# Stanford CS193p

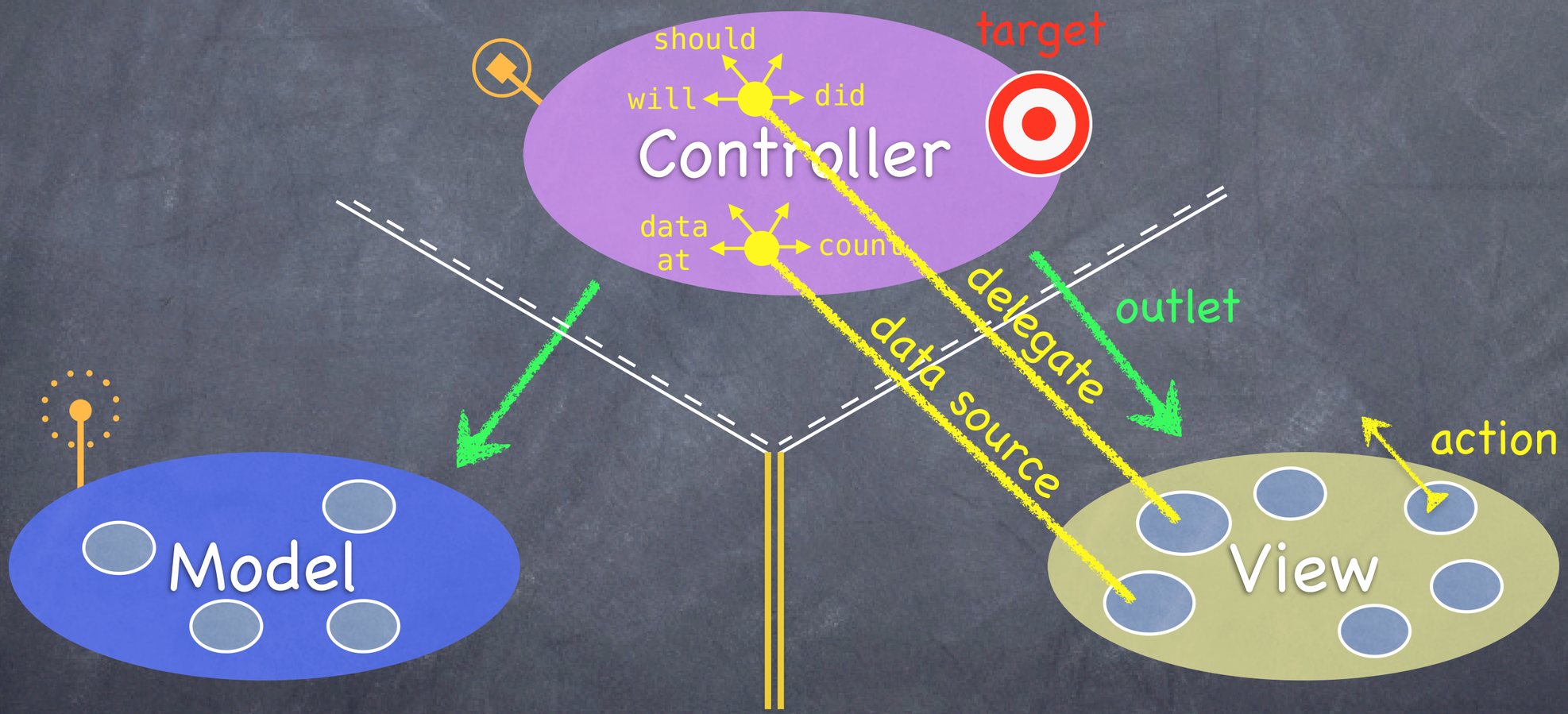
## Developing Applications for iPhone 4, iPod Touch, & iPad

## Fall 2010

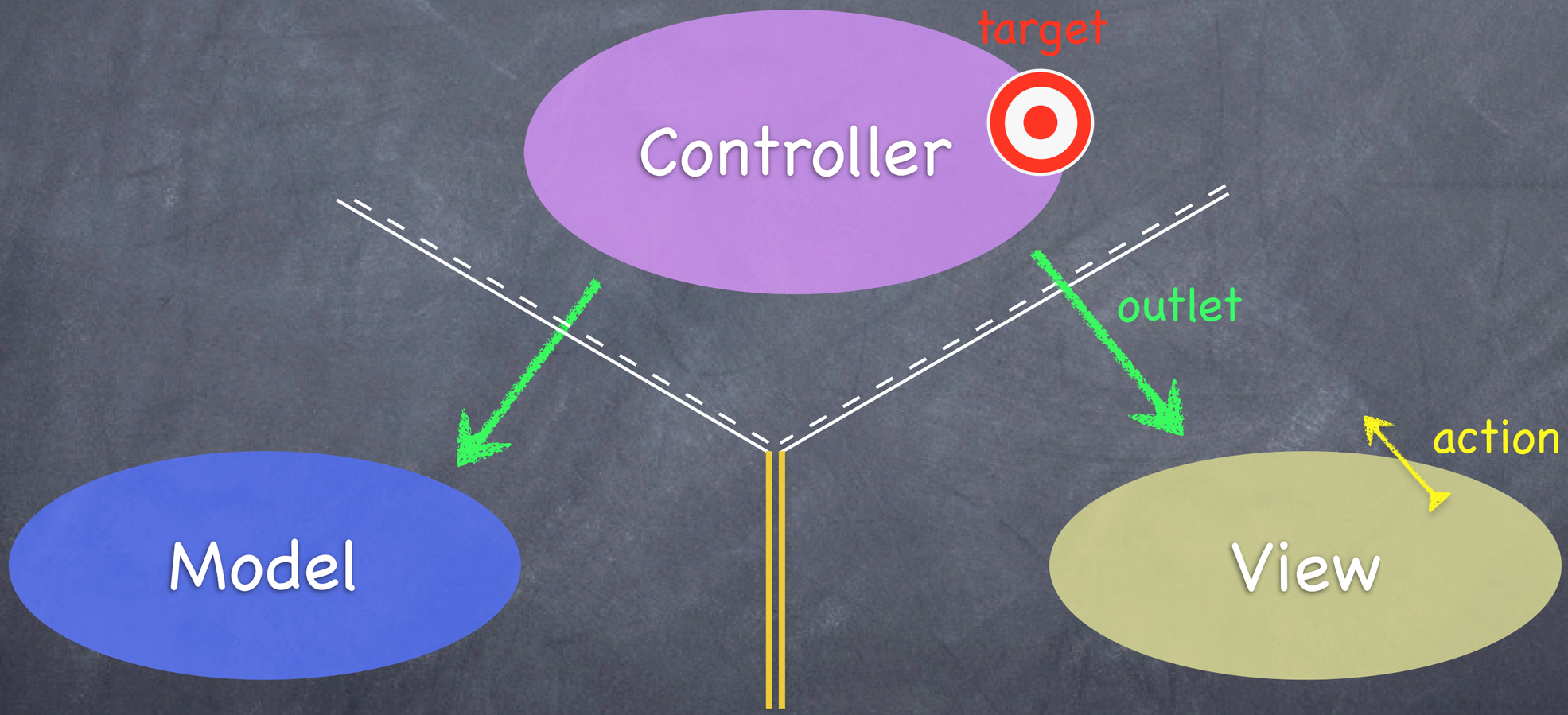
# Today

- MVC
  - Calculator
- Objective-C
  - Declaring and implementing objects
  - Sending messages between objects
- Interface Builder
  - Graphically creating your View
  - "Wiring up" objects to send messages to each other
  - Setting the properties of objects
- Xcode
  - Managing and editing your code
  - Running your application in the simulator

