Lesson 2: Touches and Gestures

INTRODUCTION

Touches and gestures add to the interactivity of iOS devices. iOS interprets when and how a device is being manipulated. This information gets passed on to the app. This lesson will take a look at incorporating touches and gestures into the app.

LESSON OBJECTIVES

By the end of this lesson, the student will be able to:

1. Differentiate between gestures, touches and taps.
2. Define gesture recognizer, event and responder chain.
3. Classify the category to which taps, touches, swipes, and pans belong.
4. Identify the different methods used in multi-touch architecture.
5. Match each gesture recognizer with its corresponding gesture.
6. Explain the requirements for developing an app for the App Store.

LEARNING SEQUENCE

Required Reading	Read the following: • Lesson 2: Touches and Gestures
Resources	View the following: • *OS Tutorial: Gesture Recognition – Recognizing Swipes* (13:56) • *Tutorial 1 – Creating Certificates – First step in creating an iPhone and/or iPad app.* (3:23) • Tutorial 2 – Adding Devices, Finding Your UDID, & Creating Test Accounts (2:49) Other resources: • iOS Quick Tip: Detecting Touches • Apple Developer Website > iOS Apps icon > iOS Developer Library Guides > Event Handling Guide for iOS > Gesture Recognizers
Assignments	Complete the following: 1. Practice – Pic a Picture 2. Quiz 2

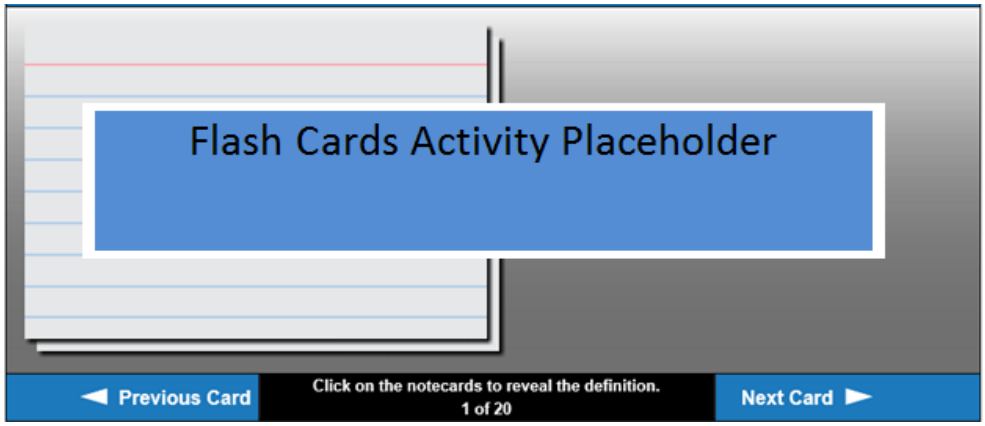
This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)

KEY TERMS

As you read your lesson, pay close attention to the key terms and phrases listed throughout the lesson. These terms and concepts are important to your understanding of the information provided in the lesson.

Karina Whetstine 2/17/2015 2:58 PM
Comment [1]: Please link to document titled:

L7_Key_Terms



INSTRUCTION

The Basics

A Gesture Recognizer is the object that watches for gestures, and interprets touches to determine whether the gesture is a swipe, pinch, drag, tap, or long press. A **touch** is defined as a finger being placed on the screen. The number of touches equals the number of fingers on the screen. A **tap** is when the user uses a single finger to quickly touch and lift the finger.

Gestures can be either discrete or continuous. A discrete gesture, such as a tap, occurs once. This type of gesture calls a single action method. A continuous gesture, such as pinching, takes place over a period of time and triggers a continuous stream of call to the action method until the gesture ends.

If a gesture is recognized, a Gesture Recognizer sends an action message to a target object (which is typically the view's view controller). The view responds to the gesture. Read Gesture Recognizers at the iOS Develop Library.

1. Go to the Apple Developer Website.
2. Select the iOS Apps icon.
3. Click iOS Developer Library Guides.
4. Scroll down and select Event Handling Guide for iOS.
5. Select Gesture Recognizers from the Table of Contents.

Events

Events happen when a user interacts with the device. **UIKit** makes it easy for an app to detect gestures. **UIButton** and **UISlider** are **UIControl** subclasses that respond to specific gestures—a tap for a button

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)

and a drag for a slider. These controls are configured to send an action message to a target object when the touch occurs. Gesture recognizers allow the developer to implement touch-event handling into the app. A developer can use one of several built-in gesture recognizers or the developer can create a new, custom gesture.

Each event has a type: multi-touch events deal with touches and gestures; a motion event is a shake or a tilt; a remote-control event lets the user control an app's multimedia.

Responder Chain

A touch can happen in an app in many different objects onscreen. The developer has to decide which object is going to respond to an event and how that object receives the event. When an event happens, the first responder is the view that is currently interacting with the user. Any class that has a **UIResponder** and UIControl is a responder. A button is tapped. If the current view, the first responder, does not know how to handle that action, it passes the request up the **responder chain** until a **responder object** is found. An action may make it all the way to the window, as shown in Figure 1. From the Window, it passes to **UIApplication** and eventually, the tap may just be ignored if the developer has not handled the action properly.

