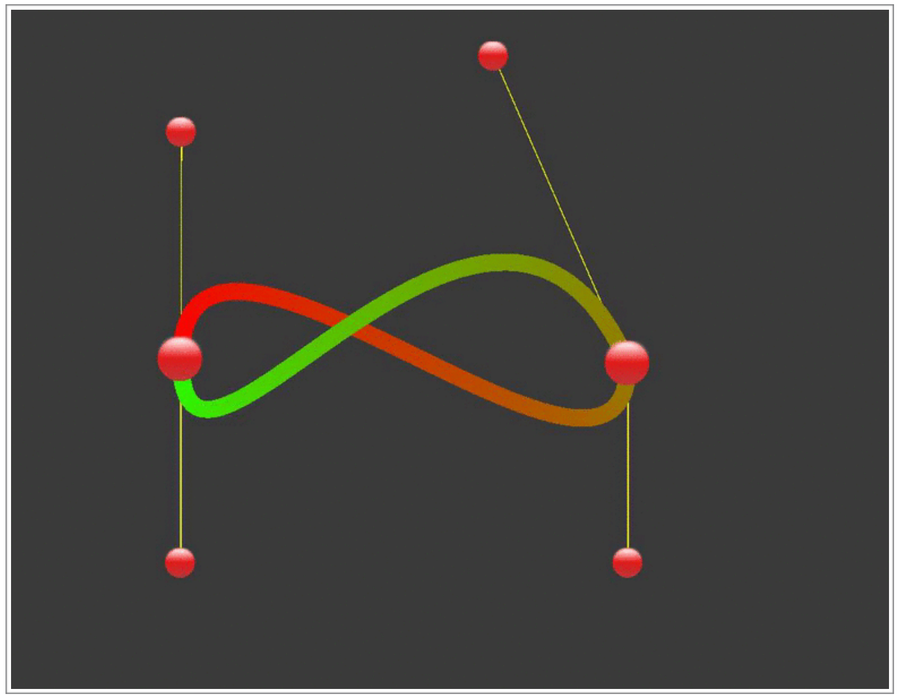
ADVANCED DRAWING TECHNIQUES WITH UIBEZIERPATH AND APPLE PENCIL



Nick Dalton

360iDev - August 15, 2017

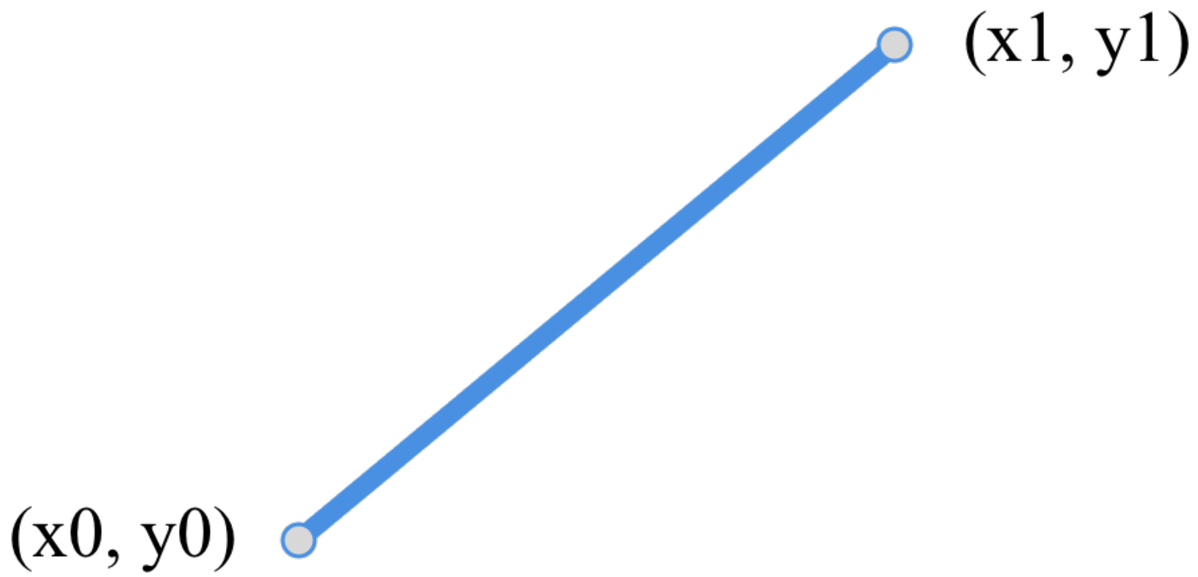
360|iDev

VECTOR GRAPHICS

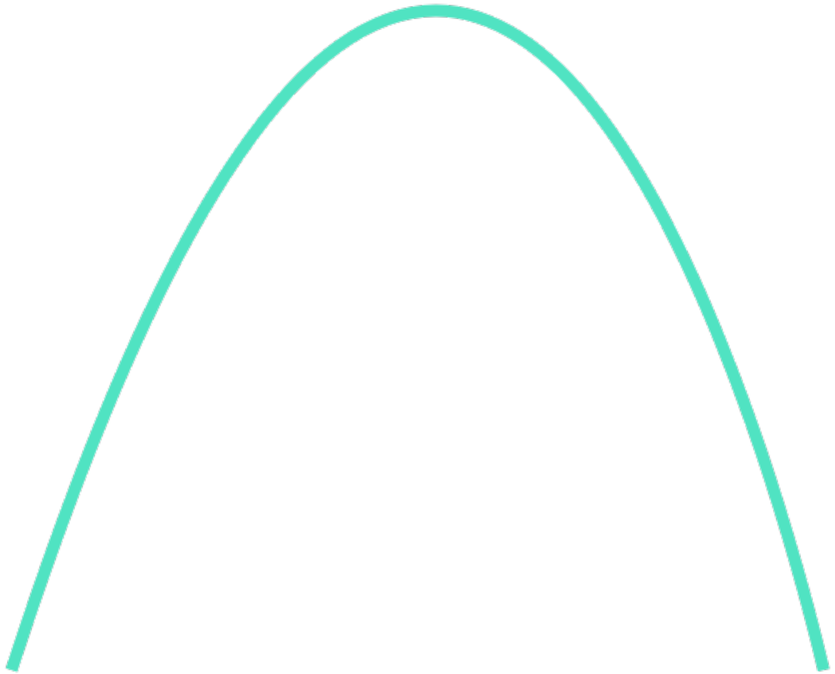
- Scalable
- Compact representation

VECTOR GRAPHICS

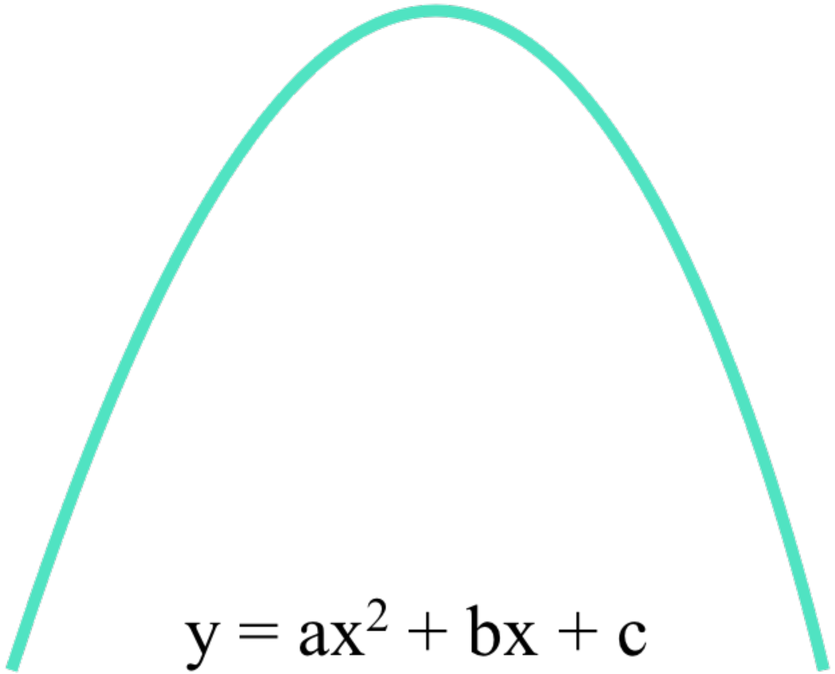
- Scalable
- Compact representation



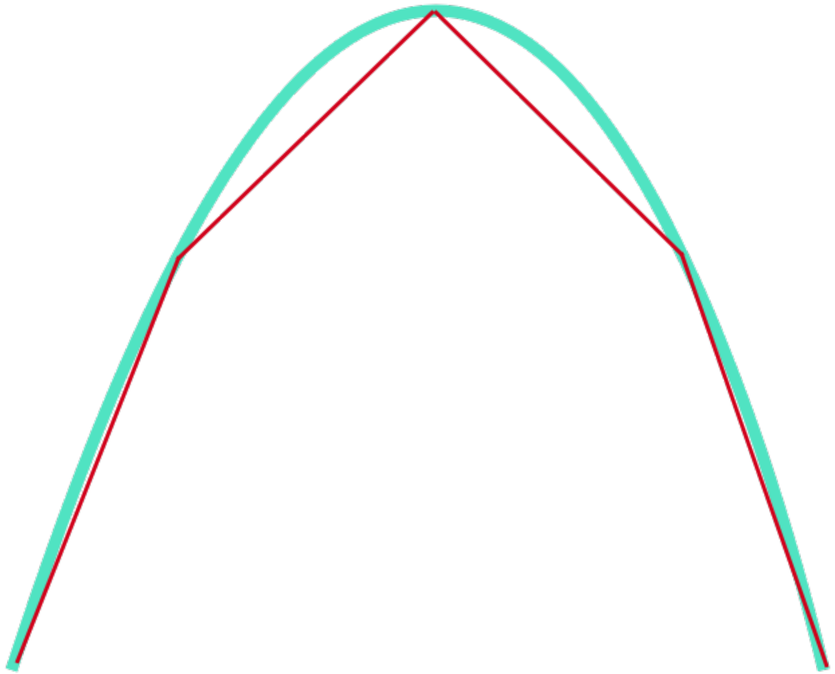
VECTOR CURVES



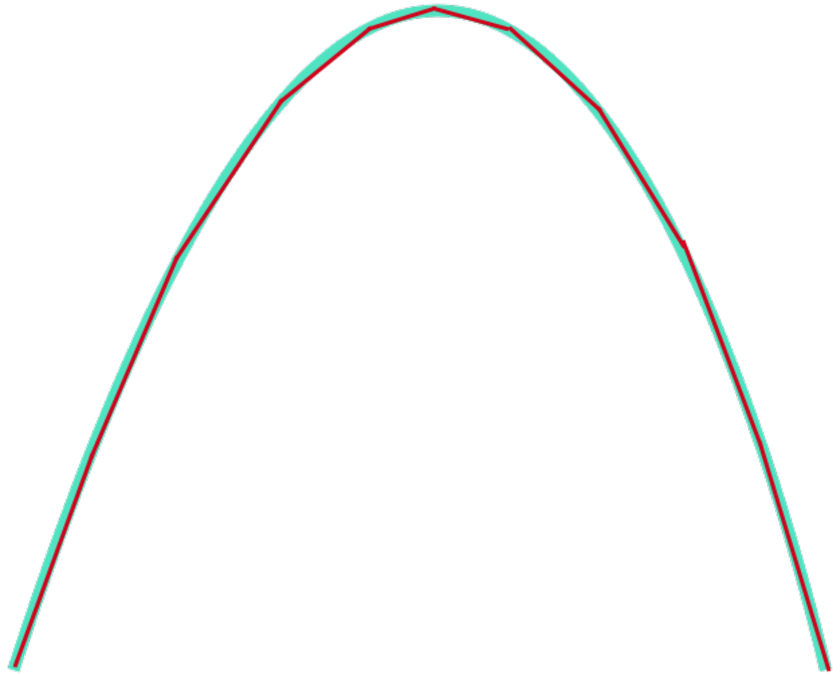
VECTOR CURVES



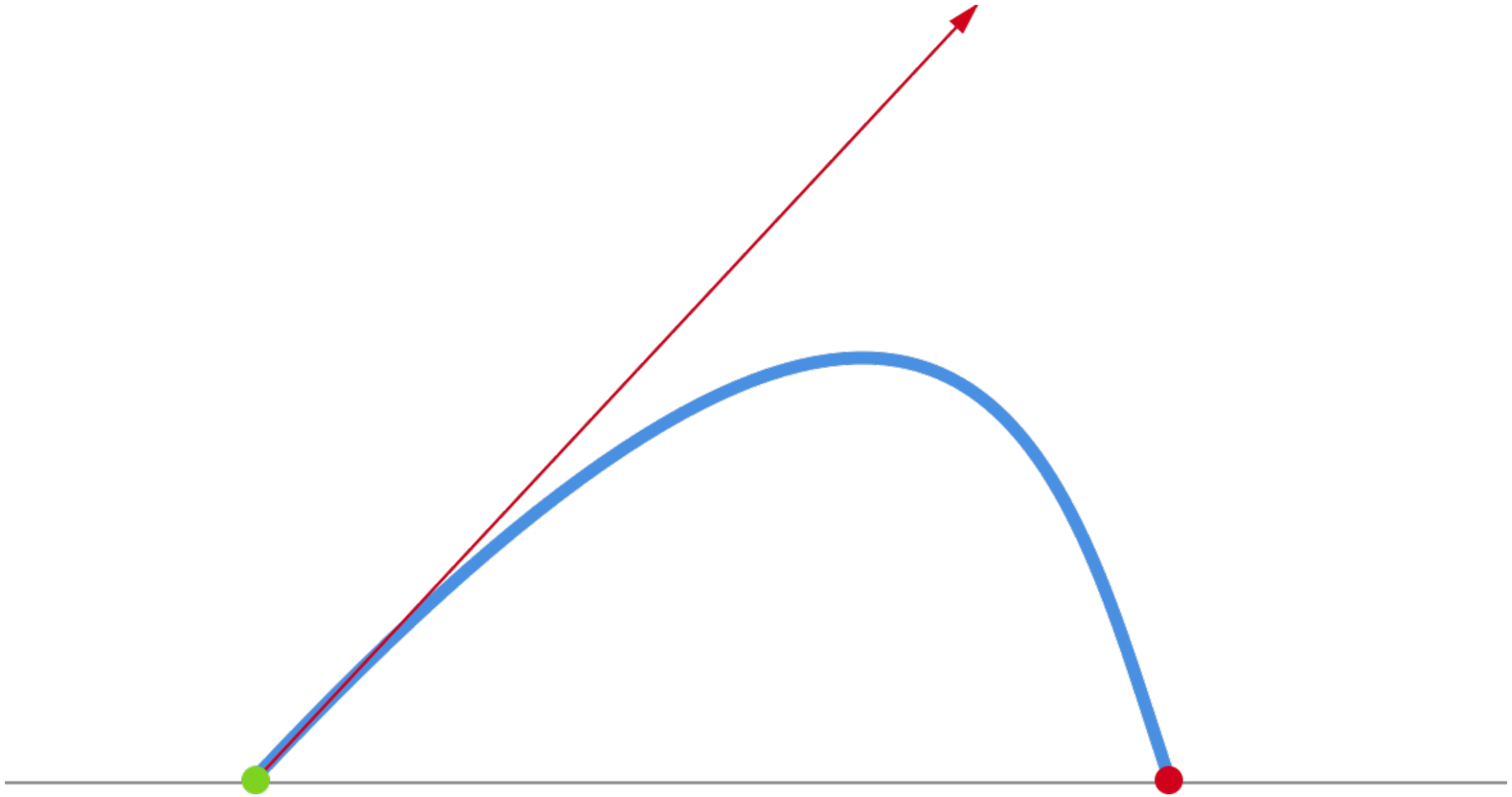
VECTOR CURVES



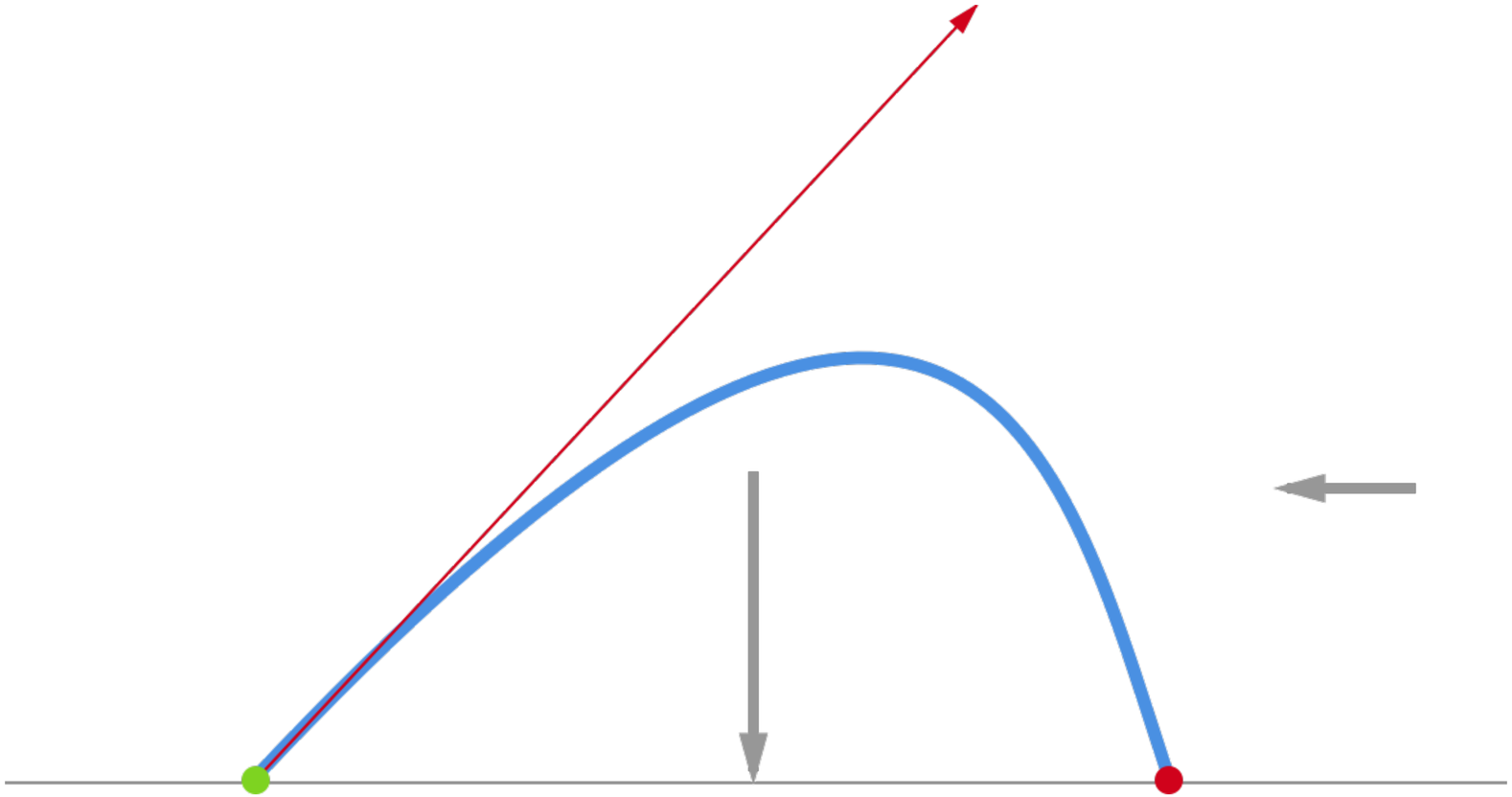
VECTOR CURVES



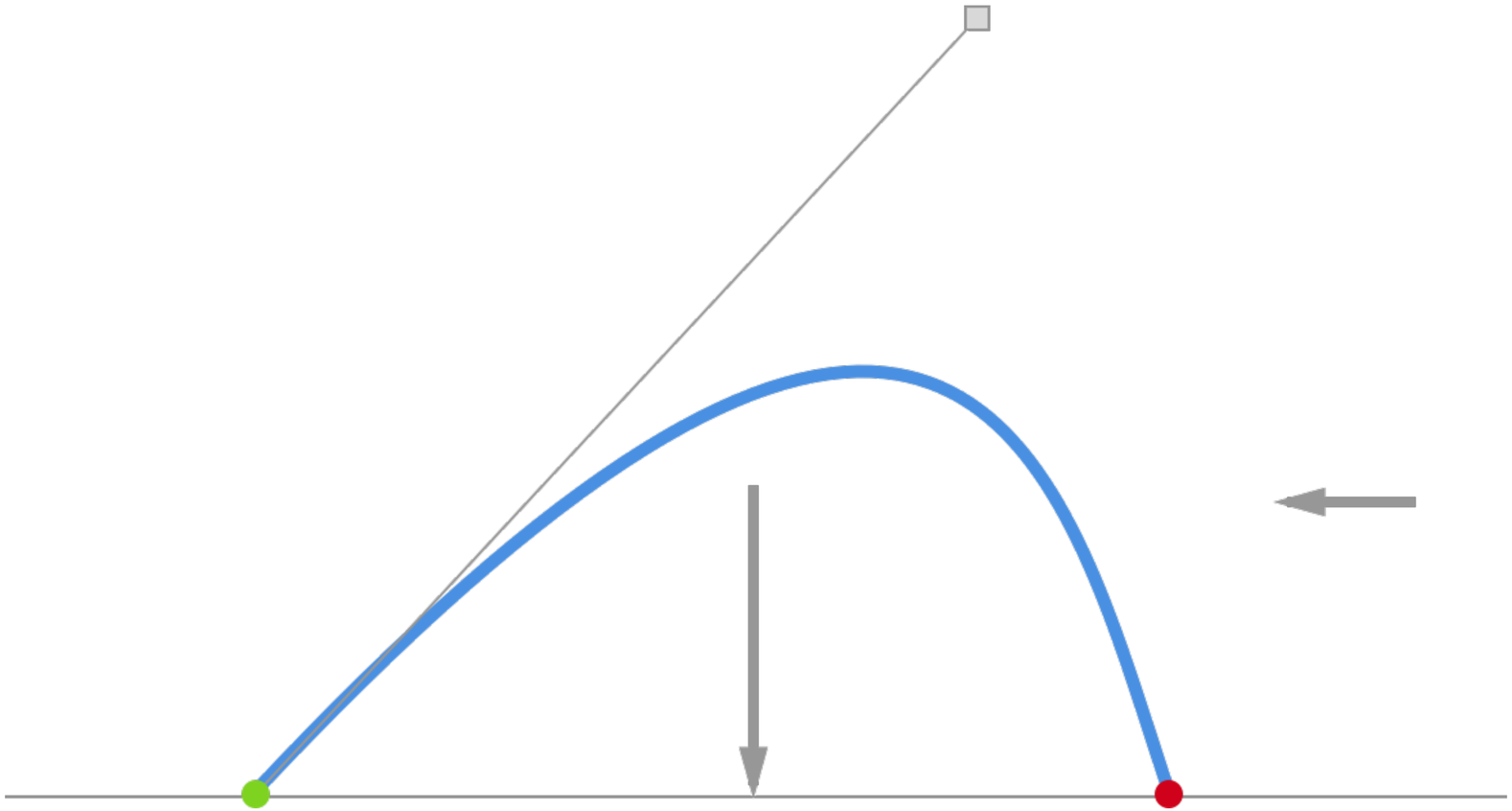
