Stanford CS193p

Developing Applications for iOS
Winter 2013

Today

- Introduction to Objective-C (con't)
  continue showing Card Game Model with `Deck`, `PlayingCard`, `PlayingCardDeck`
- Xcode 4 Demonstration
  Start building the simple Card Game

Objective-C

Deck.h

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

@interface Deck : NSObject

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

- (Card *)drawRandomCard;

@end
```

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

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>

@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

- (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
```

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

- (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
```

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.

Deck.m

```
#import "Deck.h"

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

@implementation Deck

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

- (Card *)drawRandomCard
{
}

@end
```

So we put the @property declaration here in the @implementation file.

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()
@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];
    }
}

- (Card *)drawRandomCard
{
}

@end
```

Let's take a look at a sample implementation of the addCard:atTop: method.

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;

- (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];
    }
}

- (Card *)drawRandomCard
{
}

@end
```

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

All properties start out with a value of zero.
For a pointer, zero is called `nil` and means "this property does not yet point to anything".

Sending a message (like `addObject:`) to a `nil` pointer does nothing. It does not crash, but does not invoke the method either.

Objective-C

Deck.h

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

@interface Deck : NSObject

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

- (Card *)drawRandomCard;

@end
```

Let's write our own getter to create the cards array on the fly. This is called "lazy instantiation". Now you can start to see the value of properties.

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];
    }
}

- (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
```

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];
    }
}

- (Card *)drawRandomCard
{
}



@end
```

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

Now the cards property will always at least be an empty mutable array, so this code will always work.

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()
@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 { ... }

- (Card *)drawRandomCard
{









}

@end
```

When you collapse code in the source code editor, you will see this same icon.

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()
@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
{
}

- (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
```

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

But 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.

This code will remove a random card from 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;

- (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 { ... }

- (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

```
#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
```

This is really all we need to make a Playing Card specific subclass of `Card`: the suit, the rank, and an override of the contents getter to combine them.

However, 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];
}



@end
```

Notice the @[ ] notation to create an array.

This is much better!

You can get an element out of an NSArray using this "array like" notation.

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];
}
```

Note that a Playing Card with a rank of zero (the default for a property) will show up as ? in the Card's contents.

Let's use the suit property's getter to make a "zero" (i.e. nil) suit also show up as a ?. So a PlayingCard, until someone sets its rank and suit, has contents of ??.

```
- (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 use the `suit` property's setter to make sure no one sets the card's suit to something invalid.

Notice that we can embed the array creation right inside 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
```

Normally this @synthesize is automatically created for you by the compiler, but if you implement BOTH the setter AND the getter yourself, then you also have to do the @synthesize yourself too (luckily it's quite easy).

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;


+ (NSArray *)validSuits;

@end
```

Notice the + instead of –

We've created a class method here.
This method can be invoked directly on the class (i.e. not sent to an instance).
Of course, that also means it can't use any @propertys. validSuits does not.

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
```

This is how a class method is invoked.
See how the name of the class appears in the place you'd normally see a pointer to an object?

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

Let's make our new class method 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
```

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
```

We're just collapsing these to make more space.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

+ (NSArray *)validSuits;


@end
```

Here's another class method.

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
```

Class method invocation.

Objective-C

PlayingCard.h

```
#import "Card.h"

@interface PlayingCard : Card

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

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

@end
```

Yet another class method.

We'll make this one public too.

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
```

Notice that we send rankStrings to self. Since maxRank is a class method, it can use self to invoke other class methods.

We can access a property immediately on the results of another message send. [self rankStrings] returns an array and we immediately access its count property.

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.

Yet another class method invocation.

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
```

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck









@end
```

A PlayingCardDeck is simply a subclass of Deck whose array of Cards contains all 52 PlayingCard combinations (i.e. it's a deck of playing cards).

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

- (id)init
{
}


@end
```

We only need to implement one method which is the PlayingCardDeck's initializer.
This method is always called when a PlayingCardDeck is created.
For example, a PlayingCardDeck might be created like this:
`Deck *myDeck = [[PlayingCardDeck alloc] init]`

The return type id here means "object of any (unknown) class."
We'll talk about this much more next week.
All initializers return this type.

Classes can have more complicated initializers than just plain "init".
We'll talk more about that next week as well.
PlayingCardDeck inherits its initializer from Deck which inherits it from NSObject.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

```
#import "PlayingCardDeck.h"


@implementation PlayingCardDeck

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

    if (self) {



    }

    return self;
}



@end
```

Sending a message to `super` is just like sending to `self` except that we start with our superclass's implementation first (i.e. not our own).

Yikes! We are assigning a value to `self`! This (in an initializer) is the ONLY time we ever do that. It's very weird.

The first thing we do in an initializer (we are allowed to have multiple initializers, each with different arguments) is call our class's "designated initializer" (unless the method we are implementing IS our class's designated initializer, in which case we call our superclass's "designated initializer"). This is `PlayingCardDeck`'s only (and thus its "designated") initializer, so we must call our superclass's (`Deck`'s and thus `NSObject`'s) designated initializer.

We check to see if our designated initializer returned `nil` because, if it does, this object cannot be initialized and we should return `nil` to indicate that.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

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

@implementation PlayingCardDeck

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

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

    return self;
}

@end
```

We invoke PlayingCard methods, so we must include its header file.

We're calling public class methods here!

Here's that for-in "fast enumeration" syntax again.

The implementation is quite simple.
We just go through every rank and suit and create a card and add it to ourself (we're a Deck, remember).
Here're the nested loops we need to do that.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

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

@implementation PlayingCardDeck

- (id)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 atTop:YES];
            }
        }
    }

    return self;
}

@end
```

Simply setting the `rank` and `suit` ...

... then adding the card to ourself.

`addCard:atTop:` is a `Deck` method we implemented earlier (in case you've forgotten already!).

Notice that we call `init` on `PlayingCard` here. Neither `PlayingCard` nor its superclass `Card` implements `init`, but they inherit one from NSObject.

Objective-C

PlayingCardDeck.h

```
#import "Deck.h"

@interface PlayingCardDeck : Deck

@end
```

PlayingCardDeck.m

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

@implementation PlayingCardDeck

- (id)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 atTop:YES];
            }
        }
    }

