



STANFORD UNIVERSITY
School of Engineering

Developing Apps for iOS CS193P Fall 2017:

Разработка iOS 11 приложений с помощью Swift

Пятничная Лекция 1: Debugging, Xcode Tips and Tricks
(Отладка, полезные советы и приемы работы с Xcode).

September 29, 2017

Профессор Пол Хэгертி (Paul Hegarty)
[Независимая, неавторизованная транскрипция ©
2017 Paul Hegarty]

[НАЧАЛО ЛЕКЦИИ](#)

----- 5 -ая минута лекции -----
----- 10 -ая минута лекции -----
----- 15 -ая минута лекции -----
----- 20 -ая минута лекции -----
----- 25 -ая минута лекции -----
----- 30 -ая минута лекции -----
----- 33 -ая минута лекции -----

Ладушки, ну-с, привет всем.

Меня зовут Джейсон. Я один из двух ваших младших преподавателей. Второй - это Джинни.

Сегодня её нет, а у нас на сегодня две темы.

Первая - ураганный тур по **Xcode**, просто чтобы здорово облегчить Вам жизнь на ближайшую четверть, на всю оставшуюся жизнь - познакомить Вас получше с этой средой разработки.

Вторая - это использование отладчика вообще, и **Xcode** в частности.

Занятие будет значительно короче большинства других, просто потому, что не так уж и много материала нужно усвоить, так что освободимся мы рановато. И к тому же нет ни слайдов, ничего в этом роде, ни лекции как таковой.

Я просто с места - в карьер приступаю к демонстрационному примеру. И не стесняйтесь притормозить меня в любой момент, просто крикните, поднимите руку, если что-то не ясно, если вы хотите узнать поподробнее о чем-то, о чём угодно. И я остановлюсь и постараюсь ответить.

Я понимаю, нам предстоит продираться через гигантский список комбинаций клавиш (**key commands**) - «ускорителей» (**shortcuts**). Это совсем не так, как на некоторых увлекательных лекциях, поэтому я постараюсь быть настолько «увлекательным», насколько возможно.

Но это на самом деле супер полезно.

Так что, если бы пришлось выбрать что-то одно из всей нашей беседы - вот оно: если вы хотите наловчиться использовать **Xcode** и бегло с ним обращаться, вам действительно придется запомнить множество комбинаций клавиш (**key commands**). Это вам очень поможет.

Вы уже вроде как познакомились с **Xcode** на лекциях Пола, но, собственно, изучение всех этих комбинаций клавиш (**key commands**) - «ускорителей» (**shortcuts**) сделает вашу жизнь намного проще. И я надеюсь, что смогу убедить вас сегодня, что этот материал очень даже интуитивно понятен. Это вовсе не похоже на запоминание кучи бессистемных штуковин.

Напротив, **Apple** всё разложила по полочкам, очень аккуратно и упорядоченно. Надеюсь, эту мысль мне удастся до вас донести тоже.

Итак, давайте приступим.

Я вызову **Xcode**, в котором открыт проект **Concentration** - в Среду на лекции все работало. Уловка в том, что эта версия **Concentration**, которая у меня тут - я счел её «грустной» **Концентрацией** (вот папка с проектом так и называется **SadConcentraion**), поскольку она вообще-то поломана несколькими способами.

You can see right here on this folder,

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10 class ViewController: UIViewController  
11 {  
12     var game: Concentration! {  
13         didSet {  
14             updateViewFromModel()  
15         }  
16     }  
17     var flipCount = 0 {  
18         didSet {  
19             flipCountLbl.text = "Flips: \(flipCount)"  
20         }  
21     }  
22     @IBOutlet weak var flipCountLbl: UILabel!  
23     @IBOutlet var cardButtons: [UIButton]!  
24     override func viewDidLoad() {  
25         super.viewDidLoad()  
26         game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)  
27     }  
28     var emojiChoices = ["\u{1f600}", "\u{1f601}", "\u{1f602}", "\u{1f603}", "\u{1f604}"]  
29     var emoji: String {  
30         if emoji[card.identifier] == nil {  
31             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
```

И вот это мы и будем расследовать, используя отладчик. Но прежде чем мы начнем, я отвлекусь и проведу тур на тему «**Xcode** с клавишами быстрого доступа и без».

На прошлой лекции вы видели, например, навигацию по всяkim панелям с помощью различных кнопок.

```
1 // ViewController.swift
2 // Concentration
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0
```

The screenshot shows the Xcode interface with the following details:

- Top Bar:** iPhone X, Concentration | Build Concentration Succeeded | Today at 3:31 AM
- Project Navigator:** Concentration > Concentration > ViewController.swift
- Editor:** The code editor displays `ViewController.swift` with the following content:

```
1 // ViewController.swift
2 // Concentration
3 //
4 // Created by CS193p Instructor on 9/23/17.
5 // Copyright © 2017 Stanford University. All rights reserved.
6 //
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLbl.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLbl: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices[randomIndex]
41     }
42     return emoji[card.identifier]!
43 }
```
- Utilities Pane:** Shows the following sections:
 - Identity & Type:** Name: ViewController.swift, Type: Default - Swift Source, Location: Relative to Group, Full Path: /Users/admin/Desktop/SadConcentration/Concentration/ViewController.swift
 - On Demand Resource Tags:** None
 - Target Membership:** Concentration (checkbox checked)
 - Text Settings:** Text Encoding: No Explicit Encoding, Line Endings: No Explicit Line Endings, Indent Using: Spaces, Widths: 4, Tab: 4, Wrap Lines: checkedA yellow arrow points from the "Click" label to the "Target Membership" section.
- Bottom Status Bar:** Shows "No Matches" and several small icons.

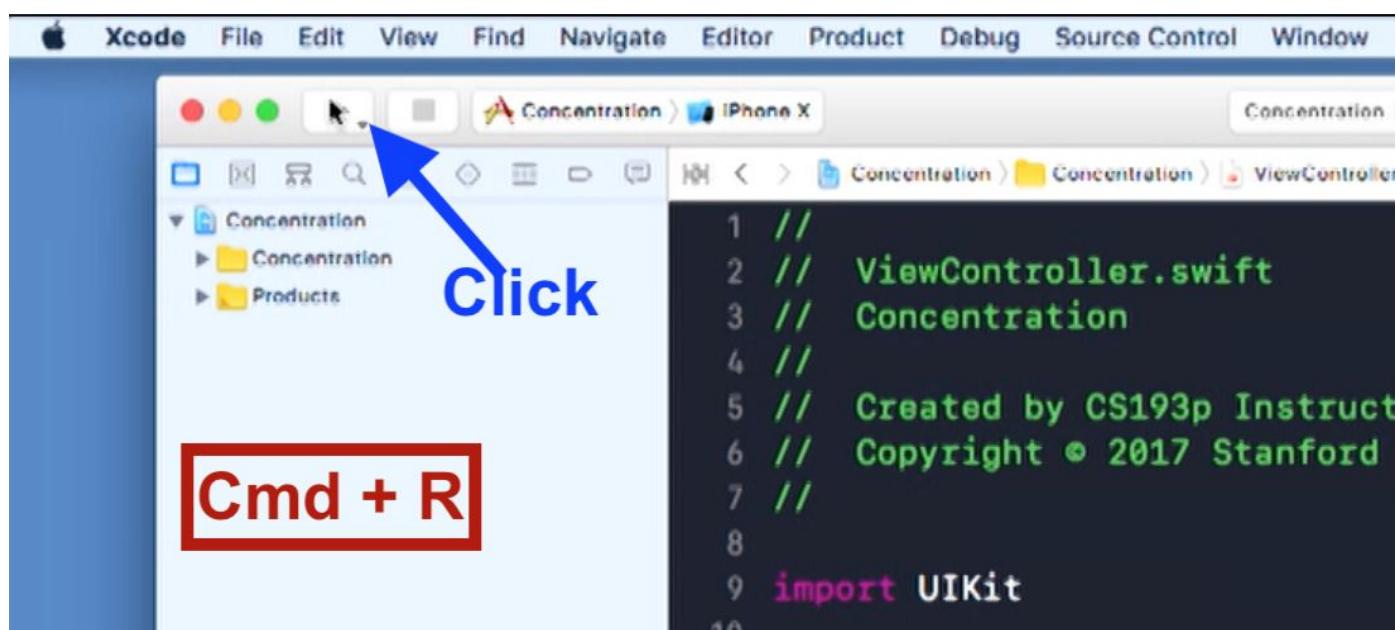
Но это всё быстро надоест при работе с кодом.

Вместо того чтобы усердно работать мышью, вы можете делать всё в пять раз быстрее, если поймете как всё устроено

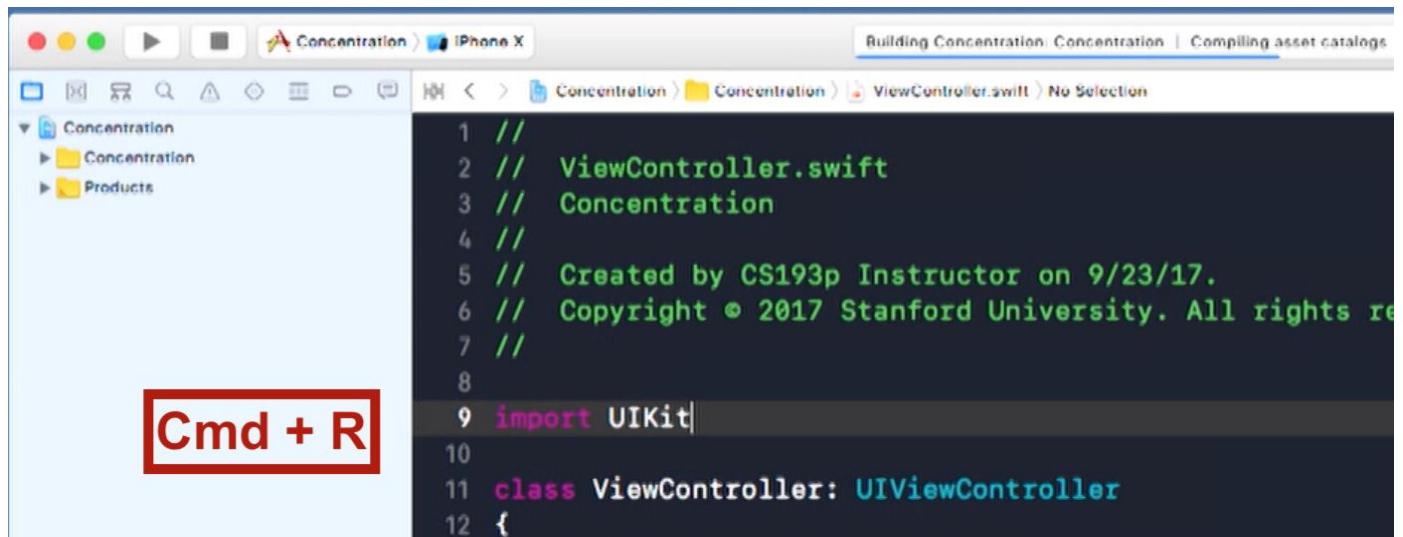
Первая вещь - начнем по порядку - по интерфейсу **Xcode** с верхнего левого угла.

Что тут можно ускорить?

Так для запуска и остановки используется сочетание клавиш **Cmd+R**.



Набираем **Cmd+R**, и приложение запускается:



```
Concentration > iPhone X
Building Concentration: Concentration | Compiling asset catalogs
Concentration > Concentration > ViewController.swift > No Selection
Concentration
Concentration
Products

1 /**
2 // ViewController.swift
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController
12 {
```

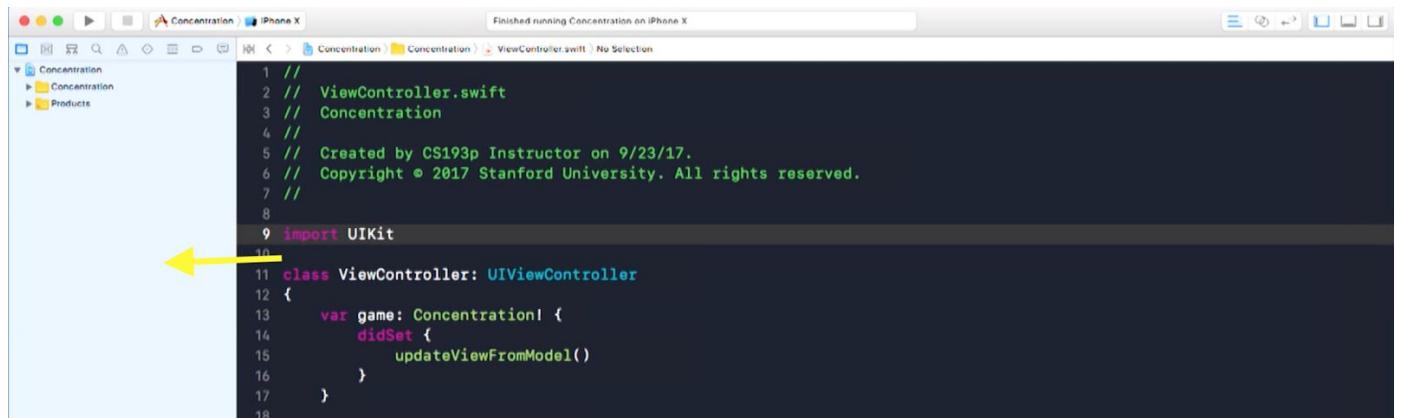
Еще раз набираем **Cmd+R**, и приложение останавливается:



```
Concentration > iPhone X
Finished running Concentration on iPhone X
Concentration > Concentration > ViewController.swift > No Selection
Concentration
Concentration
Products

1 /**
2 // ViewController.swift
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController
12 {
```

Дальше поговорим о более важном аспекте - собственно о навигации по различным областям **Xcode**. Можно отображать и скрывать по мере необходимости все эти области или отдельные панели внутри них клавишами быстрого доступа.



```
Concentration > iPhone X
Finished running Concentration on iPhone X
Concentration > Concentration > ViewController.swift > No Selection
Concentration
Concentration
Products

1 /**
2 // ViewController.swift
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18 }
```

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }
```

Identity and Type

Name: ViewController.swift
Type: Default - Swift Source
Location: Relative to Group
ViewController.swift
Full Path: /Users/admin/Desktop/GitConcentration/Concentration/ViewController.swift

On Demand Resource Tags

Target Membership

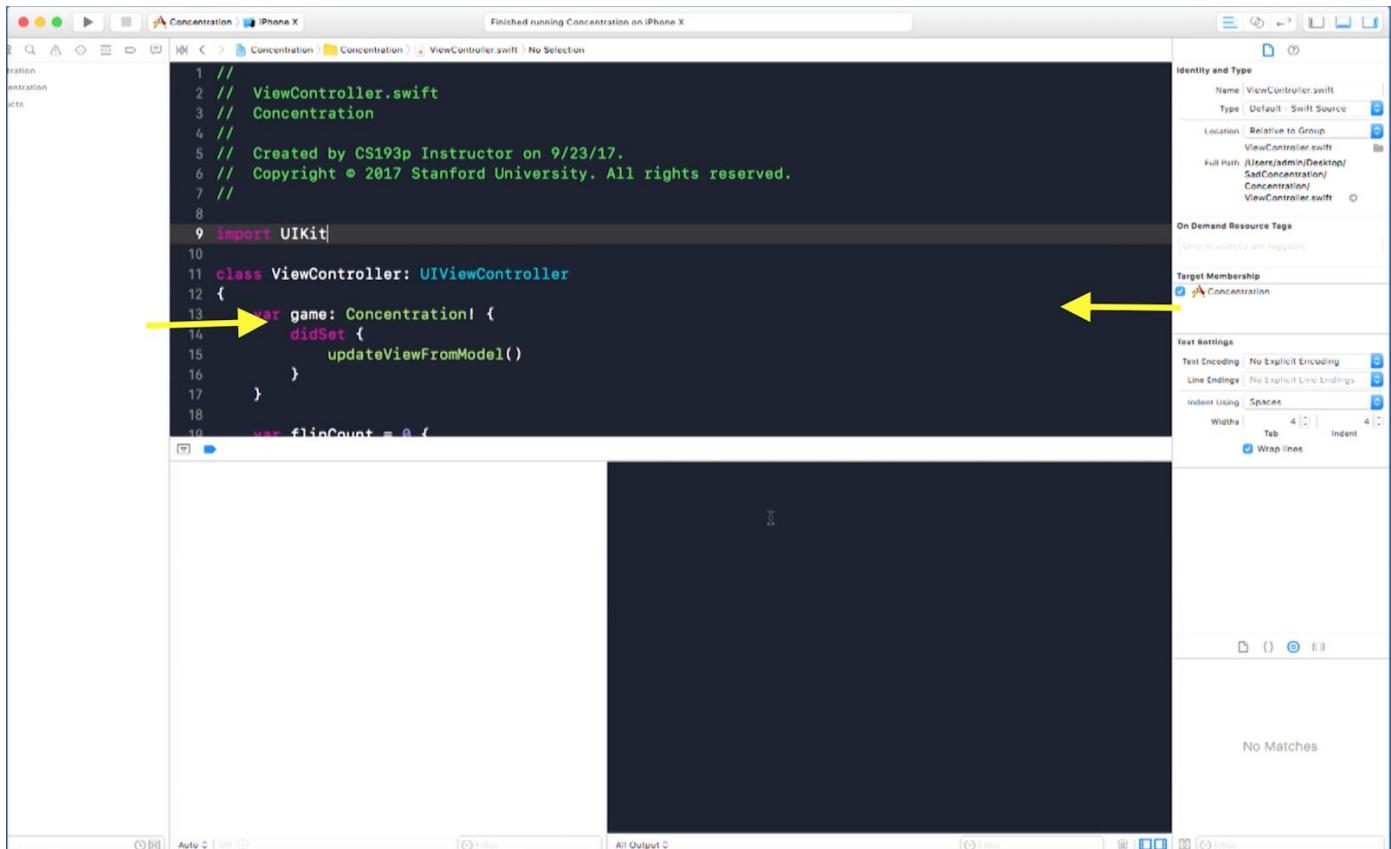
Concentration

Text Settings

Text Encoding: No Explicit Encoding
Line Endings: No Explicit Line Endings
Indent Using: Spaces
Widths: 4 | 4

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit|  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration! {  
14         didSet {  
15             updateViewFromModel()  
16         }  
17     }  
18  
19     var flipCount = 0 {  
20         didSet {  
21             flipCountLbl.text = "Flips: \(flipCount)"  
22         }  
23     }  
24  
25     @IBOutlet weak var flipCountLbl: UILabel!  
26  
27     @IBOutlet var cardButtons: [UIButton]!  
28  
29     override func viewDidLoad() {  
30         super.viewDidLoad()  
31     }  
32 }
```

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit|  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration! {  
14         didSet {  
15             updateViewFromModel()  
16         }  
17     }  
18  
19     var flipCount = 0 {  
20         didSet {  
21             flipCountLbl.text = "Flips: \(flipCount)"  
22         }  
23     }  
24  
25     @IBOutlet weak var flipCountLbl: UILabel!  
26  
27     @IBOutlet var cardButtons: [UIButton]!  
28  
29     override func viewDidLoad() {  
30         super.viewDidLoad()  
31     }  
32 }
```



```
1 // View Controller.swift
2 // Concentration
3 // Copyright © 2017 Stanford University. All rights reserved.
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLbl.text = "Flips: \(flipCount)"
22     }
23 }
```

Identity and Type

Name: ViewController.swift
Type: Default - Swift Source
Location: Relative to Group
Full Path: /Users/admin/Desktop/SadConcentration/Concentration/ViewController.swift

On Demand Resource Tags

Utility ViewControllers are Segueable

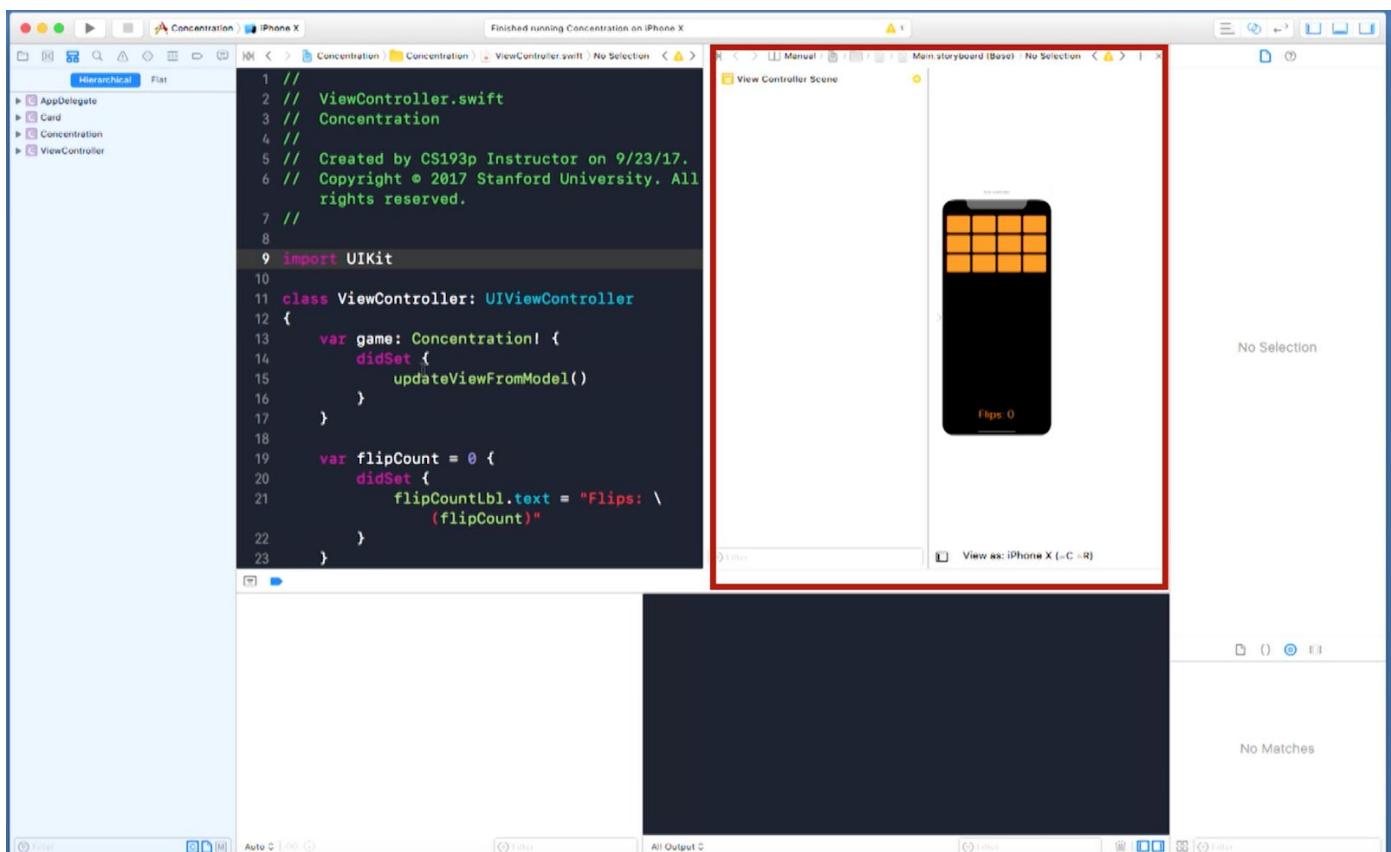
Target Membership

Concentration

Text Settings

Text Encoding: No Explicit Encoding
Line Endings: No Explicit Line Endings
Indent Using: Spaces
Width: Tab 4 Indent 4
Wrap Lines

Это вас конкретно ускоряет. Это Вспомогательный Редактор (**Assistant Editor**) открывается.:



Давайте-ка я быстренько объясню, как всё это работает. Я закрываю всё, что только что наоткрывал. И остается главный экран (**main screen**) во весь экран.

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!  
28  
29 override func viewDidLoad() {  
30     super.viewDidLoad()  
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)  
32 }  
33  
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]  
35 var emoji = Dictionary<Int, String>()  
36  
37 func emoji(for card: Card) -> String {  
38     if emoji[card.identifier] == nil {  
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
```

Вокруг главного экрана слева, справа и снизу могут отображаться дополнительные области.

В **Xcode** все расположено очень логично.

Самая левая область, панель навигатора **Navigator**, отображается с помощью комбинации клавиш

Cmd+0 ...

Cmd+0

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!  
28  
29 override func viewDidLoad() {  
30     super.viewDidLoad()  
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)  
32 }  
33  
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]  
35 var emoji = Dictionary<Int, String>()  
36  
37 func emoji(for card: Card) -> String {  
38     if emoji[card.identifier] == nil {  
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
```

... скрывается с помощью комбинации клавиш **Cmd+0**:

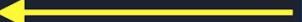


```
Concentration -> Concentration -> ViewController.swift - No Selection
```

```
1 // ViewController.swift
2 // Concentration
3 //
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLbl.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLbl: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40     }
41 }
```

Cmd+0

А самая правая область, панель Utilities, управляется комбинацией клавиш **Cmd+option+0**:



```
Concentration -> Concentration -> ViewController.swift - No Selection
```

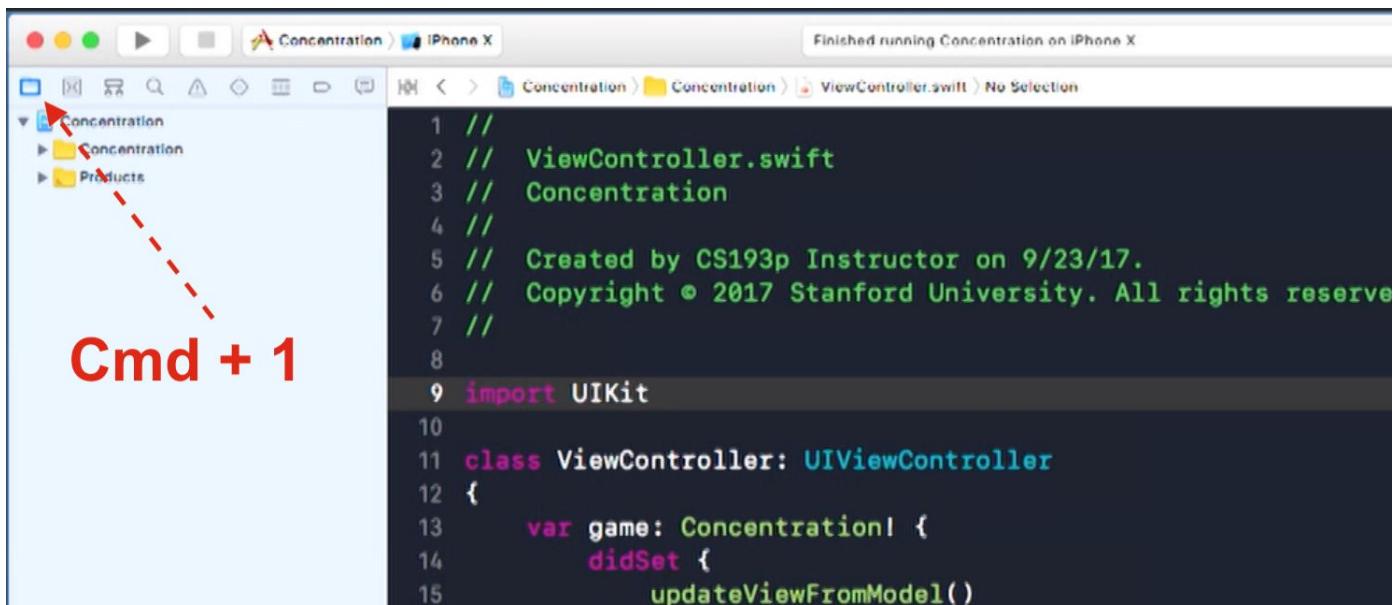
```
1 // ViewController.swift
2 // Concentration
3 //
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLbl.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLbl: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40     }
41 }
```

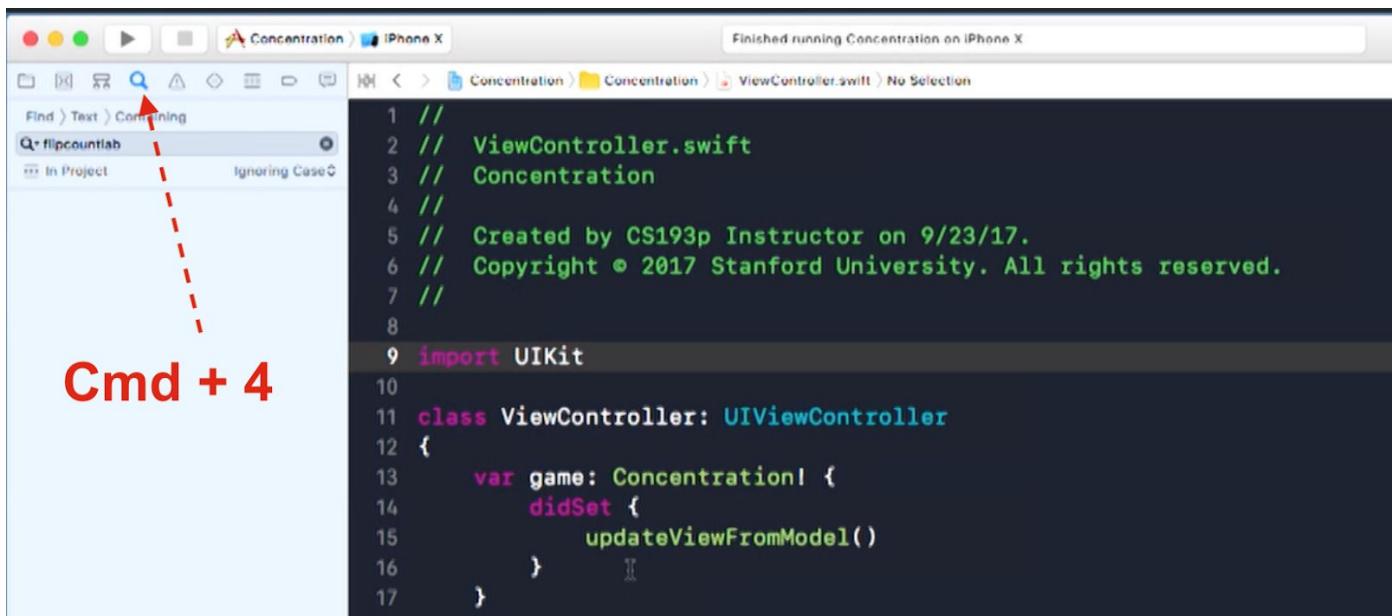
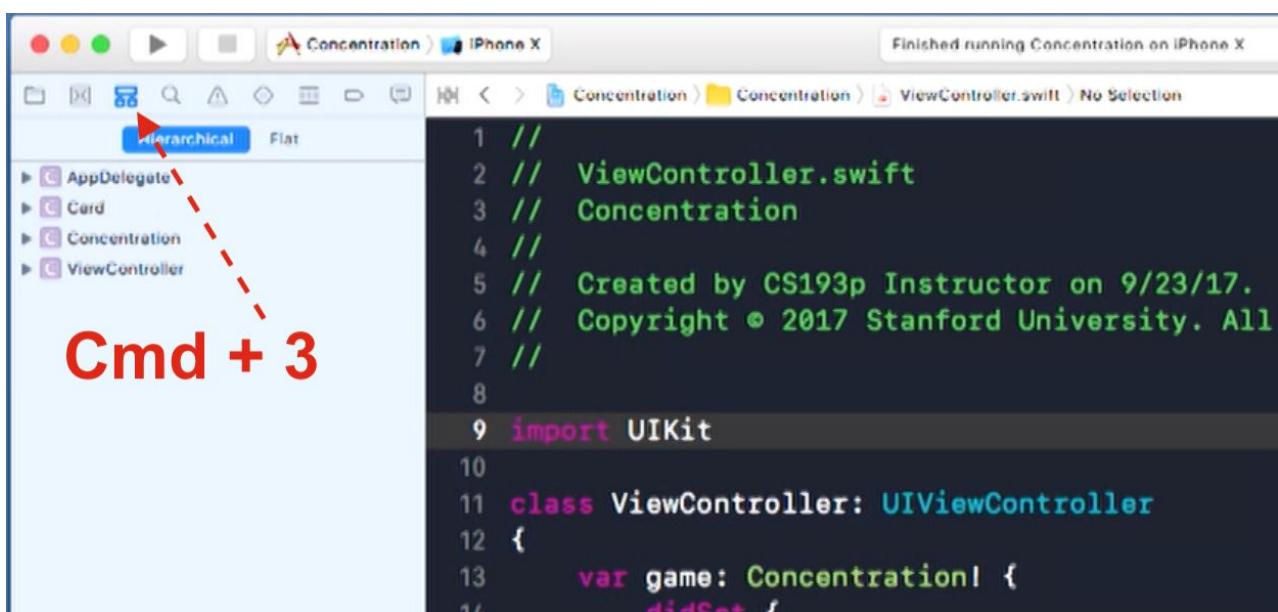
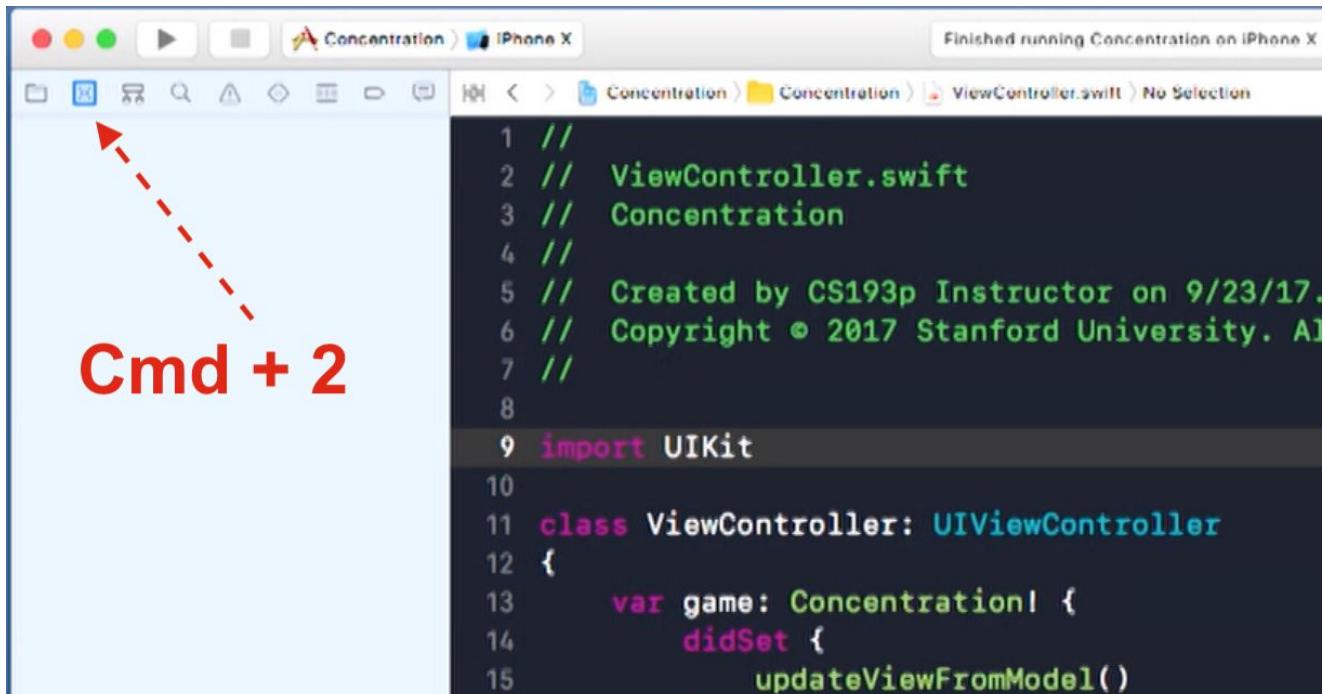
Cmd + option + 0