BÉZIER BASICS



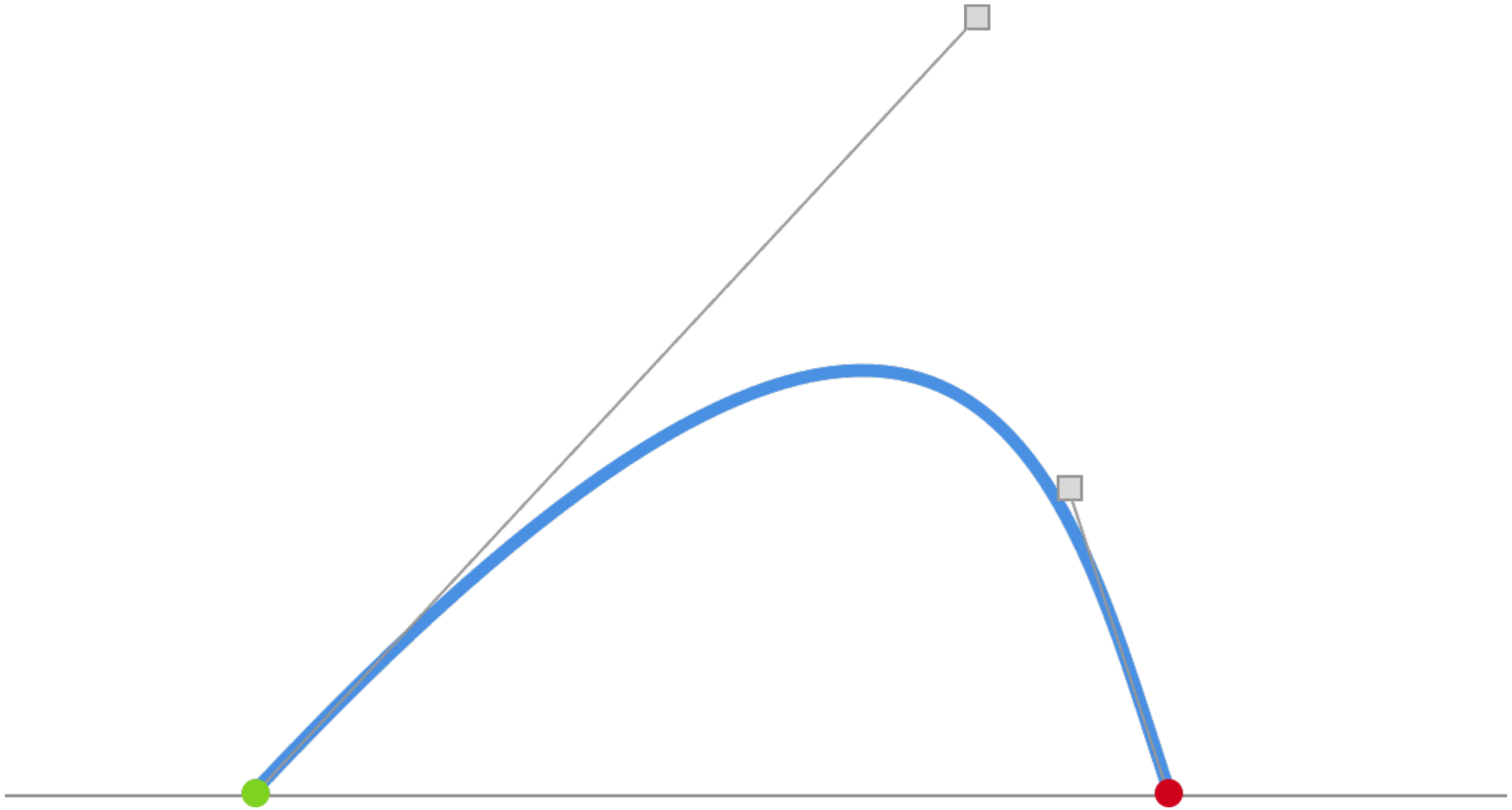
BÉZIER BASICS



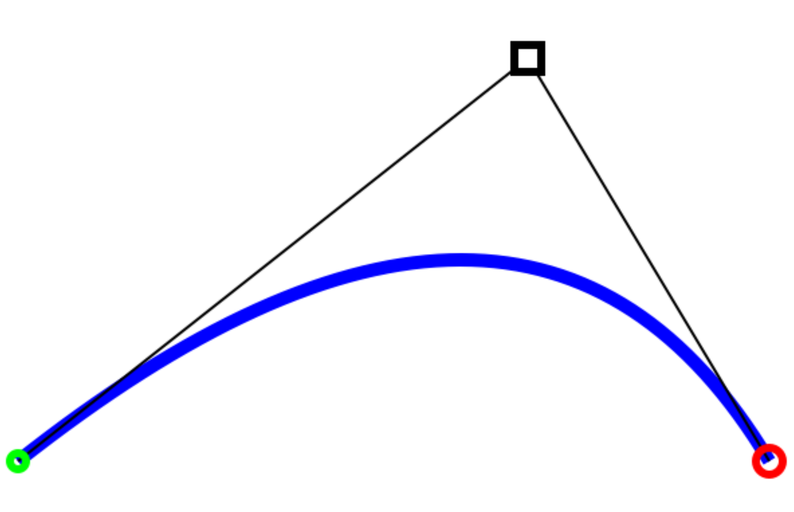
BÉZIER BASICS



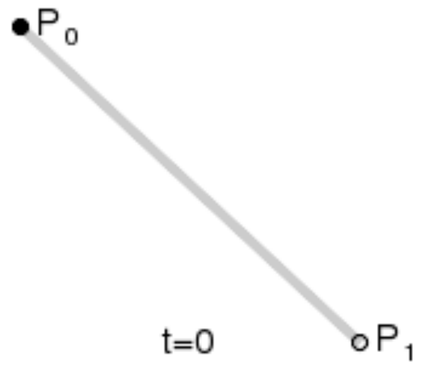
BÉZIER BASICS



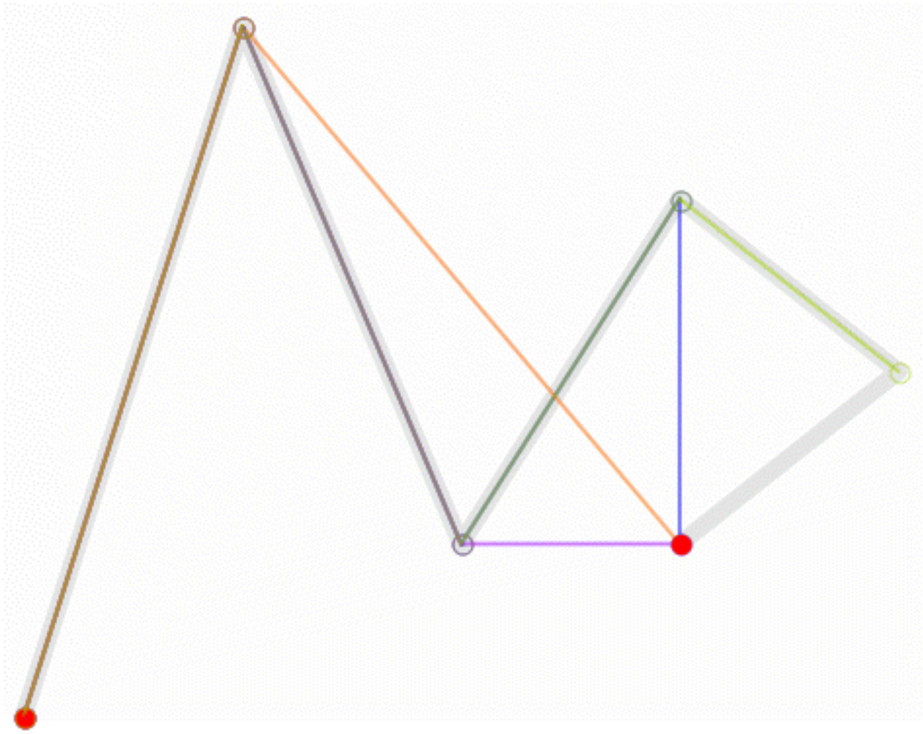
QUADRATIC BÉZIER CURVE



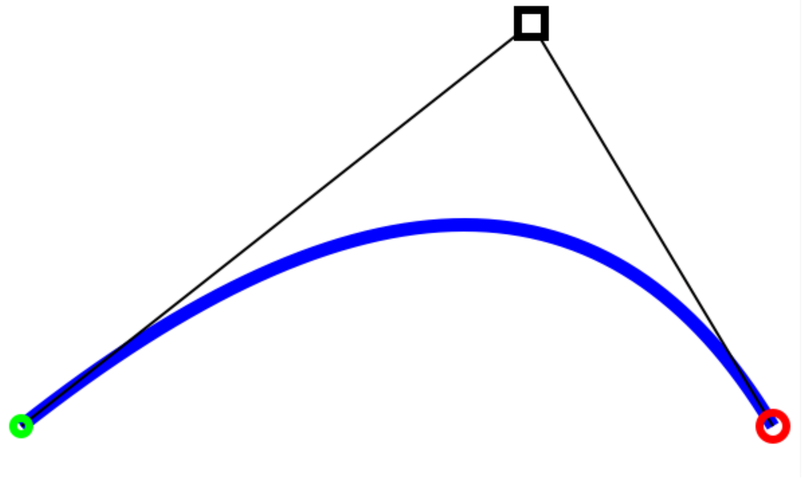
LINEAR CURVE



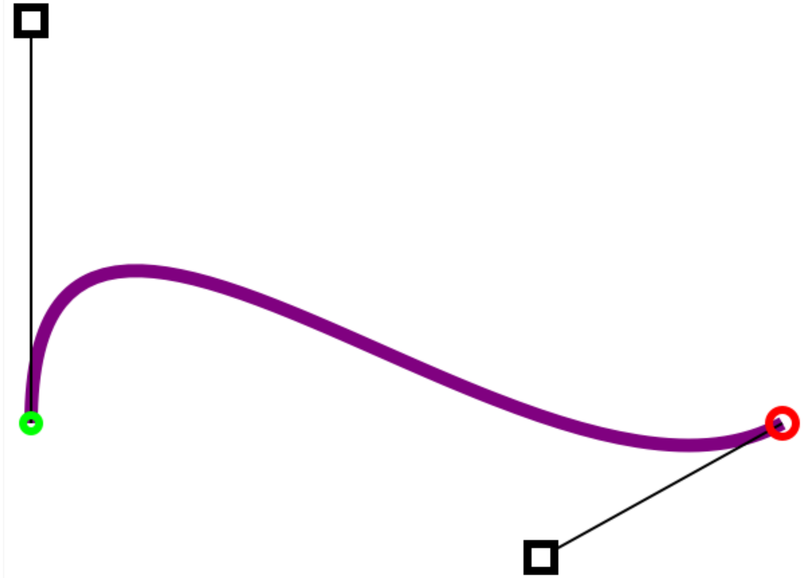
HIGHER ORDER BÉZIER CURVES



MOST COMMON BÉZIER CURVES



Quadratic



Cubic

MORE BÉZIER RESOURCES

A Primer on Bézier Curves

A free, online book for when you really need to know how to do Bézier things.

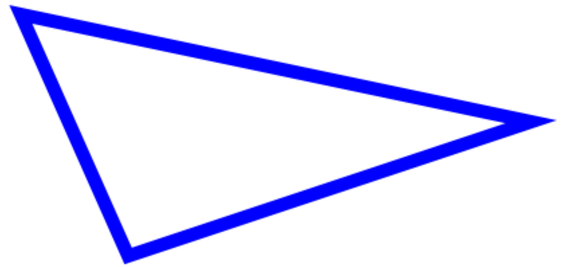
Read this in your own language: English - 日本語 - 中文

