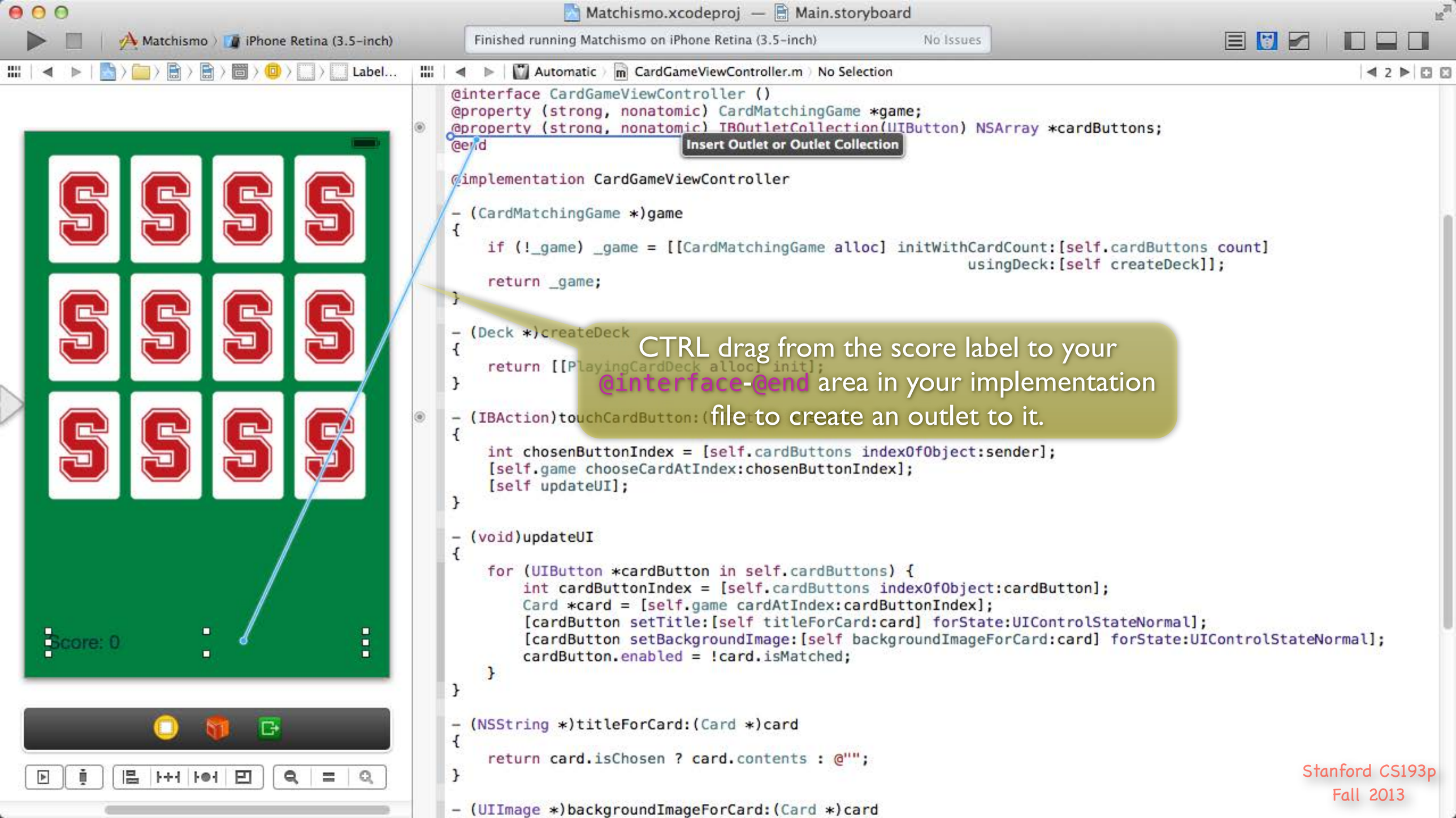
- (IBAction)touchCardButton:(UIButton *)sender
{
    int chosenButtonIndex = [self.cardButtons indexOfObject:sender];
    [self.game chooseCardAtIndex:chosenButtonIndex];
    [self updateUI];
}

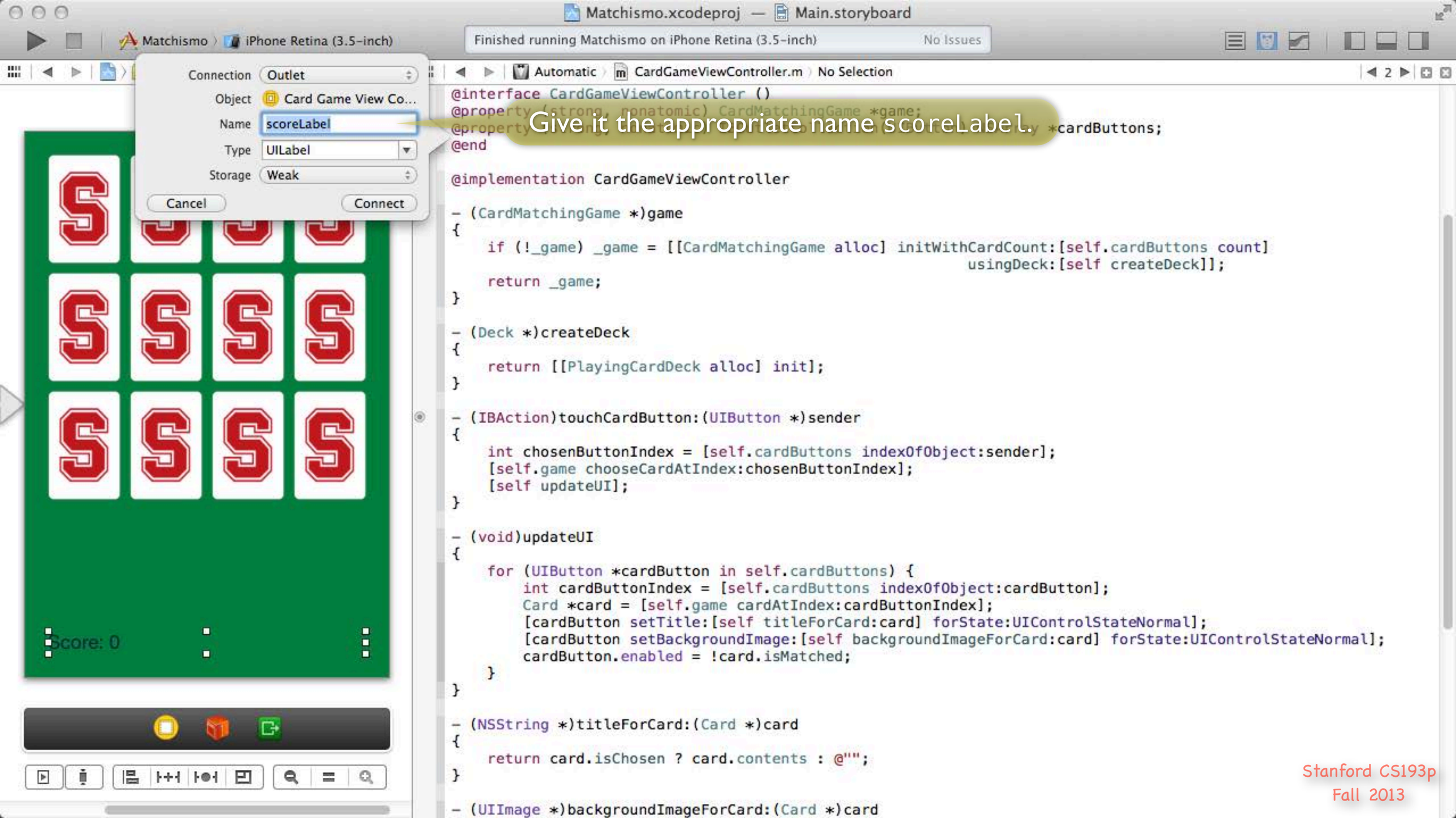
- (void)updateUI
{
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
     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

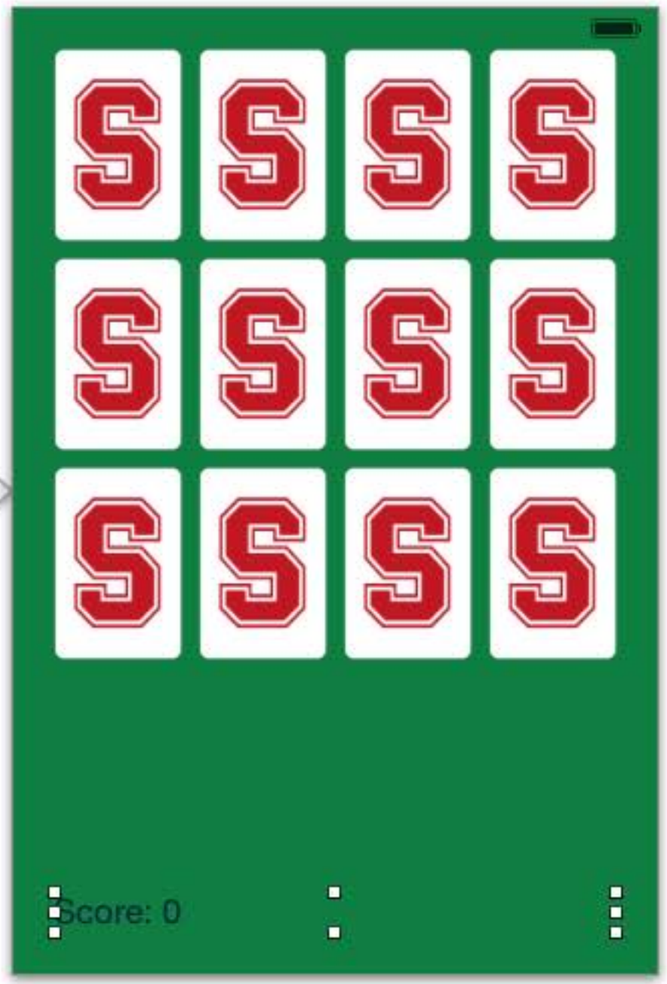
- (NSString *)titleForCard:(Card *)card
{
    return card.isChosen ? card.contents : @"";
}

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

Close Utilities again.


