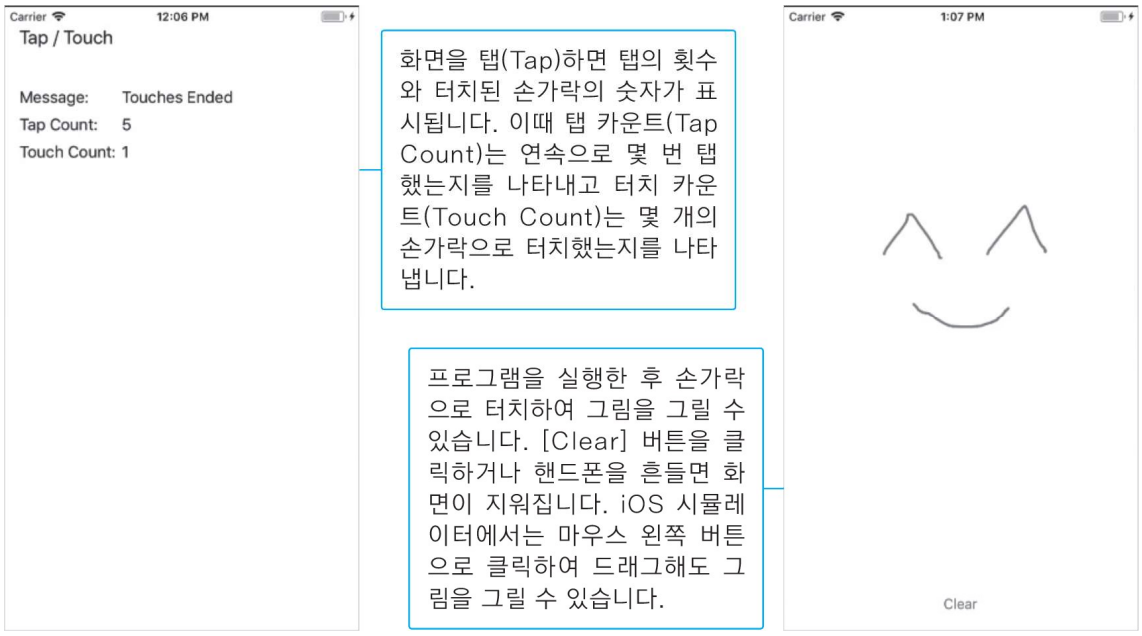
17 장. 탭과 터치를 사용해 스케치 앱 만들기

iOS 에서는 사용자의 터치로 대부분의 동작을 수행한다. 이처럼 iOS 에서 수행하는 동작의 대부분의 '터치'라고 해도 과언이 아니다. 다시 말해 화면을 터치하고, 드래그하고, 릴리즈하고, 더블 터치하는 등의 행동으로 대부분의 앱을 사용한다.

이 장에서는 이렇게 iOS 의 핵심이 되는 탭과 터치를 사용하는 방법을 알아보고, 이를 응용한 스케치 앱도 만들어 본다.

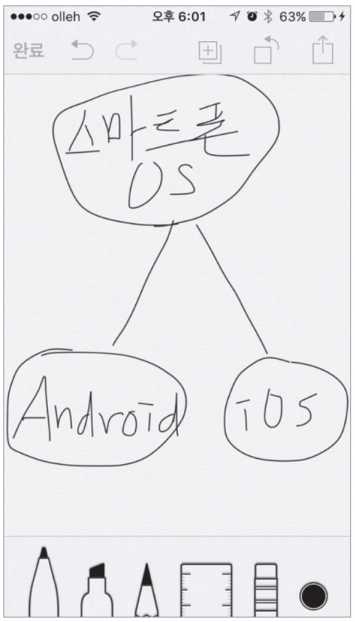


17-1 탭과 터치란?

아이폰과 아이패드에서 사용자의 입력을 받아들이는 장치는 두개의 버튼과 한개의 스위치 그리고 터치 스크린이다. 이 중에서 사용자가 가장 많이 사용하는 입력 장치는 단연 '터치 스크린'이다. 사용자는 매일, 매 순간 터치 스크린을 탭하고 터치하는 등의 동작의 연속으로 아이폰이나 아이패드를 사용하고 있다.

