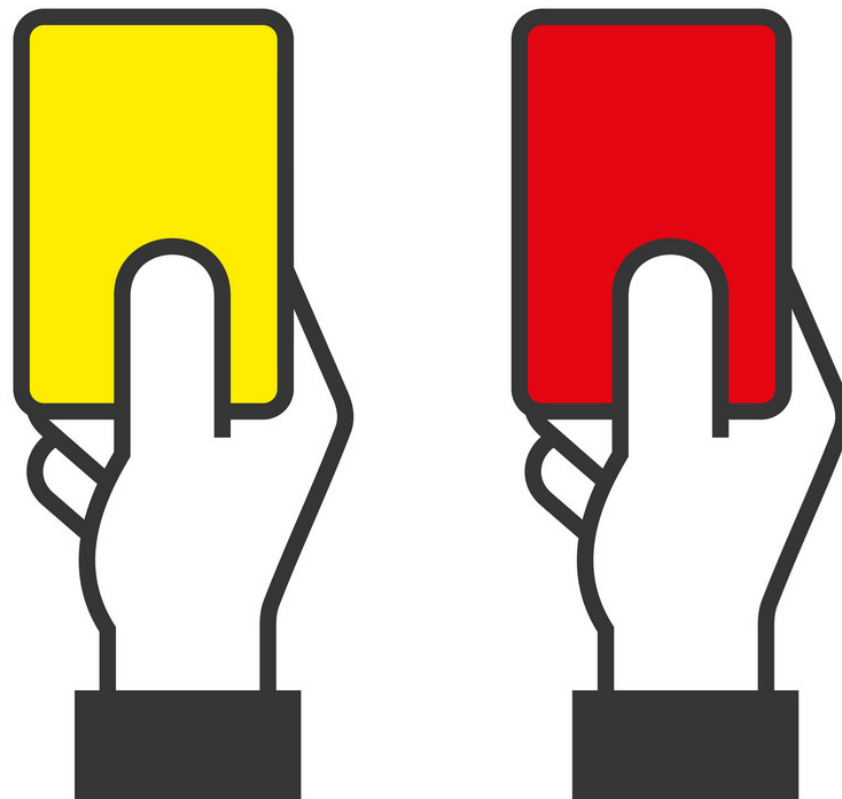
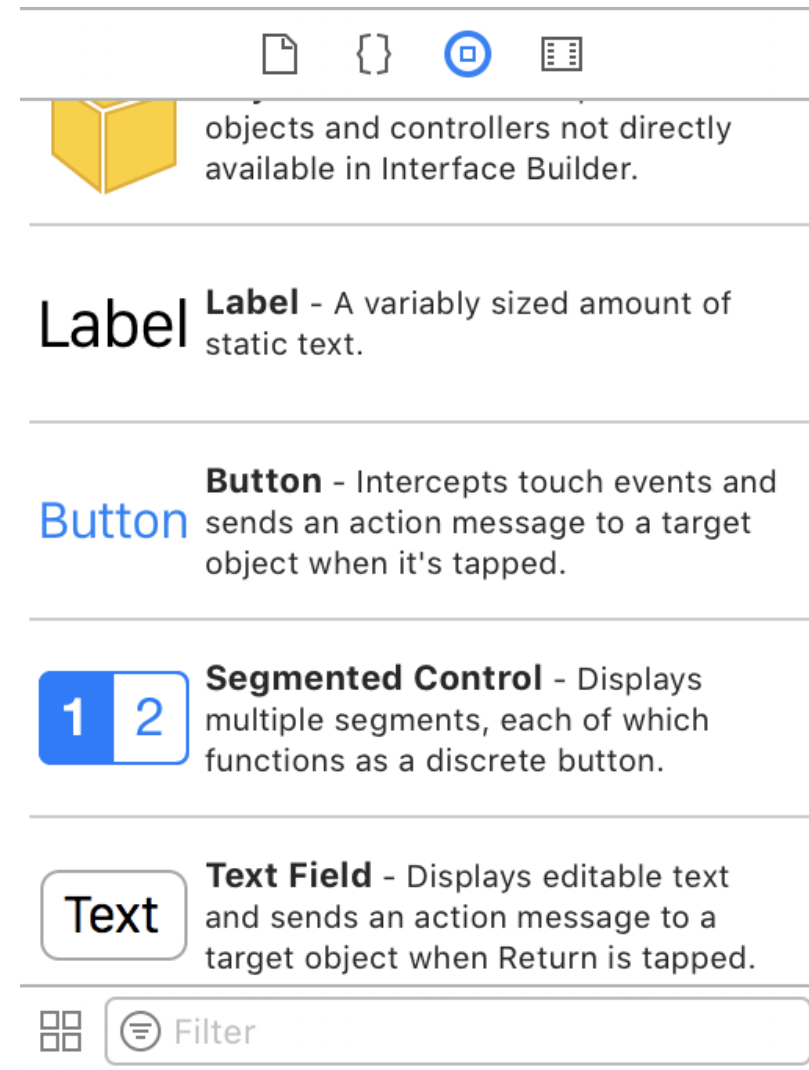


Concentration Game Tutorial

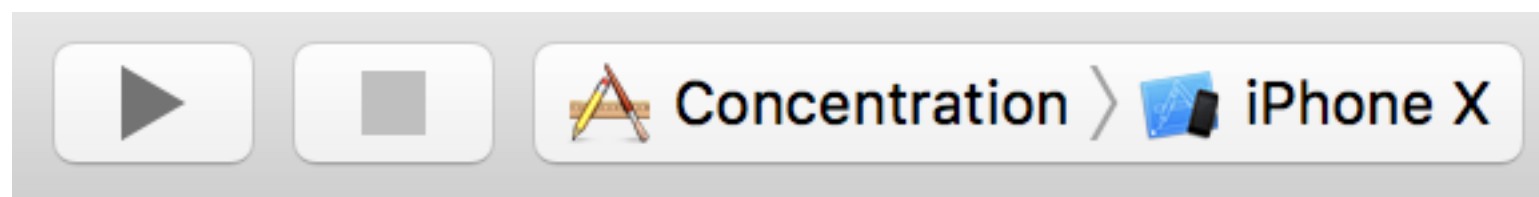


Once we have our project set up in Xcode, the **main.storyboard** file is where we design our UI. We are just going to drag objects from the **Object Library** onto this canvas and design our apps.

