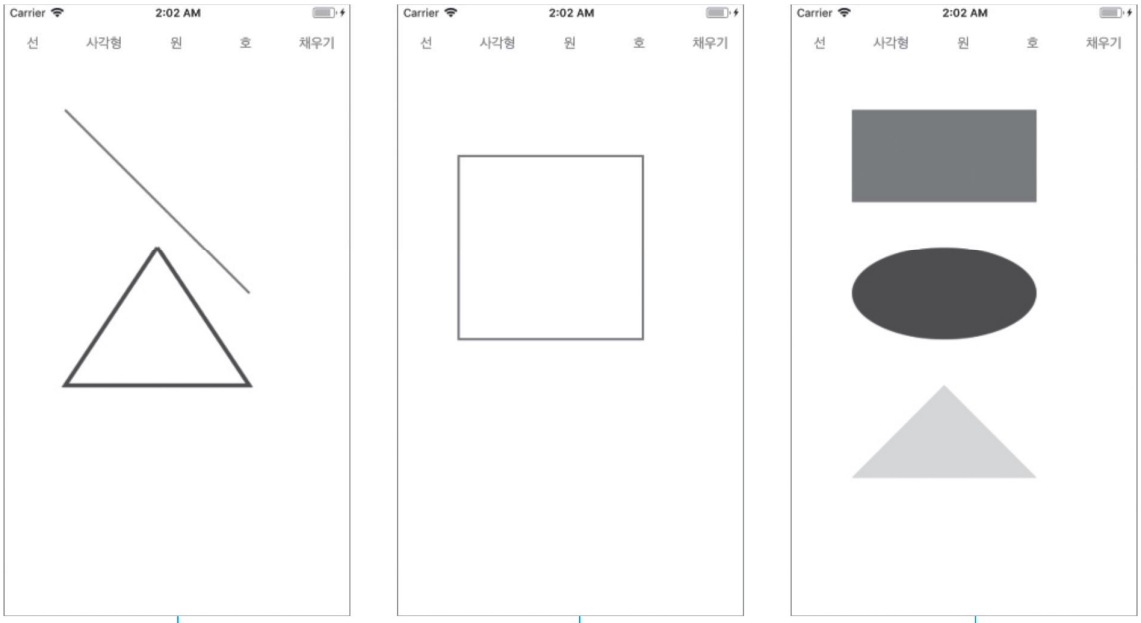
16 장. 코어 그래픽스로 화면에 그림 그리기

iOS 에서는 코어 그래픽스라는 그래픽 라이브러리를 사용하여 뷰에 그림을 그릴 수 있다. 코어 그래픽스는 선, 사각형, 원 같은 도형을 그리거나 도형에 색을 채우는 등 다양한 기능들에 활용할 수 있다.

코어 그래픽스의 다양한 라이브러리들을 사용하여 간단한 도형을 그리고 도형 내부를 특정 색으로 채우는 방법을 알아보자.

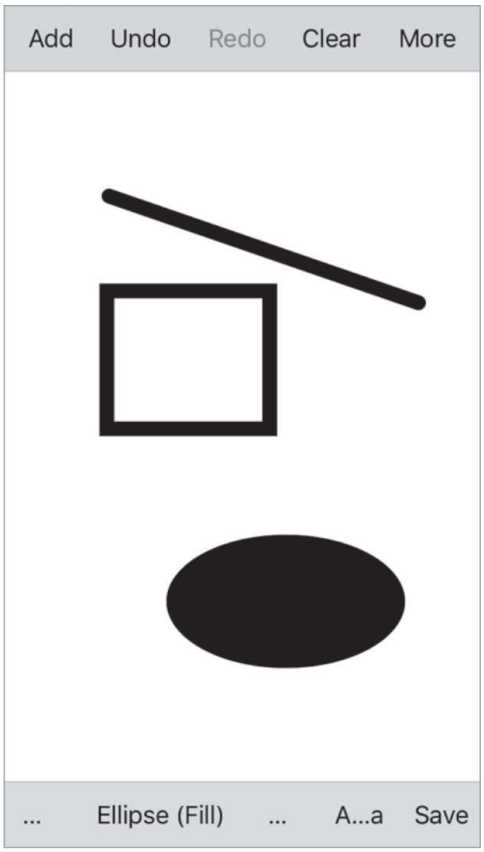


화면 위쪽의 [선], [사각형], [원], [호], [채우기] 버튼을 클릭해 해당하는 도형을 화면에 그릴 수 있습니다.

16-1 코어 그래픽스란?

코어 그래픽스(Core Graphics)란 아이폰과 아이패드에서 2 차원 그래픽을 그릴 수 있도록 제공하는 그래픽 라이브러리입니다. 코어 그래픽스는 애플이 제공하는 '쿼츠(Quartz)'라는 그래픽 라이브러리 안에 포함되어 있다.