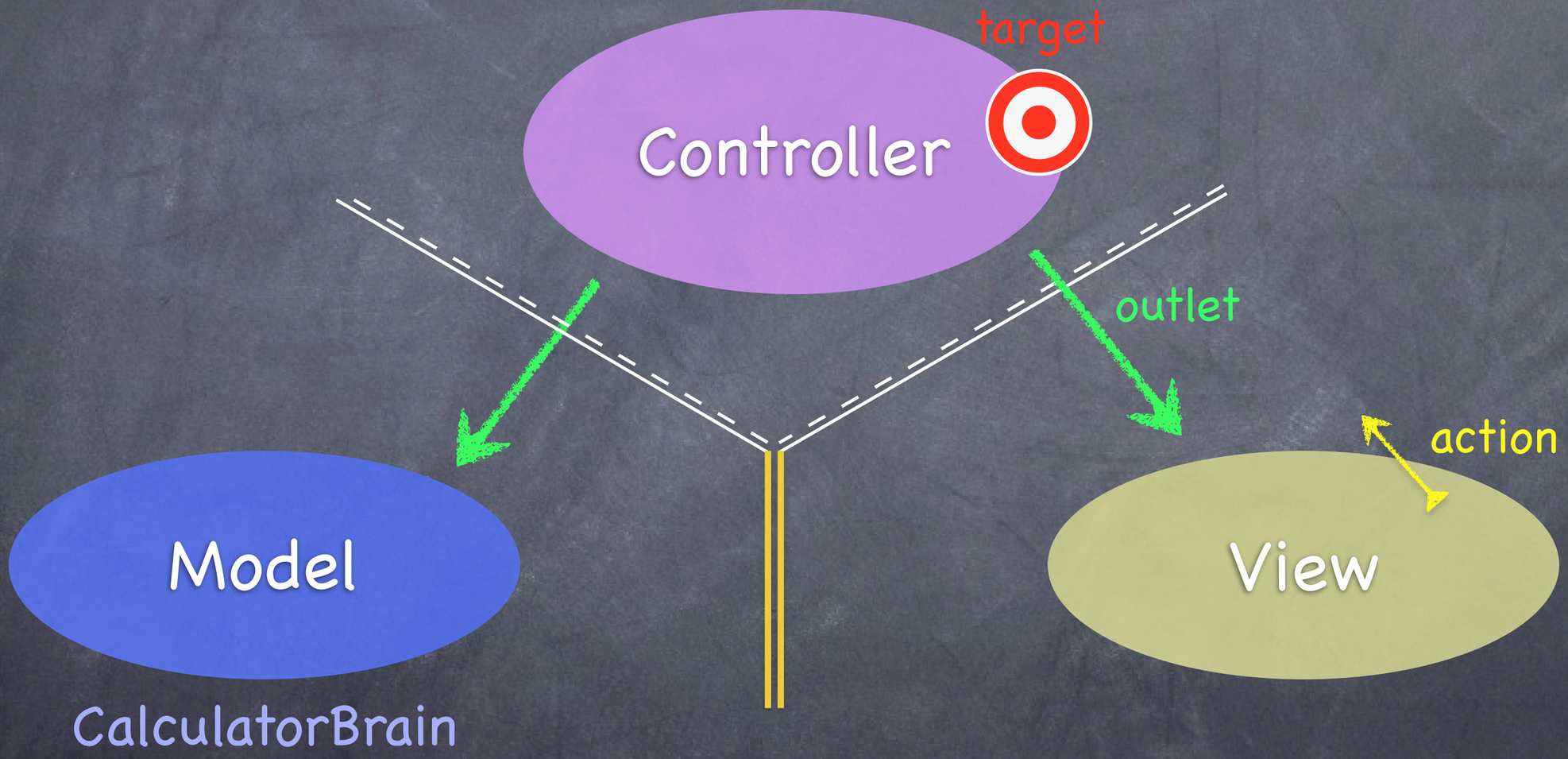
# Calculator MVC



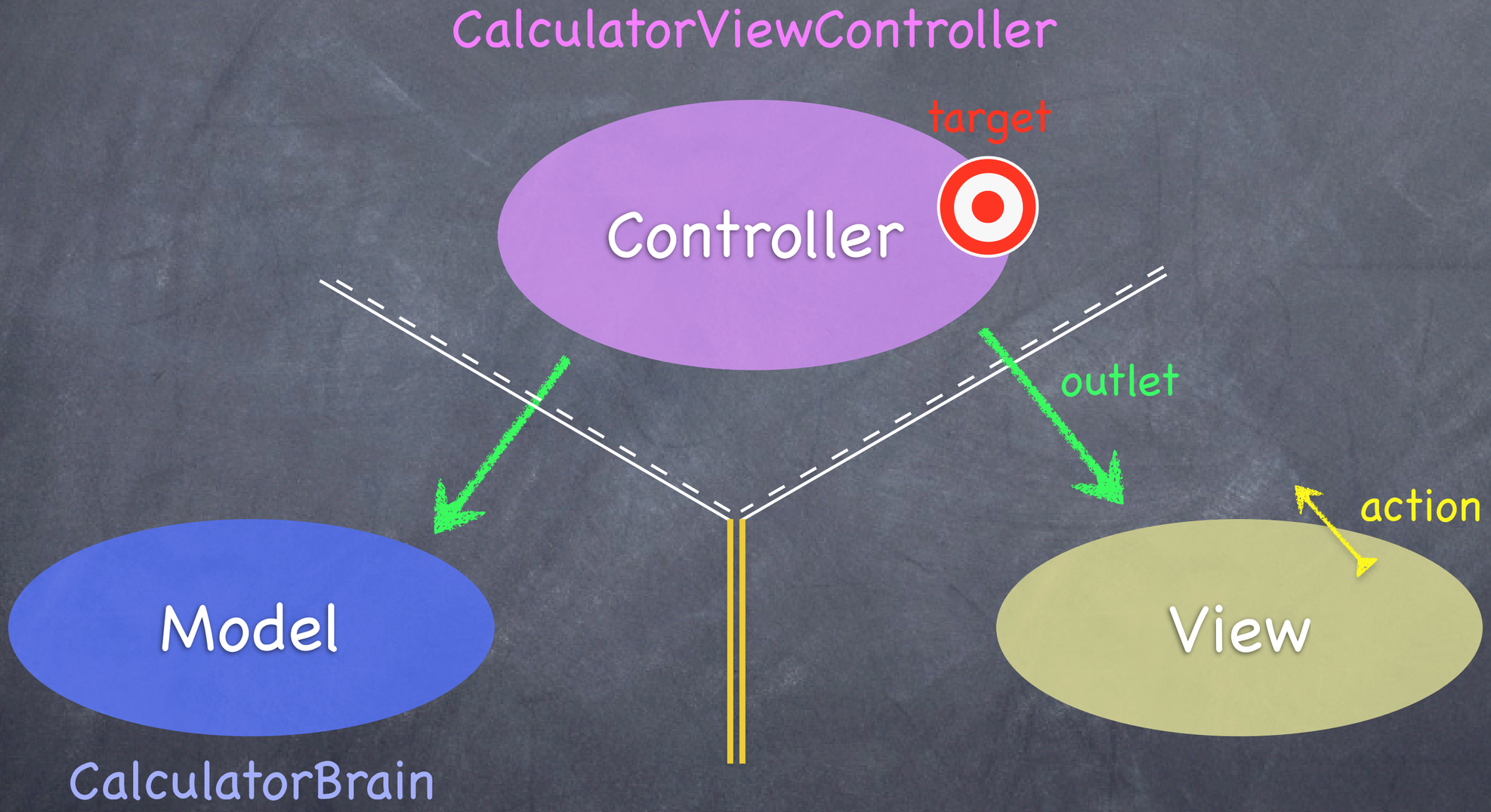
# Calculator MVC



# Calculator MVC

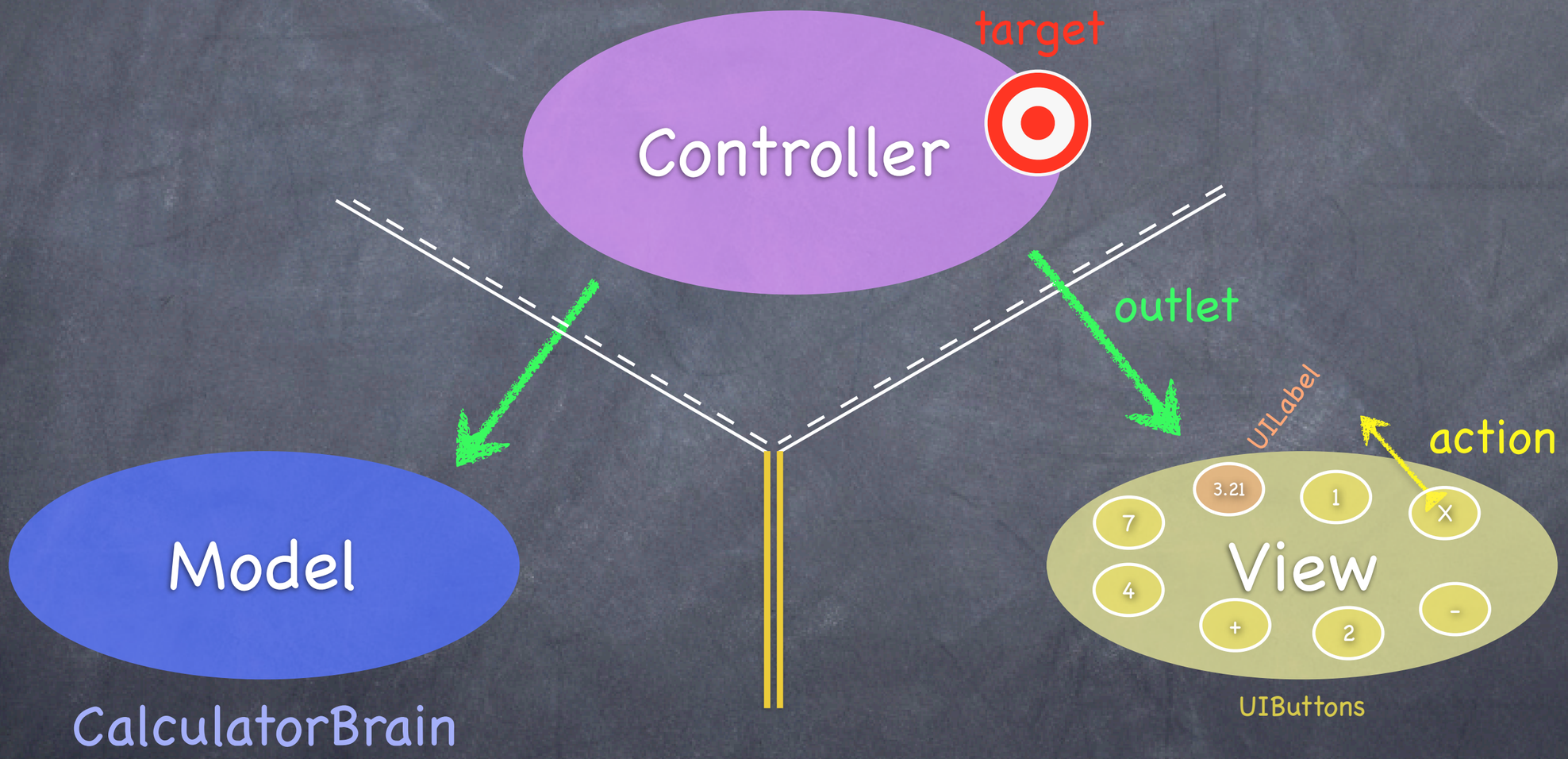


# Calculator MVC



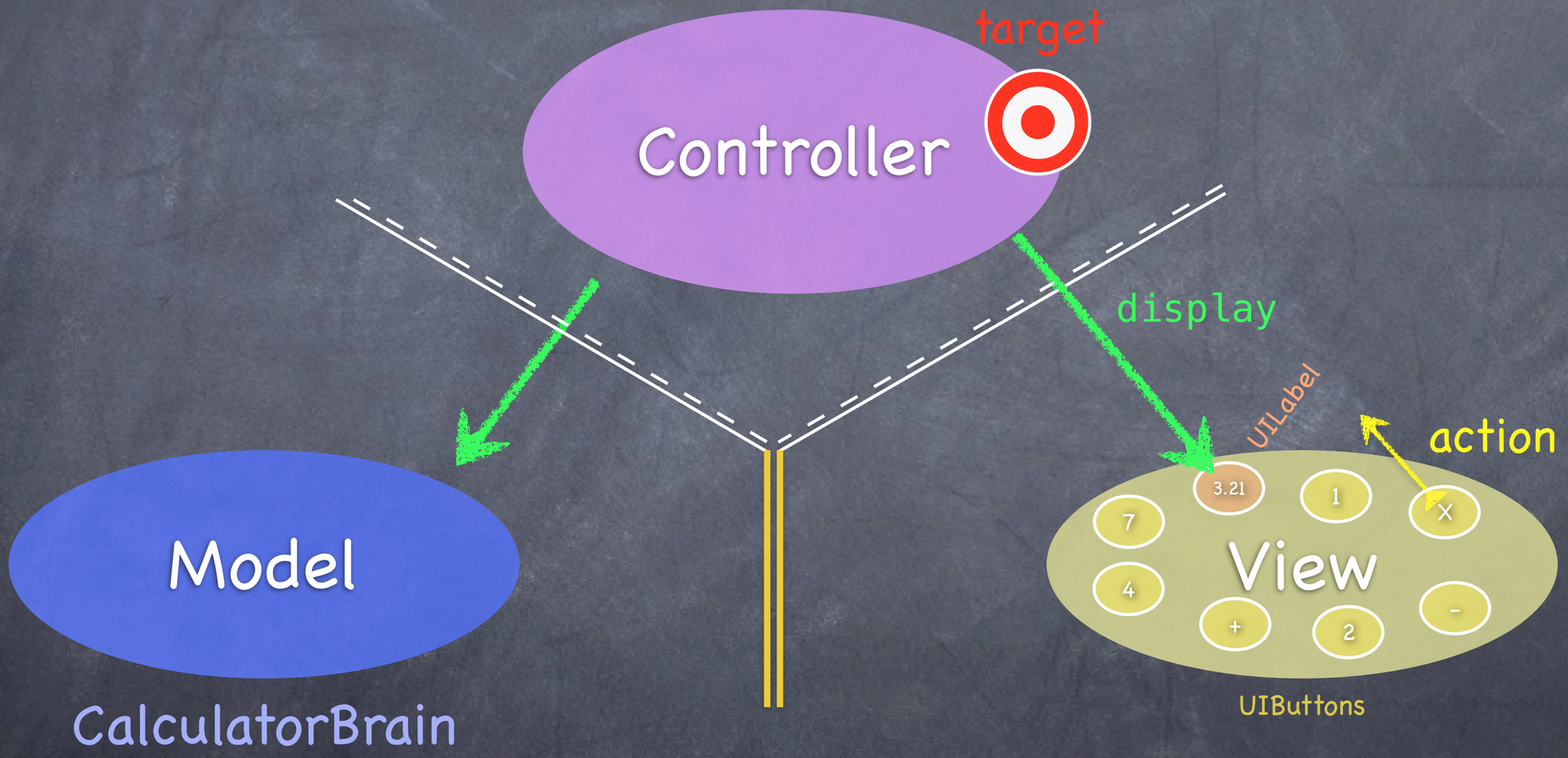
# Calculator MVC

CalculatorViewController

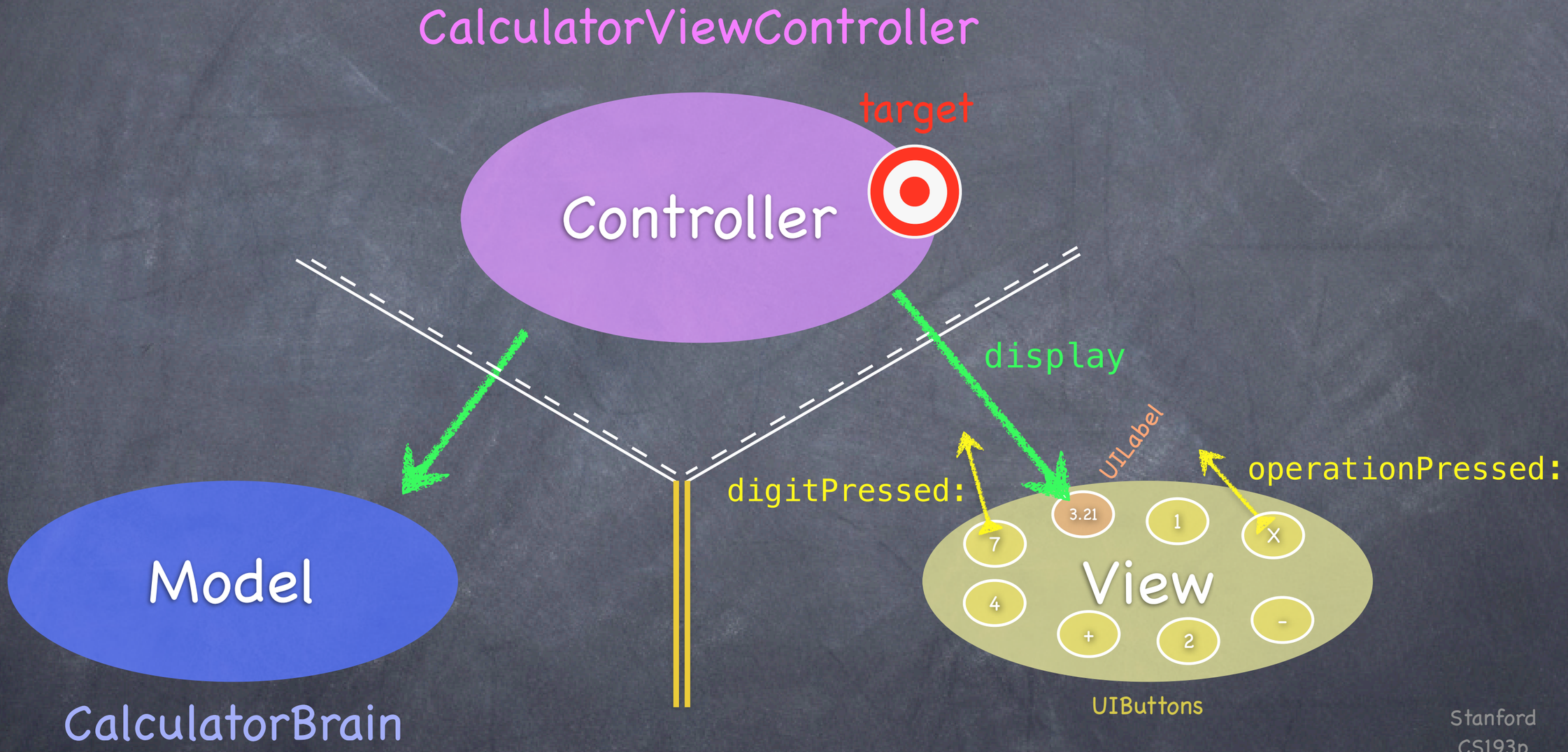


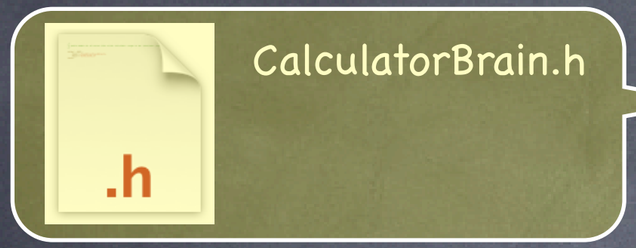
# Calculator MVC

CalculatorViewController



# Calculator MVC






This is the header file for this class.
It documents its public API.

Model


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject

@end
```

The name of this class.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject

@end
```

This class's superclass.


CalculatorBrain.h

We must import the header for our superclass.

Model

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

@interface CalculatorBrain : NSObject



@end
```


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{

}

@end
```

Instance variables go here.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}




@end
```


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

@end
```

Method declarations go here.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```

Specifying void as the return type means that this method returns no value.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```

The name of this method is "setOperand:"


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```

It takes one argument, a double called "anOperand"


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```

Don't forget a semicolon here!


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;

@end
```

This method returns a double.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}


- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```

It takes as its argument a pointer to an NSString object.
That's right, we're passing an object to this method.


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;

- (NSArray *)foo:(int)zap bar:(id)pow;

@end
```


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}


- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;

- (NSArray *)foo:(int)zap bar:(id)pow;

@end
```

This method takes two arguments and is called "foo:bar:"
(pronounced "foo colon bar colon")


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}


- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;

- (NSArray *)foo:(int)zap bar:(id)pow;

@end
```

It returns a pointer to an NSArray
(a collection class in Foundation).


CalculatorBrain.h

Model

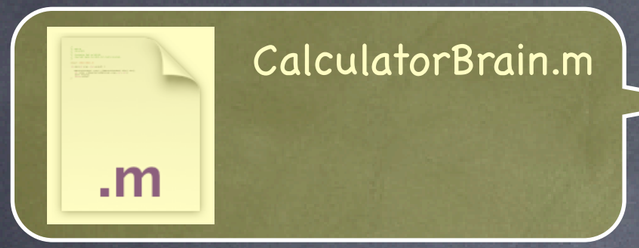
```
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;

- (NSArray *)foo:(int)zap bar:(id)pow;

@end
```

The second argument is of type "id".
This means "a pointer to any kind of object!"


CalculatorBrain.h

Model

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

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;


@end
```




This is the implementation file.
Both public and private implementation goes here.

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain



@end
```


CalculatorBrain.m

We must import our own header file.

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain



@end
```

Model


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

@end
```

Note that we don’t specify our superclass in the implementation


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    <code goes here>
}



@end
```

No semicolon this time!


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```

