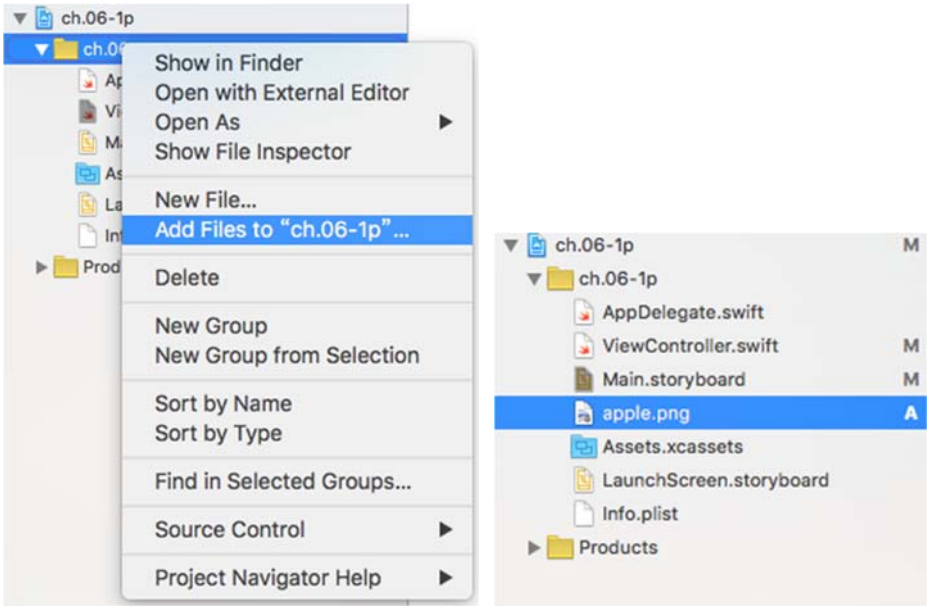
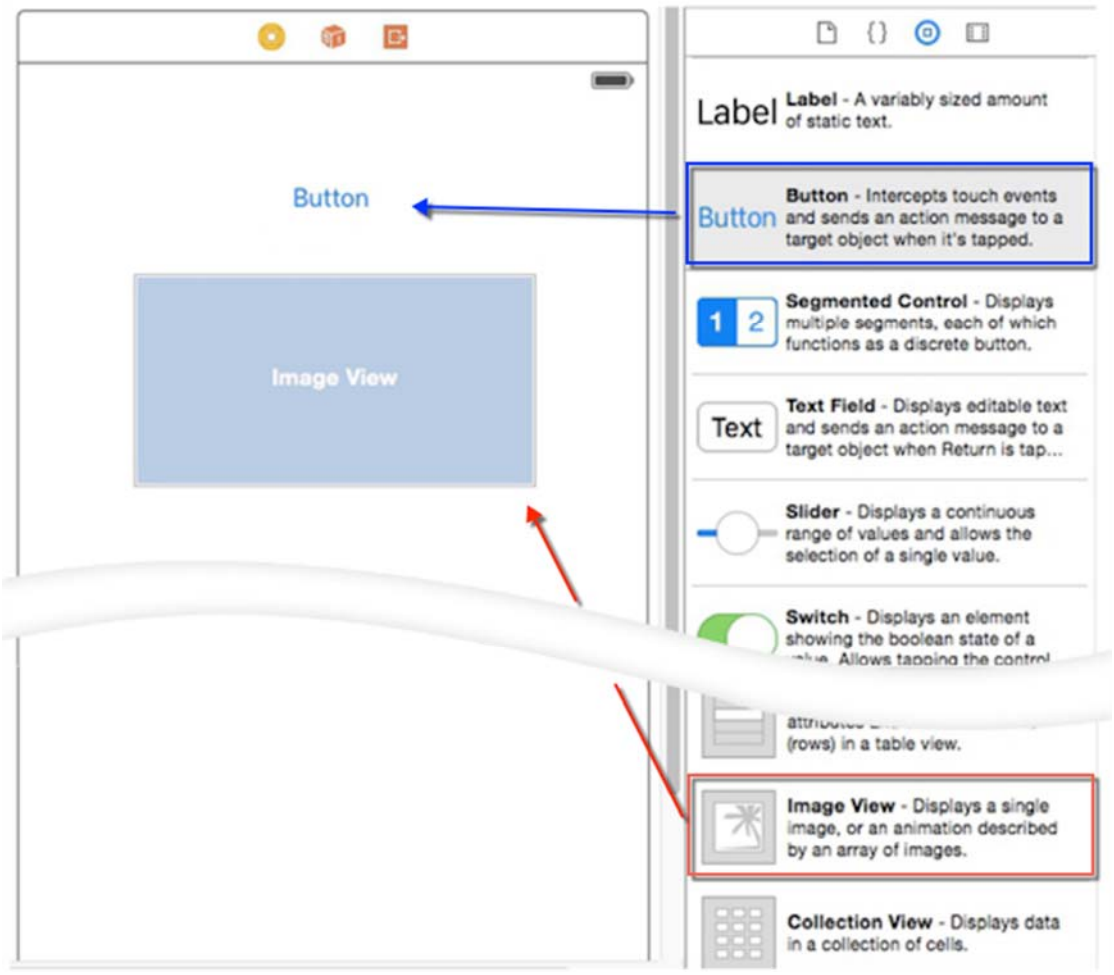
6-1 解 答

1. 將 apple.png 圖檔加入



2. 拖曳【操作按鈕】「Button」及【圖像方塊】「Image View」建立在 IB 畫面中。
3. 將【操作按鈕】「Button」名稱變更為「Show Picture」。



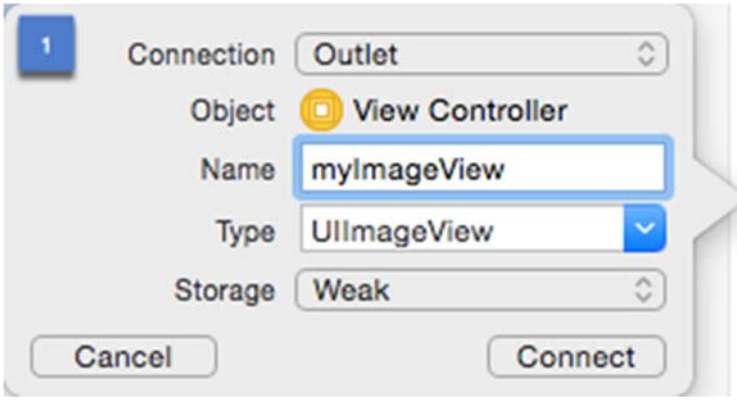
4. 在 IB 視窗上點選【Image View】(為畫面紅色方框) 及【Button】(為畫面藍色方框)，接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】建立連結。

```
import UIKit

class ViewController: UIViewController {

    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.
    }

1   @IBOutlet weak var myImageView: UIImageView!

2   @IBOutlet weak var myBtn: UIButton!

3   @IBAction func tapBtn(sender: AnyObject) {

    }

}
```