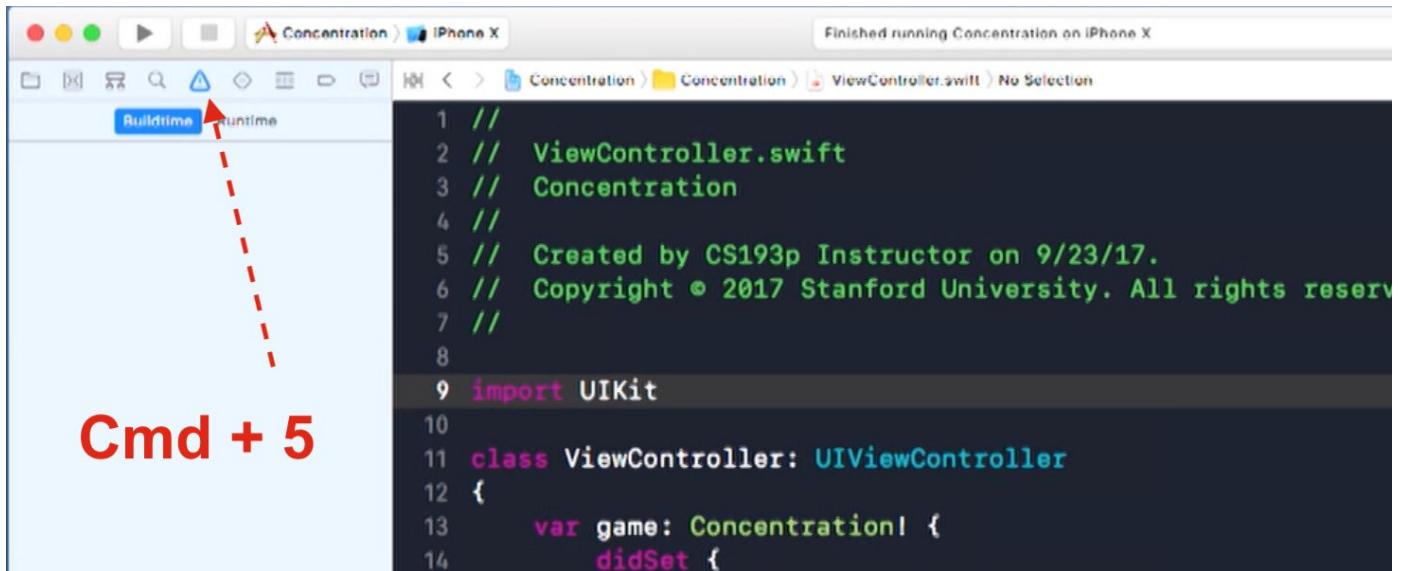
Cmd + option + 0

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 // Created by CS193p Instructor on 9/23/17.  
6 // Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration! {  
14         didSet {  
15             updateViewFromModel()  
16         }  
17     }  
18       
19     var flipCount = 0 {  
20         didSet {  
21             flipCountLbl.text = "Flips: \(flipCount)"  
22         }  
23     }  
24       
25     @IBOutlet weak var flipCountLbl: UILabel!  
26       
27     @IBOutlet var cardButtons: [UIButton]!  
28       
29     override func viewDidLoad() {  
30         super.viewDidLoad()  
31         game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)  
32     }  
33       
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]  
35     var emoji = Dictionary<Int, String>()  
36       
37     func emoji(for card: Card) -> String {  
38         if emoji[card.identifier] == nil {  
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
```

И внутри каждой области, я сейчас нахожусь в самой левой области на панели навигатора **Navigator**, нажатие комбинаций клавиш **Cmd+1,2,3,4,5** будет показывать различные вкладки (**tabs**) на панели навигатора **Navigator**.





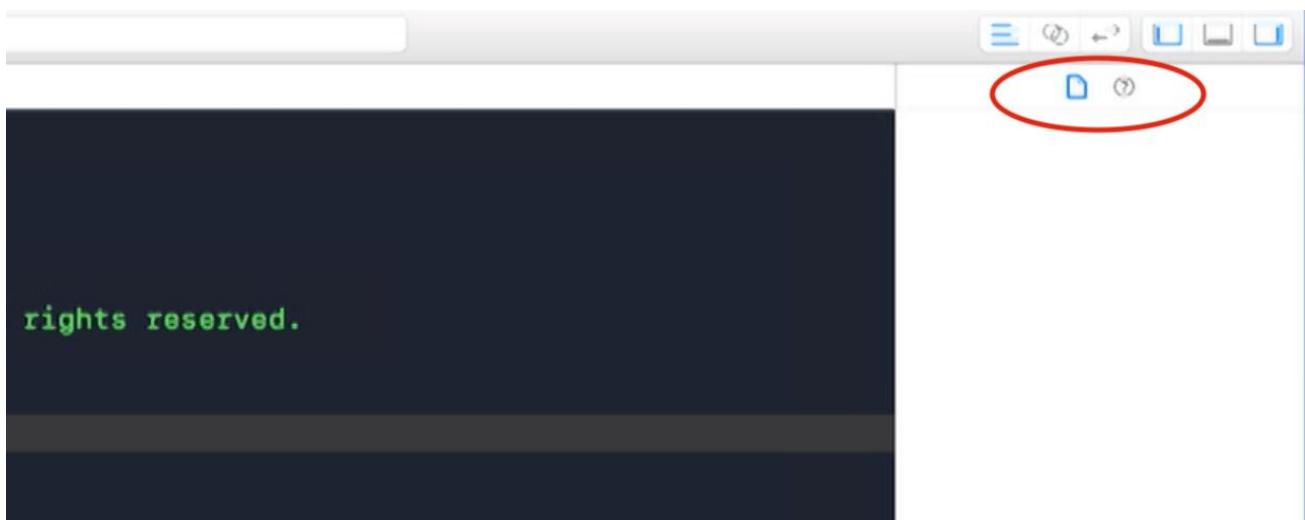


```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193P Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {
```

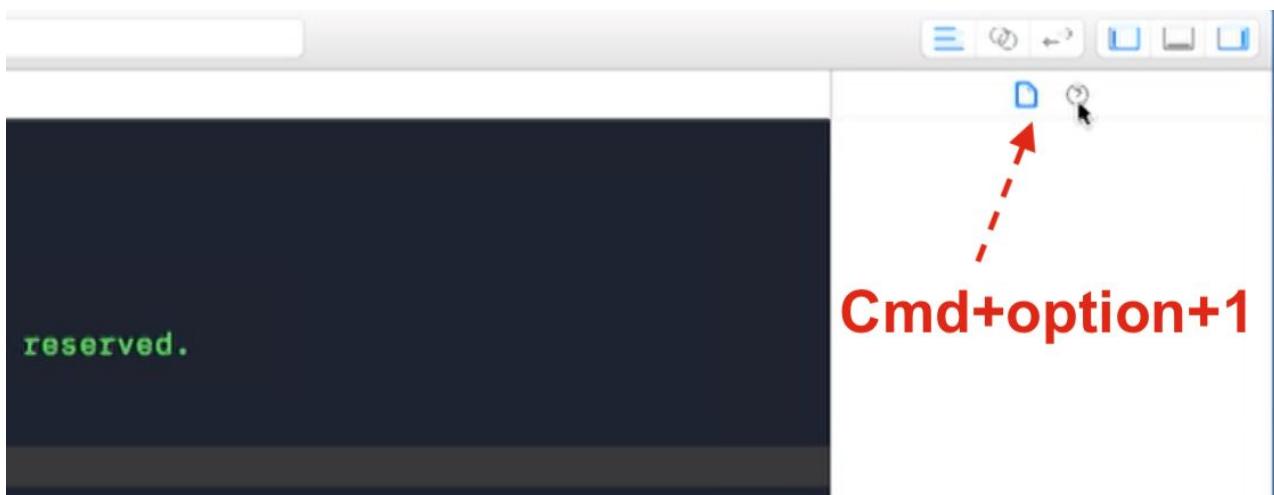
Видите как активируются по очереди вкладки вверху слева?.

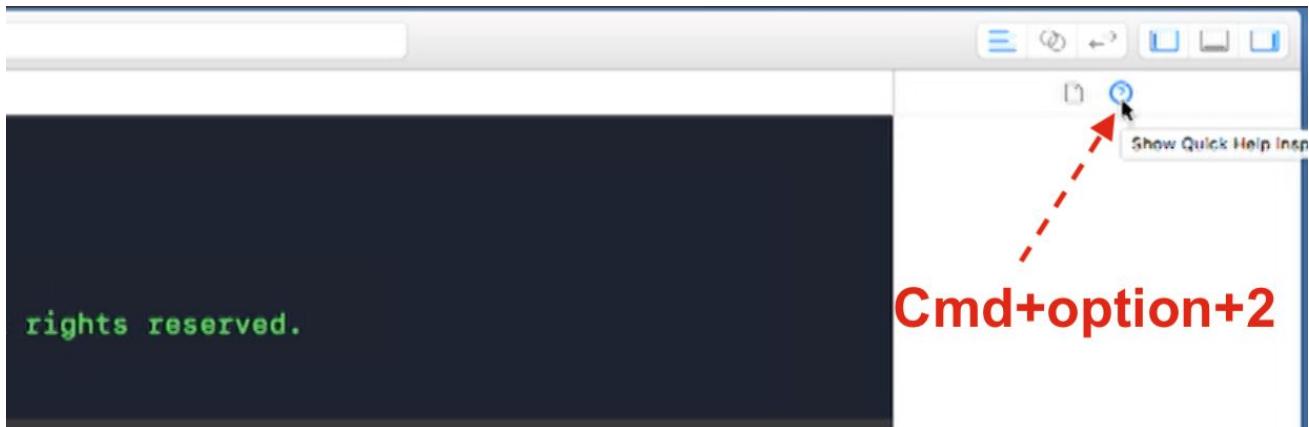
А теперь отобразим область утилит Utilities справа с помощью комбинации клавиш **Cmd+option+0**.

Взгляните - здесь такие же вкладки, как и в левой верхней панели.



Логично, что я могу пройтись по ним при помощи **Cmd + option +1,2** (раз уж левая и правая панели различаются только **option**).





Всё, что требуется запомнить - это что **Cmd+0** отображает и скрывает левую область, а остальные цифры соответствуют номерам вкладок. А если добавляется **option**, осуществляется такое же управление правой областью. Просто.

Нижняя область, содержащая панели консоли и отладчика, переключается сочетанием клавиш **Cmd+Shift+Y**. С ними ещё сегодня поработаем.

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!  
28  
29 override func viewDidLoad() {  
30     super.viewDidLoad()  
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)  
32 }  
33  
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]  
35 var emoji = Dictionary<Int, String>()  
36  
37 func emoji(for card: Card) -> String {  
38     if emoji[card.identifier] == nil {  
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))  
40         emoji[card.identifier] = emojiChoices[randomIndex]  
41     }  
42     return emoji[card.identifier]!
```

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!
```

No Tests
Click the + button to add test targets

No Selection

Cmd + shift + Y

No Matches

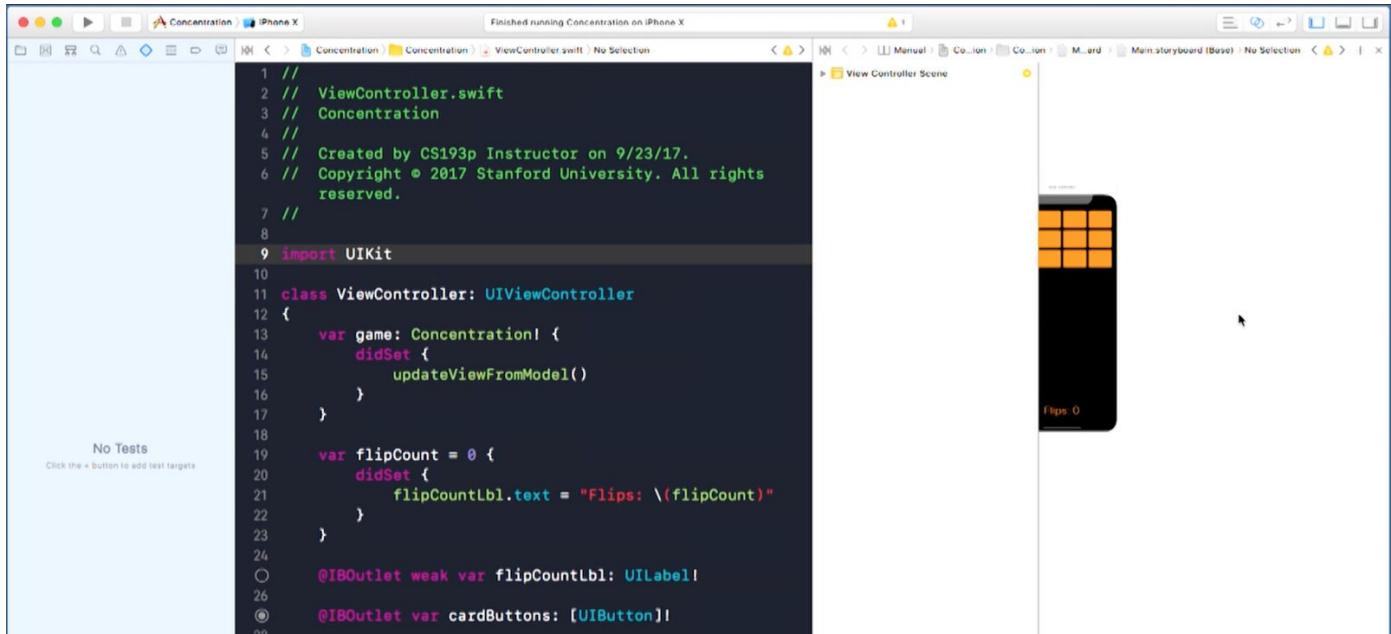
Одних только этих команд достаточно, чтобы управляться с окошками, особенно если вы программируете на ноутбуке, где экранное пространство ограничено. Здорово владеть этими сочетаниями клавиш, чтобы быстро ориентироваться в проекте.

Далее я хочу обратить ваше внимание, на то, что в общем-то вам и так известно из лекций: у **Xcode** есть два редактора. Главный - вот он - в центре и вспомогательный (**Assistant Editor**), который вызывается комбинацией клавиш **Cmd+option+Enter** и размещается справа в отдельном окне:

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!
```

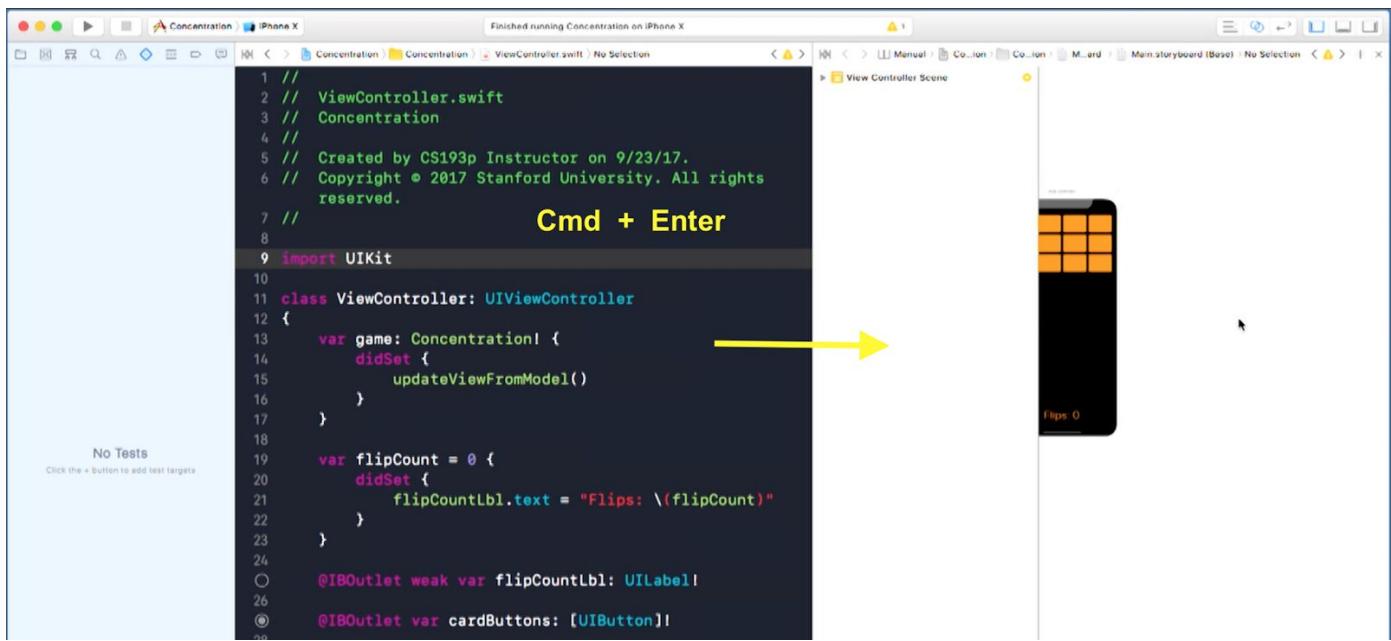
No Tests
Click the + button to add test targets

Cmd + option + Enter



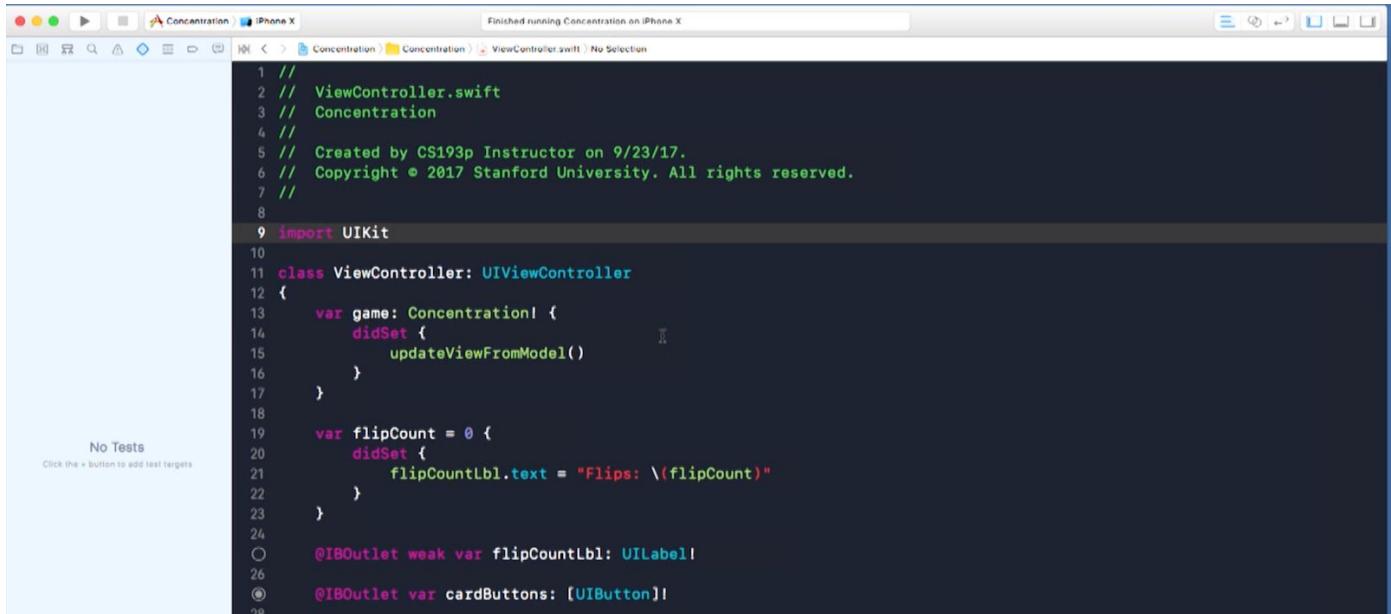
```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights  
    reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!
```

А вернуться в режим одного (главного) редактора можно клавишами **Cmd+Enter**:



Cmd + Enter

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights  
    reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!
```



```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController  
12 {  
13     var game: Concentration!  
14     didSet {  
15         updateViewFromModel()  
16     }  
17 }  
18  
19 var flipCount = 0 {  
20     didSet {  
21         flipCountLbl.text = "Flips: \(flipCount)"  
22     }  
23 }  
24  
25 @IBOutlet weak var flipCountLbl: UILabel!  
26  
27 @IBOutlet var cardButtons: [UIButton]!
```

При редактировании какая главная кнопка? Правильно **Enter**.

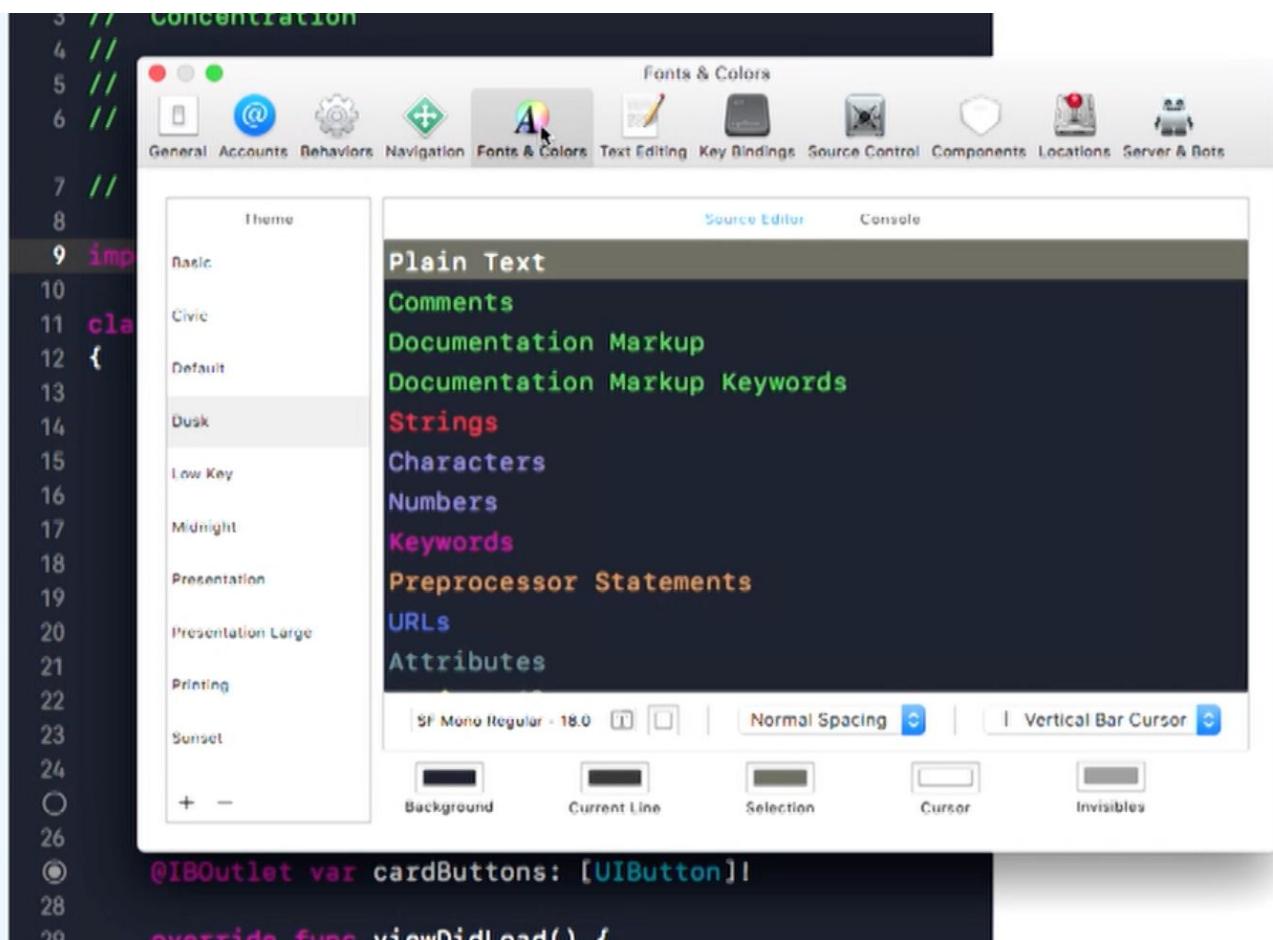
Команда **Cmd+Enter** отображает главный редактор.

А вариант редактирования с использованием вспомогательного редактора **Assistant editor** организуется с помощью комбинации клавиш **Cmd+Option+Enter**.

Все это супер полезно.

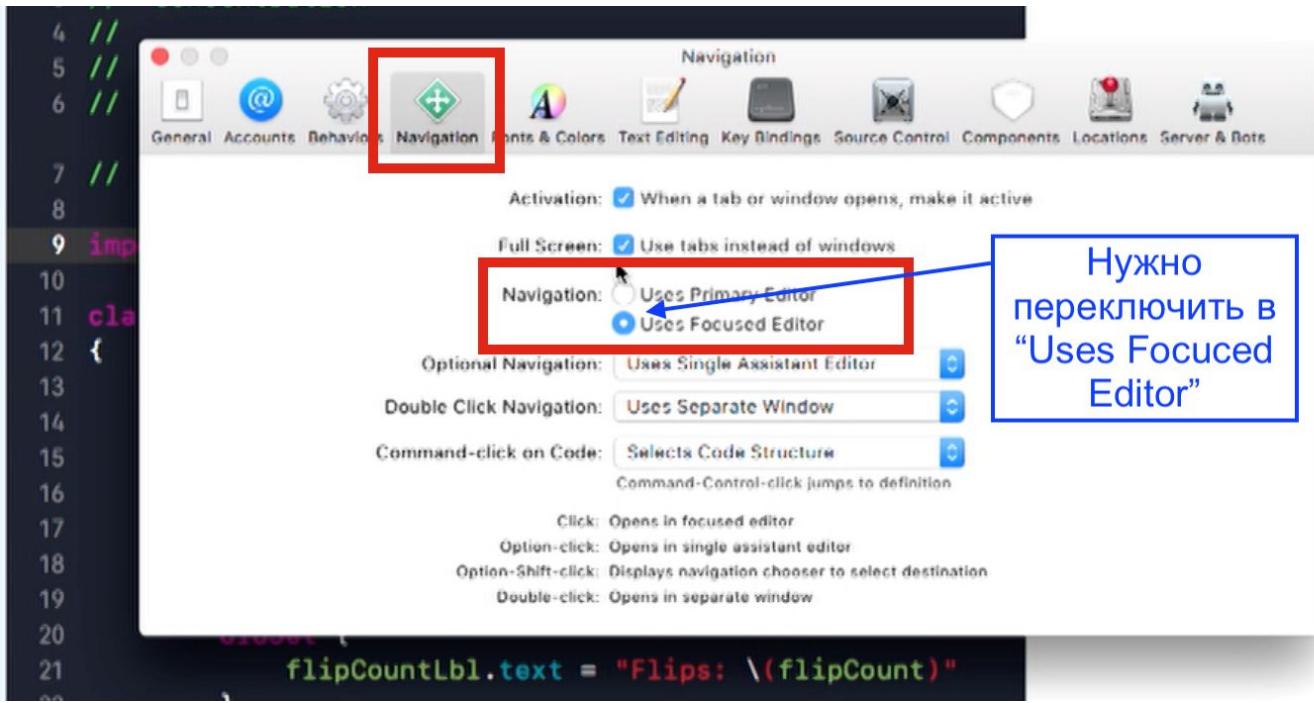
Единственная вещь, на которую я хотел бы обратить ваше внимание состоит в следующем.

Если вы пойдете в **Xcode -> Preferences**, где столько всего можно переключать:



Мы сейчас не будем гулять по настройкам, очевидно это было бы чересчур на сегодня.

Можно менять шрифты, цвета. Но вот что я бы настоятельно рекомендовал вам сделать, это переключить или проверить некоторую опцию, которая находится на панели **Navigation** в пункте **Navigation**. Это переключатель «**Uses Primary Editor**» или «**Uses Focused Editor**» по умолчанию находится в положении «**Primary**».



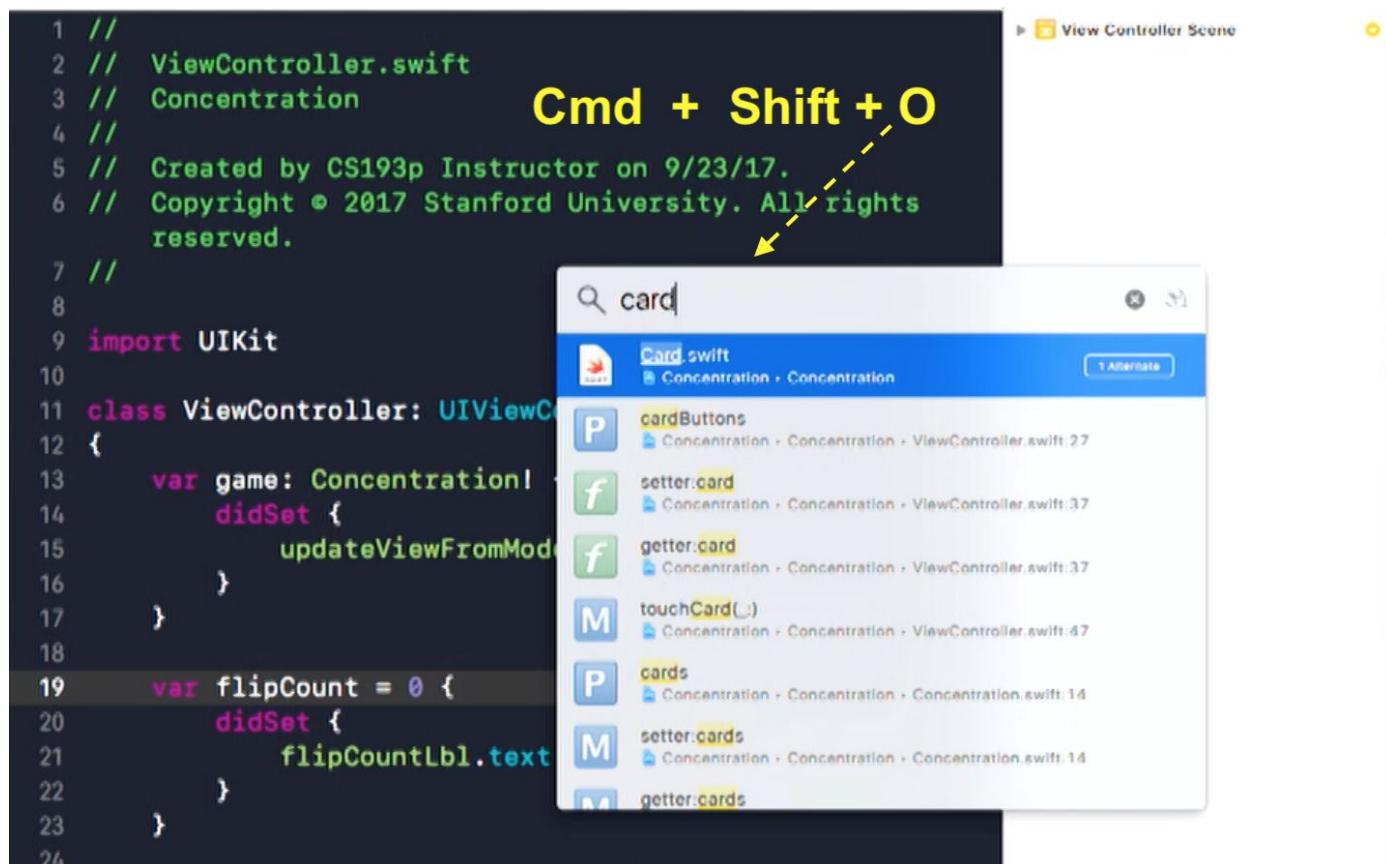
Если вы так и оставите его, это может доставить вам ряд мучений при использовании определенных сочетаний клавиш.

----- 5 -ая минута лекции -----

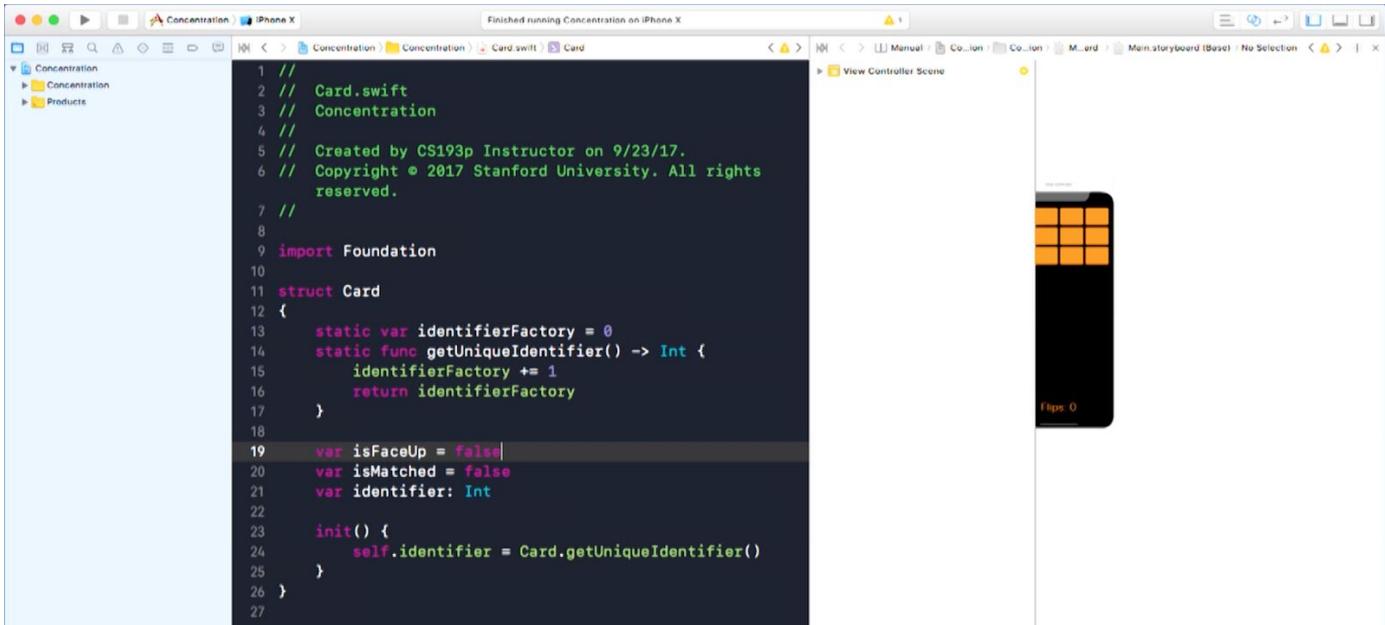
Вот пример.

Одна из команд позволяет открывать определенные файлы без необходимости находить их в навигаторе. Для этого мы используем комбинацию клавиш **Cmd+Shift+O**. **O** значит - **open**.

Мы все знаем, что в этом проекте есть файл **Card.swift**. Вбиваем часть имени искомого файла **"card"** в появившуюся поисковую строку наподобие **Spotlight ...** и готово.



Мы нашли файл **Card.swift**, кликаем на нем, и он перед нами:

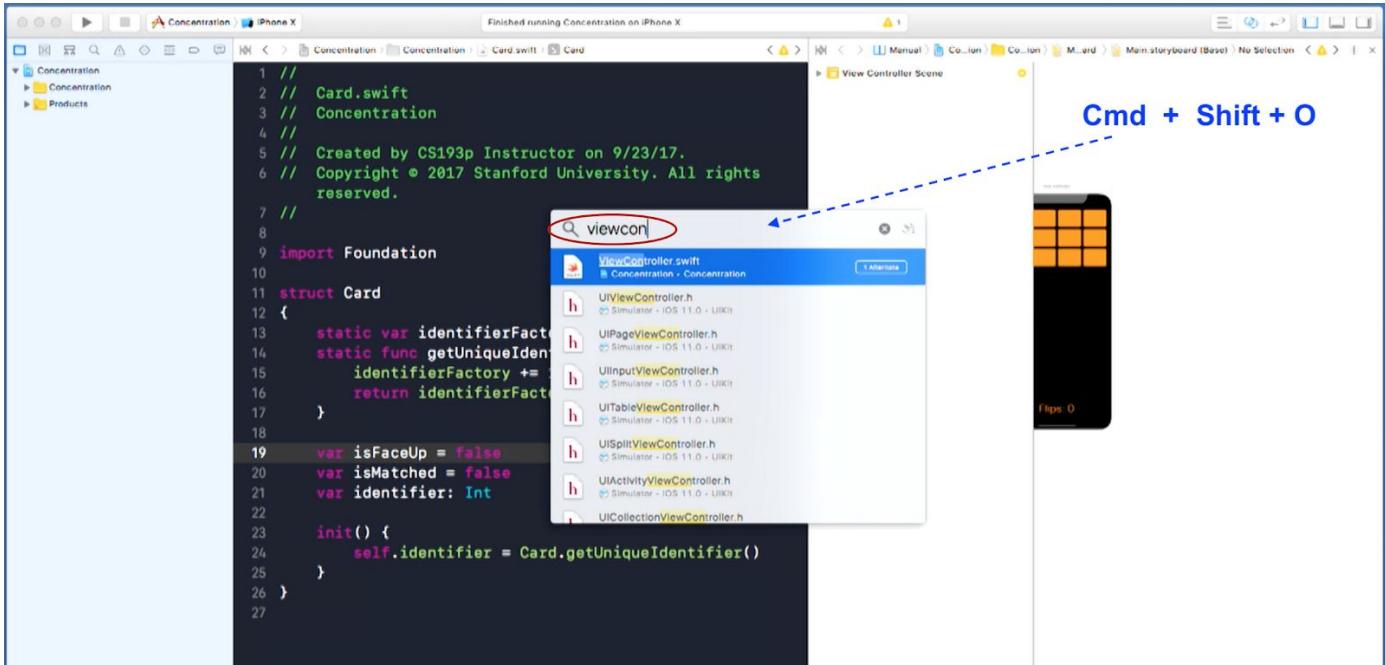


