SCIENCES SORBONNE UNIVERSITÉ

A bit of drawing with Core Graphics

Fabrice.Kordon@lip6.fr

LIP6

About Core Graphics

- Core Graphics is quite old
- Principle
  - Code embedded in a view
- In Objective-C
  - Mostly C code
- In Swift
  - Nice binding to provide native primitives

About Core Graphics

Core Graphics is quite old

Principle

- Code embedd

In Objective

- Mostly C code

In Swift

- Nice binding to provide native primitives

Objective

Only an overview
(mostly Swift, a bit of Objective-C)

Draw a rectangle

Process

- Fetch the current context
- Add the rectangle to this context
- Request the path to be drawn
- Display

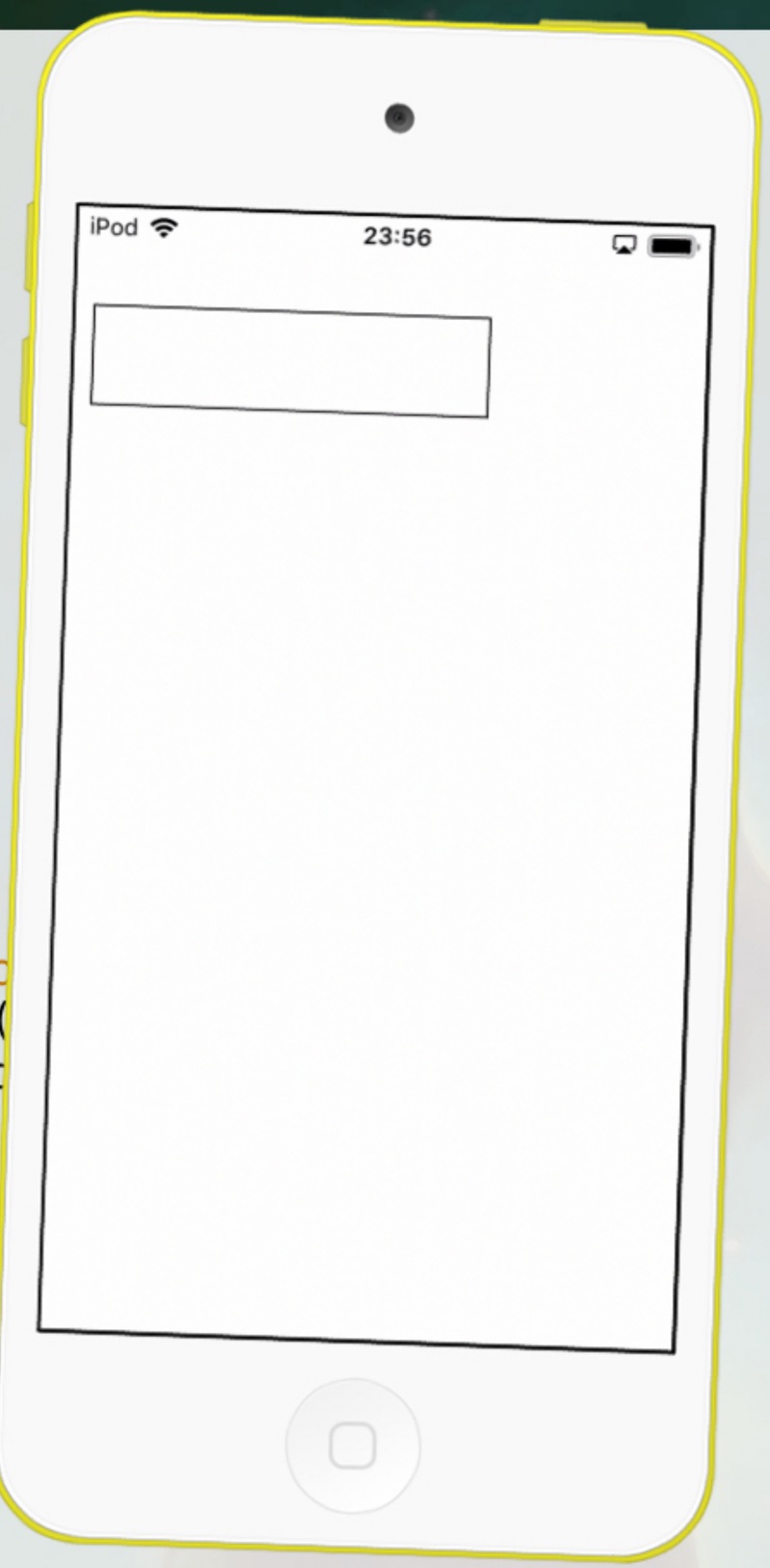
Example

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    let square = CGRect(origin: CGPoint(x: 10, y: 50),
                        size: CGSize(width: 200, height: 50))
    context?.addRect(square)
    context?.strokePath()
    // self.setNeedsDisplay()// useless because we are in draw
}
```

Draw a rectangle

Process

- Fetch the current context
- Add the rectangle to this context
- Request the path to be drawn
- Display



Example

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentCo
    let square = CGRect(origin: CGPoint(
                        size: CGSize(wid
    context?.addRect(square)
    context?.strokePath()
    // self.setNeedsDisplay()// useless
}
```

Draw a circle

Process (similar to the square)

- Fetch the current context
- Add an arc to this context
- Request the path to be drawn
- Display

