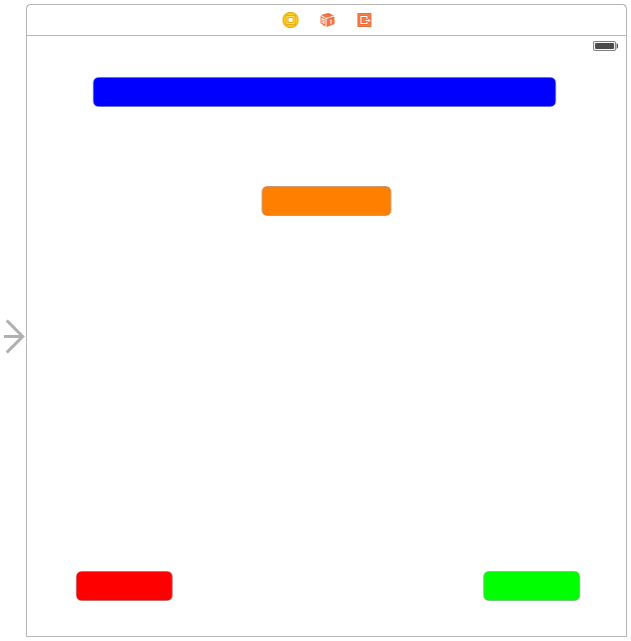
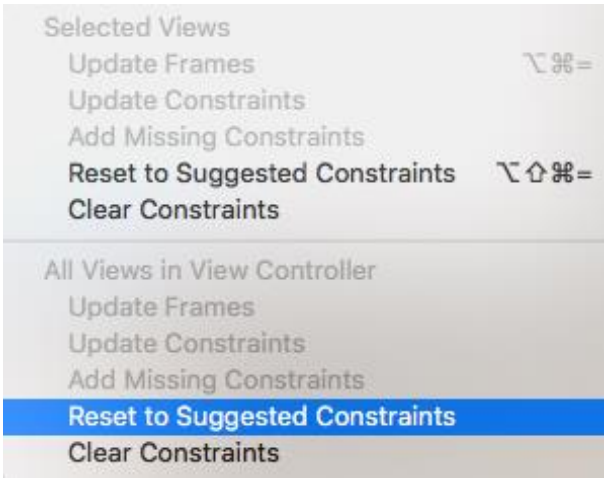
7-1