이러한 코어 그래픽스의 사용법을 익히고 나면 아이폰에 그림을 그릴 수 있다.



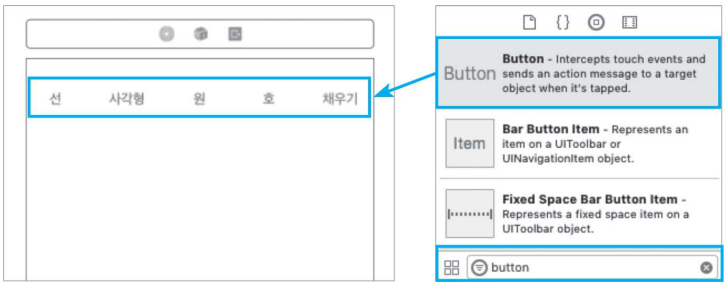
드로우 데스크(Draw Desk) 앱

16-2 코어 그래픽스를 위한기본 환경 구성하기

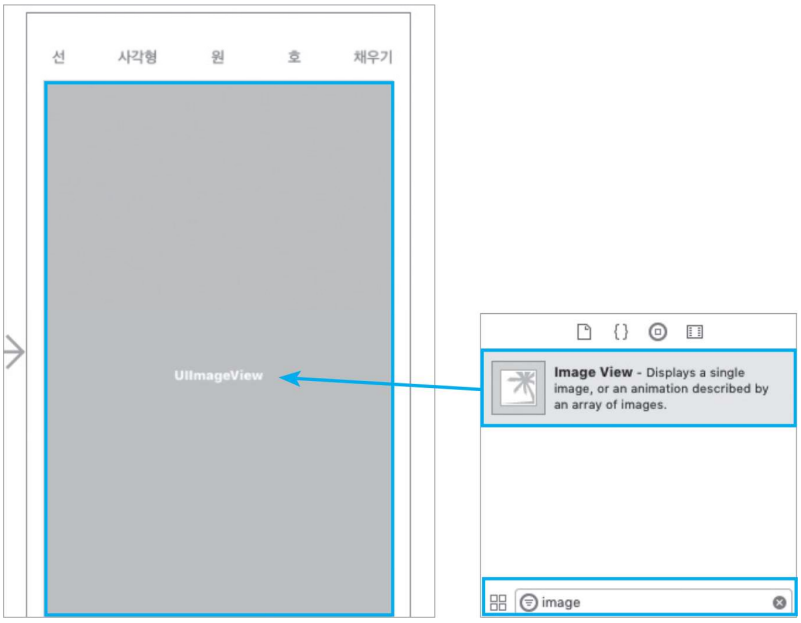
1. Xcode 를 실행한 후 'DrawGraphics'라는 이름으로 새 프로젝트를 만든다.

2. 뷰 컨트롤러 크기 조절하기

3. 오른쪽 아랫부분의 오브젝트 라이브러리에서 [버튼(Button)]을 찾아 스토리보드로 끌어와 화면의 위쪽에 배치한다. 여러 종류의 그림을 그릴 것이므로 버튼을 5 개 추가하고 텍스트를 각각 '선','사각형','원','호','채우기'로 변경한다.



4. 두번째로 오른쪽 아랫부분의 오브젝트 라이브러리에서 [이미지 뷰(Image View)]를 찾아 스토리보드로 끌어와 앞에서 배치한 버튼의 아랫부분에 갖다 놓은 후 크기 조절을 한다.



16-3 아웃렛 변수와 액션 함수 추가하기

1. 아웃렛 변수와 액션 함수를 추가하기 위해 오른쪽 윗부분의 [Show the Assistant editor] 번튼을 클릭하여 보조 편집기 영역을 연다.

2. 이미지 뷰에 대한 아웃렛 변수 추가하기
우선, 이미지 뷰 객체에 대한 아웃렛 변수를 추가한다,
스토리보드의 [이미지 뷰(Image View)]를 마우스 오른쪽 버튼으로 클릭한 후 오른쪽 보조 편집기 영역으로 드래그하면 다음 그림과 같이 연결선이 나타난다. 연결선을 뷰 컨트롤러의 선언문 바로 아래쪽에 갖다 놓은 후 마우스 버튼에서 손을 땐다.

3. 다음표를 참고하여 아웃렛 변수를 연결한다.

