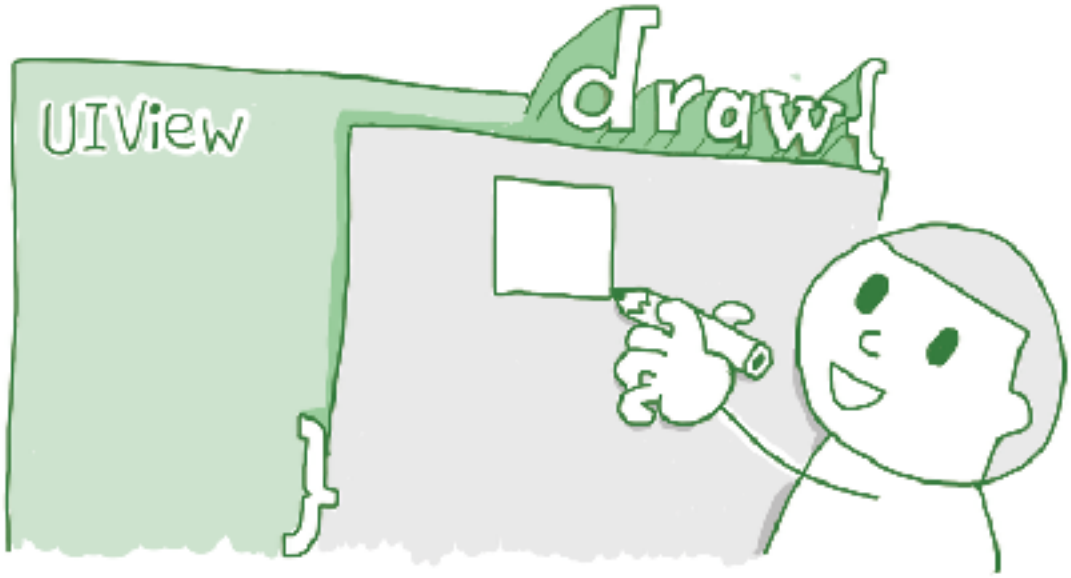
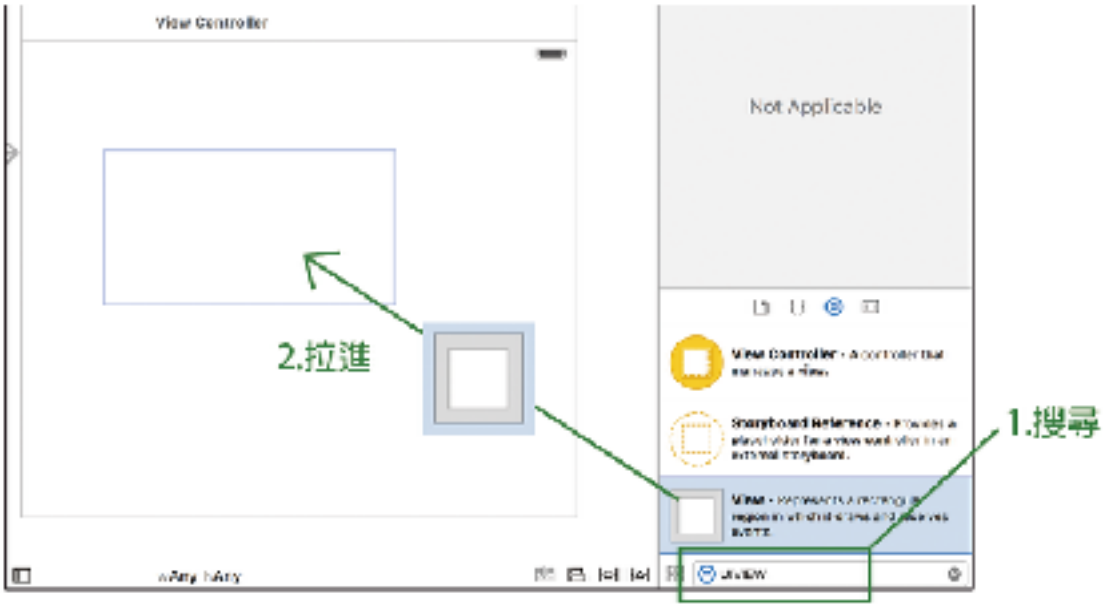
CHAPTER 繪製圖形



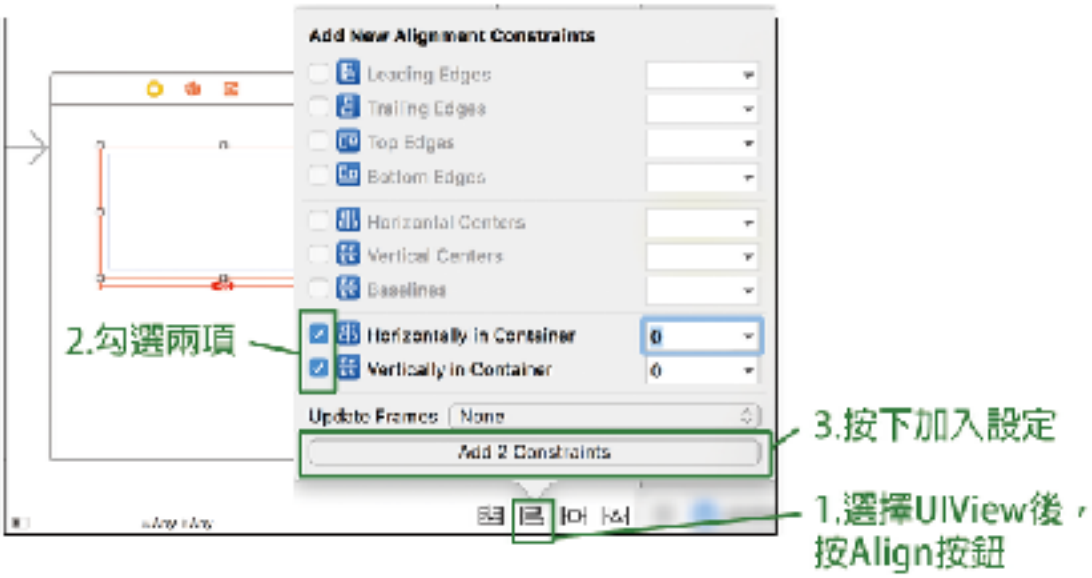
繪製直線

[問題]如何用程式碼繪製直線
[解答]取得繪圖情景後，使用CGContext的Move(to:)以及addLine(to:)繪製直線
[範例程式碼]HelloDrawLine
[取得繪圖情景]
要使用程式碼繪圖要先取得繪圖情景(Graphics Context)。本章大部分的繪圖範例將要圖像畫在UIView上。透過在UIView的draw方法，寫下畫圖的程式碼。過程記錄於下：
1.請先開一個新的專案，選擇SingleViewApplication，並且選擇適當的地方存檔。
2.打開Main.storyboard之後，在左下方搜尋UIView拉到ViewController的View上。