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.
Stanford CS193p
Fall 2013


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
Stanford CS193p
Fall 2013




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

@implementation CardGameViewController

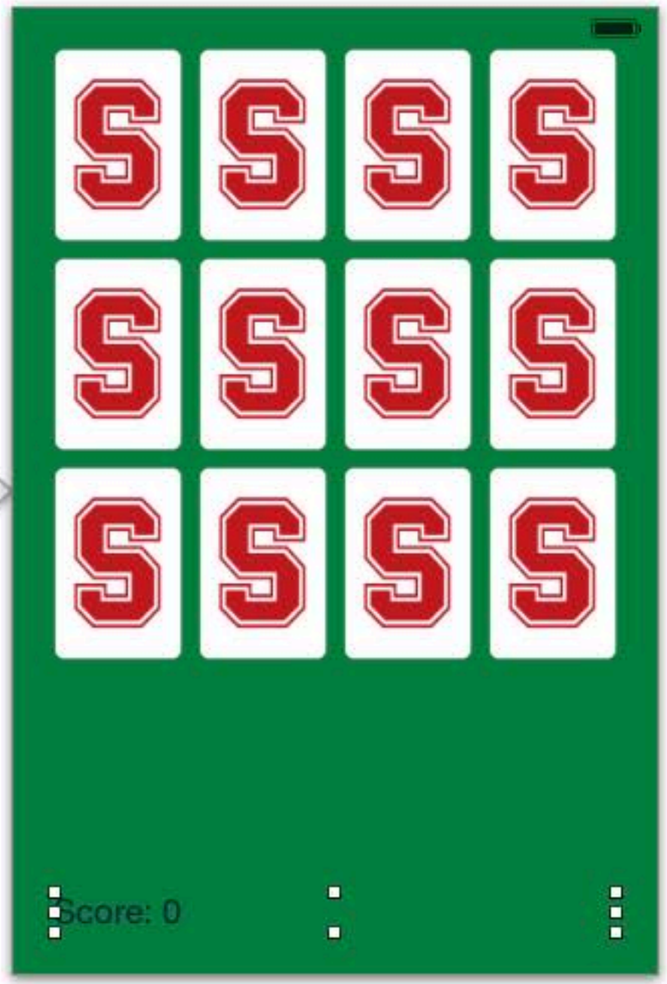
- (CardMatchingGame *)game
{
    if (!_game) _game = [[CardMatchingGame alloc] initWithCardCount:[self.cardButtons count]
                                                          usingDeck:[self createDeck]];
    return _game;
}

- (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 : @"";
}
```




```
@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

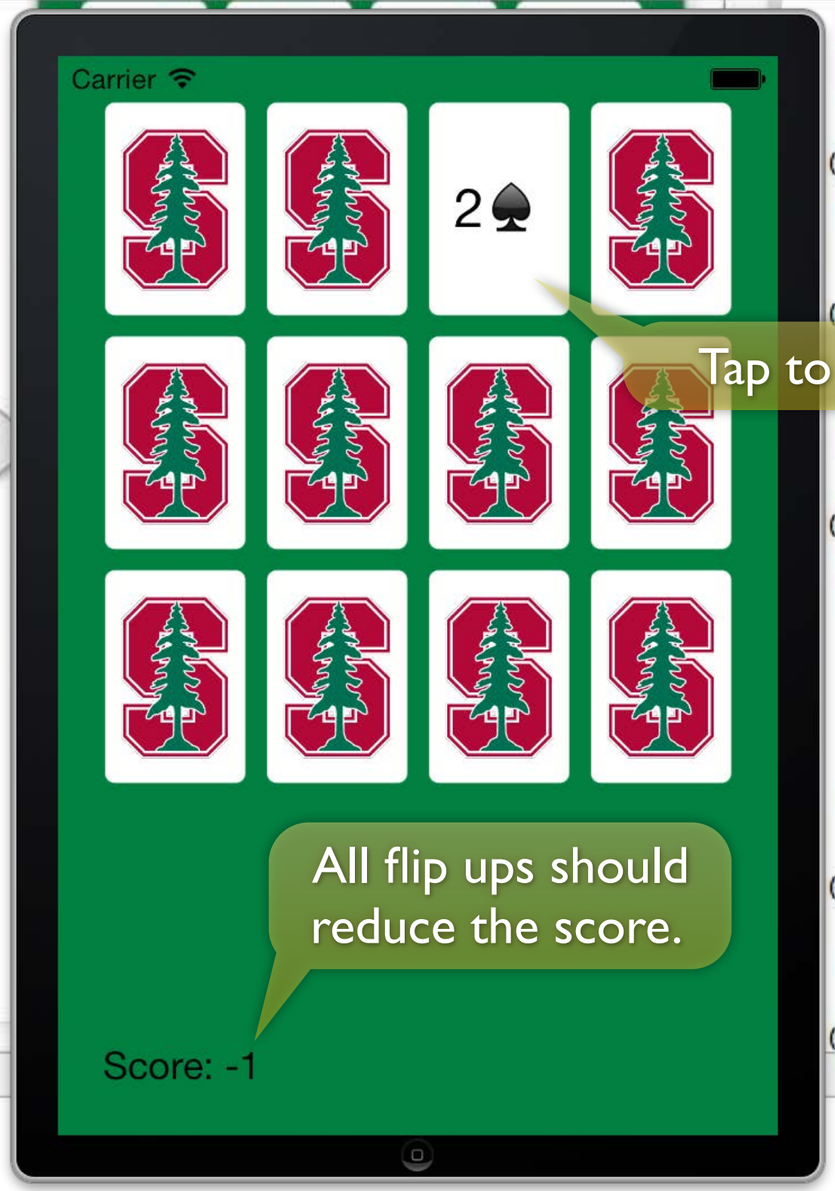
    for (UIButton *cardButton in self.cardButtons) {
        int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
        Card *card = [self.game cardAtIndex:cardButtonIndex];
        [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
        [cardButton setBackgroundImage:[self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

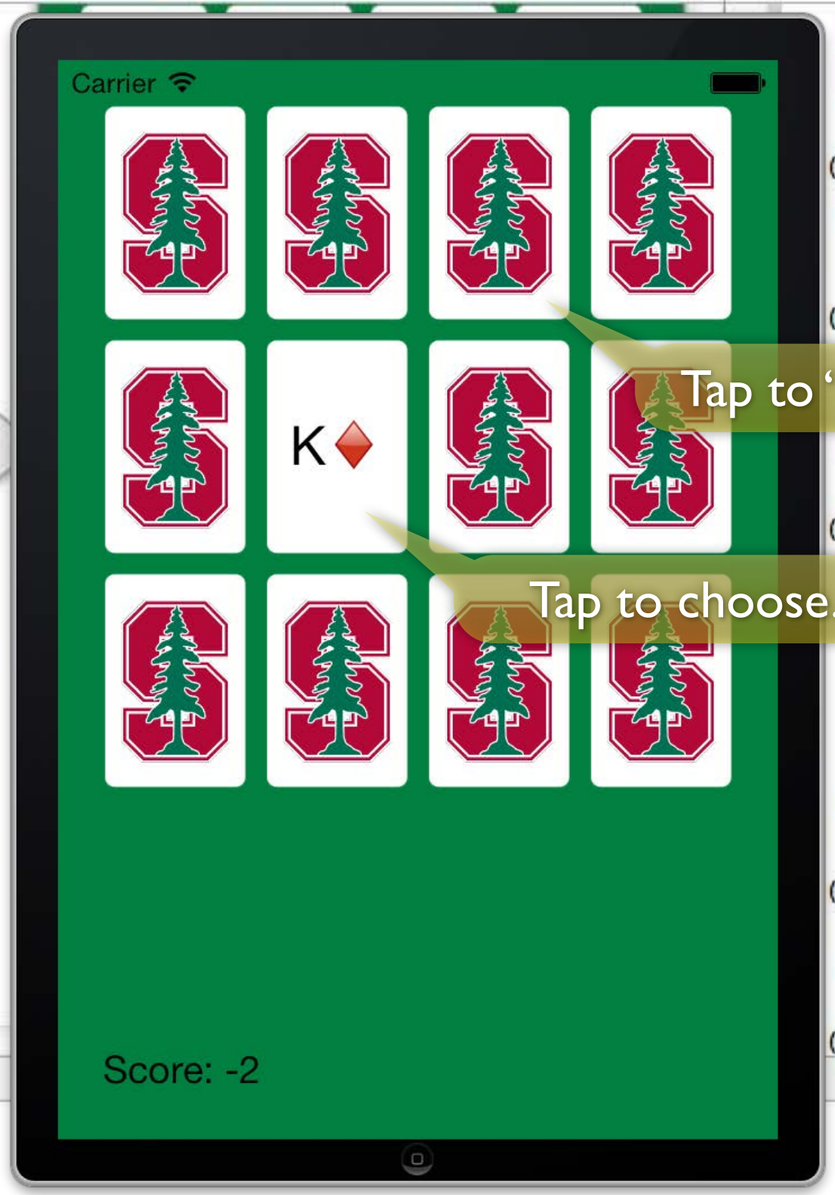
for (UIButton *cardButton in self.cardButtons) {
    int cardButtonIndex = [self.cardButtons indexOfObject:cardButton];
    Card *card = [self.game cardAtIndex:cardButtonIndex];
    [cardButton setTitle:[self titleForCard:card] forState:UIControlStateNormal];
    [cardButton setBackgroundImage:[self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

