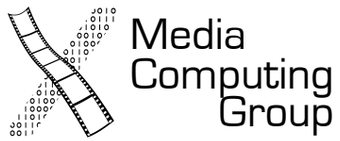


RWTH AACHEN UNIVERSITY

Mobile Application Development
L06: iOS Drawing and Animation

Jonathan Diehl (Informatik 10)
Hendrik Thüs (Informatik 9)

Views

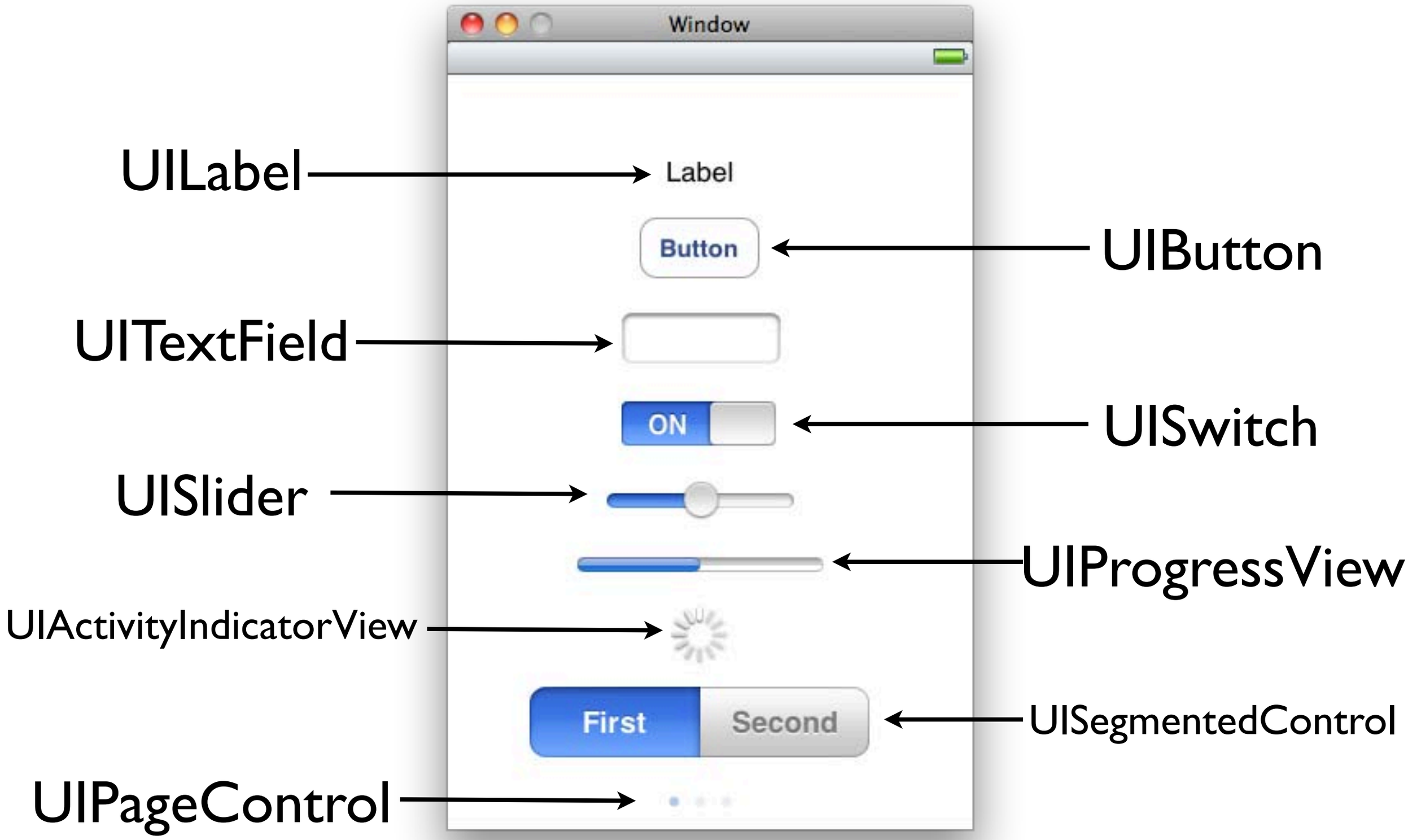
UIView

Defines a rectangular area on the screen

Responsibilities

Draw content
Arrange subviews
React to touch events

Simple Views



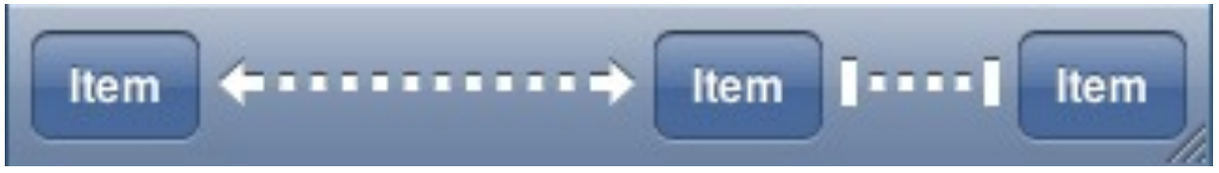
Bar Views

Status Bar

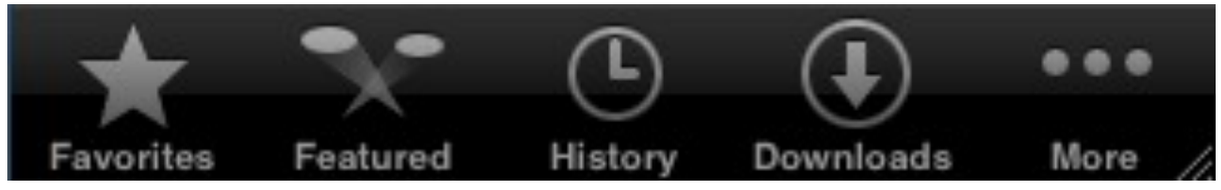
UINavigationBar

UISearchBar

UIToolbar

