TechMaster
Học là có việc

Animation, ViewController Transaction, Core Graphics

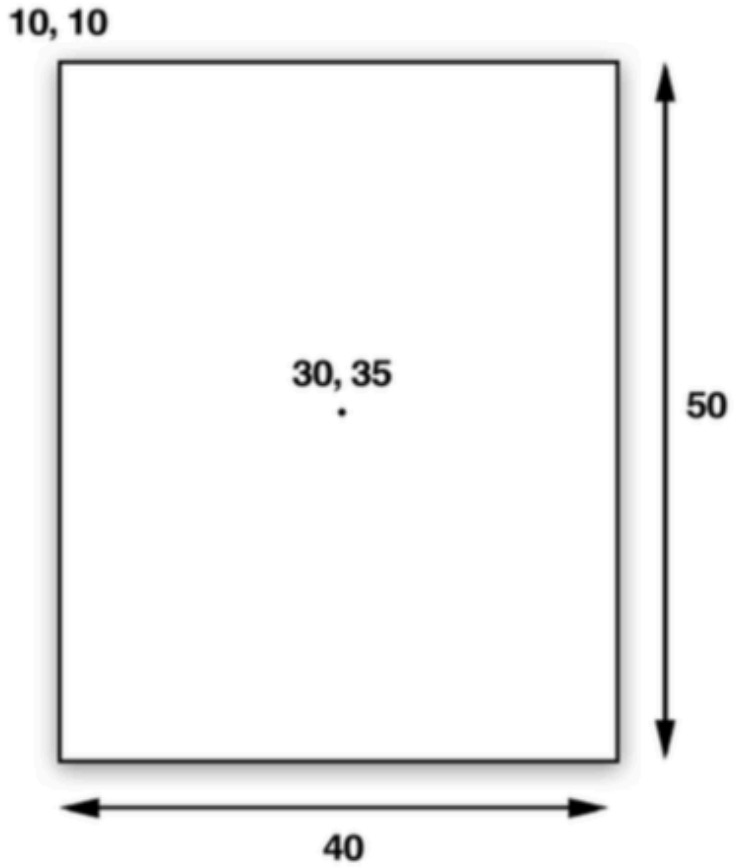
Core Amination

- Là Framework thực hiện công việc hiển thị, soạn và animate các visual elements.
- Core Animation cung cấp high frame rates và smooth animations mà không làm nặng CPU và làm chậm ứng dụng.
- Developer chỉ cần cấu hình các tham số như điểm bắt đầu và điểm kết thúc; Core Animation thực hiện phần còn lại, phân phát hầu hết công việc cho phần cứng đồ họa chuyên dụng, để tăng tốc hiển thị.
- https://developer.apple.com/documentation/quartzcore

Core Amination

- Trong iOS, tất cả các lớp giao diện đều được kế thừa từ lớp cơ sở chung UIView.
- Mỗi UIView được hỗ trợ bởi một lớp thuộc Core Animation là CALayer.
- UIViews kế thừa từ UIResponder, xử lý các sự kiện tương tác từ người dùng. Đồng thời hỗ trợ các hàm vẽ dựa trên Core Graphics.
- Tuy nhiên, vẫn còn một vài animations được thực hiện tốt nhất hoặc chỉ có thể thực hiện bằng cách tương tác với CALayer:
  - ✓ Đổ bóng, bo tròn, màu viền.
  - ✓ Biến đổi 3D.
  - ✓ Các chuyển động nhiều bước và phi tuyến tính …

Geometry - Layout



View:

frame = {10, 10, 40, 50}

bounds = {0, 0, 40, 50}

center = {30, 35}

Layer:

frame = {10, 10, 40, 50}

bounds = {0, 0, 40, 50}

position = {30, 35}

Transforms

- CGAffineTransform (Core Graphics): là một ma trận 2 cột, 3 dòng có thể nhân với một vector 2D (CGPoint) để thay đổi giá trị của nó.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

CGPoint CGAffineTransform Transformed CGPoint

- Core Graphics cung cấp các hàm cho phép biến đổi đối tượng:
  - ✓ CGAffineTransformMakeRotation(CGFloat angle)
  - ✓ CGAffineTransformMakeScale(CGFloat sx, CGFloat sy)
  - ✓ CGAffineTransformMakeTranslation(CGFloat tx, CGFloat ty)

Các options

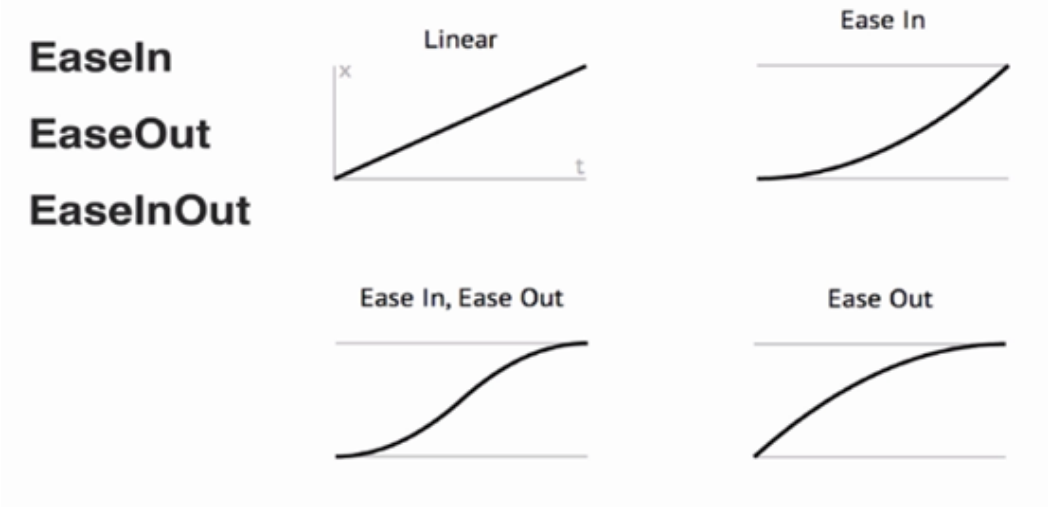
- Options điều chỉnh tốc độ:

  1. curveEaseIn // Run nhanh dần

  2. curveEaseOut // Run chận dần

  3. curveEaseInOut // kết hợp curveEaseIn vs curveEaseInOut Ban đầu nhanh dần, gần cuối chận dần