Stanford CS193p
Fall 2013



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];

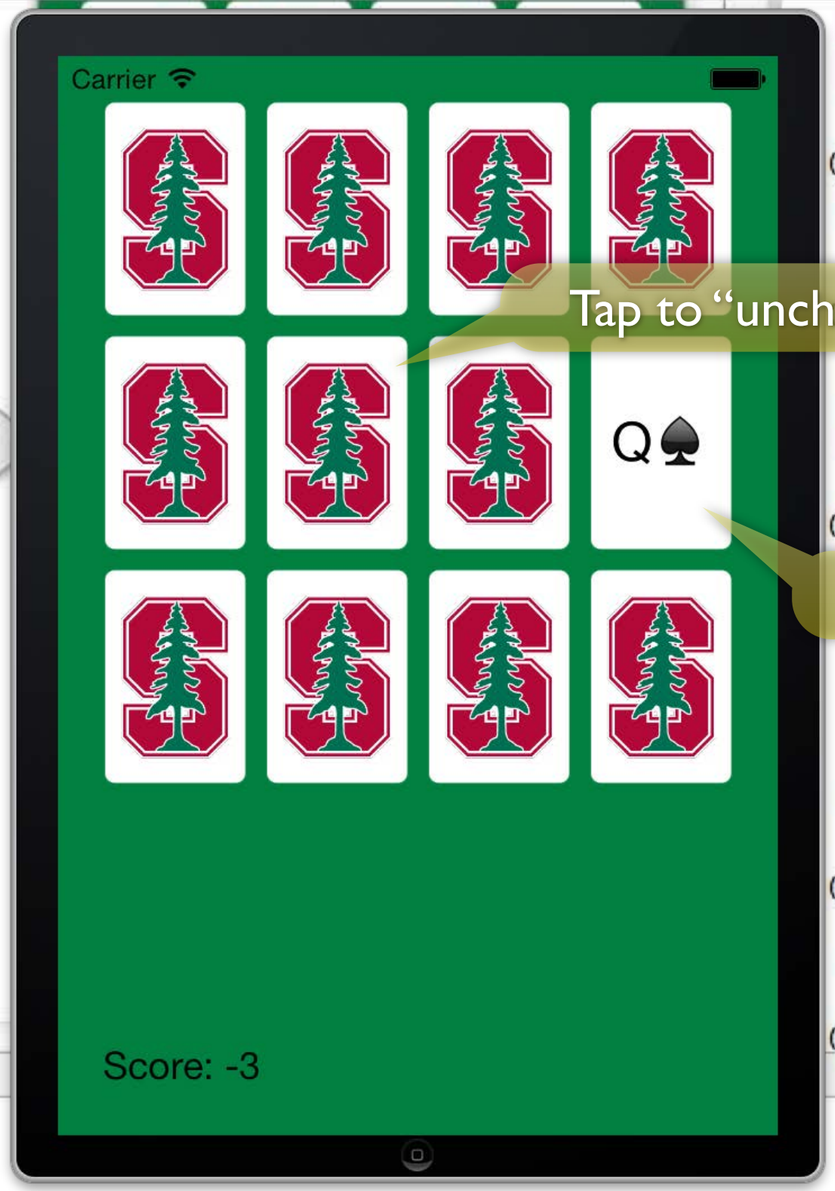
(void)updateUI

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
```



Tap to "unchoose".

Tap to choose.

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

(Deck *)createDeck

return [[PlayingCardDeck alloc] init];

touchCardButton:(UIButton *)sender

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

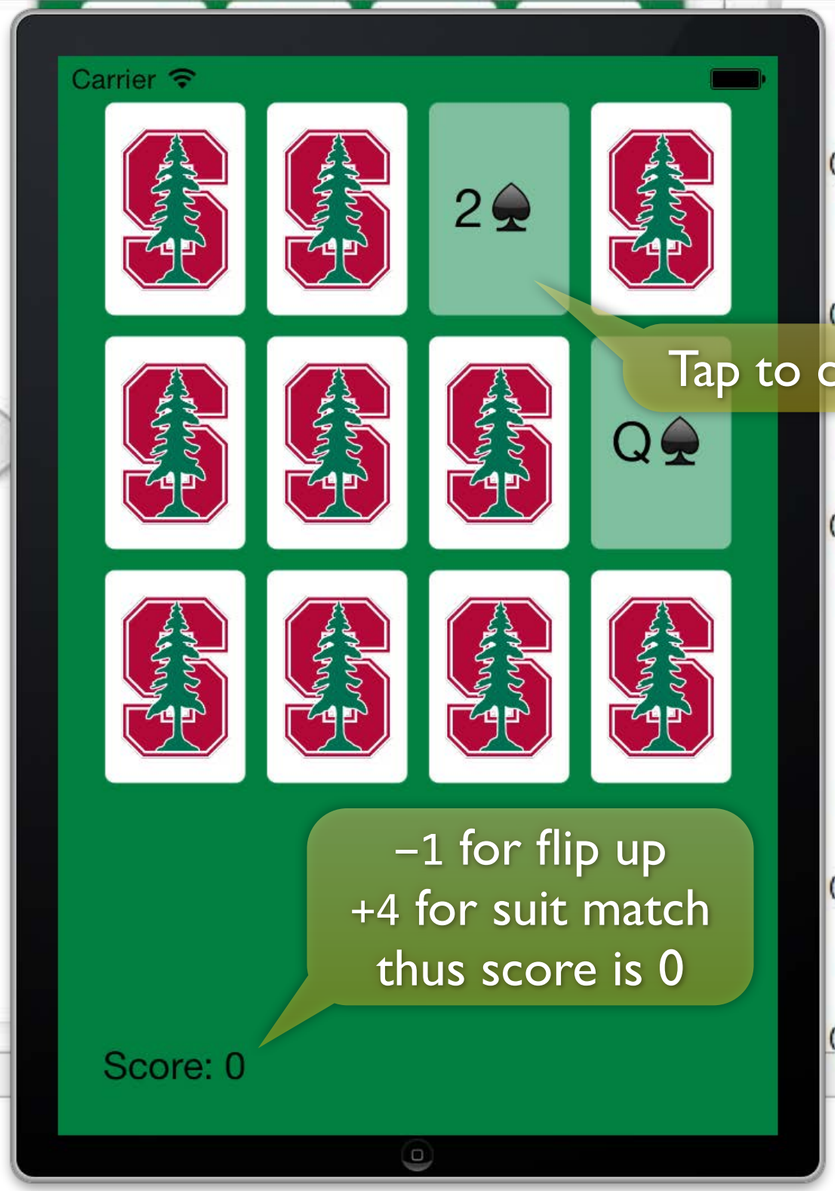
(void)updateUI

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
```




```
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.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

Stanford CS193p
Fall 2013




```
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

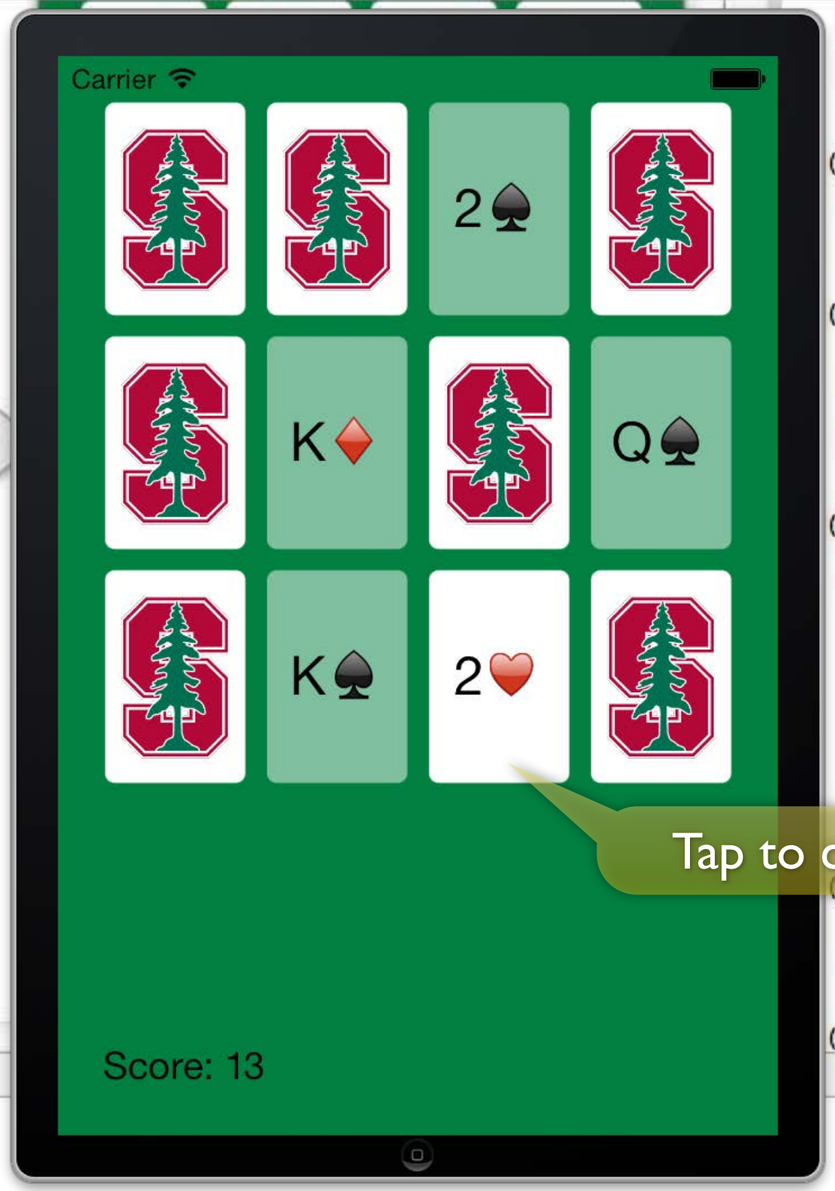
return card.isChosen ? card.contents : @"";