Don't see your language listed? Help translate this content!

https://pomax.github.io/bezierinfo/

UIBEZIERPATH - LINES

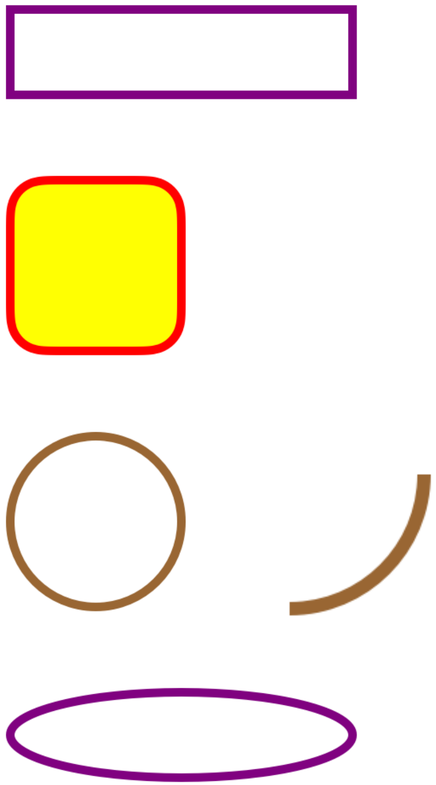
```
// Straight lines
let linesPath = UIBezierPath()
linesPath.move(to: CGPoint(x: 10, y: 10))
linesPath.addLine(to: CGPoint(x: 50, y: 100))
linesPath.addLine(to: CGPoint(x: 200, y: 50))
linesPath.close()
```



UIBEZIERPATH - SHAPES

Convenience initializers:

- Rectangle
- Rounded rectangle
- Circle
- Arc
- Oval



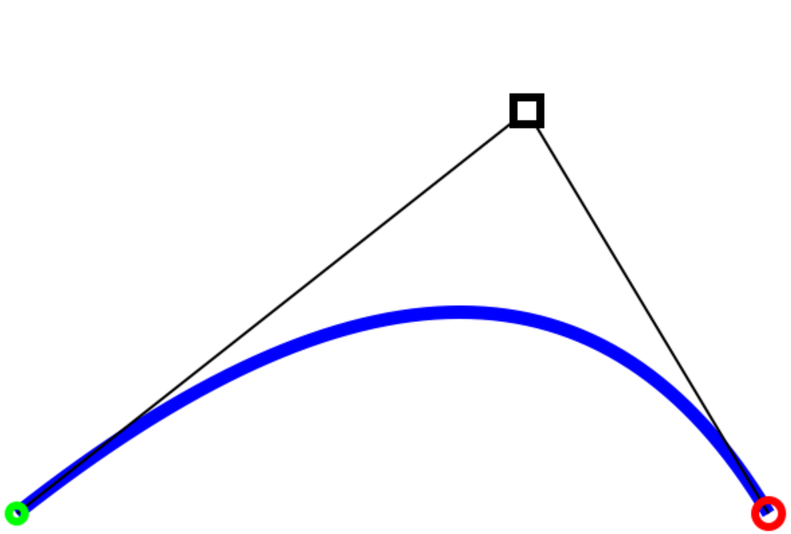
UIBEZIERPATH - DASH

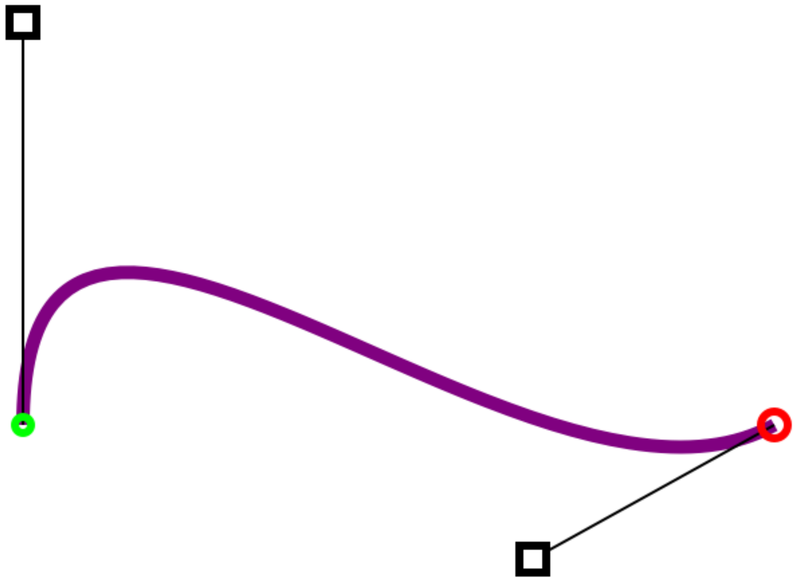
```
// Dashed line
let lineDashPath1 = UIBezierPath()