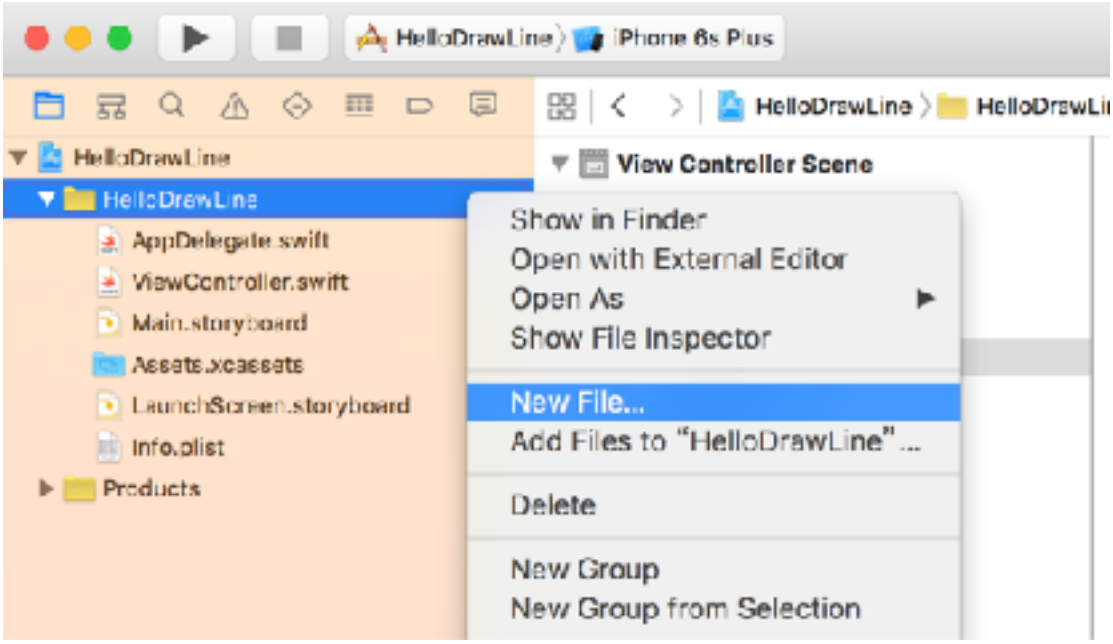


3.設定這個UIView的寬度和高度都固定在300點的大小。

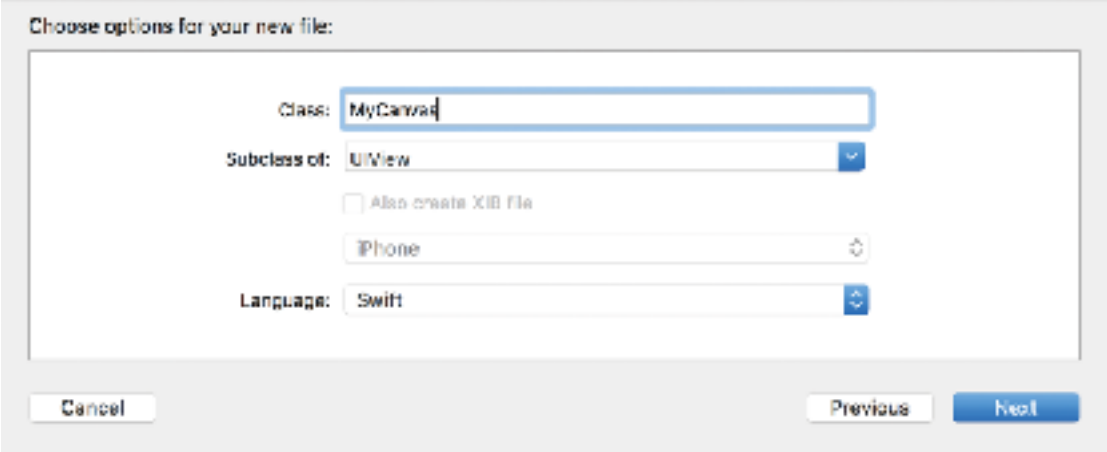
4.設定讓這個UIView置中。接下來要在這個UIView上繪圖。



5.選擇左邊欄檔案資料夾，按右鍵新增檔案。



6.選擇iOS > Source > Cocoa Touch Class。
7.新增一個UIView的子類別，範例中命名成MyCanvas。
(請注意，子類別的選項選取UIView)



8.回到Main.storyboard。選擇步驟2~4生成的UIView。將其類別設定為MyCanvas。之後就可以在MyCanvas的類別中，寫入程式碼，繪製圖像到畫面上置中、寬度為300、高度為300的UIView上。

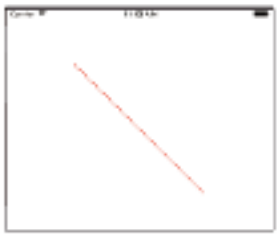
9.請打開MyCanvas.swift檔案，找到MyCanvas類別，裡面有一個註釋起來的draw方法。請把這個方法的註釋移除。
每次要繪製UIView類別的時候，程式就會呼叫draw這個方法。利用UIGraphicsGetCurrentContext方法取得繪圖情景，就可以撰寫程式碼在繪圖情景繪圖。

```
import UIKit

class MyCanvas: UIView {

    override func draw(_ rect: CGRect) {
        let context = UIGraphicsGetCurrentContext()
    }
}
```

[繪製線條]

使用程式碼繪製線條的方法很簡單，照著上面的方法取得繪圖情景後，請在draw函式中繼續寫進下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.move(to: CGPoint(x: 50, y: 50))
    context?.addLine(to: CGPoint(x: 250, y: 250))
    UIColor.red.set()
    context?.strokePath()
}
```



1.畫圖的時候，先用move(to:)，把畫筆移到UIView上座標(50,50)的地方。這個方法接受兩個參數：第一個是要移動位置的x座標，第二個是要移動位置的y座標。
2.把畫筆移動到座標(50,50)的地方準備下筆之後，呼叫addLine(to:)，從目前的座標畫一條線到座標(250,250)的地方。addLine(to:)方法也接受兩個參數：第一個是要畫線到另外一個端點的x座標，第二個是畫線到另外一個端點的y座標。
3.使用UIColor.red.set方法設定畫線與填色的顏色。範例中將顏色設定成紅

色。請依照您的情況做適當的改變。
4.呼叫strokePath方法，真的在UIView的繪圖情景上畫出一條線。

[接著畫出第二條線]
如果想要接著畫出第二條線的話，請照下面的範例，增加一行程式碼。

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.move(to: CGPoint(x: 50, y: 50))
    context?.addLine(to: CGPoint(x: 250, y: 250))
    context?.addLine(to: CGPoint(x: 50, y: 250))
    UIColor.red.set()
    context?.strokePath()
}
```

加入這一行

之前的程式碼，在畫線畫到座標(250,250)之後，如果要接著畫線，則再次呼叫addLine方法，就會從上一條線結束的座標，繼續畫一條線到座標(50,250)的地方。addLine方法接受兩個參數，分別是畫線到另外一個端點的x與y座標。

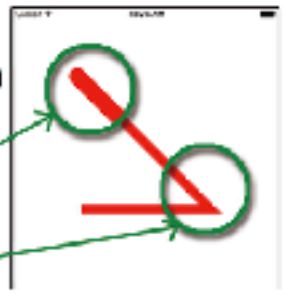
[線條粗細的設定]
之前的程式碼畫了兩條線段。不論是畫出一條線或是多條線段，如果想要調整線段的粗細，可以使用setLineWidth方法。這個方法接受一個參數：這個參數則是線段的粗細。範例中把線段的粗細調成15，所以可以看到模擬器中，線段的粗細就變粗了很多。

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.move(to: CGPoint(x: 50, y: 50))
    context?.addLine(to: CGPoint(x: 250, y: 250))
    context?.addLine(to: CGPoint(x: 50, y: 250))
    context?.setLineWidth(15)
    UIColor.red.set()
    context?.strokePath()
}
```

設定線條粗細

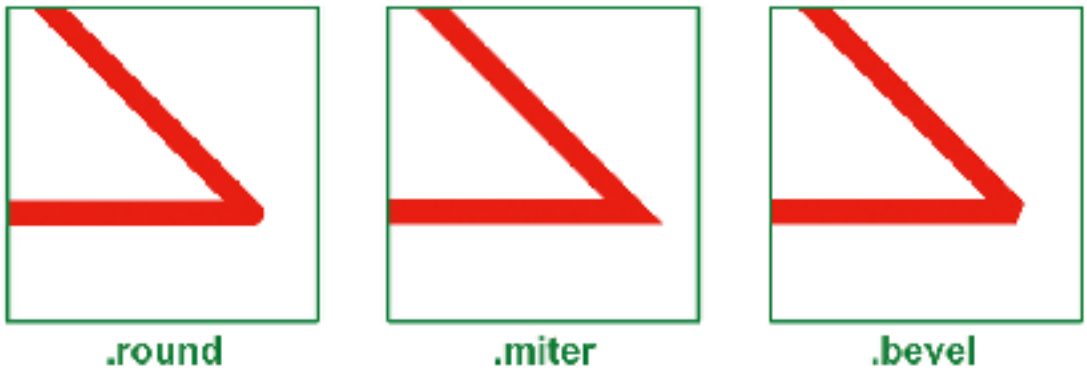
[設定線條轉折處與線條的端點]
線段還可以設定轉折處與端點的樣式，請參考下面的程式碼：