```
1 // Card.swift
2 // Concentration
3 // Created by CS193p Instructor on 9/23/17.
4 // Copyright © 2017 Stanford University. All rights reserved.
5
6 import Foundation
7
8 struct Card {
9
10     static var identifierFactory = 0
11     static func getUniqueIdentifier() -> Int {
12         identifierFactory += 1
13         return identifierFactory
14     }
15
16     var isFaceUp = false
17     var isMatched = false
18     var identifier: Int
19
20     init() {
21         self.identifier = Card.getUniqueIdentifier()
22     }
23 }
24
25
26
27
```

Но есть нюанс.

Что, если я находясь в правом окне редактирования, хочу таким же образом с помощью комбинации клавиш **Cmd+Shift+O** вернуться назад, к файлу **ViewController.swift**?

Находясь в правом окне, я использую комбинацию клавиш **Cmd+Shift+O**, выбираю **ViewController.swift** ...



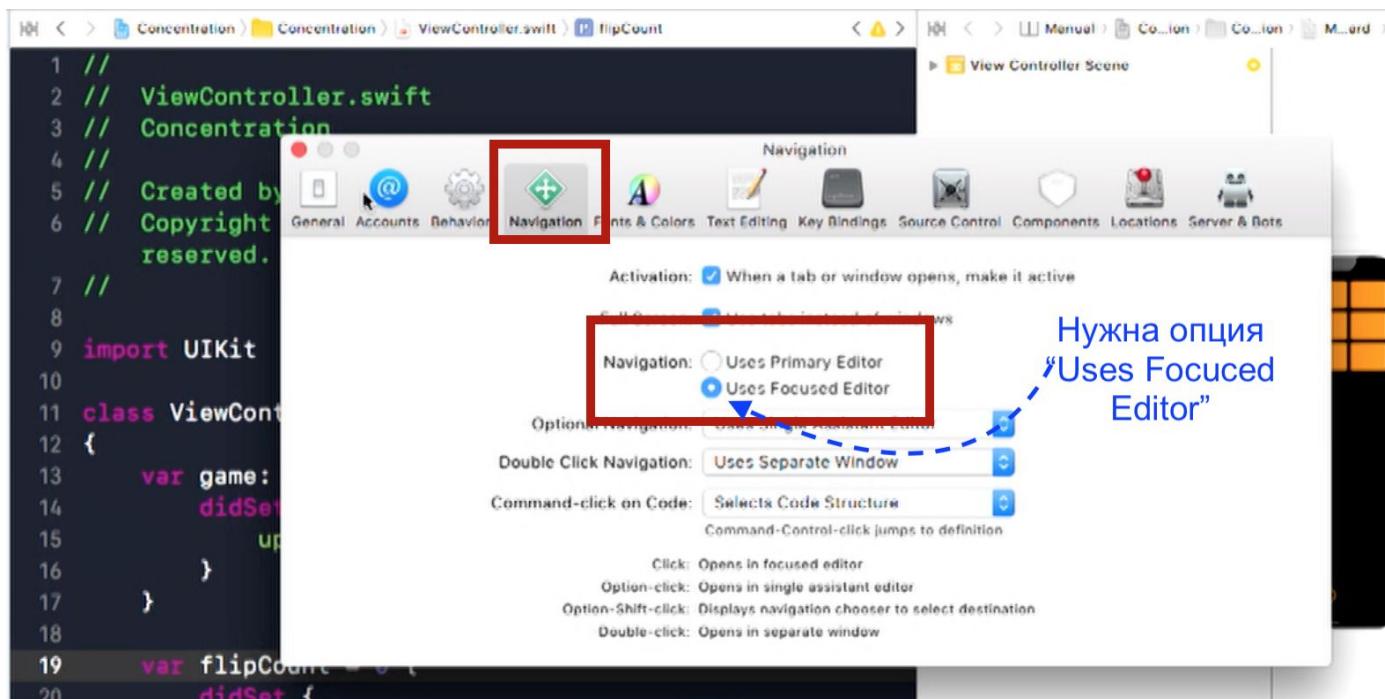
... и нужный мне файл **ViewController.swift** все равно открывается в левом окне:

```

1 /**
2 // ViewController.swift
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights
7 // reserved.
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLbl.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLbl: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!

```

То есть несмотря на то, что “фокус” находится на правом окне, нужный файл все равно открывается там, где находится основной редактор (**Primary Editor**). Если вы не согласны с такой логикой и хотите открывать файл в том редакторе, на котором в данный момент находится фокус, то переключите опцию в **Xcode preference** на панели **Navigation** в пункте **Navigation** на «**Uses Focused Editor**».



Тогда файл будет открываться в выбранном редакторе (обычно их не больше двух) - в том, на котором фокус. Очень советую.

Давайте, находясь в правом окне, с помощью комбинацию клавиш **Cmd+Shift+O**, выберем файл **Card.swift** ...

The screenshot shows the Xcode interface with the following details:

- Left Panel (Code Editor):** Displays the `ViewController.swift` file for the `Concentration` project. The code defines a `ViewController` class that initializes a `Concentration` game and tracks a `flipCount`.
- Middle Panel (Search Results):** A search results window is open, showing suggestions for the search term "card". The results include:
 - `cardButtons` (P)
 - `setter:card` (f)
 - `getter:card` (f)
 - `touchCard(_)` (M)
 - `cards` (P)
 - `setter:cards` (M)
 - `getter:cards` (M)
- Right Panel (Preview):** Shows a preview of the iPhone X storyboard scene titled "View Controller Scene". It displays a 4x4 grid of cards and a label "Flips: 0".
- Top Bar:** Shows the project name "Concentration", the file "Concentration", and the title "ViewController.swift". The status bar indicates "Main.storyboard (Base) No Selection".
- Text Overlay:** The text "Cmd + Shift + O" is overlaid in red at the top right of the image.

Теперь благодаря выбору опции «**Uses Focused Editor**», файл **Card.swift** появится в правом окне, то есть в том, на котором сейчас находится фокус:

The image shows two Xcode code editors side-by-side. The left editor contains `ViewController.swift` and the right editor contains `Card.swift`. Both files are part of the `Concentration` project.

```
// ViewController.swift
// Concentration
//
// Created by CS193p Instructor on 9/23/17.
// Copyright © 2017 Stanford University. All rights reserved.
//
// import UIKit
//
// class ViewController: UIViewController {
//     var game: Concentration!
//     didSet {
//         updateViewFromModel()
//     }
// }
//
// var flipCount = 0 {
//     didSet {
//         flipCountLbl.text = "Flips: \(flipCount)"
//     }
// }
//
// @IBOutlet weak var flipCountLbl: UILabel!
//
// @IBOutlet var cardButtons: [UIButton]!
//
// override func viewDidLoad() {
//     // ...
// }
```

```
// Card.swift
// Concentration
//
// Created by CS193p Instructor on 9/23/17.
// Copyright © 2017 Stanford University. All rights reserved.
//
// import Foundation
//
// struct Card {
//     static var identifierFactory = 0
//     static func getUniqueIdentifier() -> Int {
//         identifierFactory += 1
//         return identifierFactory
//     }
// }
//
// var isFaceUp = false
// var isMatched = false
// var identifier: Int
//
// init() {
//     self.identifier = Card.getUniqueIdentifier()
// }
```

Понятно?

Здорово.

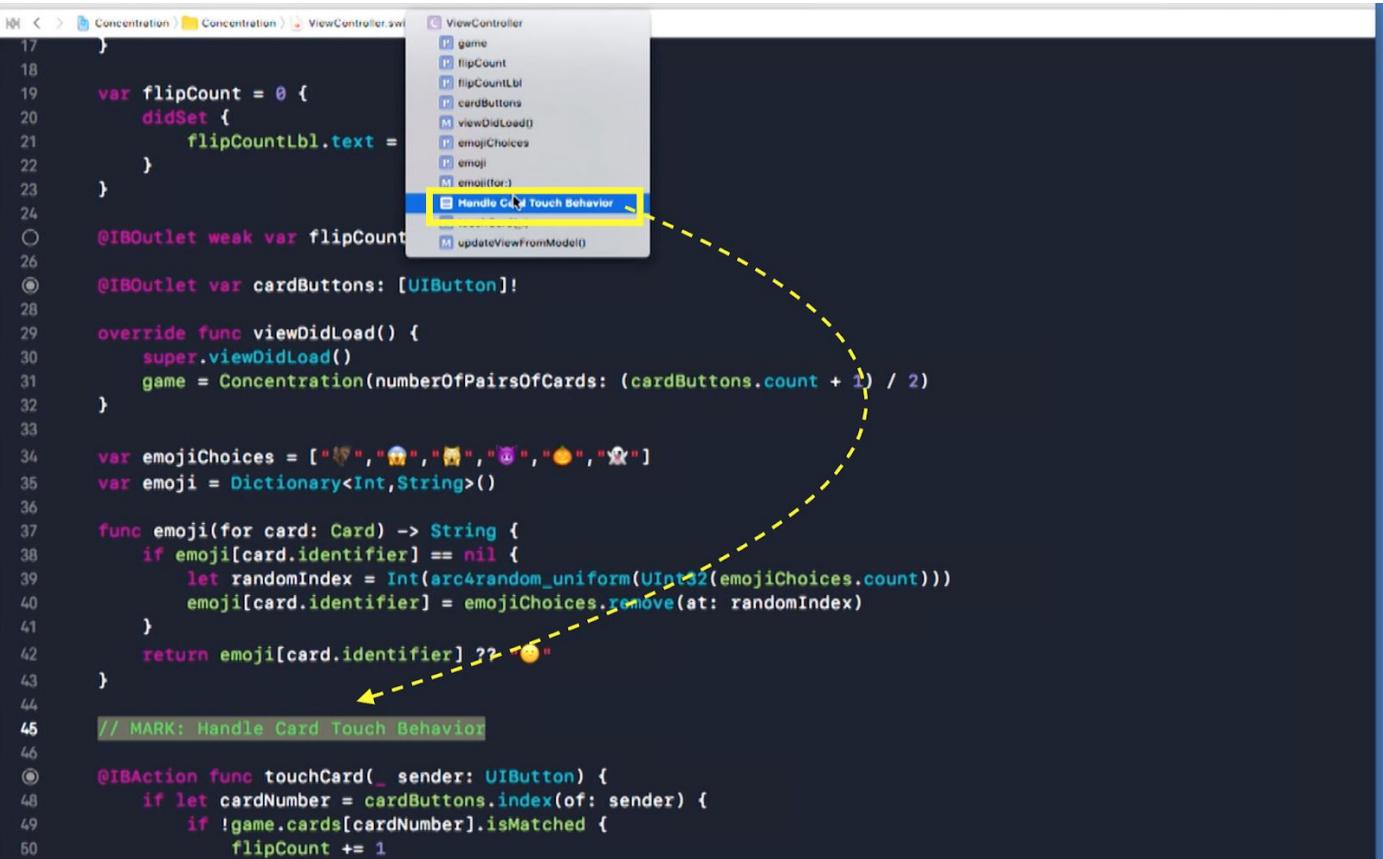
Далее очень коротко поговорим о редактировании текста в каждом отдельном окне. Хочу обратить ваше внимание на одну вещь, с которой вы еще не сталкивались, поскольку ваши проекты не вырастали до крупных размеров, но они разрастаются очень быстро, уж, поверьте. В итоге в одном проекте их могут быть сотни файлов с кодами. Среди прочего есть штука, при помощи которой вы

можете облегчить себе навигацию внутри файла с кодом, это вот такой комментарий, начинающийся с текста “// MARK”:

// MARK: Название закладки

```
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "🂷"
43     }
44
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
51         }
52         game.chooseCard(at: cardNumber)
53         updateViewModel()
54 }
```

Такой комментарий, начинающегося с текста “// MARK”, создает в тексте закладку, на которую мы можем очень быстро “прыгнуть” с помощью панели “быстрого выбора” **Jump Bar**:



можно быстро «прыгнуть» к этой закладке - также как и к любому свойству, методу (или классу если в файле их несколько). **Jump Bar** - это что-то вроде адресной строки редактора (есть и у основного, и у вспомогательного редактора), с помощью которого вы можете также быстро “прыгнуть” к любому свойству, методу или классу, если в файле их несколько.

И как видите, использование таких закладок чрезвычайно полезно, особенно для длинных файлов. Следующая вещь, о которой поговорим, касается редактирования текста. Вам очень пригодится усвоение даже простейших клавишных команд редактирования текста.

Например, выбрав много строк в коде, вы вдруг решите закомментировать их. Комбинация клавиш

Cmd+/ ...

```
34     var emojiChoices = ["👻", "🧙‍♀️", "🧙‍♂️", "😈", "🎃", "🕸️"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         // if emoji[card.identifier] == nil {
39         //     let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         //     emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         //
42         //     return emoji[card.identifier] ?? "😊"
43     }
```

Cmd + /

... сделает именно это, ...

```
34     var emojiChoices = ["👻", "🧙‍♀️", "🧙‍♂️", "😈", "🎃", "🕸️"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "😊"
43     }
```

Cmd + /

а повторное применение **Cmd+/** на закомментированных строках раскомментирует их обратно:

```
34     var emojiChoices = ["👻", "🧙‍♀️", "🧙‍♂️", "😈", "🎃", "🕸️"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "😊"
43     }
44 }
```

Еще более полезным бывает, если все отступы спутались, и из-за столь неудачного форматирования сложно разобраться с кодом (где какой блок в какой вложен). В этом случае достаточно просто выделить текст и нажать комбинацию клавиш **Ctrl+I**, где **I** означает **indent** (отступ):

```

26
○     @IBOutlet var cardButtons: [UIButton]!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31         game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32     }
33
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "🂱"
43     }
44
45     // MARK: Handle Card Touch Behavior
46
○     @IBAction func touchCard(_ sender: UIButton) {
48         if let cardNumber = cardButtons.index(of: sender) {
49             if !game.cards[cardNumber].isMatched {
50                 flipCount += 1
51             }
52             game.chooseCard(at: cardNumber)
53             updateViewFromModel()
54         }
55     }

```

Cmd + |

И он магическим образом разместит все так, как нам нужно:

```

var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
var emoji = Dictionary<Int, String>()

func emoji(for card: Card) -> String {
    if emoji[card.identifier] == nil {
        let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
        emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
    }
    return emoji[card.identifier] ?? "🂱"
}

// MARK: Handle Card Touch Behavior

@IBAction func touchCard(_ sender: UIButton) {
    if let cardNumber = cardButtons.index(of: sender) {
        if !game.cards[cardNumber].isMatched {
            flipCount += 1
        }
        game.chooseCard(at: cardNumber)
        updateViewFromModel()
    }
}

```

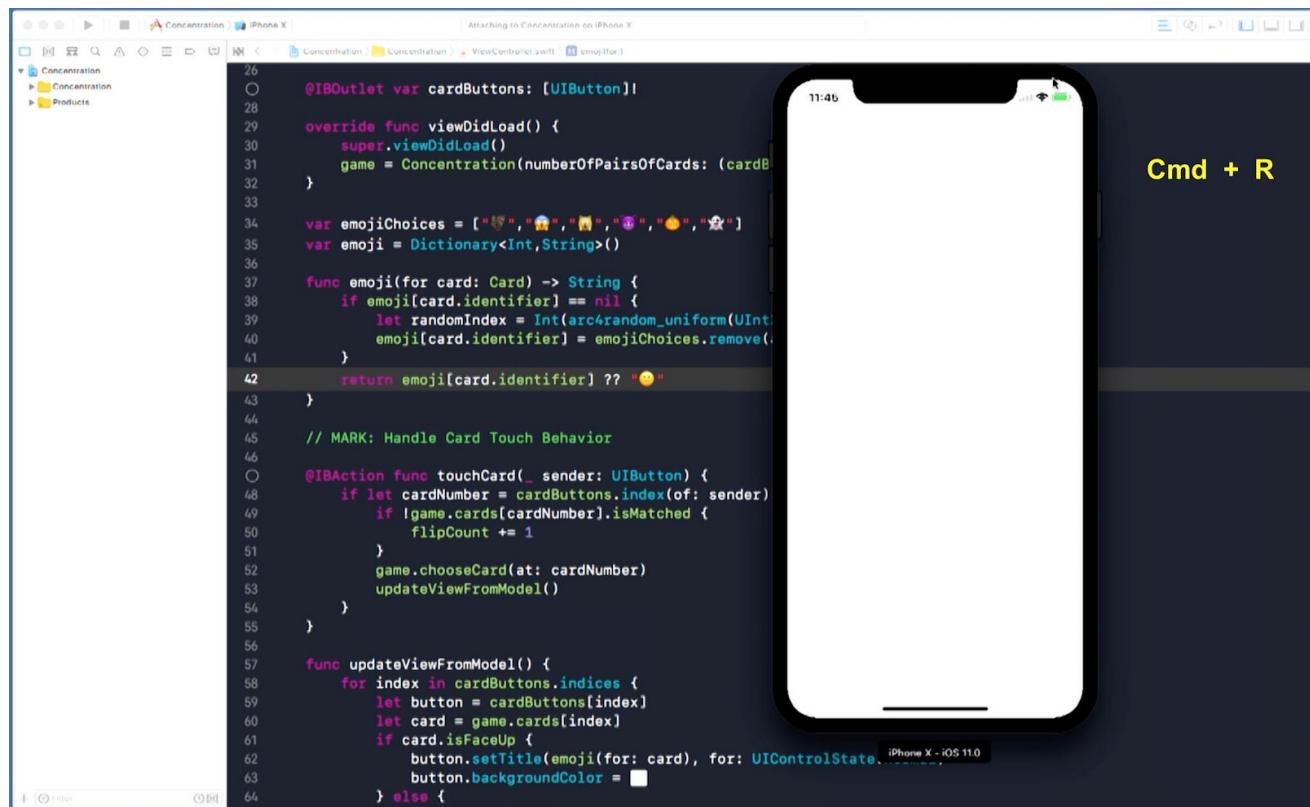
Это вам здорово сэкономит время и избавит от головной боли. Правильно расставленные отступы помогают читать код.

Всё это, кстати, есть на [Piazza](#), я просто хочу подготовить почву.

Надеюсь, мы разделились по быстрому с наиболее скучной частью лекции - с клавишными командами. Но я очень советую вам изучить весь этот материал и начать загружать его в моторную память, применяя клавишиные «ускорители» на практике при навигации в Xcode, а не мучить себя выполнением всех действий вручную.

Переходим к использованию отладчика.

Как уже было упомянуто, мы будем использовать слегка искаженную версию игры **Concentration** из предыдущей лекции. Посмотрим, что произойдёт, если мы её запустим с помощью надежной комбинации клавиш **Cmd+R**, например, на симуляторе **iPhone X**.



Мое приложение закончилось аварийно прямо с самого начала. Нас выбросило в **AppDelegate.swift**, который является главным файлом. Сбой произошел в методе **main** (как нам показывает **Debug Navigator** - левая панель), но это не слишком информативно. Огромное количество текста, выводимого на консоль.

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Shows "Concentration" as the project name and "iPhone X" as the target.
- Left Navigator:** Displays system monitoring metrics (CPU, Memory, Disk, Network) and a list of threads. The "Thread 1" entry, which corresponds to the main thread, is highlighted with a red box.
- Editor Area:** Shows the code for `AppDelegate.swift`. The first few lines are:

```
1 //  
2 // AppDelegate.swift  
3 // Concentration  
4 //  
5 // Created by CS193P Instructor on 9/23/17.  
6 // Copyright © 2017 Stanford University. All rights reserved.  
7 //
```
- Call Stack:** Located at the bottom right, it lists the stack frames from the application's main thread up to the system libraries. The stack trace includes:

```
46 CoreFoundation  
CFRunLoopRunSpecific + 409  
41 GraphicsServices  
GSEventRunModal + 62  
42 UIKit  
UIApplicationMain + 159  
43 Concentration  
65  
44 libdyld.dylib  
+ 1  
)  
libc++abi.dylib: terminating with uncaught exception of type  
NSException  
(lldb)
```

Чтобы вникнуть в происходящее, первым делом, следует взглянуть на трассировку стека (**stack trace**). А выводится она в нижней отладочной области **Debug Area** на панеле, называемой **console**:

1 //
2 // AppDelegate.swift
3 // Concentration
4 //
5 // Created by CS193p Instructor on 9/23/17.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14 var window: UIWindow?
15
16
17 func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
18 [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
19 // Override point for customization after application launch.
20 return true
21 }
22 }

Thread 1: signal SIGABRT

Трассировка стека →

```
[_UICanvasLifecycleSettingsDiffAction  
performActionsForCanvas:withUpdatedScene:settingsDiff:fromSettings:transitionContext:] + 231  
26 UIKit 0x0000000102fb7a4c -  
[_UICanvas scene:didUpdateWithDiff:transitionContext:completion:] + 392  
27 UIKit 0x000000010282dac6 -  
[UIApplication workspace:didCreateScene:withTransitionContext:completion:] + 823  
28 UIKit 0x0000000102df6623 -  
[UIApplicationSceneClientAgent scene:didInitializeWithEvent:completion:] + 369  
29 FrontBoardServices 0x0000000107a51158 -  
[FBSSceneImpl _didCreateWithTransitionContext:completion:] + 338  
30 FrontBoardServices 0x0000000107a59c4d __56-[FBSWorkspace client:handleCreateScene:withCompletion:]_block_invoke_2 + 235  
31 libdispatch.dylib 0x0000000106ccf43c  
_dispatch_client_callout + 8  
32 libdispatch.dylib 0x0000000106cd4e94
```

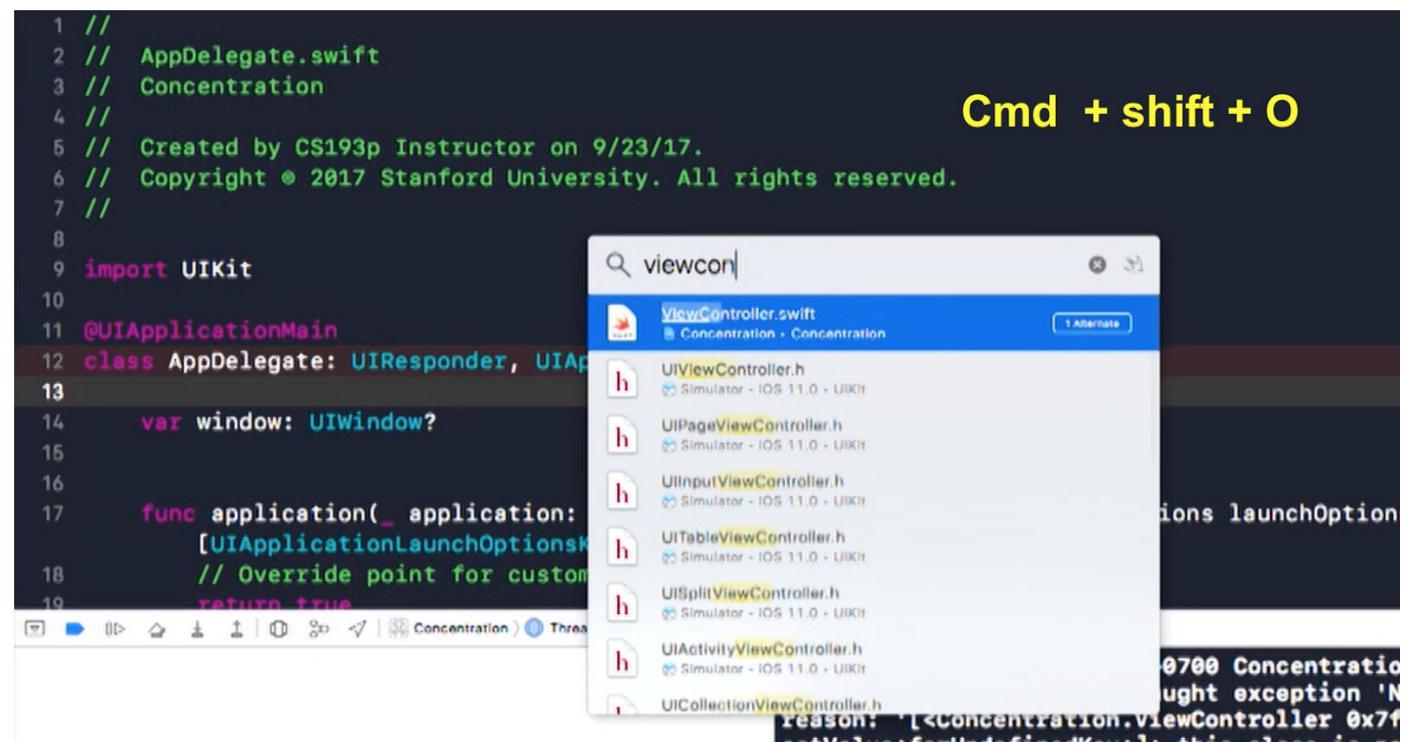
И вы видите все эти многословные логи трассировки стека, с внутренней информацией **iOS**. Там есть и вывод от **UIKit**, и от **CoreFoundation**. От всех этих адресов памяти пользы мало сейчас. А

вот если прокрутить наверх, то практически всегда вы увидите реальную причину завершения программы:

```
2017-09-29 11:45:33.262429-0700 Concentration[9410:1150045] ***
Terminating app due to uncaught exception 'NSUnknownKeyException',
reason: '[<Concentration.ViewController 0x7fc9c1706370>
setValue:forUndefinedKey:]: this class is not key value coding-
compliant for the key flipCountLabel.'
*** First throw call stack:
(
    0  CoreFoundation                      0x0000000105c221cb
    1  __exceptionPreprocess + 171         0x0000000102261f41
    2  libobjc.A.dylib                     0x00000001029c6c82 -
    3  CoreFoundation                      0x0000000105c22119 -
    4  Foundation                          0x0000000101c841e3 -
    5  NSObject(NSKeyValueCoding) setValue:forKey:] + 292
    6  UIKit                             0x0000000102cb2c40 -
    7  [UIRuntimeOutletConnection connect] + 109
    8  CoreFoundation                      0x0000000105bc557d -
```

И в нашем случае - исключение **UNKnownKeyException**. Пока мы не знаем, что это такое. Но мы знаем, что причина сбоя кроется в **Concentration.ViewController** (уже известен хотя бы файл, в котором что-то произошло). И там какая-то ошибка **setValue: forUndefinedKey**. Не вполне, понятно, что бы это могло значить, ведь мы не писали такого метода **setValue**. И после говорится «этот класс не совместим с ключом **flipCountLabel**». Вот это ближе к сути: что-то не так с **flipCountLabel**.

Идём в **ViewController** с помощью комбинации клавиш **Cmd+Shift+O...**



```
1 //
2 //  AppDelegate.swift
3 //  Concentration
4 //
5 //  Created by CS193p Instructor on 9/23/17.
6 //  Copyright © 2017 Stanford University. All rights reserved.
7 //

8
9 import UIKit
10
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14     var window: UIWindow?
15
16
17     func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
18         // Override point for customization after application launch.
19         return true
20     }
21 }
```

проверяем **flipCountLabel**...