lineDashPath1.move(to: CGPoint(x: 10, y: 650))
lineDashPath1.addLine(to: CGPoint(x: 250, y: 650))
lineDashPath1.lineWidth = 5.0

let linePattern1: [CGFloat] = [20.0, 10.0]
lineDashPath1.setLineDash(linePattern1,
  count: linePattern1.count, phase: 0.0)
```

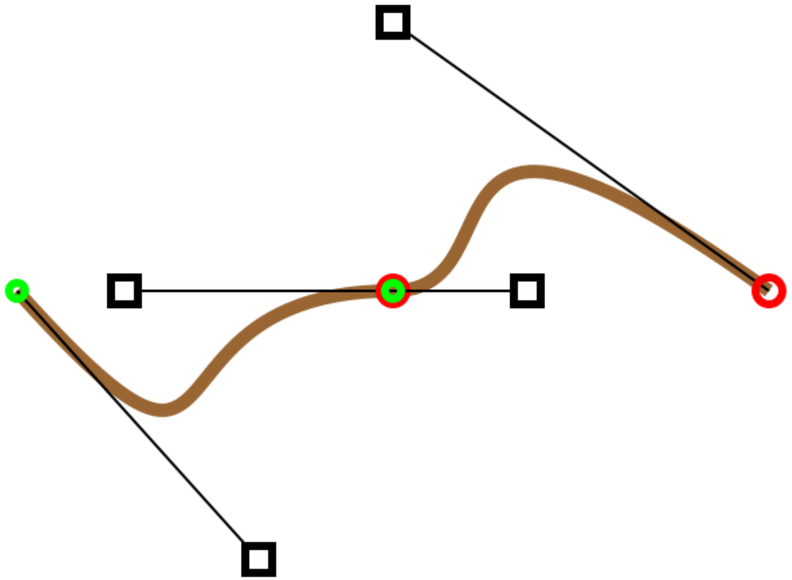
UIBEZIERPATH - DASH

```
// Dotted line
let lineDashPath1 = UIBezierPath()
lineDashPath1.move(to: CGPoint(x: 10, y: 650))
lineDashPath1.addLine(to: CGPoint(x: 250, y: 650))
lineDashPath1.lineWidth = 5.0

let linePattern2: [CGFloat] = [0.0, 10.0]
lineDashPath1.setLineDash(linePattern2,
  count: linePattern2.count, phase: 0.0)
lineDashPath1.lineCapStyle = .round
```

