A Peek at 3D Touch

Enhancing your apps for the next dimension of touch

Session 228

Tyler Fox UIKit Frameworks Engineer
Peter Hajas UIKit Frameworks Engineer
Agenda
Overview of 3D Touch
Agenda

Overview of 3D Touch

Home Screen Quick Actions
Agenda

Overview of 3D Touch
Home Screen Quick Actions
Peek and Pop
Agenda

Overview of 3D Touch
Home Screen Quick Actions
Peek and Pop
UIPreviewInteraction
Overview of 3D Touch
Tour across the system
Are you free for dinner tonight?

Absolutely! How was your vacation?

It was amazing! Check out this breathtaking waterfall I discovered.
Are you free for dinner tonight?

Absolutely! How was your vacation?

It was amazing! Check out this breathtaking waterfall I discovered.
Sydney Bailey

Dinner in the City
Hey Everyone, Want to grab dinner to up in the city this weekend? I heard good things about t...

Katie Abeles

Pick up from airport?
Hi Derek! Hope all is well with you. I am coming home from London on Friday, January 29 and...

Dave Elving

Trip to San Francisco
Derek, Visiting you in San Francisco was so much fun. We definitely need to head back so...

Amy Frost

Bucket List
Hey - I found this amazing list of adventure travel. After reading this, I definitely added a f...

Karl Bohn

The Best Songs of the 80s
Check out this list of the top 200 songs from the 80’s: http://pitchfork.com/features/staff-list...

Lauren Bosak

Louvre
Hey Derek – Even though we don’t have enough time to do it justice, I still think we should stop
Sydney Bailey
Dinner in the City
Hey Everyone, Want to grab dinner to up in the city this weekend? I heard good things about t...

Katie Abeles
Pick up from airport?
Hi Derek! Hope all is well with you. I am coming home from London on Friday, January 29 and...

Dave Elfving
Trip to San Francisco
Derek, Visiting you in San Francisco was so much fun. We definitely need to head back so...

Amy Frost
Bucket List
Hey - I found this amazing list of adventure travel. After reading this, I definitely added a f...

Karl Bohn
The Best Songs of the 80s
Check out this list of the top 200 songs from the 80’s: http://pitchfork.com/features/staff-lis...

Lauren Bosak
Louvre
Hey Derek – Even though we don’t have enough time to do it justice, I still think we should stop...
Supporting 3D Touch
Supporting 3D Touch

Accelerate access to existing features in your app
Supporting 3D Touch

Accelerate access to existing features in your app
Enable new immersive interactions
Supporting 3D Touch

Accelerate access to existing features in your app
Enable new immersive interactions
Provide a consistent experience across iOS
Home Screen Quick Actions
Leap straight into action from the home screen
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Chat from Tamsin Vantress 10 seconds ago
Chat from Lexi Torres 2 minutes ago
Chat from Natalia Maric 5 minutes ago
Chat from Nick Jones 42 minutes ago
Chat from Peter Urso 1 hour ago
Chat from Natalia Maric 4 hours ago
Chat from Tamsin Vantress 14 hours ago
Chat from Natalia Maric 1 day ago
Chat from Peter Urso 3 days ago
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Home Screen Quick Actions

Two types

Static

Dynamic
Home Screen Quick Actions
Static
Home Screen Quick Actions

Static

Defined in your app’s Info.plist
Home Screen Quick Actions

Static

Defined in your app’s Info.plist

Available as soon as your app has been installed
Home Screen Quick Actions

Static

Defined in your app’s Info.plist
Available as soon as your app has been installed

<table>
<thead>
<tr>
<th>UIApplicationShortcutItems</th>
<th>Array (1 item)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 0</td>
<td>Dictionary (3 items)</td>
</tr>
<tr>
<td>UIApplicationShortcutItemType</td>
<td>String</td>
</tr>
<tr>
<td>UIApplicationShortcutItemTypeTitle</td>
<td>String</td>
</tr>
<tr>
<td>UIApplicationShortcutItemIconType</td>
<td>String</td>
</tr>
</tbody>
</table>
Home Screen Quick Actions
Dynamic
Home Screen Quick Actions
Dynamic