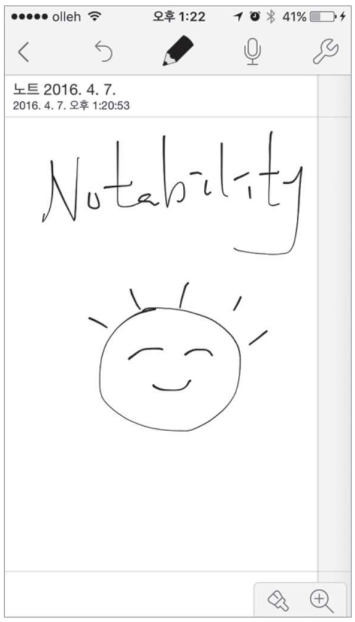
'터치(Touch)'가 화면을 만지는 모든 행위를 말한다면, '탭(Tap)'은 화면을 톡톡 두드리는 행위를 말한다. 아이폰이나 아이패드에서는 보통 화면을 확대하거나 축소할 때 탭을 사용한다. 이 터치와 탭은 화면을 만지는 순간 즉각적으로 반응한다. 마우스로 무언가를 '클릭'하는 것처럼 말이다. 이렇듯 직관적이고 당연해 보이는 탭과 터치도 사실은 프로그래밍으로 만들어낸 결과물이다.

이러한 탭과 터치의 사용법을 익히고 나면 멋진 앱을 만들 수 있다,



iOS 메모 앱

페이퍼(Paper) 앱



노타빌러티(Notability) 앱

17-2 탭과 터치 연습 앱을 위한 기본 환경 구성하기

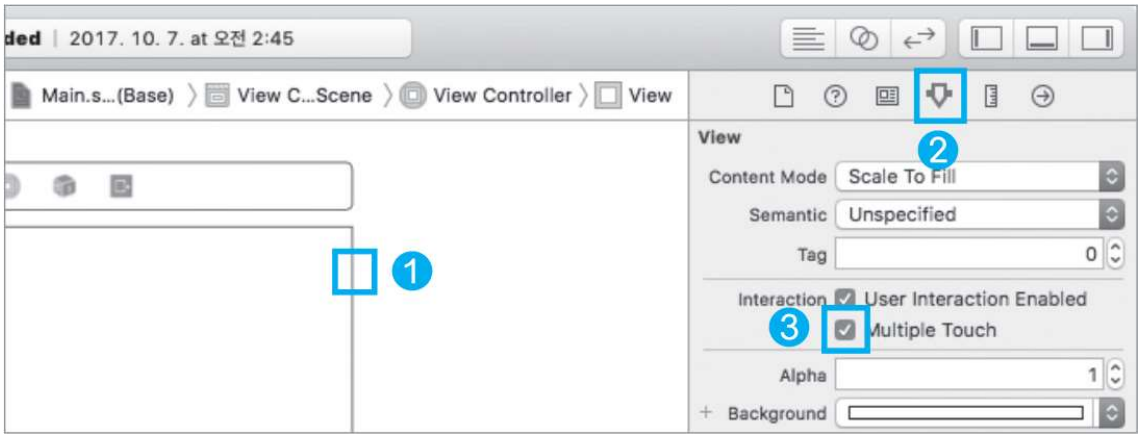
1. Xcode 를 실행한 후 'TapTouch'라는 이름으로 새 프로젝트를 만듭니다.

2. 뷰 컨트롤러 크기 조절하기

아이폰 모양의 뷰 컨트롤러 크기를 상황에 맞게 조절한다.

3. 멀티 터치 활성화하기

기본적으로 터치 이벤트는 하나의 터치에 대해서만 동작하도록 설정되어 있다. 하지만 대부분의 아이폰 앱에서는 '멀티 터치(Multiple Touch)'를 지원하기 때문에 이 예제에서는 멀티 터치를 활성화한다.