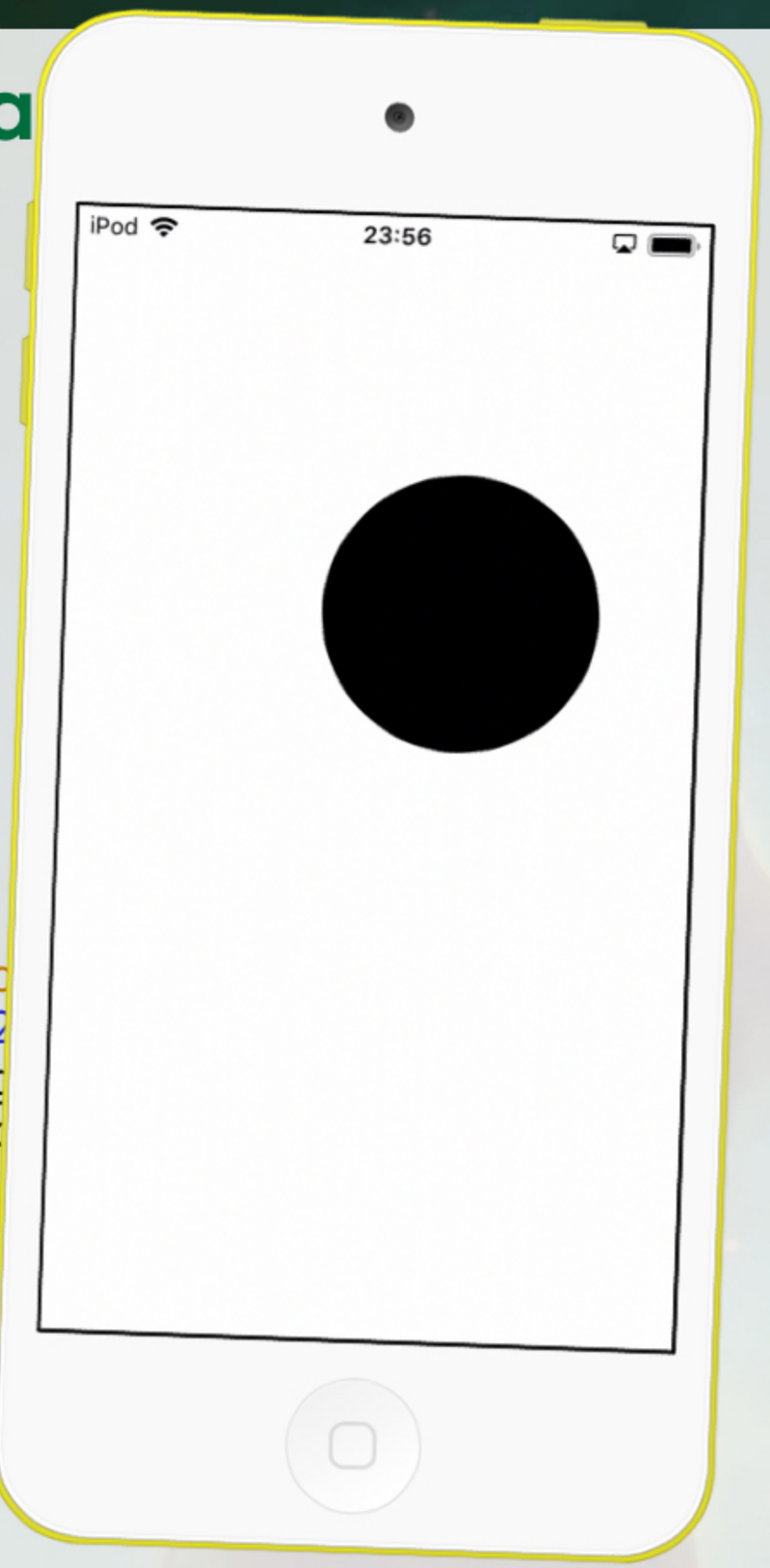
Example

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.addArc(center: CGPoint(x: 200, y: 200),
                    radius: 70, startAngle: 0,
                    endAngle: 2 * .pi, clockwise: true)
    context?.fillPath()
    // self.setNeedsDisplay()// useless because we are in draw
}
```

Draw a circle

Process (similar to the squa

- Fetch the current context
- Add an arc to this context
- Request the path to be drawn
- Display



Example

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentCo
    context?.addArc(center: CGPoint(x: 2
                    radius: 70, startAng
                    endAngle: 2 * .pi, c
    context?.fillPath()
    // self.setNeedsDisplay()// useless
}
```

