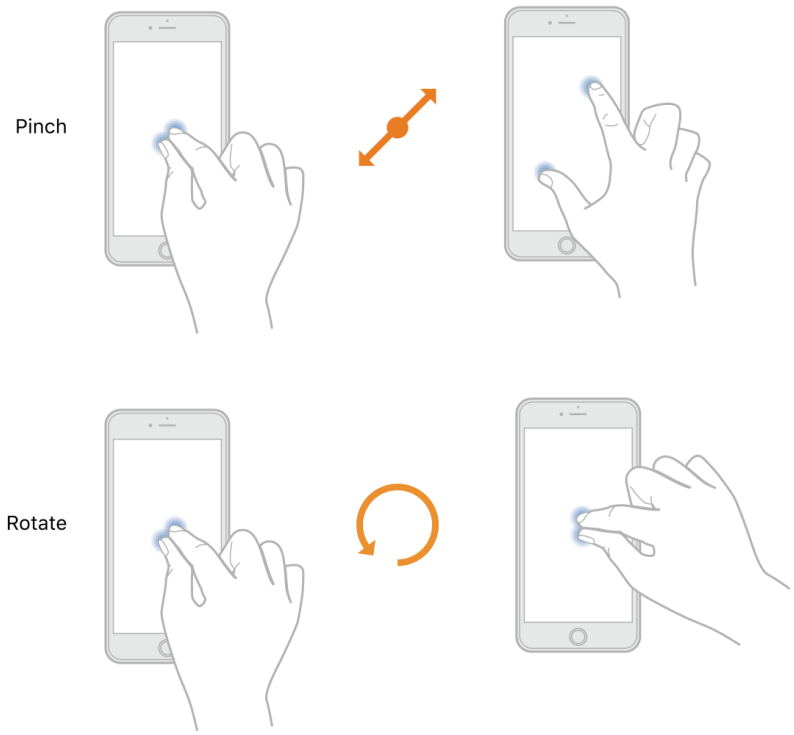
Handling Pinch and Rotation Gestures

On This Page

Pinch and rotation gestures are continuous gestures that track the movement of the first two fingers touching the screen.

- A *pinch gesture* occurs when the distance between two fingers changes. Use the `UIPinchGestureRecognizer` class to detect pinch gestures.
- A *rotation gesture* occurs when two fingers rotate around each other. Use the `UIRotationGestureRecognizer` class to detect rotation gestures.

Figure 4-1 Pinch and rotation gestures



You can attach a gesture recognizer in one of these ways:

- Programmatically. Call the `addGestureRecognizer:` method of your view.
- In Interface Builder. Drag the appropriate object from the library and drop it onto your view

Handling a Pinch Gesture

A pinch gesture recognizer reports changes to the distance between two fingers touching the screen. Pinch gestures are continuous, so your action method is called each time the distance between the fingers changes. The distance between the fingers is reported as a scale factor. At the beginning of the gesture, the scale factor is `1.0`. As the distance between the two fingers increases, the scale factor increases proportionally. Similarly, the scale factor decreases as the distance between the fingers decreases. Pinch gestures are used most commonly to change the size of objects or content onscreen. For example, map views use pinch gestures to change the zoom level of the map.

A pinch gesture recognizer enters the `UIGestureRecognizerStateBegan` state only after the distance between the two fingers changes for the first time. After that initial change, subsequent changes to the distance put the gesture recognizer into the `UIGestureRecognizerStateChanged` state and update the scale factor. When the user's fingers lift from the screen, the gesture recognizer enters the `UIGestureRecognizerStateEnded` state.

> IMPORTANT
>
> Take care when applying a pinch gesture recognizer's scale factor to your content, or you might get unexpected results. Because your action method may be called many times, you cannot simply apply the current scale factor to your content. If you multiply each new scale value by the current value of your content, which has already been scaled by previous calls to your action method, your content will grow or shrink

> exponentially. Instead, you must either cache the original value of your content, and apply each new scale factor to that value, or you must reset the scale factor to `1.0` after applying each new change.

On This Page

Listing 4-1 demc
applies the current scale factor to the view's transform and then resets the gesture recognizer's `scale` property to `1.0`. Resetting the scale factor causes the gesture recognizer to report only the amount of change since the value was reset, which results in linear scaling of the view.