```
13     var game: Concentration! {
14         didSet {
15             updateViewFromViewModel()
16         }
17     }
18
19     var flipCount = 0 {
20         didSet {
21             flipCountLbl.text = "Flips: \(flipCount)"
22         }
23     }
24
25     @IBOutlet weak var flipCountLbl: UILabel!
26
27     @IBOutlet var cardButtons: [UIButton]!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31         game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32     }

```

... и видим какое-то сокращение `flipCountLbl` вместо `flipCountLabel`. А в отладчике - `flipCountLabel`.

У кого-нибудь есть догадки, что происходит? Откуда вообще в консоли взялось это имя `flipCountLabel`?

Есть идеи?

У нас в коде нет ничего с именем `flipCountLabel`.

----- 10 -ая минута лекции -----

```
13     var game: Concentration! {
14         didSet {
15             updateViewFromViewModel()
16         }
17     }
18
19     var flipCount = 0 {
20         didSet {
21             flipCountLbl.text = "Flips: \(flipCount)"
22         }
23     }
24
25     @IBOutlet weak var flipCountLbl: UILabel!
26
27     @IBOutlet var cardButtons: [UIButton]!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31         game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32     }

```

Кто-нибудь знает, как так вышло? Нет?

Переименовали свойства в коде, а в **Storyboard** осталась ссылка на прежнее имя.

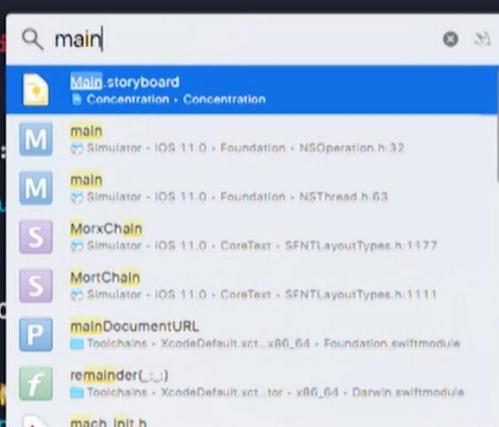
Да, именно так. Если взглянем на **Storyboard** с помощью комбинации клавиш **Cmd+Shift+O...**

```

13     var game: Concentration! {
14         didSet {
15             updateViewModel()
16         }
17     }
18
19     var flipCount = 0 {
20         didSet {
21             flipCountLbl.text = "Flips: \(flipCount)"
22         }
23     }
24
25     @IBOutlet weak var flipCountLbl: UILabel!
26
27     @IBOutlet var cardButtons: [UIButton]!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31         game = Concentration(numberOfPairsOfCards:
32             (cardButtons.count + 1) / 2)
33     }
34
35     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷"]
36     var emoji = Dictionary<Int, String>()
37
38     func emoji(for card: Card) -> String {
39         if emoji[card.identifier] == nil {
40             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))

```

Cmd + shift + O



Лучше откроем-ка код и **Storyboard** одновременно рядышком, опять же с помощью комбинации клавиш **Cmd+Option+Enter**.

```

12 {
13     var game: Concentration! {
14         didSet {
15             updateViewModel()
16         }
17     }
18
19     var flipCount = 0 {
20         didSet {
21             flipCountLbl.text = "Flips: \(flipCount)"
22         }
23     }
24
25     @IBOutlet weak var flipCountLbl: UILabel!
26
27     @IBOutlet var cardButtons: [UIButton]!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31         game = Concentration(numberOfPairsOfCards:
32             (cardButtons.count + 1) / 2)
33     }
34
35     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷"]
36     var emoji = Dictionary<Int, String>()
37
38     func emoji(for card: Card) -> String {

```

```

1 // 
2 //  Card.swift
3 //  Concentration
4 //
5 //  Created by CS193p Instructor on 9/23/17.
6 //  Copyright © 2017 Stanford University. All rights
reserved.
7 //
8
9 import Foundation
10
11 struct Card {
12
13     static var identifierFactory = 0
14     static func getUniqueIdentifier() -> Int {
15         identifierFactory += 1
16         return identifierFactory
17     }
18
19     var isFaceUp = false
20     var isMatched = false
21     var identifier: Int
22
23     init() {
24         self.identifier = Card.getUniqueIdentifier()
25     }
26

```

```

2017-09-29 11:45:33.262429-0700 Concentration[9410:1150045] ***
Terminating app due to uncaught exception 'NSUnknownKeyException',
reason: '[<Concentration.ViewController 0x7fc9c1706370>
setValue:forUndefinedKey:]: this class is not key value coding-
compliant for the key flipCountLabel.'
*** First throw call stack:
(
    0   CoreFoundation                      0x0000000105c221cb __exceptionPreprocess + 171
    1   libobjc.A.dylib                     0x0000000102261f41 objc_exception_throw + 48

```

Давайте получим слева **Storyboard** с помощью комбинации клавиш **Cmd+Shift+O...**

```
Concentration > Concentration > ViewController.swift [M emoji(for:)]
```

```
12 {  
13     var game: Concentration! {  
14         didSet {  
15             updateViewFromModel()  
16         }  
17     }  
18  
19     var flipCount = 0 {  
20         didSet {  
21             flipCountLbl.text = "Flips:  
22         }  
23     }  
24  
25     @IBOutlet weak var flipCountLbl:  
26  
27     @IBOutlet var cardButtons: [UIButton]  
28  
29     override func viewDidLoad() {  
30         super.viewDidLoad()  
31         game = Concentration(numberOfDecks:  
32             (cardButtons.count + 1))  
33     }  
34  
35     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷", "🂸", "🂹"]  
36  
37     var emoji = Dictionary<Int, String>()
```

Cmd + shift + C



The screenshot shows the Xcode interface with a search bar at the top containing the text 'main'. A sidebar on the right displays search results for 'main', listing various files and symbols found in the project. The results include 'Main storyboard', multiple entries for 'main' (Simulator - iOS 11.0), 'MorxChain' (Simulator - iOS 11.0), 'MorxChain' (Simulator - iOS 11.0 - CoreText), 'mainDocumentURL' (Toolchains - XcodeDefault.xctoolchain), 'remainder(...)' (Toolchains - XcodeDefault.xctoolchain), and 'mach_init.h'.

Cmd + shift + O

```
1 //|
2 //| Card.swift
3 //| Concentration
4 //|
5 //| Created by CS193p Instructor on 9/23/17.
6 //| Copyright © 2017 Stanford University. All rights
7 //| reserved.
8 //|
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

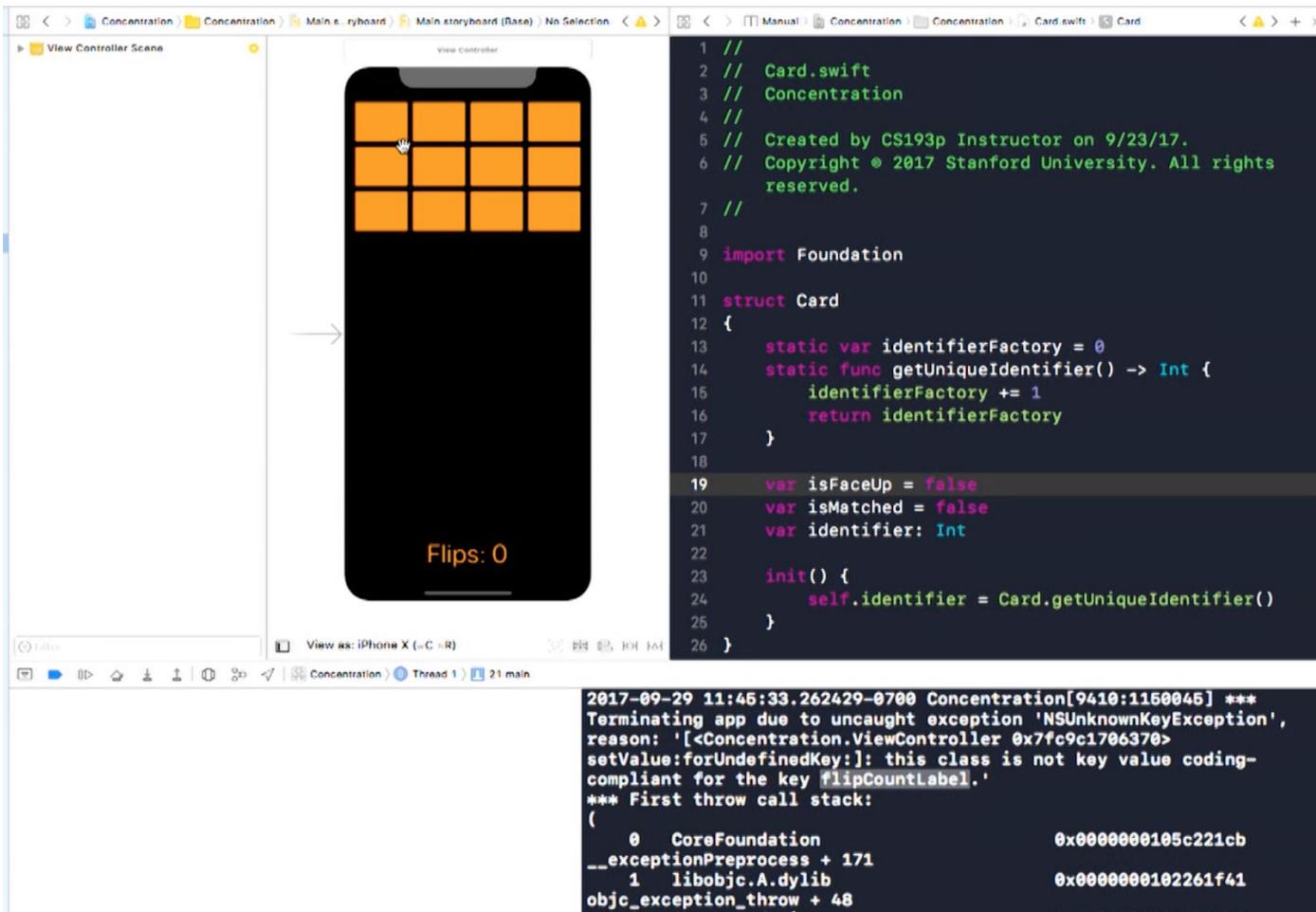
```
import Foundation
```

```
class Card {
    var identifier: Int = 0
    var faceUp: Bool = false
    var faceDown: Bool = true

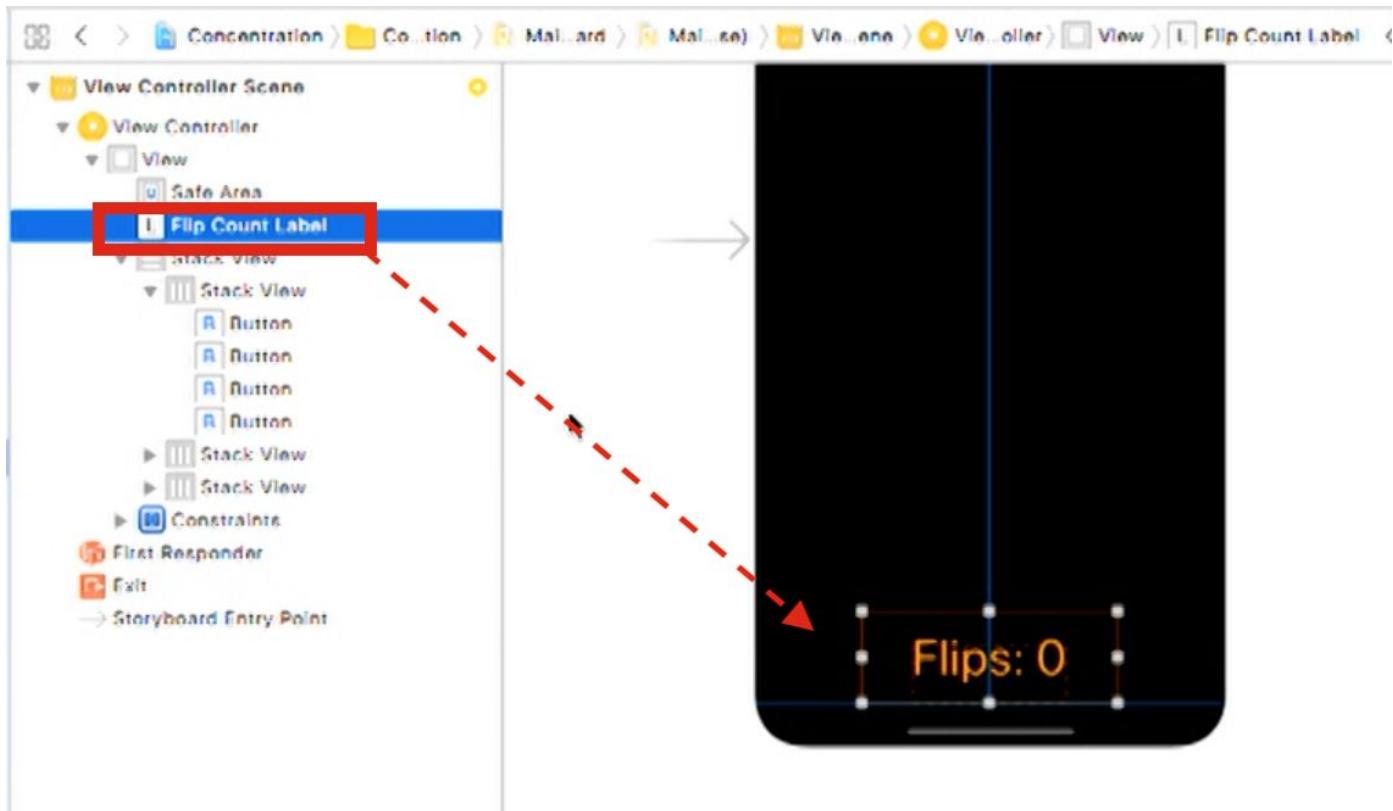
    init() {
        self.identifier = Card.getUniqueIdentifier()
    }

    func flip() {
        faceUp = !faceUp
        let completion: () -> Void = {
            self.faceDown = !self.faceUp
        }
        self.faceUp ? completion() : completion()
    }
}
```

Открылась **Storyboard**:

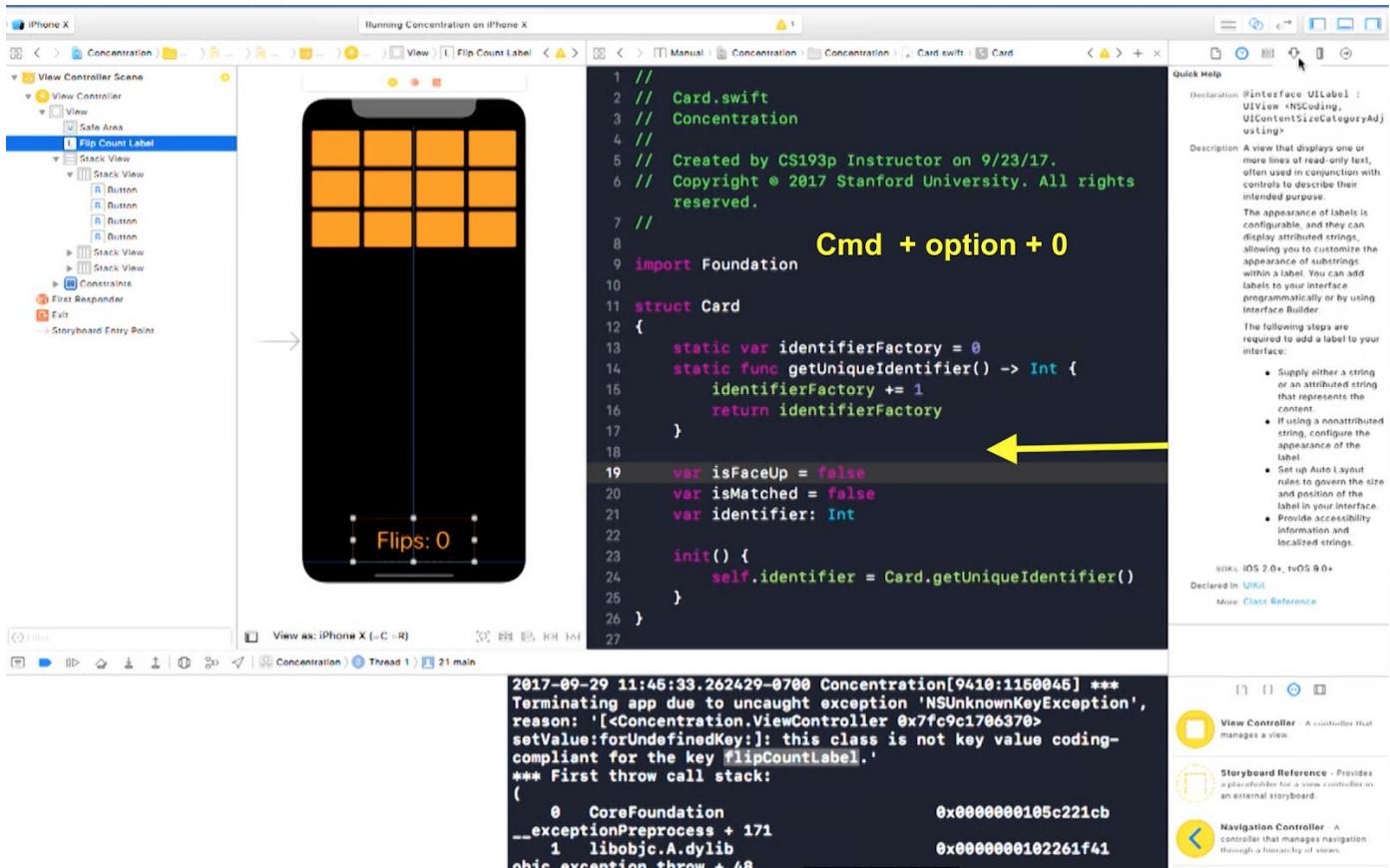


Я разверну на полный экран, чтобы было лучше видно. И вот в **Document Outline** (Схеме UI) находим «**Flip Count Label**».

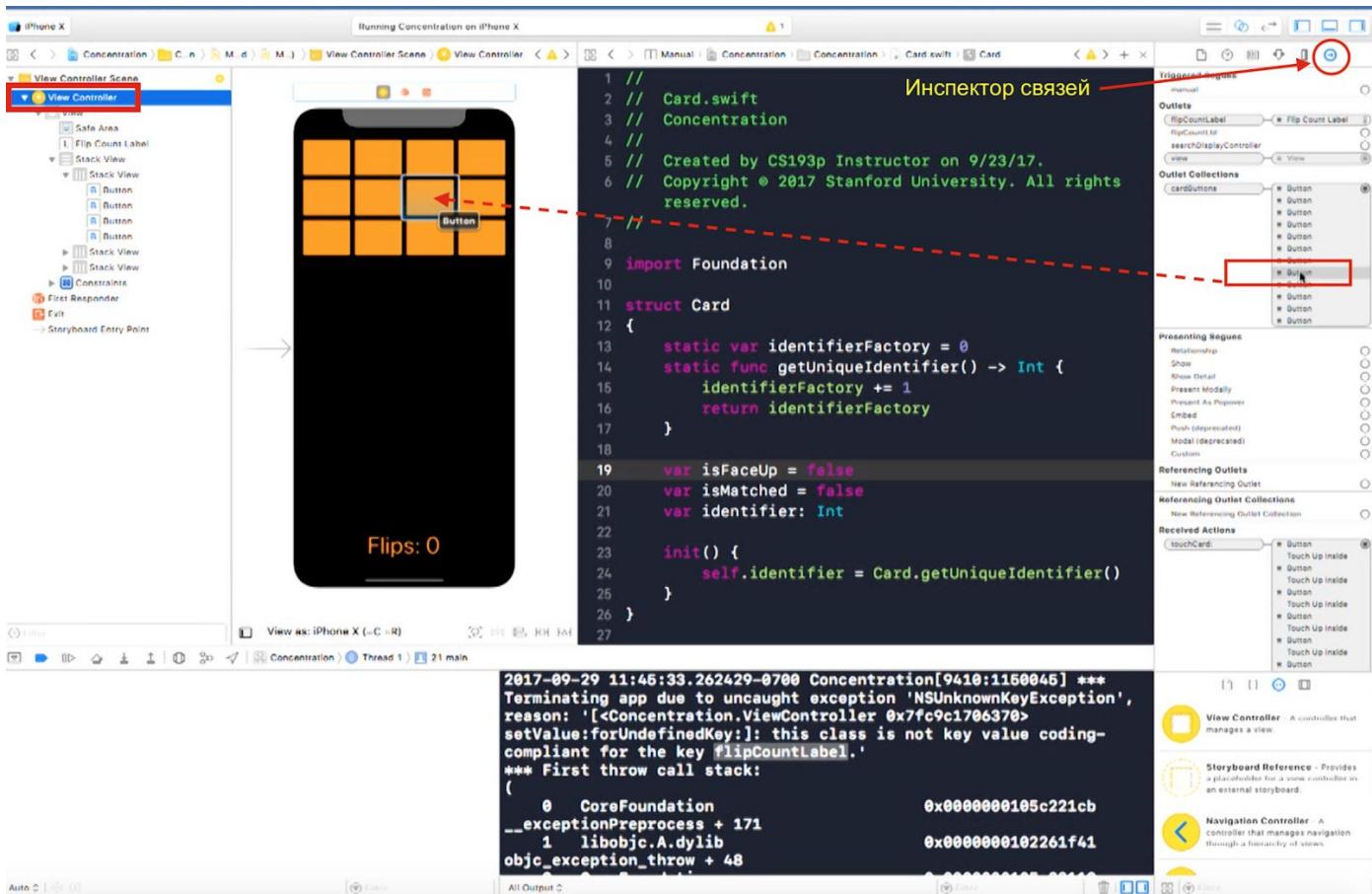


Имена, отображаемые в **Document Outline**, вообще-то не являются «официальными» именами объектов.

Отобразим область утилит Utilities Area с помощью комбинации клавиш **Cmd+Option+0** ...

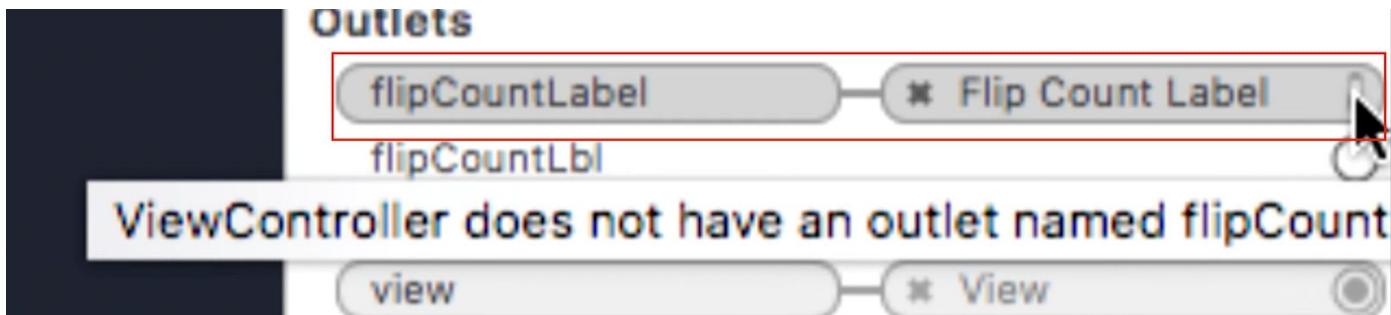


... выберем **View Controller** в **Document Outline** и перейдем на крайнюю правую закладку с помощью клавишиной команды **Cmd+Option+5**:



Там мы увидим все связи и по мере того, как я перебираю эти связи на **Storyboard** подсвечиваются объекты **UI**, к которым эти связи ведут.

В секции **Outlets** мы видим **outlet** с именем **flipCountLabel** ...



... и он связан с именем «**Flip Count Label**», то есть с именем нашей метки в **Document Outline**. Но если присмотреться, то справа от «**Flip Count Label**» стоит восклицательный знак, наводим на него мышку и получаем сообщение о том, что во **ViewController** отсутствует **outlet flipCountLabel**. А в нормальном благополучном случае на этом месте стоит жирная точка, характерная для **Outlet**.

Outlets

flipCountLabel → * Flip Count Label

flipCountLbl

searchDisplayController

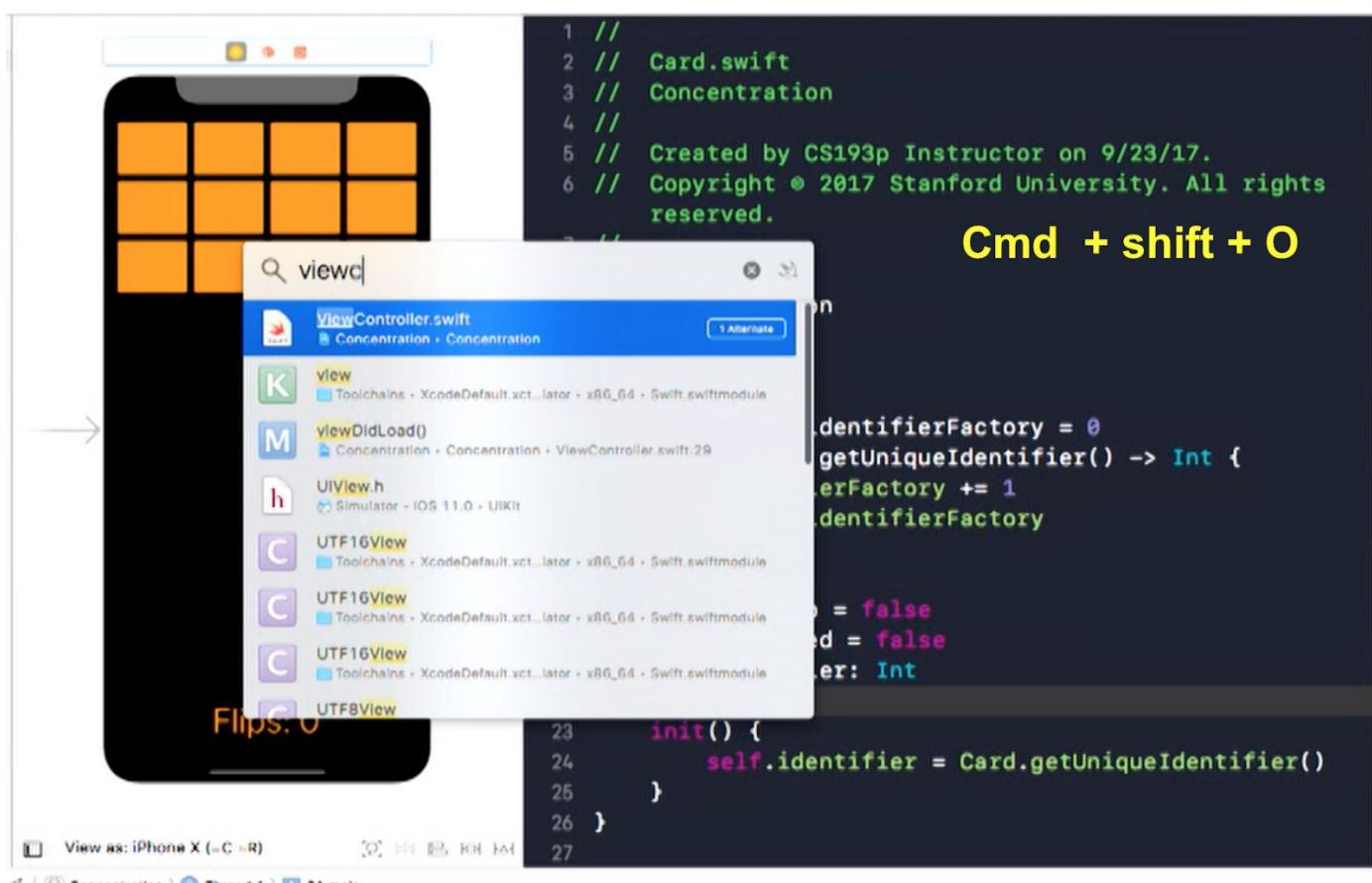
view → * View

Outlet Collections

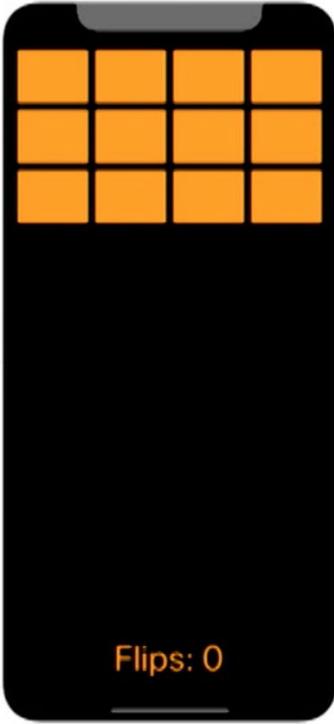
cardButtons → * Button

* Button

Идем в код класса **ViewController** ...

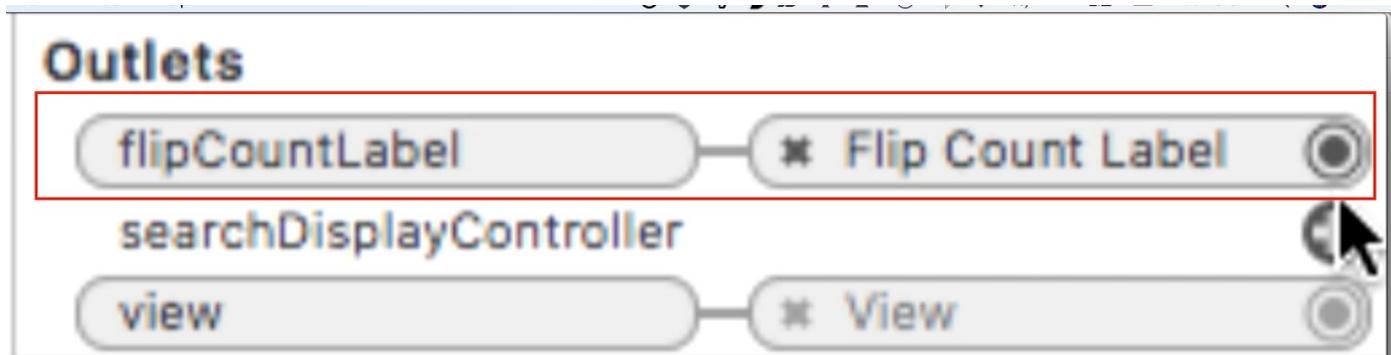


переименовываем **Outlet** как надо ...



```
11 class ViewController: UIViewController
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards:
32                           (cardButtons.count + 1) / 2)
33 }
34
35 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷"]
36 var emoji = Dictionary<Int, String>()
37
38 func emoji(for card: Card) -> String {
```

... идем обратно на **Storyboard** и видим, что ошибки больше нет, все сконнектилось.

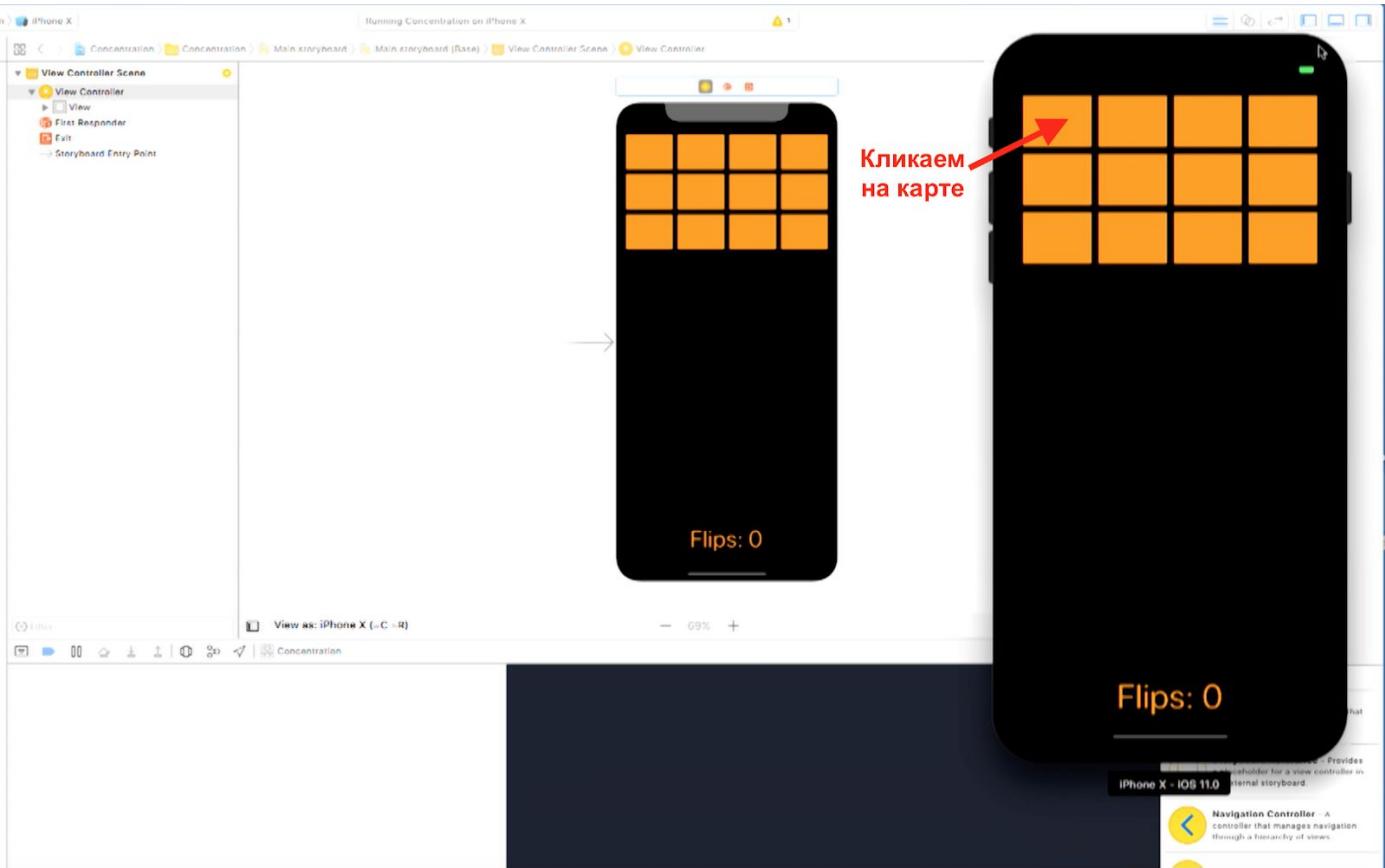


Такой баг будет поначалу вам часто встречаться. Иногда трудно избежать рассинхронизации между кодом и **Storyboard**.

