Quick Interaction Techniques for watchOS

Session 211

Tom Witkin watchOS Engineer
Miguel Sanchez watchOS Engineer
What is a quick interaction?
Glanceable

Actionable
Designing Great Apple Watch Experiences

Presidio

Wednesday 1:40PM

What’s New in watchOS 3

WWDC 2016 Videos
How long is a quick interaction?
How long is a quick interaction?
2 seconds
watchOS
1. New Features
1. New Features
2. Interaction Techniques
New WatchKit Features

APIs for quick interactions

Miguel Sanchez watchOS Engineer
watchOS 3 APIs
Glanceable, Actionable, Responsive
watchOS 3 APIs

Glanceable, Actionable, Responsive

Access new kinds of user input
watchOS 3 APIs

Glanceable, Actionable, Responsive

Access new kinds of user input

• Gesture recognizers
watchOS 3 APIs
Glanceable, Actionable, Responsive

Access new kinds of user input
- Gesture recognizers
- Digital Crown rotation
watchOS 3 APIs

Glanceable, Actionable, Responsive

Access new kinds of user input

• Gesture recognizers
• Digital Crown rotation

New ways of displaying and updating information
watchOS 3 APIs

Glanceable, Actionable, Responsive

Access new kinds of user input

• Gesture recognizers
• Digital Crown rotation

New ways of displaying and updating information

• Improved table navigation
watchOS 3 APIs
Glanceable, Actionable, Responsive

Access new kinds of user input
• Gesture recognizers
• Digital Crown rotation

New ways of displaying and updating information
• Improved table navigation
• Support for new Notifications Framework
watchOS 3 APIs
Glanceable, Actionable, Responsive

Access new kinds of user input
- Gesture recognizers
- Digital Crown rotation

New ways of displaying and updating information
- Improved table navigation
- Support for new Notifications Framework
- SceneKit and SpriteKit integration
Gesture Recognizers

Expanding your UI interactions
Gesture Recognizers
Gesture Recognizers

Discrete
Gesture Recognizers

Discrete

WKSwipeGestureRecognizer
Gesture Recognizers

Discrete

WKSwipeGestureRecognizer
WKTapGestureRecognizer
Gesture Recognizers

Discrete
WKSwipeGestureRecognizer
WKTapGestureRecognizer

Continuous
Gesture Recognizers

Discrete
- WKSwipeGestureRecognizer
- WKTapGestureRecognizer

Continuous
- WKPanGestureRecognizer
Gesture Recognizers

Discrete

WKSwipeGestureRecognizer
WKTapGestureRecognizer

Continuous

WKPanGestureRecognizer
WKLongPressGestureRecognizer
Tap Gesture Recognizer - A recognizer for tap gestures.

Swipe Gesture Recognizer - A recognizer for swipe gestures.


Pan Gesture Recognizer - A recognizer for pan gestures.
WKGestureRecognizer
Base Class
WKGestureRecognizer

Base Class

Associate recognizer with action method

```swift
IBAction func handleGesture(recognizer : WKGestureRecognizer?)
```
WKGestureRecognizer
Base Class

Associate recognizer with action method

```swift
IBAction func handleGesture(recognizer : WKGestureRecognizer?)
```

Recognizer properties

```swift
func locationInObject() -> CGPoint
func objectBounds() -> CGRect
var state : WKGestureRecognizerState
```
WKGestureRecognizerState
Discrete gestures

Action method called when recognized
WKGestureRecognizerState

Discrete gestures

Action method called when recognized
WKGestureRecognizerState

Discrete gestures

Action method called when recognized

- possible
  - .failed
  - .recognized
WKGestureRecognizerState

Continuous gestures

Action method called many times
WKGestureRecognizerState
Continuous gestures

Action method called many times
WKGestureRecognizerState

Continuous gestures

Action method called many times

- possible
  - failed
  - began
    - changed
      - cancelled
      - recognized
Tips for Using Gesture Recognizers
Tips for Using Gesture Recognizers

Add gestures to groups instead of controls
Tips for Using Gesture Recognizers

Add gestures to groups instead of controls
Set dependency relationships in storyboard
Tips for Using Gesture Recognizers

