Introducing PencilKit

Will Thimbleby, Pencil and Paper Engineer
Jenny Chen, Pencil and Paper Engineer
Pencil is the most unique aspect of iPad
New in iOS 13

Lower latency
New tool palette
PencilKit
Markup everywhere
Great pencil experiences
PencilKit
Markup everywhere
Great pencil experiences
Great pencil experiences fully use Pencil
Fully Using Pencil

Precision enables new user experiences

Force, azimuth, and altitude allow expressive marks

Pencil taps switch modes
Three Types of Pencil

1st Gen

2nd Gen

Crayon
### Three Types of Pencil

<table>
<thead>
<tr>
<th>Feature</th>
<th>1st Gen</th>
<th>2nd Gen</th>
<th>Crayon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Azimuth/Altitude</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Force</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Tap Gesture</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
# Building Great Pencil Experiences

- Rich touch data from UIKit
- Fast rendering in Metal
- Async performance with GCD

<table>
<thead>
<tr>
<th>Topic</th>
<th>WWDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leveraging Touch Input on iOS</td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Metal Performance Optimization Techniques</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>Modernizing Grand Central Dispatch Usage</td>
<td>WWDC 2017</td>
</tr>
</tbody>
</table>
How Pencil Works
The Impact of How Pencil Works

Azimuth / altitude may be estimated

Azimuth is imprecise when pencil is perpendicular

Force data is delayed
Back-fill Azimuth and Altitude
Back-fill Azimuth and Altitude
Back-fill Azimuth and Altitude
Back-fill Azimuth and Altitude
Back-fill Azimuth and Altitude
Handle Force Updates After Touches Ended
Handle Force Updates After Touches Ended
Handle Force Updates After Touches Ended
Handle Force Updates After Touches Ended
Handle Force Updates After Touches Ended
Latency Is Critical
Latency Is Critical
For Best Latency

- Render with Metal
- Use predicted touches
For Absolute Best Latency

- Render with Metal
- Use predicted touches

- Transparent Metal layers
- Overlay effects
We currently do not have a video that could replace this, see next as option.
We currently do not have a video that could replace this, see next as option.
Support Pencil Taps

UIPencilInteraction for listening to Pencil taps
Respect the user’s `preferredTapAction`

```swift
let interaction = UIPencilInteraction()
interaction.delegate = self
view.addInteraction(interaction)

func pencilInteractionDidTap(_ interaction: UIPencilInteraction) {
    switch UIPencilInteraction.preferredTapAction {
    // ...
    }
}
```
Support Pencil Taps

UIPencilInteraction for listening to Pencil taps
Respect the user’s `preferredTapAction`

```
let interaction = UIPencilInteraction()
interaction.delegate = self
view.addInteraction(interaction)

func pencilInteractionDidTap(_ interaction: UIPencilInteraction) {
    switch UIPencilInteraction.preferredTapAction {
    // ...
    }
}
```
Support Pencil Taps

UIPencilInteraction for listening to Pencil taps
Respect the user’s `preferredTapAction`

```swift
let interaction = UIPencilInteraction()
interaction.delegate = self
view.addInteraction(interaction)

func pencilInteractionDidTap(_ interaction: UIPencilInteraction) {
    switch UIPencilInteraction.preferredTapAction {
        // ...
    }
}
```
Support Pencil Taps

UIPencilInteraction for listening to Pencil taps

Respect the user’s `preferredTapAction`

```swift
let interaction = UIPencilInteraction()
interaction.delegate = self
view.addInteraction(interaction)

func pencilInteractionDidTap(_ interaction: UIPencilInteraction) {
    switch UIPencilInteraction.preferredTapAction {
        // ...
    }
}
```
Build a data model
Smooth the point data
Do most of this asynchronously
Achieve incredible latency
Pencil
Bitmap and vector eraser
Use all the coalesced touches
Handle estimated force
Avoid over-rendering to save power
Estimated azimuth/altitude
Use predicted touches
Render fast in Metal
Dark mode
Create tool and color picker UI
Design some beautiful inks
Tiling, scrolling and zoom
Crayon
Ensure double buffering
Undo/redo