4. 탭과 터치 결과를 보여 줄 레이블 추가하기

오른쪽 아랫부분의 오브젝트 라이브러리에서 [레이블(Label)]을 찾아 스토리보드로 끌어와 그림과 같이 배치한다, 그런 다음 가장 위쪽의 레이블 객체를 "Tap / Touch"라고 입력한다. 그리고 화면에 출력할 것이므로 레이블 객체를 'Message:', 'Tap Count', 'Touch Count' 로 입력한다,

그리고 각각에 해당하는 값을 출력할 레이블을 오른쪽에 추가한다.

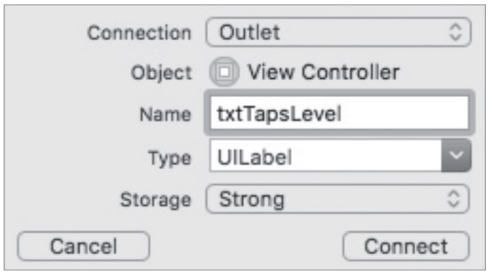
1. 보조 편집기 영역 열기
아웃렛 변수와 액션 함수를 추가하기 위해 오른쪽 윗부분의 [Show the Assistant editor] 버튼을 클릭하여 보조 편집기 영역을 연다.

2. 레이블에 대한 아웃렛 변수 추가하기
결과 출력용으로 추가한 세 레이블의 아웃렛 변수를 추가한다.
첫 번째로 'Message'에 대한 출력용 레이블을 마우스 오른쪽 버튼으로 클릭한 후 오른쪽 보조 편집기 영역으로 드래그하여 연결한다.
다음 표를 참조하여 아웃렛 변수를 추가한다,

위치	뷰 컨트롤러 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	txtMessage
유형(Type)	UILabel

3. 같은 방법으로 두개의 출력용 레이블에 대한 아웃렛 변수를 추가한다.
각 아웃렛 변수의 이름은 'txtTapsLevel'과 'txtTouchsLevel'로 입력한다.

위치	txtMessage 아웃렛 변수 바로 아래
연결(Connection)	Outlet
이름(Name)	txtTapsLevel
유형(Type)	UILabel



위치	txtTapsLevel 아웃렛 변수 바로 아래
연결(Connection)	Outlet
이름(Name)	txtTouchsLevel
유형(Type)	UILabel

17-3 탭과 터치 기능 구현하기

탭과 터치 기능을 구현하기 위한 코드를 작성해 본다. 탭과 터치 기능을 사용하기 위해서는 탭과 터치 이벤트 메서드를 작성한다.

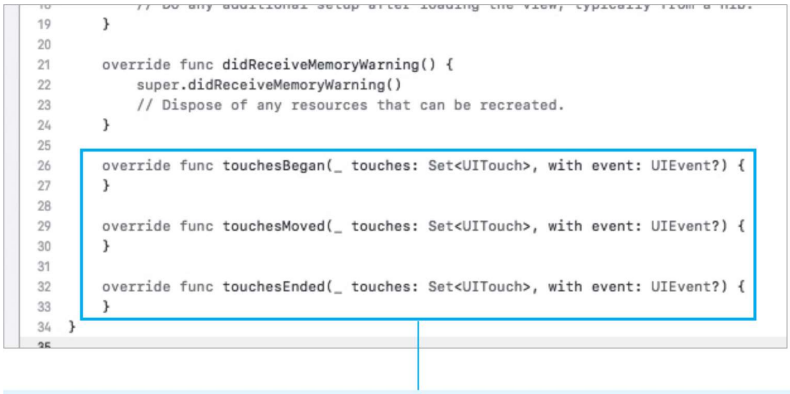
1. 스텐더드 에디터 화면 모드 수정하기

오른쪽 윗부분에서 [Show the Standard editor]버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.

2. 터치 이벤트 메서드 구현하기

터치 이벤트를 사용하기 위해서는 터치 이벤트가 발생했을 때 호출되는 메서드를 사용자가 재정의해야 한다. 재 정의란 부모 클래스에서 생성해 놓은 메서드에게 할 일을 새로 부여한다는 의미이다. 즉, 터치 이벤트가 발생했을 때 호출되는 메서드가 정해져 있는데, 해당 메서드가 무슨 일을 할지 정의한다는 의미이다.