```swift
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.move(to: CGPoint(x: 50, y: 50))
    context?.addLine(to: CGPoint(x: 250, y: 250))
    context?.addLine(to: CGPoint(x: 50, y: 250))
    context?.setLineWidth(15)
    //線條的結尾，可以選 Round,Butt，以及Square
    context?.setLineCap(.round)
    //線條的轉折處，可以設定 Round、Miter、以及Bevel
    context?.setLineJoin(.bevel)
    UIColor.red.set()
    context?.strokePath()
}
```

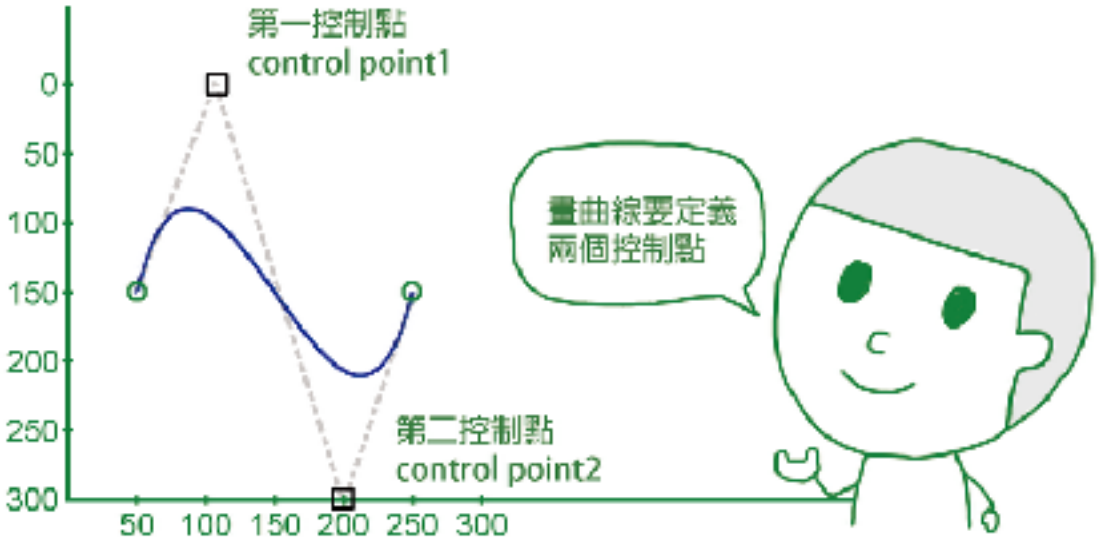


1.使用setLineCap方法，可以設定線段的端點樣式。端點的樣式有下面三種，範例中選擇圓端點(.round)。

2.使用setLineJoin方法，可以設定線條轉折處的端點樣式。端點的樣式有下面三種，範例中選擇斜角端點(.bevel)。



繪製曲線



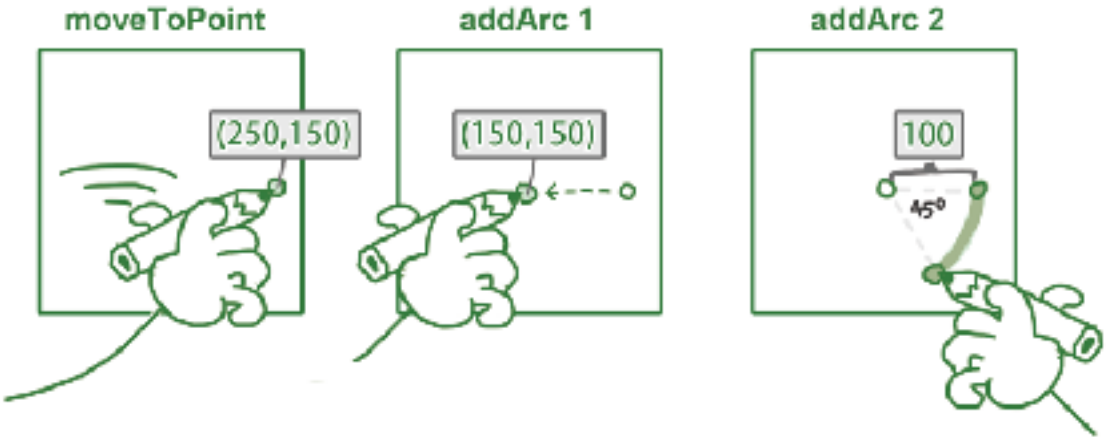
[問題]如何用程式碼繪製曲線
[解答]取得繪圖情景後，使用CGContextMoveToPoint以及CGContextAddCurveToPoint繪製曲線
[範例程式碼]HelloDrawCurveLine
[過程解說]
1.請照著本章一開始[取得繪圖情景]的作法，開啟新專案、在畫面上加入UIView、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，準備畫出曲線：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext() //取得繪圖情景
    context?.move(to: CGPoint(x: 50, y: 150))       //移到線段的第一個點
    //用CGContextAddCurveToPoint 畫曲線
    let endPoint = CGPoint(x: 250, y: 150)
    let controlPoint1 = CGPoint(x: 100, y: 0)
    let controlPoint2 = CGPoint(x: 200, y: 300)
    context?.addCurve(to: endPoint,
                      control1: controlPoint1, control2: controlPoint2)
    UIColor.blue.set()                          //設定線段顏色
    context?.strokePath()                       //畫出線段
}
```

2.先用UIGraphicsGetCurrentContext方法得到繪圖情景、接著使用move方法，設定曲線的第一個端點。範例中，曲線第一個端點座標為(50,150)
3.使用addCurve方法就可以繪製曲線。如圖示，繪製曲線需要設定兩個控制

點，於是CGContextAddCurveToPoint這個方法需要三個參數：第一個是曲線另一個端點的x與y座標、第二個與第三個是兩個控制點分別的x與y座標。
4.設定線條顏色、使用strokePath方法，就可以畫出曲線了。

