Foundation Classes
Views, Touch Events

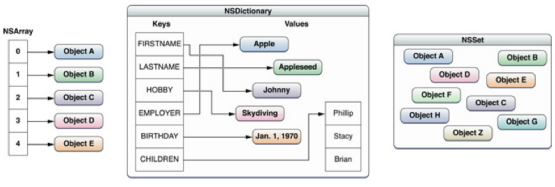
CS 344 Mobile App Development

Robert Muller

Foundation Classes

- NSString : strings are immutable! (i.e., they are *static*, as in Java), but there is an NSMutableString
- NSValue : generic wrapper for scalar values
- NSNumber : wrapper for float, double, int, … (a subclass of NSValue)
- NSData : wrappers for C-style byte buffers

Collections

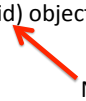


Foundation Classes

- Arrays: NSArray & NSMutableArray

- (NSUInteger) count;

- (id) objectAtIndex:(NSUInteger)index;



NB: dynamic return type!

Foundation Classes

- Sets: NSSet, NSMutableSet
- Ordered Sets: NSOrderedSet, NSMutableOrderedSet
- Maps: NSDictionary, NSMutableDictionary

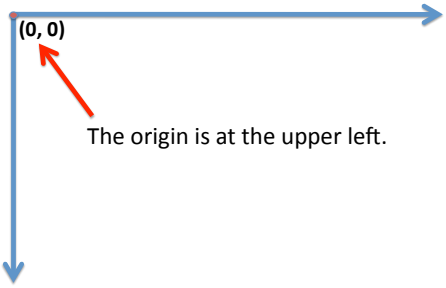
Orders

- Let A be a set and let R be a binary relation on A. (i.e., R \subseteq A x A).
- R is a partial order on A iff R is:
  - reflexive : (a, a) in R forall a in A,
  - antisymmetric: if (a, b) in R & (b, a) in R then a = b,
  - transitive: if (a, b) in R & (b, c) in R then (a, c) in R.
- R is a total order on A iff R is a partial order on A and for all a, b in A, either (a, b) in R or (b, a) in R.

Views

- A view (i.e. UIView subclass) represents a rectangular area
  - Defines a Cartesian coordinate space
  - Draws and handles events in that rectangle

View Coordinate Space



View Hierarchies

- A view has one superview:
  - (UIView *)superview;
- A view can have many subviews:
  - (NSArray *)subviews;
  - Subviews at higher indices above subviews at lower indices.
- UIWindow:
  - The UIView at the top of the view hierarchy, generally only one UIWindow in an iOS application.

Views & Core Graphics (CG)

- Views often allocated using Core Graphics:
  [[UIView alloc] initWithFrame:(CGRect)frame];
- Key types:
  - CGFloat, CGPoint, CGSize & CGRect.
  - CGFloat : typdef float CGFloat;

Points, Sizes & Rects

```
typedef struct {
    CGFloat x;
    CGFloat y;
    } CGPoint;
```

```
….
CGRect myPoint = CGMakePoint(100.0, 50.0);
CGFloat x = myPoint.x;    // NB: struct field access
….
```


Points, Sizes & Rects

```
typedef struct {
    CGFloat width;
    CGFloat height;
    } CGSize;
```

```
…
CGSize mySize = CGSizeMake(100.0, 200.);
CGFloat width = mySize.width;
….
```

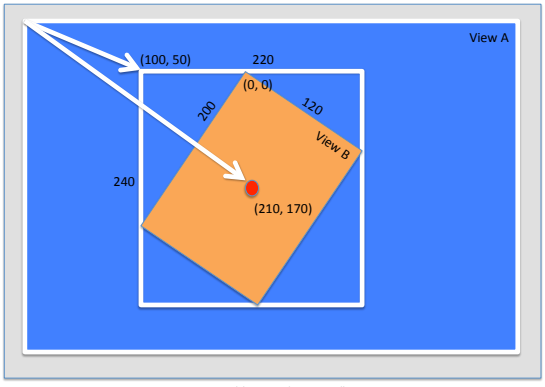
Points, Sizes & Rects

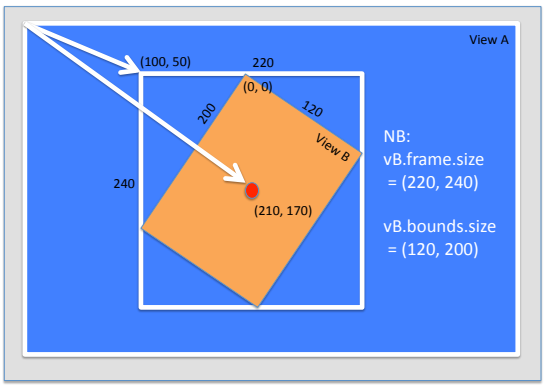
```
typedef struct {
    CGPoint origin;
    CGSize  size;
    } CGRect;
```

```
CGRect cgr = CGMakeRect(10., 20., 30., 40.);
UIView *myView =
            [[UIView alloc] initWithFrame:cgr];
[self.view addSubview:myView];
....
```


Properties of Views

- @property CGRect bounds; the internal reference frame for the view
- @property CGPoint center; relative to parent view
- @property CGRect frame; the reference frame for the view as represented by the parent view





Touch Events

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


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

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

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

}
```

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


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];
}
```


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