재 정의해야 할 터치 이벤트 메서드는 다음과 같다. ,다음 소스를 가장 아래쪽 '}'위에 추가한다.



```
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
    }

    override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
    }

    override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
    }
}
```

```
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) { ❶
}
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) { ❷
}
override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) { ❸
}
```

1) 터치가 시작될 때 호출된다.
2) 터치된 손가락이 움직였을 때 호출된다.
3) 화면에서 손가락을 떼었을 때 호출된다.

3. 이제 각 메서드들을 구현해 보자. 다음의 소스를 위에서 추가한 메서드에 각각 추가한다.

```
1   override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
2           let touch = touches.first! as UITouch
3
4           txtMessage.text = "Touches Began"
5           txtTapsLevel.text = String(touch.tapCount)
6           txtTouchsLevel.text = String(touches.count)
7   }
8
9   override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
10          let touch = touches.first! as UITouch
11
12          txtMessage.text = "Touches Moved"
13          txtTapsLevel.text = String(touch.tapCount)
14          txtTouchsLevel.text = String(touches.count)
15  }
16
17  override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
18          let touch = touches.first! as UITouch
19
20          txtMessage.text = "Touches Ended"
21          txtTapsLevel.text = String(touch.tapCount)
22          txtTouchsLevel.text = String(touches.count)
23  }
24
25
26
```

3,12,21 행은 현재 발생한 터치 이벤트를 가지고 온다.

5,14,23 행은 메서드에서 현재 발생한 이벤트의 종류를 출력한다.(메서드 마다 조금씩 다르다)

6,15,24 행은 touches 세트 안에 포함된 터치의 갯수를 출력한다.

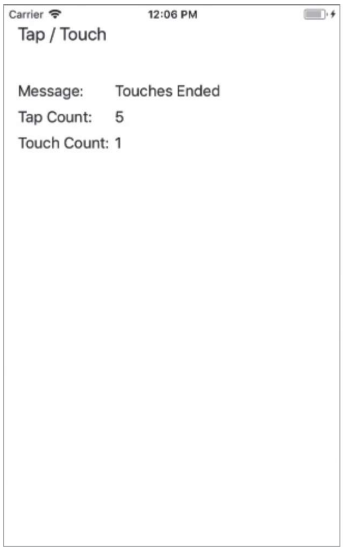
7,16,25 행은 터치 객체들 중 첫번째 객체에서 탭의 개수를 가져와 출력한다.

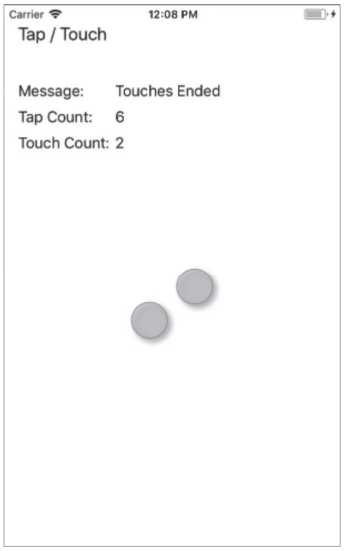
4. 결과 보기

앱을 실행한 후 화면을 탭의 횟수와 터치된 손가락의 숫자가 표시된다,

'탭 카운트(Top Count)'는 연속적으로 몇 번 탭했는지 나타내고, '터치 카운트(Touch Count)'는 몇개의 손가락을 터치했는지를 나타낸다.

iOS 시뮬레이션에서는 [option]을 누른 효과를 낼 수 있다.





17-4 스케치 앱 만들기

이번에는 탭과 터치 제스처 그리고 앞 장에서 배운 그림 그리기 기능을 이용하여 스케치 앱을 만들어 보자. 그림을 그리기 위해 이미지 뷰(Image View) 객체를 추가하고 화면 삭제를 위하여 버튼(Button)객체를 추가한다.

