iPhone Programming
CMSC 498I – Fall 2010

Accelerometer • OpenGL
Adam Porter
This Week’s Topics

• Gaming Related Functionality
  ▪ Accelerometer
  ▪ OpenGL Basics
  ▪ Audio & Video
  ▪ iPod Library Access
  ▪ GameKit
Accelerometer
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
Accelerometer Raw Data

0.75g

0.50g

0.75g
Accelerometer

- Single shared instance of `UIAccelerometer`
- Sends `UIAcceleration` values
  - Values sent to the `UIAccelerometerDelegate`
  - G-force values reported by the hardware
  - Timestamp
The Accelerometer Interface

• Getting the raw accelerometer data
  ▪ Classes
    ▪ UIAccelerometer
    ▪ UIAcceleration
  ▪ Protocol
    ▪ UIAccelerometerDelegate
Starting The Accelerometer

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

• Get the system accelerometer
• Set update interval
• Assign delegate
  ▪ Event delivery begins as soon as you assign the delegate
    ▪ Only one delegate per application
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
Defining The Delegate Object

- Processing accelerometer events

```swift
- (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
- Acceleration events delivered asynchronously to main thread
Stopping Event Delivery

- (void)disableAccelerometerEvents
{
    UIAccelerometer* theAccel = [UIAccelerometer sharedAccelerometer];
    theAccel.delegate = nil;
}

• Be a good battery citizen...
  • “turn off” updates when they aren’t needed
Using Accelerometer Data

• Accelerometer data tends to bounce around

• Use filters to isolate signals in the data

• Examples

• Low-pass filter
  - Identify constant effects of gravity by filtering out influence of instantaneous motion
  - Used to find the device orientation

• High-pass filter
  - Identify instantaneous motion by filtering out constant effects of gravity
  - Used to identify user-initiated movement
Filtering Accelerometer Data

• Consider a stationary device lying face up on a table
  ▪ Only force exerted is the force of gravity
    ▪ $x = 0.0g$, $y = 0.0g$, $z = -1.0g$

• Raw data is given in the time domain
  ▪ Data values over time

• Often more convenient to work in the frequency domain
  ▪ Frequency of data values across ranges of data values

• Traditionally, you use a Fourier transform to convert samples in the time domain to the frequency domain

$$ f(t) \implies F(\omega) $$

Fourier Transform
Filtering Accelerometer Data

- Examining the accelerometer data

\[ f(t) \]

-1.0g

\[ Z \text{ Axis} \]
Filtering Accelerometer Data

- Changing to the frequency domain

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

- But if we shake the device...

\[ f(t) \]
Filtering Accelerometer Data

- But if we shake the device...

\[ f(t) \]
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...

- No flat line at -1.0g
- Accelerometers generating waveform pattern
- But waveform centered around the -1.0g
  - gravity hasn’t gone away

\[ f(t) \]
Filtering Accelerometer Data

- We see something more interesting…
Filtering Accelerometer Data

- We see something more interesting…

\[ f(t) \]

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

- Applying a low-pass filter
  - Goal: Isolate the low-frequency and steady-state data
  - Here, steady state-state data is dominated by gravity
Filtering Accelerometer Data

- Applying a low-pass filter
  - Goal: Isolate the low-frequency and steady-state data
  - Here, steady state-state data is dominated by gravity
Filtering Accelerometer Data

- Applying a low-pass filter
  - Goal: Isolate the low-frequency and steady-state data
  - Here, steady state–state data is dominated by gravity

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

• Applying a low-pass filter (simple example)

```c
#define FILTERFACTOR 0.1

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

previousValue = value;
```

• Understanding the low-pass filter

  ▪ Rapidly changing values (indicating high frequency energy) receive less
    weight, so tend to cancel out over time

  ▪ Result is an approximation of the effects of gravity only
Filtering Accelerometer Data

- Applying a high-pass filter
  - Goal: Isolate higher-frequency
  - Filter out gravity to see just rapid changes
Filtering Accelerometer Data

- Applying a high-pass filter
  - Goal: Isolate higher-frequency
  - Filter out gravity to see just rapid changes
Filtering Accelerometer Data

- Applying a high-pass filter (simple example)