分解如下

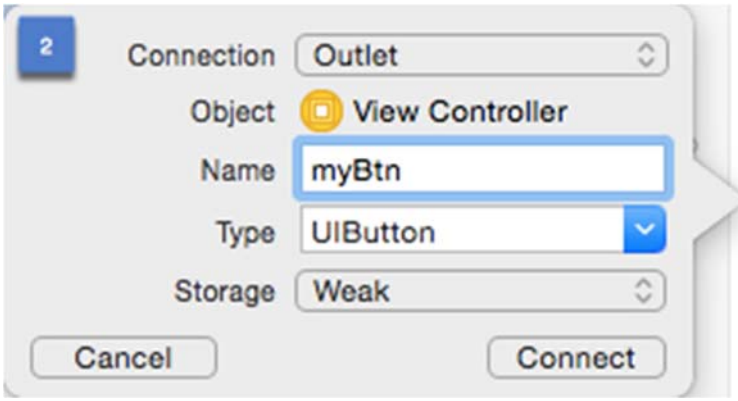
5. 由【物件區】選取之各個元件，設定名稱如下：

拖曳【Image View】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myImageView】以及在「Type」欄位點選【UIImagView】後，按【Connect】按鈕。



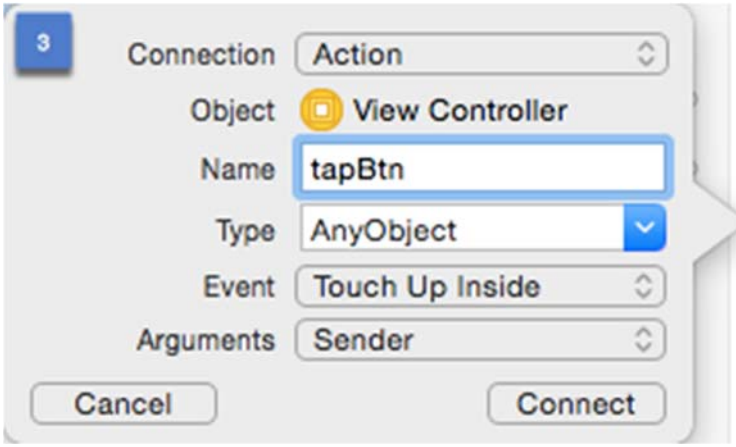
在拖曳後自動彈跳的視窗

拖曳【Button】於「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myBtn】以及在「Type」欄位點選【UIButton】後，按【Connect】按鈕。



在拖曳後自動彈跳的視窗

再一次拖曳【Button】於「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。

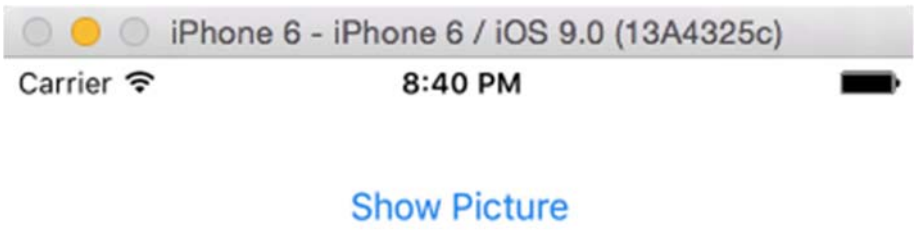


在拖曳後自動彈跳的視窗

6. 最後，我們在【紅框】中設定 swift 的程式碼在編輯區當中。

```
import UIKit

class ViewController: UIViewController {

    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.
    }

    @IBOutlet weak var myImageView: UIImageView!

    @IBOutlet weak var myBtn: UIButton!

    var x:Bool = false

    @IBAction func tapBtn(sender: AnyObject) {

        if x==false {
            myImageView.image = UIImage(named: "apple.png")
            x = true
            myBtn.setTitle("Hidden Picture", forState: .Normal)
        } else {
            myImageView.image = nil
            x = false
            myBtn.setTitle("Show Picture", forState: .Normal)
        }


    }

}
```

7. 執行畫面，出現經由【操作按鈕】「Button」後，圖像畫面出現「蘋果」圖樣，讀者們可以試試當選點時，經由在我們撰寫的【圖像方塊】「Image View」在【程式碼】串連後會出現什麼樣的變化結果。



