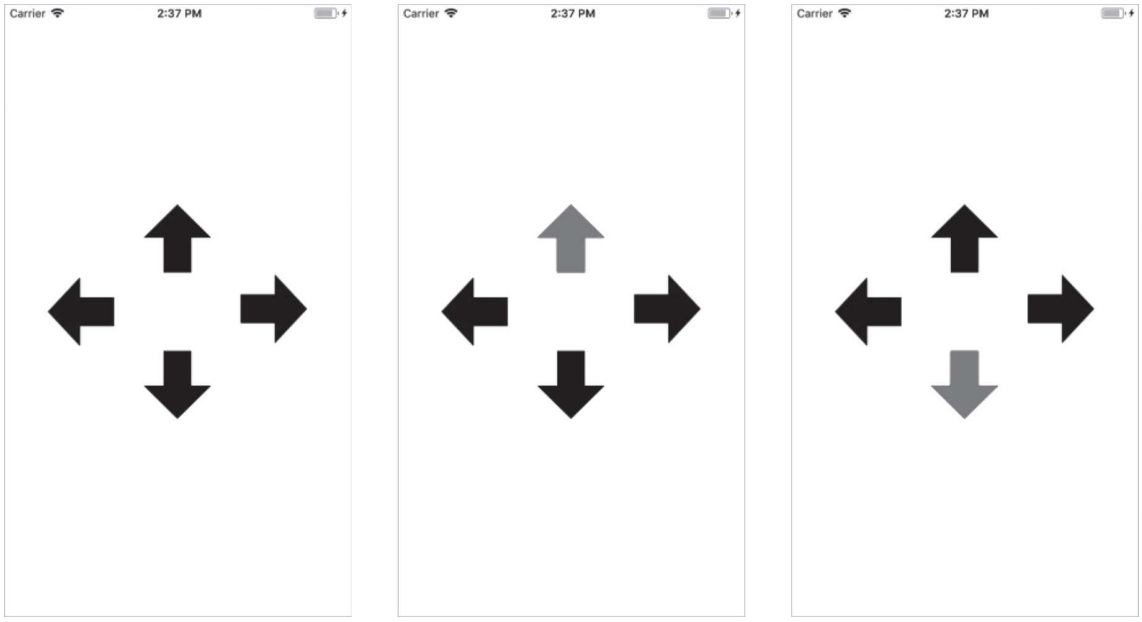
18 장. 스와이프 제스처 사용하기

페이지를 넘길 때, 항목을 삭제할 때, 게임에서 캐릭터의 방향을 움직일 때 등 많은 작업이 스와이프 제스처를 통해 이루지고 있다. 스와이프 제스처를 등록하고 사용하는 방법과 한 손가락 및 두 손가락을 이용한 제스처 활용 방법에 대해서 알아보자.

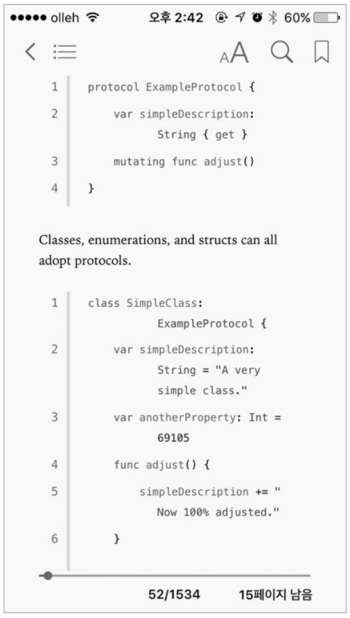


18-1 스와이프란?

손가락으로 화면을 상하좌우로 미는 동작이 바로 '스와이프' 제스처이다. 이러한 스와이프 제스처(Swipe Gesture)는 간단한 갤러리 앱 뿐만 아니라 PDF 뷰어, 키노트(Keynote)와 같이 문서 편집기 앱과 페이지를 넘기는 동작이 들어간 앱에서 많이 사용되고 있다.

한 손가락 스와이프뿐만 아니라 여러 손가락을 사용한 스와이프 제스처도 가능하다. 예를 들어 PDF 앱은 한 손가락으로 스와이프하면 밑줄이 그어지고, 두 손가락을 사용하면 페이지가 넘어간다.

iOS 사진 앱



아이북스(iBooks) 앱

키노트(Keynote) 앱

18-2 스와이프 연습 앱을 위한 기본 환경 구성하기

이번에는 스와이프 연습 앱을 만들기 위해 새 프로젝트를 만들고 앞의 '완성된 모습' 에서 본 것처럼 네 가지 방향의 화살표를 배치해 본다.

