VIEWS

• What is a view?
  • Subclass of a UIView
  • defines coord space for drawing, touch events
• There is a hierarchy
  • Top is effectively var view UIView
    • view’s bounds can change on rotation
  • Each view has a parent: var superview: UIView?
  • Potentially children: var subviews: [UIView]
    • Order matters
    • Views can be clipped to their bounds, or (usually) not
VIEW INITIALIZATION

• Avoid if possible, but:
  • init(frame: CGRect)    // from code
  • init(coder: NSCoder) // from a storyboard

• If you have one, you must have both:
  ```swift
class PeteView : UIView {
    override init(frame: CGRect) {
      super.init()
      setup()
    }

    required init?(coder aDecoder: NSCoder) {
      super.init()
      setup()
    }

    func setup() {
      // do stuff
    }
  }
  ```

COORDINATE SYSTEMS

• CGFloat
  • Used everywhere in UI
  • Must convert other numbers: let aCG = CGFloat(42)

• CGPoint
  • var pt = CGPoint(x: 42, y: 0)

• CGSize
  • var sz = CGSize(width: 42, height: 42)
CGRect

struct CGRect {
    var origin: CGPoint
    var size: CGSize
}

// And other methods and properties:
let r = CGRect(x: CGFloat, y: CGFloat,
               width: CGFloat, height: CGFloat)

Many methods and properties:
    var minX: CGFloat
    var midY: CGFloat
    intersects(CGRect) -> Bool
    contains(CGPoint) -> Bool
    ...

VIEW COORDINATE SYSTEM

- origin in upper left
- units are points, not pixels
  - Pixels are min units per-device
  - Point is an integer multiple of pixel
    - 2 or 3 for retina
    - 1 for older
      - a view's contentXscaleFactor
- A view's view of the world
  - var bounds: CGRect
- A view's place in the world
  - var center: CGPoint
  - var frame: CGRect
FRAME VS BOUNDS

frame is position in superview
bound is a view local view of itself

view.bounds = CGRect(x: 0, y: 0, width: 100, height: 100)
view.frame = CGRect(x: 100, y: 100, width: 241, height: 241)

CREATING VIEWS

• Two ways:
  • Xcode interface builder (storyboard)
    • pull out UIView from object library
    • use Identity Inspector to specify the (custom) class of the new view
  • From code:
    let r = CGRect(x: 0, y: 0, width: 42, height: 54)
    let button = UIButton(frame: r)
    button.setTitle("Hello, world", for: UIControl.State.normal)
    UIView.addSubview(button)
**Draw: UIColor**

- Colors are of type UIColor
  - some special colors: `UIColor.green`
  - try typing “colorliteral” into Xcode (you get a ColorPicker)
- Colors can have transparency (alpha)
  - `UIColor.orange.withAlphaComponent(0.25)`
- Make drawing transparent, no effect on UIButton etc.
  - `var opaque = false`
- Make entire view transparent
  - `var alpha = 0.25`

**Draw: Core Animation Layer**

- `UIView.layer`: `CALayer`
- Useful stuff:
  - `.layer.cornerRadius`
  - `.layer.borderWidth`
  - ...
- It's own color nomenclature:
  - `UIColor.green.cgColor`
- Animation
  - `next week`
**Draw: Transparency**

- Subviews are ordered
  - front view occlude those in back
  - transparent front views let back views show through
- `.isHidden: Bool`

**Draw: Fonts**

- Can change fonts in views like UIButton etc.
- Get preferred font for a given text style:
  ```swift
  static func preferredFont(forTextStyle: UIFontTextStyle) -> UIFont
  .headline
  .body
  .footnote
  ```
- Or through more flexible ways:
  ```swift
  let font = UIFont(name: "Menlo", size: 24)
  ```
  - or through a FontPicker (attributes pane)
  - or scale to user's preferred size:
    ```swift
    let metrics = UIFontMetrics(forTextStyle: .body)
    let fontToUse = metrics.scaledFont(for: font)
    ```
- Or use system fonts
  ```swift
  static func systemFont(ofSize: CGFloat) -> UIFont
  static func boldSystemFont(ofSize: CGFloat) -> UIFont
  ```
**Draw: Drawing Text**

- UILabels are easy, but...

- Draw using NSAttributedString:
  ```swift
  let text = NSAttributedString(string: "hello")
  text.draw(at: CGPoint(x: 42, y: 11))
  let textSize: CGSize = text.size
  ```

---

**Drawing**

- How?
  - Subclass UIView and override draw()
    - `override func draw(_ rest: CGRect)`
  - You can draw outside of rect.
  - Rect is just a hint
  - `.bounds` describes a view’s entire area

- You do not ever call draw(). Instead:
  - `.setNeedsDisplay()`
  - system gathers up multiple redraws into a single event
How to implement drawing?
- Either get a drawing context (printing, double-buffering), or
  - UIBezierPath (easy)
- UIBezierPath
  - create paths from lines, arcs, rounded rects etc.
  - set colors, linewidths, fonts, etc.
  - optionally set clipping to a closed path
  - stroke or fill

Create a path
```swift
let path = UIBezierPath()
```
DRAWING

- Create a path
  \texttt{let path = UIBezierPath()}

- Move, add lines and arcs
  \texttt{path.move(to: CGPoint(x: 40, y: 40))}
  \texttt{path.addLine(to: CGPoint(x: 80, y: 40))}
CREATE A PATH

```
let path = UIBezierPath()
```

MOVE, ADD LINES AND ARCS

```
path.move(to: CGPoint(x: 40, y: 40))
path.addLine(to: CGPoint(x: 80, y: 40))
path.addLine(to: CGPoint(x: 80, y: 80))
path.addLine(to: CGPoint(x: 40, y: 80))
```
• Create a path
   let path = UIBezierPath()

• Move, add lines and arcs
   path.move(to: CGPoint(x: 40, y: 40))
   path.addLine(to: CGPoint(x: 80, y: 40))
   path.addLine(to: CGPoint(x: 80, y: 80))
   path.addLine(to: CGPoint(x: 40, y: 80))

• Close (optional)
   path.close()

• Set attributes
   UIColor.blue.setStroke()
   UIColor.red.setFill()
DRAWING

- Create a path
  ```swift
  let path = UIBezierPath()
  ```

- Move, add lines and arcs
  ```swift
  path.move(to: CGPoint(x: 40, y: 40))
  path.addLine(to: CGPoint(x: 80, y: 40))
  path.addLine(to: CGPoint(x: 80, y: 80))
  path.addLine(to: CGPoint(x: 40, y: 80))
  ```

- Close (optional)
  ```swift
  path.close()
  ```

- Set attributes
  ```swift
  UIColor.blue.setStroke()
  UIColor.red.setFill()
  path.lineWidth = 3
  ```

- Fill/Stroke
  ```swift
  path.stroke()
  path.fill()
  ```
SHAPES

- Add curves:
  ```swift
  path.addCurve(to: pt1, controlPoint1: pt2, controlPoint2: pt3)
  path.addQuadCurve(to: pt1, controlPoint: pt2)
  ```

- Add common shapes:
  ```swift
  path.append(UIBezierPath(ovalIn: r))
  path.append(UIBezierPath(rect: r))
  path.append(UIBezierPath(roundedRect: r, cornerRadius: 0))
  path.append(UIBezierPath(roundedRect: r, cornerRadius: r.width / 2.0))
  ```

- `.intersects()`, etc.