CS193P - Lecture 15

iPhone Application Development

iPhone Device APIs
Location, Accelerometer & Camera

Battery Life & Power Management
Announcements

• Paparazzi 4 due *Friday* night at 11:59PM
  ▪ Late days: use ’em if you’ve got ’em
• Work on final projects!
Today’s Topics

• Hardware features
  ▪ Image Picker & Camera
  ▪ Location
  ▪ Accelerometer

• Battery Life & Power Management
Lots of Cool Features
Device Hardware

Camera
Device Hardware

Camera
Device Hardware
Core location
Device Hardware

Accelerometers
Limited Simulator Support
Image Picker
The Image Picker Interface

The camera
The Image Picker Interface

Saved photos
The Image Picker Interface

The photo library
The Image Picker Interface

Displaying the interface

• **UIImagePickerController** class
  ▪ Use as-is (no subclassing)
  ▪ Handles all user and device interactions
  ▪ UIViewController Subclass

• **UIImagePickerControllerDelegate** protocol
  ▪ Implemented by your delegate object
Displaying the Image Picker

Steps for using

• Check the source availability
• Assign a delegate object
• Present the controller modally
Displaying the Image Picker
Called from a view controller object

```swift
if ([UIImagePickerController isSourceTypeAvailable:
    UIImagePickerControllerSourceTypeCamera]) {
    UIImagePickerController* picker =
        [[UIImagePickerController alloc] init];
picker.sourceType = UIImagePickerControllerSourceTypeCamera;
picker.delegate = self;

    [self presentModalViewController:picker animated:YES];
}
```
Displaying the Image Picker
Called from a view controller object

```objective-c
if ([UIImagePickerController isSourceTypeAvailable:
    UIImagePickerControllerSourceTypeCamera]) {
    UIImagePickerController* picker =
        [[[UIImagePickerController alloc] init] autorelease];
    picker.sourceType = UIImagePickerControllerSourceTypeCamera;
    picker.delegate = self;

    [self presentModalViewController:picker animated:YES];
}
```
Displaying the Image Picker
Called from a view controller object

```swift
if ([UIImagePickerController isSourceTypeAvailable: 
    UIImagePickerControllerSourceTypeCamera])
{
    UIImagePickerController* picker = 
        [[UIImagePickerController alloc] init];
picker.sourceType = UIImagePickerControllerSourceTypeCamera;
picker.delegate = self;

    [self presentModalViewController:picker animated:YES];
}
```
Displaying the Image Picker
Called from a view controller object

if ([UIImagePickerController isSourceTypeAvailable:
    UIImagePickerControllerSourceTypeCamera])
{
    UIImagePickerController* picker =
        [[UIImagePickerController alloc] init];
    picker.sourceType = UIImagePickerControllerSourceTypeCamera;
    picker.delegate = self;

    [self presentModalViewController:picker animated:YES];
}
if ([UIImagePickerController isSourceTypeAvailable: UIImagePickerControllerSourceTypeCamera]) {
    UIImagePickerController* picker = [[UIImagePickerController alloc] init];
    picker.sourceType = UIImagePickerControllerSourceTypeCamera;
    picker.delegate = self;

    [self presentModalViewController:picker animated:YES];
}

Selecting an Image
Selecting an Image
Defining Your Delegate Object

The UIImagePickerControllerDelegate protocol

• Two methods:

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
   editingInfo:(NSDictionary*)editingInfo;

- (void)imagePickerControllerDidCancel:
   (UIImagePickerController*)picker;
Defining Your Delegate Object
The accept case

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
   editingInfo:(NSDictionary*)editingInfo
{
    // Save or use the image here.

    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Defining Your Delegate Object

The accept case

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
    editingInfo:(NSDictionary*)editingInfo
{
    // Save or use the image here.

    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Defining Your Delegate Object

The accept case

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
editingInfo:(NSDictionary*)editingInfo
{
    // Save or use the image here.

    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Defining Your Delegate Object

The accept case

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
editingInfo:(NSDictionary*)editingInfo
{
    // Save or use the image here.