Add gestures to groups instead of controls
Set dependency relationships in storyboard
Some system gestures will have precedence
Tips for Using Gesture Recognizers

Add gestures to groups instead of controls
Set dependency relationships in storyboard
Some system gestures will have precedence
  • Scrolling
  • Force Touch
Digital Crown Rotation
Increasing speed and precision of interactions
WKInterfacePicker Refresher

Introduced in watchOS 2
WKInterfacePicker Refresher

Introduced in watchOS 2

You provide

- A list of `WKPickerItems`
- Picker style
WKInterfacePicker Refresher
Introduced in watchOS 2

You provide
  • A list of WKPickerItems
  • Picker style
WKInterfacePicker Refresher

Introduced in watchOS 2

You provide

- A list of `WKPickerItems`
- Picker style
WKInterfacePicker Refresher

Introduced in watchOS 2

You provide

- A list of WKPickerItems
- Picker style
WKInterfacePicker Refresher

Introduced in watchOS 2

You provide

• A list of WKPickerItems
• Picker style

WatchKit handles

• Look and feel of interface element
• Processing of Digital Crown events
• Calling action on item selection
Digital Crown Support
Direct access to rotation events
Digital Crown Support
Direct access to rotation events

Control your own UI elements with rotation values
Digital Crown Support
Direct access to rotation events

Control your own UI elements with rotation values

WK Crown Sequencer
• Gives rotations per second
Digital Crown Support

Direct access to rotation events

Control your own UI elements with rotation values

**WKCrownSequencer**
- Gives rotations per second

**WKCrownDelegate** Protocol
- Callbacks with rotational deltas
Rotations Per Second
Useful for physics based animations
Rotations Per Second
Useful for physics based animations

Access sequencer from interface controller

```swift
var crownSequencer: WK_CrownSequencer { get }
```
Rotations Per Second
Useful for physics based animations

Access sequencer from interface controller

```swift
var crownSequencer: WKCrownSequencer { get }
```

Get RPS from sequencer

```swift
var rotationsPerSecond: Double { get }
```
Rotational Deltas
Counting number of rotations
Rotational Deltas

Counting number of rotations

Argument in WK Crown Delegate callback

```swift
func crownDidRotate(sequencer: WK Crown Sequencer?, rotationalDelta: Double)
```
Rotational Deltas

Counting number of rotations

Argument in WK Crown Delegate callback

```swift
func crownDidRotate(sequencer: WK Crown Sequencer?, rotationalDelta: Double)
```

Maintain accumulator to determine number of rotations
Rotational Deltas
Counting number of rotations

Argument in WK Crown Delegate callback

```swift
func crownDidRotate(sequencer: WK Crown Sequencer?, rotationalDelta: Double)
```

Maintain accumulator to determine number of rotations

- 1 full rotation = accumulated deltas of 1
Rotational Deltas

Counting number of rotations

Argument in WK皇冠Delegate callback

```swift
func crownDidRotate(sequencer: WK皇冠Sequencer?, rotationalDelta: Double)
```

Maintain accumulator to determine number of rotations

- 1 full rotation = accumulated deltas of 1
- Half rotation = accumulated deltas of 0.5
Rotational Deltas

Counting number of rotations

Argument in WK Crown Delegate callback

```swift
func crownDidRotate(sequencer: WK Crown Sequencer?, rotationalDelta: Double)
```

Maintain accumulator to determine number of rotations

- 1 full rotation = accumulated deltas of 1
- Half rotation = accumulated deltas of 0.5

Determine how count corresponds to sensitivity of changes in your content
Reported Values for RPS and Deltas

Sign indicates direction
Reported Values for RPS and Deltas

Sign indicates direction
Reported Values for RPS and Deltas

Sign indicates direction

Positive up rotation
Reported Values for RPS and Deltas

Sign indicates direction

Positive up rotation
Negative down rotation
Reported Values for RPS and Deltas

Sign indicates direction

Positive up rotation
Negative down rotation

Values are orientation agnostic
Digital Crown Focus

Sequencer only reports values when focused
Digital Crown Focus

Sequencer only reports values when focused

Other elements can take over Digital Crown focus

- Picker
- Slider
- Scrolling controller
Digital Crown Focus

Sequencer only reports values when focused

Other elements can take over Digital Crown focus

- Picker
- Slider
- Scrolling controller

Can use a gesture to gain focus on your UI element
Digital Crown Focus

Sequencer only reports values when focused

Other elements can take over Digital Crown focus

- Picker
- Slider
- Scrolling controller

Can use a gesture to gain focus on your UI element

Focus WKGroundSequencer to receive values

```swift
func focus()
```
Vertical Paging
Faster table row detail navigation
Hierarchical Navigation
Hierarchical Navigation
Hierarchical Navigation
Hierarchical Navigation
Vertical Paging
Vertical Paging
Make Your App More Responsive