    return self;
}









@end
```

Demonstration

- Card Matching Game

  Today (for starters): a simple UI with one card showing the A♣

  Your homework will be to flip through other cards (besides the A♣).

  What to watch for ...
  - Creating a new project!
  - Building out an MVC's View.
  - Running an application in the iOS 6 simulator.
  - Editing an MVC's Controller.
  - Accessing the Documentation from Xcode.
  - Target/Action from View to Controller
  - Connecting an Outlet from Controller to View
  - Inspecting connections between Controller and View.
  - Adding a class (the Model classes we just worked on) to your project.

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."

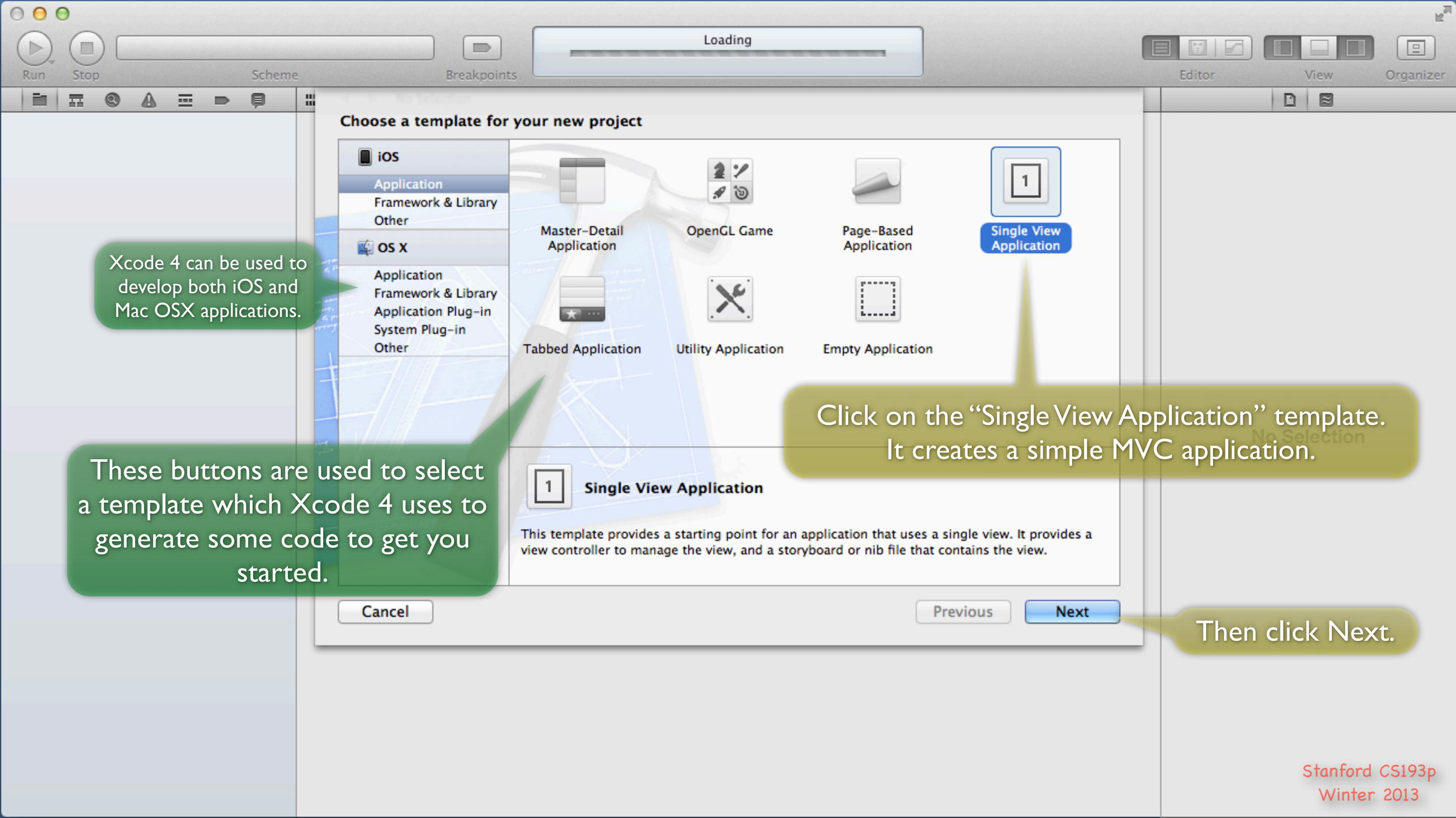
Red Bubbles
mean "important!"

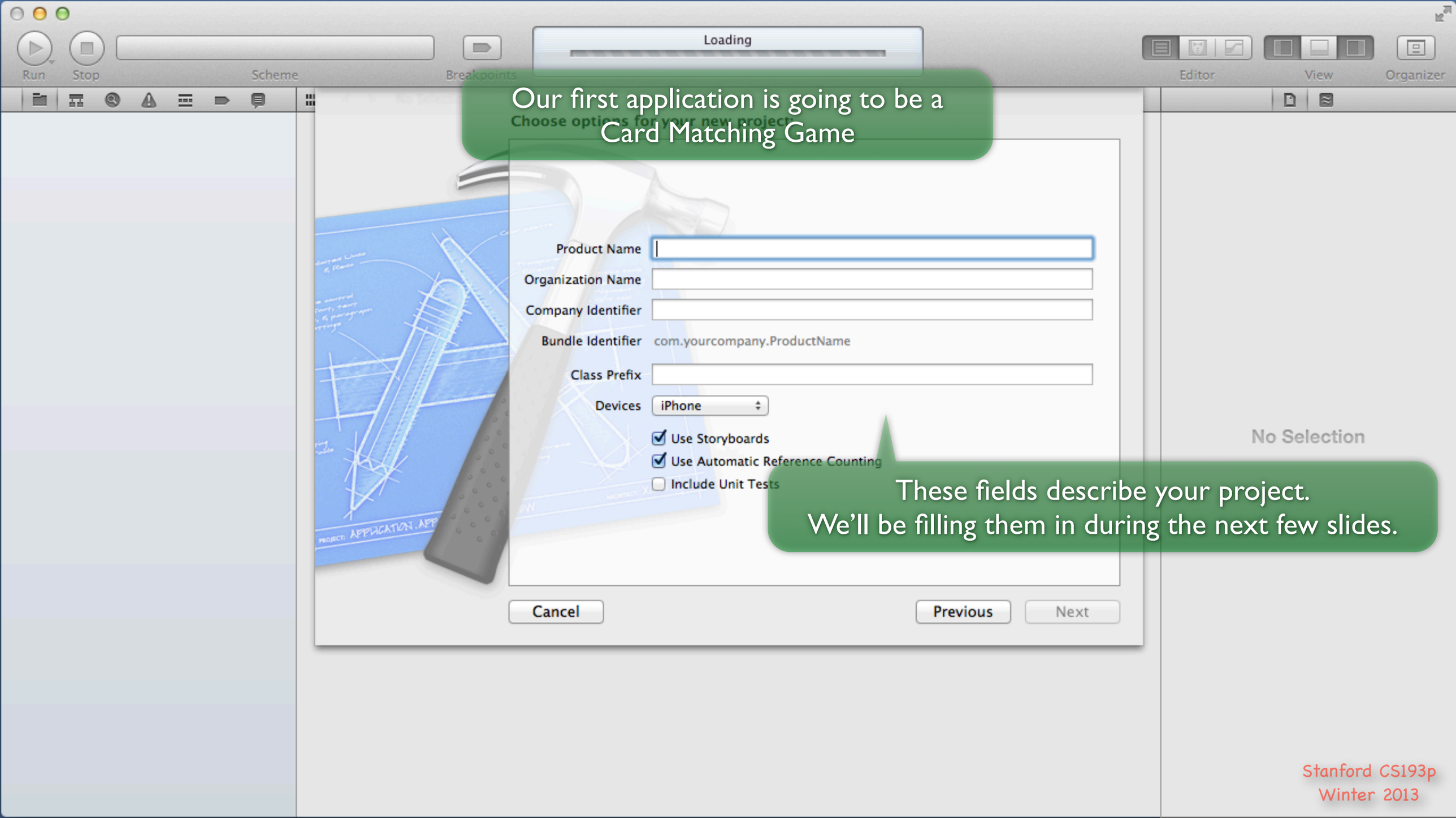
Green Bubbles
with small text is for
"minor notes."

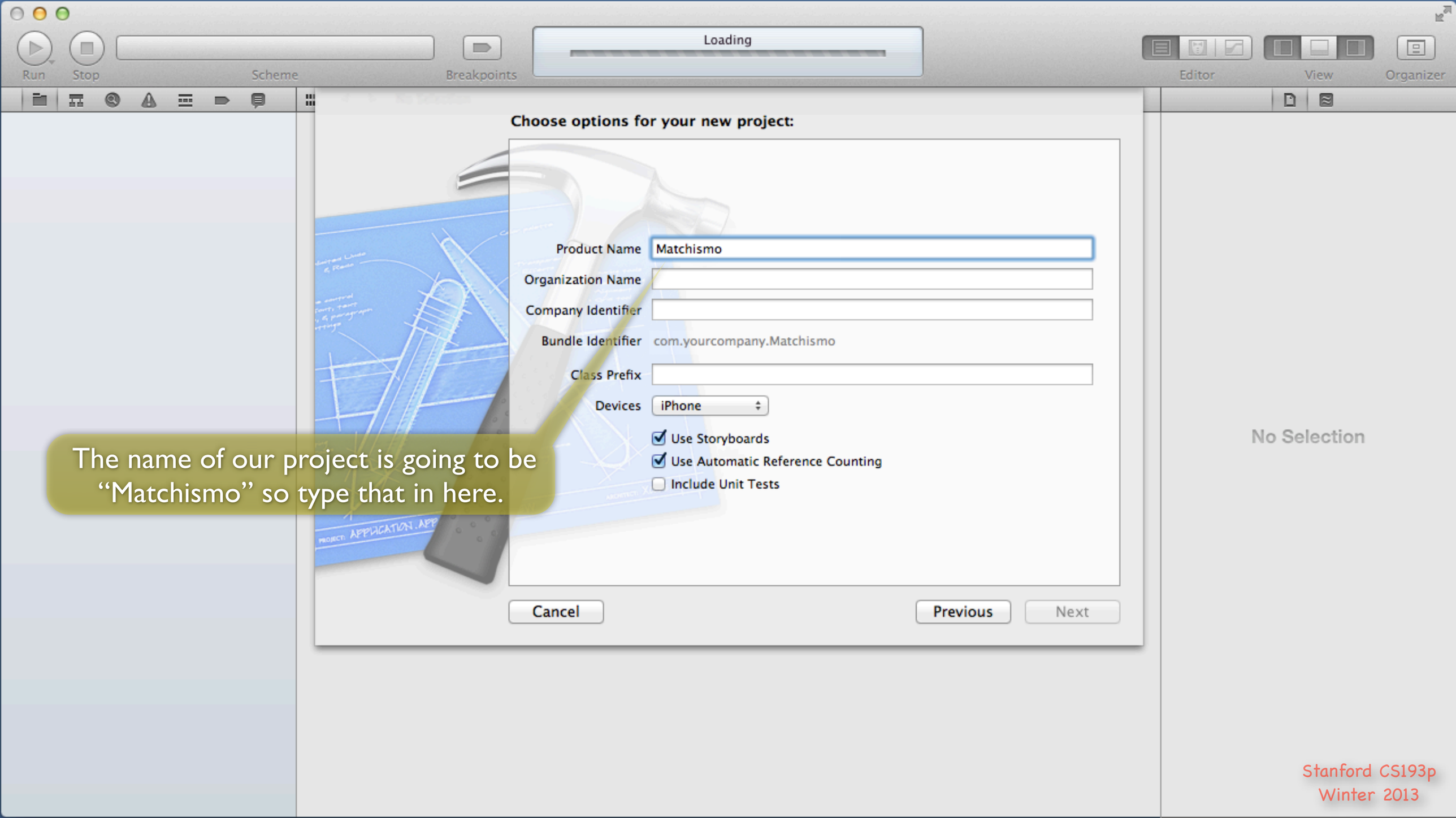
Xcode 4 Splash Screen

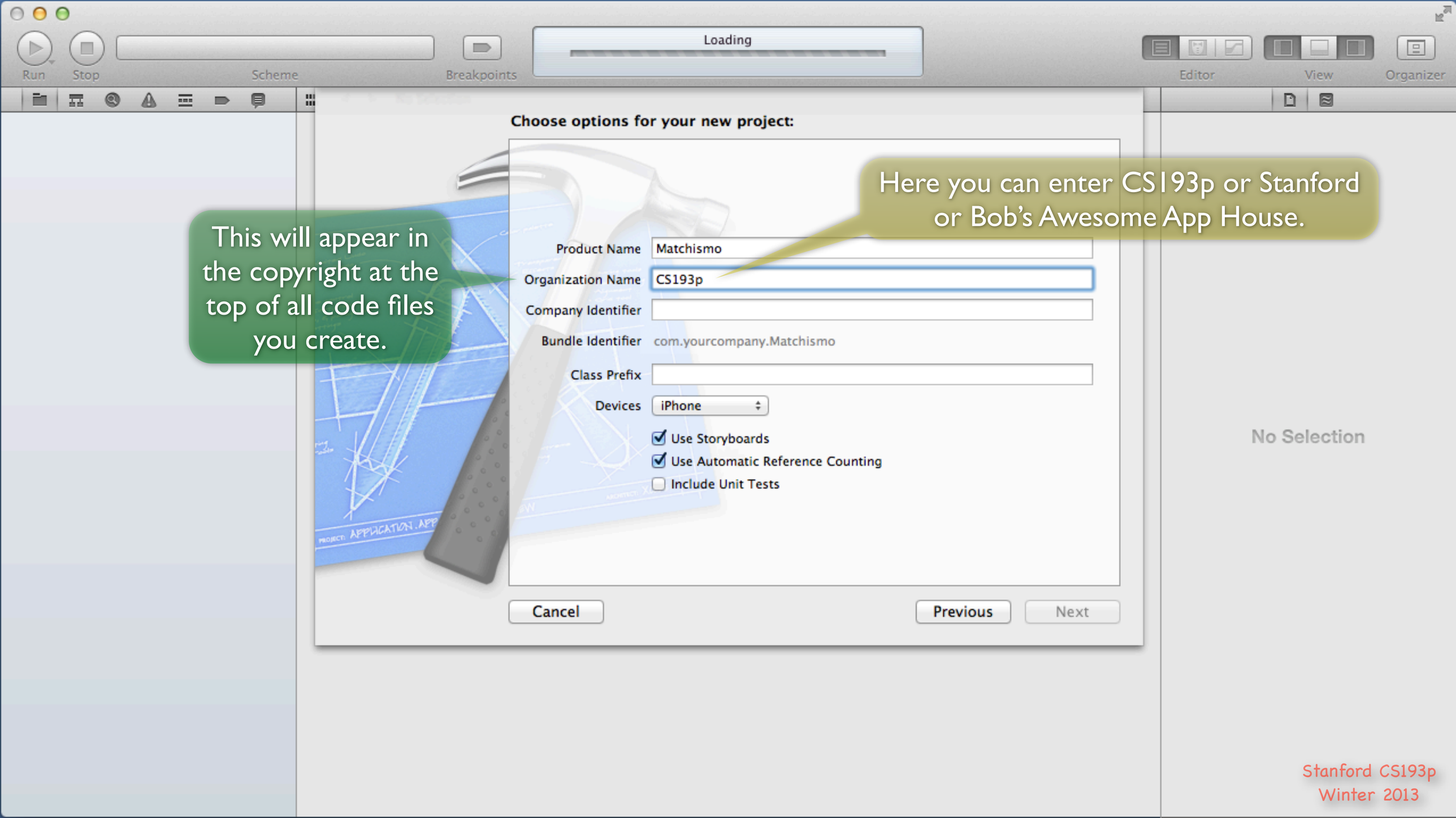
Launch Xcode 4 and click here to create a new project.

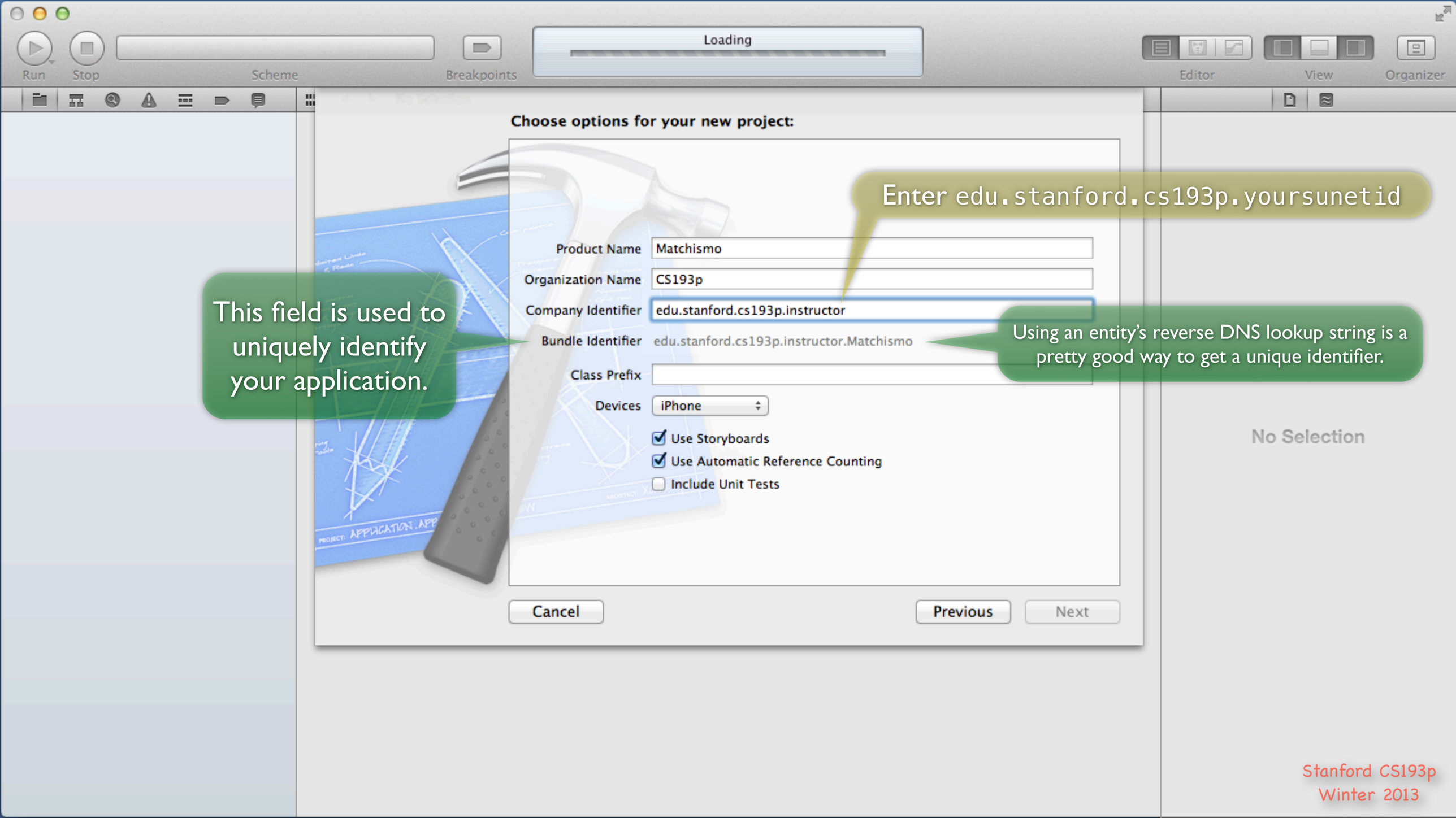
As you create projects, they will appear here.

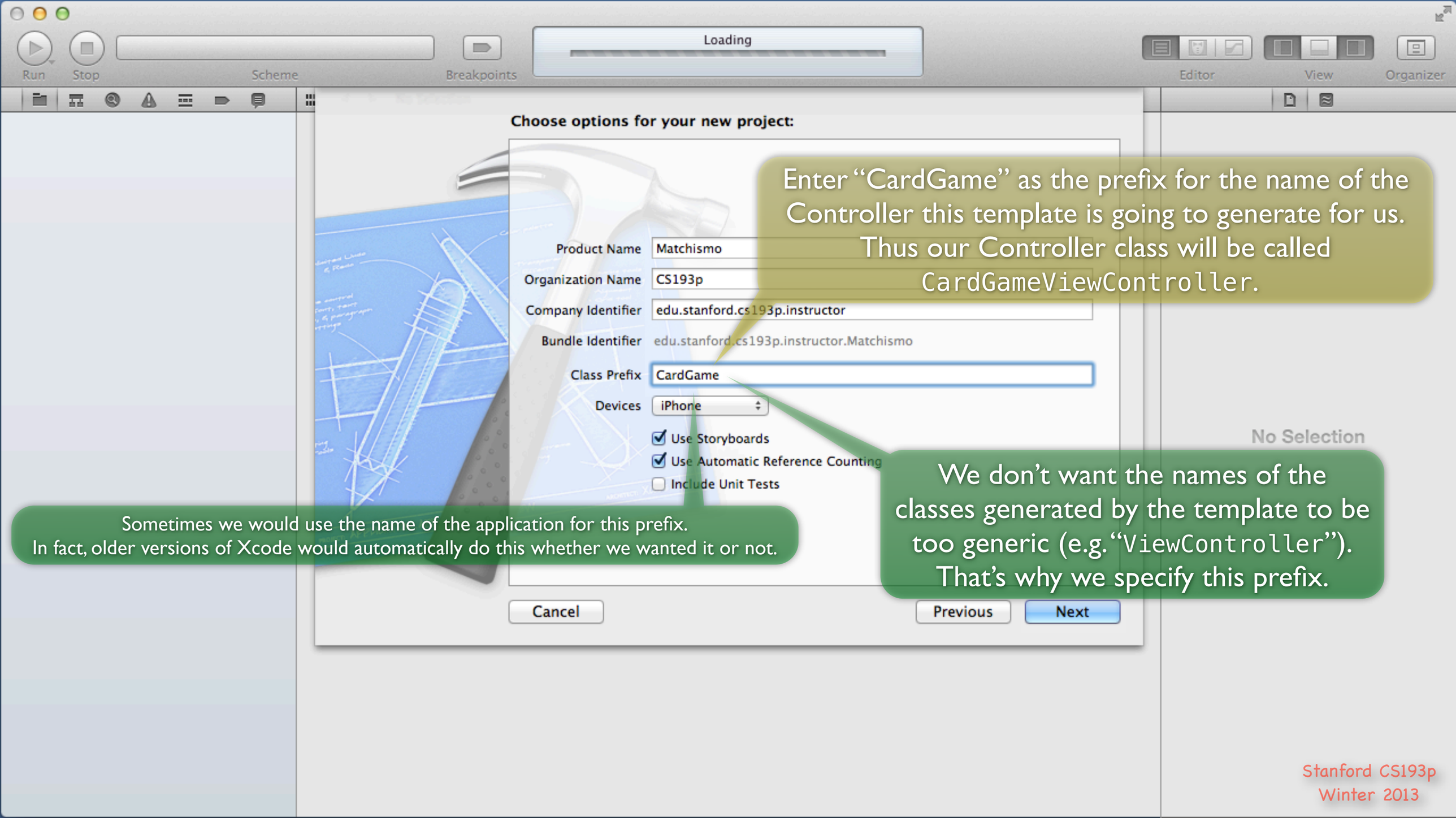

Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose a template for your new project
iOS
Application
Framework & Library
Other
OS X
Application
Framework & Library
Application Plug-in
System Plug-in
Other
Master-Detail Application
OpenGL Game
Page-Based Application
Single View Application
Tabbed Application
Utility Application
Empty Application
Single View Application
This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.
Cancel
Previous
Next
Xcode 4 can be used to develop both iOS and Mac OSX applications.
These buttons are used to select a template which Xcode 4 uses to generate some code to get you started.
Click on the "Single View Application" template. It creates a simple MVC application.
Then click Next.


Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Our first application is going to be a
Card Matching Game
Choose options for your new project:
Product Name
Organization Name
Company Identifier
Bundle Identifier
com.yourcompany.ProductName
Class Prefix
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
Cancel
Previous
Next
No Selection
These fields describe your project.
We'll be filling them in during the next few slides.
Stanford CS193p
Winter 2013


Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose options for your new project:
Product Name
Matchismo
Organization Name
Company Identifier
Bundle Identifier
com.yourcompany.Matchismo
Class Prefix
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
Cancel
Previous
Next
No Selection
The name of our project is going to be "Matchismo" so type that in here.
Stanford CS193p
Winter 2013


Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
Bundle Identifier
com.yourcompany.Matchismo
Class Prefix
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
Cancel
Previous
Next
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.
No Selection
Stanford CS193p
Winter 2013

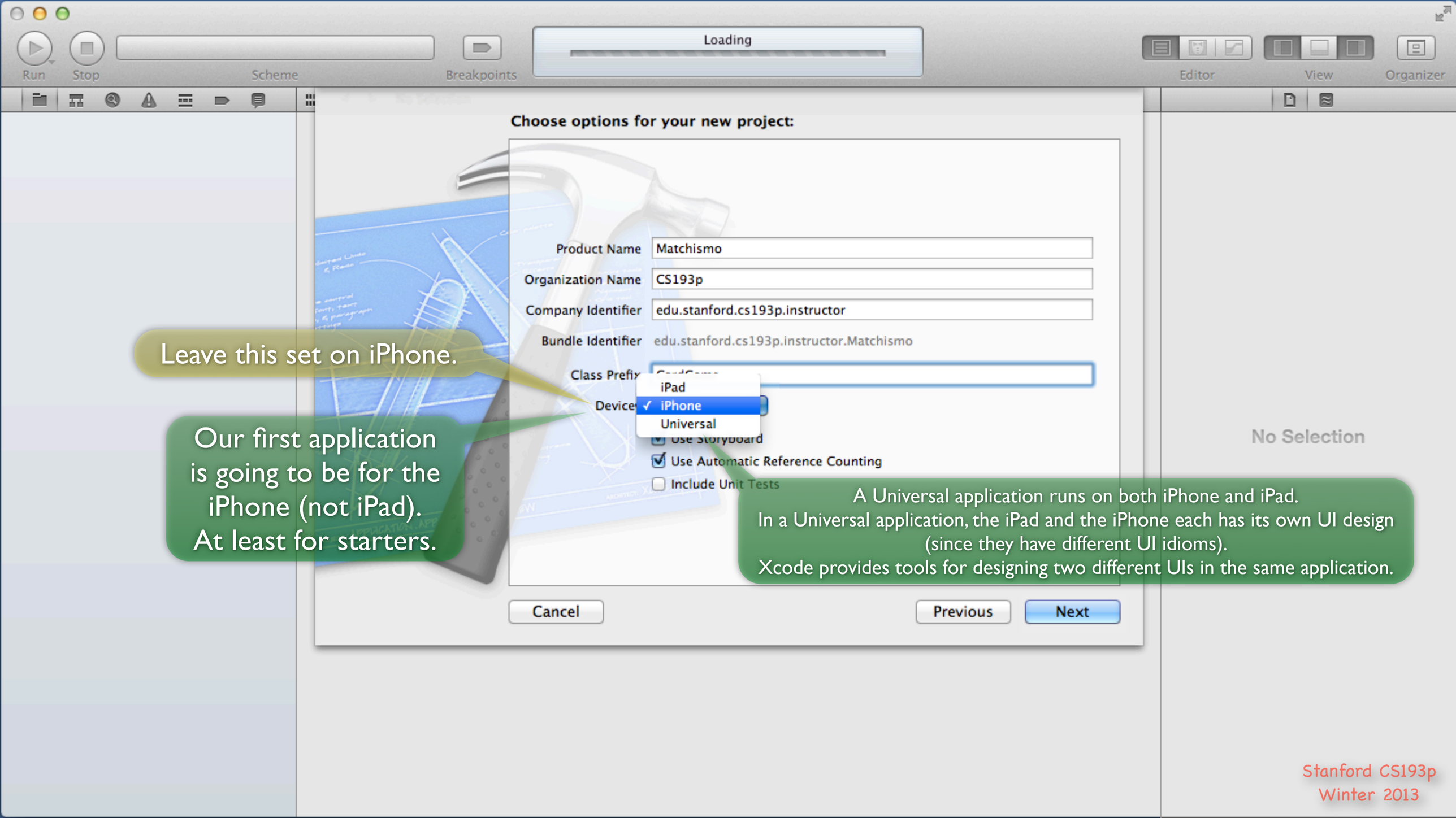

Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
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
Stanford CS193p
Winter 2013

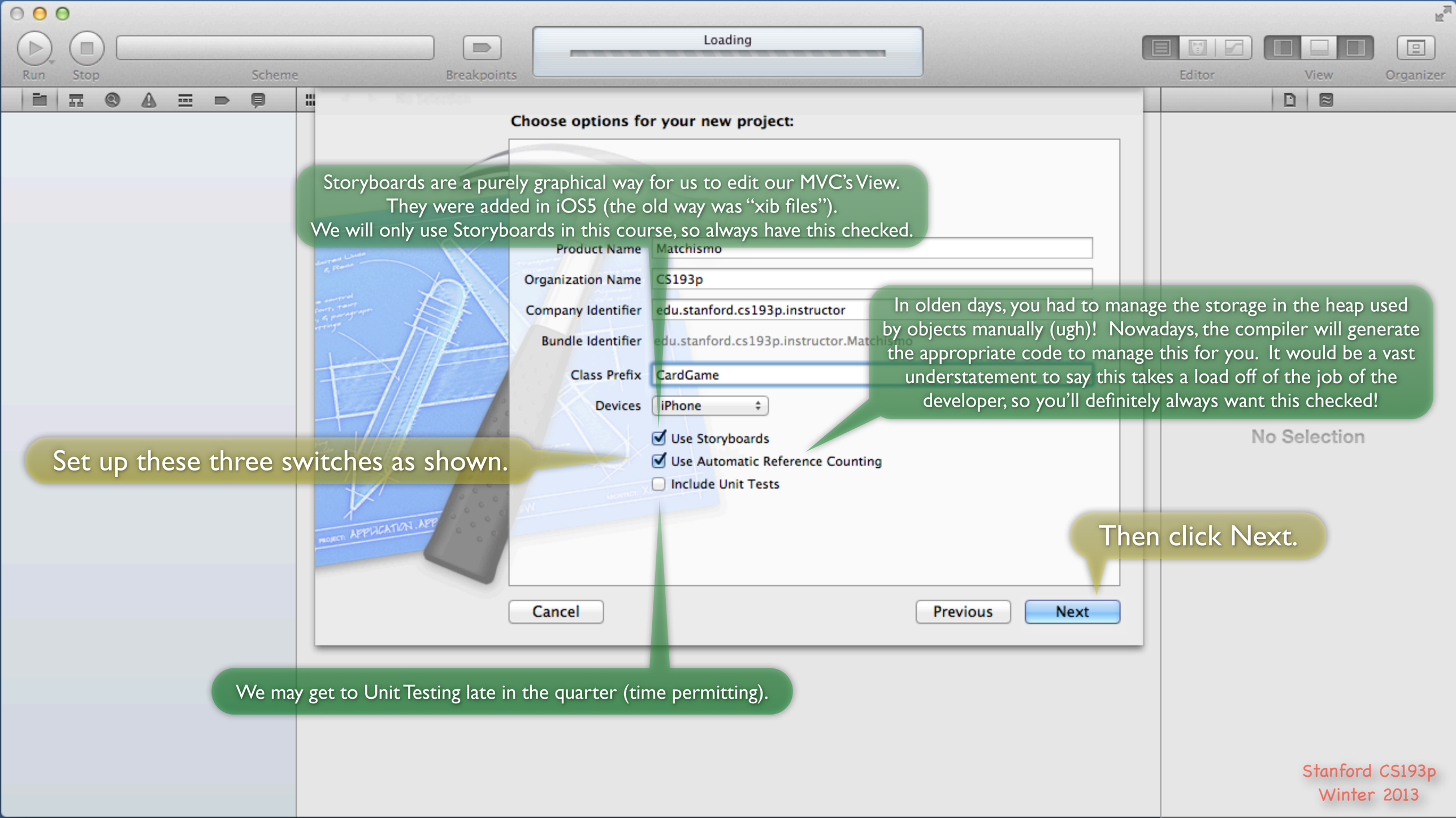


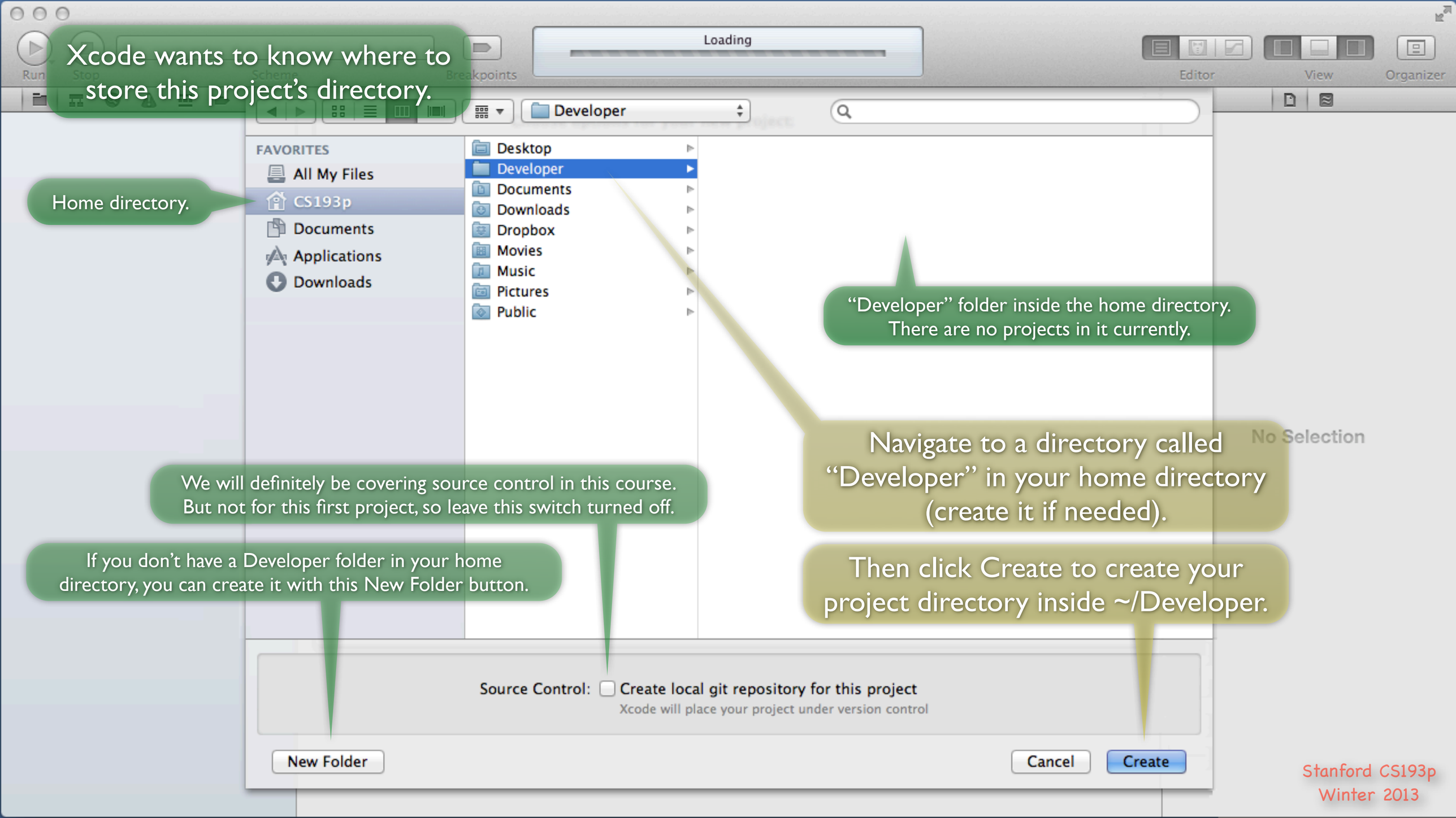
Enter "CardGame" as the prefix for the name of the Controller this template is going to generate for us. Thus our Controller class will be called `CardGameViewController`.

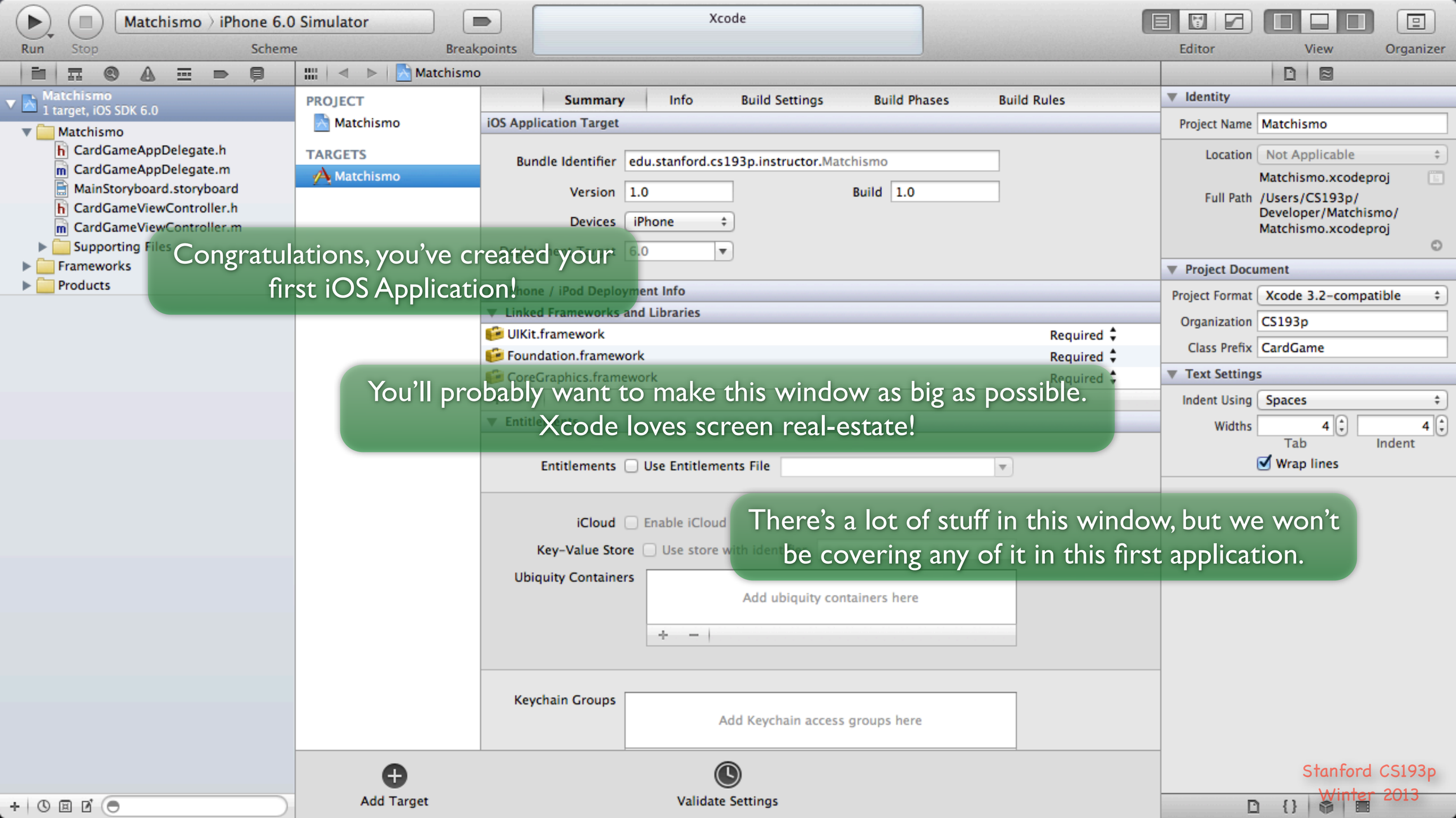
Sometimes we would use the name of the application for this prefix.
In fact, older versions of Xcode would automatically do this whether we wanted it or not.

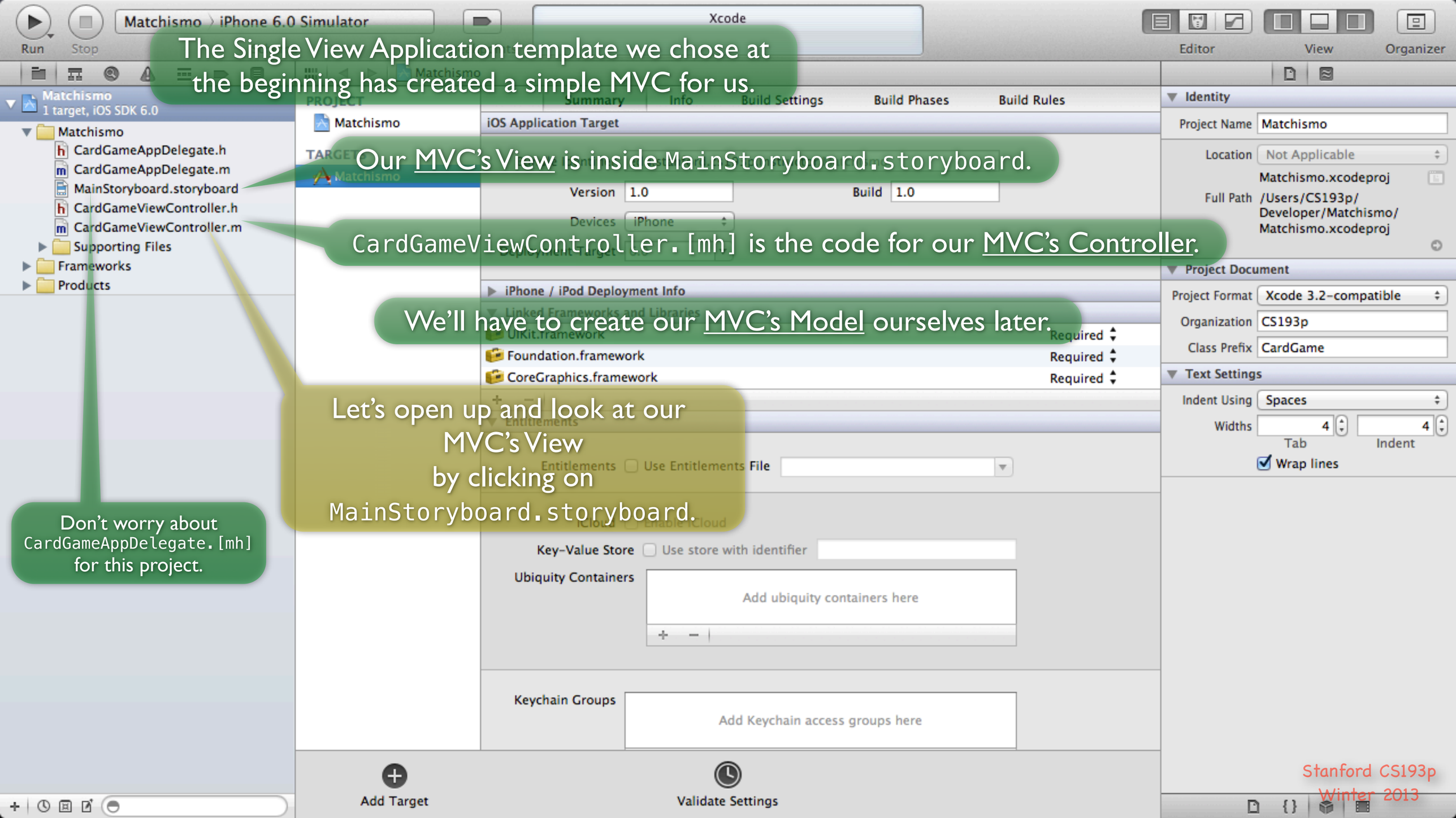
We don't want the names of the classes generated by the template to be too generic (e.g. "`ViewController`"). That's why we specify this prefix.

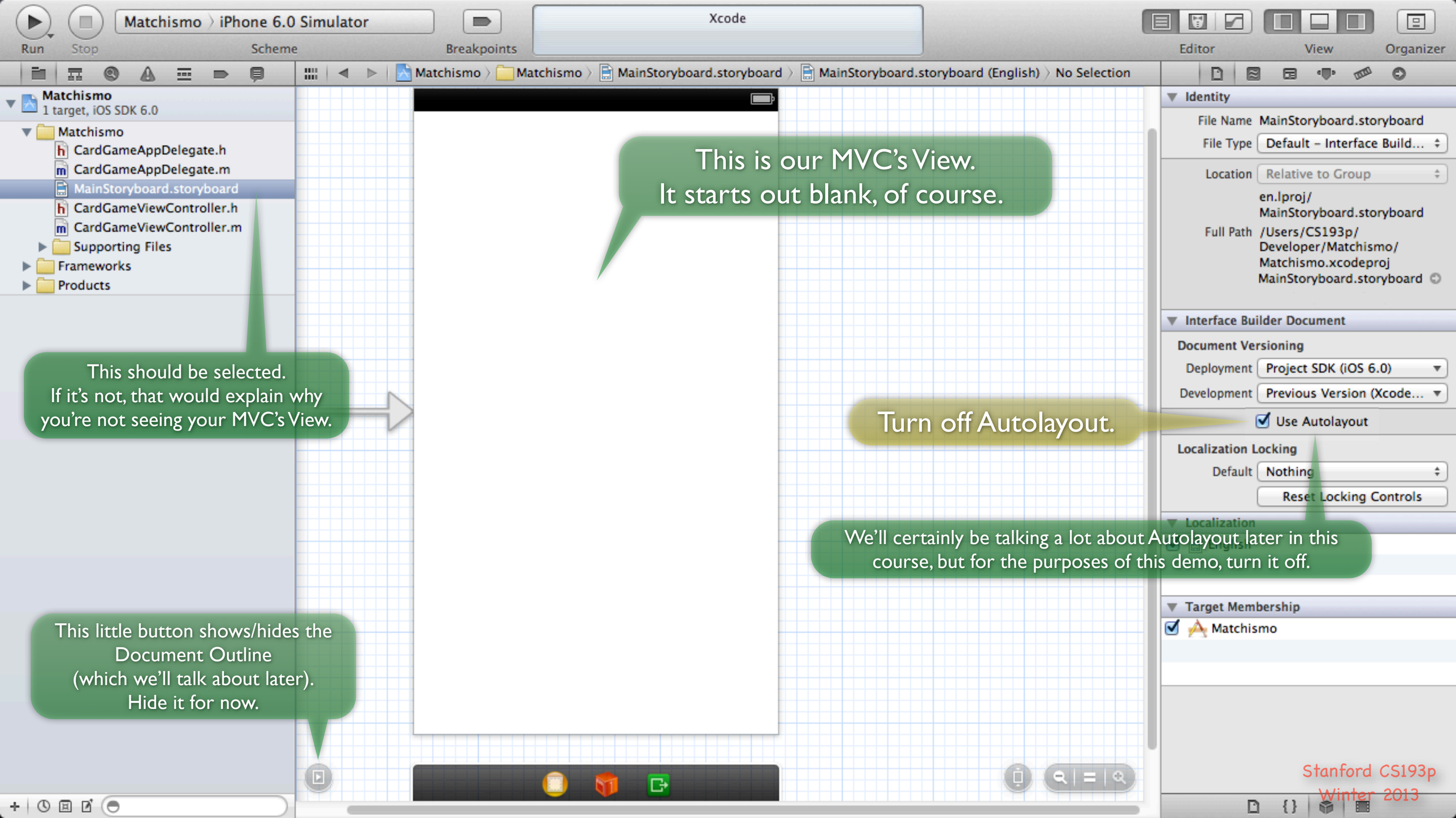

Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose options for your new project:
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
Device
iPad
iPhone
Universal
Use Storyboard
Use Automatic Reference Counting
Include Unit Tests
Cancel
Previous
Next
No Selection
Leave this set on 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.
Stanford CS193p
Winter 2013

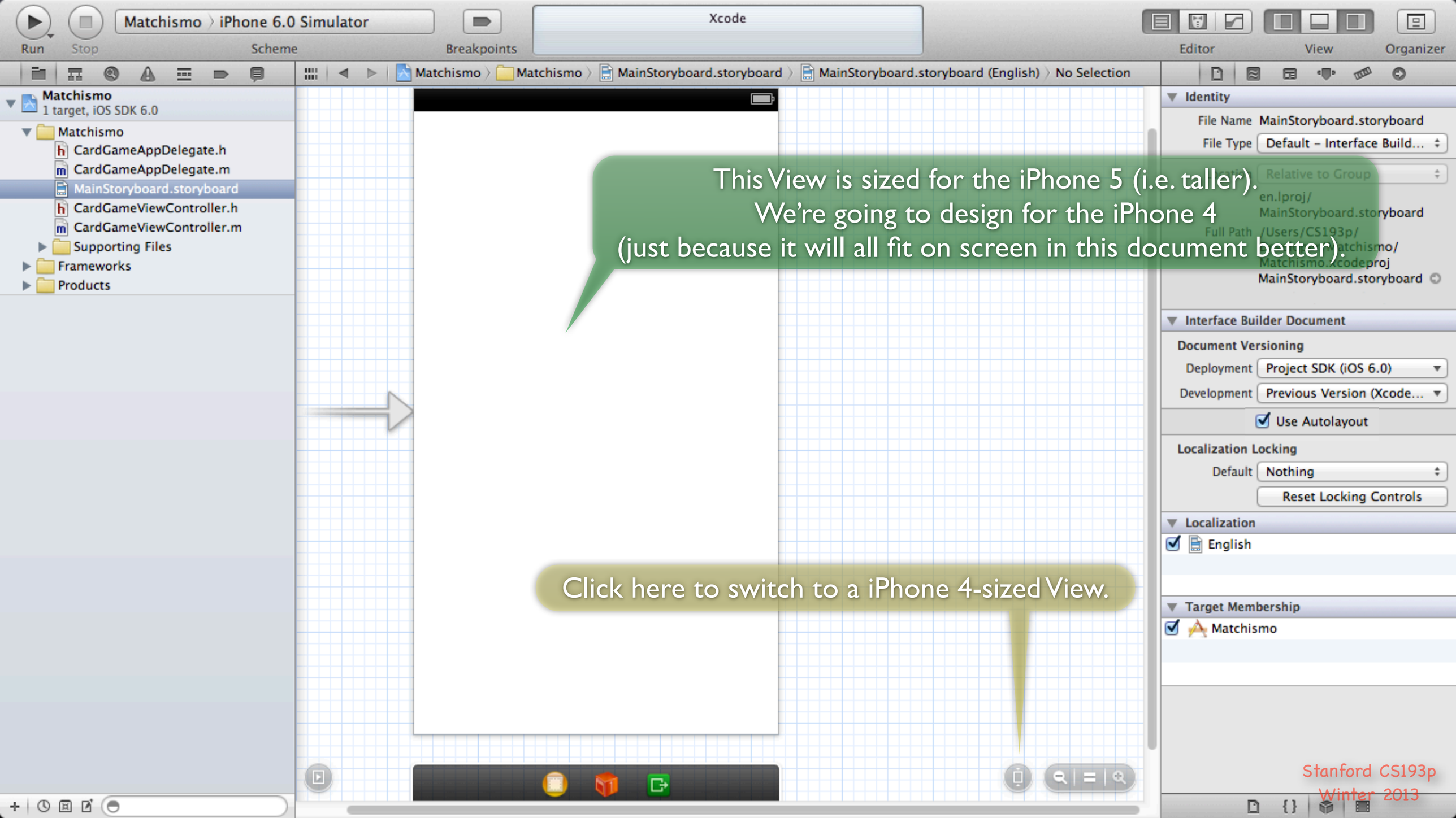

Run
Stop
Scheme
Breakpoints
Loading
Editor
View
Organizer
Choose options for your new project:
Storyboards are a purely graphical way for us to edit our MVC's View.
They were added in iOS5 (the old way was "xib files").
We will only use Storyboards in this course, so always have this checked.
Product Name
Matchismo
Organization Name
CS193p
Company Identifier
edu.stanford.cs193p.instructor
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Class Prefix
CardGame
Devices
iPhone
Use Storyboards
Use Automatic Reference Counting
Include Unit Tests
In olden days, you had to manage the storage in the heap used by objects manually (ugh)! Nowadays, the compiler will generate the appropriate code to manage this for you. It would be a vast understatement to say this takes a load off of the job of the developer, so you'll definitely always want this checked!
No Selection
Set up these three switches as shown.
Then click Next.
Cancel
Previous
Next
We may get to Unit Testing late in the quarter (time permitting).
Stanford CS193p
Winter 2013

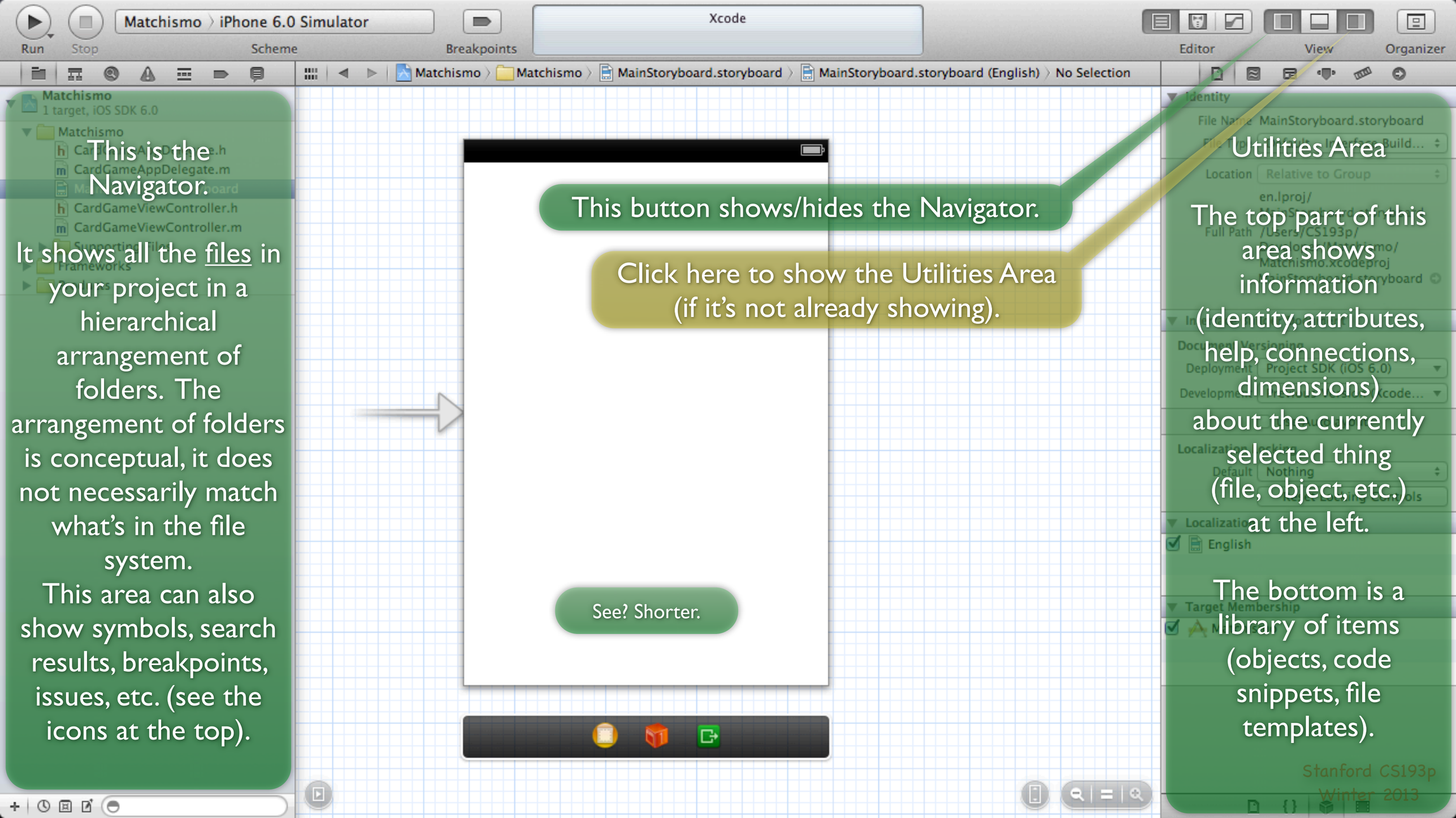

Xcode wants to know where to store this project's directory.
Loading
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
Developer
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Desktop
Developer
Documents
Downloads
Dropbox
Movies
Music
Pictures
Public
Home directory.
"Developer" folder inside the home directory. There are no projects in it currently.
No Selection
Navigate to a directory called "Developer" in your home directory (create it if needed).
We will definitely 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 local git repository for this project
Xcode will place your project under version control
New Folder
Cancel
Create

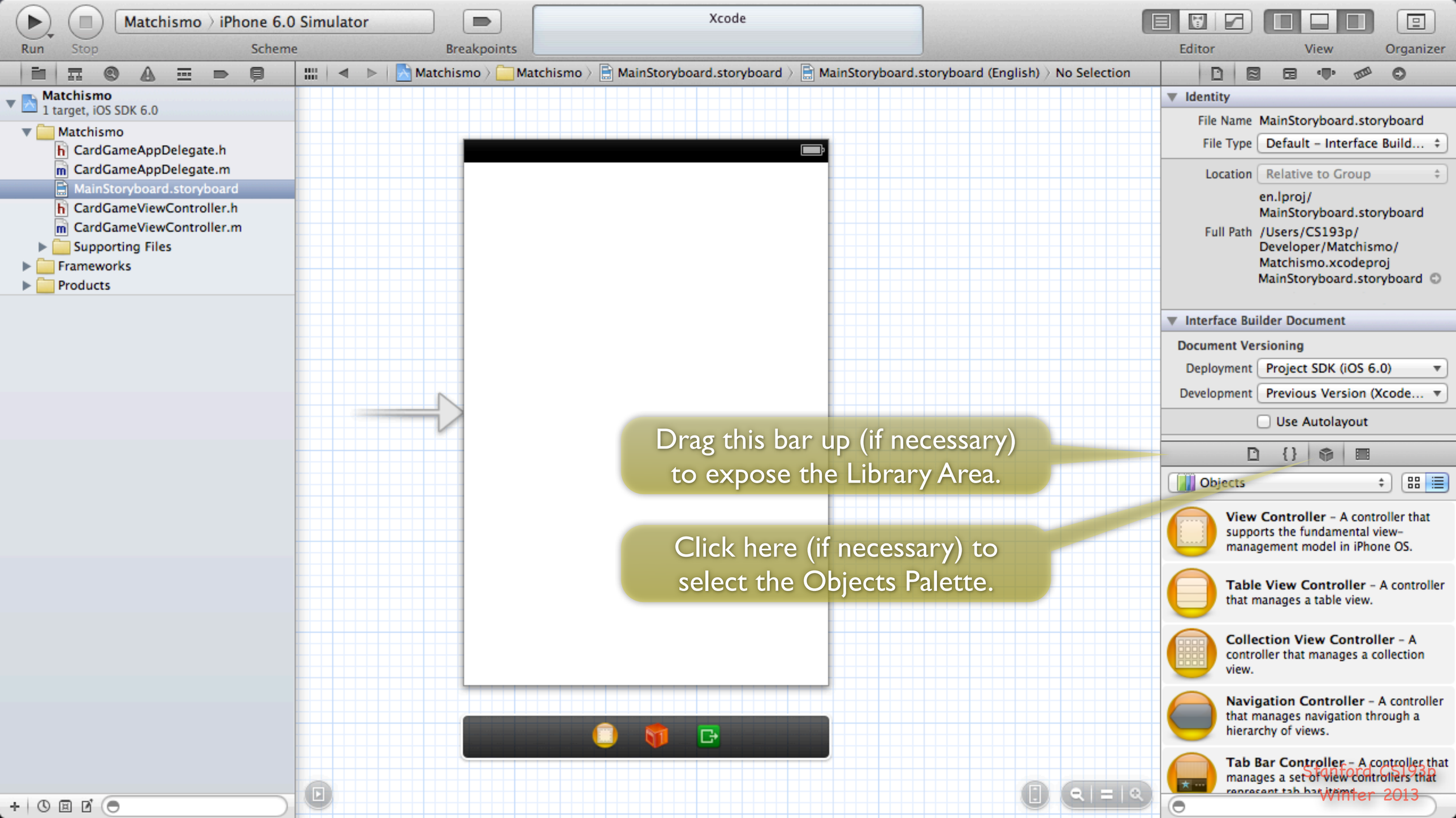

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
PROJECT
Matchismo
TARGETS
Matchismo
Summary
Info
Build Settings
Build Phases
Build Rules
iOS Application Target
Bundle Identifier
edu.stanford.cs193p.instructor.Matchismo
Version
1.0
Build
1.0
Devices
iPhone
Linked Frameworks and Libraries
UIKit.framework
Foundation.framework
CoreGraphics.framework
Required
Entitlements
Use Entitlements File
iCloud
Enable iCloud
Key-Value Store
Ubiquity Containers
Add ubiquity containers here
Keychain Groups
Add Keychain access groups here
Add Target
Validate Settings
Identity
Project Name
Matchismo
Location
Not Applicable
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
Tab
4
Indent
Wrap lines
Congratulations, you've created your first iOS Application!
You'll probably want to make this window as big as possible. Xcode loves screen real-estate!
There's a lot of stuff in this window, but we won't be covering any of it in this first application.
Stanford CS193p
Winter 2013

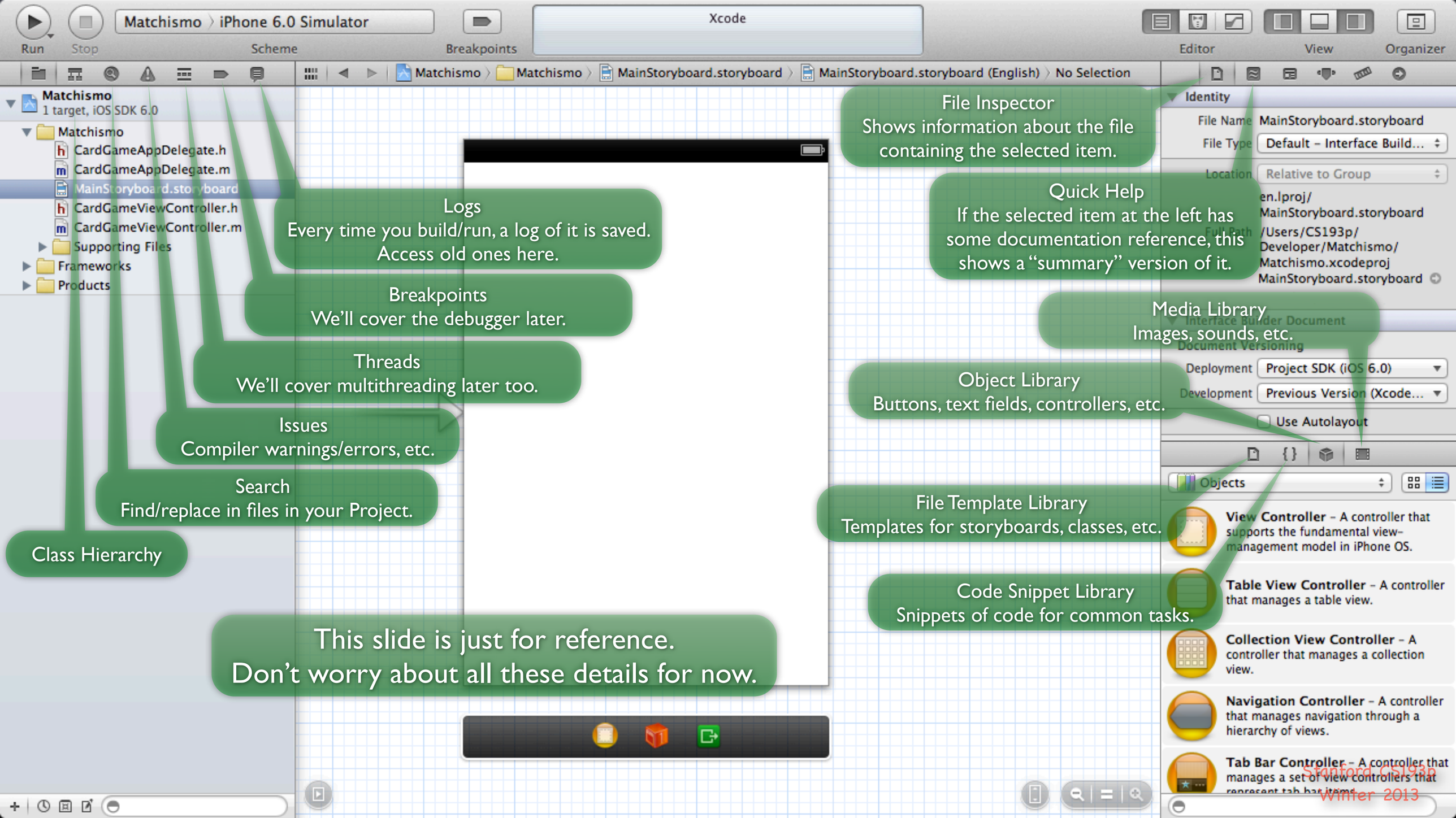

The Single View Application template we chose at the beginning has created a simple MVC for us.
Our MVC's View is inside MainStoryboard.storyboard.
CardGameViewController.[mh] is the code for our MVC's Controller.
We'll have to create our MVC's Model ourselves later.
Let's open up and look at our MVC's View by clicking on MainStoryboard.storyboard.
Don't worry about CardGameAppDelegate.[mh] for this project.
Run
Stop
Matchismo
iPhone 6.0 Simulator
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Summary
Info
Build Settings
Build Phases
Build Rules
iOS Application Target
Version 1.0
Build 1.0
Devices iPhone
iPhone / iPod Deployment Info
Linked Frameworks and Libraries
Foundation.framework Required
CoreGraphics.framework Required
Entitlements
Use Entitlements File
Key-Value Store
Use store with identifier
Ubiquity Containers
Add ubiquity containers here
Keychain Groups
Add Keychain access groups here
Add Target
Validate Settings
Identity
Project Name Matchismo
Location Not Applicable
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
Stanford CS193p
Winter 2013

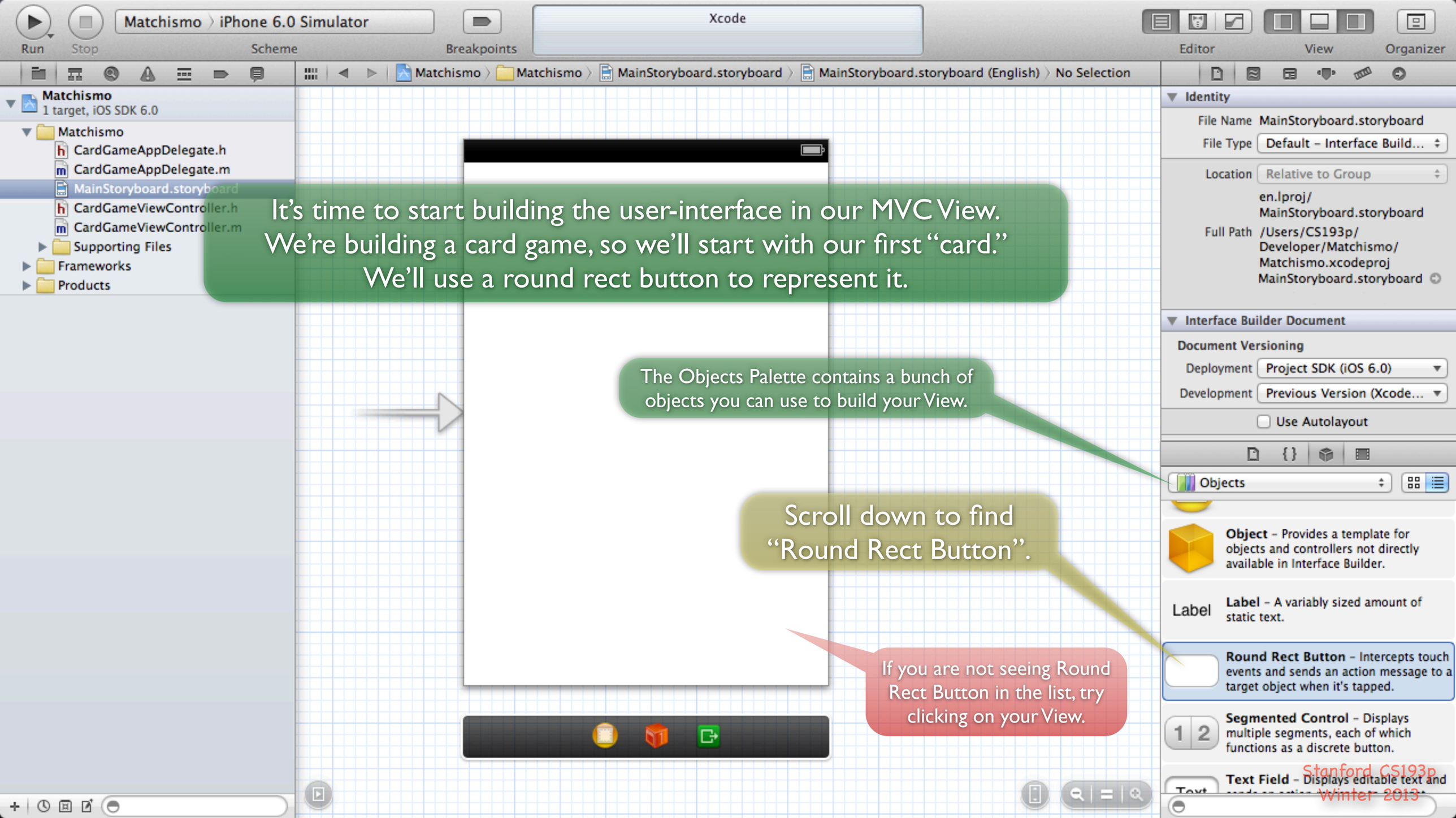

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
This is our MVC's View.
It starts out blank, of course.
This should be selected.
If it's not, that would explain why
you're not seeing your MVC's View.
Turn off Autolayout.
We'll certainly be talking a lot about Autolayout later in this course, but for the purposes of this demo, turn it off.
This little button shows/hides the Document Outline
(which we'll talk about later).
Hide it for now.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
Use Autolayout
Localization Locking
Default Nothing
Reset Locking Controls
Localization
Target Membership
Matchismo
Stanford CS193p
Winter 2013

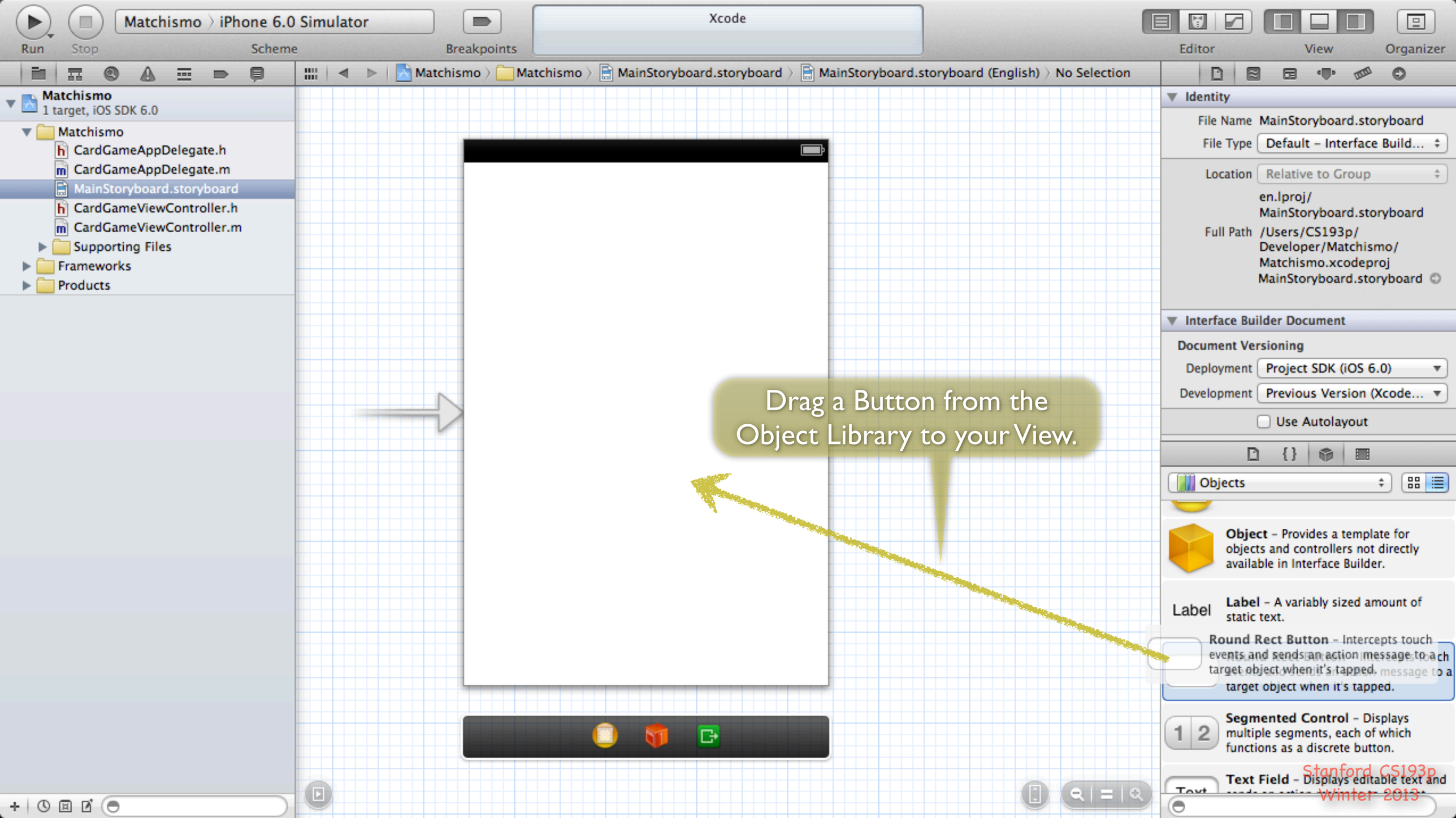

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
This View is sized for the iPhone 5 (i.e. taller).
We're going to design for the iPhone 4
(just because it will all fit on screen in this document better).
Click here to switch to a iPhone 4-sized View.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
Use Autolayout
Localization Locking
Default Nothing
Reset Locking Controls
Localization
English
Target Membership
Matchismo
Stanford CS193p
Winter 2013

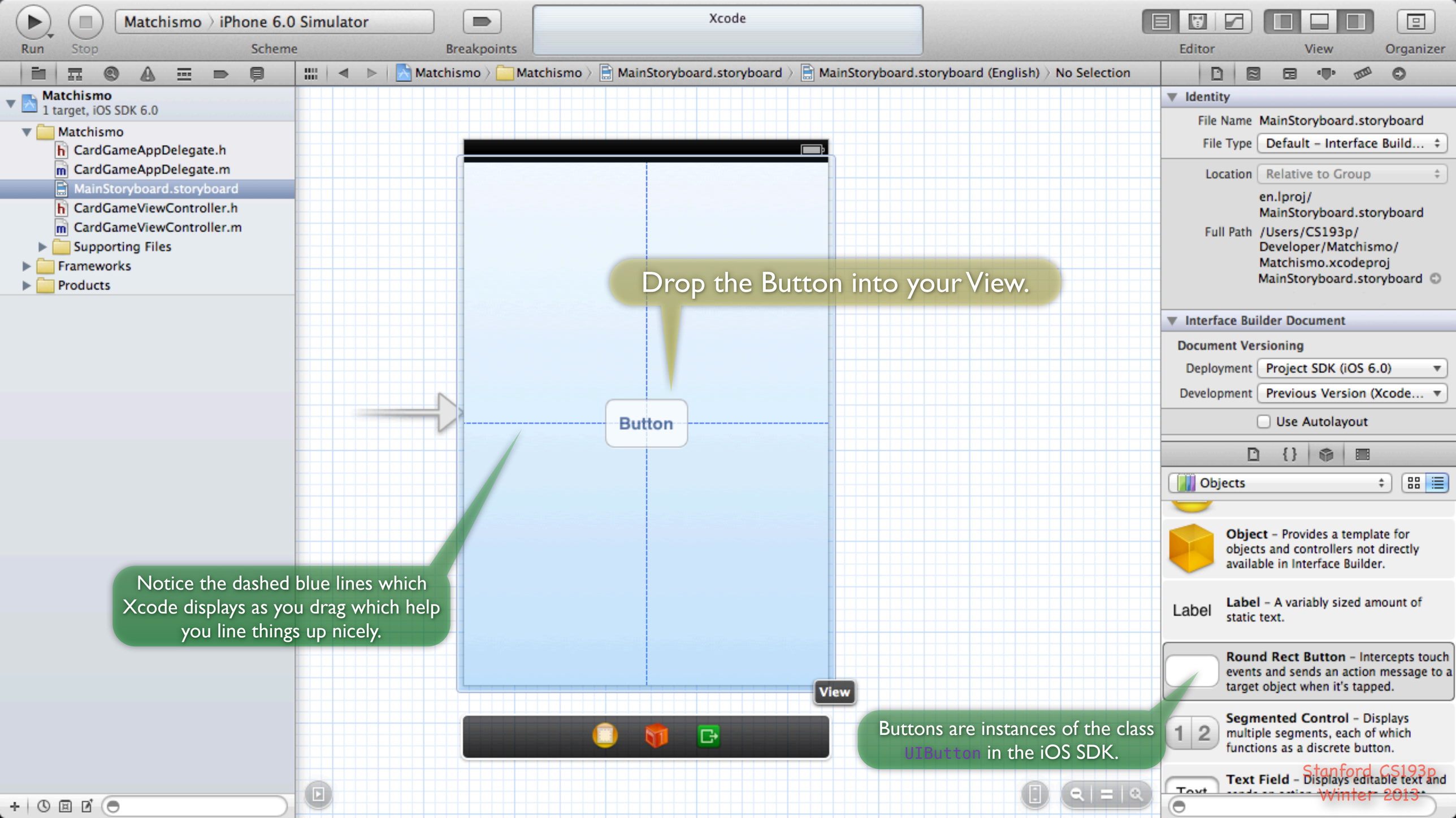

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.m
CardGameViewController.h
CardGameViewController.m
Frameworks
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).
See? Shorter.
Identity
File Name MainStoryboard.storyboard
Location Relative to Group
Full Path /Users/CS193p/Matchismo/Matchismo.xcodeproj
Deployment Project SDK (iOS 6.0)
Localization
English
Target Membership
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).
Stanford CS193p
Winter 2013

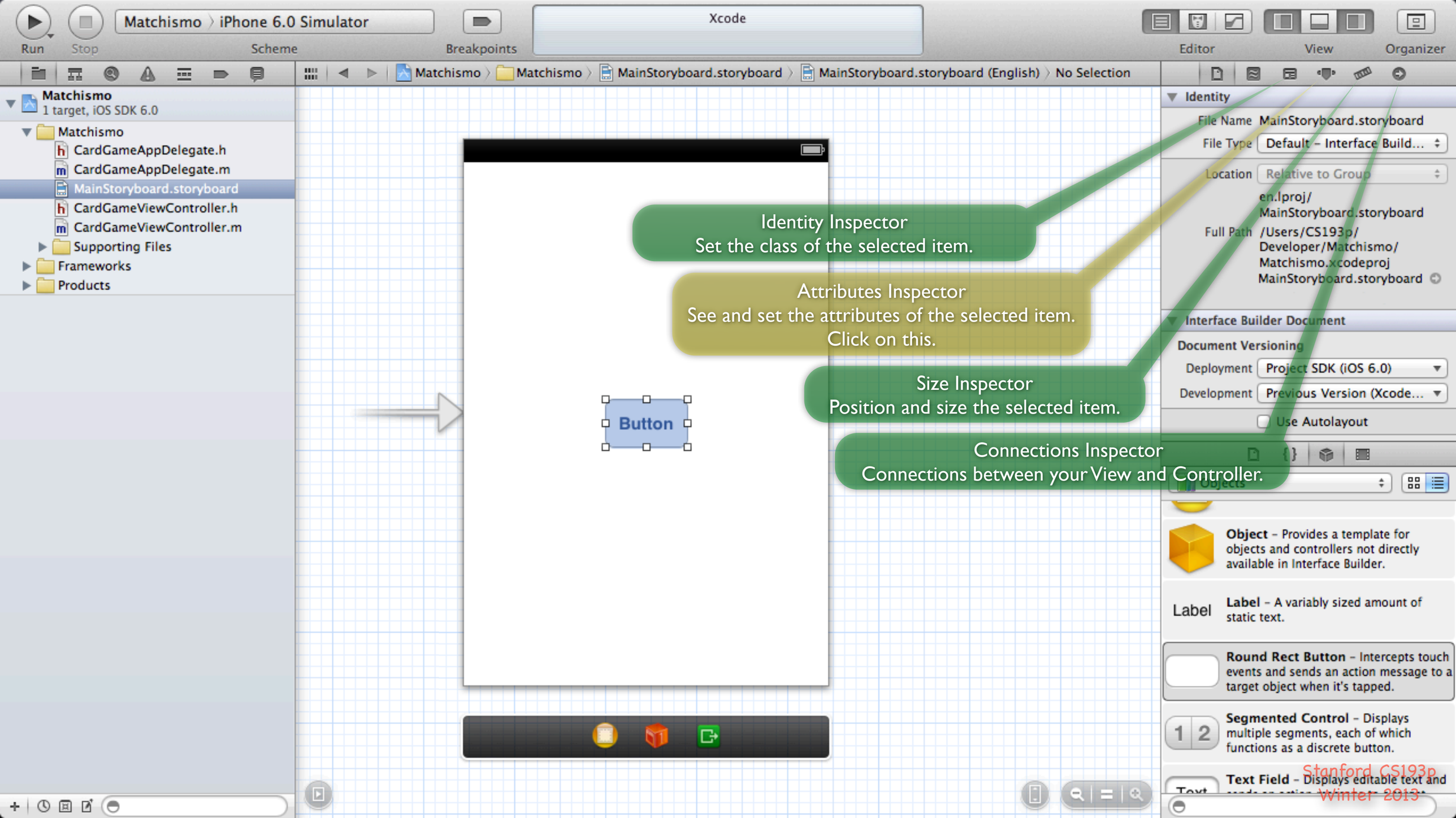

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
Use Autolayout
Drag this bar up (if necessary)
to expose the Library Area.
Click here (if necessary) to
select the Objects Palette.
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

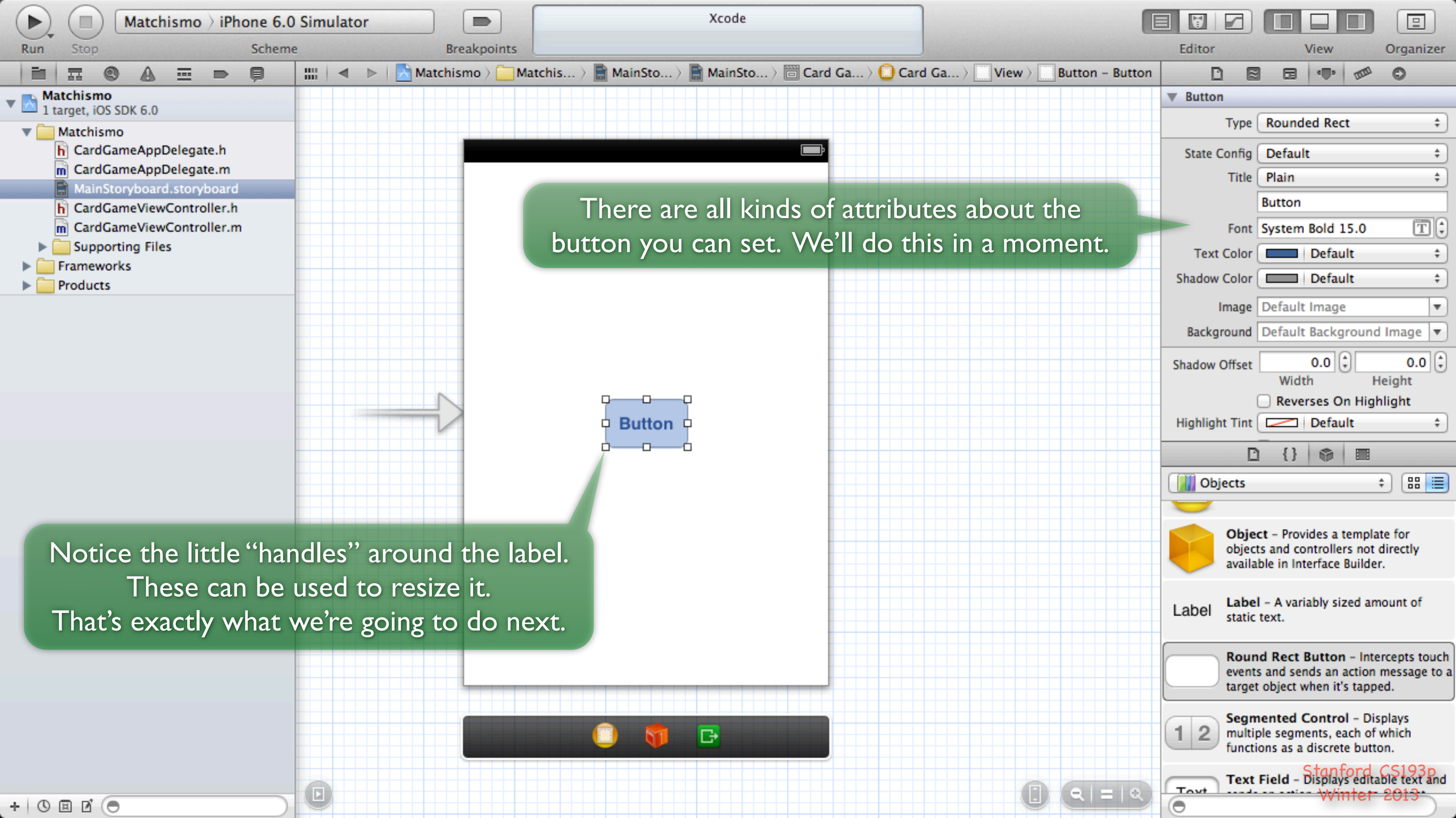

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
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.
Issues
Compiler warnings/errors, etc.
Search
Find/replace in files in your Project.
Class Hierarchy
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.
This slide is just for reference.
Don't worry about all these details for now.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Location Relative to Group
Full Path /Users/CS193p/Developer/Matchismo/Matchismo.xcodeproj/MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
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 represent tab bar items.
Stanford CS193p
Winter 2013

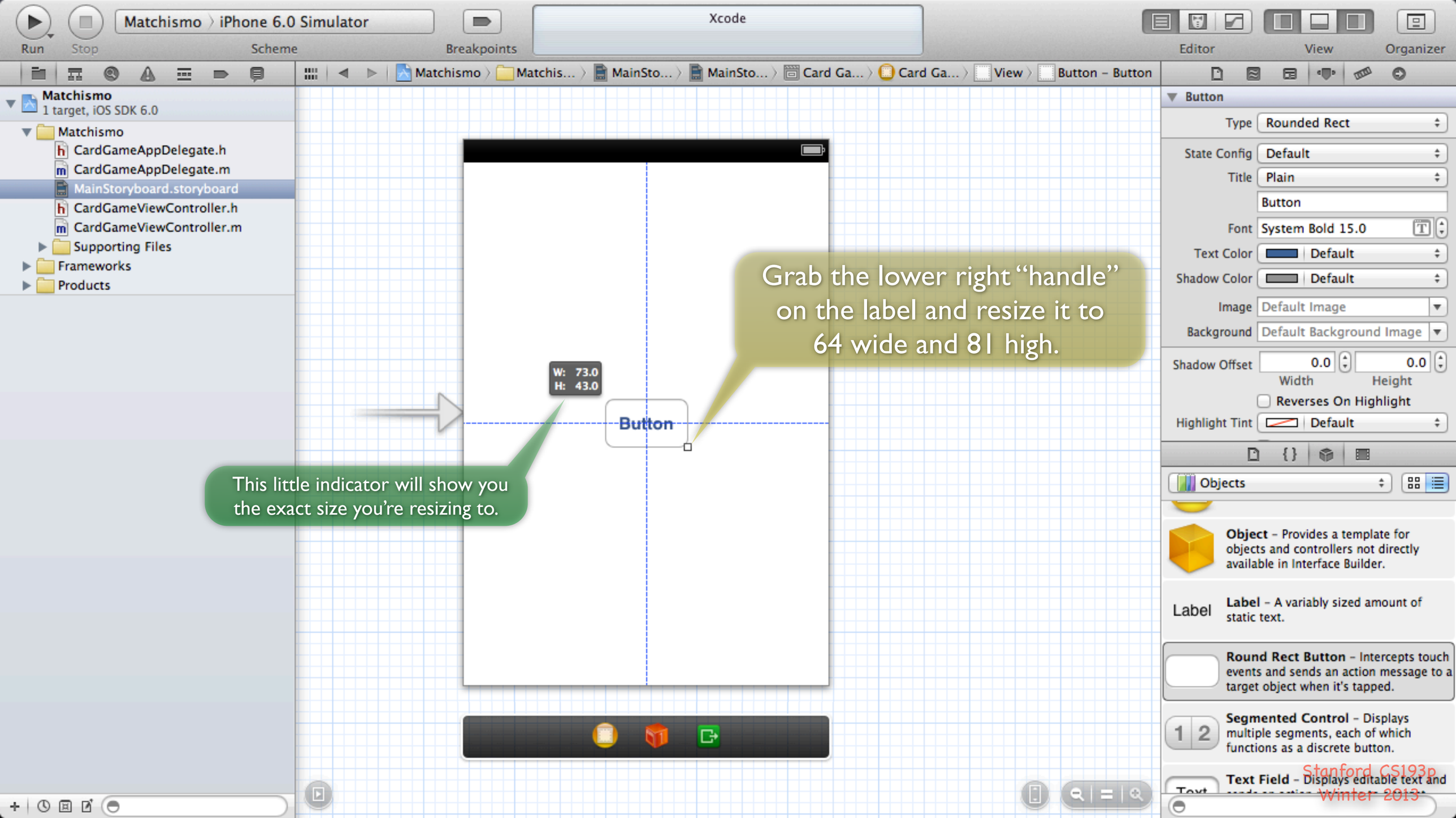

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
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 round rect button to represent it.
The Objects Palette contains a bunch of objects you can use to build your View.
Scroll down to find "Round Rect Button".
If you are not seeing Round Rect Button in the list, try clicking on your View.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build...
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode...
Use Autolayout
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

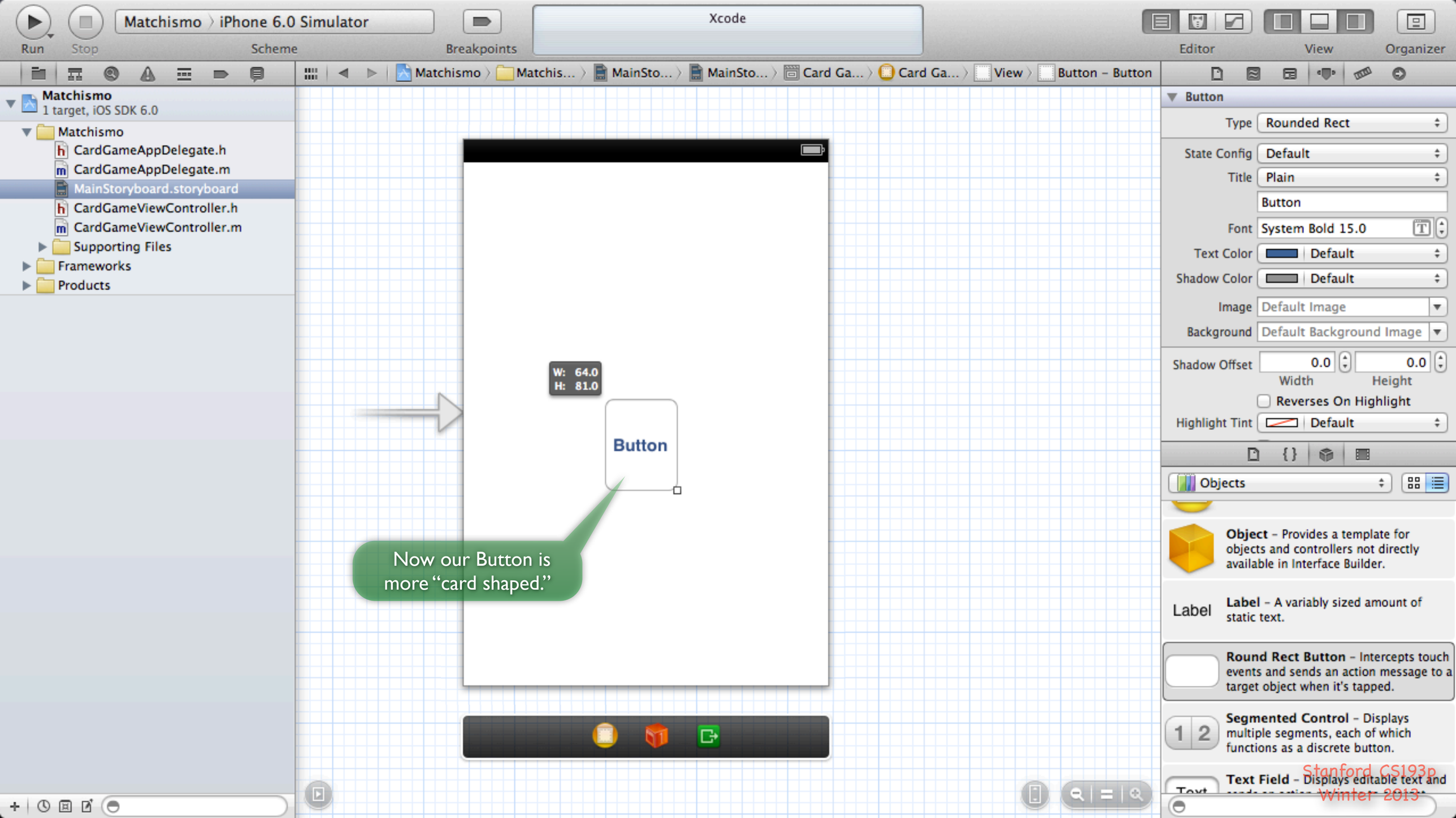

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Drag a Button from the Object Library to your View.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
Use Autolayout
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

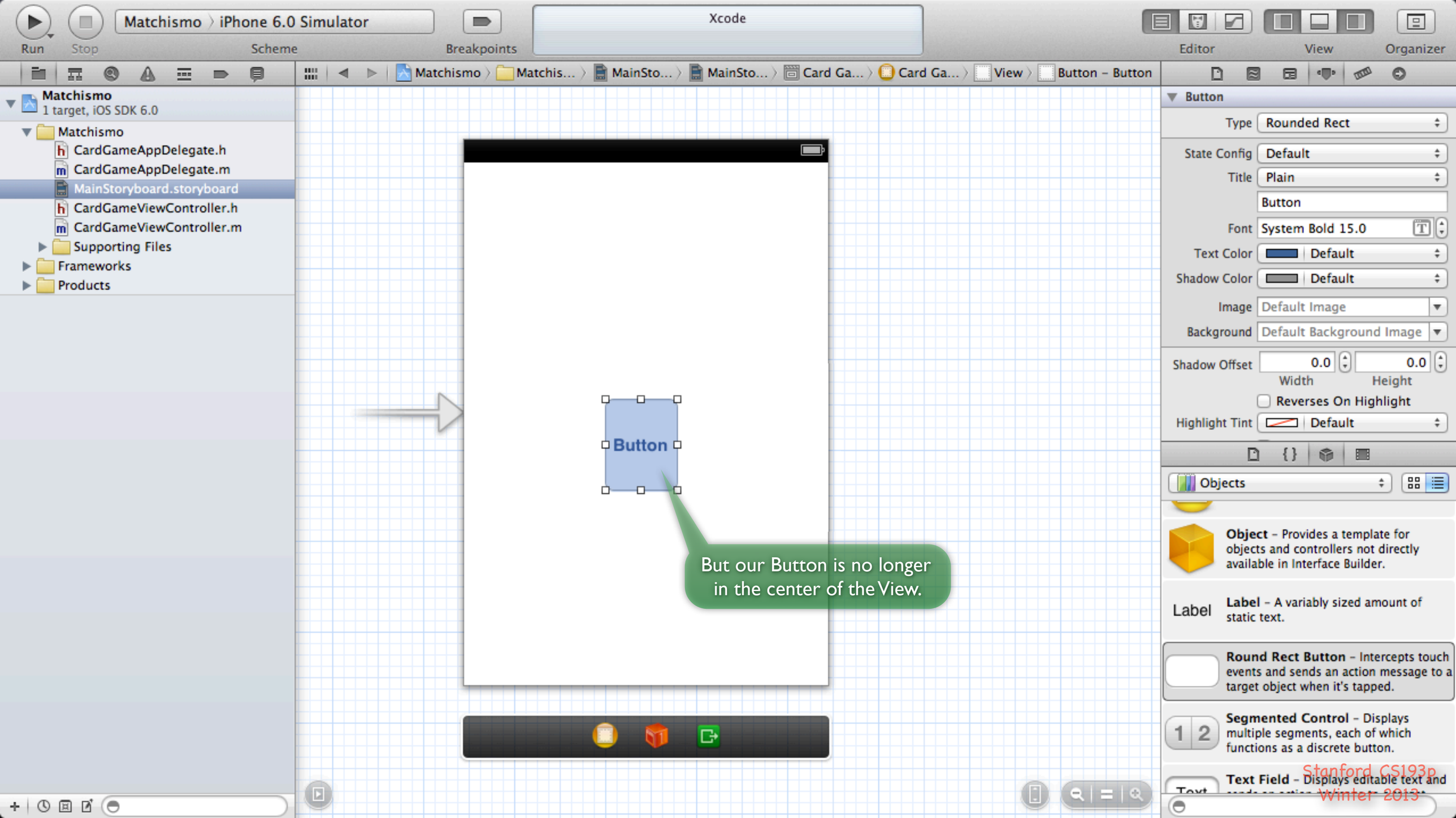

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Drop the Button into your View.
Button
View
Notice the dashed blue lines which Xcode displays as you drag which help you line things up nicely.
Buttons are instances of the class UIButton in the iOS SDK.
Identity
File Name MainStoryboard.storyboard
File Type Default – Interface Build…
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode…
Use Autolayout
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

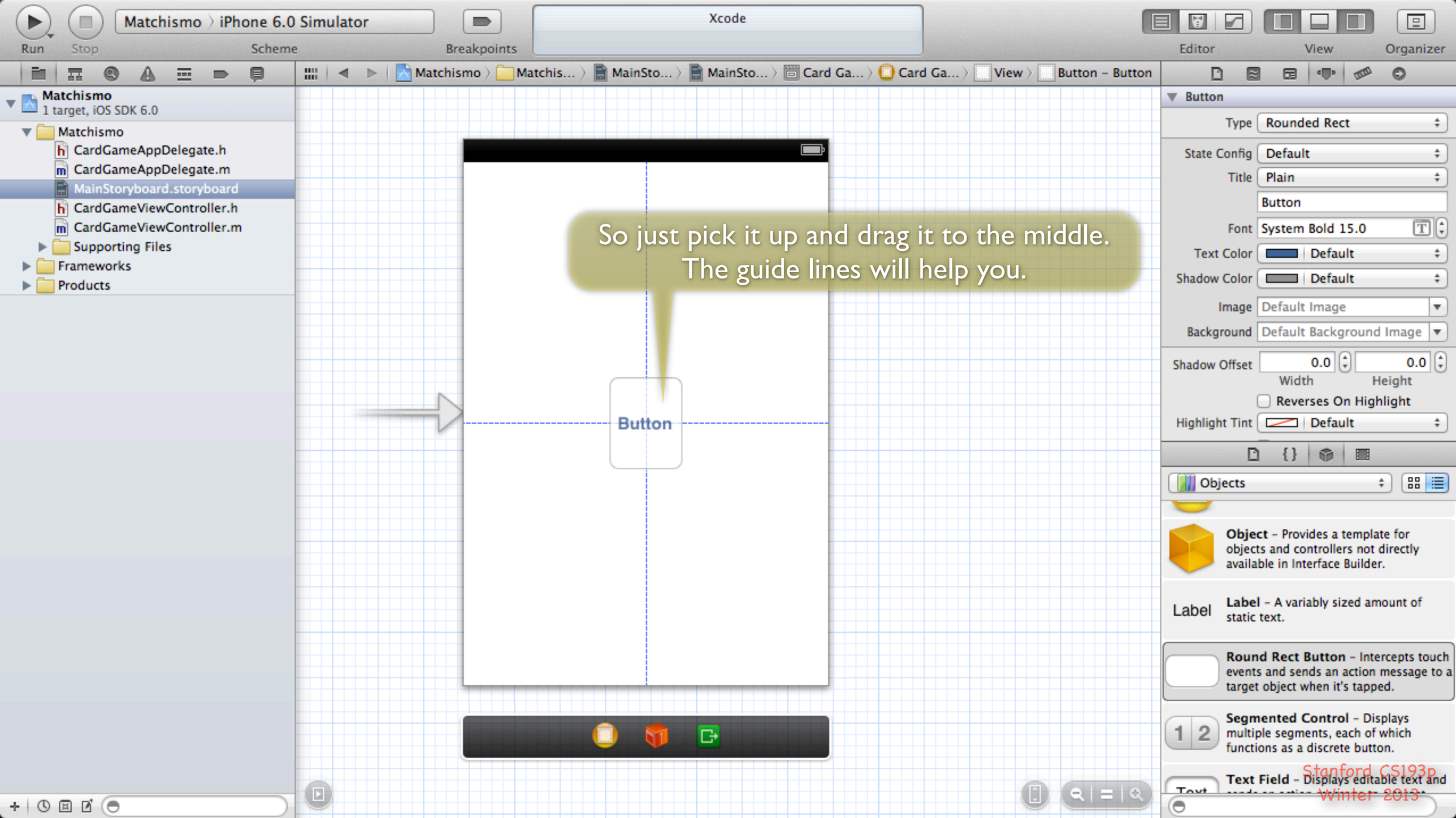

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 No Selection
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
File Name MainStoryboard.storyboard
File Type Default – Interface Build...
Location Relative to Group
en.lproj/
MainStoryboard.storyboard
Full Path /Users/CS193p/
Developer/Matchismo/
Matchismo.xcodeproj
MainStoryboard.storyboard
Interface Builder Document
Document Versioning
Deployment Project SDK (iOS 6.0)
Development Previous Version (Xcode...
Use Autolayout
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

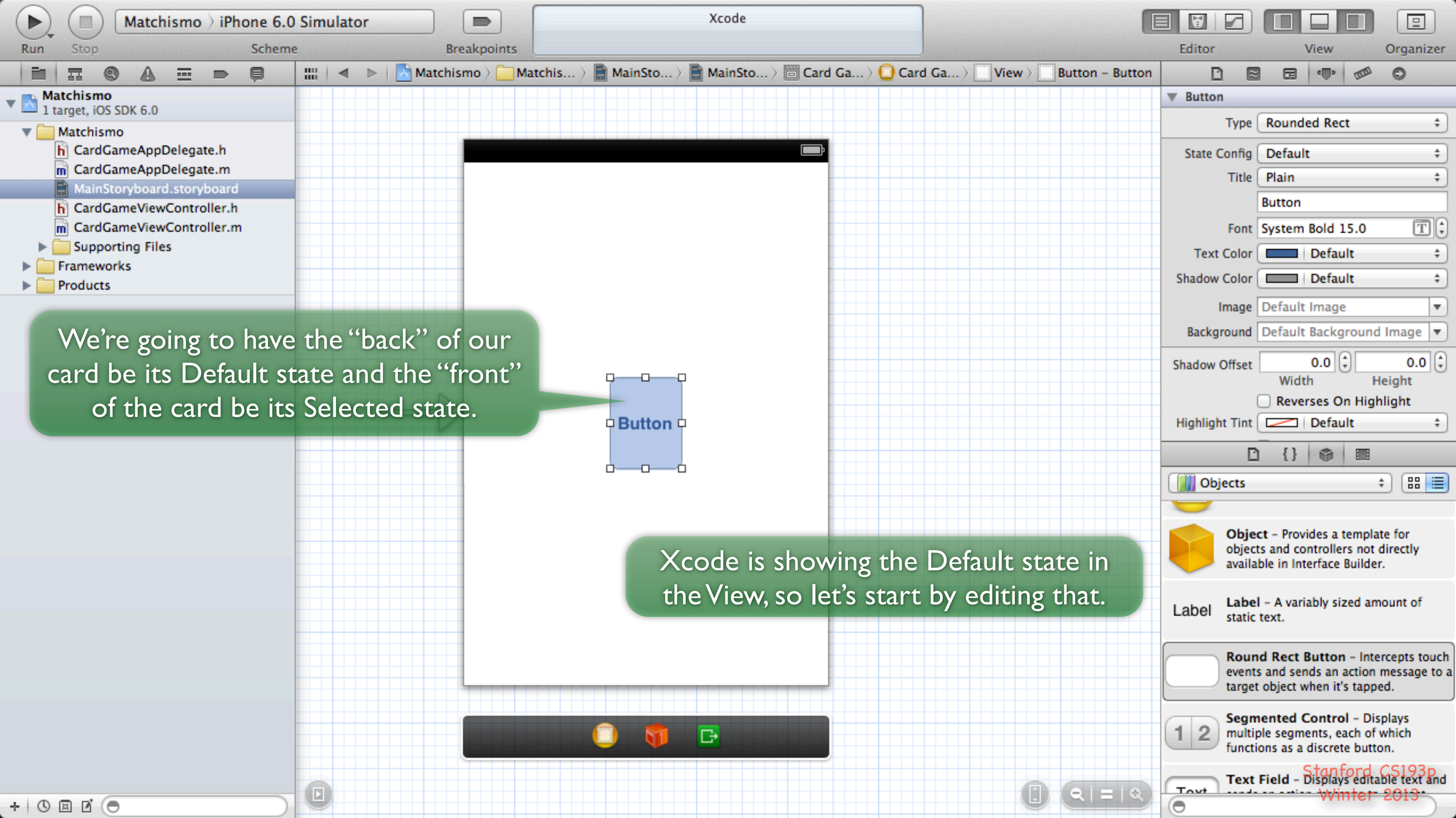

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
There are all kinds of attributes about the button you can set. We'll do this in a moment.
Button
Notice the little "handles" around the label.
These can be used to resize it.
That's exactly what we're going to do next.
Button
Type
Rounded Rect
State Config
Default
Title
Plain
Button
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

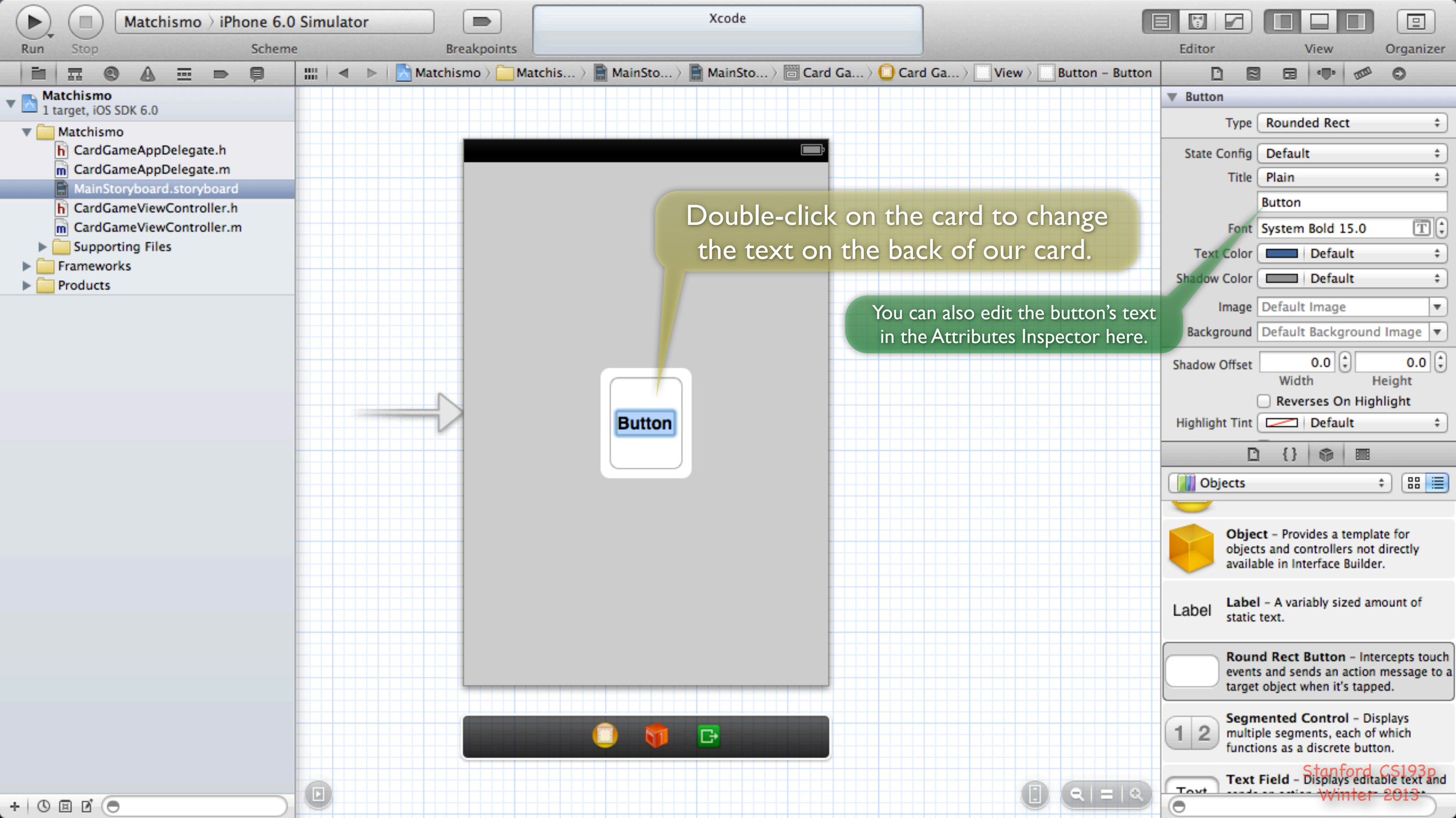

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
W: 73.0
H: 43.0
Button
Grab the lower right "handle" on the label and resize it to 64 wide and 81 high.
This little indicator will show you the exact size you're resizing to.
Button
Type
Rounded Rect
State Config
Default
Title
Plain
Button
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

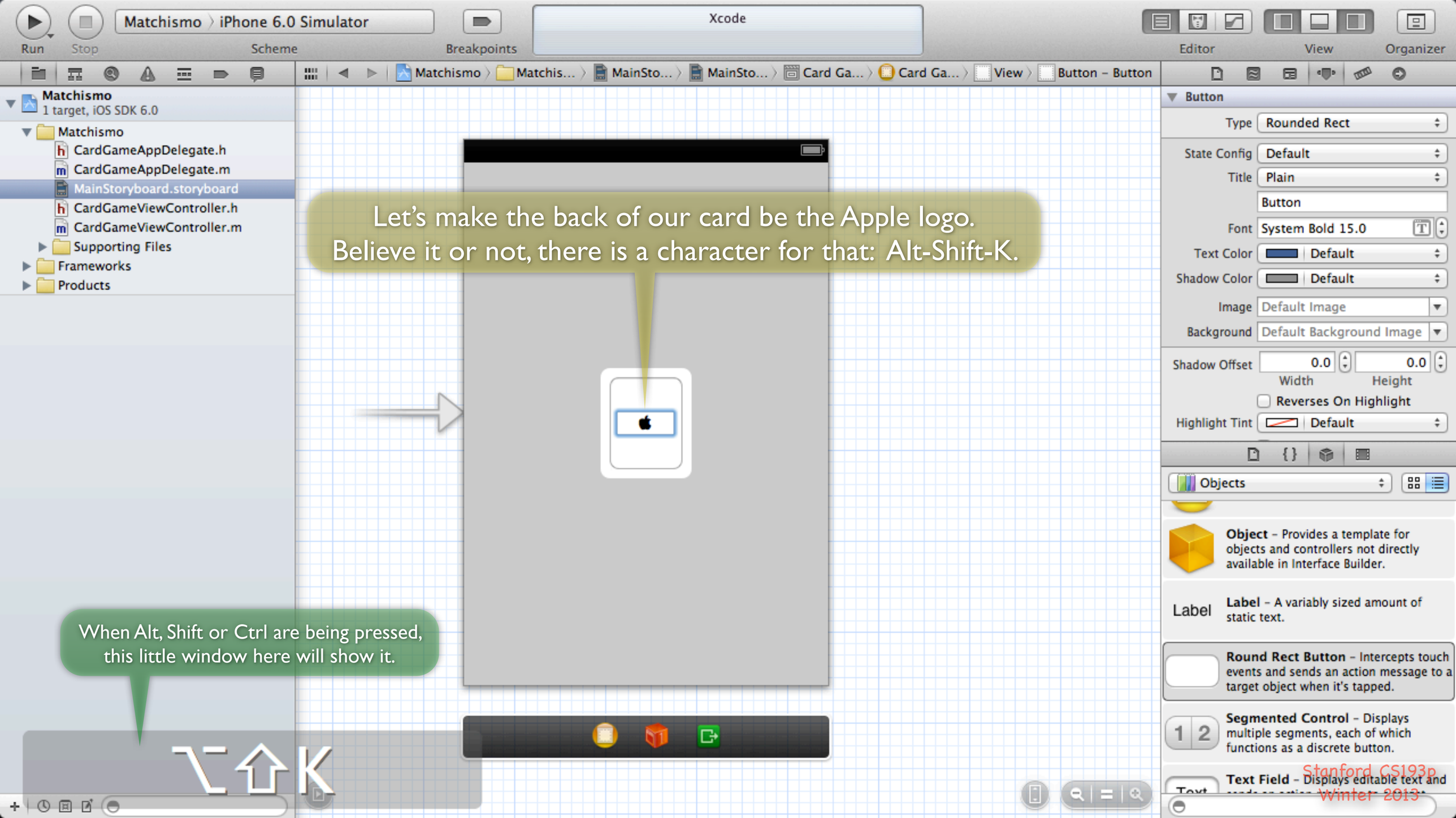

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
W: 64.0
H: 81.0
Button
Now our Button is more "card shaped."
Button
Type Rounded Rect
State Config Default
Title Plain
Button
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

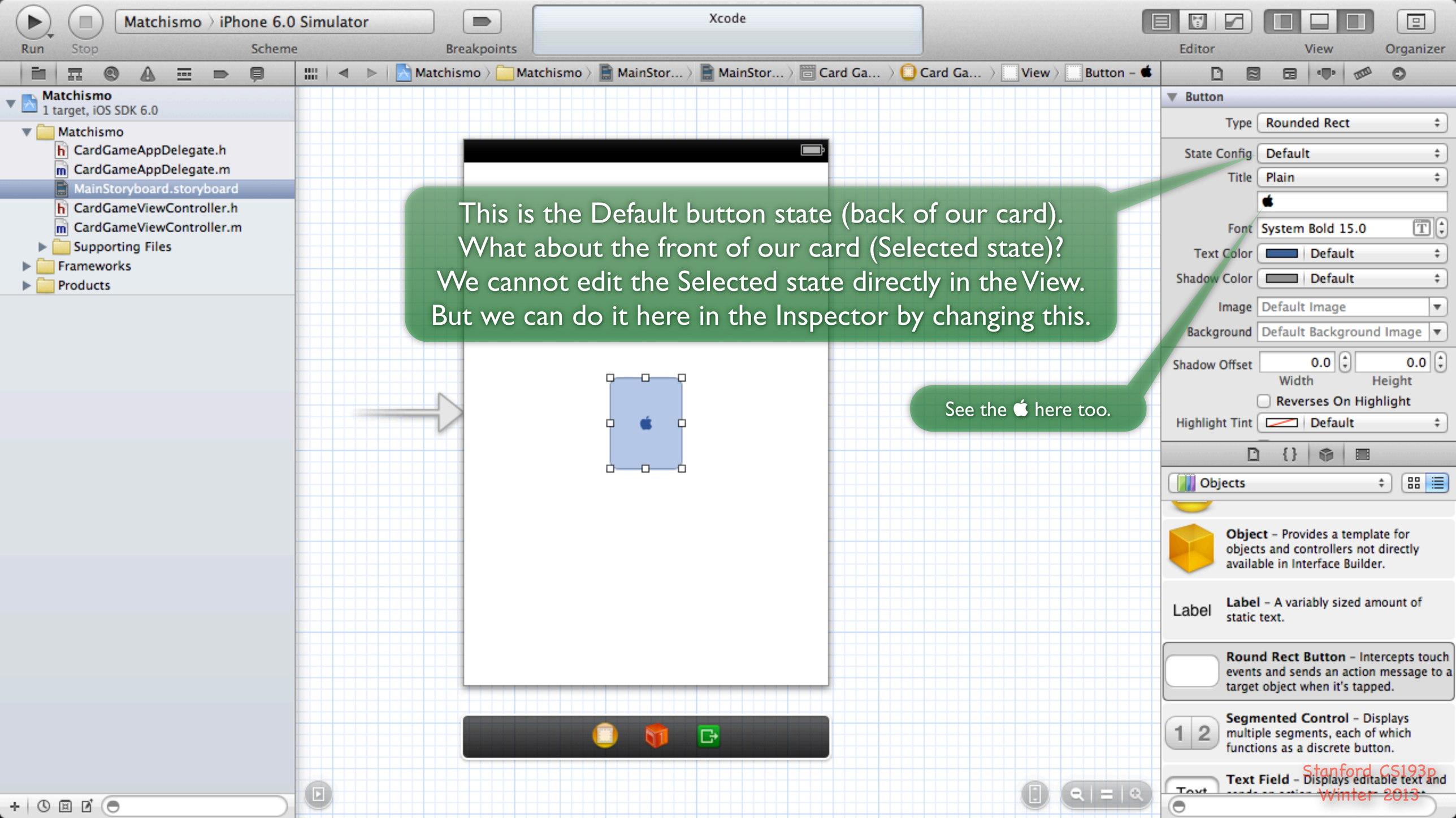

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Button – Button
Button
Type
Rounded Rect
State Config