Great Pencil Experiences

Use predicted touches
Build a data model
Support pencil taps
Achieve incredible latency
Pencil
Three lines of code
Three Lines of Code

```swift
let canvas = PKCanvasView(frame: bounds)
view.addSubview(canvas)
canvas.tool = PKInkingTool(.pen, color: .black, width: 30)
```
hello.
Demo
hello.
hello.
hello.
hello.
hello.
hello.
PKCanvasView
PKDrawing
PKToolPicker
PKTools
- PKInkingTool
- PKEraserTool
- PKLassoTool
PKCanvasView
PKDrawing
PKToolPicker
PKTools
  • PKInkingTool
  • PKEraserTool
  • PKLassoTool
PKCanvasView

UIScrollView for panning and zooming

Data model is available via **drawing**

Interaction mode is changed via **tool**

Delegate methods notify about changes
PKDrawing

Serializable to data
Generates images
Can be appended or transformed
Available on macOS
func thumbnail(drawing: PKDrawing, thumbnailRect: CGRect, traitCollection: UITraitCollection) {
    thumbnailQueue.async {
        traitCollection.performAsCurrent {
            let image = drawing.image(from: thumbnailRect, scale: 2.0)
            DispatchQueue.main.async {
                self.updateThumbnail(drawing, image: image)
            }
        }
    }
}
func thumbnail(drawing: PKDrawing, thumbnailRect: CGRect, traitCollection: UITraitCollection) {
    thumbnailQueue.async {
        traitCollection.performAsCurrent {
            let image = drawing.image(from: thumbnailRect, scale: 2.0)
            DispatchQueue.main.async {
                self.updateThumbnail(drawing, image: image)
            }
        }
    }
}
func thumbnail(drawing: PKDrawing, thumbnailRect: CGRect, traitCollection: UITraitCollection) {
    thumbnailQueue.async {
        traitCollection.performAsCurrent {
            let image = drawing.image(from: thumbnailRect, scale: 2.0)
            DispatchQueue.main.async {
                self.updateThumbnail(drawing, image: image)
            }
        }
    }
}
func thumbnail(drawing: PKDrawing, thumbnailRect: CGRect, traitCollection: UITraitCollection) {
    thumbnailQueue.async {
        traitCollection.performAsCurrent {
            let image = drawing.image(from: thumbnailRect, scale: 2.0)
            DispatchQueue.main.async {
                self.updateThumbnail(drawing, image: image)
            }
        }
    }
}
func thumbnail(drawing: PKDrawing, thumbnailRect: CGRect, traitCollection: UITraitCollection) {
    thumbnailQueue.async {
        traitCollection.performAsCurrent {
            let image = drawing.image(from: thumbnailRect, scale: 2.0)

            DispatchQueue.main.async {
                self.updateThumbnail(drawing, image: image)
            }
        }
    }
}
Tools

PKInkingTool
pen
Tools

PKInkingTool
marker
Tools

PKInkingTool
marker
Tools

PKInkingTool
pencil
Pen  Marker  Pencil
Tools

canvasView.tool = PKInkingTool(.pen, color: .blue, width: 10)
canvasView.tool = PKInkingTool(.pen, color: .blue, width: 10)
Tools

Default Width

Valid Width Range

hello → hello
Tools

PKEraserTool

vector  bitmap
PKLassoTool
canvasView.rulerActive = true
canvasView.rulerActive = true
Welcome to San Francisco
PKToolPicker
PKToolPicker
Responder-based visibility

Similar to the keyboard
Palette floats over everything
Visibility based on first responders
PKToolPicker
Responder-based visibility

```swift
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
```
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
PKToolPicker
Responder-based visibility

```swift
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
```
PKToolPicker
Responder-based visibility

```swift
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
```
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
PKToolPicker
Responder-based visibility

```swift
let toolPicker = PKToolPicker.shared(for: window)
toolPicker.addObserver(canvasView)
toolPicker.setVisible(true, forFirstResponder: canvasView)
canvasView.becomeFirstResponder()
```
Canvas is first responder
Tool picker is shown
Tool picker shows again

