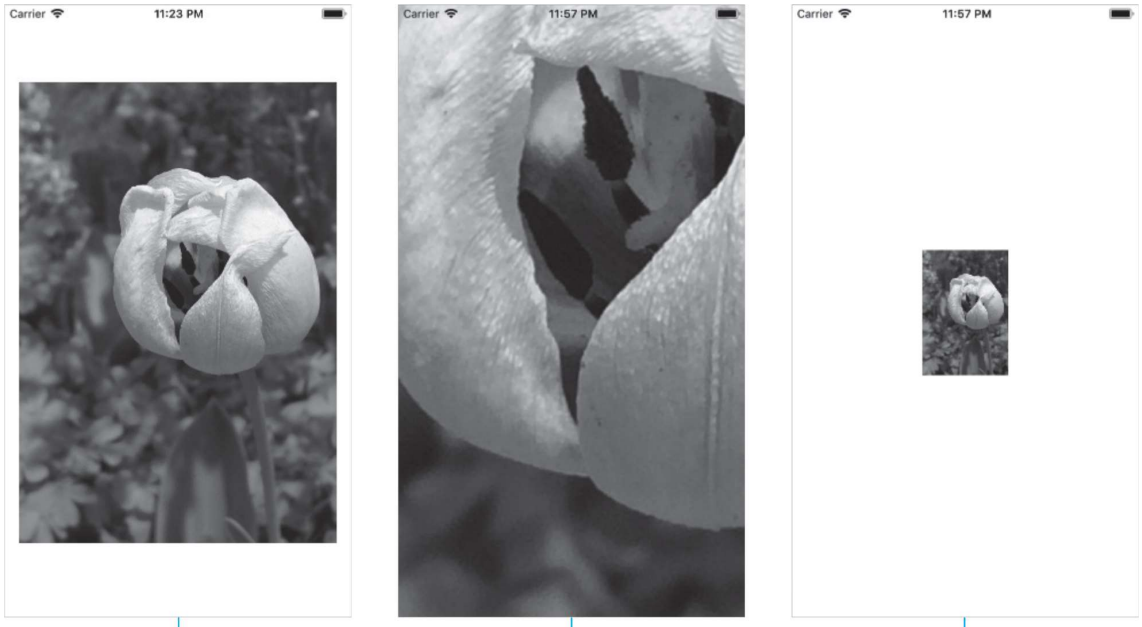
19 장. 핀치 제스처를 사용해 사진을 확대/축소 하기

핀치 제스처 또한 iOS 에서 가장 많이 사용하는 동작 중의 하나로, 엄지와 검지 두 손가락을 화면에 터치한 후 손가락의 간격을 벌리고 좁히고 동작으로 화면을 확대 / 축소할 때 많이 사용한다. 일반적으로 핀치 제스터는 갤러리의 사진을 확대 / 축소하거나 웹 페이지 또는 지도 등의 앱에서 화면을 볼 때 사용한다.

핀치 제스처를 등록하고 사용하는 방법을 알아보고 핀치 제스처를 사용해 텍스트와 이미지를 확대 / 축소해 보자.



화면을 두 손가락으로 터치한 후 손가락의 간격을 벌리거나 좁히면 이미지의 크기가 확대 / 축소됩니다.

19-1 핀치 제스처란?

핀치 제스처는 두 손가락으로 화면을 확대/축소할 때 사용하는 이벤트로, 화면 확대/축소 기능이 들어간 모든 앱에서 사용할 수 있다. 핀치 제스처를 사용하면 다음과 같은 유용한 앱을 만들 수 있다.

iOS 사진 앱

사파리(Safari) 앱

지도 앱

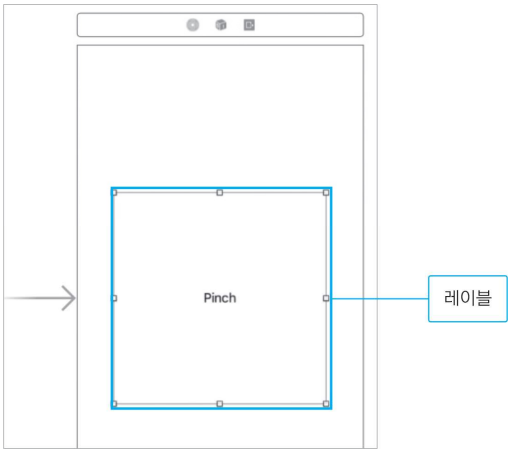
19-2 텍스트 핀치 앱을 위한 기본 환경 구성하기

핀치 제스터는 사진뿐만 아니라 텍스트에도 사용할 수 있다. 텍스트를 확대 / 축소할 수 있는 것이다.