  4. curveLinear // Run 1 các đều đều



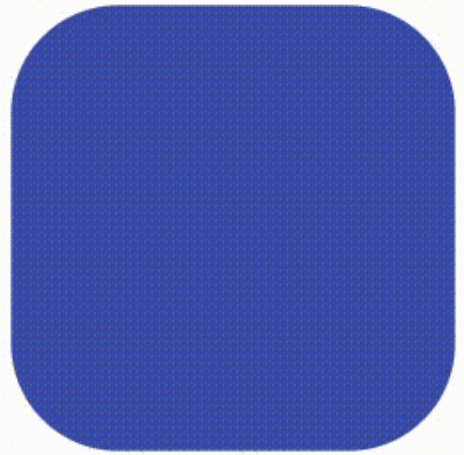
- Ngoài ra còn có

  5. Repeat : lặi lại animation

  6. AutoReverse : dùng kết hợp với Repeat ,khi bạn muốn tự động chuyển động ngược lại

Simple Animation

```
let newButtonWidth: CGFloat = 60

UIView.animate(withDuration: 2.0) {
    self.styledButton.frame = CGRect(x: 0, y: 0, width:
newButtonWidth, height: newButtonWidth)
    self.styledButton.center = self.view.center
    self.styledButton.layer.cornerRadius = newButtonWidth/2
}
```



UIView.transition

```
// 1. Dùng cho 1 view

class func transition(with view: UIView, duration: TimeInterval, options: UIViewAnimationOptions = [], animations: (() -> Void)?, completion: ((Bool) -> Void)? = nil)


// 2. Dùng chuyển đổi ẩn từ fromView và show toView

class func transition(from fromView: UIView, to toView: UIView, duration: TimeInterval, options: UIViewAnimationOptions = [], completion: ((Bool) -> Void)? = nil)
```

UIView.transition

```
// 1. Dùng cho 1 view

open class func transition(with view: UIView, duration: TimeInterval, options: UIViewAnimationOptions = [], animations: (() -> Swift.Void)?, completion: ((Bool) -> Swift.Void)? = nil)


// 2. Dùng chuyển đổi ẩn fromView và hiển thị toView

open class func transition(from fromView: UIView, to toView: UIView, duration: TimeInterval, options: UIViewAnimationOptions = [], completion: ((Bool) -> Swift.Void)? = nil) // toView added to fromView.superview, fromView removed from its superview
```

UIView.transition options

- `public static var transitionFlipFromLeft: UIViewAnimationOptions { get }`
- `public static var transitionFlipFromRight: UIViewAnimationOptions { get }`
- `public static var transitionCurlUp: UIViewAnimationOptions { get }`
- `public static var transitionCurlDown: UIViewAnimationOptions { get }`
- `public static var transitionCrossDissolve: UIViewAnimationOptions { get }`
- `public static var transitionFlipFromTop: UIViewAnimationOptions { get }`
- `public static var transitionFlipFromBottom: UIViewAnimationOptions { get }`

Inside Core Animation

- Các cá thể UIView có một thuộc tính CALayer được gọi là lớp mà animations thực hiện trên đó.

- Core Animation có hai cấu trúc cây song song:
  - Cây model layer (presentationLayer).
  - Cây presentation layer (modelLayer).

- Thông qua CABasicAnimation ta có thể animate các thuộc tính của CALayer.

Inside Core Animation

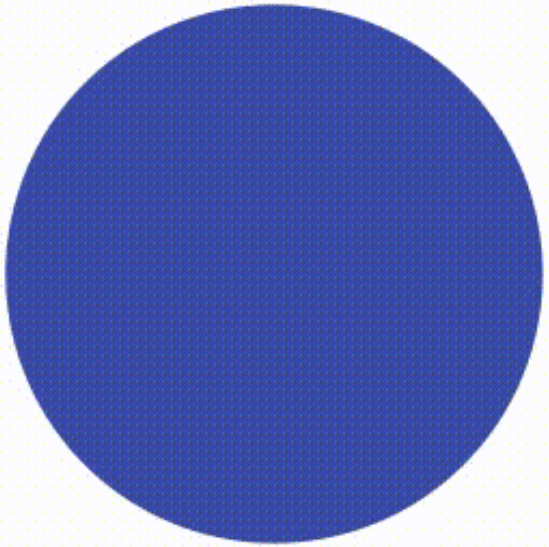
```
let cornerAnimation = CABasicAnimation(keyPath: #keyPath(CALayer.cornerRadius))
cornerAnimation.fromValue = oldValue
cornerAnimation.toValue = newValue
cornerAnimation.duration = 1.0

styledButton.layer.cornerRadius = newValue
styledButton.layer.add(cornerAnimation, forKey: #keyPath(CALayer.cornerRadius))
```

CATransaction - làm mượt animations

- CATransaction là nhóm nhiều hành động liên quan đến hoạt ảnh lại với nhau. Đảm bảo rằng các thay đổi animations được committed Core Animation cùng một lúc.

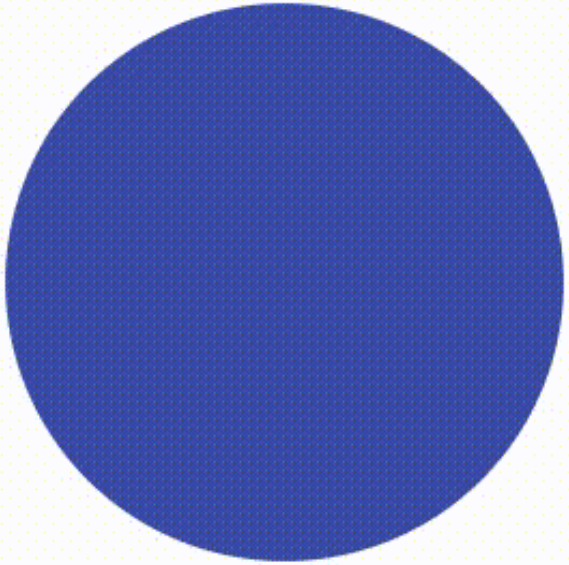
```
CATransaction.begin()
CATransaction.setAnimationDuration(0.5)

styledButton.layer.opacity = 0.5
styledButton.layer.backgroundColor = UIColor.white.cgColor

CATransaction.commit()
```

```swift
let oldValue = styledButton.frame.width/2
let newButtonWidth: CGFloat = 60

/* Do Animations */
CATransaction.begin()
CATransaction.setAnimationDuration(2.0)
CATransaction.setAnimationTimingFunction(CAMediaTimingFunction(name:
kCAMediaTimingFunctionEaseInEaseOut))

// View animations
UIView.animate(withDuration: 2.0) {
    self.styledButton.frame = CGRect(x: 0, y: 0, width: newButtonWidth, height: newButtonWidth)
    self.styledButton.center = self.view.center
}

// Layer animations
let cornerAnimation = CABasicAnimation(keyPath: #keyPath(CALayer.cornerRadius))
cornerAnimation.fromValue = oldValue
cornerAnimation.toValue = newButtonWidth/2

styledButton.layer.cornerRadius = newButtonWidth/2
styledButton.layer.add(cornerAnimation, forKey: #keyPath(CALayer.cornerRadius))

CATransaction.commit()
```