Draw a polygon

Principle

- Principle does not change…
- Here with a path contracted line by line

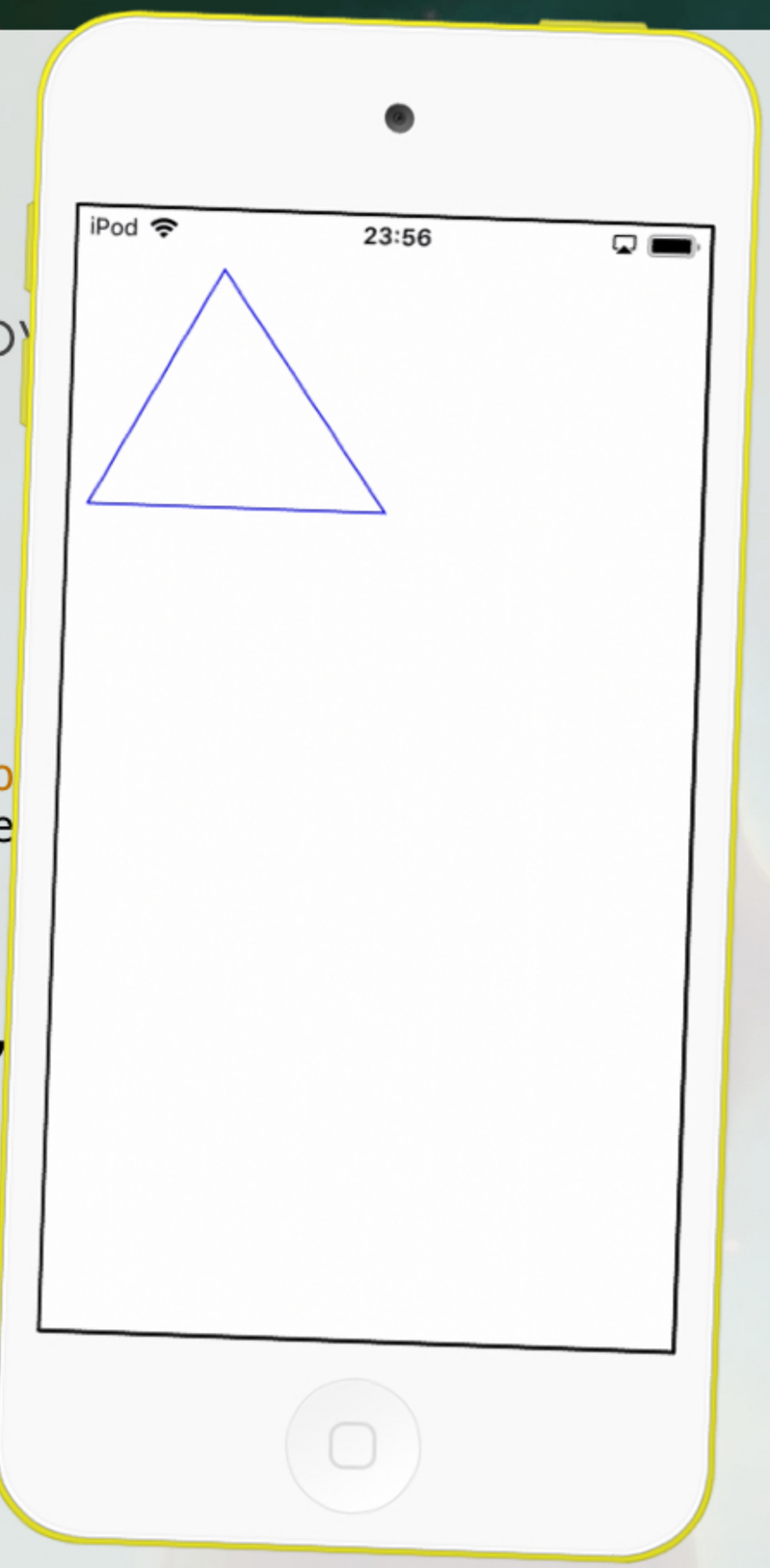
Example

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.setStrokeColor(red: 0, green: 0, blue: 1, alpha: 1)
    context?.beginPath()
    context?.move(to: CGPoint(x: 75, y: 30))
    context?.addLine(to: CGPoint(x: 10, y: 150))
    context?.addLine(to: CGPoint(x: 160, y: 150))
    context?.closePath()
    context?.strokePath()
    // self.setNeedsDisplay()// useless because we are in draw
}
```

Draw a polygon

Principle

- Principle does not change…
- Here with a path contracted line by



Example

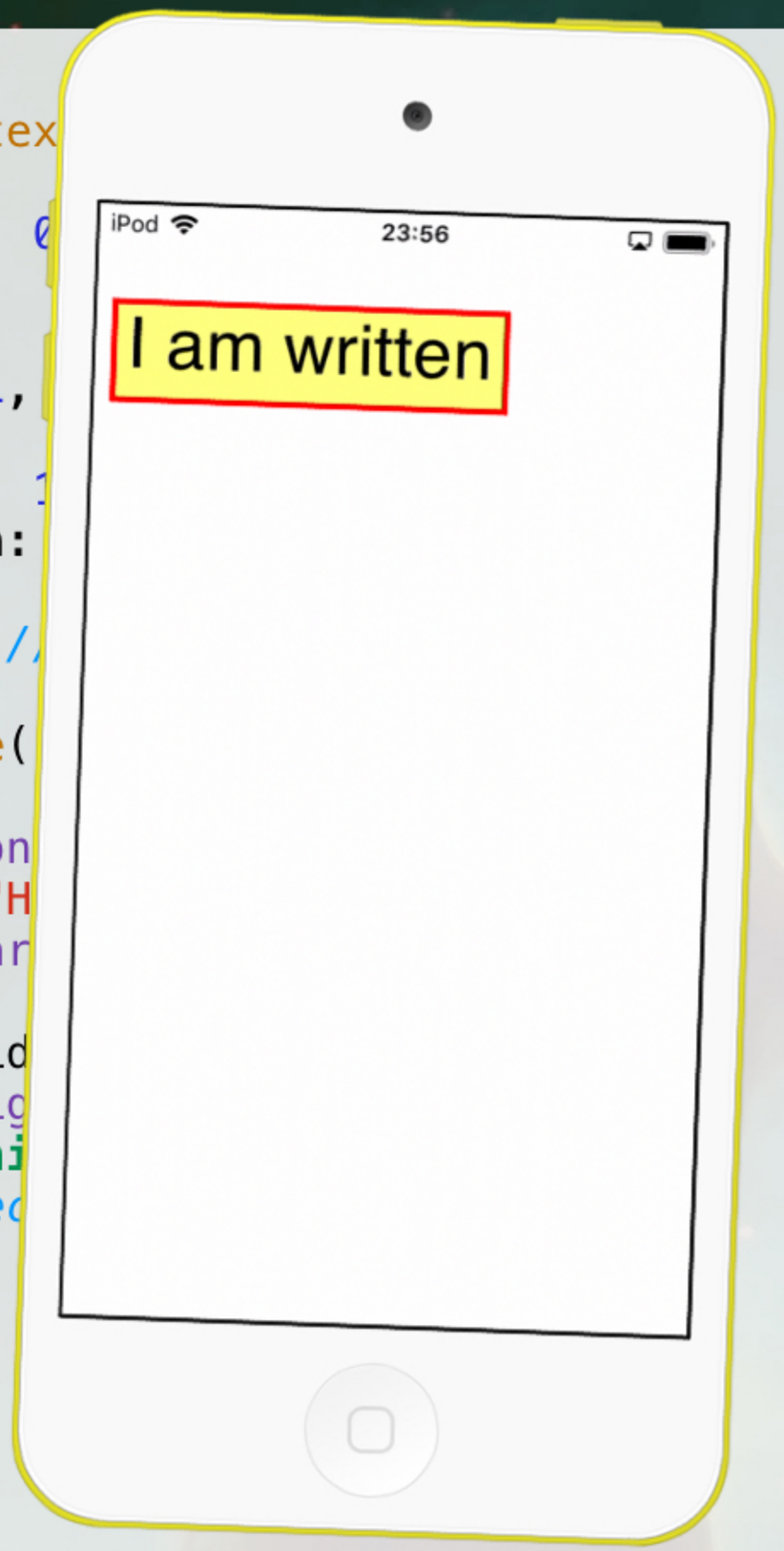