繪製弧線



[問題]如何用程式碼繪製弧線
[解答]取得繪圖情景後，使用addArc方法繪製曲線
[範例程式碼]HelloDrawArc
[過程解說]
1.請照著本章一開始[取得繪圖情景]的作法，開啟新專案、在畫面上加入UIView、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，準備畫出弧線：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext() //取得繪圖情景
    context?.move(to: CGPoint(x: 250, y: 150))      //移到線段的第一個點
    let centerPoint = CGPoint(x: 150, y: 150)
    //呼叫 addArc 方法畫曲線
    context?.addArc(center: centerPoint, radius: 100,
                startAngle: 0, endAngle: CGFloat(45.0 * M_PI)/180.0,
                    clockwise: false)
    UIColor.orange.set()                    //設定線段顏色
    context?.setLineWidth(5)               //設定線段粗細
    context?.strokePath()                   //畫出線段
}
```

2.先用UIGraphicsGetCurrentContext方法得到繪圖情景、接著使用move方法，設定弧線的第一個端點。範例中，弧線第一個端點座標為(250,150)
3.使用addArc方法就可以繪製弧線。addArc方法接受五個參數：第一個是要畫弧線其圓心的x與y座標。第二個參數是要畫弧線其圓心的半徑。第三個參數是弧線開始的弧度、第四個參數是弧線結束的弧度。第五個參數則是設定繪製弧

線時，是以順時針或是以逆時針的方向繪製弧線。此參數填false是順時針、填true則是以逆時針方向繪製弧線。
範例中，弧線圓心座標為(150,150)，弧線半徑是100，角度則是從0度畫到45度。繪製弧線時以順時針的方式繪製弧線。
4.設定線條顏色、使用strokePath方法，就可以畫出曲線了。不過範例裡，特別以setLineWidth方法，把弧線的粗細加粗一點，讓我們可以更清楚地看到畫出的弧線。

繪製虛線

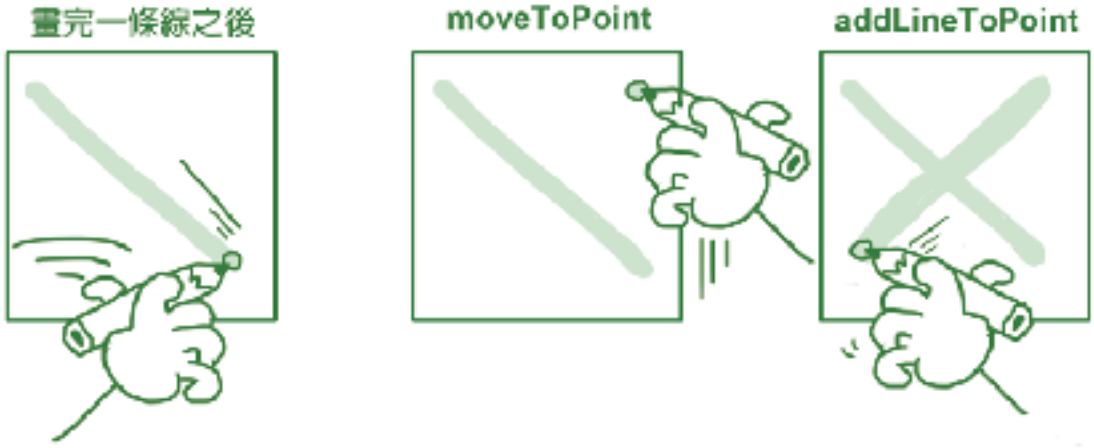
[問題]如何用程式碼繪製虛線
[解答]繪製線段時，使用CGContextSetLineDash方法來設定虛線
[範例程式碼]HelloDashLine
[過程解說]
1.請照著本章一開始[繪製直線]的作法，畫出一條直線。
2.不管是直線、曲線或是弧線，如果要設定以虛線的方式畫出線段的話，請參考下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()          //取得繪圖情景
    context?.move(to: CGPoint(x: 50, y: 150))     //移到線段的第一個點
    context?.addLine(to: CGPoint(x: 250, y: 150))        //畫出一條線
    context?.setLineDash(phase: 0, lengths: [10,20])
    context?.setLineWidth(5)                             //設定線段粗細
    UIColor.red.set()                                    //設定線段顏色
    context?.strokePath()                                  //畫出線段
}
```

3.使用setLineDash方法，可以設定以虛線的方式畫出線段。這個方法接受兩個參數：第一個要從線段的那邊開始畫虛線、第二個參數是一個陣列，裡面記錄

了虛線的長度與間隔空間。
在範例程式碼中，第一個參數設定0，意思是從線段的一開始就要畫虛線。第三個參數規定了虛線的長度與各個虛線的間隔空間分別是10與20。
4.如果把設定虛線的程式碼改成setLineDash(0, [5, 10, 10, 10])的話，就會以不同的方式畫虛線：由於第二個參數有四個數值，所以虛線會以先畫長度5的線段、空10點的距離，再畫長度10的線段、空10點的距離這樣的模式畫虛線。

繪製第二條線



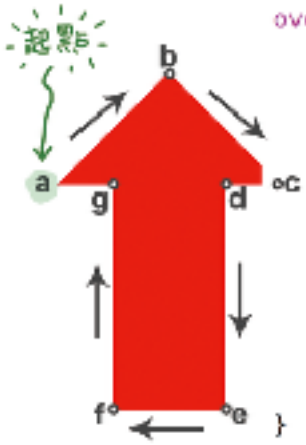
[問題]如何用程式碼繪製多條線段
[解答]多次呼叫move以及addLine方法繪製直線，就可以畫出多條線段
[範例程式碼]HelloMultiLine
[過程解說]
繪製多條線段，請參考下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext() //取得繪圖情景
    context?.move(to: CGPoint(x: 50, y: 50))        //移到線段的第一個點
    context?.addLine(to: CGPoint(x: 250, y: 250))  //畫出一條線

    context?.move(to: CGPoint(x: 250, y: 50)) //移到第二條線段的第一個點
    context?.addLine(to: CGPoint(x: 50, y: 250))  //再畫出一條線

    context?.setLineWidth(15)           //設定線段寬度
    UIColor.red.set()                     //設定線段顏色
    context?.strokePath()                 //畫出線段
}
```

想要畫出多條線段的話，只要在畫出線段之後，使用move方法，把目前繪圖的座標移到下一條線段的端點，繼續呼叫addLine方法，就可以畫出另外一條線段。

繪製圖形(不規則形狀)

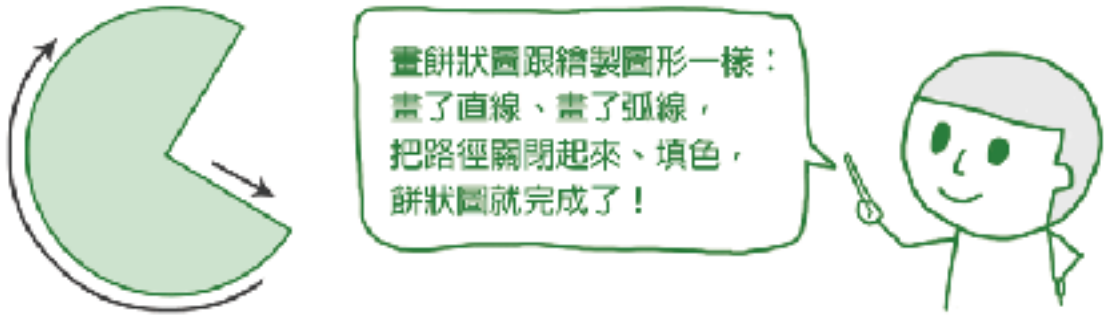
[問題]如何用程式碼繪製圖形
[解答]取得繪圖情景後，畫出多條線段，最後呼叫closePath方法，就可以繪製圖形
[範例程式碼]HelloDrawShape
[過程解說]
請參考下面的程式碼：