 Doesn’t require navigating back to table
Make Your App More Responsive

Doesn’t require navigating back to table

One checkbox in Storyboard
Make Your App More Responsive
Doesn’t require navigating back to table

One checkbox in Storyboard
Table must be using segues
Mostly the same
watchOS 3 Table API

Mostly the same

New method to segue into vertical paging

```swift
func performSegue(row: Int)
```
watchOS 3 Table API

 Mostly the same

New method to segue into vertical paging

```swift
func performSegue(row: Int)
```

WatchKit handles preheating of controllers

- Some controller lifecycle methods called when not yet visible
watchOS 3 Table API
Mostly the same

New method to segue into vertical paging

```swift
func performSegue(row: Int)
```

WatchKit handles preheating of controllers

- Some controller lifecycle methods called when not yet visible
Notifications

The original quick interactions
Notifications Refresher
Apple Watch integral part of notification flow
Notifications Refresher

Apple Watch integral part of notification flow

Quick interaction
Notifications Refresher

Apple Watch integral part of notification flow

Quick interaction

- User feels a haptic
Notifications Refresher

Apple Watch integral part of notification flow

Quick interaction
- User feels a haptic
- Sees a Short Look
Notifications Refresher

Apple Watch integral part of notification flow

Quick interaction

• User feels a haptic
• Sees a Short Look
• Sees Long Look after short interval
User Response to Notification
User Response to Notification

Invitation
Housewarming Party and...
From
Hugo Verweij

When
Saturday, Jan. 17
3PM – 4PM

Where
Our new place!
180 Montgomery
San Francisco CA

Accept
Maybe
Decline
Dismiss
User Response to Notification

Dismiss notification
User Response to Notification

Dismiss notification
Launch watch app
User Response to Notification

Dismiss notification
Launch watch app
Notifications support custom actions
User Response to Notification

Dismiss notification
Launch watch app

Notifications support custom actions
• Foreground actions make app active
• Background actions routed to iPhone app in watchOS 2
## watchOS 3 Notification Support

**New User Notification Framework**

### Introduction to Notifications

 WWDC 2016 Videos
watchOS 3 Notification Support

New User Notification Framework

Introduction to Notifications

WatchKit

WWDC 2016 Videos
watchOS 3 Notification Support

New User Notification Framework

Introduction to Notifications

WatchKit

• Notification handling APIs similar across platforms
watchOS 3 Notification Support

New User Notification Framework

Introduction to Notifications

WatchKit

- Notification handling APIs similar across platforms
- Local notifications can now be scheduled on Apple Watch
watchOS 3 Notification Support

New User Notification Framework

WatchKit

- Notification handling APIs similar across platforms
- Local notifications can now be scheduled on Apple Watch
- Background actions now delivered to extension if scheduled on Apple Watch
UNUserNotificationCenter

Register to be delegate

class ExtensionDelegate: NSObject, WKExtensionDelegate, UNUserNotificationCenterDelegate {
    func applicationDidFinishLaunching() {
        UNUserNotificationCenter.current().delegate = self;
    }
}
class ExtensionDelegate: NSObject, WKExtensionDelegate, UNUserNotificationCenterDelegate {
    func applicationDidFinishLaunching() {
        UNUserNotificationCenter.current().delegate = self;
    }
}

Register to be delegate
UNUserNotificationCenter

Register to be delegate

class ExtensionDelegate: NSObject, WKExtensionDelegate, UNUserNotificationCenterDelegate {
    func applicationDidFinishLaunching() {
        UNUserNotificationCenter.current().delegate = self;
    }
}

