iPhone Programming
CMSC 498i – Spring 2010

OpenGL • Accelerometer • Multi-Touch

Lecture #16 – Chuck Pisula
This Week’s Topics

• Gaming Related Functionality
  ▪ OpenGL Basics
  ▪ Accelerometer, and Touch Input
  ▪ Audio & Video
  ▪ iPod Library Access
  ▪ GameKit
Game Controls
Game Controls

• Need to decide how the user controls your game
Game Controls

• Need to decide how the user controls your game

• Accelerometer
  ▪ Requires user to be stationary
  ▪ Getting the feel correct requires tuning

• Touch Input
  ▪ Finger vs. Mouse – size, precision differences
  ▪ Direct control – user touches items they are manipulating
  ▪ Indirect control – user controls items without directly touching them
    ▪ Virtual Analog Joysticks
    ▪ Virtual D-Pad
    ▪ Other
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

- Easy to get up and running on iPhone
  - Xcode includes an iPhone OpenGL ES Application Template
  - Takes care of standard setup and configuration
OpenGL ES 1.1 –vs- 2.0

• Features in 2.0 – (*a big change*)
  ▪ Requires developers to be more familiar with computer graphics
  ▪ In order to render anything on screen, must write *shaders*
  ▪ Games will be super fast, look more… *awesome*…

• Device Compatibility
  ▪ 2.0 specification supported on iPhone 3GS and newer iPod Touches
  ▪ 1.1 specification supported on all platforms
  ▪ Programs written to use 2.0 features are not compatible with 1.1 hardware

• 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
• Two types
  ▪ Vertex Shaders – Foo.vsh – Executed for each vertex
  ▪ Fragment Shaders – Foo.fsh – Executed once for each fragment (~pixel)
• Mini GL programs...
  ▪ Load source, compile, attach to a program, and link
  ▪ Xcode’s sample ES2Renderer shows how to do all of this
OpenGL ES Template

• Main window consists of one EAGLView
  ▪ UIView subclass, which wraps CAEAGLLayer
  ▪ Manages GL scene animation
  ▪ Creates “best” ESRenderer for you to put your drawing code in...

• ESRenderer
  ▪ Has 2 subclasses: ES1Renderer, and ES2Renderer
  ▪ EAGLView attempts to create ES2Renderer, and falls back to ES1Renderer if the hardware does not support ES 2.0
  ▪ Drawing code goes in each -render method
  ▪ To support just ES 1.1, remove ES2Renderer references from the project
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

- **(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
Setting Up An OpenGL View

This template provides a starting point for an application that uses an OpenGL ES-based view. It provides a view into which you render your OpenGL ES scene, and a timer to allow you to animate the view.
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
 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
glyphiconPointer(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

```cpp
// 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);
gTranslate(squareSizeShown, squareSizeShown, 1.0f);
glRotatef(squareAngle, 0, 0, 1.0f);

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

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

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

// Draw as a “Triangle Strip”
gDrawArrays(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);
```
GlDrawArrays

- GL_TRIANGLES

```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

```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

```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

GL_TRIANGLE_FAN

GL_TRIANGLES
Demo
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
  - Example use – OpenGL project template in XCode

```objective-c
displayLink = [CADisplayLink displayLinkWithTarget:self
                                      selector:@selector(renderAndUpdate)-checkbox;
[displayLink setFrameInterval:2]; // 2x the display refresh rate!
[displayLink addToRunLoop:[NSRunLoop currentRunLoop]
              forMode:NSDefaultRunLoopMode];
```
What Time Is It?

- CFAbsoluteTimeGetCurrent()
  - Not necessarily monotonically increasing !!!
- mach_absolute_time()
  - Monotonically increasing
  - CPU dependent time scale
What Time Is It?

- CFAbsoluteTimeGetCurrent()
  - Not necessarily monotonically increasing !!!
- mach_absolute_time()
  - Monotonically increasing
  - CPU dependent time scale

```c
uint64_t time = mach_absolute_time();
mach_timebase_info_data_t base;
mach_timebase_info(&base);

double secondsScaleFactor = 1e-9 * (double)base.numer / (double)base.denom;
double seconds = scaleToSecondsFactor * (double)time;
```
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

0.5g

0.50g

0.75g
Accelerometer

- Single shared instance of **UILevelManager**
- Sends **UIAccelerometer** values
  - Values sent to the **UILevelManagerDelegate**
  - G-force values reported by the hardware
  - Timestamp
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
The Accelerometer Interface

- Getting the raw accelerometer data
  - Classes
    - UIAccelerometer
    - UIAcceleration
  - Protocol
    - UIAccelerometerDelegate
Configuring The Accelerometer

• Starting the event delivery