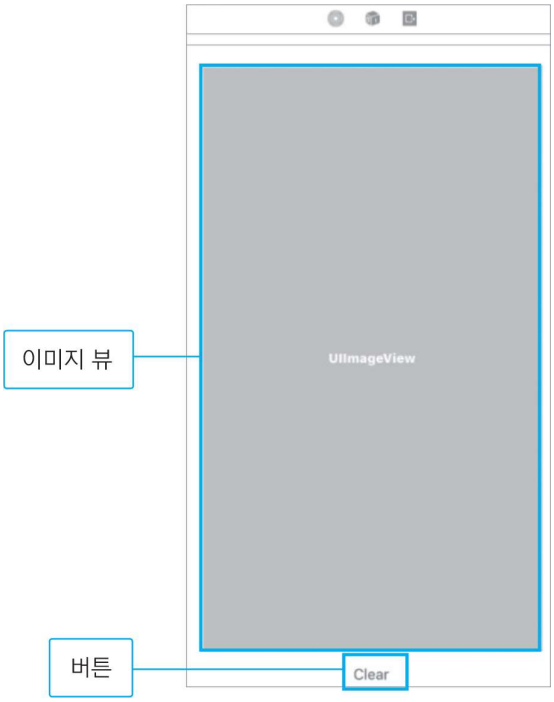
즉, 이미지 뷰 위에 그림을 그리고 버튼을 클릭하면 지금까지 그렸던 화면이 지워지는 앱을 만들어 보자.

1. 스케치 앱을 만들기 위해 Xcode 를 실행한 후 'Sketch'라는 이름으로 새 프로젝트를 만든다.

2. 스토리보드 꾸미기

오브젝트 라이브러리에서 [이미지 뷰(Image View)]와 [버튼(Button)]을 찾아 스토리보드를 끌어와 오른쪽 그림과 같이 배치하고 버튼 이름은 'Clear'로 바꾼다.

사용된 객체: 이미지 뷰, 버튼

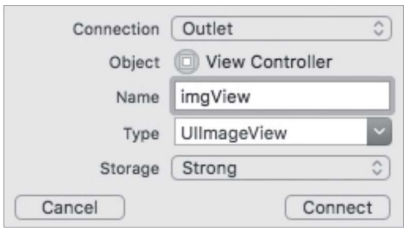


3. 이제 프로그램에서 사용할 아웃렛 변수와 액션 함수를 추가한다.

아웃렛 변수와 액션 함수를 추구하기 위해 보조 편집기 영역을 연다.

4. 그림을 그릴 이미지 뷰에 대한 아웃렛 변수를 추가한다.

위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	imgView
유형(Type)	UIImageView



5. 버튼에 대한 액션 함수 추가하기

이제는 [Clear]버튼의 액션을 추가하자. 같은 방법으로 마우스 오른쪽 버튼으로 버튼 객체를 선택한 후 드래그해서 오른쪽 보조 편집기 영역에 갖다 놓는다.

설정은 다음 표를 참조한다.

위치	뷰컨트롤러 클래스의 닫힘 괄호 '}' 바로 위
연결(Connection)	Action
이름(Name)	clearImageView
유형(Type)	UIButton

17-5 스케치 앱 기능 구현하기

1. 스탠더드 애디터로 화면 모드 수정

코딩을 위해 화면 모드를 아래 그림을 참고해 스탠더드 에디터 모드로 변경한다.

2. 변수 추가하기

필요한 변수들을 이미지 뷰의 아웃렛 변수 아래에 추가한다.

```
1  class ViewController: UIViewController {
2      @IBOutlet var imgView: UIImageView!
3
4      var lastPoint: CGPoint!
5      var lineSize:CGFloat = 2.0
6      var lineColor = UIColor.red.cgColor
```

```
7
8       override func viewDidLoad() {
9       ...
10      }
11  ...
12  }
```

4 행은 lastPoint: 바로 전에 터치하거나 이동한 위치

5 행은 lineSize : 선의 두께

6 행은 lineColor: 선의 색상

3. 버튼 동작 함수 코딩하기

[Clear]버튼을 클릭했을 때 그렸던 스케치를 지우기 위한 clearImageView 함수를 코디ㅇ한다. 추가한 버튼의 액션 함수 안에 소스를 입력한다.

```
1   @IBAction func clearImageView(_ sender: UIButton) {
2           imgView.image = nil
3       }
```

2 행에 imgView 에 nil 을 넣는다. nil 은 아무것도 없다는 의미이다. 즉, 이미지 뷰의 이미지를 없애는 것이다.