Square brackets mean “send a message.”


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```

This is the object to send the message to
(in this case, the NSString called “operation” that was passed as an argument to performOperation:).


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```

This is the message to send.


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```

And this is its one (in this case) argument.


CalculatorBrain.m

Model

```
#import “CalculatorBrain.h”

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}

@end
```

# Controller

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

@interface CalculatorViewController:UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}


- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;

@end
```

# Controller

Our Controller inherits from UIViewController. UIKit supports MVC primarily through this class.

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

@interface CalculatorViewController : UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}


- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;

@end
```

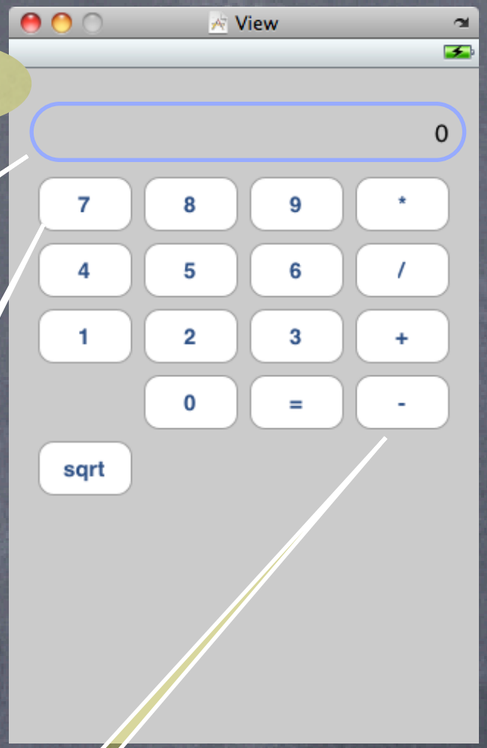
# Controller

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

@interface CalculatorViewController : UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}


- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;

@end
```

This is going to point to our Model CalculatorBrain

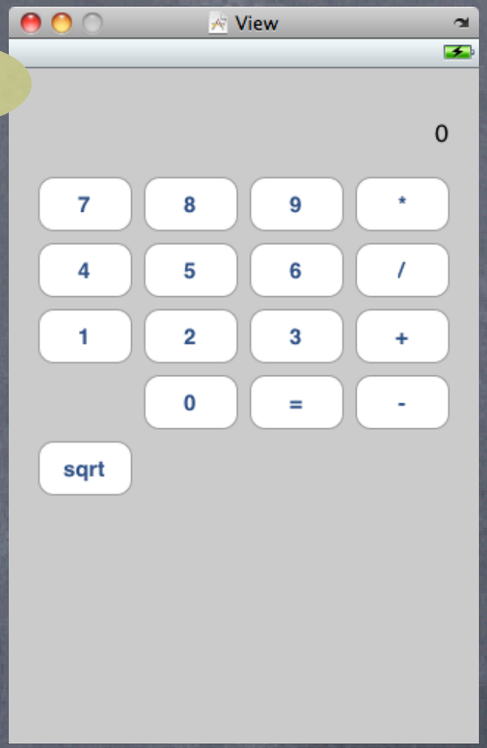
# Controller

These hook up to our View



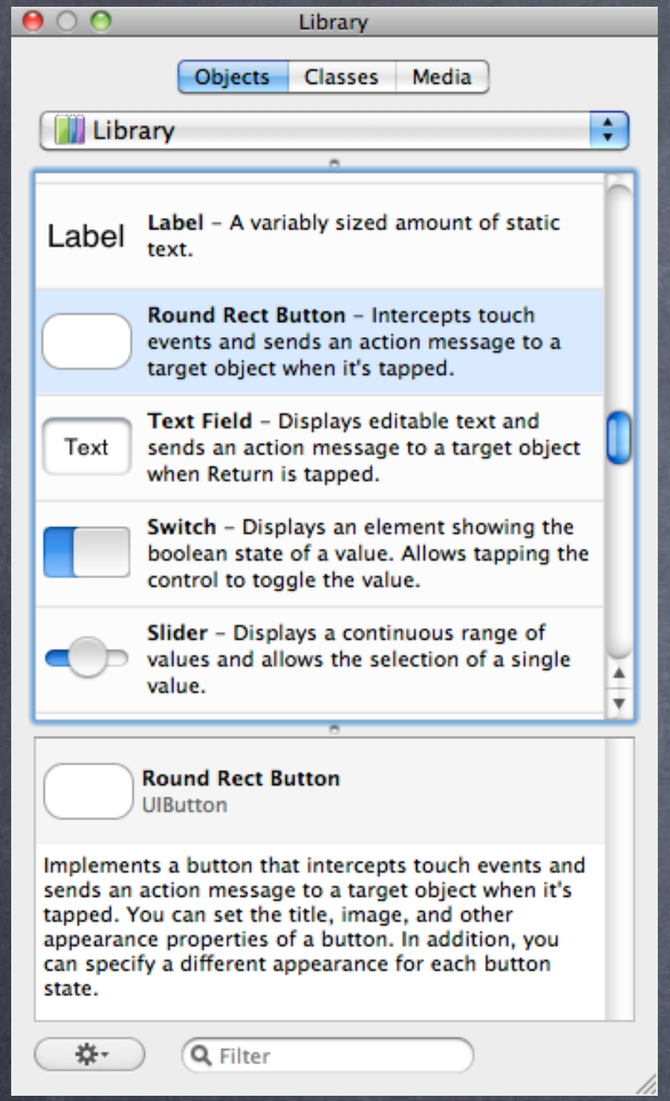

```
#import <UIKit/UIKit.h>

@interface CalculatorViewController:UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}

- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;

@end
```