```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    context?.move(to: CGPoint(x: 50, y: 100))      //a
    context?.addLine(to: CGPoint(x: 150, y: 0))    //b
    context?.addLine(to: CGPoint(x: 250, y: 100)) //c
    context?.addLine(to: CGPoint(x: 200, y: 100)) //d
    context?.addLine(to: CGPoint(x: 200, y: 300)) //e
    context?.addLine(to: CGPoint(x: 100, y: 300)) //f
    context?.addLine(to: CGPoint(x: 100, y: 100)) //g
    context?.closePath()                           //關閉路徑
    UIColor.red.set()                              //設定顏色
    //CGContextStrokePath(context)                 //畫出線段
    context?.fillPath()                            //填色
}
```

1.在範例程式碼中，先呼叫move方法、把繪圖的座標移到a點，然後依序呼叫多次addLine方法，從a點畫線到b點、b點到c點、到d點、e點、f點，然後畫線到了g點。
2.畫線到g點之後，呼叫closePath方法，就把畫筆連結回到a點、關閉了這個圖形路徑。
3.呼叫UIColor.red.set方法設定填色為紅色之後，可以呼叫strokePath方法把這個圖形的邊框畫出來。本範例程式碼是用fillPath方法把圖形整個填成紅色。

繪製餅狀圖

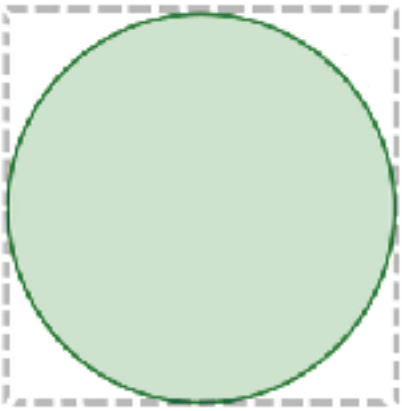


[問題]如何以程式碼繪製工作報表中的餅狀圖
[解答]結合前面幾個單元畫出直線、畫出弧線，與繪製圖形的方法，就可以繪製出餅狀圖。
[範例程式碼]HelloPieChart
[過程解說]
請參考下面的程式碼：

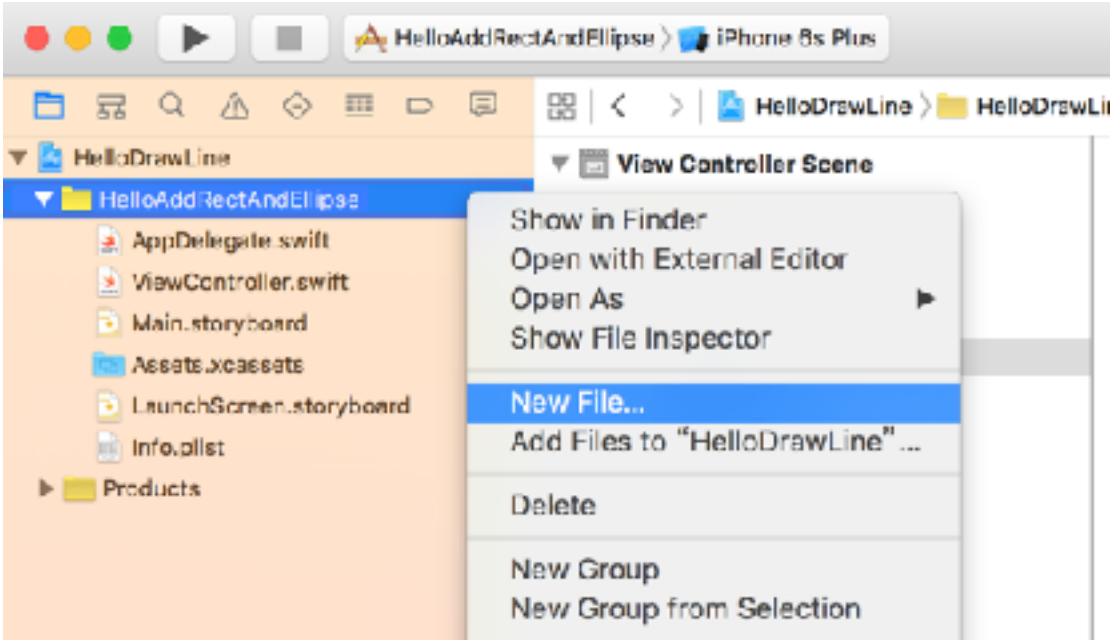
```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext() //取得繪圖情景
    context?.move(to: CGPoint(x: 150, y: 150))  //移到第一個圖形的第一個點
    let centerPoint = CGPoint(x: 150, y: 150)   //每次畫弧型的圓心
    context?.addArc(center: centerPoint, radius: 100, startAngle: 0,
        endAngle: CGFloat(300.0 * M_PI)/180.0, clockwise: true)
    context?.closePath()          //畫出弧線之後，關閉第一個圖形的路徑
    UIColor.blue.set()            //設定填色為藍色
    context?.fillPath()

    context?.move(to: CGPoint(x: 150, y: 150))  //移到第二個圖形的第一個點
    context?.addArc(center: centerPoint, radius: 100, startAngle: 0,
        endAngle: CGFloat(300.0 * M_PI)/180.0, clockwise: false)
    context?.closePath()          //畫出弧線之後，關閉第二個圖形的路徑
    UIColor.magenta.set()         //設定填色為粉紅色
    context?.fillPath()
}
```

1.在範例程式碼中，先畫出弧線、關閉路徑，將路徑填色，餅狀圖就完成了。
2.畫完了第一個餅狀圖之後，為了示範，在程式碼中又以粉紅色畫出餅狀圖的另外一個部分。這兩個圖形繪製路徑的方法很類似，只有最後一個參數不同。最後一個參數填false代表以順時針的方向繪製弧線；最後一個參數填true，代表以逆時針的方向繪製弧線。

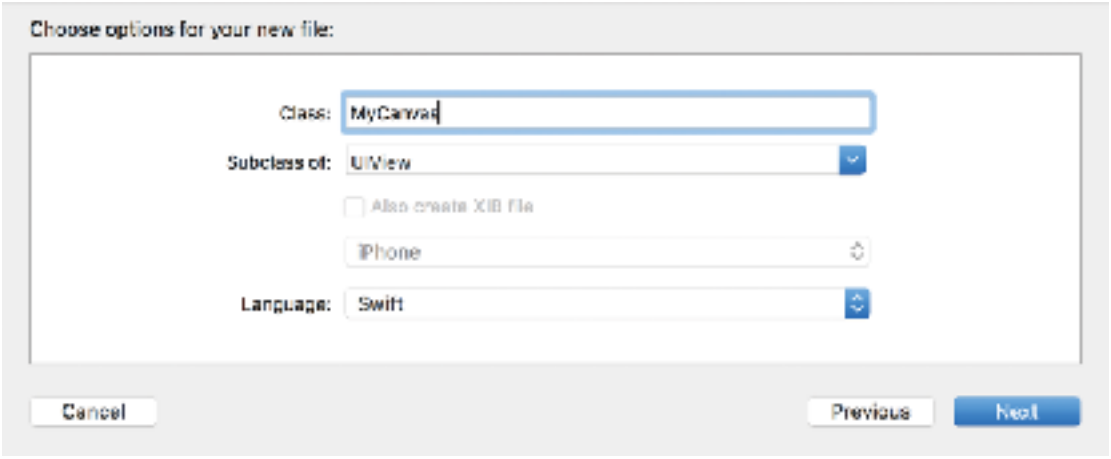
繪製方形與圓形



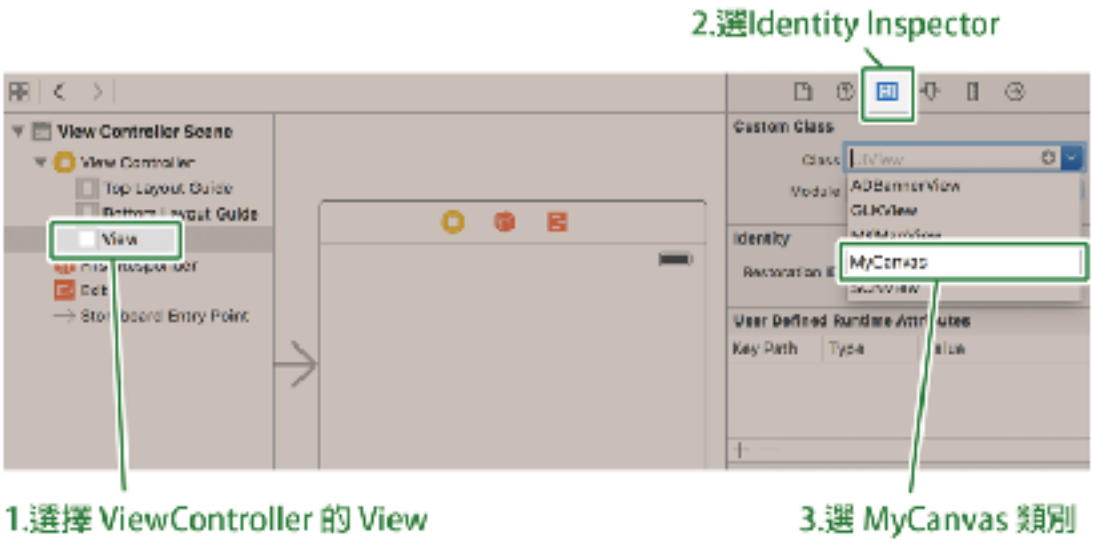
[問題]如何以程式碼繪製方形與圓形
[解答]使用addRect方法可以畫出方形；呼叫addEllipse方法可以畫出圓形或是橢圓形。
[範例程式碼]HelloAddRectAndEllipse
[以ViewController的View當作繪製圖形的畫布]
一直到本單元，各個範例都在ViewController的View上加入新的UIView子類別，繪製圖形的程式碼都寫在這個UIView子類別的draw方法中。其實繪製的圖形，也可以以ViewController類別的View當作畫布、在上面繪製圖形。以下是相關的步驟記錄：
1.請先開一個新的專案，選擇SingleViewApplication，並且選擇適當的地方存檔。
2.選擇左邊欄檔案資料夾，按右鍵新增檔案。



3.選擇iOS > Source > Cocoa Touch Class。
4.新增一個UIView的子類別，範例中命名成MyCanvas。
(請注意，子類別的選項選取UIView)



5.回到Main.storyboard。選擇ViewController的View。將其類別設定為MyCanvas。之後就可以在MyCanvas的類別中，寫入程式碼，繪製圖像到ViewController的View上。



6.請打開MyCanvas.swift檔案，找到MyCanvas類別，裡面有一個註釋起來的draw方法。請把這個方法的註釋移除。