4. 터치 이벤트 메서드 재정의하기

터치 이벤트를 사용하기 위해서는 터치 이벤트가 발생했을 때 호출되는 메서드를 재정의해야 한다. 우선 다음 소스를 가장 아래에 있는 뷰 컨트롤러 클래스 닫음 괄호 '}'위에 입력한다.

```
1   override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
2
3   }
4
5   override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
6
7   }
8
9   override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
10  }
11
12
13
```

5. TouchesBegan 메서드 구현하기

앞으로 재정의한 메서드를 차례로 구현해 보자. 첫 번째로 사용자가 화면을 터치하면 스케치를 시작하도록 touchesBegan 메서드를 구현해 보겠다.

```
1   override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
```

```
2           let touch = touches.first!
3
4           lastPoint = touch.location(in: imgView)
5   }
6
```

3 행은 현재 발생한 터치 이벤트를 가지고 온다.

5 행은 터치된 위치를 lastPoint 라는 변수에 저장된다.

6. TouchesMoved 메서드 구현하기

두 번째로 사용자가 터치한 손가락을 움직이면 스케치도 따라서 움직이도록 touchesMoved 메서드를 구현해 보겠다. 마찬가지로 앞에서 입력한 소스 안에 추가한다.

```
1   override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
2           UIGraphicsBeginImageContext(imgView.frame.size)
3       UIGraphicsGetCurrentContext()?.setStrokeColor(lineColor)
4     UIGraphicsGetCurrentContext()?.setLineCap(CGLineCap.round)
5           UIGraphicsGetCurrentContext()?.setLineWidth(lineSize)
6
7           let touch = touches.first! as UITouch
8           let currPoint = touch.location(in: imgView)
9
10          imgView.image?.draw(in: CGRect(x: 0, y: 0, width:
11  imgView.frame.size.width, height: imgView.frame.size.height))
12          UIGraphicsGetCurrentContext()?.move(to: CGPoint(x: lastPoint.x, y:
13  lastPoint.y))
            UIGraphicsGetCurrentContext()?.addLine(to: CGPoint(x: currPoint.x, y:
14  currPoint.y))
            UIGraphicsGetCurrentContext()?.strokePath()
15
16          imgView.image =
17                  UIGraphicsGetImageFromCurrentImageContext()
            UIGraphicsEndImageContext()
18
19          lastPoint = currPoint
20  }
21
```

5 행은 라인의 끝 모양을 라운드로 설정한다.

8 행은 현재 발생한 터치 이벤트를 가지고 온다.

9 행은 터치된 위치를 currPoint 로 가지고 온다

11 행은 현재 이미지 뷰(Image View)에 있는 이미지를 이미지 뷰(Image View)의 크기를 그린다.

13 행은 이전에 이동된 위치인 lastPint 로 시작 위치를 이동시킨다.

14 행은 lastPoint 에서 현재 위치인 currpoint 까지 선을 추가한다,
20 행은 터치된 위치를 lastPoint 라는 변수에 할당한다.

7. TouchesEnded 메서드 구현하기

세 번째로 사용자가 화면에서 손가락을 떼었을 때 스케치도 끝나도록 touchesEnded 메서드를 구현해 보자. touchesMoved 와 거의 똑같고 마지막 currPoint 값을 lastPoint 에 할당하는 구문만 빠진다. 11 행만 조금 다르다.

```
1  override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
2          UIGraphicsBeginImageContext(imgView.frame.size)
3      UIGraphicsGetCurrentContext()?.setStrokeColor(lineColor)
4    UIGraphicsGetCurrentContext()?.setLineCap(CGLineCap.round)
5          UIGraphicsGetCurrentContext()?.setLineWidth(lineSize)
6
7          imgView.image?.draw(in: CGRect(x: 0, y: 0, width:
8  imgView.frame.size.width, height: imgView.frame.size.height))
9          UIGraphicsGetCurrentContext()?.move(to: CGPoint(x: lastPoint.x, y:
10 lastPoint.y))
           UIGraphicsGetCurrentContext()?.addLine(to: CGPoint(x: lastPoint.x, y:
11 lastPoint.y))
           UIGraphicsGetCurrentContext()?.strokePath()
12
13         imgView.image =
                   UIGraphicsGetImageFromCurrentImageContext()
14         UIGraphicsEndImageContext()
   }
15
```