Object Library

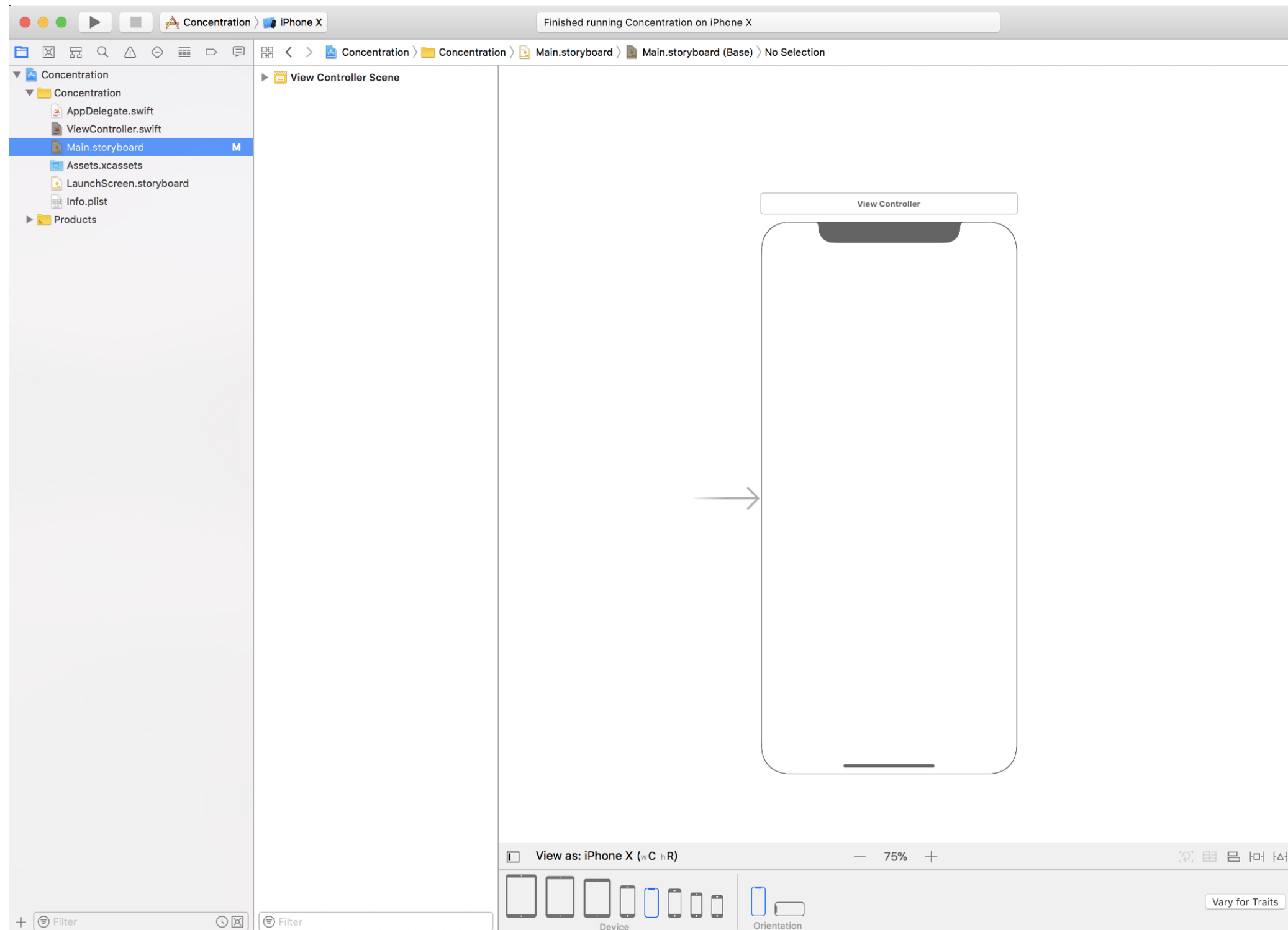


Step 1: We can choose which particular Apple device we want to design for. In this case, as can be seen at the bottom, select **View as: iPhone X**. We can also select which simulator we want to run our program on. Select **iPhone X**. (on the top of the screen next to the play button)