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

2. 뷰 컨트롤러 크기 조절하기
아이폰 모양의 뷰 컨트롤러 크기를 상황에 맞게 조절한다.

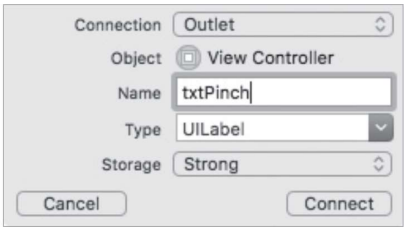
3. 스토리보드 꾸미기
오른쪽 아랫부분의 오브젝트 라이브러리에서 [레이블(Label)]을 찾아 스토리보드로 끌어와 다음 그림과 같이 배치한다.



4. 아웃렛 변수 추가하기
이제 프로그램에서 사용할 아웃렛 변수를 추가한다.
오른쪽 윗부분의 [Show the Assistant editor]버튼을 클릭하여 보조 편집기 영역을 연다.

5. 앞에서 추가한 텍스트 필드 객체에 아웃렛 변수를 추가한다, 위치와 설정은 아래 표를 참고한다.

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



19-3 텍스트 핀치 앱 구현하기

1. 스탠더드 에디터로 화면 보드 수정하기

코딩을 위해 화면 모드를 수정한다. 오른쪽 윗부분에서 [Show the Standard editor] 버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewCotroller.swift]를 선택한다.

2. 글자 크기 변수 선언하기

핀치 제스처가 발생했을 때 현재 글자 크기를 저장해 보자. 이를 위해 글자 크기 변수를 선언한다.

```
1  class ViewController: UIViewController {
2      @IBOutlet var txtPinch: UILabel!
3
4      var initialFontSize:CGFloat!
5  ...
6  }
```

4 행을 추가한다.

3. 핀치 제스처 등록하기

핀치 제스처는 UIPinchGestureRecognizer 클래스에 의해 인식되므로 이 클래스를 선언하고 핀치 제스처를 등록해야 한다. 다음 소스를 viewDidLoad 함수 안에 입력한다.

```
1  override func viewDidLoad() {
2          super.viewDidLoad()
3          // Do any additional setup after loading the view, typically from a nib.
           let pinch = UIPinchGestureRecognizer(target: self, action:
4  #selector(ViewController.doPinch(_:)))
           self.view.addGestureRecognizer(pinch)
5  }
6
```

4 행은 UIPinchGestureRecognizer 클래스 상수 pinch 를 선언한다. 액션 인수는 핀치 제스처가 인식되었을 때 실행할 메서드를 의미한다.

5 행은 뷰 객체의 addGestureRecognizer 메서드를 사용해 핀치 제스처를 등록한다.

4. 액션 메서드 구현하기

이제 핀치 제스처가 인식되었을 때 실행할 액션 메서드를 구현해 보겠다. 핀치 제스처가 처음 시작하는 상태라면 현재 글자 크기를 저장하고, 핀치 제스처가 계속 진행 중이라면 저장된 글자 크기를 확대하거나 축소할 것이므로 if 문을 사용한다.

```
1  @objc func doPinch(_ pinch: UIPinchGestureRecognizer) {
2          if (pinch.state == UIGestureRecognizerState.began) {
3              initialFontSize = txtPinch.font.pointSize
4          } else {
5              txtPinch.font = txtPinch.font.withSize(initialFontSize * pinch.scale)
```

```
            }
}
```