UNUserNotificationCenterDelegate

willPresent notification callback
UNUserNotificationCenterDelegate
willPresent notification callback

Called only if application is active

```swift
func userNotificationCenter(center: UNUserNotificationCenter,
    willPresent: UNNotification,
    withCompletionHandler: (UNNotificationPresentationOptions))
```
**UNUserNotificationCenterDelegate**

**willPresent** notification callback

Called only if application is active

```swift
func userNotificationCenter(center: UNUserNotificationCenter, willPresent: UNNotification, withCompletionHandler: (UNNotificationPresentationOptions))
```

Completion handler argument determines presentation
UNUserNotificationCenterDelegate
willPresent notification callback

Called only if application is active

```swift
func userNotificationCenter(center: UNUserNotificationCenter, willPresent: UNNotification, withCompletionHandler: (UNNotificationPresentationOptions)) {
    // Implementation
}
```

Completion handler argument determines presentation

- `.alert` Presents Long Look
- `.none` App handles notification
UNUserNotificationCenterDelegate
didReceive response callback
func userNotificationCenter(center : UNUserNotificationCenter,
    didReceive: UNNotificationResponse,
    withCompletionHandler: (UNNotificationPresentationOptions))
UNUserNotificationCenterDelegate
didReceive response callback

```swift
func userNotificationCenter(center: UNUserNotificationCenter,
    didReceive: UNNotificationResponse,
    withCompletionHandler: (UNNotificationPresentationOptions))
```

Called when a foreground action makes app active
UNUserNotificationCenterDelegate
didReceive response callback

```swift
func userNotificationCenter(center: UNUserNotificationCenter,
    didReceive: UNNotificationResponse,
    completionHandler: (UNNotificationPresentationOptions))
```

Called when a foreground action makes app active
Called for background actions on notifications
UNUserNotificationCenterDelegate
didReceive response callback

```swift
func userNotificationCenter(center: UNUserNotificationCenter,
    didReceive: UNNotificationResponse,
    withCompletionHandler: (UNNotificationPresentationOptions))
```

Called when a foreground action makes app active
Called for background actions on notifications
- If scheduled on Apple Watch
Notification Scheduling and Delivery

Notifications on iPhone forward to Apple Watch when

• iPhone screen off
• Apple Watch on-wrist and unlocked
Notification Scheduling and Delivery
Notification Scheduling and Delivery

watchOS 3 allows scheduling of local notifications on Apple Watch
Notification Scheduling and Delivery

watchOS 3 allows scheduling of local notifications on Apple Watch

- Delivered only on Apple Watch
Notification Scheduling and Delivery

watchOS 3 allows scheduling of local notifications on Apple Watch

- Delivered only on Apple Watch
- Schedule iPhone notification if required
Notification Scheduling and Delivery

watchOS 3 allows scheduling of local notifications on Apple Watch

- Delivered only on Apple Watch
- Schedule iPhone notification if required

Notification identifiers used to avoid duplicate alerts
watchOS 3 allows scheduling of local notifications on Apple Watch

- Delivered only on Apple Watch
- Schedule iPhone notification if required

Notification identifiers used to avoid duplicate alerts

- Apple Watch notifications take precedence over forwarded with same id
SpriteKit and SceneKit
Enhanced UI Interactions
Classes to Display Scenes
Classes to Display Scenes

WKInterfaceSKScene for SpriteKit
Classes to Display Scenes

WKInterfaceSKScene for SpriteKit

WKInterfaceSCNScene for SceneKit
SceneKit Scene - Displays SceneKit content.

SpriteKit Scene - Displays SpriteKit content.
Displaying the Content
Displaying the Content

Prepare the scene with SpriteKit or SceneKit APIs
Displaying the Content

Prepare the scene with SpriteKit or SceneKit APIs

Present a SpriteKit scene with function on WKInterfaceSKScene

```swift
func present(_: SKScene?)
func present(_: SKScene?, transition: SKTransition)
```
Displaying the Content

Prepare the scene with SpriteKit or SceneKit APIs

Present a SpriteKit scene with function on WKInterfaceSKScene

```swift
func present(_ : SKScene?)
func present(_ : SKScene?, transition : SKTransition)
```