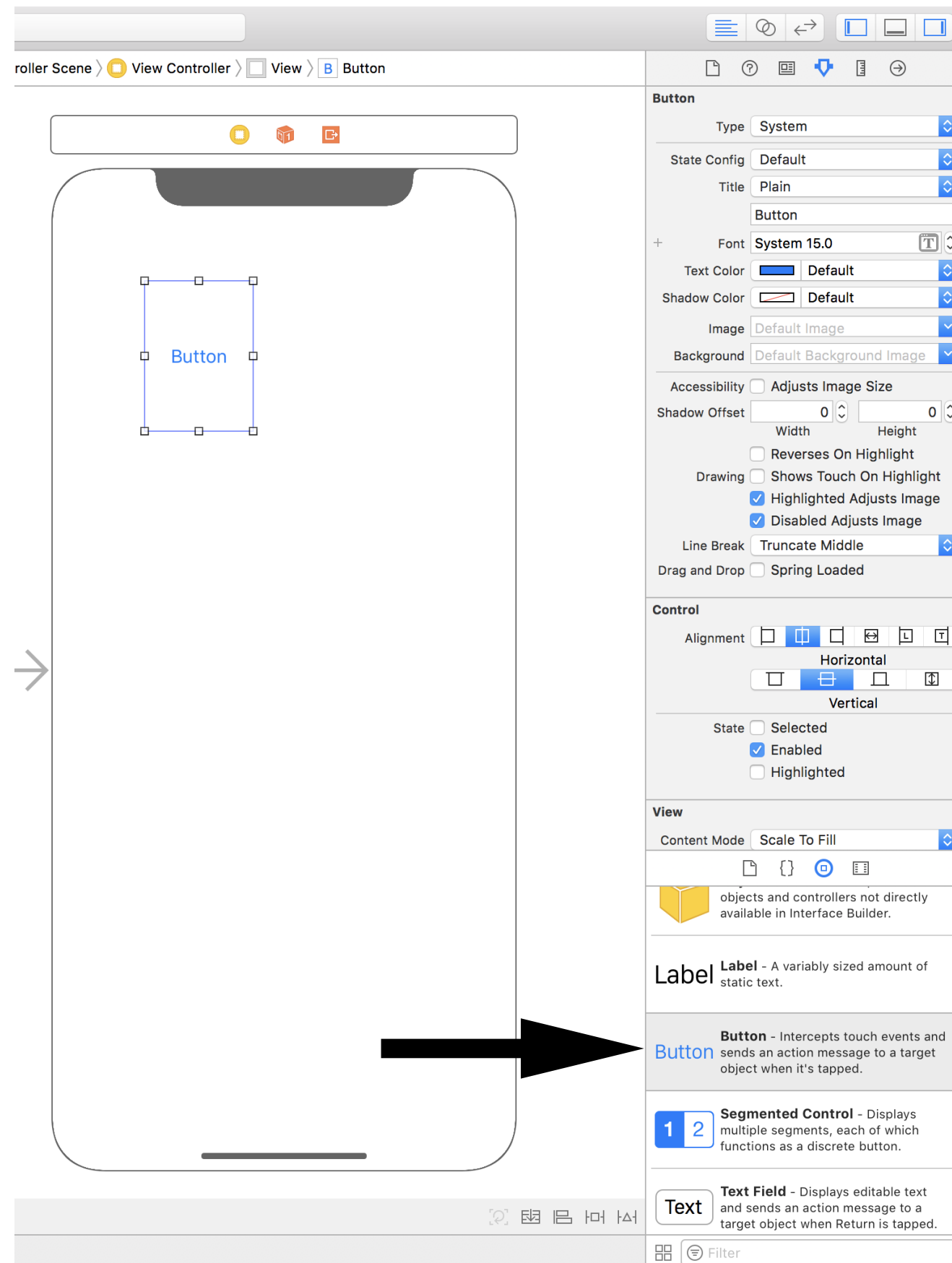


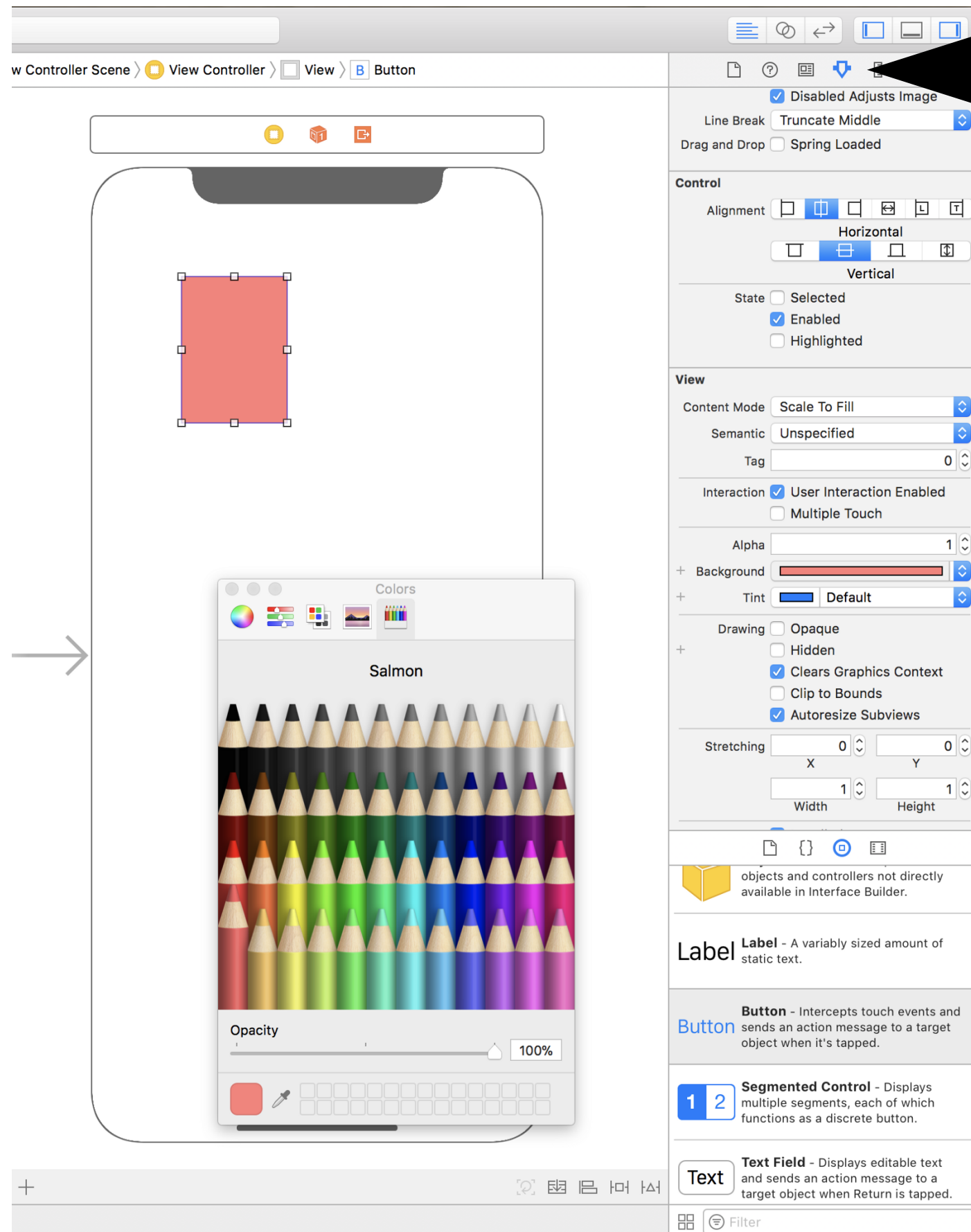
This is what your screen should look like right now



Step 2:

To find a button, we look in the *Utilities Tab*, in the **Object Library** for a button (search for it in the filter), and drag it onto our UI. The Object Library contains tons of UI Elements, some of which are very sophisticated. There are sliders, textfields, labels, map kits, augmented reality kits, etc.