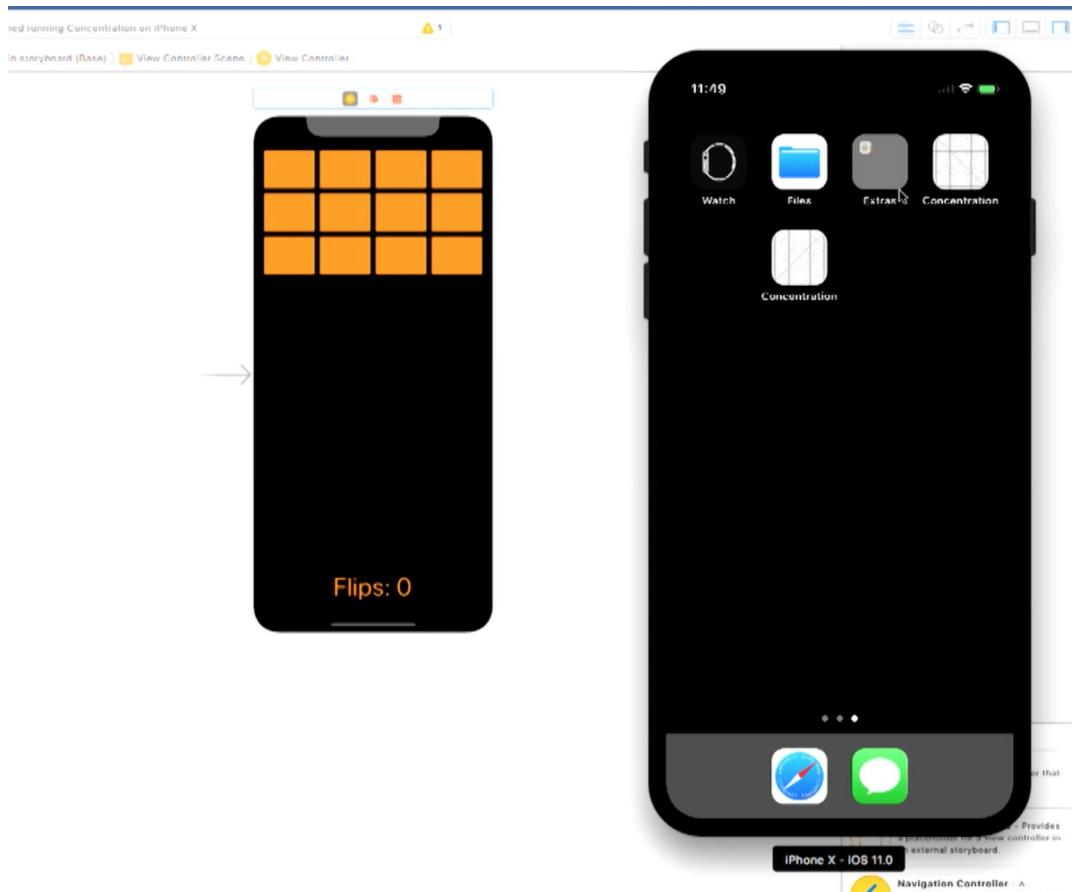
Так что запомните, что эта ошибка «**key value Coding compliance**» чаще всего говорит о таком рассинхронизированном состоянии. В этом случае вам следует выбрать **View Controller** в **Document Outline** и заглянуть в Инспектор Связей (**Connections Inspector**) на крайнюю правую закладку с помощью клавиши комбинации **Cmd+Option+5**. Это поможет вам избавиться от таких ошибок.

Запускаем наше приложение с помощью комбинации клавиш **Cmd + R**.

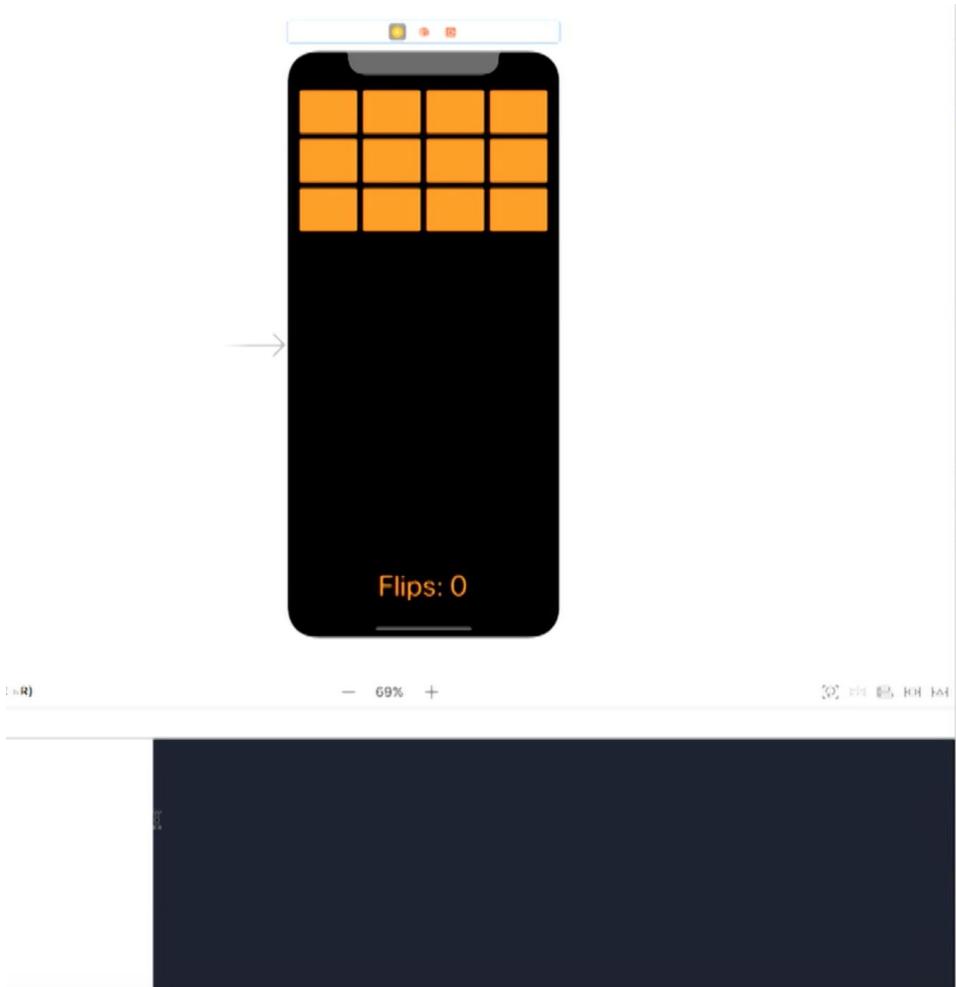
Ну вот, теперь запустилось.



Давайте кликнем на карту и поиграем в игру. Но как только я кликнул на первой карте, приложение вылетело:



Смотрим вниз в console.



Вообще никаких сообщений! Ни словечка в консоль о том, почему программа вылетела.

Запустим-ка еще раз, посмотрим удастся ли нам выяснить, что заставляет приложение вылетать.

```

12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices[randomIndex]
41     }
42     return emoji[card.identifier]!
43 }
44
45 func updateViewFromModel() {
46     for button in cardButtons {
47         if let card = button.tag {
48             button.setTitle(emoji(for: card), for: .normal)
49         }
50     }
51 }
52
53 extension Concentration {
54     static func numberOfPairsOfCards(_ count: Int) -> Int {
55         return count * (count - 1) / 2
56     }
57 }

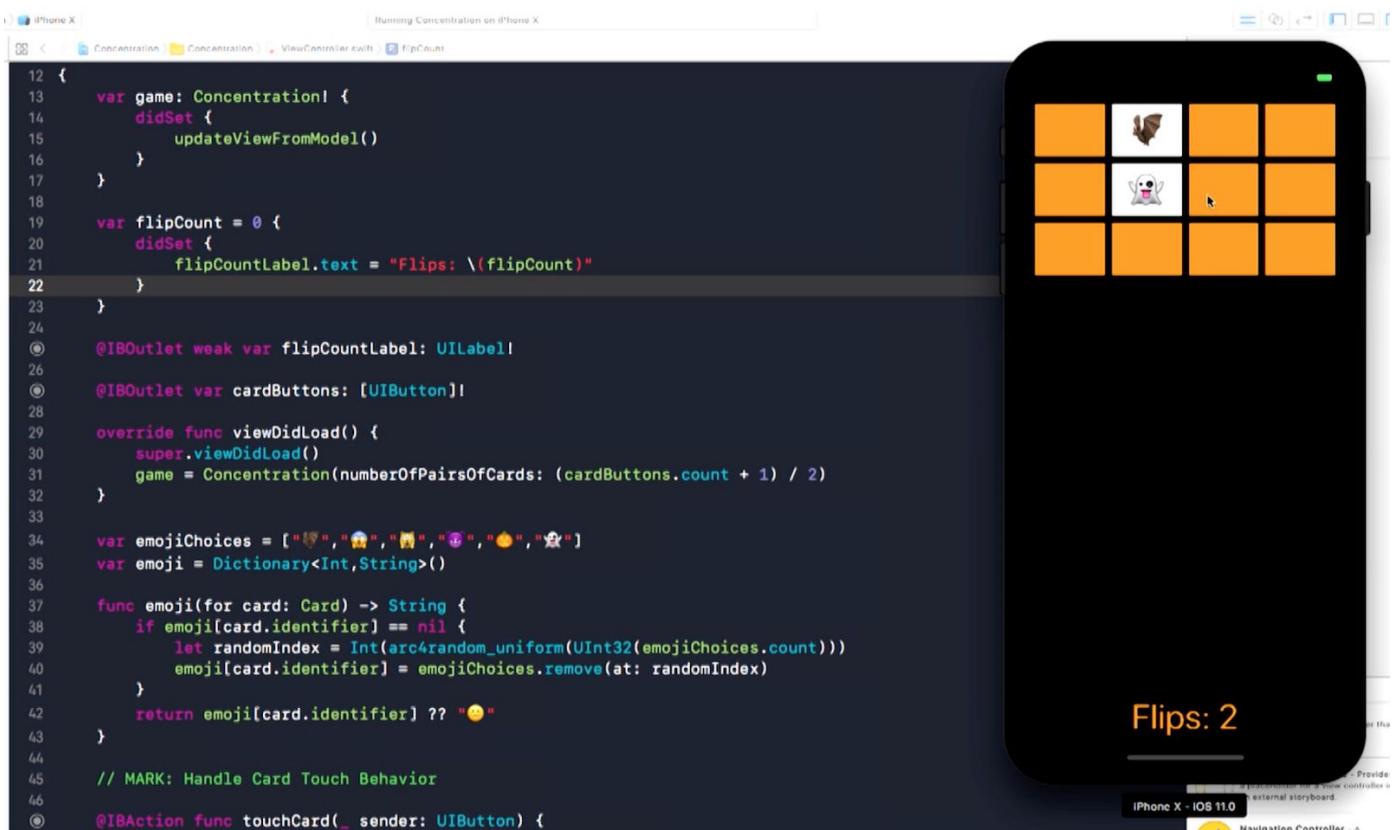
```

На этот раз мне удалось открыть одну карту, а когда я кликнул на второй карте, приложение

вылетело.

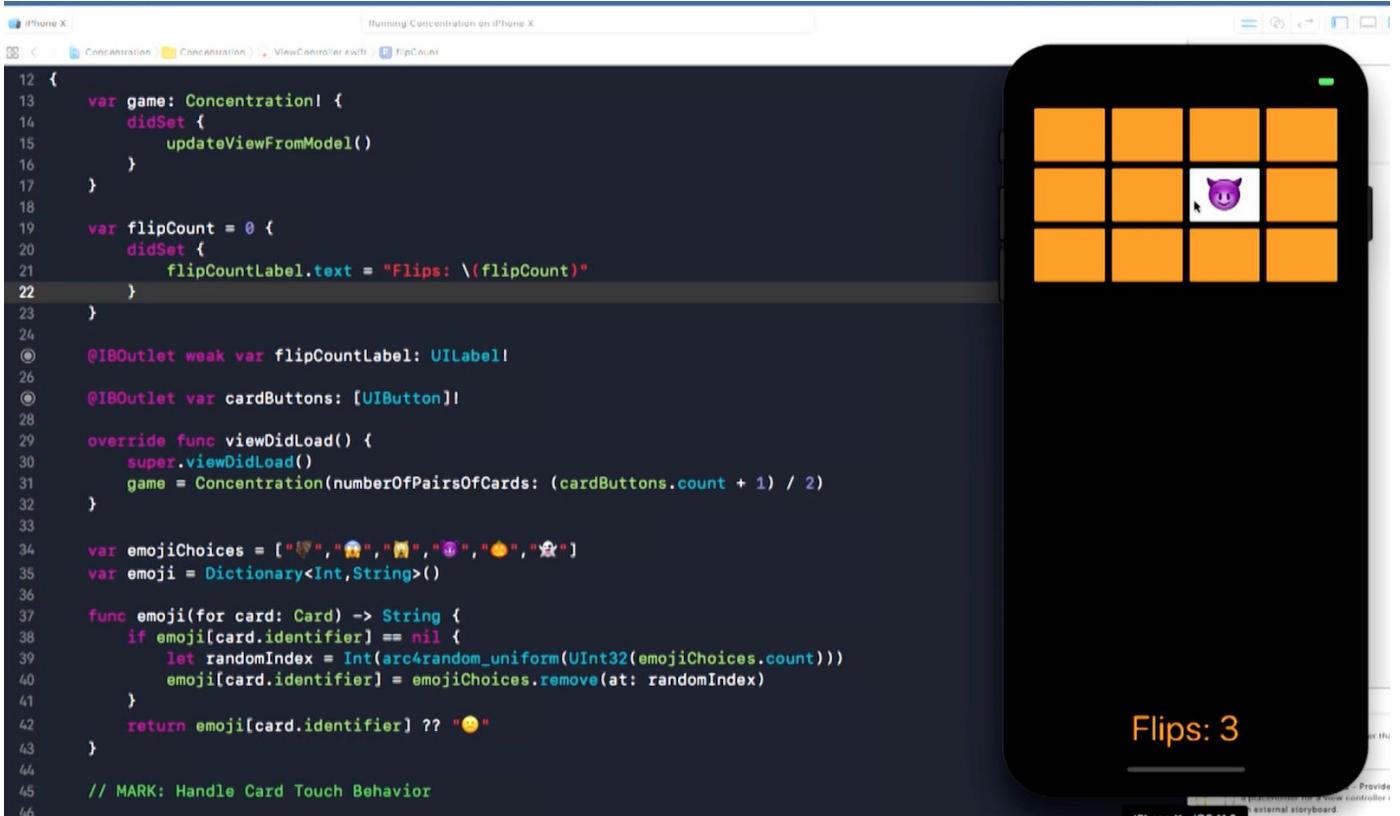
Попробуем ещё разок. Хоть бы какую-то закономерность выявить, что происходит-то.

На этот раз нам удалось открыть две карты:



```
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41     }
42     return emoji[card.identifier] ?? "🂱"
43 }
44
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
```

Кликаем на 3-ей карте:



```
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41     }
42     return emoji[card.identifier] ?? "🂱"
43 }
44
45 // MARK: Handle Card Touch Behavior
46
```

Кликаем на 4-ой карте:

```

12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromViewModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41     }
42     return emoji[card.identifier] ?? "🂱"
43 }
44
45 // MARK: Handle Card Touch Behavior

```

В этот раз вылетает после 4-х карт.

```

12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromViewModel()
16     }
17 }
18
19 var flipCount = 0 {
20     didSet {
21         flipCountLabel.text = "Flips: \(flipCount)"
22     }
23 }
24
25 @IBOutlet weak var flipCountLabel: UILabel!
26
27 @IBOutlet var cardButtons: [UIButton]!
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31     game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1) / 2)
32 }
33
34 var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35 var emoji = Dictionary<Int, String>()
36
37 func emoji(for card: Card) -> String {
38     if emoji[card.identifier] == nil {
39         let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40         emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41     }
42     return emoji[card.identifier] ?? "🂱"
43 }
44
45 // MARK: Handle Card Touch Behavior

```

Есть у кого идеи с чего начать отладку вот этого?

Такое впечатление, что приложение вылетает случайно после нескольких нажатий на карты.

Какие-нибудь идеи? Даже если вы думаете, что это глупо, просто скажите что-нибудь.

- А что если выводить что-нибудь на печать с помощью `print()`?

OK. Предложение использовать печать `print()`.

А если программа большая, не можем же мы в каждой точке каждого метода размещать `print()`.

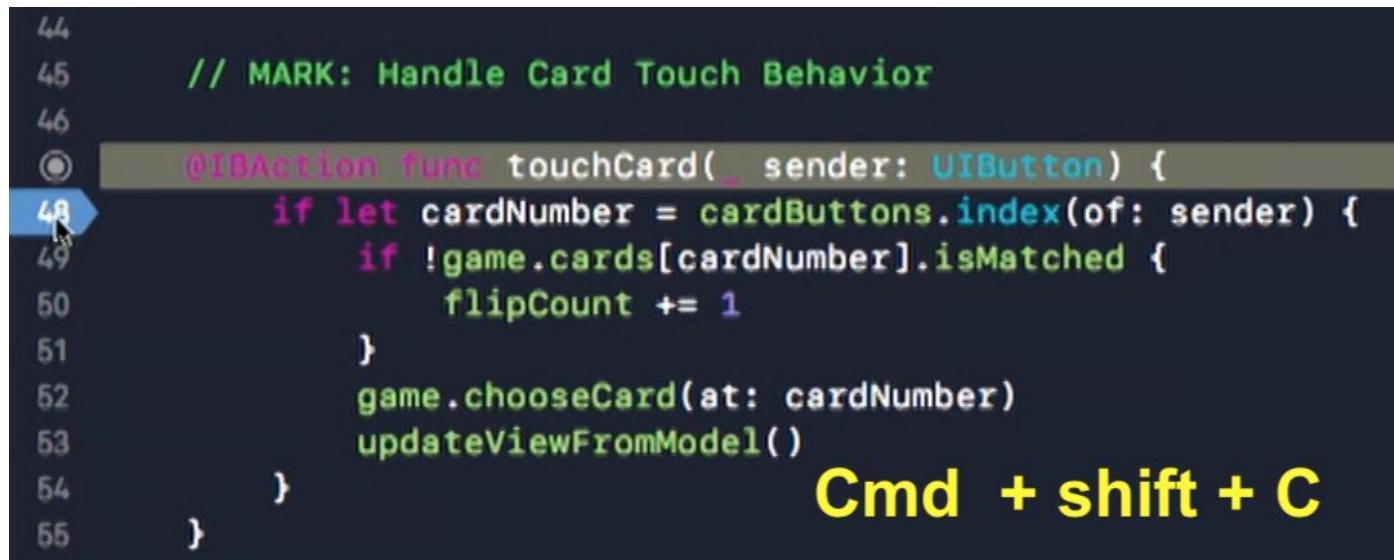
Представьте, для программы из 100000 строк это будет удручающее.

- А что если попытаться понять в какой точке происходит вылет? Это происходит, когда Вы переворачиваете карту. Можно ведь «точку прерывания» **Breakpoint** внутри соответствующего метода разместить.

Бинго! Вы утверждаете, что просто с помощью «точки прерывания» **Breakpoint** в нужном методе мы сможем в случае вылета запустить отладку в пошаговом режиме, и выяснить что происходит.

В **Xcode**, как и в других средах разработки, «точки прерывания» **Breakpoints** работают следующим образом: отладчик ставит на паузу выполнение вашей программы на определенной строке кода.

Переместимся внутрь метода **touchCard**, который, как мы знаем, вызывается в первую очередь при клике на карте. Если мы кликаем на номере строки, то появится такая маленькая синяя стрелка - так выглядит «точка прерывания» **Breakpoint**.



```

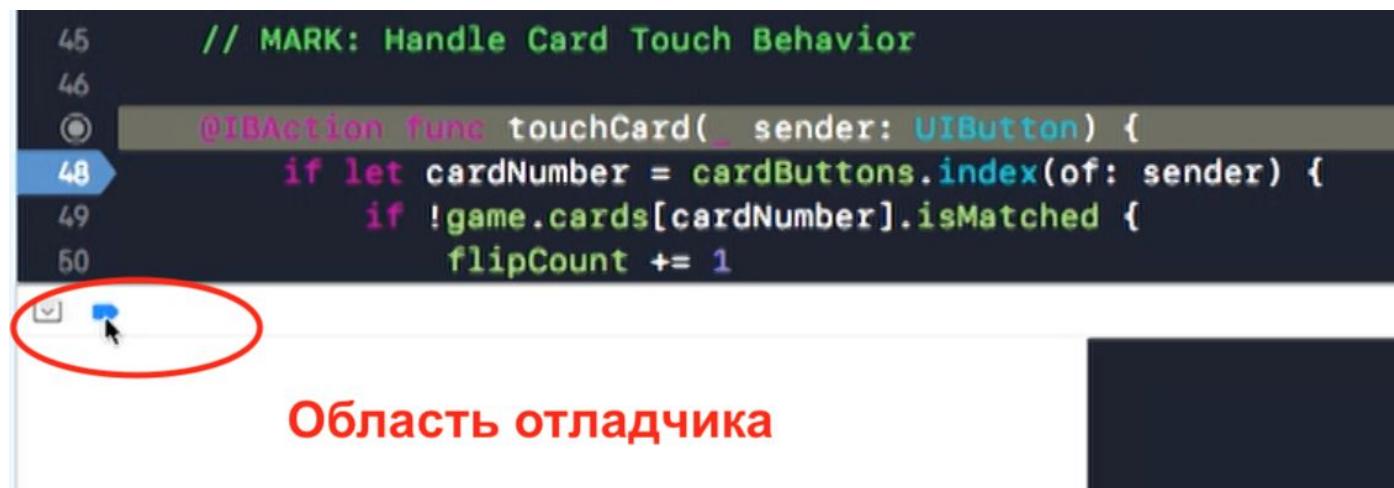
44
45     // MARK: Handle Card Touch Behavior
46
47     @IBAction func touchCard(_ sender: UIButton) {
48         if let cardNumber = cardButtons.index(of: sender) {
49             if !game.cards[cardNumber].isMatched {
50                 flipCount += 1
51             }
52             game.chooseCard(at: cardNumber)
53             updateViewFromModel()
54         }
55     }

```

Cmd + shift + C

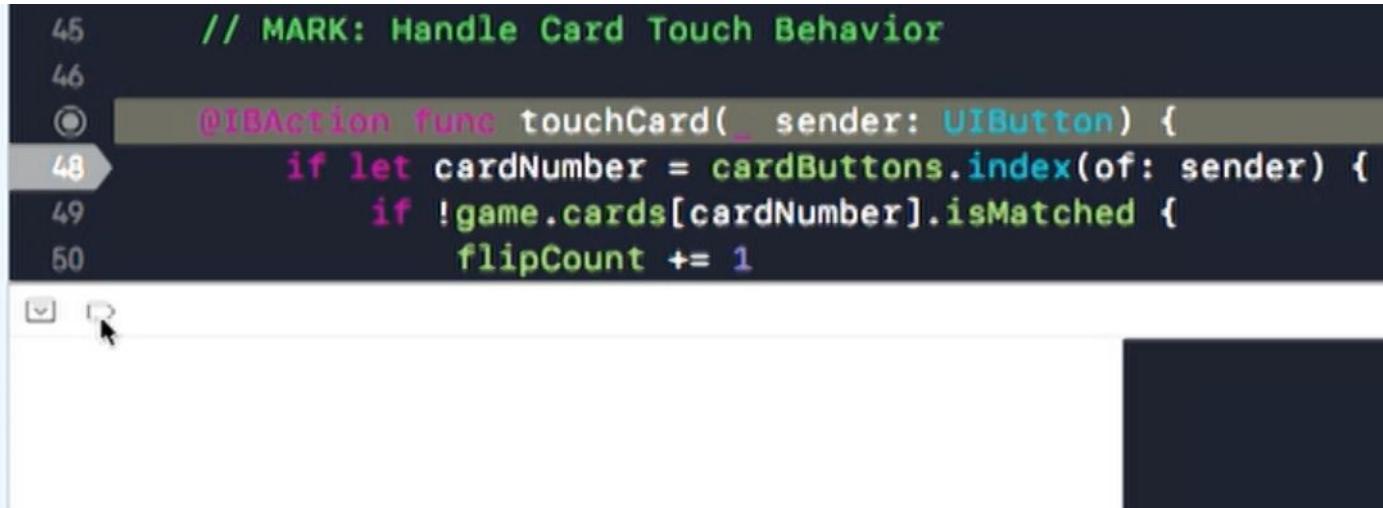
Это инструкция для отладчика остановиться в указанном месте.

Отобразим область отладчика **Debugger Area** с помощью комбинации клавиш **Cmd+Shift+C** (C - консоль **console**).



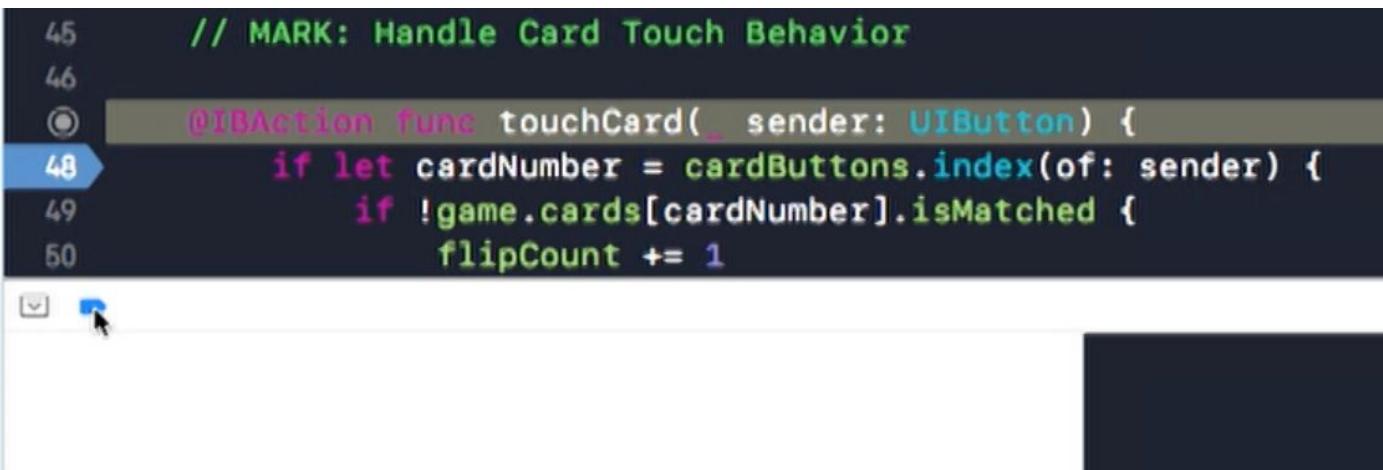
Область отладчика

Видите, в верхней полоске (называется она **Debug Bar**) такая же маленькая синяя стрелка. Если кликнуть на ней, то все «точки прерывания» **Breakpoints** в коде становятся серыми (как и сама кнопка), что означает что все **Breakpoints** дезактивированы.



```
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
```

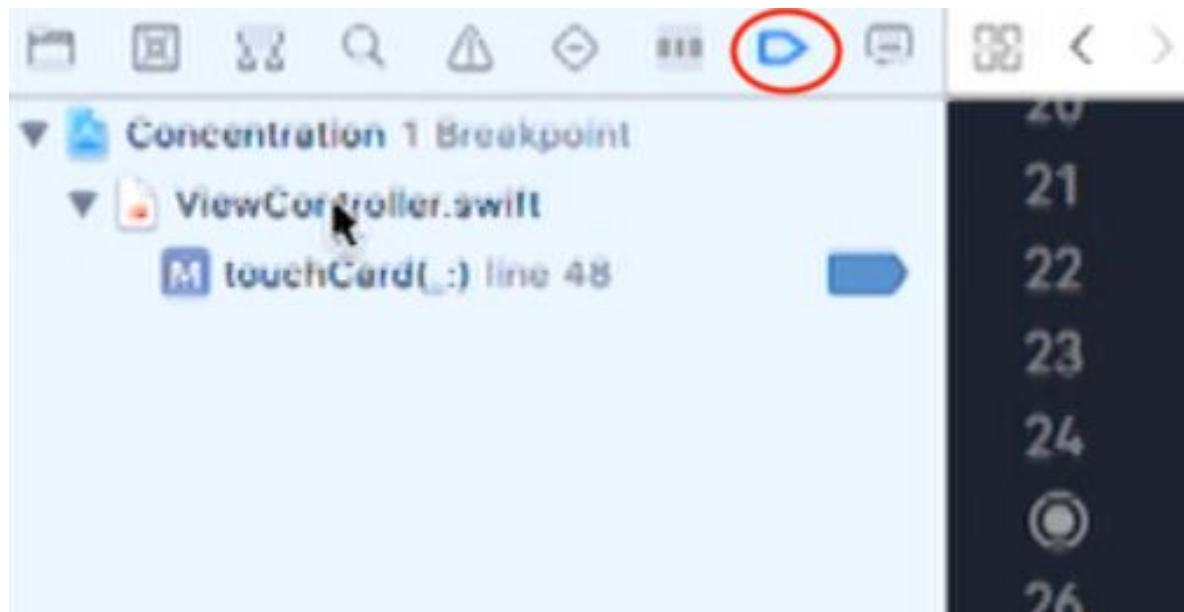
Все “точки прерывания” во всей программе. А вот повторное нажатие активирует только те **Breakpoints**, которые были дезактивированы вот таким вот массовым способом.



```
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
```

Это всё видно по цветам стрелок.

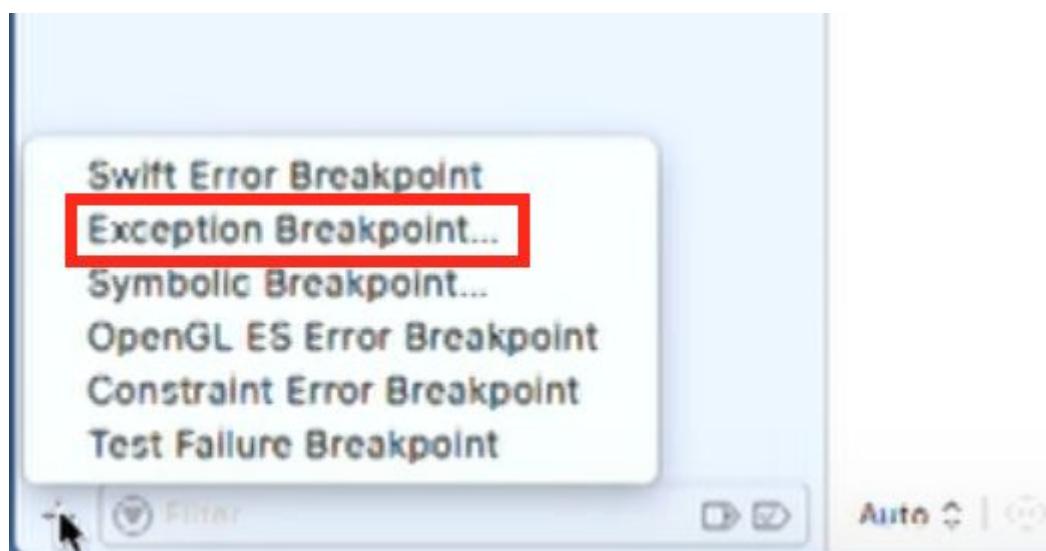
В навигаторе **Navigator**, который вызывается комбинацией клавиш **Cmd+0**, есть панель **Breakpoint Navigator**, на которую можно перейти с помощью комбинаций клавиш **Cmd+8**, имеет пиктограмму с такой же стрелкой.



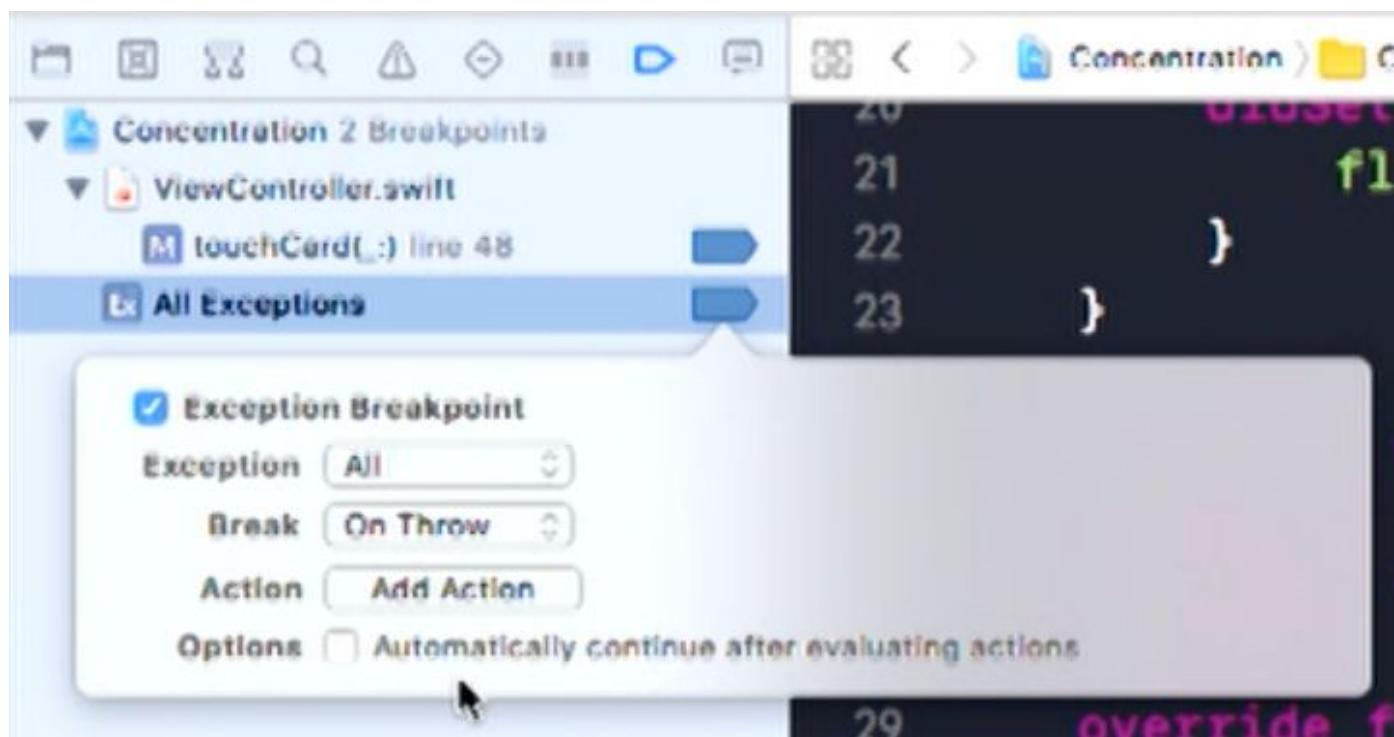
Она показывает вам список всех “точек прерывания” **Breakpoints** во всем проекте **Xcode**.