讀者們，請試著點選「Show Picture」後，會出現什麼變化。

沒錯，蘋果圖像經由點選後出現在畫面中。

接著，我們再試試當點選「Hidden Picture」後，會再回到隱藏圖片畫面。

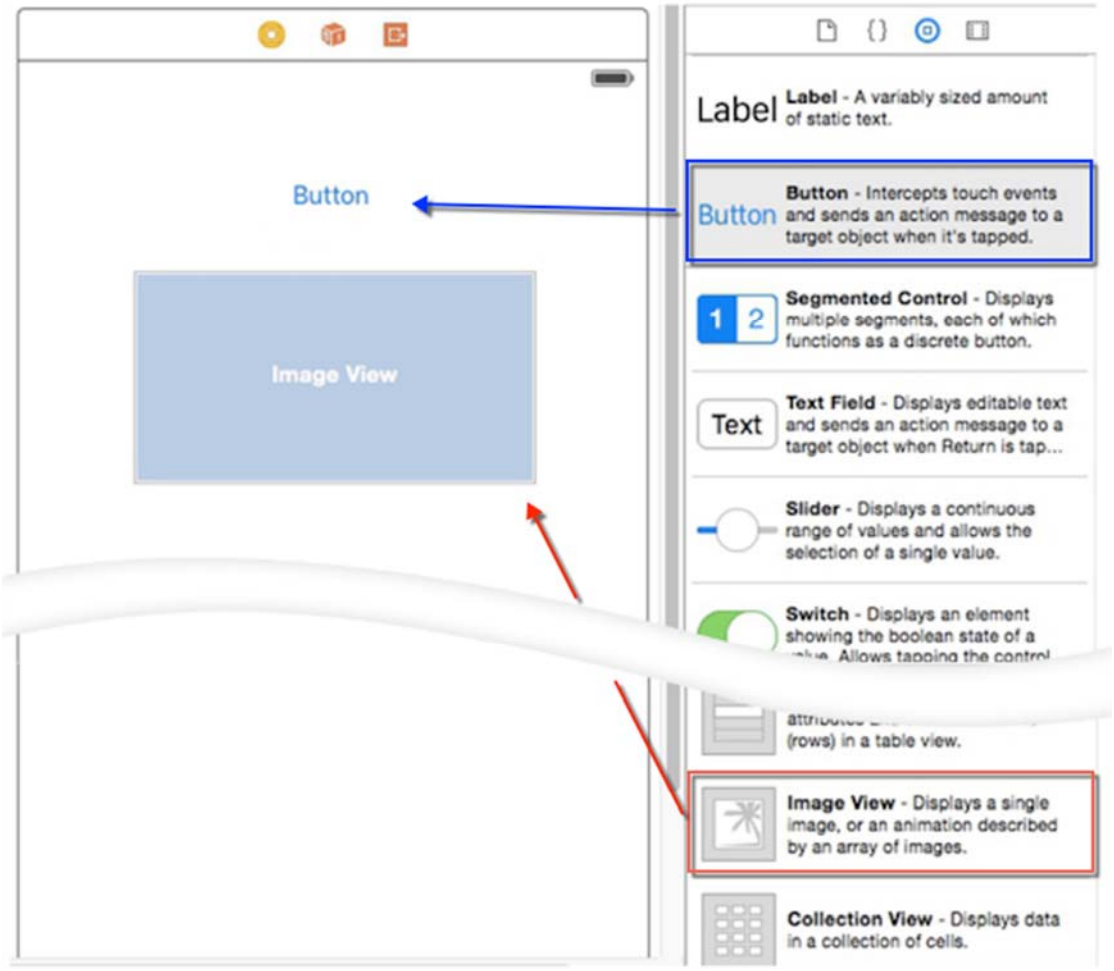
在這裡是不是學習到更多的技法，這次我們巧妙運用了【操作按鈕】「Button」及【圖像方塊】「Image View」二項物件之基礎運用，足足邁進了好幾大步。

編編建議讀者們，別忘了有關於詳細的 swift 語法，還是可以多翻閱第三章相關說明唷。

加油！

6-2 解 答

1. 拖曳【操作按鈕】「Button」及【圖像方塊】「Image View」建立在 IB 畫面中。



拖曳 Image View 後調整適當的大小

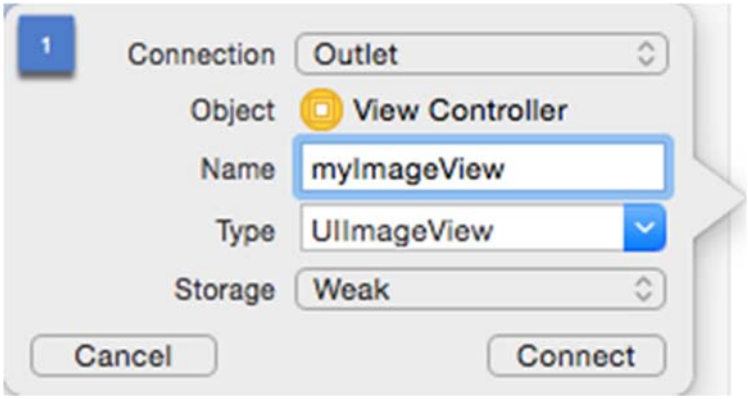
2. 在 IB 視窗上點選【Image View】(為畫面紅色方框) 及【Button】(為畫面藍色方框)，接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】建立連結。

```
import UIKit

class ViewController: UIViewController {

    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.
    }
1   @IBOutlet weak var myImageView: UIImageView!
2   @IBAction func tapBtn(sender: AnyObject) {

    }

}
```

分解如下

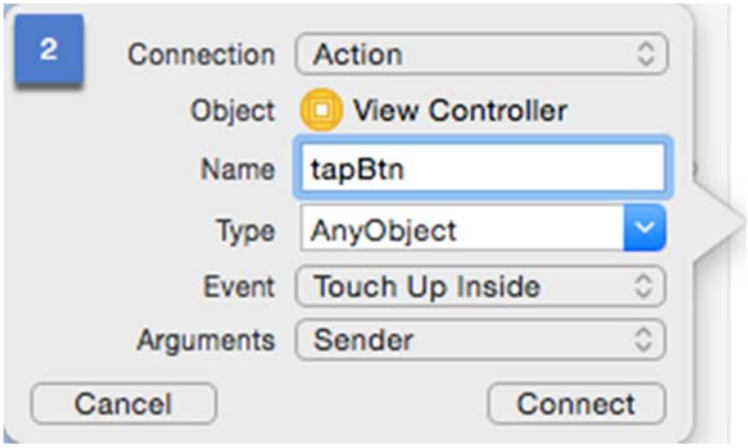
3. 由【物件區】選取之各個元件，設定名稱如下：

   拖曳【Image View】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myImageView】以及在「Type」欄位點選【UIImageView】後，按【Connect】按鈕。



在拖曳後自動彈跳的視窗

拖曳【Button】於「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。

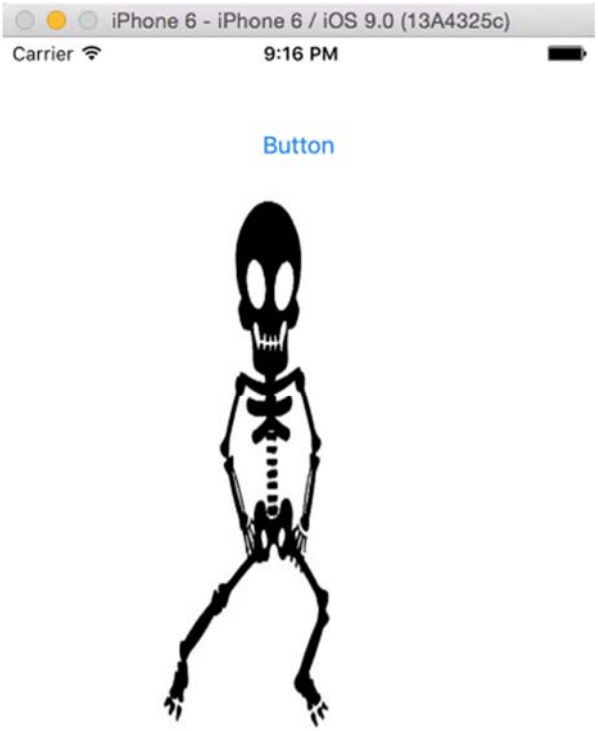


在拖曳後自動彈跳的視窗

4. 最後，我們在【紅框】中設定 swift 的程式碼在編輯區當中。

```
import UIKit

class ViewController: UIViewController {

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

        myImageView.image = UIImage(named: "dance1.png")
    }

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

    @IBOutlet weak var myImageView: UIImageView!

    var x:Bool = true

    @IBAction func tapBtn(sender: AnyObject) {

        let imageArray: [UIImage] = [
            UIImage(named: "dance1.png")!,
            UIImage(named: "dance2.png")!,
            UIImage(named: "dance3.png")!,
            UIImage(named: "dance4.png")!
        ]

        myImageView.animationImages = imageArray
        myImageView.animationDuration = 1.0
        myImageView.animationRepeatCount = 0
        //myImageView.startAnimating()


        if (x == true){
            myImageView.startAnimating()
            x = false
        } else {
            myImageView.stopAnimating()
            x = true
        }
    }
}
```