解 答

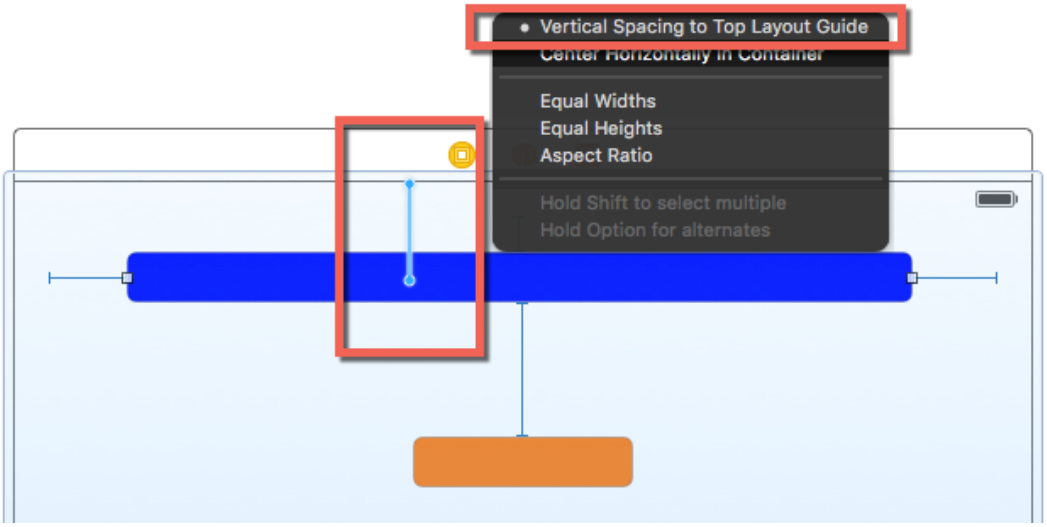
1. 將四個文字框加入畫面中。



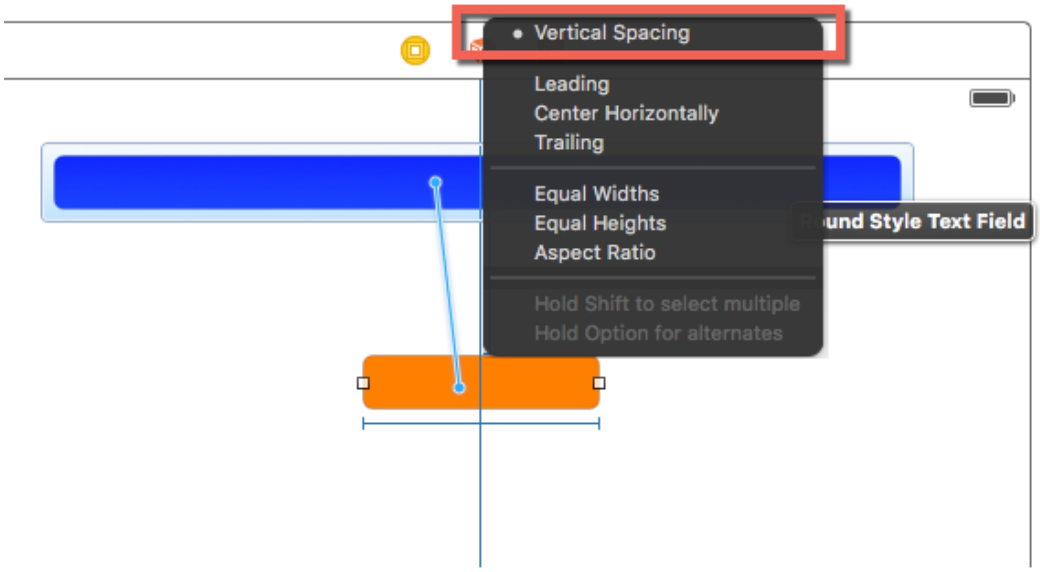
2. 最簡單的方法是採用 Reset to Suggested Constraints