Default
Title
Plain
Button
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Button
But our Button is no longer in the center of the View.
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

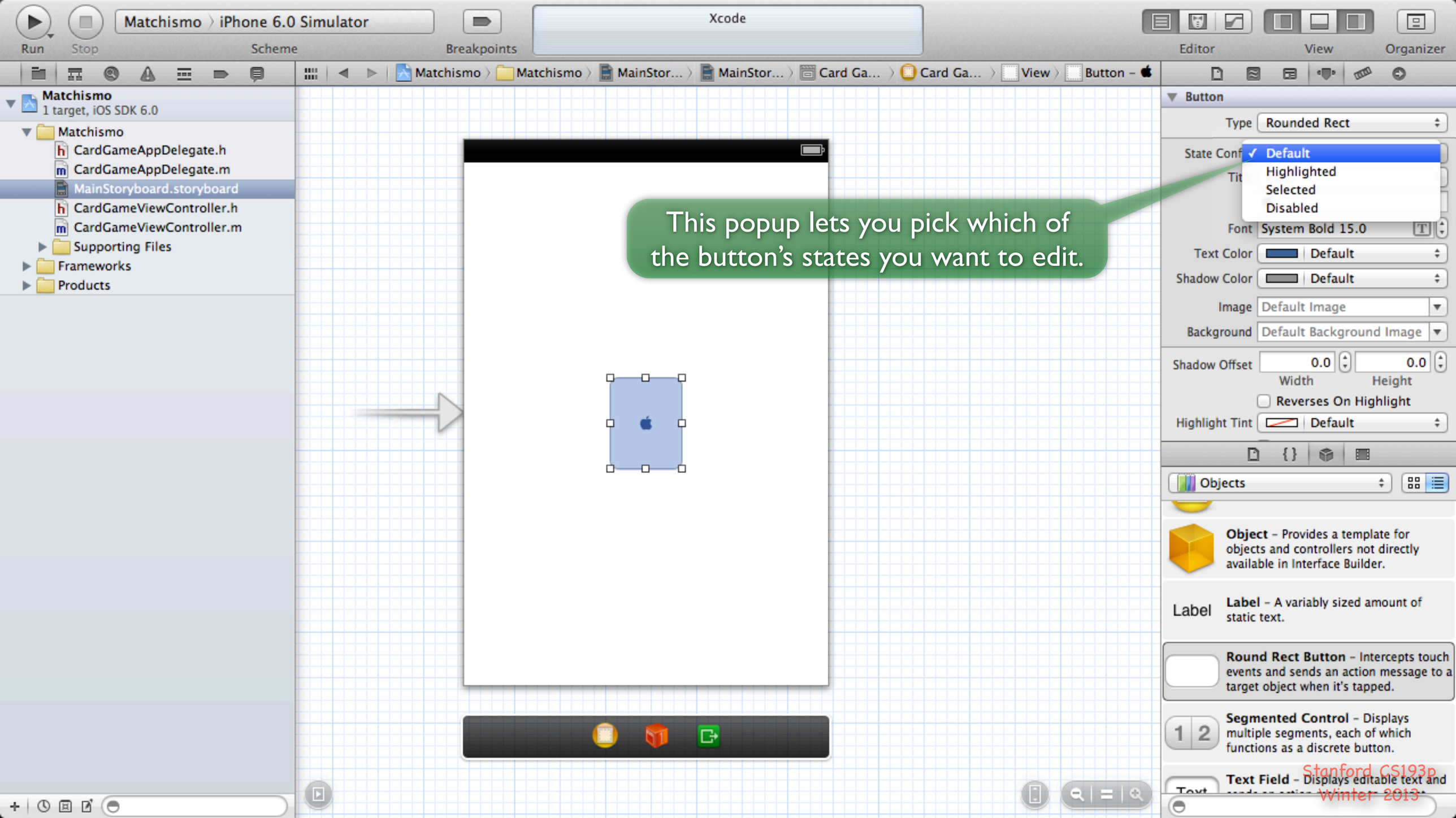

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
So just pick it up and drag it to the middle.
The guide lines will help you.
Button
Button
Type Rounded Rect
State Config Default
Title Plain
Button
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

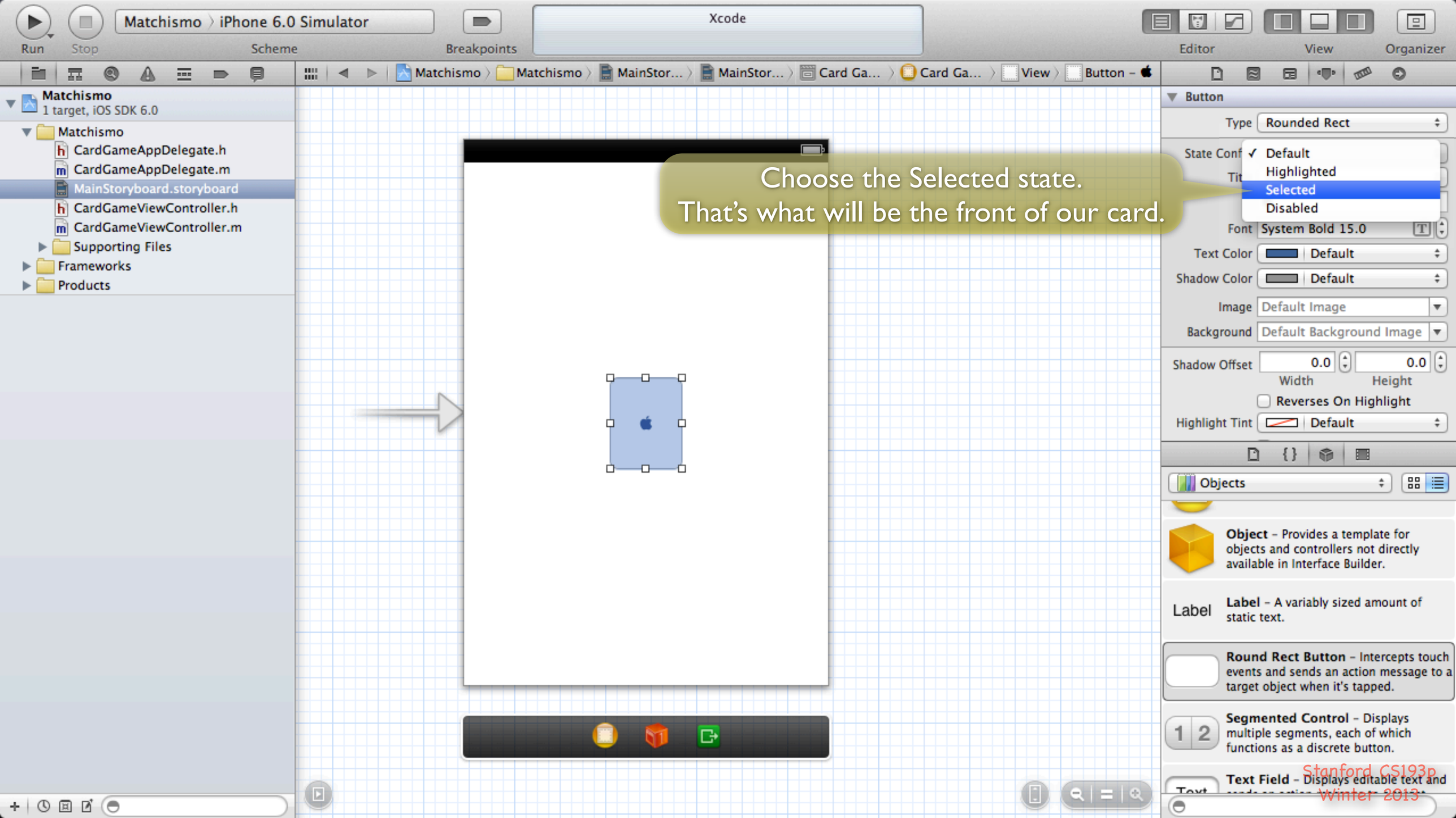

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
We're going to have the "back" of our card be its Default state and the "front" of the card be its Selected state.
Button
Xcode is showing the Default state in the View, so let's start by editing that.
Button
Type Rounded Rect
State Config Default
Title Plain
Button
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

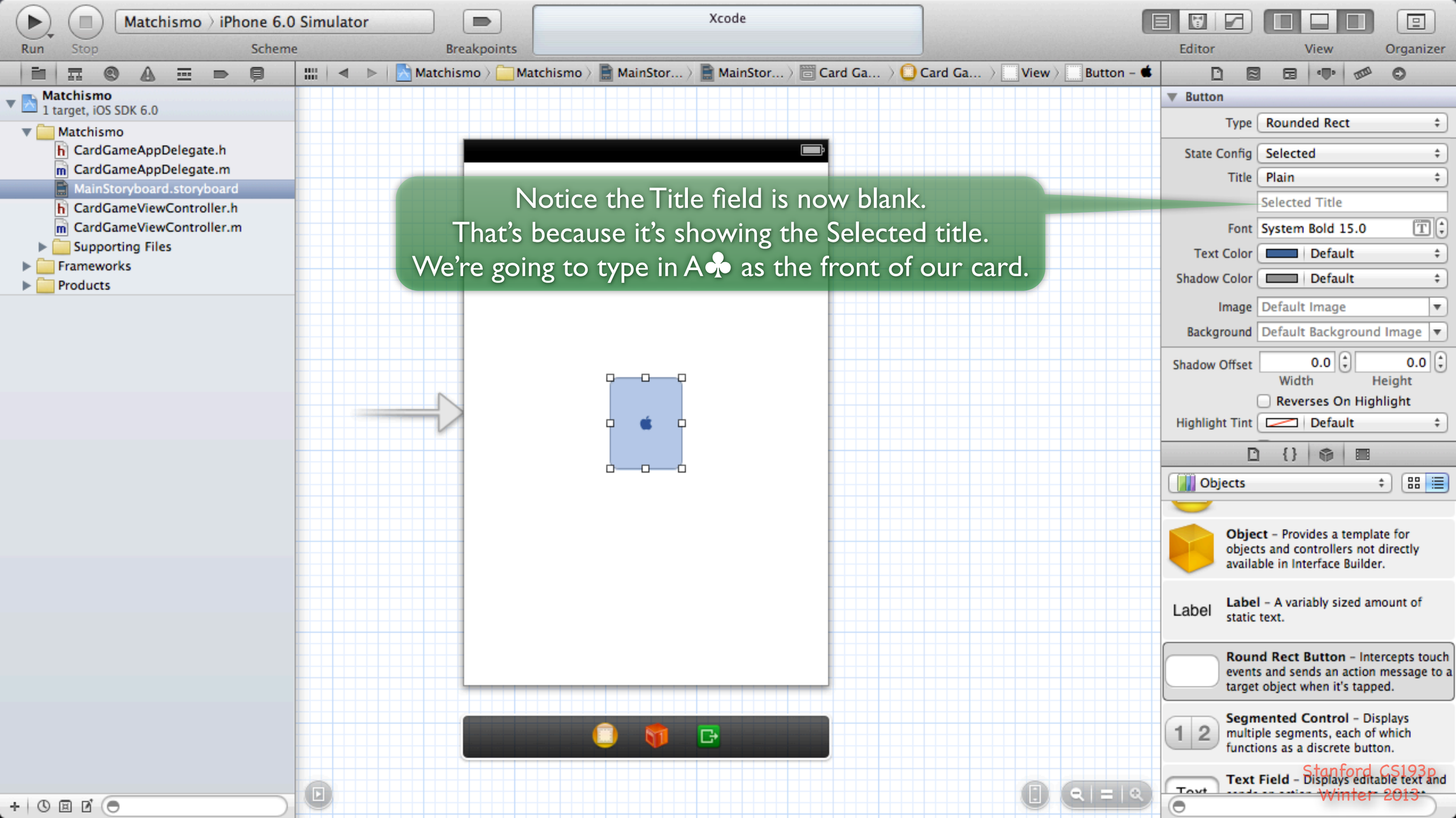

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
Double-click on the card to change the text on the back of our card.
You can also edit the button's text in the Attributes Inspector here.
Button
Button
Type
Rounded Rect
State Config
Default
Title
Plain
Button
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

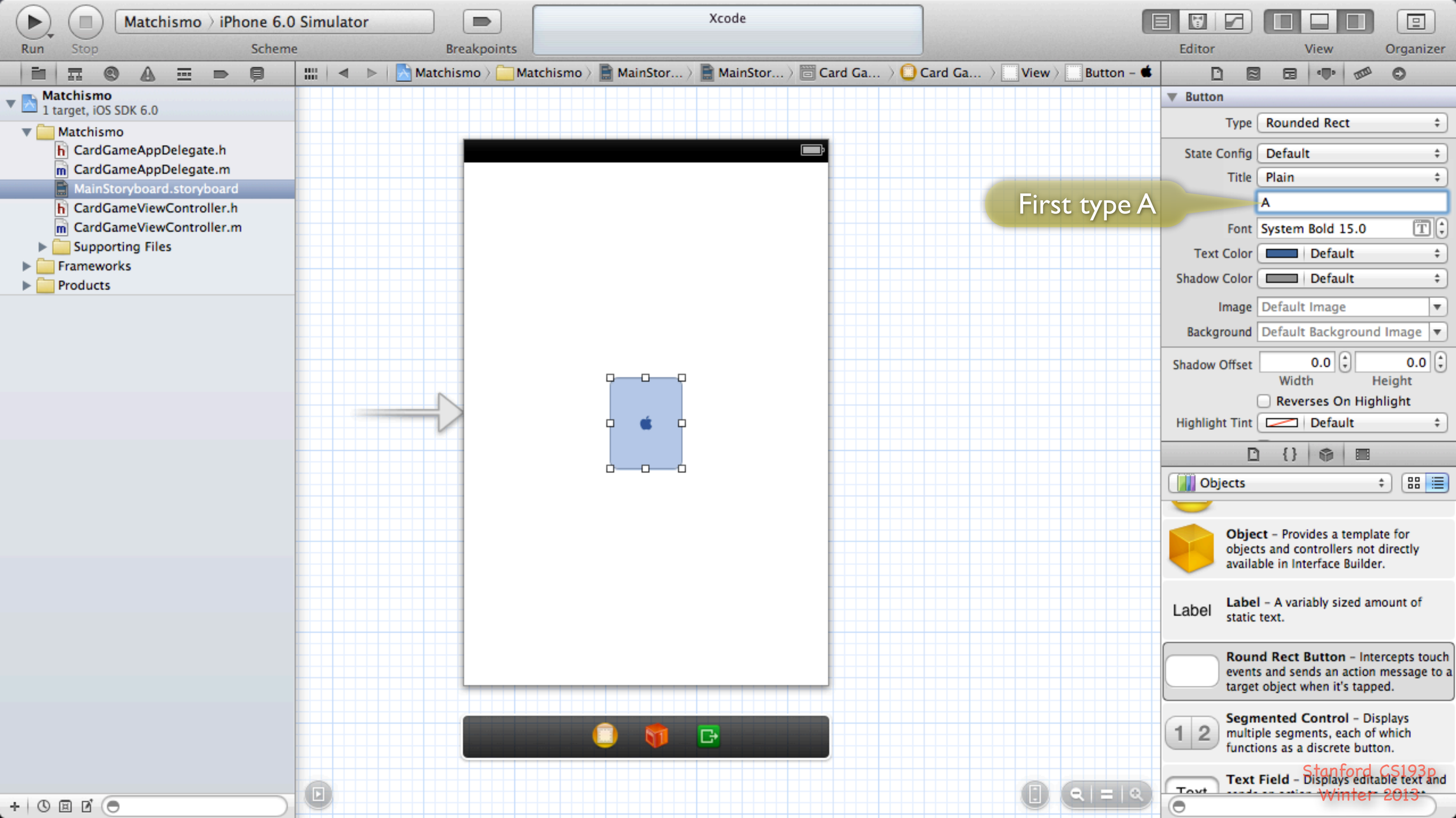

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchis... 〉 MainSto... 〉 MainSto... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button – Button
Let's make the back of our card be the Apple logo.
Believe it or not, there is a character for that: Alt-Shift-K.
When Alt, Shift or Ctrl are being pressed,
this little window here will show it.
⌥⇧K
Button
Type Rounded Rect
State Config Default
Title Plain
Button
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

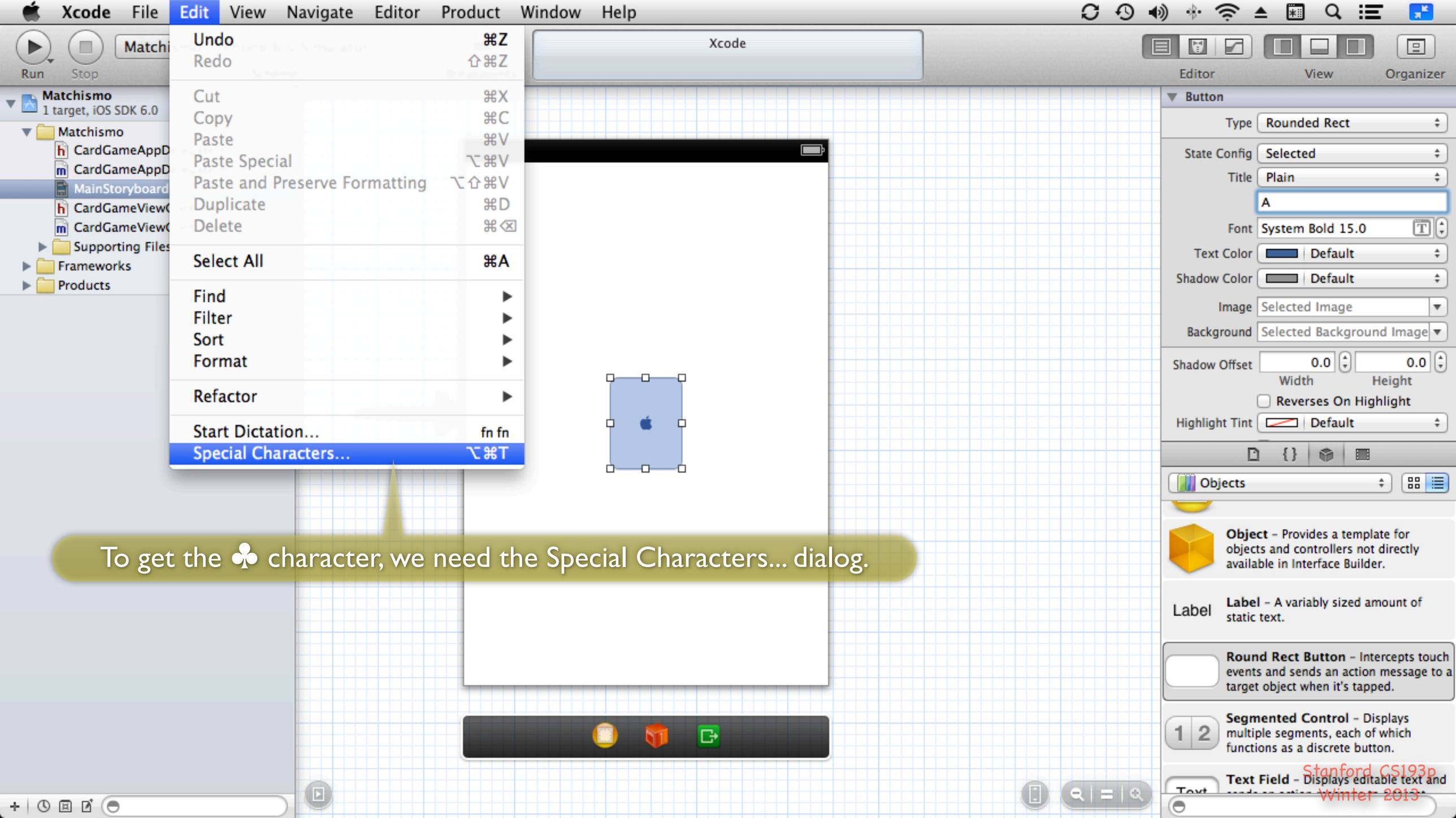

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button –
This is the Default button state (back of our card).
What about the front of our card (Selected state)?
We cannot edit the Selected state directly in the View.
But we can do it here in the Inspector by changing this.
See the here too.
Button
Type
Rounded Rect
State Config
Default
Title
Plain
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

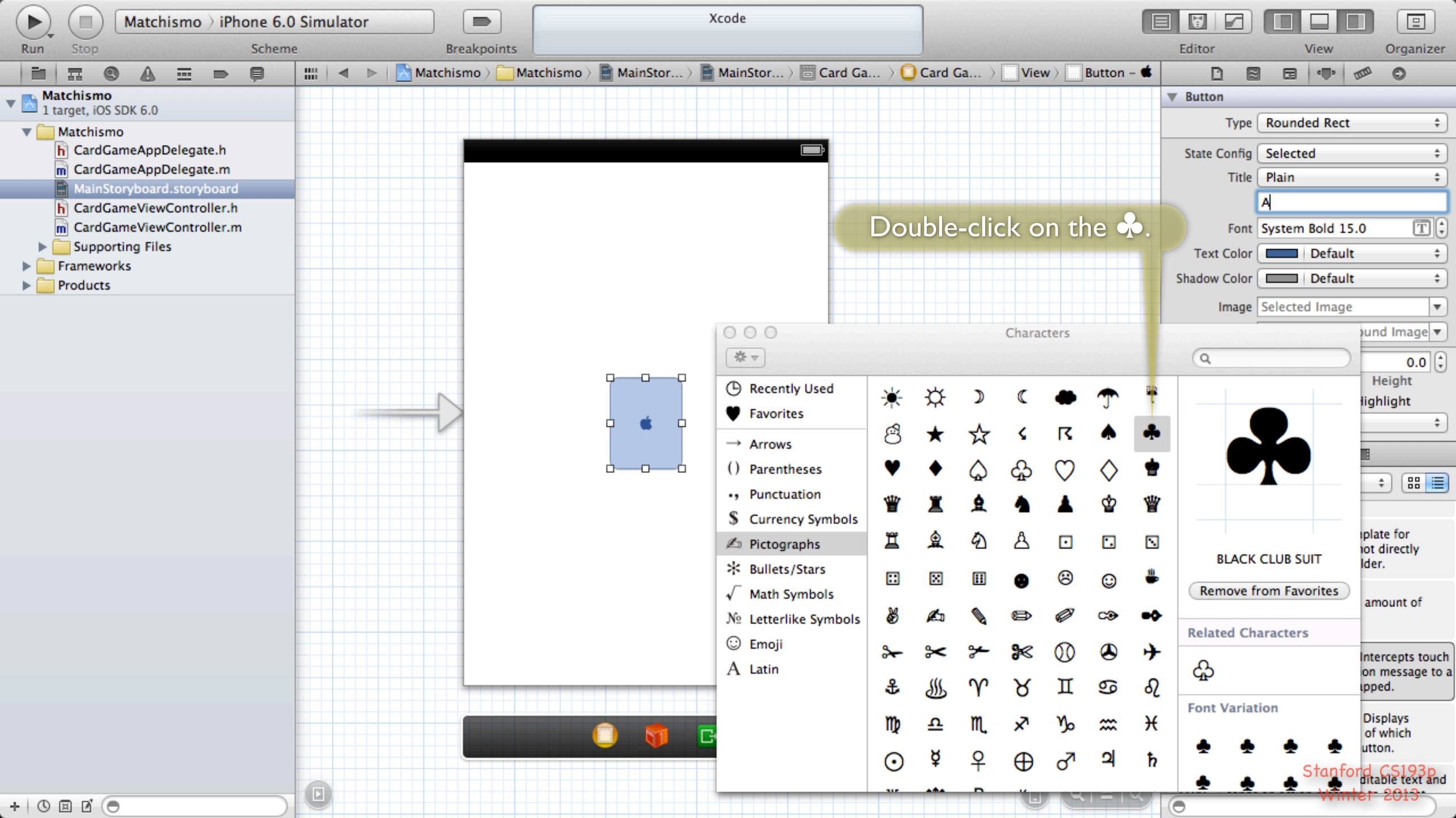

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
This popup lets you pick which of the button's states you want to edit.
Button
Type
Rounded Rect
State Conf
Default
Highlighted
Selected
Disabled
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

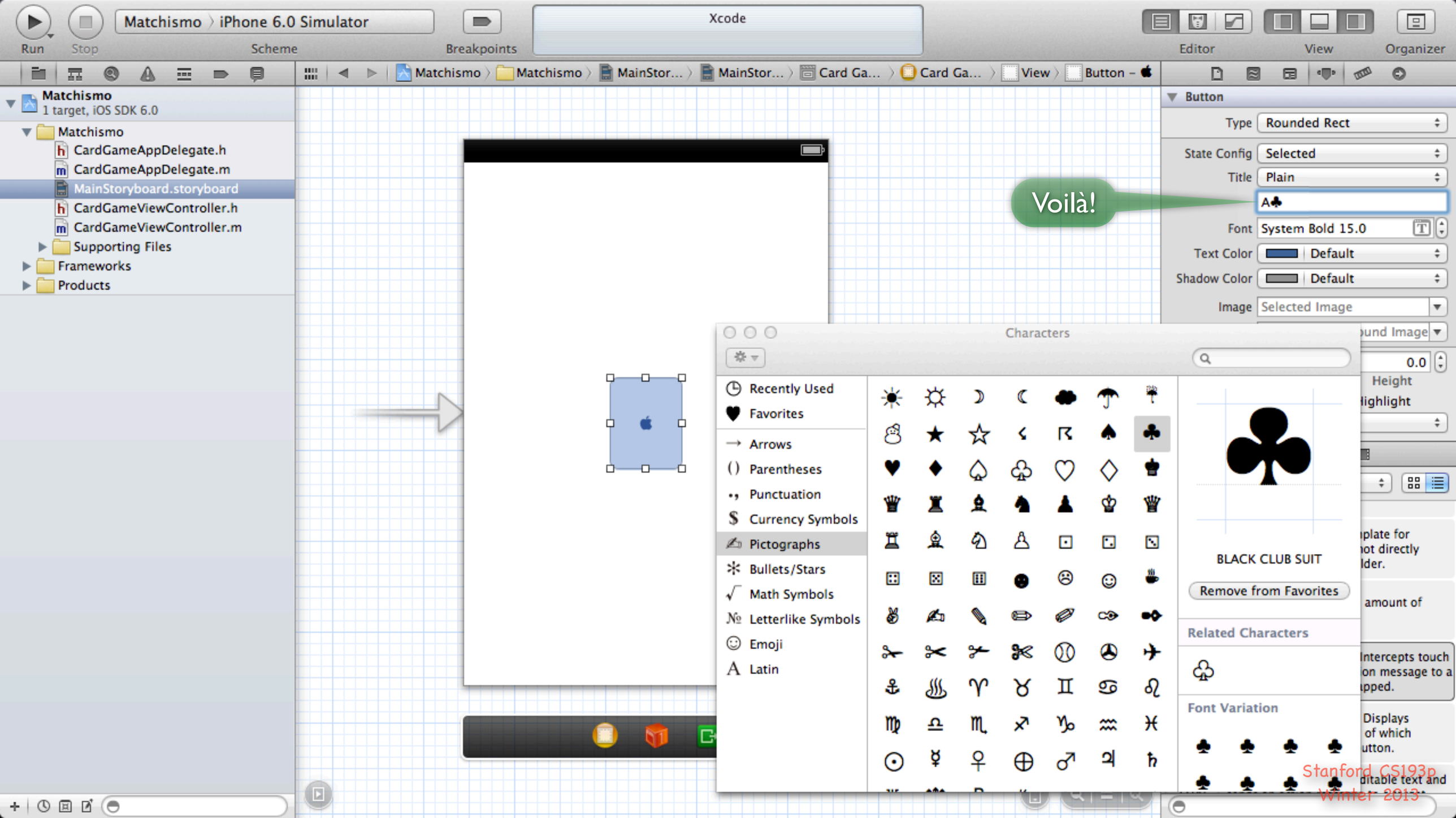

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Choose the Selected state.
That's what will be the front of our card.
Button
Type
Rounded Rect
State Conf
Default
Highlighted
Selected
Disabled
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

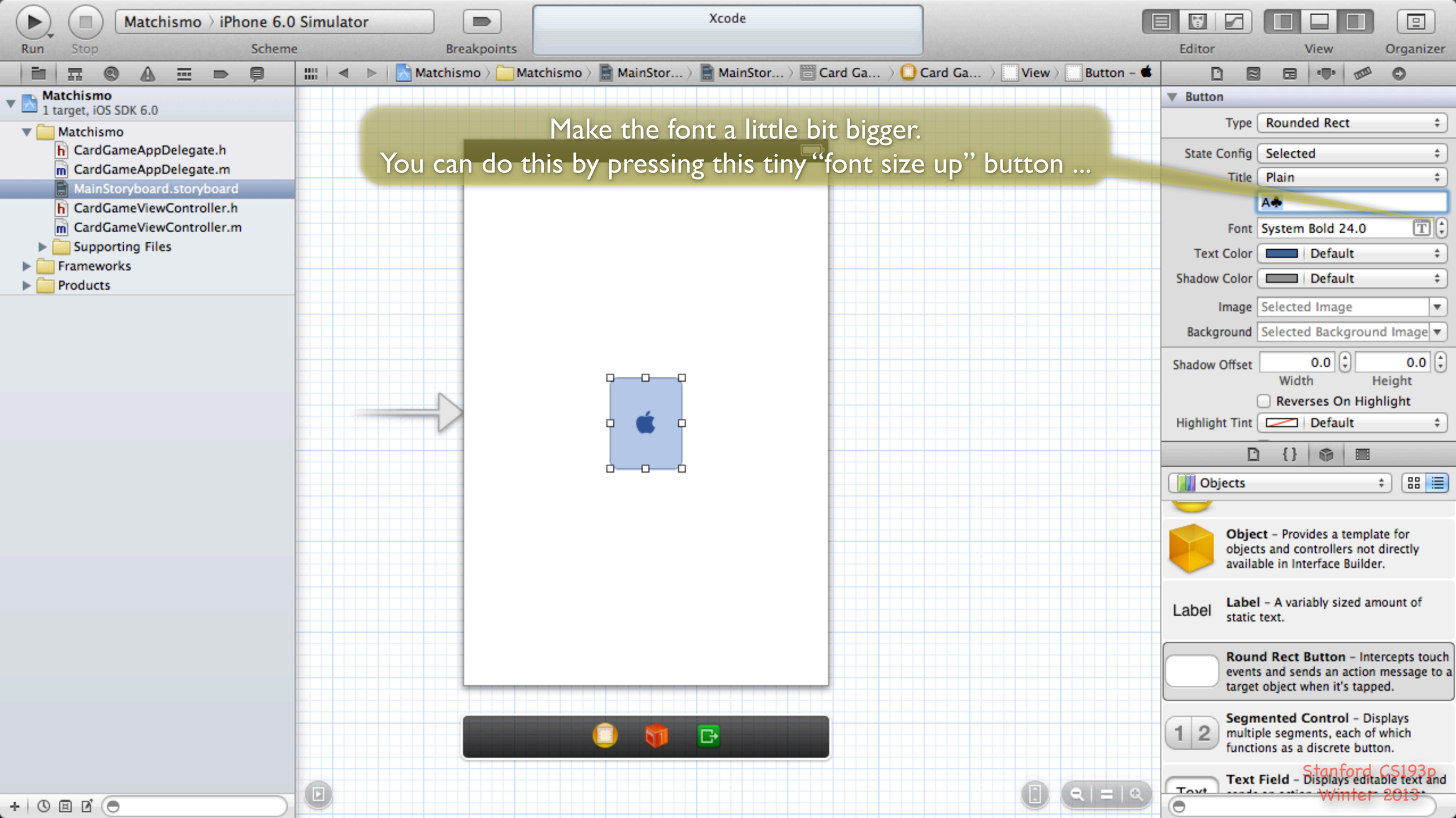

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Notice the Title field is now blank.
That's because it's showing the Selected title.
We're going to type in A♣ as the front of our card.
Button
Type Rounded Rect
State Config Selected
Title Plain
Selected Title
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

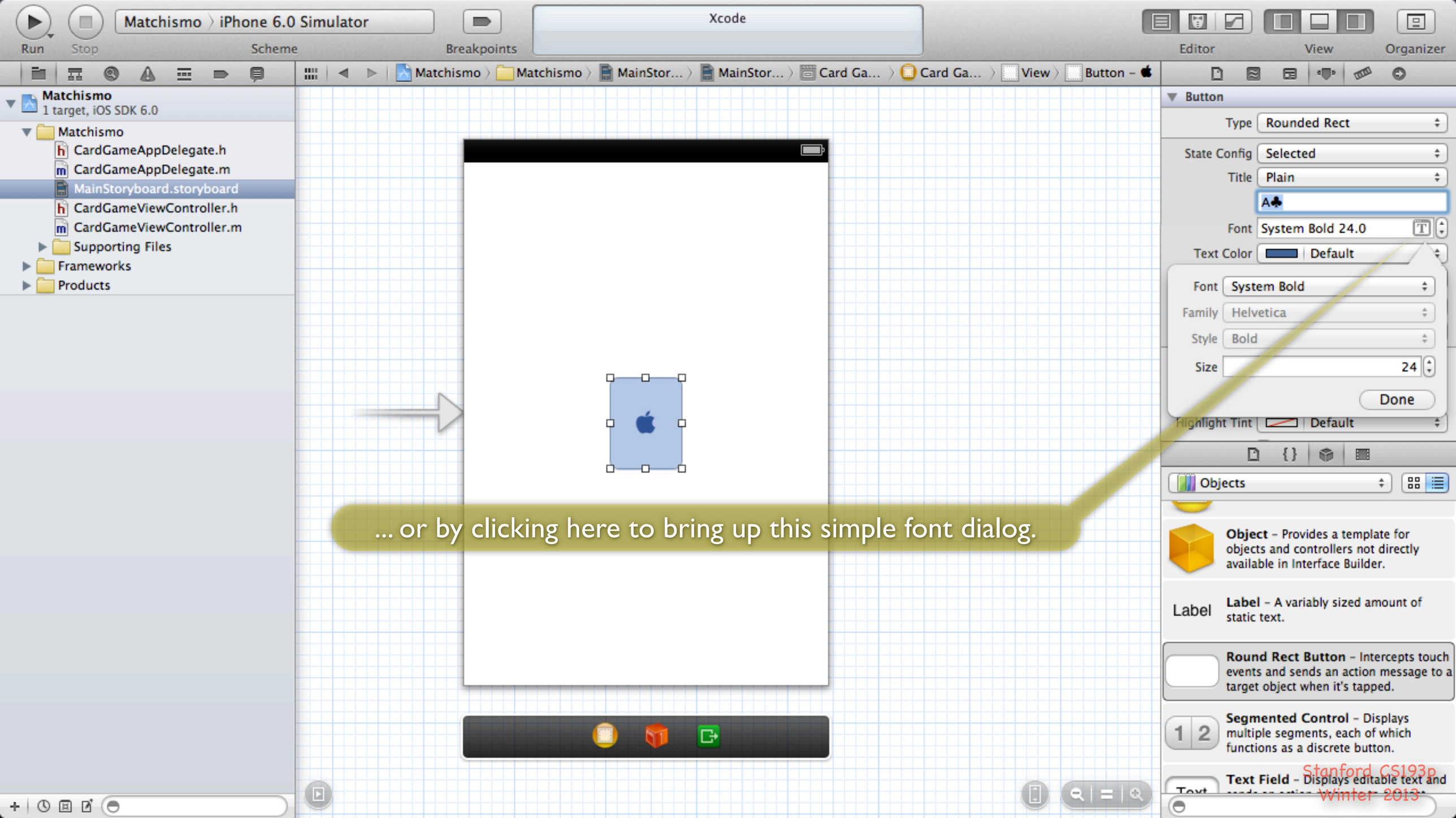

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
First type A
Button
Type
Rounded Rect
State Config
Default
Title
Plain
A
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

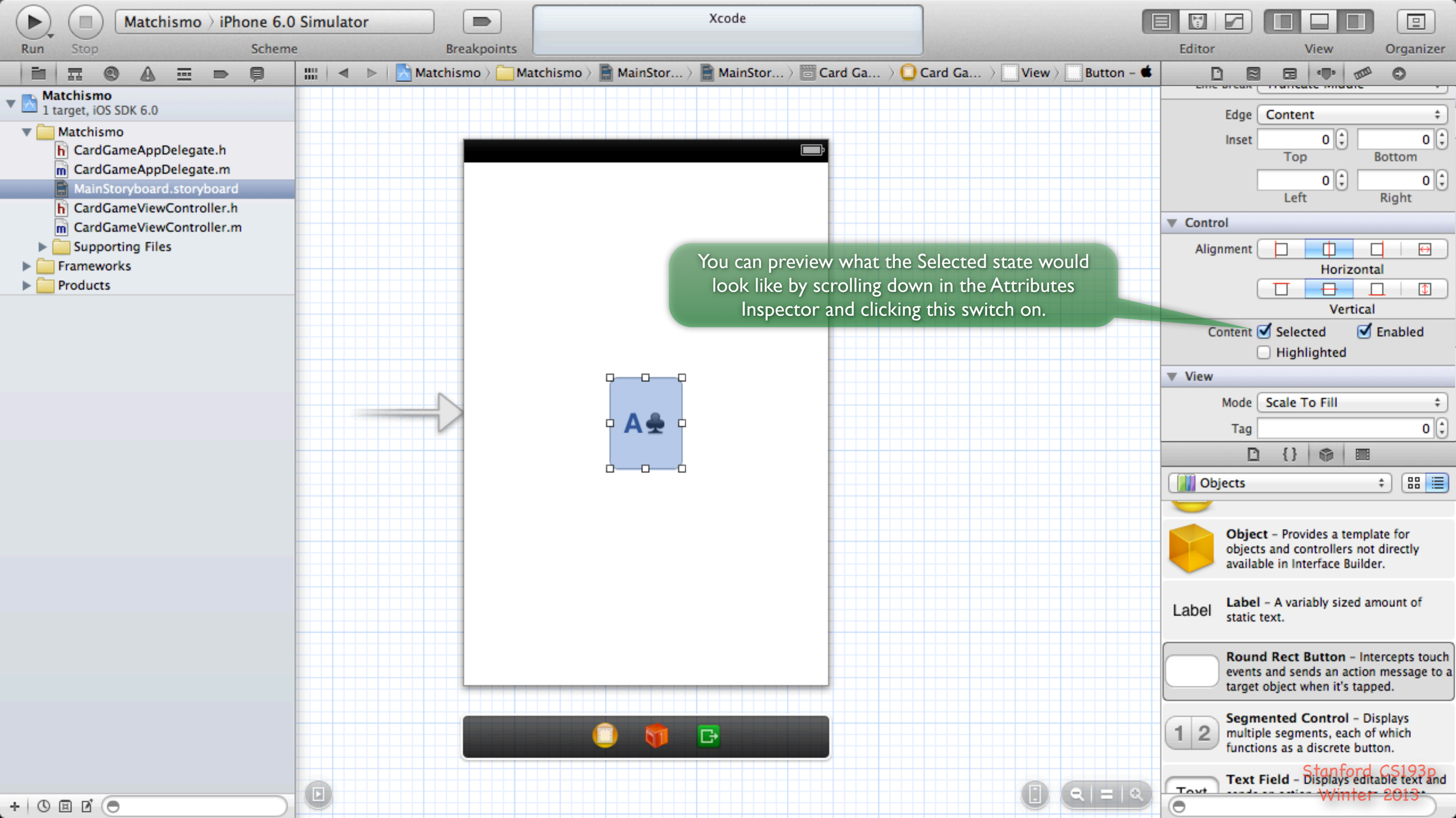

Xcode File Edit View Navigate Editor Product Window Help
Undo ⌘Z
Redo ⇧⌘Z
Cut ⌘X
Copy ⌘C
Paste ⌘V
Paste Special ⌥⌘V
Paste and Preserve Formatting ⌥⇧⌘V
Duplicate ⌘D
Delete ⌘⌫
Select All ⌘A
Find
Filter
Sort
Format
Refactor
Start Dictation... fn fn
Special Characters... ⌥⌘T
Run
Stop
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppD
CardGameAppD
MainStoryboard
CardGameView
CardGameView
Supporting Files
Frameworks
Products
Button
Type Rounded Rect
State Config Selected
Title Plain
A
Font System Bold 15.0
Text Color Default
Shadow Color Default
Image Selected Image
Background Selected Background Image
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Highlight Tint Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
To get the ♣ character, we need the Special Characters... dialog.
Stanford CS193p
Winter 2013

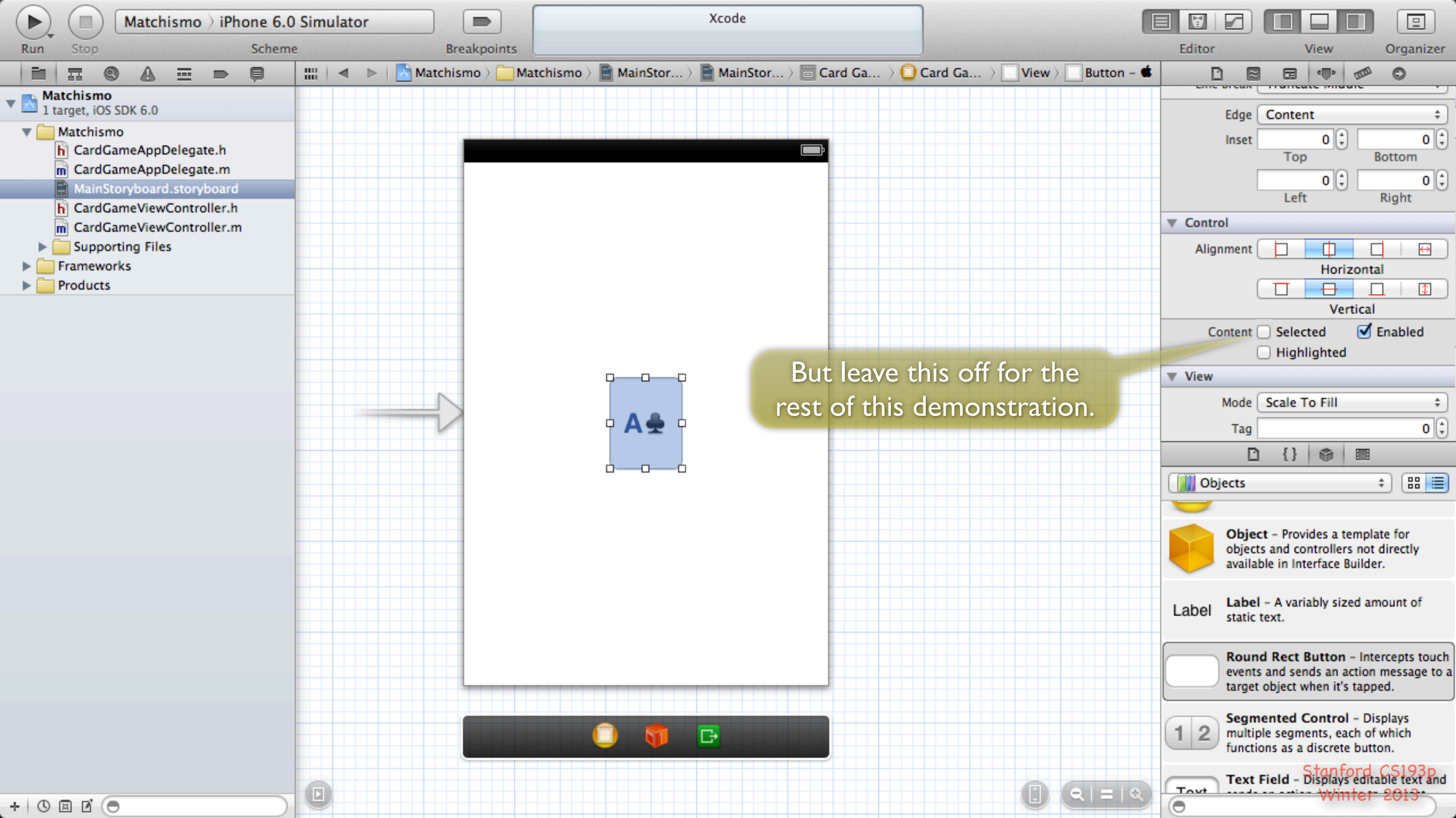

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Double-click on the ♣.
Characters
Recently Used
Favorites
Arrows
Parentheses
Punctuation
Currency Symbols
Pictographs
Bullets/Stars
Math Symbols
Letterlike Symbols
Emoji
Latin
BLACK CLUB SUIT
Remove from Favorites
Related Characters
Font Variation
Height
Stanford CS193p
Winter 2013

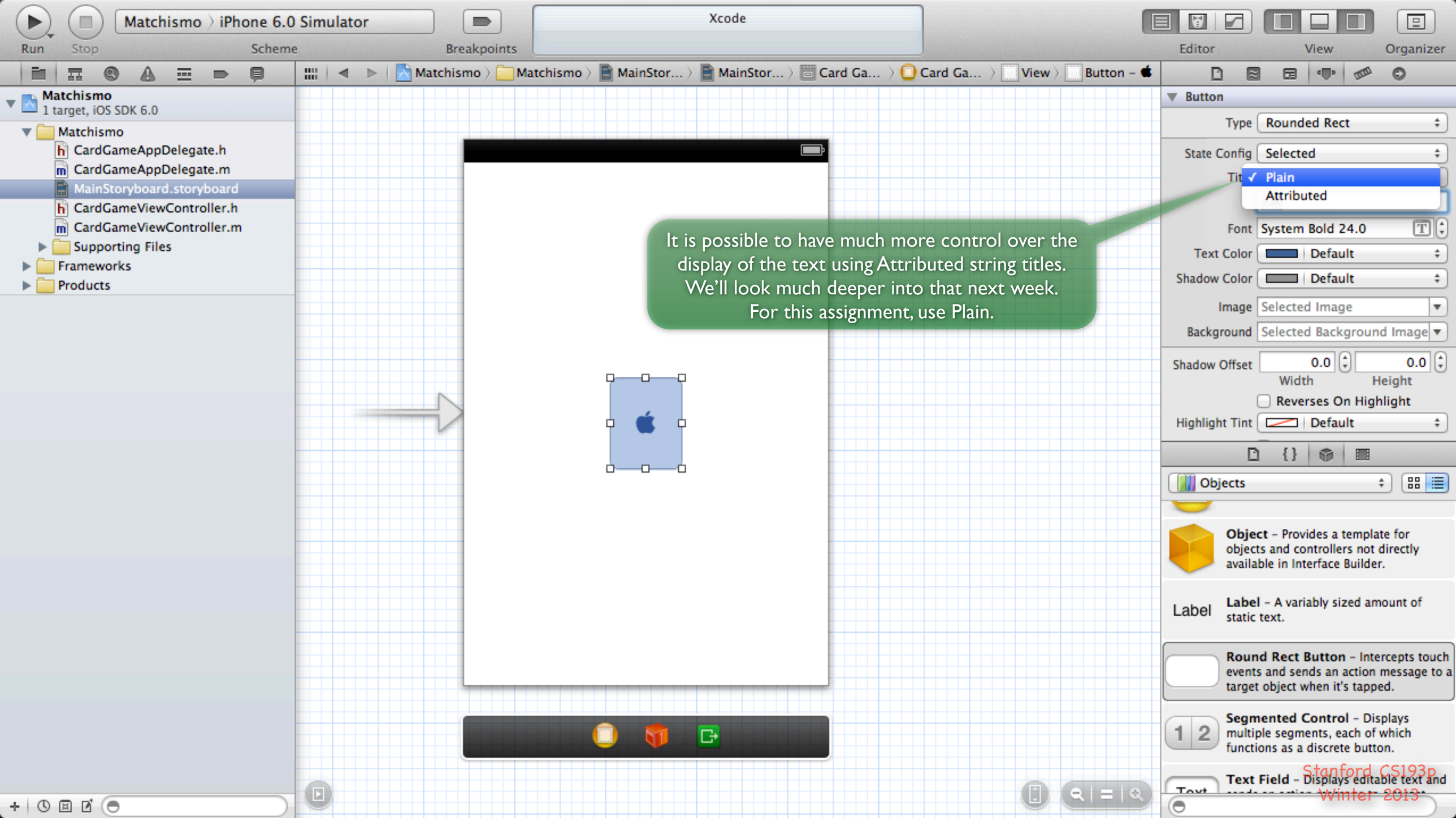

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A♣
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Voilà!
Characters
Recently Used
Favorites
Arrows
Parentheses
Punctuation
Currency Symbols
Pictographs
Bullets/Stars
Math Symbols
Letterlike Symbols
Emoji
Latin
BLACK CLUB SUIT
Remove from Favorites
Related Characters
Font Variation
Height
Stanford CS193p
Winter 2013

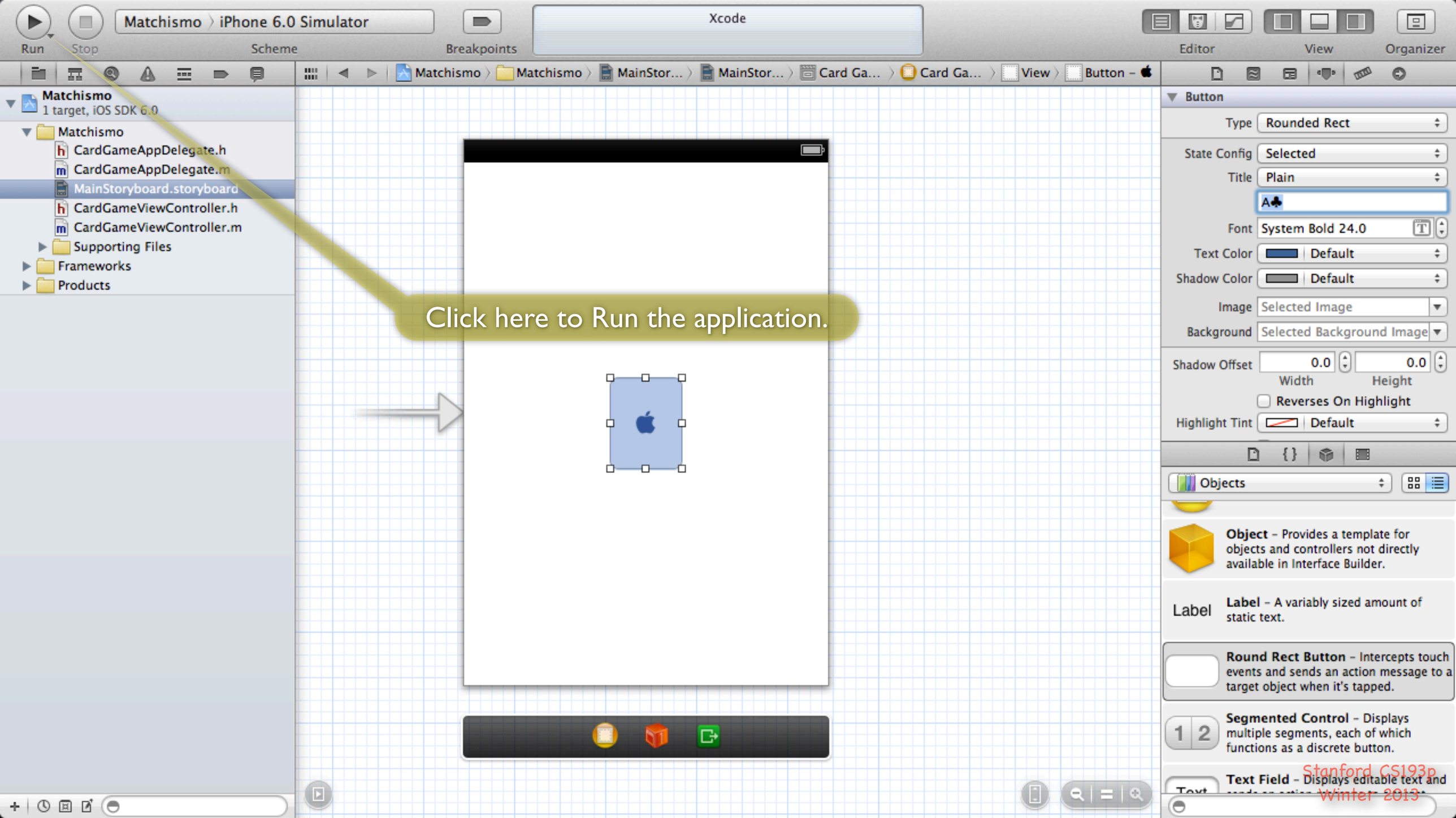

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Make the font a little bit bigger.
You can do this by pressing this tiny "font size up" button ...
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A♣
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

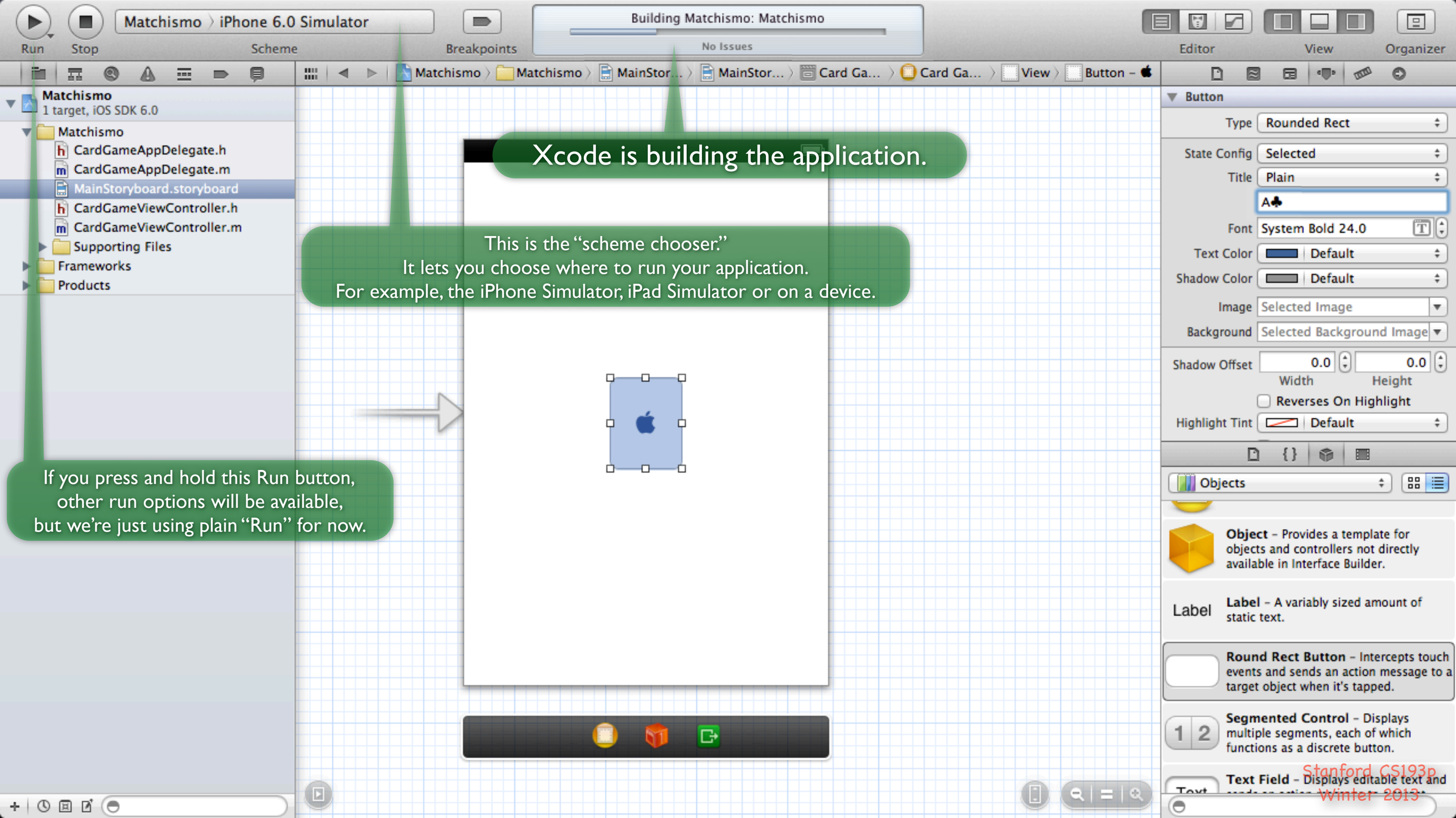

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A♣
Font
System Bold 24.0
Text Color
Default
Font
System Bold
Family
Helvetica
Style
Bold
Size
24
Done
Highlight Tint
Default
... or by clicking here to bring up this simple font dialog.
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

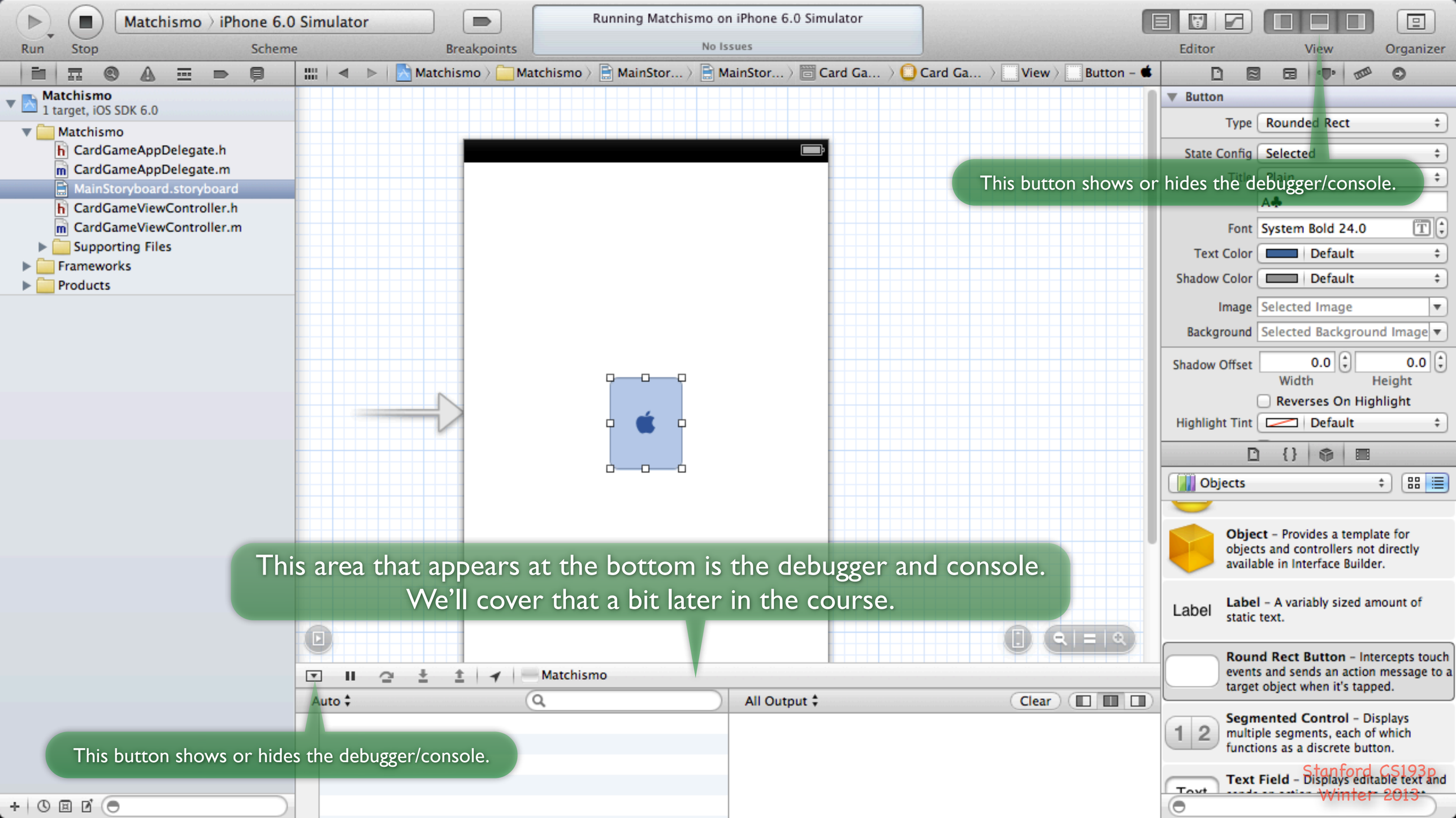

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
You can preview what the Selected state would look like by scrolling down in the Attributes Inspector and clicking this switch on.
Edge
Content
Inset
0
Top
Bottom
Left
Right
Control
Alignment
Horizontal
Vertical
Content
Selected
Enabled
Highlighted
View
Mode
Scale To Fill
Tag
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

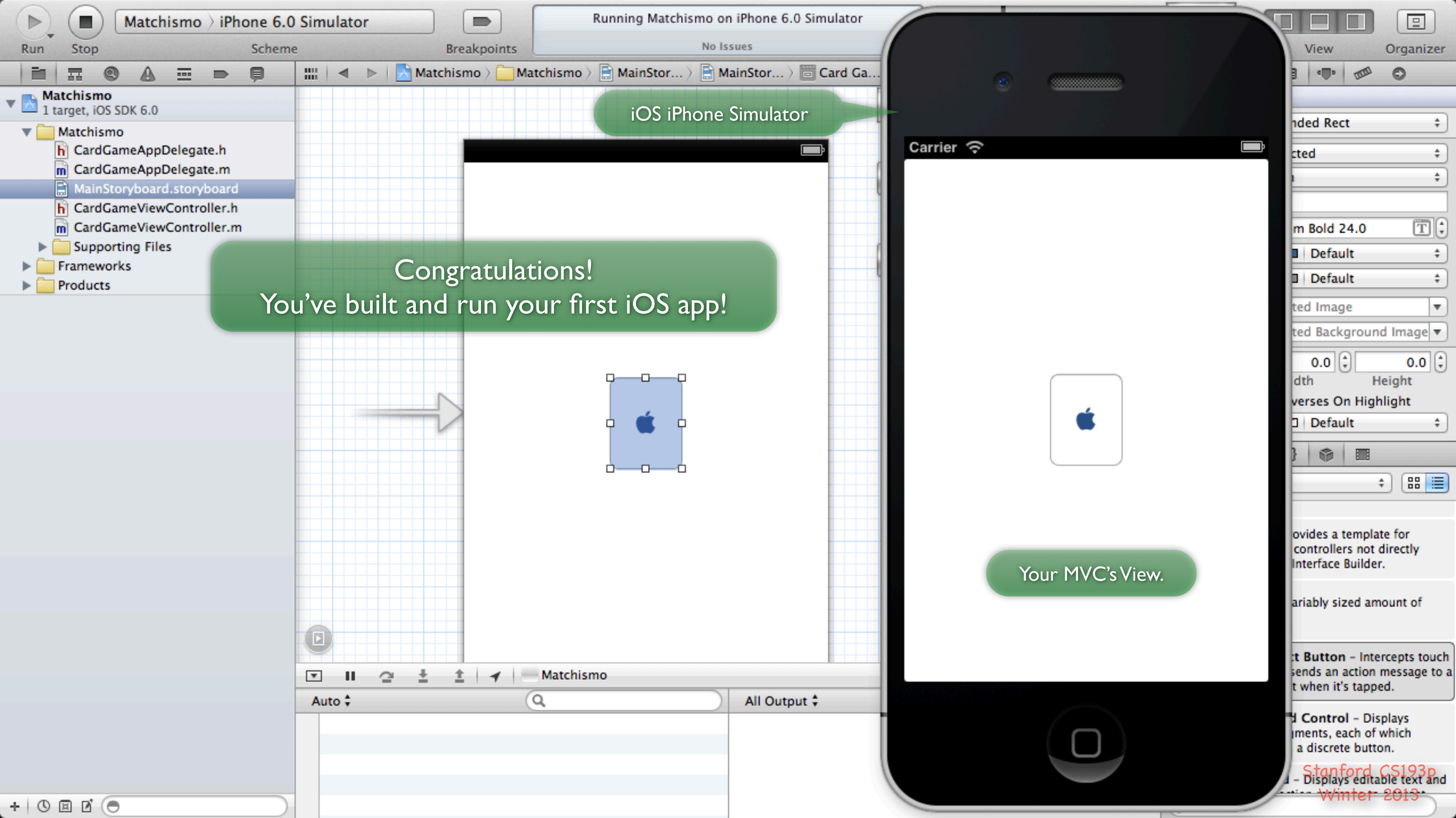

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
But leave this off for the rest of this demonstration.
Edge
Content
Inset
Top
Bottom
Left
Right
Control
Alignment
Horizontal
Vertical
Content
Selected
Enabled
Highlighted
View
Mode
Scale To Fill
Tag
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

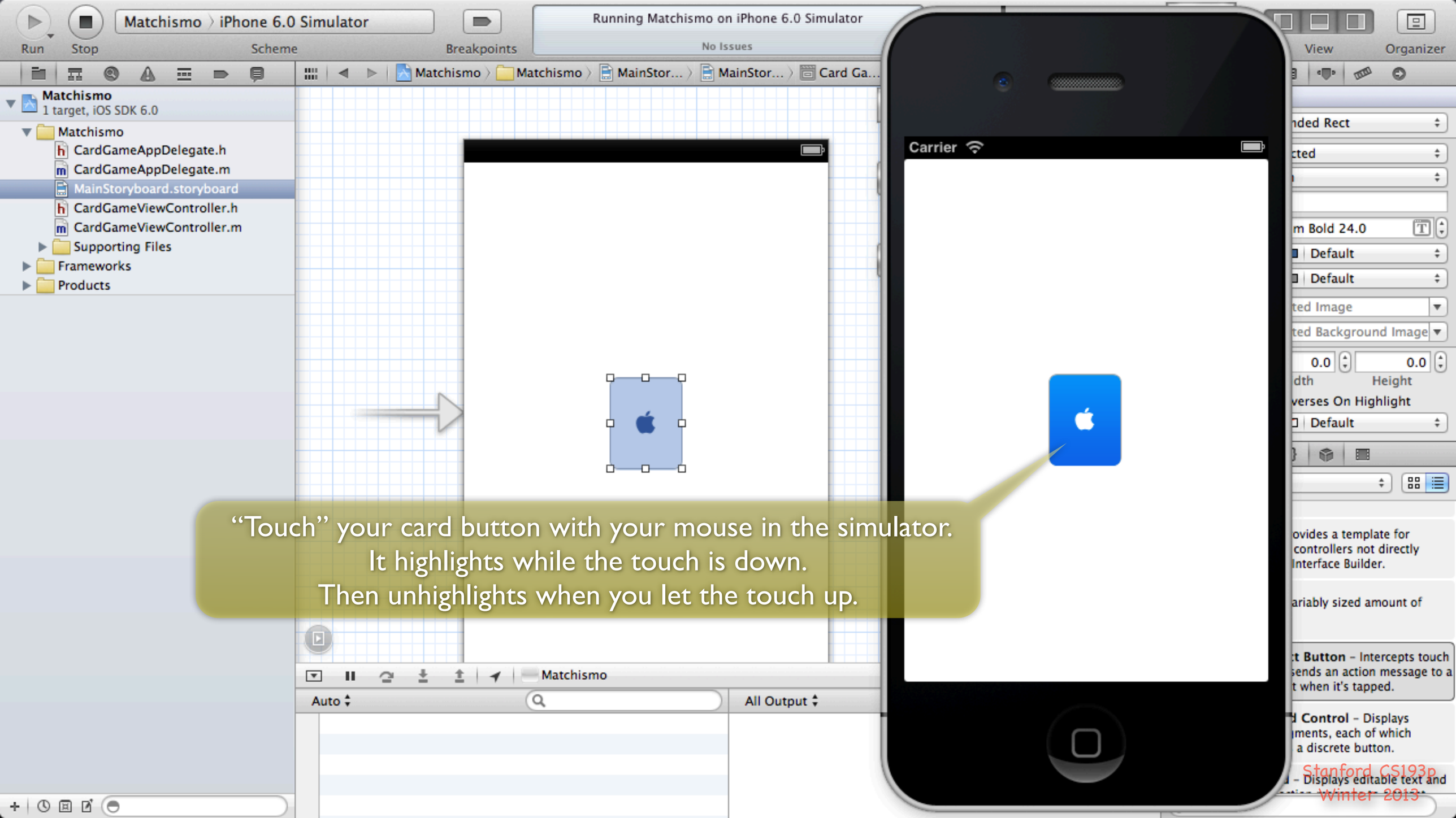

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Button
Type
Rounded Rect
State Config
Selected
Plain
Attributed
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
It is possible to have much more control over the display of the text using Attributed string titles. We'll look much deeper into that next week. For this assignment, use Plain.
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

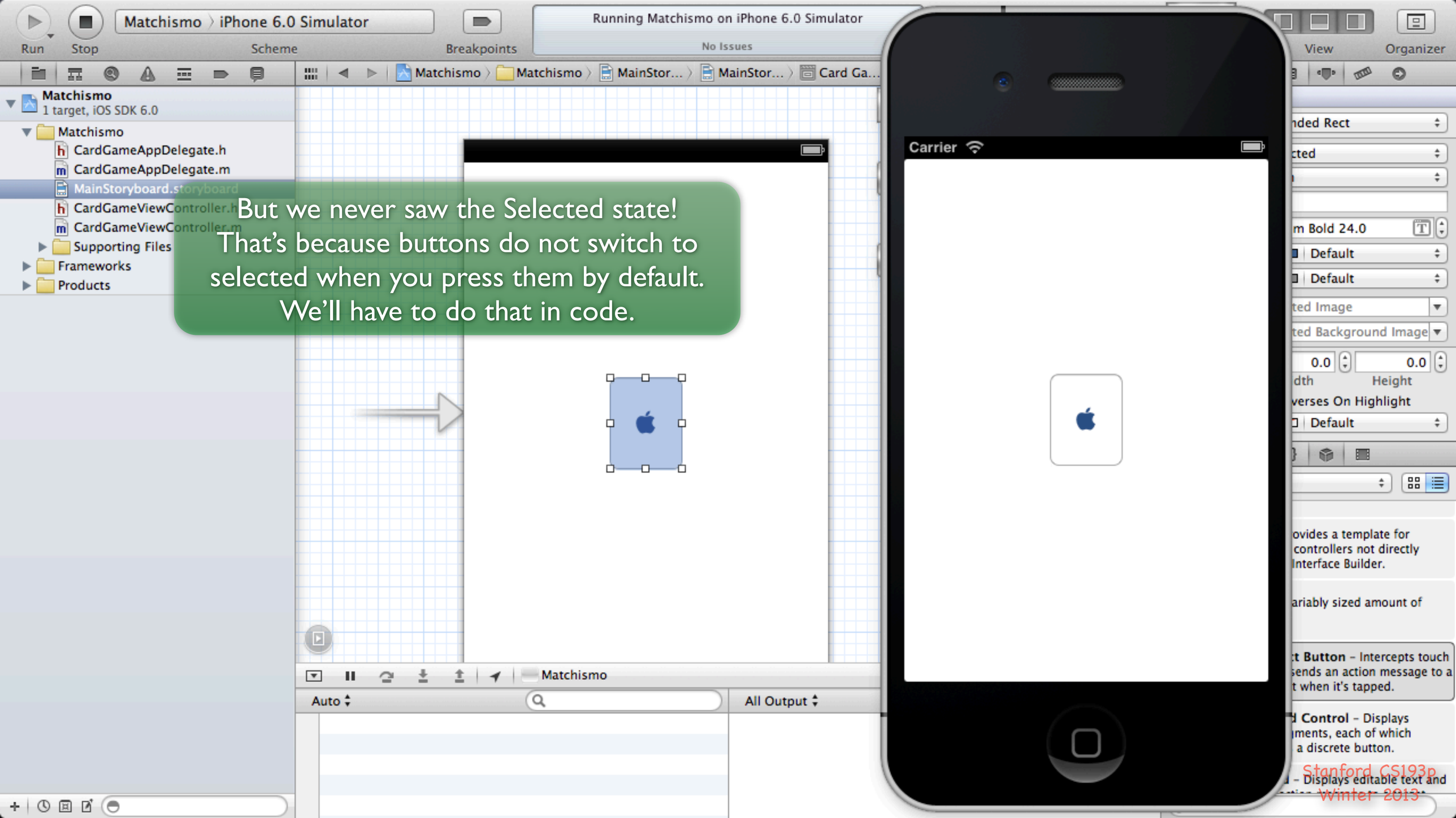

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Xcode
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Click here to Run the application.
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A♣
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

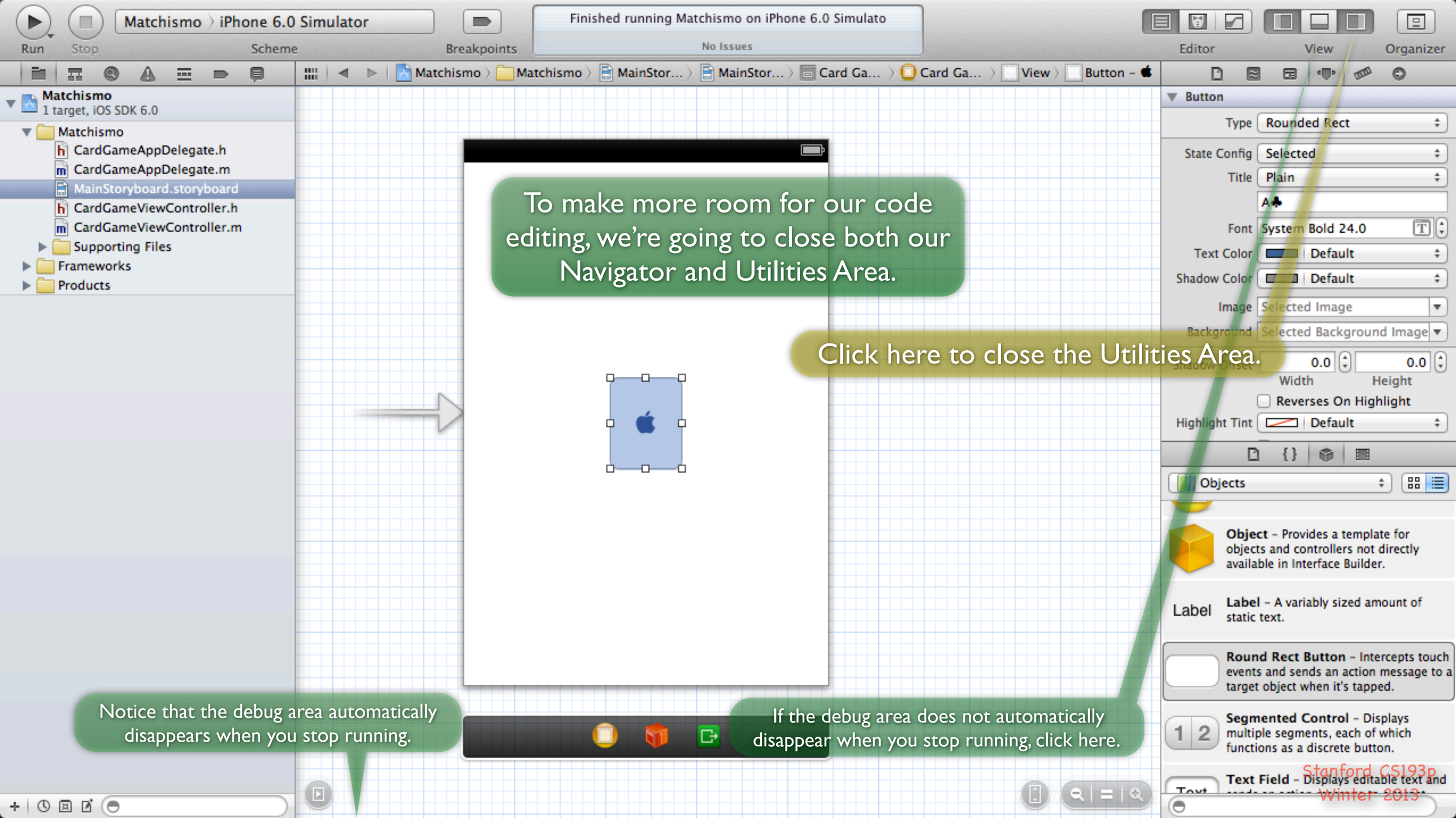

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Building Matchismo: Matchismo
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Xcode is building the application.
This is the "scheme chooser."
It lets you choose where to run your application.
For example, the iPhone Simulator, iPad Simulator or on a device.
If you press and hold this Run button,
other run options will be available,
but we're just using plain "Run" for now.
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
A♣
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

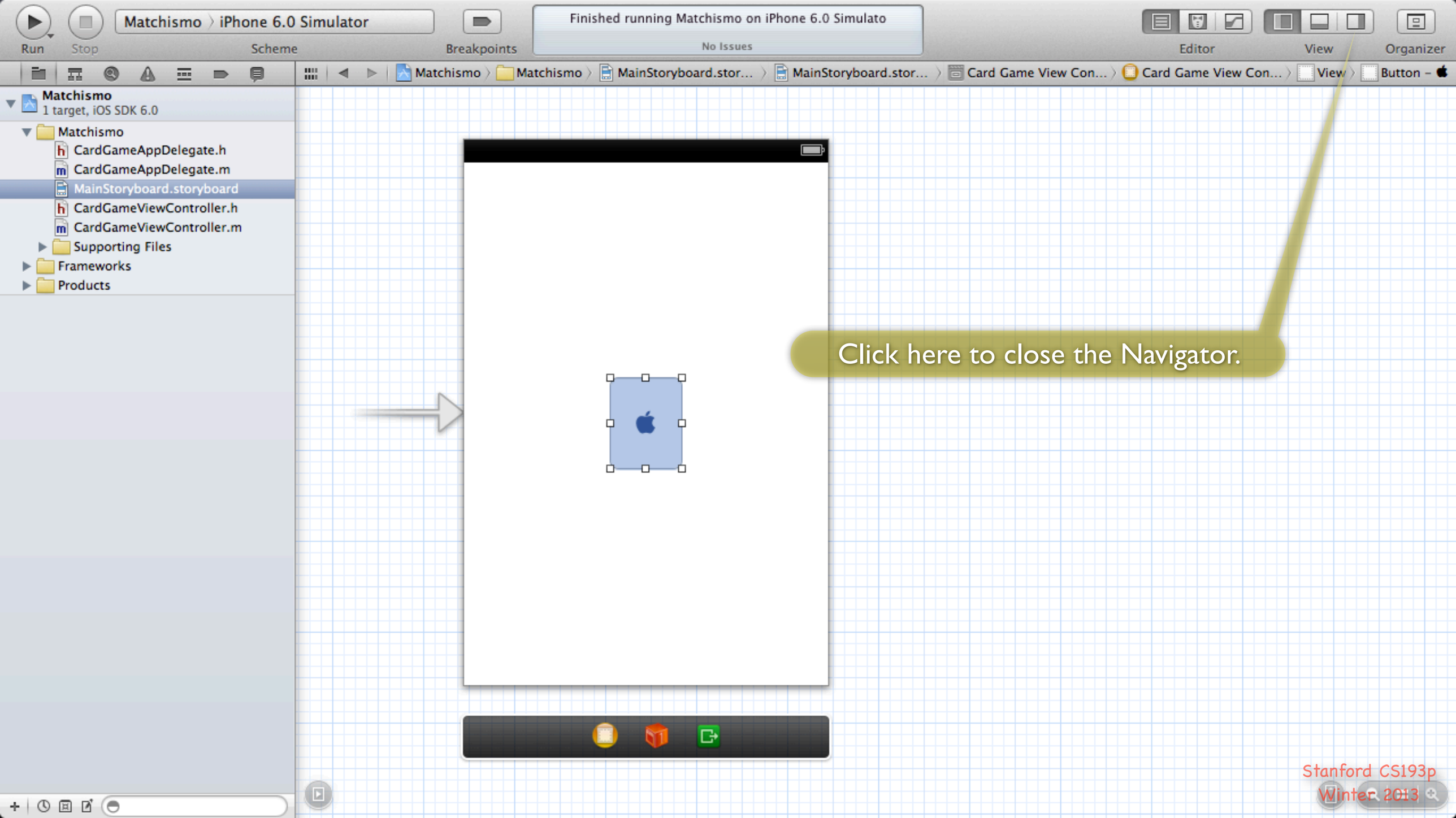

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
Button
Type
Rounded Rect
State Config
Selected
This button shows or hides the debugger/console.
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
This area that appears at the bottom is the debugger and console.
We'll cover that a bit later in the course.
Matchismo
Auto
All Output
Clear
This button shows or hides the debugger/console.
Stanford CS193p
Winter 2013

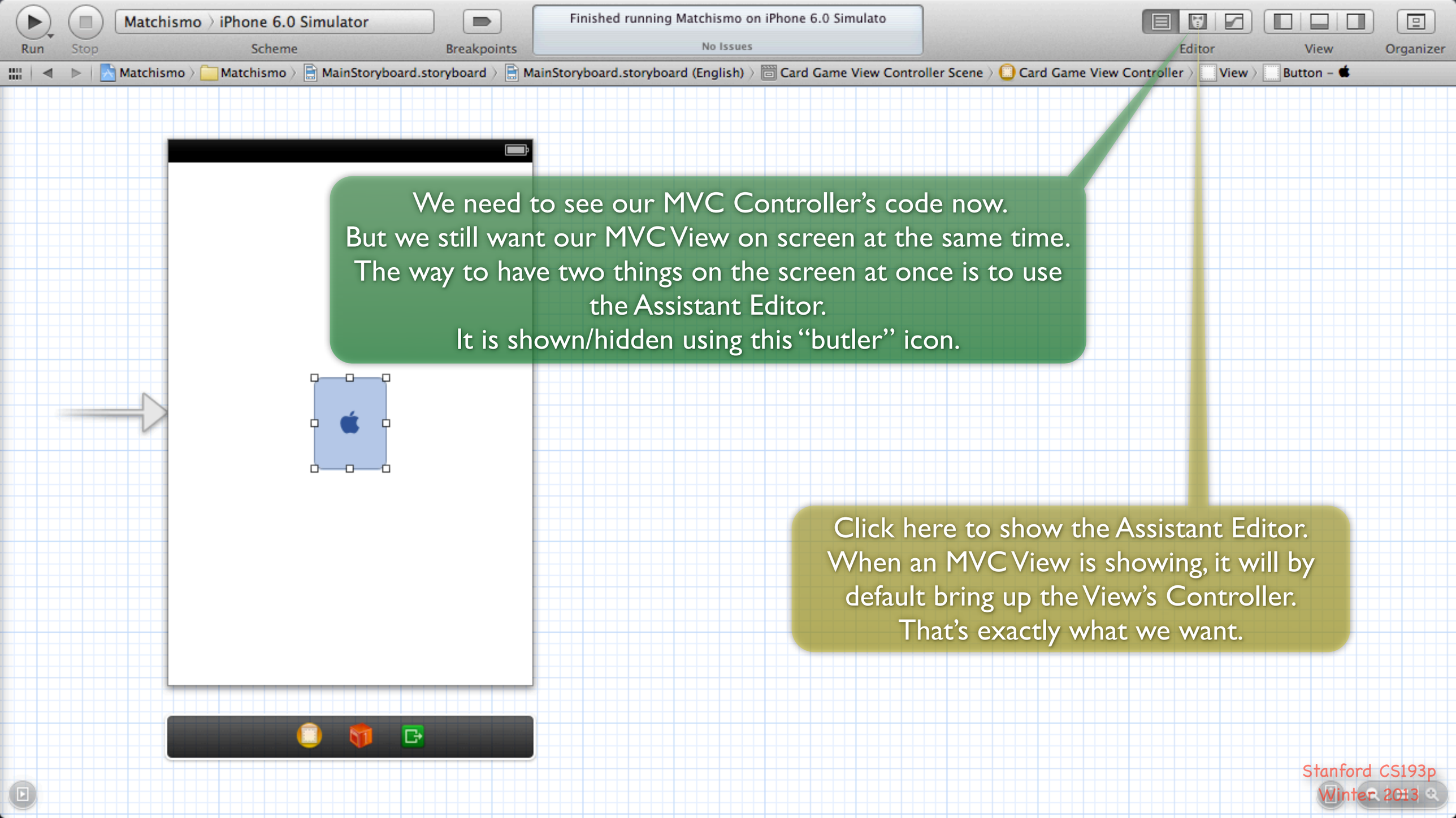

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga...
iOS iPhone Simulator
Congratulations!
You've built and run your first iOS app!
Carrier
Your MVC's View.
Matchismo
Auto
All Output
Stanford CS193p
Winter 2013

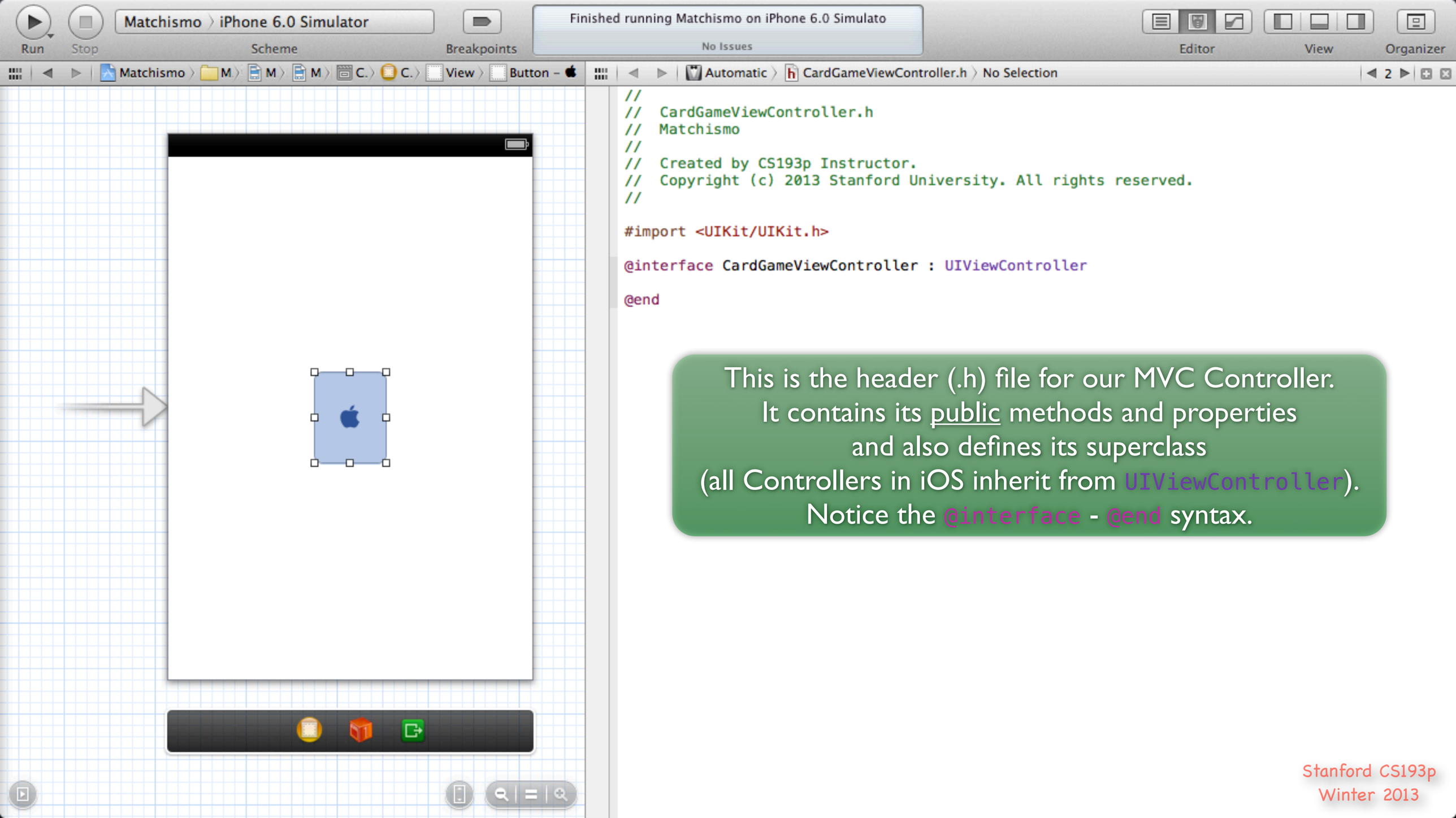

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Carrier
"Touch" your card button with your mouse in the simulator.
It highlights while the touch is down.
Then unhighlights when you let the touch up.
Auto
All Output
Stanford CS193p
Winter 2013

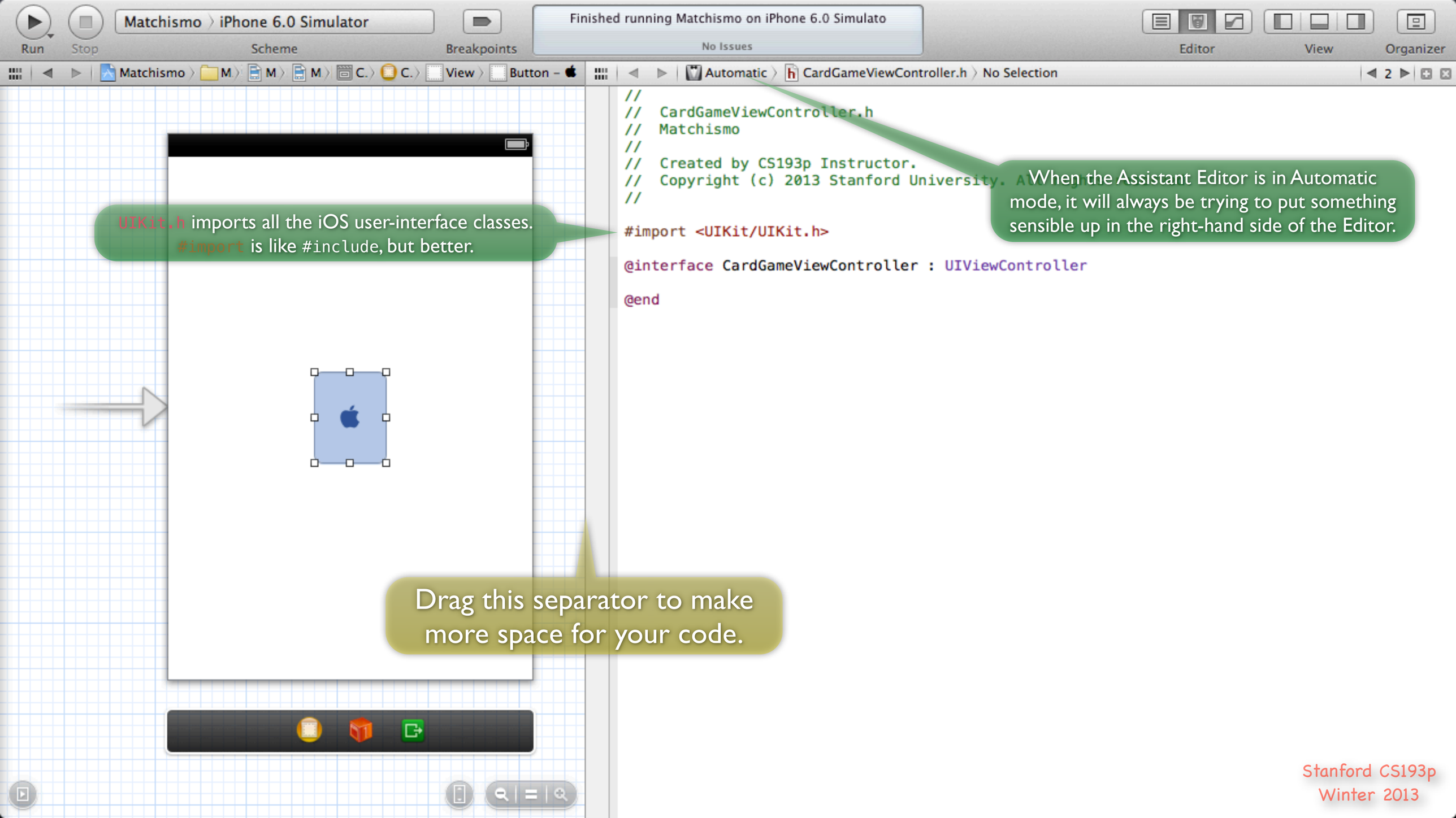

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
But we never saw the Selected state!
That's because buttons do not switch to selected when you press them by default.
We'll have to do that in code.
Carrier
Auto
All Output
Stanford CS193p
Winter 2013

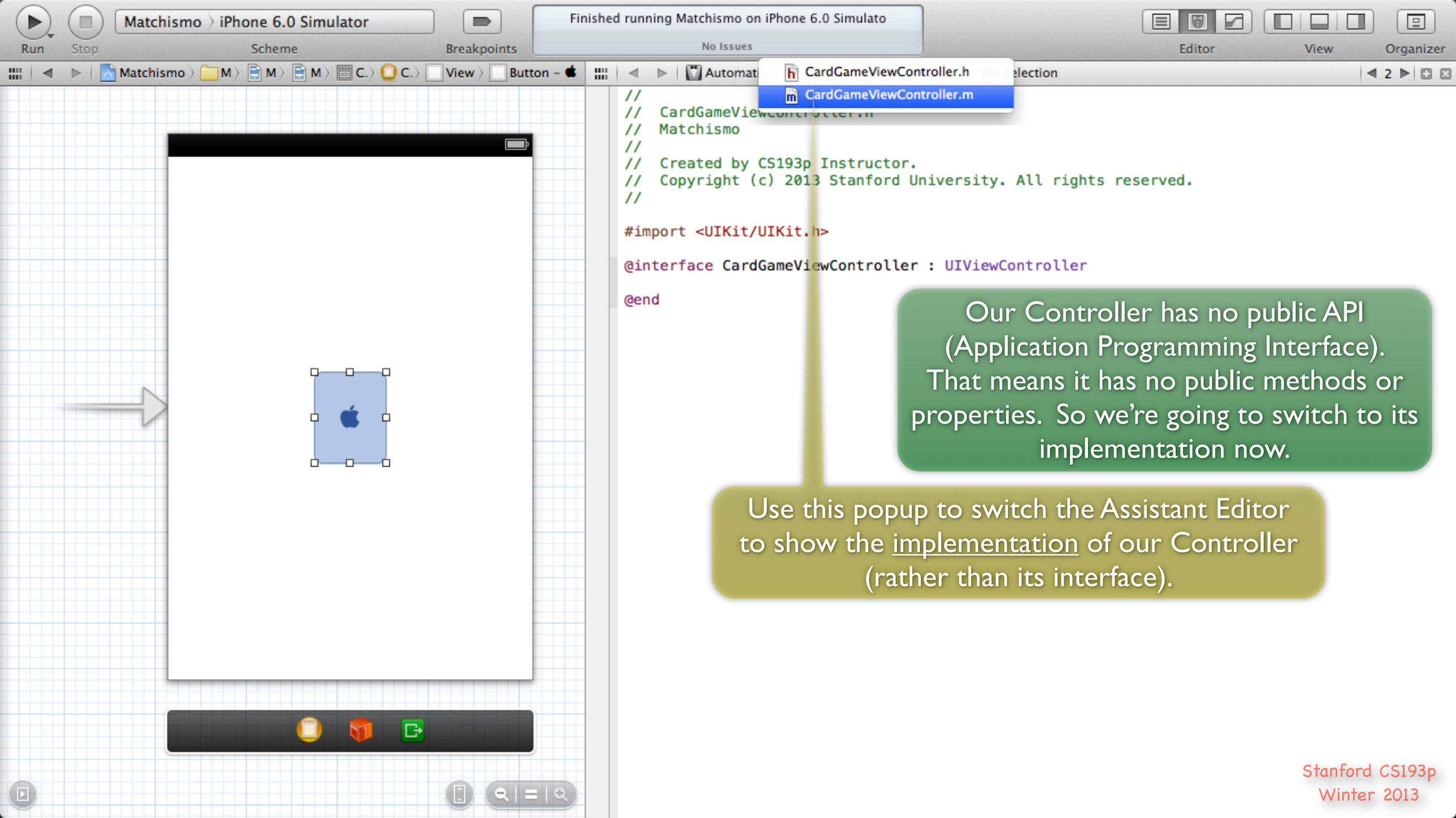

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matchismo 〉 MainStor... 〉 MainStor... 〉 Card Ga... 〉 Card Ga... 〉 View 〉 Button
To make more room for our code editing, we're going to close both our Navigator and Utilities Area.
Click here to close the Utilities Area.
Button
Type
Rounded Rect
State Config
Selected
Title
Plain
Font
System Bold 24.0
Text Color
Default
Shadow Color
Default
Image
Selected Image
Background
Selected Background Image
Width
Height
Reverses On Highlight
Highlight Tint
Default
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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
Notice that the debug area automatically disappears when you stop running.
If the debug area does not automatically disappear when you stop running, click here.
Stanford CS193p
Winter 2013

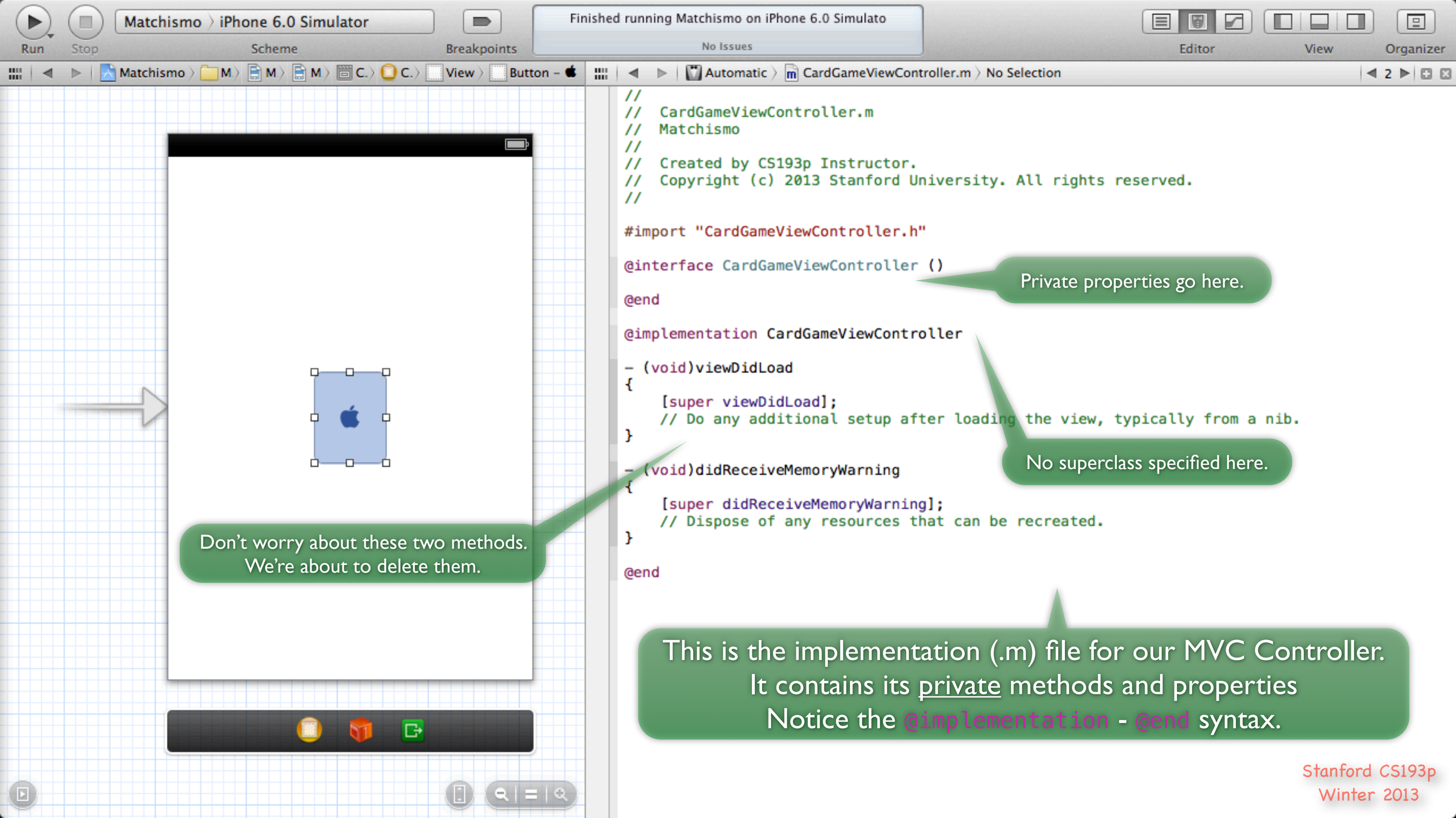

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.stor... 〉 MainStoryboard.stor... 〉 Card Game View Con... 〉 Card Game View Con... 〉 View 〉 Button
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Click here to close the Navigator.
Stanford CS193p
Winter 2013

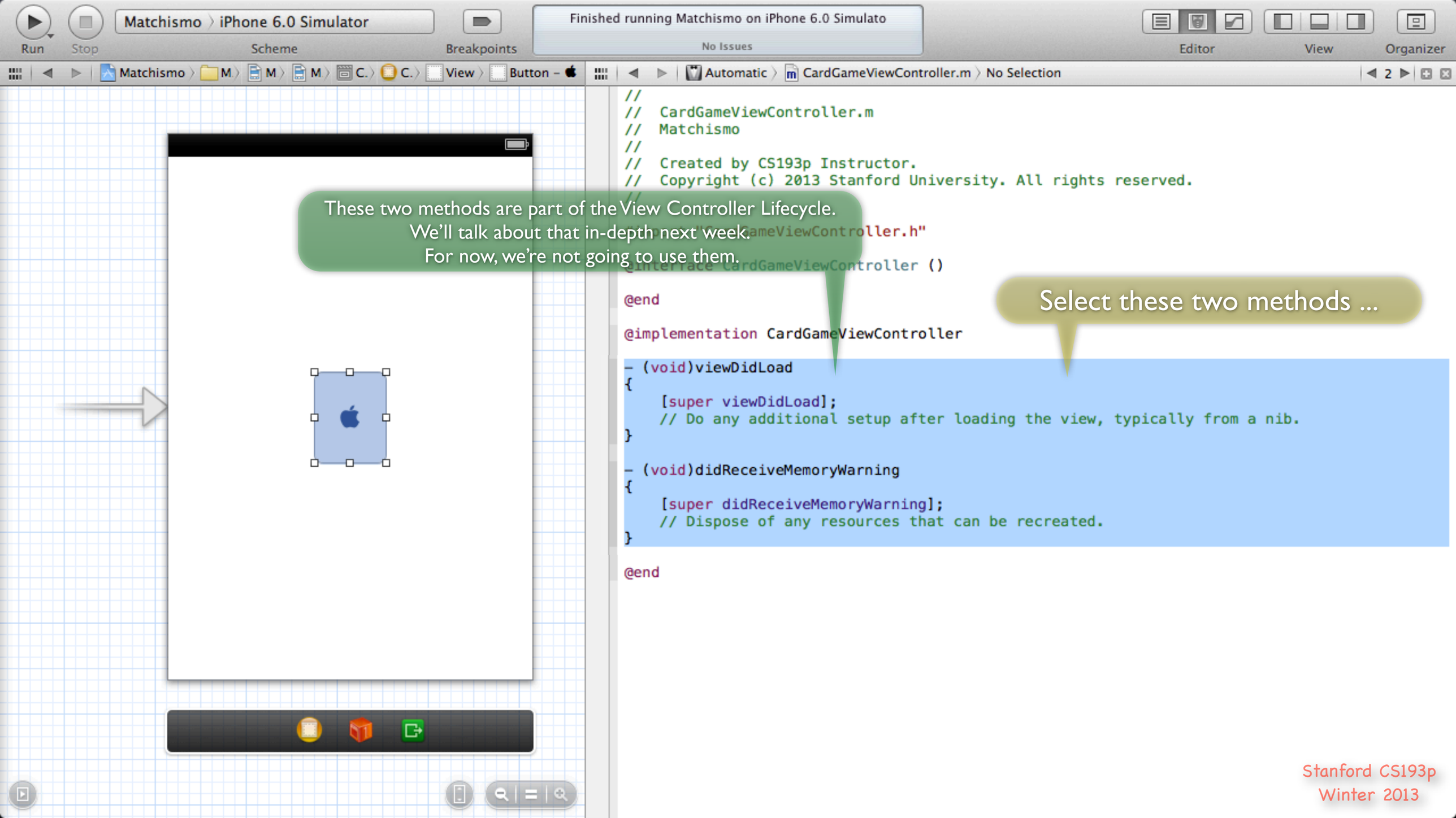

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 Matchismo 〉 MainStoryboard.storyboard 〉 MainStoryboard.storyboard (English) 〉 Card Game View Controller Scene 〉 Card Game View Controller 〉 View 〉 Button –
We need to see our MVC Controller's code now.
But we still want our MVC View on screen at the same time.
The way to have two things on the screen at once is to use the Assistant Editor.
It is shown/hidden using this "butler" icon.
Click here to show the Assistant Editor.
When an MVC View is showing, it will by default bring up the View's Controller.
That's exactly what we want.

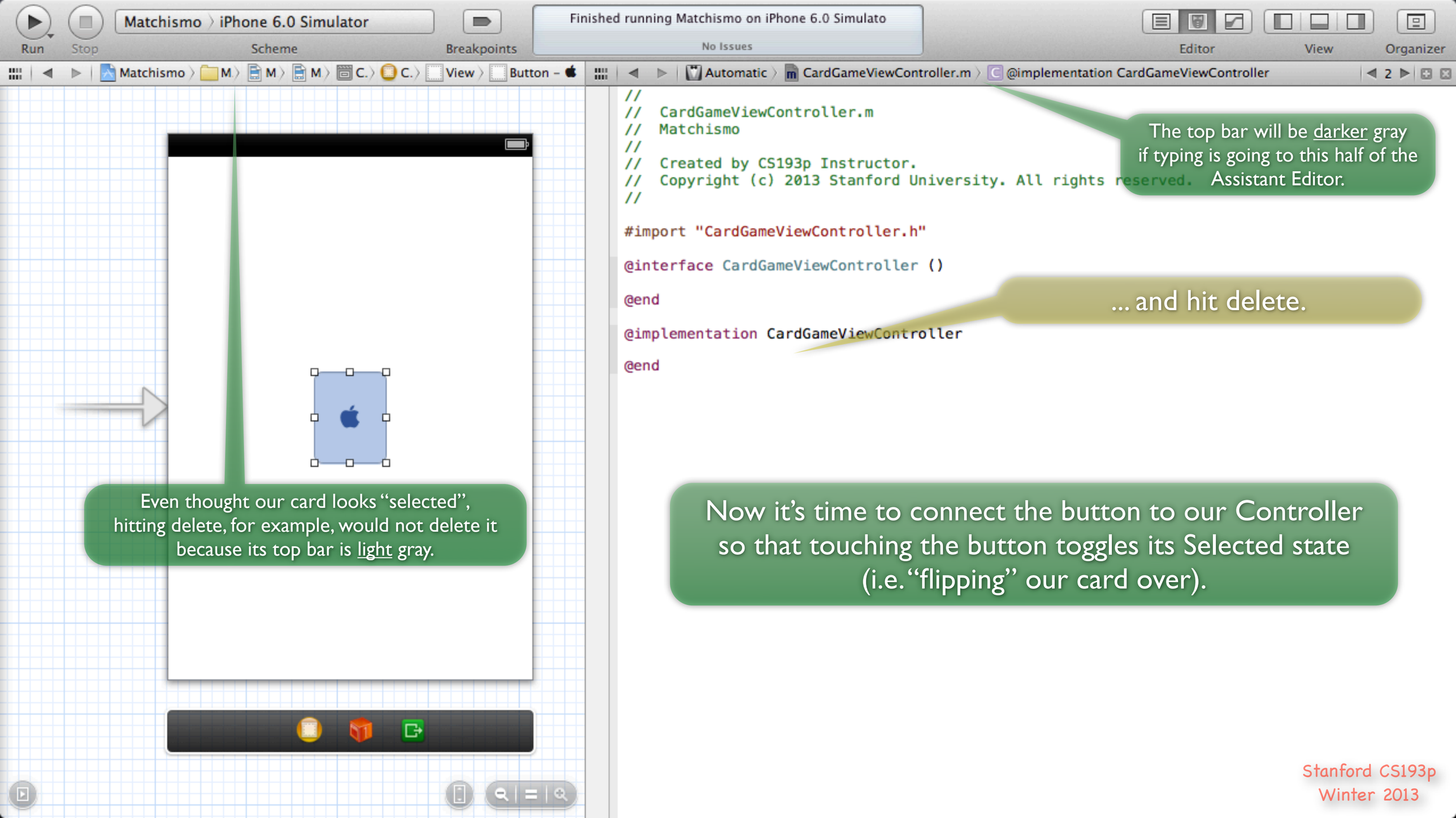

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.h 〉 No Selection
2
//
// CardGameViewController.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface CardGameViewController : UIViewController
@end
This is the header (.h) file for our MVC Controller.
It contains its public methods and properties
and also defines its superclass
(all Controllers in iOS inherit from UIViewController).
Notice the @interface - @end syntax.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.h 〉 No Selection
2
//
// CardGameViewController.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All
//
#import <UIKit/UIKit.h>
@interface CardGameViewController : UIViewController
@end
When the Assistant Editor is in Automatic mode, it will always be trying to put something sensible up in the right-hand side of the Editor.
UIKit.h imports all the iOS user-interface classes.
#import is like #include, but better.
Drag this separator to make more space for your code.
Stanford CS193p
Winter 2013

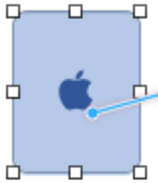

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automat
CardGameViewController.h
CardGameViewController.m
election
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface CardGameViewController : UIViewController
@end
Our Controller has no public API (Application Programming Interface). That means it has no public methods or properties. So we're going to switch to its implementation now.
Use this popup to switch the Assistant Editor to show the implementation of our Controller (rather than its interface).
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 No Selection
2
//
// 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
Private properties go here.
No superclass specified here.
Don't worry about these two methods.
We're about to delete them.
This is the implementation (.m) file for our MVC Controller.
It contains its private methods and properties
Notice the @implementation - @end syntax.
Stanford CS193p
Winter 2013

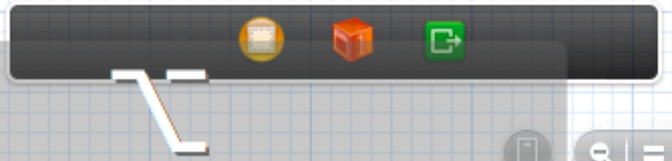

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 No Selection
2
These two methods are part of the View Controller Lifecycle.
We'll talk about that in-depth next week.
For now, we're not going to use them.
Select these two methods ...
//
// 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
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
The top bar will be darker gray if typing is going to this half of the Assistant Editor.
... and hit delete.
Even thought our card looks "selected", hitting delete, for example, would not delete it because its top bar is light gray.
Now it's time to connect the button to our Controller so that touching the button toggles its Selected state (i.e. "flipping" our card over).
Stanford CS193p
Winter 2013

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 ...



Don't forget this little window.
It means CTRL is being held down.



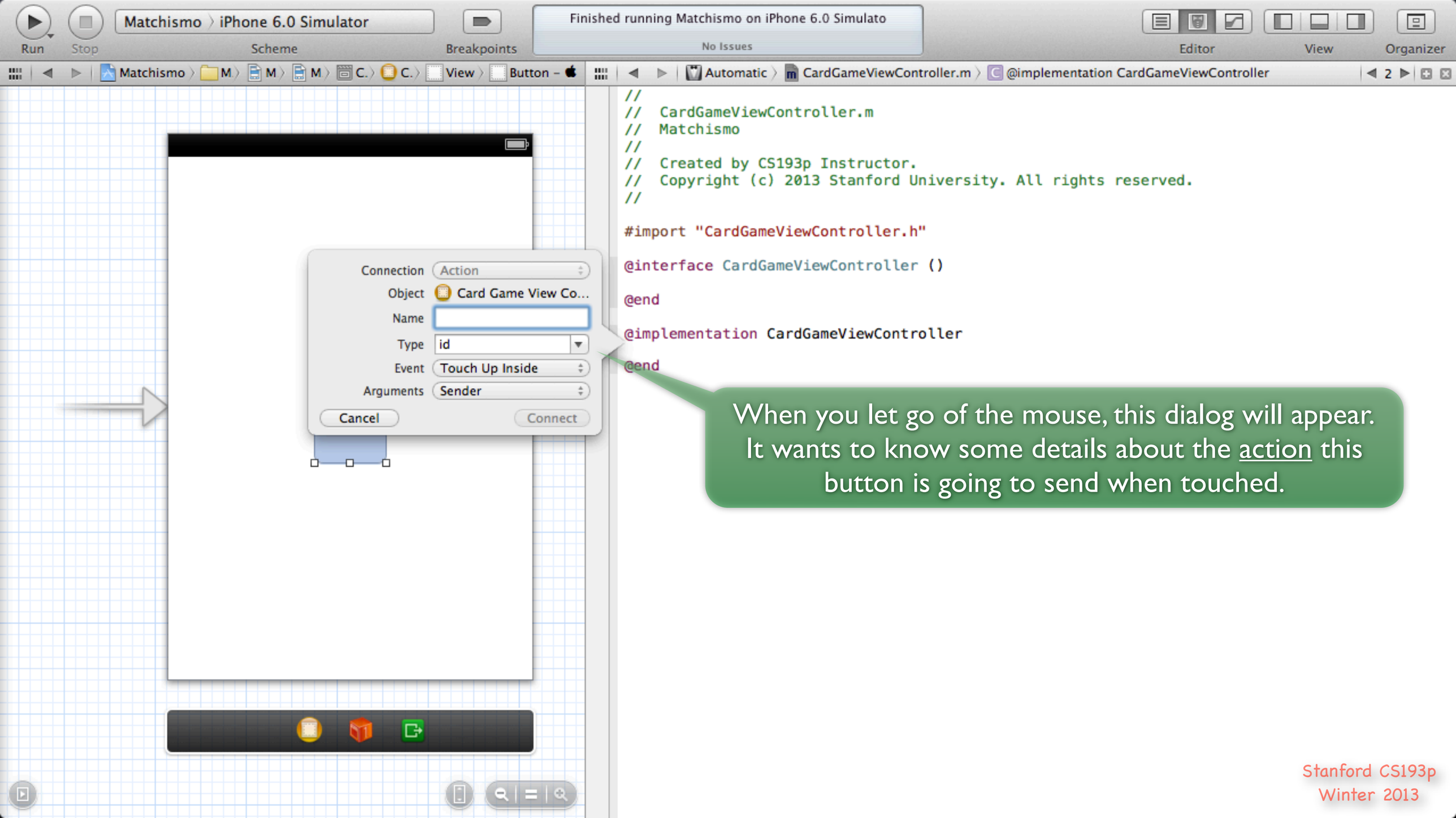
```
//
//  CardGameViewController.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University. All rights reserved.
//