```
import UIKit

class MyCanvas: UIView {

    override func draw(_ rect: CGRect) {
        let context = UIGraphicsGetCurrentContext()
    }
}
```

每次要繪製UIView類別的時候，程式就會呼叫draw這個方法。設定MyCanvas為UIView的子類別、連結MyCanvas類別到storyboard中UIViewController的View後，在MyCanvas的draw方法裡，利用UIGraphicsGetCurrentContext方法取得繪圖情景，就可以撰寫程式碼在繪圖情景繪圖。

[繪製方形與圓形]
請參考下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let context = UIGraphicsGetCurrentContext()
    //設定方形的區域
    let rectArea = CGRect(x: 50, y: 50, width: 100, height: 100)
    context?.addRect(rectArea)  //加入方形

    //設定圓形的區域
    let ellipseArea = CGRect(x: 50, y: 200, width: 100, height: 100)
    context?.addEllipse(in: ellipseArea) //加入圓形

    UIColor(red: 0.34,green:0.43,blue:0.68,alpha:1).set()    //設定顏色
    context?.fillPath()
}
```

1.畫方形的話，先使用CGRect的初始化方法設定方形的區域。CGRectMake方法接受四個參數：第一個是方形區域的x座標、第二個參數是方形區域的y座標、第三個參數是方形區域的寬度，最後的參數是方形區域的高度。
2.定義好要畫的方形區域後，呼叫addRect方法就可以標出要畫的方形區域。
3.範例的程式碼會在x座標50、y座標50的地方，畫出寬度為100、高度為100的正方形。如果要畫出長方形的話，則只需在定義方形區域的過程中，改變方形區域的寬度與高度即可。
4.畫圓形的話，先使用CGRect的初始化方法設定圓形的區域。請參考本單元的附圖：雖然要畫圓形，但是也可以用方形來限制圓形的區域。
5.定義好要畫圓形的方形區域後，呼叫AddEllipse方法就可以標式要畫出的圓形區域。
6.範例的程式碼會在x座標50、y座標200的地方，畫出寬度為100、高度為100的圓形。如果要畫出橢圓形的話，則只需在定義方形區域的過程中，改變方形區域的寬度與高度即可。
7.設定完成之後，呼叫 fillPath，就把步驟2與5加入的圖形填入顏色。

繪製文字

[問題]如何繪製文字
[解答]使用NSString中的draw方法，就可以繪製文字
[範例程式碼]HelloDrawString
[過程解說]
1.請照著之前[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，準備畫出文字：

```swift
override func draw(_ rect: CGRect) {
    let fontName = "Arial-BoldMT"     //設定字型名稱，之後用此名稱設定UIFont
    let ArialBold = UIFont(name: fontName, size: 30.0)
    let swiftString:NSString = "I Love Swift"          //這是要顯示的文字

    //用draw方法繪製文字
    swiftString.draw(at: CGPoint(x: 100, y: 100),
        withAttributes: [NSFontAttributeName: ArialBold!,
            NSForegroundColorAttributeName: UIColor.red])
}
```

2.範例程式碼中，先設定字型名稱，之後用此名稱產生出UIFont。產生UIFont的方法中接受兩個參數：第一個參數是字型的名稱，第二個參數則是字型的大小。
3.把要顯示的文字存在NSString型別的常數swiftString中。
4.使用NSString型別的draw方法，就可以繪製文字。該方法接受兩個參數：第一個參數是要繪製文字的座標，範例中使用CGPoint初始化方法，設定文字的座標為(100,100)；NSString的draw方法中第二個參數是一個Dictionary，設定繪製文字相關的資訊。範例中設定文字的字型以及文字的顏色。

繪製圖片

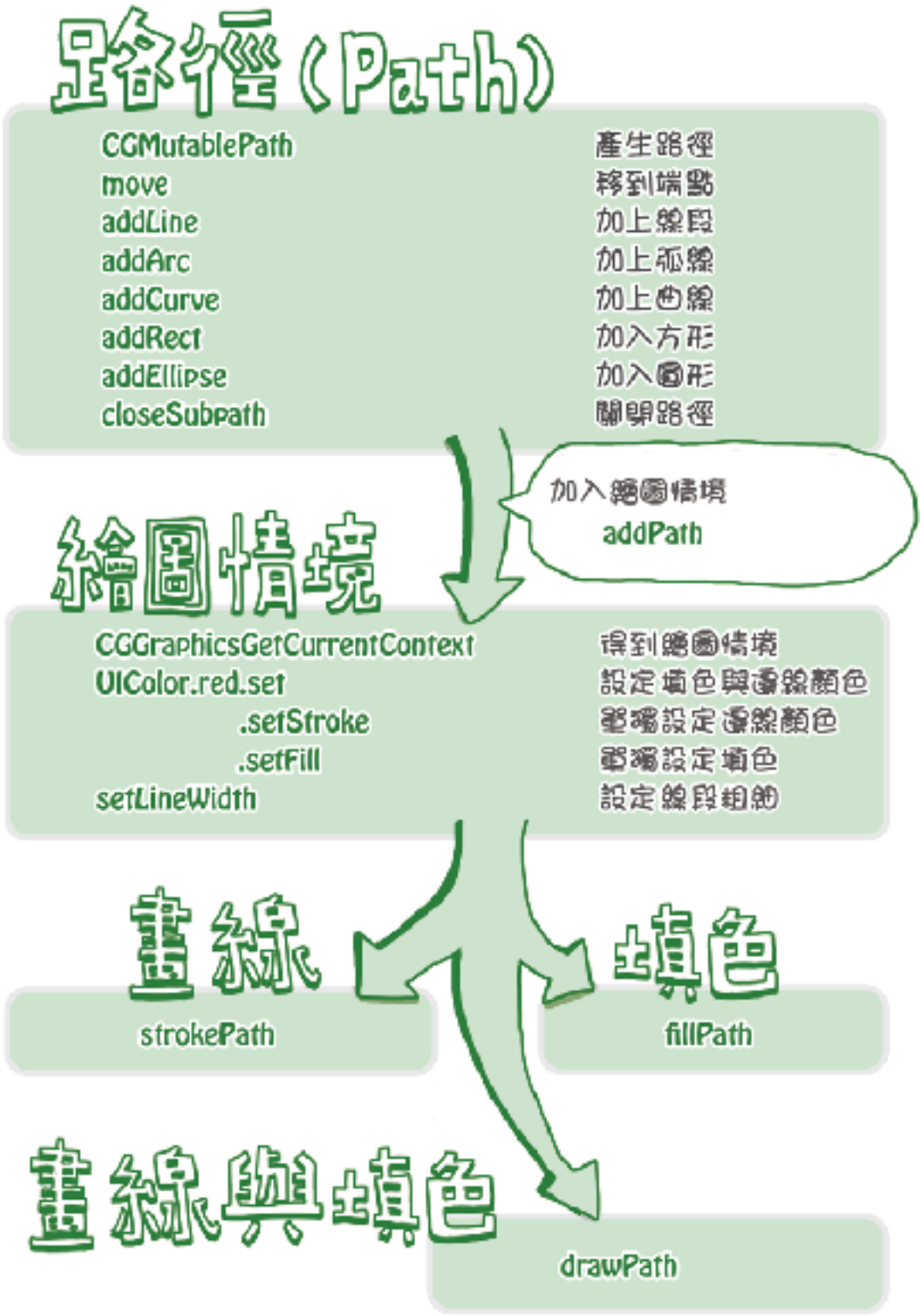
[問題]如何繪製圖片
[解答]使用UIImage的draw(at:)方法或是draw(in:)方法，都可以繪製圖片
[範例程式碼]HelloDrawImage
[過程解說]
1.在繪製圖片之前，先把要顯示的圖片匯入的專案中。這個範例裡，匯入了檔名為「Beautiful」的圖片。