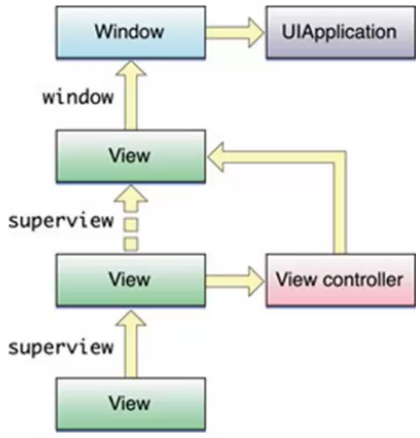


Figure 1: Responder chain

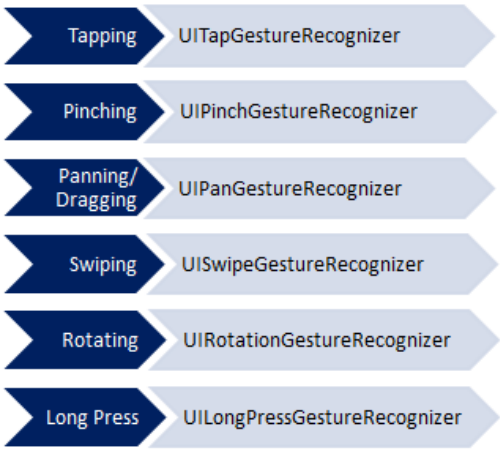
Automatic Gesture Recognition

Automatic gesture recognition was added in iOS4. **UIGestureRecognizer** is pre-built code that is used instead of using manual coding methods. For example to differentiate the number of taps, the following would be used:

```
currenttap requireGestureRecognizerToFail: nexttap
```

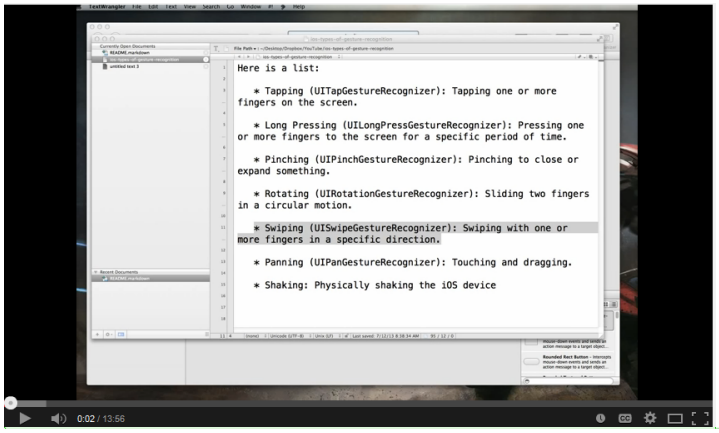
Use the following chart as a guide to the available gestures that correspond to its UIKit class.

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)



Most touch events can be handled with the standard controls and gesture recognizers in UIKit. Otherwise, the developer can create a subclass of UIResponder.

View the *iOS Tutorial: Gesture Recognition – Recognizing Swipes* (13:56).



Karina Whetstine 2/17/2015 2:58 PM
Comment [2]:
https://www.youtube.com/watch?v=5Ou8Oqhngm8

embed code
<iframe width="420" height="315" src="//www.youtube.com/embed/5Ou8Oqhngm8" frameborder="0" allowfullscreen></iframe>

Multi-touch Architecture

When the gesture recognizers are not being used, touch notification methods are used.

- touchesBegan:withEvent: indicates when one or more fingers touch down in a view or window
- touchesMoved:withEvent: indicates when one or more fingers associated with an event move within a view or window
- touchesEnded:withEvent: indicates when one or more fingers are raised from a view or window
- touchesCancelled:withEvent: indicates when a system event cancels a touch event

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)

For more information, read iOS Quick Tip: Detecting Touches.

Developing for the App Store

All apps must be signed with a certificate issued by Apple during development or the app will not build to a device. A testing device has to be associated with a project. In order to do this step, a paid developer membership is required. If part of a development team, each team member must have a certificate and his or her own identity, with a public and private key. Watch the video, *Tutorial 1 – Creating Certificates – First step in creating an iPhone and/or iPad app.* (3:23).

Karina Whetstine 2/17/2015 2:58 PM
Comment [3]:
http://www.youtube.com/watch?v=6_eCFDajDUA

<iframe width="560" height="315" src="//www.youtube.com/embed/6_eCFDajDUA" frameborder="0" allowfullscreen></iframe>

There are three digital assets involved: Keychain on the Mac computer, Xcode installed on the Mac computer, and the Apple Development website. Figure 2 below shows how the digital assets required for iOS development interact so that a device can be associated with a project.