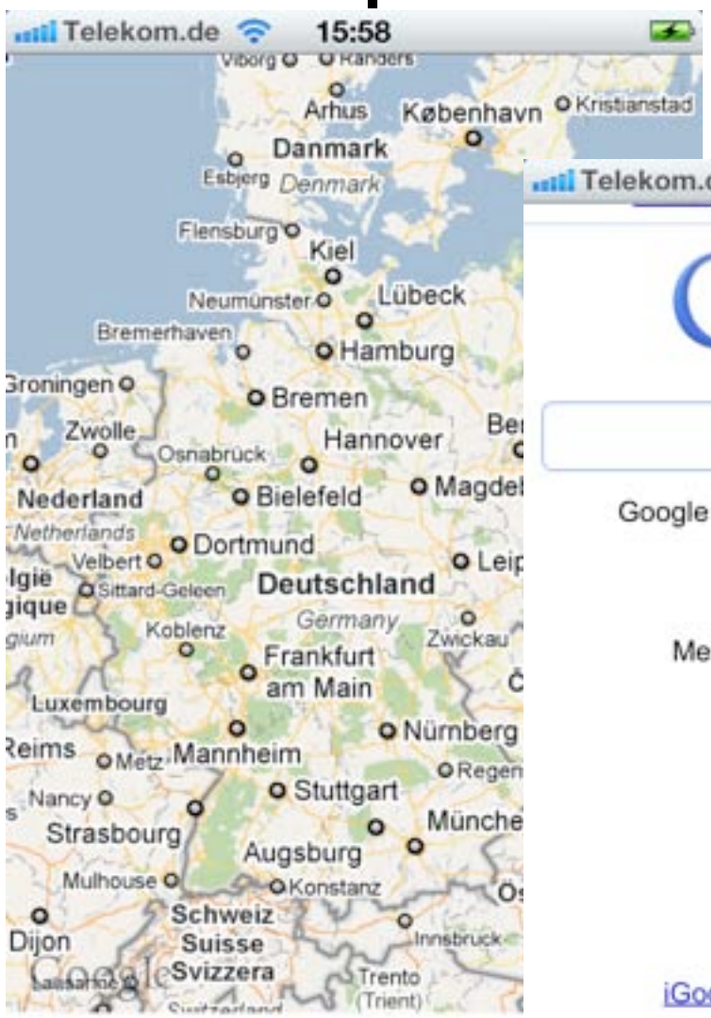


UITabBar



Complex Views

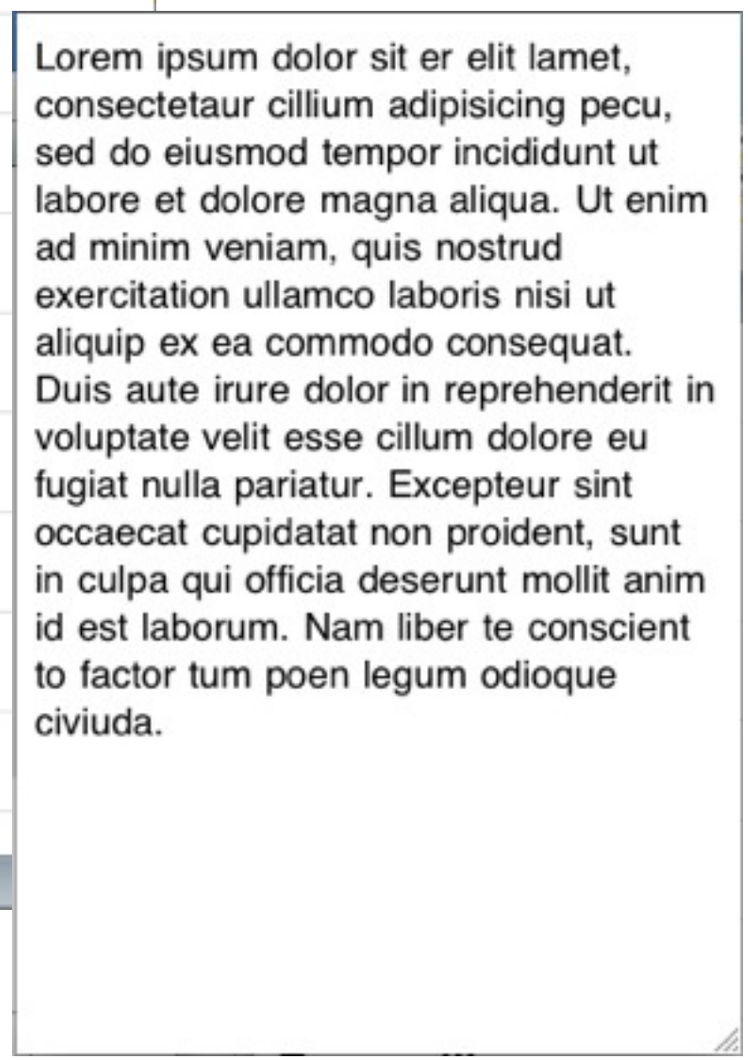
MKMapView



UITableView

UIWebView

UITextView



Draw content

- How? CoreGraphics
  - Get the graphics context
  - Configure the brush (color, line width, etc.)
  - Define a shape
  - Stroke or fill the shape
- Where? Subclass UIView
  - override - (void)drawRect:(CGRect)rect
  - For redraw: [view setNeedsDisplay]

Drawing Example

```
- (void)drawRect:(CGRect)rect {

    // Get the graphics context
    CGContextRef context = UIGraphicsGetCurrentContext();

    // Set stroke and fill color
    [[UIColor whiteColor] set];

    // Define a line as the shape to be drawn
    CGContextMoveToPoint(context, 10.0, 30.0);
    CGContextAddLineToPoint(context, 310.0, 30.0);

    // Stroke the line
    CGContextStrokePath(context);

}
```

Layout (Sub-) Views

- Where?
  - In the interface description (.xib)
  - In a View Controller (e.g., loadView)
  - In a View (initWithFrame:)
- How?
  - Manage the view hierarchy (subviews, superviews)
  - Manage the frame and bounds