2.請照著之前[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，準備繪製圖片：

```
override func draw(_ rect: CGRect) {
    let image = UIImage(named: "Beautiful")              //讀入圖片
    image?.draw(at: CGPoint(x: 0, y: 0))                //第一種方法
    image?.draw(in: CGRect(x: 0, y: 350, width: 300,
                                      height: 300))      //第二種方法
}
```

3.使用產生UIImage的方法把步驟1放入專案的圖片存在常數image裡面。
4.使用UIImage的draw(at:)方法可以繪製圖片。這個方法接受一個參數，就是圖片要繪製的位置座標。本範例設定繪製在座標(0,0)的地方繪製圖片。
5.除了使用UIImage的draw(at:)方法以外，繪製圖片也可以使用UIImage的draw(in:)方法來繪製圖形。draw(in:)方法接受一個參數，就是繪製圖形的範圍。範例程式碼設定要在x座標0、y座標350的地方，、繪製寬度300、高度300的圖片。

以路徑(Path)繪製圖形與線條



[問題]使用路徑(Path)來繪製線條與圖形
[解答]先設定路徑、把路徑加入繪圖情景之後，使用drawPath或其他畫線與填色的方法繪製各種圖形
[範例程式碼]HelloPath
[以路徑的方式繪製線段]
1.請照著之前[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，準備以路徑的方式繪製線段：

```
override func draw(_ rect: CGRect) {
    let path = CGMutablePath()                    //產生路徑
    path.move(to: CGPoint(x: 50, y: 50))          //移到線段端點
    path.addLine(to: CGPoint(x: 250, y: 250))     //加上線段

    let context = UIGraphicsGetCurrentContext() //得到繪圖情境
    context?.addPath(path)                        //把路徑加入繪圖情境

    UIColor.red.setStroke()                       //設定顏色
    context?.setLineWidth(10)                     //設定線段粗細
    context?.strokePath()                         //畫出線段
    //context?.drawPath(using: .stroke)           //畫線段的另外一個方法
}
```

2.使用路徑的方式繪製圖形或是線條的話，要先用CGMutablePath方法產生路徑。範例中，把這個路徑存在常數path中。
3.接下來呼叫CGPath的move方法，設定線段的端點的座標為(50,50)。move方法接受一個參數：此參數為線段端點的x與y座標。
4.然後呼叫addLine方法加上線段。此方法接受一個參數：此參數為線段另外一個端點的x與y座標。
5.呼叫UIGraphicsGetCurrentContext方法得到繪圖情境，使用addPath方法，把要繪製的路徑加入到繪圖情境中。
6.設定顏色、線段粗細之後，呼叫strokePath方法，就可以畫出線段了。範例中使用setStroke方法設定畫線段的顏色是紅色。除了setStroke以外，還可以使用setFill方法設定填色，或是使用set方法同時設定填色與邊線的顏色。
7.除了使用strokePath方法以外，另外也可以呼叫註釋起來的drawPath方法。drawPath方法接受一個參數：此參數是繪製時的的繪製模式。範例中模式設定為「.stroke」，於是只會畫出線段。

[使用路徑跟沒有使用路徑繪製線段的分別]

使用路徑繪製圖形或線條跟之前沒有使用路徑、直接繪製圖形跟線條很像。只不過呼叫的方法從呼叫CGContext的move方法，改成呼叫CGPath的move方法；原本呼叫CGContext的addLine方法，也改為呼叫CGPath的addLine方法。

[以路徑的方式繪製形狀]

如果要以路徑的方式繪製形狀，請把上面繪製線段的drawRect方法改成下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let path = CGMutablePath()
    path.move(to: CGPoint(x: 0, y: 110))
    path.addLine(to: CGPoint(x: 100, y: 90))
    path.addLine(to: CGPoint(x: 150, y: 0))
    path.addLine(to: CGPoint(x: 200, y: 90))
    path.addLine(to: CGPoint(x: 300, y: 110))
    path.addLine(to: CGPoint(x: 230, y: 185))
    path.addLine(to: CGPoint(x: 243, y: 286))
    path.addLine(to: CGPoint(x: 150, y: 243))
    path.addLine(to: CGPoint(x: 57, y: 286))
    path.addLine(to: CGPoint(x: 70, y: 185))
    path.closeSubpath()                              //關閉路徑

    let context = UIGraphicsGetCurrentContext()//得到繪圖情境
    context?.addPath(path)                           //把路徑加入繪圖情境

    UIColor.yellow.setFill()                         //設定顏色
    context?.fillPath()                              //填色
    //context?.drawPath(using: .fill)                //填色的另外一個方法
}
```

1.使用CGMutablePath方法產生路徑、呼叫CGPath的move方法，設定線段的端

點的座標之後，多次利用CGPath的addLine方法畫出多條線，最後用closeSubpath方法把繪圖路徑封閉起來，就定義好了一個繪圖路徑。
2.呼叫UIGraphicsGetCurrentContext方法得到繪圖情境，並用CGContext的qddPath方法，把要繪製的路徑加入到繪圖情境中。
3.設定顏色、線段粗細之後，呼叫CGContext的fillPath方法，就可以畫出圖形了。範例中使用setFill方法設定填色的顏色是黃色。
4.除了使用CGContext的fillPath方法以外，也可以使用在路徑中填色的作法，呼叫註釋起來的CGContext的drawPath方法、參數設定為「.fill」，就可以在路徑中填色。
5.請參考隨書附贈的範例程式碼，其中還示範了以路徑的方式繪製曲線、弧線、方形與圓形的方法。

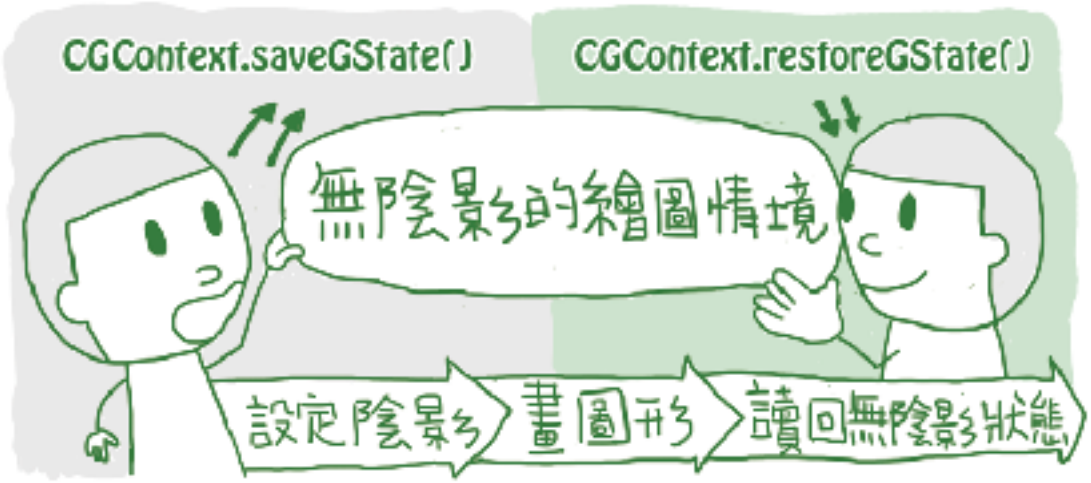
幫圖形加上陰影

[問題]如何幫繪製的圖形加入陰影
[解答]使用setShadow方法，就可以幫繪製的圖形加上陰影
[範例程式碼]HelloAddShadow
[過程解說]
1.請照著之前[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，寫下下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let path = CGMutablePath()                          //產生路徑
    let rectArea =
        CGRect(x: 50, y: 50, width: 200, height: 100)//設定方形範圍
    path.addRect(rectArea)                              //加入方形到路徑中
    let context = UIGraphicsGetCurrentContext()         //得到繪圖情境
    context?.addPath(path)                             //把路徑加入繪圖情境
    UIColor.magenta.setFill()                           //設定顏色
    context?.setShadow(                                 //設定陰影
        offset: CGSize(width: 10,height: 10),
        blur: 20,
        color: UIColor.lightGray.cgColor)
    context?.fillPath()                                 //填色
}
```