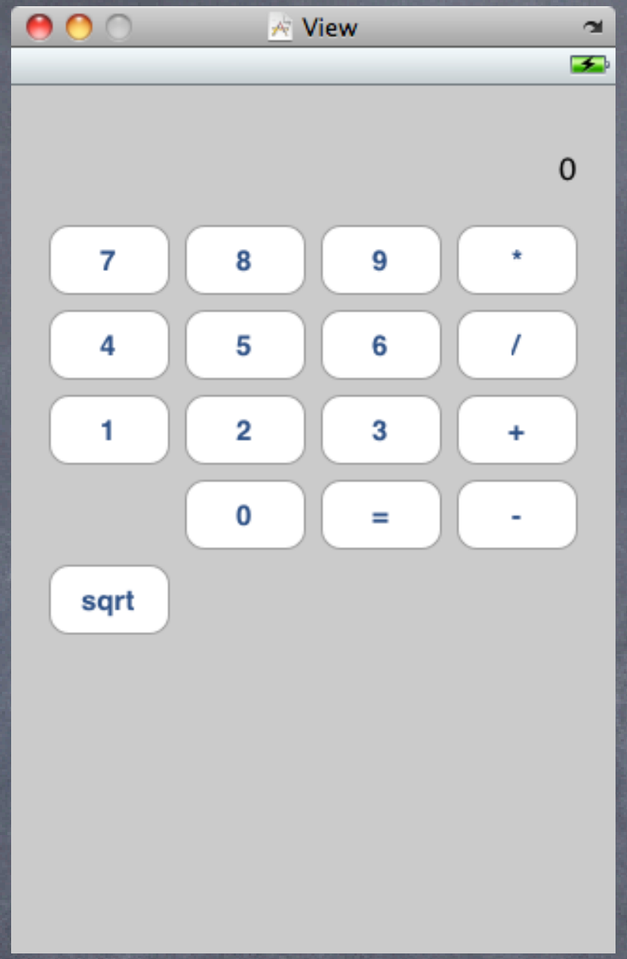
# Controller

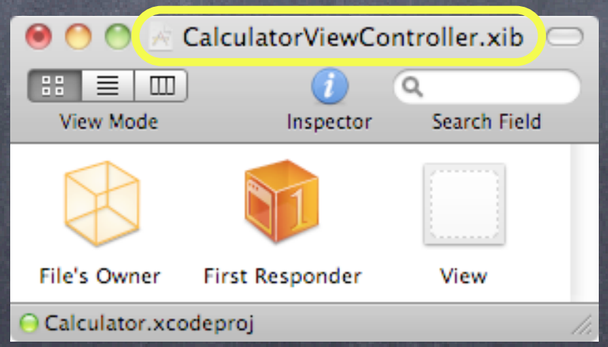


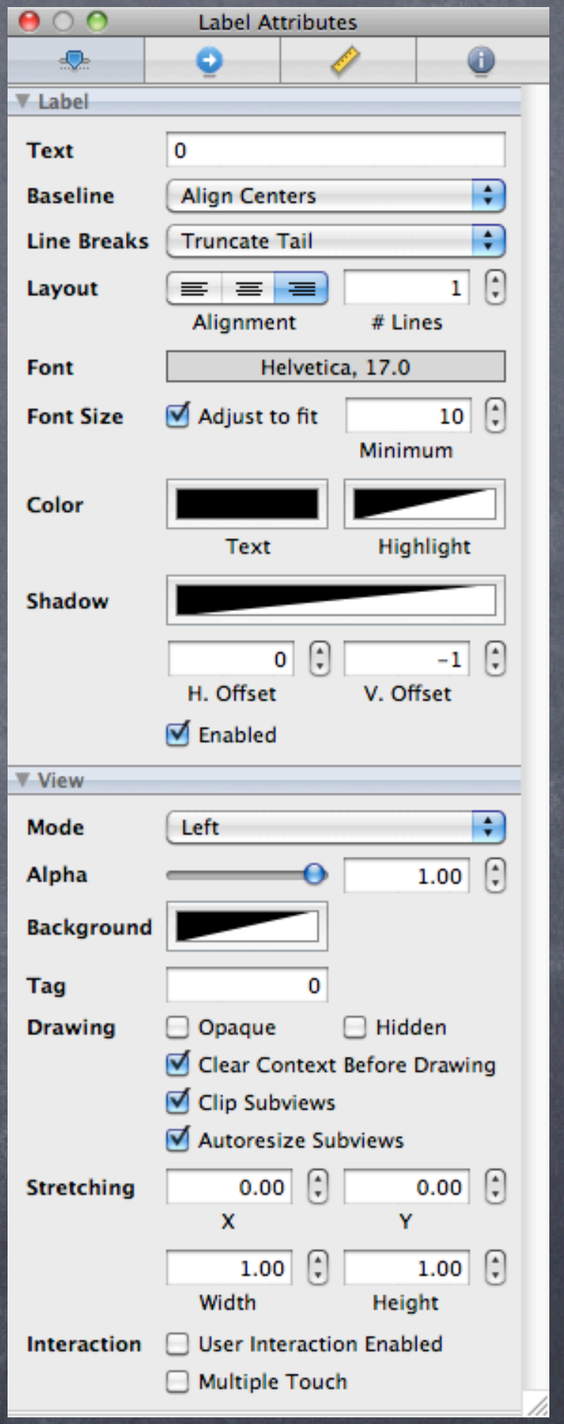

```
#import <UIKit/UIKit.h>

@interface CalculatorViewController:UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}

- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;

@end
```