Listing 4-1 Scaling a view

```
@IBAction func scalePiece(_ gestureRecognizer : UIPinchGestureRecognizer) {
    // Move the anchor point of the view's layer to the touch point
    // so that scaling the view and layer becomes simpler.
    self.adjustAnchorPoint(gestureRecognizer: gestureRecognizer)

    // Scale the view by the current scale factor.
    if gestureRecognizer.state == .began || gestureRecognizer.state == .changed {
        gestureRecognizer.view?.transform =
            (gestureRecognizer.view?.transform.scaledBy(x: gestureRecognizer.scale,
                                                        y:
  gestureRecognizer.scale))!
        // Set the scale factor to 1.0 to avoid exponential growth
        gestureRecognizer.scale = 1.0
    }
}
```

If the code for your pan gesture recognizer is not called, or is not working correctly, check to see if the following conditions are true, and make corrections as needed:

- The `userInteractionEnabled` property of the view is set to `YES`. Image views and labels set this property to `NO` by default.
- At least two fingers are touching the screen.
- You are applying scale factors to your content correctly. Exponential growth of a value happens when you simply apply the scale factor to the current value.

Handling a Rotation Gesture

Use a rotation gesture recognizer when you want to use rotational movements as input to your app. Rotational gestures are commonly used to manipulate objects onscreen. For example, you might use them to rotate a view or update the value of a custom control. Rotation gestures are continuous, so your action method is called whenever the rotation value changes, giving you a chance to update your content.

The gesture recognizer reports rotation values in radians. If you imagine a line between the user's fingers, the line created by the fingers at their initial positions represents the initial point for measurements, and therefore represents a rotation angle of 0. As the user's fingers move, a new line is created between the fingers at each new location. The gesture recognizer measures the angle between the initial line and each new line and places the resulting value in its `rotation` property.

A rotation gesture recognizer enters the `UIGestureRecognizerStateBegan` state as soon as the position of the user's fingers changes in a way that indicates that rotation has begun. After that initial change, subsequent changes cause the gesture recognizer to enter the `UIGestureRecognizerStateChanged` state and update the angle of rotation. When the user's fingers lift from the screen, the gesture recognizer enters the `UIGestureRecognizerStateEnded` state.

> IMPORTANT
>
> Take care when applying rotation values to your content, or you might get unexpected results. The rotation reported by the gesture recognizer represents the angle between the current finger position and the initial finger position. If you apply each new rotation value as is to your content, each new value compounds the previous one, causing your content to rotate too fast. Instead, you must either cache the original value of your content and modify that value, or you must reset the `rotation` property to `0.0` after applying each new change.

Listing 4-2 demonstrates how to use a rotation gesture recognizer to rotate a view in a way that matches the user's fingers. The method sets the view's anchor point to the center point of the user's two fingers, which causes the view to rotate around that point and appear more natural. After applying the rotation value, the method resets the rotation property to `0.0`. The next time this method is called, the gesture recognizer reports only the new amount of rotation since the beginning of the gesture.

On This Page

Listing 4-2 Rotating a view

```
@IBAction func rotatePiece(_ gestureRecognizer : UIRotationGestureRecognizer) {
    // Move the anchor point of the view's layer to the center of the
    // user's two fingers. This creates a more natural looking rotation.
    self.adjustAnchorPoint(gestureRecognizer: gestureRecognizer)

    // Apply the rotation to the view's transform.
    if gestureRecognizer.state == .began || gestureRecognizer.state == .changed {
        gestureRecognizer.view?.transform =
            (gestureRecognizer.view?.transform.rotated(by:
  gestureRecognizer.rotation))!

        // Set the rotation to 0 to avoid compounding the
        // rotations in the view's transform.
        gestureRecognizer.rotation = 0.0
    }
}
```

If the code for your rotation gesture recognizer is not called, or is not working correctly, check to see if the following conditions are true, and make corrections as needed:

- The `userInteractionEnabled` property of the view is set to `YES`. Image views and labels set this property to `NO` by default.
- At least two fingers are touching the screen.
- You are applying rotation factors to your content correctly. Over-rotation happens when you apply the same rotation value more than once. To fix this problem, set the `rotation` property to `0.0` after applying the current rotation value to your content.