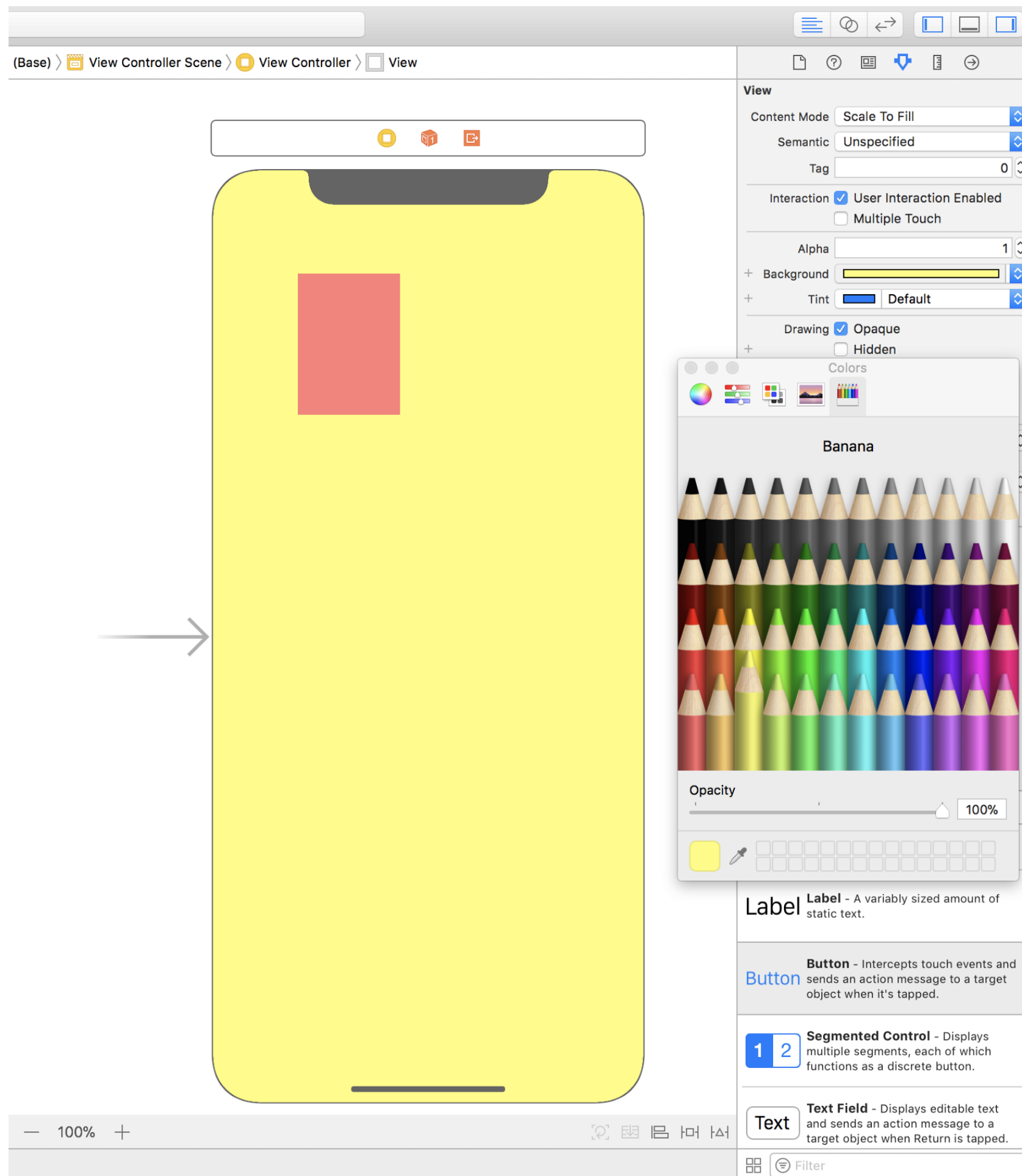
Attributes Inspector

Notice that as we added the button, we can use the controls surrounding it to change its size.

Furthermore, once we add the button and select it, we see a lot of options in the **attributes inspector** to edit and change the button as we see fit.

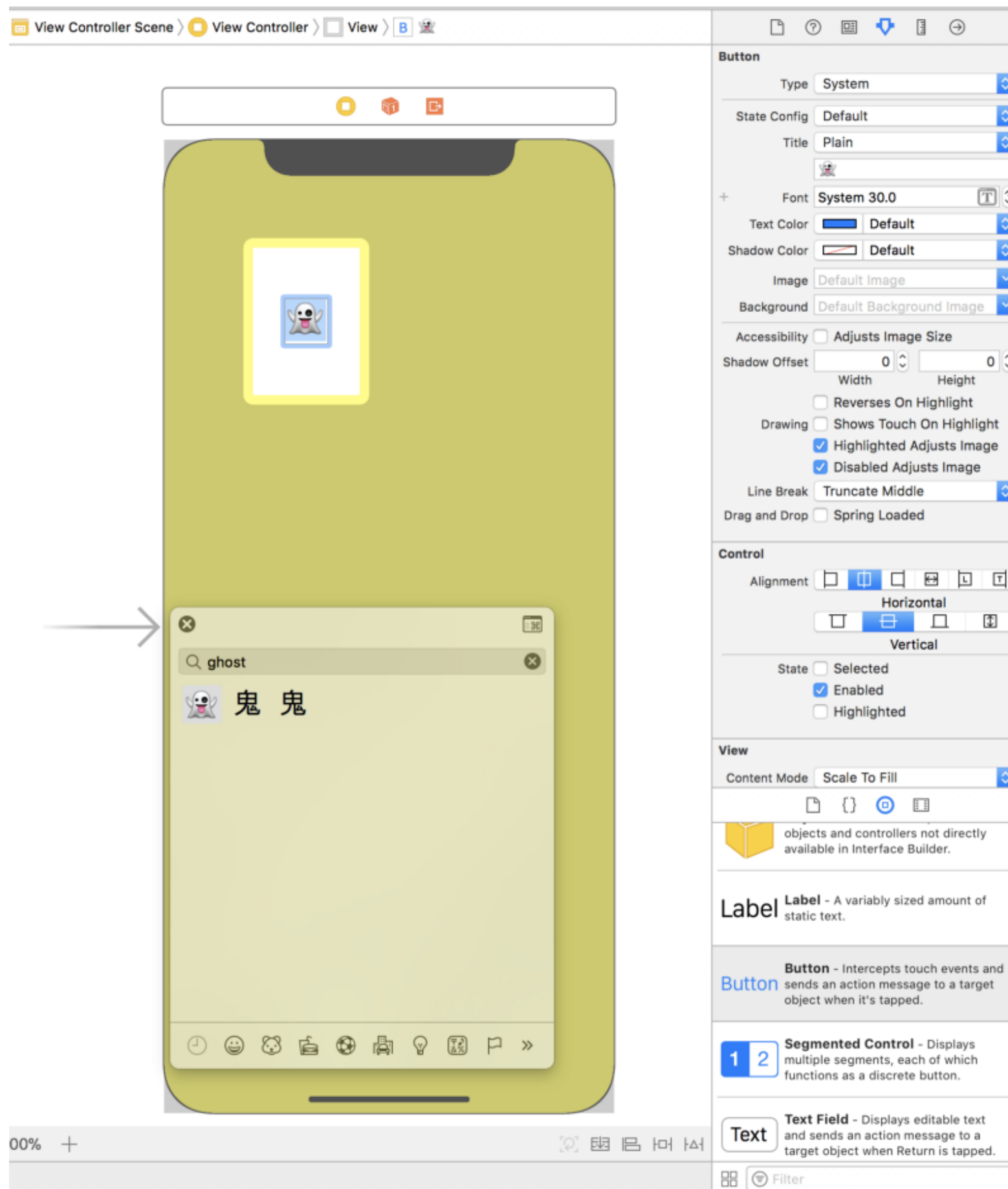
Step 3:

Delete the word “button” and change the button’s background colour to anything you like.