(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

Stanford CS193p
Fall 2013



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 : @"";

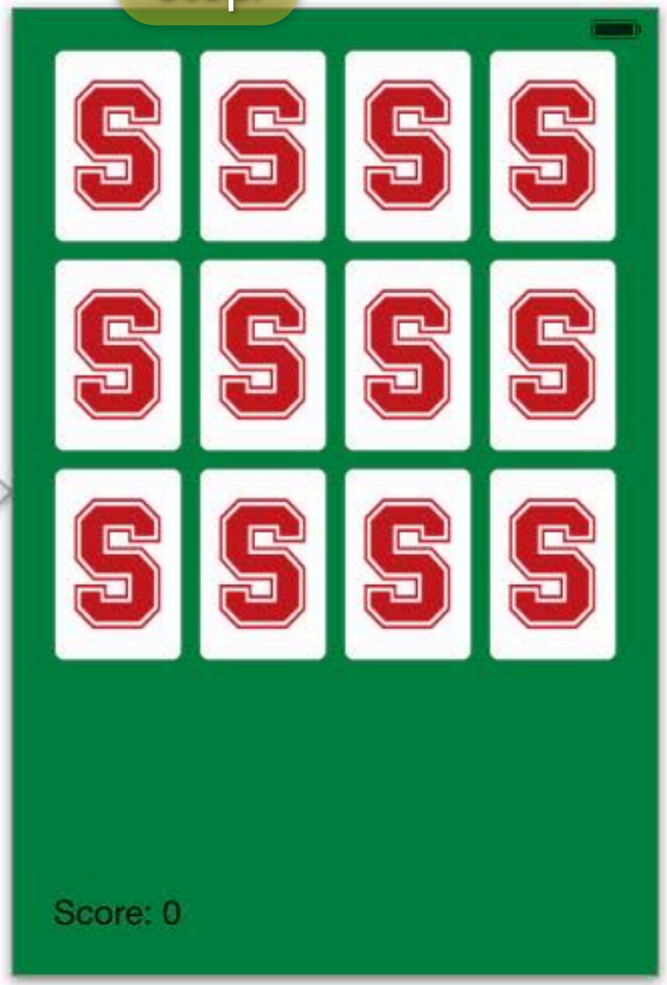
(UIImage *)backgroundImageForCard:(Card *)card
```

Stanford CS193p
Fall 2013

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 `alloc`ate and `init`ialize 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

Stanford CS193p

Developing Applications for iOS
Fall 2013-14



Today

- More Objective-C
  - Creating objects
  - `nil`
  - The very important type `id` (and the concept of "dynamic binding")
  - Introspection
  - Demo (improving `match:` with introspection)
- Foundation
  - `NSObject`, `NSArray`, `NSNumber`, `NSData`, `NSDictionary`, et. al.
  - Property Lists and `NSUserDefaults`
  - `NSRange`
  - `UIFont` and `UIColor` (not actually Foundation, they're in UIKit)
  - `NSAttributedString` (and its UIKit extensions)
  - Attributed strings in `UITextView` and `UILabel`

Creating Objects

- Most of the time, we create objects with alloc and init...

```
NSMutableArray *cards = [[NSMutableArray alloc] init];
CardMatchingGame *game = [[CardMatchingGame alloc] initWithCardCount:12 usingDeck:d];
```

- Or with class methods

```
NSString's + (id)stringWithFormat:(NSString *)format, ...
NSString *moltuae = [NSString stringWithFormat:@"%d", 42];
UIButton's + (id)buttonWithType:(UIButtonType)buttonType;
NSMutableArray's + (id)arrayWithCapacity:(int)count;
NSArray's + (id)arrayWithObject:(id)anObject;
```

- Sometimes both a class creator method and init method exist

`[NSString stringWithFormat:…]` same as `[[NSString alloc] initWithFormat:…]`

Don't be disturbed by this. Using either version is fine.

iOS seems to be moving more toward the alloc/init versions with new API, but is mostly neutral.

Creating Objects

- You can also ask other objects to create new objects for you

  NSString's `- (NSString *)stringByAppendingString:(NSString *)otherString;`
  NSArray's `- (NSString *)componentsJoinedByString:(NSString *)separator;`
  NSString's & NSArray's `- (id)mutableCopy;`

- But not all objects given out by other objects are newly created

  NSArray's `- (id)lastObject;`
  NSArray's `- (id)objectAtIndex:(int)index;`
  Unless the method has the word "copy" in it, if the object already exists, you get a pointer to it. If the object does not already exist (like the first 2 examples at the top), then you're creating.

nil

- Sending messages to `nil` is (mostly) okay. No code executed.

  If the method returns a value, it will return zero.

  ```
  int i = [obj methodWhichReturnsAnInt]; // i will be zero if obj is nil
  ```

  It is absolutely fine to depend on this and write code that uses this (don't get too cute, though).

  But be careful if the method returns a C struct. Return value is undefined.

  ```
  CGPoint p = [obj getLocation];  // p will have an undefined value if obj is nil
  ```

Dynamic Binding

- Objective-C has an important type called `id`
  - It means "pointer to an object of unknown/unspecified" type.
  - `id myObject;`
  - Really all object pointers (e.g. `NSString *`) are treated like `id` at runtime.
  - But at compile time, if you type something `NSString *` instead of `id`, the compiler can help you.
  - It can find bugs and suggest what methods would be appropriate to send to it, etc.
  - If you type something using `id`, the compiler can't help very much because it doesn't know much.
  - Figuring out the code to execute when a message is sent at runtime is called "dynamic binding."
- Is it safe?
  - Treating all object pointers as "pointer to unknown type" at runtime seems dangerous, right?
  - What stops you from sending a message to an object that it doesn't understand?
  - Nothing. And your program crashes if you do so. Oh my, Objective-C programs must crash a lot!
  - Not really.
  - Because we mostly use static typing (e.g. `NSString *`) and the compiler is really smart.

Dynamic Binding

- Static typing

```
NSString *s = @"x";   // "statically" typed (compiler will warn if s is sent non-NSString messges).
id obj = s;           // not statically typed, but perfectly legal; compiler can't catch [obj rank]
NSArray *a = obj;     // also legal, but obviously could lead to some big trouble!
```

Compiler will not complain about assignments between an `id` and a statically typed variable.

Sometimes you are silently doing this. You have already done so!

```
- (int)match:(NSArray *)otherCards
{
    …
    PlayingCard *otherCard = [otherCards firstObject]; // firstObject returns id!
    …
}
```

Never use "`id *`" by the way (that would mean "a pointer to a pointer to an object").

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];
```

No compiler warning.
Perfectly legal since `s` "isa" `Vehicle`.
Normal object-oriented stuff here.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
```

No compiler warning.
Perfectly legal since `s` "isa" `Vehicle`.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];
```

Compiler warning!

Would not crash at runtime though.
But only because we know `v` is a `Ship`.
Compiler only knows `v` is a `Vehicle`.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
```

No compiler warning.

The compiler knows that the method `shoot` exists,
so it's not impossible that `obj` might respond to it.
But we have not typed `obj` enough for the compiler to be sure it's wrong.
So no warning.
Might crash at runtime if `obj` is not a `Ship`
(or an object of some other class that implements a `shoot` method).

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
[obj someMethodNameThatNoObjectAnywhereRespondsTo];
```

Compiler warning!

Compiler has never heard of this method.
Therefore it's pretty sure `obj` will not respond to it.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
[obj someMethodNameThatNoObjectAnywhereRespondsTo];

NSString *hello = @"hello";
[hello shoot];
```

Compiler warning.
The compiler knows that NSString objects do not respond to shoot.
Guaranteed crash at runtime.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
[obj someMethodNameThatNoObjectAnywhereRespondsTo];

NSString *hello = @"hello";
[hello shoot];
Ship *helloShip = (Ship *)hello;
```

No compiler warning.
We are "casting" here.
The compiler thinks we know what we're doing.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
[obj someMethodNameThatNoObjectAnywhereRespondsTo];

NSString *hello = @"hello";
[hello shoot];
Ship *helloShip = (Ship *)hello;
[helloShip shoot];
```

No compiler warning!

We've forced the compiler to think that the `NSString` is a `Ship`. "All's well," the compiler thinks. Guaranteed crash at runtime.

Object Typing

```
@interface Vehicle
- (void)move;
@end
@interface Ship : Vehicle
- (void)shoot;
@end

Ship *s = [[Ship alloc] init];
[s shoot];
[s move];

Vehicle *v = s;
[v shoot];

id obj = ...;
[obj shoot];
[obj someMethodNameThatNoObjectAnywhereRespondsTo];

NSString *hello = @"hello";
[hello shoot];
Ship *helloShip = (Ship *)hello;
[helloShip shoot];
[(id)hello shoot];
```

No compiler warning!

We've forced the compiler to ignore the object type by "casting" in line. "All's well," the compiler thinks. Guaranteed crash at runtime.

Dynamic Binding

- So when would we ever intentionally use this dangerous thing!
  When we want to mix objects of different classes in a collection (e.g. in an NSArray).
  When we want to support the "blind, structured" communication in MVC (i.e. delegation).
  And there are other generic or blind communication needs.
  But to make these things safer, we're going to use two things: Introspection and Protocols.
- Introspection
  Asking at runtime what class an object is or what messages can be sent to it.
- Protocols
  A syntax that is "in between" `id` and static typing.
  Does not specify the class of an object pointed to, but does specify what methods it implements.
  Example ...