UIBEZIERPATH - CURVES







Live Playgrounds 😱

NSBezierPath

- NSBezierPath expects angles in degrees, UIBezierPath uses radians.
- NSBezierPath has methods to access the elements of the path:
  - elementCount
  - element(at: Int)

NSBezierPath

```
extension UIBezierPath {
    var elements: [PathElement] {
        var pathElements = [PathElement]()
        withUnsafeMutablePointer(to: &pathElements) { elementsPointer in
            let rawElementsPointer = UnsafeMutableRawPointer(elementsPointer)
            cgPath.apply(info: rawElementsPointer) { userInfo, nextElementPointer in
                let nextElement = PathElement(element: nextElementPointer.pointee)
                let elementsPointer = userInfo?.assumingMemoryBound(to: [PathElement].self)
                elementsPointer?.pointee.append(nextElement)
            }
        }
        return pathElements
    }
}
```

https://oleb.net/blog/2015/06/c-callbacks-in-swift/

PRACTICAL APPLICATION: A SIMPLE DRAWING APP

TOUCHESBEGAN

```
override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {
    super.touchesBegan(touches, with: event)

    guard let touch = touches.first else {
      return
    }

    let point = touch.location(in: self)
    newPath().move(to: point)
}
```

TOUCHESMOVED

```
override func touchesMoved(_ touches: Set<UITouch>, with event:
UIEvent?) {
    super.touchesMoved(touches, with: event)

    guard let touch = touches.first else {
      return
    }

    // Add straight line
    let point = touch.location(in: self)
    currentPath()?.addLine(to: point)

    // Cause drawRect to be called
    setNeedsDisplay()
}
```

DRAWRECT

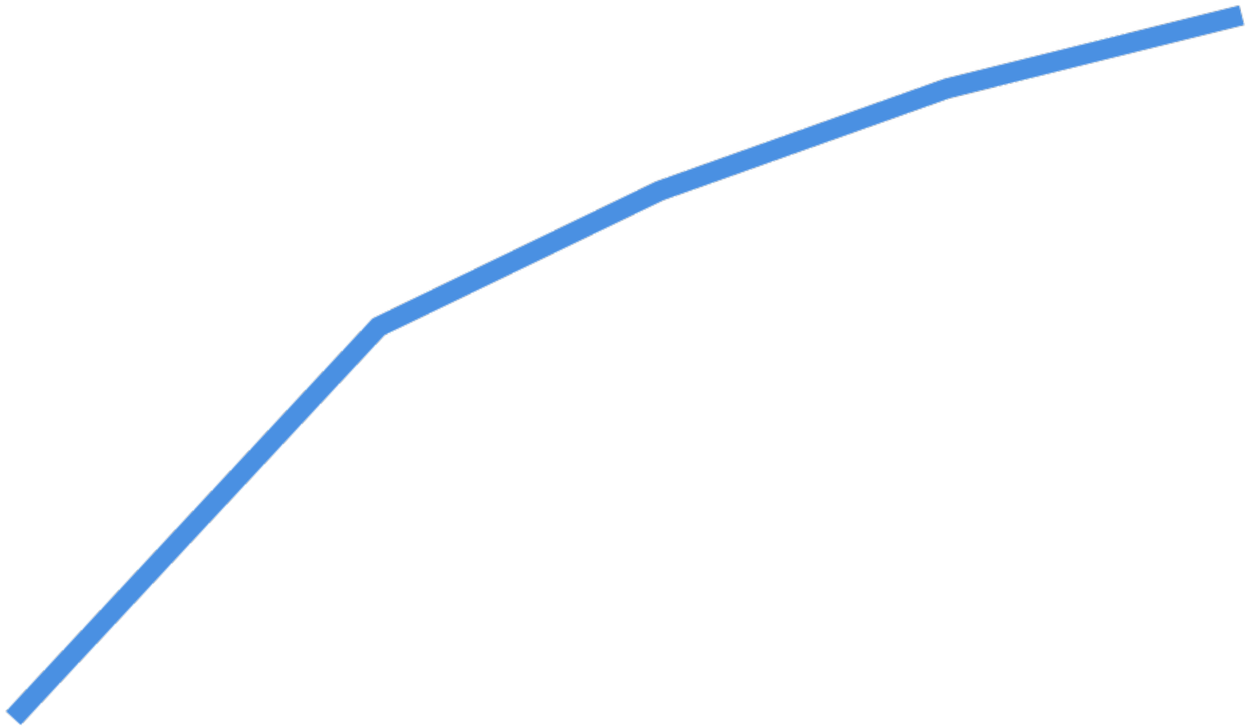
```
override func draw(_ rect: CGRect) {
    // Set stroke color
    UIColor.blue.setStroke()

    // Draw each path
    for path in self.paths {
      path.stroke()
    }
}
```

PERFORMANCE PROBLEMS

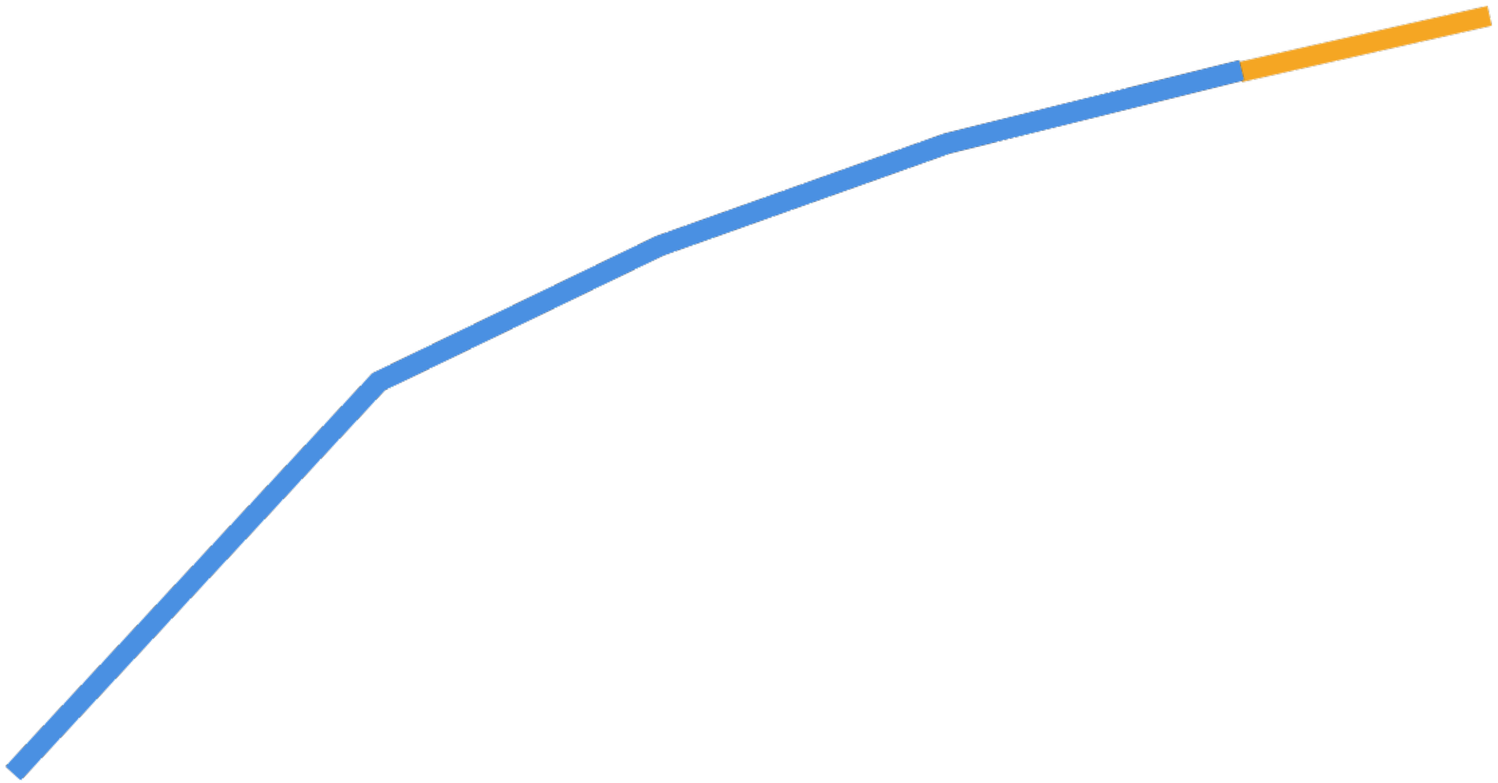
- Current implementation is very naïve.
- This is a problem in touchesMoved:

```
// Cause drawRect to be called
setNeedsDisplay()
```