위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	imgView
유형(Type)	UIImageView

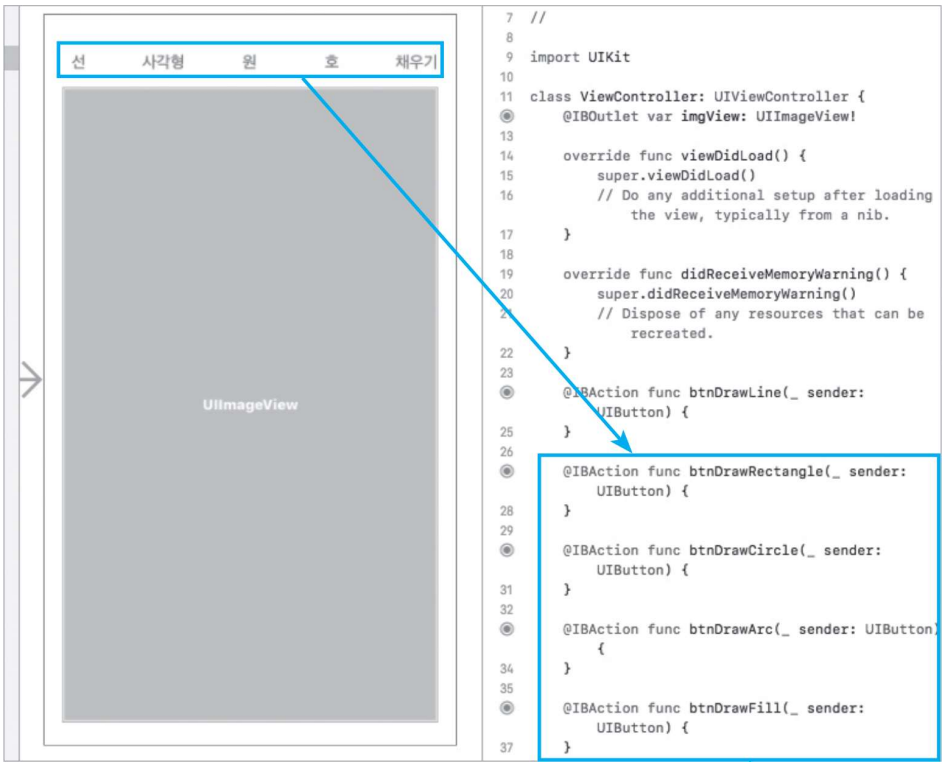
4. 버튼에 대한 액션 함수 추가하기
두번째로 버튼들에 대한 액션 함수를 추가한다. 마우스 오른쪽 버튼으로 첫번째 버튼을 클릭한 후 드래그해서 오른쪽 소스의 가장 아래쪽에 있는 닫힘 '}'의 바로 위에 갖다 놓는다,

5. 다음표를 참고하여 액션 함수를 추가한다.

위치	뷰 컨트롤러의 클래스 닫힘 괄호 '}' 바로 위
연결(Connection)	Action
이름(Name)	btnDrawLine
유형(Type)	UIButton

6. 네 개의 버튼에 대한 액션 함수를 추가한다. 각 버튼에 대한 액션 함수의 이름은 다음과 같다.
(사각형: btnDrawRectangle | 원: btnDrawCircle | 호: btnDrawArc | 채우기: btnDrawFill)

위치	뷰 컨트롤러의 클래스 닫힘 괄호 '}' 바로 위
연결(Connection)	Action
이름(Name)	- [사각형] 버튼: btnDrawRectangle - [원] 버튼: btnDrawCircle - [호] 버튼: btnDrawArc - [채우기] 버튼: btnDrawFill
유형(Type)	UIButton




```
//

import UIKit

class ViewController: UIViewController {
    @IBOutlet var imgView: UIImageView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading
            the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be
            recreated.
    }

    @IBAction func btnDrawLine(_ sender:
        UIButton) {
    }

    @IBAction func btnDrawRectangle(_ sender:
        UIButton) {
    }

    @IBAction func btnDrawCircle(_ sender:
        UIButton) {
    }

    @IBAction func btnDrawArc(_ sender: UIButton)
        {
    }

    @IBAction func btnDrawFill(_ sender:
        UIButton) {
    }
```

16-4 화면에 그림 그리기 기능 구현하기

1. 스텐더드 에디터로 화면 모드 수정하기

코딩을 위해 화면 모드를 수정한다. 오른쪽 윗부분에서 [Show the Standard editor]버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.

2. 선 그리기 기능 구현하기

우선 그림을 그리기 위하여 콘텍스트(Context)를 생성한다. 이때 콘텍스트는 그림을 그리는 도화지라고 생각하면된다.

다음 소스를 btnDrawLine 함수안에 입력한다.

```
1   @IBAction func btnDrawLine(_ sender: UIButton) {
2           UIGraphicsBeginImageContext(imgView.frame.size)
3           let context = UIGraphicsGetCurrentContext()!
4
5           // Draw Line
6           context.setLineWidth(2.0)
7           context.setStrokeColor(UIColor.red.cgColor)
8
9           context.move(to: CGPoint(x: 50, y: 50))
10          context.addLine(to: CGPoint(x: 250, y: 250))
11
12          context.strokePath()
13
14          // Draw Triangle
15          context.setLineWidth(4.0)
16          context.setStrokeColor(UIColor.blue.cgColor)
17
18          context.move(to: CGPoint(x: 150, y: 200))
19          context.addLine(to: CGPoint(x: 250, y: 350))
20          context.addLine(to: CGPoint(x:  50, y: 350))
21          context.addLine(to: CGPoint(x: 150, y: 200))
22          context.strokePath()
23
24          imgView.image =
                    UIGraphicsGetImageFromCurrentImageContext()
            UIGraphicsEndImageContext()
25  }
26
```

2 행은 콘텍스트를 이미지 뷰의 크기와 같게 생성한다.