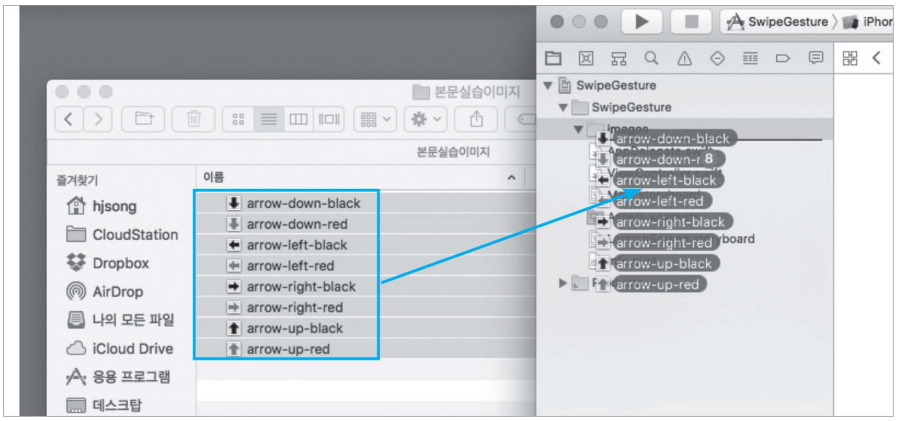
1. Xcode 를 실행한 후 'SwipeGesture'라는 이름으로 새 프로젝트를 만든다.

2. 뷰 컨트롤러 크기 조절하기

아이폰 모양의 뷰 컨트롤러 크기를 상황에 맞게 조절한다.

3.이미지 추가하기

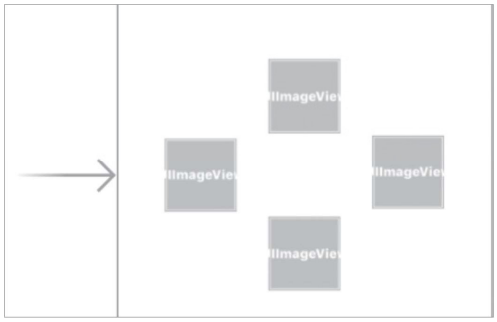
앱에서 사용할 이미지를 프로젝트에 추가한다. [Images]라는 폴더를 추가한 후 파인더(Finder)에서 원하는 이미지를 선택해 내비게이터 영역으로 드래그 앤 드롭하여 추가한다,

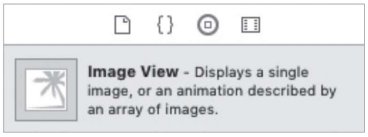


4. 스터토리보드 꾸미기

오른쪽 아랫부분의 오브젝트 라이브러리에서 [이미지 뷰(Image View)]를 찾아 네개의 [이미지 뷰]를 스토리보드로 끌어와 아래 그림과 같이 배치한다.

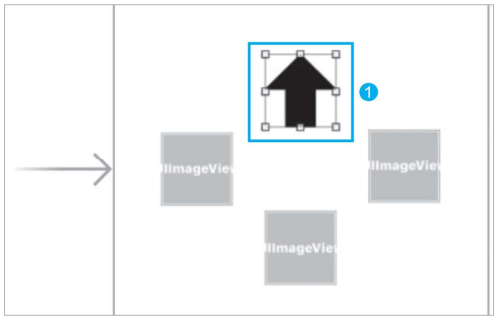
각각의 [이미지 뷰]에 위쪽, 아래 쪽, 왼쪽, 오른쪽 방향의 화살표 그림을 넣을 것이다.

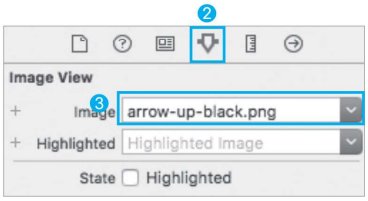




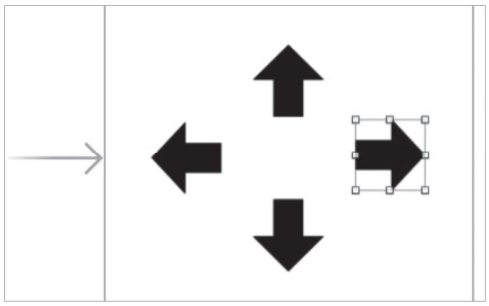
5. 네 개의 이미지 뷰 중에서 우선 맨 위의 [이미지 뷰]를 선택한 후 위쪽 방향의 화살표를 넣기 위해 오른쪽 인스펙터 영역에서 [Atrributes inspector]버튼을 클릭한다.

이미지 뷰 객체의 속성 중 Image 에서 [arrow-up-black.png]를 선택한다.





6. 같은 방법으로 나머지 [이미지 뷰]에 화살표 이미지를 넣는다.