С “точками прерывания” **Breakpoints** можно производить дополнительные действия. В левом нижнем углу есть значок «+», он позволяет устанавливать некоторые особые точки останова. Например, есть одна под названием «**Exception Breakpoint**».

----- 15 -ая минута лекции -----

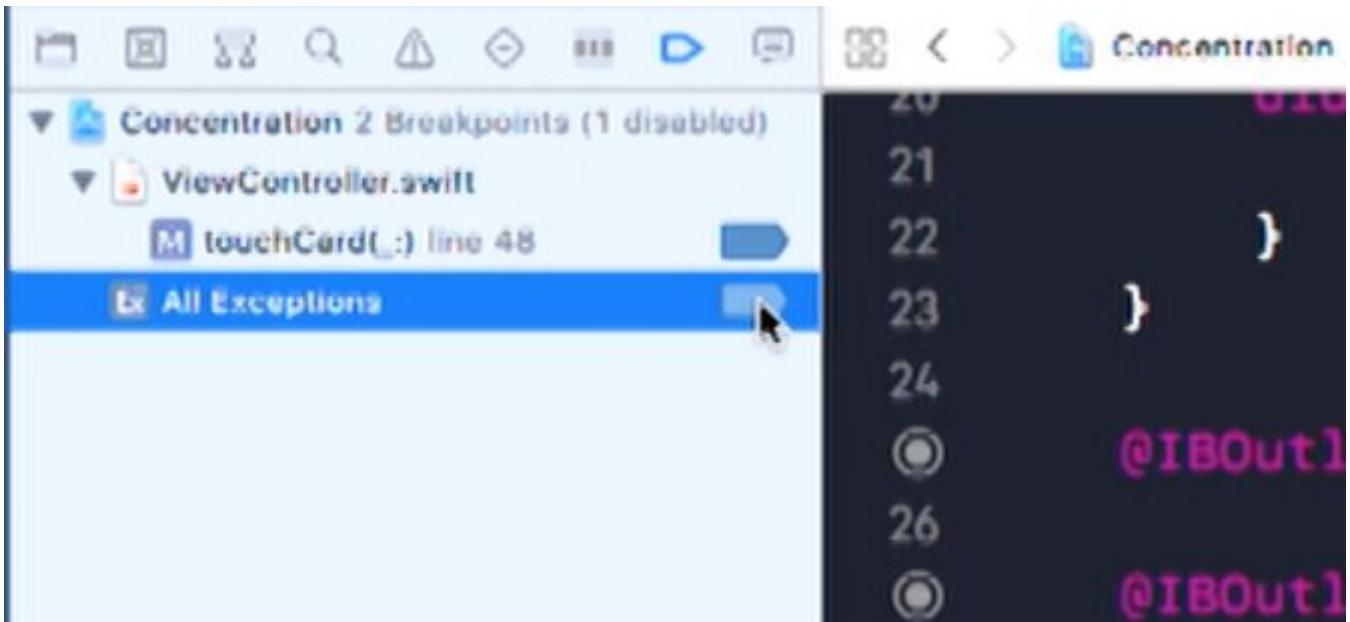


Если я добавлю такую “точку прерывания” **Breakpoint**, это приведет к тому, что каждый раз, когда выбрасывается исключение, вместо вывода на консоль (или вылета) - это уж как повезет, вместо этого отладчик будет остановлен на проблемной строке кода.



Поэтому, если у вас всегда будет такая вот особая “точка прерывания” **Breakpoint** вида «**Exception Breakpoint**», то это вам сильно поможет.

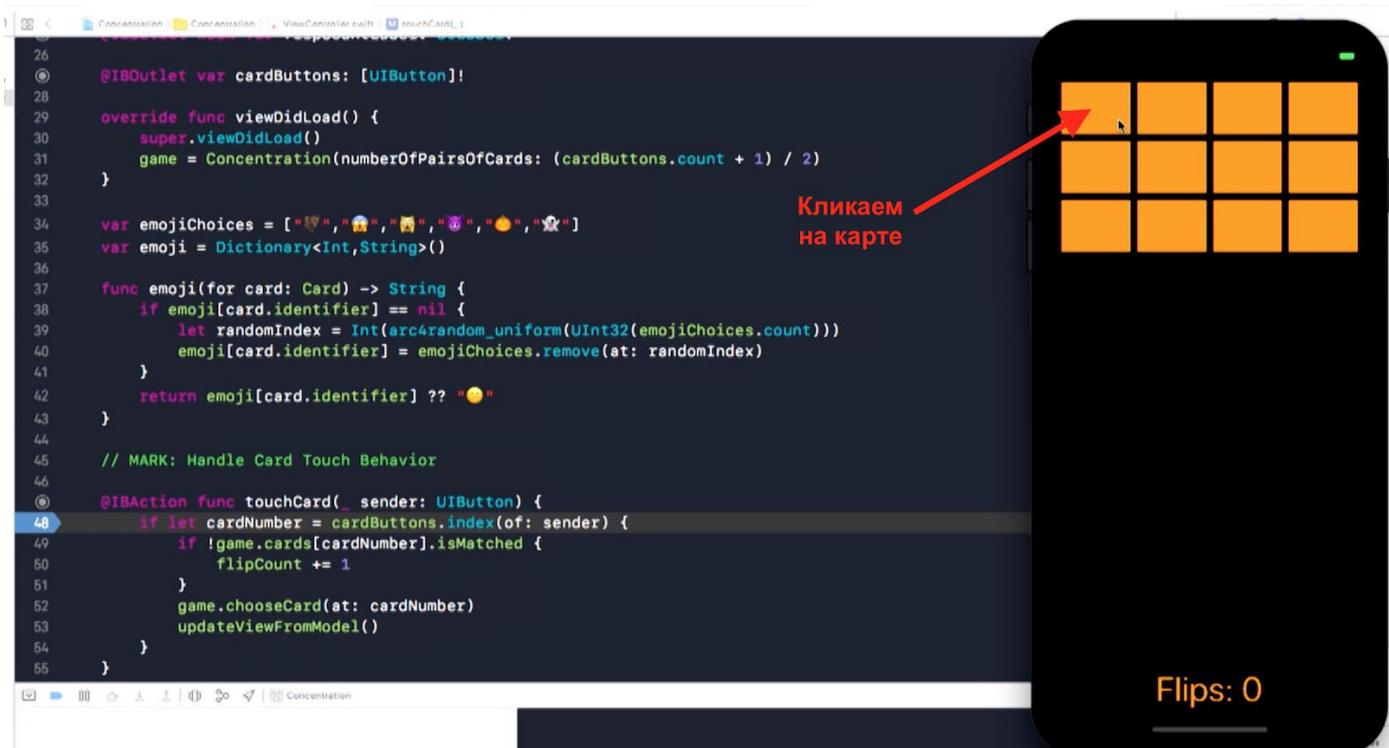
В закладке **Breakpoint Navigator** можно кастомизировать поведение “точек прерывания” **Breakpoints** очень гибко. А отдельные **Breakpoints** можно здесь деактивировать...



(а ещё удалять и перемещать). В одном месте собраны **Breakpoints** всего проекта. Мы видим здесь две “точки прерывания” **Breakpoints** и одна из них не активирована. Это “точка прерывания” **Breakpoint** вида «**Exception Breakpoint**», о которой мы только что говорили.

Зачастую, все что вам потребуется, это в редакторе кода в каждом индивидуальном файле расставить “точки прерывания” **Breakpoints**.

Запустим-ка программу. Жмем на карту.



Мы оказываемся внутри метода **touchCard**, в “точке прерывания” на строке **48**.

```
43    }
44
45    // MARK: Handle Card Touch Behavior
46
47    @IBAction func touchCard(_ sender: UIButton) {
48        if let cardNumber = cardButtons.index(of: sender)
49            if !game.cards[cardNumber].isMatched {
50                flipCount += 1

```

Concentration > Thread 1 > ViewController.touchCard(_)

▶ A **sender** = (UIButton) 0x00007fbc0ca19600
▶ A **self** = (Concentration.ViewController) 0x00007fbc09c16080
L **cardNumber** (Array.Index) 106377751104592

(lldb)

Программа приостановлена в месте, где мы устанавливаем **cardNumber**. Первое, на что стоит обратить внимание - это то, что в отладчике у нас появляется трассировка всего, что хранится в памяти в данный момент. **self** в данном случае относится к **Concentration.ViewController**, есть адрес объекта в куче.

Но и это не все. Можно раскрыть список и увидеть все переменные экземпляра класса **Concentration.ViewController**, хранящиеся в в данном экземпляре указанного класса.

Concentration > (lldb)

▶ A **sender** = (UIButton) 0x00007fd3dd806fd0
▼ A **self** = (Concentration.ViewController) 0x00007fd3dbc056c0
 ▶ **UIKit.UIViewController** (UIViewController)
 ▶ **game** = (Concentration.Concentration?) 0x00006000019a5f80
 flipCount = (Int) 1
 ▶ **flipCountLabel** = (UILabel?) 0x00007fd3dbc0a280
 ▶ **cardButtons** = ([UIButton]?) 12 values
 ▶ **emojiChoices** = ([String]) 9 values
 ▶ **emoji** = ([Int : String]) 0 key/value pairs
L **cardNumber** = (Int) 0

У нас есть переменная **game** - экземпляр класса **Concentration.Concentration**:



```
▶ A sender = (UIButton) 0x00007fd3dd806fd0
▼ A self = (Concentration.ViewController) 0x00007fd3dbc056c0
  ▶ UIKit.UIViewController (UIViewController)
    ▼ game = (Concentration.Concentration?) 0x00006000019a5f80
      ▶ cards = ([Concentration.Card]) 12 values
        indexOfOneAndOnlyFaceUpCard = (Int?) nil
        flipCount = (Int) 1
    ▶ flipCountLabel = (UILabel?) 0x00007fd3dbc0a280
    ▶ cardButtons = ([UIButton]?) 12 values
    ▶ emojiChoices = ([String]) 9 values
    ▶ emoji = ([Int : String]) 0 key/value pairs
  L cardNumber = (Int) 0
```

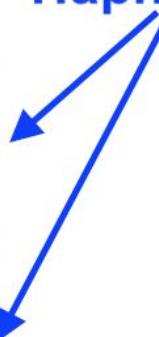
В ней находится массив карт **cards** и переменная **indexOfOneAndOnlyFaceUpCard**.

Можем раскрыть карты и посмотреть на каждую в отдельности:



```
▶ A sender = (UIButton) 0x00007fd3dd806fd0
▼ A self = (Concentration.ViewController) 0x00007fd3dbc056c0
  ▶ UIKit.UIViewController (UIViewController)
    ▼ game = (Concentration.Concentration?) 0x00006000019a5f80
      ▶ cards = ([Concentration.Card]) 12 values
        ▶ [0] (Concentration.Card)
        ▶ [1] (Concentration.Card)
        ▶ [2] (Concentration.Card)
        ▶ [3] (Concentration.Card)
        ▶ [4] (Concentration.Card)
        ▶ [5] (Concentration.Card)
        ▶ [6] (Concentration.Card)
        ▶ [7] (Concentration.Card)
        ▼ [8] (Concentration.Card)
          ▶ isFaceUp = (Bool) false
          ▶ isMatched = (Bool) false
          identifier = (Int) 5
        ▼ [9] (Concentration.Card)
          ▶ isFaceUp = (Bool) false
          ▶ isMatched = (Bool) false
          identifier = (Int) 5
        ▶ [10] (Concentration.Card)
        ▶ [11] (Concentration.Card)
      indexOfOneAndOnlyFaceUpCard = (Int?) nil
```

Парные карты



И вдруг увидеть, что у карты с индексом 9 идентификатор **identifier** равен 5.

Мы видим значения всех переменных, что позволяет нам проверить что-либо или убедиться в чем-то. Это просто замечательно. Например, мы можем увидеть, что у каждой карты есть пара и только одна пара. Вовсе не обязательно для этого использовать **print()**.

Ещё мы можем сделать следующее. Вот правее находится консоль отладчика, сокращенно **LLDB** (**DB** означает **debugger**). Мы можем вывести на печать любой объект, который хотим проинспектировать. Я набираю «**po game**», **po** означает **print object**. Мгновенье и, смотрите-ка,

```
(lldb) po game
↳ Optional<Concentration>
  ↳ some : <Concentration: 0x608000244fb0>

(lldb)
```

Выводится **Optional<Concentration>**, ведь тип переменной **game** - **Optional**.

В данном случае это - **implicitly unwrapped optional**, то есть “неявно развернутое **Optional**”:

```
9 import UIKit
10
11 class ViewController: UIViewController
12 {
13     var game: Concentration!
14     didSet {
15         updateViewFromModel()
16     }
17 }
```

Значение **some** говорит о том, что переменная не равна **nil**. Значит значение у **game** есть, то есть игра **Concentration** установлена.

Еще можно распечатывать командой «**p game**» (**p** означает **print()**), и здесь прямо магия.

```
(Concentration.Concentration?) $R1 = 0x0000608000244fb0 {
    cards = 12 values {
        [0] = (isFaceUp = false, isMatched = false, identifier = 5)
        [1] = (isFaceUp = false, isMatched = false, identifier = 3)
        [2] = (isFaceUp = false, isMatched = false, identifier = 4)
        [3] = (isFaceUp = false, isMatched = false, identifier = 4)
        [4] = (isFaceUp = false, isMatched = false, identifier = 1)
        [5] = (isFaceUp = false, isMatched = false, identifier = 5)
        [6] = (isFaceUp = false, isMatched = false, identifier = 6)
        [7] = (isFaceUp = false, isMatched = false, identifier = 2)
        [8] = (isFaceUp = false, isMatched = false, identifier = 2)
        [9] = (isFaceUp = false, isMatched = false, identifier = 1)
        [10] = (isFaceUp = false, isMatched = false, identifier = 6)
        [11] = (isFaceUp = false, isMatched = false, identifier = 3)
    }
    indexOfTheOneAndOnlyFaceUpCard = nil
}
(lldb)
```

В консоль выводится информация об объекте в удобном для восприятия виде. Информация обо всех переменных в очень хорошо отформатированном виде. Кстати, видно, что у нас все 12 карт со своими свойствами и `indexOfOneAndOnlyFaceUpCard = nil` пока, ведь мы только только “тапнули” на карте.

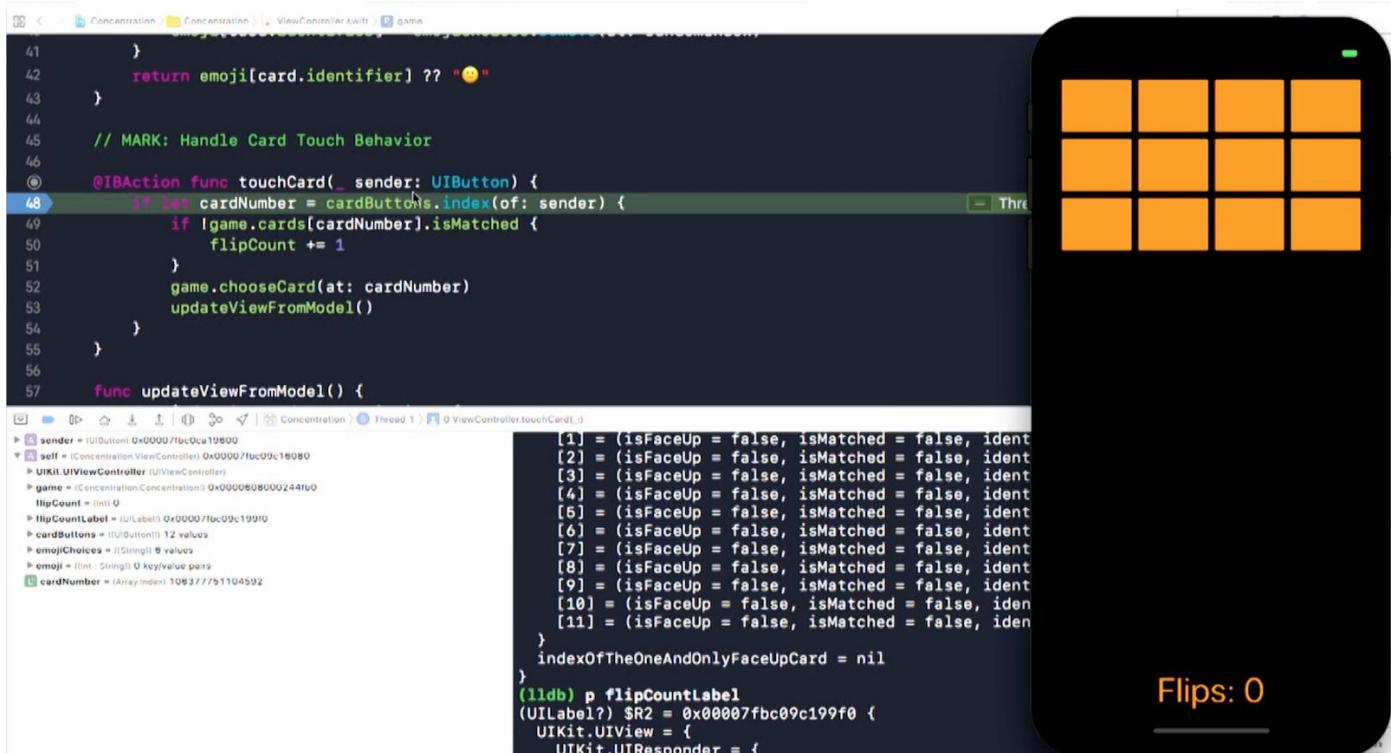
Можно сделать так для любого объекта. Я набираю «`p flipCountLabel`».

```
[1] = (isFaceUp = false, isMatched = false, identifier = 3)
[2] = (isFaceUp = false, isMatched = false, identifier = 4)
[3] = (isFaceUp = false, isMatched = false, identifier = 4)
[4] = (isFaceUp = false, isMatched = false, identifier = 1)
[5] = (isFaceUp = false, isMatched = false, identifier = 5)
[6] = (isFaceUp = false, isMatched = false, identifier = 6)
[7] = (isFaceUp = false, isMatched = false, identifier = 2)
[8] = (isFaceUp = false, isMatched = false, identifier = 2)
[9] = (isFaceUp = false, isMatched = false, identifier = 1)
[10] = (isFaceUp = false, isMatched = false, identifier = 6)
[11] = (isFaceUp = false, isMatched = false, identifier = 3)
}
indexOfTheOneAndOnlyFaceUpCard = nil
}
11db) p flipCountLabel
UILabel?) $R2 = 0x00007fbc09c199f0 {
    UIKit.UIView = {
        UIKit.UIResponder = {
            ObjectiveC.NSObject = {}
        }
    }
}
(11db)
```

Судя по выводу консоли его тип - `Optional<UILabel>` - это `UIView`, у которого есть `UIResponder`...

Суть в том, что “`p`” - очень симпатичная и полезная команда, если нужно что-то вывести в консоль, чтобы это ни было.

Вернемся непосредственно к нашему багу. Мы исследуем проблему, мы кликнули на карте и остановились в этой точке, а теперь пытаемся понять, отчего же эта штука вылетает.



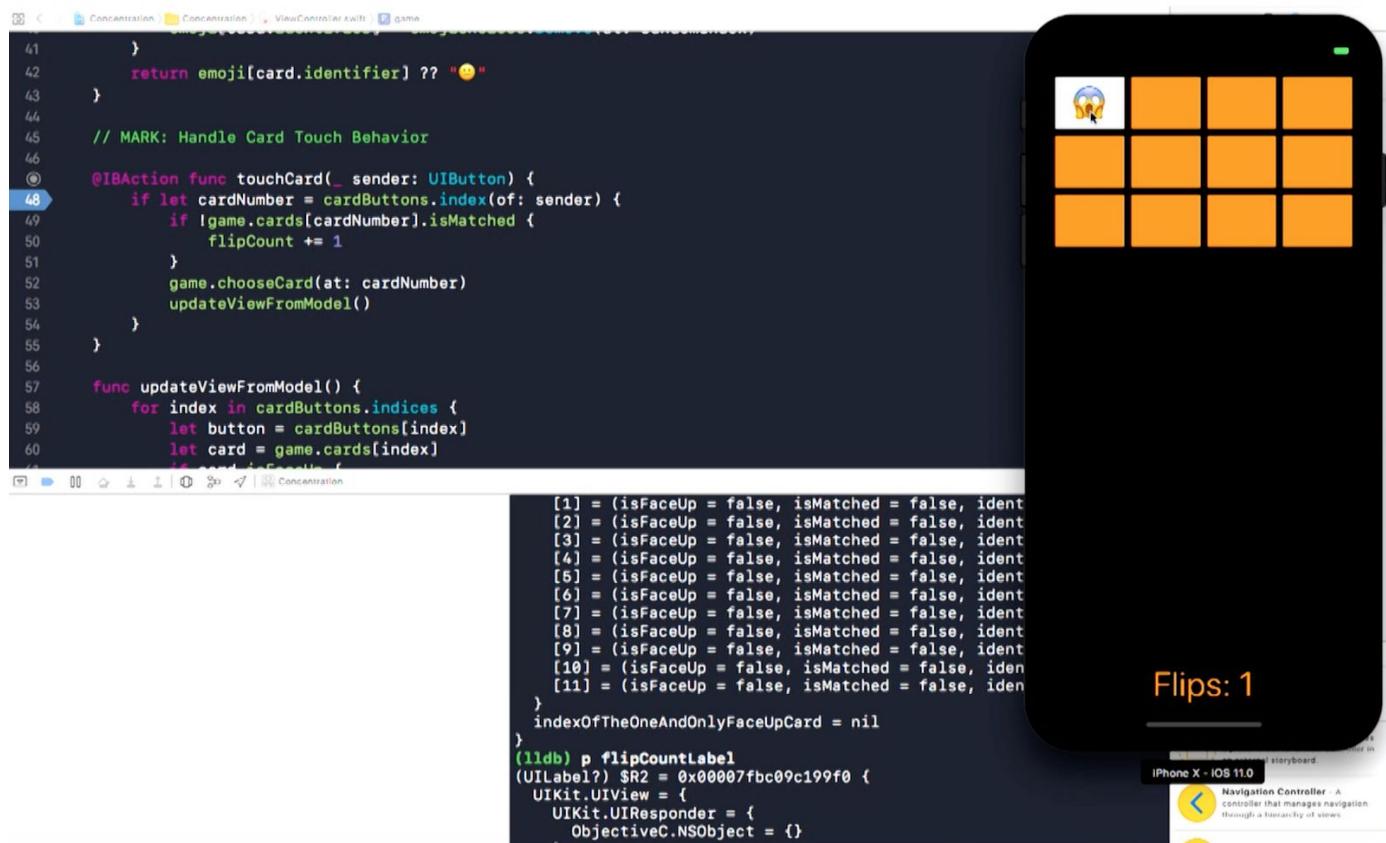
Первое, что можно сделать - это выяснить, на какой строке происходит вылет в данном случае. Ведь исключение не выбрасывается. Воспользуемся возможностями навигации внутри отладчика и взглянем на кнопочки в отладочной полоске **Debug Bar**. У нас здесь несколько возможных опций:

```
func updateViewFromModel() {
    for index in cardButtons.indices {
        let button = cardButtons[index]
        let card = game.cards[index]
        if card.isFaceUp {
            if card.isFaceUp {
                if card.isFaceUp {
                    if card.isFaceUp {
                        if card.isFaceUp {
                            if card.isFaceUp {
                                if card.isFaceUp {
                                    if card.isFaceUp {
                                        if card.isFaceUp {
                                            if card.isFaceUp {
                                                if card.isFaceUp {
                                                    if card.isFaceUp {
                                                        if card.isFaceUp {
                                                            if card.isFaceUp {
                                                                if card.isFaceUp {
                                                                    if card.isFaceUp {
................................................................
```

The screenshot shows the Xcode interface during debugging. The top part displays Swift code for updating a view from a model. Below the code is a toolbar with several icons, including a step-over icon (represented by a right-pointing arrow) which is highlighted with a red box. The bottom part shows a stack trace in the Debug Navigator, listing the current stack frames and their memory addresses.

Там есть опция **Continue program execution** - означает это следующее: мы остановились в “точке прерывания” **Breakpoint**, а теперь продолжим до следующей точке прерывания” **Breakpoint**, а, если больше не встречаются **Breakpoints**, то - до конца.

Испытаем и кликаем на кнопке **Continue program execution**.



В этот раз без вылета. Программа не остановлена.

Что ж, кликнем ещё раз на карте:

```

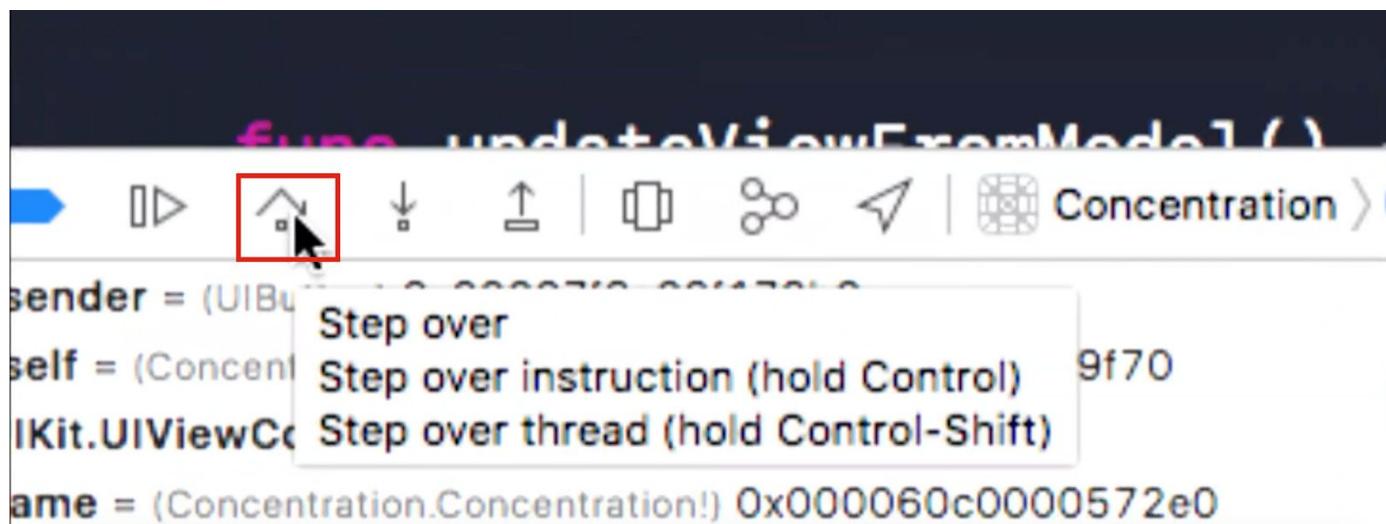
41 }
42     return emoji[card.identifier] ?? "😊"
43 }
44
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
51         }
52         game.chooseCard(at: cardNumber)
53         updateViewFromModel()
54     }
55 }
56
57 func updateViewFromModel() {
58     for index in cardButtons.indices {
59         let button = cardButtons[index]
60         let card = game.cards[index]
61         if card.isFaceUp {
62             button.setTitle(emoji(for: card), for: .normal)
63             button.backgroundColor = #fff
64         } else {
65             button.setTitle("", for: .normal)
66             button.backgroundColor = #eee
67         }
68         if card.isMatched {
69             button.setTitle("?", for: .normal)
70             button.backgroundColor = #ccc
71         }
72     }
73 }
74
75 var flipCount = 0
76
77 var emoji: [Card] = [
78     [1] = (isFaceUp = false, isMatched = false, identifier: "one"),
79     [2] = (isFaceUp = false, isMatched = false, identifier: "two"),
80     [3] = (isFaceUp = false, isMatched = false, identifier: "three"),
81     [4] = (isFaceUp = false, isMatched = false, identifier: "four"),
82     [5] = (isFaceUp = false, isMatched = false, identifier: "five"),
83     [6] = (isFaceUp = false, isMatched = false, identifier: "six"),
84     [7] = (isFaceUp = false, isMatched = false, identifier: "seven"),
85     [8] = (isFaceUp = false, isMatched = false, identifier: "eight"),
86     [9] = (isFaceUp = false, isMatched = false, identifier: "nine"),
87     [10] = (isFaceUp = false, isMatched = false, identifier: "ten"),
88     [11] = (isFaceUp = false, isMatched = false, identifier: "jack"),
89     [12] = (isFaceUp = false, isMatched = false, identifier: "queen"),
90     [13] = (isFaceUp = false, isMatched = false, identifier: "king"),
91     [14] = (isFaceUp = false, isMatched = false, identifier: "ace")
92 ]
93
94 var indexOfTheOneAndOnlyFaceUpCard: Int?
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
986
987
988
989
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
21
```

первой строки метода **touchCard**:

```
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
51         }
52         game.chooseCard(at: cardNumber)
53         updateViewFromModel()
54     }
55 }
```

Как же нам получить больше подробностей?

Обратим внимание на навигацию внутри самого отладчика:



Давайте в “точке прерывания” **Breakpoint** вместо того чтобы продолжить, выберем вариант «**Step Over**» -пошаговое выполнение. В подсказке видно, что это вариант имеет три опции.

И вот мы идем по шагам...

```
45 // MARK: Handle Card Touch Behavior
46
47 @IBAction func touchCard(_ sender: UIButton) {
48     if let cardNumber = cardButtons.index(of: sender) {
49         if !game.cards[cardNumber].isMatched {
50             flipCount += 1
51         }
52         game.chooseCard(at: cardNumber)
53         updateViewFromModel()
54     }
55 }
```

... идем на следующую строку...

```
①      @IBAction func touchCard(_ sender: UIButton) {  
48        if let cardNumber = cardButtons.index(of: sender) {  
49            if !game.cards[cardNumber].isMatched {  
50                flipCount += 1  
51            }  
52            game.chooseCard(at: cardNumber)  
53            updateViewFromModel()  
54        }  
55    }  
56
```

... следующая строка...

```
45    // MARK: Handle Card Touch Behavior  
46  
①    @IBAction func touchCard(_ sender: UIButton) {  
48        if let cardNumber = cardButtons.index(of: sender) {  
49            if !game.cards[cardNumber].isMatched {  
50                flipCount += 1  
51            }  
52            game.chooseCard(at: cardNumber)  
53            updateViewFromModel()  
54        }  
55    }  
56
```

... надеюсь, время аварийного завершения приложения приближается, и мы смотрим на каком шаге споткнемся..

```
45    // MARK: Handle Card Touch Behavior  
46  
①    @IBAction func touchCard(_ sender: UIButton) {  
48        if let cardNumber = cardButtons.index(of: sender) {  
49            if !game.cards[cardNumber].isMatched {  
50                flipCount += 1  
51            }  
52            game.chooseCard(at: cardNumber)  
53            updateViewFromModel()  
54        }  
55    }  
56
```

Однако все прошло гладко. Пошаговая отладка не помогла. Ещё идеи? Что дальше?

- Установить “точку прерывания” **Breakpoint** внутри метода `game.chooseCard(...)`.

Действительно метод `touchCard` вызывает другие методы внутри себя. Есть ещё метод `updateViewFromModel ()`. Вылет может происходить внутри методов `game.chooseCard(...)` и `updateViewFromModel ()`. Можно, конечно, пойти и поставить “точку прерывания” **Breakpoint** в методе `updateViewFromModel ()`:

```

    @IBAction func touchCard(_ sender: UIButton) {
48        if let cardNumber = cardButtons.index(of: sender) {
49            if !game.cards[cardNumber].isMatched {
50                flipCount += 1
51            }
52            game.chooseCard(at: cardNumber)
53            updateViewModel()
54        }
55    }
56
57    func updateViewModel() {
58        for index in cardButtons.indices {
59            let button = cardButtons[index]

```

Можно поставить “точку прерывания” **Breakpoint** в методе `chooseCard(...)`:

```

39        let randomIndex =
40            Int(arc4random_uniform(UInt32(emojiChoices.count)))
41            emoji[card.identifier] =
42            emojiChoices.remove(at: randomIndex)
43        }
44        return emoji[card.identifier] ?? "🂡"
45
46        // MARK: Handle Card Touch Behavior
47
48        @IBAction func touchCard(_ sender: UIButton) {
49            if let cardNumber = cardButtons.index(of: sender) {
50                if !game.cards[cardNumber].isMatched {
51                    flipCount += 1
52                }
53                game.chooseCard(at: cardNumber)
54            }
55        }

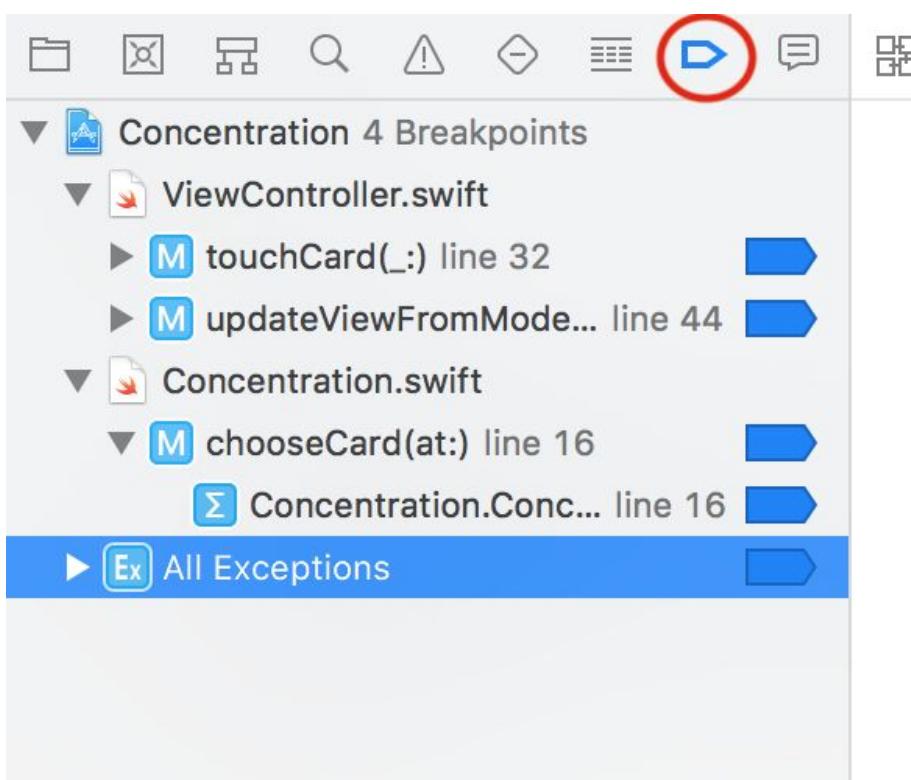
```

```

11    class Concentration
12    {
13        var cards = [Card]()
14
15        var indexOfTheOneAndOnlyFaceUpCard: Int?
16
17        func chooseCard(at index: Int) {
18            if !cards[index].isMatched {
19                if let matchIndex =
20                    indexOfTheOneAndOnlyFaceUpCard, index != matchIndex {
21                        if cards[index].identifier ==
22                            cards[matchIndex].identifier {
23                            cards[index].isMatched = true
24                            cards[matchIndex].isMatched = true
25                        }
26                        cards[index].isFaceUp = true
27                        indexOfTheOneAndOnlyFaceUpCard = nil
28                    } else {
29                        for flipDownIndex in cards.indices {
30                            cards[flipDownIndex].isFaceUp =

```

У нас будет 3 “точки прерывания” **Breakpoints**, что видно во вкладке Навигатора **Debug Navigator**:



Но мы пойдем другим путём. Удаляем две добавленные “точки прерывания” **Breakpoints**, Воспользуемся опцией отладчика «**Step Into**».