5. 執行畫面，出現經由【操作按鈕】「Button」後，圖像畫面出現「機器人」圖像，讀者們可以試試當選點時，經由在我們撰寫的【圖像方塊】「Image View」在【程式碼】串連後會出現什麼樣的變化結果。

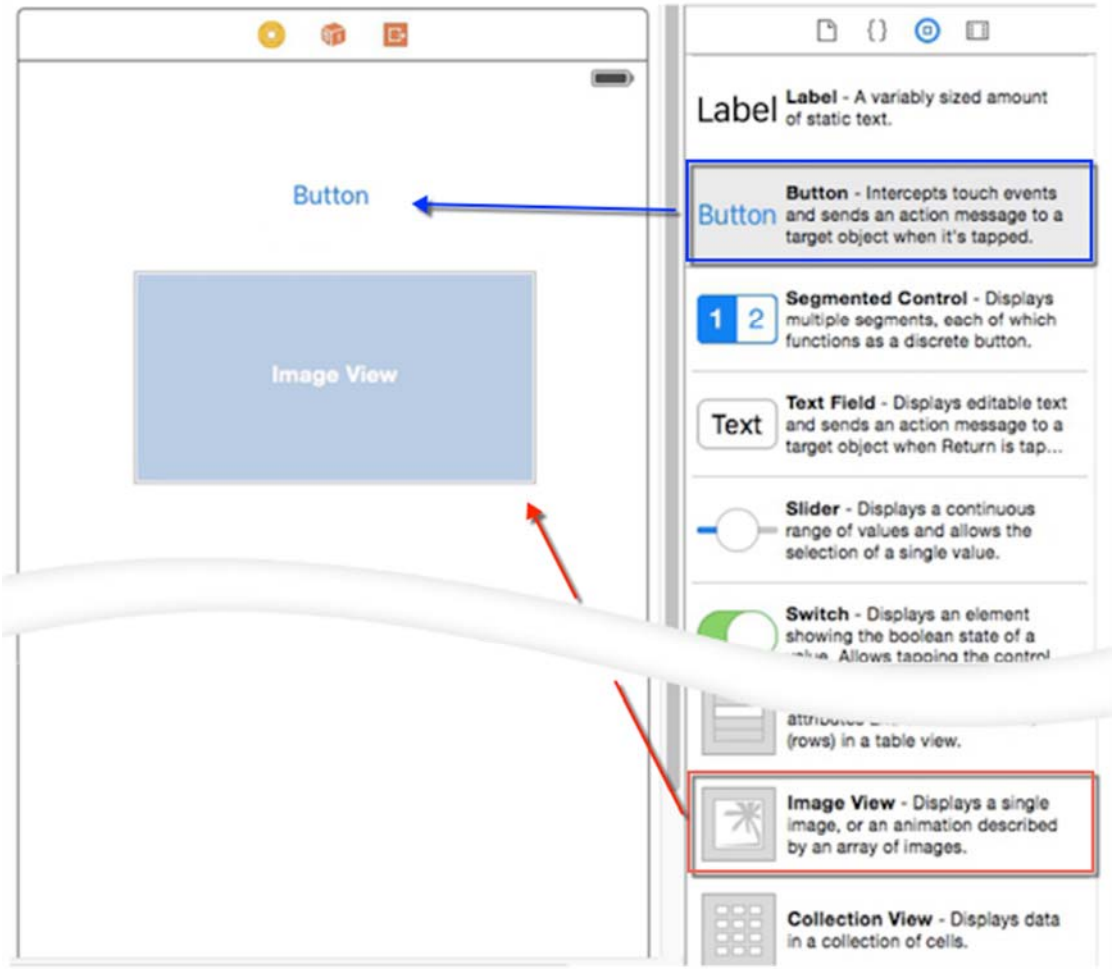


在點選「Button」後，圖像會出現「啟動」及「停止」在畫面中。

【圖像方塊】「Image View」不再是那樣單調沒有變化的。雖然只是基礎的運用，但在這裡我們透過 swife 的屬性，建立動畫播放的「順序」、「時間」、「重覆次數」以及「啟動停止播放」……等技法，是不是挑起了熱愛小撇步的你呢。編編建議讀者們，別忘了有關於詳細的 swift 語法，還是可以多翻閱第三章相關說明唷。