- 위쪽 화살표: arrow-up-black.png
- 아래쪽 화살표: arrow-down-black.png
- 왼쪽 화살표: arrow-left-black.png
- 오른쪽 화살표: arrow-right-black.png

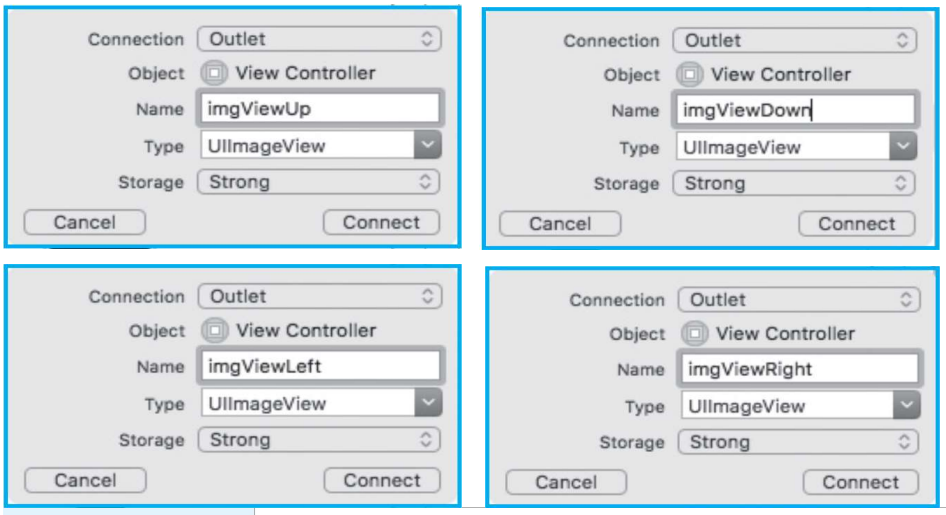
18-3 아웃렛 변수 추가하기

1. 보조 편집기 영역 열기

아웃렛 변수를 추가하기 위해 보조 편집기 영역을 연다.

2. 이미지 뷰에 대한 아웃렛 변수 추가하기

[이미지 뷰]에 대한 아웃렛 변수를 추가하자.



위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	- 위쪽 화살표: imgViewUp - 아래쪽 화살표: imgViewDown - 왼쪽 화살표: imgViewLeft - 오른쪽 화살표: imgViewRight
유형(Type)	UIImageView

```
1  import UIKit
2
3  class ViewController: UIViewController {
4      @IBOutlet var imgViewUp: UIImageView!
5      @IBOutlet var imgViewDown: UIImageView!
6      @IBOutlet var imgViewLeft: UIImageView!
7      @IBOutlet var imgViewRight: UIImageView!
8      ...
9  }
```

18-4 한 손가락 스와이프 기능 구현하기

한 손가락으로 스와이프하면 해당하는 화살표가 빨간색이 되는 것을 앞의 '완성된 모습'에서 미리 만나봤는데, 이 기능을 구현해 보자.

1. 스텐더드 에디터로 화면 모드 수정하기

코딩을 위해 화면 모드를 스탠더드 에디터 모드로 수정한다.

2. 이미지 배열 선언하기

각 방향 별로 검은색과 빨간색 이미지를 저장하기 위해 다음과 같이 배열을 선언한다.

```
1   import UIKit
2
3   class ViewController: UIViewController {
4       @IBOutlet var imgViewUp: UIImageView!
5       @IBOutlet var imgViewDown: UIImageView!
6       @IBOutlet var imgViewLeft: UIImageView!
7       @IBOutlet var imgViewRight: UIImageView!
8       var imgLeft = [UIImage]()
9       var imgRight = [UIImage]()
10      var imgUp = [UIImage]()
11      var imgDown = [UIImage]()
12      ...
13  }
```

3. 이미지 배열에 이미지 할당하기

앞에서 만든 이미지 배열에 이미지를 추가하기 위해 viewDidLoad 함수에 다음 소스를 입력한다. 각 배열은 append 메서드를 사용하여 값을 추가한다.

```
1   override func viewDidLoad() {
2           super.viewDidLoad()
3           // Do any additional setup after loading the view, typically from a nib.
4           imgUp.append(UIImage(named: "arrow-up-black.png")!)
5           imgUp.append(UIImage(named: "arrow-up-red.png")!)
6           imgDown.append(UIImage(named: "arrow-down-black.png")!)
7           imgDown.append(UIImage(named: "arrow-down-red.png")!)
8           imgLeft.append(UIImage(named: "arrow-left-black.png")!)
9           imgLeft.append(UIImage(named: "arrow-left-red.png")!)
10          imgRight.append(UIImage(named: "arrow-right-black.png")!)
11          imgRight.append(UIImage(named: "arrow-right-red.png")!)
12          ...
13  }
14
15
```

배열을 UIImage 형으로 만들었기 때문에 append 의 인수로 UIImage 형의 값을 입력한다.

스와이프하면 검은색 화살표가 빨간색이 되도록 하기 위해 처음에는 검은색 화살표의 파일명, 그 뒤에는 빨간색 화살표의 파일명을 입력하는 식으로 입력한다. 첫번째로 추가한 이미지는 imgUp[0]에 저장되고, 두번째로 추가한 이미지는 imgUp[1]에 저장된다.