Created by your app at runtime
Home Screen Quick Actions

Dynamic

Created by your app at runtime
Available after the first launch of your app
Home Screen Quick Actions

Dynamic

Created by your app at runtime
Available after the first launch of your app
Shown after any static quick actions (space permitting)
Home Screen Quick Actions

Dynamic

Created by your app at runtime
Available after the first launch of your app
Shown after any static quick actions (space permitting)
Can include a system icon, custom icon, or Address Book contact
Dynamic Quick Actions
Using a contact for the icon

```swift
let contactName = "Lexi Torres"
var contactIcon: UIApplicationShortcutIcon? = nil
```
Dynamic Quick Actions
Using a contact for the icon

```swift
let contactName = "Lexi Torres"
var contactIcon: UIApplicationShortcutIcon? = nil

// Make sure to request access to the user's contacts first
if CNContactStore.authorizationStatus(for: .contacts) == .authorized {
```

```swift
}
```
Using a contact for the icon

```swift
let contactName = "Lexi Torres"
var contactIcon: UIApplicationShortcutIcon? = nil

// Make sure to request access to the user's contacts first
if CNContactStore.authorizationStatus(for: .contacts) == .authorized {
    let predicate = CNContact.predicateForContacts(matchingName: contactName)
    let contacts = try? CNContactStore().unifiedContacts(matching: predicate, keysToFetch: [])
}
```
let contactName = "Lexi Torres"

var contactIcon: UIApplicationShortcutIcon? = nil

// Make sure to request access to the user's contacts first
if CNContactStore.authorizationStatus(for: .contacts) == .authorized {
    let predicate = CNContact.predicateForContacts(matchingName: contactName)
    let contacts = try? CNContactStore().unifiedContacts(matching: predicate, keysToFetch: [])
    if let contact = contacts?.first {
        contactIcon = UIApplicationShortcutIcon(contact: contact)
    }
}
Dynamic Quick Actions
Using a contact for the icon

```swift
let contactName = "Lexi Torres"

var contactIcon: UIApplicationShortcutIcon? = nil

// Make sure to request access to the user's contacts first
if CNContactStore.authorizationStatus(for: .contacts) == .authorized {
    let predicate = CNContact.predicateForContacts(matchingName: contactName)
    let contacts = try? CNContactStore().unifiedContacts(matching: predicate, keysToFetch: [])
    if let contact = contacts?.first {
        contactIcon = UIApplicationShortcutIcon(contact: contact)
    }
}

let icon = contactIcon ?? UIApplicationShortcutIcon(type: .message)
```
Dynamic Quick Actions
Creating and registering

```swift
// Create a dynamic quick action using the icon
let type = "com.company.app.sendChatTo"
let subtitle = "Send a chat"
let shortcutItem1 = UIApplicationShortcutItem(type: type, localizedTitle: contactName,
    localizedSubtitle: subtitle, icon: icon)
```
Dynamic Quick Actions
Creating and registering

// Create a dynamic quick action using the icon
let type = "com.company.app.sendChatTo"
let subtitle = "Send a chat"
let shortcutItem1 = UIApplicationShortcutItem(type: type, localizedTitle: contactName,
                                            localizedSubtitle: subtitle, icon: icon)

// Repeat as needed for any additional dynamic quick actions...
Dynamic Quick Actions
Creating and registering

// Create a dynamic quick action using the icon
let type = "com.company.app.sendChatTo"
let subtitle = "Send a chat"
let shortcutItem1 = UIApplicationShortcutItem(type: type, localizedTitle: contactName,
                                           localizedSubtitle: subtitle, icon: icon)

// Repeat as needed for any additional dynamic quick actions...

let shortcutItems = [shortcutItem1, shortcutItem2, shortcutItem3]
Dynamic Quick Actions
Creating and registering

// Create a dynamic quick action using the icon
let type = "com.company.app.sendChatTo"
let subtitle = "Send a chat"
let shortcutItem1 = UIApplicationShortcutItem(type: type, localizedTitle: contactName,
                                          localizedSubtitle: subtitle, icon: icon)

// Repeat as needed for any additional dynamic quick actions...
let shortcutItems = [shortcutItem1, shortcutItem2, shortcutItem3]

// Register the dynamic quick actions to display on the home screen
application.shortcutItems = shortcutItems
Handling Quick Actions