3 행은 생성한 콘텍스트의 정보를 가져온다.

6 행은 이제 콘덱스트에 대한 여러 가지 설정을 한다. 가장 먼저 선을 굵기를 설정한다, 여기서는 2.0 으로 입력했지만 원하는 값으로 설정해도 된다.

7 행은 해당 콘택스트의 선 색상을 설정한다. 여기서는 빨간색으로 설정한다.

9 행은 그림을 그리기 위하여 시작 위치로 커서를 이동한다. (0,0)은 화면의 왼쪽 윗부분

의 지표이며, 여기서는 시작 위치를 (50,50)으로 지정한다.
10 행은 현재 위치에서 지정한 위치까지 선을 추가한다. 그리고 싶은 선은 계속 이어서 추가할 수 있다. 여기서는 임의로 (250, 250)을 입력한다.
12 행은 추가한 경로를 콘텍스트에 그린다.
15~22 행은 Draw Line 에서 사용한 방법으로 삼각형을 그린다. 색상은 파란색으로 설정하고, 삼각형의 꼭지점이 세개이므로 context.addLine좌표를 다르게 하여 세번 입력한다.
24 행은 현재 콘텍스트에 그려진 이미지를 가지고 와서 이미지 뷰에 나타낸다.
25 행은 그림 그리기를 끝낸다.

3. 사각형 그리기 기능 구현하기
다음은 화면에 사각형을 그리는 방법을 알아본다. 앞에서 배운 소스를 응용해서 임의로 입력한 좌표에서 시작하고 폭과 높이를 갖는 사각형을 추구한다. 다음 소스를 btnDrawRectangle 함수 안에 입력한다.

```
1   @IBAction func btnDrawRectangle(_ sender: UIButton) {
2           UIGraphicsBeginImageContext(imgView.frame.size)
3           let context = UIGraphicsGetCurrentContext()!
4
5           // Draw Rectangle
6           context.setLineWidth(2.0)
7           context.setStrokeColor(UIColor.red.cgColor)
8
9           context.addRect(CGRect(x: 50, y: 100, width: 200, height: 200))
            context.strokePath()
10
11          imgView.image = UIGraphicsGetImageFromCurrentImageContext()
12          UIGraphicsEndImageContext()
13  }
14
15
```

9 행은 X,Y 좌표(50, 100) 에서 시작하고 폭이 200 픽셀(px), 높이가 200 픽셀(px), 높이가 200 픽셀(px)인 사각형을 그린다. 이때 시작하는 좌표인 (50, 100)은 완성된 사각형에서 왼쪽 위의 꼭지점을 말한다.

4. 원 및 타원 그리기 기능 구현하기
이번에는 화면에 원 및 타원을 그리는 방법을 알아본다, 입력한 좌표에서 시작하고 폭과 높이를 갖는 직사각형에 내접한 타원을 추구한다. 원을 그리려면 폭과 높이를 같게 하면 된다. 다음 소스를 btnDrawCircle 함수 안에 입력한다.

```
1   @IBAction func btnDrawCircle(_ sender: UIButton) {
2           UIGraphicsBeginImageContext(imgView.frame.size)
3           let context = UIGraphicsGetCurrentContext()!
```

```
4
5               // Draw Ellipse
6               context.setLineWidth(2.0)
7               context.setStrokeColor(UIColor.red.cgColor)
8
9               context.addEllipse(in: CGRect(x: 50, y: 50, width: 200, height: 100))
10              context.strokePath()

                // Draw Circle
11              context.setLineWidth(5.0)
12              context.setStrokeColor(UIColor.green.cgColor)
13
14              context.addEllipse(in: CGRect(x: 50, y: 200, width: 200, height: 200))
15              context.strokePath()
16
17              imgView.image = UIGraphicsGetImageFromCurrentImageContext()
                UIGraphicsEndImageContext()
18      }
19
20
21
22
```

10 행은 X,Y 좌표(50, 50)에서 시작하고 폭이 200 픽셀, 높이가 100 픽셀인 사각형 안에 내접하는 타원을 그린다.

17 행은 X, Y 좌표(50, 200)에서 시작하고 폭이 200 픽셀, 높이가 200 픽셀인 사각형 안에 내접하는 원을 그린다. 폭과 높이를 같게 설정하면 원을 그릴 수 있다.