    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Defining Your Delegate Object

The cancel case

- (void)imagePickerControllerDidCancel:
  (UIImagePickerController*)picker
{
  // Dismiss the image picker.
  [self dismissModalViewControllerAnimated:YES];
  [picker release];
}
Defining Your Delegate Object

The cancel case

- (void)imagePickerControllerDidCancel:
    (UIImagePickerController*)picker
{
    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Defining Your Delegate Object

The cancel case

- (void)imagePickerControllerDidCancel: (UIImagePickerController*)picker
{
    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}"
Manipulating the Returned Image
Allowing users to edit returned images

- If `allowsImageEditing` property is YES:
  - User allowed to crop the returned image
  - Image metadata returned in `editingInfo`
Manipulating the Returned Image

Allowing users to edit returned images
Manipulating the Returned Image

Allowing users to edit returned images
Manipulating the Returned Image

The editingInfo dictionary

- (void)imagePickerController:(UIImagePickerController*)picker
didFinishPickingImage:(UIImage*)image
editingInfo:(NSDictionary*)editingInfo
{
    // Save or use the image here.

    // Dismiss the image picker.
    [self dismissModalViewControllerAnimated:YES];
    [picker release];
}
Manipulating the Returned Image

The editingInfo dictionary

- Original image in `UIImagePickerControllerOriginalImage` key
- Crop rectangle in `UIImagePickerControllerCropRect` key
Augmented Reality

Walk around looking through a camera. What could possibly go wrong?

@property BOOL showsCameraControls;
@property(retain) UIView cameraOverlayView;
@property CGAffineTransform cameraViewTransform;
The Image Picker Interface

Custom Camera Interface
Managing Image Data

Avoid retaining images
Saving Images
Writing to the saved photos album

- `UIImageWriteToSavedPhotosAlbum` function
  - Photos can be downloaded to iPhoto by user
  - Optional completion callback
Saving Videos
Writing to the saved photos album

- **(UIVideoAtPathIsCompatibleWithSavedPhotosAlbum** function
- **UISaveVideoAtPathToSavedPhotosAlbum** function
  - Videos can be downloaded to iPhoto by user
  - Optional completion callback
Available in the Simulator
Key Tips

Using UIImagePickerController effectively

• Always check the source availability
• Your delegate methods do the cleanup
• Be frugal with images
• Available in the simulator
Core Location
Core Location

What is it?
Core Location

What is it?
Core Location

What is it?

Location ring

Activate service
Core Location

How?
Core Location

How?
Core Location

How?
Core Location

How?
Core Location

How?
Core Location
Location Technologies
Core Location

Location Technologies

Bootstrap
Core Location
Location Technologies

Cross-check
Core Location
Location Technologies

Complement
Core Location Framework
Core Location Framework

The core classes and protocols

- **Classes**
  - CLLocationManager
  - CLLocation
- **Protocol**
  - CLLocationManagerDelegate
Core Location Framework

CLLocationManagerDelegate protocol

• Two optional methods

- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
fromLocation:(CLLocation*)oldLocation;

- (void)locationManager:(CLLocationManager*)manager
didFailWithError:(NSError*)error;

• Called asynchronously on main thread
• Issues movement-based updates
Core Location Framework

CLLocationManagerDelegate protocol

• Two optional methods

```c
- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
fromLocation:(CLLocation*)oldLocation;
```

```c
- (void)locationManager:(CLLocationManager*)manager
didFailWithError:(NSError*)error;
```

• Called asynchronously on main thread
• Issues movement-based updates
Core Location Framework

CLLocationManagerDelegate protocol

• Two optional methods

- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
fromLocation:(CLLocation*)oldLocation;

- (void)locationManager:(CLLocationManager*)manager
didFailWithError:(NSError*)error;

• Called asynchronously on main thread
• Issues movement-based updates
Getting a Location

Starting the location service

```objective-c
CLLocationManager* locManager =
    [[CLLocationManager alloc] init];

locManager.delegate = self;
[locManager startUpdatingLocation];
```
Getting a Location
Starting the location service

CLLocationManager* locManager =
    [[CLLocationManager alloc] init];

locManager.delegate = self;
[locManager startUpdatingLocation];
Getting a Location

Starting the location service

```objective-c
CLLocationManager* locManager = 
    [[CLLocationManager alloc] init];
locManager.delegate = self;
[locManager startUpdatingLocation];
```
Getting a Location
Using the event data

- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
    fromLocation:(CLLocation*)oldLocation
{

    // Use the coordinate data.
    double lat = newLocation.coordinate.latitude;
    double lon = newLocation.coordinate.longitude;
}

Friday, February 26, 2010
- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
fromLocation:(CLLocation*)oldLocation
{
    NSTimeInterval howRecent =
        [newLocation.timestamp timeIntervalSinceNow];
    if (howRecent < -10) return;

    // Use the coordinate data.
    double lat = newLocation.coordinate.latitude;
    double lon = newLocation.coordinate.longitude;
}
Getting a Location