2 행은 우선 핀치 제스처의 상태를 state 속성을 사용하여 확인한다

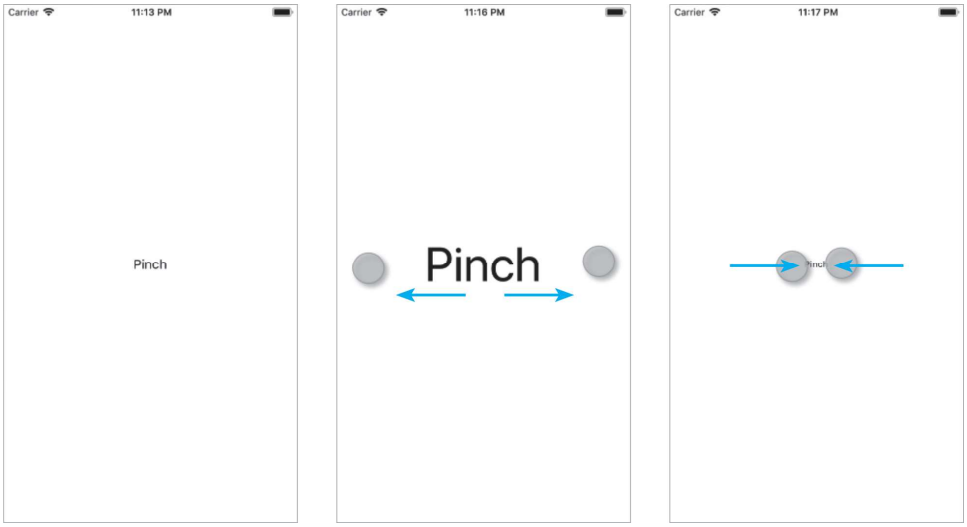
3 행은 핀치 제스처의 상태가 시작이면 앞에서 선언한 initialFontSize 변수에 현재 텍스트의 글자 크기를 저장한다.

5 행은 핀치 제스처의 상태가 시작이 아니라면 핀치 제스처가 계속 진행되고 있는 상태이므로 initialFontSize 에 저장해 둔 글자 크기 값에 scale 속성을 곱하여 텍스트의 글자의 크기에 반영한다.

5. 결과 보기

[실행] 버튼을 클릭하여 iOS 시뮬레이터를 실행한다. 프로그램이 실행되면 화면을 두 손으로 터치한 후 손가락의 간격을 벌리거나 좁히면 텍스트의 크기가 확대 / 축소된다.

시뮬레이터에서는 [option]키를 누른 채 마우스로 드래그하면 된다.