Model



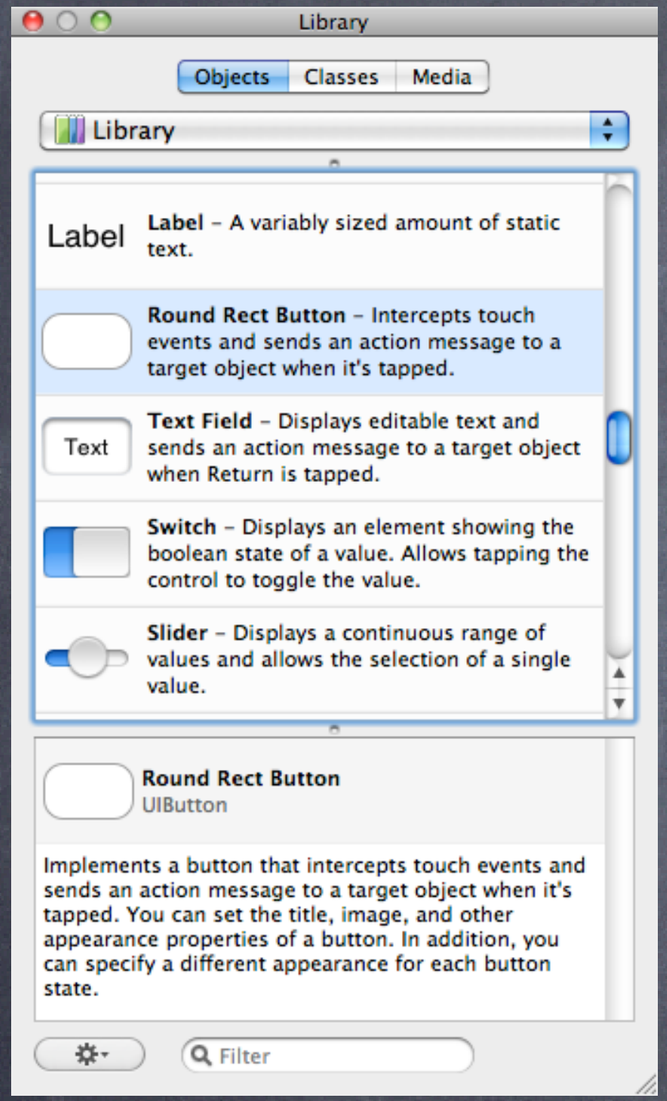



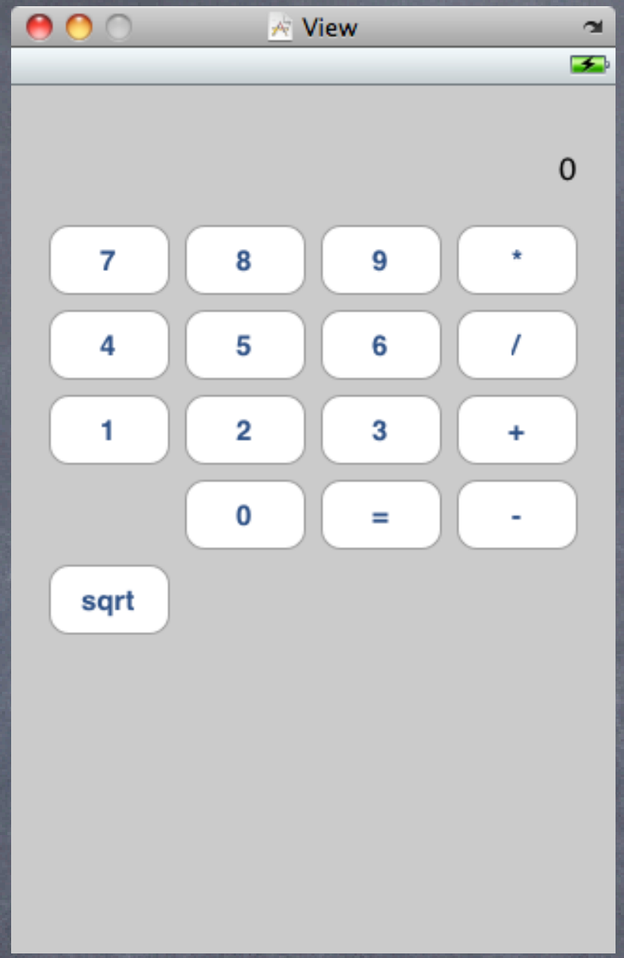



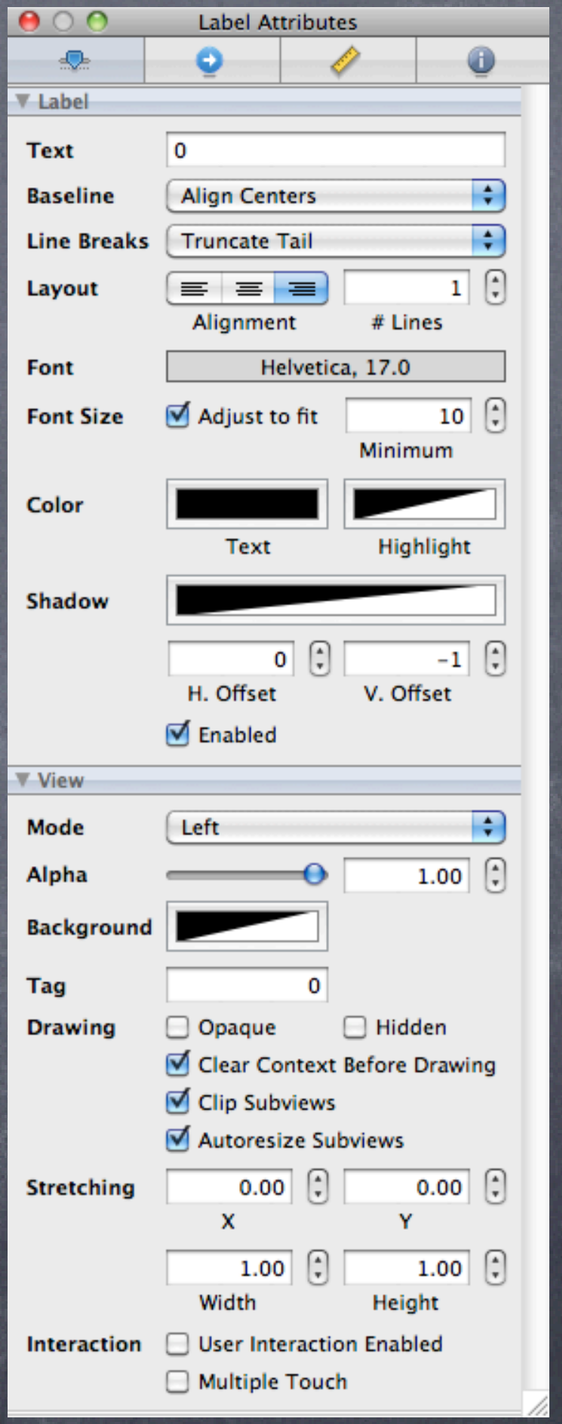



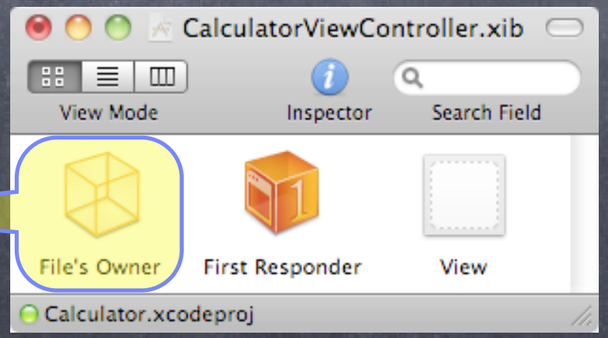

CalculatorViewController.xib

Stanford
CS193p
Fall 2010



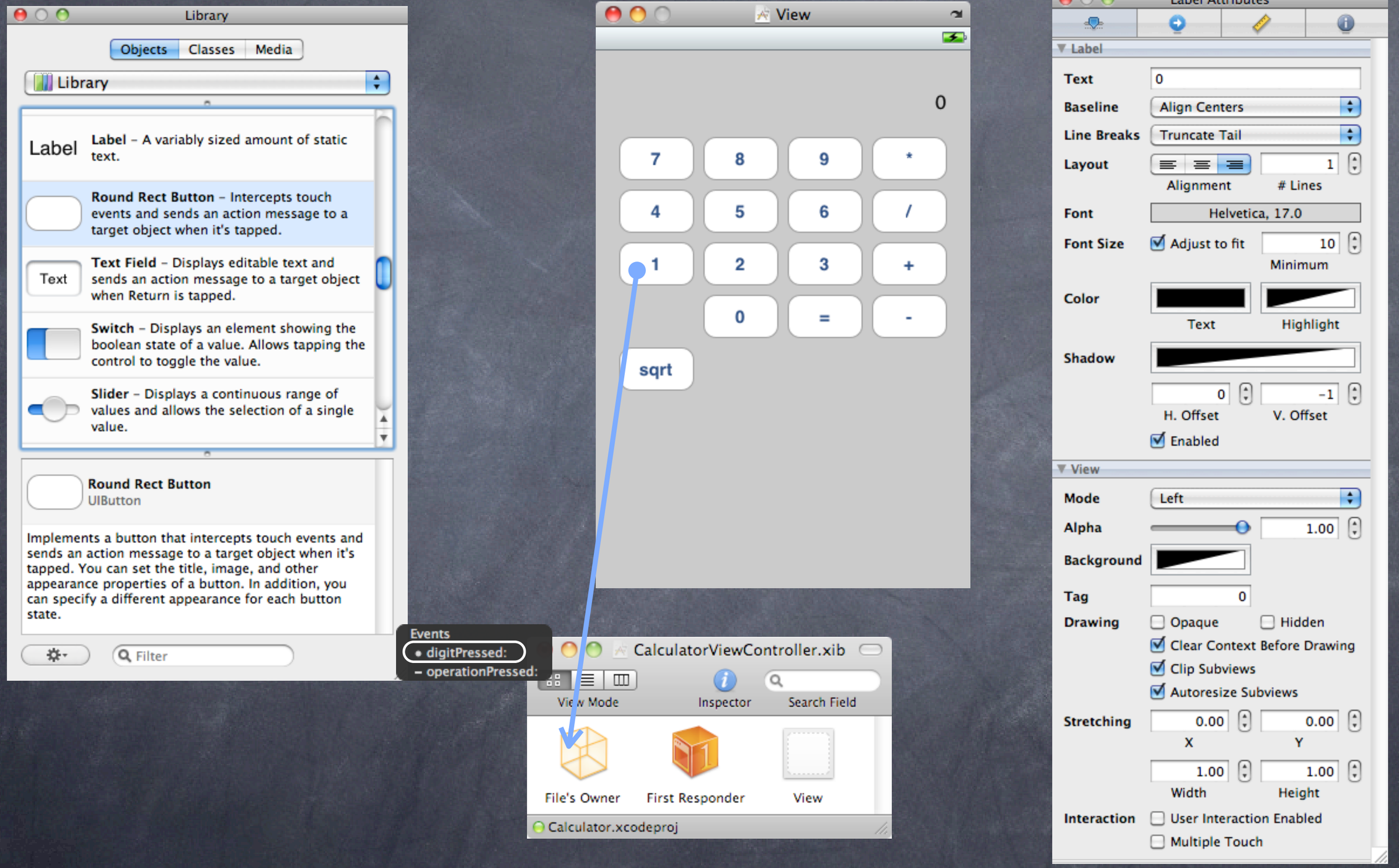