```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentCo
    context?.setStrokeColor(red: 0, gree
    context?.beginPath()
    context?.move(to: CGPoint(x: 75, y:
    context?.addLine(to: CGPoint(x: 10,
    context?.addLine(to: CGPoint(x: 160,
    context?.closePath()
    context?.strokePath()
    // self.setNeedsDisplay()// useless
}
```

Using fonts & colors


```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    // line size & color
    context?.setStrokeColor(red: 1, green: 0, blue: 0, alpha: 1)
    context?.setLineWidth(3)
    //fill color
    context?.setFillColor(red: 1, green: 1, blue: 0, alpha: 0.5)
    //draw a square
    let square = CGRect(origin: CGPoint(x: 10, y: 50),
                        size: CGSize(width: 200, height: 50))
    context?.addRect(square)
    context?.drawPath(using: .fillStroke) // stroke & fill
    // add a text
    let parStyle = NSMutableParagraphStyle()
    parStyle.alignment = .center
    let attrs = [NSAttributedString.Key.font:
                            UIFont(name: "Helvetica", size: 36)!,
                 NSAttributedString.Key.paragraphStyle: parStyle]
    let str = "I am written"
    str.draw(with: CGRect(x: 10, y: 50, width: 200, height: 50),
             options: .usesLineFragmentOrigin,
             attributes: attrs, context: nil)
    // self.setNeedsDisplay()// useless because we are in draw
}
```

Using fonts & colors