On app activation

```swift
func application(application: UIApplication,
    performActionForShortcutItem shortcutItem: UIApplicationShortcutItem,
    completionHandler: Bool -> Void) {

}
```
Handling Quick Actions
On app activation

```swift
func application(application: UIApplication,
    performActionForShortcutItem shortcutItem: UIApplicationShortcutItem,
    completionHandler: Bool -> Void) {

    let didHandle: Bool = /* handle the quick action using shortcutItem */
    completionHandler(didHandle)
}
```
Handling Quick Actions

On app activation

```swift
func application(application: UIApplication,
    performActionForShortcutItem shortcutItem: UIApplicationShortcutItem,
    completionHandler: Bool -> Void) {
    let didHandle: Bool = /* handle the quick action using shortcutItem */
    completionHandler(didHandle)
}
```
Handling Quick Actions

On app launch

```swift
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
}
```
Handling Quick Actions

On app launch

```swift
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    var performAdditionalHandling = true
    if let shortcutItem = launchOptions?[UIApplicationLaunchOptionsShortcutItemKey] as? UIApplicationShortcutItem {
        /* handle the quick action using shortcutItem */
        performAdditionalHandling = false
    }
    return performAdditionalHandling
}
```
Handling Quick Actions
On app launch

```swift
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    var performAdditionalHandling = true
    if let shortcutItem = launchOptions?[UIApplicationLaunchOptionsShortcutItemKey] as? UIApplicationShortcutItem {
        /* handle the quick action using shortcutItem */
        performAdditionalHandling = false
    }
    return performAdditionalHandling
}
```
func application(application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    var performAdditionalHandling = true
    if let shortcutItem = launchOptions?[UIApplicationLaunchOptionsShortcutItemKey]
        as? UIApplicationShortcutItem {
        /* handle the quick action using shortcutItem */
        performAdditionalHandling = false
    }
    return performAdditionalHandling
}
Home Screen Quick Actions

Best practices
Home Screen Quick Actions

Best practices

Every app should provide quick actions
Home Screen Quick Actions

Best practices

Every app should provide quick actions

Focus on providing quick access to high-value tasks
Home Screen Quick Actions

Best practices

Every app should provide quick actions
Focus on providing quick access to high-value tasks
Make quick actions predictable
Home Screen Quick Actions

Best practices

Every app should provide quick actions
Focus on providing quick access to high-value tasks
Make quick actions predictable
Be prepared to handle dynamic quick actions from a previous version of your app
Home Screen Quick Actions

Best practices

- Every app should provide quick actions
- Focus on providing quick access to high-value tasks
- Make quick actions predictable
- Be prepared to handle dynamic quick actions from a previous version of your app
- Don’t add functionality that is only accessible using quick actions
Peek and Pop
Seamlessly preview and navigate to content

Peter Hajas UIKit Frameworks Engineer
<table>
<thead>
<tr>
<th>Chat from Tamsin Vantress</th>
<th>10 seconds ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chat from Lexi Torres</td>
<td>2 minutes ago</td>
</tr>
<tr>
<td>Chat from Natalia Maric</td>
<td>5 minutes ago</td>
</tr>
<tr>
<td>Chat from Nick Jones</td>
<td>42 minutes ago</td>
</tr>
<tr>
<td>Chat from Peter Urso</td>
<td>1 hour ago</td>
</tr>
<tr>
<td>Chat from Natalia Maric</td>
<td>4 hours ago</td>
</tr>
<tr>
<td>Chat from Tamsin Vantress</td>
<td>14 hours ago</td>
</tr>
<tr>
<td>Chat from Natalia Maric</td>
<td>1 day ago</td>
</tr>
<tr>
<td>Chat from Peter Urso</td>
<td>3 days ago</td>
</tr>
</tbody>
</table>
Chat from Tamsin Vantress
10 seconds ago
Chat from Lexi Torres
2 minutes ago
Chat from Natalia Maric
5 minutes ago
Chat from Nick Jones
42 minutes ago
Chat from Peter Urso
1 hour ago
Chat from Natalia Maric
4 hours ago
Chat from Tamsin Vantress
14 hours ago
Chat from Natalia Maric
1 day ago
Chat from Peter Urso
3 days ago
Chat from Tamsin Vantress
10 seconds ago
Chat from Lexi Torres
2 minutes ago
Chat from Natalia Maric
5 minutes ago
Chat from Nick Jones
42 minutes ago
Chat from Peter Urso
1 hour ago
Chat from Natalia Maric
4 hours ago
Chat from Tamsin Vantress
14 hours ago
Chat from Natalia Maric
1 day ago
Chat from Peter Urso
3 days ago
AppChat

Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Commit
Adding Peek and Pop to Your App

Components of the interaction
Adding Peek and Pop to Your App

Components of the interaction

Registered View Controller
Adding Peek and Pop to Your App

Components of the interaction

Registered View Controller

[Image of chat interface with a focus on a specific chat message from Natalia Maric]
Adding Peek and Pop to Your App

Components of the interaction

Registered View Controller

Previewed View Controller

Source

Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Adding Peek and Pop to Your App

Components of the interaction

Registered View Controller

Previewed View Controller

Source
Adding Peek and Pop to Your App
Conforming to UIViewControllerPreviewingDelegate

class ChatTableViewController : UITableViewController,
UIViewControllerPreviewingDelegate
Adding Peek and Pop to Your App
Conforming to UIViewControllerPreviewingDelegate

class ChatTableViewController: UITableViewController, 
UIViewControllerPreviewingDelegate
Adding Peek and Pop to Your App
Registering for previewing

override func viewDidLoad() {
    super.viewDidLoad()
    registerForPreviewing(with: self, sourceView: tableView)
}
Adding Peek and Pop to Your App

Registering for previewing

```swift
override func viewDidLoad() {
    super.viewDidLoad()

    registerForPreviewing(with: self, sourceView: tableView)
}
```
Adding Peek and Pop to Your App

Registering for previewing

```swift
override func viewDidLoad() {
    super.viewDidLoad()
    registerForPreviewing(with: self, sourceView: tableView)
}
```
Adding Peek and Pop to Your App
Providing a preview view controller

```swift
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController?
{
}
```
Adding Peek and Pop to Your App

Providing a preview view controller

```swift
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController? {
    // Your implementation goes here
}
```
Adding Peek and Pop to Your App
Providing a preview view controller

```swift
func previewingContext(_ previewingContext: UIViewControllerPreviewing, viewControllerForLocation location: CGPoint) -> UIViewController? {
    guard let indexPath = tableView.indexPathForRow(at: location) else { return nil }
}
```
func previewingContext(_ previewingContext: UIViewControllerPreviewing, 
viewControllerForLocation location: CGPoint) -> UIViewController? {
    guard let indexPath = tableView.indexPathForRow(at: location) else { return nil }

    let chatDetailViewController = ...
    chatDetailViewController.chatItem = chatItem(at: indexPath)
}

Providing a preview view controller

Adding Peek and Pop to Your App
func previewingContext(_ previewingContext: UIViewControllerPreviewing, viewControllerForLocation location: CGPoint) -> UIViewController? {
    guard let indexPath = tableView.indexPathForRow(at: location) else { return nil }

    let chatDetailViewController = ...
    chatDetailViewController.chatItem = chatItem(at: indexPath)

    let cellRect = tableView.rectForRow(at: indexPath)
    let sourceRect = previewingContext.sourceView.convert(cellRect, from: tableView)
    previewingContext.sourceRect = sourceRect
}

Providing a preview view controller

Adding Peek and Pop to Your App
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
    viewControllerForLocation location: CGPoint) -> UIViewController? {
    guard let indexPath = tableView.indexPathForRow(at: location) else { return nil }
    let chatDetailViewController = ...
    chatDetailViewController.chatItem = chatItem(at: indexPath)
    let cellRect = tableView.rectForRow(at: indexPath)
    let sourceRect = previewingContext.sourceView.convert(cellRect, from: tableView)
    previewingContext.sourceRect = sourceRect
    return chatDetailViewController
}
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
func previewingContext(_ previewingContext: UIViewControllerPreviewing, commit viewControllerToCommit: UIViewController) {
    show(viewControllerToCommit, sender: self)
}
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
    commit viewControllerToCommit: UIViewController) {

    show(viewControllerToCommit, sender: self)
}

Adding Peek and Pop to Your App
Committing a preview view controller
Adding Peek and Pop to Your App