8. motionEnded 메서드 구현하기

이제 사용자가 iOS 기기를 흔들었을 때 이미지 뷰 위에 그렸던 스케치가 지워지도록 motionEnded 메서드를 구현해 보자.

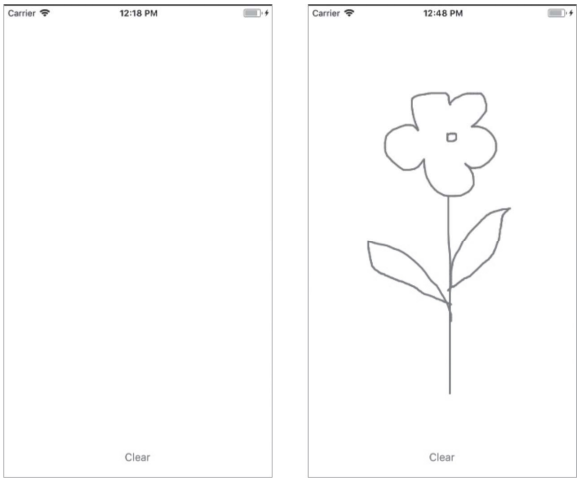
```
1  override func motionEnded(_ motion: UIEventSubtype, with event: UIEvent?) {
2          if motion == .motionShake {
3              imgView.image = nil
4          }
5  }
6
```

3 행은 '세이크(Shake)' 라는 모션 이벤트가 발생할 경우
4 행은 이미지 뷰의 이미지를 nil 로 대체하여 이미지를 삭제한다.

9. 결과 보기

이제 [실행]버튼을 클릭하여 앱을 실향한다. 프로그램이 실행되면 손가락으로 터치하고

이동하며 그림을 그릴 수 있다. 또한 기기를 흔들거나 [Clear]버튼을 클릭해 화면을 지울 수도 있다.



iOS 시뮬레이터에서 세이크 이벤트 만들기

iOS 시뮬레이터에서도 세이크(Shake)이벤트를 만들 수 있다. 메뉴에서 [Hardware ->

Shake Gesture]를 선택하면 세이크 이벤트를 만들어진다.
또한 [control]+[command]+[Z]를 눌러도 된다.


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

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

import UIKit

class ViewController: UIViewController {

    @IBOutlet var txtMessage: UILabel!
    @IBOutlet var txtTapsLevel: UILabel!
    @IBOutlet var txtTouchsLevel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }
    // 터치가 시작될 때 호출
    override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {
        //현재 터치 이벤트를 가져옴
        let touch = touches.first! as UITouch
        // 이벤트 종류를 출력
        txtMessage.text = "Touches Began"
        //터치의 갯수가져옴
        txtTapsLevel.text = String(touch.tapCount)
        //탭의 갯수 가져옴
        txtTouchsLevel.text = String(touches.count)
    }
    // 터치된 손가락이 움직였을 때 호출된다.
    override func touchesMoved(_ touches: Set<UITouch>, with event:
UIEvent?) {
        let touch = touches.first! as UITouch

        txtMessage.text = "Touches Moved"
        txtTapsLevel.text = String(touch.tapCount)
        txtTouchsLevel.text = String(touches.count)
    }
```