Custom Keyframe Animations

```
let timingFunction = CAMediaTimingFunction(controlPoints:
0.65, -0.55, 0.27, 1.55)
//...

CATransaction.setAnimationTimingFunction(timingFunction)
//... Do your UIKit animation
```



CATransition

- CATransition là class con của CAAnimation. Cung cấp một animation transition giữa các trạng thái của một layer.
- CATransition có type và subtype được sử dụng để xác định các hiệu ứng chuyển động.

```
/* Common transition types. */

@available(iOS 2.0, *)
public let kCATransitionFade: String

@available(iOS 2.0, *)
public let kCATransitionMoveIn: String

@available(iOS 2.0, *)
public let kCATransitionPush: String

@available(iOS 2.0, *)
public let kCATransitionReveal: String
```

```
/* Common transition subtypes. */

@available(iOS 2.0, *)
public let kCATransitionFromRight: String

@available(iOS 2.0, *)
public let kCATransitionFromLeft: String

@available(iOS 2.0, *)
public let kCATransitionFromTop: String

@available(iOS 2.0, *)
public let kCATransitionFromBottom: String
```

ViewControllers transition

Trong iOS để chuyển từ viewcontroller này sang viewcontroller khác chúng ta có 2 cách:

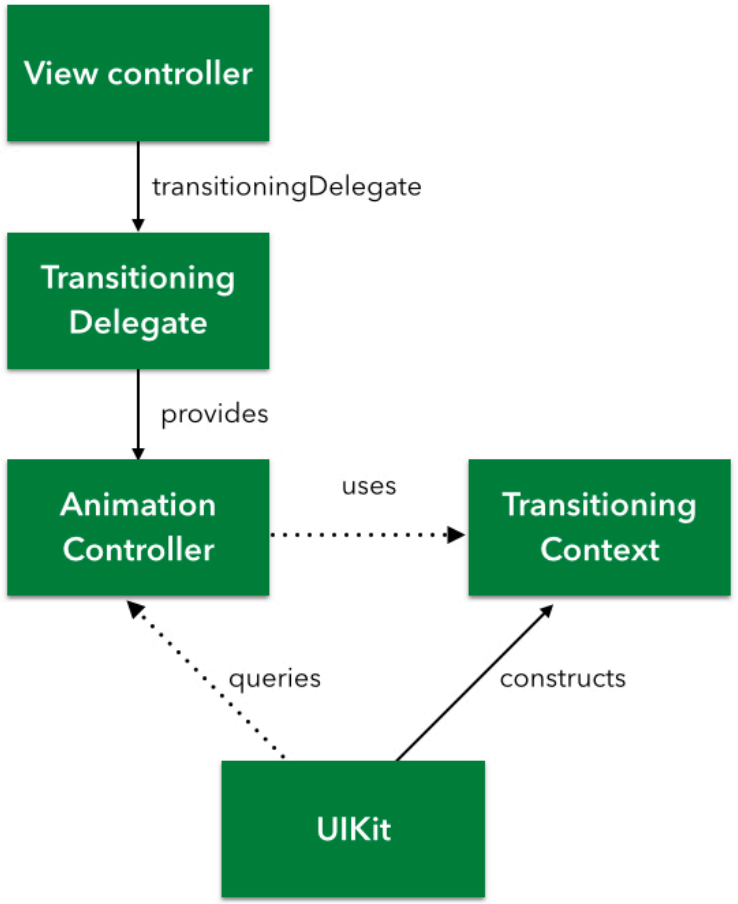
- Đơn giản nhất là từ viecontroller A, sử dụng hàm:

```
func present(_ viewControllerToPresent: UIViewController, animated flag: Bool, completion: (() -> Swift.Void)? = nil)
```

- Cách thứ 2: dùng navigationController quản lý việc chuyển đổi các viewController này.

Tuy nhiên 2 cách trên thì được apple xây dựng sẵn cho chúng ta nên không thể tuỳ chỉnh việc các viewController xuất hiện: thêm hiệu ứng, tuỳ chỉnh thời gian chuyển đổi ...

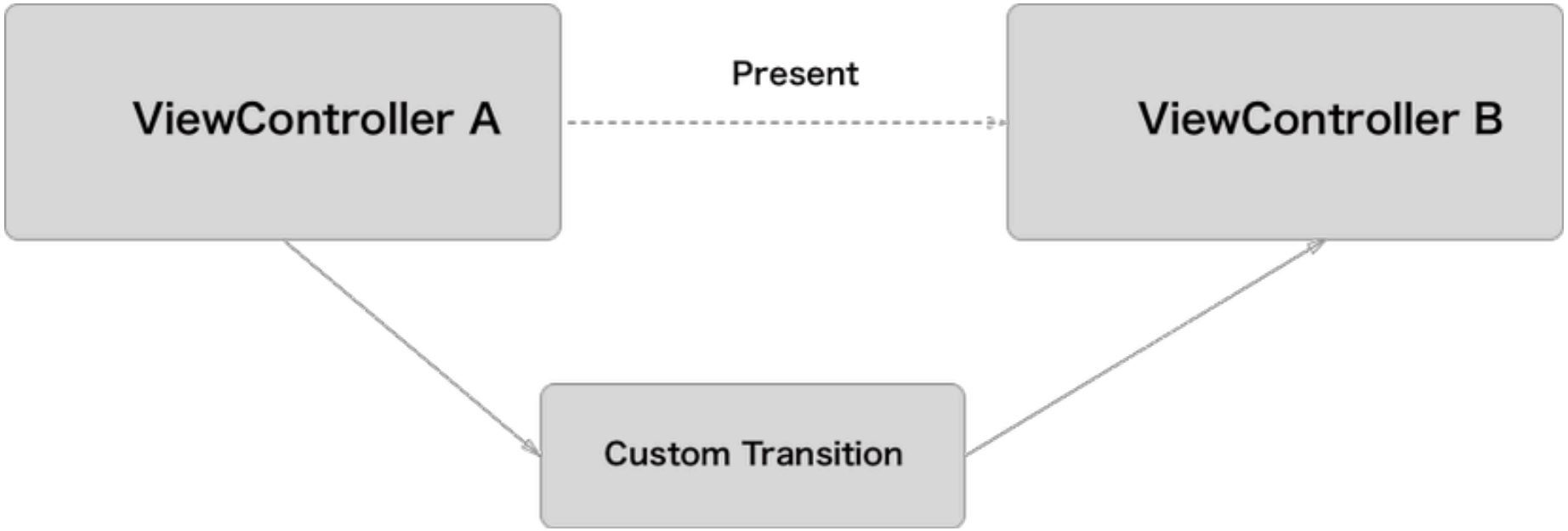
Transitioning diagram



Transitioning process

- Developer kích hoạt quá trình chuyển đổi, bằng code hoặc thông qua segue.
- UIKit hỏi to-viewcontroller có transitioning delegate hay không. Nếu nó không có, UIKIt sử dụng trasition chuẩn, tích hợp sẵn.
- UIKit sau đó hỏi transitioning delegate có animation controller không thông qua animationController(forPresented:presenting:source:). Nếu hàm này trả về nil, quá trình chuyển đổi sẽ sử dụng animation mặc định.
- UIKit xây dựng transitioning context.
- UIKit hỏi bộ animation controller về duration của animation bằng cách gọi transitionDuration(using:).
- UIKit gọi transitionDuration(using:) trên animation controller để thực hiện animation cho quá trình chuyển đổi.
- Cuối cùng, bộ điều khiển hoạt ảnh gọi hàm completeTransition (_ :) trên transitioning context để cho biết rằng animation hoàn tất.