Frame vs. Bounds

Frame
Origin: 0, 0
Size : 320, 480

Bounds
Origin: 300, 100
Size : 320, 480

Frame vs. Bounds

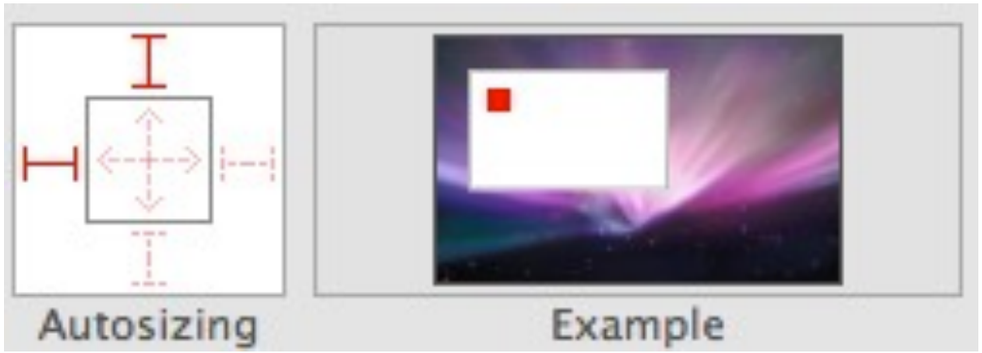
Frame
Origin: 0, **100**
Size : 320, 480

Bounds
Origin: 300, 100
Size : 320, 480

Frame vs. Bounds

Frame
Origin: 0,   0
Size : 320, **380**

Bounds
Origin: 300, 100
Size : 320, **380**



Automatic Layout

- parent.autoresizesSubviews = YES
- Define an autoresize mask
  - In the interface description (.xib)
  - view.autoresizingMask = ...
- Resizing Behavior
  - flexible left / top / right / bottom margin
  - flexible width / height

React to Touch Events

- How?
  - view.userInteractionAllowed = YES
  - Multitouch: view.multipleTouchEnabled = YES
- Where?
  - Subclass UIView
  - Override event handling methods
  - Events are also forwarded to the View Controller!

Touch Events

```
// initial touch
- (void)touchesBegan:(NSSet *)touches
           withEvent:(UIEvent *)event

// updated touch
- (void)touchesMoved:(NSSet *)touches
           withEvent:(UIEvent *)event

// cancelled touch (by external event)
- (void)touchesCancelled:(NSSet *)touches
               withEvent:(UIEvent *)event

// finished touch
- (void)touchesEnded:(NSSet *)touches
           withEvent:(UIEvent *)event
```

Animation

Core Animation

Implicit Animations

Changing animatable properties triggers implicit animations

Explicit Animations

Create an animation object that defines and controls the animation

Implicit Animations

```
   static int i = 0;

// start an implicit animation
[UIView animateWithDuration:1.0 animations:^{

   // move the view
   view.center = CGPointMake(50+random() % 220,
                             50+random() % 360);

   // rotate the view
   view.transform = CGAffineTransformMakeRotation(
                        random() % 360);

   // change the color
   view.backgroundColor = i++ % 2 ? [UIColor blueColor] :
                                    [UIColor redColor];
}];
```

Explicit Animations

- Create Animation Object
  - CABasicAnimation
  - CAKeyframeAnimation
- Configure animation
  - Duration
  - Timing function
  - Key-path of animated property
  - From and to value

Example: Move Animation

```
CGPoint position = CGPointMake(100, 200);

// create an animation object
CABasicAnimation *move = [[CABasicAnimation alloc] init];

// configure the key path and value
move.keyPath = @"position";
move.toValue = [NSValue valueWithCGPoint: position];

// set the animation duration
move.duration = 1.0;

// start the animation
[timeLabel.layer addAnimation:move
                       forKey:@"moveAnimation"];
```

Transformations

- Animations can use affine transformations to manipulate the layer's geometry
  - CGAffineTransform
    - Every rendered pixel is transformed
- Common transformations:
  - CGContextTranslateCTM: move origin
  - CGContextScaleCTM: change size
  - CGContextRotateCTM: rotate around anchorPoint