3. 若要個別設定，則點擊某一元件後，可透過滑鼠右鍵按住不放，拖拉出一條線指向邊界或對某一元件。
   例如：藍色框對頂端的垂直距離



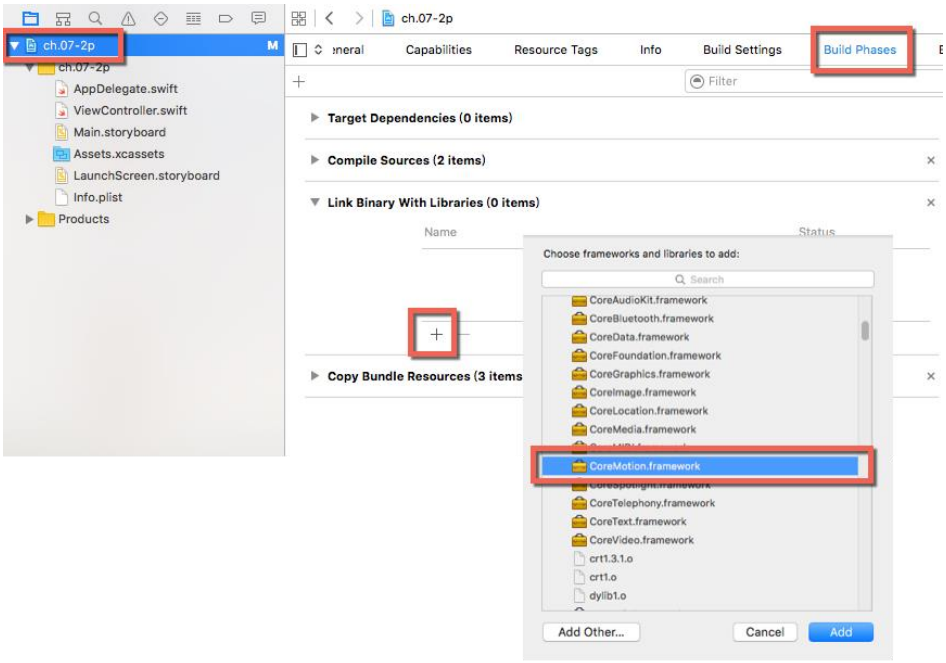
桔色文字框對藍色文字框的垂直距離



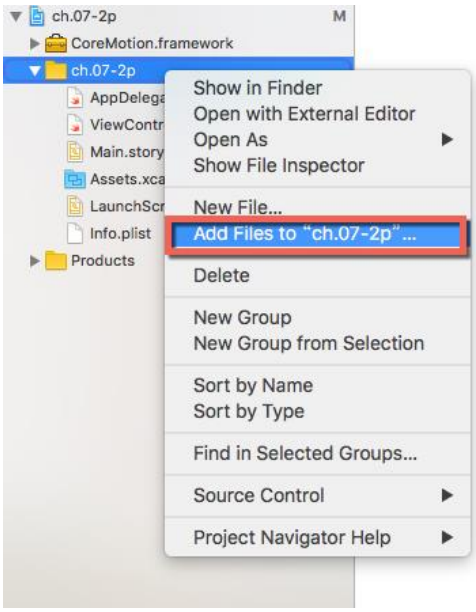
7-2

解 答

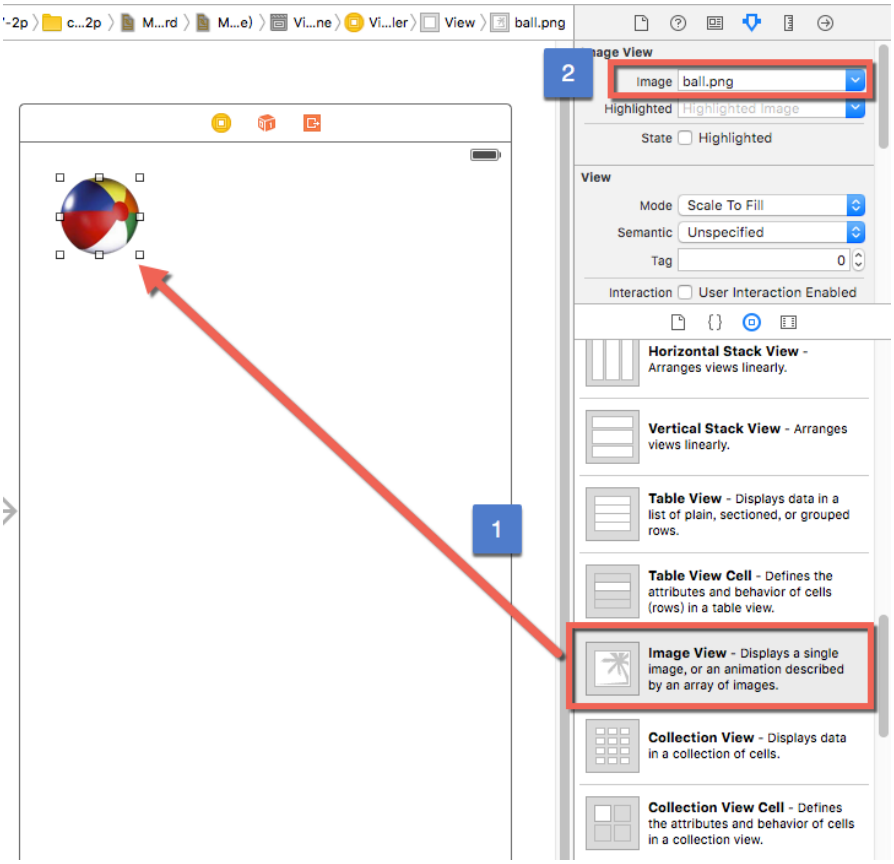
1. 將 CoreMotion.framework 加入所要連結使用的函式庫。



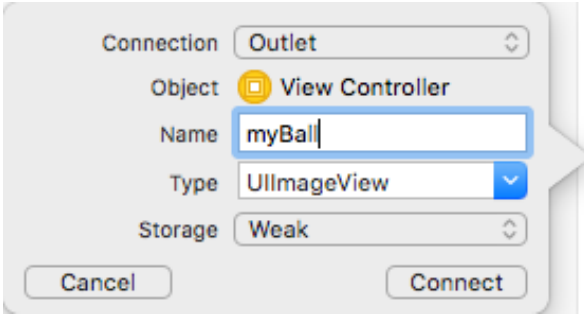
2. 將圖檔 ball.png 加入專案中。



3. 拖拉一個 ImageView 元件到畫面中，並將其圖檔選定為 ball.png



4. 將 ImageView 建立 Outlet， 命名為 myBall



5. 開啟 ViewController.swift 將 CoreMotion 函式庫加入。

```
import CoreMotion
```

6. 撰寫啟動加速器偵測的程式，由於加速度直會在 -1.0~1.0 之間變化，若直接使用做為改變運動的距離會太小，而顯的求移動的速度太慢，因此將加速度值放大 20 倍後，再用做計算新座標。