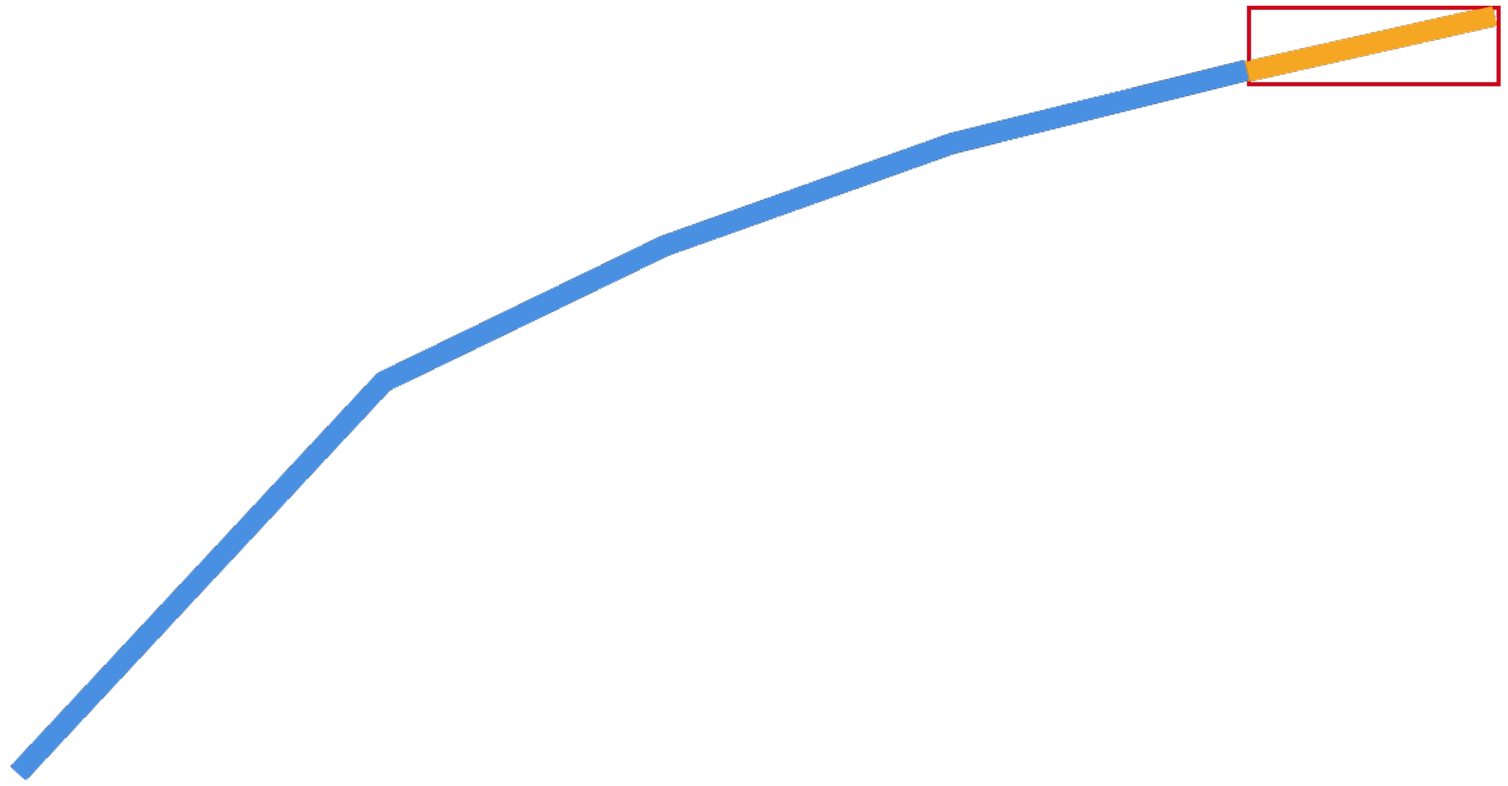
PERFORMANCE PROBLEMS



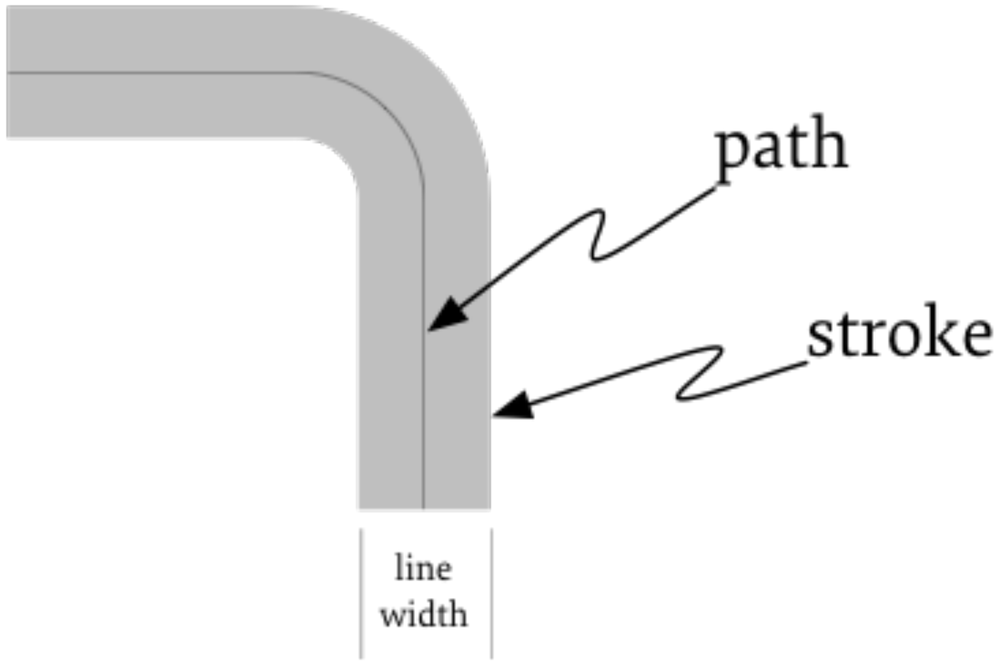
PERFORMANCE PROBLEMS



PERFORMANCE PROBLEMS



PATH VS. STROKE



TOUCHESMOVED

```
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
    ...

    let point = touch.location(in: self)

    // Calculate rect to refresh
    var refreshRect = CGRect.zero
    if let previousPoint = currentPath()?.currentPoint,
       let lineWidth = currentPath()?.lineWidth {
      refreshRect = CGRect(p1: previousPoint, p2: point)
                    .insetBy(dx: -lineWidth, dy: -lineWidth)
    }

    // Add straight line
    currentPath()?.addLine(to: point)

    // Cause drawRect to be called
    setNeedsDisplay(refreshRect)
}
```

DRAWING DEMO

Live iPad Pro Demo 😱

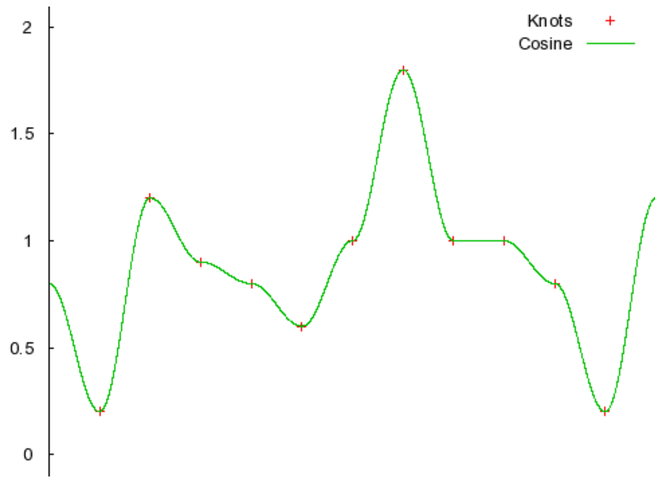
BRING THE CURVES

Drawing straight line paths is not very pretty:

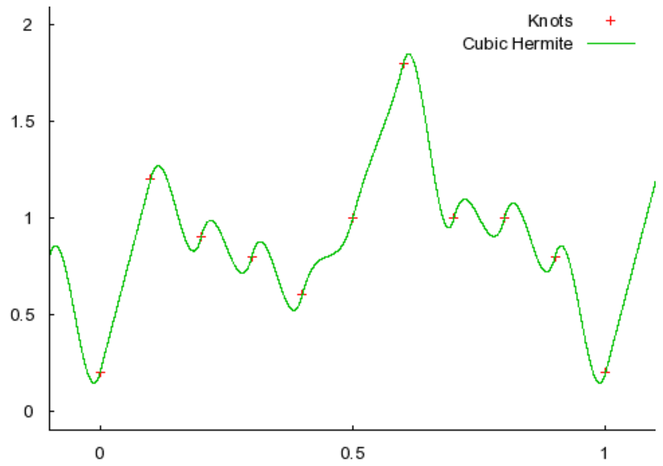
A LINE SMOOTHING ALGORITHM

Back to the Playground

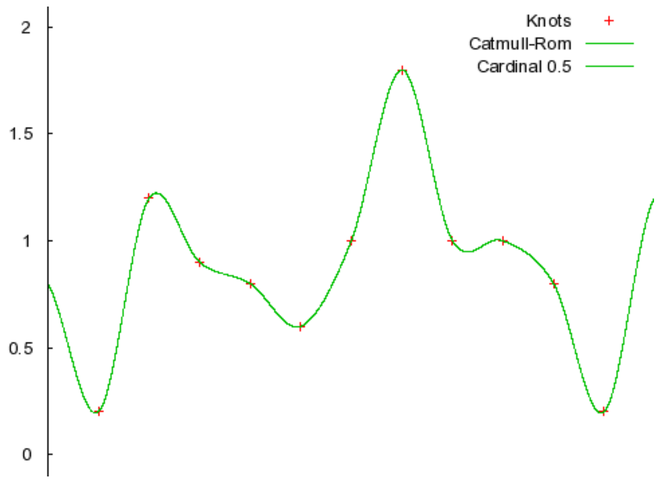
OTHER SMOOTHING ALGORITHMS



Cosine Interpolation



Cubic Hermite Splines

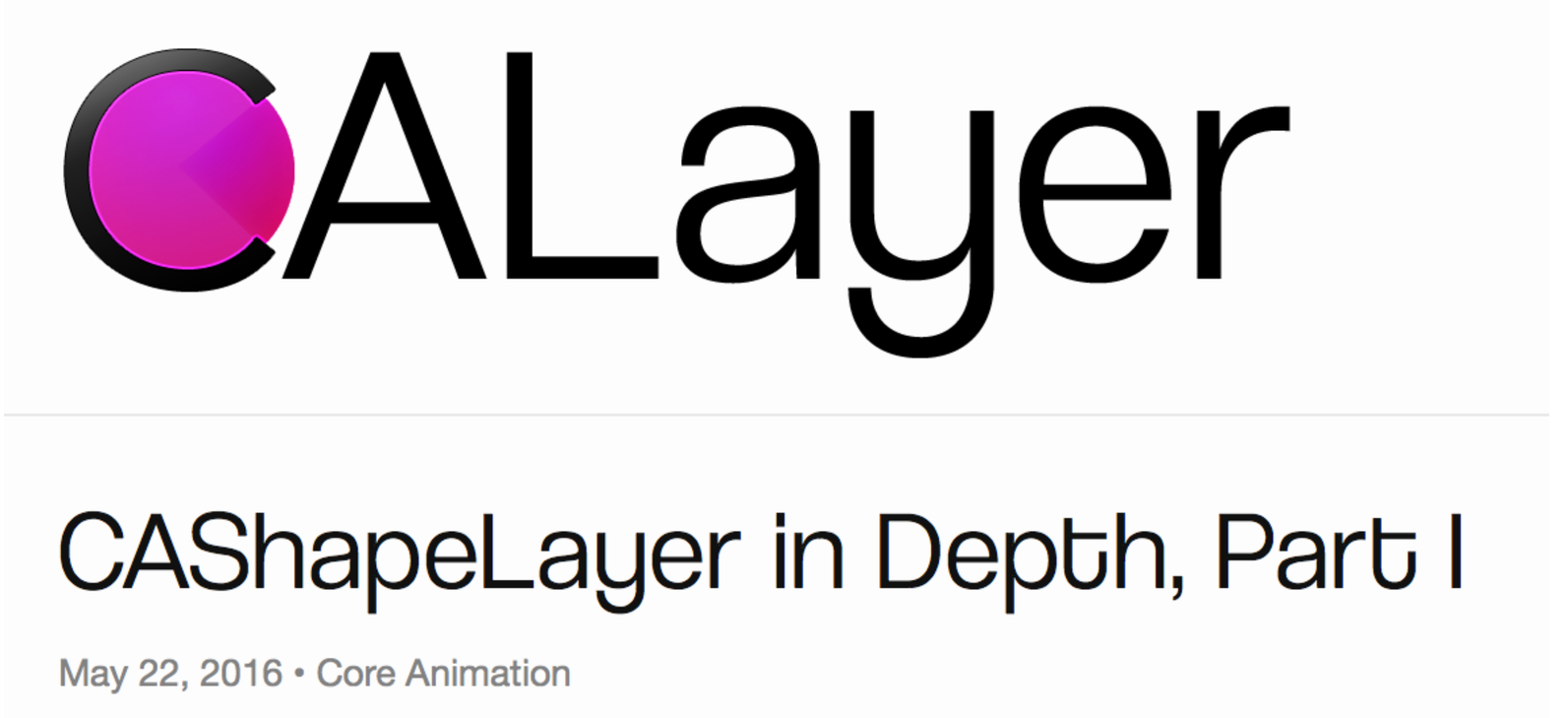


Catmull-Rom Splines

CAShapeLayer

```
let roundedRectPath = UIBezierPath(
  roundedRect: CGRect(x: 10, y: 10, width: 100, height: 100),
  cornerRadius: 20.0)

let shapeLayer1 = CAShapeLayer()
shapeLayer1.path = roundedRectPath.cgPath
shapeLayer1.lineWidth = 5.0
shapeLayer1.fillColor = UIColor.yellow.cgColor
shapeLayer1.strokeColor = UIColor.blue.cgColor
myView.layer.addSublayer(shapeLayer1)
```

CASHAPELAYER - RESOURCES



http://calayer.com/core-animation/2016/05/22/cashapelayer-in-depth.html

APPLE PENCIL TRICKS

- Detect stylus
- Coalesced touches
- Predicted touches
- Altitude and azimuth angles
- Force

DETECT APPLE PENCIL

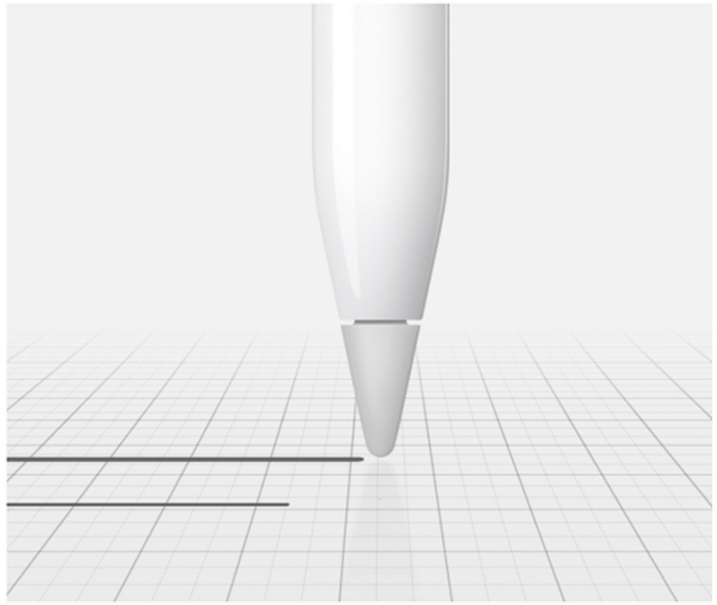
- There is no (official) way to detect if an Apple Pencil is paired with the iPad.
- But for each UITouch:

```
if touch.type == .stylus
```

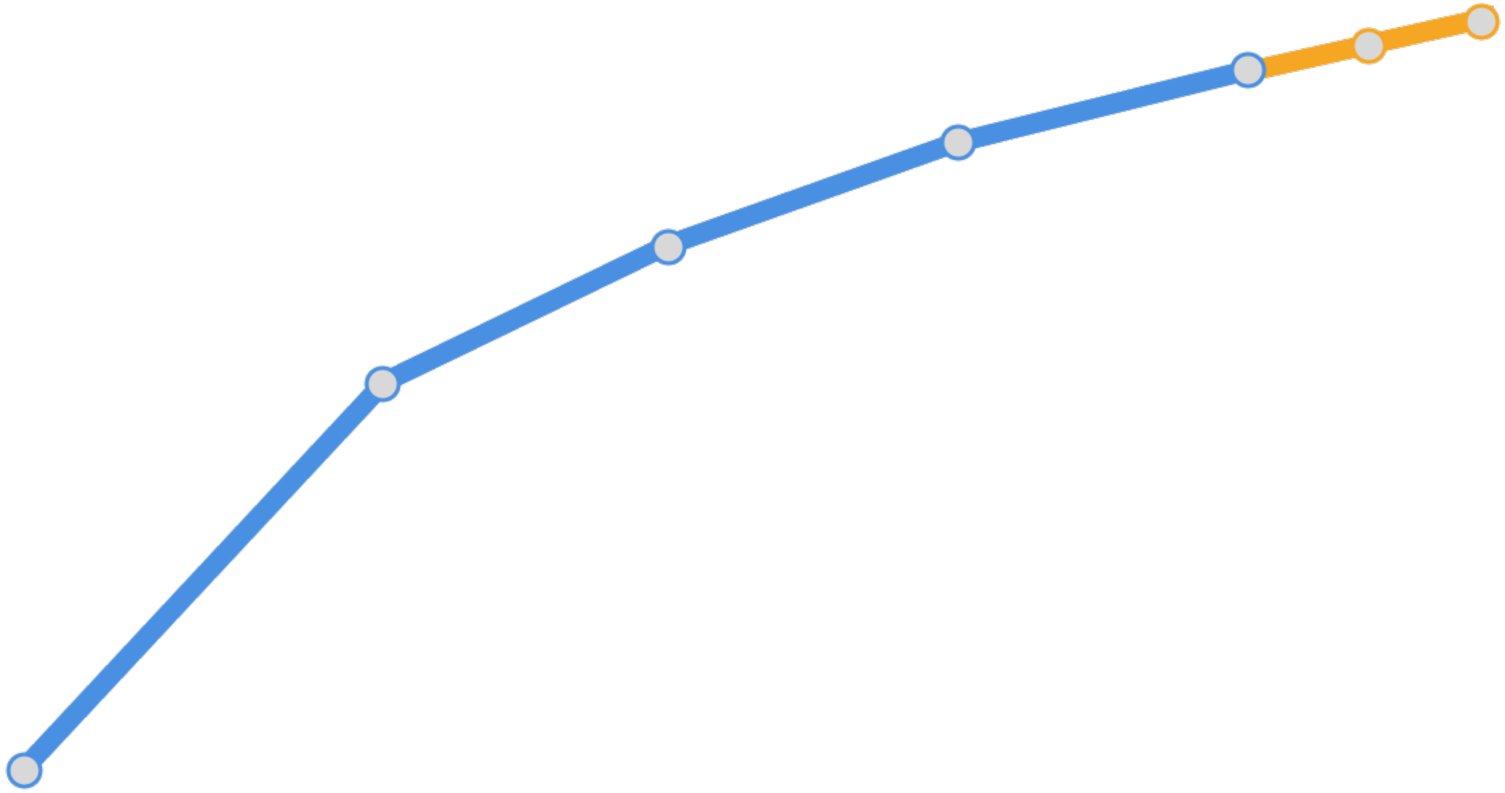
COALESCED TOUCHES

```
override func touchesMoved(_ touches: Set<UITouch>, with event:
UIEvent?) {

    ...

    // Gather coalesced touches if available (iPad Pro only)
    var allTouches = [UITouch]()
    if let coalescedTouches = event?.coalescedTouches(for: touch) {
        allTouches.append(contentsOf: coalescedTouches)
    } else {
        // Just add the single touch point
        allTouches.append(touch)
    }
```