4. 각 이미지 뷰에 이미지 할당하기

```
1   override func viewDidLoad() {
2           super.viewDidLoad()
3           // Do any additional setup after loading the view, typically from a nib.
4           imgUp.append(UIImage(named: "arrow-up-black.png")!)
5           imgUp.append(UIImage(named: "arrow-up-red.png")!)
6           imgDown.append(UIImage(named: "arrow-down-black.png")!)
7           imgDown.append(UIImage(named: "arrow-down-red.png")!)
8           imgLeft.append(UIImage(named: "arrow-left-black.png")!)
9           imgLeft.append(UIImage(named: "arrow-left-red.png")!)
10          imgRight.append(UIImage(named: "arrow-right-black.png")!)
11          imgRight.append(UIImage(named: "arrow-right-red.png")!)
12
13          imgViewUp.image = imgUp[0]
14          imgViewDown.image = imgDown[0]
15          imgViewLeft.image = imgLeft[0]
16          imgViewRight.image = imgRight[0]
17
18  }
19
20
```

15~18 행의 각 배열의 첫 번째 값은 검은색 화살표를 나타내며, 인자 값[0]을 사용하여 참조할 수 있다. 또한 두 번째 값은 빨간색 화살표를 나타내며, 인자 값[1]을 사용하며 참조할 수 있다.

5. 스와이프 제스퍼 인식하기

스와이프 제스처는 UISwipeGestureRecognizer 클래스에 의해 인식한다. 즉, UISwipeGestureRecognizer 클래스 상수의 direction 속성에 원하는 방향을 설정한 후 뷰 객체의 addGestureRecognizer 메소드를 사용해 원하는 방향의 스와이프 제스처를 등록하여 인식하게 된다.

```
1   override func viewDidLoad() {
2           super.viewDidLoad()
3           // Do any additional setup after loading the view, typically from a nib.
4           imgUp.append(UIImage(named: "arrow-up-black.png")!)
5           imgUp.append(UIImage(named: "arrow-up-red.png")!)
```

```
6               imgDown.append(UIImage(named: "arrow-down-black.png")!)
7               imgDown.append(UIImage(named: "arrow-down-red.png")!)
8               imgLeft.append(UIImage(named: "arrow-left-black.png")!)
9               imgLeft.append(UIImage(named: "arrow-left-red.png")!)
10              imgRight.append(UIImage(named: "arrow-right-black.png")!)
11              imgRight.append(UIImage(named: "arrow-right-red.png")!)
12
13              imgViewUp.image = imgUp[0]
14              imgViewDown.image = imgDown[0]
15              imgViewLeft.image = imgLeft[0]
16              imgViewRight.image = imgRight[0]
17
18
19              let swipeUp = UISwipeGestureRecognizer(target: self, action:
20  #selector(ViewController.respondToSwipeGesture(_:)))
21              swipeUp.direction =
                                        UISwipeGestureRecognizerDirection.up
22              self.view.addGestureRecognizer(swipeUp)

23              let swipeDown = UISwipeGestureRecognizer(target: self, action:
24  #selector(ViewController.respondToSwipeGesture(_:)))
25              swipeDown.direction =
                                      UISwipeGestureRecognizerDirection.down
26              self.view.addGestureRecognizer(swipeDown)

27              let swipeLeft = UISwipeGestureRecognizer(target: self, action:
28  #selector(ViewController.respondToSwipeGesture(_:)))
29              swipeLeft.direction =
                                      UISwipeGestureRecognizerDirection.left
30              self.view.addGestureRecognizer(swipeLeft)

31              let swipeRight = UISwipeGestureRecognizer(target: self, action:
32  #selector(ViewController.respondToSwipeGesture(_:)))
33              swipeRight.direction =
                                    UISwipeGestureRecognizerDirection.right
34              self.view.addGestureRecognizer(swipeRight)
    }
35
36
```

21 행은 UISwipeGestureRecognizer 클래스 상수 swipeUp 을 선언한다. 액션(action)인수는 해당 스와이프 제스처를 행했을 때 실행할 메서드를 의미한다.

22 행은 위에서 선언한 UISwipeGestureRecognizer 클래스 상수 swipeUp 의 direction 속성을 설정한다.