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController
@end
```

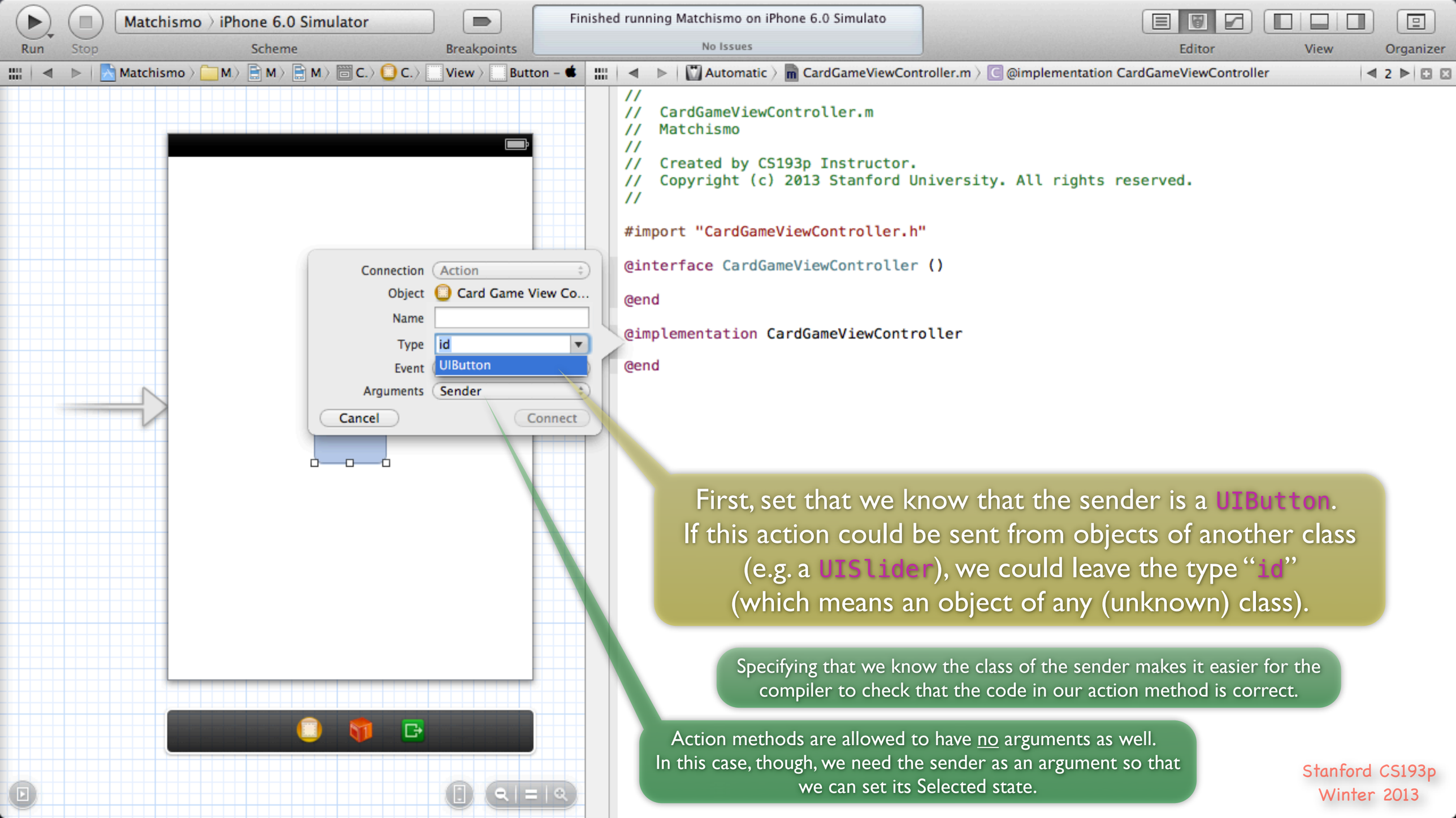
Insert Action

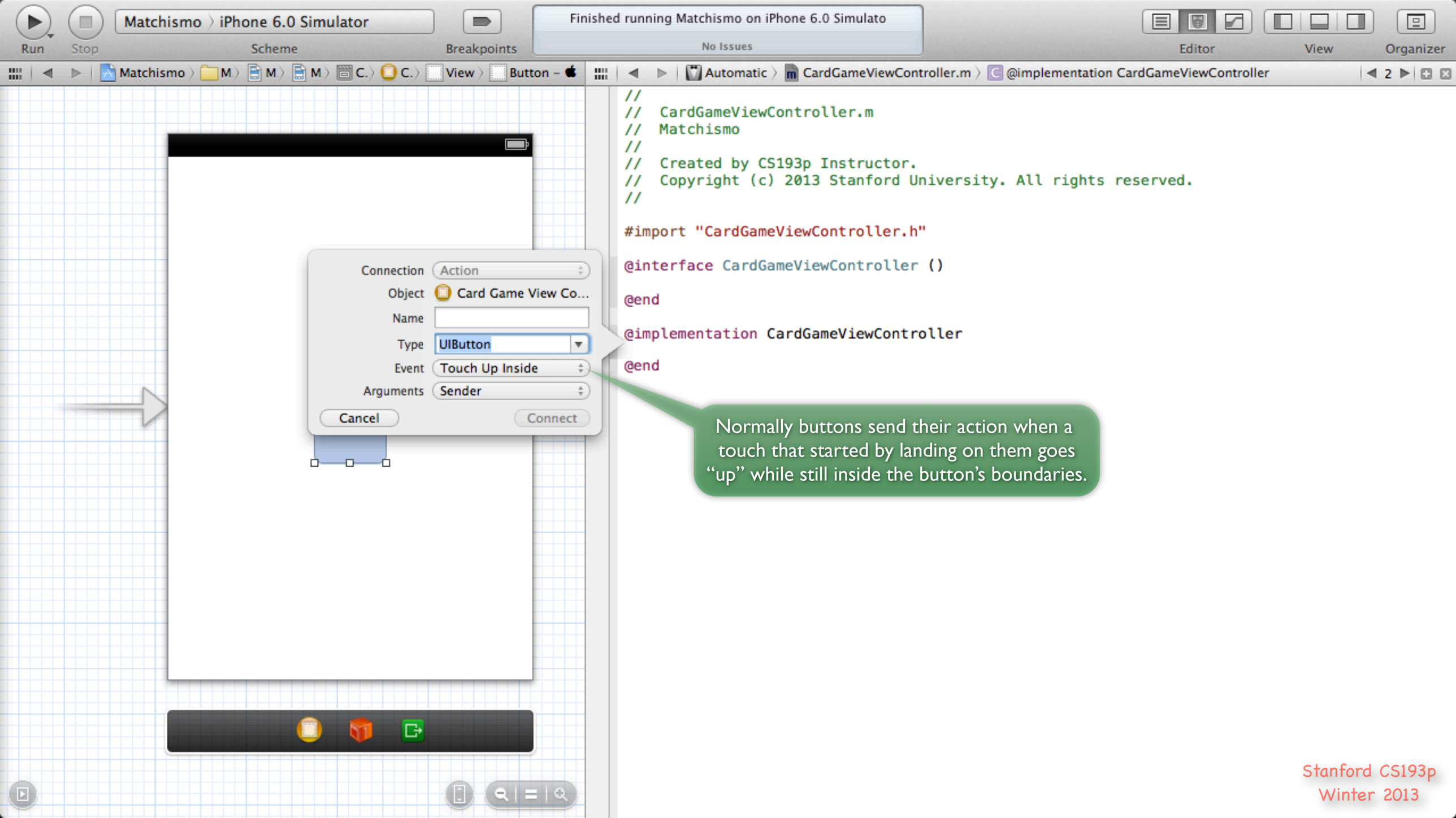
Hold down the CTRL key while dragging the mouse from your button to the code (then let go).
Since we're creating an "action" here, we drag somewhere between the `@implementation` and the `@end`.

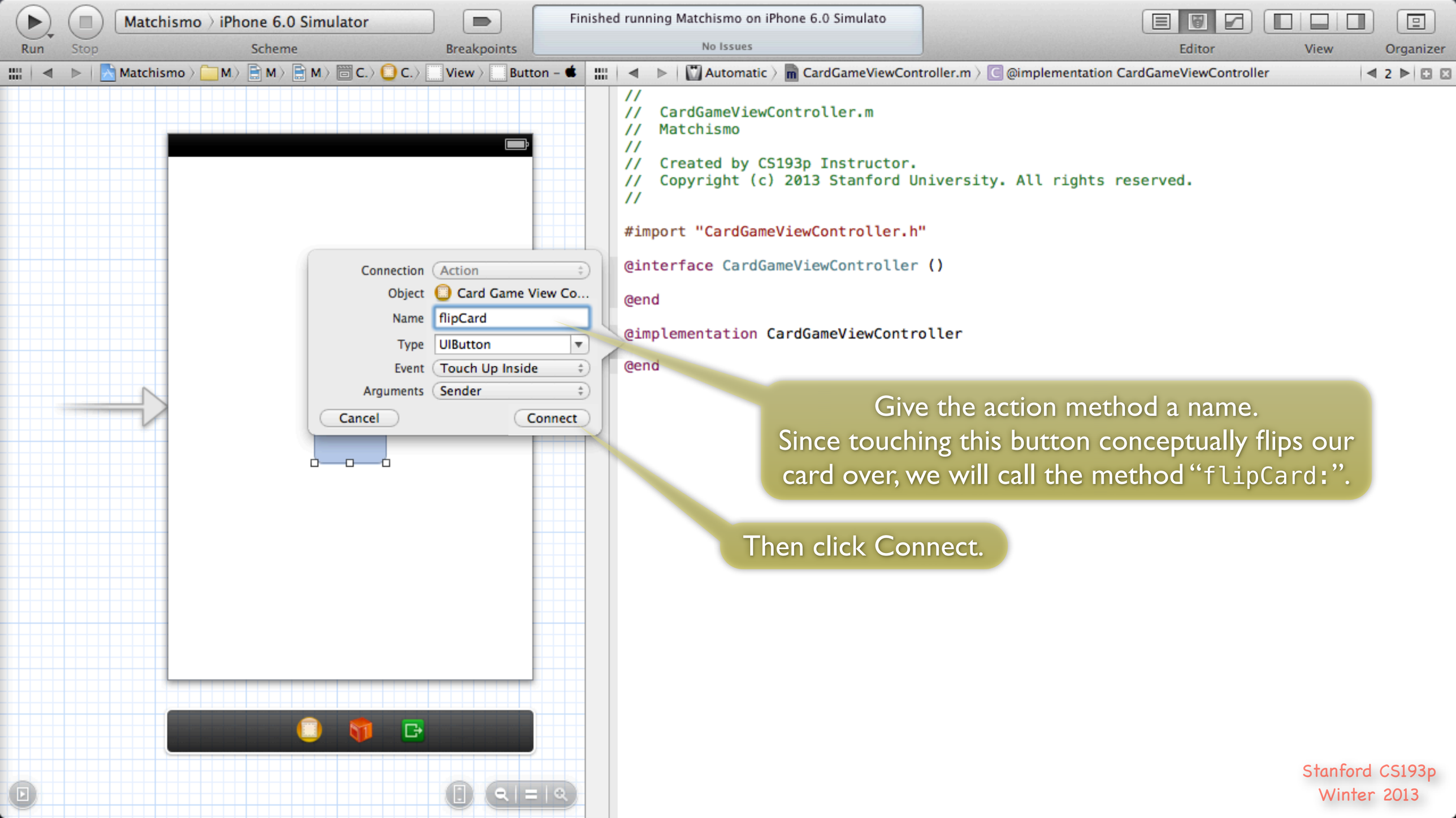
If we were making an "outlet" connection here, we'd drag it up between the `@interface` and the `@end` above.

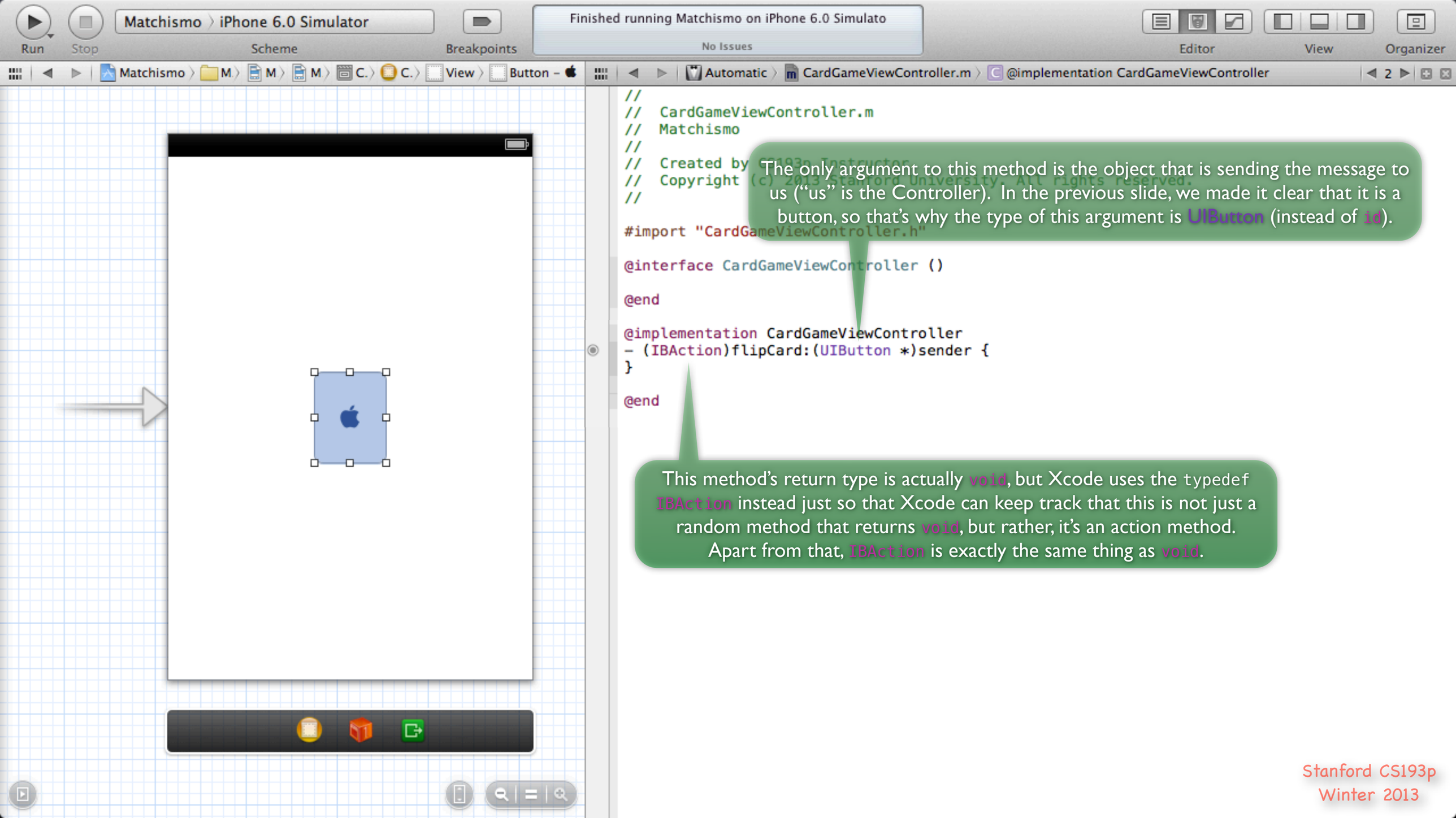
You can drag from your View to the header (.h) file of your Controller if you want to make a public connection (rare).

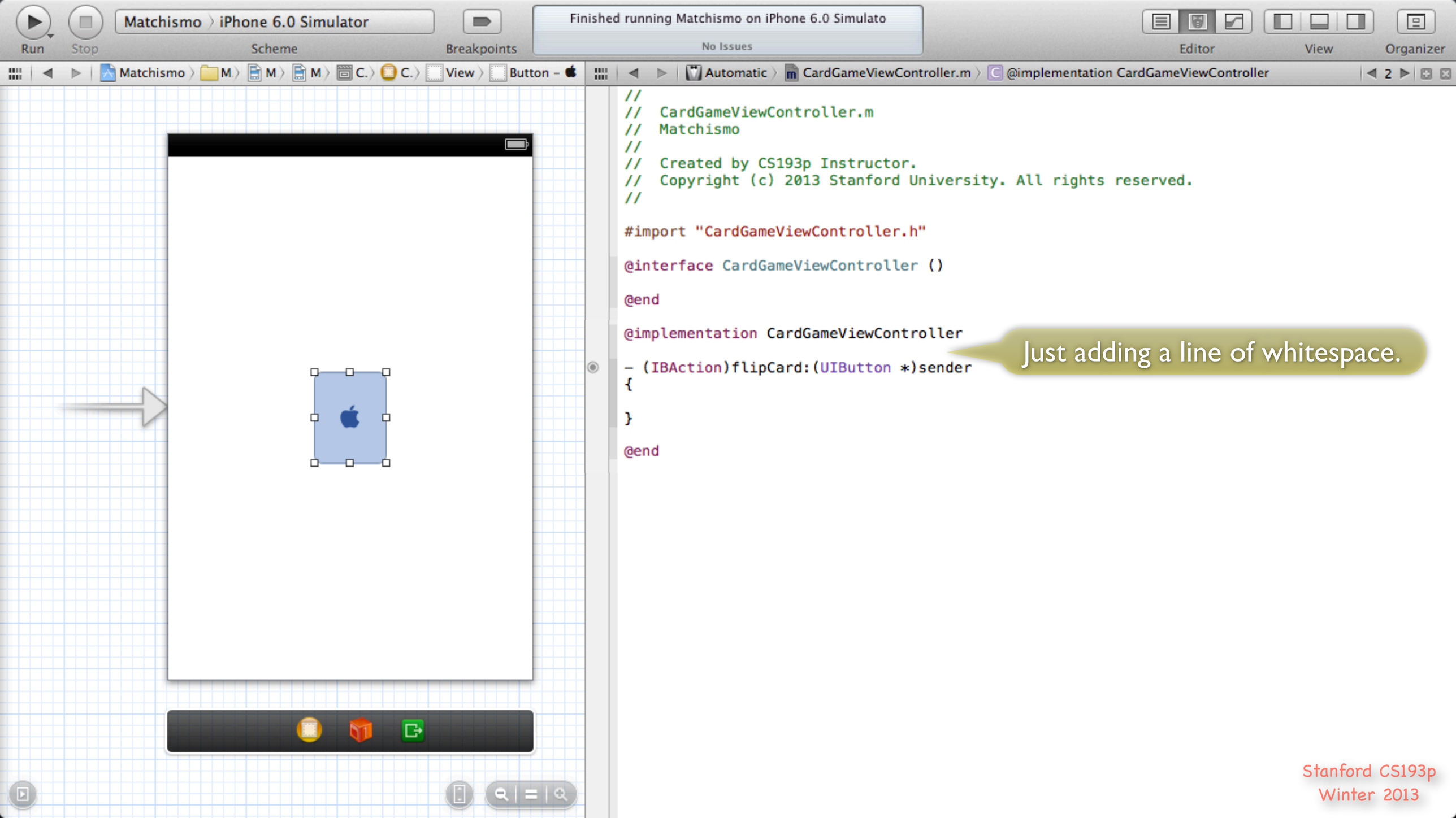

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
Connection Action
Object Card Game View Co...
Name
Type id
Event Touch Up Inside
Arguments Sender
Cancel
Connect
//
// 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 this button is going to send when touched.
Stanford CS193p
Winter 2013

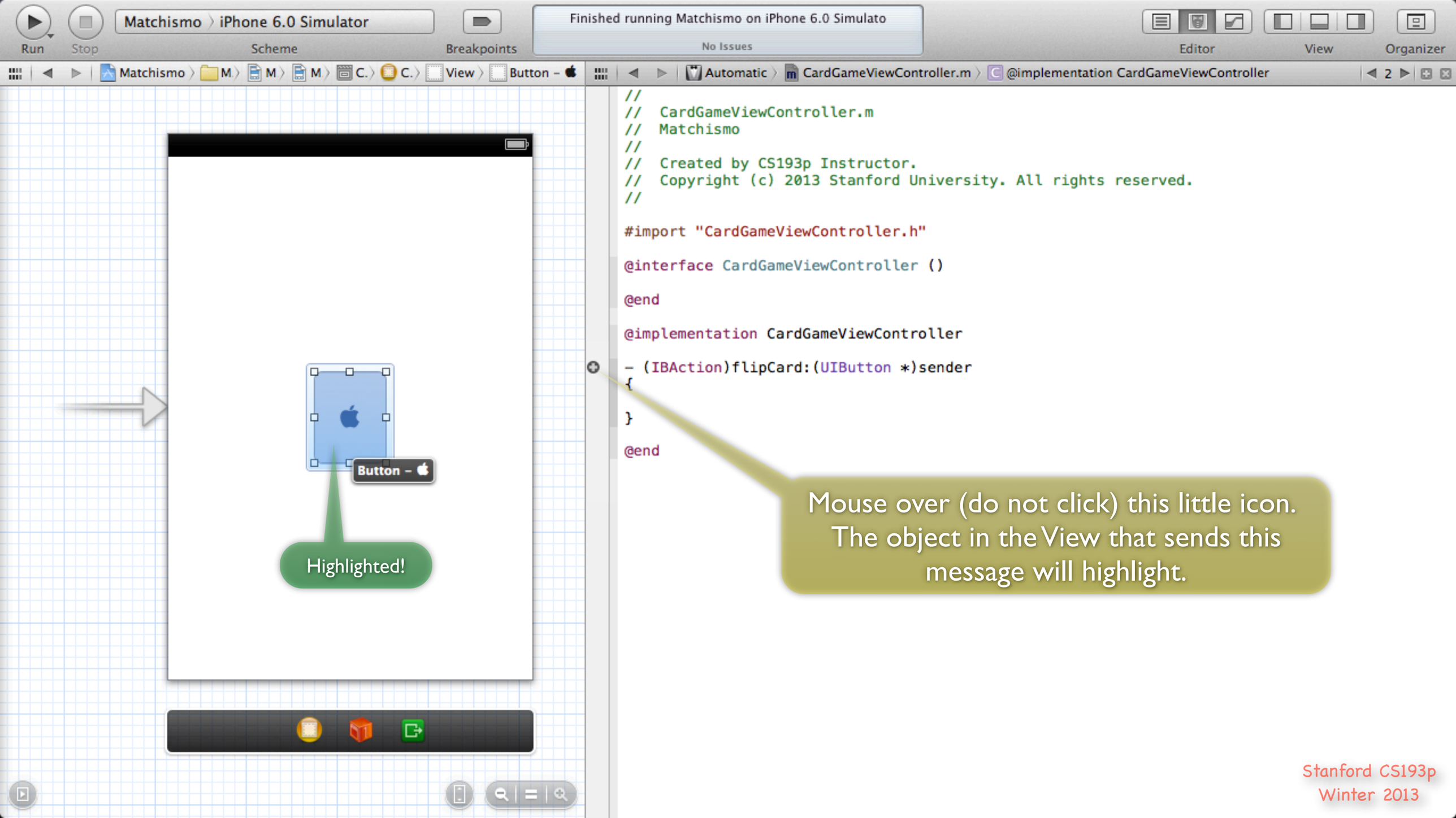

Matchismo › iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo › M › M › M › C. › C. › View › Button
Automatic › CardGameViewController.m › @implementation CardGameViewController
2
Connection Action
Object Card Game View Co...
Name
Type id
Event
UIButton
Arguments Sender
Cancel
Connect
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
First, set that we know that the sender 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 correct.
Action methods are allowed to have no arguments as well.
In this case, though, we need the sender as an argument so that we can set its Selected state.
Stanford CS193p
Winter 2013

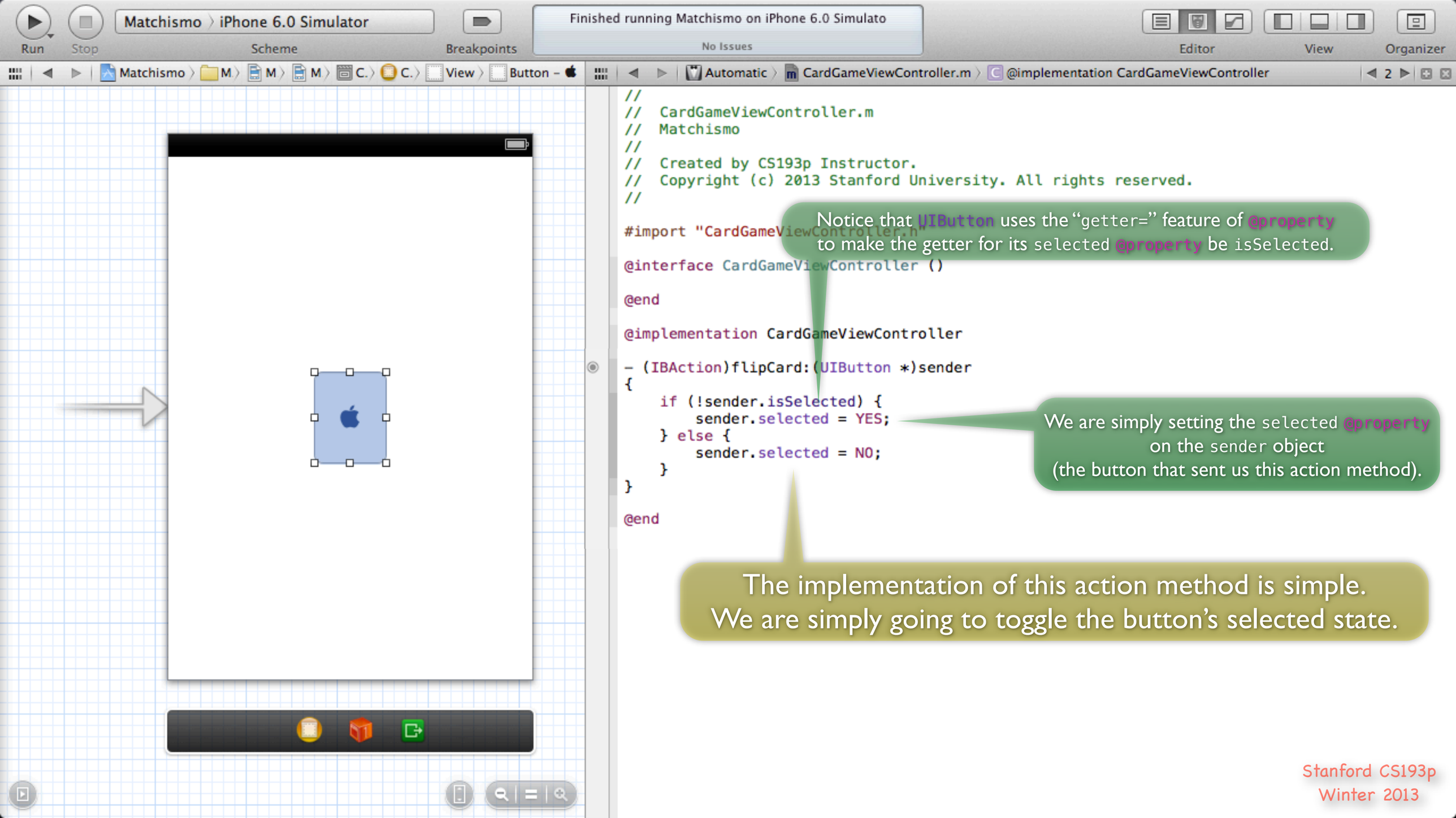

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
Connection Action
Object Card Game View Co...
Name
Type UIButton
Event Touch Up Inside
Arguments Sender
Cancel
Connect
//
// 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.
Stanford CS193p
Winter 2013

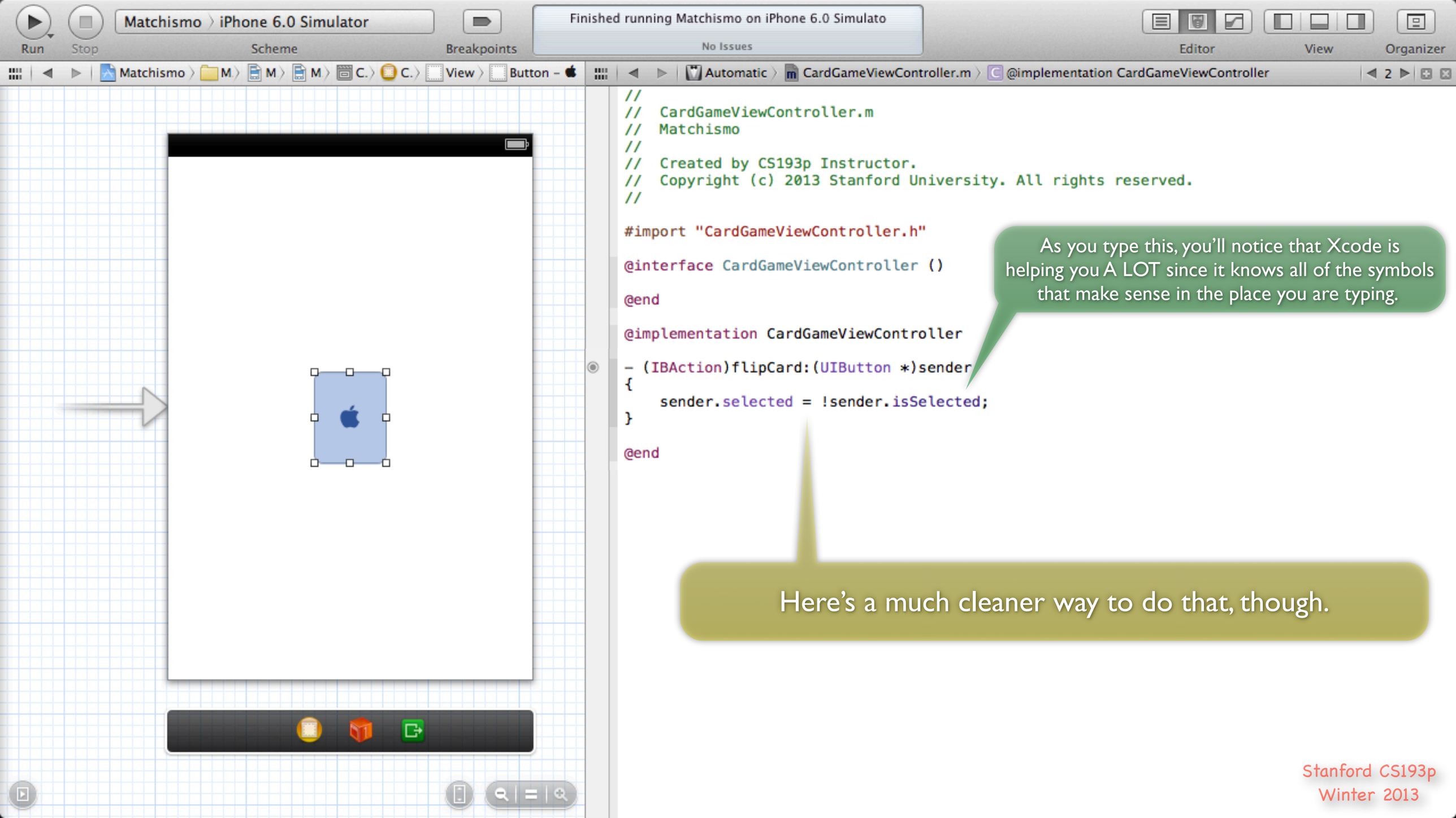

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
Connection Action
Object Card Game View Co...
Name flipCard
Type UIButton
Event Touch Up Inside
Arguments Sender
Cancel
Connect
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
@end
Give the action method a name.
Since touching this button conceptually flips our card over, we will call the method "flipCard:".
Then click Connect.
Stanford CS193p
Winter 2013

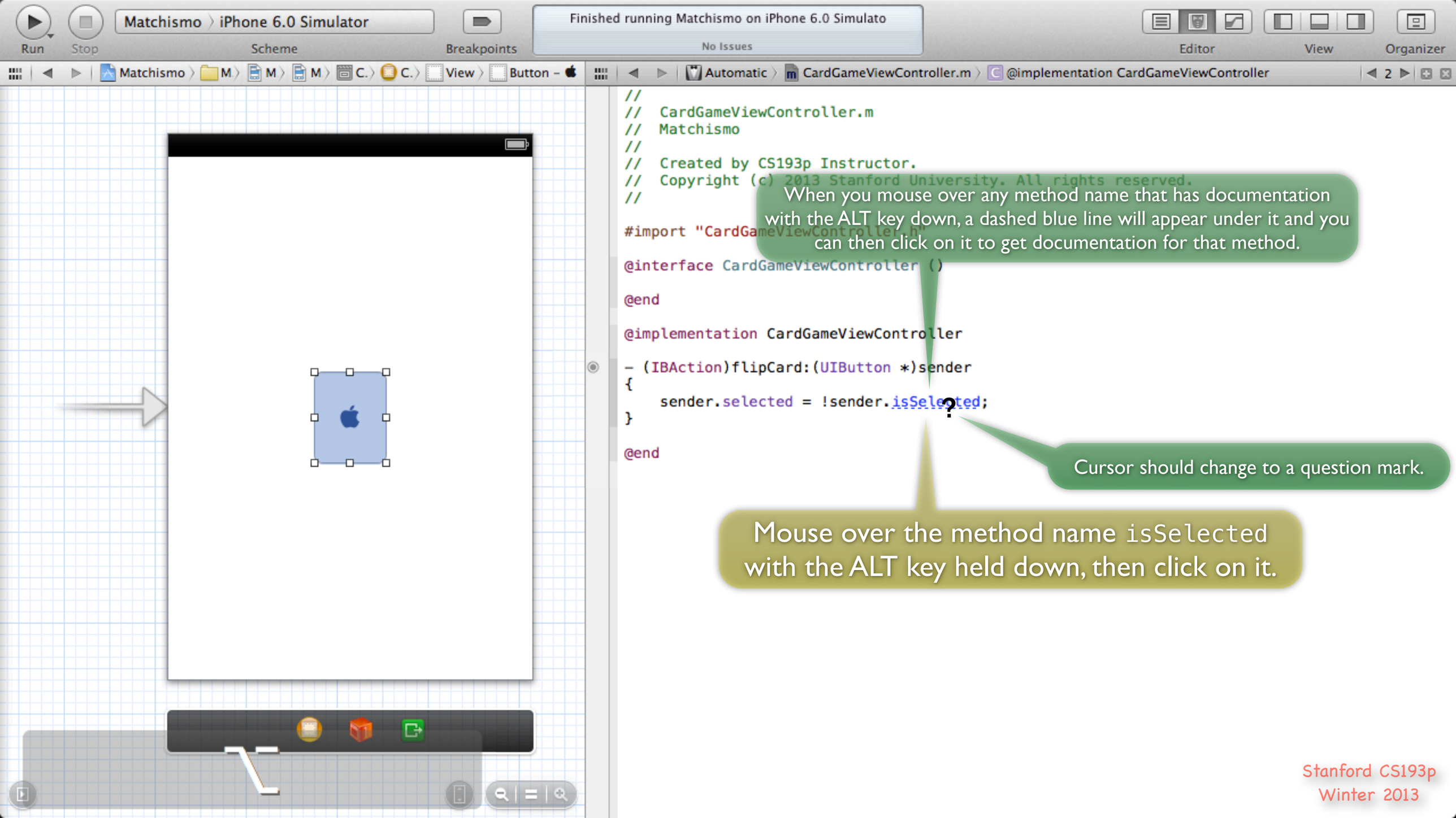

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(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).
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.

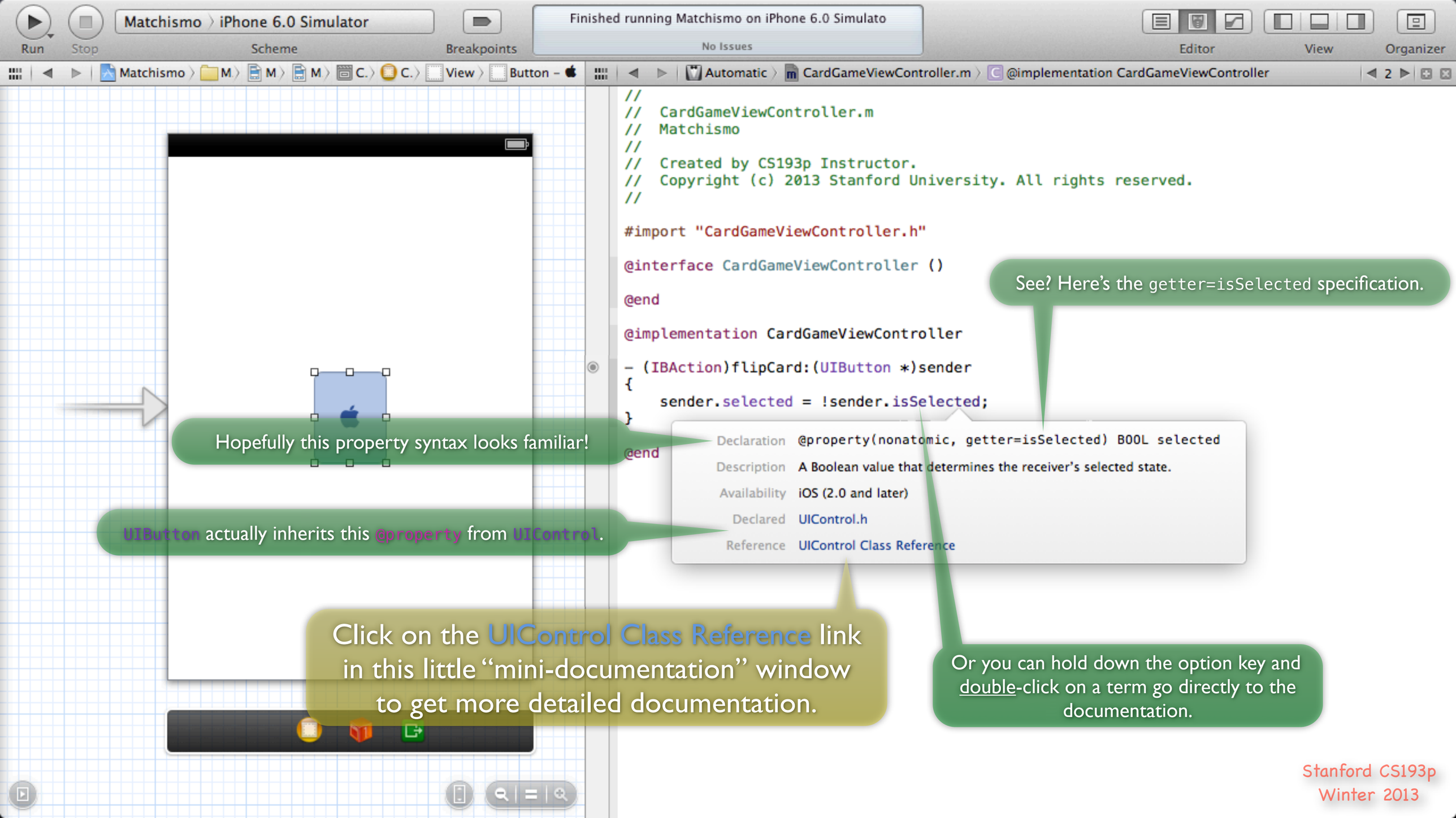

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
}
@end
Just adding a line of whitespace.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
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)flipCard:(UIButton *)sender
{
}
@end
Button –
Highlighted!
Mouse over (do not click) this little icon. The object in the View that sends this message will highlight.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
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)flipCard:(UIButton *)sender
{
    if (!sender.isSelected) {
        sender.selected = YES;
    } else {
        sender.selected = NO;
    }
}
@end
Notice that UIButton uses the "getter=" feature of @property to make the getter for its selected @property be isSelected.
We are simply setting the selected @property on the sender object (the button that sent us this action method).
The implementation of this action method is simple. We are simply going to toggle the button's selected state.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
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)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}
@end
As you type this, you'll notice that Xcode is helping you A LOT since it knows all of the symbols that make sense in the place you are typing.
Here's a much cleaner way to do that, though.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
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)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}
@end
When you mouse over any method name that has documentation with the ALT key down, a dashed blue line will appear under it and you can then click on it to get documentation for that method.
Cursor should change to a question mark.
Mouse over the method name isSelected with the ALT key held down, then click on it.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 @implementation CardGameViewController
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)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}
@end
Declaration @property(nonatomic, getter=isSelected) BOOL selected
Description A Boolean value that determines the receiver's selected state.
Availability iOS (2.0 and later)
Declared UIControl.h
Reference UIControl Class Reference
See? Here's the getter=isSelected specification.
Hopefully this property syntax looks familiar!
UIButton actually inherits this @property from UIControl.
Click on the UIControl Class Reference link in this little "mini-documentation" window to get more detailed documentation.
Or you can hold down the option key and double-click on a term go directly to the documentation.
Stanford CS193p
Winter 2013

This is the Organizer window.

Devices Repositories Projects Archives Documentation

The Organizer window shows documentation, source control, device management, etc.

iOS 6.0 Documentation Set › User Experience › Controls › UIControl Class Reference

Hits must: contain search term
Languages: All Languages
Find in: 1 of 6 Doc Sets

selected

A Boolean value that determines the receiver's selected state.