19-4 이미지 핀치 앱을 위한 기본 환경 구성하기

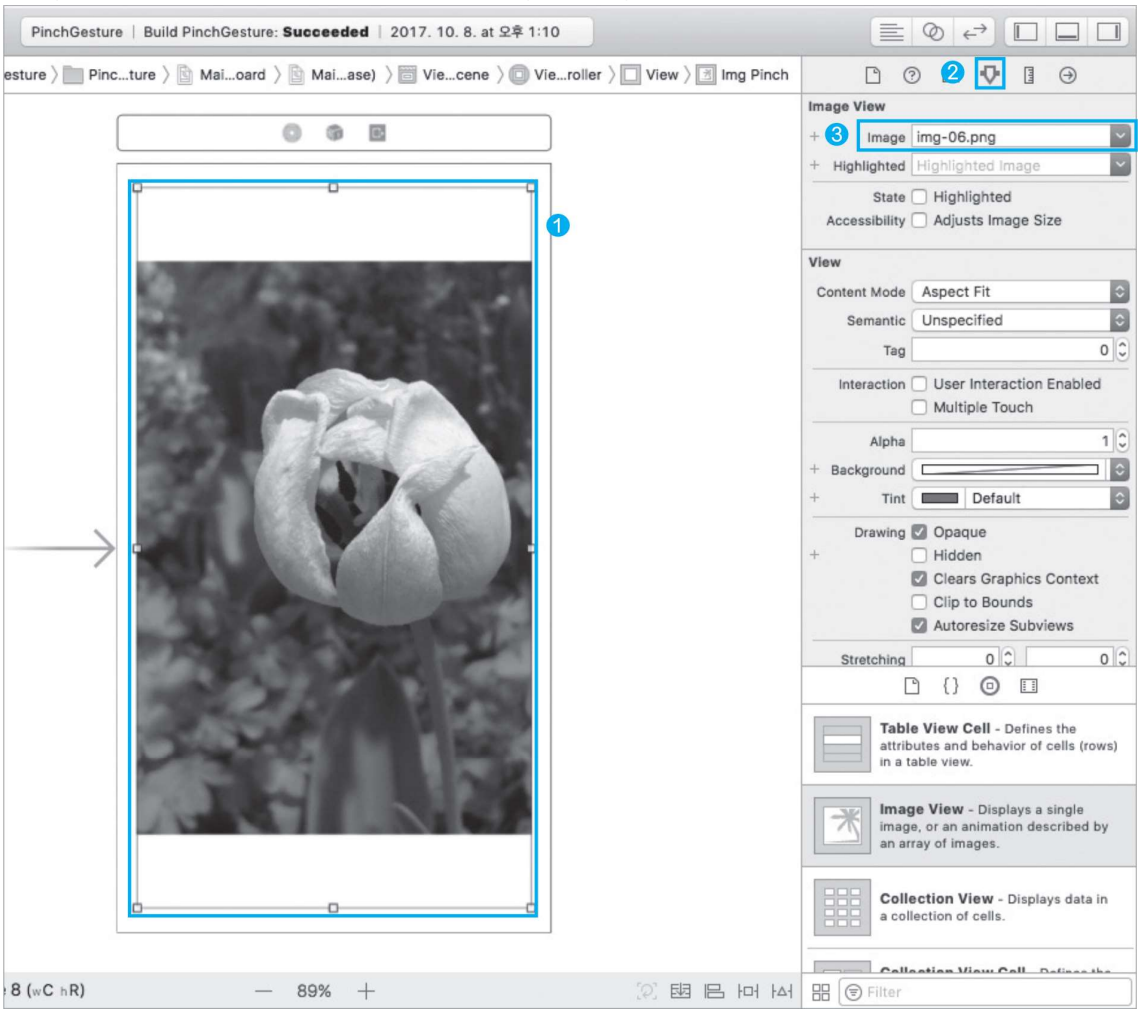
핀치 제스처를 사용해 이미지를 확대 / 축소하는 앱을 만들어 보자.

1. 이미지를 확대 / 축소하기 위해 프로젝트에 'img-06.png'이미지를 추가한다.

2. 스토리보드 수정하기

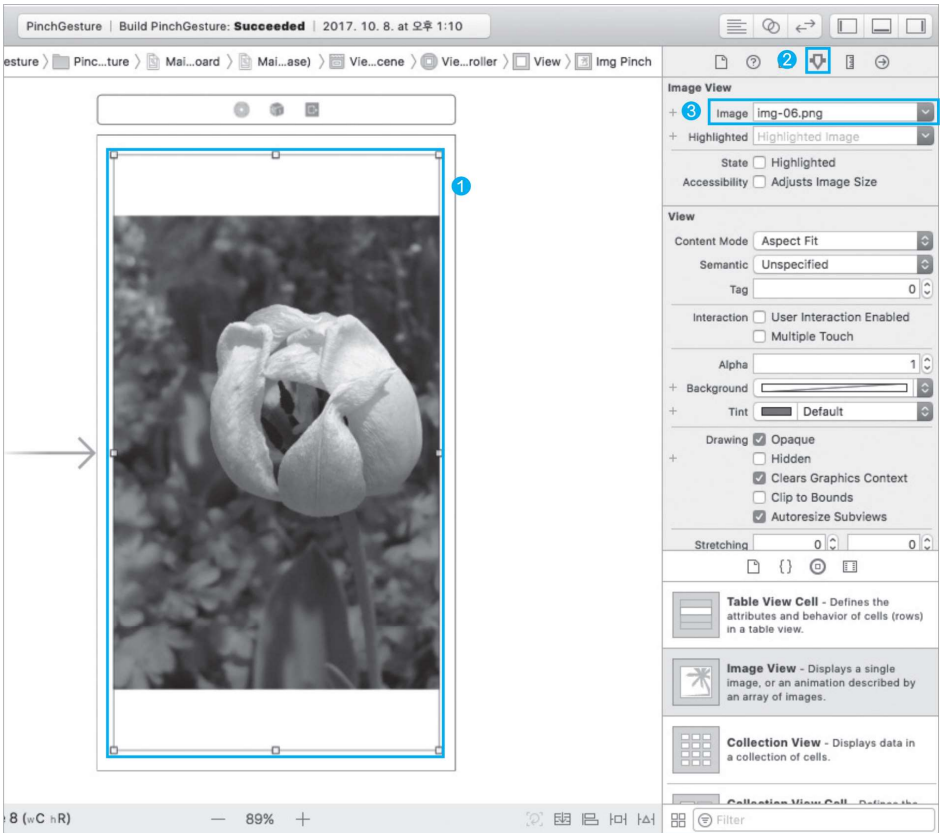
스토리보드에서 앞의 예제에서 추가한 [레이블(Label)]을 삭제한 후 오른쪽 아랫부분의 오브젝트 라이브러리에서 [이미지 뷰(Image View)]를 찾아 스토리보드로 끌어와 아래 그림과 같이 배치한다.