Dismiss Signature View Controller
Automatically resigns first responder
PKToolPicker
Responder-based visibility

Keep palette visible with other responders

```swift
public func set(_ visible: Bool for responder: UIResponder)
```
PKToolPicker
Responder-based visibility

Keep palette visible with other responders

```swift
public func set(_ visible: Bool for responder: UIResponder)
```
PKToolPicker
Tool Picker UI — Compact layout
PKToolPicker
Tool Picker UI — Compact layout
PKToolPicker
Tool Picker UI — Compact layout

Fixed and docked to the bottom
PKToolPicker
Tool Picker UI — Compact layout

iPad
PKToolPicker
Tool Picker UI — Compact layout
PKToolPicker
Tool Picker UI — Compact layout
PKToolPicker
Tool Picker UI — Compact layout

Listen for tool picker's frame changes

```swift
optional func toolPickerFramesObscuredDidChange(_ toolPicker: PKToolPicker)
```

Adjust content accordingly

```swift
public func frameObscured(in view: UIView) -> CGRect
```
Tool Picker
Tool Picker UI — Undo buttons
Tool Picker
Tool Picker UI — Undo buttons
Tool Picker
Tool Picker UI — Undo buttons
Advanced Behaviors
PKCanvasView
Delegate callbacks

- pencil/touch down

canvasViewDidBeginUsingTool
PKCanvasView
Delegate callbacks

- pencil/touch down
  - canvasViewDidBeginUsingTool
- pencil/touch up
  - canvasViewDidEndUsingTool
- final force values
  - canvasViewDrawingDidChange
PKCanvasView
Delegate callbacks

PKCanvasView -> Set drawing
PKCanvasView
Delegate callbacks
PKCanvasView
Delegate callbacks

PKCanvasView → Set drawing → canvasViewDidFinishRendering
one finger
one finger
# Scrolling and Touch

**PKCanvasView**

<table>
<thead>
<tr>
<th>allowsFingerDrawing</th>
<th>Drawing</th>
<th>Scrolling</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1 Finger and Pencil Draw</td>
<td>2 fingers scroll</td>
</tr>
<tr>
<td>false</td>
<td>Pencil Draws</td>
<td>1 finger scrolls</td>
</tr>
</tbody>
</table>
## Scrolling and Touch

**PKCanvasView**

```swift
var allowsFingerDrawing: Bool { get set }
```

<table>
<thead>
<tr>
<th>allowsFingerDrawing</th>
<th>Drawing</th>
<th>Scrolling</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1 Finger and Pencil Draw</td>
<td>2 fingers scroll</td>
</tr>
<tr>
<td>false</td>
<td>Pencil Draws</td>
<td>1 finger scrolls</td>
</tr>
</tbody>
</table>
# Scrolling and Touch

**PKCanvasView**

```swift
var allowsFingerDrawing: Bool { get set }
```

<table>
<thead>
<tr>
<th>allowsFingerDrawing</th>
<th>Drawing</th>
<th>Scrolling</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1 Finger and Pencil Draw</td>
<td>2 fingers scroll</td>
</tr>
<tr>
<td>false</td>
<td>Pencil Draws</td>
<td>1 finger scrolls</td>
</tr>
</tbody>
</table>
Scrolling and Touch
PKCanvasView

```swift
var drawingGestureRecognizer: UIGestureRecognizer { get }
```
canvasView.opaque = false
canvasView.backgroundColor = .clear

Drawing Over Content
Dark Mode

Fundamentals of Color Science
August 13, 2018 10:30 am

Estimating Color Differences
- CIE LUV space
  - Superseded by CIE Lab color space
  - Based on opponent process theory of color vision
  - Non-linear version of XYZ
    - Designed to be perceptually uniform
    - \( \Delta E_{ab} \) 3D euclidean distance