Committing a preview view controller

```swift
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
commit viewControllerToCommit: UIViewController) {
    show(viewControllerToCommit, sender: self)
}
```
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Chat from Tamsin Vantress
10 seconds ago

Chat from Lexi Torres
2 minutes ago

Chat from Natalia Maric
5 minutes ago

Chat from Nick Jones
42 minutes ago

Chat from Peter Urso
1 hour ago

Chat from Natalia Maric
4 hours ago

Chat from Tamsin Vantress
14 hours ago

Chat from Natalia Maric
1 day ago

Chat from Peter Urso
3 days ago
Peek and Pop
Preview quick actions

Registered View Controller

Previewed View Controller

Source
override func previewActionItems() -> [UIPreviewActionItem] {

    let heart = UIPreviewAction(title: "❤", style: .default) { (action, viewController) in
        // Send a heart
    }

    return [heart]
}
Adding Peek and Pop to Your App

Preview quick actions

```swift
override func previewActionItems() -> [UIPreviewActionItem] {
    let heart = UIPreviewAction(title: "❤", style: .default) {
        // Send a heart
    }

    return [heart]
}
```
let replyActions = [UIPreviewAction(title: "❤", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😄", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "👍", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😯", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😢", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😈", style: .default, handler: replyActionHandler)],

let sendReply = UIPreviewActionGroup(title: "Send Reply…", style: .default, actions: replyActions)
Adding Peek and Pop to Your App

Preview quick action groups

```swift
let replyActions = [
    UIPreviewAction(title: "❤", style: .default, handler: replyActionHandler),
    UIPreviewAction(title: "😄", style: .default, handler: replyActionHandler),
    UIPreviewAction(title: "👍", style: .default, handler: replyActionHandler),
    UIPreviewAction(title: "😯", style: .default, handler: replyActionHandler),
    UIPreviewAction(title: "😢", style: .default, handler: replyActionHandler),
    UIPreviewAction(title: "😈", style: .default, handler: replyActionHandler)
]

let sendReply = UIPreviewActionGroup(title: "Send Reply…", style: .default, actions: replyActions)
```
let replyActions = [UIPreviewAction(title: "❤", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😄", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "👍", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😯", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😢", style: .default, handler: replyActionHandler),
                   UIPreviewAction(title: "😈", style: .default, handler: replyActionHandler)]

let sendReply = UIPreviewActionGroup(title: "Send Reply…",
                                      style: .default,
                                      actions: replyActions)
Adding Peek and Pop to Your App

Other preview quick action styles

```swift
let save = UIPreviewAction(title: "Chat Saved", style: .selected, handler: saveHandler)
let block = UIPreviewAction(title: "Block", style: .destructive, handler: blockHandler)
```
Peek and Pop

Best practices
Peek and Pop

Best practices

Content that can be tapped should support Peek and Pop
Peek and Pop

Best practices

Content that can be tapped should support Peek and Pop
Return a preview view controller consistently
Best practices

Content that can be tapped should support Peek and Pop
Return a preview view controller consistently
Don’t take too long in the previewing delegate
Peek and Pop

Best practices

Content that can be tapped should support Peek and Pop
Return a preview view controller consistently
Don’t take too long in the previewing delegate
Set the previewing context: sourceRect
UIPreviewInteraction

Peek and Pop feel with your user interface
UIPreviewInteraction

- Same Peek and Pop Force Processing
- Automatic Haptic Feedback
- Your User Interface
extension ChatDetailViewController : UIPreviewInteractionDelegate
override func viewDidLoad() {
    super.viewDidLoad()

    replyPreviewInteraction = UIPreviewInteraction(view: view)
    replyPreviewInteraction.delegate = self
}

UIPreviewInteraction
Creating the UIPreviewInteraction
override func viewDidLoad() {
    super.viewDidLoad()
    replyPreviewInteraction = UIPreviewInteraction(view: view)
    replyPreviewInteraction.delegate = self
}