```
@property(nonatomic, getter=isSelected) BOOL selected
```

Discussion
Specify `YES` if the control is selected; otherwise `NO`. The default is `NO`. For many controls, this state has no effect on behavior or appearance. But other subclasses (for example, UISwitchControl) or the application object might read or set this control state.

Availability
Available in iOS 2.0 and later.

See Also
`@property state`

Related Sample Code
oalTouch
Table View Animations and Gestures

Declared In
`UIControl.h`

Xcode opened up the documentation for `UIControl` (`UIButton`'s superclass) to show you the documentation for the `selected` property.

state

A Boolean value that indicates the state of the receiver. (read-only)

```
@property(nonatomic, readonly) UIControlState state
```

Discussion
One or more `UIControlState` bit-mask constants that specify the state of the `UIControl` object; for information on these constants, see "Control State." Note that the control can be in more than one state, for example, both disabled and selected `(UIControlStateDisabled | UIControlStateSelected)`.This attribute is read only—there is no corresponding setter method.

Availability
Available in iOS 2.0 and later.

See Also
`@property enabled`
`@property selected`
`@property highlighted`

UIControl Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) UIAppearance (UIView) UIAppearanceContainer (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.
Declared in	UIControl.h
Related sample code	Formulaic Regions UICatalog

Scroll up to the top to see an Overview of the `UIControl` class.

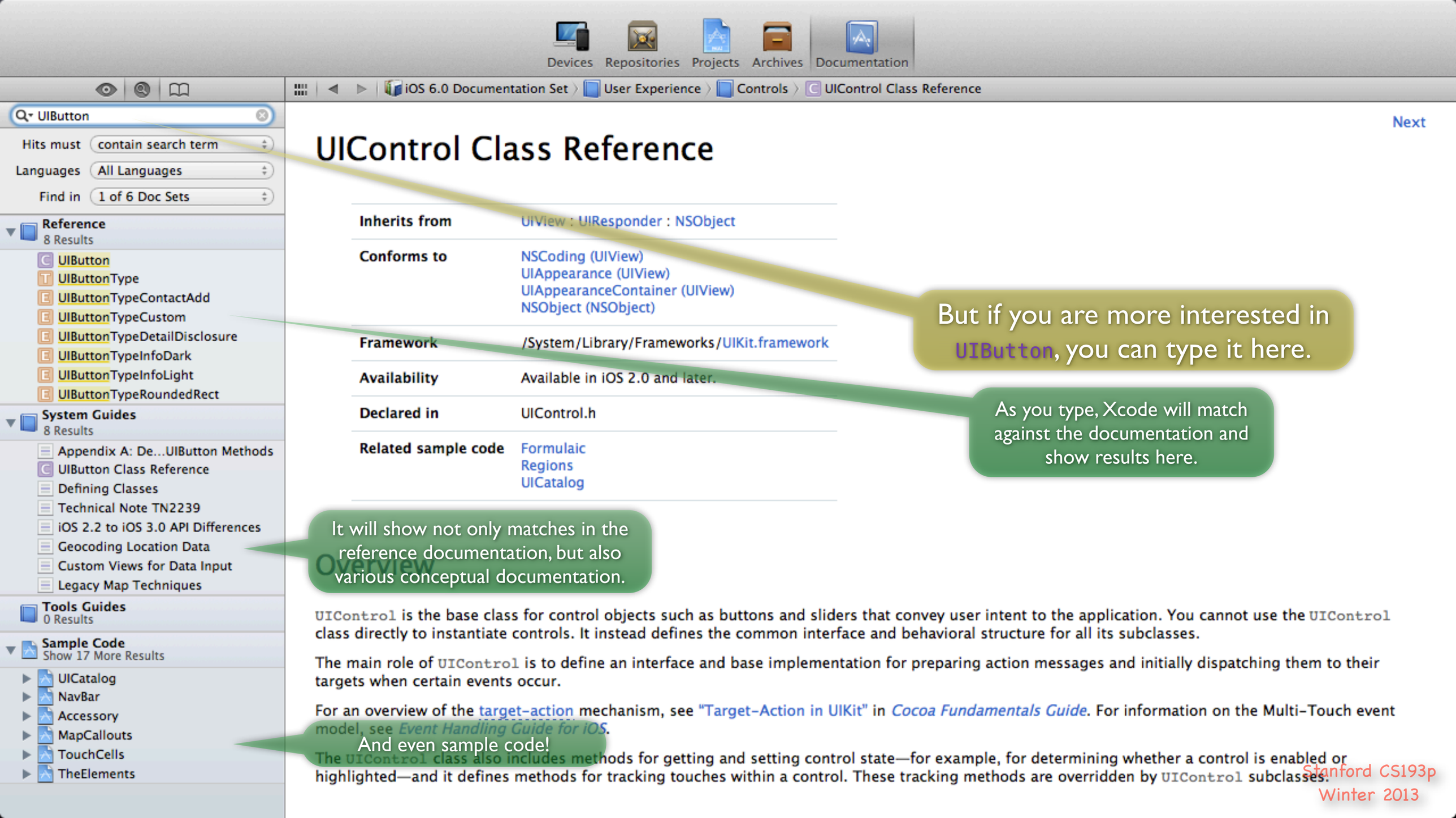
Overview

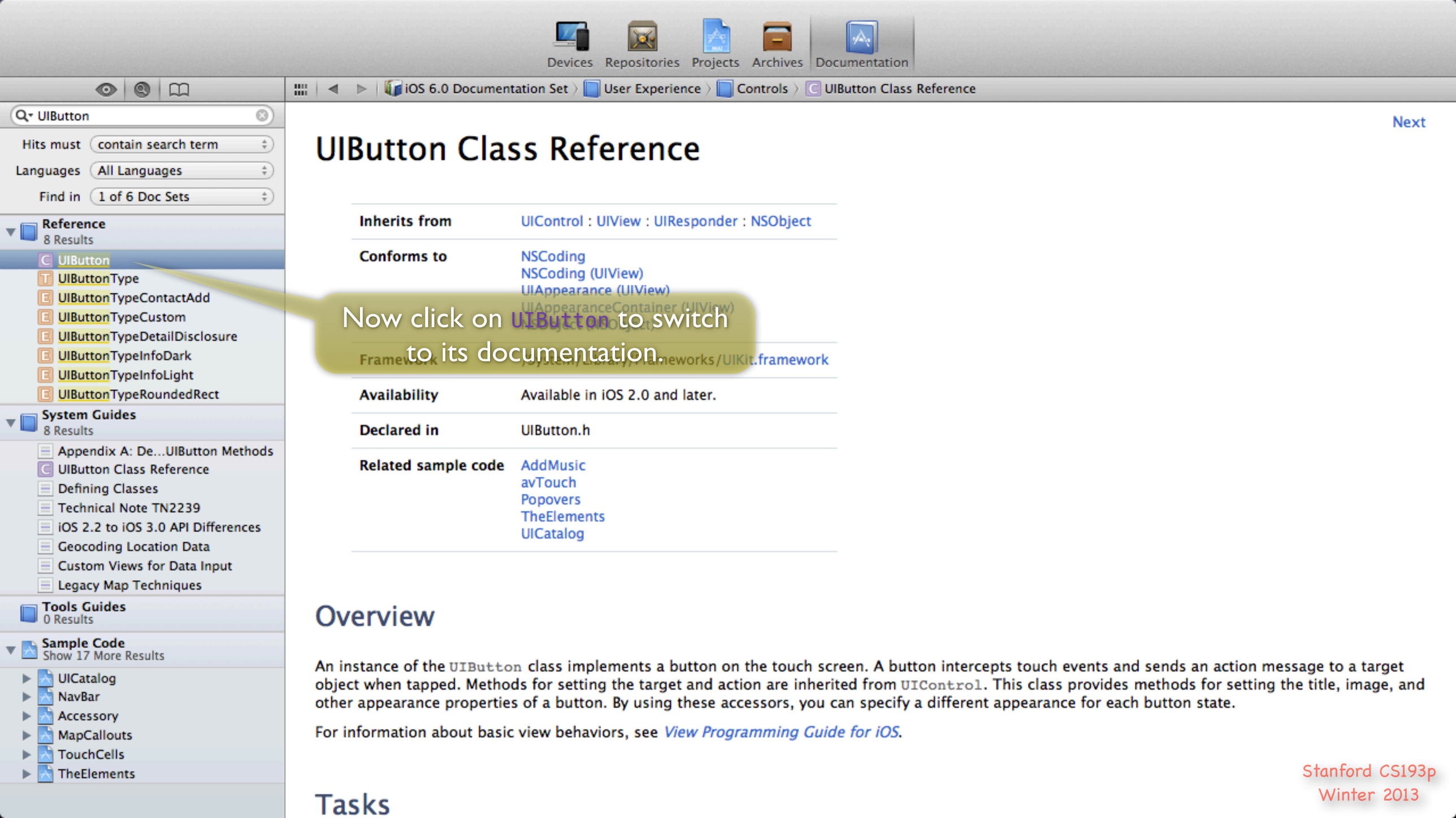
`UIControl` is the base class for control objects such as buttons and sliders that convey user intent to the application. You cannot use the `UIControl` class directly to instantiate controls. It instead defines the common interface and behavioral structure for all its subclasses.

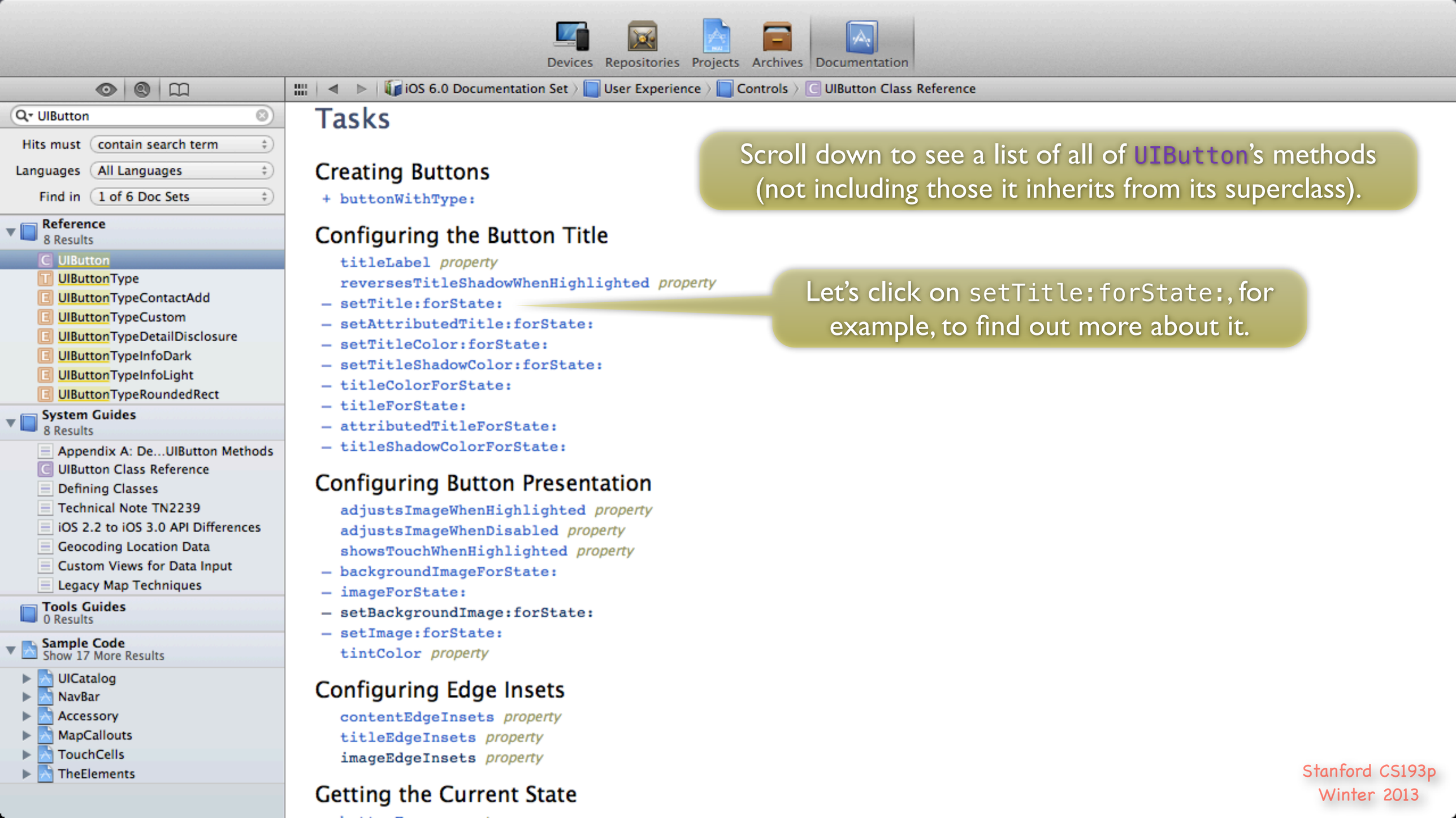
The main role of `UIControl` is to define an interface and base implementation for preparing action messages and initially dispatching them to their targets when certain events occur.

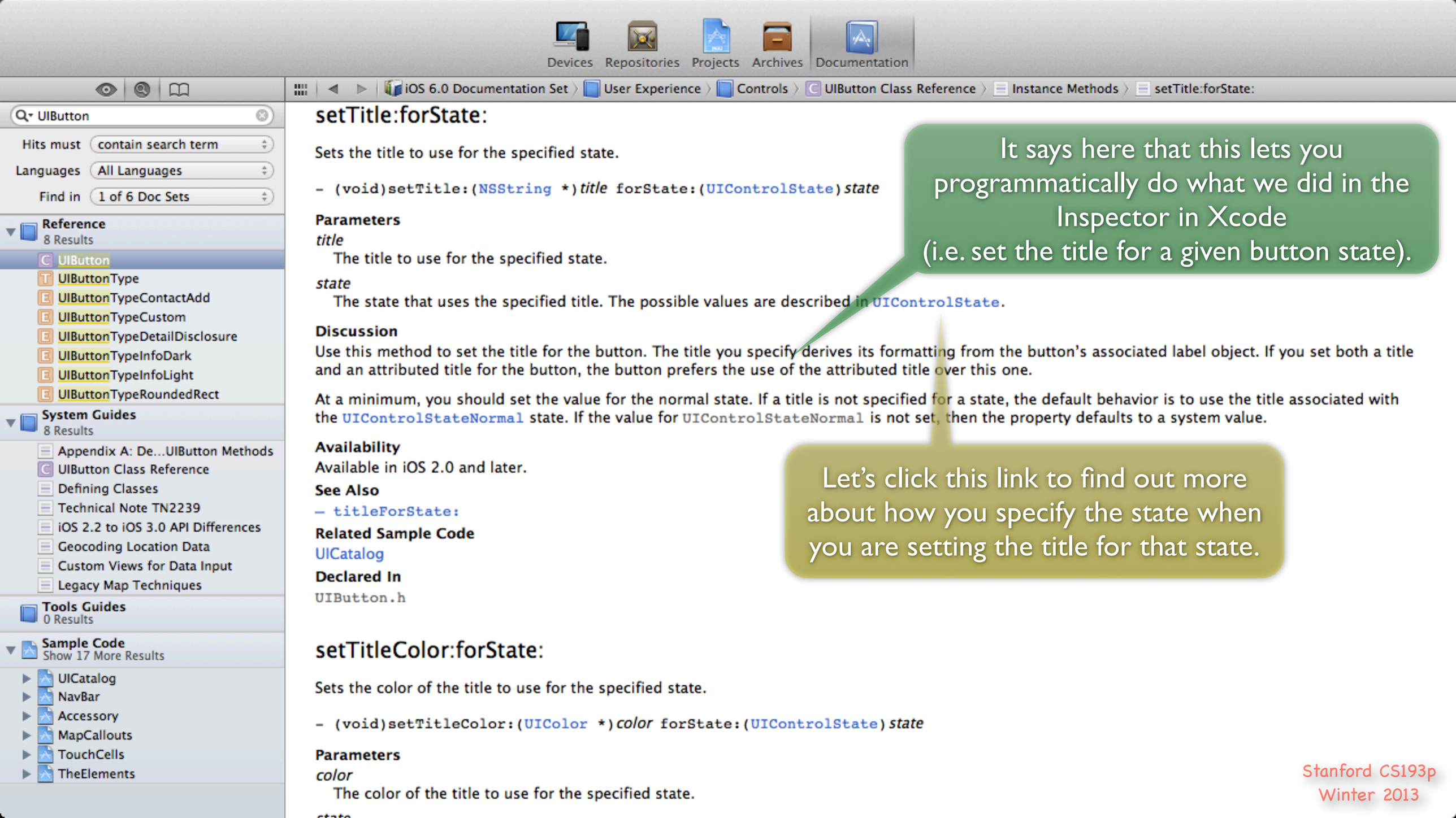
For an overview of the target-action mechanism, see "Target-Action in UIKit" in *Cocoa Fundamentals Guide*. For information on the Multi-Touch event model, see *Event Handling Guide for iOS*.

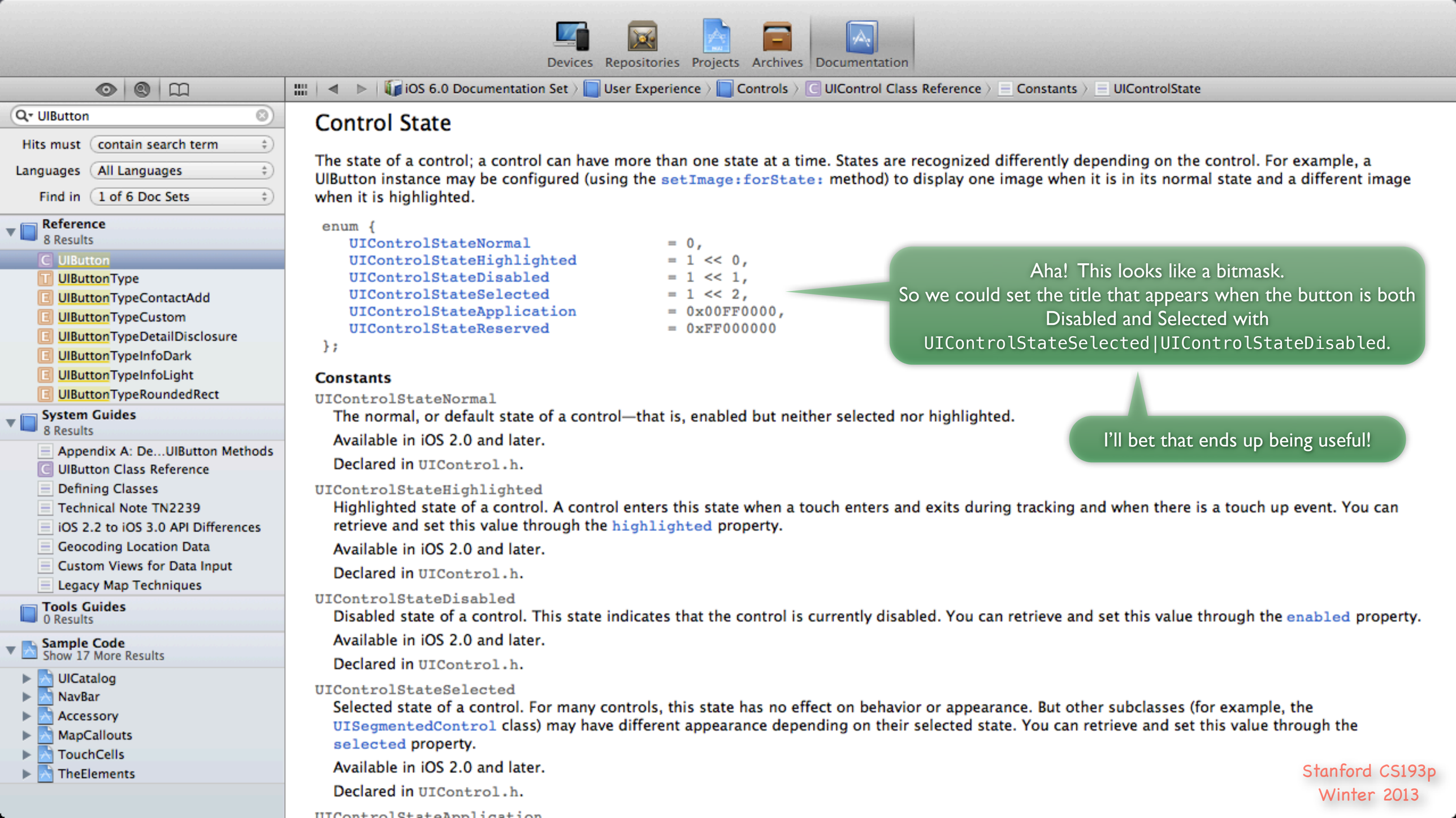
The `UIControl` class also includes methods for getting and setting control state—for example, for determining whether a control is enabled or highlighted—and it defines methods for tracking touches within a control. These tracking methods are overridden by `UIControl` subclasses.


Devices
Repositories
Projects
Archives
Documentation
iOS 6.0 Documentation Set
User Experience
Controls
UIControl Class Reference
UIButton
Hits must
contain search term
Languages
All Languages
Find in
1 of 6 Doc Sets
Reference
8 Results
UIButton
UIButtonType
UIButtonTypeContactAdd
UIButtonTypeCustom
UIButtonTypeDetailDisclosure
UIButtonTypeInfoDark
UIButtonTypeInfoLight
UIButtonTypeRoundedRect
System Guides
8 Results
Appendix A: De...UIButton Methods
UIButton Class Reference
Defining Classes
Technical Note TN2239
iOS 2.2 to iOS 3.0 API Differences
Geocoding Location Data
Custom Views for Data Input
Legacy Map Techniques
Tools Guides
0 Results
Sample Code
Show 17 More Results
UICatalog
NavBar
Accessory
MapCallouts
TouchCells
TheElements
Next
UIControl Class Reference
Inherits from
UIView : UIResponder : NSObject
Conforms to
NSCoding (UIView)
UIAppearance (UIView)
UIAppearanceContainer (UIView)
NSObject (NSObject)
Framework
/System/Library/Frameworks/UIKit.framework
Availability
Available in iOS 2.0 and later.
Declared in
UIControl.h
Related sample code
Formulaic
Regions
UICatalog
But if you are more interested in UIButton, you can type it here.
As you type, Xcode will match against the documentation and show results here.
It will show not only matches in the reference documentation, but also various conceptual documentation.
Overview
UIControl is the base class for control objects such as buttons and sliders that convey user intent to the application. You cannot use the UIControl class directly to instantiate controls. It instead defines the common interface and behavioral structure for all its subclasses.
The main role of UIControl is to define an interface and base implementation for preparing action messages and initially dispatching them to their targets when certain events occur.
For an overview of the target-action mechanism, see "Target-Action in UIKit" in Cocoa Fundamentals Guide. For information on the Multi-Touch event model, see Event Handling Guide for iOS.
And even sample code!
The UIControl class also includes methods for getting and setting control state—for example, for determining whether a control is enabled or highlighted—and it defines methods for tracking touches within a control. These tracking methods are overridden by UIControl subclasses.
Stanford CS193p
Winter 2013

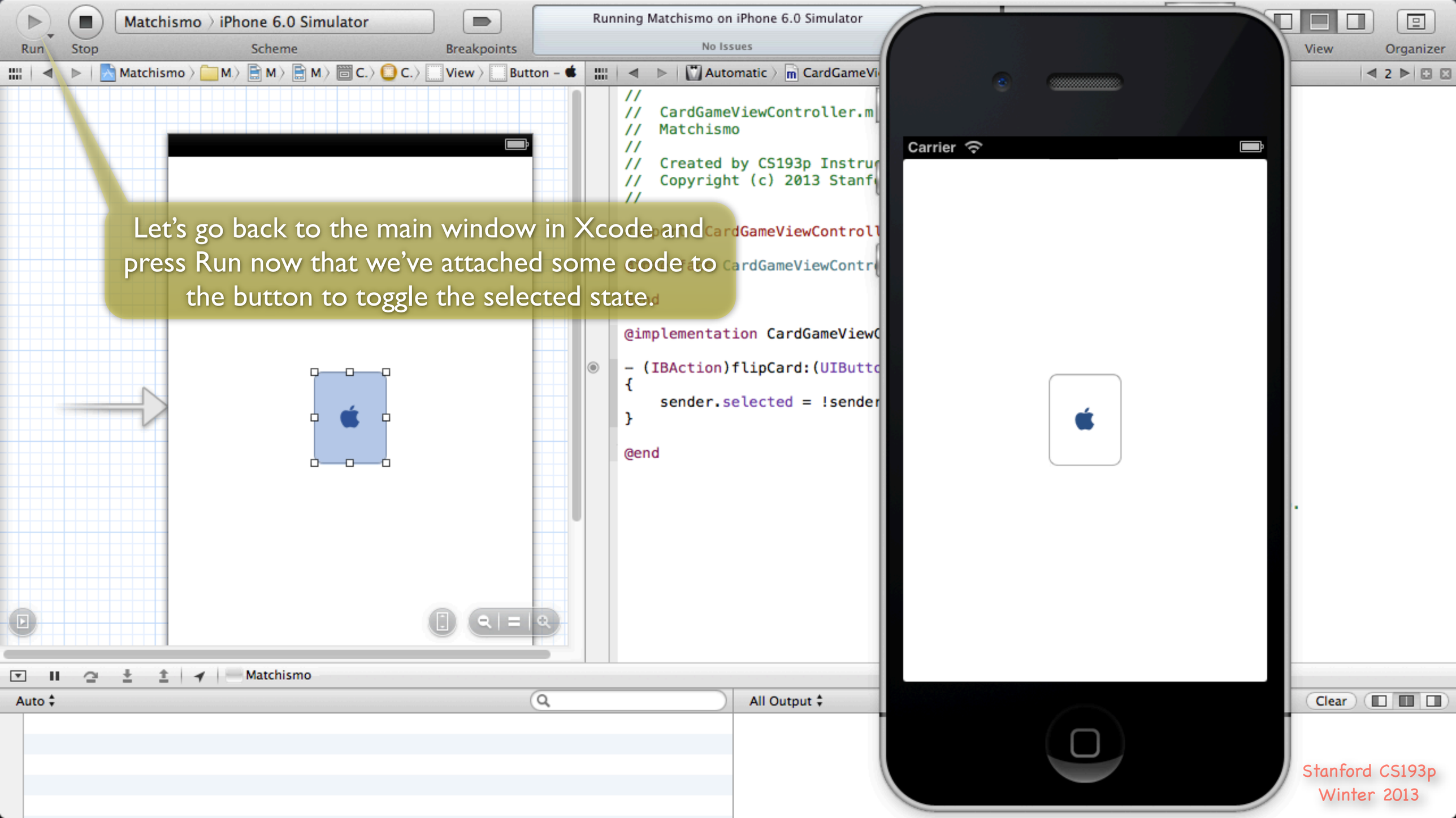

Devices
Repositories
Projects
Archives
Documentation
iOS 6.0 Documentation Set
User Experience
Controls
UIButton Class Reference
UIButton
Hits must contain search term
Languages All Languages
Find in 1 of 6 Doc Sets
Reference
8 Results
UIButton
UIButtonType
UIButtonTypeContactAdd
UIButtonTypeCustom
UIButtonTypeDetailDisclosure
UIButtonTypeInfoDark
UIButtonTypeInfoLight
UIButtonTypeRoundedRect
System Guides
8 Results
Appendix A: De...UIButton Methods
UIButton Class Reference
Defining Classes
Technical Note TN2239
iOS 2.2 to iOS 3.0 API Differences
Geocoding Location Data
Custom Views for Data Input
Legacy Map Techniques
Tools Guides
0 Results
Sample Code
Show 17 More Results
UICatalog
NavBar
Accessory
MapCallouts
TouchCells
TheElements
Next
UIButton Class Reference
Inherits from UIControl : UIView : UIResponder : NSObject
Conforms to NSCoding
NSCoding (UIView)
UIAppearance (UIView)
UIAppearanceContainer (UIView)
Framework /System/Library/Frameworks/UIKit.framework
Availability Available in iOS 2.0 and later.
Declared in UIButton.h
Related sample code AddMusic
avTouch
Popovers
TheElements
UICatalog
Overview
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.
Tasks
Now click on UIButton to switch to its documentation.
Stanford CS193p
Winter 2013


Devices
Repositories
Projects
Archives
Documentation
iOS 6.0 Documentation Set
User Experience
Controls
UIButton Class Reference
UIButton
Hits must
contain search term
Languages
All Languages
Find in
1 of 6 Doc Sets
Reference
8 Results
UIButton
UIButtonType
UIButtonTypeContactAdd
UIButtonTypeCustom
UIButtonTypeDetailDisclosure
UIButtonTypeInfoDark
UIButtonTypeInfoLight
UIButtonTypeRoundedRect
System Guides
8 Results
Appendix A: De...UIButton Methods
UIButton Class Reference
Defining Classes
Technical Note TN2239
iOS 2.2 to iOS 3.0 API Differences
Geocoding Location Data
Custom Views for Data Input
Legacy Map Techniques
Tools Guides
0 Results
Sample Code
Show 17 More Results
UICatalog
NavBar
Accessory
MapCallouts
TouchCells
TheElements
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
adjustsImageWhenHighlighted property
adjustsImageWhenDisabled property
showsTouchWhenHighlighted property
– backgroundImageForState:
– imageForState:
– setBackgroundImage:forState:
– setImage:forState:
tintColor property
Configuring Edge Insets
contentEdgeInsets property
titleEdgeInsets property
imageEdgeInsets property
Getting the Current State
Scroll down to see a list of all of UIButton's methods (not including those it inherits from its superclass).
Let's click on setTitle:forState:, for example, to find out more about it.
Stanford CS193p
Winter 2013

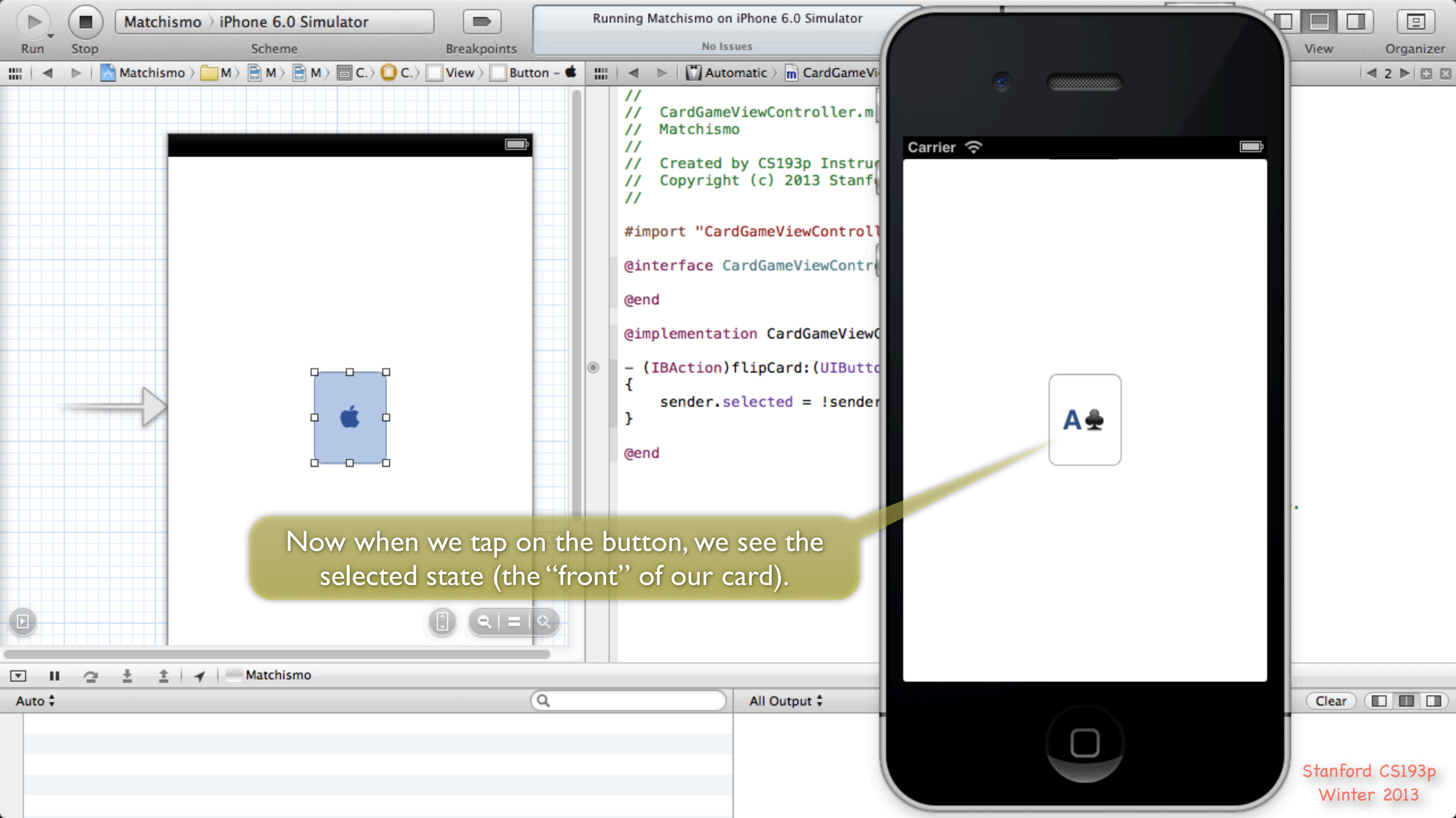

Devices
Repositories
Projects
Archives
Documentation
iOS 6.0 Documentation Set
User Experience
Controls
UIButton Class Reference
Instance Methods
setTitle:forState:
UIButton
Hits must
contain search term
Languages
All Languages
Find in
1 of 6 Doc Sets
Reference
8 Results
UIButton
UIButtonType
UIButtonTypeContactAdd
UIButtonTypeCustom
UIButtonTypeDetailDisclosure
UIButtonTypeInfoDark
UIButtonTypeInfoLight
UIButtonTypeRoundedRect
System Guides
8 Results
Appendix A: De...UIButton Methods
UIButton Class Reference
Defining Classes
Technical Note TN2239
iOS 2.2 to iOS 3.0 API Differences
Geocoding Location Data
Custom Views for Data Input
Legacy Map Techniques
Tools Guides
0 Results
Sample Code
Show 17 More Results
UICatalog
NavBar
Accessory
MapCallouts
TouchCells
TheElements
setTitle:forState:
Sets the title to use for the specified state.
- (void)setTitle:(NSString *)title forState:(UIControlState)state
Parameters
title
The title to use for the specified state.
state
The state that uses the specified title. The possible values are described in UIControlState.
Discussion
Use this method to set the title for the button. The title you specify derives its formatting from the button's associated label object. If you set both a title and an attributed title for the button, the button prefers the use of the attributed title over this one.
At a minimum, you should set the value for the normal state. If a title is not specified for a state, the default behavior is to use the title associated with the UIControlStateNormal state. If the value for UIControlStateNormal is not set, then the property defaults to a system value.
Availability
Available in iOS 2.0 and later.
See Also
– titleForState:
Related Sample Code
UICatalog
Declared In
UIButton.h
setTitleColor:forState:
Sets the color of the title to use for the specified state.
- (void)setTitleColor:(UIColor *)color forState:(UIControlState)state
Parameters
color
The color of the title to use for the specified state.
It says here that this lets you programmatically do what we did in the Inspector in Xcode (i.e. set the title for a given button state).
Let's click this link to find out more about how you specify the state when you are setting the title for that state.
Stanford CS193p
Winter 2013

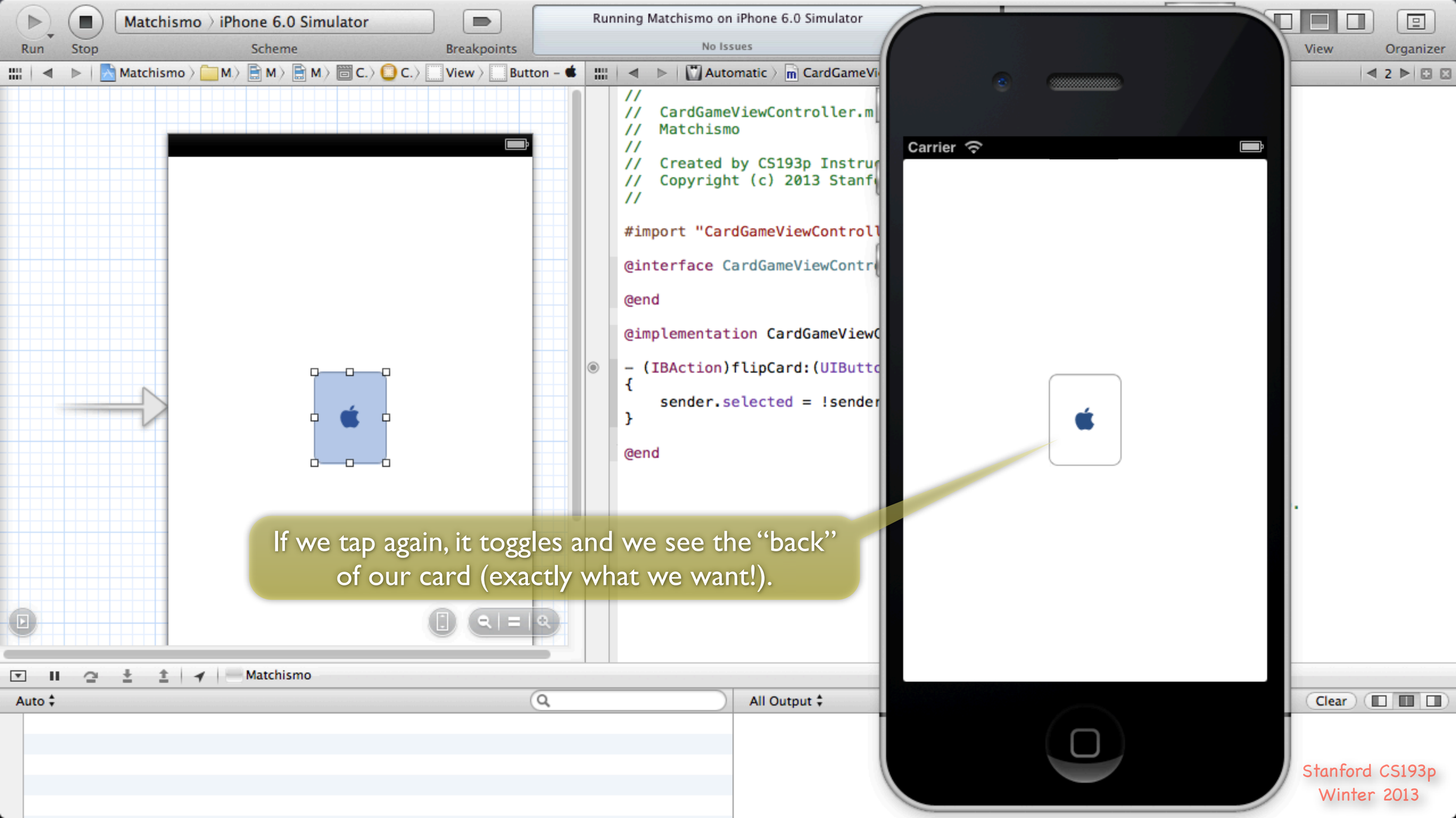

Devices
Repositories
Projects
Archives
Documentation
iOS 6.0 Documentation Set
User Experience
Controls
UIControl Class Reference
Constants
UIControlState
UIButton
Hits must
contain search term
Languages
All Languages
Find in
1 of 6 Doc Sets
Reference
8 Results
UIButton
UIButtonType
UIButtonTypeContactAdd
UIButtonTypeCustom
UIButtonTypeDetailDisclosure
UIButtonTypeInfoDark
UIButtonTypeInfoLight
UIButtonTypeRoundedRect
System Guides
8 Results
Appendix A: De...UIButton Methods
UIButton Class Reference
Defining Classes
Technical Note TN2239
iOS 2.2 to iOS 3.0 API Differences
Geocoding Location Data
Custom Views for Data Input
Legacy Map Techniques
Tools Guides
0 Results
Sample Code
Show 17 More Results
UICatalog
NavBar
Accessory
MapCallouts
TouchCells
TheElements
Control State
The state of a control; a control can have more than one state at a time. States are recognized differently depending on the control. For example, a UIButton instance may be configured (using the setImage:forState: method) to display one image when it is in its normal state and a different image when it is highlighted.
enum {
UIControlStateNormal = 0,
UIControlStateHighlighted = 1 << 0,
UIControlStateDisabled = 1 << 1,
UIControlStateSelected = 1 << 2,
UIControlStateApplication = 0x00FF0000,
UIControlStateReserved = 0xFF000000
};
Aha! This looks like a bitmask.
So we could set the title that appears when the button is both Disabled and Selected with UIControlStateSelected|UIControlStateDisabled.
I'll bet that ends up being useful!
Constants
UIControlStateNormal
The normal, or default state of a control—that is, enabled but neither selected nor highlighted.
Available in iOS 2.0 and later.
Declared in UIControl.h.
UIControlStateHighlighted
Highlighted state of a control. A control enters this state when a touch enters and exits during tracking and when there is a touch up event. You can retrieve and set this value through the highlighted property.
Available in iOS 2.0 and later.
Declared in UIControl.h.
UIControlStateDisabled
Disabled state of a control. This state indicates that the control is currently disabled. You can retrieve and set this value through the enabled property.
Available in iOS 2.0 and later.
Declared in UIControl.h.
UIControlStateSelected
Selected state of a control. For many controls, this state has no effect on behavior or appearance. But other subclasses (for example, the UISegmentedControl class) may have different appearance depending on their selected state. You can retrieve and set this value through the selected property.
Available in iOS 2.0 and later.
Declared in UIControl.h.
Stanford CS193p
Winter 2013

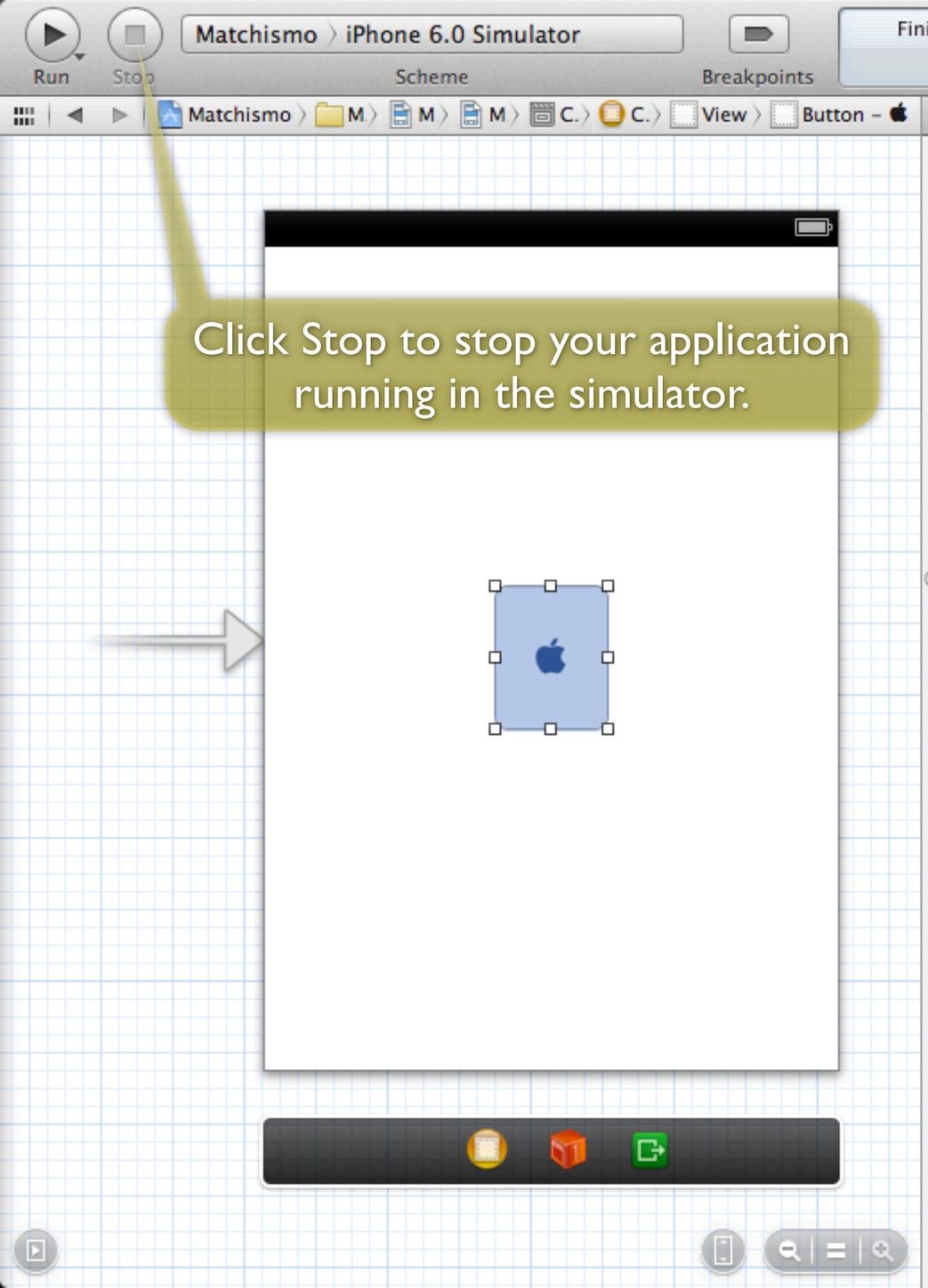
Let's go back to the main window in Xcode and press Run now that we've attached some code to the button to toggle the selected state.

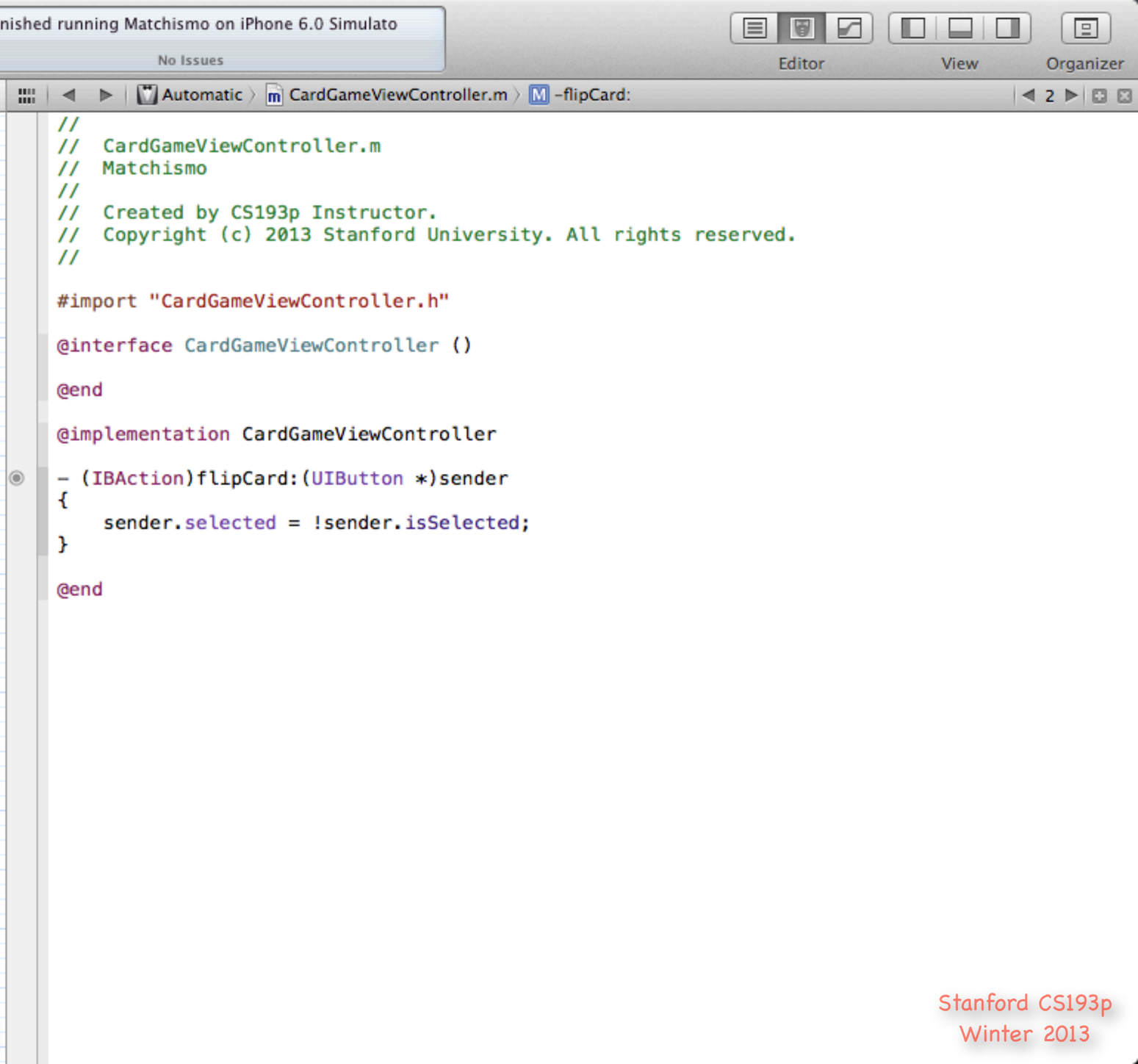


Stanford CS193p
Winter 2013

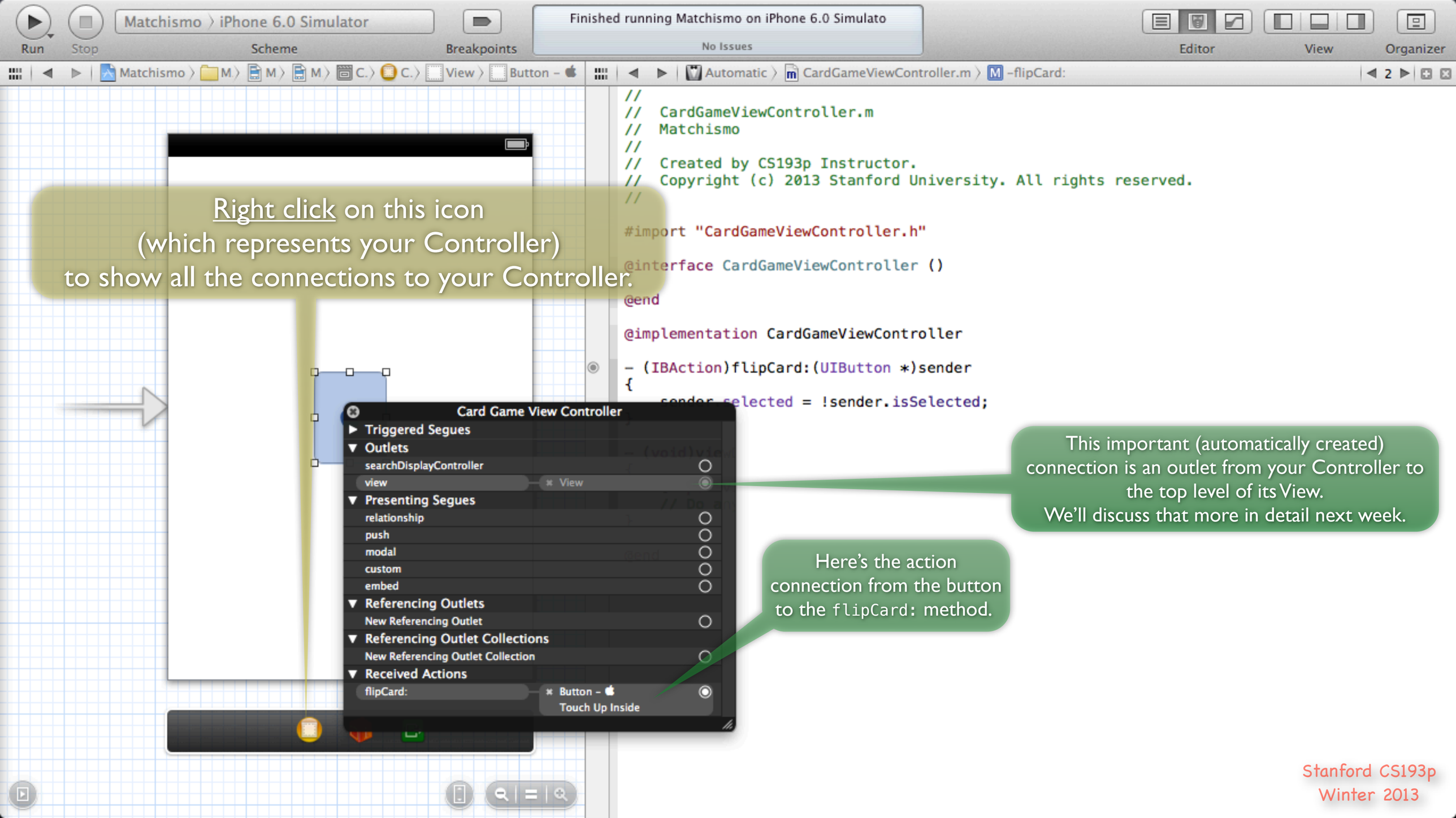

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameVi
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instruc
// Copyright (c) 2013 Stanf
//
#import "CardGameViewControll
@interface CardGameViewContr
@end
@implementation CardGameViewC
- (IBAction)flipCard:(UIButto
{
sender.selected = !sender
}
@end
Carrier
A♣
Now when we tap on the button, we see the selected state (the “front” of our card).
Matchismo
Auto
All Output
Clear
Stanford CS193p
Winter 2013

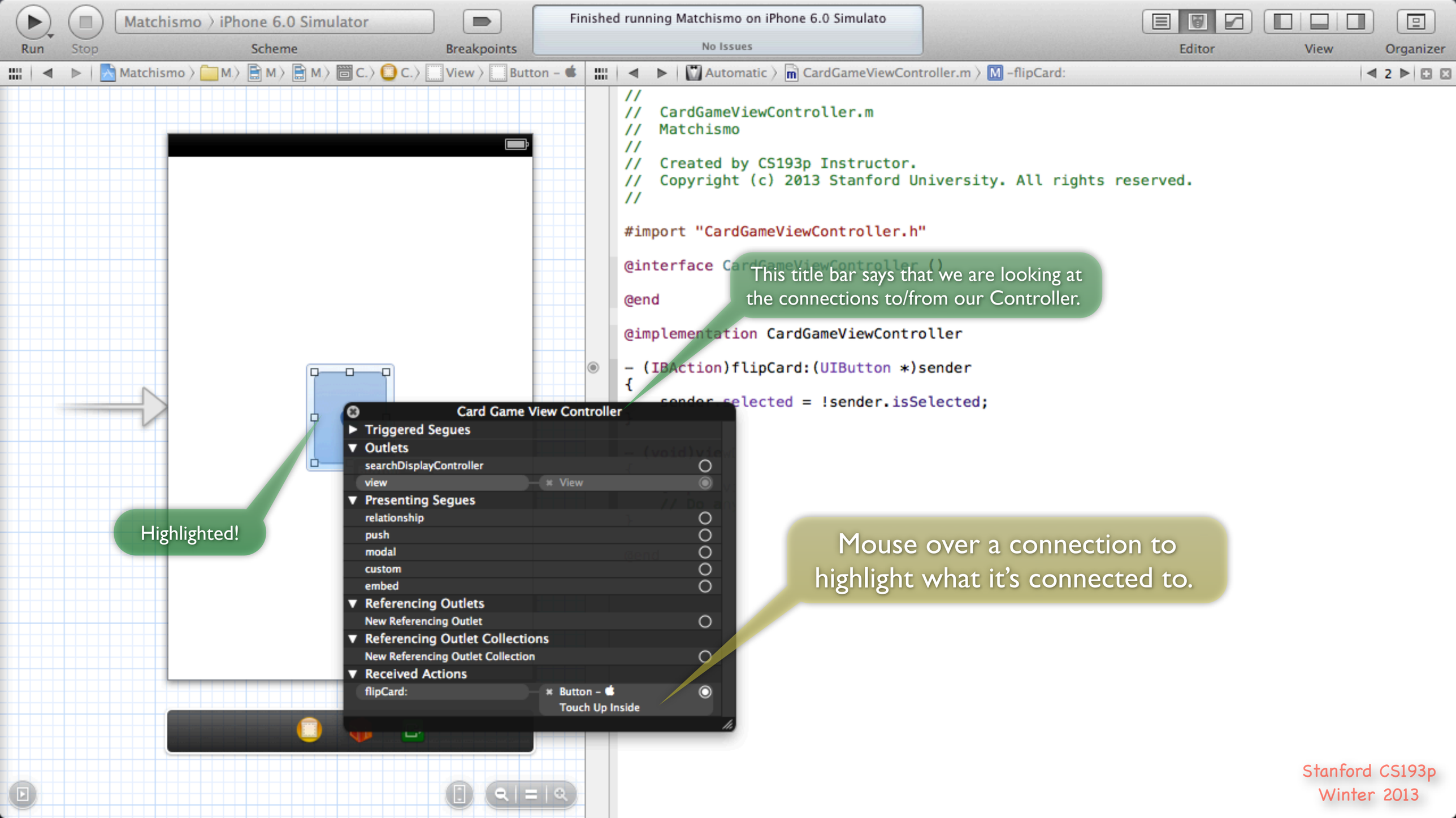

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button – 〉
Automatic 〉 CardGameVi
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instru
// Copyright (c) 2013 Stanf
//
#import "CardGameViewControll
@interface CardGameViewContr
@end
@implementation CardGameViewC
- (IBAction)flipCard:(UIButto
{
sender.selected = !sender
}
@end
Carrier
If we tap again, it toggles and we see the “back” of our card (exactly what we want!).
Matchismo
Auto
All Output
Clear
Stanford CS193p
Winter 2013

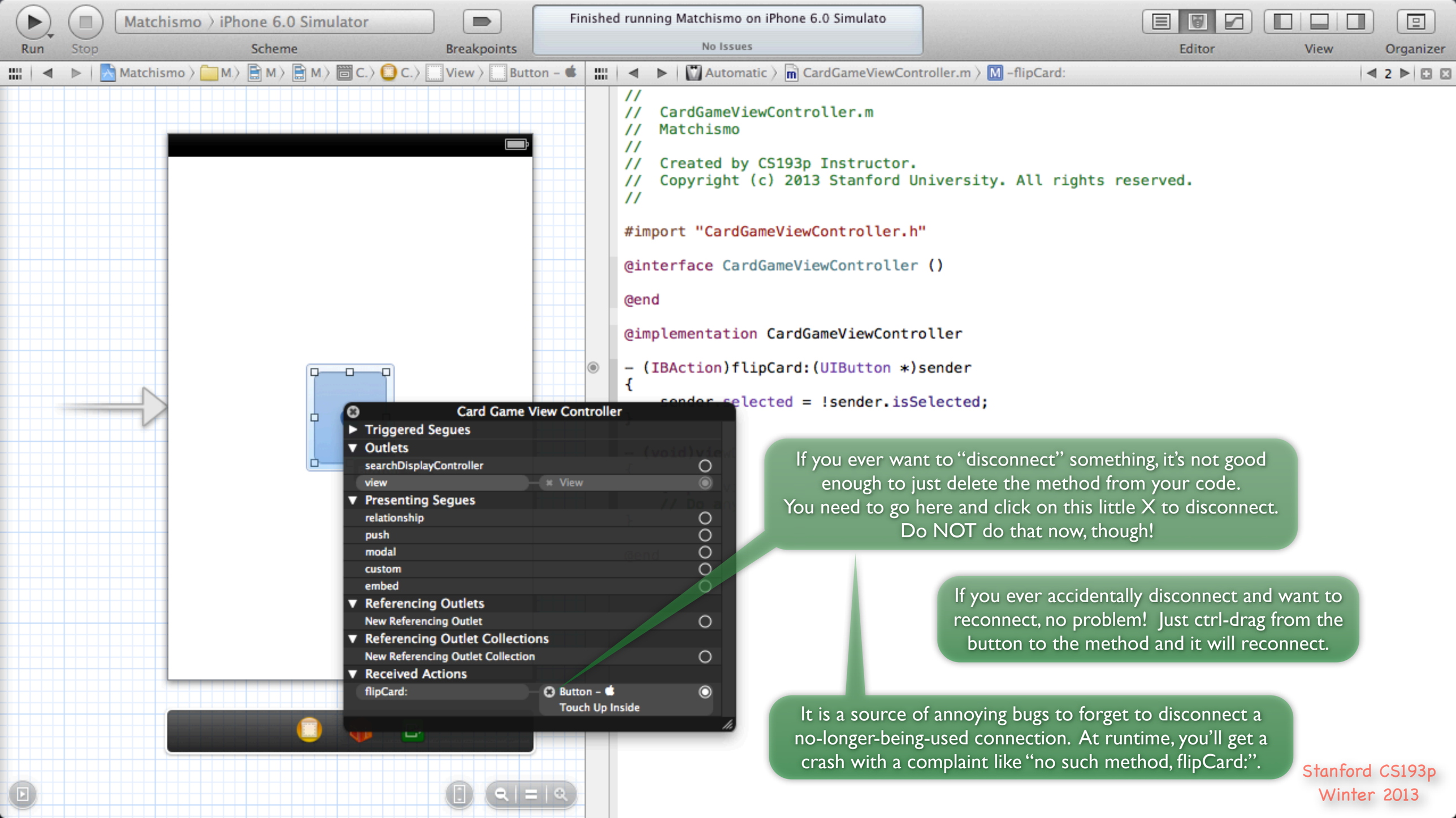


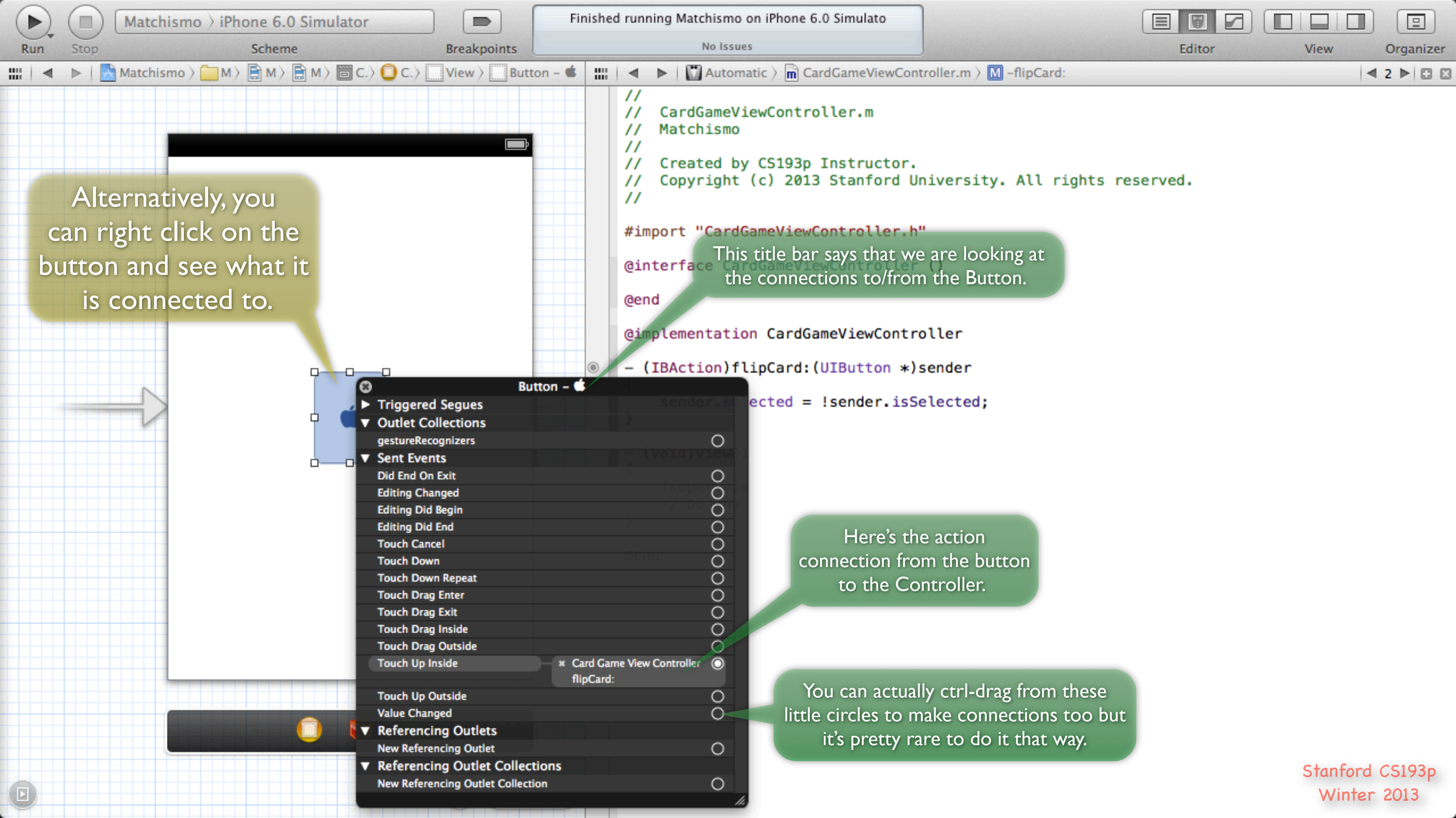


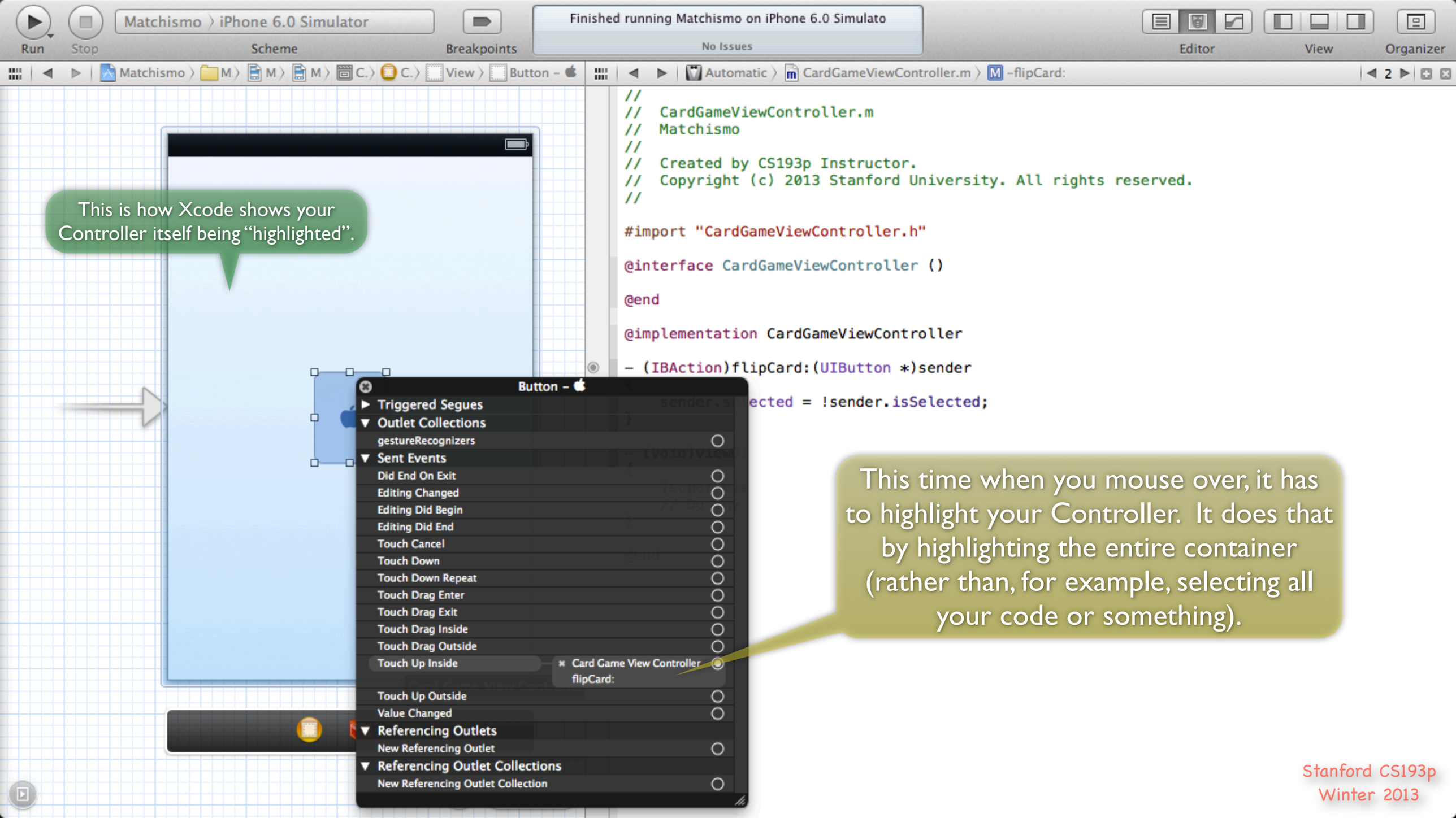