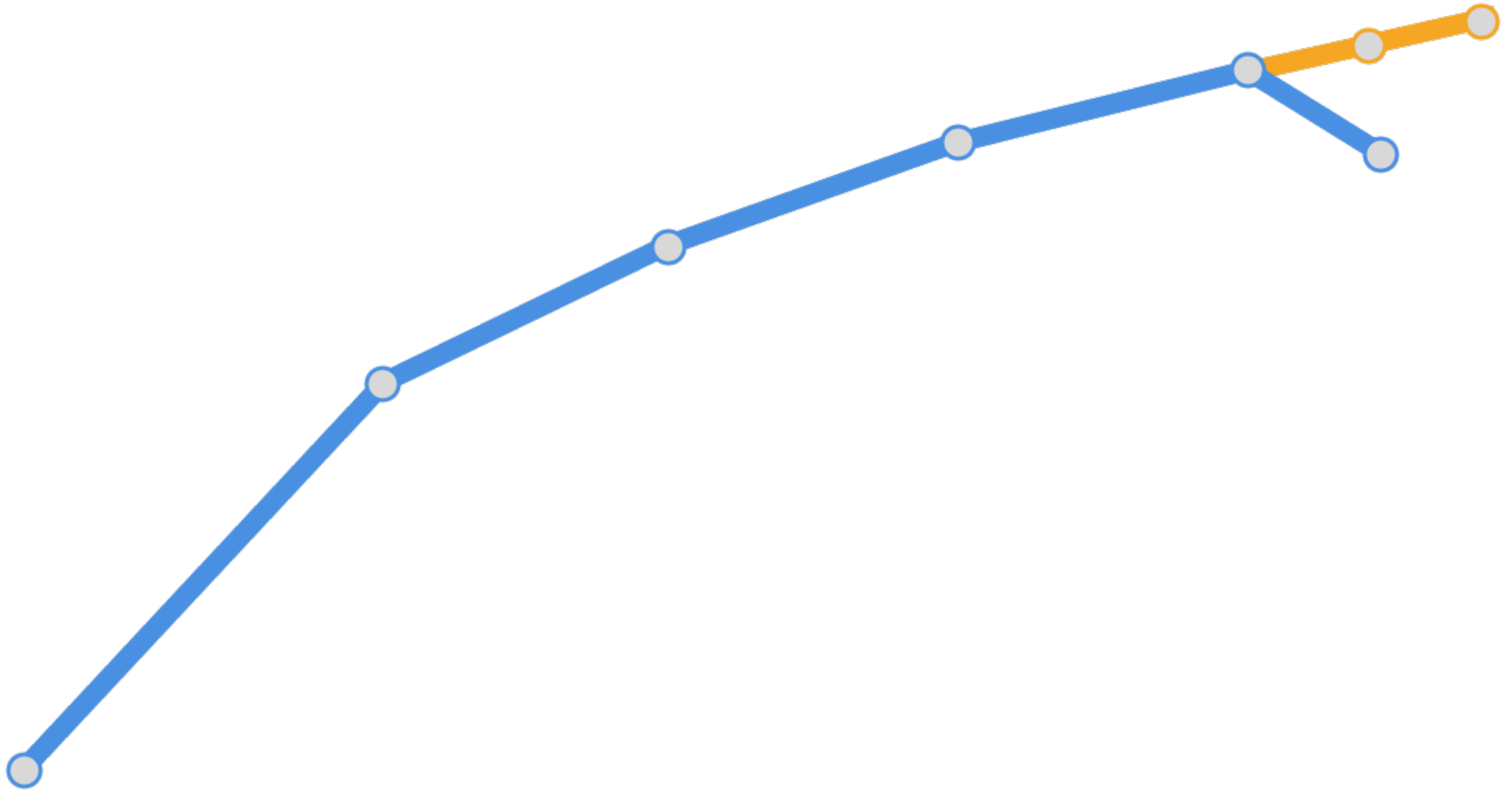
PREDICTED TOUCHES



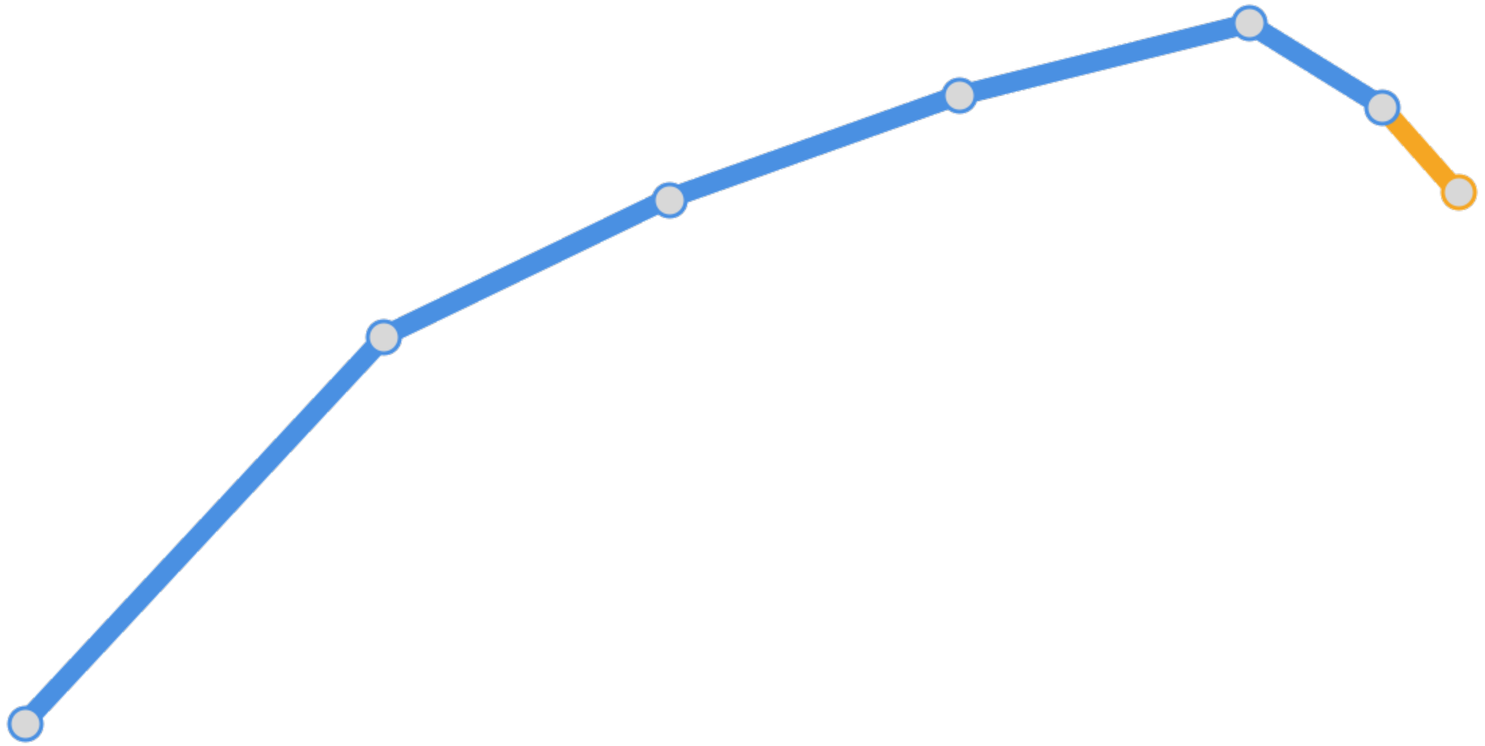
PREDICTED TOUCHES



PREDICTED TOUCHES



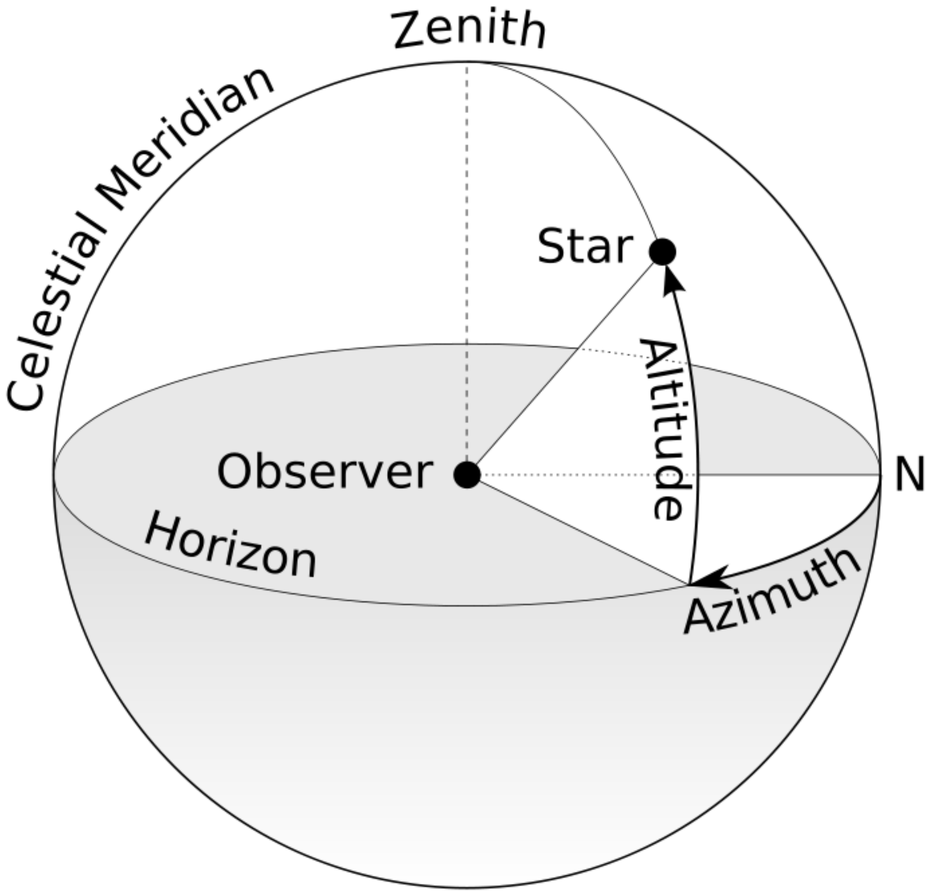
PREDICTED TOUCHES



PREDICTED TOUCHES

```
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {

    ...

    // Draw predicted path if available (iPad Pro only)
    if let predictedTouches = event?.predictedTouches(for: touch),
        predictedTouches.count > 0 {

      let predictedPath = UIBezierPath()
      predictedPath.move(to: touch.location(in: self))

      for aTouch in predictedTouches {
        let point = aTouch.location(in: self)
        predictedPath.addLine(to: point)
      }

      self.predictedPath = predictedPath
    } else {
      self.predictedPath = nil
    }
```

ALTITUDE AND AZIMUTH



ALTITUDE ANGLE

```
// Adjust the line width using the stylus altitude angle
if touch.altitudeAngle < .pi / 2.0 {

   let altitude = (.pi / 2.0 - touch.altitudeAngle) / .pi / 2.0
   let altitudeLineWidth = altitude * altitudeAngleMultiplier
   self.lineWidth = altitudeLineWidth

} else {
   // No stylus
   self.lineWidth = self.userLineWidth
}
```

TOUCH FORCE

```
// Adjust the line width using the stylus touch force (Apple
Pencil only)
if touch.force > 0.0 {
   let forceLineWidth = touch.force * touchForceMultiplier
   self.currentPath()?.lineWidth = forceLineWidth
   self.lineWidth = forceLineWidth
   self.setNeedsDisplay()
} else {
   // No stylus
   self.lineWidth = self.userLineWidth
}
```

DEMO

- Sample drawing app
- ShadowDraw

THANK YOU!

Nick Dalton @TheAppCoach github.com/iNick