Step 4:

Click on the white background, and change the background colour to anything you like.



Step 5:

Change the background colour of the button back to white, and add a ghost 👻 emoji.

(control+command
+space to enter emoji
keyboard
Alt: Edit -> Emoji &
Symbols)

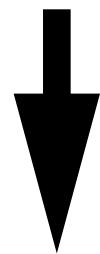
Note that the original size of the emoji is too small for a user to see without straining their eyes, so we can change the font-size to 30, or even 50 in the attributes inspector.



The behaviour of our UI is written in code. Currently we will write that in the class ViewController. Open the file named ViewController.swift from the left panel.

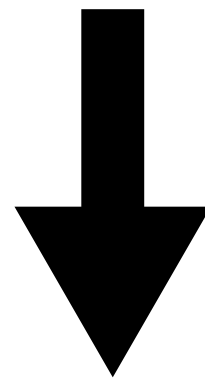
When we select **ViewController.swift**, it comes preloaded with some methods. Delete these for now.

```
1
2 import UIKit
3
4 class ViewController: UIViewController {
5
6     override func viewDidLoad() {
7         super.viewDidLoad()
8         // Do any additional setup after loading the view, typically from a nib.
9     }
10
11    override func didReceiveMemoryWarning() {
12        super.didReceiveMemoryWarning()
13        // Dispose of any resources that can be recreated.
14    }
15
16
17 }
18
```



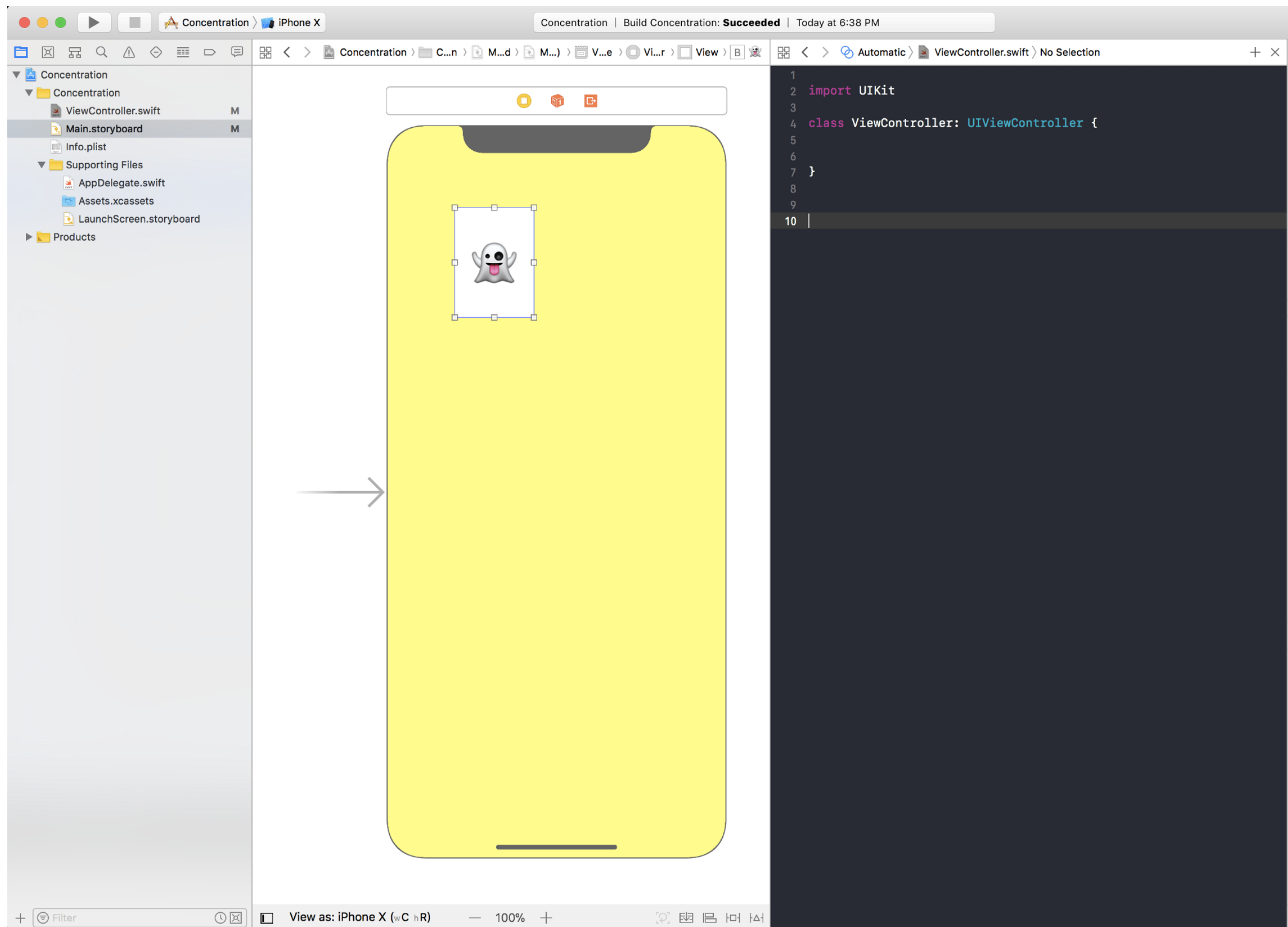
```
1
2 import UIKit
3
4 class ViewController: UIViewController {
5
6
7 }
```


So now what we want, is that when we click on the button, it should invoke a method in our class that flips the button. To do this we need both our code and the main storyboard file on the screen at the same time. You get this by clicking the **assistant editor**.