23 행은 뷰 객체의 addGestureRecognizer 메서드를 사용하여 위쪽 방향의 스와프 제스

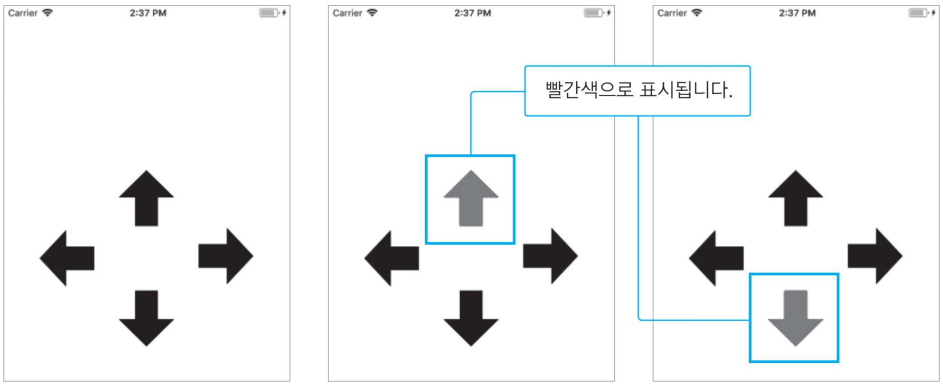
처를 등록한다.
같은 방식으로 아래쪽, 왼쪽, 오른쪽 방향의 스와이프 제스처를 등록한다. 앞에서 'up'을 입력한 부분에 각각 'down', 'left', 'right'를 입력하면 된다.

6. 액션 메서드 구현하기
스와이프 제스처를 등록할 때 입력한 액션(action)인수는 스와이프 제스처를 행했을 때 실행을 때 실행할 메서드를 의미한다. 그럼스와이프 제스처를 행했을 때 실행할 액션 메서드를 구현해 보자. 작업하던 곳에서 조금 내려가 respondToSwipeGesture 함수에 다음 내용을 입력한다.

```
1  @objc func respondToSwipeGesture(_ gesture: UIGestureRecognizer) {
2
3          if let swipeGesture = gesture as?
4                                  UISwipeGestureRecognizer {
           imgViewUp.image = imgUp[0]
5          imgViewDown.image = imgDown[0]
6          imgViewLeft.image = imgLeft[0]
7          imgViewRight.image = imgRight[0]
8
9          switch swipeGesture.direction {
10         case UISwipeGestureRecognizerDirection.up:
11             imgViewUp.image = imgUp[1]
12         case UISwipeGestureRecognizerDirection.down:
13             imgViewDown.image = imgDown[1]
14         case UISwipeGestureRecognizerDirection.left:
15             imgViewLeft.image = imgLeft[1]
16         case UISwipeGestureRecognizerDirection.right:
17             imgViewRight.image = imgRight[1]
18         default:
19             break
20         }
21     }
22 }
23
```

4 행은 만일 제스처가 있다면
5~8 행은 우선 전체 이미지 뷰를 검은 색 화살표로 초기화한다.(배열의 인자 값[1] 사용)

7.결과 확인
이제 [실행] 버튼을 클릭하여 앱을 실행한다. 프로그램이 실행된 후 한 손가락으로 위쪽, 아래쪽, 왼쪽, 오른쪽으로 스와이프하면 해당 방향으로 화살표가 빨간색으로 변한다.



18-5 멀티 터치 스와이프 제스처 인식하기

두 손가락 이상의 스와이프 제스처를 인식해 기능을 수행하도록 하자.

1. 두 손가락 이상의 스와이프 제스처를 인식해야 하므로 원하는 터치의 개수를 '2'로 설정한다. 다음 소스를 뷰 컨트롤러의 클래스 선언문 바로 아래에 추가한다.

```
1  import UIKit
2
3  class ViewController: UIViewController {
4      let numOfTouchs = 2
5      ...
6  }
```

2. 이제 앞에서 각 방향마다 입력했던 코드에 다음과 같이 한 줄씩 추가하겠다. 멀티 터치 스와이프 제스처를 등록 할 때는 numberOfTouchesRequired 속성이 필요하다. 이 속성에는 앞에서 설정한 터치의 개수인 'numOfTouchs'를 등록한다.

```
1  let swipeUp = UISwipeGestureRecognizer(target: self, action:
   #selector(ViewController.respondToSwipeGesture(_:)))
2          swipeUp.direction =
                   UISwipeGestureRecognizerDirection.up
3
4          swipeUp.numberOfTouchesRequired = numOfTouchs
5
6          self.view.addGestureRecognizer(swipeUp)
```

```
7
8           let swipeDown = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
9           swipeDown.direction =
                                UISwipeGestureRecognizerDirection.down
10
11          swipeDown.numberOfTouchesRequired = numOfTouchs
12
13          self.view.addGestureRecognizer(swipeDown)
14
15          let swipeLeft = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
16          swipeLeft.direction =
                                UISwipeGestureRecognizerDirection.left
17          swipeLeft.numberOfTouchesRequired = numOfTouchs
18
19          self.view.addGestureRecognizer(swipeLeft)
20
21          let swipeRight = UISwipeGestureRecognizer(target: self, action:
22  #selector(ViewController.respondToSwipeGesture(_:)))
            swipeRight.direction =
23                              UISwipeGestureRecognizerDirection.right
24          swipeRight.numberOfTouchesRequired = numOfTouchs
25          self.view.addGestureRecognizer(swipeRight)
26
27
```

4 행, 11 행, 18 행, 25 행을 추가한다.