Tuỳ chỉnh viewcontroller transition



Tuỳ chỉnh viewcontroller transition

- Đầu tiên ta tạo 1 file CustomPresentModalAnimator
  - ✓
```
class CustomPresentModalAnimator: NSObject,
UIViewControllerAnimatedTransitioning {}
```
- Để file CustomPresentModalAnimator adpot protocol UIViewControllerAnimatedTransitioning ta thêm vào 2 hàm sau:
  - ✓
```
func transitionDuration(using transitionContext:
UIViewControllerContextTransitioning?) -> TimeInterval
```
  - ✓
```
func animateTransition(using transitionContext:
UIViewControllerContextTransitioning)
```

Tuỳ chỉnh viewcontroller transition

- Tiếp đó ở CustomPresentModalViewController ta adopt protocol UIViewControllerTransitioningDelegate và thêm 2 hàm sau:
  - ✓
```
func animationController(forPresented presented: UIViewController,
presenting: UIViewController, source: UIViewController) ->
UIViewControllerAnimatedTransitioning?
```

- viewcontroller sẽ nhận object trả về ở đây để làm animation khi ta present viewcontroller
  - ✓
```
func animationController(forDismissed dismissed:
UIViewController) -> UIViewControllerAnimatedTransitioning?
```

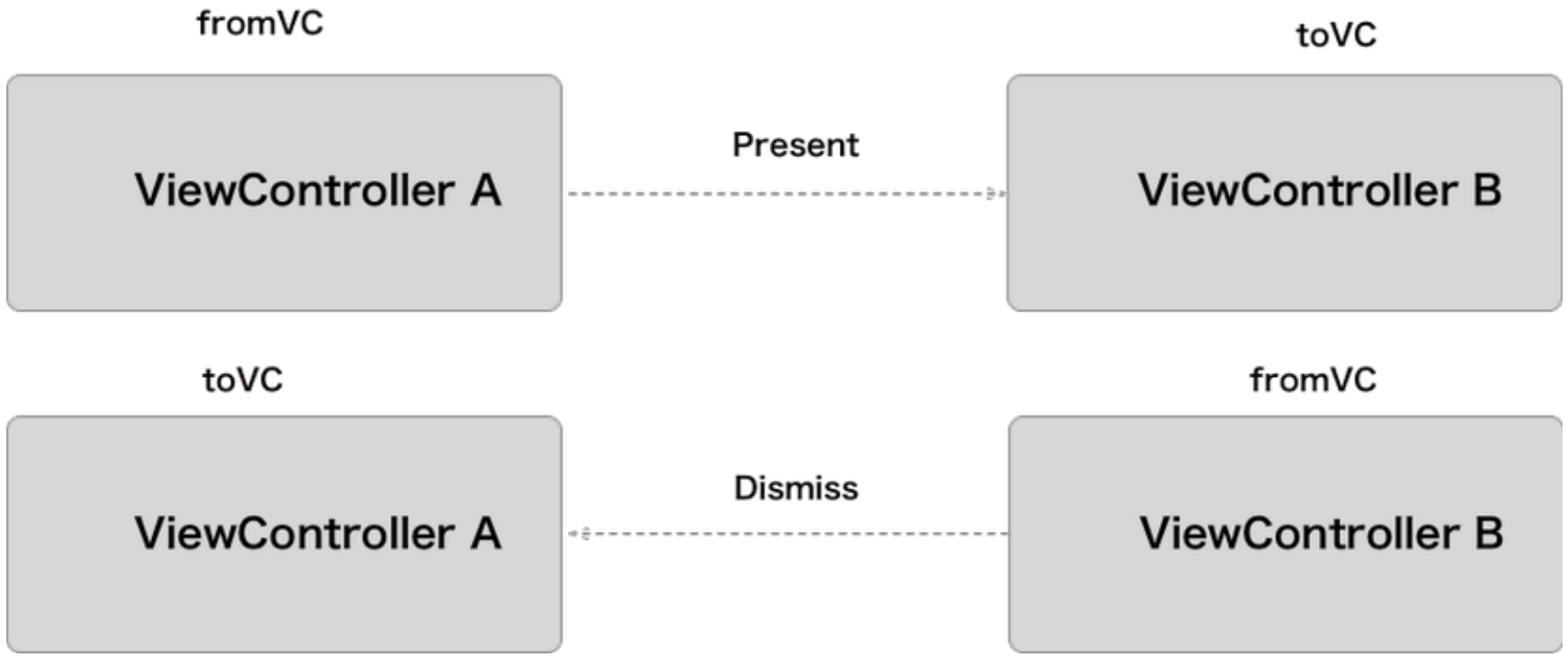
- Viewcontroller sẽ nhận object trả về ở đây để làm animation khi dismiss viewcontroller vừa present lên

Tuỳ chỉnh viewcontroller transition

- Mở lại file CustomPresentModalAnimator ta sẽ thêm vào 1 số thuộc tính sau:
  - ✓ `let duration = 0.4`
  - ✓ `var presenting = true`
  - ✓ `var xScaleFactor: CGFloat = 0.0`
  - ✓ `var yScaleFactor: CGFloat = 0.0`
  - ✓ `var originFrame = CGRect.zero`
  - ✓ `var initialFrame = CGRect.zero`
  - ✓ `var finalFrame = CGRect.zero`