```
//
//  CardGameViewController.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University. All rights reserved.
//

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}

@end
```

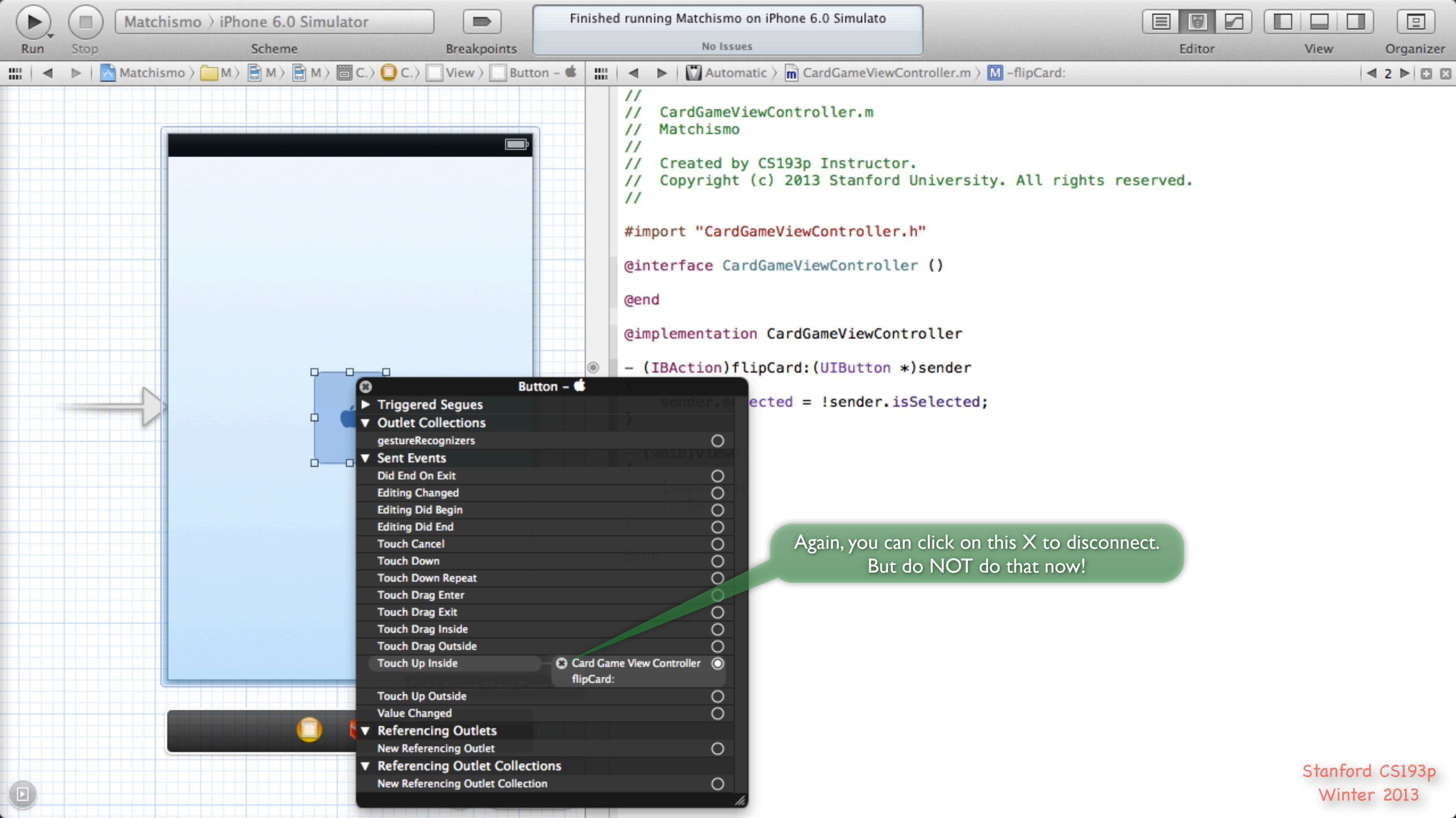

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
Right click on this icon
(which represents your Controller)
to show all the connections to your Controller.
Card Game View Controller
Triggered Segues
Outlets
searchDisplayController
view
View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
flipCard:
Button –
Touch Up Inside
This important (automatically created) connection is an outlet from your Controller to the top level of its View.
We'll discuss that more in detail next week.
Here's the action connection from the button to the flipCard: method.
Stanford CS193p
Winter 2013

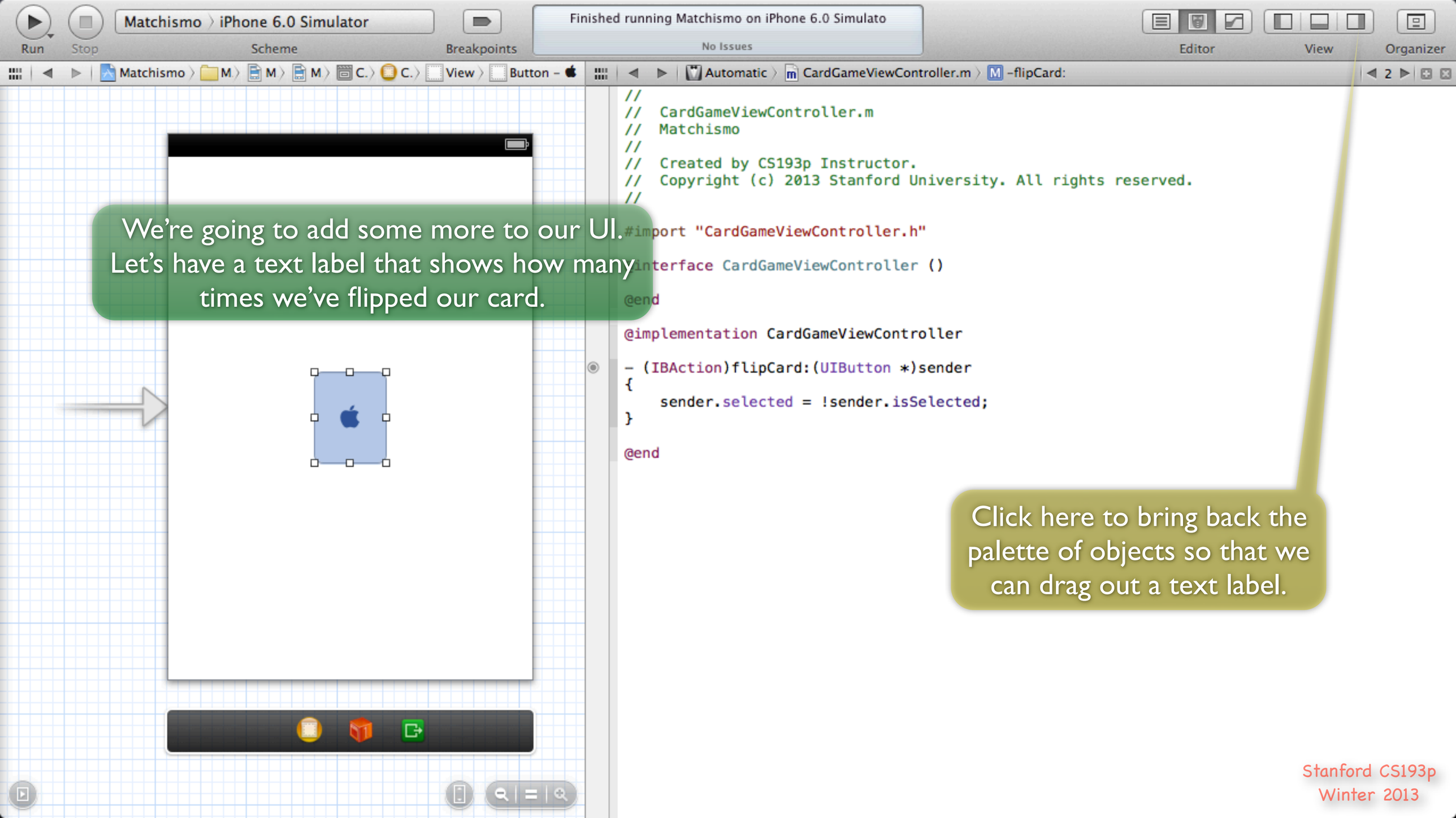

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
This title bar says that we are looking at the connections to/from our Controller.
Highlighted!
Card Game View Controller
Triggered Segues
Outlets
searchDisplayController
view
View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
flipCard:
Button –
Touch Up Inside
Mouse over a connection to highlight what it's connected to.
Stanford CS193p
Winter 2013

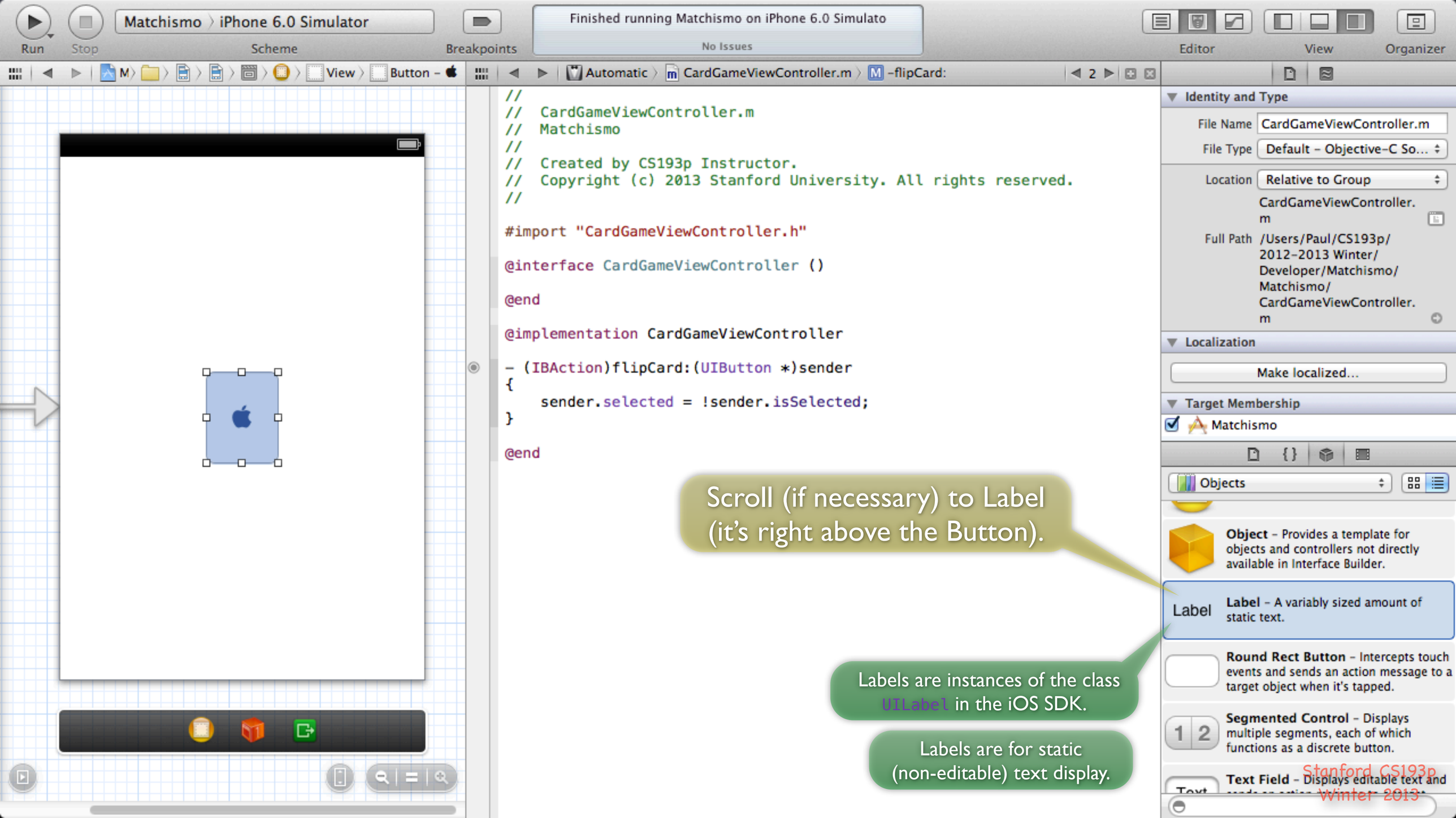

Run
Stop
Matchismo 〉iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉M 〉M 〉M 〉C 〉C 〉View 〉Button –
Automatic 〉CardGameViewController.m 〉-flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
Card Game View Controller
Triggered Segues
Outlets
searchDisplayController
view
View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
flipCard:
Button –
Touch Up Inside
If you ever want to "disconnect" something, it's not good enough to just delete the method from your code.
You need to go here and click on this little X to disconnect.
Do NOT do that now, though!
If you ever accidentally disconnect and want to reconnect, no problem! Just ctrl-drag from the button to the method and it will reconnect.
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, flipCard:".
Stanford CS193p
Winter 2013

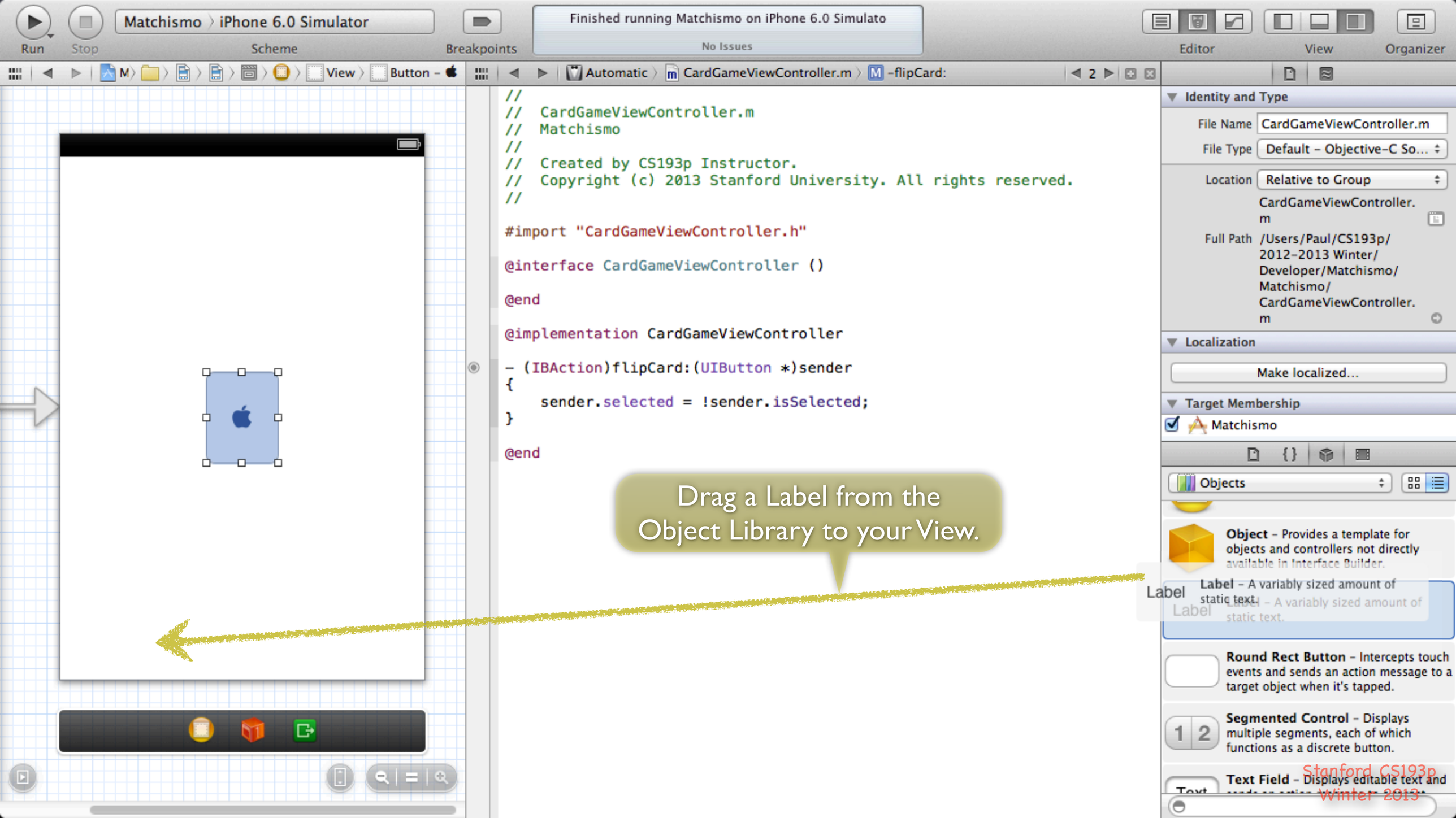

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
ected = !sender.isSelected;
Alternatively, you can right click on the button and see what it is connected to.
This title bar says that we are looking at the connections to/from the Button.
Button –
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
flipCard:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Here's the action connection from the button to the Controller.
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
Winter 2013

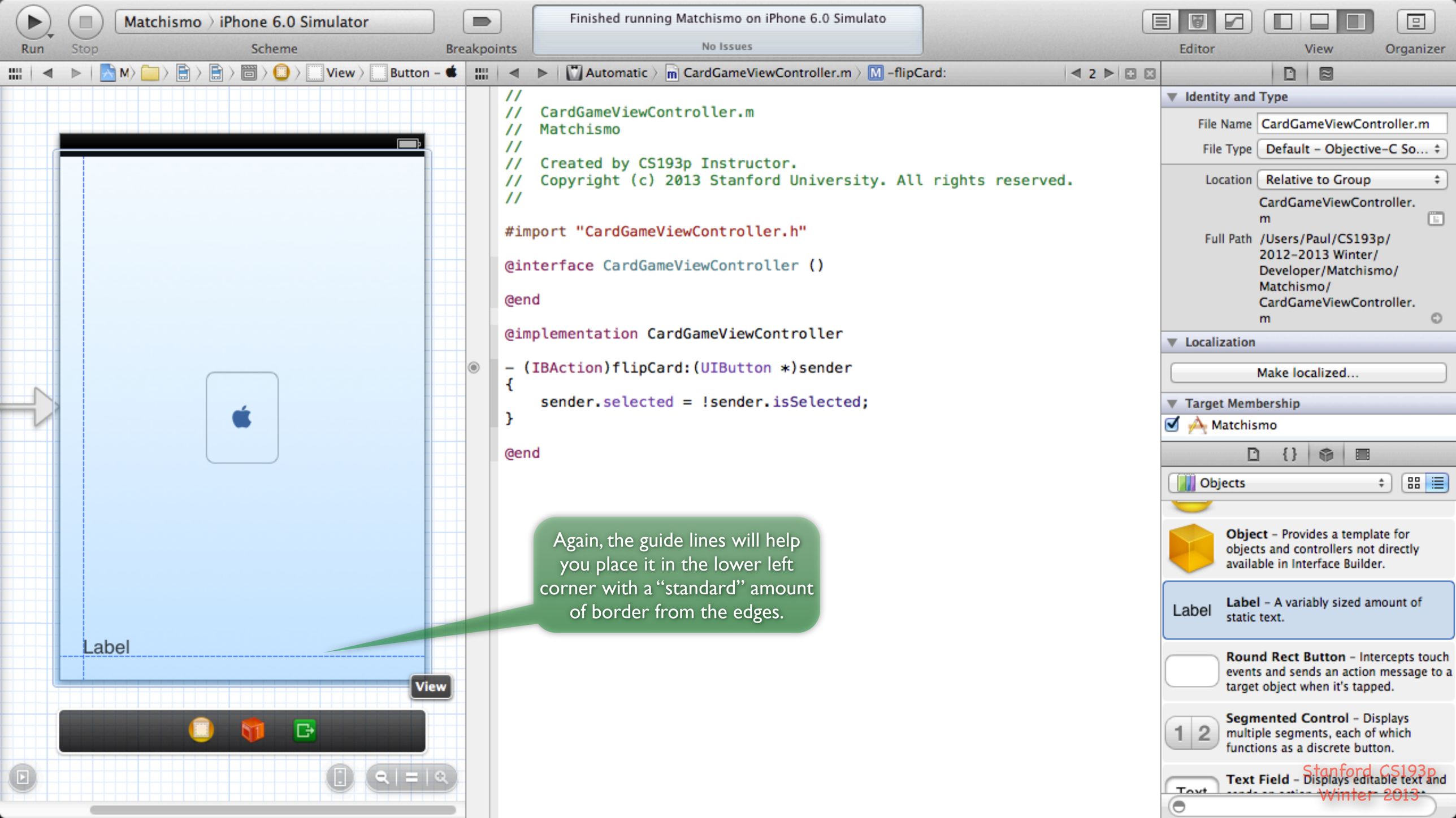

Run
Stop
Matchismo 〉iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉M 〉M 〉M 〉C. 〉C. 〉View 〉Button –
Automatic 〉CardGameViewController.m 〉-flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
ected = !sender.isSelected;
This is how Xcode shows your Controller itself being "highlighted".
Button –
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
flipCard:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
This time when you mouse over, it has to highlight your Controller. It does that by highlighting the entire container (rather than, for example, selecting all your code or something).
Stanford CS193p
Winter 2013

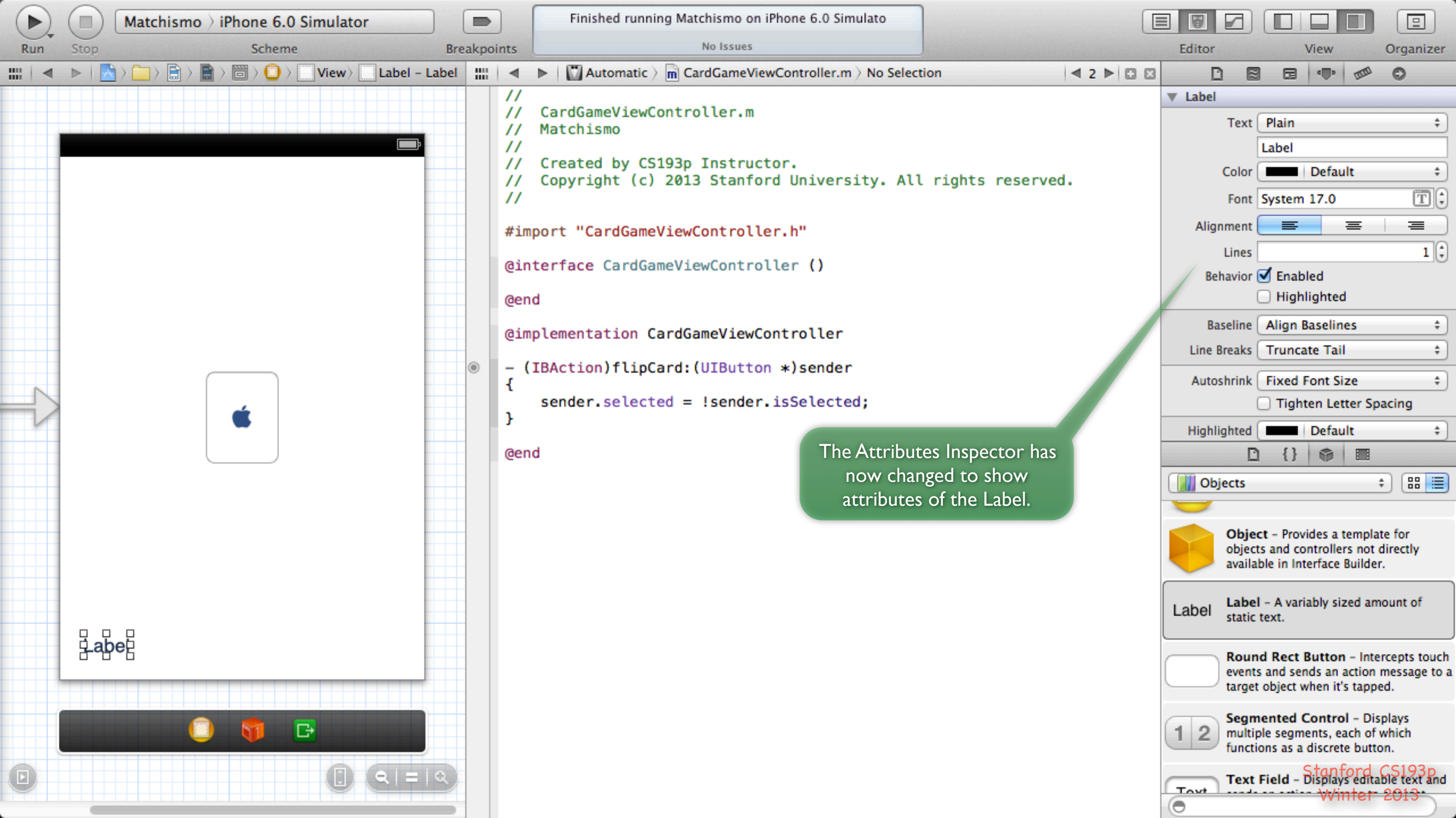

Run
Stop
Matchismo 〉iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉M 〉M 〉M 〉C. 〉C. 〉View 〉Button –
Automatic 〉CardGameViewController.m 〉-flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
ected = !sender.isSelected;
Button –
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
flipCard:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Again, you can click on this X to disconnect.
But do NOT do that now!
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo 〉 M 〉 M 〉 M 〉 C. 〉 C. 〉 View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
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)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}
@end
We're going to add some more to our UI. Let's have a text label that shows how many times we've flipped our card.
Click here to bring back the palette of objects so that we can drag out a text label.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Identity and Type
File Name CardGameViewController.m
File Type Default – Objective-C So…
Location Relative to Group
CardGameViewController.m
Full Path /Users/Paul/CS193p/2012-2013 Winter/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized…
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Scroll (if necessary) to Label (it's right above the Button).
Labels are instances of the class UILabel in the iOS SDK.
Labels are for static (non-editable) text display.
Stanford CS193p
Winter 2013

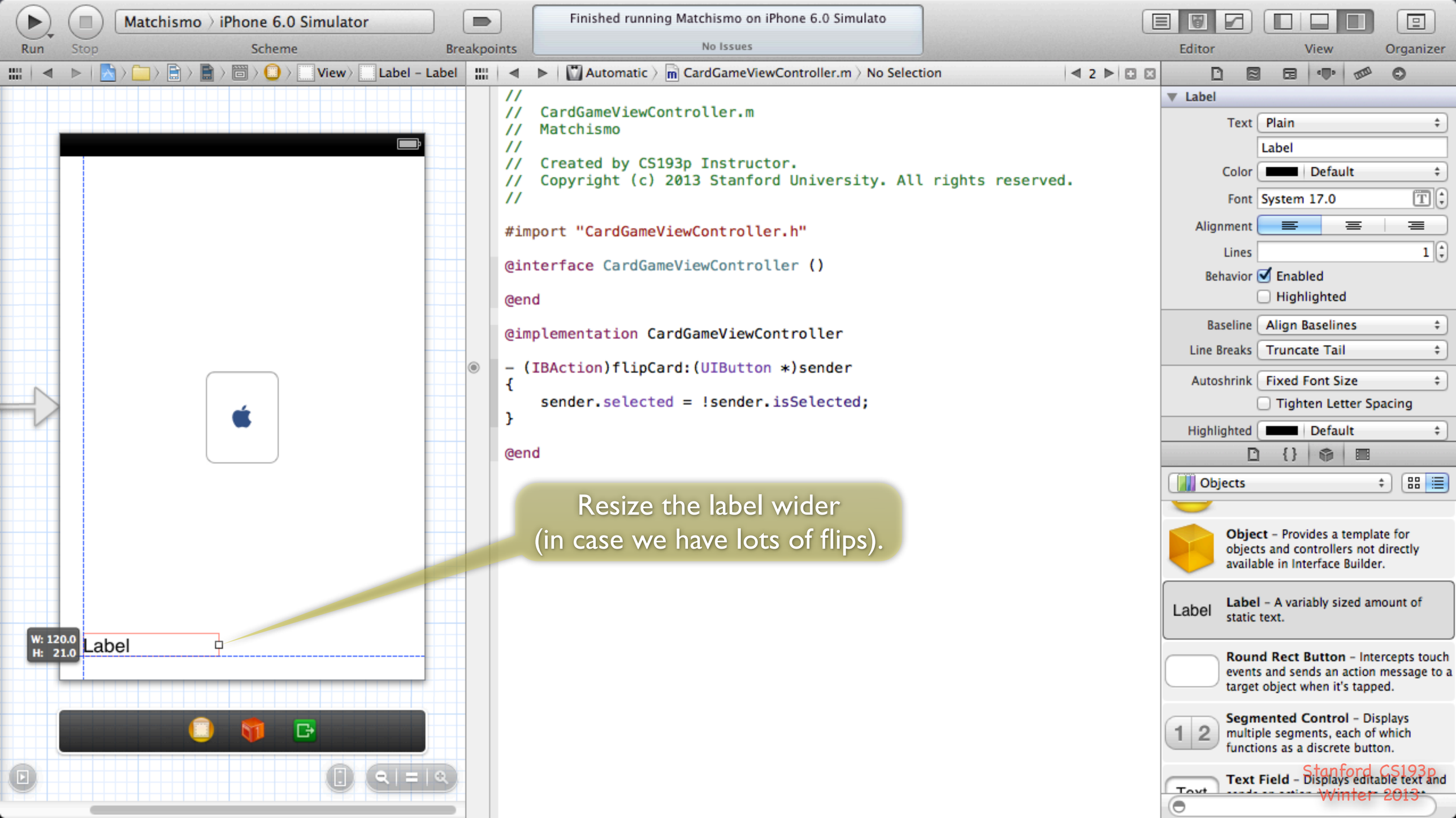

Matchismo 〉 iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Automatic 〉 CardGameViewController.m 〉 -flipCard:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Drag a Label from the Object Library to your View.
Identity and Type
File Name CardGameViewController.m
File Type Default – Objective-C So...
Location Relative to Group
CardGameViewController.m
Full Path /Users/Paul/CS193p/2012-2013 Winter/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized...
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

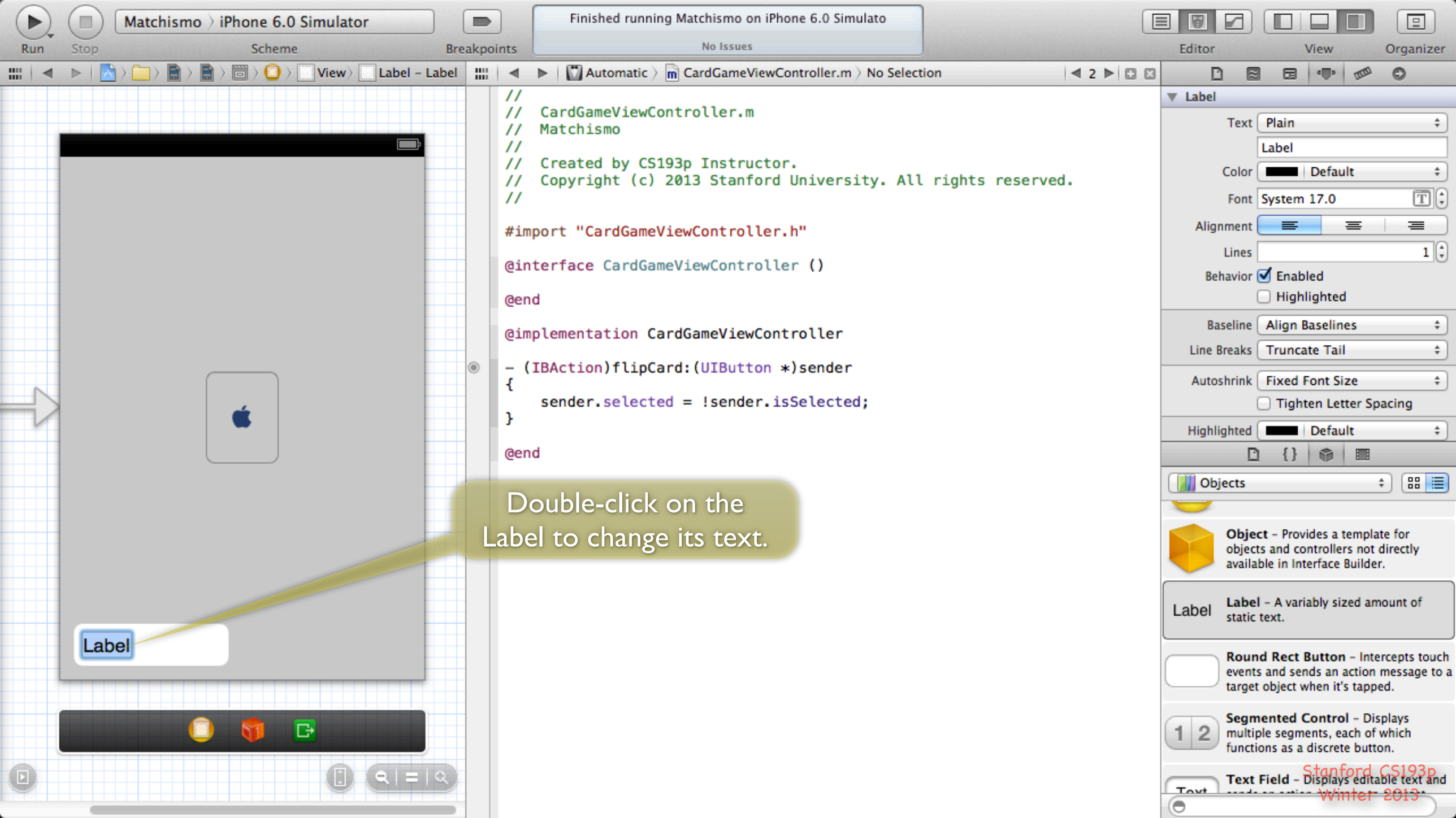

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
View 〉 Button –
Automatic 〉 CardGameViewController.m 〉 -flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Label
View
Again, the guide lines will help you place it in the lower left corner with a "standard" amount of border from the edges.
Identity and Type
File Name CardGameViewController.m
File Type Default – Objective-C So...
Location Relative to Group
CardGameViewController.m
Full Path /Users/Paul/CS193p/2012-2013 Winter/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized...
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013



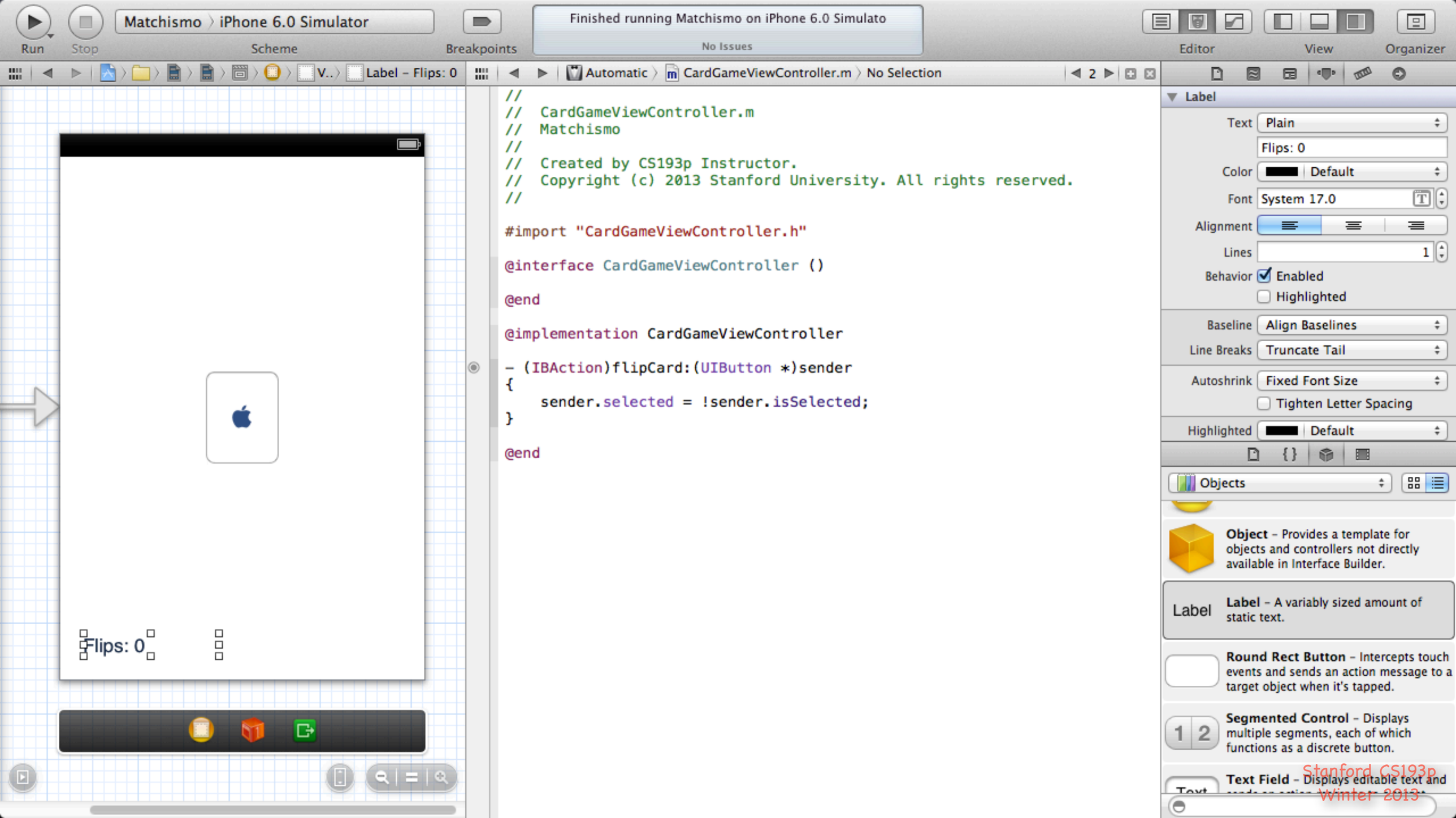

```
//
//  CardGameViewController.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University. All rights reserved.
//

#import "CardGameViewController.h"

@interface CardGameViewController ()

@end

@implementation CardGameViewController

- (IBAction)flipCard:(UIButton *)sender
{
    sender.selected = !sender.isSelected;
}