```c
#define FILTERFACTOR  0.1

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

previousValue = value;
```

- Understanding the high-pass filter
  - Approximate the effects of motion by giving more weight to new samples
  - Takes the new sample, and subtracts out the low-pass value
  - Result is an approximation of the effects ignoring gravity
Filtering Accelerometer Data

- These approximations do work in real life
- Apple’s Bubble Level sample code uses a simple low-pass filter
OpenGL Basics
OpenGL

- We will cover
  - Getting an iPhone OpenGL app off the ground
  - OpenGL basics concepts

- Learning OpenGL in detail – not a goal of this lecture

- Lots of tutorials and resources for learning all about OpenGL
OpenGL ES

- Open Graphics Library
  - API for 2D/3D graphics display
  - Cross platform, efficient, popular rendering APIs
  - No physical interaction, just display

- OpenGL ES is a subset targeted for mobile devices
- Xcode includes an iPhone OpenGL ES Application Template
  - Takes care of standard setup and configuration
OpenGL ES 1.1 –vs- 2.0

- OpenGL ES 2.0 requires developers to be more familiar with computer graphics
  - Must implement shaders in order to render anything on screen

- Device Compatibility
  - 2.0 supported on iPhone from 3GS and on iPod Touches from 3rd gen.
  - 1.1 supported on all platforms

- Xcode’s OpenGL ES Template has a lot of boiler plate code to hide version selection details
What Is A Shader?

- Shaders are mini programs that run on the graphics hardware
- Transform input data (vertices, states) into images on screen
- Written in the OpenGL shared language - GLSL
- Vertex Shaders – Foo.vsh – Executed for each vertex, computes position
- Fragment Shaders – Foo.fsh – Executed once for each fragment (~pixel), computes color
- Mini GL programs...
  - Load source, compile, attach to a program, and link
  - See Apple’s sample code GLES2Sample
GLES2Sample Sample Application

- Main window consists of one EAGLView
  - UIView subclass, which wraps CAEAGLLayer
  - Manages GL scene animation & creates ESRenderer for your drawing code

- ESRenderer
  - ES1Renderer (v1.1) and ES2Renderer (v 2.0)
  - EAGLView first tries to create ES2Renderer; then falls back to ES1Renderer
  - Drawing code goes in each subclass’ -render method
  - To support only ES 1.1, remove ES2Renderer references

- Sample code at:
Display Updates

• Games are constantly drawing to the screen
  ▪ One option is to set up a NSTimer to fire repeatedly and refresh your UI
  ▪ Another option (iPhoneOS 3.1+) is CADisplayLink

• CADisplayLink
  ▪ Lets you synchronize drawing with refresh rate of the screen
  ▪ Invokes a method you specify in sync with the screen refresh

```objective-c
displayLink = [NSClassFromString (@"CADisplayLink")
  displayLinkWithTarget:self
  selector:@selector(drawView:)];

[displayLink setFrameInterval:animationFrameInterval];

[displayLink addToRunLoop:[NSRunLoop currentRunLoop]
  forMode:NSDefaultRunLoopMode];
```
Anatomy Of An OpenGL ES App

**EAGLView** : UIView

**ES2Renderer** : NSObject

- (id)init;
  // Standard setup for you (frame buffer, etc)
  // Creates a ES2 context if possible
  // Returns *nil* if ES2 not supported

- (void)resizeFromLayer:(CAEAGLLayer *)layer
  // Recreates frame buffers for you

- (void)render {
  // Some boilerplate setup code
  // Replace with your own drawing
}

// Shader Helpers
- (void)loadShaders... // Load your shaders here
- (void)compileShader...
- (void)linkProgram...
- (void)validateProgram....
Anatomy Of An OpenGL ES App

**EAGLView** : **UIView**

**ES1Renderer** : **NSObject**

```objective-c
-(id)init;
  // Standard setup for you (frame buffer, etc)
  // Creates a ES2 context if possible
  // Returns *nil* if ES2 not supported

-(void)resizeFromLayer:(CAEAGLLayer *)layer
  // Recreates frame buffers for you

-(void)render {
  // Some boilerplate setup code
  // Replace with your own drawing
}