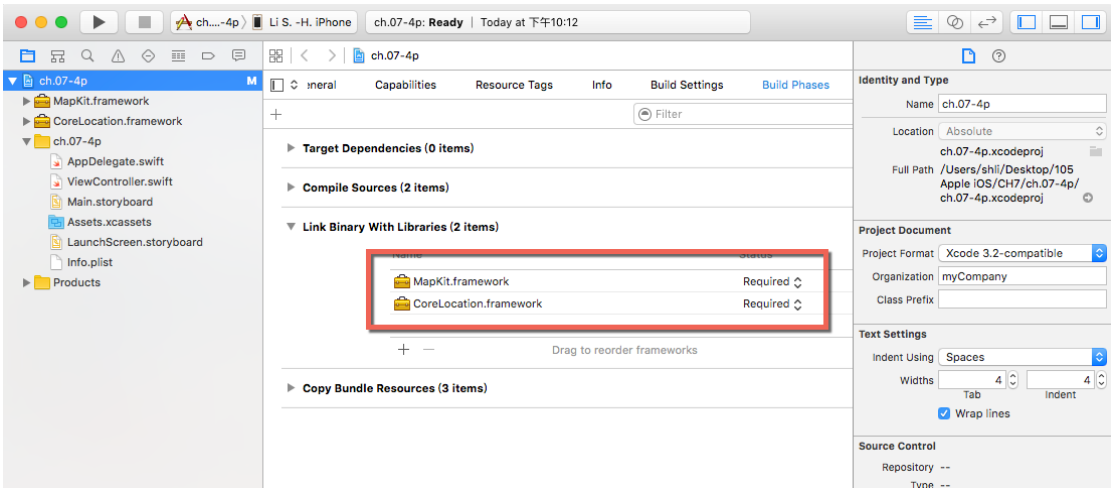
```
import UIKit
import CoreMotion

class ViewController: UIViewController {

    let motionManager: CMMotionManager = CMMotionManager()

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

        motionManager.accelerometerUpdateInterval = 0.02

        motionManager.startAccelerometerUpdatesToQueue(NSOperationQueue.mainQueue())
            { [weak self] (data: CMAccelerometerData?, error: NSError?) in

                //設定球的移動 X 與 Y 值
                var wx = self!.myBall.center.x + CGFloat( data!.acceleration.x ) * 20
                var wy = self!.myBall.center.y - CGFloat( data!.acceleration.y ) * 20

                //設定 球 移動到邊界時 需要停住不可超出範圍
                if wx < 25 { wx = 25 }
                else if wx > 295 { wx = 295 }
                if wy < 25 { wy = 25 }
                else if wy > 545 { wy = 545}

                //依照座標值，設定球的中心座標
                self!.myBall.center = CGPointMake(CGFloat(wx), CGFloat(wy) )
        }

    }

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


    @IBOutlet weak var myBall: UIImageView!
}
```

7. 在啟動「RUN」之前， 需要先選擇實體裝置，才可將程式安裝於手機上以做為測試使用。

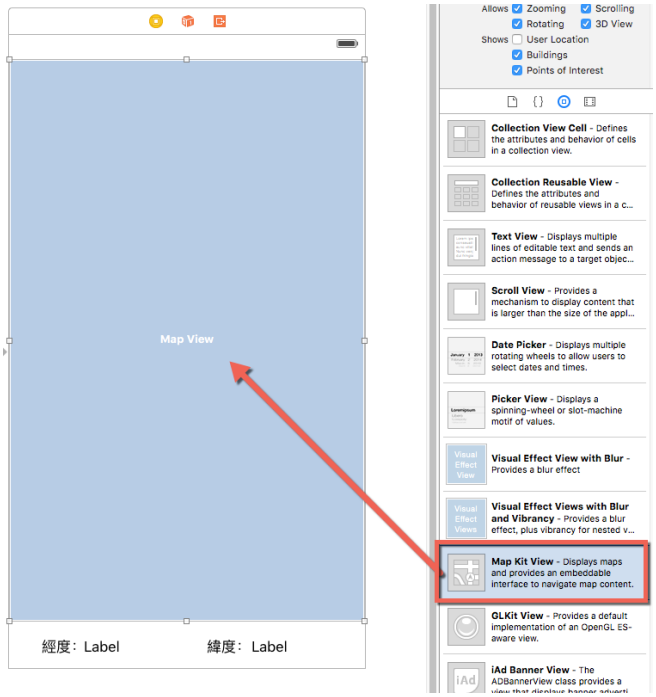
7-4

解 答

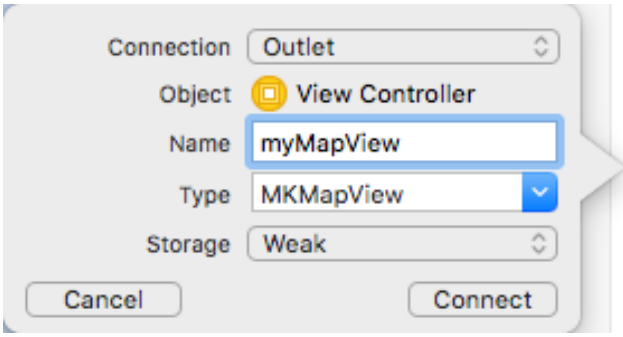
1. 將 MapKit.framework 與 CoreLocation.framework 加入所要連結使用的函式庫。