Using the event data

- (void)locationManager:(CLLocationManager*)manager
didUpdateToLocation:(CLLocation*)newLocation
  fromLocation:(CLLocation*)oldLocation
{
    NSTimeInterval howRecent =
        [newLocation.timestamp timeIntervalSinceNow];
    if (howRecent < -10) return;

    if (newLocation.horizontalAccuracy > 100) return;

    // Use the coordinate data.
    double lat = newLocation.coordinate.latitude;
    double lon = newLocation.coordinate.longitude;
}
Getting a Heading

Using the event data

```objc
- (void)locationManager:(CLLocationManager *)manager
didUpdateHeading:(CLHeading *)newHeading
{
    // Use the coordinate data.
    CLLocationDirection heading = newHeading.trueHeading;
}
```
Power Play (beat Canada again):
CLLocationManager Properties
Desired Accuracy

Choosing an appropriate accuracy level

```swift
CLLocationManager* locManager =
    [[CLLocationManager alloc] init];
locManager.desiredAccuracy = kCLLocationAccuracyBest;
```

- Choose an appropriate accuracy level
  - Higher accuracy impacts power consumption
  - Lower accuracy is “good enough” in most cases
- Can change accuracy setting later if needed
- Actual accuracy reported in `CLLocation` object
Distance Filter

Choosing an appropriate update threshold

CLLocationManager* locManager =
    [[CLLocationManager alloc] init];

locManager.distanceFilter = 3000;

• New events delivered when threshold exceeded
Stopping the Service

CLLocationManager* locManager =
    [[CLLocationManager alloc] init];
[locManager startUpdatingLocation];
...

[locManager stopUpdatingLocation];

• Restart the service later as needed
Responding to Errors

User may deny use of the location service

• Results in a `kCLErrorDenied` error
• Protects user privacy
• Occurs on a per-application basis
Responding to Errors
Location may be unavailable

• Results in a `kCLErrorLocationUnknown` error
• Likely just temporary
• Scan continues in background
Limited Simulator Support
Accelerometers
What Are Accelerometers?

Measure changes in force
What Are Accelerometers?
Measure changes in force
What Are Accelerometers?

Measure changes in force
What Are Accelerometers?

Measure changes in force
What Are Accelerometers?

Measure changes in force
What Are Accelerometers?

Measure changes in force
Accelerometers

What are the uses?
Accelerometers

What are the uses?
Kinds of Orientation
The physical vs the interface

• Physical Orientation
  ▪ How is the device positioned?
• Interface Orientation
  ▪ Where is the status bar?
• Examples: Photos & Safari
Orientation-Related Changes

Getting the physical orientation

- **UIDevice** class
  - Start notifications
    - `beginGeneratingDeviceOrientationNotifications`
  - Get Orientation
    - `UIDeviceOrientationDidChangeListener` Notification delivered to registered observers
    - `orientation` property
  - Stop notifications
    - `endGeneratingDeviceOrientationNotifications`
Orientation-Related Changes

Getting the interface orientation

• **UIApplication** class
  - `statusBarOrientation` property
  - Defines interface orientation, not device orientation

• **UIViewController** class
  - `interfaceOrientation` property

- `(BOOL)shouldAutorotateToInterfaceOrientation:`
  `(UIInterfaceOrientation)`interfaceOrientation
Shake

Undo!

• **UIEvent** type
  - `@property(readonly) UIEventType type;`
  - `@property(readonly) UIEventSubtype subtype;`

- `UIEventTypeMotion`
- `UIEventSubtypeMotionShake`
Orientation changes are nice, but…
Wii™ Want Raw Data

0.5g

0.75g

1.0g

0.5g
The Accelerometer Interface
Getting the raw accelerometer data

• Part of the UIKit framework
• Delivers 3-axis data
• Configurable update frequency (approx 10–100Hz)
• Delegate-based event delivery
Device Axis Orientation

- X
- Y
+ Z
+ X
- Z
+ Y
The Accelerometer Interface

Getting the raw accelerometer data

- Classes
  - UIAccelerometer
  - UIAcceleration
- Protocol
  - UIAccelerometerDelegate
Configuring the Accelerometer

Starting the event delivery

- (void)enableAccelerometerEvents
{
    UIAccelerometer* theAccel =
        [UIAccelerometer sharedAccelerometer];
    theAccel.updateInterval = 1/50; // 50 Hz
    theAccel.delegate = self;
}
Configuring the Accelerometer