// Shader Helpers
-(void)loadShaders... // Load your shaders here
-(void)compileShader...
-(void)linkProgram...
-(void)validateProgram....
```
Understanding The OpenGL World

- OpenGL programming consists of our basic steps:
  - Define your model
  - Define your viewport - orthographic, projection
  - Set up OpenGL state - contexts, frame buffers, etc…
  - Draw geometry
Drawing Shapes - render

- Consult your model
- Everything is drawn as triangles

- Apply, push, pop transformations – scale, rotate, etc…
- Enable client state
- Define vertex arrays
- Define attributes – normal, color, texture coordinates
- Draw array
Drawing Shapes - render

- Consult your model
- Everything is drawn as triangles
- Apply, push, pop transformations – scale, rotate, etc…
- Enable client state
- Define vertex arrays
- Define attributes – normal, color, texture coordinates
- Draw array
Drawing Shapes -render

- Consult your model
- Everything is drawn as triangles
- Apply, push, pop transformations – scale, rotate, etc…
- Enable client state
- Define vertex arrays
- Define attributes – normal, color, texture coordinates
- Draw array
Example Drawing Code

• Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);
glScalef(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };
glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```
Example Drawing Code

• Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);
glScalef(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };
glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
GLfloat squareColors[] = {0, 0, 0, 1};
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```
Example Drawing Code

- Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);
glScalef(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };
glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```
Example Drawing Code

- Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);
glScalef(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };
glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```
Example Drawing Code

- Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);
setScale(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };
glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```
Example Drawing Code

- Replace drawing parts of -render

```c
// Clear the background and set it to gray
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);	glClear(GL_COLOR_BUFFER_BIT);

// Apply transformations
glTranslatef(squareCenter.x, squareCenter.y, 0.0f);	glScalef(squareSizeShown, squareSizeShown, 1.0f);	glRotatef(squareAngle, 0, 0, 1.0f);

// Enable states - without this glVertexPointer, glColorPointer do nothing
glEnableClientState(GL_VERTEX_ARRAY);	glEnableClientState(GL_COLOR_ARRAY);

// Give OpenGL engine an array of vertices to render
static const GLfloat squareVertices[] = { // 2D points on a square ..... };	glVertexPointer(2, GL_FLOAT, 0, squareVertices);

// Define colors for each vertex
glColorPointer(4, GL_UNSIGNED_BYTE, 0, squareColors);

// Draw as a “Triangle Strip”	glDrawArrays(GL_TRIANGLES, 0, 4);
```
// clear background and set it to gray

glClearColor(0.5f, 0.4f, 0.5f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);

// use shader program
    glUseProgram(program);

// apply transformations
...

// projection matrix * modelview matrix
mat4f_MultiplyMat4f(proj, modelview, modelviewProj);

// update uniform values
glUniformMatrix4fv(uniforms[UNIFORM_MODELVIEW_PROJECTION_MATRIX],
    1, GL_FALSE, modelviewProj);

// update attribute values
glVertexAttribPointer(ATTRIB_VERTEX, 2, GL_FLOAT, 0, 0, squareVertices);
glEnableVertexAttribArray(ATTRIB_VERTEX);

// draw

glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
**GIDrawArrays**

- **GL_TRIANGLE_STRIP**

```c
static const GLfloat squareVertices[] = {
    -0.5, -0.5, // point 0
    0.5, -0.5, // point 1
    -0.5, 0.5, // point 2
    0.5, 0.5, // point 3
};
```
GlDrawArrays

- GL_TRIANGLES_STRIP

```c
static const GLfloat squareVertices[] = {
    -0.5, -0.5, // point 0
    0.5, -0.5, // point 1
    -0.5,  0.5, // point 2
    0.5,  0.5  // point 3
};
```

0 1
GLDrawArrays

- GL_TRIANGLE_STRIP

```cpp
static const GLfloat squareVertices[] = {
    -0.5, -0.5, // point 0
    0.5, -0.5, // point 1
    -0.5,  0.5, // point 2
    0.5,  0.5 // point 3
};
```
**GlDrawArrays**

- **GL_TRIANGLE_STRIP**

```c
static const GLfloat squareVertices[] = {
    -0.5, -0.5, // point 0
    0.5, -0.5, // point 1
    -0.5,  0.5, // point 2
    0.5,  0.5  // point 3
};
```
GlDrawArrays

- GL_TRIANGLE_STRIP