```
    // 화면에서 손가락을 떼었을 때 호출
    override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {
        let touch = touches.first! as UITouch

        txtMessage.text = "Touches Ended"
        txtTapsLevel.text = String(touch.tapCount)
        txtTouchsLevel.text = String(touches.count)
    }
}
```

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

import UIKit

class ViewController: UIViewController {

    @IBOutlet var imgView: UIImageView!
    //바로 전에 터치하거나 이동한 위치
    var lastPoint: CGPoint!
    //선의 두께
    var lineSize:CGFloat = 2.0
    //선의 색상
    var lineColor = UIColor.red.cgColor

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    @IBAction func clearImageView(_ sender: UIButton) {
        // clear버튼을 누르면 화면을 지우기 위해 이미지뷰에 nil
        imgView.image = nil
    }
    override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {
```

```
        // 현재 발생한 터치이벤트를 가져옴
        let touch = touches.first!
        // 터치된 위치를 lastPoint라는 변수에 저장된다
        lastPoint = touch.location(in: imgView)

    }
    override func touchesMoved(_ touches: Set<UITouch>, with event:
UIEvent?) {

        UIGraphicsBeginImageContext(imgView.frame.size)
        UIGraphicsGetCurrentContext()?.setStrokeColor(lineColor)
       /* 라인의 끝 모양을 라운드로 설정한다 */
UIGraphicsGetCurrentContext()?.setLineCap(CGLineCap.round)
        // 선의 크기 지정
        UIGraphicsGetCurrentContext()?.setLineWidth(lineSize)
        //현재 발생한 터치 이벤트를 가지고 온다
        let touch = touches.first! as UITouch
        // 터치된 위치를 가지고 온다
        let currPoint = touch.location(in: imgView)
        // 현재 이미지 뷰(Image View)에 있는 이미지를 이미지 뷰(Image
View)의 크기를 그린다.
        imgView.image?.draw(in: CGRect(x: 0, y: 0, width:
imgView.frame.size.width, height: imgView.frame.size.height))
        // 이전에 이동된 위치인 lastPint로 시작 위치를 이동시킨다.
        UIGraphicsGetCurrentContext()?.move(to: CGPoint(x: lastPoint.x, y:
lastPoint.y))
        // lastPoint에서 현재 위치인 currpoint까지 선을 추가한다
        UIGraphicsGetCurrentContext()?.addLine(to: CGPoint(x: currPoint.x,
y: currPoint.y))
        // 콘택스트에 추가
        UIGraphicsGetCurrentContext()?.strokePath()
        // 콘텍스트에 그려진 이미지를 가지고 와서 이미지 뷰에 나타낸다
        imgView.image =
            UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        // 터치된 위치를 lastPoint라는 변수에 할당한다.
        // 현재 위치를 lastPoint에 저장
```

```
            lastPoint = currPoint

        }
        // 화면에서 손가락을 떼었을 때 스케치도 끝나도록
        override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {

            UIGraphicsBeginImageContext(imgView.frame.size)
            UIGraphicsGetCurrentContext()?.setStrokeColor(lineColor)
           /* 라인의 끝 모양을 라운드로 설정한다 */
UIGraphicsGetCurrentContext()?.setLineCap(CGLineCap.round)
            // 선의 크기 지정
            UIGraphicsGetCurrentContext()?.setLineWidth(lineSize)
            // 현재 이미지 뷰(Image View)에 있는 이미지를 이미지 뷰(Image
View)의 크기를 그린다.
            imgView.image?.draw(in: CGRect(x: 0, y: 0, width:
imgView.frame.size.width, height: imgView.frame.size.height))
            // 이전에 이동된 위치인 lastPint로 시작 위치를 이동시킨다.
            UIGraphicsGetCurrentContext()?.move(to: CGPoint(x: lastPoint.x, y:
lastPoint.y))
            // lastPoint에서 현재 위치인 currpoint까지 선을 추가한다
            UIGraphicsGetCurrentContext()?.addLine(to: CGPoint(x: lastPoint.x, y:
lastPoint.y))
            // 콘택스트에 추가
            UIGraphicsGetCurrentContext()?.strokePath()
            // 콘텍스트에 그려진 이미지를 가지고 와서 이미지 뷰에 나타낸다
            imgView.image =
                UIGraphicsGetImageFromCurrentImageContext()
            //그림 그리기를 끝낸다
            UIGraphicsEndImageContext()
        }
        override func motionEnded(_ motion: UIEvent.EventSubtype, with event:
UIEvent?) {
            // shake가 발생한다면
            if motion == .motionShake {
                imgView.image = nil
            }
```

```
    }
}
```