- duration: là thời gian dùng để hiển thị animation trong quá trình chuyển đổi viewcontroller.
- presenting: xác định khi nào là present , khi nào dismiss viewcontroller
- xScaleFactor, yScaleFactor: tỉ lệ avatar imageview khi tiến hành animation.
- originFrame, initialFrame, finalFrame: frame của avatar imageview thay đổi trong quá trình present lên và dismiss

present và dismiss

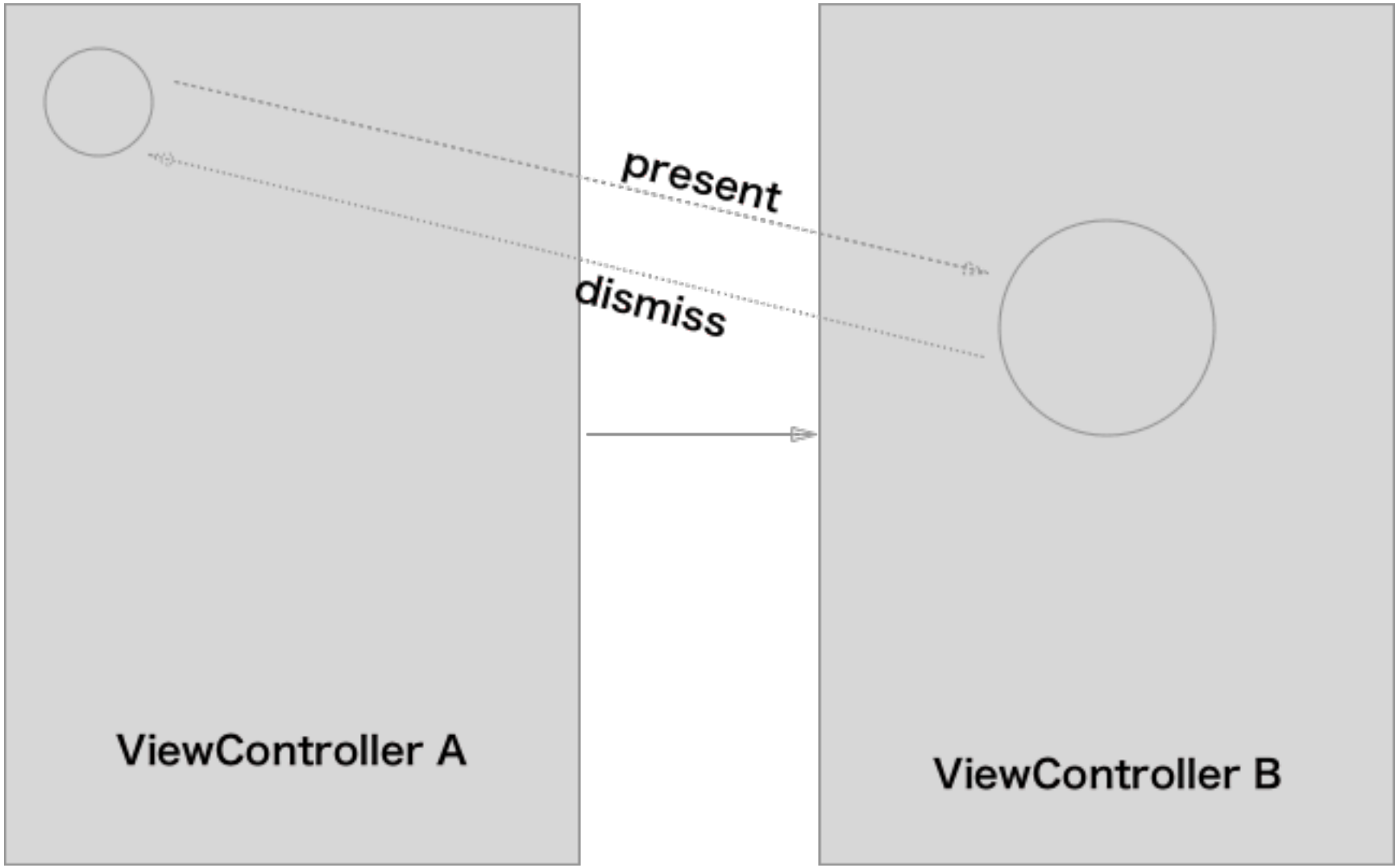


Xác định fromVC và toVC

- Để xác định đc viewcontroller fromVC và toVC ta thêm đoạn code sau vào hàm animateTransition:

```
guard let fromVC = transitionContext.viewController(forKey:
UITransitionContextViewControllerKey.from) else {
            return
        }

guard let toVC = transitionContext.viewController(forKey:
UITransitionContextViewControllerKey.to) else {
            return
        }
```

Custom animation



Custom animation

Ẩn avatar của 2 viewcontroller:

```
let navigationControllerFirst = presenting ? (fromVC as! UINavigationController) : (toVC as! UINavigationController)
let customPresentModalVC = navigationControllerFirst.viewControllers.last as! CustomPresentModalViewController

// get avatar image view
let topView = customPresentModalVC.view.subviews.compactMap{ $0 }.filter{ $0.tag == 68 }.first!
let avatarImageViewFirst = topView.subviews.compactMap{ $0 }.filter{ $0.tag == AvatarImageViewTag }.first!

// hide avatar image view
 avatarImageViewFirst.layer.opacity = 0.0
```

Custom animation

Tạo fakeAvatarImageView:

```
// make fake avatar image view for animation
let fakeAvatarImageView = UIImageView(image: UIImage(named: "image.jpeg"))
fakeAvatarImageView.contentMode = .scaleAspectFill
fakeAvatarImageView.clipsToBounds = true
```

Custom animation

Tính toán tỉ lệ scale của avatar imageview:

```
// when present it will move from top to center
// when dismiss it will move from center to top
if presenting {
    // when present init frame will be avatar image frame in CustomPresentModalViewController
    initialFrame = originFrame

    // when present final frame will be avatar image frame in AvatarViewController
    finalFrame = avatarImageViewSecond.frame

    // set fake avatar image frame
    fakeAvatarImageView.frame = initialFrame
    fakeAvatarImageView.setRounded()
} else {
    // when dismiss init frame will be avatar image frame in AvatarViewController
    initialFrame = avatarImageViewSecond.frame

    // when dismiss final frame will be avatar image frame in CustomPresentModalViewController
    finalFrame = originFrame

    // set fake avatar image frame
    fakeAvatarImageView.frame = initialFrame
    fakeAvatarImageView.setRounded()
}
```

Custom animation

Tạo animation:

```
// make animation for transition between controllers
UIView.animate(withDuration: transitionDuration(using: transitionContext),
               delay: 0,
               options: .curveEaseInOut,
               animations: {
  fakeAvatarImageView.transform = scaleTransform
  fakeAvatarImageView.center = CGPoint(x: self.finalFrame.midX, y: self.finalFrame.midY)
  toVC.view.layer.opacity = 1.0
}) { finished in
  // show avatar in both view controller
  avatarImageViewFirst.layer.opacity = 1.0
  avatarImageViewSecond.layer.opacity = 1.0

  // remove fake avatar image view
  fakeAvatarImageView.removeFromSuperview()

  // complete
  transitionContext.completeTransition(!transitionContext.transitionWasCancelled)
}
```

Core Graphic

iOS cập nhật context bằng cách gọi draw (_ :) bất cứ khi nào view cần được cập nhật.

Điều này xảy ra khi:

✓ Hiển thị view mới lên màn hình.

✓ Các view khác trên top được di chuyển.

✓ Thuộc tính hidden của view được thay đổi.