3. 이미지 뷰를 선택한 후 오른쪽 윗부분의 [Atrributes inspector]버튼을 클릭하고 image 에서 앞에서 추가한 이미지인 [img-06.png]를 선택한다.



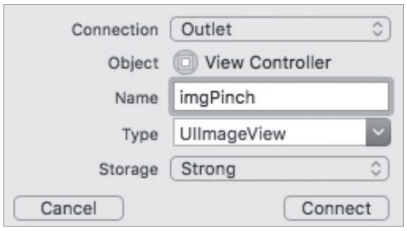
4. 아웃렛 변수 추가하기

이제 프로젝트에서 사용할 아웃렛 변수를 추가한다. 아웃렛 변수와 액션 함수를 추가하기 위해 오른쪽 윗부분의 [Show the Assistant editor]버튼을 클릭하여 보조 편집기 영역을 연다.



5. 앞에서 추가한 이미지 뷰에 아웃렛 변수를 추가한다. 텍스트 핀치 앱을 만들 때 추가했던 레이블의 아웃렛 변수를 삭제한 후, [이미지 뷰(Image View)]를 마우스 오른쪽 버튼으로 클릭하여 보조 편집기 영역에 갖다 놓는다.

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



19-5 이미지 핀치 앱 구현하기

텍스트 핀치 앱처럼 이미지 핀치 앱을 구현해 보자

1.스텐더드 에디터로 화면 모드를 수정하기

코딩을 위해 화면 모드를 수정한다. 오른쪽 윗부분에서 [Show the Standard editor]버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.

2. 액션 메서드 구현하기

핀치 제스처를 등록할 때 입력한 액션 인수는 핀치 제스처가 인식되었을 때 실행할 메서드의 인수를 의미한다. 그럼 핀치 제스처가 인식되었을 때 실행할 액션 메서드를 구현하자. doPinch 함수 안에 텍스트 핀치 앱을 만들 때 입력했던 부분을 지우고 다음 내용으로 대체한다.

```
1  @objc func doPinch(_ pinch: UIPinchGestureRecognizer) {
2          imgPinch.transform = imgPinch.transform.scaledBy(x: pinch.scale, y:
   pinch.scale)
3          pinch.scale = 1
4  }
```