3. 결과 확인

이제 실행 버튼을 클릭하여 앱을 실행한다. 앞의 프로그램과는 다르게 두 손가락으로 스와이프 해도 동작을 인식한다.

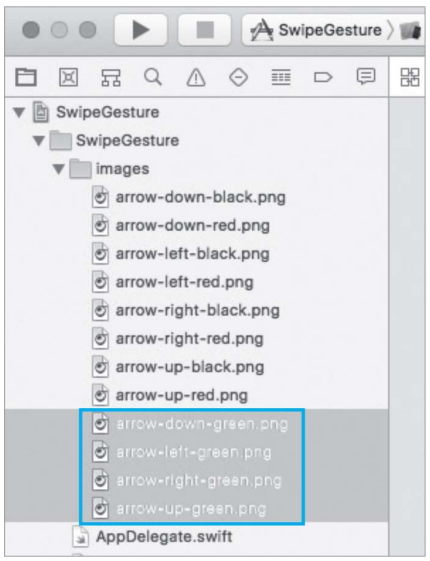
두 손가락을 사용하려면 [option]을 눌러야 한다. 하지만 스와이프하면 반대 방향으로 움직이므로 손가락을 같은 방향으로 스와이프 하려면 [option]과 함께 [shift]를 눌르고 움직이면 된다.

18-5 멀티 터치 스와이프 제스처 인식하기

한 손가락과 두 손가락의 스와이프 제스처를 각각 인식하게 하는 벙법을 알아보자.

1. 이미지 추가하기

두 손가락으로 스와이프할 때 표시할 초록색 화살표 이미지를 추가한다.



2. 이미지 배열에 초록색 화살표 추가하기

앞에서 검은색 화살표 파일명, 빨간색 화살표 파일명 순으로 입력했던 각 이미지 배열에 초록색 화살표 이미지를 각각 추가한다.

```
1   override func viewDidLoad() {
2           super.viewDidLoad()
3           // Do any additional setup after loading the view, typically from a nib.
4           imgUp.append(UIImage(named: "arrow-up-black.png")!)
5           imgUp.append(UIImage(named: "arrow-up-red.png")!)
6           imgUp.append(UIImage(named: "arrow-up-green.png")!)
7           imgDown.append(UIImage(named: "arrow-down-black.png")!)
8           imgDown.append(UIImage(named: "arrow-down-red.png")!)
9           imgDown.append(UIImage(named: "arrow-down-green.png")!)
10          imgLeft.append(UIImage(named: "arrow-left-black.png")!)
11          imgLeft.append(UIImage(named: "arrow-left-red.png")!)
12          imgLeft.append(UIImage(named: "arrow-left-green.png")!)
13          imgRight.append(UIImage(named: "arrow-right-black.png")!)
14          imgRight.append(UIImage(named: "arrow-right-red.png")!)
            imgRight.append(UIImage(named: "arrow-right-green.png")!)
15  ...
16  }
17
18
```

7 행, 10 행, 13 행, 16 행을 각 배열에 추가한다.

3. 스와이프 제스처 등록하기

앞에서 한 손가락으로 스와이프했을 때 인식할 상수로 swipeUp 을 선언하고, 액션 메서드로 'respondToSwipeGesture'를 입력했다. 그 아래에 이번에는 두 손가락으로 스와이프했을 때 인식할 상수를 선언한다. 이때, 두 손가락으로 스와이프했을 때 인식할 swipeUp, swipeDown 등을 앞에서 이미 선언했으니 이 부분을 먼저 삭제한다. 그리고 새로운 swipeUpMulti 상수를 선언하고 액션 메서드는 'respondTo SwipeGesture Multi'를 입력한다. 또한 swipeUpMulti 상수의 numberOfTouches Required 속성에 numOfTouches 값을 입력한다.

```
1   let swipeUp = UISwipeGestureRecognizer(target: self, action:
    #selector(ViewController.respondToSwipeGesture(_:)))
2           swipeUp.direction =
                    UISwipeGestureRecognizerDirection.up
3
4           swipeUp.numberOfTouchesRequired = numOfTouchs
5
6           self.view.addGestureRecognizer(swipeUp)
```

```
7
8           let swipeDown = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
9           swipeDown.direction =
                                UISwipeGestureRecognizerDirection.down
10
11          swipeDown.numberOfTouchesRequired = numOfTouchs
12
13          self.view.addGestureRecognizer(swipeDown)
14
15          let swipeLeft = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
16          swipeLeft.direction =
                                UISwipeGestureRecognizerDirection.left
17          swipeLeft.numberOfTouchesRequired = numOfTouchs
18
19          self.view.addGestureRecognizer(swipeLeft)
20
21          let swipeRight = UISwipeGestureRecognizer(target: self, action:
22  #selector(ViewController.respondToSwipeGesture(_:)))
            swipeRight.direction =
23                              UISwipeGestureRecognizerDirection.right
24          swipeRight.numberOfTouchesRequired = numOfTouchs
25          self.view.addGestureRecognizer(swipeRight)
26
27          let swipeUpMulti = UISwipeGestureRecognizer(target: self, action:
28  #selector(ViewController.respondToSwipeGestureMulti(_:)))
            swipeUpMulti.direction =
                                  UISwipeGestureRecognizerDirection.up
29          swipeUpMulti.numberOfTouchesRequired = numOfTouchs
            self.view.addGestureRecognizer(swipeUpMulti)
30
31          let swipeDownMulti = UISwipeGestureRecognizer(target: self, action:
32  #selector(ViewController.respondToSwipeGestureMulti(_:)))
33          swipeDownMulti.direction =
                                UISwipeGestureRecognizerDirection.down
            swipeDownMulti.numberOfTouchesRequired = numOfTouchs
34          self.view.addGestureRecognizer(swipeDownMulti)
35          let swipeLeftMulti = UISwipeGestureRecognizer(target: self, action:
36  #selector(ViewController.respondToSwipeGestureMulti(_:)))
37          swipeLeftMulti.direction =
38                                UISwipeGestureRecognizerDirection.left
            swipeLeftMulti.numberOfTouchesRequired = numOfTouchs
            self.view.addGestureRecognizer(swipeLeftMulti)
```

```
39          let swipeRightMulti = UISwipeGestureRecognizer(target: self, action:
    #selector(ViewController.respondToSwipeGestureMulti(_:)))
40          swipeRightMulti.direction =
41                                  UISwipeGestureRecognizerDirection.right
42          swipeRightMulti.numberOfTouchesRequired = numOfTouchs
43          self.view.addGestureRecognizer(swipeRightMulti)
44
45
46
```