Запускаем приложение.

Кликаем на какой-нибудь карте и попадаем на нашу “точку прерывания” **Breakpoint**. Затем идем по шагам с помощью уже знакомой нам опции «**Step Over**» - пошаговое выполнение.

Доходим до **chooseCard** и воспользуемся опцией «**Step Into**» - заход внутрь метода или функции.



Попадаем внутрь метода **chooseCard** в самое начало:

```
8 //
9
10 import Foundation
11
12 class Concentration {
13
14     var cards = [Card]()
15
16     var indexOfTheOneAndOnlyFaceUpCard: Int?
17
18     func chooseCard(at index: Int) {
19         if !cards[index].isMatched {
20             if let matchIndex = indexOfTheOneAndOnlyFaceUpCard, index != matchIndex {
21                 if cards[index].identifier == cards[matchIndex].identifier {
22                     cards[index].isMatched = true
23                     cards[matchIndex].isMatched = true
24                 }
25                 cards[index].isFaceUp = true
26                 indexOfTheOneAndOnlyFaceUpCard = nil
27             } else {
28                 for flipDownIndex in cards.indices {
29                     cards[flipDownIndex].isFaceUp = false
30                 }
31             }
32         }
33     }
34 }
```

Можно заодно и код почитать, убедиться, что всё в порядке. Добавить дополнительные “точки прерывания” **Breakpoints**, если заметим что-то подозрительное.

Продвигаемся дальше по этому методе с помощью опции «**Step Over**» - пошаговое выполнение.

В циклах можно застрять надолго. Но ничего не случилось в методе **chooseCard**.

Идем внутрь метода **updateViewModel**, опять с помощью опции «**Step Into**» - заход внутрь метода или функции:

```
57     func updateViewFromModel() {
58         for index in cardButtons.indices {
59             let button = cardButtons[index]
60             let card = game.cards[index]
61             if card.isFaceUp {
62                 button.setTitle(emoji(for: card), for: UIControlState.normal)
63                 button.backgroundColor = □
64             } else {
65                 button.setTitle("", for: UIControlState.normal)
66                 button.backgroundColor = card.isMatched ? □ : □
67             }
68         }
69     }
```

Продолжаем продвигаться по методу `updateViewFromModel` с опции «Step Over» -пошаговое выполнение. Вот как раз цикл в `updateViewFromModel` - придется потерпеть. Мы прошли по всем индексам массива `cardButtons.indices`.

Никто не замечает ничего подозрительного?

Скажите, если вы увидите что-то странное в этом методе. Ну, кто-нибудь?

```
func updateViewFromModel() {
    for index in cardButtons.indices {
        let button = cardButtons[index]
        let card = game.cards[index]
        if card.isFaceUp {
            button.setTitle(emoji(for: card), for: UIControlState.normal)
            button.backgroundColor = □
        } else {
            button.setTitle("", for: UIControlState.normal)
            button.backgroundColor = card.isMatched ? □ : □
        }
        if (flipCount > Int(arc4random_uniform(UInt32(128)))) { exit(-1); }
    }
}
```

Вот же она: худшая строчка кода в мире.

Что это такое?

Мы выводим `exit (-1)` из приложения с кодом **-1** как только число переворотов карт `flipCount` превысит случайное целое число, которое генерируется в диапазоне **0 - 128**?

Эта строка кода вообще не имеет никакого смысла!

Именно поэтому отладчик не дает нам никакой информации на консоле. Зато она помогла нам разобраться, как пользоваться отладчиком.

Избавимся от неё ...

```

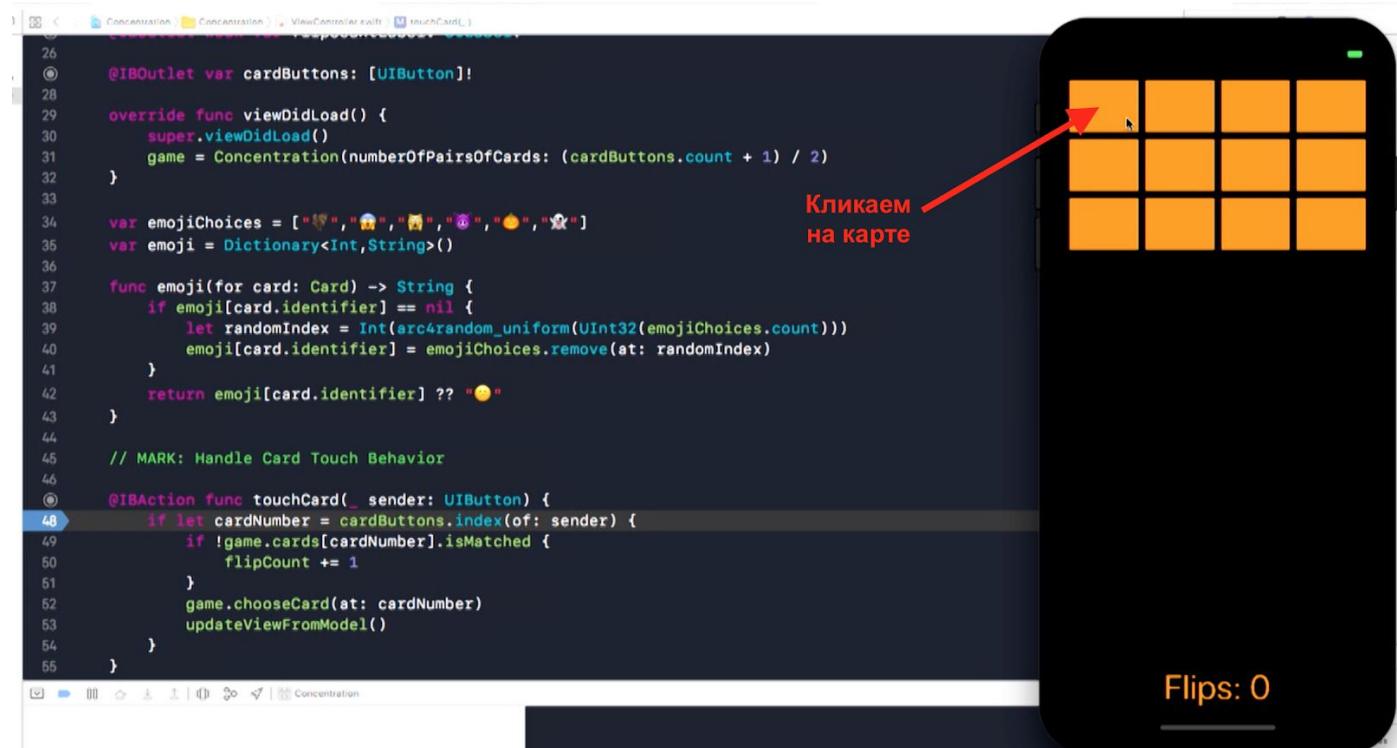
func updateViewFromModel() {
    for index in cardButtons.indices {
        let button = cardButtons[index]
        let card = game.cards[index]
        if card.isFaceUp {
            button.setTitle(emoji(for: card), for: UIControlState.normal)
            button.backgroundColor = white
        } else {
            button.setTitle("", for: UIControlState.normal)
            button.backgroundColor = card.isMatched ? green : orange
        }
    }
}

```

... и снова запустим приложение.

“Точку прерывания” **Breakpoint** оставим на всякий случай.

Кликаем на карте:



И снова попадаем на нашу “точку прерывания” **Breakpoint**:

```

33
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices
40                 emoji[card.identifier] = emojiChoices.remove(at: randomIndex
41             })
42             return emoji[card.identifier] ?? "🂴"
43         }
44
45     // MARK: Handle Card Touch Behavior
46
47     @IBAction func touchCard(_ sender: UIButton) {
48         if let cardNumber = cardButtons.index(of: sender) {
49             if !game.cards[cardNumber].isMatched {
50                 flipCount += 1
51             }
52             game.chooseCard(at: cardNumber)
53             updateViewFromModel()

```

Concentration > Thread 1 > ViewController.touchCard()

На этот раз мы делаем нашу “точку прерывания” **Breakpoint** неактивной и жмем на кнопке “Continue”:

```

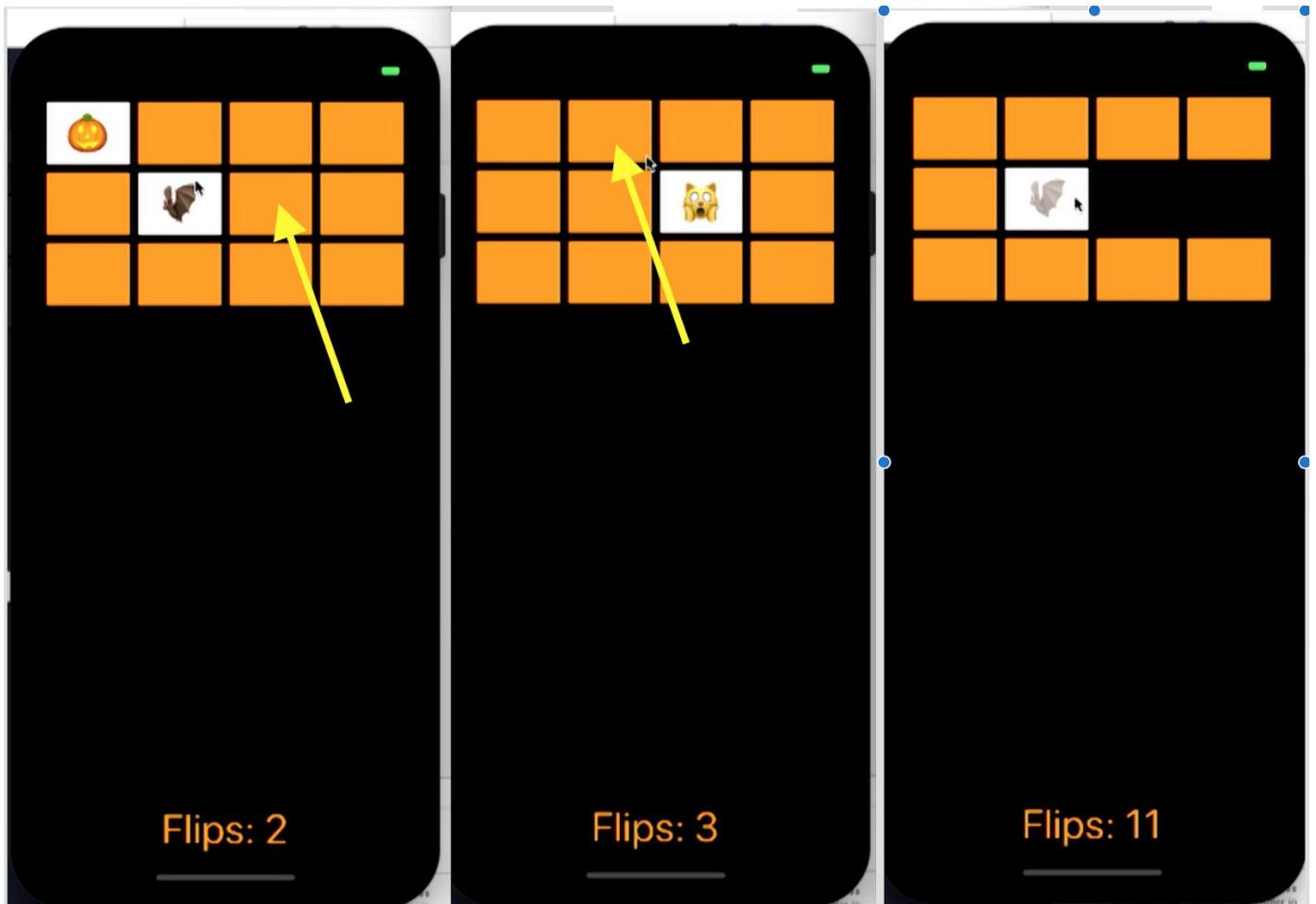
44
45     // MARK: Handle Card Touch Behavior
46
47     @IBAction func touchCard(_ sender: UIButton) {
48         if let cardNumber = cardButtons.index(of: sender) {
49             if !game.cards[cardNumber].isMatched {
50                 flipCount += 1
51             }
52             game.chooseCard(at: cardNumber)
53             updateViewFromModel()

```

Concentration > Thread 1 > ViewController.touchCard()

Сыграем в игру **Concentration**. Посмотрим, нет ли чего странного. В этот раз она работает значительно лучше:

----- 25 -ая минута лекции -----



И кстати, я сейчас не стараюсь; уверяю вас, я не настолько плохо играю в эту игру. Похоже, проблема устранена. Ну одну-то проблему мы точно устранили - теперь в нашем коде нет этого неуместного **exit** кода.

Вопросы по использованию отладчика?

Всем понятна общая идея применения отладчика?

Она довольно проста.

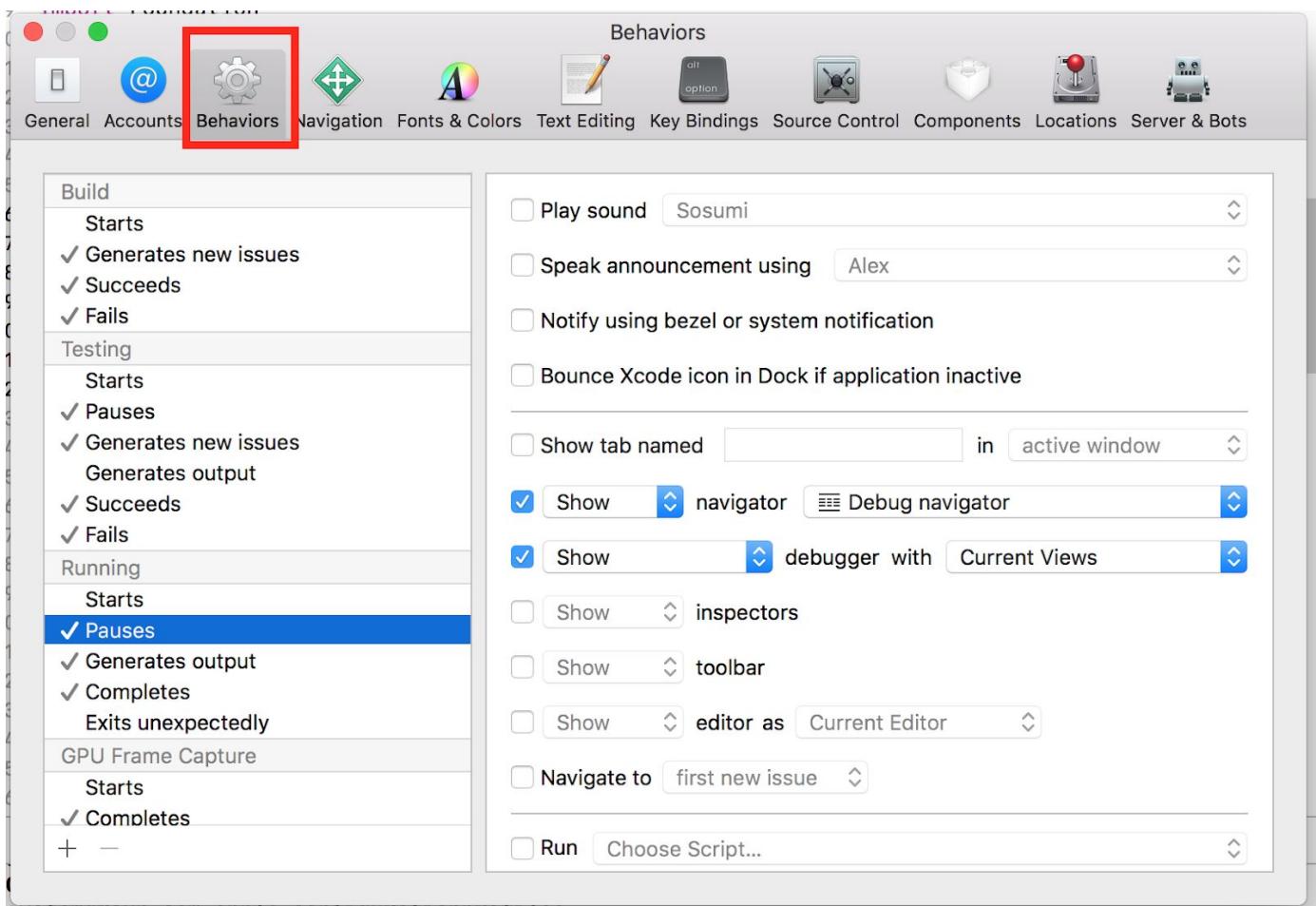
Перемещение по иерархии объектов в памяти, а затем использование этих вариантов пошаговых команд отладчика для “прохода” по коду.

Всем понятно?

Ещё хочу показать вам, что отладчик не так прост, как может показаться по его интерфейсу в нижней части.



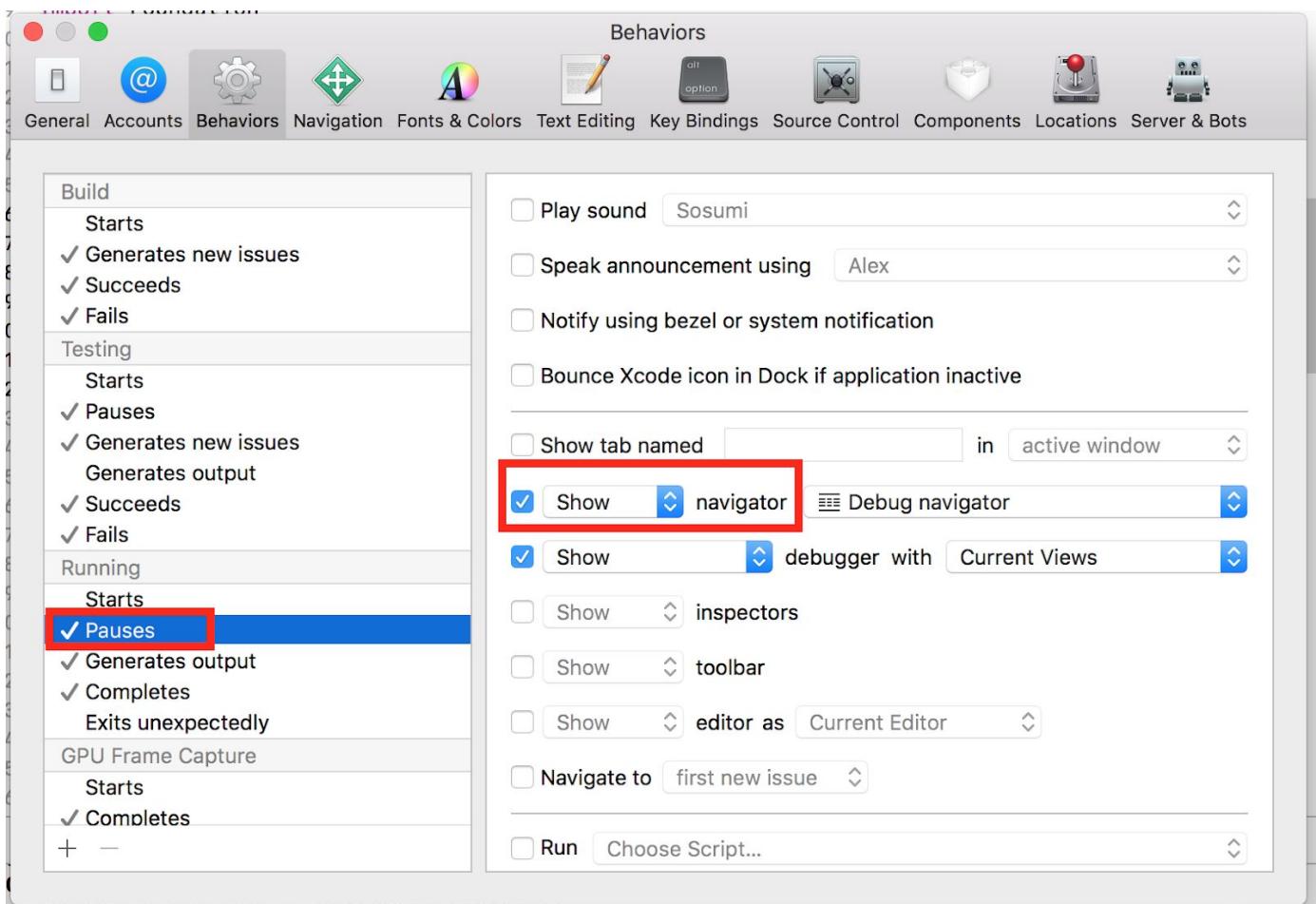
Оказывается есть полезная и крутая штука, в настройках **Xcode -> Preferences -> Behaviours**, которые весьма обширны и очень полезны:



svstemarou.com.apple.configurationprototypes

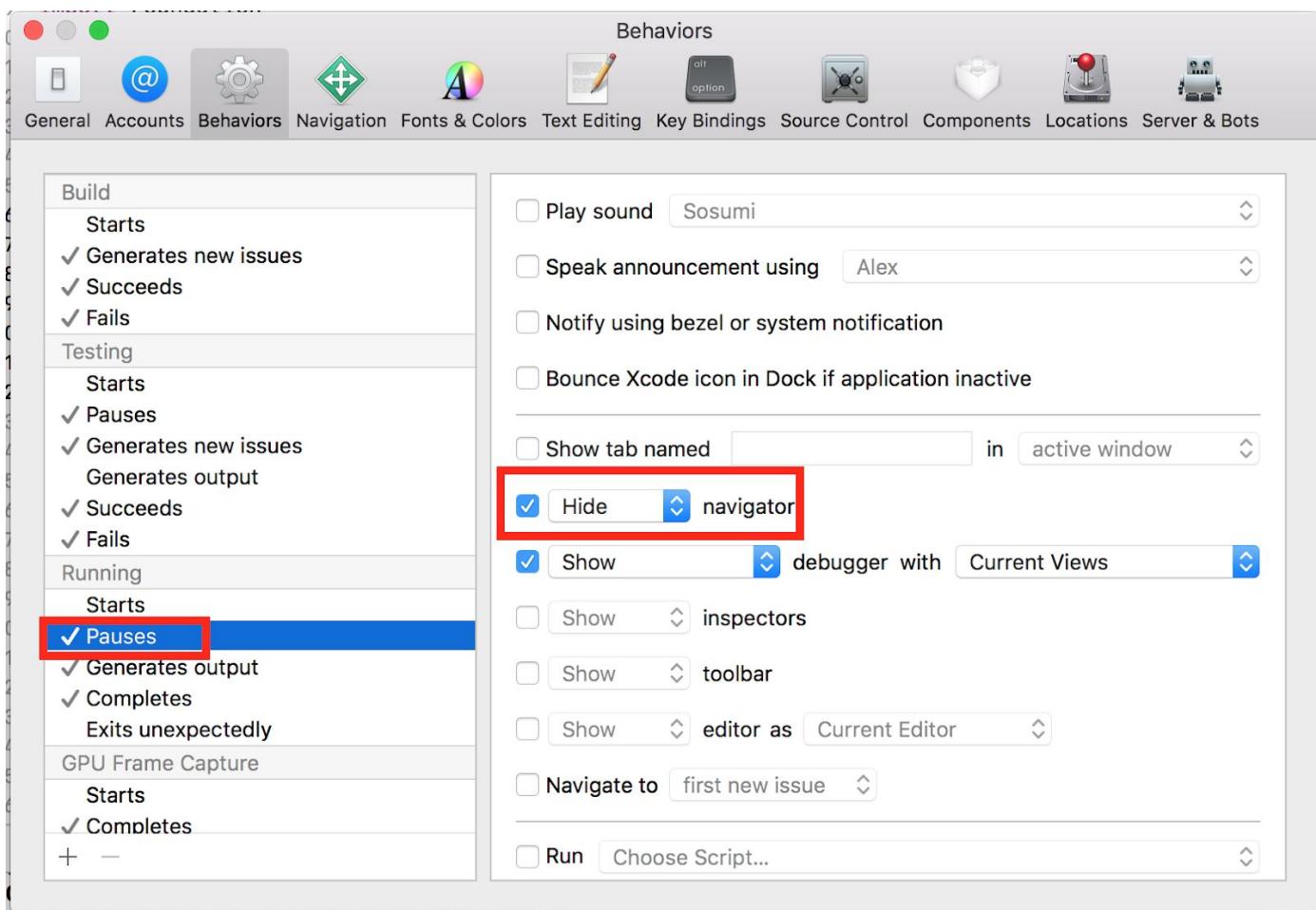
Behaviours (“поведения”) - управляют разными аспектами запуска вашего приложения и дают возможность создавать кастомные конфигурации для определённых событий. Можно сказать: «Если происходит такое-то событие, **Xcode**, сделай то-то и то-то».

Например, если мы приостанавливаем (**pause**) выполнение приложения где-то внутри этого приложения с помощью “точки прерывания” **Breakpoint**, то у нас есть опция «**Show**» показа навигатора **Navigator**:



systematique.com.apple.configurationprototypes

А если изменить опцию «Show» на «Hide» navigator...



systematique.com.apple.configurationprototypes

... то смотрите, что будет происходить в “точке прерывания” **Breakpoint**.

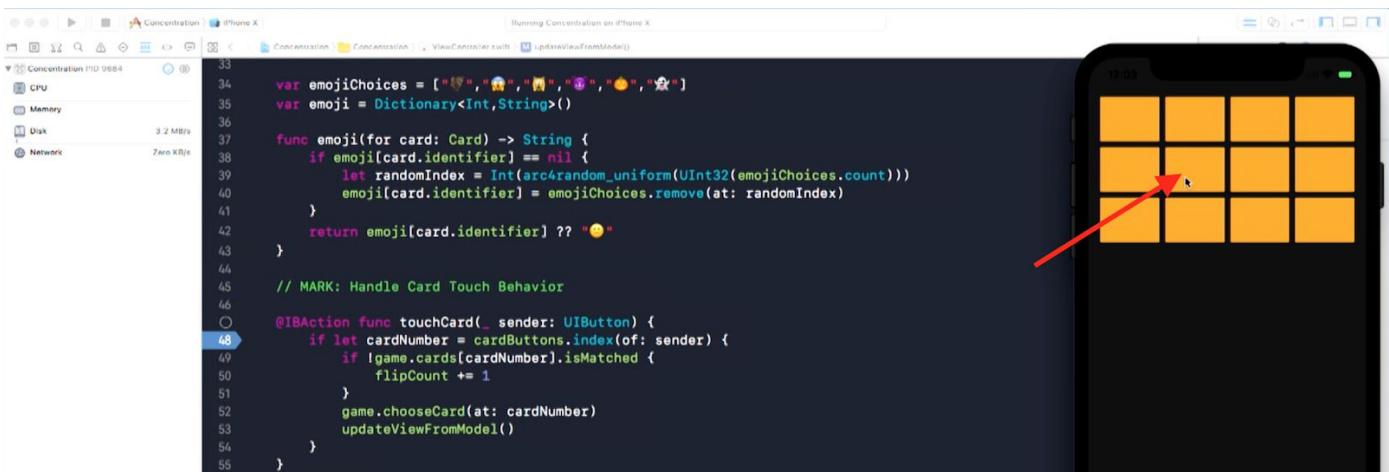
```

33
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "🂱"
43     }
44
45     // MARK: Handle Card Touch Behavior
46
47
48     @IBAction func touchCard(_ sender: UIButton) {
49         if let cardNumber = cardButtons.index(of: sender) {
50             if !game.cards[cardNumber].isMatched {
51                 flipCount += 1
52             }
53             game.chooseCard(at: cardNumber)
54             updateViewModel()
55         }
56     }

```

Cmd + R

Когда мы коснулись карты...



... заметили, что слева панель навигатора **Navigator** исчезла?

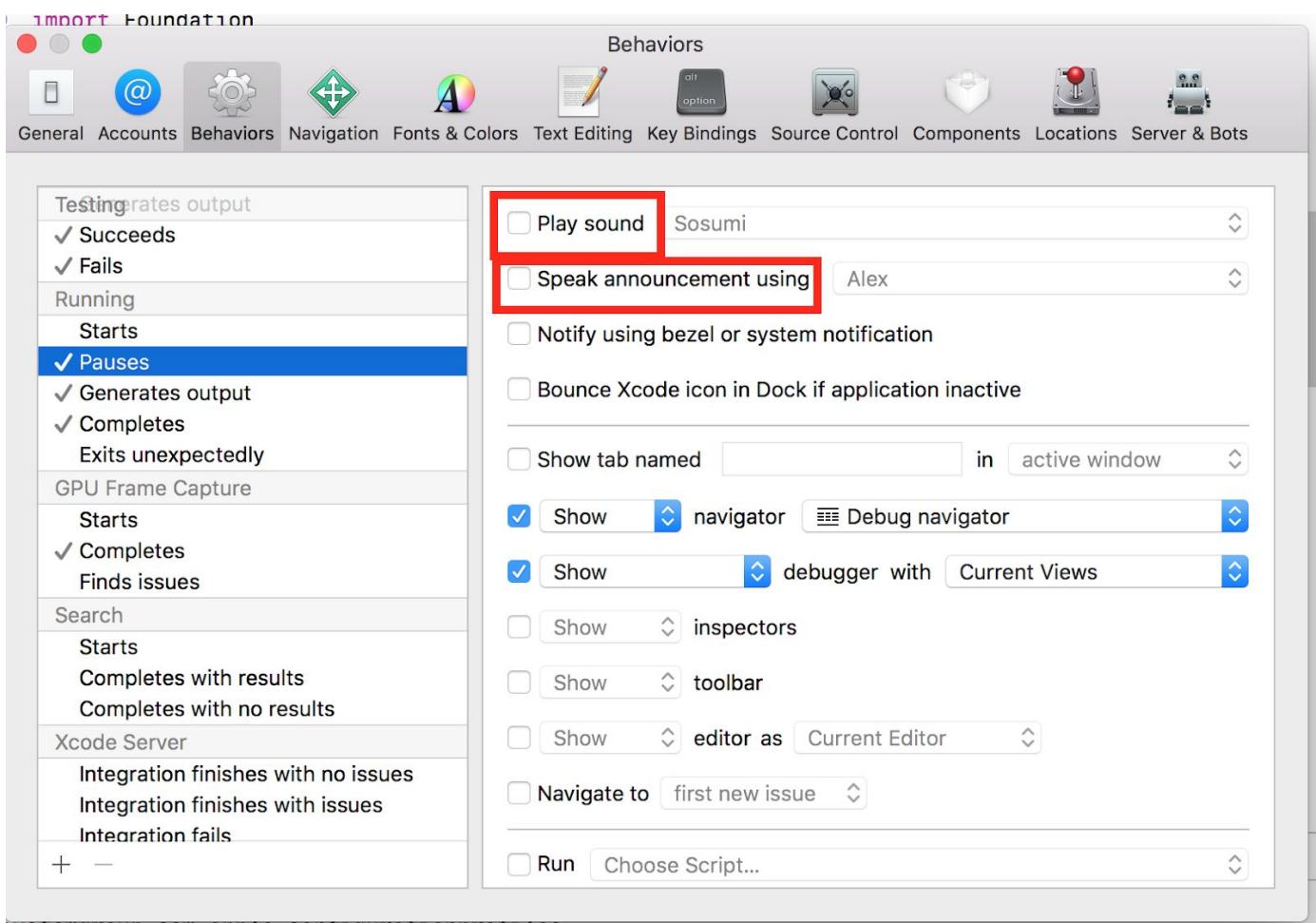
```

33
34     var emojiChoices = ["🂱", "🂲", "🂳", "🂴", "🂵", "🂶", "🂷"]
35     var emoji = Dictionary<Int, String>()
36
37     func emoji(for card: Card) -> String {
38         if emoji[card.identifier] == nil {
39             let randomIndex = Int(arc4random_uniform(UInt32(emojiChoices.count)))
40             emoji[card.identifier] = emojiChoices.remove(at: randomIndex)
41         }
42         return emoji[card.identifier] ?? "🂱"
43     }
44
45     // MARK: Handle Card Touch Behavior
46
47
48     @IBAction func touchCard(_ sender: UIButton) {
49         if let cardNumber = cardButtons.index(of: sender) {
50             if !game.cards[cardNumber].isMatched {
51                 flipCount += 1
52             }
53             game.chooseCard(at: cardNumber)
54             updateViewModel()
55         }
56     }

```

Мы заставили **Xcode** перейти в определенное состояние при определенном событии. Можете настроить это его “поведение” **Behavior** как вам удобно.