UIPreviewInteraction
Creating the UIPreviewInteraction
UIPreviewInteraction

Starting the interaction

previewInteractionShouldBegin()
UIPreviewInteraction
State transitions

Preview
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

0.80

Preview
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

Preview

0.10
UIPreviewInteraction

State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

Preview

1.00
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
didUpdatePreviewTransition transitionProgress: CGFloat,
ended: Bool) {

  updateForPreview(progress: transitionProgress)

  if ended {
    completePreview()
  }
}
func previewInteraction(_ previewInteraction: UIPreviewInteraction, 
    didUpdatePreviewTransition transitionProgress: CGFloat, 
    ended: Bool) {

    updateForPreview(progress: transitionProgress)

    if ended {
        completePreview()
    }
}
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
    didUpdatePreviewTransition transitionProgress: CGFloat,
    ended: Bool) {
    updateForPreview(progress: transitionProgress)

    if ended {
        completePreview()
    }
}
func previewInteractionDidCancel(_ previewInteraction: UIPreviewInteraction) {
    UIView.animate(withDuration: 0.4) {
        self.updateForPreview(progress: 0)
        self.resetToInitialAppearance()
    }
}
func previewInteractionDidCancel(_ previewInteraction: UIPreviewInteraction) {
    UIView.animate(withDuration: 0.4) {
        self.updateForPreview(progress: 0)
        self.resetToInitialAppearance()
    }
}
UIPreviewInteraction

State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

Preview
UIPreviewInteraction

State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

Preview

previewInteraction(didUpdateCommitTransition:ended:)

Commit
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

0.20
0.40
0.60
0.80

Preview
Commit

0.80
UIPreviewInteraction

State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

← 0.10

Preview

Commit
UIPreviewInteraction

State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

1.00
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

Preview 1.00
Commit 0.80
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

1.00
0.10
UIPreviewInteraction
State transitions

previewInteraction(didUpdatePreviewTransition:ended:)

previewInteraction(didUpdateCommitTransition:ended:)

Preview

Commit
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
    didUpdateCommitTransition transitionProgress: CGFloat,
    ended: Bool) {

    updateForCommit(progress: transitionProgress)

    if ended {
        completeCommit()
    }
}

Commit transition
func previewInteraction(_ previewInteraction: UIPreviewInteraction,
                   didUpdateCommitTransition transitionProgress: CGFloat,
                   ended: Bool) {

    updateForCommit(progress: transitionProgress)

    if ended {
        completeCommit()
    }
}
func previewInteraction(_ previewInteraction: UIPreviewInteraction, didUpdateCommitTransition transitionProgress: CGFloat, ended: Bool) {
    updateForCommit(progress: transitionProgress)

    if ended {
        completeCommit()
    }
}
Low-Level Force API
Low-Level Force API

Normalized access to force data
Low-Level Force API

Normalized access to force data

Properties on UITouch: `force` and `maximumPossibleForce`
Low-Level Force API

Normalized access to force data

Properties on UI Touch: `force` and `maximumPossibleForce`

Available on devices that support 3D Touch or Apple Pencil
Low-Level Force API

Normalized access to force data

Properties on UITouch: force and maximumPossibleForce

Available on devices that support 3D Touch or Apple Pencil

Leveraging Touch Input on iOS
Summary

Home screen quick actions let you jump straight into action
Summary

Home screen quick actions let you jump straight into action
Peek and Pop allow you to quickly preview and navigate to content
Summary

Home screen quick actions let you jump straight into action

Peek and Pop allow you to quickly preview and navigate to content

UIPreviewInteraction opens up new possibilities to make your app more immersive
Summary

Home screen quick actions let you jump straight into action

Peek and Pop allow you to quickly preview and navigate to content

UIPreviewInteraction opens up new possibilities to make your app more immersive

Users expect your apps to support 3D Touch
More Information

https://developer.apple.com/wwdc16/228
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advances in UIKit Animations and Transitions</td>
<td>Pacific Heights</td>
<td>Wednesday 5:00PM</td>
</tr>
<tr>
<td>Leveraging Touch Input on iOS</td>
<td>Pacific Heights</td>
<td>Thursday 10:00AM</td>
</tr>
</tbody>
</table>
Labs

Cocoa Touch and 3D Touch Lab

Frameworks Lab C  Friday 10:30AM