```
id <UIScrollViewDelegate> scrollViewDelegate;
```

  We'll cover how to declare and use protocols next week.

Introspection

- All objects that inherit from `NSObject` know these methods …

  `isKindOfClass:` returns whether an object is that kind of class (inheritance included)

  `isMemberOfClass:` returns whether an object is that kind of class (no inheritance)

  `respondsToSelector:` returns whether an object responds to a given method

  It calculates the answer to these questions at runtime (i.e. at the instant you send them).

- Arguments to these methods are a little tricky

  Class testing methods take a `Class`

  You get a `Class` by sending the class method `class` to a class (not the instance method `class`).

```
if ([obj isKindOfClass:[NSString class]]) {
    NSString *s = [(NSString *)obj stringByAppendingString:@"xyzzy"];
}
```

Introspection

- Method testing methods take a selector (SEL)

  Special `@selector()` directive turns the name of a method into a selector

```
if ([obj respondsToSelector:@selector(shoot)]) {
    [obj shoot];
} else if ([obj respondsToSelector:@selector(shootAt:)]) {
    [obj shootAt:target];
}
```

- SEL is the Objective-C "type" for a selector

```
SEL shootSelector = @selector(shoot);
SEL shootAtSelector = @selector(shootAt:);
SEL moveToSelector = @selector(moveTo:withPenColor:);
```

Introspection

- If you have a SEL, you can also ask an object to perform it …

Using the performSelector: or performSelector:withObject: methods in NSObject

```
[obj performSelector:shootSelector];
[obj performSelector:shootAtSelector withObject:coordinate];
```

Using makeObjectsPerformSelector: methods in NSArray

```
[array makeObjectsPerformSelector:shootSelector]; // cool, huh?
[array makeObjectsPerformSelector:shootAtSelector withObject:target]; // target is an id
```

In UIButton, - (void)addTarget:(id)anObject action:(SEL)action …;

```
[button addTarget:self action:@selector(digitPressed:) …];
```

Demo

- How `match:` might be improved with introspection

Foundation Framework

- NSObject

  Base class for pretty much every object in the iOS SDK

  Implements introspection methods discussed earlier.

  `- (NSString *)description` is a useful method to override (it's `%@` in `NSLog()`).
  Example … `NSLog(@"array contents are %@", myArray);`
  The `%@` is replaced with the results of invoking `[myArray description]`.

  Copying objects. This is an important concept to understand (why mutable vs. immutable?).

```
- (id)copy;          // not all objects implement mechanism (raises exception if not)
- (id)mutableCopy;   // not all objects implement mechanism (raises exception if not)
```

  It's not uncommon to have an array or dictionary and make a `mutableCopy` and modify that.
  Or to have a mutable array or dictionary and `copy` it to "freeze it" and make it immutable.
  Making copies of collection classes is very efficient, so don't sweat doing so.

Foundation Framework

- NSArray

Ordered collection of objects.
Immutable. That's right, once you create the array, you cannot add or remove objects.
All objects in the array are held onto strongly.
Usually created by manipulating other arrays or with @[].

You already know these key methods …

```
- (NSUInteger)count;
- (id)objectAtIndex:(NSUInteger)index; // crashes if index is out of bounds; returns id!
- (id)lastObject;  // returns nil (doesn't crash) if there are no objects in the array
- (id)firstObject;  // returns nil (doesn't crash) if there are no objects in the array
```

But there are a lot of very interesting methods in this class. Examples …

```
- (NSArray *)sortedArrayUsingSelector:(SEL)aSelector;
- (void)makeObjectsPerformSelector:(SEL)aSelector withObject:(id)selectorArgument;
- (NSString *)componentsJoinedByString:(NSString *)separator;
```

Foundation Framework

- NSMutableArray

  Mutable version of NSArray.

  Create with alloc/init or …

```
+ (id)arrayWithCapacity:(NSUInteger)numItems; // numItems is a performance hint only
+ (id)array;  // [NSMutableArray array] is just like [[NSMutableArray alloc] init]
```

  NSMutableArray inherits all of NSArray's methods.

  Not just count, objectAtIndex:, etc., but also the more interesting ones mentioned last slide.

  And you know that it implements these key methods as well …

```
- (void)addObject:(id)object;  // to the end of the array (note id is the type!)
- (void)insertObject:(id)object atIndex:(NSUInteger)index;
- (void)removeObjectAtIndex:(NSUInteger)index;
```

Enumeration

- Looping through members of an array in an efficient manner

  Language support using `for-in`.

  Example: `NSArray` of `NSString` objects

```
NSArray *myArray = ...;
for (NSString *string in myArray) { // no way for compiler to know what myArray contains
    double value = [string doubleValue]; // crash here if string is not an NSString
}
```

  Example: `NSArray` of `id`

```
NSArray *myArray = ...;
for (id obj in myArray) {
   // do something with obj, but make sure you don't send it a message it does not respond to
   if ([obj isKindOfClass:[NSString class]]) {
      // send NSString messages to obj with no worries
   }
}
```

Foundation Framework

- NSNumber

  Object wrapper around primitive types like `int`, `float`, `double`, `BOOL`, `enum`s, etc.

```
NSNumber *n = [NSNumber numberWithInt:36];
float f = [n floatValue]; // would return 36.0 as a float (i.e. will convert types)
```

  Useful when you want to put these primitive types in a collection (e.g. `NSArray` or `NSDictionary`).

  New syntax for creating an `NSNumber` in iOS 6: `@()`

```
NSNumber *three = @3;
NSNumber *underline = @(NSUnderlineStyleSingle); // enum
NSNumber *match = @([card match:@[otherCard]]); // expression that returns a primitive type
```

- NSValue

  Generic object wrapper for some non-object, non-primitive data types (i.e. C structs).

  e.g. `NSValue *edgeInsetsObject = [NSValue valueWithUIEdgeInsets:UIEdgeInsetsMake(1,1,1,1)]`

  Probably don't need this in this course (maybe when we start using points, sizes and rects).

Foundation Framework

- `NSData`

  "Bag of bits." Used to save/restore/transmit raw data throughout the iOS SDK.

- `NSDate`

  Used to find out the time right now or to store past or future times/dates.
  See also `NSCalendar`, `NSDateFormatter`, `NSDateComponents`.
  If you are going to display a date in your UI, make sure you study this in detail (localization).

- `NSSet` / `NSMutableSet`

  Like an array, but no ordering (no `objectAtIndex:` method).
  `member:` is an important method (returns an object if there is one in the set `isEqual:` to it).
  Can union and intersect other sets.

- `NSOrderedSet` / `NSMutableOrderedSet`

  Sort of a cross between `NSArray` and `NSSet`.
  Objects in an ordered set are distinct. You can't put the same object in multiple times like array.

Foundation Framework

- NSDictionary

Immutable collection of objects looked up by a key (simple hash table).
All keys and values are held onto strongly by an NSDictionary.

Can create with this syntax: `@{ key1 : value1, key2 : value2, key3 : value3 }`

```
NSDictionary *colors = @{ @“green” : [UIColor greenColor],
                          @“blue” : [UIColor blueColor],
                          @“red” : [UIColor redColor] };
```

Lookup using “array like” notation …

```
NSString *colorString = …;
UIColor *colorObject = colors[colorString]; // works the same as objectForKey: below
```

```
- (NSUInteger)count;
- (id)objectForKey:(id)key; // key must be copyable and implement isEqual: properly
```

NSStrings make good keys because of this.
See NSCopying protocol for more about what it takes to be a key.

Foundation Framework

- NSMutableDictionary

  Mutable version of NSDictionary.

  Create using alloc/init or one of the + (id)dictionary… class methods.

  In addition to all the methods inherited from NSDictionary, here are some important methods …

```
- (void)setObject:(id)anObject forKey:(id)key;
- (void)removeObjectForKey:(id)key;
- (void)removeAllObjects;
- (void)addEntriesFromDictionary:(NSDictionary *)otherDictionary;
```

Enumeration

- Looping through the keys or values of a dictionary

  Example:

```
NSDictionary *myDictionary = ...;
for (id key in myDictionary) {
    // do something with key here
    id value = [myDictionary objectForKey:key];
    // do something with value here
}
```

Property List

- The term "Property List" just means a collection of collections
  It's just a phrase (not a language thing). It means any graph of objects containing only:
  `NSArray`, `NSDictionary`, `NSNumber`, `NSString`, `NSDate`, `NSData` (or mutable subclasses thereof)
- An `NSArray` is a Property List if all its members are too
  So an `NSArray` of `NSString` is a Property List.
  So is an `NSArray` of `NSArray` as long as those `NSArray`'s members are Property Lists.
- An `NSDictionary` is one only if all keys and values are too
  An `NSArray` of `NSDictionary`s whose keys are `NSString`s and values are `NSNumber`s is one.
- Why define this term?
  Because the SDK has a number of methods which operate on Property Lists.
  Usually to read them from somewhere or write them out to somewhere. Example:

```
- (void)writeToFile:(NSString *)path atomically:(BOOL)atom;
```

  This can (only) be sent to an NSArray or NSDictionary that contains only Property List objects.

Other Foundation

- NSUserDefaults

  Lightweight storage of Property Lists.

  It's basically an NSDictionary that persists between launches of your application.

  Not a full-on database, so only store small things like user preferences.

  Read and write via a shared instance obtained via class method standardUserDefaults …

```
[[NSUserDefaults standardUserDefaults] setArray:rvArray forKey:@"RecentlyViewed"];
```

  Sample methods:

```
- (void)setDouble:(double)aDouble forKey:(NSString *)key;
- (NSInteger)integerForKey:(NSString *)key;  // NSInteger is a typedef to 32 or 64 bit int
- (void)setObject:(id)obj forKey:(NSString *)key; // obj must be a Property List
- (NSArray *)arrayForKey:(NSString *)key; // will return nil if value for key is not NSArray
```

  Always remember to write the defaults out after each batch of changes!

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Other Foundation

- NSRange

C struct (not a class)

Used to specify subranges inside strings and arrays (et. al.).