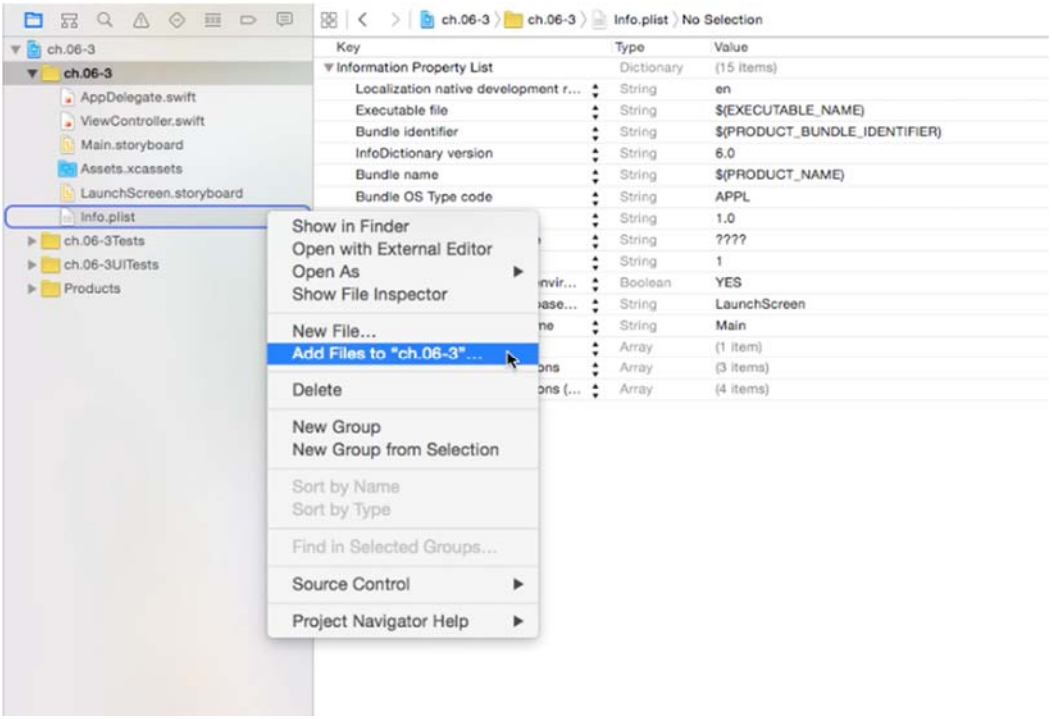
6-3 解 答

1. 拖曳【操作按鈕】「Button」及【圖像方塊】「Image View」建立在 IB 畫面中。

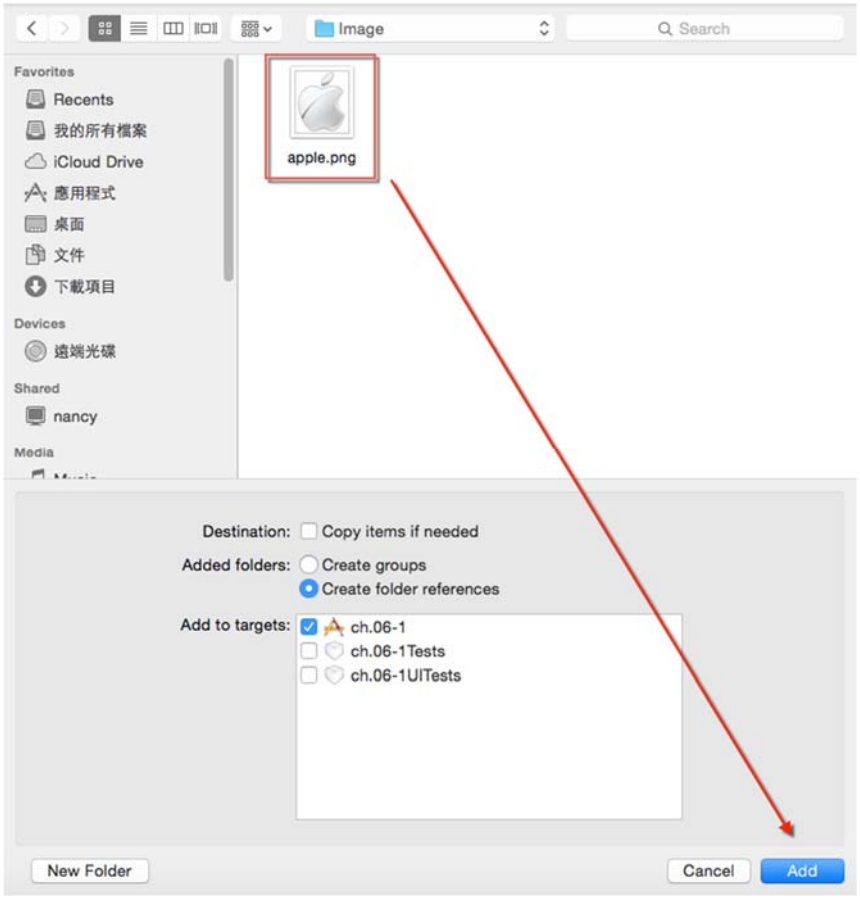


拖曳 Image View 後調整適當的大小

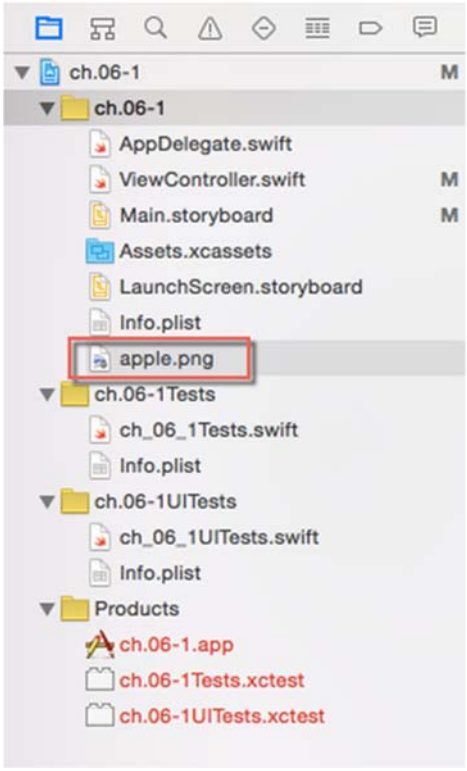
(1) 在這裡我們選擇一張圖片，展開 ch.06-3 目錄並且在 info.plint 檔名按右鍵→Add Files to “ch.06-3….”。



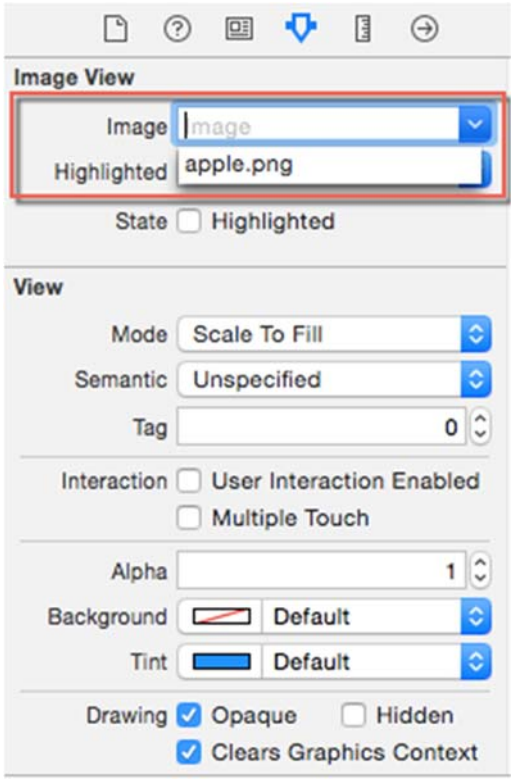
(2) 在這裡我們選擇選取的檔案名稱為＜apple.png＞選取後按「Add」。