Using Colorimetry
- Spot Color
  - Process color produced by halftoning RGB/CMYK primaries
  - How can we create systems with end-to-end fidelity?
Dark Mode

canvasView.overrideUserInterfaceStyle = .light
Markup Everywhere
Screenshots
iPhone XR from $19.99/mo. or $479.
Two great ways to buy. Just trade in your current iPhone online or at an Apple Store.*

Buy > Learn more >

iPhone XR
All-screen design. Longest battery life ever in an iPhone. Fastest performance. Studio-quality photos.

Learn more > Buy >
iPhone XR from $19.99/mo. or $479.
Two great ways to buy. Just trade in your current
iPhone online or at an Apple Store.*
Buy  Learn more

iPhone XR
All-screen design. Longest battery life ever in an iPhone.
Fastest performance. Studio-quality photos.
Learn more  Buy
Fundamentals of Color Science
August 13, 2018 10:30 am

Estimating Color Differences
- CIE LUV space
  - marginally better, still used in video
  - superseded by
- CIE Lab color space
  - based on opponent process theory of color vision
  - non-linear version of XYZ
  - designed to be perceptually uniform
  - $\Delta E_{ab}$ - 3D euclidean distance
Fundamentals of Color Science
August 13, 2018 10:30 am

Estimating Color Differences

- CIE LUV space
  - marginally better, still used in video
    - superseded by

  - CIE Lab color space
  - based on opponent process theory of color vision
  - non-linear version of XYZ
  - designed to be perceptually uniform
  - \( \Delta E_{ab} \): 3D euclidean distance
Apple Park
Corporate Headquarters - 36 mi

Apple Park is the corporate headquarters of Apple Inc., located at 1 Apple Park Way in Cupertino, California, United States. It opened to employees in April 2017, while construction was still underway. Its research and development facilities are occupied with over 2,000 people. It replaced the original headquarters at 1 Infinite Loop, which opened in 1993.
Apple Park
Corporate Headquarters - 36 mi

Directions
44 min drive

Photos on Wikipedia
More Photos on Wikipedia

Address
Apple Inc.
One Apple Park Way
Cupertino, CA 95014
United States

Phone
+1 (408) 996-1010

Website
apple.com

Wikipedia
Open Wikipedia

Apple Park is the corporate headquarters of Apple Inc., located at 1 Apple Park Way in Cupertino, California, United States. It opened to employees in April 2017, while construction was still underway. Its research and development facilities are occupied with over 10,000 people. It replaced the original headquarters at 1 Infinite Loop, which opened in 1993.
Introducing Multiple Windows on iPad

UIScreenshotService on UIWindowScene

class DrawingViewController: UIScreenshotServiceDelegate {
    override func viewWillAppear() {
        super.viewWillAppear()
        view.window.windowScene.screenshotService.delegate = self
    }
}

WWDC 2019
func screenshotService(
    UIScreenshotService,
    generatePDFRepresentationWithCompletion: (Data?, Int, CGRect) -> Void)

Full content delivered as PDF data

Introducing PDFKit on iOS
Screenshots

indexOfCurrentPage
 iPhone Xe from $19.99/mo. or $479.
Two-great ways to buy. Just trade in your current!
iPhone online or at an Apple Store.*
Buy > Learn more >

iPhone X
All-screen design. Longest battery life ever in an iPhone.
Fastest performance. Studio-quality photos.
Learn more > Buy >
Screenshots
Screenshots

rectInCurrentPage

Expected in PDF coordinates
Screenshots

Expected in PDF coordinates

(rectInCurrentPage)

(0, 0)

View Coordinates
Screenshots

Expected in PDF coordinates

View Coordinates

PDF Coordinates

rectInCurrentPage
Build Your Own Engine
Build Your Own Engine

PencilKit

Other Great Pencil APIs
More Information

developer.apple.com/wwdc19/221