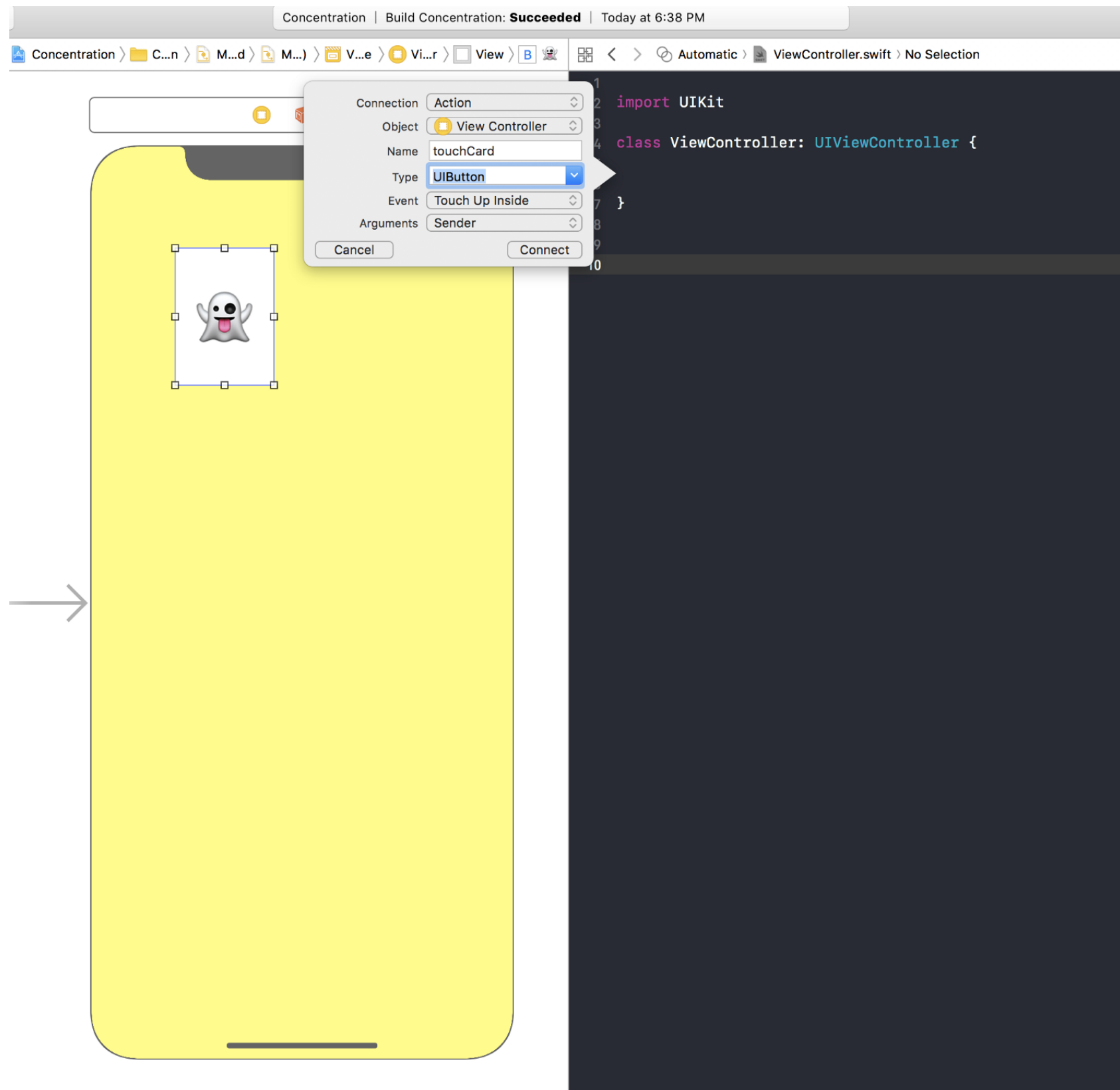




This is what you should see right now



Step 6: Press control and drag from the button into our code.

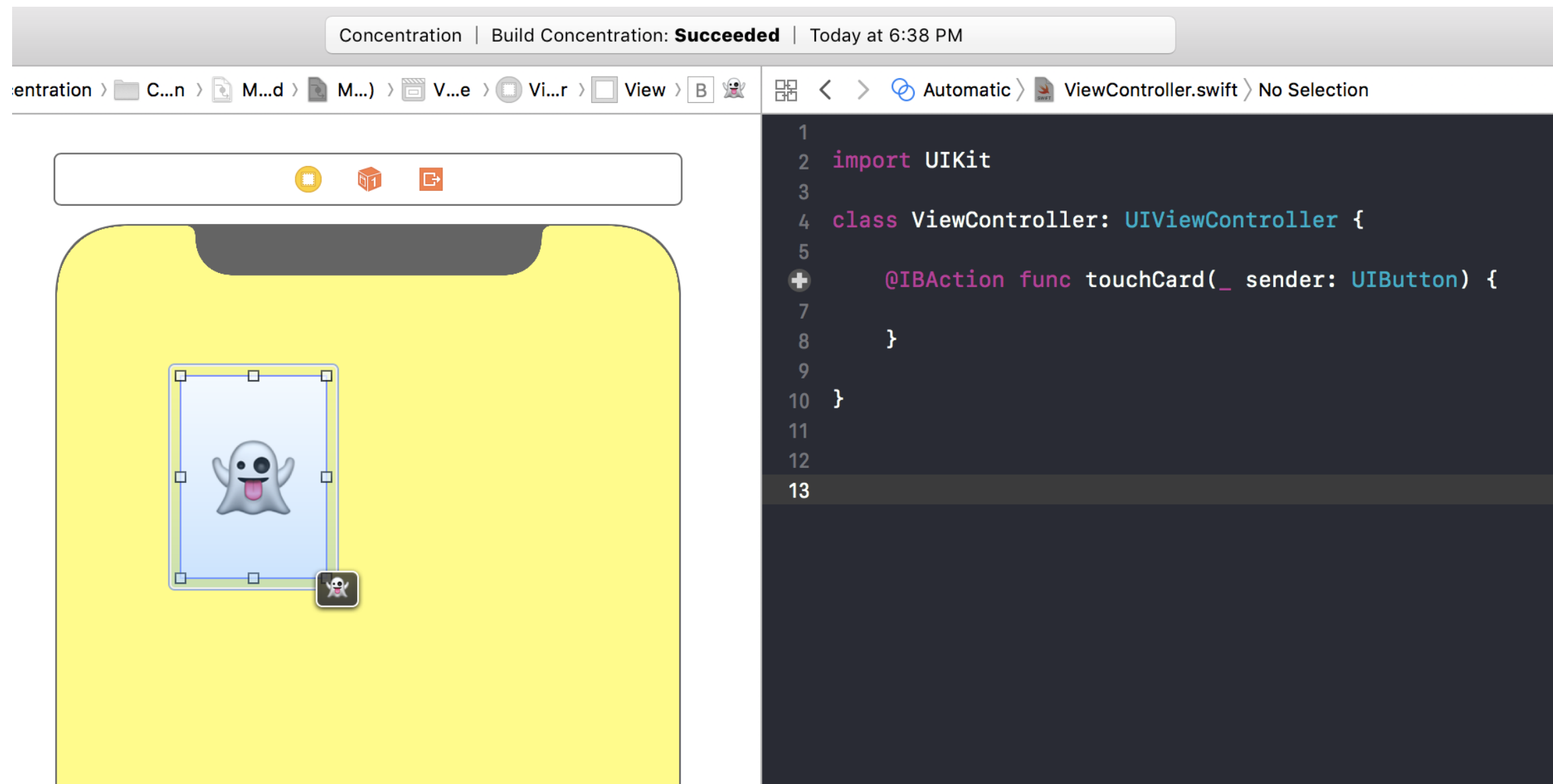




Once you make the connection, you have some options (Outlet, Outlet Collection, or Action) to indicate what kind of connection you want to make between the user interface element and the *Controller*. Choose Action because it is used for allowing something that happens in the UI (some action) to invoke a method in the *Controller*.