(3)　完成後，接著會看到顯示出個檔名為＜apple.png＞。



(4)　我們在右邊「屬性檢視」的視窗，點選我們所要加入點片的檔案名稱＜apple.png＞



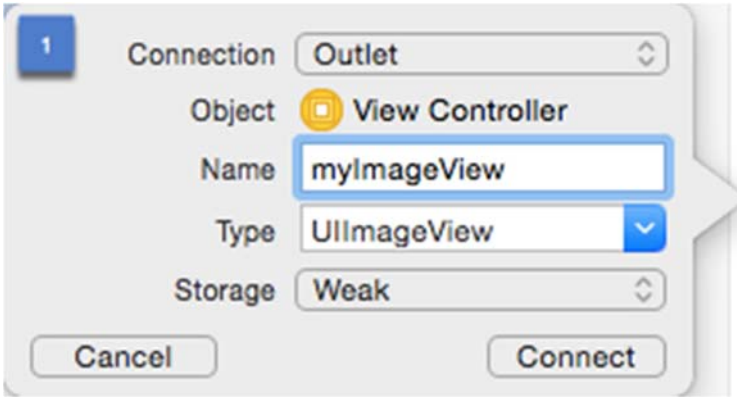
2. 在 IB 視窗上點選【圖像方塊】「Image View」（為畫面紅色方框）及【操作按鈕】、「Button」（為畫面藍色方框），接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】建立連結。

```
import UIKit

class ViewController: UIViewController {

    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.
    }
1   @IBOutlet weak var myImageView: UIImageView!
2   @IBAction func tapBtn(sender: AnyObject) {

    }

}
```

分解如下

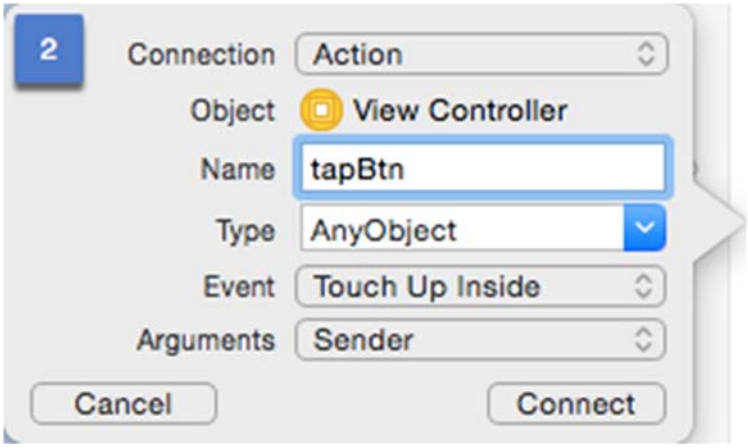
3. 由【物件區】選取之各個元件，設定名稱如下：

拖曳【Image View】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myImageView】以及在「Type」欄位點選【UIImageView】後，按【Connect】按鈕。



在拖曳後自動彈跳的視窗

拖曳【Button】於「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。

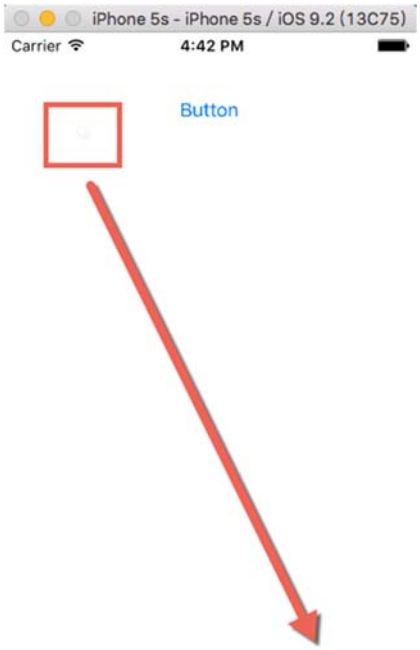


在拖曳後自動彈跳的視窗

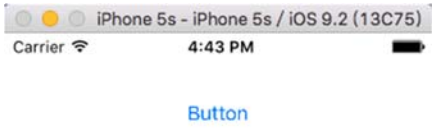
4. 最後，我們在【紅框】中設定 swift 的程式碼在編輯區當中。

```
import UIKit

class ViewController: UIViewController {

    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.
    }

    @IBOutlet weak var myImageView: UIImageView!

    @IBAction func tapBtn(sender: AnyObject) {

        self.myImageView.center = CGPointMake(50,50)
        self.myImageView.alpha = 0.1
        self.myImageView.transform = CGAffineTransformMakeScale(0.1, 0.1)

        UIView.beginAnimations("animation", context: nil)
        UIView.setAnimationDuration(2)

        myImageView.center=CGPointMake(250, 480)
        myImageView.alpha = 1.0
        myImageView.transform = CGAffineTransformMakeScale(1.0, 1.0)

        UIView.commitAnimations()

    }
}
```

5. 執行畫面，出現經由【操作按鈕】「Button」後，圖像畫面出現「蘋果」圖像，當在選點時，會經由在我們撰寫的【圖像方塊】「Image View」在【程式碼】串連後會出現的變化結果。