@end
```

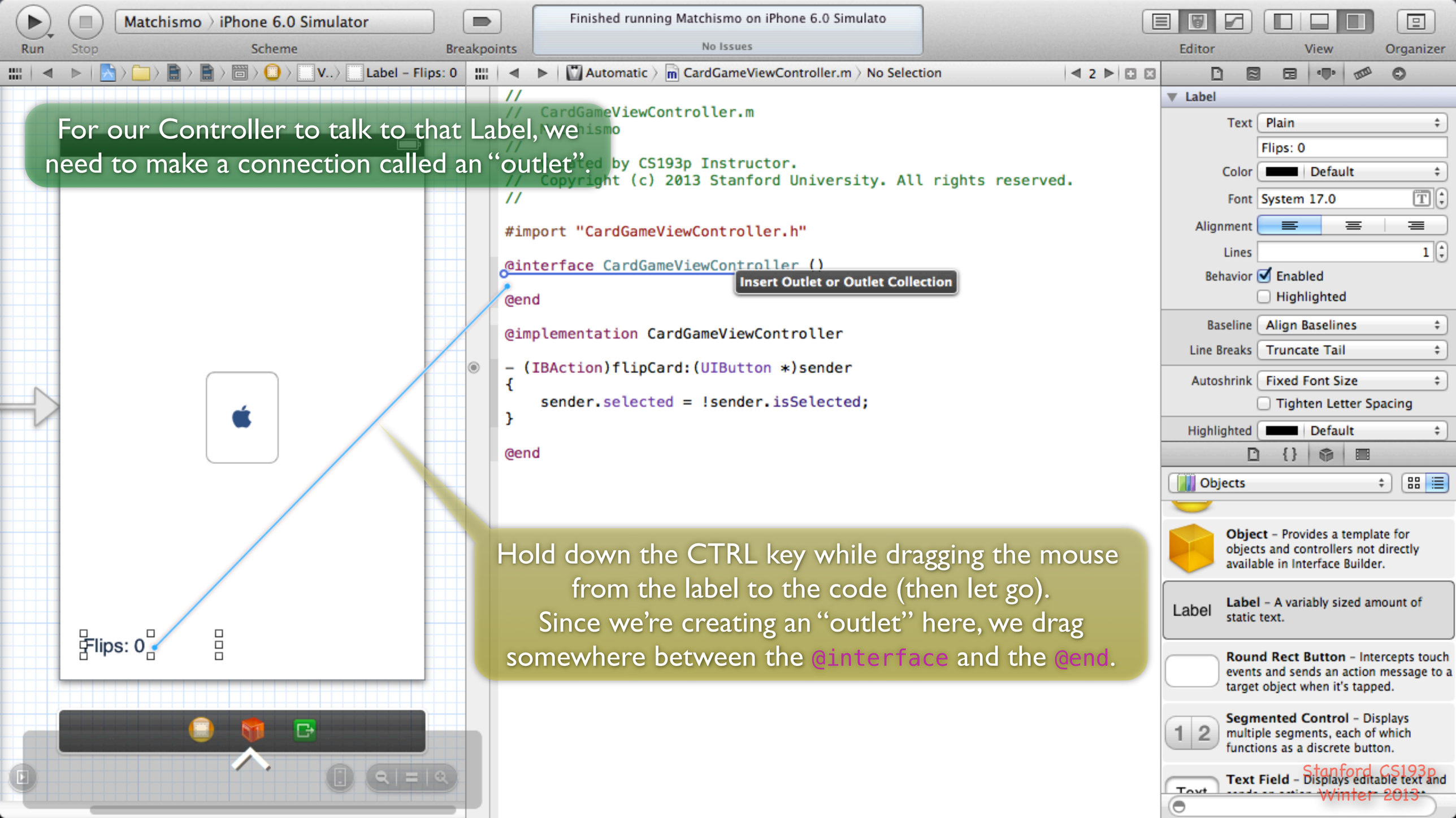
The Attributes Inspector has now changed to show attributes of the Label.

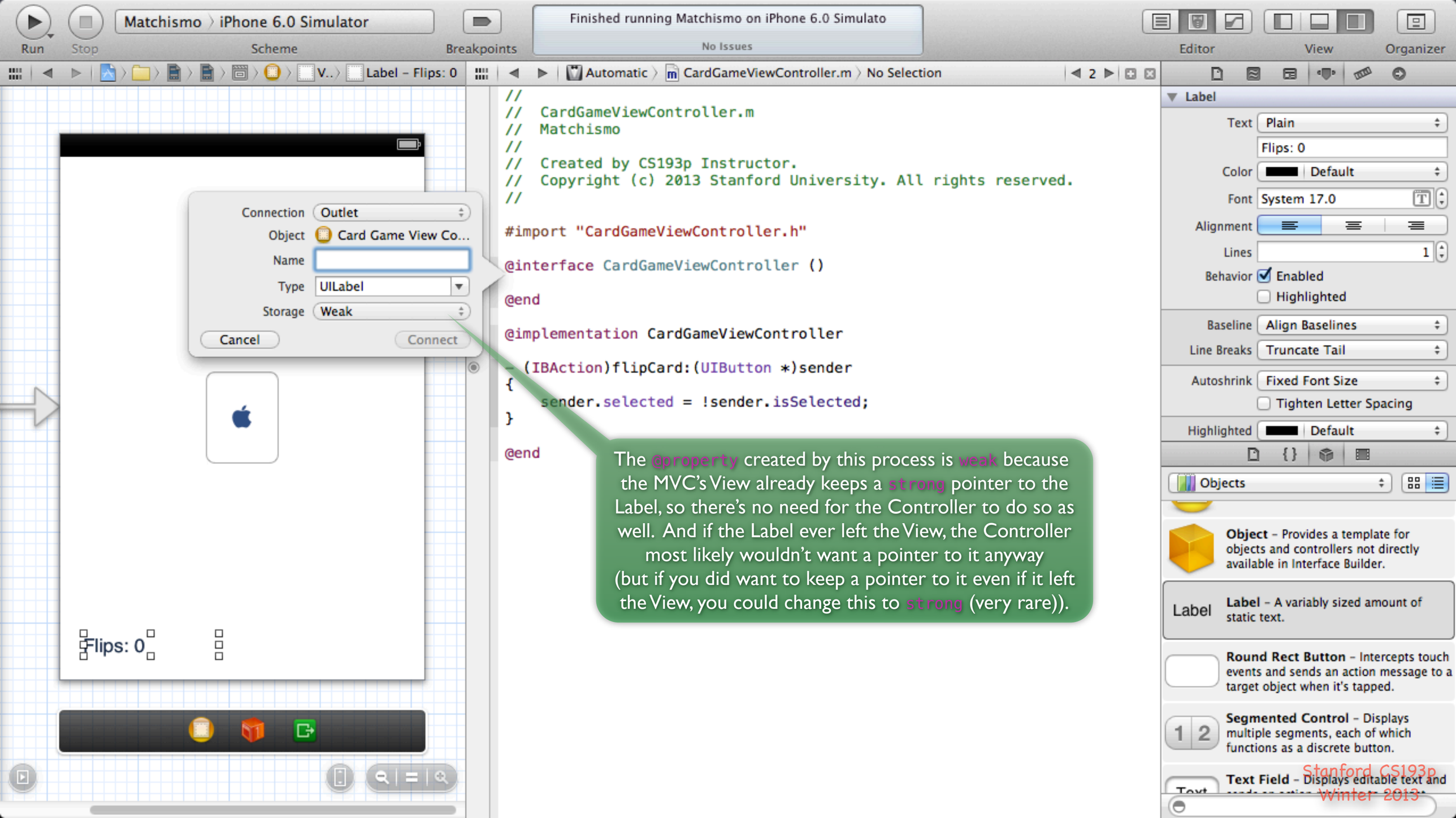

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
View 〉 Label – Label
Automatic 〉 CardGameViewController.m 〉 No Selection
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Label
Text Plain
Label
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
W: 120.0
H: 21.0
Label
Resize the label wider
(in case we have lots of flips).
Stanford CS193p
Winter 2013

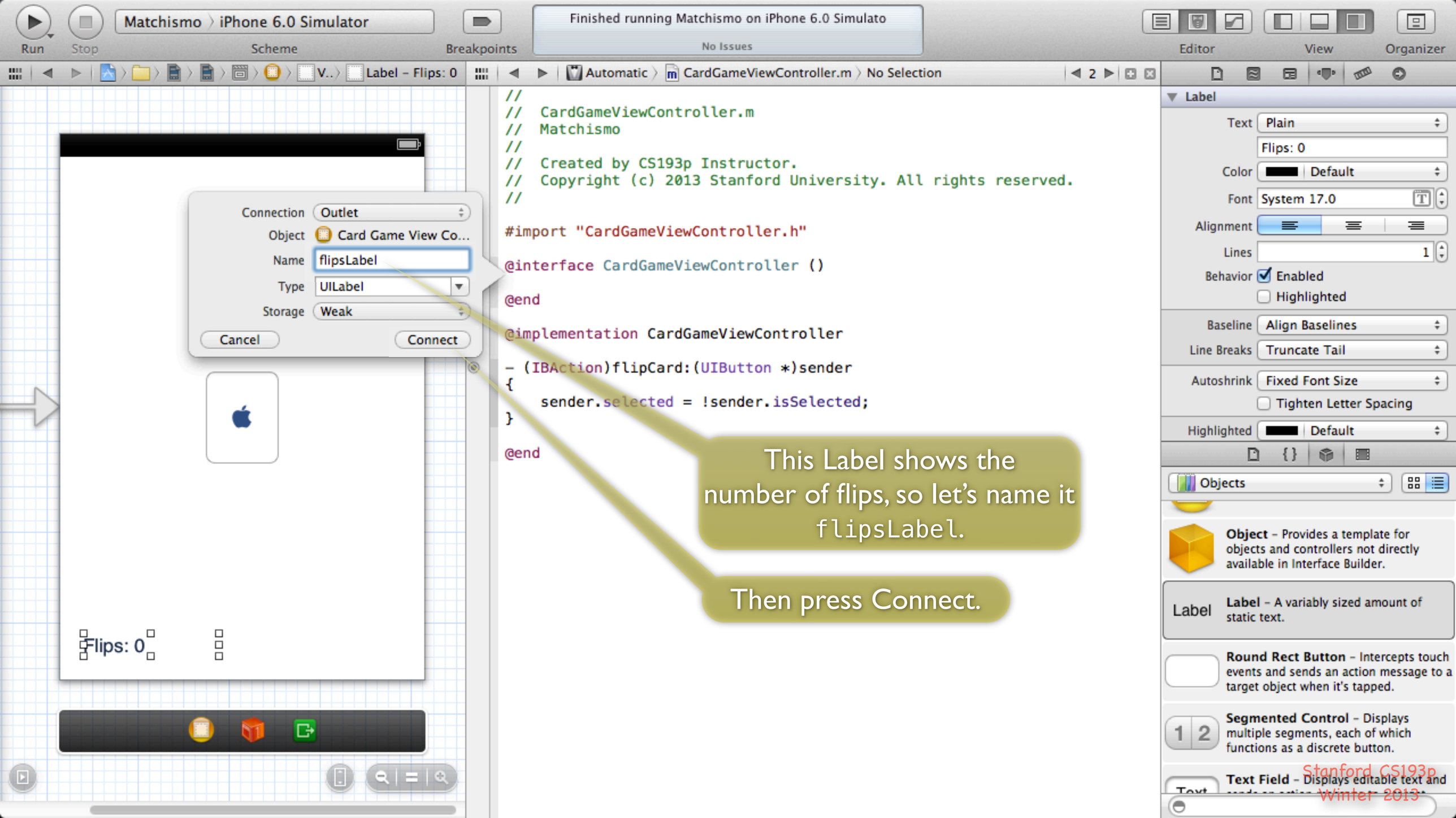

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
View 〉 Label – Label
Automatic 〉 CardGameViewController.m 〉 No Selection
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Label
Double-click on the
Label to change its text.
Label
Text Plain
Label
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

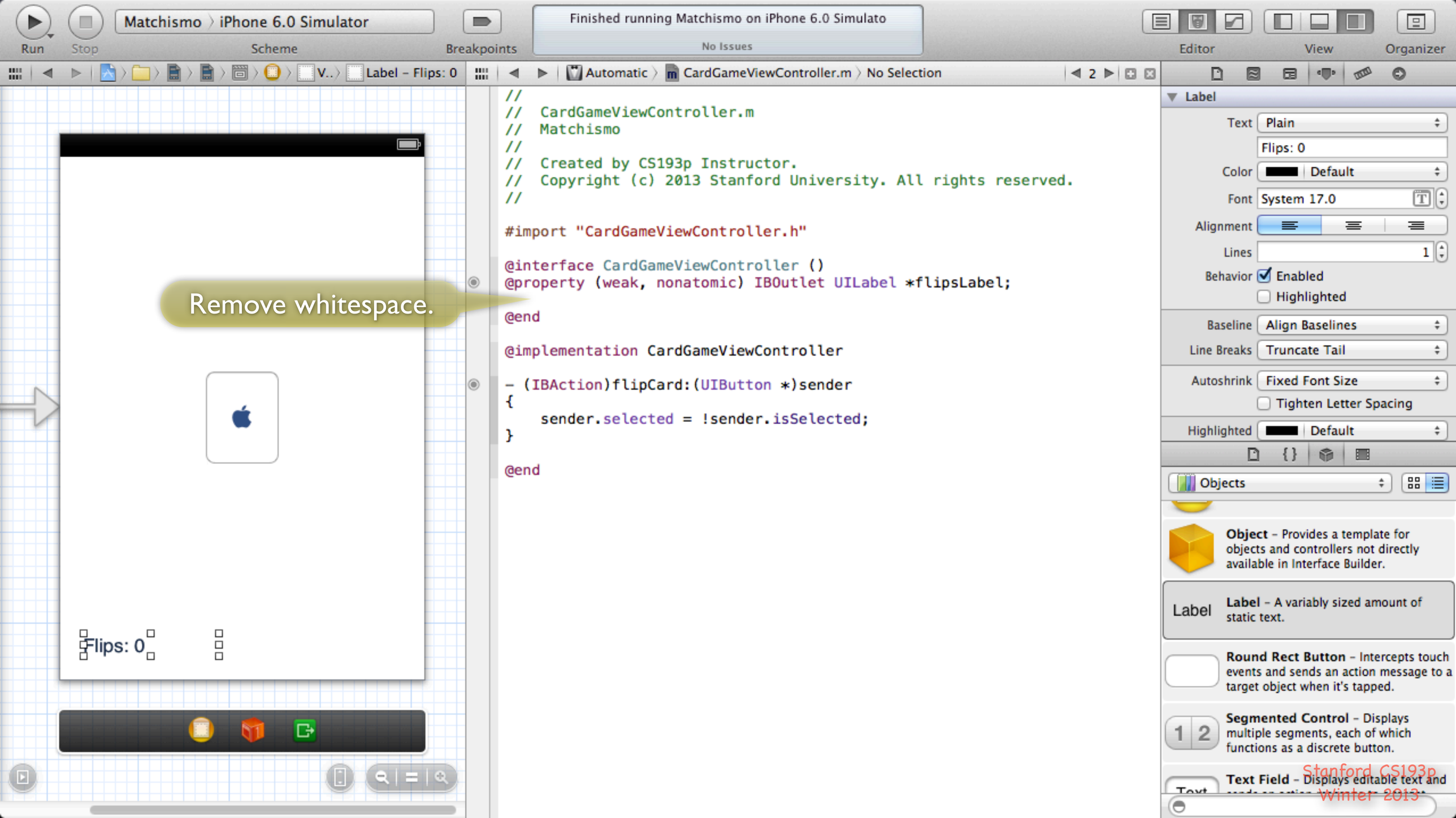

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
View 〉 Label – Label
Automatic 〉 CardGameViewController.m 〉 No Selection
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
(IBAction) flipCard:(UIButton *)sender
sender.selected = !sender.isSelected;
}
@end
We're actually going to set this text entirely from our code, but putting "Flips: 0" here just helps remind us what this Label is for when we're here in Xcode.
Set the text to "Flips: 0"
Flips: 0
Label
Text Plain
Label
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

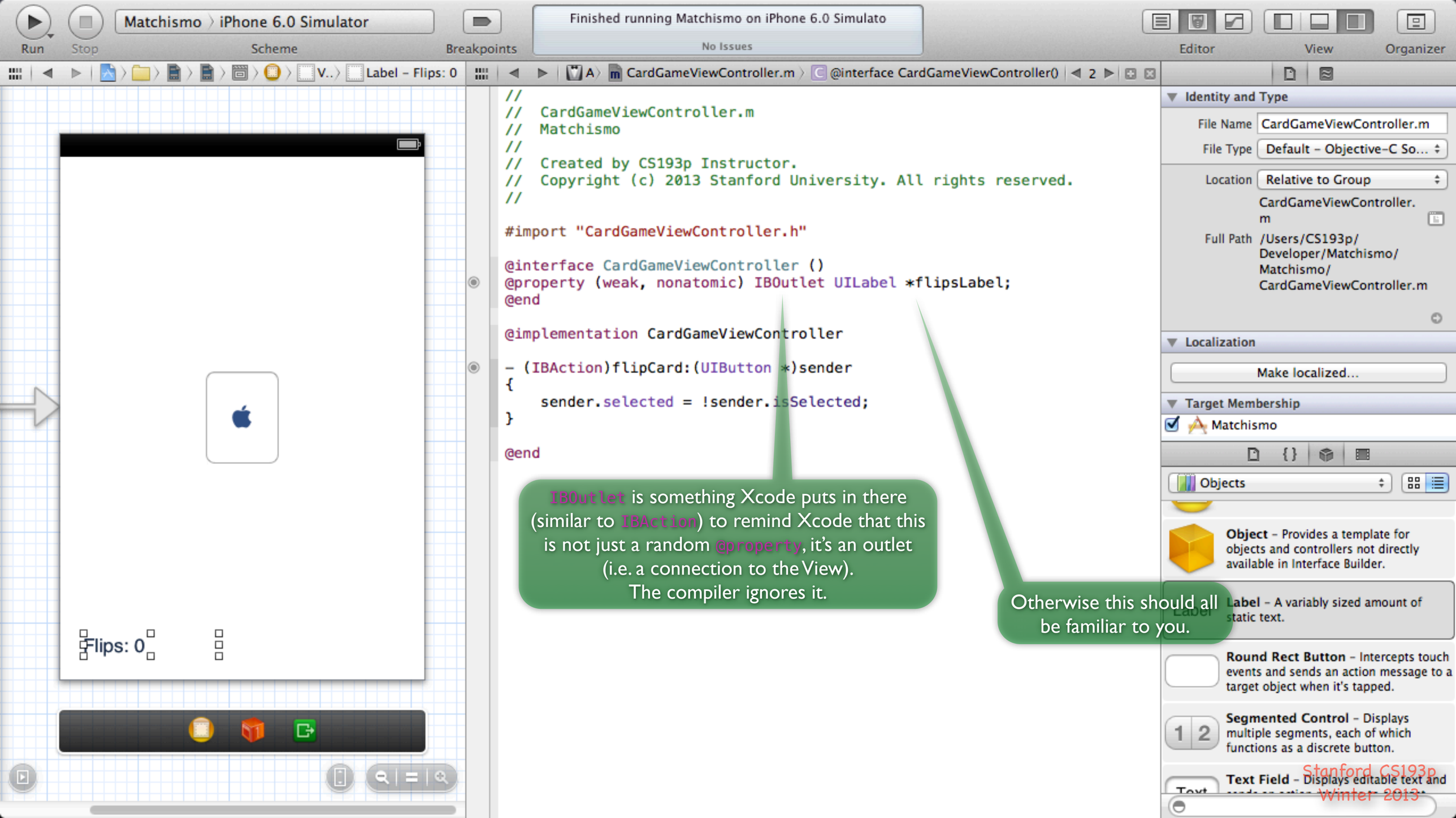

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Automatic 〉 CardGameViewController.m 〉 No Selection
2
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Label
Text Plain
Flips: 0
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

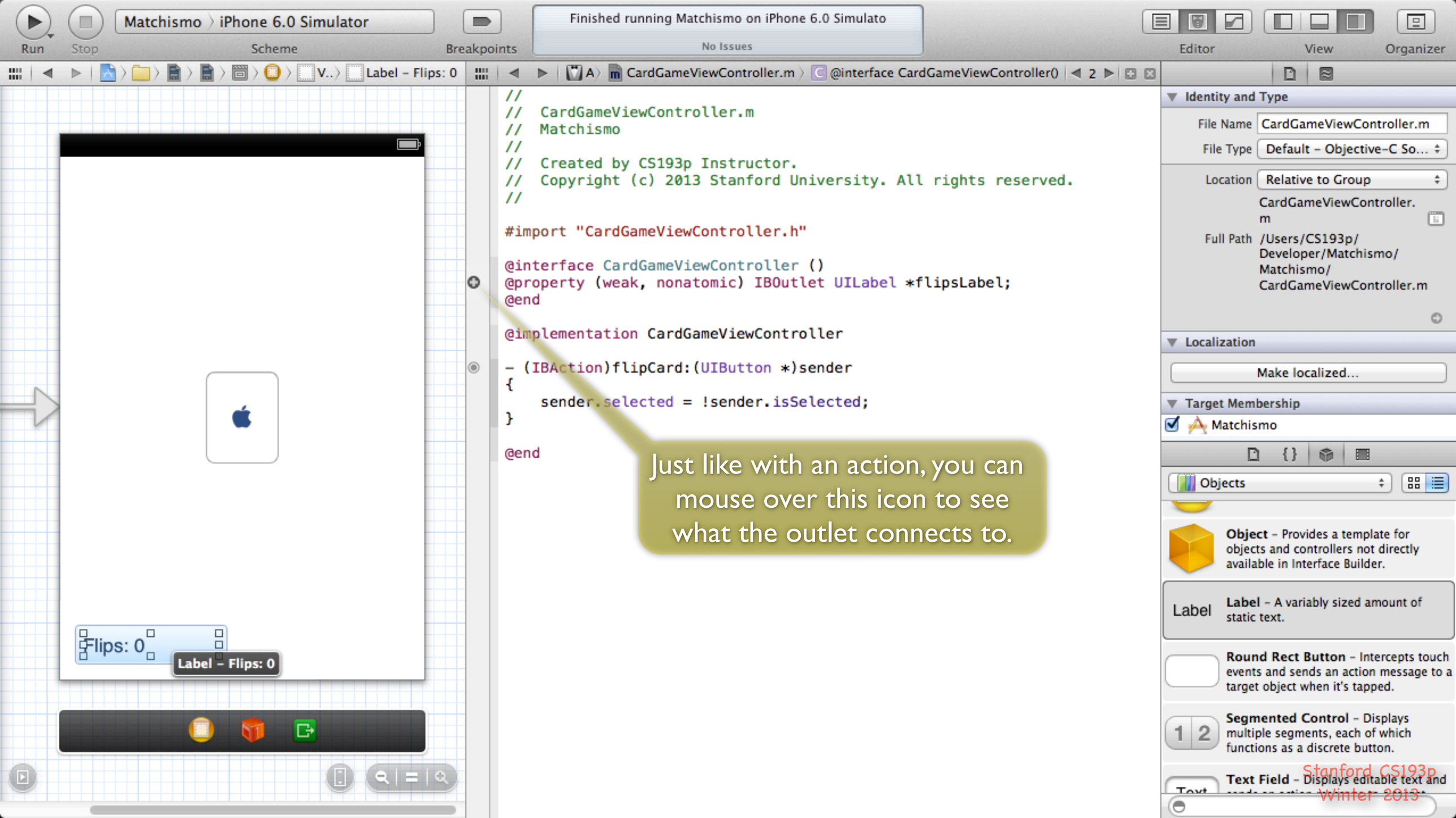

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Automatic
CardGameViewController.m
No Selection
For our Controller to talk to that Label, we need to make a connection called an "outlet".
//
// 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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Flips: 0
Hold down the CTRL key while dragging the mouse from the label to the code (then let go).
Since we're creating an "outlet" here, we drag somewhere between the @interface and the @end.
Label
Text Plain
Flips: 0
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

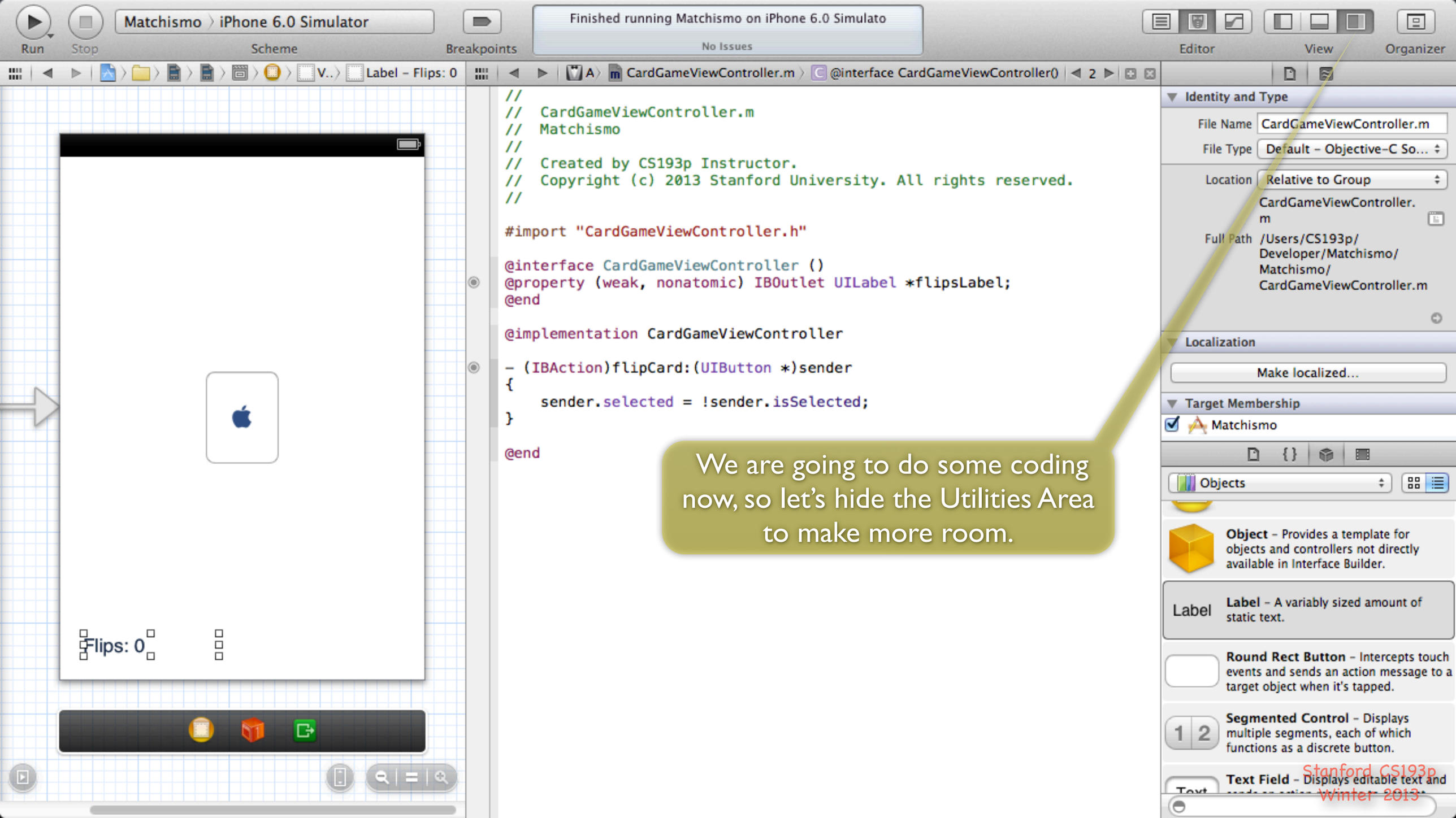

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Automatic 〉 CardGameViewController.m 〉 No Selection
Connection Outlet
Object Card Game View Co...
Name
Type UILabel
Storage Weak
Cancel
Connect
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
The @property created by this process is weak because the MVC's View already keeps a strong pointer to the Label, so there's no need for the Controller to do so as well. And if the Label 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 Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

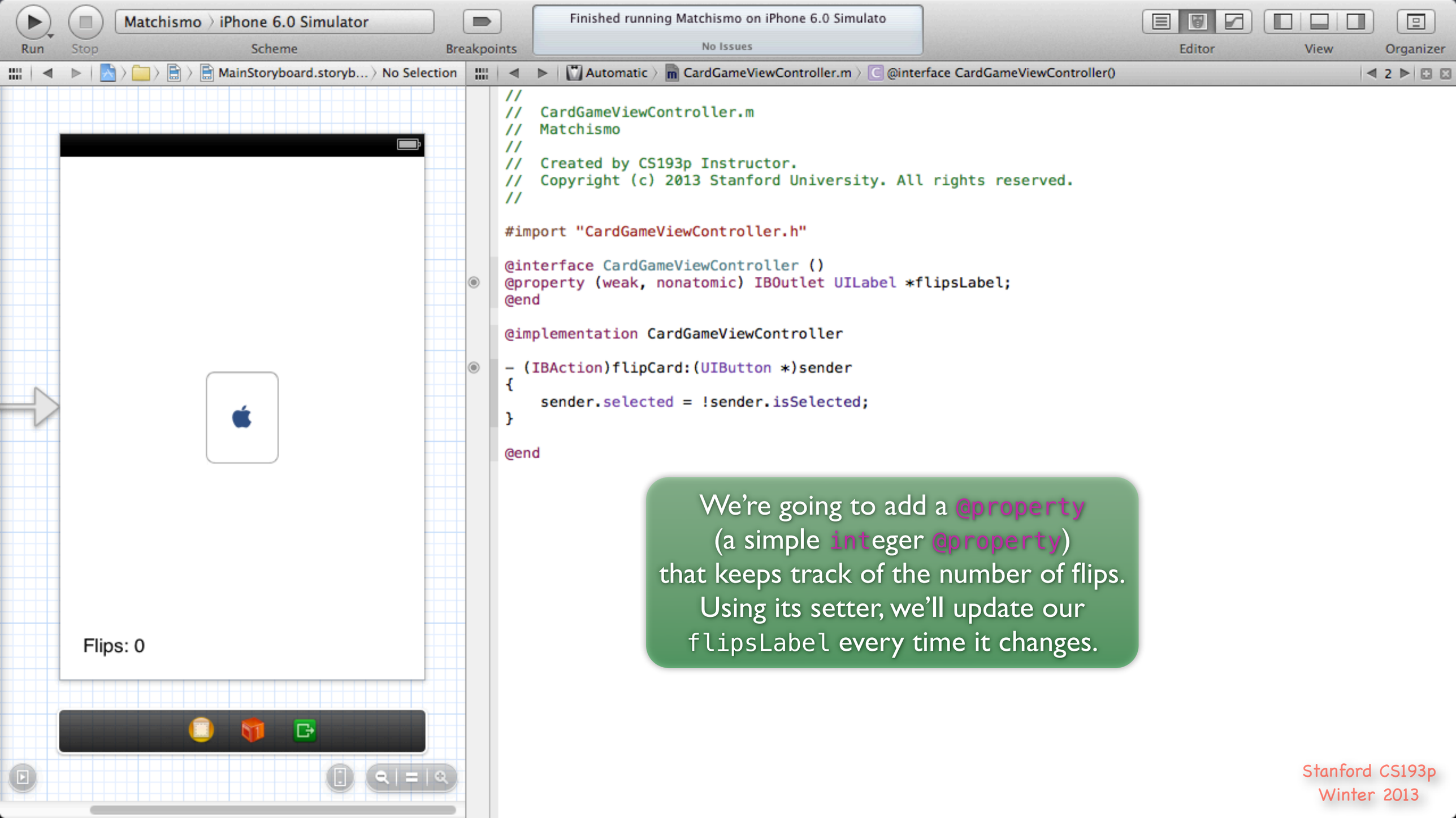

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Automatic 〉 CardGameViewController.m 〉 No Selection
Connection Outlet
Object Card Game View Co...
Name flipsLabel
Type UILabel
Storage Weak
Cancel
Connect
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
This Label shows the number of flips, so let's name it flipsLabel.
Then press Connect.
Label
Text Plain
Flips: 0
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

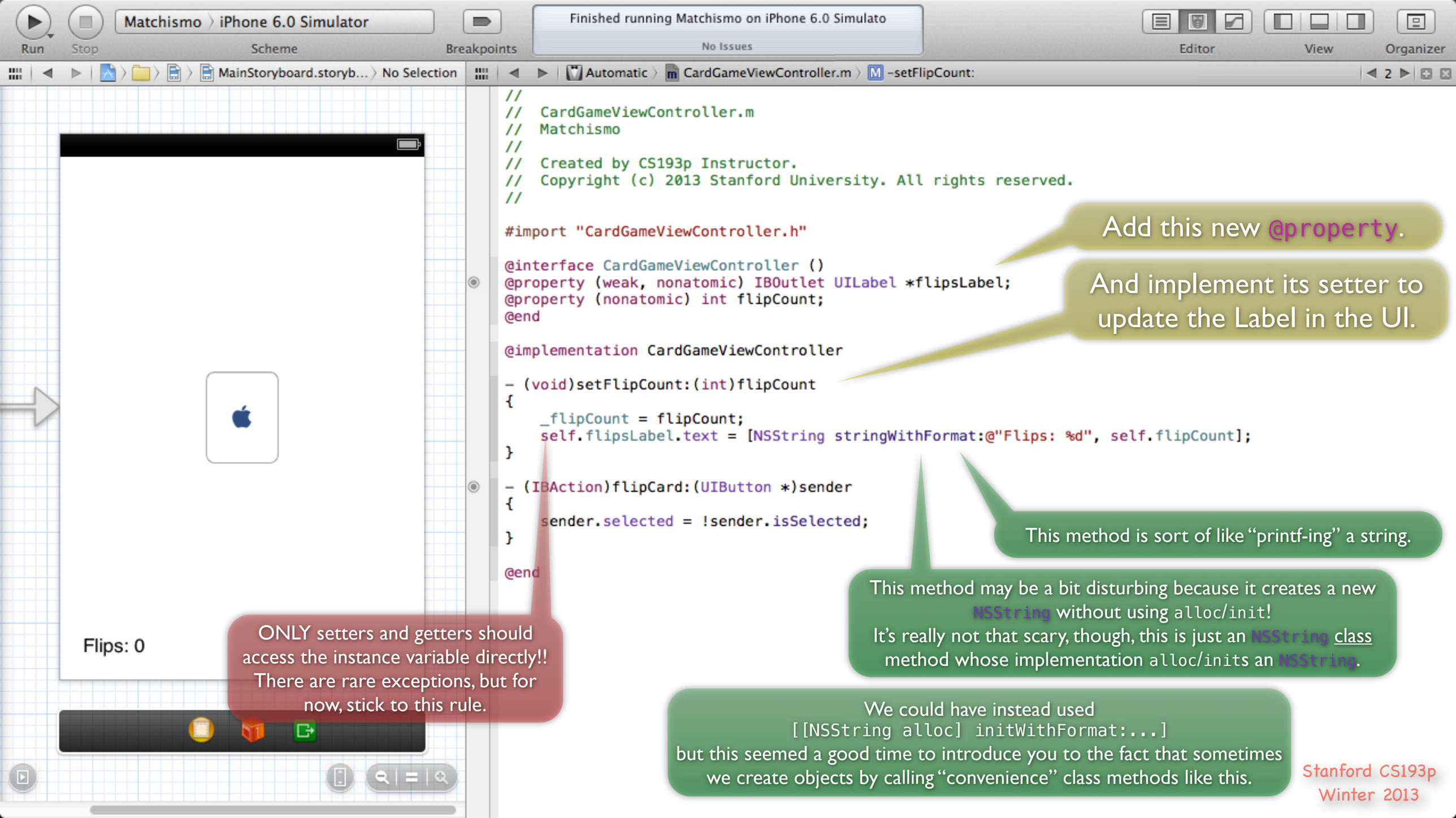

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Automatic
CardGameViewController.m
No Selection
Remove whitespace.
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Label
Text Plain
Flips: 0
Color Default
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
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

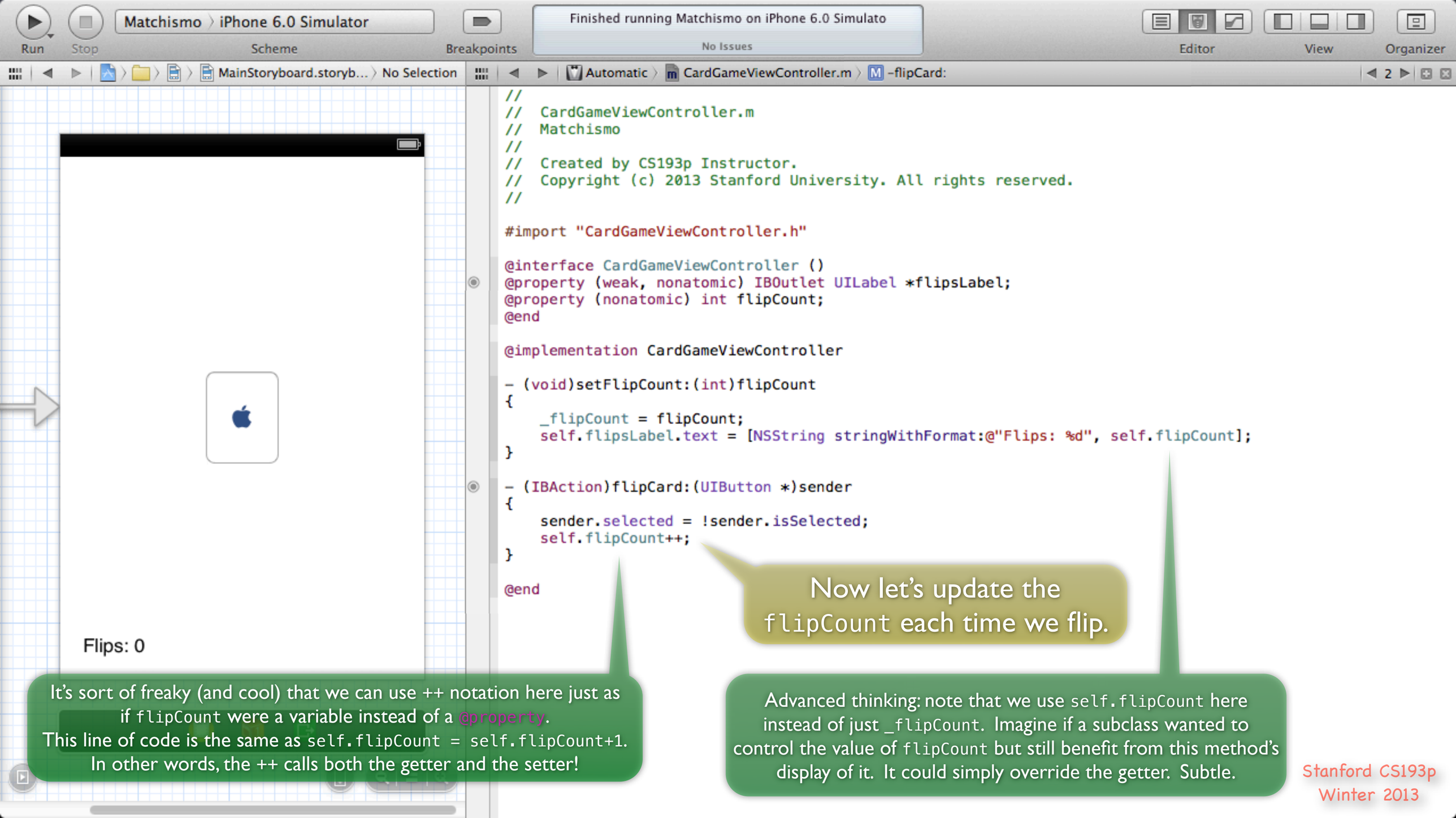

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
Flips: 0
CardGameViewController.m
@interface CardGameViewController()
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
IBOutlet is something Xcode puts in there (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 should all be familiar to you.
Identity and Type
File Name
CardGameViewController.m
File Type
Default – Objective-C So...
Location
Relative to Group
CardGameViewController.m
Full Path
/Users/CS193p/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized...
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

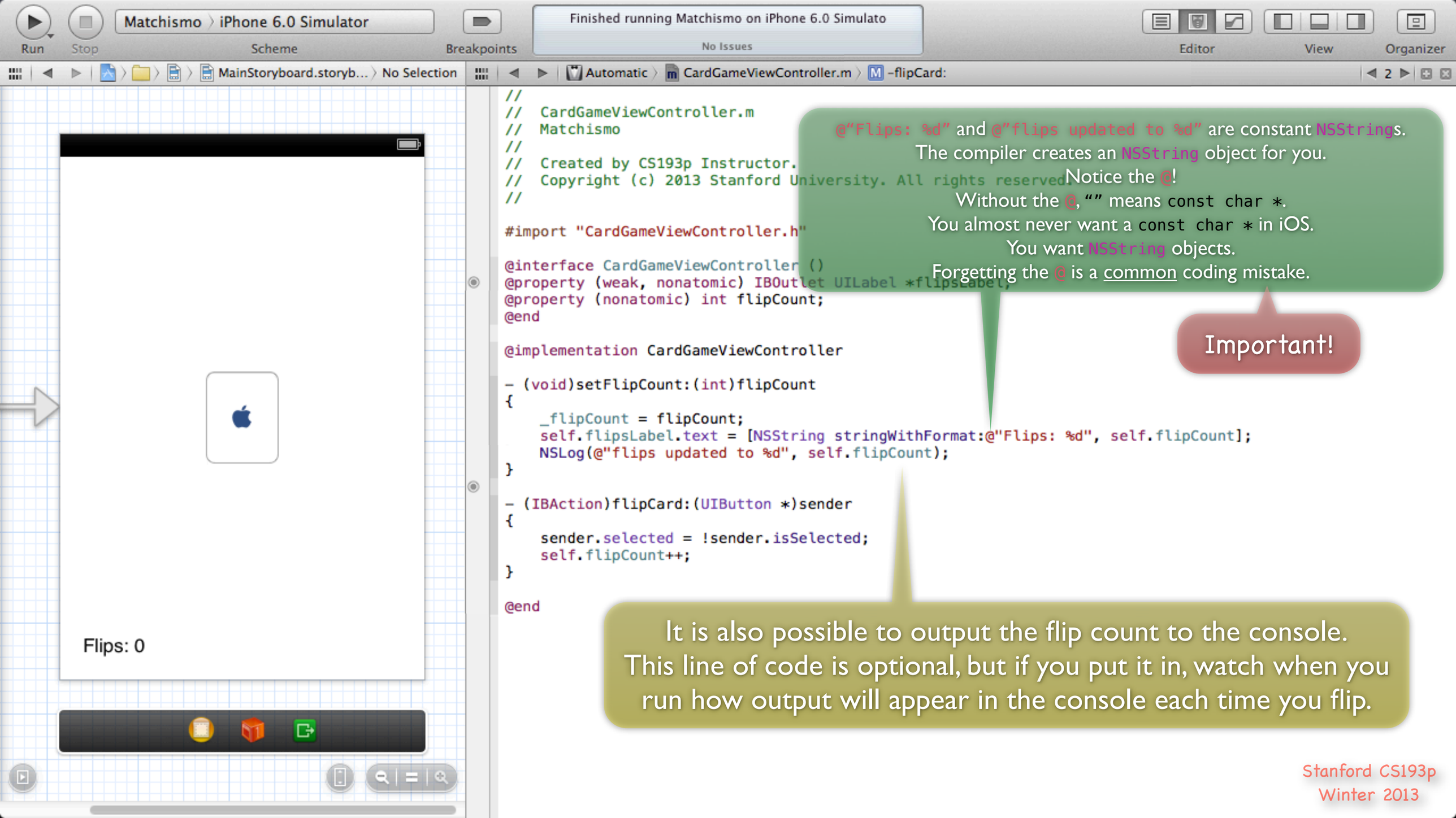

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
A 〉 CardGameViewController.m 〉 @interface CardGameViewController() 2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Just like with an action, you can mouse over this icon to see what the outlet connects to.
Flips: 0
Label – Flips: 0
Identity and Type
File Name CardGameViewController.m
File Type Default – Objective-C So...
Location Relative to Group
CardGameViewController.m
Full Path /Users/CS193p/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized...
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label – A variably sized amount of static text.
Round Rect 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
Stanford CS193p
Winter 2013

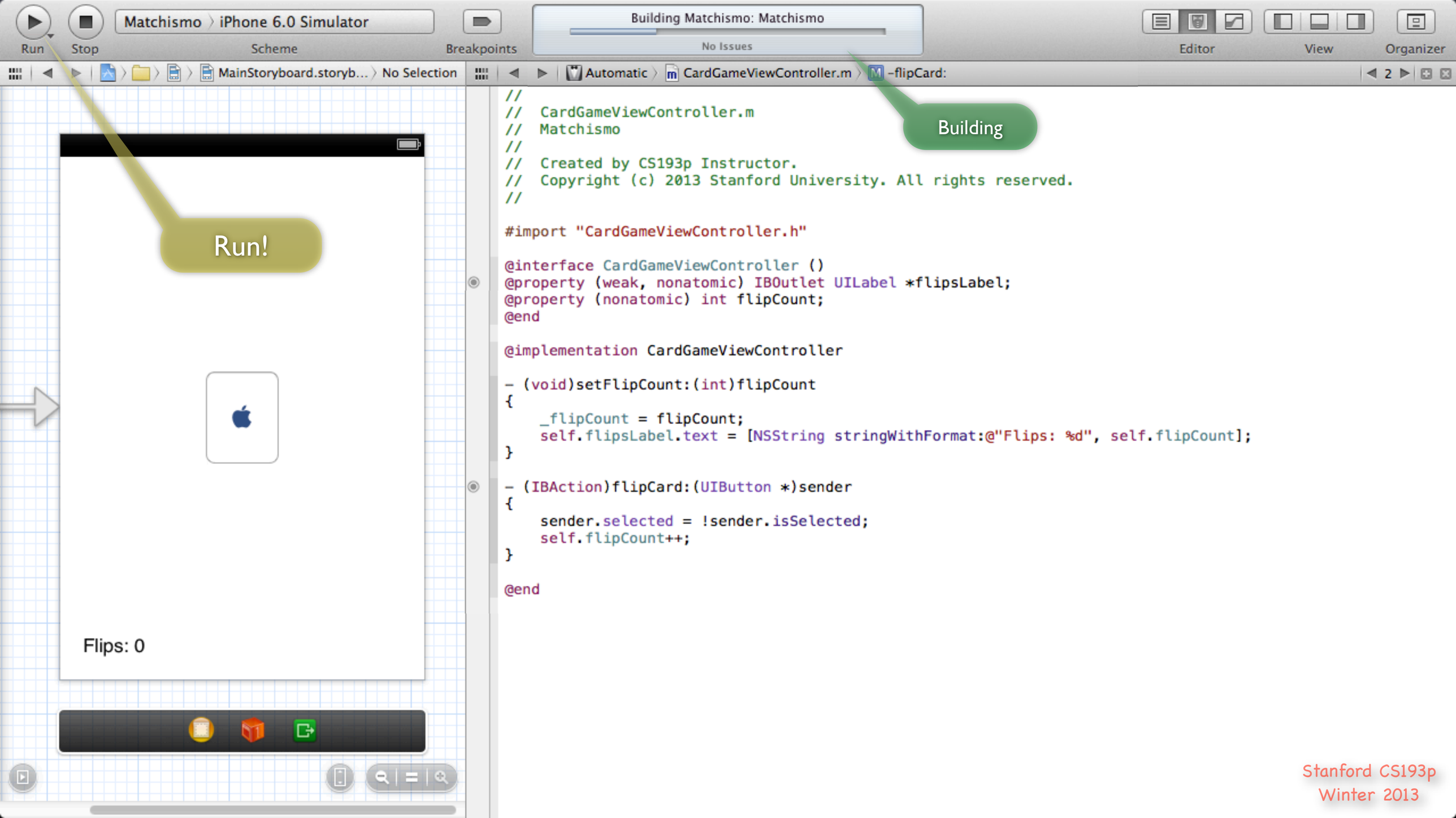

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Label – Flips: 0
A 〉 CardGameViewController.m 〉 @interface CardGameViewController() 2
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
@implementation CardGameViewController
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
We are going to do some coding now, so let's hide the Utilities Area to make more room.
Identity and Type
File Name
CardGameViewController.m
File Type
Default – Objective-C So...
Location
Relative to Group
CardGameViewController.m
Full Path
/Users/CS193p/Developer/Matchismo/Matchismo/CardGameViewController.m
Localization
Make localized...
Target Membership
Matchismo
Objects
Object – Provides a template for objects and controllers not directly available in Interface Builder.
Label
Label – A variably sized amount of static text.
Round Rect 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 text and
Stanford CS193p
Winter 2013

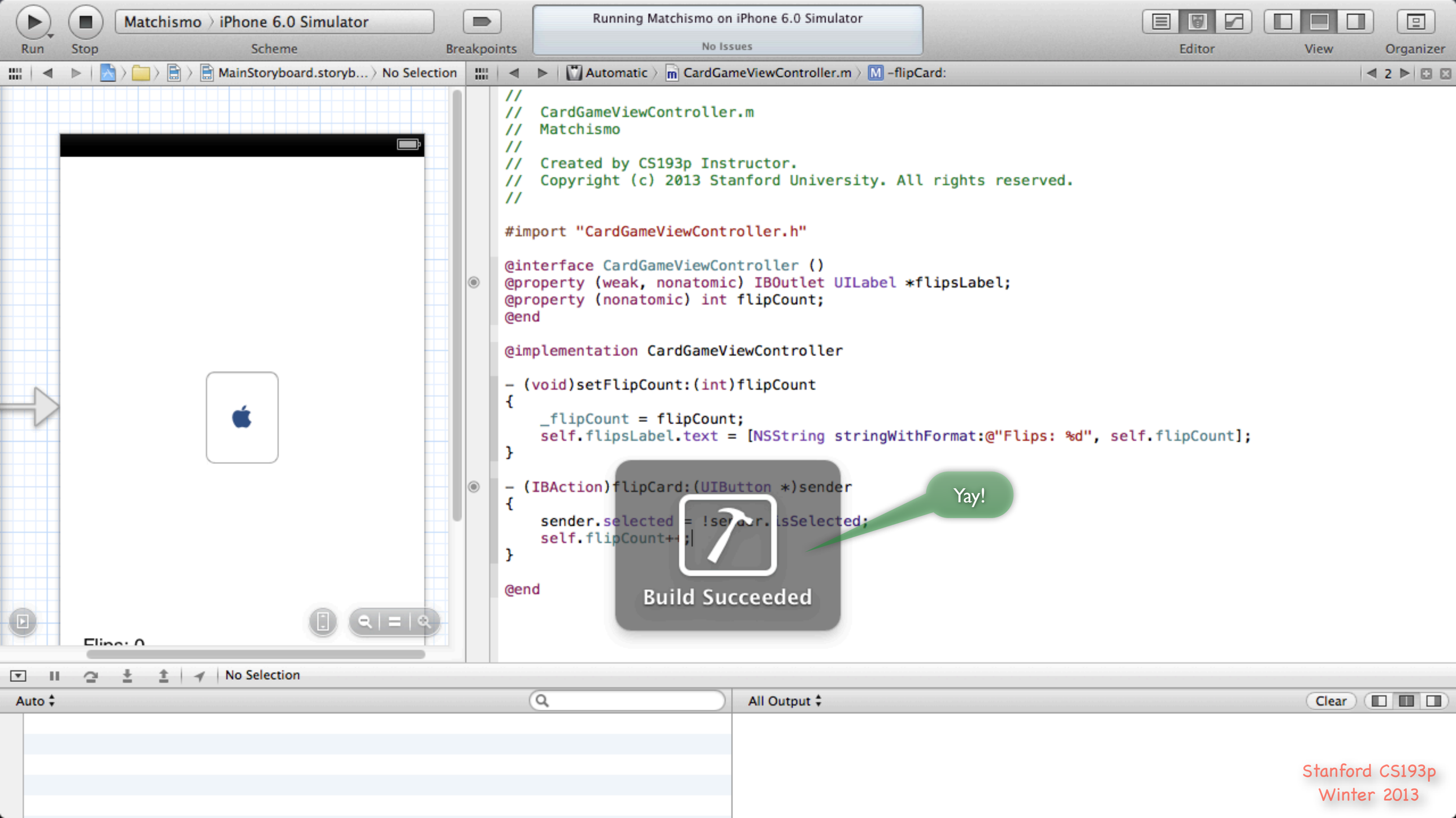

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 @interface CardGameViewController()
2
Flips: 0
//
// CardGameViewController.m
// 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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
We're going to add a @property
(a simple integer @property)
that keeps track of the number of flips.
Using its setter, we'll update our
flipsLabel every time it changes.

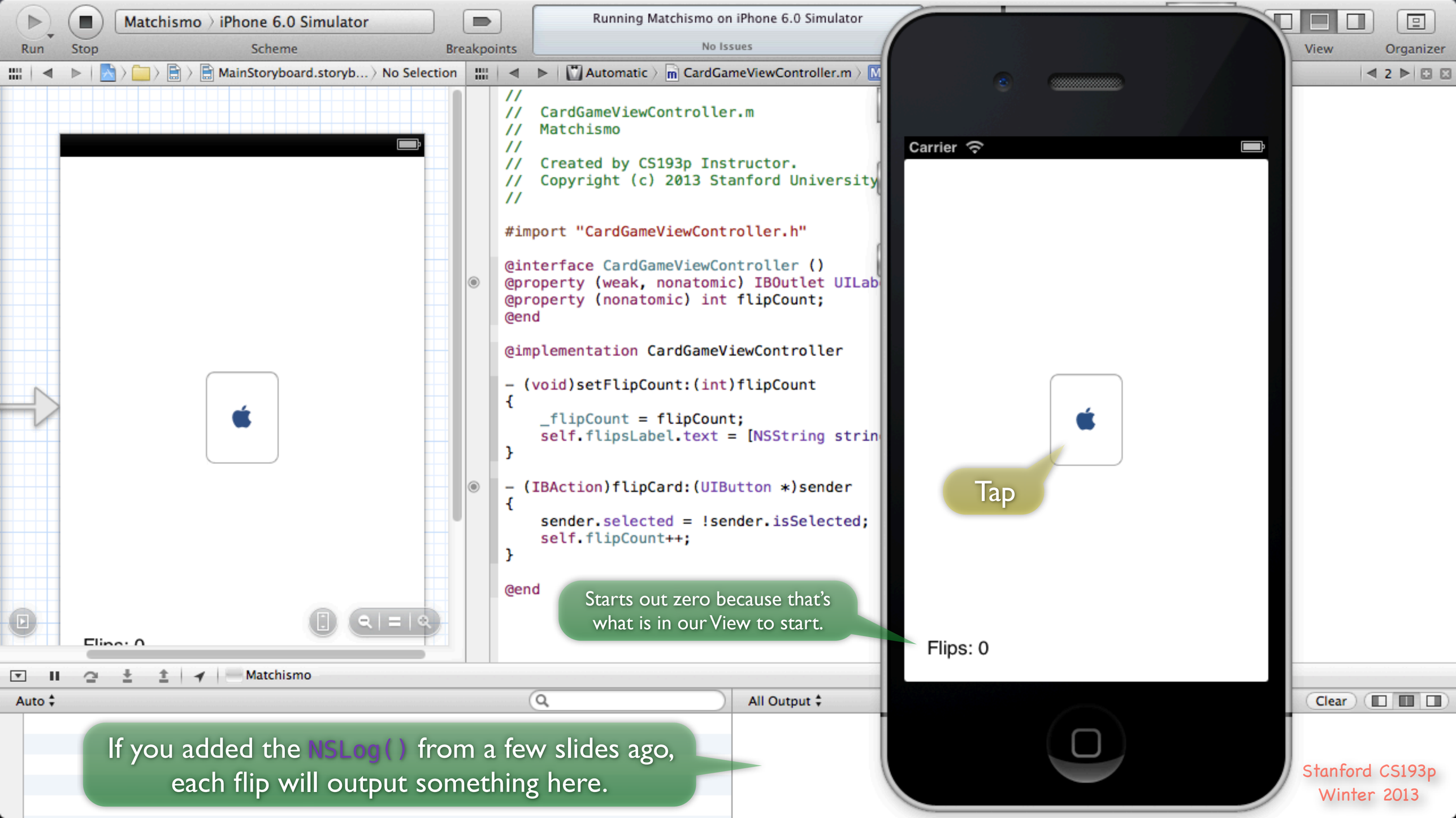

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 -setFlipCount:
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created 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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
}
@end
Add this new @property.
And implement its setter to update the Label in the UI.
This method is sort of like "printf-ing" a string.
This method may be a bit disturbing because it creates a new NSString without using alloc/init!
It's really not that scary, though, this is just an NSString class method whose implementation alloc/inits an NSString.
ONLY setters and getters should access the instance variable directly!! There are rare exceptions, but for now, stick to this rule.
We could have instead used
[[NSString alloc] initWithFormat:...]
but this seemed a good time to introduce you to the fact that sometimes we create objects by calling "convenience" class methods like this.
Stanford CS193p
Winter 2013

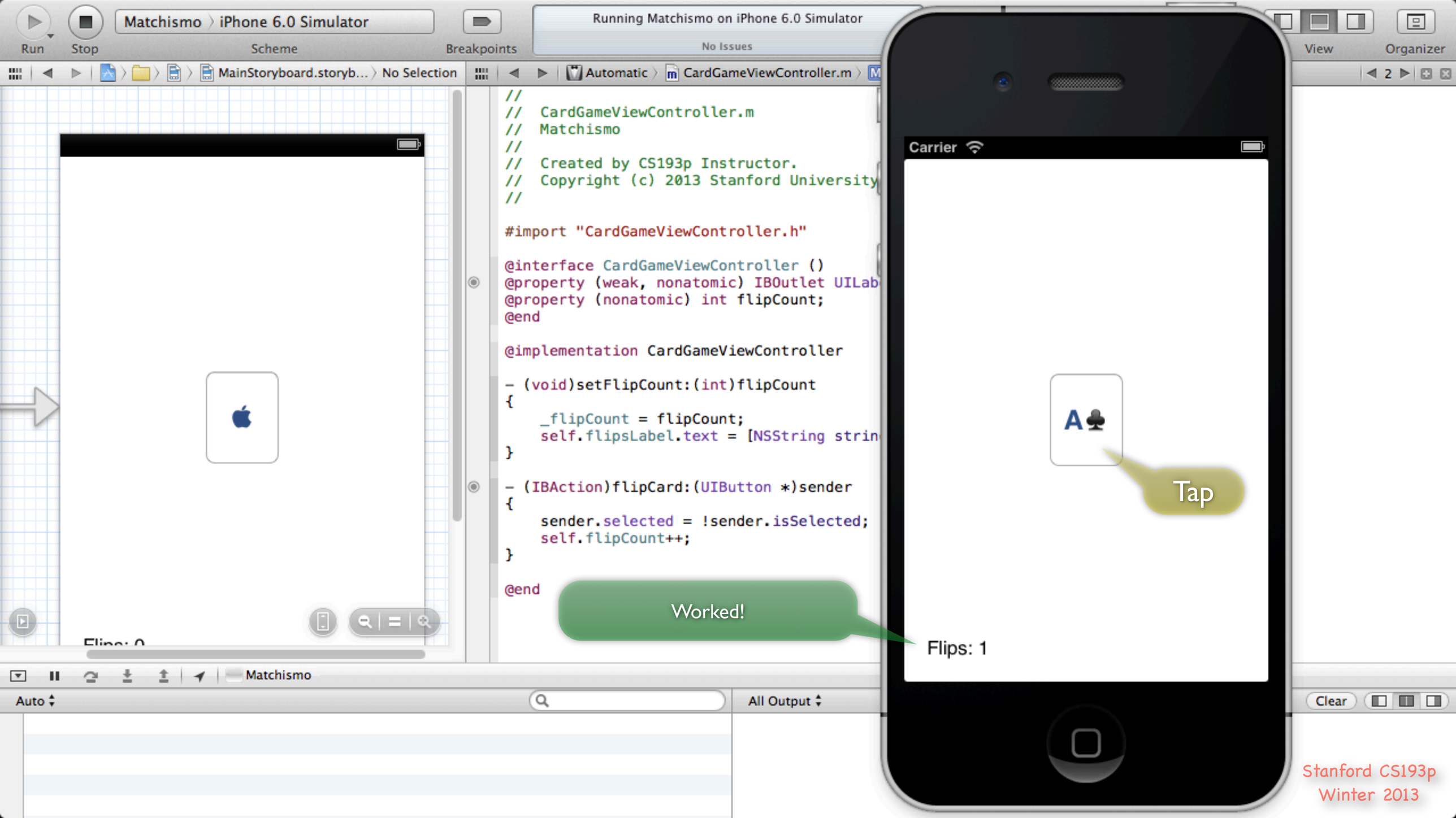

Run
Stop
Matchismo
iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
MainStoryboard.storyb...
No Selection
Automatic
CardGameViewController.m
-flipCard:
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// 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];
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Flips: 0
Now let's update the flipCount each time we flip.
It's sort of freaky (and cool) that we can use ++ notation here just as if flipCount were a variable instead of a @property.
This line of code is the same as self.flipCount = self.flipCount+1.
In other words, the ++ calls both the getter and the setter!
Advanced thinking: note that we use self.flipCount here instead of just _flipCount. Imagine if a subclass wanted to control the value of flipCount but still benefit from this method's display of it. It could simply override the getter. Subtle.
Stanford CS193p
Winter 2013

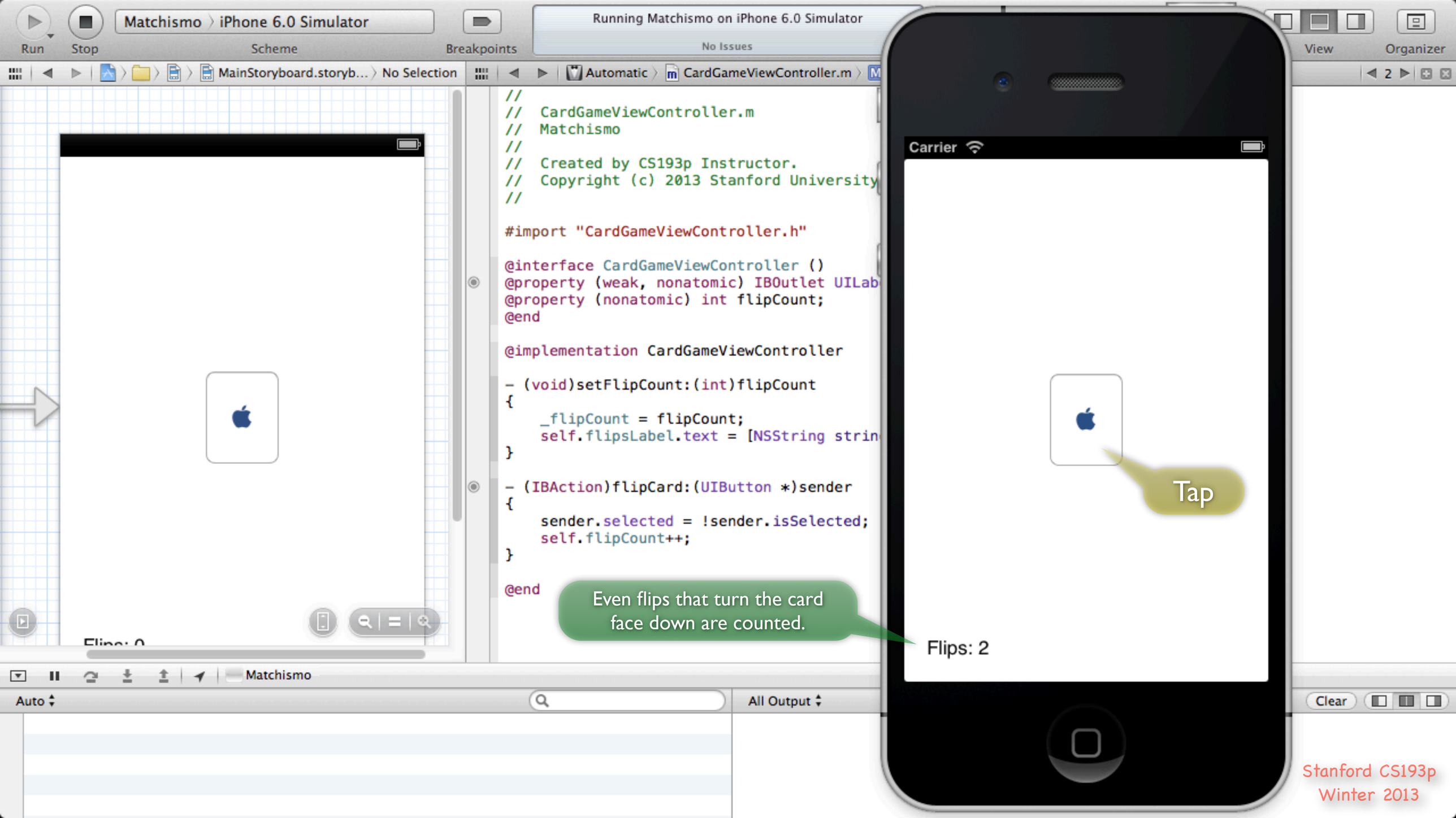

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 -flipCard:
2
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created 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];
NSLog(@"flips updated to %d", self.flipCount);
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
@"Flips: %d" and @"flips updated to %d" are constant NSStrings.
The compiler creates an NSString object for you.
Notice the @!
Without the @, "" means const char *.
You almost never want a const char * in iOS.
You want NSString objects.
Forgetting the @ is a common coding mistake.
Important!
It is also possible to output the flip count to the console.
This line of code is optional, but if you put it in, watch when you run how output will appear in the console each time you flip.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Building Matchismo: Matchismo
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 -flipCard:
Run!
Building
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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Stanford CS193p
Winter 2013

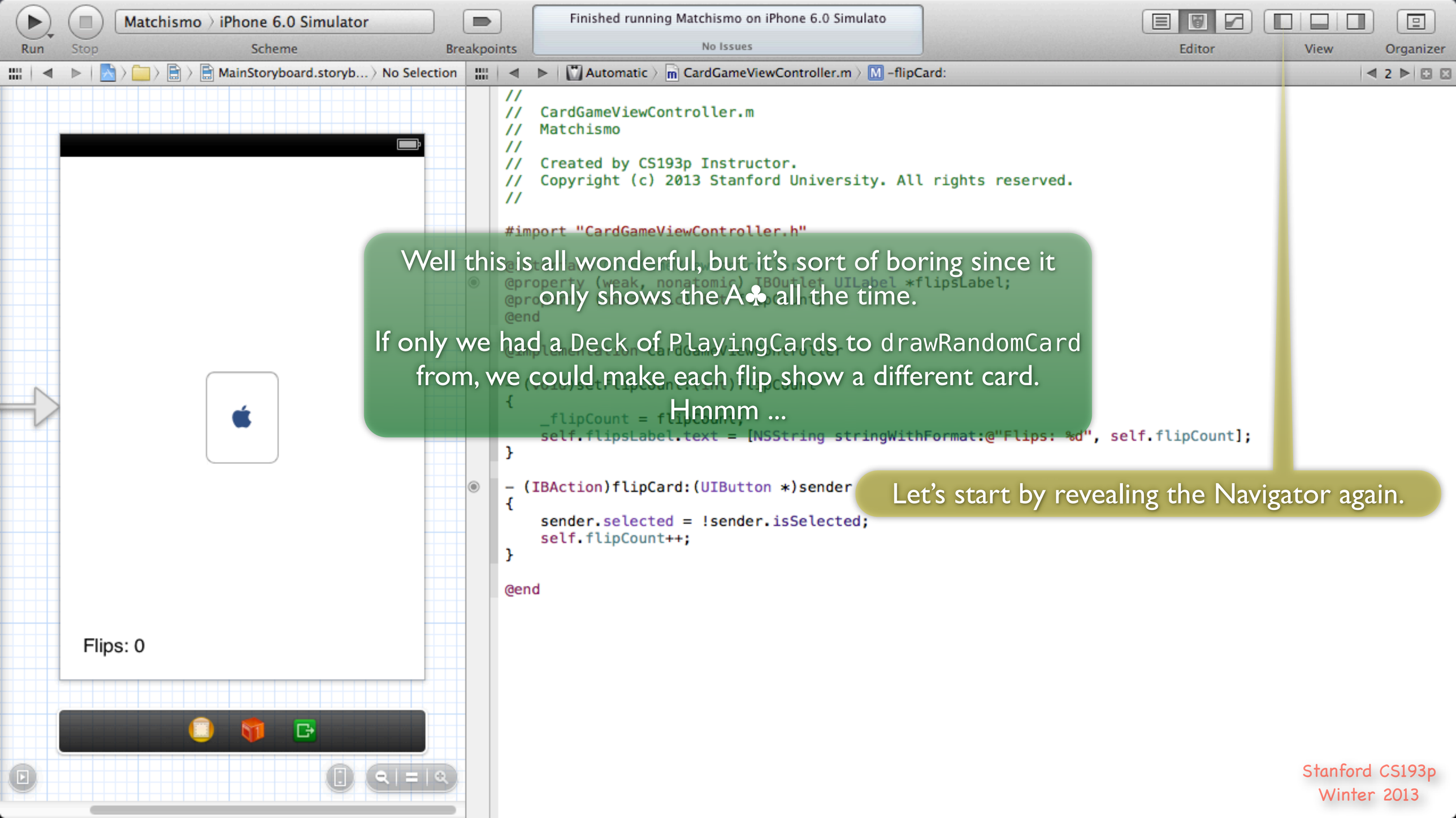

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 -flipCard:
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// 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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Yay!
Build Succeeded
Flips: 0
No Selection
Auto
All Output
Clear
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILab
@property (nonatomic) int flipCount;
@end
@implementation CardGameViewController
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString strin
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Starts out zero because that's what is in our View to start.
Carrier
Tap
Flips: 0
Matchismo
Auto
All Output
Clear
If you added the NSLog() from a few slides ago, each flip will output something here.
Stanford CS193p
Winter 2013

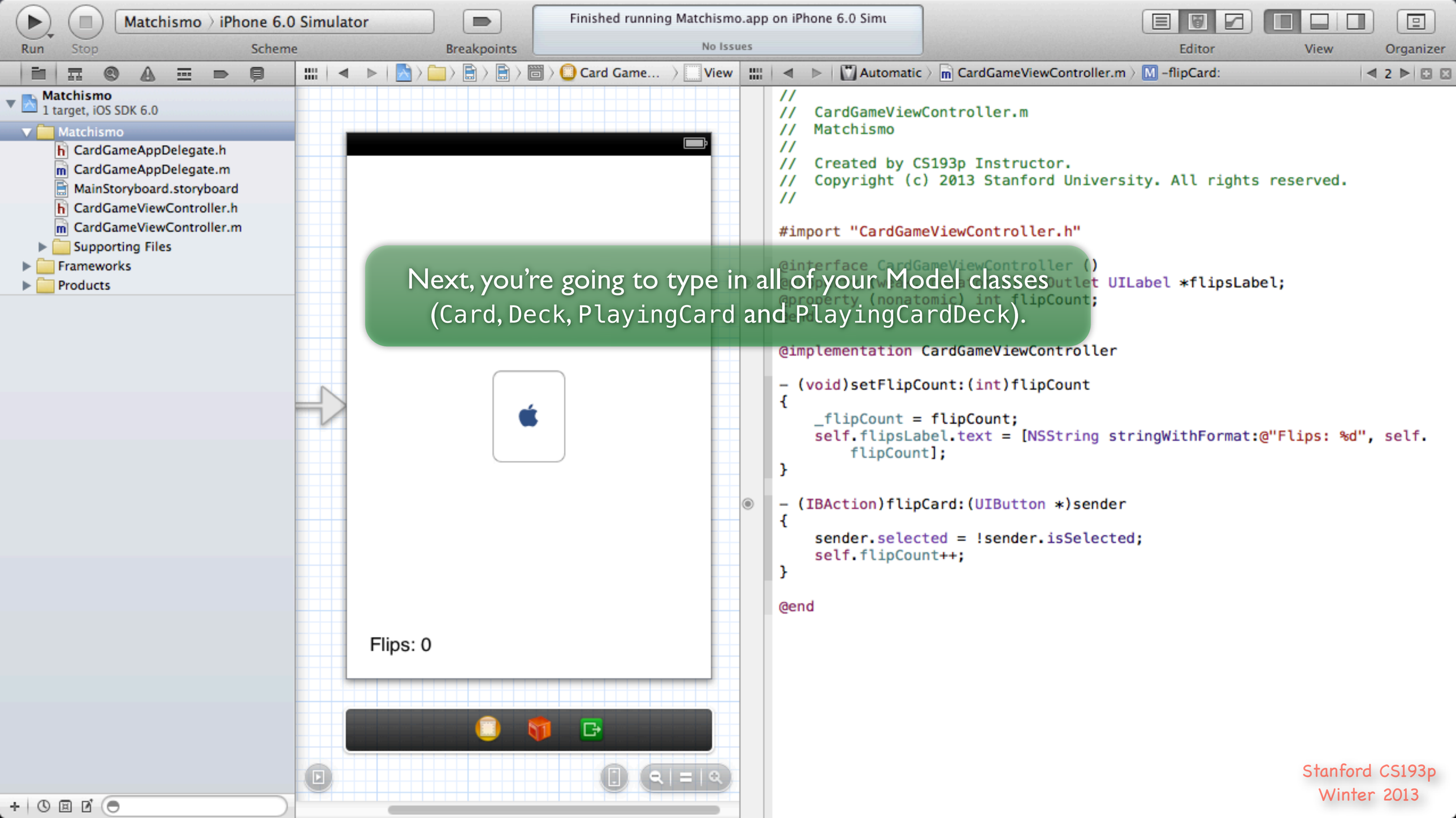

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
MainStoryboard.storyb...
No Selection
Automatic
CardGameViewController.m
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILab
@property (nonatomic) int flipCount;
@end
@implementation CardGameViewController
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString strin
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Worked!
Matchismo
Auto
All Output
Clear
Carrier
A♣
Tap
Flips: 1
Stanford CS193p
Winter 2013

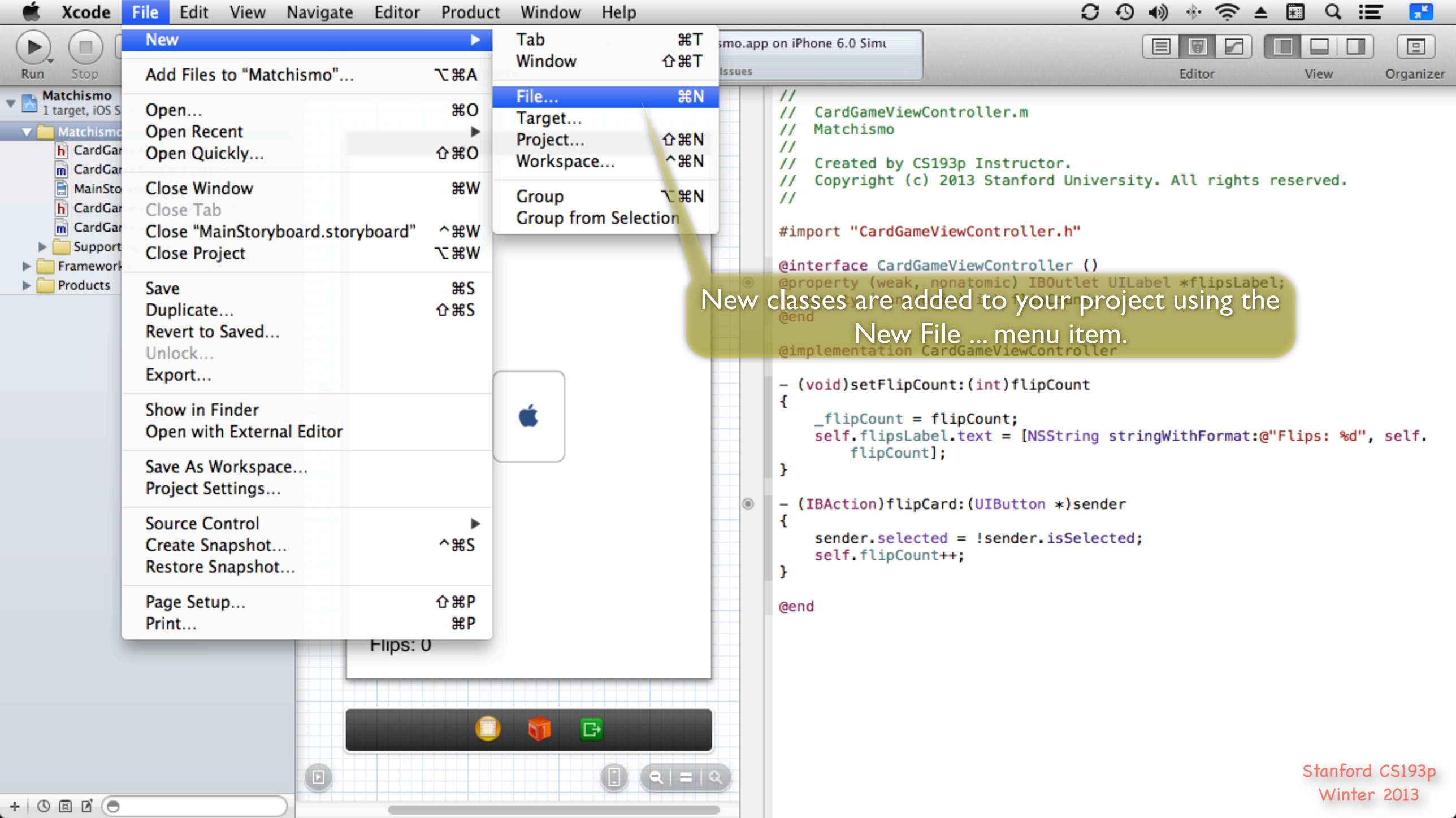

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
MainStoryboard.storyb...
No Selection
Automatic
CardGameViewController.m
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILab
@property (nonatomic) int flipCount;
@end
@implementation CardGameViewController
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString strin
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Even flips that turn the card face down are counted.
Flips: 0
Matchismo
Auto
All Output
Clear
Carrier
Tap
Flips: 2
Stanford CS193p
Winter 2013

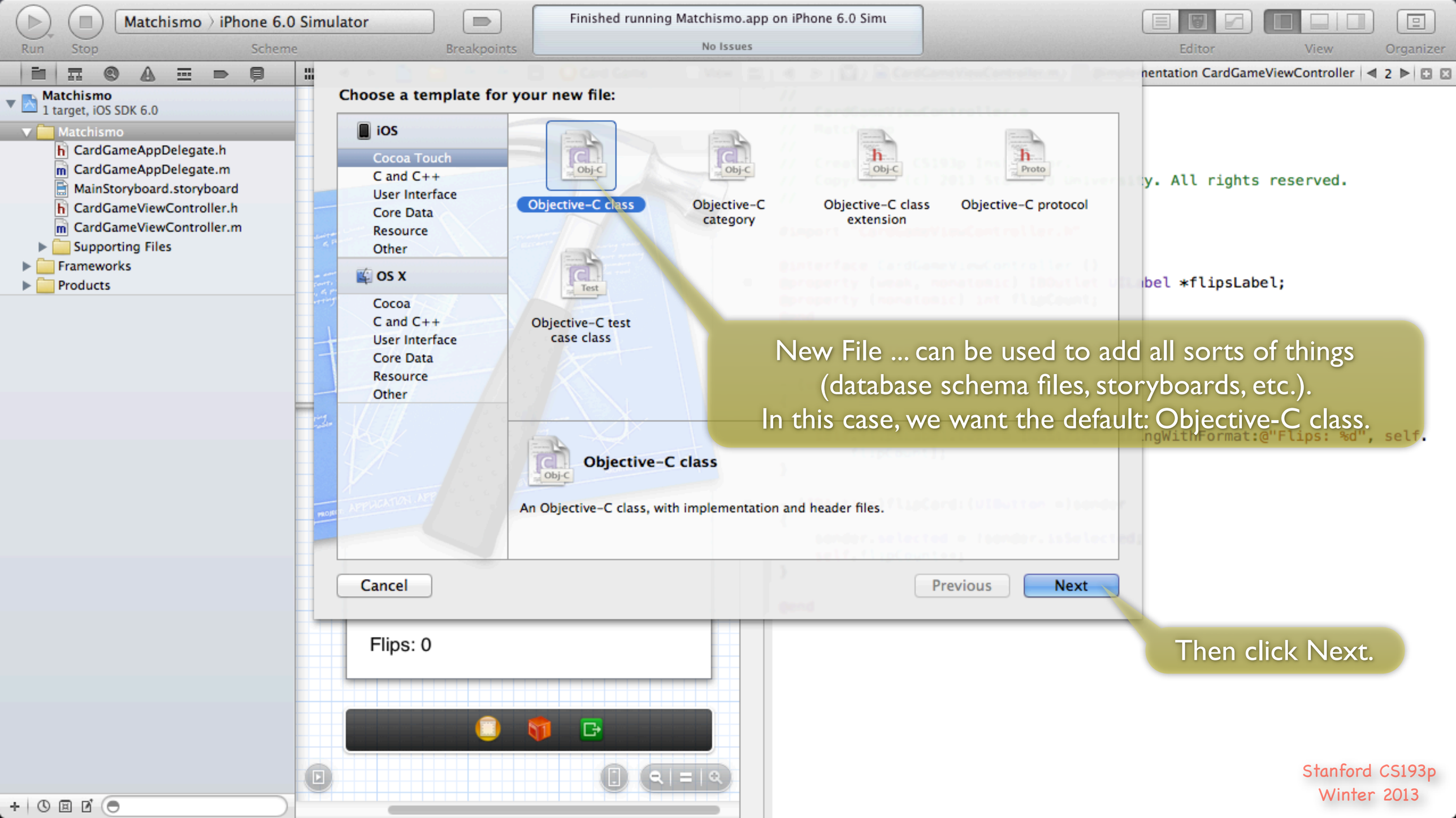

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Running Matchismo on iPhone 6.0 Simulator
No Issues
View
Organizer
MainStoryboard.storyb...
No Selection
Automatic
CardGameViewController.m
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University
//
#import "CardGameViewController.h"
@interface CardGameViewController ()
@property (weak, nonatomic) IBOutlet UILab
@property (nonatomic) int flipCount;
@end
@implementation CardGameViewController
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString strin
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Hopefully your flip count is going up and up as you tap.
Matchismo
Auto
All Output
Clear
Carrier
A♣
Keep tapping!
Flips: 29
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
MainStoryboard.storyb... 〉 No Selection
Automatic 〉 CardGameViewController.m 〉 -flipCard:
Flips: 0
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "CardGameViewController.h"
@property (weak, nonatomic) IBOutlet UILabel *flipsLabel;
@end
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
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.
Stanford CS193p
Winter 2013

Next, you're going to type in all of your Model classes (`Card`, `Deck`, `PlayingCard` and `PlayingCardDeck`).

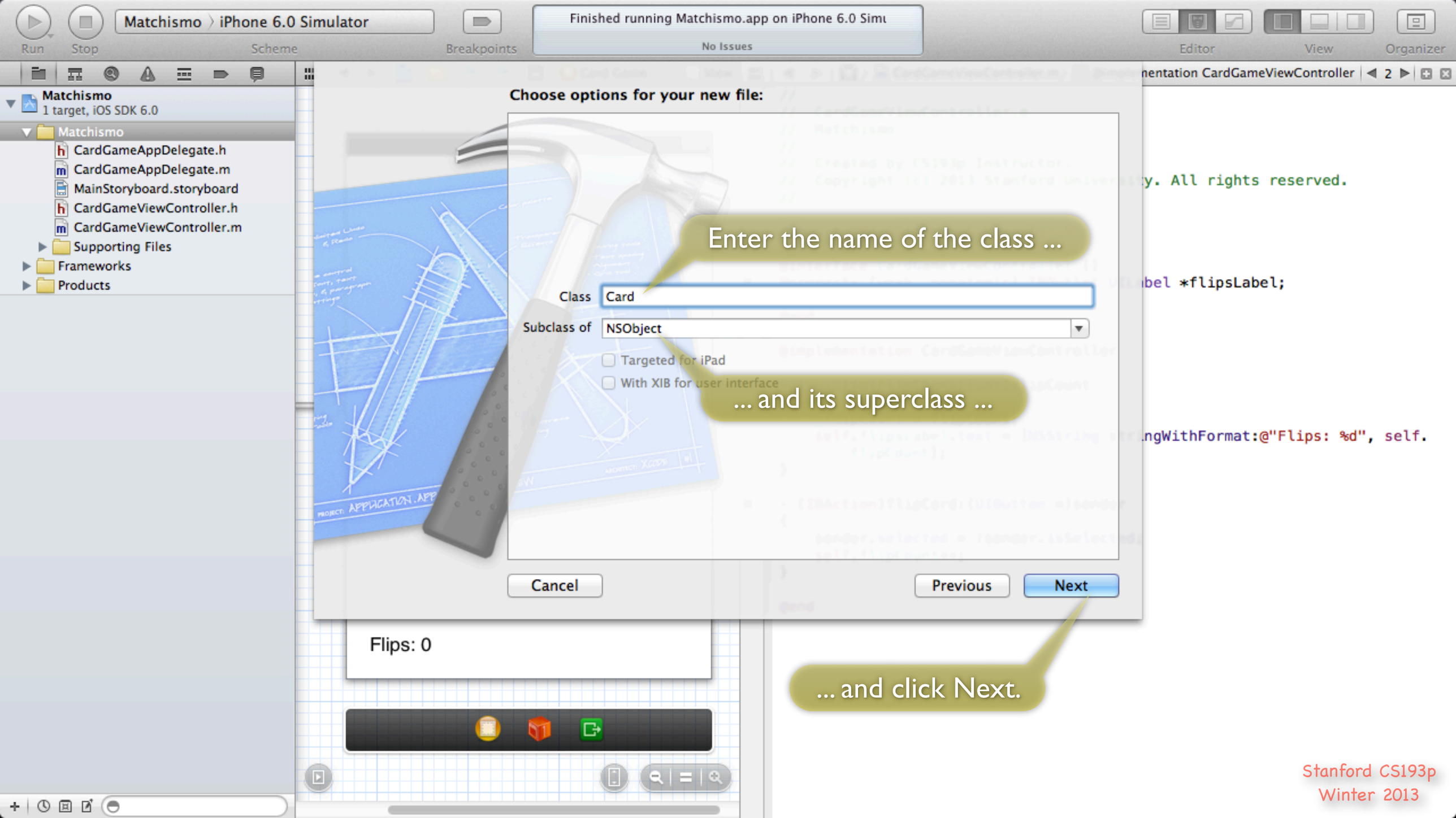



Xcode File Edit View Navigate Editor Product Window Help
New
Add Files to "Matchismo"...
Open...
Open Recent
Open Quickly...
Close Window
Close Tab
Close "MainStoryboard.storyboard"
Close Project
Save
Duplicate...
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Source Control
Create Snapshot...
Restore Snapshot...
Page Setup...
Print...
Tab
Window
File...
Target...
Project...
Workspace...
Group
Group from Selection
Run
Stop
Editor
View
Organizer
Matchismo
1 target, iOS S
Flips: 0
New classes are added to your project using the New File ... menu item.
// 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
- (void)setFlipCount:(int)flipCount
{
_flipCount = flipCount;
self.flipsLabel.text = [NSString stringWithFormat:@"Flips: %d", self.flipCount];
}
- (IBAction)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Stanford CS193p
Winter 2013

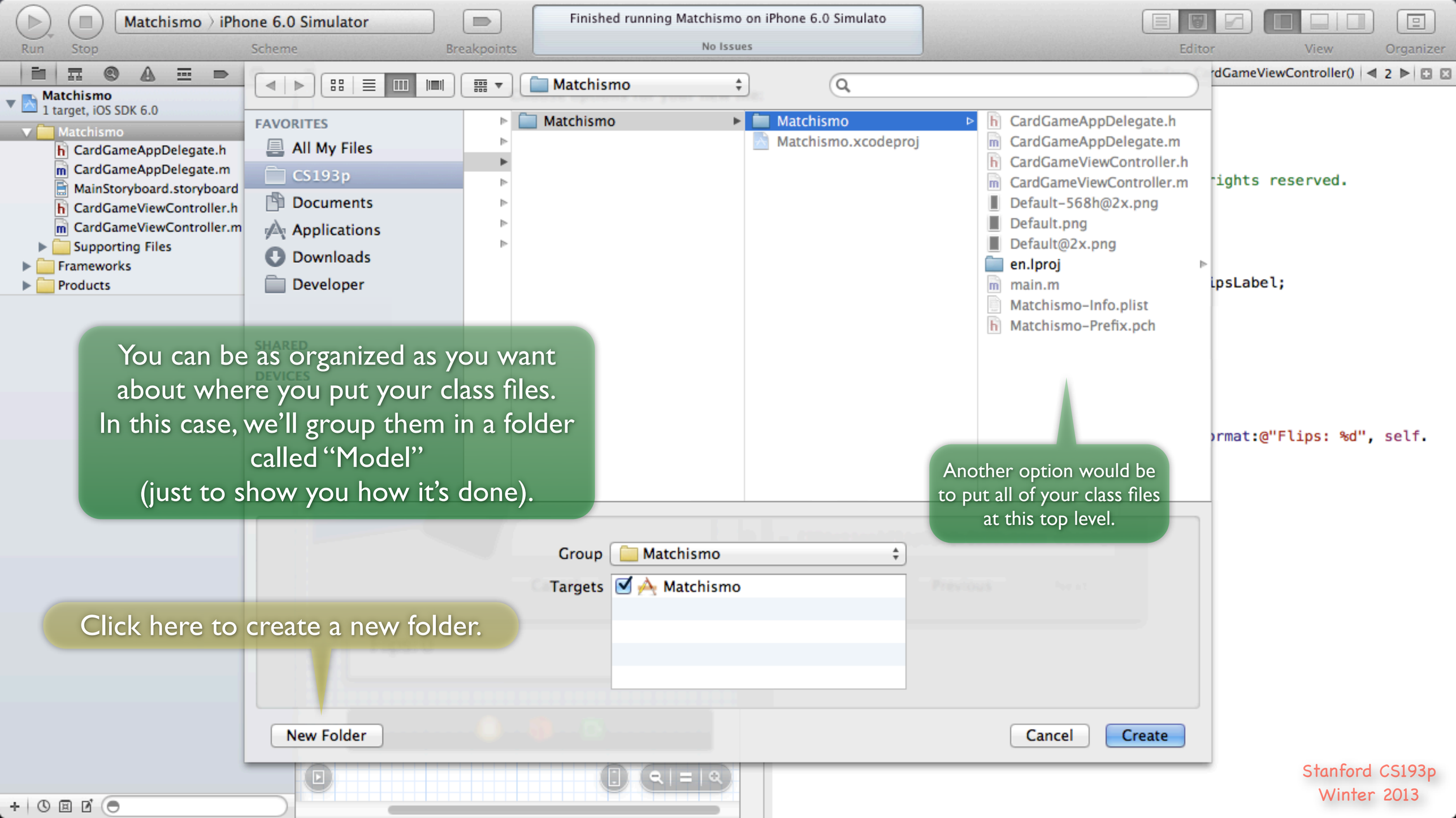


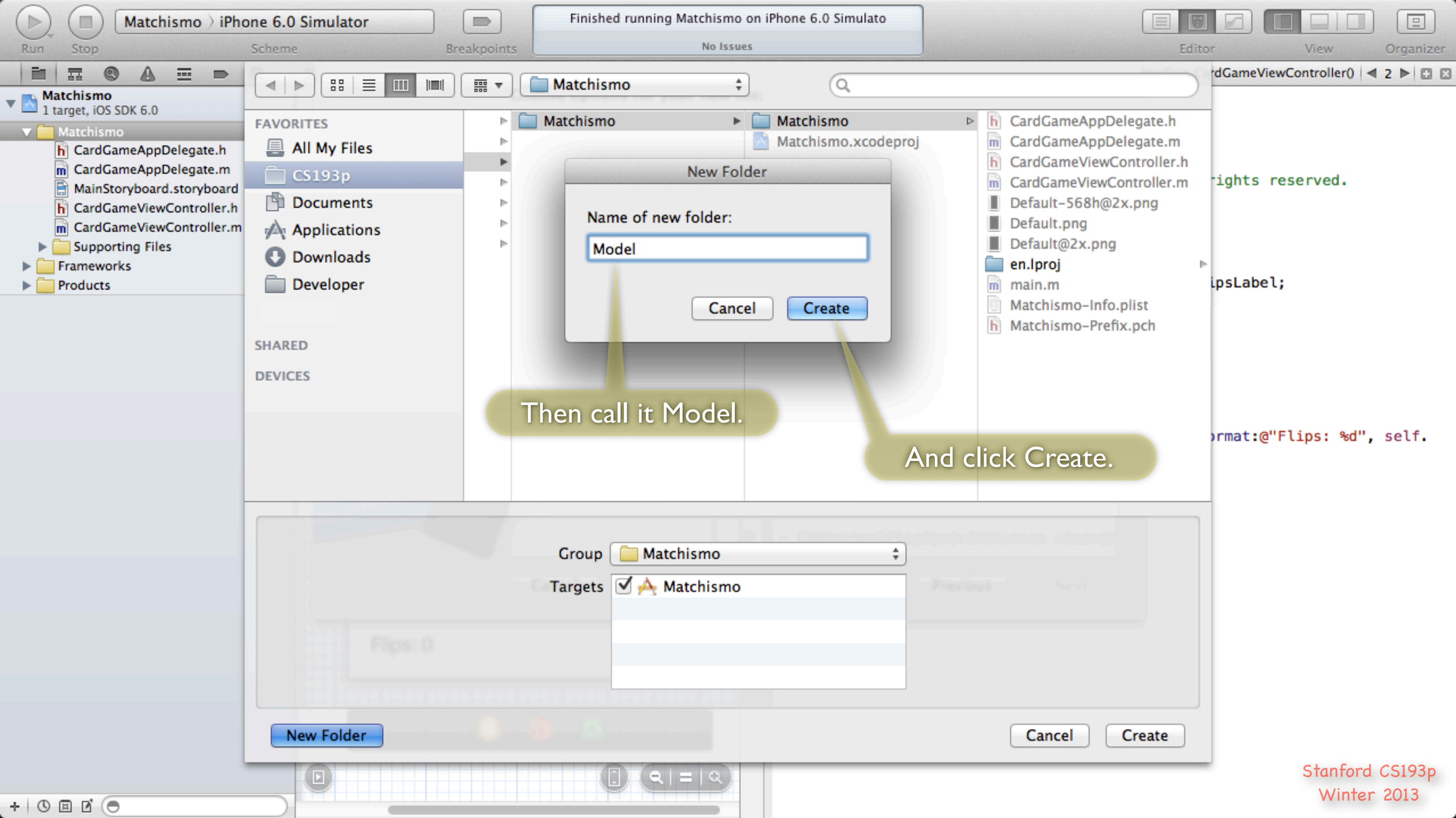
New File ... can be used to add all sorts of things (database schema files, storyboards, etc.). In this case, we want the default: Objective-C class.

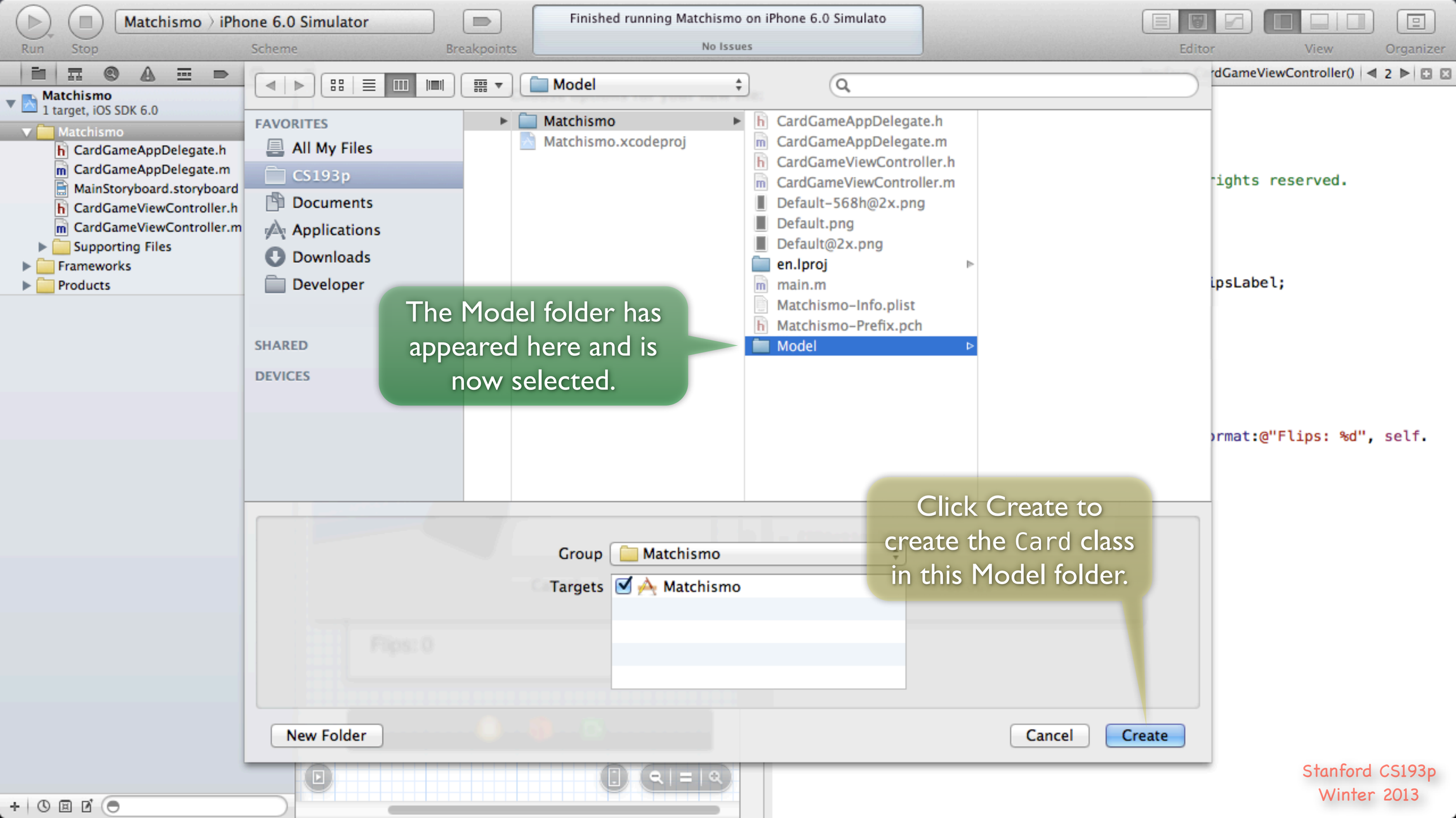
Then click Next.

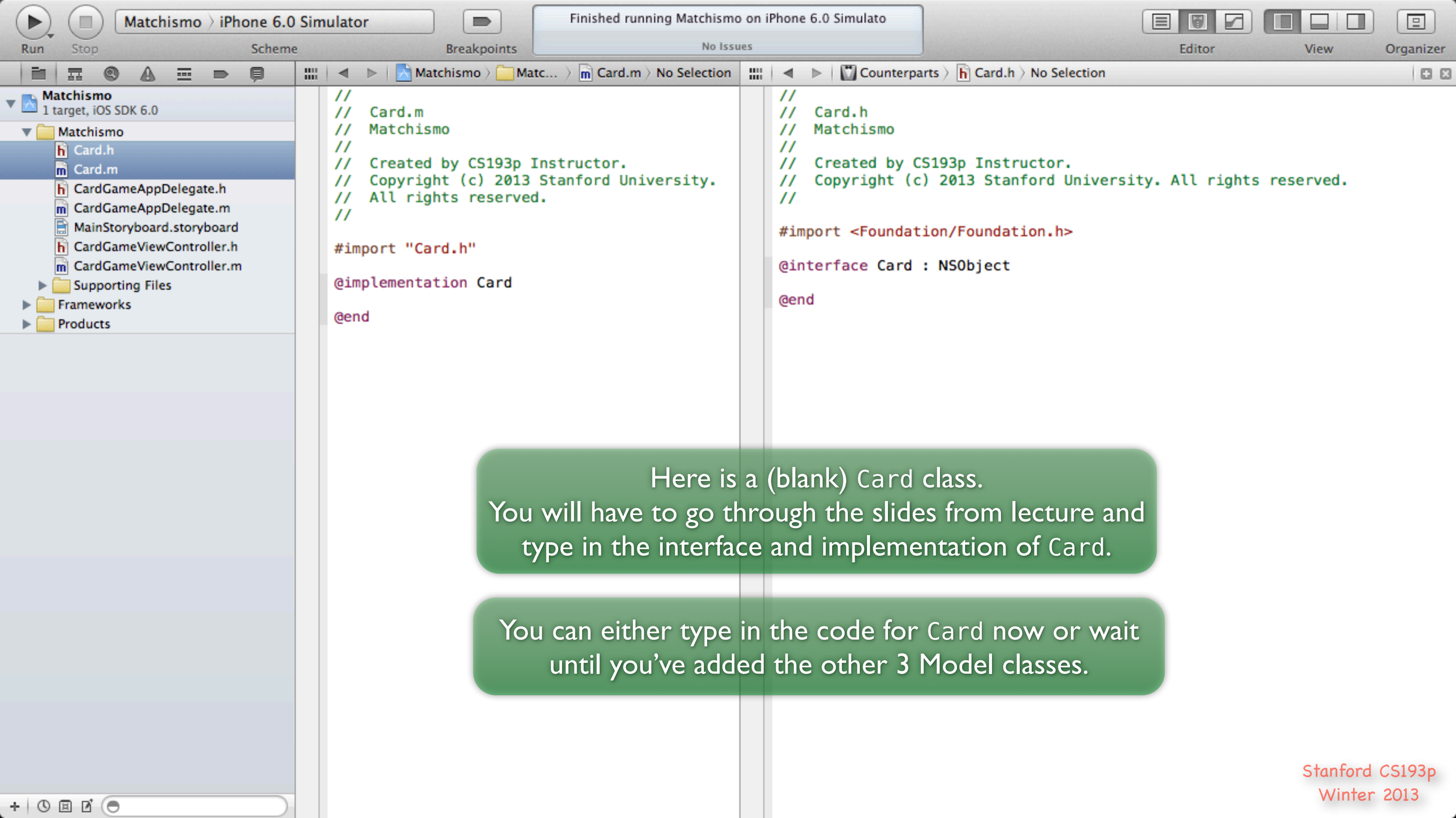


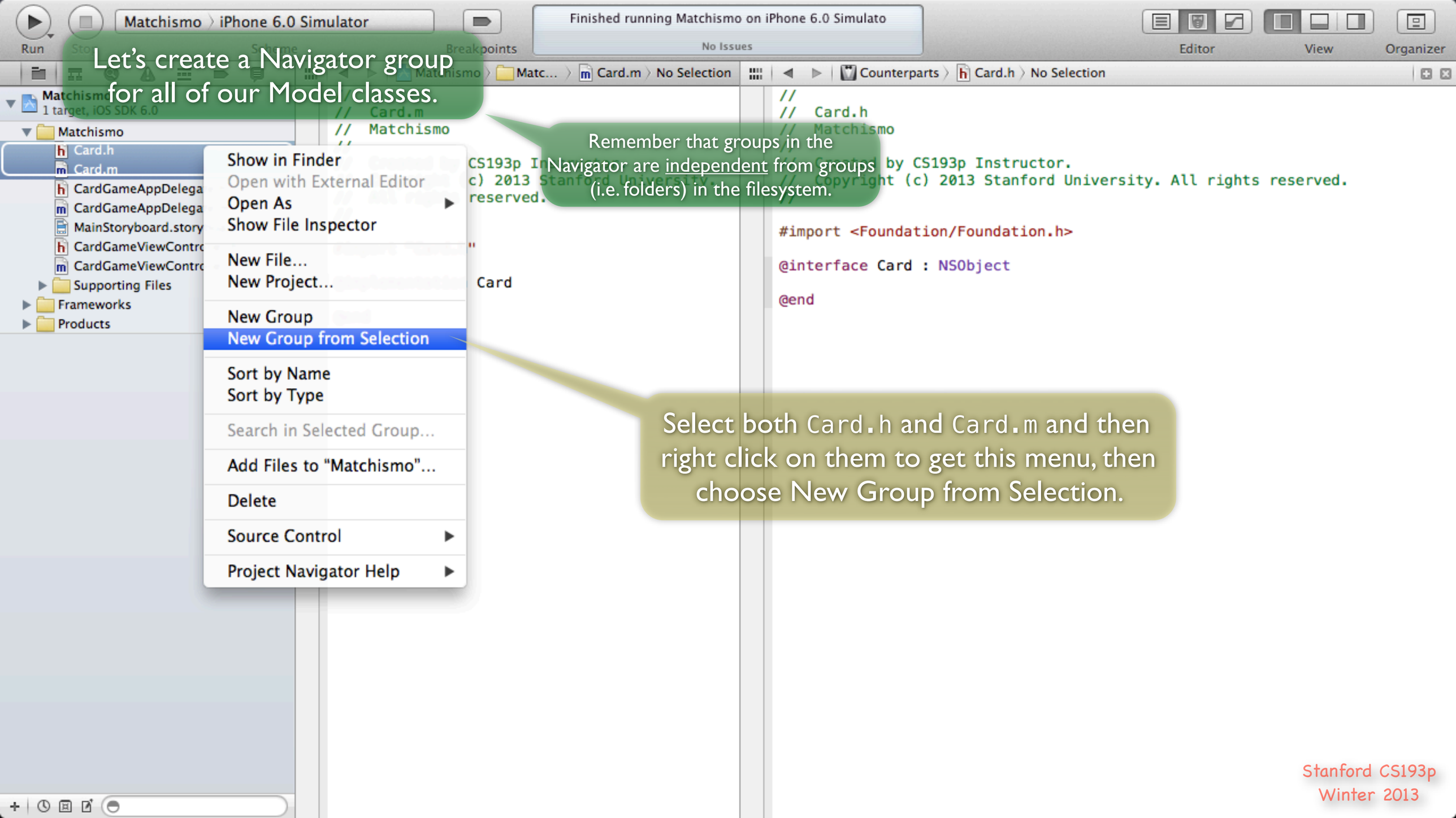
Stanford CS193p
Winter 2013


Matchismo 〉 iPhone 6.0 Simulator
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Developer
Matchismo.xcodeproj
Default-568h@2x.png
Default.png
Default@2x.png
en.lproj
main.m
Matchismo-Info.plist
Matchismo-Prefix.pch
You can be as organized as you want about where you put your class files. In this case, we'll group them in a folder called "Model" (just to show you how it's done).
Another option would be to put all of your class files at this top level.
Group
Targets
Click here to create a new folder.
New Folder
Cancel
Create
rights reserved.
ipsLabel;
rmat:@"Flips: %d", self.
Stanford CS193p
Winter 2013

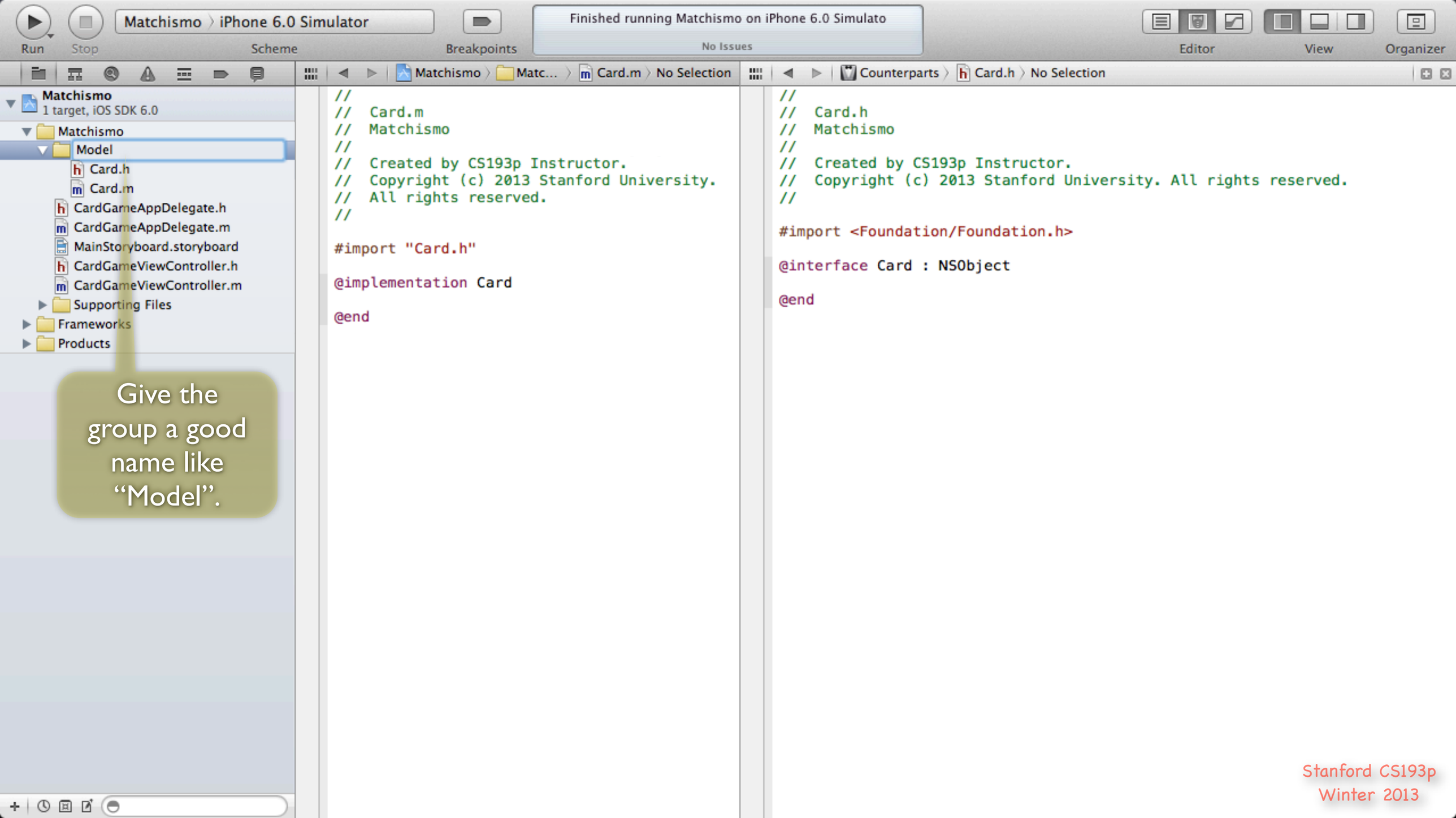

Matchismo
iPhone 6.0 Simulator
Run
Stop
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Developer
SHARED
DEVICES
Matchismo
Matchismo.xcodeproj
CardGameAppDelegate.h
CardGameAppDelegate.m
CardGameViewController.h
CardGameViewController.m
Default-568h@2x.png
Default.png
Default@2x.png
en.lproj
main.m
Matchismo-Info.plist
Matchismo-Prefix.pch
New Folder
Name of new folder:
Model
Cancel
Create
Then call it Model.
And click Create.
Group
Matchismo
Targets
Matchismo
New Folder
Cancel
Create
rights reserved.
ipsLabel;
rmat:@"Flips: %d", self.
Stanford CS193p
Winter 2013


Matchismo 〉 iPhone 6.0 Simulator
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Model
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Developer
SHARED
DEVICES
Matchismo
Matchismo.xcodeproj
CardGameAppDelegate.h
CardGameAppDelegate.m
CardGameViewController.h
CardGameViewController.m
Default-568h@2x.png
Default.png
Default@2x.png
en.lproj
main.m
Matchismo-Info.plist
Matchismo-Prefix.pch
Model
The Model folder has appeared here and is now selected.
Group
Matchismo
Targets
Matchismo
Click Create to create the Card class in this Model folder.
New Folder
Cancel
Create
rights reserved.
ipsLabel;
rmat:@"Flips: %d", self.
Stanford CS193p
Winter 2013

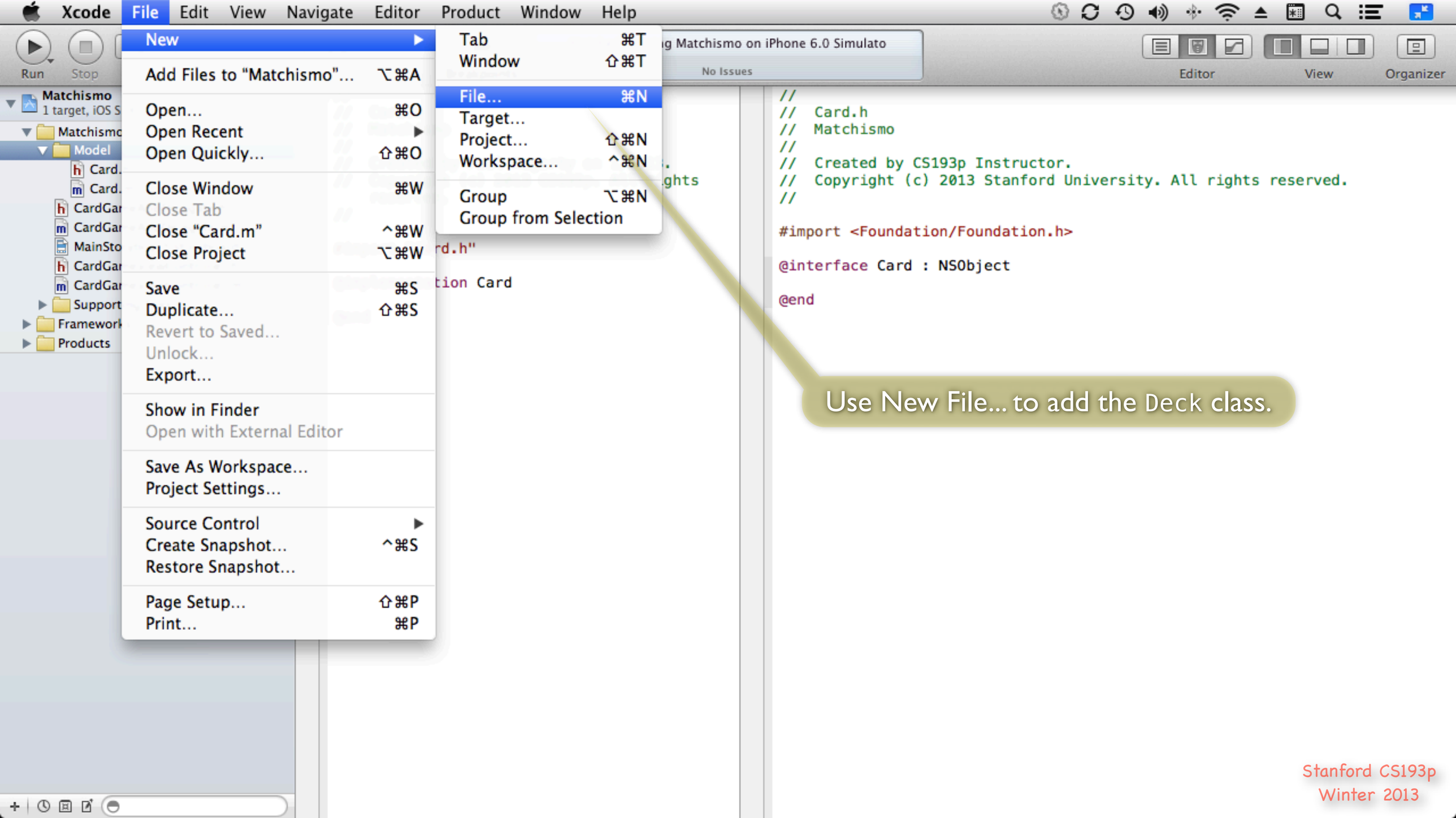

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Card.h
Card.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matc... 〉 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
Counterparts 〉 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
Here is a (blank) Card class.
You will have to go through the slides from lecture and type in the interface and implementation of Card.
You can either type in the code for Card now or wait until you've added the other 3 Model classes.

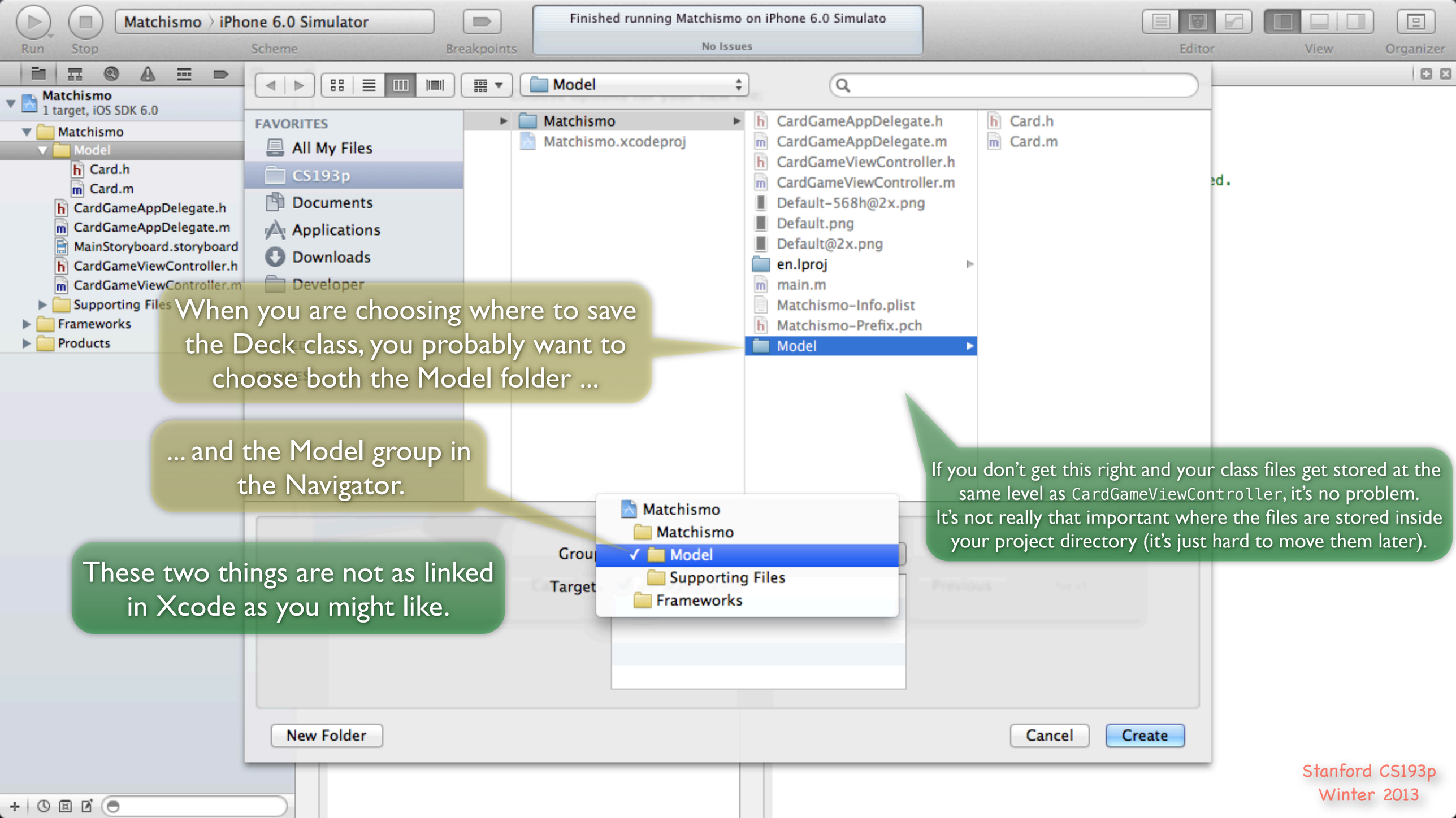


Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matc... 〉 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
Counterparts 〉 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
Give the group a good name like "Model".


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 Matc... 〉 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
Counterparts 〉 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
Okay, let's move on to creating templates for the Deck, PlayingCard and PlayingCardDeck classes.
Stanford CS193p
Winter 2013

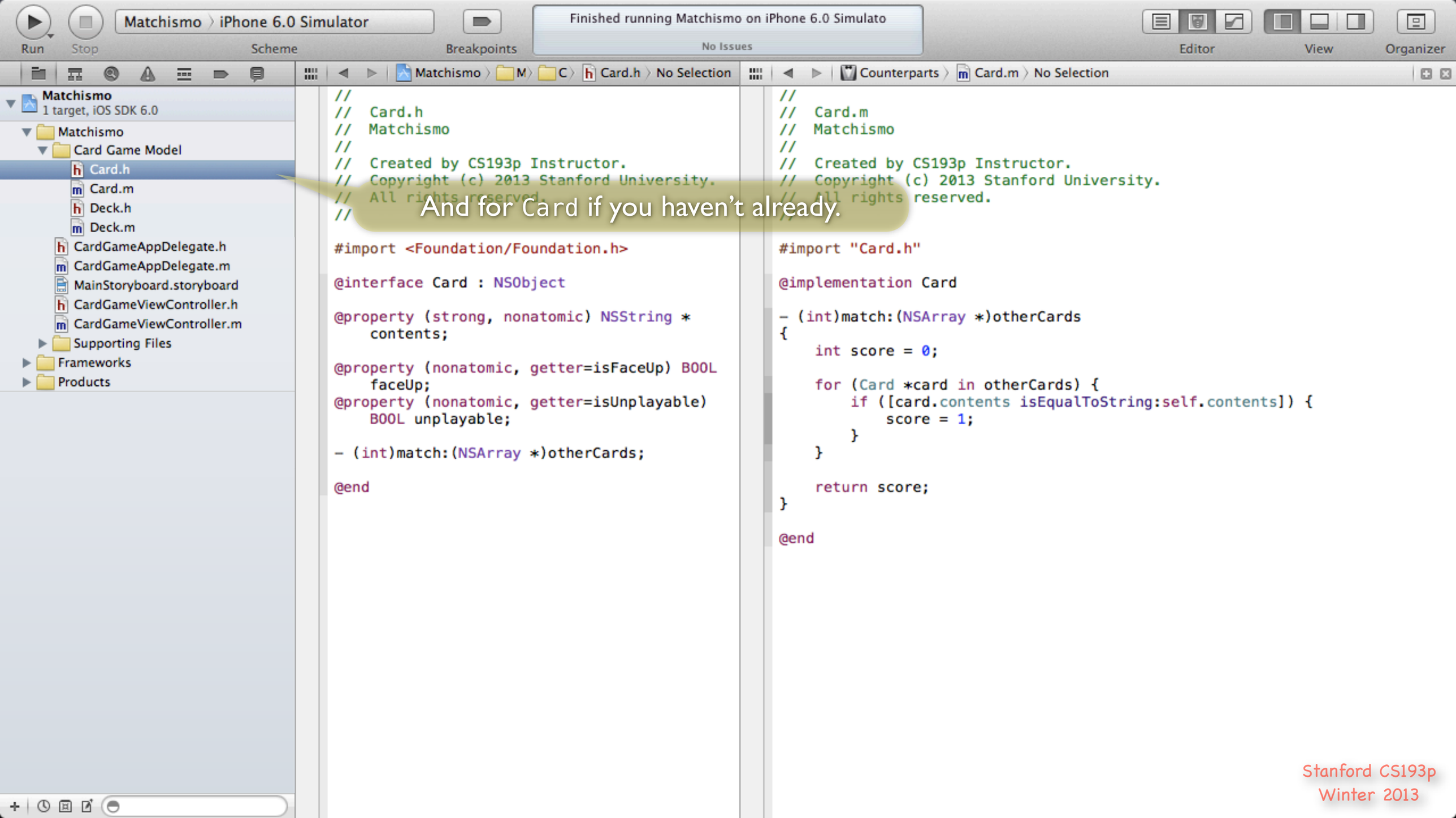

Xcode
File
Edit
View
Navigate
Editor
Product
Window
Help
New
Add Files to "Matchismo"...
Open...
Open Recent
Open Quickly...
Close Window
Close Tab
Close "Card.m"
Close Project
Save
Duplicate...
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Source Control
Create Snapshot...
Restore Snapshot...
Page Setup...
Print...
Tab
Window
File...
Target...
Project...
Workspace...
Group
Group from Selection
Run
Stop
Matchismo
Model
Editor
View
Organizer
No Issues
//
// Card.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <Foundation/Foundation.h>
@interface Card : NSObject
@end
Use New File... to add the Deck class.
Stanford CS193p
Winter 2013

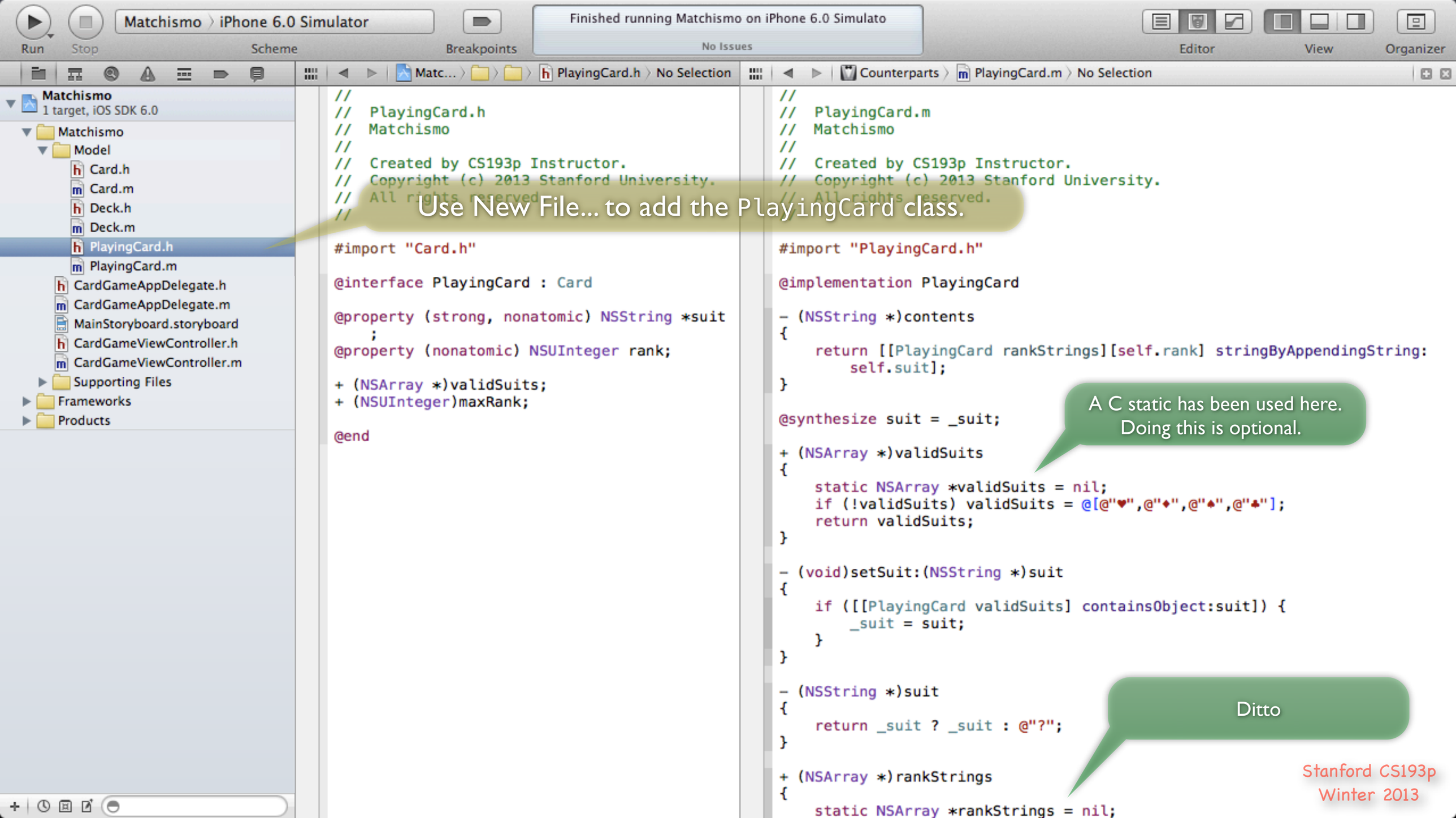

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Model
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Developer
Matchismo
Matchismo.xcodeproj
CardGameAppDelegate.h
CardGameAppDelegate.m
CardGameViewController.h
CardGameViewController.m
Default-568h@2x.png
Default.png
Default@2x.png
en.lproj
main.m
Matchismo-Info.plist
Matchismo-Prefix.pch
Model
Card.h
Card.m
When you are choosing where to save the Deck class, you probably want to choose both the Model folder ...
... and the Model group in the Navigator.
These two things are not as linked in Xcode as you might like.
If you don't get this right and your class files get stored at the same level as CardGameViewController, it's no problem. It's not really that important where the files are stored inside your project directory (it's just hard to move them later).
Matchismo
Matchismo
Model
Supporting Files
Frameworks
Group
Target
New Folder
Cancel
Create
Stanford CS193p
Winter 2013

Now you can type in the code for Deck.

All the code doesn't fit here, so use the other lecture slides to enter this code.

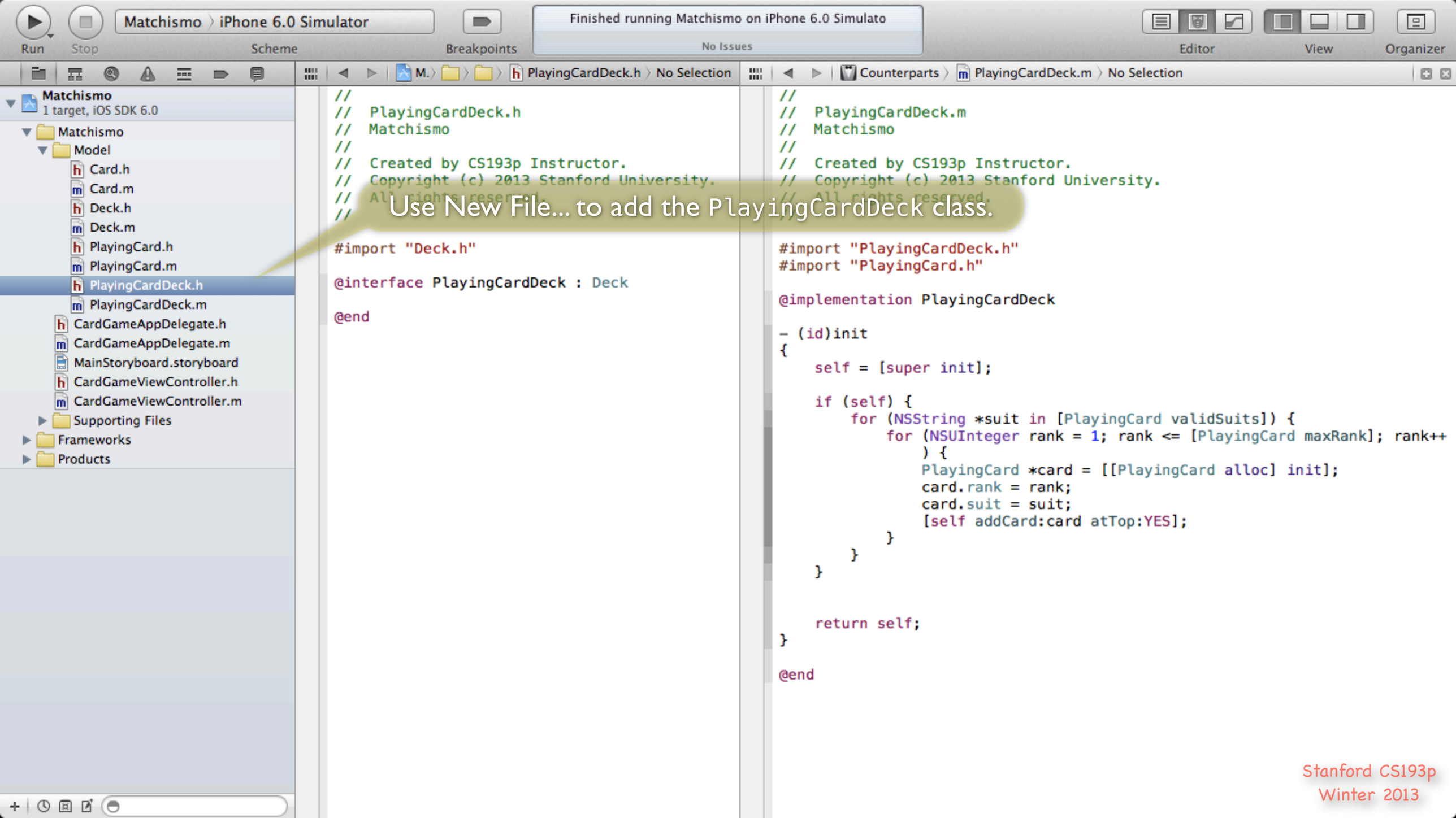
Stanford CS193p
Winter 2013

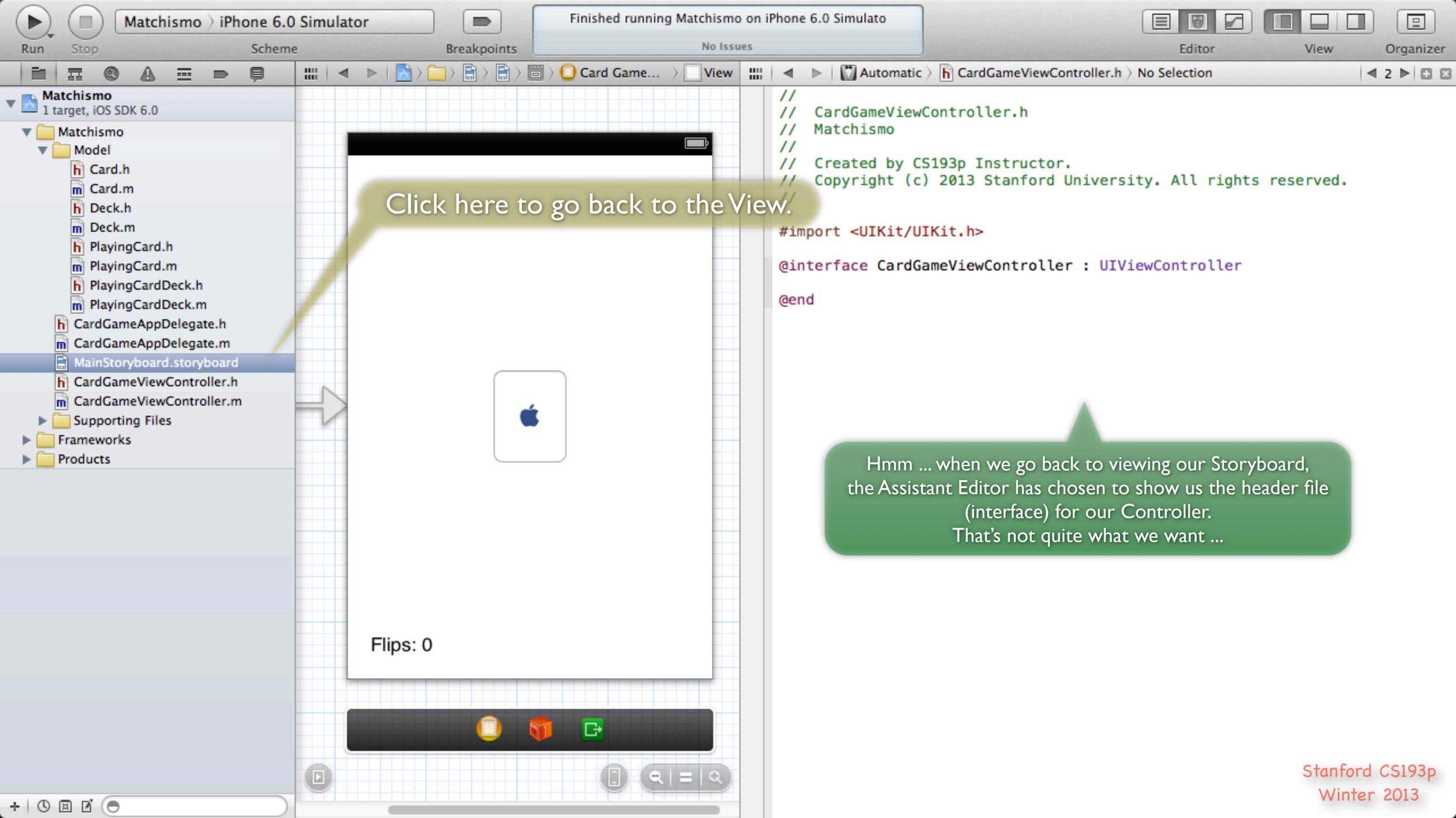

Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Card Game Model
Card.h
Card.m
Deck.h
Deck.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Matchismo 〉 M 〉 C 〉 Card.h 〉 No Selection
Counterparts 〉 Card.m 〉 No Selection
And for Card if you haven't already.
// 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=isFaceUp) BOOL faceUp;
@property (nonatomic, getter=isUnplayable) BOOL unplayable;
- (int)match:(NSArray *)otherCards;
@end
// 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
Winter 2013

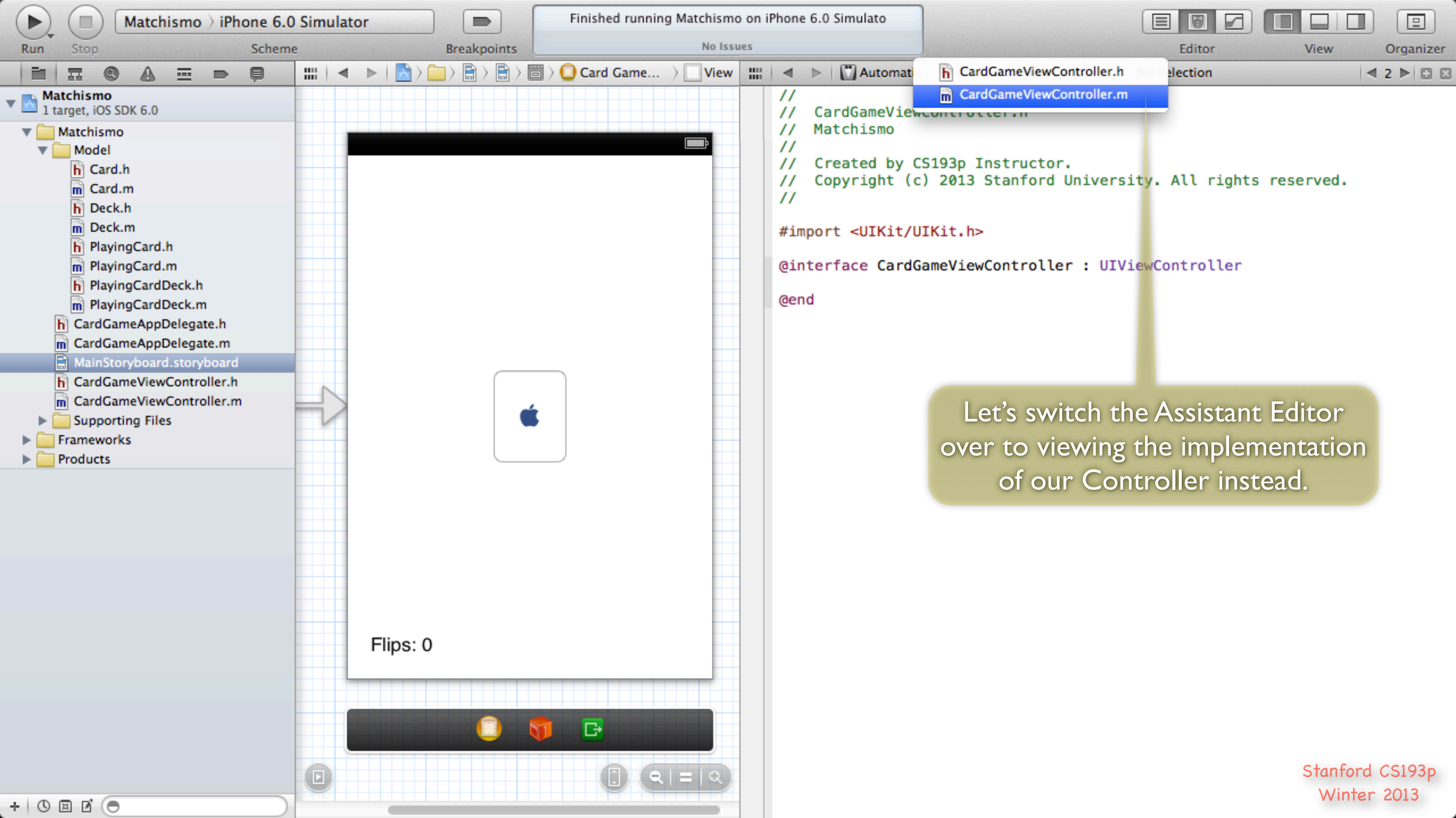


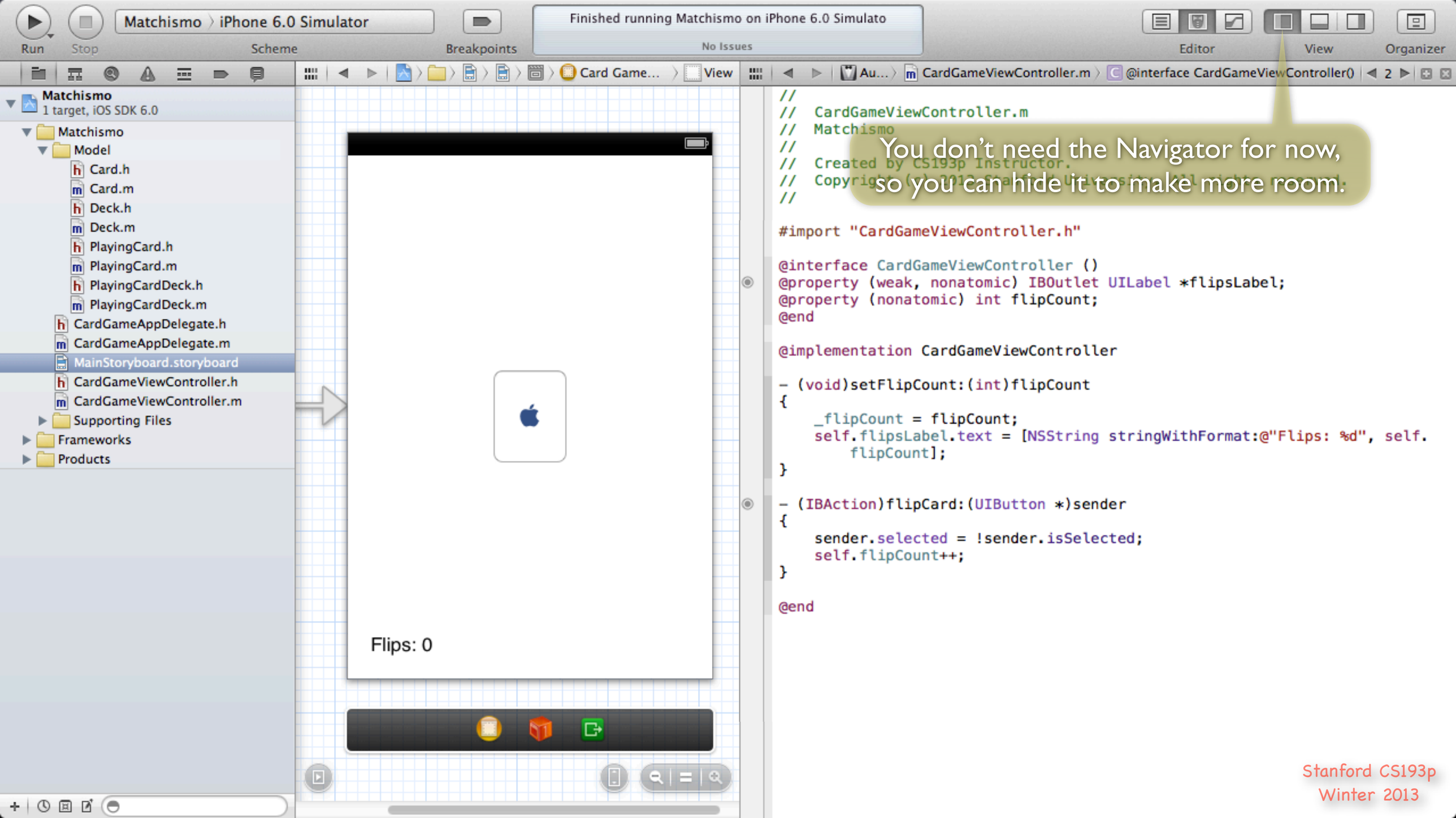