5. 호 그리기 기능 구현하기

이번에는 화면에 호를 그리는 방법에 대해서 알아본다. 호는 양 끝점과 반 지름을 이용해 그릴 수 있다. 다음 소스를 btnDrawArc 함수 안에 입력한다.

```
1   @IBAction func btnDrawArc(_ sender: UIButton) {
2               UIGraphicsBeginImageContext(imgView.frame.size)
3               let context = UIGraphicsGetCurrentContext()!
4
5               // Draw Arc
6               context.setLineWidth(5.0)
7               context.setStrokeColor(UIColor.red.cgColor)
8
9               context.move(to: CGPoint(x: 50, y: 50))
10              context.addArc(tangent1End: CGPoint(x: 200, y:50), tangent2End:
    CGPoint(x:200, y:200), radius: CGFloat(50))
11              context.addLine(to: CGPoint(x: 200, y: 200))
12
13
```

```
14              context.move(to: CGPoint(x: 10, y: 250))
15              context.addArc(tangent1End: CGPoint(x: 250, y:250), tangent2End:
16  CGPoint(x:100, y:400), radius: CGFloat(20))
                context.addLine(to: CGPoint(x: 100, y: 400))
17
18
19              context.strokePath()
20
21              imgView.image = UIGraphicsGetImageFromCurrentImageContext()
                UIGraphicsEndImageContext()
22      }
```

10 행은 현재 위치에서 두 개의 접점(200, 50), (200, 200)사이에 내접한 반지름이 50 인 호를 그리겠다는 뜻이다.

6. 채우기 기능 구현하기

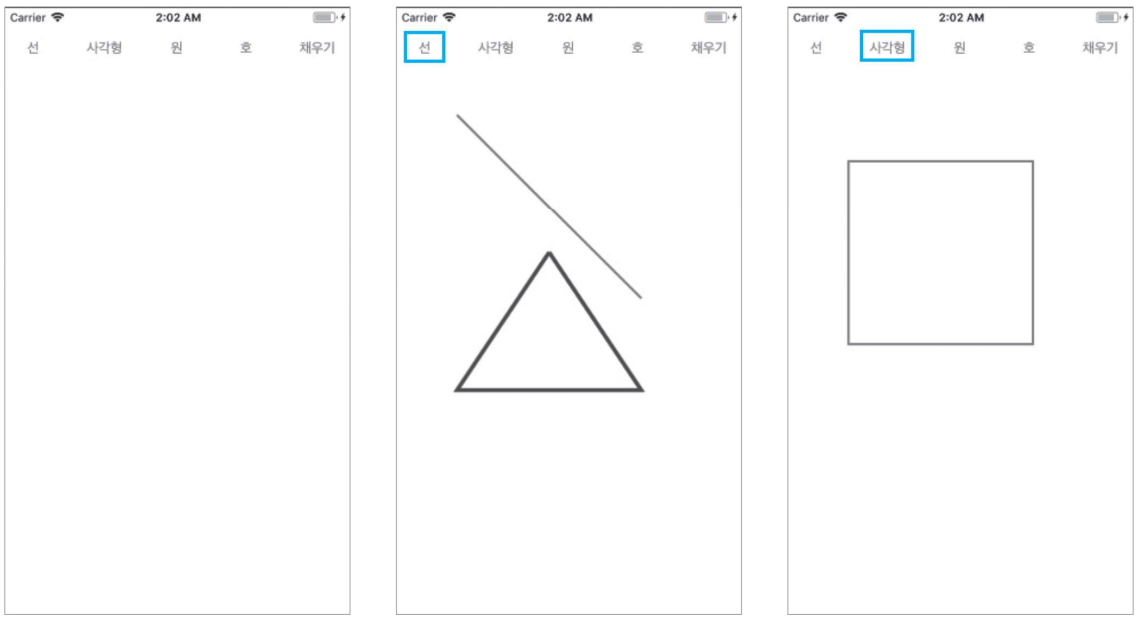
도형 내부를 특정 색상으로 채워 본다. 해당 콘텍스트의 채우기 색상을 설정한다. 이 때 사용하는 소스는 다음과 같다.