在點選「Button」後，此時，蘋果由左上方由原圖 10 分 1 大小，半透明方式出現在右上的畫面中。

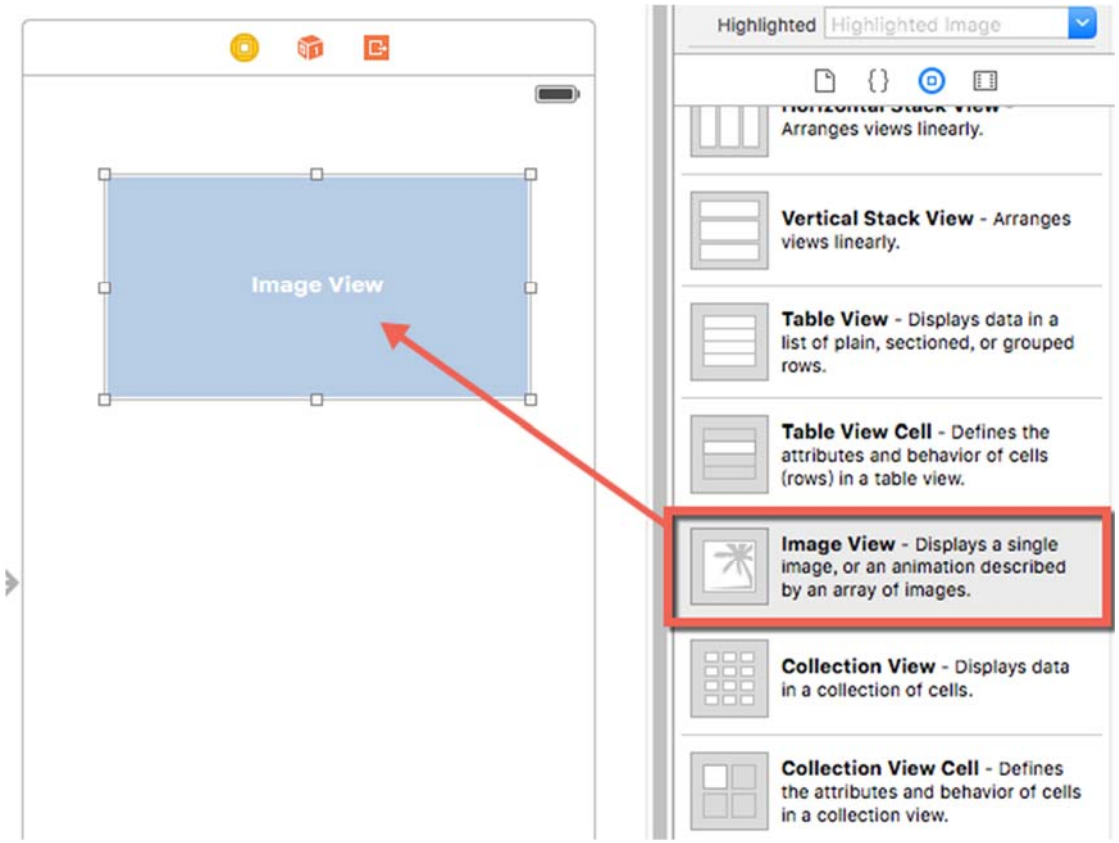


在經過 2 秒的過程中，圖像會慢慢變大出現在右下的畫面中。

【圖像方塊】「Image View」不再是那樣單調沒有變化的。雖然只是基礎的運用，但在這裡我們透過 swife 的屬性，建立動畫播放的「順序」、「時間」、「重覆次數」以及「啟動停止播放」……等技法，是不是挑起了熱愛小撇步的你呢。編編建議讀者們，別忘了有關於詳細的 swift 語法，還是可以多翻閱第三章相關說明唷。

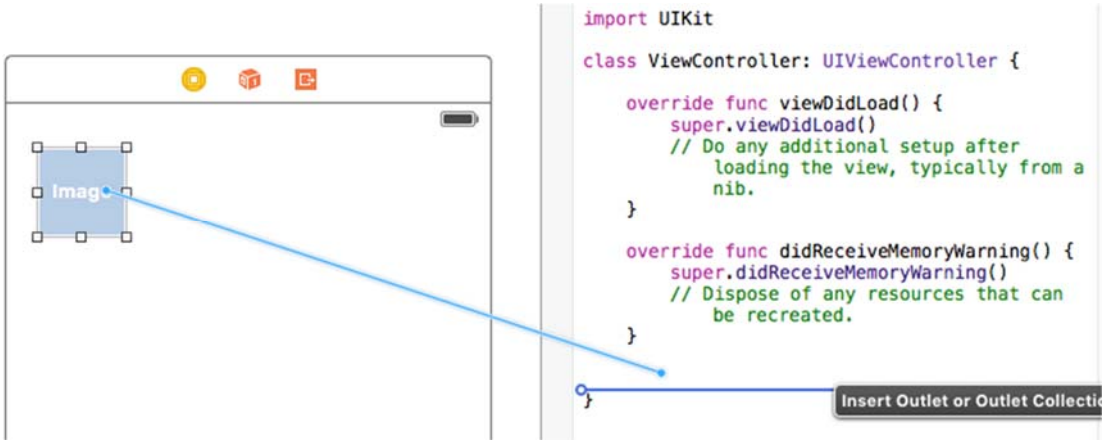
6-4 解 答

1. 拖曳【Image View】建立在 IB 畫面中。



拖曳【Image View】到適當的位置

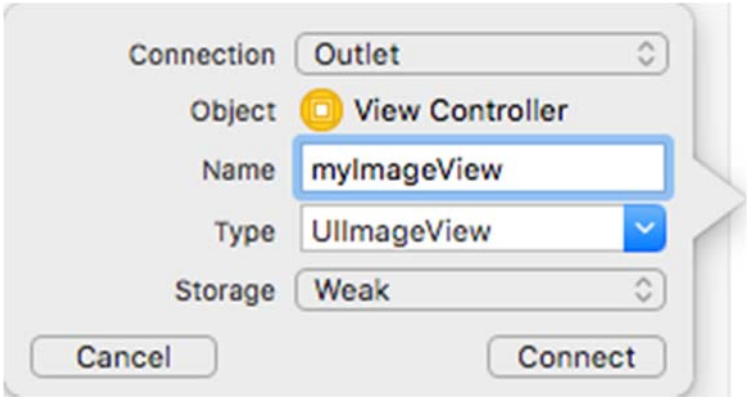
2. 調整【Image View】到左上角，在 IB 視窗上點選【Image View】，接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】建立連結。



按住【Control】用滑鼠拖曳【ImageView】元件。

3. 由【物件區】選取之元件，設定名稱如下：

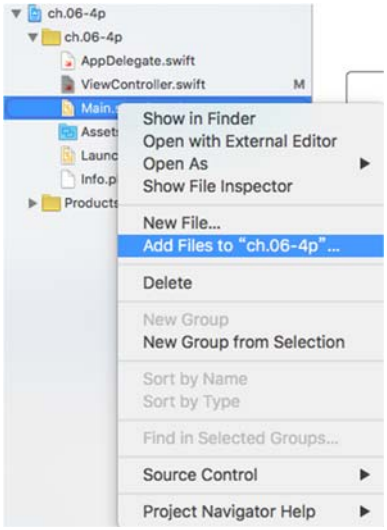
在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myImageView】以及在「Type」欄位點選【UIImageView】後，按【Connect】按鈕。