```
typedef struct {
    NSUInteger location;
    NSUInteger length;
} NSRange;
```

Important `location` value `NSNotFound`.

```
NSString *greeting = @“hello world”;  NSString *hi = @“hi”;
NSRange r = [greeting rangeOfString:hi];  // finds range of hi characters inside greeting
if (r.location == NSNotFound) { /* couldn’t find hi inside greeting */ }
```

`NSRangePointer` (just an `NSRange *` … used as an out method parameter).

There are C functions like `NSEqualRanges()`, `NSMakeRange()`, etc.

Colors

- `UIColor`

  An object representing a color.

  Initializers for creating a color based on RGB, HSB and even a pattern (`UIImage`).

  Colors can also have alpha (`UIColor *color = [otherColor colorWithAlphaComponent:0.3]`).

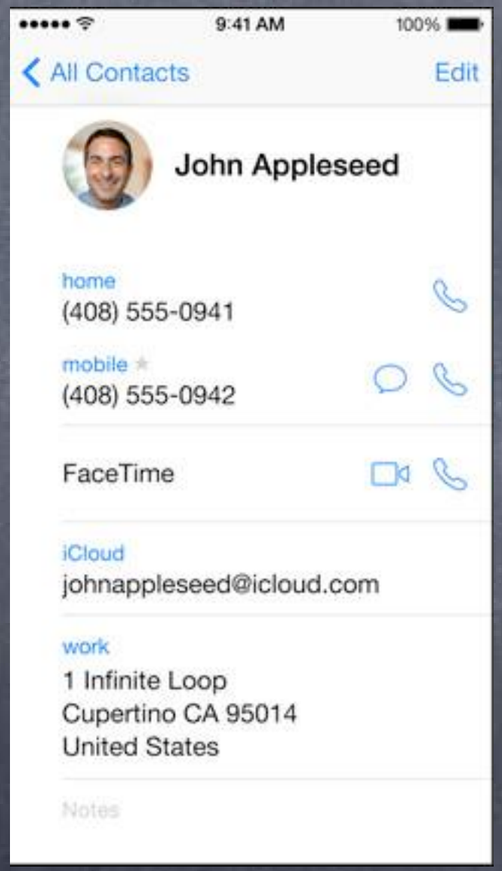
  A handful of "standard" colors have class methods (e.g. `[UIColor greenColor]`).
  A few "system" colors also have class methods (e.g. `[UIColor lightTextColor]`).

Fonts

- UIFont

  Fonts in iOS 7 are very important to get right.









Fonts

- UIFont

  It is best to get a UIFont by asking for the preferred font for a given text style …

  ```
  UIFont *font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
  ```

  Some other styles (see UIFontDescriptor documentation for even more styles) …

  UIFontTextStyleHeadline, UIFontTextStyleCaption1, UIFontTextStyleFootnote, etc.

  There are also "system" fonts.

  They are used in places like button titles.

  ```
  + (UIFont *)systemFontOfSize:(CGFloat)pointSize;
  + (UIFont *)boldSystemFontOfSize:(CGFloat)pointSize;
  ```

  You should never uses these for your user's content.

  Use preferredFontForTextStyle: for that.

Fonts

- `UIFontDescriptor`

  Fonts are designed by artists.
  They aren't always designed to fit any sort of categorization.
  Some fonts have Narrow or Bold or Condensed faces, some do not.
  Even "size" is sometimes a designed-in aspect of a particular font.

  A `UIFontDescriptor` attempts to categorize a font anyway.
  It does so by family, face, size, and other attributes.
  You can then ask for fonts that have those attributes and get a "best match."

  Understand that a best match for a "bold" font may not be bold if there's no such designed face.

Fonts

- UIFontDescriptor

  You can get a font descriptor from an existing UIFont with this UIFont method …

  ```
  - (UIFontDescriptor *)fontDescriptor;
  ```

  You might well have gotten the original UIFont using preferredFontForTextStyle:.

  Then you might modify it to create a new descriptor with methods in UIFontDescriptor like …

  ```
  - (UIFontDescriptor *)fontDescriptorByAddingAttributes:(NSDictionary *)attributes;
  ```

  (the attributes and their values can be found in the class reference page for UIFontDescriptor)

  You can also create a UIFontDescriptor directly from attributes (though this is rare) using …

  ```
  + (UIFontDescriptor *)fontDescriptorWithFontAttributes:(NSDictionary *)attributes;
  ```

- Symbolic Traits

  Italic, Bold, Condensed, etc., are important enough to get their own API in UIFontDescriptor …

  ```
  - (UIFontDescriptorSymbolicTraits)symbolicTraits;
  - (UIFontDescriptor *)fontDescriptorWithSymbolicTraits:(UIFontDescriptorSymbolicTraits)traits;
  ```

  Some example traits (again, see UIFontDescriptor documentation for more) …

  UIFontDescriptorTraitItalic, UIFontDescriptorTraitBold, UIFontDescriptorTraitCondensed, etc.

Fonts

- UIFontDescriptor

  Once you have a UIFontDescriptor that describes the font you want, use this UIFont method:

  ```
  + (UIFont *)fontWithDescriptor:(UIFontDescriptor *)descriptor size:(CGFloat)size;
  ```

  (specify size of 0 if you want to use whatever size is in the descriptor)

  You will get a "best match" for your descriptor given available fonts and their faces.

- Example

  Let's try to get a "bold body font" ...

  ```
  UIFont *bodyFont = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
  UIFontDescriptor *existingDescriptor = [bodyFont fontDescriptor];
  UIFontDescriptorSymbolicTraits traits = existingDescriptor.symbolicTraits;
  traits |= UIFontDescriptorTraitBold;
  UIFontDescriptor *newDescriptor = [existingDescriptor fontDescriptorWithSymbolicTraits:traits];
  UIFont *boldBodyFont = [UIFont fontWithFontDescriptor:newDescriptor size:0];
  ```

  This will do the best it can to give you a bold version of the UIFontTextStyleBody preferred font. It may or may not actually be bold.

Attributed Strings

- How text looks on screen

  The font has a lot to do with how text looks on screen.
  But there are other determiners (color, whether it is "outlined", stroke width, underlining, etc.).
  You put the text together with a font and these other determiners using `NSAttributedString`.

- `NSAttributedString`

  Think of it as an `NSString` where each character has an `NSDictionary` of "attributes".
  The attributes are things like the font, the color, underlining or not, etc., of the character.
  It is not, however, actually a subclass of `NSString` (more on this in a moment).

- Getting Attributes

  You can ask an `NSAttributedString` all about the attributes at a given location in the string.

```
- (NSDictionary *)attributesAtIndex:(NSUInteger)index
                     effectiveRange:(NSRangePointer)range;
```

  The `range` is returned and it lets you know for how many characters the attributes are identical.
  There are also methods to ask just about a certain attribute you might be interested in.
  `NSRangePointer` is essentially an `NSRange *`. It's okay to pass `NULL` if you don't care.

Attributed Strings

- NSAttributedString is not an NSString

  It does not inherit from NSString, so you cannot use NSString methods on it.
  If you need to operate on the characters, there is this great method in NSAttributedString …

```
- (NSString *)string;
```

  For example, to find a substring in an NSAttributedString, you could do this …

```
NSAttributedString *attributedString = …;
NSString *substring = …;
NSRange r = [[attributedString string] rangeOfString:substring];
```

  The method string is guaranteed to be high performance but is volatile.
  If you want to keep it around, make a copy of it.

Attributed Strings

- NSMutableAttributedString

  Unlike NSString, we almost always use mutable attributed strings.

- Adding or setting attributes on the characters

  You can add an attribute to a range of characters …

  ```
  - (void)addAttributes:(NSDictionary *)attributes range:(NSRange)range;
  ```

  … which will change the values of attributes in attributes and not touch other attributes.

  Or you can set the attributes in a range …

  ```
  - (void)setAttributes:(NSDictionary *)attributes range:(NSRange)range;
  ```

  … which will remove all other attributes in that range in favor of the passed attributes.

  You can also remove a specific attribute from a range …

  ```
  - (void)removeAttribute:(NSString *)attributeName range:(NSRange)range;
  ```

- Modifying the contents of the string (changing the characters)

  You can do that with methods to append, insert, delete or replace characters.

  Or call the NSMutableAttributedString method - (NSMutableString *)mutableString
  and modify the returned NSMutableString (attributes will, incredibly, be preserved!).

Attributed Strings

So what kind of attributes are there?

```
@{ NSFontAttributeName :
     [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline] }
```



Attributed Strings

So what kind of attributes are there?

```
@{ NSFontAttributeName :
      [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
   NSForegroundColorAttributeName : [UIColor blueColor] }
```



Attributed Strings

So what kind of attributes are there?

```
@{ NSFontAttributeName :
      [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
   NSForegroundColorAttributeName : [UIColor greenColor] }
```

Be careful with colored text.
Color is one of the primary ways a user knows what's "clickable."
Be Consistent.



Attributed Strings

So what kind of attributes are there?