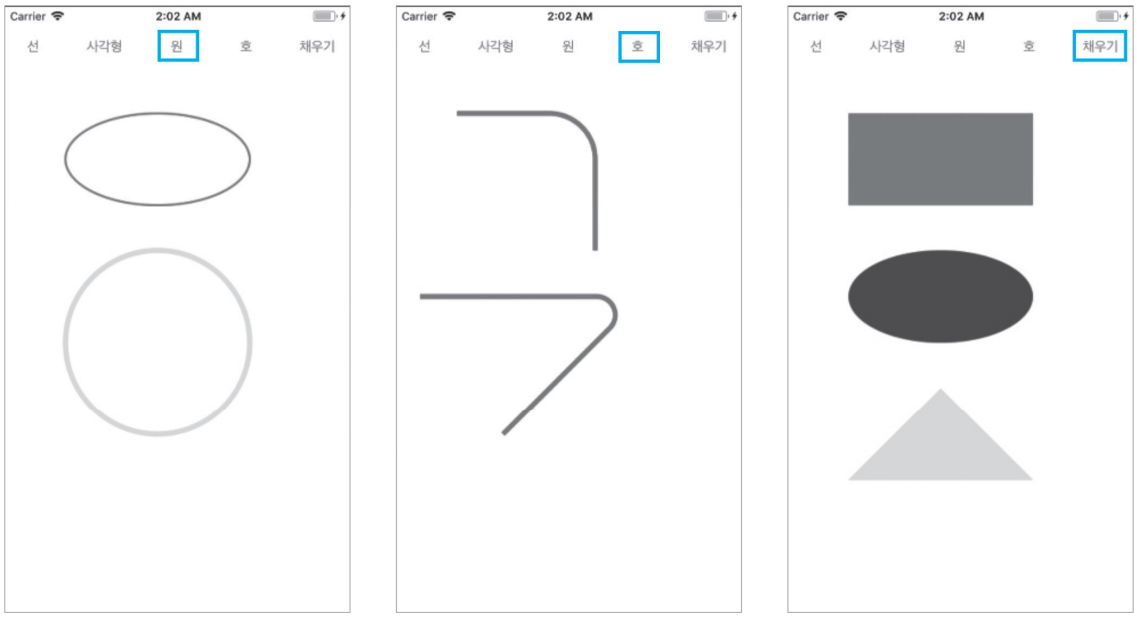
```
1   @IBAction func btnDrawFill(_ sender: UIButton) {
2               UIGraphicsBeginImageContext(imgView.frame.size)
3               let context = UIGraphicsGetCurrentContext()!
4
5               // Draw Rectangle
6               context.setLineWidth(1.0)
7               context.setStrokeColor(UIColor.red.cgColor)
8               context.setFillColor(UIColor.red.cgColor)
9
10              let rectangel = CGRect(x: 50, y: 50, width: 200, height: 100)
                context.addRect(rectangel)
11              context.fill(rectangel)
12              context.strokePath()
13
14              // Draw Circle
15              context.setLineWidth(1.0)
16              context.setStrokeColor(UIColor.blue.cgColor)
17              context.setFillColor(UIColor.blue.cgColor)
18
19              let circle = CGRect(x: 50, y: 200, width: 200, height: 100)
20              context.addEllipse(in: circle)
                context.fillEllipse(in: circle)
                context.strokePath()
21
22              // Draw Triangle
23              context.setLineWidth(1.0)
24              context.setStrokeColor(UIColor.green.cgColor)
25              context.setFillColor(UIColor.green.cgColor)
```

```
26
27              context.move(to: CGPoint(x: 150, y: 350))
28              context.addLine(to: CGPoint(x: 250, y: 450))
29              context.addLine(to: CGPoint(x:  50, y: 450))
30              context.addLine(to: CGPoint(x: 150, y: 350))
31              context.fillPath()
32              context.strokePath()
33
34              imgView.image = UIGraphicsGetImageFromCurrentImageContext()
35              UIGraphicsEndImageContext()
36  }
37
38
39
```

8, 18, 28 행은 도형의 내부를 색상으로 채운다. red 자리에 blue, green 등의 다른 색상 값이 들어갈 수 있다.
12 행은 사각형의 내부를 색상으로 채운다.
22 행은 원, 타원의 내부를 색상으로 채운다.
34 행은 선의 내부를 색상으로 채운다.

7. 결과 확인



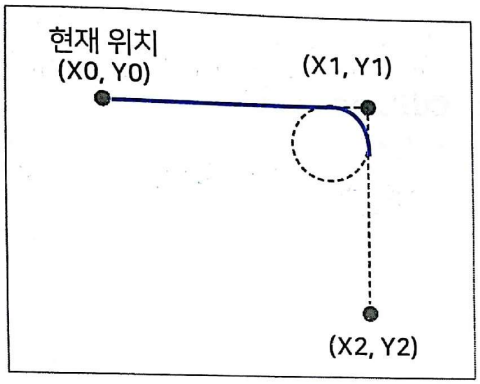


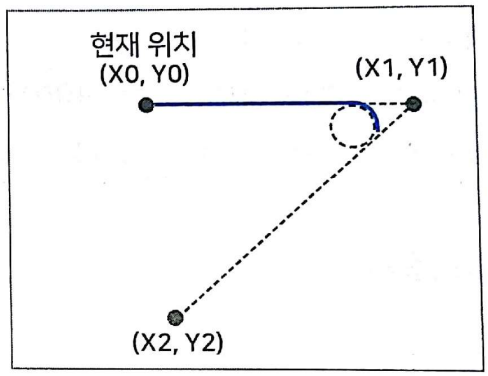
*** 호 그리기 ***
호는 두개의 접점과 반지름을 사용하여 그릴 수 있다. 어떤 원리로 호를 그리는지 좀 더 자세히 알아보자.