✓ Ứng dụng gọi phương thức setNeedsDisplay () hoặc setNeedsDisplayInRect () trên view.

Core Graphic

Có hai bước để custom drawings:

- Tạo một lớp con UIView.
- Override hàm draw(_:) và thêm một số Core Graphics drawing code.

Draw a shape in Core Graphics

Có ba nguyên tắc cơ bản cần biết về đường path:

- Một path có thể được stroked và filled.
- Một stroke vạch ra đường path trong stroke color hiện tại.
- Một fill sẽ đổ đầy vào một đường path khép kín với fill color hiện tại.

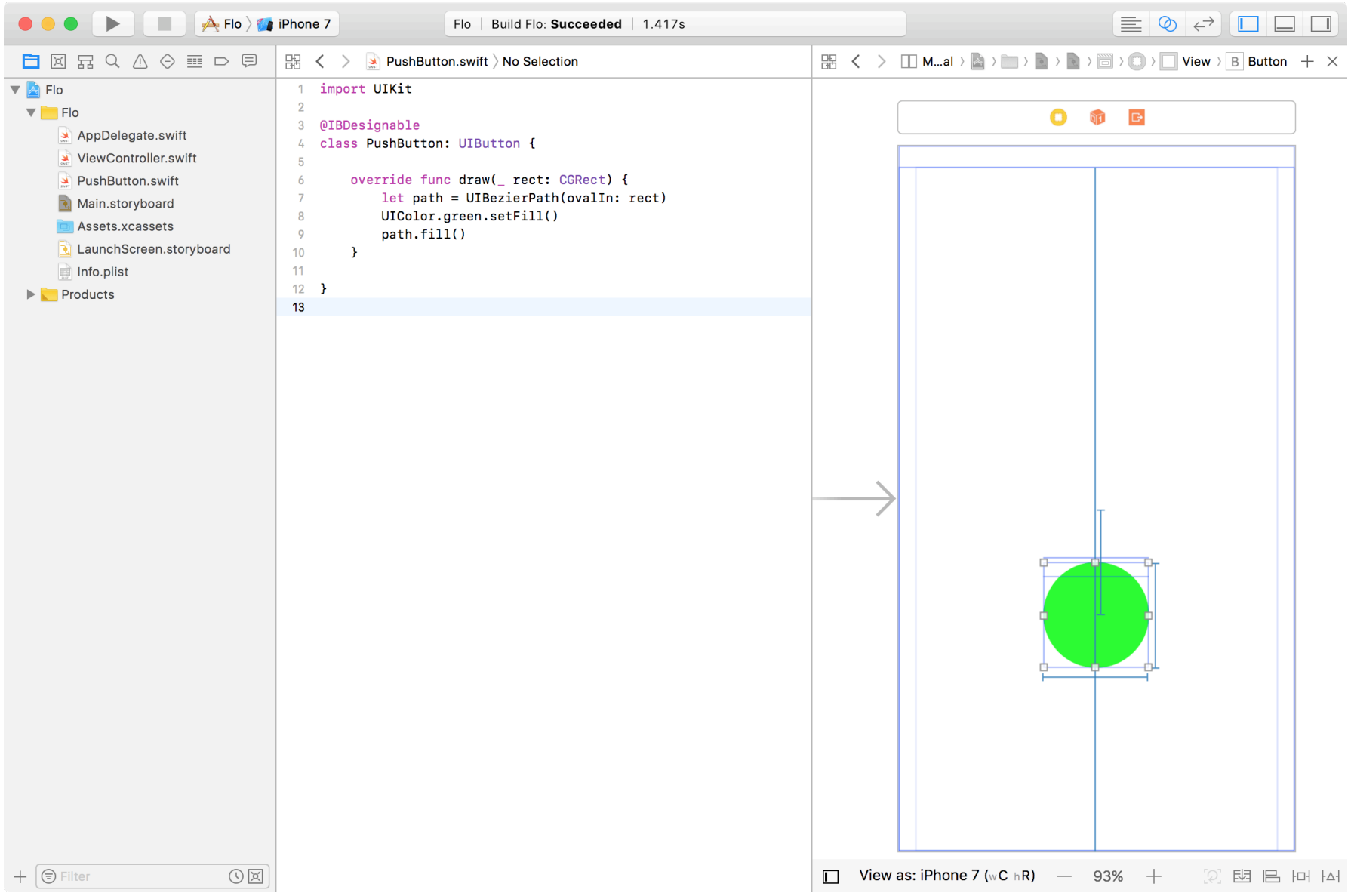
Một cách dễ dàng để tạo ra một đường path Core Graphics là thông qua một lớp gọi là UIBezierPath

@IBDesignable và @IBInspectable

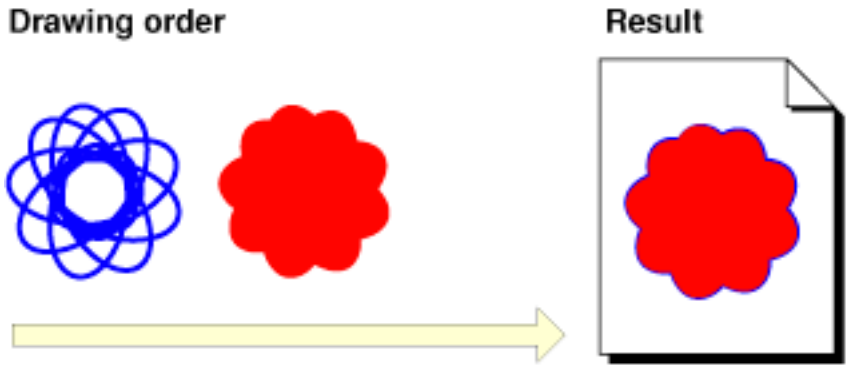
- @IBDesignable giúp thấy được sự thay đổi trực tiếp trên các giao diện như Xib, Storyboard. Điều này cho phép xem các chế độ tùy chỉnh của bạn sẽ xuất hiện mà không cần xây dựng và chạy ứng dụng của bạn sau mỗi thay đổi.

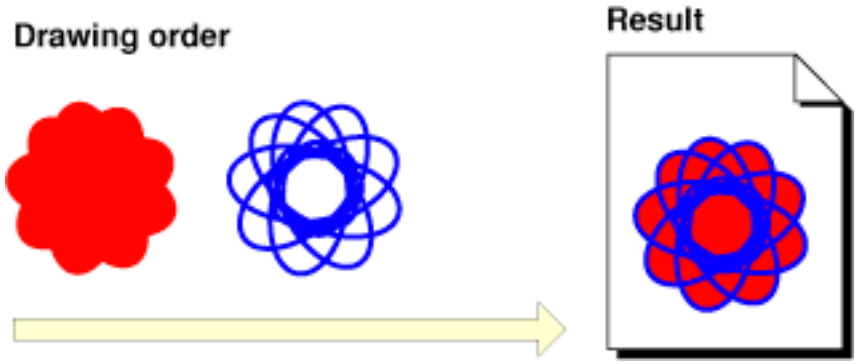
- @IBInspectable là một attribute mà bạn có thể thêm vào một property làm cho nó có thể đọc được bởi Interface Builder.

