More on Core Graphics, Animation Working with Images

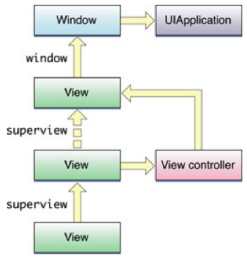
CS 344 Mobile App Development

Robert Muller

Touch Events

- UIView is a subclass of UIResponder
- Responding to Touch Events:
  - touchesBegan:withEvent:
  - touchesMoved:withEvent:
  - touchesEnded:withEvent:
  - touchesCancelled:withEvent:

Event Responder Chain



```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch *aTouch = [touches anyObject];    // NB: type downcast

    CGPoint touched = [aTouch locationInView:self.view];

    myViewA.center = touched;

    NSLog([NSString stringWithFormat:@"touched at (%g, %g)",
                touched.x, touched.y]);
}
```


UIEvent Types

```
typedef enum {
   UIEventTypeTouches,
   UIEventTypeMotion,
   UIEventTypeRemoteControl,
} UIEventType;
```

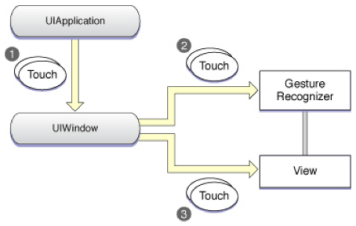

UIEvent Subtypes

```
typedef enum {
   UIEventSubtypeNone                      = 0,

   UIEventSubtypeMotionShake               = 1,

   UIEventSubtypeRemoteControlPlay         = 100,
   UIEventSubtypeRemoteControlPause        = 101,
   UIEventSubtypeRemoteControlStop         = 102,
   ...
   UIEventSubtypeRemoteControlEndSeekingForward   = 109,
} UIEventSubtype;
```

Gestures

Gesture	UIKit class
Tapping (any number of taps)	`UITapGestureRecognizer`
Pinching in and out (for zooming a view)	`UIPinchGestureRecognizer`
Panning or dragging	`UIPanGestureRecognizer`
Swiping (in any direction)	`UISwipeGestureRecognizer`
Rotating (fingers moving in opposite directions)	`UIRotationGestureRecognizer`
Long press (also known as "touch and hold")	`UILongPressGestureRecognizer`

Gestures and Touch Events



Graphics

- 2D Graphics
  - Core Graphics : Quartz 2D Graphics Engine
    - Bitmaps, vector graphics, PDF, …
  - Convenience methods in UIKit
- 3D Graphics:
  - OpenGL ES (Open GL for Embedded Systems)

Core Graphics/Quartz

- Contexts : contains drawing parameters and all device-specific information that the drawing system needs to perform any subsequent drawing commands
- UIKit creates a graphics context for drawing and attaches it to your UIVew
  - adjusts the transform of that context so that its origin matches the origin of view's bounds
- Retrieve graphics context using the UIGraphicsGetCurrentContext function

- (void)drawRect:(CGRect)rect

- UIView defines drawRect, a 2D graphics application should override drawRect
- Never call drawRect explicitly
- setNeedsDisplay property of the view

Core Graphics Contexts

```
(CGContextRef) UIGraphicsGetCurrentContext();

  … Construct a CG Path  …

  CGContextAddPath(CGContextRef, CGPathRef);
  CGContextDrawPath(context, kCGPathFill);
```

Core Graphics Paths

- CGPathMoveToPoint
- CGPathAddArc
- Etc
- Can save the path but must add path to a CGContext to be displayed.

CG Contexts Convenience Procedures

```
(CGContextRef) UIGraphicsGetCurrentContext();

  CGContextBeginPath(CGContextRef);

  … Some CG Path operations …

  CGContextDrawPath(context, kCGPathFill);
```


CG Context Convenience Procedures

- Instead of explicitly constructing a CGPath, you can use CGContext convenience procedures:
  - CGContextMoveToPoint
  - CGContextAddArc
  - Etc.

Other Convenience Operations

- Convenience methods in UIKit wrapping underlying routines in CoreGraphcs:
  - UIRectFill
  - UIRectFrame
  - UIRectClip
  - …
  - UIColor, UIFont

Images

- UIImage
- UIIMageView
- UIScrollView

Animation

- Framework : Core Animation + Wrappers
- Animation Blocks:
  beginAnimations:context:

  … Move Things Around…

  commitAnimations

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    [UIView beginAnimations:@"Shift" context:nil];
    [UIView setAnimationDuration:1];

    if (myViewA.center.y == 115) {

        myViewA.center = CGPointMake(80, 345);
        myViewB.center = CGPointMake(240, 115);
    } else
    {
        myViewA.center = CGPointMake(80, 115);
        myViewB.center = CGPointMake(240, 345);
    }
    [UIView commitAnimations];
}
```



```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event  {

    [UIView beginAnimations:@"Shift" context:nil];
    [UIView setAnimationDuration:1];

    if (myViewA.center.y == 115) {

    myViewB.alpha = .5;
    myViewA.center = CGPointMake(80, 345); // x was 115
    myViewB.center = CGPointMake(240, 115);
    } else {
    myViewA.center = CGPointMake(80, 115);
    myViewB.center = CGPointMake(240, 345);
    }
    CGAffineTransform transform = CGAffineTransformRotate([self transform],
                                                          -3.14159 / 2.0);
    CGRect newBounds = CGRectApplyAffineTransform([myViewA bounds], transform);
    myViewA.bounds = newBounds;
    [UIView commitAnimations];
}
```