2. 拖拉一個 【Map Kit View】、【Label】 元件到畫面中。



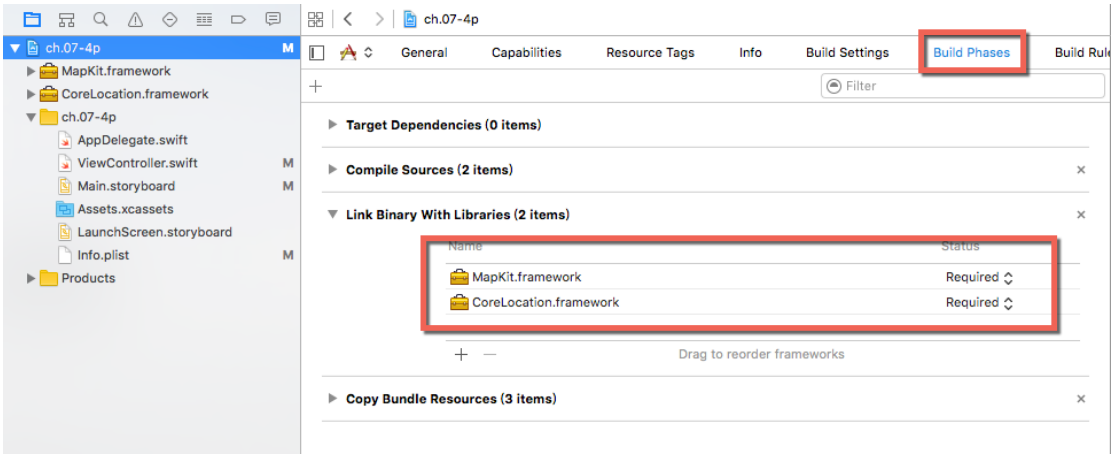
3. 將 Map Kit View 建立 IBOutlet， 命名為 myMapView



4. 將【文字欄位】和【名稱連結】，分別命名 myLngLabel, myLatLabel
5. 開啟 ViewController.swift 將 CoreLocation，MapKit 函式庫加入。

```
import CoreLocation
import MapKit
```

6. 加入 CoreLocation.framework， MapKit.framework 到函式庫



7. 撰寫程式碼

(1) 加入函式庫

```
import CoreLocation
import MapKit
```

(2) 於 class ViewController 之後需加上 CLLocationManagerDelegate 變數，

```
class ViewController: UIViewController , CLLocationManagerDelegate ,
MKMapViewDelegate {
```

(3) 建立位置感應器的物件

```
let manager = CLLocationManager()
```

(4) 宣告兩個變數資料型態是 NSString 用來接收傳出來的座標值

```
var LatitudeGPS = NSString()
var LongitudeGPS = NSString()
```

(5) 設定位置感應器物件的屬性。

讓位置感應器的作用於主程式上

```
self.manager.delegate = self
```

設定精細度為最高

```
self.manager.desiredAccuracy = kCLLocationAccuracyBest
```

設定多遠的距離才更新位置資訊，目前設定為一移動就更新

```
self.manager.distanceFilter = kCLDistanceFilterNone
```

獲得手機權限

```
self.manager.requestWhenInUseAuthorization()
```

開始執行取得位置資訊

```
self.manager.startUpdatingLocation()
```

讓地圖顯示現在的座標點

```
myMapView.showsUserLocation = true
```

(6) 手動撰寫位置更新時的動作。

讀出經緯度的數值並將數值轉換為文字。

隨著座標值的更動要不斷更新地圖顯示範圍，並標示現在所在點。

```
func locationManager(manager: CLLocationManager,
didUpdateLocations locations: [CLLocation]){

        LatitudeGPS = String(format: "%.6f",
manager.location!.coordinate.latitude)
        LongitudeGPS = String(format: "%.6f",
manager.location!.coordinate.longitude)
        myLatLabel.text = String( LatitudeGPS )
        myLngLabel.text = String( LongitudeGPS )

        location.longitude = manager.location!.coordinate.longitude
        location.latitude = manager.location!.coordinate.latitude
        let span = MKCoordinateSpanMake(0.001, 0.001)
        let region = MKCoordinateRegion(center: location, span:
span)
        myMapView.setRegion(region, animated: true)

    }
```

8. 在啟動「RUN」之前， 需要先選擇實體裝置，才可將程式安裝於手機上以做為測試使用。