우선 호를 그리기 위한 함수는 다음과 같다.
addArc(tangent1End: CGPoint, tangent2End: CGPoint, radius: CGFloat)

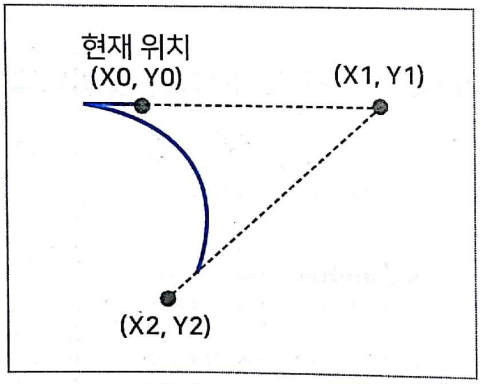
인수로는 그림을 그릴 콘텍스트 c 와 첫번째 접점 좌표인 tangent1End(x1, y1), 두번째 접점 좌표인 tangent2End(x2, y2) 그리고 반지름 radius 가 있다.

호를 그리는 방법은 다음과 같다, 우선 첫번째 접점 (x1, y1)을 기준으로 현재 위치 (x0, y0)와 두번째 접점 (x2, y2) 까지 가상의 선을 그린 후 그려진 선 안에 들어가는 반지름이 radius 인 가상의 원을 그린다. 그런다음 현재 위치 (x0, y0)에서 원의 첫번째 접선을 지나 두 번째 접선까지 그리면 된다.





만약 그리는 원의 반지름이 너무 크면 다음과 같이 그려지므로 주의하기 바란다.