When the button is touched we will call some action method to be performed. Call this method, **touchCard**. We could have no arguments, but in this case we want to know which button, or card was touched so that we can flip it over. Therefore, we keep *sender*, which is the button as the argument.

Notice that it fills a little dot in the gutter (where the line numbers are) such that when we mouse over that dot we see which of our interface elements is invoking that method.
(Move your mouse over the little dot to see)





Step 7:

Fill the touchCard method. Also, define the flipCard method as shown below



```
1
2 import UIKit
3
4 class ViewController: UIViewController {
5
6     @IBAction func touchCard(_ sender: UIButton) {
7         flipCard(withEmoji: "👻", on: sender)
8     }
9
10    func flipCard(withEmoji emoji: String, on button: UIButton) {
11    }
12
13 }
```

We still need to complete our flipCard method. This basically involves toggling. We first check to see if the card has a ghost on it. If it does we flip the card to show a background colour with no text, indicating its back face. If it's not the ghost emoji then we just add the emoji with the white background to the button. How do we know which emoji the button has? **We need to be able to get the current title of the button.**



Step 8:

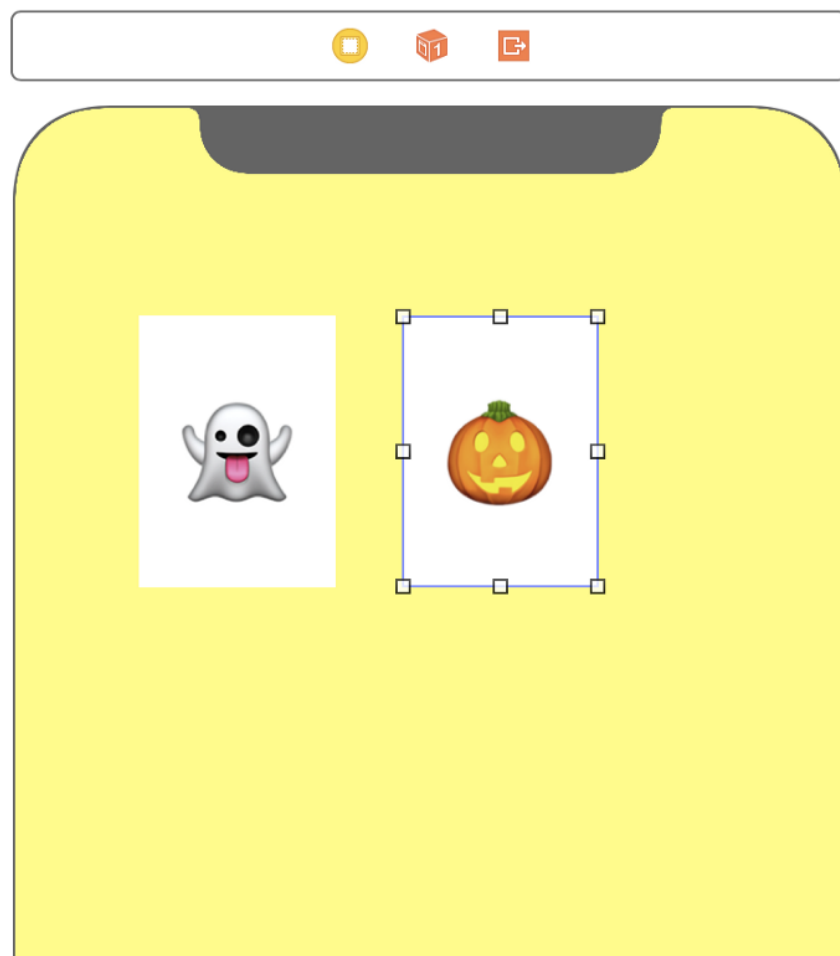
Fill the flipCard method. We get the colours, by typing “color literal”, which places a box that enables us to choose the colour.

```
func flipCard(withEmoji emoji: String, on button: UIButton) {  
    if button.currentTitle == emoji {  
        button.setTitle("", for: UIControlState.normal)  
        button.backgroundColor =   
    } else {  
        button.setTitle(emoji, for: UIControlState.normal)  
        button.backgroundColor =   
    }  
}
```

Step 9:

Challenge

Drag another button and repeat the same process. This time use the 🎃 emoji. Create an action for this button and fill the code as follows.

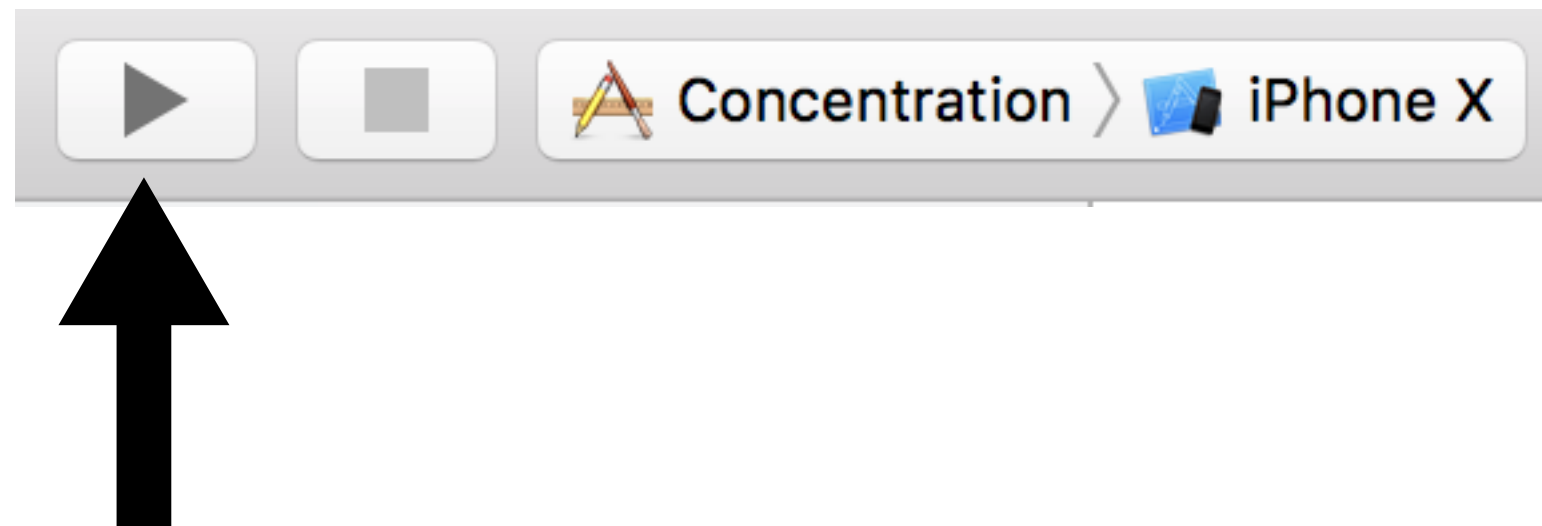


```

1  import UIKit
2
3
4  class ViewController: UIViewController {
5
6      @IBAction func touchCard(_ sender: UIButton) {
7          flipCard(withEmoji: "👻", on: sender)
8      }
9
10     @IBAction func touchSecondCard(_ sender: UIButton) {
11         flipCard(withEmoji: "🎃", on: sender)
12     }
13
14     func flipCard(withEmoji emoji: String, on button: UIButton) {
15         if button.currentTitle == emoji {
16             button.setTitle("", for: UIControlState.normal)
17             button.backgroundColor = #f00
18         } else {
19             button.setTitle(emoji, for: UIControlState.normal)
20             button.backgroundColor = #fff
21         }
22     }
23
24 }