```
@{ NSFontAttributeName :
      [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
  NSForegroundColorAttributeName : [UIColor greenColor],

  NSStrokeWidthAttributeName : @-5,

  NSStrokeColorAttributeName : [UIColor orangeColor] }
```

Negative number means "fill and stroke."
Positive number is stroke only (outline).



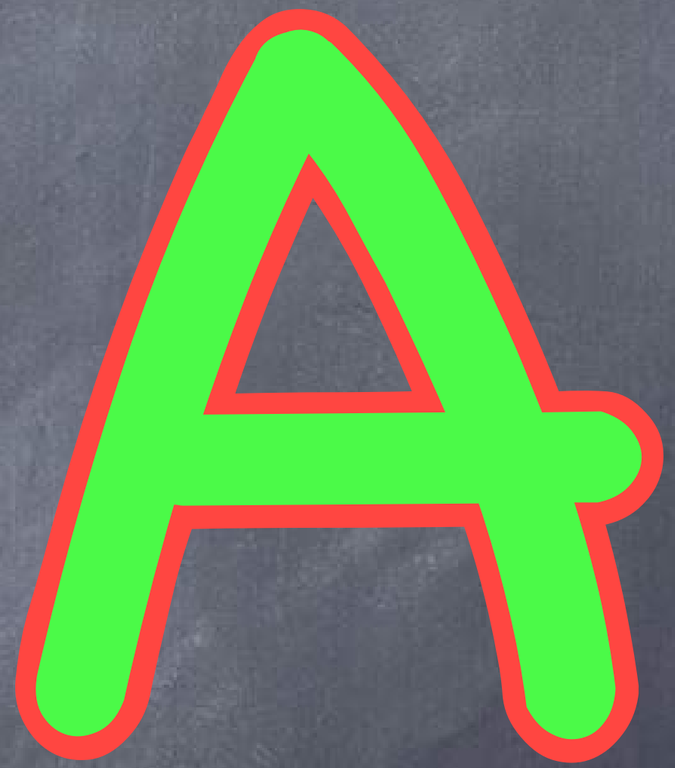
Attributed Strings

So what kind of attributes are there?

```
@{ NSFontAttributeName :
      [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
  NSForegroundColorAttributeName : [UIColor greenColor],

  NSStrokeWidthAttributeName : @-5,

  NSStrokeColorAttributeName : [UIColor redColor] }
```



Attributed Strings

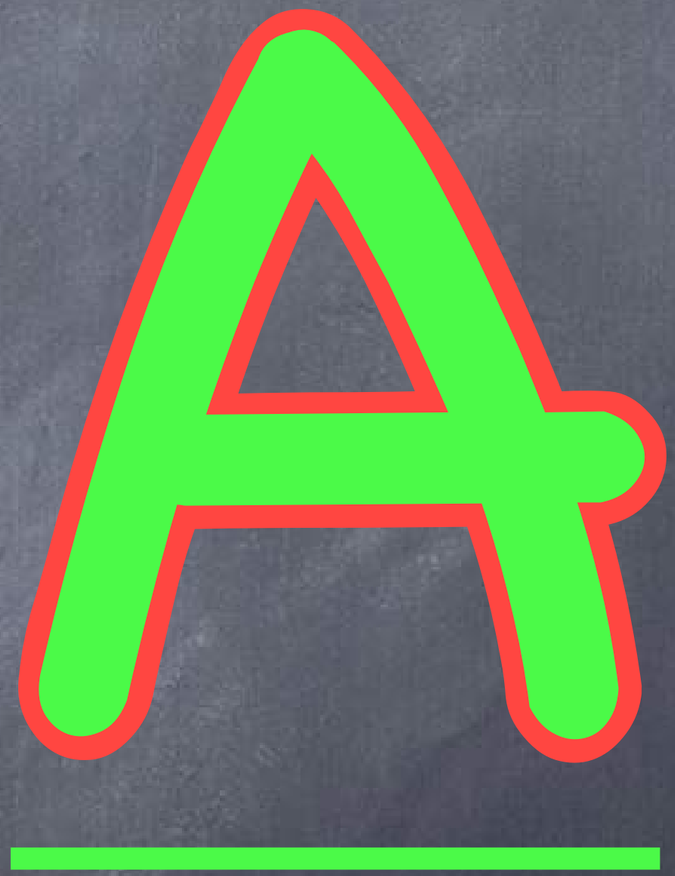
So what kind of attributes are there?

```
@{ NSFontAttributeName :
     [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
  NSForegroundColorAttributeName : [UIColor greenColor],

  NSStrokeWidthAttributeName : @-5,

  NSStrokeColorAttributeName : [UIColor redColor],

  NSUnderlineStyleAttributeName : @(NSUnderlineStyleSingle) }
```



Attributed Strings

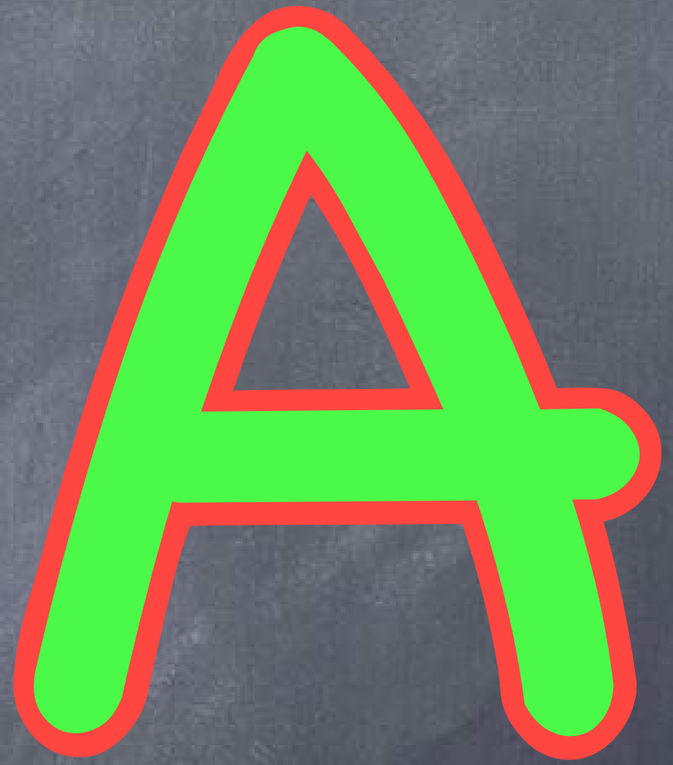
So what kind of attributes are there?

```
@{ NSFontAttributeName :
      [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
  NSForegroundColorAttributeName : [UIColor greenColor],

  NSStrokeWidthAttributeName : @-5,

  NSStrokeColorAttributeName : [UIColor redColor],

  NSUnderlineStyleAttributeName : @(NSUnderlineStyleNone) }
```



Attributed Strings

```
UIColor *yellow = [UIColor yellowColor];
UIColor *transparentYellow = [yellow colorWithAlphaComponent:0.3];
 @{ NSFontAttributeName :
       [UIFont preferredFontWithTextStyle:UIFontTextStyleHeadline]
   NSForegroundColorAttributeName : [UIColor greenColor],
   NSStrokeWidthAttributeName : @-5,
   NSStrokeColorAttributeName : [UIColor redColor],
   NSUnderlineStyleAttributeName : @(NSUnderlineStyleNone),
   NSBackgroundColorAttributeName : transparentYellow }
```

You could use transparent colors in other attributes as well.



NSAttributedString

- Where do attributed strings get used?

  UIButton's `- (void)setAttributedTitle:(NSAttributedString *)title forState:…;`

  UILabel's `@property (nonatomic, strong) NSAttributedString *attributedText;`

  UITextView's `@property (nonatomic, readonly) NSTextStorage *textStorage;`

- UIButton

  Attributed strings on buttons will be extremely use for your homework.

- Drawing strings directly

  Next week we'll see how to draw things directly on screen.

  NSAttributedStrings know how to draw themselves on screen, for example …

  `- (void)drawInRect:(CGRect)aRect;`

  Don't worry about this too much for now. Wait until next week.

UILabel

- UILabel
  You have been setting its contents using the NSString property text.
  But it also has a property to set/get its text using an NSAttributedString …
  ```
  @property (nonatomic, strong) NSAttributedString *attributedText;
  ```
- Note that this attributed string is not mutable
  So, to modify what is in a UILabel, you must make a mutableCopy, modify it, then set it back.
  ```
  NSMutableAttributedString *labelText = [myLabel.attributedText mutableCopy];
  [labelText setAttributes:…];
  myLabel.attributedText = labelText;
  ```
- Don't need this very often
  There are properties in UILabel like font, textColor, etc., for setting look of all characters.
  The attributed string in UILabel would be used mostly for "specialty labels".

Next Time

- Friday Section
  - Location and Time
  - Debugging
  - Xcode Tips and Tricks
- Monday
  - `UITextView`
  - `NSNotification` (radio station)
  - Demo of Attributed Strings, etc.
  - View Controller Lifecycle

Stanford CS193p

Developing Applications for iOS
Fall 2013-14



Today

- UITextView

  Scrollable, editable/selectable view of a mutable attributed string.

- View Controller Lifecycle

  Finding out what's happening as a VC is created, hooked up to the View, appears/disappears, etc.

- NSNotification

  The "radio station" communication mechanism of MVC.

  Today we'll just talk about "tuning in."

- Demo

  UITextView

  NSAttributedString, NSMutableAttributedString, NSTextStorage

  UIFont, UIColor

  NSNotification

  View Controller Lifecycle

UITextView

- UITextView
  Like UILabel, but multi-line, selectable/editable, scrollable, etc.
- Set its text and attributes via its NSMutableAttributedString
  Obtain the NSMutableAttributedString representing the text in the UITextView using …
  `@property (nonatomic, readonly) NSTextStorage *textStorage;`
  NSTextStorage is a subclass of NSMutableAttributedString.
  You can simply modify it and the UITextView will automatically update. New in iOS 7.
- Setting the font
  Fonts can, of course, vary from character to character in UITextView.
  But there is also a property that applies a font to the entire UITextView …
  `@property (nonatomic, strong) UIFont *font;`

  Changing the font of every character will not reset other attributes (e.g. color, underline, etc.).
  However, you will lose your symbolic traits!

UITextView

- Advanced layout in `UITextView` with TextKit

  This property defines "where text can be" in the `UITextView` …