```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContex
    // line size & color
    context?.setStrokeColor(red: 1, green: 0
    context?.setLineWidth(3)
    //fill color
    context?.setFillColor(red: 1, green: 1,
    //draw a square
    let square = CGRect(origin: CGPoint(x: 1
                        size: CGSize(width:
    context?.addRect(square)
    context?.drawPath(using: .fillStroke) //
    // add a text
    let parStyle = NSMutableParagraphStyle(
    parStyle.alignment = .center
    let attrs = [NSAttributedString.Key.fon
                            UIFont(name: "H
                 NSAttributedString.Key.par
    let str = "I am written"
    str.draw(with: CGRect(x: 10, y: 50, wid
             options: .usesLineFragmentOrig
             attributes: attrs, context: ni
    // self.setNeedsDisplay()// useless bec
}
```



Objective-C & Core Graphics?

This is old… and C

- For a square

```
- (void)drawRect:(CGRect)rect {
    CGContextRef context1 = UIGraphicsGetCurrentContext();
    CGRect square = CGRectMake(10, 50, 200, 50);
    CGContextAddRect(context1, square);
    CGContextStrokePath(context1);
}
```

- For a circle

```
- (void)drawRect:(CGRect)rect {
    CGContextRef context2 = UIGraphicsGetCurrentContext();
    CGContextAddArc(context2, 200, 200, 70, 0, M_PI * 2, YES);
    CGContextFillPath(context2);
}
```

- For a polygon

```
- (void)drawRect:(CGRect)rect {
    CGContextRef context3 = UIGraphicsGetCurrentContext();
    CGContextBeginPath (context3);
    CGContextMoveToPoint (context3, 75, 30);
    CGContextAddLineToPoint (context3, 10, 150);
    CGContextAddLineToPoint (context3, 160, 150);
    CGContextClosePath (context3);
    CGContextStrokePath(context3);
}
```

Objective-C & Core Graphics?

This is old… and C

- For a square

```
- (void)drawRect:(CGRect)rect {
    CGContextRef context1 = UIGraphicsGetCurrentContext();
    CGRect square = CGRectMake(10, 50, 200, 50);
    CGContextAddRect(context1, square);
    CGContextStroke…ntext1);
}
```

- For a circle

```
- (void)drawRect:(
    CGContextRef c…
    CGContextAddAr…                                    …YES);
    CGContextFillP…
}
```

- For a polygon

```
- (void)drawRect:(CGRect)rect {
    CGContextRef context3 = UIGraphicsGetCurrentContext();
    CGContextBeginPath (context3);
    CGContextMoveToPoint (context3, 75, 30);
    CGContextAddLineToPoint (context3, 10, 150);
    CGContextAddLineToPoint (context3, 160, 150);
    CGContextClosePath (context3);
    CGContextStrokePath(context3);
}
```

Ave you noticed?

No [and]…

As a conclusion…

This is 2D drawing

- Library: Quartz2D
- Old traditional API (MacOS and iOS)

For games?

- OpenGL ES
  - Being unsupported
- Metal
  - Pushed by Apple
  - Relies on hardware acceleration
- SceneKit
  - For 3D
  - Relies on hardware acceleration

As a conclusion…

This is 2D drawing

- Library: Quartz2D
- Old traditional API (MacOS ar

For games?

- OpenGL ES
  - Being unsupported
- Metal
  - Pushed by Apple
  - Relies on hardw
- SceneKit
  - For 3D
  - Relies on hardware acceleration