```


Step 10: Build and Run!



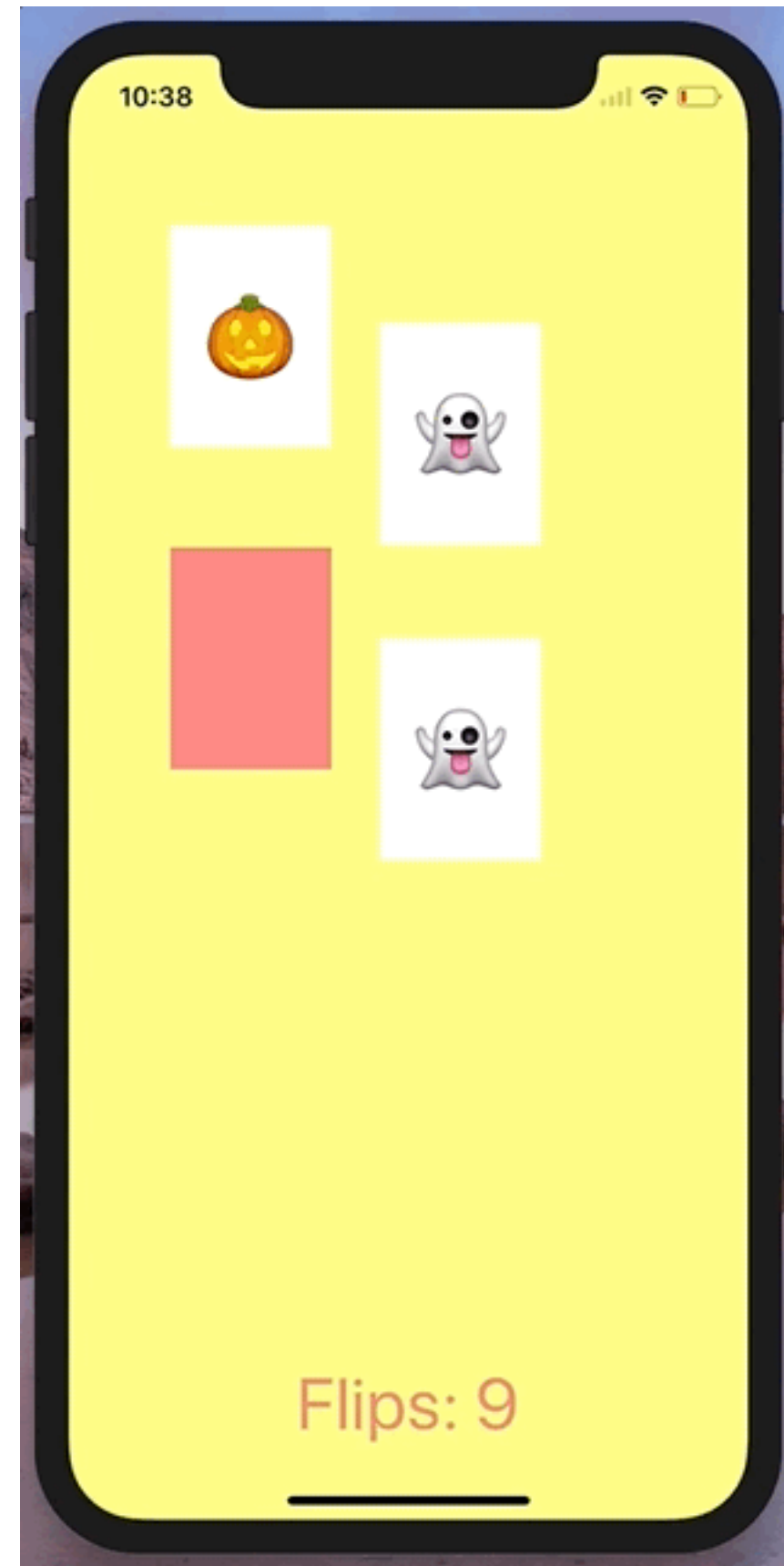
Challenge:

Now add multiple cards such that each card has a pair and all the cards are shuffled.

And there you have it, your very first iOS app!

Additionally, you may try to add a counter for the number of flips.

(Solution on the next slide)





```
2 import UIKit
3
4 class ViewController: UIViewController
5 {
6     var flipCount = 0 {
7         didSet {
8             flipCountLabel.text = "Flips: \(flipCount)"
9         }
10    }
11
12    var emojiChoices = ["🍁", "👻", "🍁", "👻"]
13
14    @IBOutlet weak var flipCountLabel: UILabel!
15
16    @IBOutlet var cardButtons: [UIButton]!
17
18    @IBAction func touchCard(_ sender: UIButton) {
19        flipCount += 1
20        if let cardNumber = cardButtons.index(of: sender) {
21            flipCard(withEmoji: emojiChoices[cardNumber], on: sender)
22        } else {
23            print("chosen card was not in cardButtons")
24        }
25    }
26
27    func flipCard(withEmoji emoji: String, on button: UIButton) {
28        if button.currentTitle == emoji {
29            button.setTitle("", for: UIControlState.normal)
30            button.backgroundColor = #f08080
31        } else {
32            button.setTitle(emoji, for: UIControlState.normal)
33            button.backgroundColor = #ffffff
34        }
35    }
36
37 }
38
```