Starting the event delivery

- (void)enableAccelerometerEvents
{
    UIAccelerometer* theAccel =
        [UIAccelerometer sharedAccelerometer];
    theAccel.updateInterval = 1/50; // 50 Hz
    theAccel.delegate = self;
}
Configuring the Accelerometer

Starting the event delivery

- (void)enableAccelerometerEvents
{
    UIAccelerometer* theAccel = [UIAccelerometer sharedAccelerometer];
    theAccel.updateInterval = 1/50; // 50 Hz
    theAccel.delegate = self;
}
- (void)enableAccelerometerEvents
{
    UIAccelerometer* theAccel = [UIAccelerometer sharedAccelerometer];
    theAccel.updateInterval = 1/50;  // 50 Hz
    theAccel.delegate = self;
}
Event delivery begins as soon as you assign the delegate
Defining Your Delegate Object

Processing the accelerometer data

- (void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration
{
    // Get the event data
    UIAccelerationValue x, y, z;

    x = acceleration.x;
    y = acceleration.y;
    z = acceleration.z;

    // Process the data...
}
Defining Your Delegate Object

Processing the accelerometer data

- (void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration
{
    // Get the event data
    UIAccelerationValue  x, y, z;

    x = acceleration.x;
    y = acceleration.y;
    z = acceleration.z;

    // Process the data...
}
Defining Your Delegate Object

Processing the accelerometer data

- (void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration
{
    // Get the event data
    UIAccelerationValue x, y, z;

    x = acceleration.x;
    y = acceleration.y;
    z = acceleration.z;

    // Process the data...
}

• Only one delegate per application
• Delivered asynchronously to main thread
Configuring the Accelerometer

Choosing an appropriate update frequency

• System range is approximately 10–100Hz
• Frequency should be based on need
  ▪ Determine the minimum frequency for your needs
  ▪ Don’t update too frequently
• Target ranges
  ▪ Game input: 30–60 Hz
  ▪ Orientation detection: 10–20 Hz
Disabling Event Delivery
Stopping the event delivery

- (void)disableAccelerometerEvents
{
    UIAccelerometer* theAccel =
        [UIAccelerometer sharedAccelerometer];

    theAccel.delegate = nil;
}

FilteringAccelerometerData

Usefilters to isolate data components

• Low-pass filter
  ▪ Isolates constant acceleration
  ▪ Used to find the device orientation
• High-pass filter
  ▪ Shows instantaneous movement only
  ▪ Used to identify user-initiated movement
Filtering Accelerometer Data

Examining the accelerometer data
Filtering Accelerometer Data

Examining the accelerometer data
Filtering Accelerometer Data

Examining the accelerometer data

-1.0g
Filtering Accelerometer Data

Examining the accelerometer data

\[ f(t) \]

-1.0g
Filtering Accelerometer Data

But, to apply a filter…
Filtering Accelerometer Data

But, to apply a filter...

\[ f(t) \Rightarrow F(\omega) \]

Fourier Transform
Filtering Accelerometer Data

Changing to the frequency domain

\[ f(t) \]
Filtering Accelerometer Data

Changing to the frequency domain
Filtering Accelerometer Data
But if we shake the device...
Filtering Accelerometer Data

But if we shake the device…
Filtering Accelerometer Data

We see something more interesting…

\[ f(t) \]
Filtering Accelerometer Data

We see something more interesting…

\[ f(t) \]
Filtering Accelerometer Data

We see something more interesting…

\[ f(t) \]

\[ F(\omega) \]
Filtering Accelerometer Data

We see something more interesting…
Filtering Accelerometer Data

Applying a low-pass filter

\[ f(t) \]

\[ F(\omega) \]
Filtering Accelerometer Data

Applying a low-pass filter

\[ f(t) \]

\[ F(\omega) \]
Filtering Accelerometer Data

Applying a low-pass filter

\[ f(t) \]

\[ F(\omega) \]
Filtering Accelerometer Data

Applying a low-pass filter

- Simple low-pass filter example

```c
#define FILTERFACTOR 0.1

value = (newAcceleration * FILTERFACTOR) +
       (previousValue * (1.0 - FILTERFACTOR));

previousValue = value;
```
Filtering Accelerometer Data

Applying a high-pass filter

\[ f(t) \quad F(\omega) \]
Filtering Accelerometer Data

Applying a high-pass filter

\( f(t) \)

\( F(\omega) \)
Filtering Accelerometer Data

Applying a high-pass filter

- Simple high-pass filter example

```c
#define FILTERFACTOR  0.1

lowPassValue = (newAcceleration * FILTERFACTOR) +
               (previousValue * (1.0 - FILTERFACTOR));

previousLowPassValue = lowPassValue;

highPassValue = newAcceleration - lowPassValue;
```
Filtering Accelerometer Data