4 행, 11 행, 18 행, 25 행은 삭제한다.

28 행부터 46 행까지 추가한다.

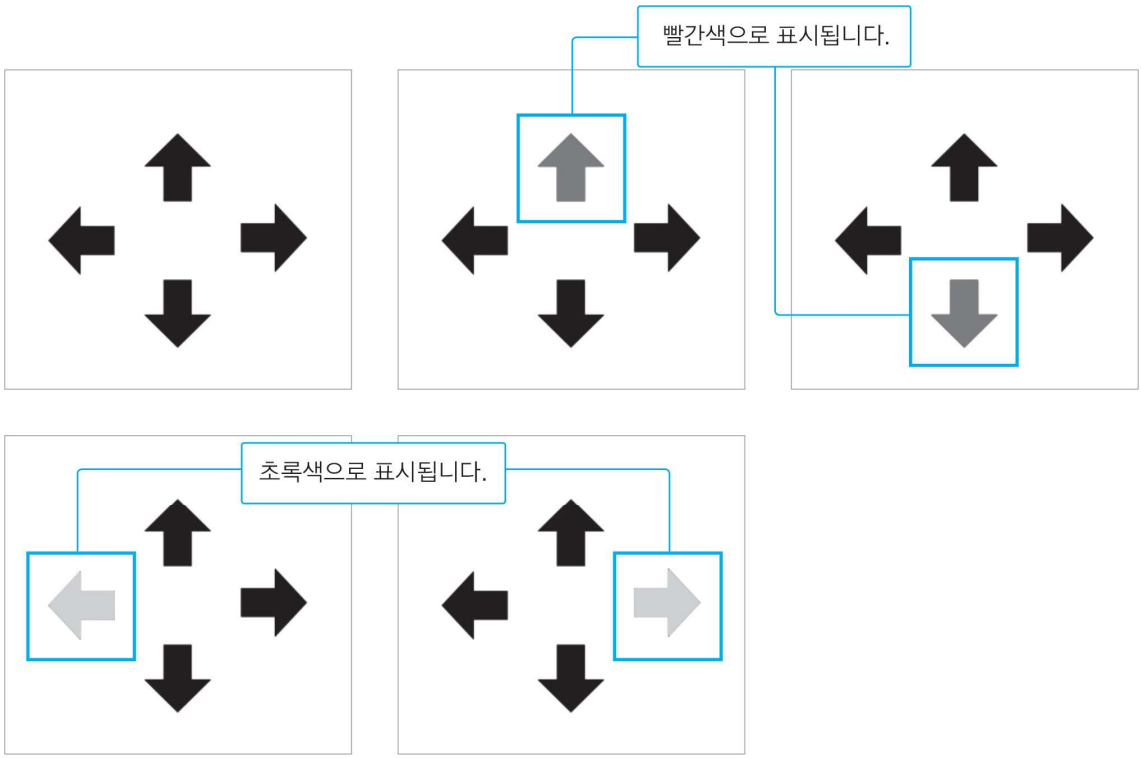
4. 액션 메서드 구현하기

두 손가락으로 스와이프했을 때 실행할 액션 메서드를 추가하자. 맨 아래로 내려가 다음 소스를 입력하면 이미지 뷰에 초록색 화살표 이미지가 할당된다. 각 배열의 세 번째 값은 초록색 화살표를 나타내며 인자 값 [2]을 사용하여 참조할 수 있다.

```
1   @objc func respondToSwipeGestureMulti(_ gesture:
                                            UIGestureRecognizer) {
2           if let swipeGesture = gesture as?
                                        UISwipeGestureRecognizer {
3               imgViewUp.image = imgUp[0]
4               imgViewDown.image = imgDown[0]
5               imgViewLeft.image = imgLeft[0]
6               imgViewRight.image = imgRight[0]
7
8               switch swipeGesture.direction {
9               case UISwipeGestureRecognizerDirection.up:
10                  imgViewUp.image = imgUp[2]
11              case UISwipeGestureRecognizerDirection.down:
12                  imgViewDown.image = imgDown[2]
13              case UISwipeGestureRecognizerDirection.left:
14                  imgViewLeft.image = imgLeft[2]
15              case UISwipeGestureRecognizerDirection.right:
16                  imgViewRight.image = imgRight[2]
17              default:
18                  break
19              }
20          }
21  }
```

5. 결과 보기

이제 [실행] 버튼을 클릭하여 앱을 실행한다. 한 손가락으로 스와이프하면 스와이프한 방향의 화살표가 빨간색으로 표시되고, 두 손가락으로 스와이프하면 스와이프한 방향의 화살표가 초록색으로 표시된다.



전체 소스 보기

```
//
//  ViewController.swift
//  SwipeGesture
```

```
//
//  Created by SoongM00 Rhee on 2018. 12. 6..
//  Copyright © 2018년 SoongMoo Rhee. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var imgViewUp: UIImageView!
    @IBOutlet var imgViewDown: UIImageView!
    @IBOutlet var imgViewLeft: UIImageView!
    @IBOutlet var imgViewRight: UIImageView!
    // 이미지를 저장하기 위한 배열 선언
    var imgLeft = [UIImage]()
    var imgRight = [UIImage]()
    var imgUp = [UIImage]()
    var imgDown = [UIImage]()
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        // 배열에 이미지 저장
        imgUp.append(UIImage(named: "arrow-up-black.png")!)
        imgUp.append(UIImage(named: "arrow-up-red.png")!)
        imgDown.append(UIImage(named: "arrow-down-black.png")!)
        imgDown.append(UIImage(named: "arrow-down-red.png")!)
        imgLeft.append(UIImage(named: "arrow-left-black.png")!)