Set the SceneKit scene property on WKInterfaceSCNScene

```swift
var scene : SCNScene
```
Your watchOS 3 API Toolkit
Glanceable, Actionable, Responsive

Gesture recognizers
Digital Crown rotation
Improved table navigation
New Notifications Framework
SceneKit and SpriteKit integration
Quick Interaction Techniques

Tom Witkin watchOS Engineer
Quick Interaction Techniques
Quick Interaction Techniques

Actionable complications
Quick Interaction Techniques

Actionable complications
Glanceable notifications
Quick Interaction Techniques

- Actionable complications
- Glanceable notifications
- Simple navigation
Quick Interaction Techniques

- Actionable complications
- Glanceable notifications
- Simple navigation
- Drop wrist
Actionable Complications
Why Complications
Why Complications

Highly glanceable
Why Complications

Highly glanceable
Frequent updates
Why Complications

Highly glanceable
Frequent updates
Keeps app in memory
Why Complications

Highly glanceable
Frequent updates
Keeps app in memory
Accelerates user into your app
More Information

“Adding Complications to the Gallery”
developer.apple.com/library/prerelease/content
Glanceable Notifications
Why Notifications

- In 60 minutes
  - Pick-up Dad

Where
- San Francisco International Airport
Why Notifications

Customizable
Why Notifications

Customizable
Displays valuable information
Why Notifications

Customizable
Displays valuable information
Removes the necessity to launch your app
Why Notifications

Customizable
Displays valuable information
Removes the necessity to launch your app
Now supports:
• SpriteKit
• SceneKit
• Inline video
SpriteKit

SceneKit
Game Technologies for Apple Watch

<table>
<thead>
<tr>
<th>Mission</th>
<th>Friday 3:00PM</th>
</tr>
</thead>
</table>

SpriteKit

SceneKit
Ivy Calhoun has shared 10 new photos.
WKInterfaceSKScene
Ivy Calhoun has shared 10 new
Simple Navigation
Digital Crown
Drop Wrist
| Keeping Your Watch App Up to Date | Mission | Thursday 9:00AM |
Responding to User Action
Instant Feedback
Your order is on its way!
Choose Item
A digital watch interface showing a purchase screen with the following details:

- **Purchase**: 10:09
- **Roasted Chicken**: $12.00
- **Buy** button

The watch is set against a black background.
Processing After Dropping Wrist
Processing After Dropping Wrist

Background NSURLSession
Processing After Dropping Wrist

Background NSURLSession
Background task assertion
Processing After Dropping Wrist

Background URLSession
Background task assertion
2 seconds
Focus on quick interactions
Focus on quick interactions
Use complications and notifications
Focus on quick interactions
Use complications and notifications
Create glanceable UIs with SceneKit / SpriteKit
Focus on quick interactions
Use complications and notifications
Create glanceable UIs with SceneKit / SpriteKit
Simplify navigation
Focus on quick interactions
Use complications and notifications
Create glanceable UIs with SceneKit / SpriteKit
Simplify navigation
Don’t make the user wait
More Information

https://developer.apple.com/wwdc16/211
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in watchOS 3</td>
<td>WWDC 2016 Videos</td>
<td></td>
</tr>
<tr>
<td>Introduction to Notifications</td>
<td>WWDC 2016 Videos</td>
<td></td>
</tr>
<tr>
<td>Designing Great Apple Watch Experiences</td>
<td>Presidio</td>
<td>Wednesday 1:40PM</td>
</tr>
<tr>
<td>Keeping Your Watch App Up to Date</td>
<td>Mission</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Architecting for Performance on watchOS 3</td>
<td>Mission</td>
<td>Thursday 3:00PM</td>
</tr>
<tr>
<td>Labs</td>
<td>Frameworks Lab</td>
<td>Time</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>WatchKit and ClockKit Lab</td>
<td>B</td>
<td>Wednesday 12:00PM</td>
</tr>
<tr>
<td>WatchKit and Background Tasks Lab</td>
<td>C</td>
<td>Thursday 10:30AM</td>
</tr>
<tr>
<td>WatchKit and WatchConnectivity Lab</td>
<td>B</td>
<td>Friday 2:00PM</td>
</tr>
</tbody>
</table>