2.上面方法中，要畫出一個粉紅色的方形。過程中，在先設定要畫出方形的路徑，在真的幫路徑填色之前，呼叫了CGContext的setShadow方法，這個方法可以幫繪製的圖形加上陰影。此方法接受四個參數：第一個是CGSize型別的參數，用來設定陰影偏移的量。範例中設定陰影往右偏移10、往左偏移10。setShadow方法的第二個參數是調整模糊效果強弱的數值；最後一個參數則是陰影的顏色。設定好了這些，繪圖時就會出現陰影。

[儲存繪圖區域狀態]



照著上述方法繪製出有陰影的圖形之後，繼續繪製其他圖形，會發現其他圖形也會有陰影。如果要做到有些圖形有陰影、有些圖形沒有陰影的話，要在設定陰影之前，把當時的繪圖情境存起來，等到之後需要的時候，再回復沒有設定陰影繪圖情境：

```
override func draw(_ rect: CGRect) {
    var path = CGMutablePath()                                  //產生路徑
    let rectArea =
          CGRect(x: 50, y: 50, width: 200, height: 100) //設定方形範圍
    path.addRect(rectArea)                                      //加入方形到路徑中
    let context = UIGraphicsGetCurrentContext()                 //得到繪圖情境
    context?.addPath(path)                                      //把路徑加入繪圖情境

    //***** 儲存沒有陰影的狀態 *****
    context?.saveGState()
    UIColor.magenta.setFill()                                   //設定顏色
    context?.setShadow(offset: CGSize(width: 10,height: 10),//設定陰影
        blur: 20,
        color: UIColor.lightGray.cgColor)
    context?.fillPath()                                         //填色

    //***** 回復沒有陰影的狀態 *****
    context?.restoreGState()

    path = CGMutablePath()                                  //產生另外一個方形的路徑
    let anotherRectArea =
        CGRect(x: 50, y: 200, width: 200, height: 100)//設定另外方形範圍
    path.addRect(anotherRectArea)                           //加入另外一個方形到路徑中
    context?.addPath(path)                                  //把路徑加入繪圖情境
    UIColor.blue.setFill()                                  //設定顏色
    context?.fillPath()                                     //填色
}
```

如上程式碼，在設定陰影前，先使用CFContext的saveGState方法，儲存當時的繪圖情境。設定陰影畫出有陰影的圖形之後，再呼叫CGContext的restoreGState方法，恢復還沒有設定陰影狀態的繪圖情境。之後畫出的圖形就不會有陰影了。

繪製中空的圖形

[問題]如何繪製中空的圖形
[解答]呼叫CGContextDrawPath方法時，改變填色的模式可以繪製出中空的圖形
[範例程式碼]HelloHollowCircle
[過程解說]
1.請照著之前[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，寫下下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let path = CGMutablePath()              //產生路徑
    let largeCircleArea =
        CGRect(x: 50, y: 50, width: 300, height: 300)  //設定大圓形範圍
    let smallCircleArea =
        CGRect(x: 100, y: 100, width: 200, height: 200)//設定小圓形範圍

    path.addEllipse(in: largeCircleArea) //加入大圓形範圍
    path.addEllipse(in: smallCircleArea) //加入小圓形範圍

    let context = UIGraphicsGetCurrentContext() //得到繪圖情境
    context?.addPath(path)                  //把路徑加入繪圖情境

    context?.setLineWidth(10)               //設定邊線粗細
    UIColor.yellow.setStroke()                //設定邊線顏色
    UIColor.magenta.setFill()                 //設定顏色
    context?.drawPath(using: .eoFillStroke)    //填色
}
```

2.範例中在路徑中加入了兩個圓形的區域。小圓在大圓的中間。
3.要做出中空的圖形，只需在填色時，選擇eoFillStroke或是eoFill模式，就可以繪製出中空的圖形。選擇eoFillStroke模式不僅會填色，也會畫出邊線；選擇eoFill模式則只會填色，並不會畫出邊線。

移動、放大縮小或是旋轉繪製圖形

[問題]如何移動、放大縮小或是旋轉繪製的圖形
[解答]使用CGAffineTransform的各種方法來變形繪製的圖形
[範例程式碼]HelloTransformShape
[將繪製的圖形移動]
1.請照著[繪製方形與圓形]解答中、[以ViewController的View當作繪製圖形的畫布]的作法，開啟新專案、新增一個UIView的子類別，最後做好連結。在新增的UIView子類別的draw方法裡，寫下下面的程式碼：

```
override func draw(_ rect: CGRect) {
    let path = CGMutablePath()              //產生路徑
    let rectArea =
        CGRect(x: 50, y: 50, width: 100, height: 100) //設定方形範圍

    //移動圖形
    let transform = CGAffineTransform(translationX: 100, y: 200)
    //加入路徑時，附帶變形設定
    path.addRect(rectArea, transform: transform)

    let context = UIGraphicsGetCurrentContext() //得到繪圖情境
    context?.addPath(path)                  //把路徑加入繪圖情境

    UIColor.magenta.setFill()               //設定顏色
    context?.drawPath(using: .fill)    //填色
}
```

2.使用CGAffineTransform(translationX: y:)方法，可以移動繪製圖形的位置。這個方法需要兩個參數：第一個參數是x方向移動的距離；第二個參數則是y方向移動的距離。範例中的程式碼將讓圖形從本來的位置往右移動100，往下移動200。
3.設定好了移動的位置之後，在加入路徑時，把變形的設定當作參數代入，畫出來的圖形，就會照著設定移動到正確的位置。

[將繪製的圖形放大或縮小]
要將繪製圖形放大或縮小的話，請把上面範例設定移動圖形位置的程式碼，代換成下面的：

```
//縮放圖形
let transform = CGAffineTransform(scaleX: 2.0, y: 2.0)
```

使用CGAffineTransform(scaleX: ,y:)方法，可以放大或縮小繪製的圖形。這個方法需要兩個參數：第一個參數是x方向放大縮小的比例；第二個參數則是y方向放大縮小的比例。範例中的程式碼將讓圖形放大成原本的兩倍大。如果要縮小圖形的話，比方說要縮小成一半的大小，請把放大縮小的數值改成0.5。

[旋轉繪製的圖形]
要旋轉繪製的圖形，請再把上面範例設定縮放圖形的程式碼，代換成下面的：

```
//旋轉圖形
let radian = CGFloat((10.0 * M_PI)/180.0)    //設定旋轉角度
let transform = CGAffineTransform(rotationAngle: radian)
```

使用CGAffineTransform(rotationAngle:)方法，可以旋轉繪製的圖形。範例中把角度轉換成徑度之後，把該數值當作參數呼叫CGAffineTransform(rotationAngle:)方法，圖形就會旋轉10度。