        imgLeft.append(UIImage(named: "arrow-left-red.png")!)
        imgRight.append(UIImage(named: "arrow-right-black.png")!)
        imgRight.append(UIImage(named: "arrow-right-red.png")!)

        imgViewUp.image = imgUp[0]
        imgViewDown.image = imgDown[0]
        imgViewLeft.image = imgLeft[0]
        imgViewRight.image = imgRight[0]
        // 스와이프를 생성하기 위해 UISwipeGestureRecognizer 클래스를
사용한다.
        // 스와이프가 실행된 정보를 스와이프 생성시 전달
```

```
        let swipeUp = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
        swipeUp.direction = UISwipeGestureRecognizerDirection.up
        self.view.addGestureRecognizer(swipeUp)

        let swipeDown = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
        swipeDown.direction = UISwipeGestureRecognizerDirection.down
        self.view.addGestureRecognizer(swipeDown)

        let swipeLeft = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
        swipeLeft.direction = UISwipeGestureRecognizerDirection.left
        self.view.addGestureRecognizer(swipeLeft)

        let swipeRight = UISwipeGestureRecognizer(target: self, action:
#selector(ViewController.respondToSwipeGesture(_:)))
        swipeRight.direction = UISwipeGestureRecognizerDirection.right
        self.view.addGestureRecognizer(swipeRight)
    }
    // 스와프 제스처가 실행되었을 때의 메소드
    @objc func respondToSwipeGesture(_ gesture: UIGestureRecognizer)
    {

        if let swipeGesture = gesture as?
            UISwipeGestureRecognizer {
            // 스와이처가 실행되면 초기값을 [0]으로
            imgViewUp.image = imgUp[0]
            imgViewDown.image = imgDown[0]
            imgViewLeft.image = imgLeft[0]
            imgViewRight.image = imgRight[0]
            // 스와이프 방향에 따라 이미지 뷰 그림 변경
            switch swipeGesture.direction {
            case UISwipeGestureRecognizerDirection.up:
                imgViewUp.image = imgUp[1]
            case UISwipeGestureRecognizerDirection.down:
                imgViewDown.image = imgDown[1]
            case UISwipeGestureRecognizerDirection.left:
                imgViewLeft.image = imgLeft[1]
            case UISwipeGestureRecognizerDirection.right:
```

```
                imgViewRight.image = imgRight[1]
            default:
                break
            }

        }
    }

}
```