@IBDesignable và @IBInspectable



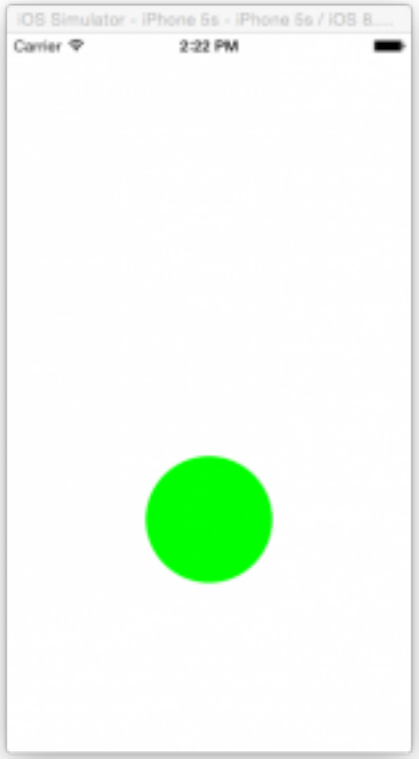
Drawing Into the Context





Vẽ button đơn giản

```
override func draw(_ rect: CGRect) {
  let path = UIBezierPath(ovalIn: rect)
  UIColor.green.setFill()
  path.fill()
}
```



Định nghĩa thông số để vẽ

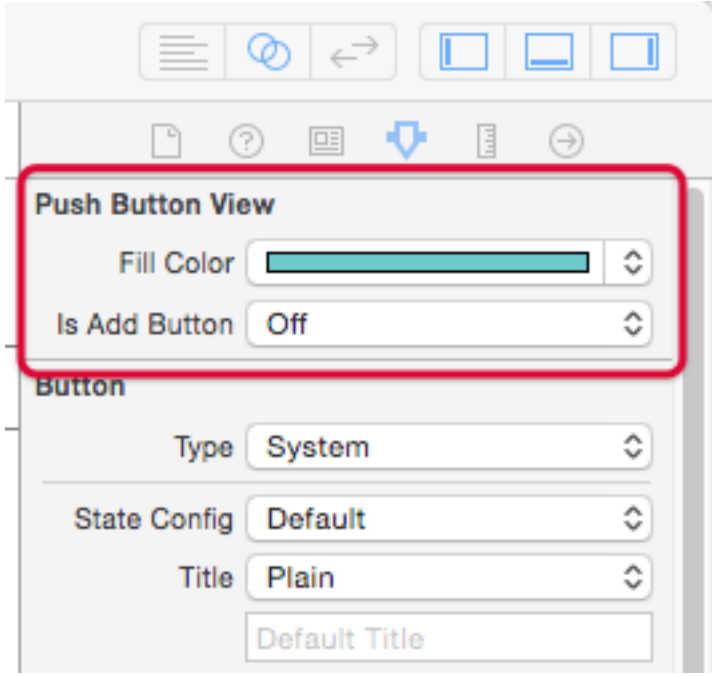
```
private struct Constants {
  static let plusLineWidth: CGFloat = 3.0
  static let plusButtonScale: CGFloat = 0.6
  static let halfPointShift: CGFloat = 0.5
}

private var halfWidth: CGFloat {
  return bounds.width / 2
}

private var halfHeight: CGFloat {
  return bounds.height / 2
}
```

Vẽ button

```
//set up the width and height variables
//for the horizontal stroke
let plusWidth: CGFloat = min(bounds.width, bounds.height) * Constants.plusButtonScale
let halfPlusWidth = plusWidth / 2

//create the path
let plusPath = UIBezierPath()

//set the path's line width to the height of the stroke
plusPath.lineWidth = Constants.plusLineWidth

//move the initial point of the path
//to the start of the horizontal stroke
plusPath.move(to: CGPoint(
  x: halfWidth - halfPlusWidth,
  y: halfHeight))

//add a point to the path at the end of the stroke
plusPath.addLine(to: CGPoint(
  x: halfWidth + halfPlusWidth,
  y: halfHeight))

//set the stroke color
UIColor.white.setStroke()

//draw the stroke
plusPath.stroke()
```

Custom Storyboard Properties

```
@IBInspectable var fillColor: UIColor = UIColor.green
@IBInspectable var isAddButton: Bool = true

override func draw(_ rect: CGRect) {
    ...
    fillColor.setFill()
    ...

    if isAddButton {
      //move the initial point of the path
      //to the start of the horizontal stroke
      plusPath.move(to: CGPoint(
        x: halfWidth - halfPlusWidth + Constants.halfPointShift,
        y: halfHeight + Constants.halfPointShift))

   //add a point to the path at the end of the stroke
      plusPath.addLine(to: CGPoint(
        x: halfWidth + halfPlusWidth + Constants.halfPointShift,
        y: halfHeight + Constants.halfPointShift))
    }

   ...
  }
}
```