```c
- (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
  - Only one delegate per application

• Update Interval – use only what you need
  - Game input: 30–60 Hz
  - Orientation detection: 10–20 Hz
Defining Your Delegate Object

- Processing the accelerometer data

```c
- (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
Disabling Event Delivery

- Stopping the event delivery

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

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

• Use filters to isolate data components

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

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

• Consider a stationary device on a table…
  ▪ Only force exerted is the force of gravity
  ▪ The accelerometer z-axis has a steady value of -1.0 g
  ▪ This data comes to us in the time domain

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

• Consider a stationary device on a table…
  ▪ Only force exerted is the force of gravity
  ▪ The accelerometer z-axis has a steady value of -1.0 g
  ▪ This data comes to us in the time domain

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

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

Fourier Transform
Filtering Accelerometer Data

- Examining the accelerometer data
Filtering Accelerometer Data

- Examining the accelerometer data
Filtering Accelerometer Data

- Examining the accelerometer data

\[ f(t) \]

\[-1.0g\]
Filtering Accelerometer Data

- Changing to the frequency domain

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

- We see something more interesting…

\[ f(t) \]
Filtering Accelerometer Data

- We see something more interesting…
- No flat line at -1.0g
Filtering Accelerometer Data

- We see something more interesting…
- No flat line at -1.0g
- Accelerometers generating waveform pattern
Filtering Accelerometer Data

- We see something more interesting...

- No flat line at -1.0g
- Accelerometers generating waveform pattern
- Waveform centered around the -1.0g

\[ f(t) \]
Filtering Accelerometer Data

- We see something more interesting…

- No flat line at -1.0g
- Accelerometers generating waveform pattern
- Waveform centered around the -1.0g
- This is because gravity hasn’t gone away, and is still affecting the accelerometer data

\[ f(t) \]
Filtering Accelerometer Data

• We see something more interesting…

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

\[ F(\omega) \]
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-data is dominated by gravity
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

\[ F(\omega) \]
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
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
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
- Bubble Level sample (low-pass filter)
Multi-Touch Events
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

```objective-c
@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

- (NSSet *)allTouches;
- (NSSet *)allTouches;
UIEvent

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

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

Window B

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

View A

- (NSSet *)touchesForView:(UIView *)view;
View B

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

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:
Single Touch Sequence

touchesMoved:withEvent:
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

UITouch 0x123
Phase: Began
Location: 120, 200

touchesBegan:withEvent:
Multiple Touch Sequence

UITouch 0x123
Phase: Moved
Location: 120, 240

touchesMoved:withEvent:
Multiple Touch Sequence

touchesBegan:withEvent:
Multiple Touch Sequence

UITouch 0x123
Phase: Moved
Location: 120, 280

UITouch 0xabc
Phase: Moved
Location: 200, 240

touchesMoved:withEvent:
Multiple Touch Sequence

touchesMoved:withEvent:

UITouch 0x123
Phase: Stationary
Location: 120, 280

UITouch 0xabc
Phase: Moved
Location: 200, 280
Multiple Touch Sequence

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

UITouch 0x123
Phase: Began
Location: 120, 200

touchesBegan:
withEvent:
Multiple Views Touch Sequence

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

touchesMoved:
withEvent:
Multiple Views Touch Sequence

touchesBegan:
withEvent:

UITouch 0x123
Phase: Stationary
Location: 120, 240

UITouch 0xabc
Phase: Began
Location: 200, 200
Multiple Views Touch Sequence

UITouch 0x123
Phase: Moved
Location: 120, 280

UITouch 0xabc
Phase: Moved
Location: 200, 240

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

touchesMoved:
withEvent:

UITouch 0x123
Phase: Stationary
Location: 120, 280

UITouch 0xabc
Phase: Moved
Location: 200, 280
Multiple Views Touch Sequence

UITouch 0x123
Phase: Ended
Location: 120, 280

UITouch 0xabc
Phase: Ended
Location: 200, 280

touchesEnded: withEvent:
touchesEnded: withEvent:
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

UI View

superview

UI View
Responder Chain
Responder Chain

UIView
  superview
  UIView
    superview
    UIView
      UIViewController
        UIView
          UIViewController
Responder Chain
Responder Chain

(UIView
superview
(UIView
superview
(UIView
(UIWindow
window
(UIViewController
(UIApplication

66
Hit Testing
Hit Testing

Window

View

Subview
Hit Testing

hitTest:withEvent:
Hit Testing

- `hitTest:withEvent:`
- `userInteractionEnabled`
Hit Testing

- `userInteractionEnabled`
- `hidden/alpha`

`hitTest:withEvent:`
Hit Testing

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

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