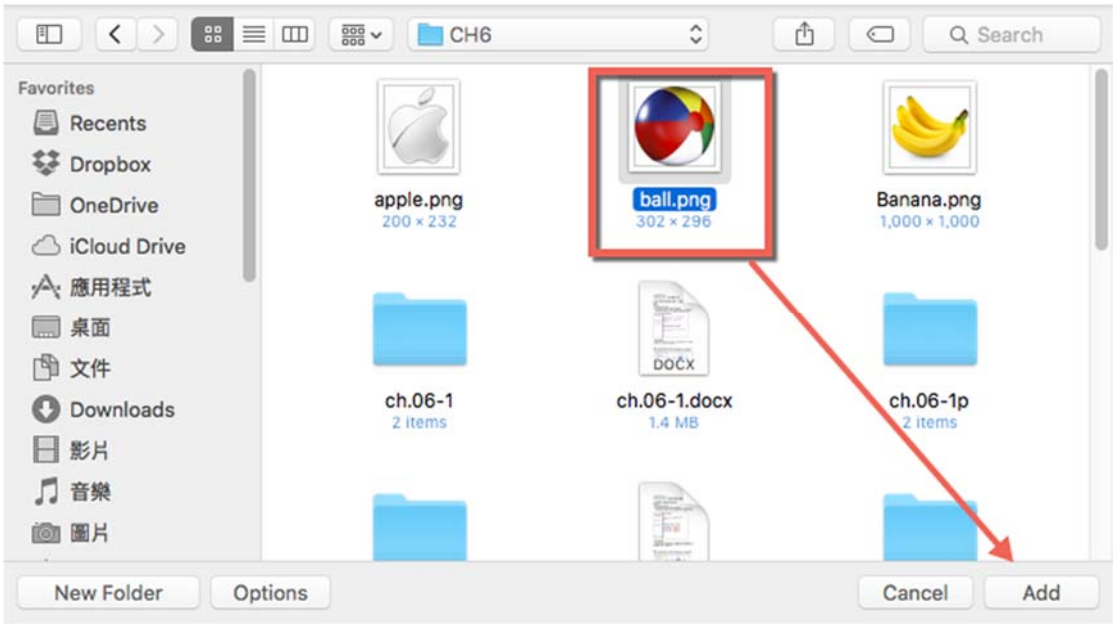
在拖曳後自動彈跳的視窗

4. 加入圖檔於 【ImageView】元件

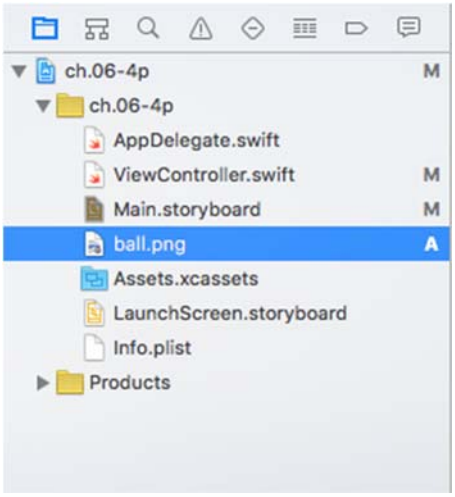
(1) 在這裡我們選擇一張圖片，展開 ch.06-4p 目錄並且按右鍵→Add Files to “ch.06-4p….”。



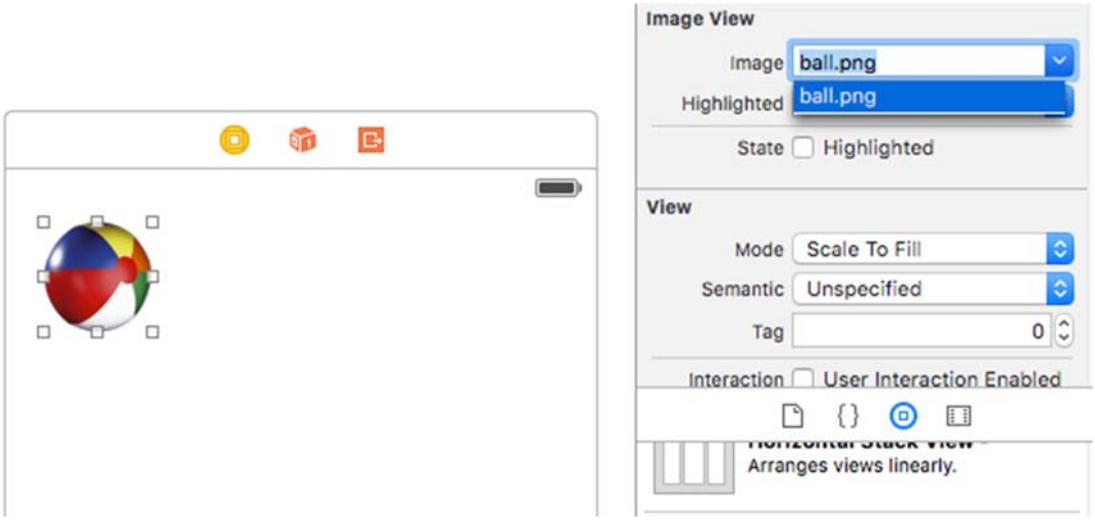
(2) 在這裡我們選擇選取的檔案名稱為＜ball.png＞選取後按「Add」。



(3)　完成後，接著會看到顯示出個檔名為＜ball.png＞。



(4)　在【ImageView】元件的「屬性檢視」的視窗，點選我們所要加入點片的檔案名稱＜ball.png＞

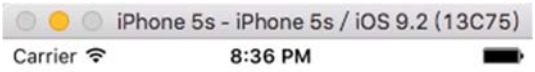


5. 最後，我們在【紅框】中設定 swift 的程式碼在編輯區當中。

```
import UIKit

var myTimer : NSTimer = NSTimer()

class ViewController: UIViewController {

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

        myTimer = NSTimer.scheduledTimerWithTimeInterval(0.05, target:
            self, selector:("moveLabel"), userInfo: nil, repeats:
            true)
    }

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

    @IBOutlet weak var myImageView: UIImageView!

    var right:Bool = true
    var down:Bool = true

    func moveLabel() {
        var wx, wy:CGFloat
        wx = myImageView.center.x
        wy = myImageView.center.y

        // 控制左右方向運動
        if (wx > 320 && right) {
            right = false
        }

        if (wx < 0 && !right) {
            right = true
        }

        if (right) {
            wx = wx + 10
        } else {
            wx = wx - 10
        }

        // 控制上下方向運動
        if (wy > 480 && down) {
            down = false
        }
        if (wy < 0 && !down) {
            down = true
        }
        if (down) {
            wy = wy + 5
        } else {
            wy = wy - 5
        }


        myImageView.center = CGPointMake(wx, wy)
    }
}
```

手動輸入紅框內的程式

6. 在啟動「RUN」後，會出現「球」不停的移動時，當遇到邊框時會自動彈回的效果。



在這裡我們僅以簡單的【ImageView】與程式結合執行。發揮創意採行純粹基礎的運用，並透過 swife 的屬性運用，建立以計時器性質的動畫巧思……等技法。除了平日的演練之外，編編建議讀者們，別忘了有關於詳細的 swift 語法，還是可以多翻閱第三章相關練說明唷。