```
@property (readonly) NSTextContainer *textContainer;
```

  This object will read characters from `textStorage` and lays down glyphs into `textContainer` …

```
@property (readonly) NSLayoutManager *layoutManager;
```

  These objects are quite powerful.

  For example, `textContainer` can have "exclusion zones" specified in it (flowing around images).

  Check them out if you're interested in typography.

Demo

- Attributor

  `UITextView`

  `NSAttributedString`, `NSMutableAttributedString`, `NSTextStorage`

  `UIFont`, `UIColor`

View Controller Lifecycle

- View Controllers have a "Lifecycle"
  A sequence of messages is sent to them as they progress through it.
- Why does this matter?
  You very commonly override these methods to do certain work.
- The start of the lifecycle ...
  Creation.
  MVCs are most often instantiated out of a storyboard (as you've seen).
  There are ways to do it in code (rare) as well which we may cover later in the quarter.
- What then?
  Outlet setting.
  Appearing and disappearing.
  Geometry changes.
  Low-memory situations.
  At each stage, iOS invokes method(s) on the Controller ...

View Controller Lifecycle

- After instantiation and outlet-setting, `viewDidLoad` is called

  This is an exceptionally good place to put a lot of setup code.

```
- (void)viewDidLoad
{
    [super viewDidLoad];          // always let super have a chance in lifecycle methods
    // do some setup of my MVC
}
```

  But be careful because the geometry of your view (its `bounds`) is not set yet!
  At this point, you can't be sure you're on an iPhone 5-sized screen or an iPad or ???.
  So do not initialize things that are geometry-dependent here.

View Controller Lifecycle

- Just before the view appears on screen, you get notified
  (argument is just whether you are appearing instantly or over time via animation)
  `- (void)viewWillAppear:(BOOL)animated;`

  Your view will only get "loaded" once, but it might appear and disappear a lot.
  So don't put something in this method that really wants to be in `viewDidLoad`.
  Otherwise, you might be doing something over and over unnecessarily.

  Do something here if things you display are changing while your MVC is off-screen.

  Later we will use this to optimize performance by waiting until this method
  (as opposed to `viewDidLoad`) to kick off an expensive operation (probably in another thread).

  View's geometry is set here, but there are other (better?) places to react to geometry.

View Controller Lifecycle

- And you get notified when you will disappear off screen too

  This is where you put "remember what's going on" and cleanup code.

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];  // call super in all the viewWill/Did... methods
    // let's be nice to the user and remember the scroll position they were at ...
    [self rememberScrollPosition];  // we'll have to implement this, of course
    // do some other clean up now that we've been removed from the screen
    [self saveDataToPermanentStore]; // maybe do in did instead?
    // but be careful not to do anything time-consuming here, or app will be sluggish
    // maybe even kick off a thread to do what needs doing here (again, we'll cover threads later)
}
```

- There are "did" versions of both of the appearance methods

```
- (void)viewDidAppear:(BOOL)animated;
- (void)viewDidDisappear:(BOOL)animated;
```

View Controller Lifecycle

- Geometry changed?

  Most of the time this will be automatically handled with Autolayout (next week).

  `- (void)view{Will,Did}LayoutSubviews;`

  Called any time a view's `frame` changed and its `subviews` were thus re-layed out.

  For example, autorotation.

  You can reset the `frames` of your `subviews` here or set other geometry-affecting properties.

  Between "`will`" and "`did`", autolayout will happen (we'll talk about that next week).

View Controller Lifecycle

- Autorotation

  When the device is rotated, the top level view controller will have its bounds reoriented iff …

  The view controller returns YES from `shouldAutorotate`.

  The view controller returns the new orientation in `supportedInterfaceOrientations`.

  The application allows rotation to that orientation (defined in `Info.plist` file).

  Generally it is a good idea to try to support rotation in MVCs. We'll talk about that next week.

- Specific notification that rotation will/did happen

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)anOrientation
                                duration:(NSTimeInterval)seconds;
- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOriention)orient
                                duration:(NSTimeInterval)seconds;
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)anOrientation;
```

  This property will have the current orientation when each of the above is called …

```
@property UIInterfaceOrientation interfaceOrientation;
```

  It's pretty rare to implement these. Autolayout and `viewWillLayoutSubviews` usually suffice.

View Controller Lifecycle

- In low-memory situations, `didReceiveMemoryWarning` gets called …
  This rarely happens, but well-designed code with big-ticket memory uses might anticipate it.
  Examples: images and sounds.
  Anything "big" that can be recreated should probably be released (i.e. set `strong` pointer to `nil`).

View Controller Lifecycle

- awakeFromNib

  This method is sent to all objects that come out of a storyboard (including your Controller).
  Happens before outlets are set! (i.e. before the MVC is "loaded")
  Put code somewhere else if at all possible (e.g. viewDidLoad or viewWillAppear:).

  Anything that would go in your Controller's init method would have to go in awakeFromNib too
  (because init methods are not called on objects that come out of a storyboard).

```
- (void)setup { };  // do something which can't wait until viewDidLoad
- (void)awakeFromNib { [self setup]; }
// UIViewController's designated initializer is initWithNibName:bundle: (ugh!)
- (instancetype)initWithNibName:(NSString *)name bundle:(NSBundle *)bundle
{
    self = [super initWithNibName:name bundle:bundle];
    [self setup];
    return self;
}
```

  It is unlikely you will ever need to use awakeFromNib in this course.

View Controller Lifecycle

- Summary

Instantiated (from storyboard - many ways for this to happen which we'll cover later)
awakeFromNib
outlets get set
viewDidLoad
(when geometry is determined)
  viewWillLayoutSubviews: and viewDidLayoutSubviews:
(next group can happen repeatedly as your MVC appears and disappears from the screen ...)
  viewWillAppear: and viewDidAppear:
    (whenever geometry changes again while visible, e.g. device rotation)
    viewWillLayoutSubviews: and viewDidLayoutSubviews:
    (if it is autorotation, then you also get will/didRotateTo/From messages--rare to use these)
  viewWillDisappear: and viewDidDisappear:
(possibly if memory gets low ...)
  didReceiveMemoryWarning
(there is no "unload" anymore, so that's all there is)

Demo

- Attributor

  View Controller Lifecycle

  How (and when in the VCL) would we "outline" the text on the Outline button?

NSNotification

- Notifications

  The "radio station" from the MVC slides.

- NSNotificationCenter

  Get the default "notification center" via `[NSNotificationCenter defaultCenter]`

  Then send it the following message if you want to "listen to a radio station":

```
- (void)addObserver:(id)observer      // you (the object to get notified)
          selector:(SEL)methodToInvokeIfSomethingHappens
              name:(NSString *)name // name of station (a constant somewhere)
            object:(id)sender;      // whose changes you're interested in (nil is anyone's)
```

- You will then be notified when there are broadcasts

```
- (void)methodToInvokeIfSomethingHappens:(NSNotification *)notification
{
    notification.name     // the name passed above
    notification.object   // the object sending you the notification
    notification.userInfo // notification-specific information about what happened
}
```

NSNotification

- Be sure to "tune out" when done listening

```
[center removeObserver:self];
```

or

```
[center removeObserver:self name:UIContentSizeCategoryDidChangeNotification object:nil];
```

Failure to remove yourself can sometimes result in crashers.
This is because the NSNotificationCenter keeps an "unsafe retained" pointer to you.

A good place to remove yourself is when your MVC's View goes off screen.
We'll see how to find out about that later in this lecture.

Or you can remove yourself in a method called `dealloc` (called when you leave the heap).

```
- (void)dealloc
{
    // be careful in this method! can't access properties! you are almost gone from heap!
    [[NSNotificationCenter defaultCenter] removeObserver:self];
}
```

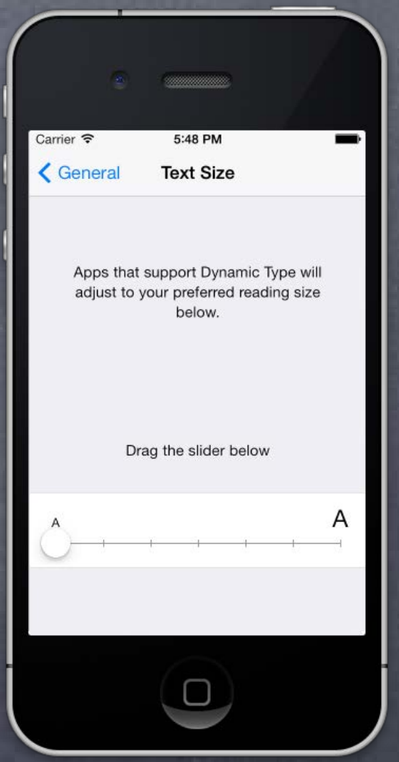
NSNotification

- Example

  Watching for changes in the size of preferred fonts (user can change this in Settings) …

```
NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
[center addObserver:self
           selector:@selector(preferredFontsSizeChanged:)
               name:UIContentSizeCategoryDidChangeNotification
             object:nil]; // this station's broadcasts aren't object-specific

- (void)preferredFontsSizeChanged:(NSNotification *)notification
{
    // re-set the fonts of objects using preferred fonts
}
```



Demo

- Attributor

  NSNotification

  More View Controller Lifecycle

Next Time

- Wednesday

  Multiple MVCs in one application

  `UITabBarController`

  `UINavigationController`

  Using inheritance with Controllers

  Assignment 3 Handed Out (due next Wednesday)

- Friday

  No Section