Ради сохранения времени и рассудка я не стану перебирать все эти опции, присваивать звуки событиям и всё такое.



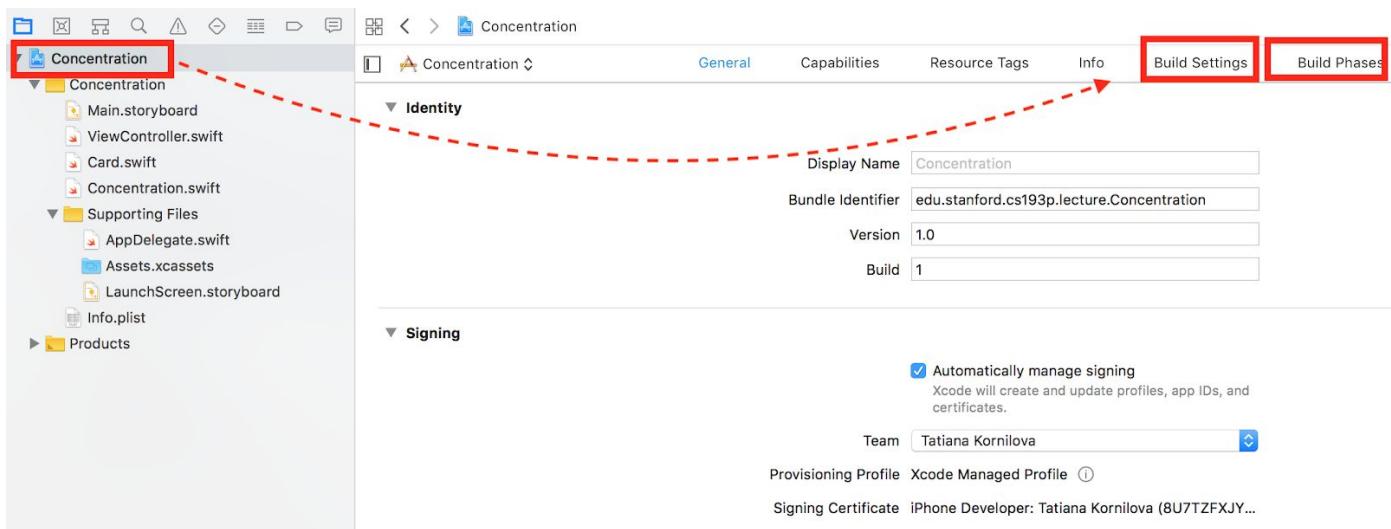
Просто пройдитесь по “поведениям” **Behaviors** и настройте их на свой вкус.

Тоже полезная вещь. Вопросы по **Behaviors**?

Замечательно!

Следующий пунктом рассмотрим, что представляет собой файл проекта в **Xcode**, с первого раза людям бывает сложно в этом разобраться.

Вот файл вашего проекта. Он содержит все эти вкладки внутри себя; здесь много всего происходит.



Детали я опущу, но в целом, я бы хотел сподвигнуть вас к тому, чтобы вы разобрались в двух вкладках: **Build Settings** и **Build Phases**.

Во вкладке **Build Settings** мы можем определиться с версией используемого языка программирования. Это влияет на компиляцию.

Давайте-ка даже пролистаем содержимое этой вкладки до этой опции. Версия **Swift 4.2**.

The screenshot shows the Xcode project settings for a target named "Concentration". The "Build Settings" tab is selected, indicated by a red box around the "Build Settings" button in the top navigation bar. The "Combined" tab is also highlighted in blue. The "Swift Compiler - Language" section is expanded, showing the "Swift Language Version" setting, which is also highlighted with a red box. The value "Swift 4.2" is listed next to it.

Есть удобный поиск, поскольку опций много. Вбил «**Swift language**» в поисковую строку и нашел ту же самую настройку:

The screenshot shows the Xcode search interface. The search bar at the top contains the text "Swift language", which is also highlighted with a red box. Below the search bar, the "Build Settings" tab is selected, indicated by a red box around the "Build Settings" button in the top navigation bar. The "Combined" tab is also highlighted in blue. The "Swift Compiler - Language" section is expanded, showing the "Swift Language Version" setting, which is also highlighted with a red box. The value "Swift 4.2" is listed next to it.

Я призываю вас пройтись по опциям **Build Settings**, посмотреть какие настройки есть, на что их можно поменять. Чем более сложным будет ваш проект, тем больше настроек в нём может появиться.

Вкладка **Build Phases** - это другая тема. Я бы хотел отметить, что здесь вы можете увидеть все файлы, используемые в программе. Вот исходники, а чуть ниже - бинарные библиотеки.

The screenshot shows the Xcode interface with the 'Build Phases' tab selected. In the 'Compile Sources' section, four Swift files are listed: ViewController.swift, Concentration.swift, Card.swift, and AppDelegate.swift. A red box highlights the list of files. In the 'Link Binary With Libraries' section, there is a '+' button highlighted with a red box. A dashed blue arrow points from the 'Compile Sources' section to the word 'Исходники' (Sources) above it. Another dashed blue arrow points from the '+' button in the 'Link Binary With Libraries' section to the word 'Библиотеки' (Libraries) above it.

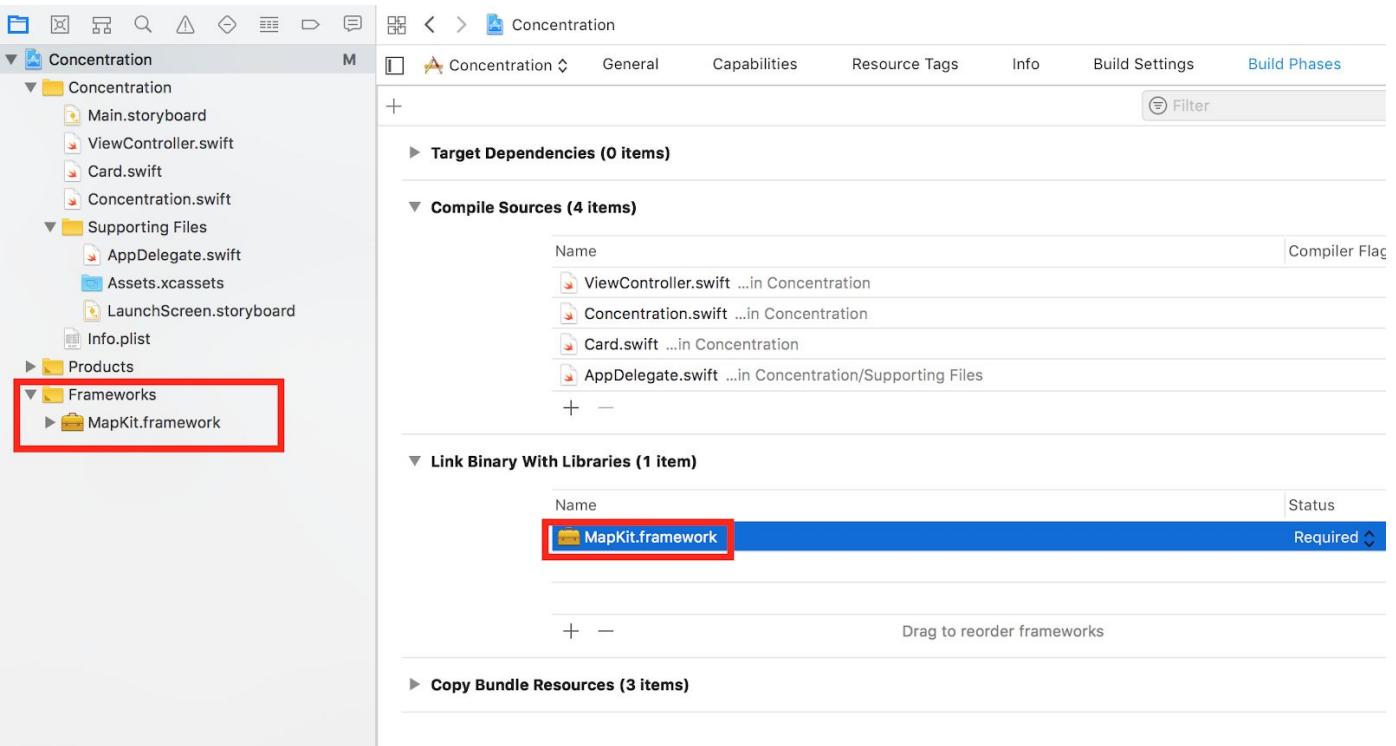
Так что если хотите подключить двоичную библиотеку к проекту (это мы с сделаем позже на этом курсе), то нужно кликнуть на кнопке «+».

Это используется тогда, когда вы в своем приложении используете некоторый элемент **UI**, который сложнее кнопки **UIButton** или метки **UILabel**, такой, которого нет в стандартной библиотеке **UIKit**. Допустим, нам нужна карта (**map**) для нашего специального приложения. Тогда мы кликаем на кнопке «+», вбиваем в окне поиска «**map**», находим **MapKit**, жмём «**Add**» («Добавить»).

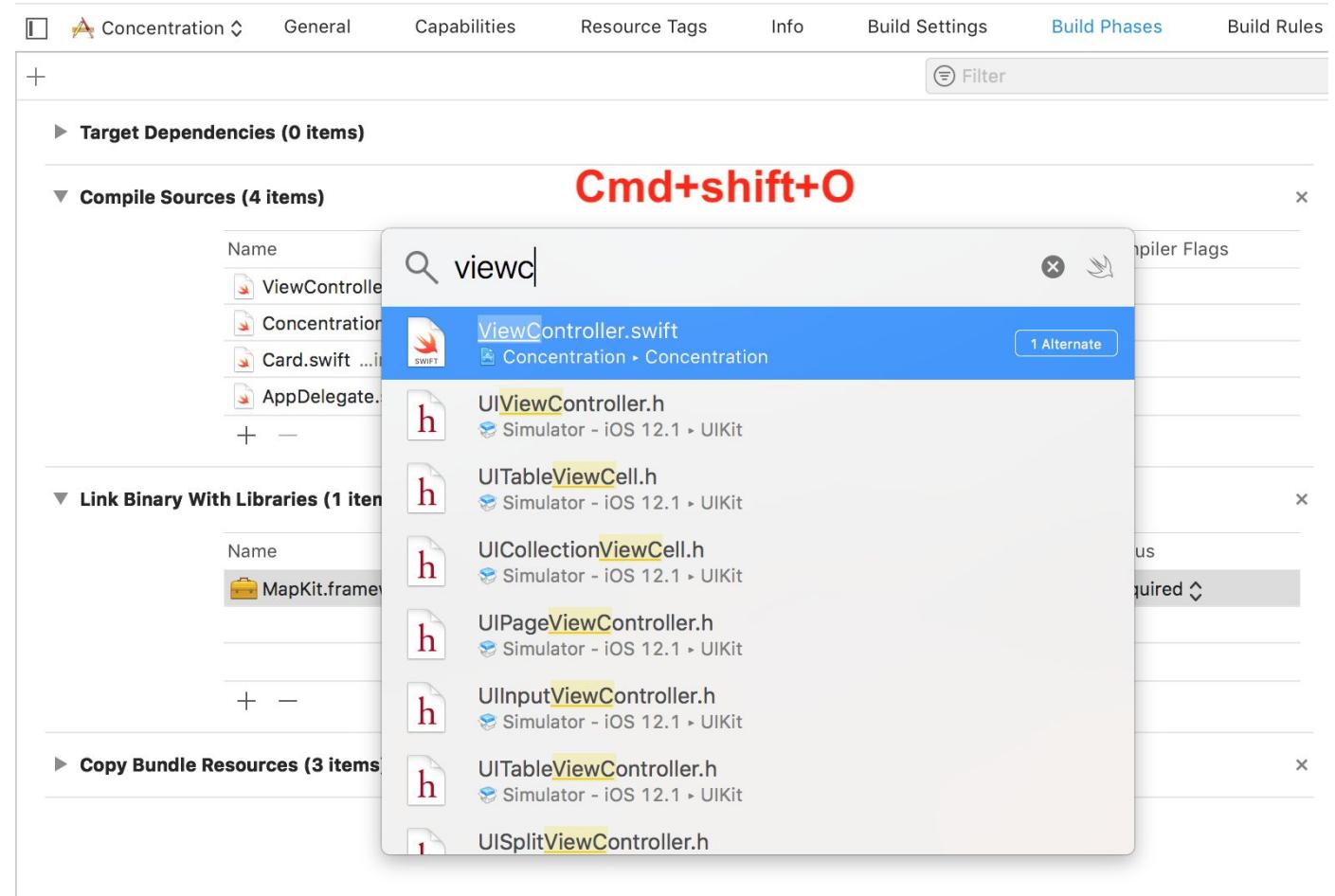
The screenshot shows the 'Choose frameworks and libraries to add' dialog box. A search bar at the top contains the text 'map'. Below the search bar, a list shows 'iOS 12.1' expanded, with 'MapKit.framework' selected and highlighted with a red box. A large blue arrow points from the 'Add' button at the bottom right of the dialog to the word 'Кликаем' (Click) in blue text above it. The background of the dialog shows other framework categories like 'Developer Frameworks'.

Вот так и добавляют фреймворки.

И мы видим, что добавился новый фреймворк, а в дереве Навигатора **Navigator** проекта появилась новая папка **Frameworks**, а внутри этой папки - папка **MapKit.framework**.



Теперь можно перейти в любой файл, например, **ViewController.swift** ...



... и вверху добавить директиву **import MapKit**.

```
1 //  
2 //  ViewController.swift  
3 //  Concentration  
4 //  
5 //  Created by CS193p Instructor on 9/23/17.  
6 //  Copyright © 2017 Stanford University. All rights reserved.  
7 //  
8  
9 import UIKit  
10 import MapKit  
11  
12 class ViewController: UIViewController  
13 {  
14     var game: Concentration! {  
15         didSet {  
16             updateViewFromModel()  
17         }  
18     }  
19  
20     var flipCount = 0 {  
21         didSet {
```

Всё, можно пользоваться. Понятно?

С файлами проекта проектом пока всё.

Со временем, всё это станет для вас более привычным.

----- 30 -ая минута лекции -----

Есть одна засада, о которой я расскажу, и она связана с файлом проекта Xcode ***.xcodeproj** при использовании **Git** или другой системы контроля версий. В этом случае вы будете “комитить” (**commit**) сам файл проекта ***.xcodeproj** в репозиторий. Представьте, что вы работаете совместно с другими людьми онлайн. Если у двоих возникнет **merge**-конфликт относительно файла проекта ***.xcodeproj**, этого большого файла, хранящего кипу данных, будет довольно неприятно разруливать такой конфликт. Имеет смысл при совместной работе изучить способы онлайн управления файлом проекта **Xcode *.xcodeproj** - это не так просто, как управление версиями обычного файла кода.

Возможно вам захочется узнать, каково внутреннее устройство файла проекта **Xcode *.xcodeproj**.

На этом сегодняшний план выполнен. У кого-нибудь есть вопросы?

- Можно ли осуществить подобие контроля версий?

Я как бы намекал на это, говоря о собственно файле проекта **Xcode *.xcodeproj**.

Ответ - ДА, вы можете использовать контроль версий.

Большинство, да почти все, сегодня применяют **GIT**. Очень рекомендую изучить **GIT**. Если не знаете **GIT**, зайдите в интернет, и я бы посоветовал нечто под названием **PRO GIT**. Есть такая бесплатная онлайн книга (в том числе и на русском языке):



git

--distributed-even-if-your-workflow-isnt

About

Documentation

Reference

Book

Videos

External Links

Downloads

Community

This book is available in [English](#).

Full translation available in

български език,

Español,

Français,

日本語,

한국어,

Nederlands,

Русский,

Українська

简体中文,

Partial translations available in

Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).



2nd Edition (2014)
Switch to 1st Edition

Download Ebook



1. Введение

- 1.1 О системе контроля версий
- 1.2 Краткая история Git
- 1.3 Основы Git
- 1.4 Командная строка
- 1.5 Установка Git
- 1.6 Первоначальная настройка Git
- 1.7 Как получить помощь?
- 1.8 Заключение

2. Основы Git

- 2.1 Создание Git-репозитория
- 2.2 Запись изменений в репозиторий
- 2.3 Просмотр истории коммитов
- 2.4 Операции отмены

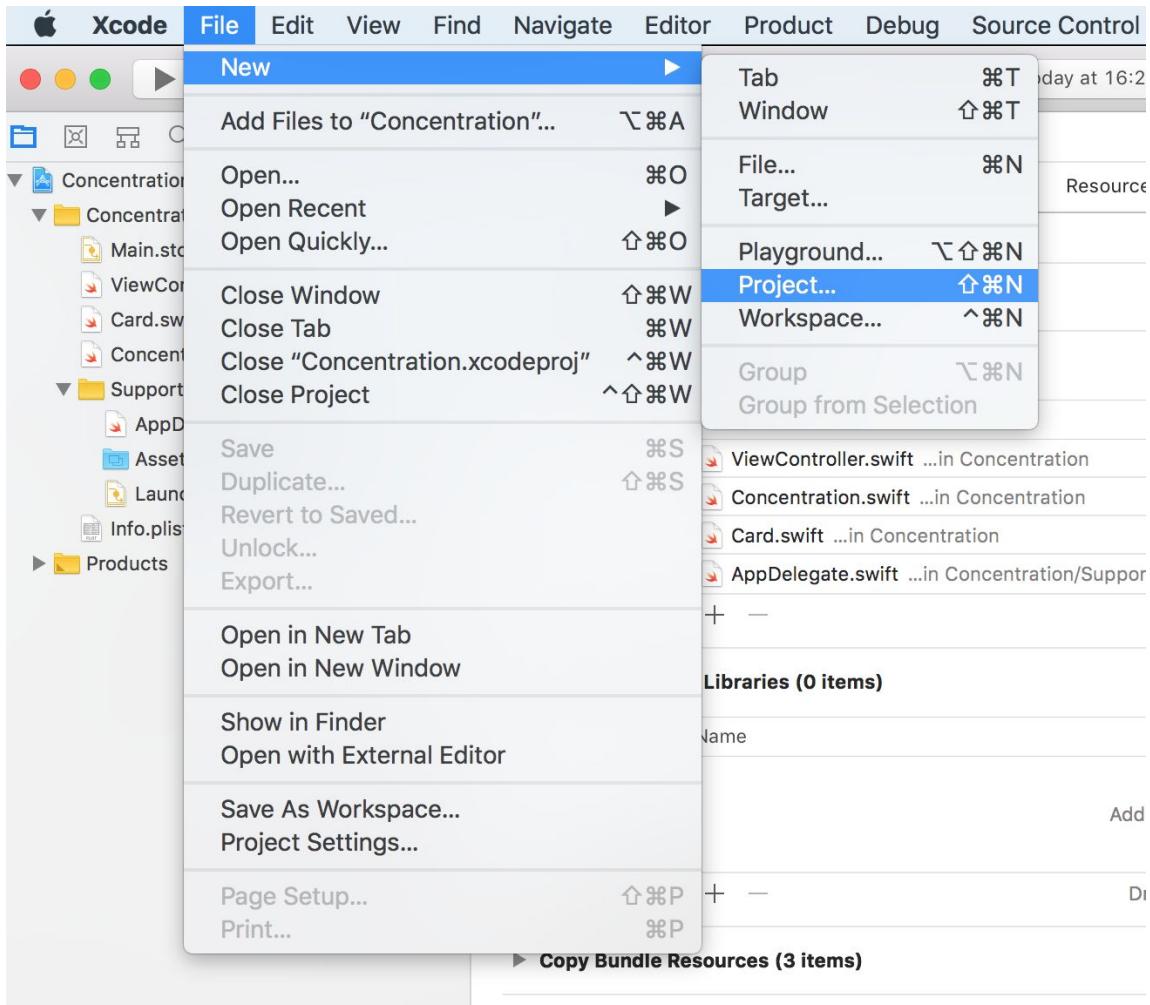
Если вы не имели дела с **GIT**, лучше сначала почитайте что-то вроде **PRO GIT**, а не бросайтесь сразу пробовать. Так вы узнаете как работает **GIT** и как он внутри выполняет контроль версий. Это вам очень поможет, поскольку это одна из болевых точек для людей при знакомстве с контролем версий. Они выучат пару команд для онлайн комита и отправки изменений другим людям. Но потом, если столкнутся с какой-либо проблемой, то не знают, как отменить изменения, вызвавшие эту проблему. Сложно понять, что происходит, если нет понимания того, как именно организовано хранение в **GIT** внутри.

Поэтому необходимо пройти через чтение подобной книги, чтобы понять как работает **GIT**.

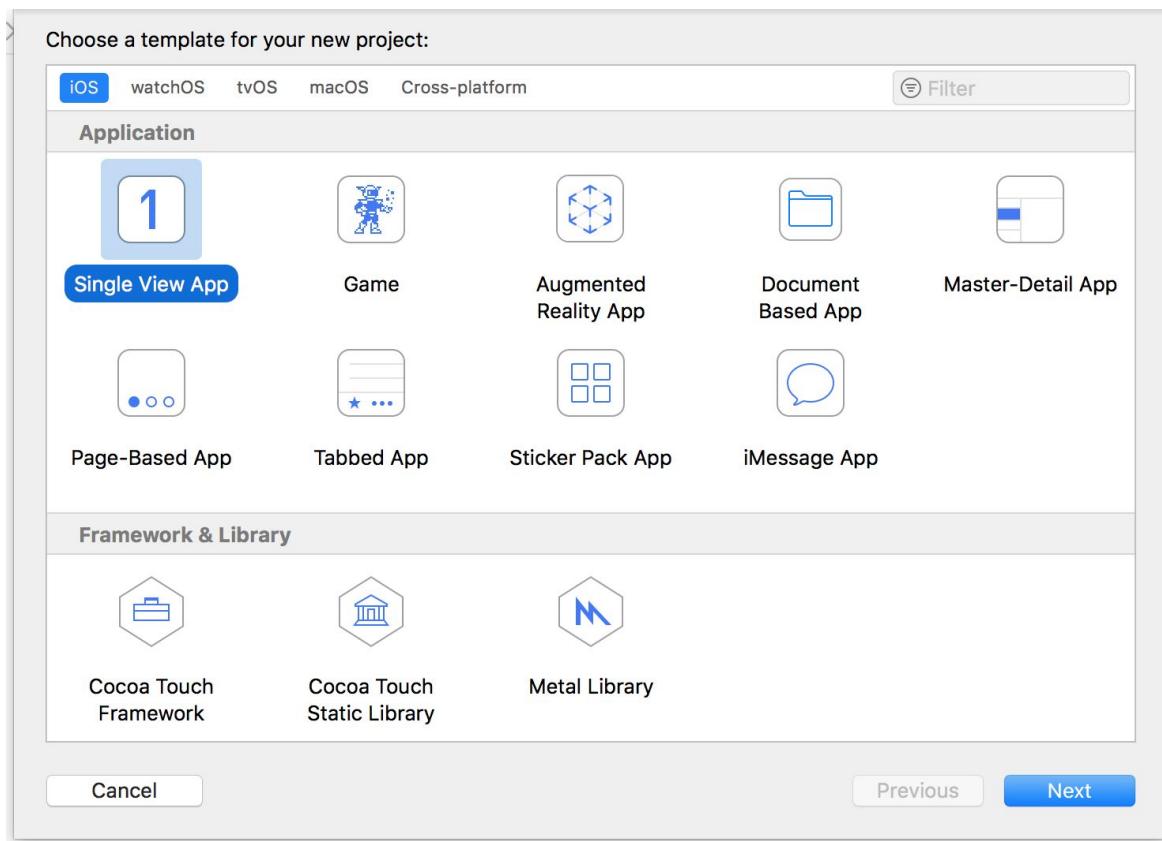
И если у вас нет аккаунта на [github](#) (удаленная версия **GIT**), зайдите на [github.com](#) и немедленно создайте его. Начните создавать свой репозиторий и изучать все команды, которые лучше объяснены в этой книге. Рекомендую.

Возвращаясь собственно к вопросу.

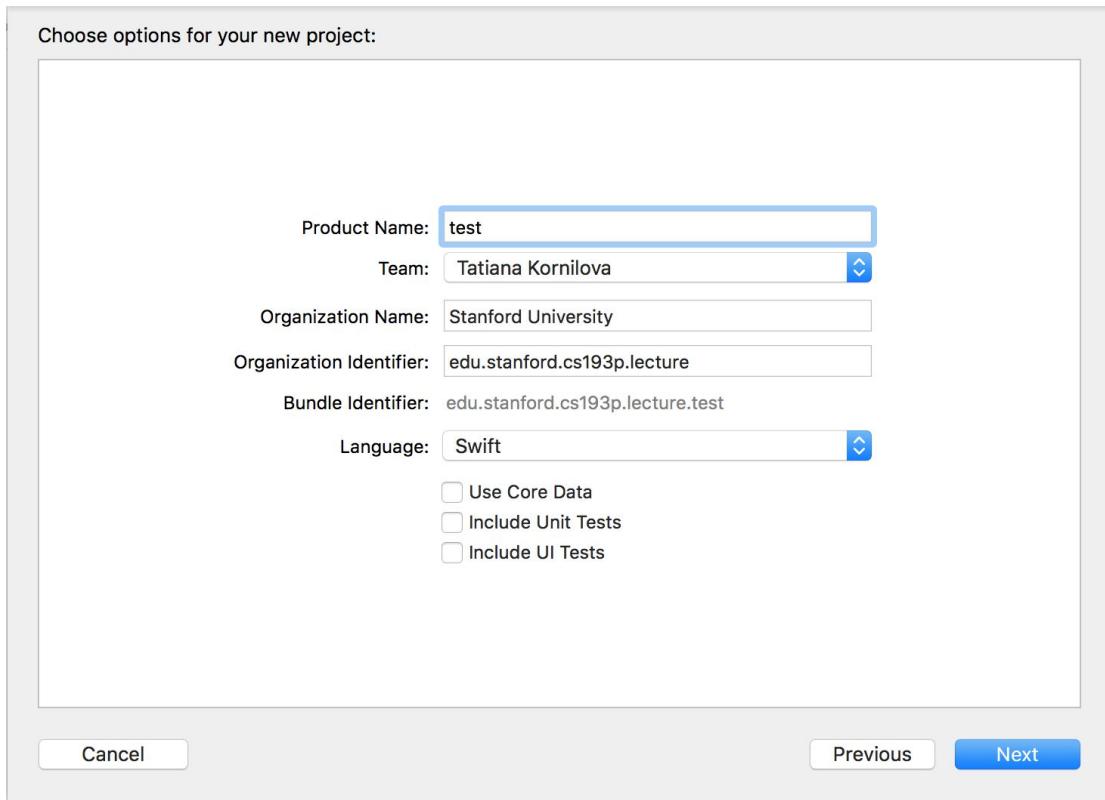
Создадим новый проект **File -> New -> Project ...**



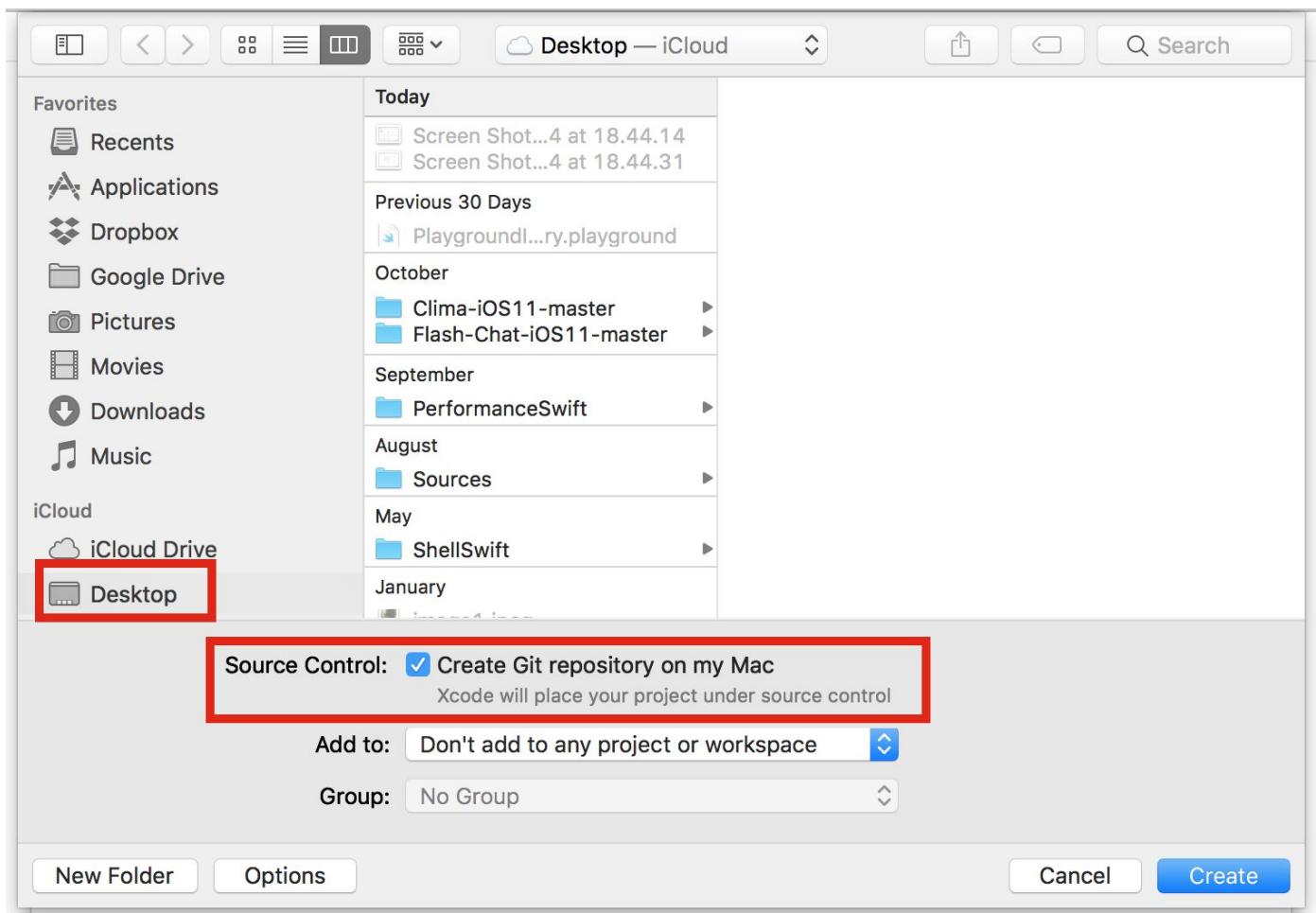
...я создаю **Single View App** ...



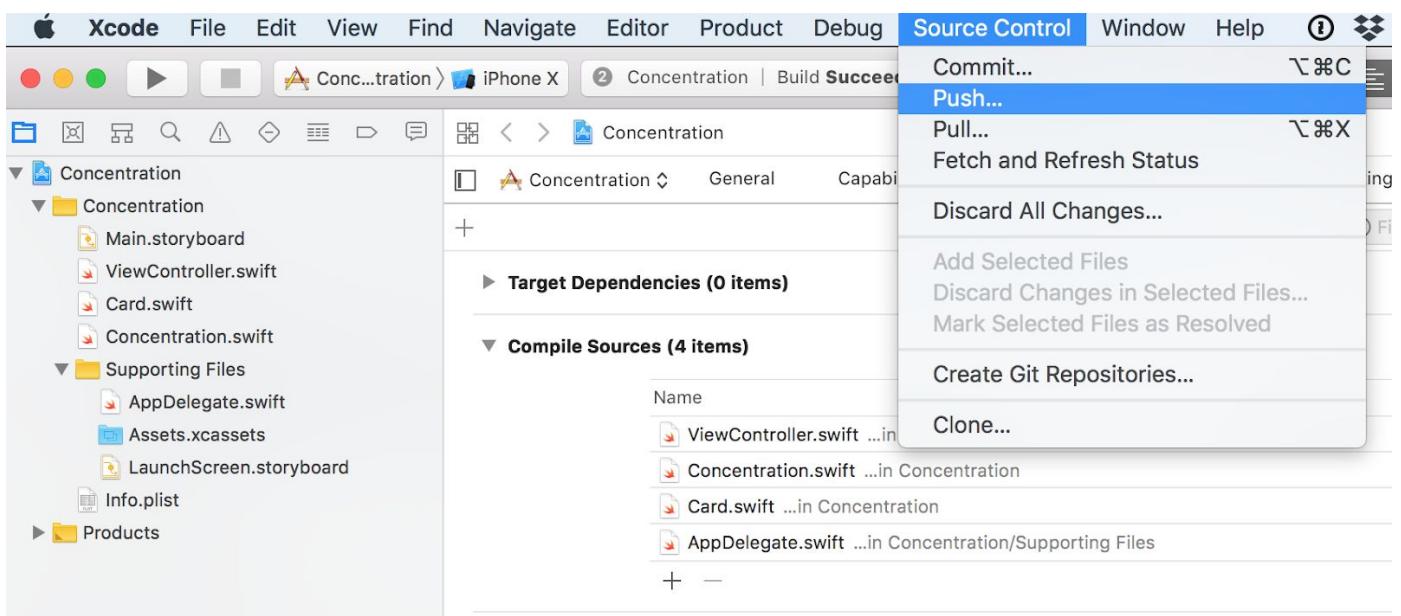
...и называю его **test**:



При создании проекта я размещу на Desktop, и у меня есть возможность активировать контроль версий с помощью чекбокса «Create Git repository on my Mac»:



Если я отмечу это чекбокс, то в директории проекта появится файл .git и можно с помощью меню Source Control -> Push всё это отправить на github.com:



Все произойдет автоматически, и вам не придется возиться вручную.

Вопросы?

Хорошо, если кто-то хочет поговорить отдельно, обращайтесь.

В остальном, на сегодня всё.

Всем спасибо.

----- 33-ая минута лекции -----

----- КОНЕЦ ПЯТНИЧНОЙ ЛЕКЦИИ 1-----

[На НАЧАЛО ПЯТНИЧНОЙ ЛЕКЦИИ 1](#)