전체 소스

```
//
//  ViewController.swift
//  DrawGraphics
//
//  Created by SoongMoo Rhee on 2018. 12. 6..
//  Copyright © 2018년  SoongMoo Rhee. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var imgView: UIImageView!
    override func viewDidLoad() {
        super.viewDidLoad()
```

```
        // Do any additional setup after loading the view, typically from a nib.
}

@IBAction func btnDrawLine(_ sender: UIButton) {
        // 그림을 그리기 위해서는 먼저 context가 생성되어야 한다.
        // ImageView크기 만큼 context생성
        UIGraphicsBeginImageContext(imgView.frame.size)
        // 생성된 Context정보를 가져온다.
        let context = UIGraphicsGetCurrentContext()!
        // Draw Line
        // 선의 굵기 사이즈
        context.setLineWidth(2.0)
        // 선의 색상을 정해준다.
        context.setStrokeColor(UIColor.red.cgColor)
        // 선의 시작 위치 지정
        context.move(to: CGPoint(x: 50, y: 50))
        // 선이 끝나는 위치를 지정해둔다.
        context.addLine(to: CGPoint(x: 250, y: 250))
        // 콘택스트에 선 추가
        context.strokePath()

        // Draw Triangle
        // 선의 굴기
        context.setLineWidth(4.0)
        // 선의 색
        context.setStrokeColor(UIColor.blue.cgColor)
        // 선의 시작점
        context.move(to: CGPoint(x: 150, y: 200))
        // 선이 끝나는 지점1
        context.addLine(to: CGPoint(x: 250, y: 350))
        // 선이 끝나는 지점2
        context.addLine(to: CGPoint(x:   50, y: 350))
        // 선이 끝나는 자점3
        context.addLine(to: CGPoint(x: 150, y: 200))
        // 콘택스트에 선 추가
        context.strokePath()
```

```
        //콘택스트의 이미지를 가져와 view에 출력
        imgView.image =
            UIGraphicsGetImageFromCurrentImageContext()
        // Context에서 그리기를 끝낸다.
        UIGraphicsEndImageContext()
    }

    @IBAction func btnDrawRectangle(_ sender: UIButton) {
        // ImageView크기 만큼 context생성
        UIGraphicsBeginImageContext(imgView.frame.size)
        // 생성된 Context정보를 가져온다.
        let context = UIGraphicsGetCurrentContext()!

        // Draw Rectangle
        // 선의 굴기
        context.setLineWidth(2.0)
        // 선의 색
        context.setStrokeColor(UIColor.red.cgColor)
        // 시작점을 x,y로 하고 너비는 200 높이는 200인 사가형
        context.addRect(CGRect(x: 50, y: 100, width: 200, height: 200))
        // 콘택스트에 사각형 추가
        context.strokePath()
        //콘택스트의 이미지를 가져와 view에 출력
        imgView.image = UIGraphicsGetImageFromCurrentImageContext()
        // Context에서 그리기를 끝낸다.
        UIGraphicsEndImageContext()
    }
    @IBAction func btnDrawCircle(_ sender: UIButton) {
        // ImageView크기 만큼 context생성
        UIGraphicsBeginImageContext(imgView.frame.size)
        // 생성된 Context정보를 가져온다.
        let context = UIGraphicsGetCurrentContext()!

        // Draw Ellipse
        // 선의 굴기
        context.setLineWidth(2.0)
        // 선의 색
```

```
        context.setStrokeColor(UIColor.red.cgColor)
        // 사각형의 시작점이 x,y각각 50인 위치에서 폭 200이고 높이가
100인 사각형안을 내접하는 타원
        context.addEllipse(in: CGRect(x: 50, y: 50, width: 200, height: 100))
        // 콘택스트에 원 추가
        context.strokePath()

        // Draw Circle
        // 선의 굴기
        context.setLineWidth(5.0)
        // 선의 색
        context.setStrokeColor(UIColor.green.cgColor)
        // 사각형의 시작점이 x,y각각 50인 위치에서 폭 200이고 높이가
200인 사각형안을 내접하는 원
        context.addEllipse(in: CGRect(x: 50, y: 200, width: 200, height: 200))
        // 콘택스트에 원 추가
        context.strokePath()
         //콘택스트의 이미지를 가져와 view에 출력
        imgView.image = UIGraphicsGetImageFromCurrentImageContext()
        // Context에서 그리기를 끝낸다.
        UIGraphicsEndImageContext()
    }
    @IBAction func btnDrawArc(_ sender: UIButton) {
        // ImageView크기 만큼 context생성
        UIGraphicsBeginImageContext(imgView.frame.size)
        // 생성된 Context정보를 가져온다.
        let context = UIGraphicsGetCurrentContext()!

        // Draw Arc
        // 선의 굴기
        context.setLineWidth(5.0)
        // 선의 색
        context.setStrokeColor(UIColor.red.cgColor)
        // 시작점 x,y
        context.move(to: CGPoint(x: 50, y: 50))
```

```
        // x1은 200, y1은 50   x1,y1에서 x2,y2까지의 위치 각각 200, 200 반지름이 50인 호
        context.addArc(tangent1End: CGPoint(x: 200, y:50), tangent2End: CGPoint(x:200, y:200), radius: CGFloat(50))
        // x1,y1부터 x2,y2까지 선 그리기 추가
        context.addLine(to: CGPoint(x: 200, y: 200))

        // 시작점 x,y
        context.move(to: CGPoint(x: 10, y: 250))
        // x1은 250, y1은 250   x1,y1에서 x2,y2까지의 위치 각각 100, 400 반지름이 20인 호
        context.addArc(tangent1End: CGPoint(x: 250, y:250), tangent2End: CGPoint(x:100, y:400), radius: CGFloat(20))
        // x1,y1부터 x2,y2까지 선 그리기 추가
        context.addLine(to: CGPoint(x: 100, y: 400))
        // 콘택스트에 원 추가
        context.strokePath()
         //콘택스트의 이미지를 가져와 view에 출력
        imgView.image = UIGraphicsGetImageFromCurrentImageContext()
        // Context에서 그리기를 끝낸다.
        UIGraphicsEndImageContext()
    }
    @IBAction func btnDrawFill(_ sender: UIButton) {
        UIGraphicsBeginImageContext(imgView.frame.size)
        let context = UIGraphicsGetCurrentContext()!

        // Draw Rectangle
        context.setLineWidth(1.0)
        context.setStrokeColor(UIColor.red.cgColor)
        context.setFillColor(UIColor.red.cgColor)

        let rectangel = CGRect(x: 50, y: 50, width: 200, height: 100)
        context.addRect(rectangel)
        context.fill(rectangel)
        context.strokePath()

        // Draw Circle
        context.setLineWidth(1.0)
```

```
        context.setStrokeColor(UIColor.blue.cgColor)
        context.setFillColor(UIColor.blue.cgColor)

        let circle = CGRect(x: 50, y: 200, width: 200, height: 100)
        context.addEllipse(in: circle)
        context.fillEllipse(in: circle)
        context.strokePath()

        // Draw Triangle
        context.setLineWidth(1.0)
        context.setStrokeColor(UIColor.green.cgColor)
        context.setFillColor(UIColor.green.cgColor)

        context.move(to: CGPoint(x: 150, y: 350))
        context.addLine(to: CGPoint(x: 250, y: 450))
        context.addLine(to: CGPoint(x:   50, y: 450))
        context.addLine(to: CGPoint(x: 150, y: 350))
        context.fillPath()
        context.strokePath()

        imgView.image = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
    }
}
```