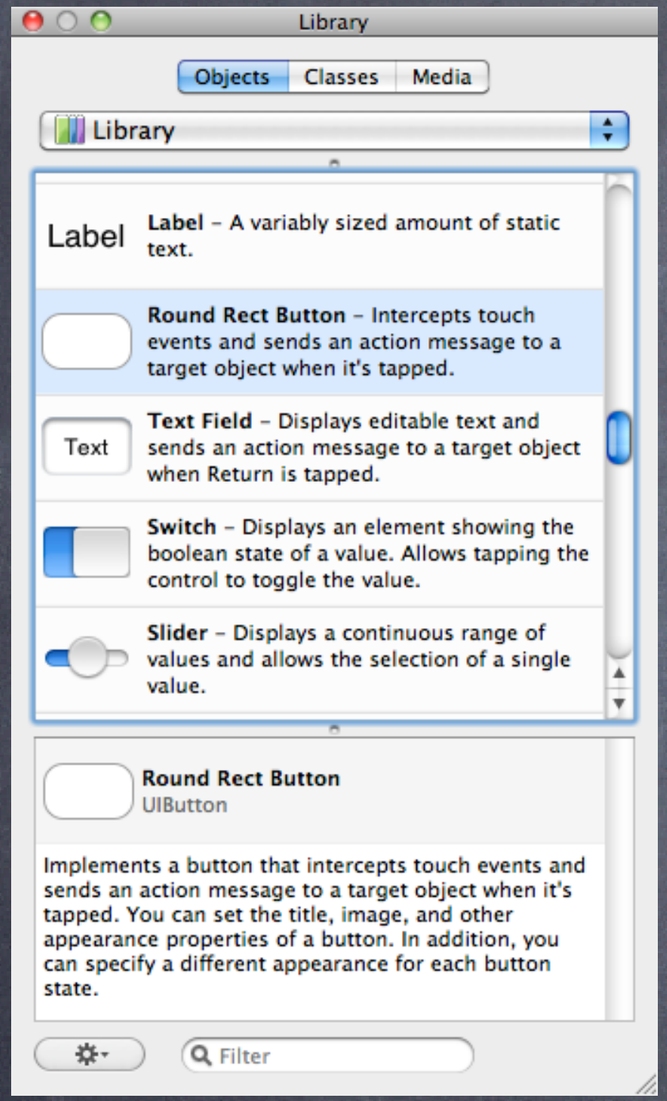

"File's Owner" is our Controller

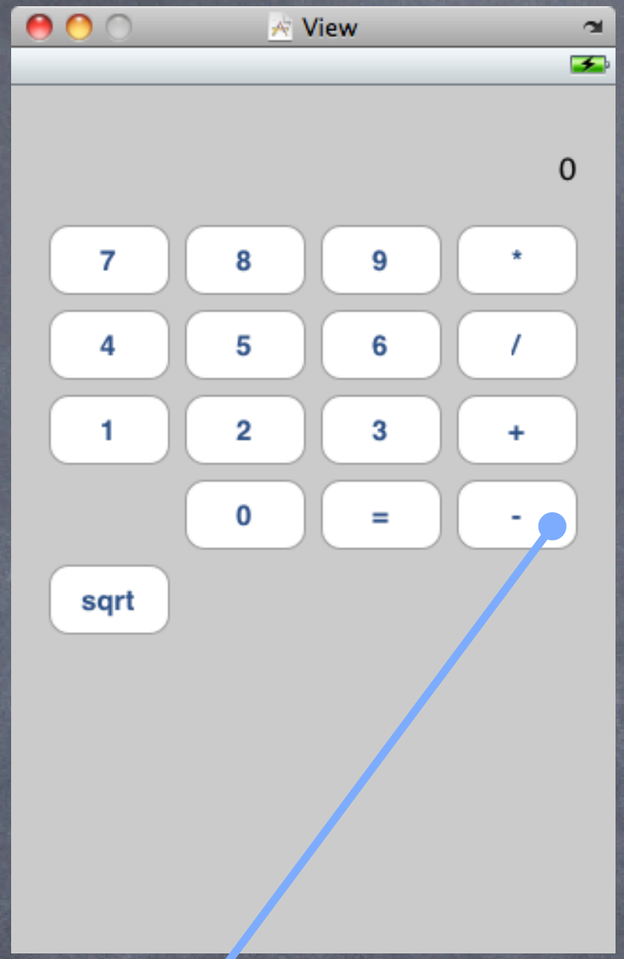
CalculatorViewController.xib

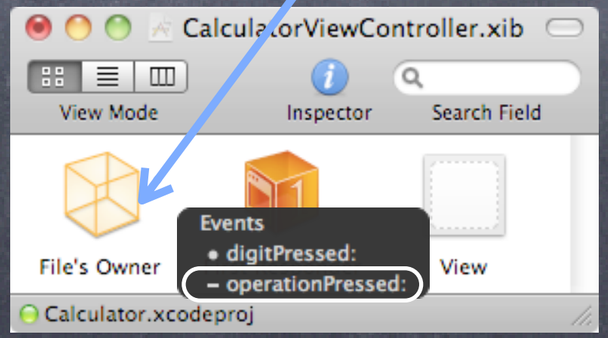
Stanford
CS193p
Fall 2010

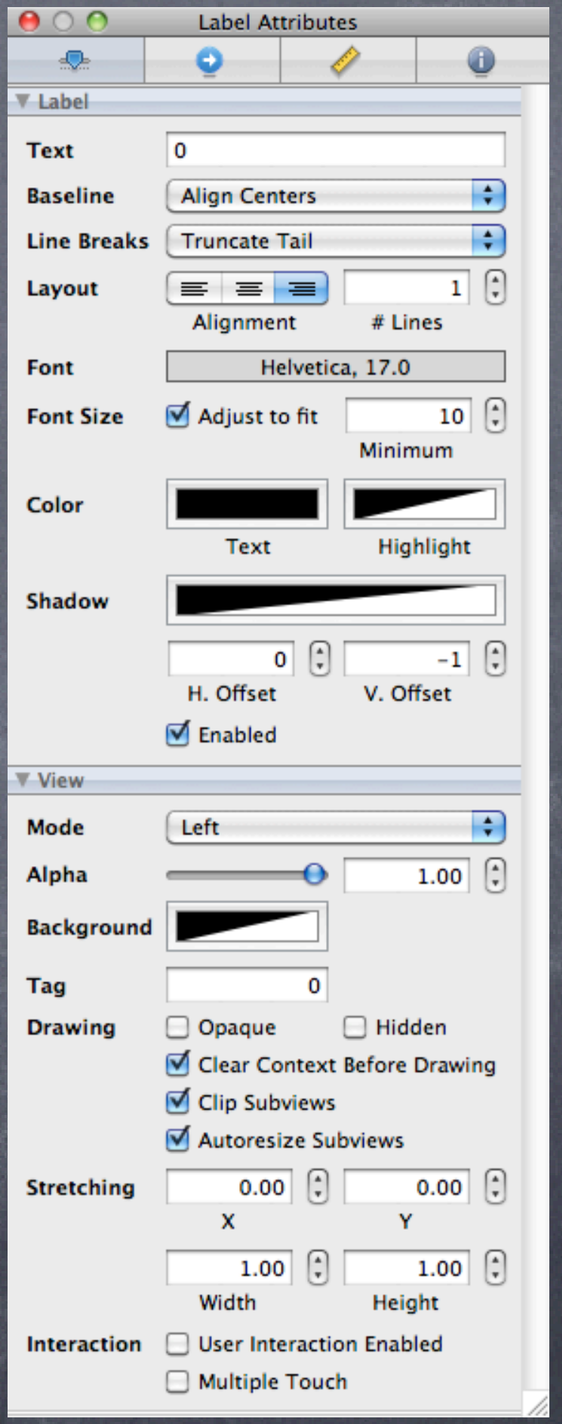



Stanford
CS193p
Fall 2010