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)

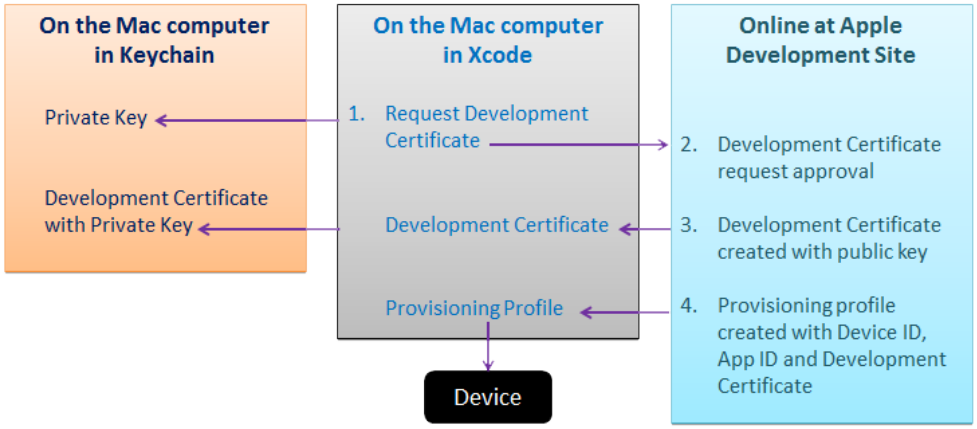


Figure 2: The Keychain on the Mac contains the private key of the development certificate which was received by registering at the Apple Developer website.

The provisioning profile is required on the device to identify the developer and the device itself.

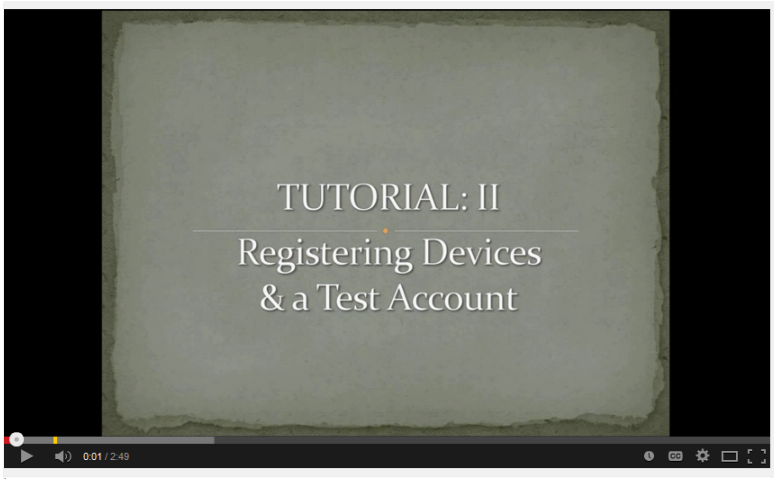
Follow the steps below to set up a device for a project:

1. Choose Window > Organizer to open the Organizer window.
2. Click Devices to display the devices organizer.
3. Plug in the device, and select it in the devices list.
4. Click Use for Development.
5. Copy the device identifier from the identifier text field.
6. If part of a team, send a message containing the device identifier to the team agent requesting that the device be added to the team's list of devices.
7. The device must be added to the team's devices list before continuing.
8. In the devices organizer, select Provisioning Profiles in the Library section, and click Refresh.
9. If the developer does not have a developer certificate, Xcode offers to request one:
    a. Have Xcode request the developer certificate.
    b. If part of a team, notify the team agent that Xcode requested a developer certificate.
10. A developer certificate must be issued before continuing.
11. Ensure that the device is plugged into the Mac, and that it is listed in the devices organizer.
12. Select Provisioning Profiles in the Library section, and click Refresh.
13. Xcode installs the developer certificate in a keychain (if the certificate is not there already), and installs the team provisioning profile on the device.

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)

As part of the certificate process, a private key is stored in keychain on the developer's computer. If this key is lost, it cannot be recovered. If a developer changes computers, he or she must also copy the private key to the new computer.

Watch the video, *Tutorial 2 – Adding Devices, Finding Your UDID, & Creating Test Accounts* (2:49).



Karina Whetstine 2/12/2015 7:20 AM
Comment [4]: CP: Please embed.

http://www.youtube.com/watch?v=MUAmjWCEAIg

embed code

<iframe width="560" height="315" src="//www.youtube.com/embed/MUAmjWCEAIg" frameborder="0" allowfullscreen></iframe>

It is important to also test an app on an actual device. The app feels slower in real-time and the controls feel smaller.

SUMMARY

This lesson discussed the addition of touch and gestures by using gesture recognizers. If a developer wants to distribute the app, he or she needs a paid membership. A paid membership allows the developer to associate a testing device with the app so that the app can be tested on an actual device. The lesson goes through the steps required to set up a device for a project.

ASSIGNMENTS

1. Practice Example—Touches
2. Quiz 2

This work by the National Information Security and Geospatial Technologies Consortium (NISGTC), and except where otherwise noted, is licensed under the Creative Commons Attribution 3.0 Unported License.
Authoring Organization: Collin College
Written by: Original Author, Elizabeth Pannell; Edited Version, Susan Sands
Copyright: © National Information Security, Geospatial Technologies Consortium (NISGTC)