```c
static const GLfloat squareVertices[] = {
    -0.5, -0.5, // point 0
    0.5, -0.5, // point 1
    -0.5,  0.5, // point 2
    0.5,  0.5  // point 3
};
```
**GLDrawArrays**

---

**GL_TRIANGLE_STRIP**

- Vertices: 0, 2, 4
- Indices: 0, 2, 4

**GL_TRIANGLE_FAN**

- Vertices: 3, 2, 1, 0
- Indices: 3, 2, 1, 0

**GL_TRIANGLES**

- Vertices: 5, 3, 4, 0, 1
- Indices: 5, 3, 4, 0, 1
Multi-Touch Events

• Touch Sequences
• Touch and Event Objects
• Touch Delivery
• Single Touch
• Multiple Touches
• Multiple Views
• Touch Routing
Multi-Touch Events

- Touch Sequences
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
Single Touch Sequence
Single Touch Sequence

- Begins when the finger contacts the screen
Single Touch Sequence

- Begins when the finger contacts the screen
- Continues as the finger moves along the surface of the screen
Single Touch Sequence

• Begins when the finger contacts the screen
• Continues as the finger moves along the surface of the screen
• Ends when the finger lifts from the screen.
Multi-Touch Events

- Touch Sequences
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
UITouch

- Represents a single finger

```objc
@property(nonatomic,readonly) NSTimeInterval      timestamp;
@property(nonatomic,readonly) UITouchPhase        phase;
@property(nonatomic,readonly) NSUInteger          tapCount;

@property(nonatomic,readonly,retain) UIWindow    *window;
@property(nonatomic,readonly,retain) UIView      *view;

- (CGPoint)locationInView:(UIView *)view;
- (CGPoint)previousLocationInView:(UIView *)view;
```

UITouchPhase

-UITouchPhaseBegan
-UITouchPhaseMoved
-UITouchPhaseStationary
-UITouchPhaseEnded
-UITouchPhaseCancelled
UIEvent

- A container for one or more touches

@property(nonatomic,readonly) NSTimeInterval timestamp;

- (NSSet *)allTouches;
- (NSSet *)TouchesForWindow:(UIWindow *)window;
- (NSSet *)TouchesForView:(UIView *)view;
UIEvent
UIEvent
UIEvent
UIEvent
UIEvent

UIEvent

UITouch

UITouch

UITouch

Window A

View A

UITouch

UITouch

UITouch

Window B

View B

View C

Window B
UIEvent

- (NSSet *)allTouches;
UIEvent

- (NSSet *)allTouches;
UIEvent

- (NSSet *)touchesForWindow:(UIWindow *)window;
UIEvent

Window A

- (NSSet *)touchesForWindow:(UIWindow *)window;
Window B

- (NSSet *)touchesForWindow:(UIWindow *)window;
- (NSSet *)touchesForView:(UIView *)view;
UIEvent

View A

- (NSSet *)touchesForView:(UIView *)view;
**UIEvent**

View B

```objective-c
-(NSSet *)touchesForView:(UIView *)view;
```
View C

- (NSSet *)touchesForView:(UIView *)view;
Multi-Touch Events

- Touch Sequences
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
Receiving Touches

UIResponder

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;

• **Any** responder can receive touch events
  
  - UIView
  - UIControl
  - UIViewController!
  - UIApplication
Multi-Touch Events

- Touch Sequences
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
Single Touch Sequence
Single Touch Sequence

touchesBegan:withEvent:

UITouch 0x123
Phase: Began
Location: 160, 120
Single Touch Sequence

touchesMoved:withEvent:

UITouch 0x123
Phase: Moved
Location: 160, 160
Single Touch Sequence

touchesMoved:withEvent:

UITouch 0x123
Phase: Moved
Location: 160, 200
Single Touch Sequence

touchesMoved:withEvent:
Single Touch Sequence

touchesEnded:withEvent:
Today’s Topics

• Touch Sequences
• Touch and Event Objects
• Touch Delivery
• Single Touch
• Multiple Touches
• Multiple Views
• Touch Routing
Multiple Touch Sequence

• By default, only first touch within a view is delivered

• Enable multiple touch tracking
  • -[view setMultipleTouchEnabled: YES]
Multiple Touch Sequence
Multiple Touch Sequence

UI Touch 0x123
Phase: Began
Location: 120, 200

touchesBegan:withEvent:
Multiple Touch Sequence

UI Touch 0x123
Phase: Moved
Location: 120, 240

touchesMoved:withEvent:
Multiple Touch Sequence

- UITouch 0xabc
  - Phase: Began
  - Location: 200, 200

- UITouch 0x123
  - Phase: Stationary
  - Location: 120, 240