```
//
//  PlayingCard.h
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "Card.h"

@interface PlayingCard : Card

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

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

@end
```

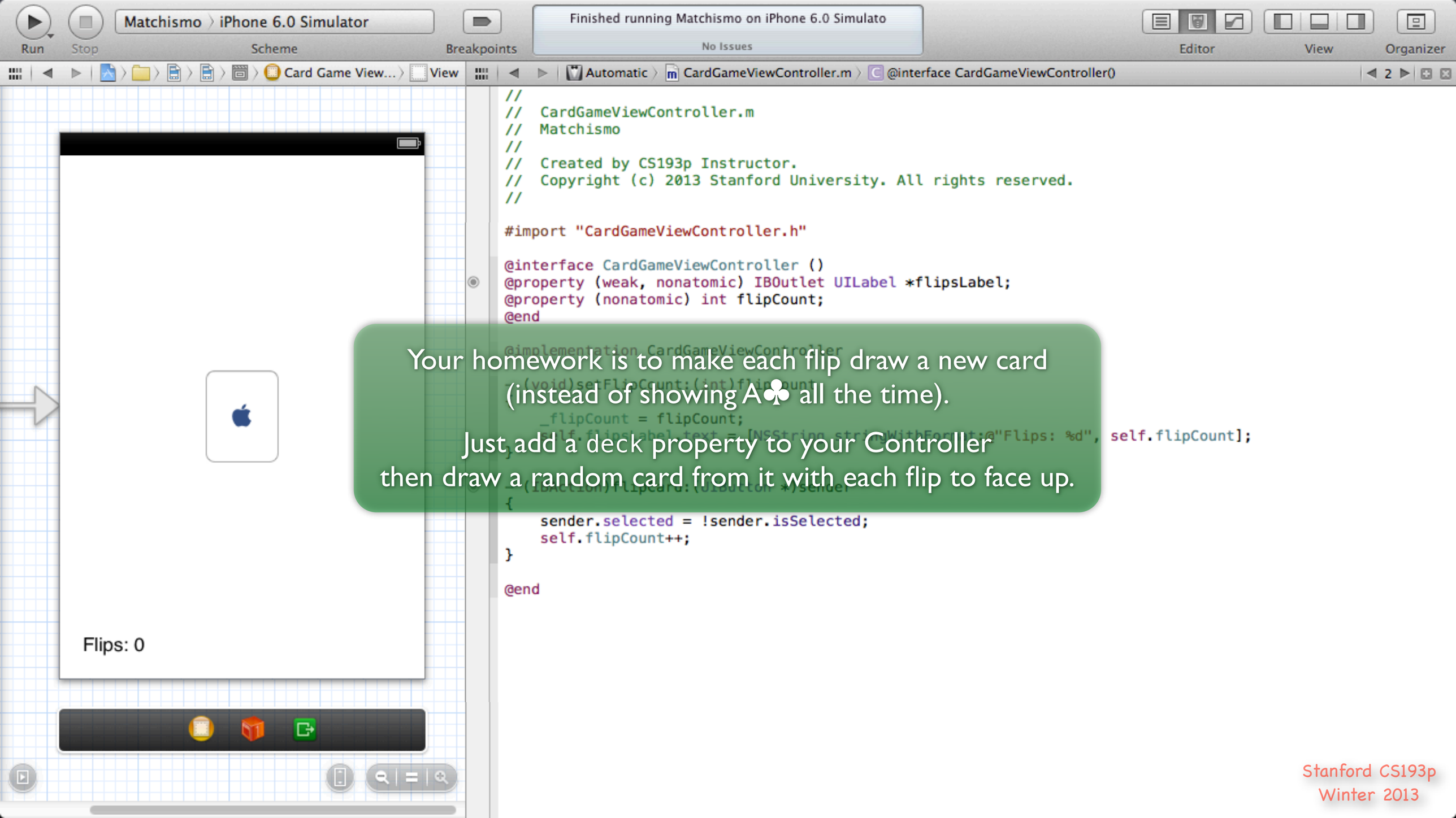
```
//
//  PlayingCard.m
//  Matchismo
//
//  Created by CS193p Instructor.
//  Copyright (c) 2013 Stanford University.
//  All rights reserved.
//

#import "PlayingCard.h"

@implementation PlayingCard

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

@synthesize suit = _suit;

+ (NSArray *)validSuits
{
    static NSArray *validSuits = nil;
    if (!validSuits) validSuits = @[@"♥",@"♦",@"♠",@"♣"];
    return validSuits;
}

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

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

+ (NSArray *)rankStrings
{
    static NSArray *rankStrings = nil;
```


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
Deck.h
Deck.m
PlayingCard.h
PlayingCard.m
PlayingCardDeck.h
PlayingCardDeck.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
PlayingCardDeck.h 〉 No Selection
Counterparts 〉 PlayingCardDeck.m 〉 No Selection
Use New File... to add the PlayingCardDeck class.
//
// PlayingCardDeck.h
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "Deck.h"
@interface PlayingCardDeck : Deck
@end
//
// PlayingCardDeck.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "PlayingCardDeck.h"
#import "PlayingCard.h"
@implementation PlayingCardDeck
- (id)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 atTop:YES];
}
}
}
return self;
}
@end
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
Deck.h
Deck.m
PlayingCard.h
PlayingCard.m
PlayingCardDeck.h
PlayingCardDeck.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Card Game...
View
Click here to go back to the View.
Flips: 0
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
Hmm ... when we go back to viewing our Storyboard,
the Assistant Editor has chosen to show us the header file
(interface) for our Controller.
That's not quite what we want ...


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
Deck.h
Deck.m
PlayingCard.h
PlayingCard.m
PlayingCardDeck.h
PlayingCardDeck.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Card Game...
View
Flips: 0
CardGameViewController.h
CardGameViewController.m
//
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface CardGameViewController : UIViewController
@end
Let's switch the Assistant Editor over to viewing the implementation of our Controller instead.


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Matchismo
1 target, iOS SDK 6.0
Matchismo
Model
Card.h
Card.m
Deck.h
Deck.m
PlayingCard.h
PlayingCard.m
PlayingCardDeck.h
PlayingCardDeck.m
CardGameAppDelegate.h
CardGameAppDelegate.m
MainStoryboard.storyboard
CardGameViewController.h
CardGameViewController.m
Supporting Files
Frameworks
Products
Card Game...
View
Flips: 0
CardGameViewController.m
@interface CardGameViewController()
2
//
// CardGameViewController.m
// Matchismo
//
// Created by CS193p Instructor.
// Copyright (c) 2011 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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
You don't need the Navigator for now,
so you can hide it to make more room.
Stanford CS193p
Winter 2013


Run
Stop
Matchismo 〉 iPhone 6.0 Simulator
Scheme
Breakpoints
Finished running Matchismo on iPhone 6.0 Simulato
No Issues
Editor
View
Organizer
Card Game View... 〉 View
Automatic 〉 CardGameViewController.m 〉 @interface CardGameViewController()
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)flipCard:(UIButton *)sender
{
sender.selected = !sender.isSelected;
self.flipCount++;
}
@end
Your homework is to make each flip draw a new card
(instead of showing A♣ all the time).
Just add a deck property to your Controller
then draw a random card from it with each flip to face up.
Stanford CS193p
Winter 2013

Coming Up

- Next Lecture
  - More Xcode Demonstration!
  - Much more sophisticated Card Game Model and UI
- And later next week ...
  - Objective-C language in depth
  - Foundation classes: arrays, dictionaries, strings, etc.
  - Dynamic vs. static typing
  - Protocols, categories and much, much more!