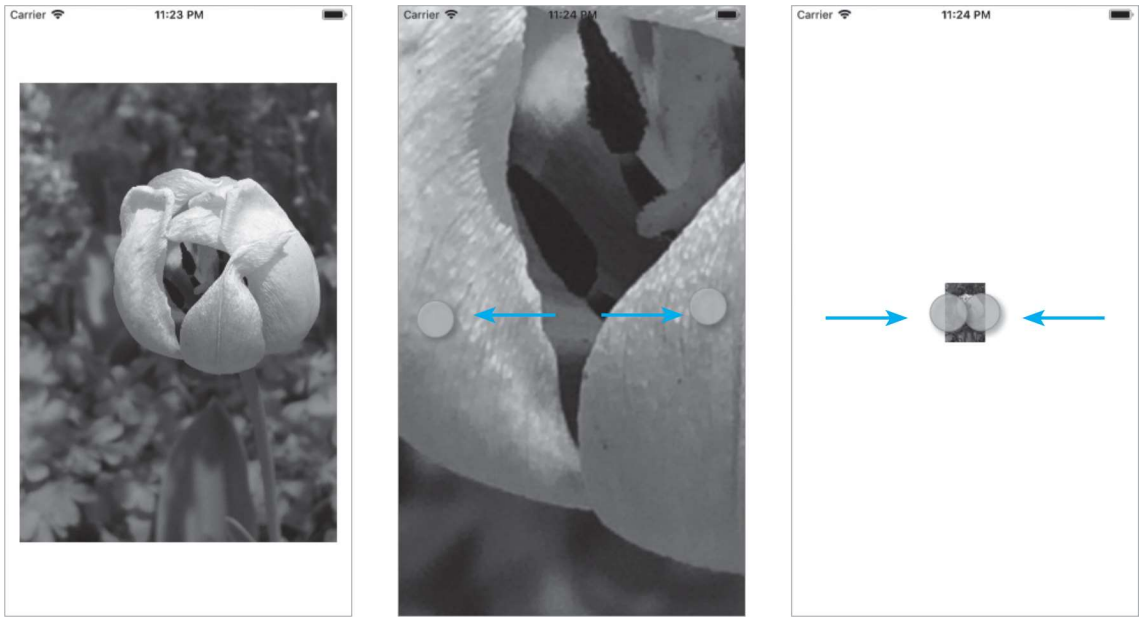
2 행은 이미지 imgPinch 를 scale 에 맞게 변한다,

3 행은 다음 변환을 위하여 핀치(pinch)의 스케일(scale)속성을 1 로 설정한다.

3. 결과 보기

[실행] 버튼을 클릭하여 앱을 실행한다. 화면을 두 손가락으로 터치한 후 손가락의 간격을 벌리거나 좁히면 이미지의 크기가 확대 / 축소된다.

[option]을 누른 채 드래그하면 된다.



텍스트 핀치 앱

```
/
//   ViewController.swift
//   PinchGesture
//
//   Created by SoongM00 Rhee on 2018. 12. 7..
//   Copyright © 2018년  SoongMoo Rhee. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var txtPinch: UILabel!
    // 글자 크기를 지정하기 위한 변수 선언
    var initialFontSize:CGFloat!
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        //UIPinchGestureRecognizer클래스로 핀치제스처 생성
        //
        let pinch = UIPinchGestureRecognizer(target: self, action:
#selector(ViewController.doPinch(_:)))
        //핀치 제즈처를 ViewController에 등록
        self.view.addGestureRecognizer(pinch)
```

```
    }
    @objc func doPinch(_ pinch: UIPinchGestureRecognizer)
    // UIPinchGestureRecognizer클래스의 pinch 매개변수로 핀치제스처를
받아옴
    {
        //핀치제스처의 state속성을 사용하여 상태를 확인
        // 시직 상태라면
        if (pinch.state == UIGestureRecognizer.State.began) {
            // 핀치제스처가 확인되면 현재 사이즈를 initialFontSize 변수에
저장
            initialFontSize = txtPinch.font.pointSize
        } else { // 계속 진행 상태라면 initialFontSize에 scale만큼 곱한
크기로 변경
            txtPinch.font = txtPinch.font.withSize(initialFontSize * pinch.scale)
        }
    }
}
```

이미지 핀치 앱

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet var imgPinch: UIImageView!

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

        let pinch = UIPinchGestureRecognizer(target: self, action:
#selector(ViewController.doPinch(_:)))
        self.view.addGestureRecognizer(pinch)
    }

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

    @objc func doPinch(_ pinch: UIPinchGestureRecognizer) {
        imgPinch.transform = imgPinch.transform.scaledBy(x: pinch.scale, y: pinch.scale)
        pinch.scale = 1
    }
}
```