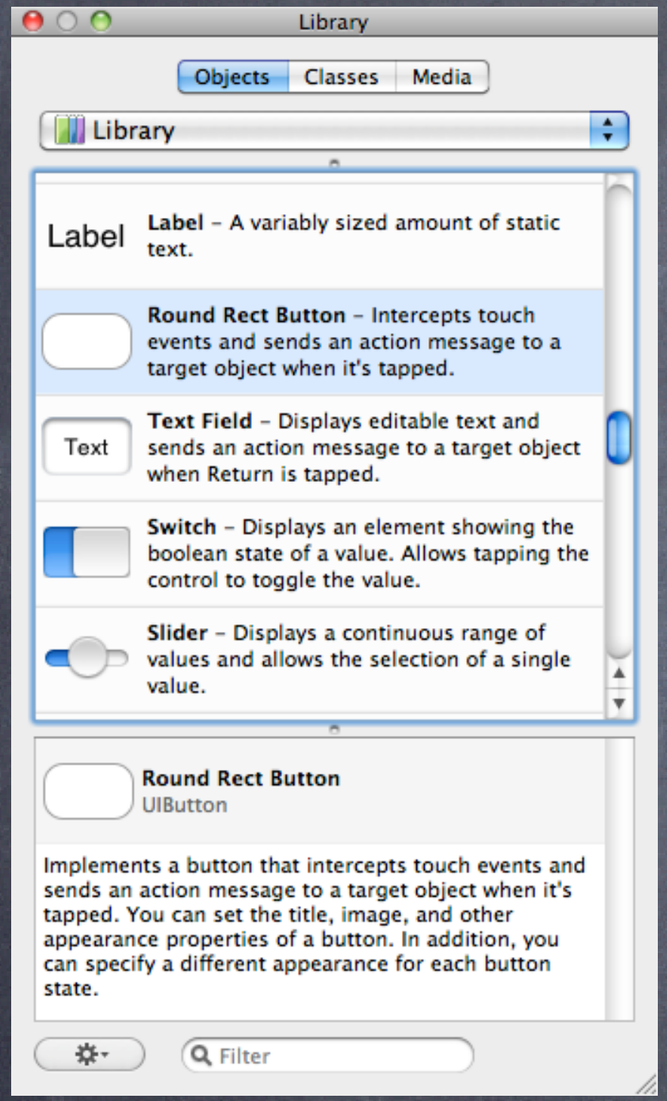


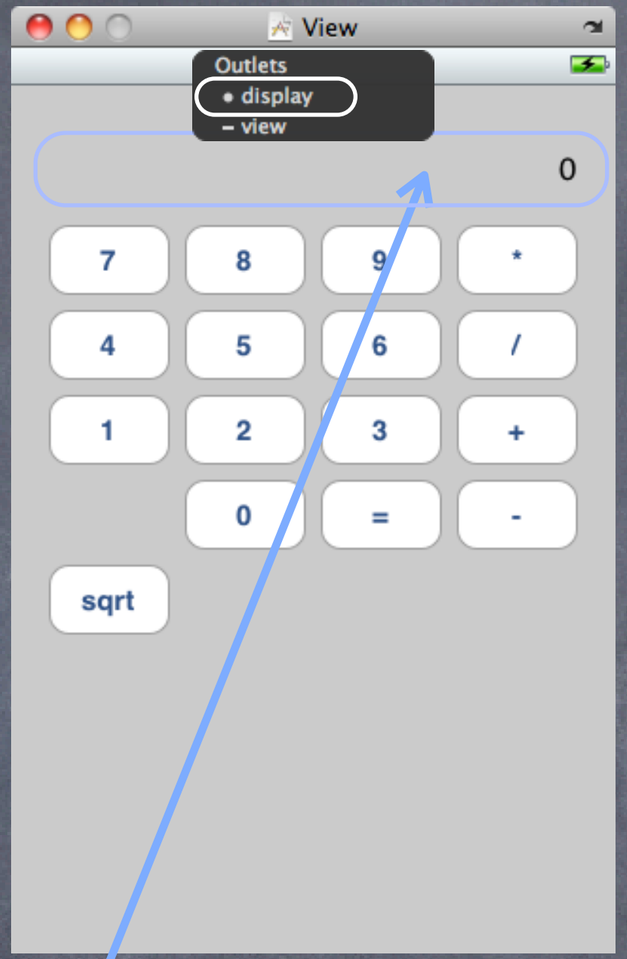



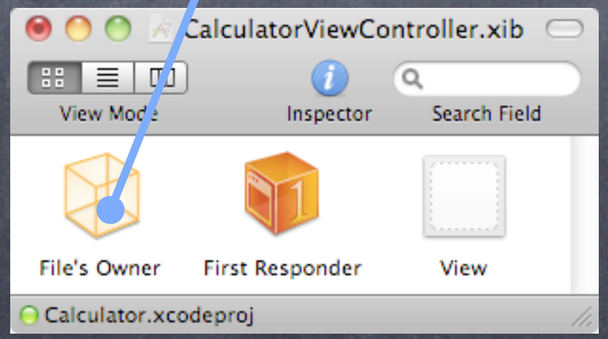



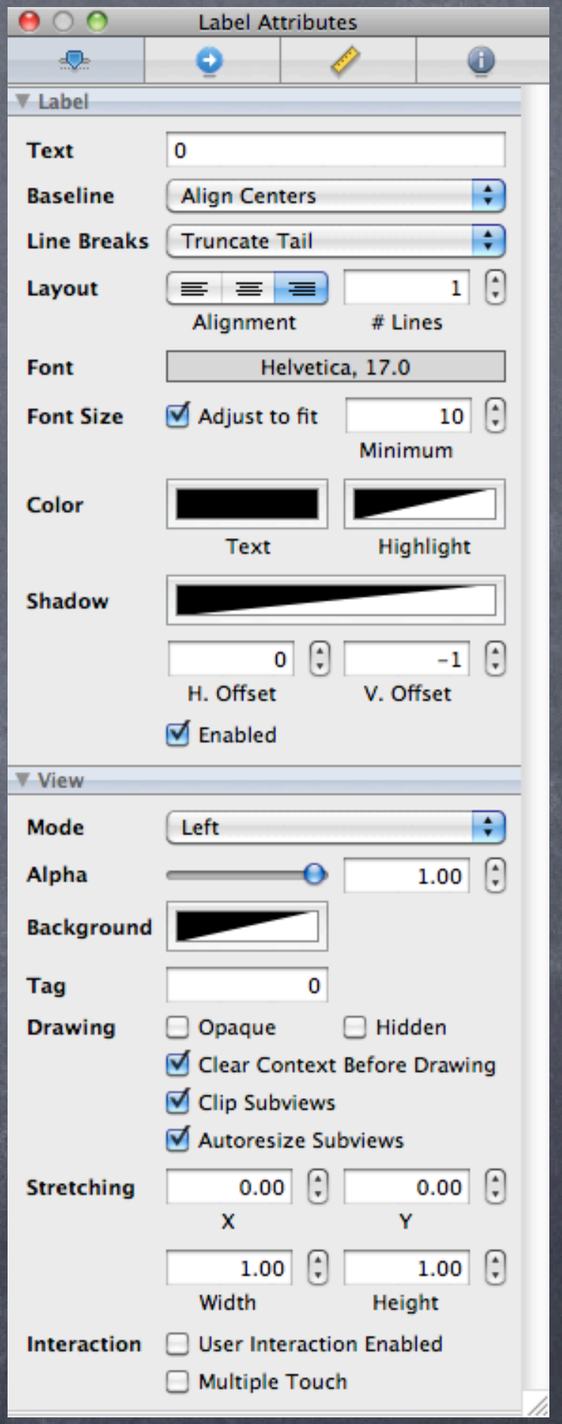




Stanford
CS193p
Fall 2010




Stanford
CS193p
Fall 2010

# My First Project

A picture (or demo) is worth 1,000 words.