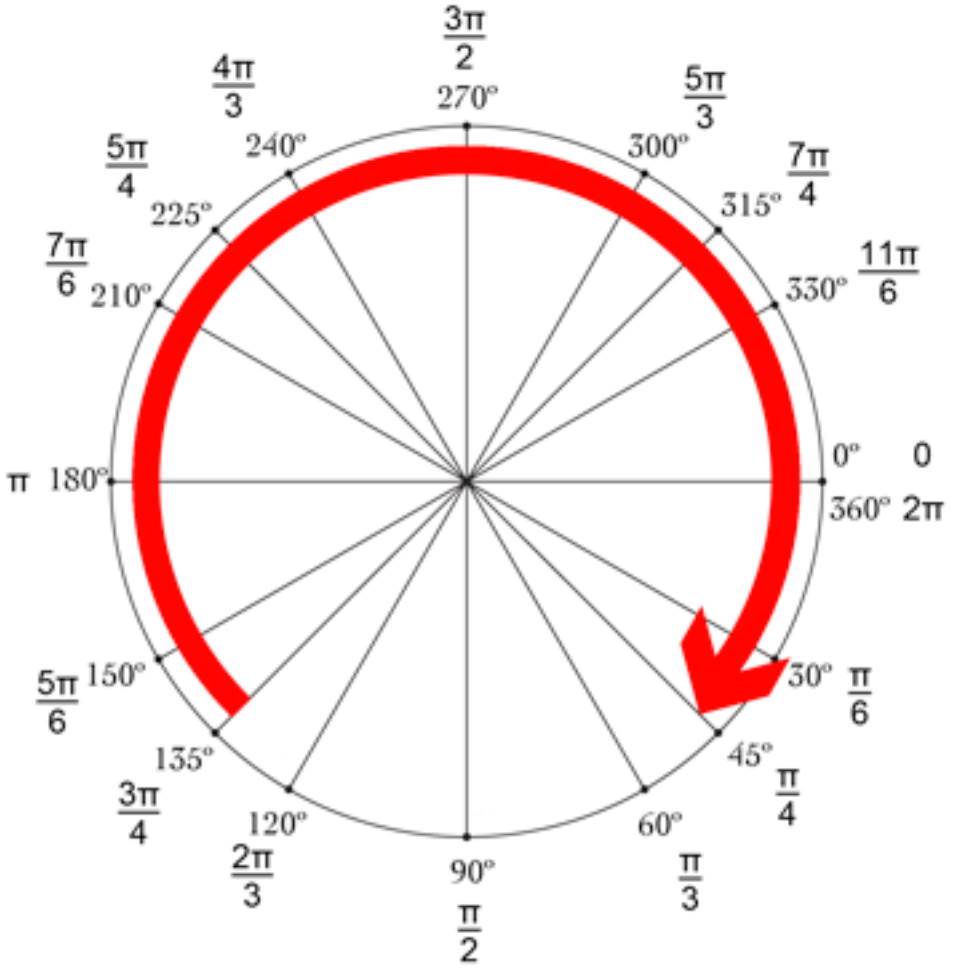


Arcs with UIBezierPath


```
@IBDesignable class CounterView: UIView {

  private struct Constants {
    static let numberOfGlasses = 8
    static let lineWidth: CGFloat = 5.0
    static let arcWidth: CGFloat = 76

    static var halfOfLineWidth: CGFloat {
      return lineWidth / 2
    }
  }

  @IBInspectable var counter: Int = 5
  @IBInspectable var outlineColor: UIColor = UIColor.blue
  @IBInspectable var counterColor: UIColor = UIColor.orange

  override func draw(_ rect: CGRect) {

  }
}
```

Arcs with UIBezierPath

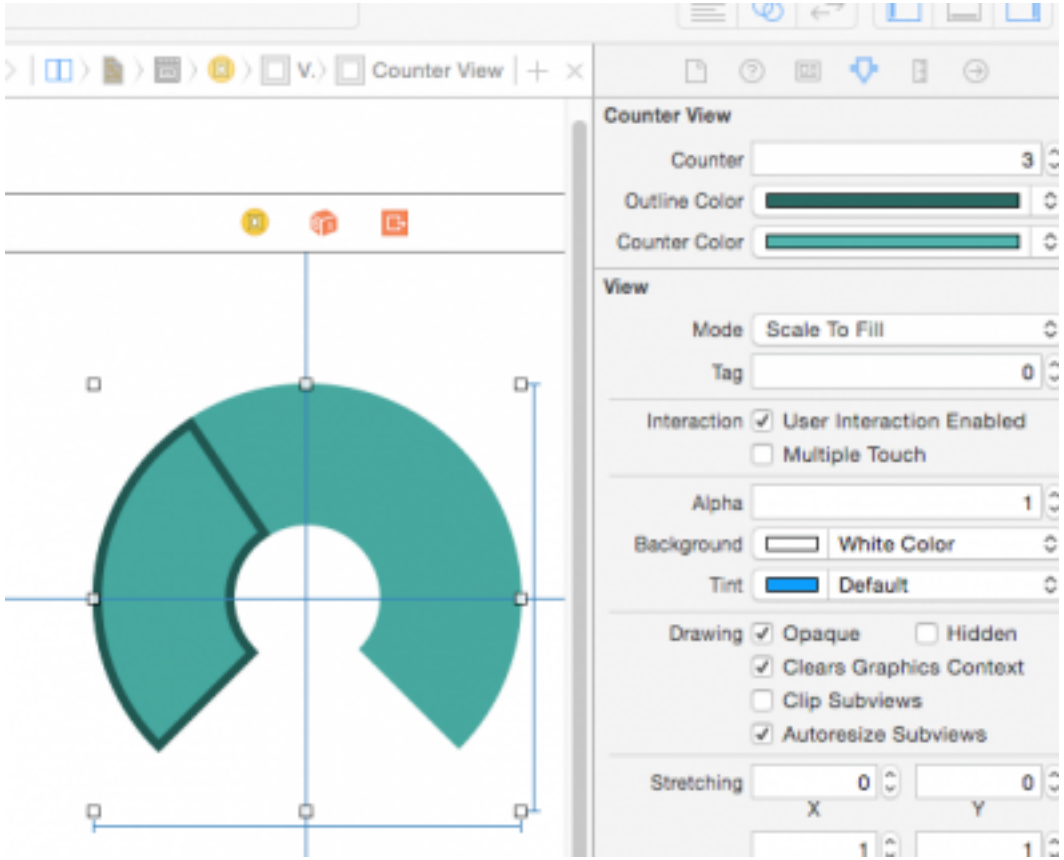
```
// 1
let center = CGPoint(x: bounds.width / 2, y: bounds.height / 2)

// 2
let radius: CGFloat = max(bounds.width, bounds.height)

// 3
let startAngle: CGFloat = 3 * .pi / 4
let endAngle: CGFloat = .pi / 4

// 4
let path = UIBezierPath(arcCenter: center,
                        radius: radius/2 - Constants.arcWidth/2,
                    startAngle: startAngle,
                      endAngle: endAngle,
                     clockwise: true)

// 5
path.lineWidth = Constants.arcWidth
counterColor.setStroke()
path.stroke()
```



Outlining the Arc

```
//Draw the outline

//1 - first calculate the difference between the two angles
//ensuring it is positive
let angleDifference: CGFloat = 2 * .pi - startAngle + endAngle
//then calculate the arc for each single glass
let arcLengthPerGlass = angleDifference / CGFloat(Constants.numberOfGlasses)
//then multiply out by the actual glasses drunk
let outlineEndAngle = arcLengthPerGlass * CGFloat(counter) + startAngle

//2 - draw the outer arc
let outlinePath = UIBezierPath(arcCenter: center,
                               radius: bounds.width/2 - Constants.halfOfLineWidth,
                           startAngle: startAngle,
                             endAngle: outlineEndAngle,
                            clockwise: true)

//3 - draw the inner arc
outlinePath.addArc(withCenter: center,
                       radius: bounds.width/2 - Constants.arcWidth + Constants.halfOfLineWidth,
                   startAngle: outlineEndAngle,
                     endAngle: startAngle,
                    clockwise: false)

//4 - close the path
outlinePath.close()

outlineColor.setStroke()
outlinePath.lineWidth = Constants.lineWidth
outlinePath.stroke()
```

Kết quả


Counter View
Counter View
Counter
3
Outline Color
Counter Color
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
White Color
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
1
1