AR Kit 入門

Danny Shen shenfive@gmail.com

先決條件

只支援 iOS 11 之後的作業系統，Xcode 9 之後的版本開發

只支援 iPhone 6s 或 iPad 2017 之後 的版本 (A9 以上處理器）

必需要有實體機器，模擬器可執行，但沒有任何意義 (沒有加速器，陀螺儀，相機等元件）

若有 LiDAR 的機種，效果會好很多 (iPad 2020)

什麼是 AR?

擴增實境(Augmented Reality，簡稱AR)，也有對應VR虛擬實境一詞的翻譯稱為實擬虛境或擴張現實，是指透過攝影機影像的位置及角度精算並加上圖像分析技術，讓螢幕上的虛擬世界能夠與現實世界場景進行結合與互動的技術。

重要的 AR 案例

Google Glass

Project Tengo

Microsoft Hololans

Hello AR Kit

使用 AR 模版，建立一個新的 Project

進行必要設定 (如 Target OS) 等

執行

新增一個檔案在 art.scnassts 下

試著拉幾個元件 node，並著色

修改 ViewController 中的 Scene

要觀查的項目～與空白專案的不同

觀查 sample scn 檔

這是場景檔，目前有一架飛機，使用了一個材質

觀查 StoryBoard

在第一個場景，多了一個ARSCeneView

觀查 ViewController

支援了 ARSCNViewDelegat 與相關的 code，並在 viewDidLoad 時，載入了 scn 檔，viewWillAppear 時，加入了 scene 的 configuration

空間概念

在場景初始化時，手機在空間的位置為 (0,0,0)

位置表示法為 (x,y,z) x 代表左右，y 代表上下，z 代表前後

左/下/前方向為負值，右/上/後 方向為正值

所有的東西都是以【公尺】為計算單位

任何物件都是以 node 為實體，node 是一個或多個幾何形狀組成

幾何形狀的外圍需有材質貼圖才能看的到

加上這一行 code 以協助了解座標

```
sceneView.debugOptions = [.showWorldOrigin]
```

建立一個簡單的方塊

在 viewDidLoad 中加入方塊(或其他形狀)，其原為為物體中央，可依狀況來貼不同的材質，

```
sceneView.delegate = self
sceneView.showsStatistics = true
let scene = SCNScene() //使用空白場景
// Set the scene to the view
sceneView.scene = scene
let box = SCNBox(width: 0.1, height: 0.1, length: 0.1, chamferRadius: 0.01) //新增一個 BOX
let material = SCNMaterial() //新增材質
material.diffuse.contents = UIColor.red  //材質內容為紅色
box.materials = [material] //把 box 的貼圖材質加進去
let node = SCNNode(geometry: box) //新增一個 Box
node.position = SCNVector3(0, 0, -0.5) //設定 node 在空間的位置
sceneView.scene.rootNode.addChildNode(node) //把 node 加入到目前的 scene 上
```

建一個文字形狀

文字的材質，計算原點的方式與其他幾何形狀有點不同，原點並不是在中央，而是左下方，大小的計算也與字數相關

```
let text = SCNText(string: "Hello Text in AR", extrusionDepth: 1.0)
text.firstMaterial?.diffuse.contents = UIColor.blue

let textNode = SCNNode(geometry: text)
textNode.position = SCNVector3(0, 0.05, -0.5)
textNode.scale = SCNVector3(0.01, 0.01, 0.01)

sceneView.scene.rootNode.addChildNode(textNode)
```

加入球形物件並使用圖片貼圖材質

不同的幾何形狀需不同的參數，如 SCNSphere 是球形，只需要半徑即可。除了純色之外，也可以用圖版材質貼圖

```
let earth = SCNSphere(radius: 0.3)
earth.firstMaterial?.diffuse.contents = UIImage(named: "worldmap")
let earthNode = SCNNode(geometry: earth)
earthNode.position = SCNVector3(0, 0, -1)
sceneView.scene.rootNode.addChildNode(earthNode)
```

如何在 AR 中點選物件？

先加入一個手勢感應器到場景中

```
let gesture = UITapGestureRecognizer(target: self, action: #selector(taped(sender:)))
sceneView.addGestureRecognizer(gesture)
```

當點擊手勢發生時，試試看有沒有點到東西

```
@objc func taped(sender:UIGestureRecognizer){
    let view = sender.view as! SCNView //由傳送者取得 ARView 的實體
    let location = sender.location(in: view) //取得點選的畫面座標
    let hitResult = view.hitTest(location, options: nil) //試試看能不能點到東西
    if hitResult.isEmpty != true{
        print("some thing!")
    }else{
        print("nothing!")
    }
}
```

改變點選物件的顏色

使用亂數變色，若點到東西，可以取得 node 並用同樣的方式設定材質即可，arc4random() 為亂數產生器，產生 0~UInt32.max 之間的亂數

```
@objc func taped(sender:UIGestureRecognizer){
    let view = sender.view as! SCNView //由傳送者取得 ARView 的實體
    let location = sender.location(in: view) //取得點選的畫面座標
    let hitResult = view.hitTest(location, options: nil) //試試看能不能點到東西
    if hitResult.isEmpty != true{
        let randomColor = UIColor(
            red: CGFloat(arc4random()) / CGFloat(UInt32.max),
            green: CGFloat(arc4random()) / CGFloat(UInt32.max),
            blue:  CGFloat(arc4random()) / CGFloat(UInt32.max),
            alpha: 1.0)
        hitResult[0].node.geometry?.materials[0].diffuse.contents = randomColor
    }
}
```

尋找空間中的平台(PLANE)

Plane 是用來放置 node 的位置，例如空間中有一個桌子，東西就應該放到桌子上，而不是浮在空中，也可以開啟物理引擎，當東西掉到地版或桌面時，就應停步掉落，或彈回來，ARKit 會依據影像與 LiDAR 來偵測 Plane 的實際位置存在。

如需偵測 plane 只需在AR 的 configuration 告知設定即可，系統會自動偵測

```
configuration.planeDetection = .horizontal
```

如有偵測到新的 plane，就會在以下方法回應

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {}
```

要顯示偵測到的平台

Plane 本身是不會顯示的，需要我們自己加入一個 Overlay 並在找到其 plane 時，蓋一層自訂義的上去

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    if !(anchor is ARPlaneAnchor) { return } //確定找到加入的是 plane
    let plane = OverlayPlane(anchor: anchor as! ARPlaneAnchor) //產出自訂義的可視平台
    self.planes.append(plane) //新增到 ViewController 的記錄中
    node.addChildNode(plane) //把自訂義的可視元件，蓋一層到平台上
}
```

自訂義的可視 node ~ 1

```
import UIKit
import SceneKit
import ARKit

class OverlayPlane: SCNNode {
    var anchor :ARPlaneAnchor
    var planeGeometry :SCNPlane!

    init(anchor :ARPlaneAnchor) {
        self.anchor = anchor
        super.init()
        setup()
    }
    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
```

自訂義的可視 node ~ 2

```
    func update(anchor :ARPlaneAnchor) {
        self.planeGeometry.width = CGFloat(anchor.extent.x);
        self.planeGeometry.height = CGFloat(anchor.extent.z);
        self.position = SCNVector3Make(anchor.center.x, 0, anchor.center.z)
    }

    func setup() {
        self.planeGeometry = SCNPlane(width: CGFloat(self.anchor.extent.x), height: CGFloat(self.anchor.extent.z))
        let material = SCNMaterial()
        material.diffuse.contents = UIImage(named:"overlay_grid.png")
        self.planeGeometry.materials = [material]
        let planeNode = SCNNode(geometry: self.planeGeometry)
        planeNode.position = SCNVector3Make(anchor.center.x, 0, anchor.center.z);
        planeNode.transform = SCNMatrix4MakeRotation(Float(-Double.pi / 2.0), 1, 0.0, 0.0)
        //向(1,0,0) 座標轉 二分之π 即 90°
        self.addChildNode(planeNode)
    }
}
```

3D 的旋轉

```
planeNode.transform = SCNMatrix4MakeRotation(Float(-Double.pi / 2.0), 1, 0.0, 0.0)
```

中的第一個參數為角度，一個 PI 為 半徑，所以 PI/2 即 九十度

後面三個是方向座標，即 X, Y, Z 三軸的目標方向

以上案例為 X 軸轉九十度，其他兩個方向不變

Plane 的更新

Plane 是會一直偵測與更新的，所以在更新時，也要做顯示的修正，寫在 renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) 方法中

可用 .filter 來找是那一個 plane

```
func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) {
    if let plane = self.planes.filter { plane in
        return plane.anchor.identifier == anchor.identifier
    }.first{
        plane.update(anchor: anchor as! ARPlaneAnchor)
    }
}
```

加入物理引擎

ARKit 中已包含物理引擎，可以做類似物品掉落的功能，先修改點擊時新增物件的方法

```
@objc func taped(sender:UIGestureRecognizer){
    let view = sender.view as! ARSCNView
     //由傳送者取得 ARView 的實體, 必需為 ARSCNView 才能偵測 plane
    let location = sender.location(in: view)
    let hitResult = view.hitTest(location, types: .existingPlaneUsingExtent) //試試是否是點到 plane
    if let firstHitResults = hitResult.first{
        self.addSphere(hitResult: firstHitResults)
    }
}
```

新增一個球形 node 加上物理特性

只需在 node 設定 .physicsBody 並設好參數，即可加入物理物特性

```
    @objc func addSphere(hitResult:ARHitTestResult){
        let sphere = SCNSphere(radius: 0.075)
        let material = SCNMaterial()
        material.diffuse.contents = UIImage(named: "worldmap")
        sphere.materials = [material]
        let sphereNode = SCNNode(geometry: sphere)
        sphereNode.position = SCNVector3(hitResult.worldTransform.columns.3.x,
hitResult.worldTransform.columns.3.y+0.5, hitResult.worldTransform.columns.3.z)
        sphereNode.physicsBody = SCNPhysicsBody(type: .dynamic, shape: nil)//加上物理特性並啟動
        self.sceneView.scene.rootNode.addChildNode(sphereNode)
    }
```

在 OverlayPlane 也加入物理特性

在 setup 時

```
planeNode.physicsBody = SCNPhysicsBody(type: .static, shape: SCNPhysicsShape(geometry: self.planeGeometry, options: nil))
```

在 update 時

```
planeNode.physicsBody = SCNPhysicsBody(type: .static, shape: SCNPhysicsShape(geometry: self.planeGeometry, options: nil))
```