Bubble Level sample (low-pass filter)
Demo
Filtering Accelerometer Data

Bubble Level sample (low-pass filter)

```c
- (void)accelerometer:(UIAccelerometer*)accelerometer
didAccelerate:(UIAcceleration*)acceleration
{
    accelerationX = acceleration.x * kFilteringFactor +
                    accelerationX * (1.0 - kFilteringFactor);
    accelerationY = acceleration.y * kFilteringFactor +
                    accelerationY * (1.0 - kFilteringFactor);

    currentRawReading = atan2(accelerationY,accelerationX);
    float calibratedAngle = [self calibratedAngleFromAngle:
                            currentRawReading];

    [levelView updateToInclinationInRadians:calibratedAngle];
}
```
Demo
No Simulator Support
Key Tips

Using the Accelerometers Effectively

• Use UIViewController
• Use filters to isolate raw data components
• Disable accelerometer updates when not needed
  ▪ Set your accelerometer delegate to nil
Summary

• Take advantage of the device APIs, but…
• For image picker, always check source availability
• For hardware-based features, turn them off when not needed
Battery Life &
Power Management
Power Management
Power Management

Small devices need advanced power management
Power Management

Small devices need advanced power management

• Total power consumption
Power Management

Small devices need advanced power management

- Total power consumption
  - Laptops: ~20-60W
Power Management

Small devices need advanced power management

• Total power consumption
  ▪ Laptops: ~20-60W
  ▪ iPhone: 500 mW to 2.5W
Power Management

Small devices need advanced power management

• Total power consumption
  ▪ Laptops: ~20-60W
  ▪ iPhone: 500 mW to 2.5W

• Dynamic clocking
Power Management

Small devices need advanced power management

- Total power consumption
  - Laptops: ~20-60W
  - iPhone: 500 mW to 2.5W
- Dynamic clocking
- Clock gating and power gating
Power Management

Small devices need advanced power management

• Total power consumption
  ▪ Laptops: ~20-60W
  ▪ iPhone: 500 mW to 2.5W

• Dynamic clocking

• Clock gating and power gating
  ▪ Turning blocks on and off continuously
Power Consumption
Power Consumption

Everything consumes power
Power Consumption

Everything consumes power

- Radios – up to ~2W
Power Consumption

Everything consumes power

• Radios – up to ~2W
  ▪ Baseband, Wi-Fi, Bluetooth, GPS
Power Consumption

Everything consumes power

• Radios – up to ~2W
  ▪ Baseband, Wi-Fi, Bluetooth, GPS

• CPU/GPU – up to ~800 mW
Power Consumption

Everything consumes power

- Radios – up to ~2W
  - Baseband, Wi-Fi, Bluetooth, GPS
- CPU/GPU – up to ~800 mW
- Display – up to ~200 mW
Power Consumption

Everything consumes power

- Radios – up to ~2W
  - Baseband, Wi-Fi, Bluetooth, GPS
- CPU/GPU – up to ~800 mW
- Display – up to ~200 mW
- Hardware modules – ~10s of mWs
Power Consumption

Everything consumes power

- Radios – up to ~2W
  - Baseband, Wi-Fi, Bluetooth, GPS
- CPU/GPU – up to ~800 mW
- Display – up to ~200 mW
- Hardware modules – ~10s of mWs
- Keeping the system awake – enormous impact
Battery Life

Be aware of power consumption
Battery Life

Be aware of power consumption
Power Consumption - Radios

The network

- Transmitting is the most expensive operation
- Minimize the amount of transmitted data
- Avoid chatty protocols
- Transmit/receive in bursts
- Use compact data formats
- Core Location
  - Stop the location service once you have a location fix
  - Request only the location accuracy that you need
Power Consumption - CPU/GPU

All about performance

- Reduce CPU usage
- Use Sample or Shark
- Stress the GPU less – fewer layers, smaller textures, etc.
Power Consumption - Hardware Modules

Accelerometer, NAND, others

• Turn off what you don’t need
• Accelerometer
  ▪ Set the UIAccelerometer delegate to nil
  ▪ Support orientation changes only as needed
• NAND
  ▪ Access the disk less – use the System Usage instrument
Power Consumption - Standby

Let the system sleep

• Battery life drops from 250+ hours to <12 hours without sleep
• Don’t disable the idle timer
• Don’t play audio except when you need to
Questions?