Sample Code

Illustrating the force, altitude, and azimuth properties of touch input

Capture Apple Pencil and touch input in views.

Download

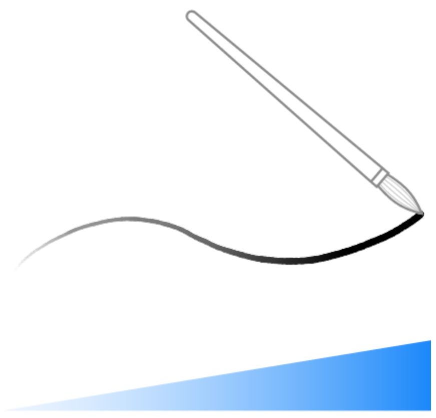
iOS 11.0+ | iPadOS 11.0+ | Xcode 10.1+

Overview

Touch Canvas illustrates responsive handling of Apple Pencil and touch input, focusing on the force, altitude, and azimuth properties of `UITouch`. The sample creates a visualization of force using line thickness, and creates a visualization of altitude and azimuth with an interactive diagram. To build on the concepts demonstrated in this sample and learn about using Apple Pencil and touch input in a drawing app, see Leveraging touch input for drawing apps.

Calculate the force of a touch

You can use the force of a touch applied by a finger on 3D Touch-enabled devices or from the tip of Apple Pencil to create effects in your app. For example, the force of a touch can change the width of a line on a canvas.



The current force is reported by the `force` property of `UITouch`.

```
force = touch.force
```

The force value input affects the result of handling a `UITouch`. In this sample, force is interpreted as a value representing the magnitude of a point in a line, including a lower bound on the force value usable by the app.

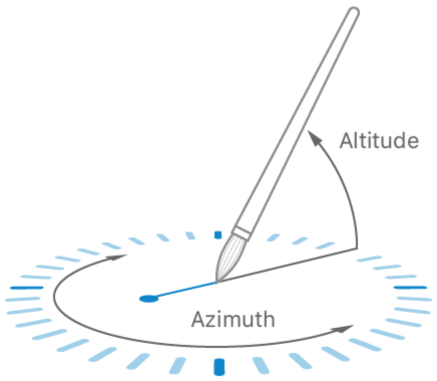
```
var magnitude: CGFloat {
    return max(force, 0.025)
}
```

This sample uses the magnitude value to affect drawing on the canvas, including the line width value.

```
context.setLineWidth(point.magnitude)
```

Create a visualization of Apple Pencil's altitude and azimuth

Touch Canvas contains a visualization of the altitude and azimuth for Apple Pencil as you draw on the screen when the *Debug* mode is enabled. This visualization is a diagram which continuously updates based on Apple Pencil's motion.



Apple Pencil reports its altitude as an angle relative to the device surface through the altitudeAngle property on UITouch.

```
let altitudeAngle = touch.altitudeAngle
```

In this sample project, the line length extends to the edge of the diagram when Apple Pencil is fully horizontal. If Apple Pencil is perfectly vertical, the line length reduces to a dot under Apple Pencil's tip. The line length calculation transforms the altitude angle relative to the radius of the diagram.

```
/*
 Make the length of the indicator's line representative of the `altitudeAngle`. When
 zero radians (parallel to the screen surface) the line will be at its longest. At `
 only the dot on top of the indicator will be visible directly beneath the touch loc
 */
let altitudeRadius = (1.0 - altitudeAngle / ( CGFloat.pi / 2)) * radius
var lineTransform = CGAffineTransform(scaleX: altitudeRadius, y: 1)
```

Apple Pencil reports its direction, or azimuth, relative to the view Apple Pencil interacts with. A drawing app might use azimuth information to change the shape or strength of a particular drawing tool. Access the azimuth information with the azimuthAngle(in:) and azimuthUnitVector(in:) methods of UITouch.

```
let azimuthAngle = touch.azimuthAngle(in: canvasView)
let azimuthUnitVector = touch.azimuthUnitVector(in: canvasView)
```

The interactive diagram demonstrates how altitude, azimuth angle, and azimuth unit vector values can be used together. Here, the azimuth angle rotates around the diagram opposite the actual azimuth value, and the dot at the end of the altitude line moves by combining the altitude and azimuth unit vector properties. A transform efficiently applies the calculated rotation of the line and position of the dot to the diagram so that it remains responsive to small changes in Apple Pencil's position.

```
// Draw the azimuth indicator line as opposite the azimuth by rotating `.pi` radians
var rotationTransform = CGAffineTransform(rotationAngle: azimuthAngle)
rotationTransform = rotationTransform.rotated(by: CGFloat.pi)

var dotPositionTransform = CGAffineTransform(translationX: -azimuthUnitVector.dx * a
dotPositionTransform = dotPositionTransform.concatenating(centeringTransform)
```

Toggle debug drawing

Touch Canvas contains a debug drawing mode that allows you to view the operation of the properties in detail for different types of input, such as the difference between strokes drawn at different speeds with Apple Pencil. The debug mode enables the interactive diagram for altitude and azimuth, and changes the color of individual line segments to identify if the `UIEvent` for the line segment included data from `predictedTouches(for:)` or `coalescedTouches(for:)`.

The sample uses the double-tap feature of the second generation Apple Pencil to toggle *Debug* mode when the user configures the preferred double tap action to switch tools. The sample app ignores the other preferred actions. See Apple Pencil interactions for more information.

```
func pencilInteractionDidTap(_ interaction: UIPencilInteraction) {
    guard UIPencilInteraction.preferredTapAction == .switchPrevious else { return }

    /* The tap interaction is a quick way for the user to switch tools within an app
     Toggling the debug drawing mode from Apple Pencil is a discoverable action, as
     for debug mode is on screen and visually changes to indicate what the tap inte
     */
    toggleDebugDrawing(sender: debugButton)
}
```

See Also

Touches

Handling touches in your view

Use touch events directly on a view subclass if touch handling is intricately linked to the view's content.

Handling input from Apple Pencil

Learn how to detect and respond to touches from Apple Pencil.

Tracking the force of 3D Touch events

Manipulate your content based on the force of touches.

Leveraging touch input for drawing apps

Capture touches as a series of strokes and render them efficiently on a drawing canvas.

`class UITouch`

An object representing the location, size, movement, and force of a touch occurring on the screen.