```
touchesBegan:withEvent:
```
Multiple Touch Sequence

UITouch 0x123
Phase: Moved
Location: 120, 280

UITouch 0xabc
Phase: Moved
Location: 200, 240

touchesMoved:withEvent:
Multiple Touch Sequence

UI Touch 0x123
Phase: Stationary
Location: 120, 280

UI Touch 0xabc
Phase: Moved
Location: 200, 280

touchesMoved:withEvent:
Multiple Touch Sequence

UITouch 0x123
Phase: Ended
Location: 120, 280

UITouch 0xabcd
Phase: Ended
Location: 200, 280

touchesEnded:withEvent:
Today’s Topics

- Touch Sequences
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
Multiple Views Touch Sequence
Multiple Views Touch Sequence

UI Touch 0x123
Phase: Began

touchesBegan:
withEvent:
Multiple Views Touch Sequence

touchesMoved:
withEvent:

UITouch 0x123
Phase: Moved
Multiple Views Touch Sequence

touchesBegan:
withEvent:

UITouch 0x123
Phase: Stationary

UITouch 0xabc
Phase: Began

Tuesday, November 2, 2010
Multiple Views Touch Sequence

touchesMoved: touchesMoved:
withEvent: withEvent:
Multiple Views Touch Sequence

touchesMoved:
withEvent:

UITouch 0x123
Phase: Stationary

UITouch 0xabc
Phase: Moved

Tuesday, November 2, 2010
Multiple Views Touch Sequence

UITouch 0x123
Phase: Ended

UITouch 0xabc
Phase: Ended

touchesEnded: touchesEnded:
withEvent: withEvent:

Tuesday, November 2, 2010
Multiple Views

• By default, multiple views can receive touches simultaneously

• Change this behavior if you need
  
  ▪ Example: avoid multiple buttons being simultaneously clicked
Multiple Views Demo
Multiple Views Demo
Today’s Topics

- Touch Sequences and Phases
- Touch and Event Objects
- Touch Delivery
- Single Touch
- Multiple Touches
- Multiple Views
- Touch Routing
Responder Chain

UIView
Responder Chain

UIView

superview

UIView
Responder Chain

UIView

superview

UIView

superview

UIView
Responder Chain

UIView
superview
UIView
superview
UIView

UIViewController
Responder Chain

UIView

superview

UIView

superview

UIView

window

UIViewController

UIView Controller
Responder Chain

UIView

superview

UIView

superview

UIView

window

UIViewController

UIApplication

62
Hit Testing
Hit Testing

Window

View

Subview
Hit Testing

Window

View

Subview

hitTest:withEvent:
Hit Testing

- userInteractionEnabled

```swift
hitTest:withEvent:
```
Hit Testing

- `userInteractionEnabled`
- `hidden/alpha`

`hitTest:withEvent:`
Hit Testing

- userInteractionEnabled
- hidden/alpha
- pointInside:withEvent:
Hit Testing

- userInteractionEnabled
- hidden/alpha
- pointInside:withEvent:
Hit Testing

Window

View

Subview

UITouch

view

window

Tuesday, November 2, 2010
Making View Easier To Hit

• Small items can be hard to touch

• Make UI easy to touch
  
  ▪ Bigger is better
  
  ▪ Use standard controls
    
    ▪ UIKit automagically treats taps “close enough” to a view as a tap on the view
    
    ▪ This includes any UIButton subclasses, etc...

  ▪ Cheat – lie about your view’s hit region to UIKit

- (BOOL)pointInside:(CGPoint)point withEvent:(UIEvent *)event;
  
  // default returns YES if point is in bounds

- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event;
  
  // recursively calls pointInside:withEvent: