國立臺東大學
執行教育部109年度高等教育深耕計畫
2020打造綠色國際大學

校園示範點校園創作成果報告書

議題：C1-2：
多項技術之校園導引系統

主持人：王忍成 助理教授

學生：莫翔宇　10611231
陳冠樺　10611240
侯天龍　10611242
魏發正　10611245

中　華　民　國　　１１０　年　6　月

一、 摘要

關於新生或是遊客來到一個陌生的校園所面對的第一個問題，那就是方向的問題，而紙張的校園導覽不僅可能會有看不懂或是不夠詳盡的問題，更造成了許多紙張的浪費，這些紙本的校園導覽地圖，很可能僅僅是被使用一次之後就被丟棄，那麼如果將其改成電子的呢?更讓其行程只要輸入想去的地方就可以幫你自動找尋最佳的路徑的 APP 呢?

這個 APP 在戶外可以使用 GPS 衛星定位來達到高準確度的定位，並且在你打開 APP 之後首先使用者必須先選擇目前所在的地方為室內或者室外，這樣可以使得我們程式能精確的切換室內外的定位方式，之後可以先從下拉式選單來選擇你目的地的大範圍(例如:理工學院、師範學院、人文學院....等等)，選擇完你的大範圍之後再選擇小範圍的部分(例如:哪一間教室)，選擇完畢之後 APP 將會運用 A*路徑演算法來計算最佳路徑。

而如果到了室內 GPS 收不到的範圍則會提醒使用者必須切換室內外定位，會在螢幕下方有一顆室內外的開關，只要將其打開則會切換成 Wi-Fi 定位和感測器定位來輔助定位使用者所在的位置，並且使用者可以透過手機螢幕上的提示訊息來讓使用者知道是否到達自己的目的地，並且當使用者看不懂地圖的時候也可以跟著螢幕左下角所顯示羅盤指針方向走即可，這樣可以讓人們更輕鬆地找到自己想去的地方，並且也能更盡興地遊覽這個校園。

。

關鍵字： A*、定位、建模

目　錄

圖目錄

表目錄

二、 研究背景及動機

研究動機：

隨著時代的發展，科技的進步，紙張製作的地圖已然不能滿足人們的需求，而且紙張製作的地圖大多都只用一次就會被丟棄，既浪費資源也不環保，據統計，每箱 A4 紙約 25 公斤（5,000 張紙），每噸紙漿約需砍伐 24 棵平均高度 12 公尺、直徑 15 至 20 公分的樹木，每噸紙漿可生產 40 箱紙，因此生產 1 箱紙約須砍伐 24／40＝0.6 棵樹，一棵 50 年的大樹，按木材產出計算其價格約為新台幣 10 萬，但這只是樹木真正價值的一小部分，度加爾各答農業大學(Haryana Agricultural University)達斯教授(Dr. Dess)曾經對一棵樹的生態價值進行計算，一棵樹齡 50 歲的樹，以累計計算，每年平均釋放一噸的氧氣計，50 年產生氧氣的價值約新台幣 100 萬元、防止大氣污染的價值約新台幣 210 萬、增加土壤肥力價值約新台幣 100 萬元、產生蛋白質價值約新台幣 8.5 萬元，這其中不包含花、果實和木材價值，綜合出的生態價值約新台幣 418.5 萬元。就以實體發票的浪費為例，從 2018 年全年數據來看，整年共開出 71.8 億張，其中只有約 12.8 億張利用載具存進雲端，多數消費者仍索取紙本「電子發票證明聯」，印出來浪費的紙張高達 59 億張，59 億張電子發票證明聯，相當於等於砍了 11.6 座大安森林公園的樹、可以堆出 1,160 棟的 101 大樓。

藉有電子發票的靈感，以及保護我們賴以生存的家園——地球，我們想以自己所學，為保護地球做出貢獻，減少紙張的使用量，為此，打造一款校園導覽的 APP，透過 APP 來進行引路，以到達最短路徑，既可以快速到達想要去的地方，也不浪費時間，在 APP 上透過 AR 技術，也就是擴增合虛擬與現實，可以標註校園地標性建築，也可以對建築進行簡短的介紹，讓學生、遊客及第一次到校園來的人可以快速了解這棟建築。

圖 1、台東大學校區平面圖

圖 2、台東大學知本校區平面圖

研究背景：

圖 1 圖為使用 Google 導航的台東大學路線圖，圖二為台東大學知本校區平面圖，可以看出兩者的共同點——非最佳路徑，平面圖只標出了大概的路線，而 Google 導航圖標示的路大多是給車開的，如果用步行，會浪費許多時間，即不適合用走的，而我們要做的 APP 是人步行時所走的最佳路線，為此 ，我們將功能分為感測器、A star 演算法、Wi-Fi、建模等四個部份來完成研究。而重點會集中在感測器、A star 演算法、Wi-Fi 定位來進行說明。

感測器研究背景：

感測器的輔助就是使用手機的各種感測器測量得到的數值來做為運算位移的方式[1]，而通常礙於手機的感測器較於廉價，所以得出來的值誤差也會比較大，所以最後算出的位移也會有所誤差。

首先我們可以用 G-SENSOR 以及 GYROSEOPE 得到的值，也就是使用所謂的重力加速度感測器加上陀螺儀來取得他的加速度值，之後根據牛頓第二運動定律[2]。

$$\vec{a} = \frac{d\vec{v}}{dt} \quad \text{and}$$

$$\vec{s} \int \vec{v} \cdot dt = \int (\int \vec{a} \cdot dt) dt \quad \vec{v} = \frac{d\vec{s}}{dt} \Rightarrow \vec{a} = \frac{d(d\vec{s})}{dt^2}$$

圖 3、加速度轉位移公式

加速度是指一個物體速度對時間的變化率，而速度則是該物體的位置對時間的變化率。用數學專有名詞來說，速度就是位置對時間的微分；加速度則是速度對時間的微分。假定速度為零之下，可將牛頓第二運動定律簡化成積分是微分的逆運算，當得知某物體的加速度資訊時，便可利用連續兩次積分將加速度的資訊轉換成位移。

Wi-Fi 研究背景：

室內定位發展已久，因應不同條件需求而有多項不同的定位方式，以下我們將研究數種 Wi-Fi 定位方法來完成我們的室內定位。

1. TOA 定位法(Time of Arrival)：又稱為 TOF(Time of Flight)，利用移動點發送訊號到基地台的傳遞時間以計算他們之間的距離。電磁波傳遞為光速，得到傳遞的時間差值可以計算移動站與基地台的之間的距離，當存在三台以上基地台時我們 可以利用公式來得出移動站的座標。[3]

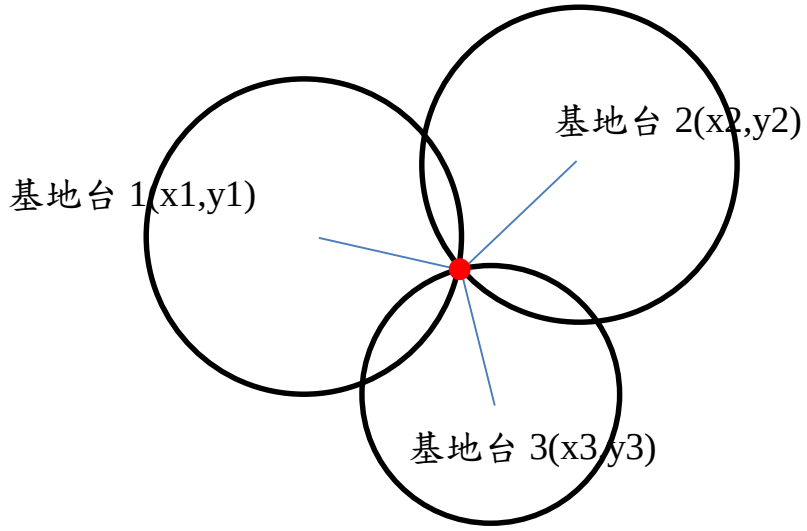

圖 4、TOA 定位法示意圖

$$r_i^2 = (X_i - X_m)^2 + (Y_i - Y_m)^2$$

圖 5、TOA 公式

i=1,2,3……
xi,yi=基地台 i 得座標
xm,ym=移動站 m 的座標
r=移動站 m 與基地台 i 之間的紀離

2. TDOA 定位法(Time Difference of Arrival)：TOA 算法的延伸以電磁波傳遞時間差作為計算距離的方法，以多個基地台的收到訊號的時間差來做計算移動點的位置。[4]

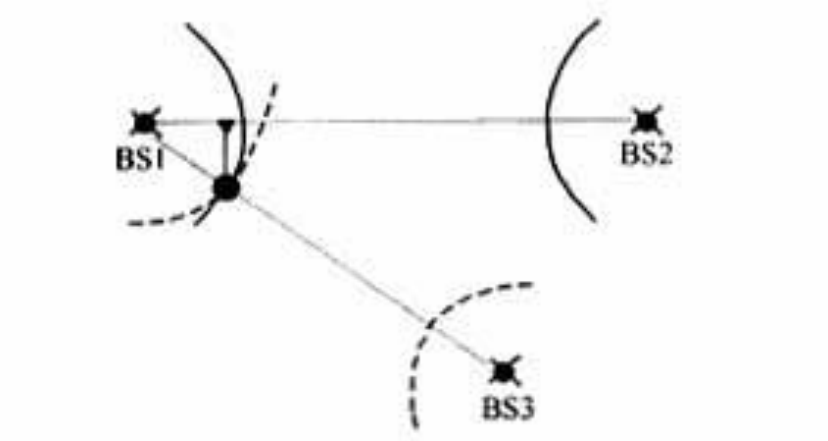

圖 6、TDOA 定位法示意圖

公式：

$$\sqrt{(x-x_1)^2+(y-y_1)^2}-\sqrt{(x-x_3)^2+(y-y_3)^2}=c(t_1-t_3)$$

$$\sqrt{(x-x_2)^2+(y-y_2)^2}-\sqrt{(x-x_3)^2+(y-y_3)^2}=c(t_2-t_3)$$

圖 7、TDOA 公式

c=3*10^8m/s　移動點(x,y)

基地台 1 (x_1,y_1) 基地台 2 (x_2,y_2) 基地台 3 (x_3,y_3)

3. 訊號強度定位法：利用訊號強度對於距離的衰減狀況取得距離以基地台為圓心形成圓，需要最少 3 個基地台的圓來進行定位。[5]

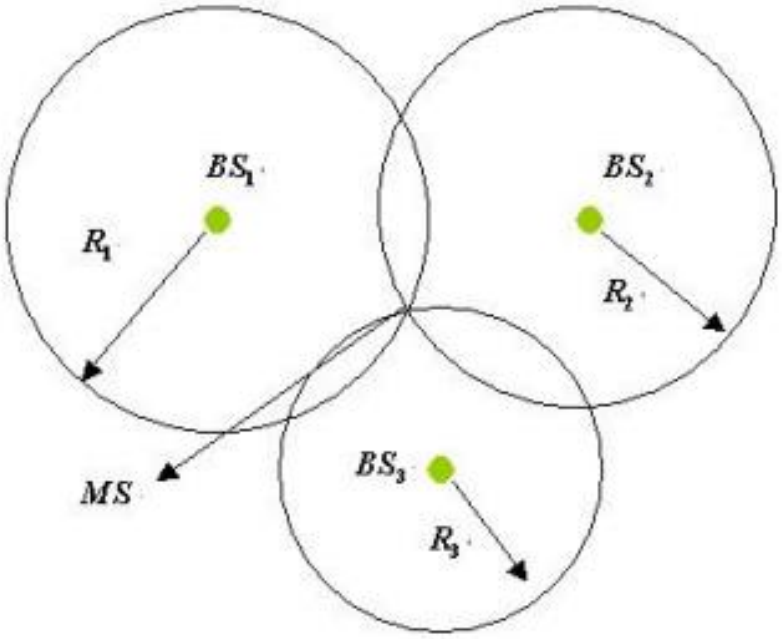

圖 8、訊號強度定位法示意圖

4. Wi-Fi 指紋：在 2000 年由 Bahl 等學者提出訊號紋比對法(Fingerprinting)，利用 Wi-Fi 技術進行室內定位，將其定位系統命名為 RADAR (Radio Detection and Ranging) (Bahl & Padmanabhan, 2000)。[6]

   分為離線階段與在線階段；離線階段，採樣定位區域所有的採樣點，建立指紋庫。伺服器把指紋庫中相同採樣點的指紋形成指紋圖（Radio Map），並使得指紋圖表達採樣點的特徵。在線階段為定位階段，從伺服器下載指紋圖，並測量信號強度（RSSI），然後利用合適的匹配算法確定位置[7]

A star 演算法研究背景：

A star 演算法（A* search algorithm）最初發表於 1968 年，由 Stanford 研究院的 Peter Hart,Nils Nilsson 以及 Bertram Raphael 發表。A star 是一種在圖形平面上，有多個節點的路徑，求出最低通過成本的演算法。常用於遊戲中的 NPC 的移動計算，或網路遊戲的 BOT 的移動計算上，是通常用來解決最短路徑(Shortest Path)問題的一種演算法，相對於另一個知名的 Dijkstra 演算法來說，Dijkstra 演算法雖然可以保證找到一條最短的路徑，但不如 A star 演算法這樣簡捷快速。同時，Dijkstra 的搜尋深度在某些情形下也容易顯得不適用。A star 演算法便是為了這些情形而出現的,可以算是 Dijkstra 演算法的一種改良。由於 A star 有啟發函式的引導，使它擁有很好的性能。[8]

啟發函式：F(n)=G(n)+H(n)，F(n)是節點 n 的綜合優先級。當我們選擇下一個要遍歷的節點時，我們總會選取綜合優先級最高（值最小）的節點。 G(n)是節點 n 距離起點的代價。H(n)是節點 n 距離終點的預計代價。[8]

1. 如果 G(n)為 0，即只計算任意頂點到目標的啟發函式，而不計算起點到頂點的距離，則演算法轉化為使用貪心策略的最良優先搜尋，速度最快，但可能得不出最佳解。[8]
2. 如果 H(n)不大於頂點到目標頂點的實際距離，則一定可以求出最佳解，而且越小，需要計算的節點越多，演算法效率越低，常見的評估函式有──歐幾里德距離、曼哈頓距離、切比雪夫距離。
3. 如果 H(n)為 0，即只需求出起點到任意頂點的最短路徑，而不計算任何評估函式，則轉化為單源最短路徑問題，即 Dijkstra 演算法，此時需要計算最多的頂點。

照我們預想來看，我們的 APP 所看到的畫面會是 3D 立體的，所以我們的 A star 演算法就需要改變。目前有三個方法，第一個是分為兩個平面，垂直平面與水平平面，分別來計算；第二個方法則是，在一個平面設置一個跳點，當使用者經過跳點，就會顯示另一個平面；第三個方法是做三維的 A star，將各個樓層的模型對齊之後，記

錄好樓梯節點，藉由樓梯節點完成跨樓層，以實現三維 A star 演算法。

Unity 簡介：

Unity 作為跨平台 2D/3D 遊戲引擎，無論是 Windows、Linux 還是 MacOS 都可以使用 Unity 來開發，在我們專題構想初期，是一個不錯的工具，畢竟它還可以結合 3D 建模，A star 演算法也可以直接在此平台上 coding，且 Student 免費版或 Personal 免費版提供的功能就已經足夠使用，加上 Unity 有許多的線上教學資料，不需要自己從頭摸索，節約了不少時間，但本身還有一個最大的問題，多人使用 APP 請求路徑規劃，系統就會沒辦法負荷，效能存在很大的問題，在之後遇到的困難，以及直接做在 Android Studio 原生平台更方便等原因，我們停止了對 Unity 相關研究，僅是作為 3D 轉 2D 平面座標點的工具。

Blender GIS：

GIS(Geographic Information System)為地理資訊系統，是資訊系統的一類，能藉由空間資料與屬性資料兩種資料描述地理環境。Blender GIS 是一款免費套件，能直接在 Blender 軟體中使用 Google Map 選定區域，並藉由 Google 提供的地理參照資料生成 3D 模型。

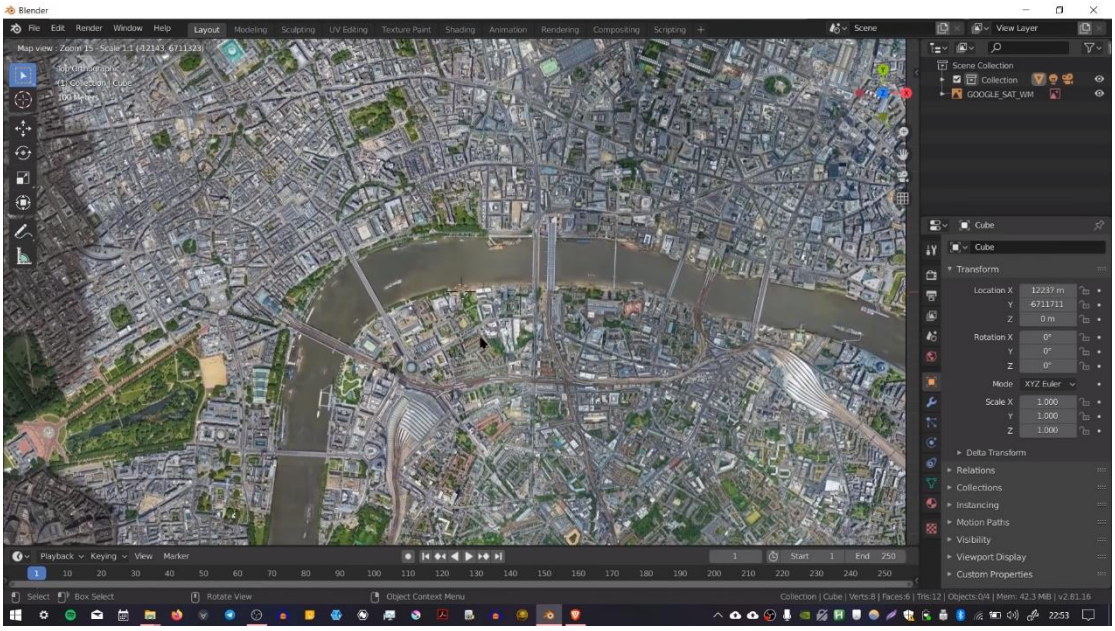

圖 9、使用 Google Map 選取區域

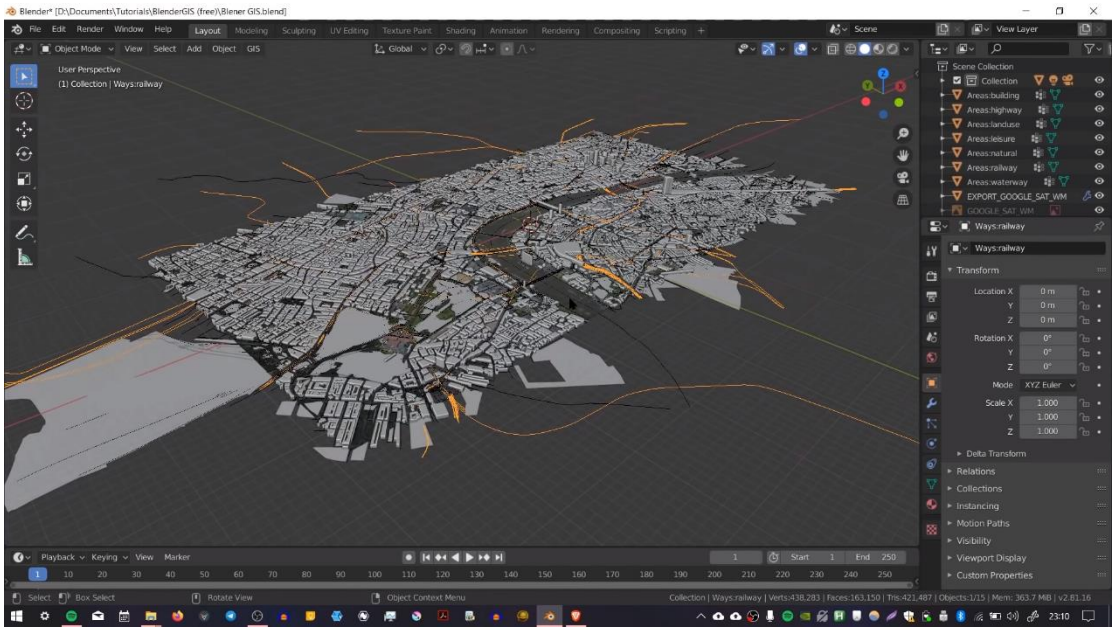

圖 10、使用 GIS 功能產生建築物模型

ARCore 研究背景：

為 Google 所開發的軟體開發套件，使用者可以利用這個套件來創建擴增現實的應用程式。

其中擁有三項關鍵的技術，分別為:

1.「六自由度」:

使手機可以了解並且去追蹤設備所在的相對位置

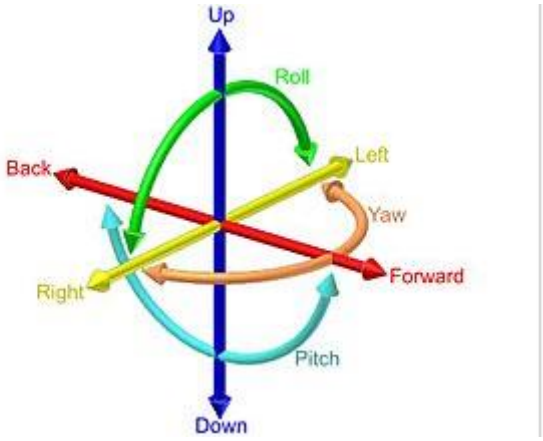

圖 11、六自由度

2.「環境理解」:

手機可以藉由此種技術去探測地面或桌面等水平面的大小和位置。

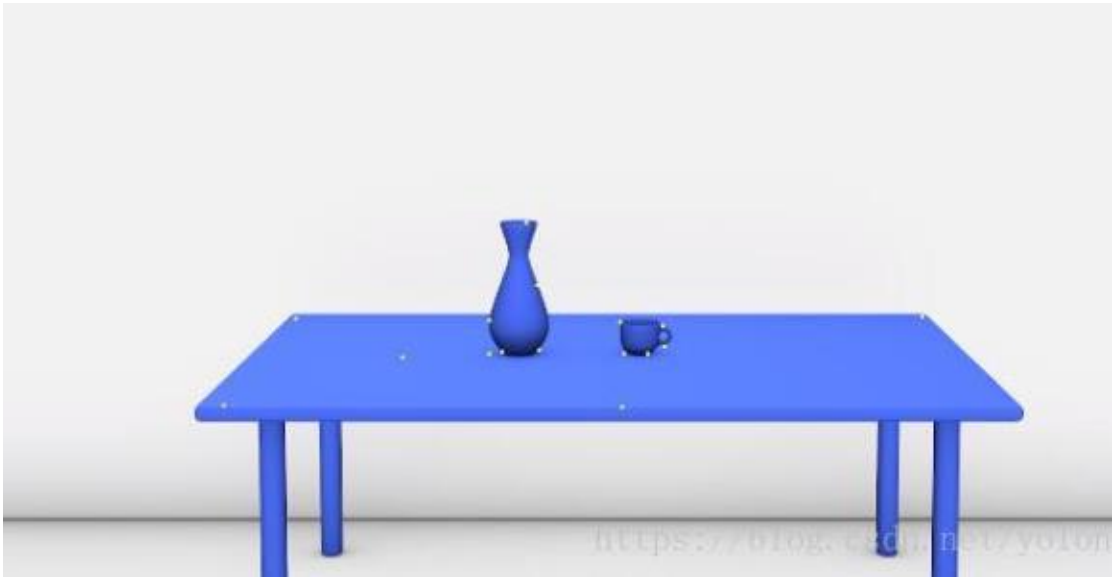

圖 12、環境理解

3.「光線評估」:

手機可以藉此估算出當前環境的照明情形，並且使得添加的 AR 物件擁有光影感，能夠更貼近周遭環境。

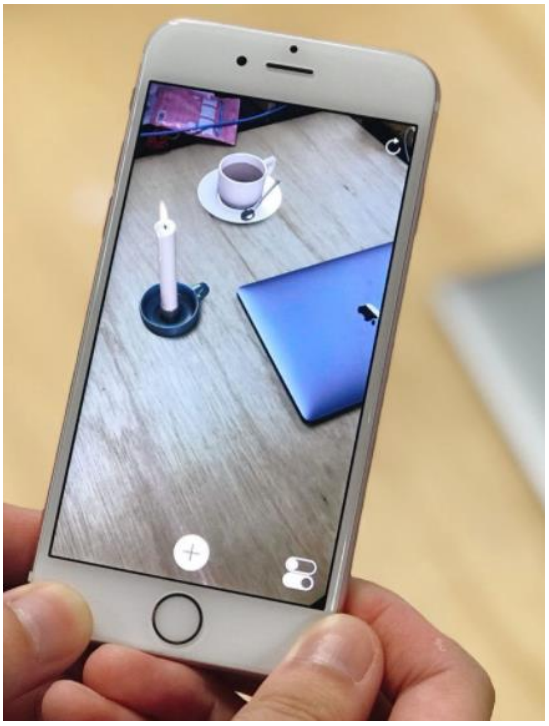

圖 13、光線評估

想要使用這些技術需要使用者手機本身的系統在 Android 8.1 以上才可以運行，所以這些技術在舊式的手機是無法運行的。

此外 Google 本身也有提供各種功能的範例，如下圖

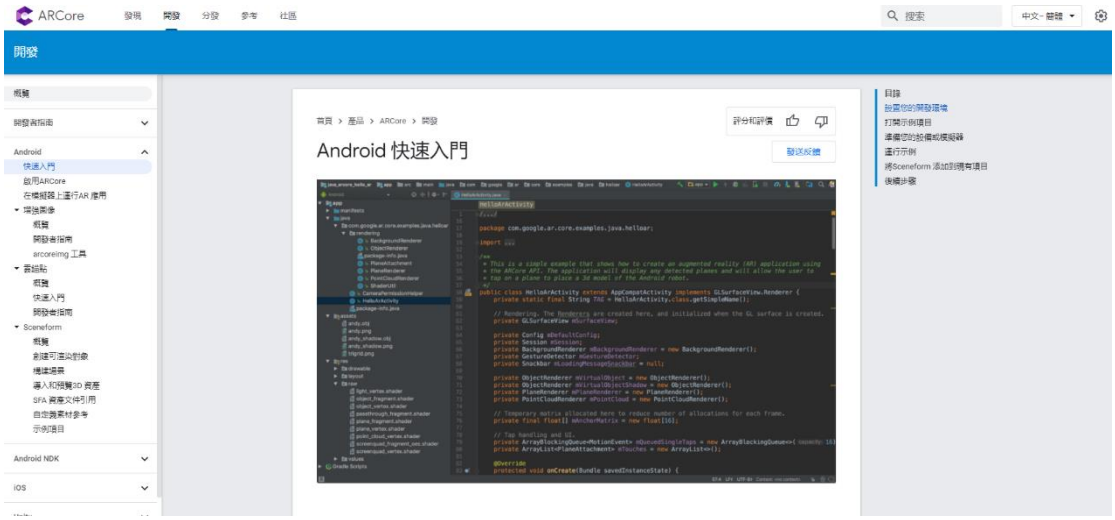

圖 14、Google 提供範例

透過先呼叫 ARCore 的 API 之後即可使用許多其相關功能，其中包括了:

表格 1、ARCore API

識別 & 增強圖像	借助 ARCore 中的識別和增強圖像功能，開發者可提供一張平面的圖像，程式將會在掃描到那張圖像後會連同位置將其鎖定，之後就可以透過提供其他 3D 模型將掃描到的圖片做變化，可參考圖 15。
雲錨點	此功能可以透過將同一環境中的使用者所掃描到的錨點，或是編輯的物件上傳到共同的雲端，來使得同一環境中的多個用戶可以彼此做交互並且共享，可參考圖 16。
Sceneform	使用此 API 可以透過觀察周圍光影變化等等因素來來使得使用者添加到環境中的物件會根據光影做變化，使得物件可以更加的融入周遭環境，可參考圖 17。

圖 15、識別&增強圖像

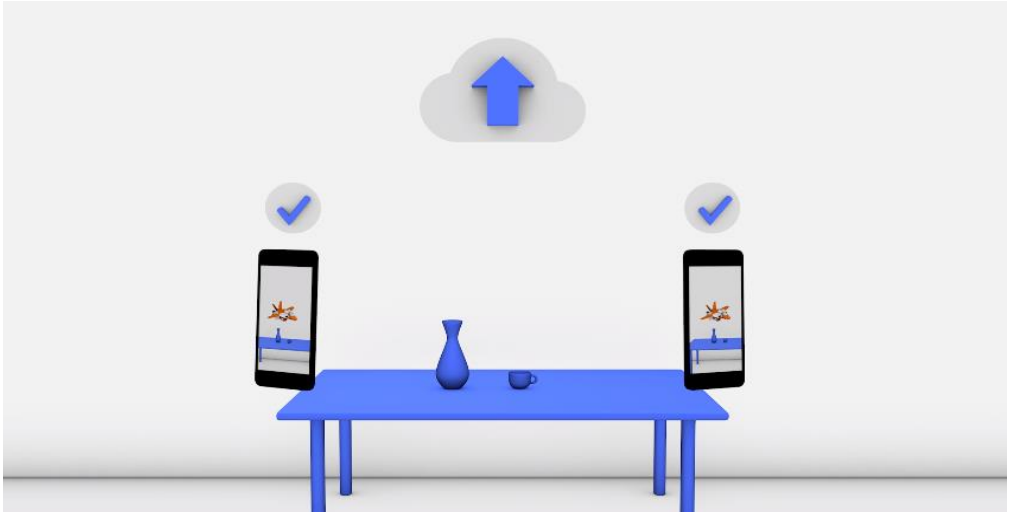

圖 16、雲錨點

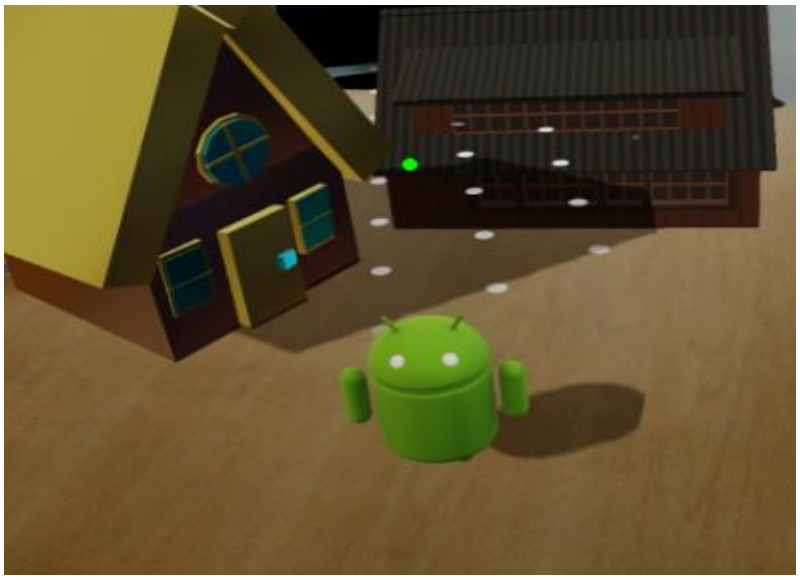

圖 17、Sceneform

AR Location-Based 技術：

AR Location-Based 技術其實是針對手機、實景、定位、AR 幾項技術所整合技術的統稱，透過手機本身的定位功能來知道使用者目前所在地，並且透過將 AR 物件添加到固定的位置(固定經緯度之類的)方式，使得使用者可以透過鏡頭看到添加的 AR 物件處於固定的位置上，這項技術應用最為人知的就是由 Nintendo 開發的 Pokemon GO 這款遊戲了，此外使用這項技術也有許多公司的 API 可供開發者使用，例如:Wikitude、Mapbox、Placenote…..等等，不過這些功能通常會根據數據流量來做收費。

圖 18、AR Location-Based

RFID 標籤：

RFID 全名為「Radio Frequency Identification」，中文稱為「無線射頻辨識」，是一種無線通訊技術，透過無線電訊號來識別特定目標並且填寫相關數據，而無須將識別系統和目標之間做接觸。

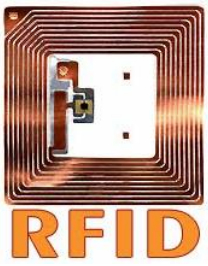

圖 19、RFID

無線電的訊號是通過調成無線電頻率的電磁場，把數據從附著在物品上的標籤上傳送出去，以自動辨識與追蹤該物品。某些標籤在識別時從識別器發出的電磁場中就可以得到能量，並不需要電池；也有標籤本身擁有電源，並可以主動發出無線電波（調成無線電頻率的電磁場）。標籤包含了電子儲存的資訊，數公尺之內都可以識別。與條形碼不同的是，射頻標籤不需要處在識別器視線之內，也可以嵌入被追蹤物體之內。

偵測方式分為兩種：

被動式標籤：

內部沒有電源，透過接收的電磁波驅動內部積體電路，接收由 RFID 讀取器發出的電磁波，當電磁波訊號強度足夠會發出數據，數據包含代碼、預存在標籤內的數據。

由於被動式標籤具有價格低廉，體積小巧，無需電源等優點。目前市場所運用的 RFID 標籤以被動式為主。

但由於被動式標籤電源驅動來自 RFID 讀取器發出的電磁波，且回傳的訊號為散射的，所以在傳播路徑衰減的影響下影響了傳播距離。

主動式標籤：

主動式標籤具有內部電源，用來供應內部需求及可隨時主動發射訊號，與被動式標籤相比擁有較長的讀取距離及較大的記憶體存附加訊息。

表格 2、RFID 種類

120 到 150 千赫茲	低頻/LF	無規定	0.1m	低速	動物識別， 工廠數據的收集	1 元
13.56 百萬赫茲	高頻/HF	全世界通用 ISM 頻段	1m	低速到中速	小卡片	0.50 元
433 百萬赫茲	特高頻/UHF	近距離設備 SRD	1-100m	中速	國防應用(主動式標籤)	5 元
868 到 870 百萬赫茲(歐洲) 902 到 928 百萬赫茲(北美)	特高頻/UHF	ISM 頻段	1-2m	中速到高速	歐洲商品編碼，各種標準	0.15 元(被動式標籤)
2450 到 5800 百萬赫茲	微波/microwave	ISM 頻段	1-2m	高速	802.11WLAN(無線局域網)，藍牙標準	25 元(主動式)
3.1 到 10 吉赫茲	微波/microwave	超寬頻	最高 200m	高速	需要半主動或主動標籤	設計為 5 元

雖然被動式標籤成本低廉，但讀取距離不足，若要在走廊運用需要每一兩公尺貼一張，可能會影響校園美觀，若要運用可能在轉角或樓梯等特別地方使用；而主動式標籤則跟藍芽一樣，由於需要電源，後續維護不易，故兩種標籤本次並不採用。

手機本身能夠透過許多於 Google Play 商店或是 App store 中能夠下載的免費軟體來達到掃描辨識 RFID 標籤的功能

手機內建感測器：

手機本身擁有許多的感測器，包括:距離感測器、光線感測器、加速度感測器(G-SENSOR)、陀螺儀、磁場感測器……等等，其中在我們的專題中，我們將會應用其中的加速度感測器、陀螺儀以及磁場感測器來實現我們的功能。

表格 3、感測器

加速度感測器	又被稱之為重力測測器，能夠感測出手機本身的重力加速度
磁場感測器	能夠藉由地球的磁場來分辨出東西南北方位。
陀螺儀	將手機本身比喻為一顆陀螺，能夠感測手機自身在空間中的角度變化。

三、 研究方法與步驟

我們將功能分為感測器、Wi-Fi、建模、演算法、ARCore 五個部份來分開研究最後再做統整

建置環境：

我們選擇在 Android Studio 平台上開發 Android app 安裝 Andoroid Studio 的步驟如下。

1. 安裝 Java 開發工具包(Java Development Kit , JDK)
首先我們置至 Java 下載點下載適當的 JDK 版本，點選 JDK Download 下載下來安裝即可。
https://javadl.oracle.com/webapps/download/AutoDL?BundleId=243737_61ae65e088624f5aaa0b1d2d801acb16

圖 20、java 下載點

2. 安裝 Android Studio

我們至 Android Studio 官網點選 Download Android Studio

https://redirector.gvt1.com/edgedl/android/studio/install/4.1.1.0/android-studio-ide-201.6953283-windows.exe

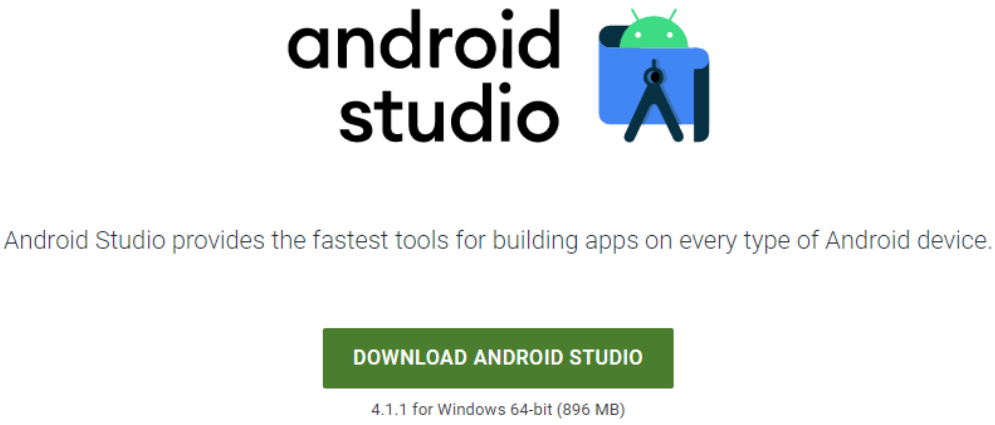

圖 21、android studio 官網

點擊安裝完成後我們將我們的附在光碟中附錄的設定檔匯入，以減少因環境設定的 bug。

點選 File Manage → IDE Setting → Import Setting 匯入設定檔。

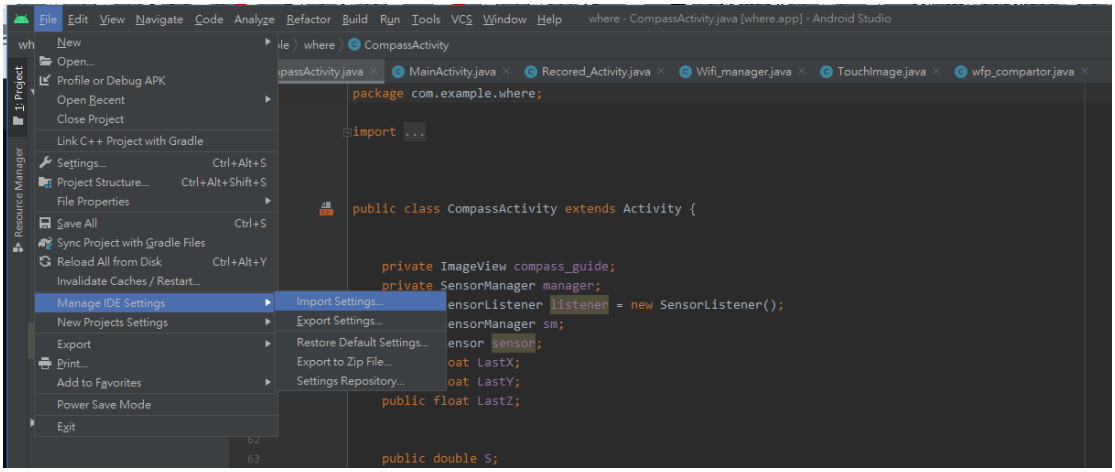

圖 22、匯入設定檔

1. 感測器研究方法與步驟

I. 室內感測器定位：

在感測器的實作方面一開始我們嘗試使用了 G-SENSOR 跟陀螺儀配合開始和結束的時間來計算出使用者位移，不過這樣做有些缺陷就是因為手機的感測器通常沒辦法很精準的算出實際上的值(會有誤差)，而那個誤差的值會令得我們算出的值越大時誤差會慢慢變大到最後跟實際上的位移的落差越來越大。

甚至可能在手機放置於桌上完全沒有動的狀況陀螺儀也會有數值上的變動，那這樣的精確度在實際應用上可能還需要更改自己的演算法，之後因為誤差太不穩定所以去嘗試了其他計算方法。

例如：使用加速度感測器來算出使用者所走的步數，手機本身也有步數感測器...等有很多的方法來使用，使用上可以使用加速度感測器來計算自己的步數乘上普通人每步平均距離，普通人每步的平均距離落在 45 到 60 公分，之後就可以在程式中呼叫感測器的函式來做使用，且我們必須要讓顯示於手機畫面中的步數是會隨著感測器得到新的數值之後更新的，所以我們將我們的 text 訊息放在一個 on Sensor Change 裡面，這樣就可以隨著使用者每走一步之後就持續更新步數訊息了，並且我們將每人一步的距離取 48 公分來做推估，以下是我們顯示訊息的方法。

```
//------------------------------------------------------------
private SensorEventListener SL=new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        long CurrentTime=System.currentTimeMillis();

        long xTime=CurrentTime-LastTime;

        if(xTime<80) return;;


        float x=event.values[0];
        float y=event.values[1];
        float z=event.values[2];

        float DX=x-LastX;
        float DY=y-LastY;
        float DZ=z-LastZ;

        S=Math.sqrt(Math.pow(DX,2)+Math.pow(DY,2)+Math.pow(DZ,2))/xTime *1000;

        if (S>=150 &&S<=200) {
            // tv.setText("初位置:" + x + "," + y + "," + z + ",末位置:" + DX + "," + DY + "," + DZ);
            i++;
            t0.setText(String.format("當前步數:%s步", String.valueOf(i)));
            String taa;
            taa = String.valueOf(i * 48);
            // t1.setText("當前觸發時間:" + CurrentTime + "，上次觸發時間" + LastTime + "，觸發間隔:" + xTime + "，速度值為" + S);
```

圖 23、顯示移動路徑長度的訊息

圖 24、感測器實作之計步器

由上頁圖片中我們可以看到使用者目前所走的步數在我們的計算中為 42 步，但當時我們在實驗時我走的步數大約為 100 步左右，所以可以看出感測器結合的步數器的精確度還有待加強或者用其他計算方式來加權使得精確度可以有所提高，此外，因為是用感測器判斷手機的晃動程度來判斷使用者是否有行走，所以若是使用者在原地一動也不動，但是卻一直晃動手機的話步數也會一直增加，這是目前如果需要將感測器定位實作在我們的程式中需要去克服的一點。

更可以設計成讓使用者在室外(收得到 GPS 的地方)，走大約 10 步，之後根據 GPS 所得到的精確位移來算出這個人的平均每步的距離為多少，這樣就可以更精確地為每個使用者量身設計他每步行走的距離，使得精確度能夠提高，不過目前做完後的測試結果顯示計步器其實常常會少算到許多步數，例如：當使用者在使用此APP時手拿的比較穩時，手機的計步器就常常算不到使用者走的步數，所以後來我們使用的室內定位中暫時放棄了以上的幾種方法。

之後我們認為還是可以利用其他的方法並且做一些權重上面的分配以達到更高的精確度，例如：使用鏡頭來跑機器學習等方法來嘗試做計算，此種計算方式我們認為在之後是個可以提高許多精確度的辦法，但是也可以預想到的是效能和精確度要如何分配的問題也會隨之產生。

II. 室外定位方法：

室外定位方法我們分成兩部分來做說明，分別為：

1. 得到當前位置訊息
2. 將當前位置訊息和校園地圖結合。

一、 得到當前位置資訊(經緯度)：

室外定位我們採取 GPS 定位的方法來做定位，GPS 定位的啟用權限可以概略分為兩種，第一種是精確位置(下圖上面行程式碼)、另一種是概略位置(下圖下面行程式碼)。

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

圖 25、啟用權限

表格 4、精確與粗略定位

精確位置	通常被稱為 GPS，此功能允許應用程式存取 GPS。 GPS 提供者和被動式提供者選項的必要項 (被動提供者需要存取另一個應用程式或服務) 所收集之 GPS 資料的許可權。 網路提供者的選擇性許可權。 耗電量高，但相對的精確度也更高，缺點是在室內時基本上很難定位到。
概略位置	此功能也被稱為 AGPS，允許應用程式存取行動電話通訊和 Wi-Fi 位置，通常用基地台等方式定位。 耗電較低，但相對的精度也較低，優點是在室內時的定位相較於前一種方法來的更為精準。

我們採用的定位方法：

在我們的專題中我們透過GPS以及網路的交互(AGPS)來得到我們的經緯度值，若是在完全空曠的室外我們直接採用 GPS 得到的值來做使用者當前的位置，若是在接近室內正準備切換室內外地圖時則可以採到 AGPS 的到的值來做使用者的當前位置，但因為我們的這種定位方法基本上只是為了室外定位而設計的，所以經過測試之後我們認為就單純只使用GPS的定位方法的話跟混合了 AGPS 定位得到的位置上的偏差也不是很明顯，這算是我們做完後比較得到的結論。

二、 將當前位置訊息和校園地圖結合：

之後配合建模好的地圖站在我們較為確定站的位置的 XY 座標的地方，透過寫一個程式持續獲得當前位置的經緯度座標，並將每次獲得的經緯度座標寫進 txt 文字檔裡，之後取得了三個點分別為 1.方外人間 2.理工學院 C 棟通往師範學院的師範學院進室內前的點 3.共教大樓 A 棟門口各 1000 次的經緯度變化(如下圖)的各種資料。

靜心書院	121.0666116	22.7367178
	121.0666349	22.7367233
	121.0666576	22.736725
	121.0666894	22.7367126
	121.0666571	22.7367156
	121.0666998	22.7367078
	121.0666976	22.7367112
	121.0666967	22.7367118
	121.0666967	22.7367117
	121.0666967	22.7367117

圖 26、經緯度變化資料

之後用將經度跟緯度的座標分別丟進 Excel 中透過平均的方式得出每個點的平均經緯度各為多少，之後去計算每兩個點的每一格X以及Y座標分別會相差多少經度和緯度，將得出的每種可能再做一次平均得出大致上一格的 XY 座標會對應多少經度緯度，之後去推估得到的經緯度為多少時會定位在地圖上

的哪一個座標，此方法因為為推估的方式，所以有些時候會因為當天天氣得到的經緯度變化而導致有不同的誤差，並且可能因為建模時的地圖跟實際上可能也有一些誤差導致定位的位置跟當前使用者位置會有 2x2 到 3x3 的位置誤差，下圖是我們當前推估出來的經緯度以及誤差。

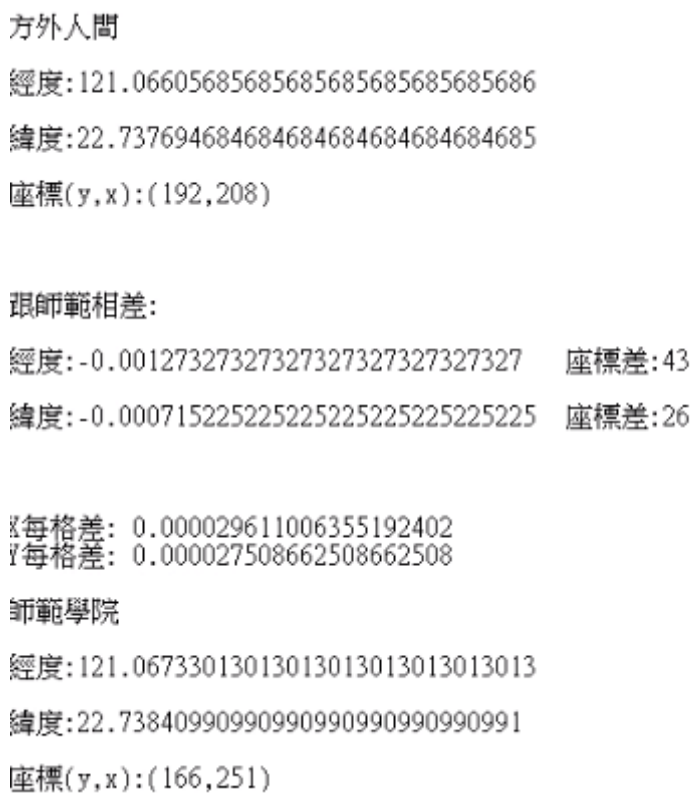

圖 27、推估經緯度以及誤差

如果想再繼續增加精準度及改良推估的方法的話可能要去計算從各個衛星中得到的位置訊息並且對其做處理，而推估地圖每隔座標相差距離的方法可能也就不能單純只用平均的方式來做計算了。

III. 路徑指引方式：

我們將路徑指引分成幾種方式：

1. 顯示使用者當前位置
2. 羅盤路徑指引
3. 文字路徑指引

一、 顯示使用者當前位置：

為了讓使用者知道自身當前位置，我們的程式會將使用者當前位置畫在螢幕上方的小地圖上來供使用者觀看，以下程式碼為顯示使用者當前位置的程式碼片段。

```
private int link_map(int i) { return file_array[i]; }

private void map_show(int floor, int num) {
    if (num > grid.size() - 1) {
        numctl = 0;
        num = 0;
        map_touchImage.init(map_ctl[num]);
    }
    if (floor > grid.get(num).length - 1) {
        floor_ctl = grid.get(num).length - 1;
        floor = grid.get(num).length - 1;
        map_touchImage.init(map_ctl[num]);
    }
    if (floor < 0) {
        floor_ctl = 0;
        floor = 0;
        map_touchImage.init(map_ctl[num]);
    }
    bitmap = drawBitmap(floor, grid.get(num), map_ctl[num]);
    int finalNum = num;
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
           // map_imageView.setImageBitmap(bitmap);
            //map_imageView.invalidate();
            map_touchImage.setImageBitmap(bitmap);
            map_touchImage.invalidate();
        }
    });
}
```

圖 28、顯示使用者當前位置

二、 羅盤指引路徑：

使用手機內建感測器可以感測地球磁場來判斷東西南北方位的功能，利用指針可以指向地球的絕對方位的能力來實做羅盤指引功能，以下將拆成三個部分來講解。

i. 指向絕對方位：

將羅盤指引到地球的正北時夾角為 0 度，並且依照逆時針的方式，90 度為西、180 度為南、270 度為東這種方法來將羅盤指到地球的各個絕對方位，並且可以透過將夾角設為 45 這種的中間值將羅盤指向西北之類的更精確方向，之後利用手機內建的陀螺儀來感測手機轉動的方向，並且因為羅盤的底座和指針都是圖片的關係，我們可以利用下圖的方法來將我們感測到手機轉了多少角度就將指針圖片往相反的方向轉相同的角度。

```
float degree = event.values[0];
RotateAnimation animation = new RotateAnimation(-predegree, degree,  //北方
        Animation.RELATIVE_TO_SELF,  pivotXValue: 0.5f,
        Animation.RELATIVE_TO_SELF,  pivotYValue: 0.5f);
animation.setDuration(200);
compass_guide.startAnimation(animation);
predegree = degree;
last_x_location = x_int;
last_y_location = y_int;
```

圖 29、指針旋轉函式

ii. 如何指向路徑的下一個點：

在最一開始時會將由前面的路徑演算法得到的路徑存成一個陣列傳過來，之後依照室內使用 Wi-Fi 定位、室外使用 GPS 定位得到的當前位置來將指針依照他們的相對位置來做指向，為了達成這個功能，當初在建模時室外地圖就有依照正上方為北方、左方為西方、下方為南方、右方為東方的方式來做建模，所以我們只要透過看路徑的下一個點在地圖的(x,y)座標中和目前位置來做指向就可以達成

這個功能，但因為室內地圖於建模時還沒想到這個狀況，所以每間室內的地圖的上下左右方位並不代表實際的北南西東方向，例如：理工學院的上下左右對到的方位和室外不同、共教大樓的上下左右對到的方位也跟理工學院不同，當我們發現這個問題時距離專題展的時間已經不足以我們做地圖的修改了，所以我們目前透過去每個地方測量他的夾角並且對照建模的地圖來得到他的地圖上下左右所對到的方向夾角應該是多少(可能出現地圖上方對應的夾角為 217 度這種狀況)，之後根據當前定位靠近路徑的地圖座標的 2x2 範圍內後就將指引的方向指向下一個更靠近終點的路徑，因為如果指定為剛好到達路徑的那個點才往下一個方向指引的話可能會因為定位的精確度問題導致可能已經走過了那個地方可是因為定位得到的位置是那格的附近一些些誤差的格子而判定使用者並沒有到這邊，從而不將使用者指向下一個該前往的路徑，下圖為理工學院下一個點在地圖中是目前位置的正上方時的狀況，地圖建模正上方為地球方位的西方。

```
if (current_build_path.get(i_location) - current_y <= -1 && Math.abs(current_build_path.get(i_location + 1) - current_x) <= 10) {
    float degree = event.values[0] + 90;
    RotateAnimation animation = new RotateAnimation(-predegree, degree, //上方
            Animation.RELATIVE_TO_SELF,  pivotXValue: 0.5f,
            Animation.RELATIVE_TO_SELF,  pivotYValue: 0.5f);
    animation.setDuration(200);
    compass_guide.startAnimation(animation);
    predegree = degree;
    last_x_location = current_x;
    last_y_location = current_y;
}
```

圖 30、部分預判函式

iii. 預判使用者路徑使得指針不會亂跳：

在我們的路徑演算法中和建模配合得出的路徑中，因為我們的建模皆以 XY 座標的方式將地圖切成一格一格的正方形，且當初障礙物建的時候沒擋到斜著走的緣故(左上左下右上右下)，所以我們將路徑演算法斜著走的功能拿掉了，所以當路徑為斜著走時他會一格向地圖的 Y 方向(上下)移動，下一格在向 X 方向(左右)移動，之後再向 Y 方向(上下)移動.....這樣反覆循環，使得如果只是純粹將指針指向下一個路徑點的方向的話在走斜線時指針導引會令使用者走得很不直覺而且浪費時間，所以我們透過將使用者後面幾步路徑的座標點放進來預判的方式來判斷，若使用者後面幾步的路徑是斜著走的話，我們就將路徑指向斜著走的方向，例如：路徑是上左上左上左，那我們就知道他是向左上方移動，我們直接將指針指向左上方來導引使用者路徑，之後因為我們地圖的格子點切的蠻細的，所以像是室外有些比較寬的路，像是共教大樓外面那條馬路的寬度就已經超過了 2 格，以及前面提到的定位精確度問題，因為我們定位時得到的格子點也可能會有偏差，但如果再將定位範圍放到更寬的話也可能影響精確度，所以在路徑為一條直線時，若他是往地圖上方走的話我們會將他的左右方向的格子模糊到 5 格內都可以算是到這附近，而上下方的格子依然是 2 格內，將模糊定位變成一個 5x2 的狀況(x,y)，他往左右走一條直線時則是 2x5(x,y)，而室內因為走道較為窄小就沒這方面的問題。

三、 文字指引路徑：

在室內時會遇到需要上下樓梯的情況，以及當室內和室外切換時在我們的程式中是需要使用者本身手動作切換的，在這種時候就需要文字來提醒使用者是否需要上下樓以及切換是內外定位。

圖 31、文字指引

上下樓以及切換室內外功能時會向上圖紅字顯示的方法來提醒使用者，其中的程式碼部份我們以一個 text 來顯示我們所要的訊息，如下圖所示。

```
if (under_or_up_floor!=0){
    if (upDn==1){
        t4.setText("請下至"+under_or_up_floor+"樓 ∠( ᐊ ┘∠)__");
    }
    else if (upDn==2){
        t4.setText("請上至"+under_or_up_floor +"樓 ∠( ᐊ ┘∠)__");
    }
}
```

圖 32、顯示訊息

2. 室內定位研究方法及步驟

I. Wi-Fi 指紋庫實作：

A. 資料庫建立：

因為指紋庫定位頻繁為了避免延遲我們將資料庫放在手機本機端省略了伺服器的建置。我們使用 SQLite Database。

SQLite 是遵守 ACID 的關聯式資料庫管理系統，它包含在一個相對小的 C 程式庫中。與許多其它資料庫管理系統不同，SQLite 不是一個客戶端/伺服器結構的資料庫引擎，而是被整合在使用者程式中。[9]

SQLite 遵守 ACID，實現了大多數 SQL 標準。它使用動態的、弱類型的 SQL 語法。[9]它作為嵌入式資料庫，是應用程式，如網頁瀏覽器，在客戶端儲存資料的常見選擇。它可能是最廣泛部署的資料庫引擎，因為它正在被一些流行的瀏覽器、作業系統、嵌入式系統所使用[10]

我們的指紋庫使用到的 table，有 2 個 table：

Build_Code：

Code(建築物的代碼)、
Build_Name(名稱)、
M_id(prime_key)、
floor(樓層數)。

Wi-Fi_Finger_Print：

floor(樓層數)。

Bssid(Wi-Fi AP 的 MAC 位址)、

Level(訊號強度)、

X(X 座標)、

Y(Y 座標)、

Z(Z 座標)。

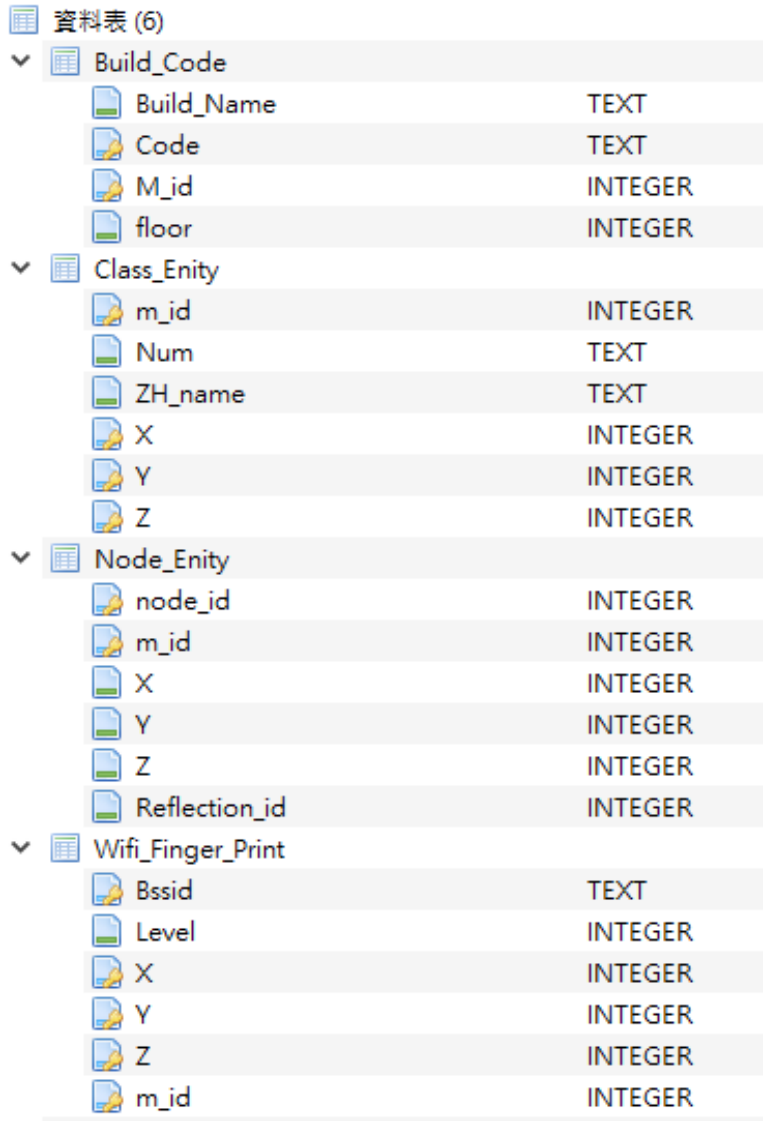

圖 33、資料庫 Table

Wi-Fi_Finger_Print 的資料在後面的資料採集部分介紹，而其他部分由附檔的 where\app\src\main\res\raw 資料夾裡面的 build.txt 裡匯入資料，格式順序為 Build_code、code、floor、M_id。

B. 資料採集：

為了使用 Wi-Fi 指紋庫我們需要採集各建築物裡的各個位置的 Wi-Fi 訊號，而採集的數量影響到精準度的計算所以需要大量採集，一個點、一個點採集太過耗時，需要能快速採集，在我們的 APP 裡面額外寫了一個透過行走來採集 Wi-Fi 訊號的程式，採集結束直接輸入進資料庫，也方便我們攜帶去各建築採集。

資料記錄方式，選擇要記錄樓層，進入紀錄頁面，輸入當前 XY 座標與終點 XY 座標，按下 Task 按鈕後開始記錄。程式會先記錄當前 Wi-Fi 訊號，且開始計時會每三秒紀錄一次 Wi-Fi 訊號，當抵達終點再次按下 Task 按鈕會停止紀錄，並計算平均位移，來將移動中紀錄的 Wi-Fi 訊號給與座標並存進資料庫。而 Record 按鈕則能紀錄單點座標(只能計算直線的 Wi-Fi 訊號，移動中請勿轉彎，須保持等速，避免造成誤差)

圖 34、Wi-Fi 訊號採集頁面

C. 定位計算：

我們嘗試了訊號強度定位法與TDOA跟Wi-Fi指紋庫三種方法。

訊號強度定位法：

為了運行訊號強度定位法，我們需要將學校架設的基地台位置一一給定座標，記入至檔案，當函式被呼叫時會對附近基地台進行掃描，將不屬於學校基地台的訊號資料排除，取出訊號最佳的三筆資料，進行計算。

但架設在天花板的 Wi-Fi 基地台跟使用者不同平面距離約 1 公尺，若忽略這個將會有更大的誤差，所以我們需要將 Z 軸加入我們的座標計算，但因為Wi-Fi多徑效應影響，他不會剛好交於一點，增加了我們計算上的困難度，及基地台分布位置不均在走廊最旁邊兩側的區塊將無法使用，所以我們放棄這定位法。

TDOA 定位法：

一樣為了進行座標計算我們將學校的基地台記入至檔案內，呼叫函式後掃描取出訊號最佳且屬於學校架設的基地台的訊號，利用圖[7]的公式進行計算，為了避免誤差也同樣加入 Z 軸進行運算，但後面我們遇到了個問題是手機測量得到的 timestamp 精度不足，移動點與基地台距離是利用光速*時間差值求得，但在手機端只能測得微秒級別的時間，光速 3*10^8(m/s)/10^6(s)=300(m)，計算誤差可能達到 300 公尺，再加上多徑路徑影響所以最後也放棄此定位法。

訊號紋比對法：

當函式被呼叫會進行一次 Wi-Fi 掃描，掃描使用者當前座標後，將訊號最佳的 Bssid 進行搜尋，找出資料庫內所有包含此 Bssid 的點，若搜尋不到則搜尋次佳的 Bssid，若搜尋不到則回傳 false。再將前面搜尋到的點的座標再進行搜尋，得到各點的所有的 Wi-Fi 訊號再與當前點的 Wi-Fi 訊號比對取相似度前三高的訊號的座標平均得到當前座標。

而訊號比對方式，我們嘗試過將相同 Bssid 的訊號取其均方差，但在訊號強度低時會造成極大的誤差，例在訊號強度-96 與-90 時會判定成比-36 與-30 誤差更大，所以我們改計算相同 Bssid 的 Wi-Fi 訊號強度的斜率，當他們強度的斜率會愈趨近於 1。

Lm=移動站訊號強度
Li 則為資料庫的基地台 i。

均方差比較公式： $\sum_{i=0}^{n}(L_m - L_i)^2$

斜率比較公式：$\sum_{i=0}^{n}(L_m/L_i)$,if $(L_m>L_i)$
$\sum_{i=0}^{n}(L_i / L_m)$,if$(L_m < L_i)$

因為強度-36:-30 跟-30:-36 對我們來說他們的距離一樣，這樣當強度相當時他們斜率為 1，他們值為最小，方便我們排序。

II. 成果實測：

A. 資料採集：

在實測中再存入資料庫的功能並沒有太大的問題，下圖為在理工學院五樓的紀錄畫面，另一張則為在資料庫裡的部分資料紀錄。測完理工大樓 C 棟部分就有上萬筆資料，在測量全校後可能會造成定位計算的延遲，且我們在測量資料時沒有排除移動基地台，若基地台移動會對定位造成影響。

	Bssid	Level	X	Y	Z	m_id
	過濾	過濾	…	…	…	過濾
19	90:6c:ac:5d:3b:8d	-68	91	154	4	1
20	a2:6c:ac:5d:3b:8d	-68	91	154	4	1
21	b2:6c:ac:5d:3b:8d	-68	91	154	4	1
22	c2:6c:ac:5d:3b:8d	-68	91	154	4	1
23	40:9b:cd:3b:b9:ec	-68	91	154	4	1
24	b2:6c:ac:5d:4c:13	-73	91	154	4	1
25	b2:6c:ac:5d:4c:0b	-73	91	154	4	1
26	c2:6c:ac:5d:4c:0b	-73	91	154	4	1
27	90:6c:ac:5d:4c:13	-74	91	154	4	1
28	a2:6c:ac:5d:4c:13	-74	91	154	4	1
29	c2:6c:ac:5d:4c:13	-74	91	154	4	1
30	90:6c:ac:5d:4c:0b	-74	91	154	4	1
31	a2:6c:ac:5d:4c:0b	-74	91	154	4	1
32	2a:5b:0e:0f:40:6a	-74	91	154	4	1
33	3a:5b:0e:0f:40:6a	-74	91	154	4	1
34	0c:9d:92:59:d3:24	-76	91	154	4	1
35	1a:5b:0e:0f:40:6a	-77	91	154	4	1
36	74:da:38:db:74:06	-78	91	154	4	1
37	c2:6c:ac:7d:d5:af	-79	91	154	4	1
38	b2:6c:ac:5d:3b:73	-79	91	154	4	1
39	90:6c:ac:5d:3b:73	-79	91	154	4	1
40	c2:6c:ac:5d:3b:73	-80	91	154	4	1

圖 35、資料庫部分資料

B. 定位實測：

在上述功能皆完成後，我們開始進行實測，因為時間不足的緣故，所以我們指採集了理工學院 C 棟的指紋來進行實測，我們將定位出的結果顯示在 UI 地圖上，使我們方便比對我們現在的位置。而比對結果程式在走廊時效果最佳，誤差約在 5 公尺內，但在或者樓梯中段、空曠處(連接理工各棟的橋)誤差極大甚至可能差距好幾層樓。

3. 3D 建模研究方法與步驟

由於室內的定位技術精度有限，需要其他方法來做為輔助以避免過大的誤差，建築的 3D 建模為本專題輔助定位的想法之一。

I. 第一階段：

i. 建模軟體選用及操作學習

ii. 取得建模所需資料

II. 第二階段：

本階段邁入建模的實作與演算法相結合，並依據結合後的測試回饋，修改模型中不利於計劃進行的部分，其步驟如下：

i. 各學院建模實作：

實際建模時依據先前實地考察所繪製完成的各學院平面圖，依其測量數值在 Blender 上實作。實作時將每一學院的各樓層獨立建模，方便後續測試及修改使用。

A. 理工學院：

最先實作時以理工大樓開始，初次建模時在學習操作並使用上花費了許多時間，也因為沒事先規劃好後續結合演算法時遇上的問題，最初的模型每一面牆

都完整呈現，並在各房間入口皆設立了門的模型。

雖然成品與實際情況較為相近，但也耗掉了近百小時的時間，後續測試也因為定位並不會完全精確，準確

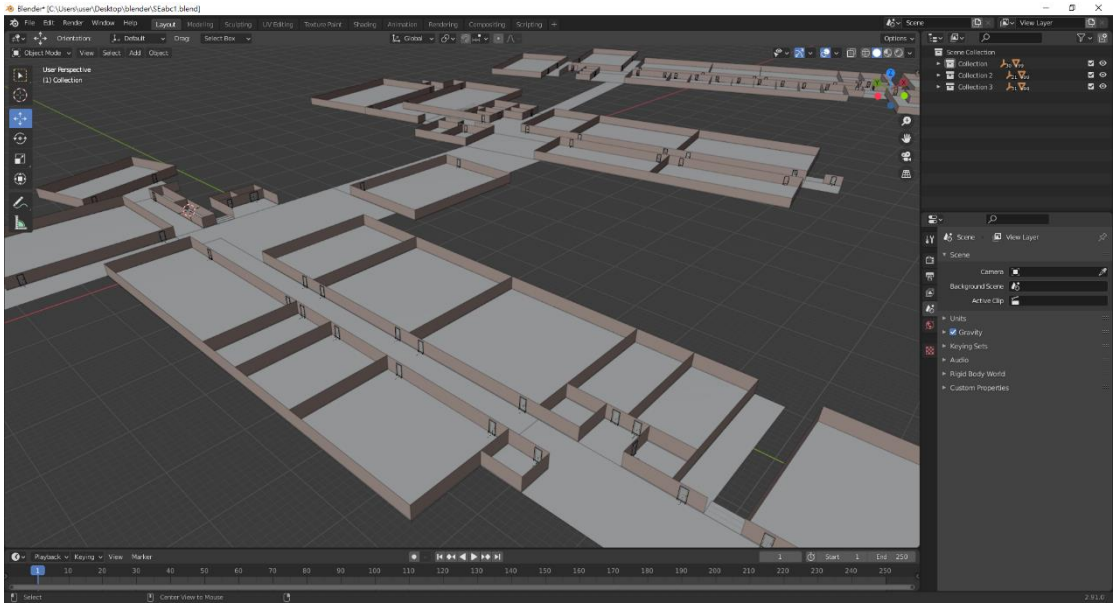

圖 36、初版理工學院一樓建模

設置的門並不一定能正常通過，可能會因為誤差而導致實際導引時會不斷地被牆壁給擋住。為了修正這個問題， 最後採用在房間面向走道方向的牆移除，不使用門的作法來解決，雖然在入口的標示上無法像原先預計的準確，但可以在避免前述問題的狀況下導引至入口大致位置。

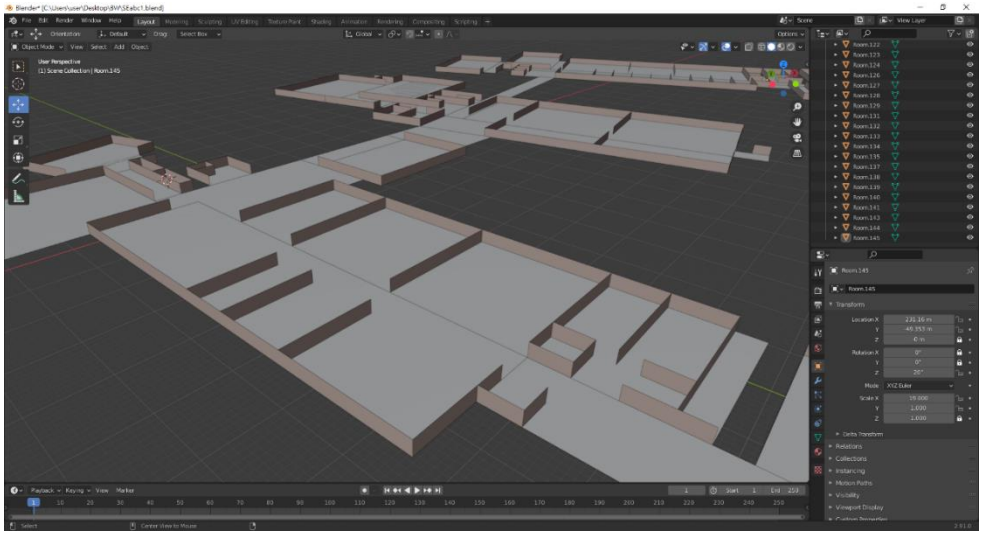

圖 37、修改後移除走道側牆壁與門的理工學院一樓模型

共同教學大樓：

鑒於理工學院建模獲得的經驗，在共同教學大樓建模操作方面沒有太多問題。僅因為共同教學大樓有些地板形狀不是方正的需要裁切。另外有可以直接從校園一樓進入共同教學大樓二樓的樓梯入口該如何表示也是在進行這棟建築建模的時候才發現的問題，當下討論後的結果是在演算法方面在樓層間採用節點的方式呈現。

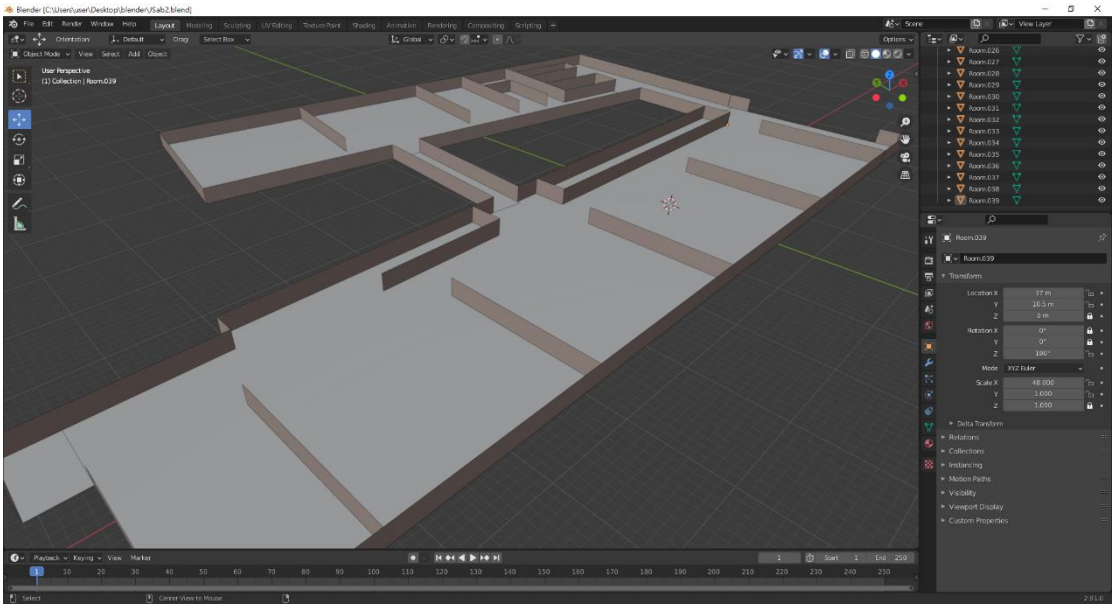

圖 38、共同教學大樓 AB 棟二樓模型

B. 行政大樓：

建模方面問題較小，反而是取得平面資料時較難實行，假日時二樓以上的樓層都關上鐵門，無法上去。平日因公務有許多人出入，且上班時間內因建築設計採光良好，雷射測距儀運作受到影響，需反覆測量才能得到數據。等到下班時間又會關上鐵門，使行政大樓在繪製平面圖時花費了大量時間才得以完成。

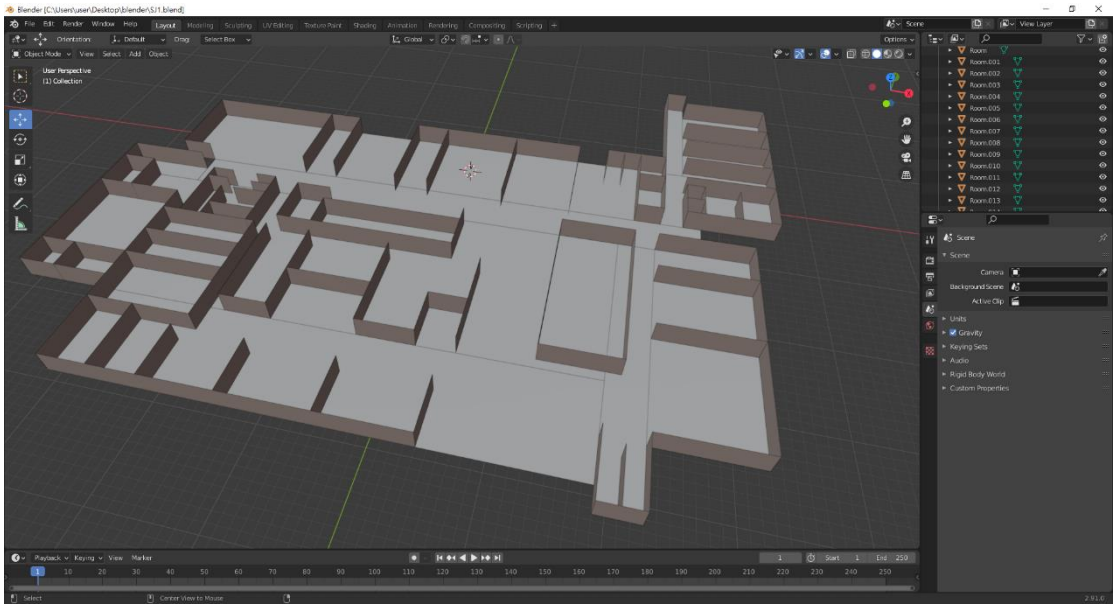

圖 39、行政大樓一樓模型

C.　師範學院：

蒐集資料時在各棟樓間的角度上遇到困難，最後是利用每棟樓之間連接的通道與該樓的夾角獲得相對可靠的角度數據後完成平面圖，至此建模的操作已經算熟練，有了平面圖輔助，後續建模基本沒有問題。

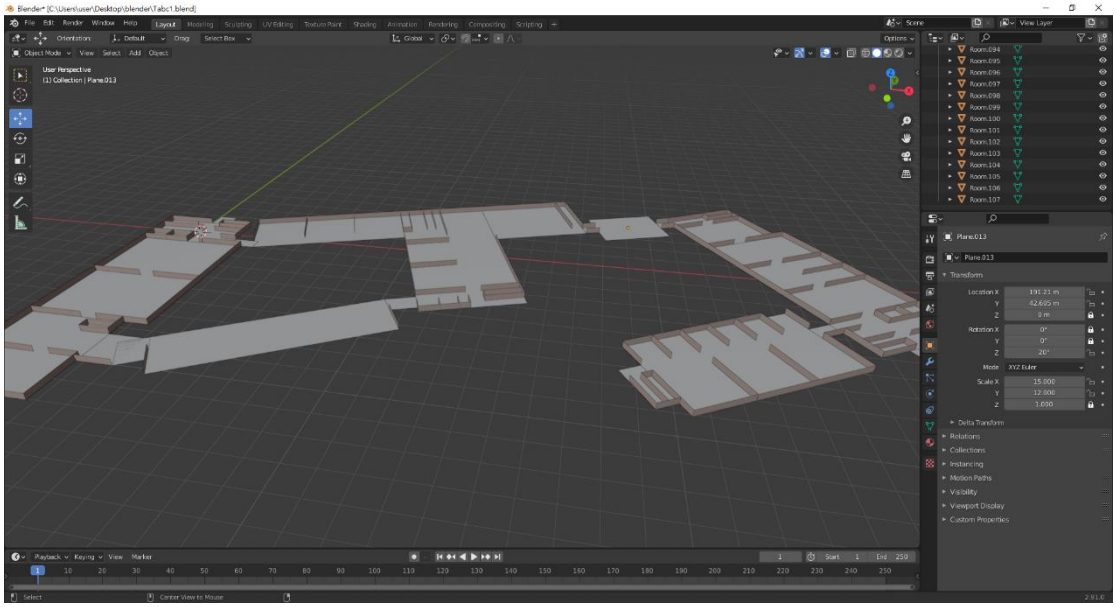

圖 40、師範學院一樓模型

D. 人文學院：

人文學院設計特殊，該棟一、二樓設計讓大多數人都是由二樓進入該學院。入口部分如同共同教學大樓是直接進入二樓的樓梯，考慮到演算法的執行方式，經討論後決定在之後校園全圖一樓製作完畢與演算法結合時再做調整，人文學院先正常建模即可。另外也因為後續樓層格局變動較多，人文學院建模花費了較多的時間，不像其他學院可以藉由類似的樓層配置方便修改。

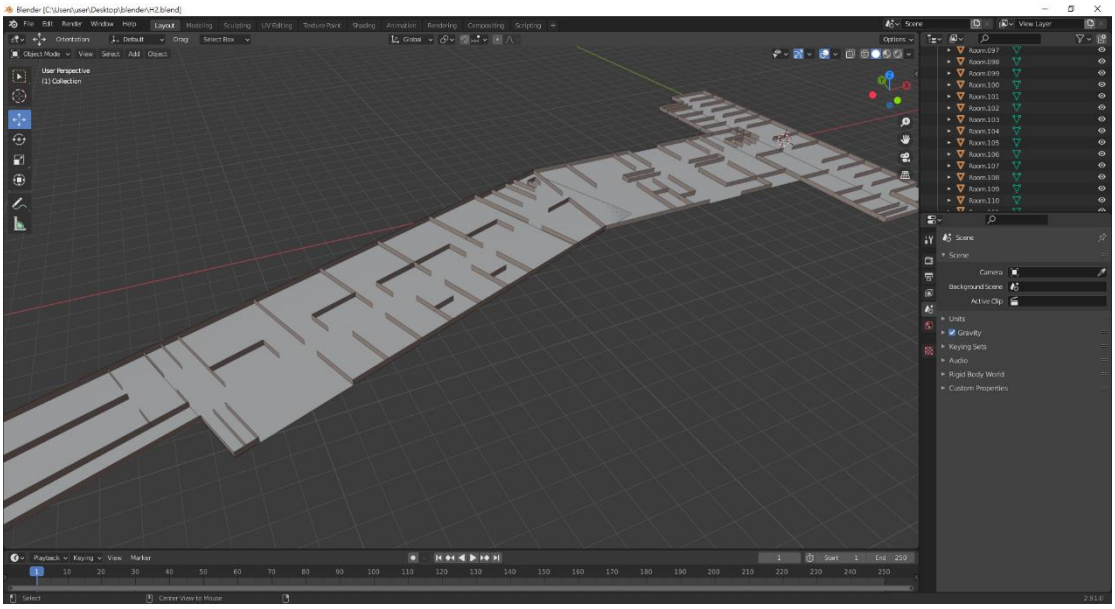

圖 41、人文學院二樓模型

ii. 將各學院導入校園全圖：

完成各學院的建模之後，下個目標是完成整體校園的一樓全圖，原定計畫是使用 Blender 內支援 Google Map 的 Blender GIS 功能，該功能可以直接套用 Google Map 的地理位置資料生成模型，免去建模所需的心力。但是礙於 Google 對於本校園區的資料採樣不足，呈現的效果差強人意。

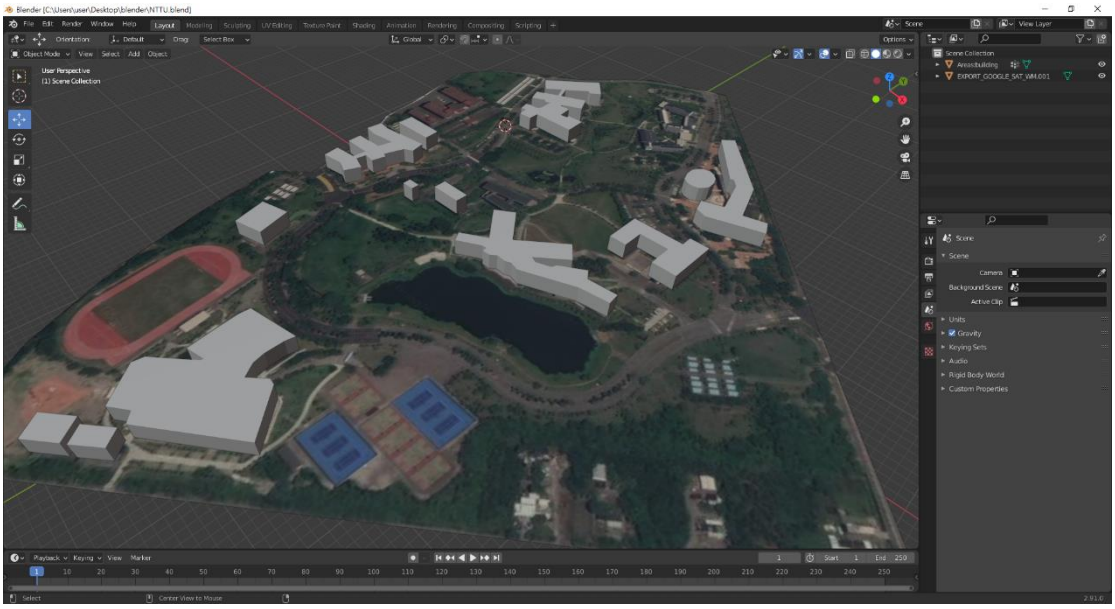

圖 42、校園使用 Blender GIS 的實作結果

最後改用以校園平面地圖為主，Google Map 的衛星雲圖及自己實際經驗為輔，自行從平面切割出符合校園內道路的校區平面模型。之後配合方便演算法執行，調整模型角度方位符合現實情況。

圖 43、參考用的校園平面地圖

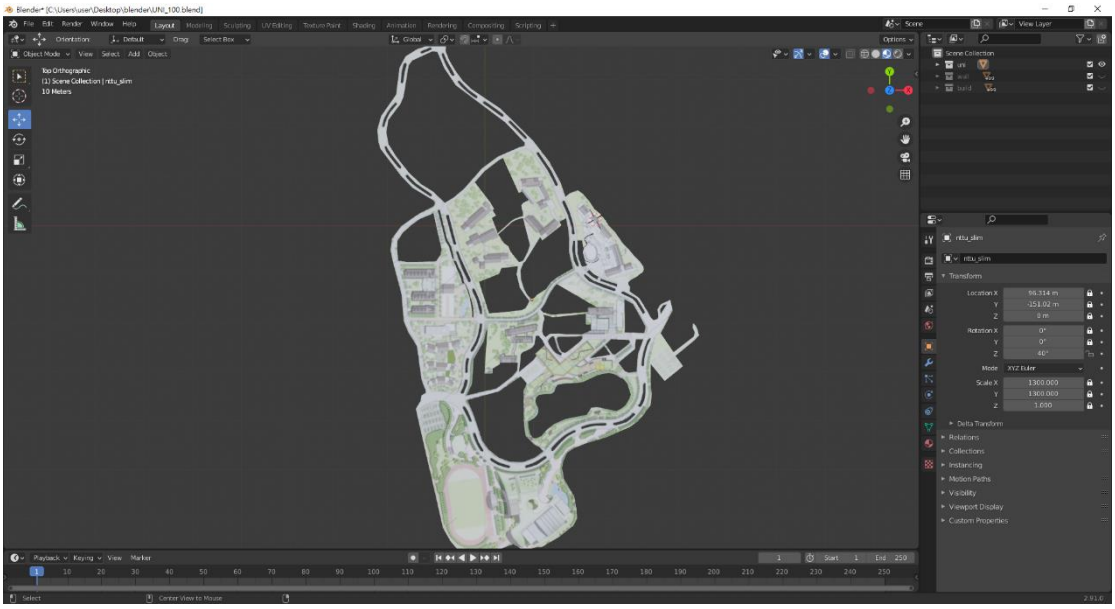

圖 44、切割調整大小與角度後的結果

完成校園平面地圖後將之前完成的各學院及其他建築的一樓模型導入，實際誤差不大，各個建築的模型幾乎完好覆蓋於該建築在校園平面上預定的範圍之內。之後將非校內人員無法隨意進出之建築設置牆壁作為阻擋後，可供測試的初期版本完成。

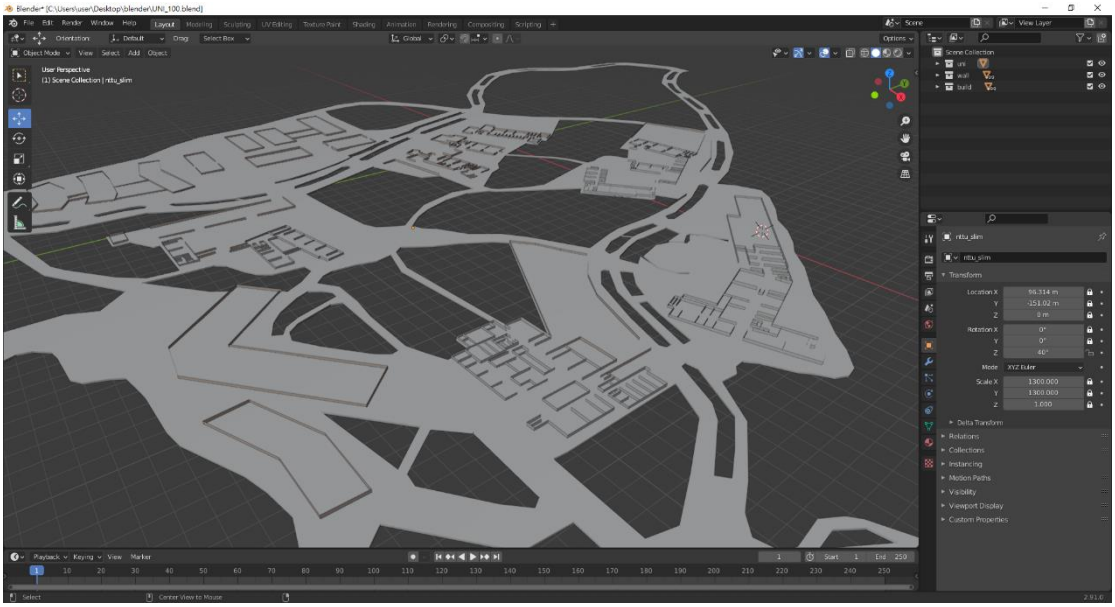

圖 45、將一樓建築模型導入校園平面實作結果

iii. 建模與演算法結合：

將完成的模型轉成 Autodesk FBX Interchange File(.fbx)檔導入 Unity 結合 A star 演算法實作地圖內點對點的較佳路徑規劃。Unity 與 A star 方面將由演算法部分詳細說明。

iv. 配合演算法修正模型以達開發需求：

根據從 Unity 上測試得到的回饋，將原本模型中不利演算法執行的因素修正，由於 Unity 上障礙物是以碰撞箱的方式訂立。Blender 方面為了操作方便進行物件群組化或是變動了物件原點的部分都需要重新修正，令碰撞箱出現的位置與大小符合物件。實際配合定位測試時，模型位置或角度偏離實際位置的微調，重新考察與修正，逐漸改善模型與演算法及定位的契合，實作精度也漸漸提高，直至能符合計畫需求。

4. 演算法研究方法與步驟

針對以上計畫目標，我們預計以二個階段完成演算法。這二個階段包括：文獻搜集、Dijkstra 分析、Floyd 分析、A star 分析、A star 實作、成果實測、未來展望。

1. 第一階段：

第一階段為演算法分析，用 Dijkstra 和 Floyd 演算法與 A 我們會使 star 做比較。我們會用的是 A star 演算法，這是非常成熟的技術，但想要使用，還是得從最基本的找資料，進行分析，以至於達到實作的結果。

I. 文獻搜集與閱讀：

我們大量閱讀有關路徑演算法的資料，例如 Dijkstra、Floyd 等等。透過之前從 Patrick Lester 和 Myopic Rhino 發佈的 A star 演算法初學者教學開始學習[11]。由此延伸出我們想要收集的資料與閱讀分析。

II. Dijkstra 分析：

由荷蘭電腦科學家 Dijkstra 發現的演算法，dijkstra 原始版本僅適用於找到兩個頂點之間的最短路徑，後來更常見的變體固定了一個頂點作為源結點然後找到該頂點到圖中所有其它結點的最短路徑，產生一個最短路徑樹。Dijkstra 雖然能找到最短路徑，而所花費的時間也會很多，以使用者的角度來看，五秒還可以等待，如果超過十秒，應該就不會使用這款 APP 了。

時間複雜度：O(|E|+|V|log|V|)。

III. Floyd 分析：

Floyd 演算法同樣可以找到兩點之間的最短路徑，Floyd 演算法本質上是一種 Dynamic Programming 的演算法，將一個問題，拆解成不同的子問題，再根據子問題的解，得出原問題的答案。優點很明顯，容易理解，程

式碼編寫相對簡單，缺點也很明顯，時間複雜度高，不適合計算大量的資料。

時間複雜度：$O(N^3)$。

IV. A star 分析：

由 Stanford 研究院的 Peter Hart, Nils Nilsson 以及 Bertram Raphael 發表。它可以被認為是 Dijkstra 算法的擴展。A star 演算法，我們路徑規劃的核心，藉 Patrick Lester 和 Myopic Rhino 發佈的 A star 演算法初學者教學開始研讀，A star 常用於 NPC 遊戲的移動的計算，我們相信它在計算時間和最佳路徑的平衡上能有很好的表現，因此決定用它來進行路徑規劃，但實際的效果，還需要實作才能做評估。

啟發函式：F(n)=G(n)+H(n)

2. 第二階段：

我們決定做三維的 A star 演算法，也就是通過將樓層模型對齊的方式來實現。之前我們打算用紅色的線條將路徑描繪出來，可是路徑在地圖上很慢才顯示出來，我們以為是手機無法負荷大量的計算，原本打算將計算交到伺服器上做。後來經過討論，又決定在資料庫上做。畢竟校園導引的最佳路徑是固定，例如理工 C 棟三樓到共教大樓 D 棟 1 樓的最佳路徑是固定的，我們只要預先計算過一次，之後將算的路徑儲存在資料庫，就可以很快的顯示路徑。但在不斷的試錯過程中，我們發現在 Android studio 建立節點的速度太慢，以至於在大量資料下，計算速度表現不佳，只要將原本的資料結構從 linked list 改為 hash，計算速度有了大幅度的提升，伺服器及資料庫即作為備選方案。

I. A star 實作：

A. 模型匯入：

這步驟與我們選擇用 3D 建模息息相關，我們用的方法是借由 Unity 這個跨平台 2D/3D 的遊戲引擎將建好的校園模型轉為 2D 平面座標點。

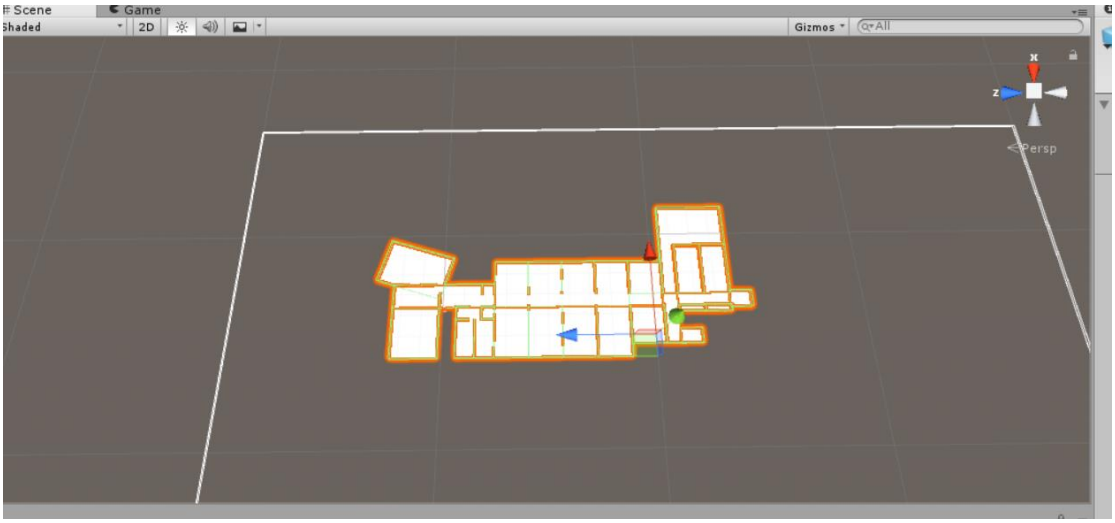

圖 46、理工大樓 C 棟五樓模型

接下來就是將模型轉化為程式的 Input 部分。我們將障礙物設為 false，可以走的地方，例如走廊，設 true，可以看到，上述圖 46 中，橘色的部分就是障礙物，白色的部分就是可以走的地方。

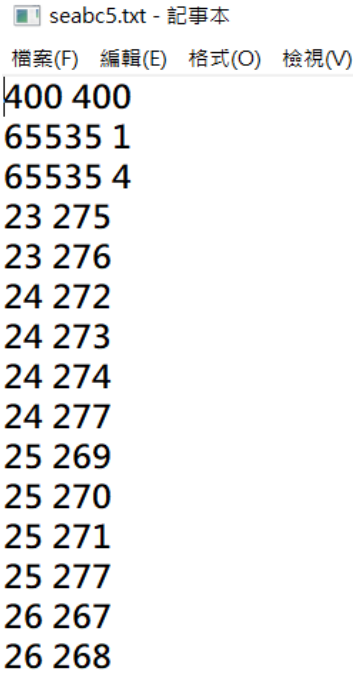

圖 47、理工大樓 C 棟五樓座標

我們寫了一個程式跑 2 維 A star 演算法，將模型中，我們設為障礙物的地方，輸出到一個 TXT 檔案，第一行:400 400 為矩陣大小，第二行：65535 1 為有幾棟樓，這裡的 1，就是有一棟樓，65535 為特定用途的數字，即做跨樓層的時候要用到的數字，第三行：65535 4，即五樓(由 0 起算)，之後的數字就是障礙物的座標點。

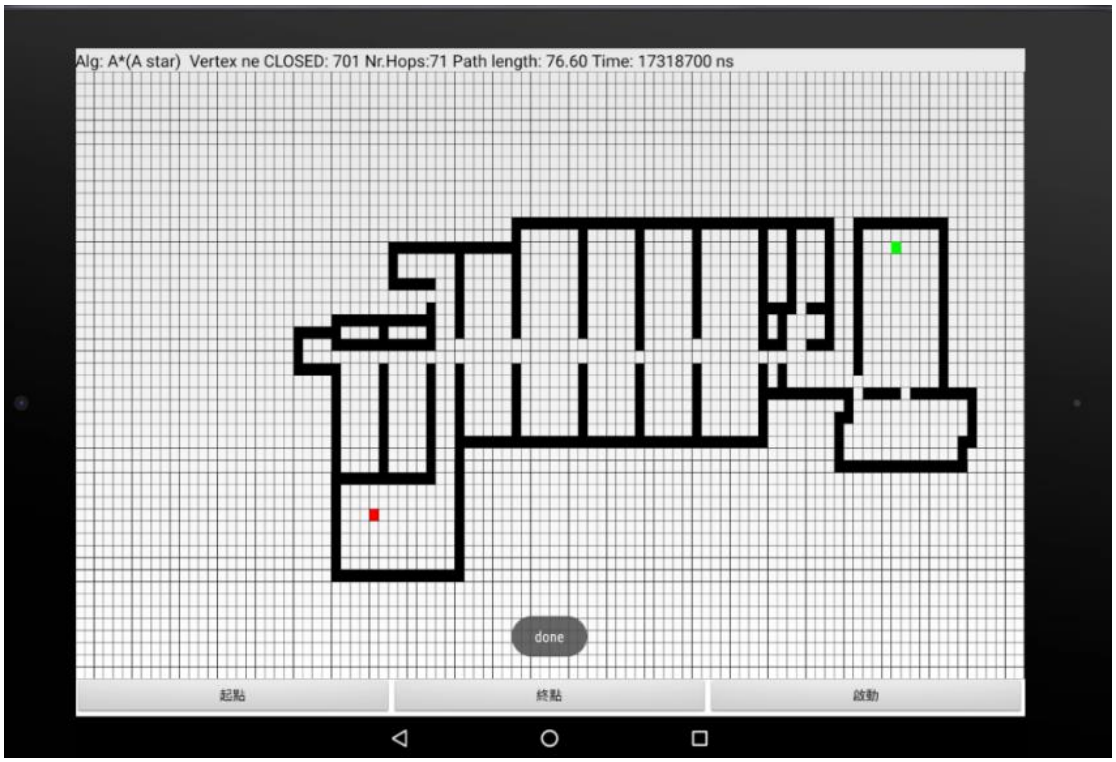

圖 48、理工大樓 C 棟五樓座標圖

如圖 48 所示，黑色的格子就是我們所設的障礙物，白色的格子就是可以走的地方。

圖 49、室外地圖

如上圖 49 所示，由 3D 建模轉平面座標點，在室外表現差強人意，邊邊角角的地方，顯示的不是十分準確，在 Unity 上添加 box collider 及修改物件的 state 的方式，可能不是程式 input 的最佳解。值得一提的是：添加 box collider 時，注意不要將模型旋轉，可能會導致整個畫面都是黑的，因為它能顯示的只能是平面。

5. AR 相關研究方法與步驟

基於室內定位如果用 GPS、Wi-Fi 等定位技術，因其誤差過大，我們打算使用 ARCore 裡的 Cloud Anchor 的擴充實境技術輔助室內定位，我們會以三個階段完成這部分。這部分包括資料收集與閱讀、Cloud Anchor 實作、使用者界面測試。

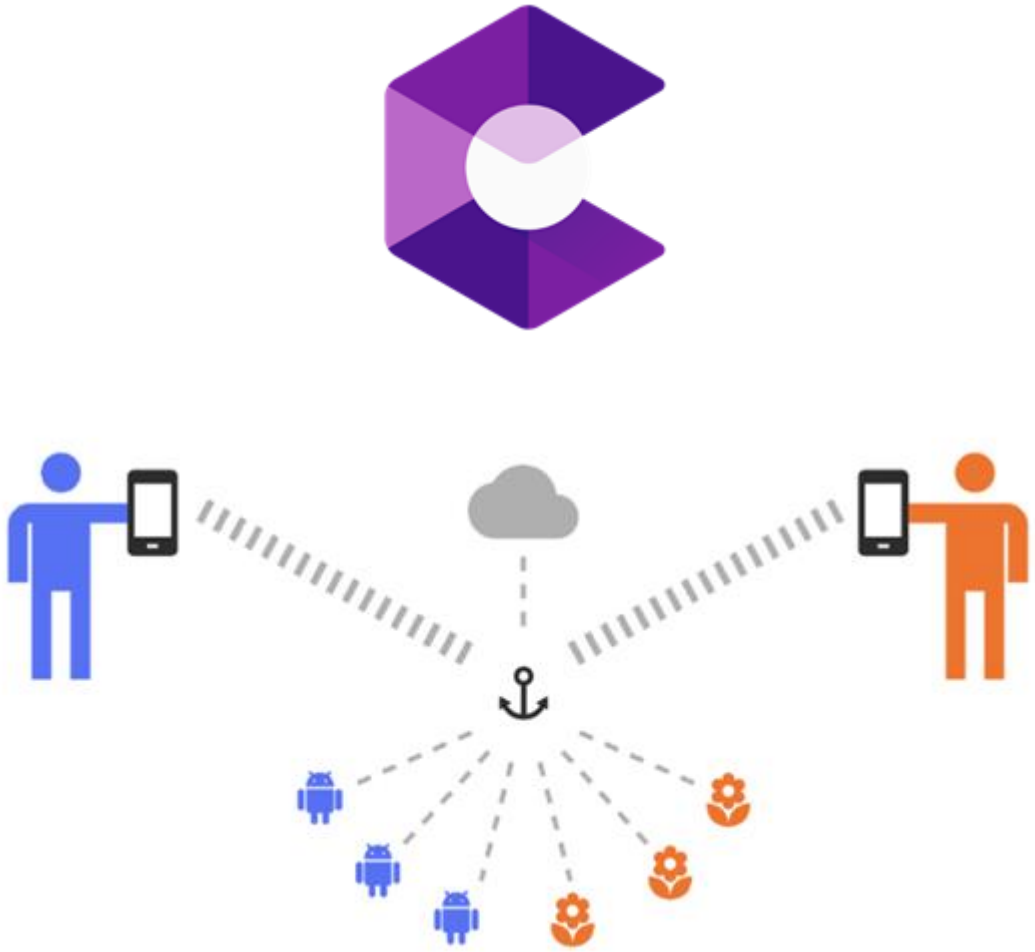

圖 50、Cloud Anchor 示意圖

第一階段以資料收集與閱讀為主，因為 ARCore 這套軟體開發套件被發行使用有一段時間了，但從網上找到的資料十分有限，我們會嘗試從 Google develop 開始。ARCore 中的 Cloud Anchor，也就是雲錨點。

第二階段進入實作階段，雲錨點可以讓同一環境中的使用者可以將雲錨點添加到他們在自己的設備上看到的AR場景中，也就是你的應用可以渲染連接到雲錨點的3D對象，從而讓使用者能夠查看對象並同步與對象交互。首先，我們會託管錨點，ARCore 會將相關的資料從使用者的環境發送到 Google 伺服器，上傳後，此資料會被處理成稀疏的點圖，之後，進行解析雲錨點，可讓一定空間內的多台設備使用之前的錨點來建立共同的參照框架。雲錨點解析請求會將可以看到的特征發送到伺服器，伺服器會嘗試將可視特征與雲錨點中的稀疏點圖匹配。[24]

第三階段進入使用者界面測試，透過寫好的雲錨點在手機上實測，看是否與我們預期的那樣，可以在每間教室上放一個錨點，以達到輔助定位的效果，讓使用者知道目前的所在位置，如若未達預期，我們會進一步修改。

I. ARCore：

首先我們透過 Google 本身所提供的教學內容來一步一步的做實作來達到練習的目的，分為：

1. 辨識&增強影像
2. 雲錨點
3. Sceneform

i. 識別&增強影像：

首先我們先照著辨識&增強影像的文檔來做實作以達到鎖定圖片的功能，此外增強影像的功能因為需要繪製3D圖形以及我們的專題本身沒有迫切需要，所以我們在實作時沒有去實現這項功能，我們希望能透過鎖定某些特定圖案來達到輔助使用者定位的功能，以下為我們實作出來的狀況。

圖 51、圖片偵測到前

圖 52、圖片偵測到後

我們可以從圖片中看到，我們的手機將我們預設的圖片所辨識，並且將其鎖定，鎖定後畫面左下角也會出現一行文字提醒我們圖片已經鎖定，我們最初認為通過這個功能可以讓手機畫面鎖定每間教室的牌子來達到定位功能，但後來經過實作後發現這個功能鎖定的圖像必須先通過一個叫做辨識度測試的測試，以我們教室牌子這種的圖像來說辨識度幾乎趨近於 0，所以無法靠著牌子實作這項功能，此外我們也思考過將學校各處貼上辨識度夠高的圖片來達到這項功能，但是因為認為這樣張貼需要學校的同意，以及可能因為其他因素被人所撕下，所以放棄了這項功能。

圖 53、教室牌子

ii. 雲錨點：

雲錨點這項功能我們在實做的時候需要去申請其 API，將 API 放入程式中之後才可以使用其相關功能，此外其 API 會根據使用的流量來收費，若低於一定的流量的話則免費，下圖為申請 API 的畫面。

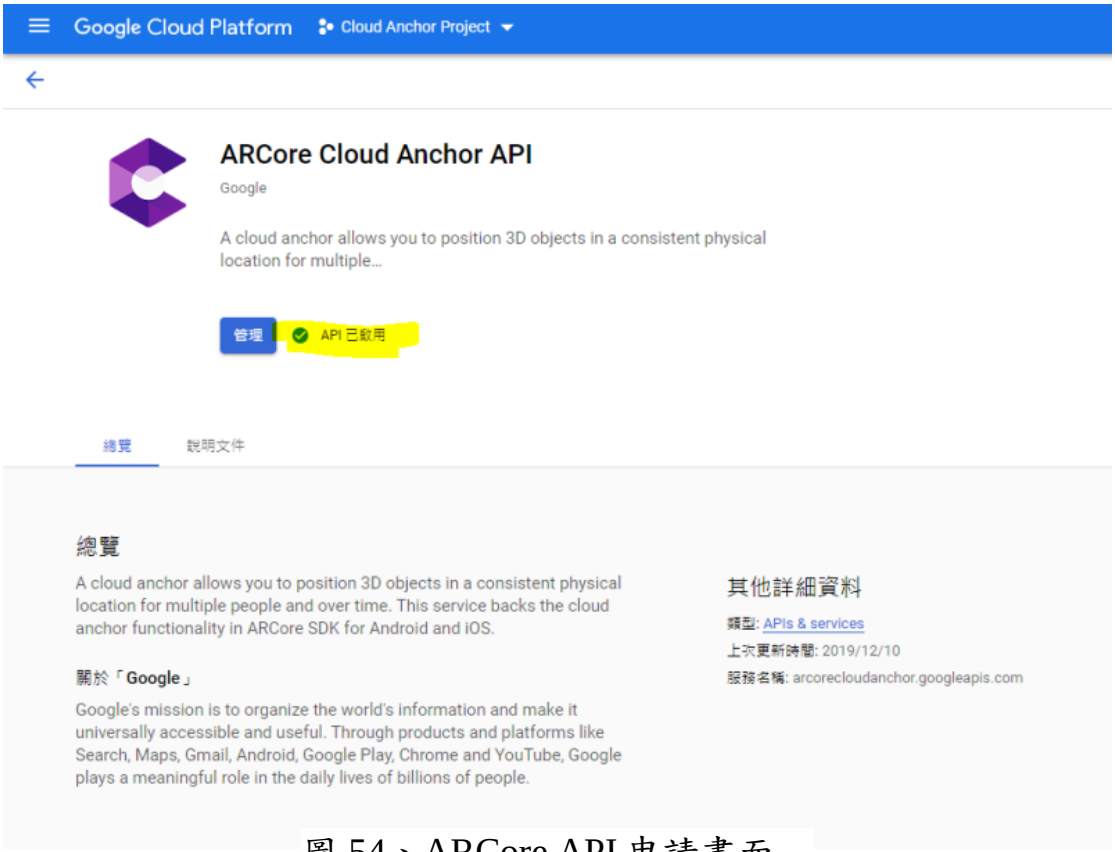

圖 54、ARCore API 申請畫面

申請完成之後即可將 API 放進自身所寫的 Android 程式中並且對其做使用，我們最初的想法是將此功能實作於我們的程式中，來達到另不同使用者可以做互動的情形，透過發布一項活動的訊息到其舉辦地點並且傳送到雲端，這樣其他同時使用這項程式的使用者若是在其地點使用我們的程式就可以看到此地將有什麼活動即將開始，能夠成為各社團、科系甚至校內活動的一種新的宣傳方式。

經過實作後我們則發現此功能並不具有即刻性，我們傳送上去的圖檔必須先經過 24 小時的處理後才能做使用，

並且其圖檔僅會幫忙保存七天，所以不適合做為長期的物件，跟一開始設想的差不多，僅適合短期的物件，但因為沒有立即性，必須等待 24 小時的問題使得其功能侷限性提高，所以我們在評估此項功能之後決定將此項功能作為之後完善了基本功能後再做的額外功能。

iii. Sceneform：

Sceneform 這項功能我們透過使用人家所提供的範例圖片以及步驟去做實作，通過將我們的 AR 物件加入光影，使其能夠更加地融入我們的周遭環境，以下圖片可以看到我們所實作的狀況。

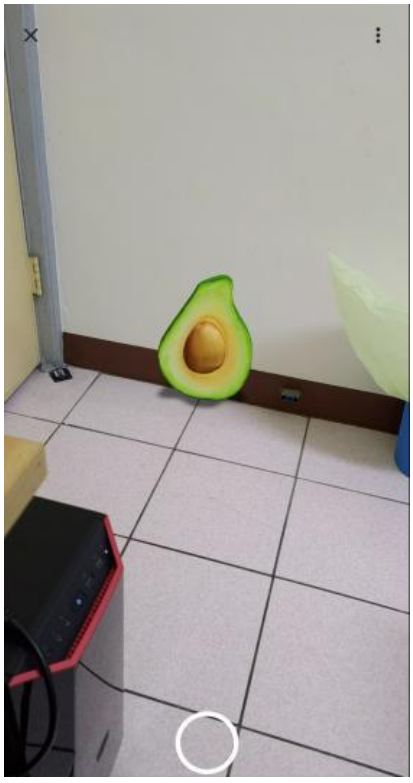

圖 55、Sceneform

II. AR Location-Based：

透過將 AR 的技術結合我們的定位技術能夠將我們所想要的 AR 物件呈現於我們所指定的位置上，可以藉此功能來介紹每個場館的功能，並且列出其更多資訊，使得使用者可以更佳的了解這些地方的功用，其中除了自己土法煉鋼來寫完全部的程式碼來實現以外我們也試過使用 Wiketude 的 API 來作為實作，其中 Wikitude 的 API 因為是商業用途，有些功能需要收費，所以我們最後放棄使用此 API 決定自行寫一個出來。

之後經過實行後我們的想法是通過我們所得到的 GPS 座標之後，將圖片定於我們所指定的經緯度，之後讓使用者在一定的範圍內後就可以從鏡頭中看到圖片，並且我們認為這算是一個不錯的想法，不過因為我們的實力以及時間精力有限，所以我們後來決定先暫時放棄這一塊部分，先將我們的其他定位指向等相關功能完善後，若有多餘時間我們會再回來做這個部分。

6. Unity 研究方法及步驟

I. 版本安裝：

到 Unity 官方網站，找到 Download 底下的 Unity，點擊之後就可以選擇下載版本。

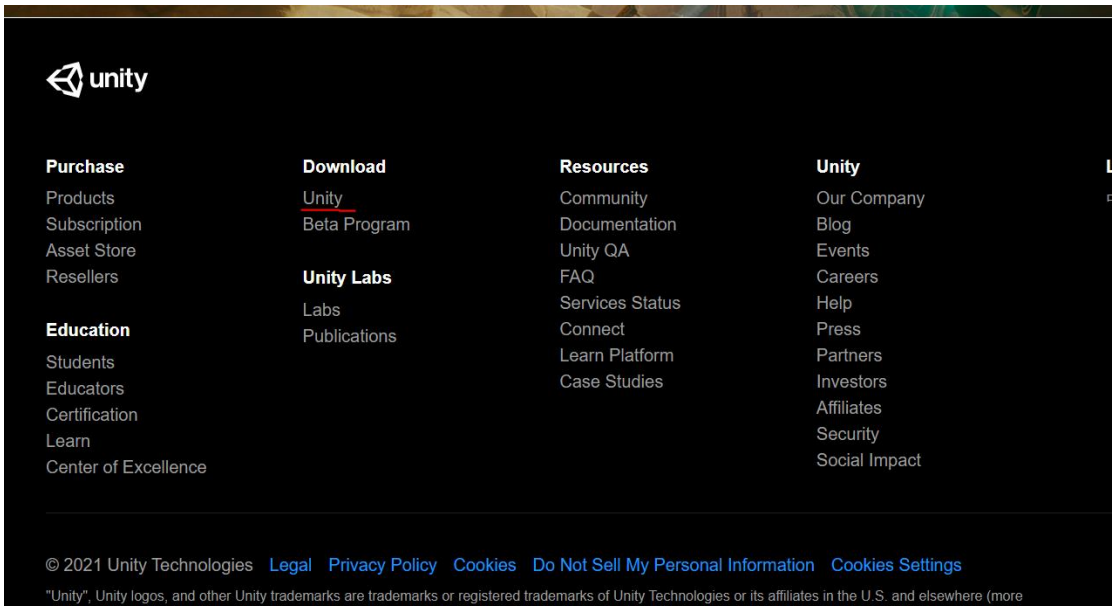

圖 56、Unity 下載

II. 版本介紹：

版本建議 Unity5.X 版，如果是新的專案，可以使用 2017 之後的 Unity 版本，2017 版本，渲染效率優化了不少，特性也變多了，也就是說你可以調控的東西變多了，Unity 對版本重新按年份來定義，這相對於原來 5.X 版本的重大改進。我們研究版本為 5.6.7f1。版本不是越新越好，Unity 更新頻率之快常常你沒適應完新功能，就又出了一個 beta 版，Unity5.X 版相對較為穩定，想想看，如果你的電腦老是更新，更新過程中還會出錯，導致整個系統需要重灌，那是非常糟糕的一件事。評估三大點：穩定性，效能，功能。穩定性，有再好的優化、功能，專案一執行就會當掉，那都沒有意義，選擇版本，首要考慮穩定性；再來是效能，在有穩定基礎功能上，看效能在哪得到了很大的提升，首要考慮對自己的專案有沒有幫助；功能，了解選擇的版本有哪些特性是對專案有用的，並且比較與舊版本的差別為何，完成這些，就可以避免一些不可預期之錯誤，節約大量砍掉重練的時間。

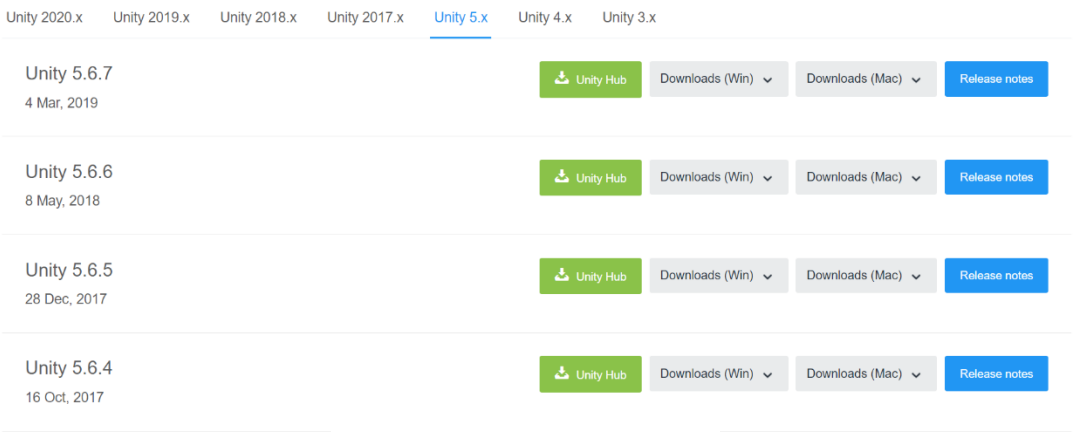

圖 57、Unity 選擇版本

III. 路徑演算法在 Unity 上的實作：

1. 在 Scene 建立四個 cube，分類為障礙物,再建立兩個 cube，分別命名為起點與終點。

2. 將分類為障礙物的四個 cube 的狀態在 inspector 上改為 static

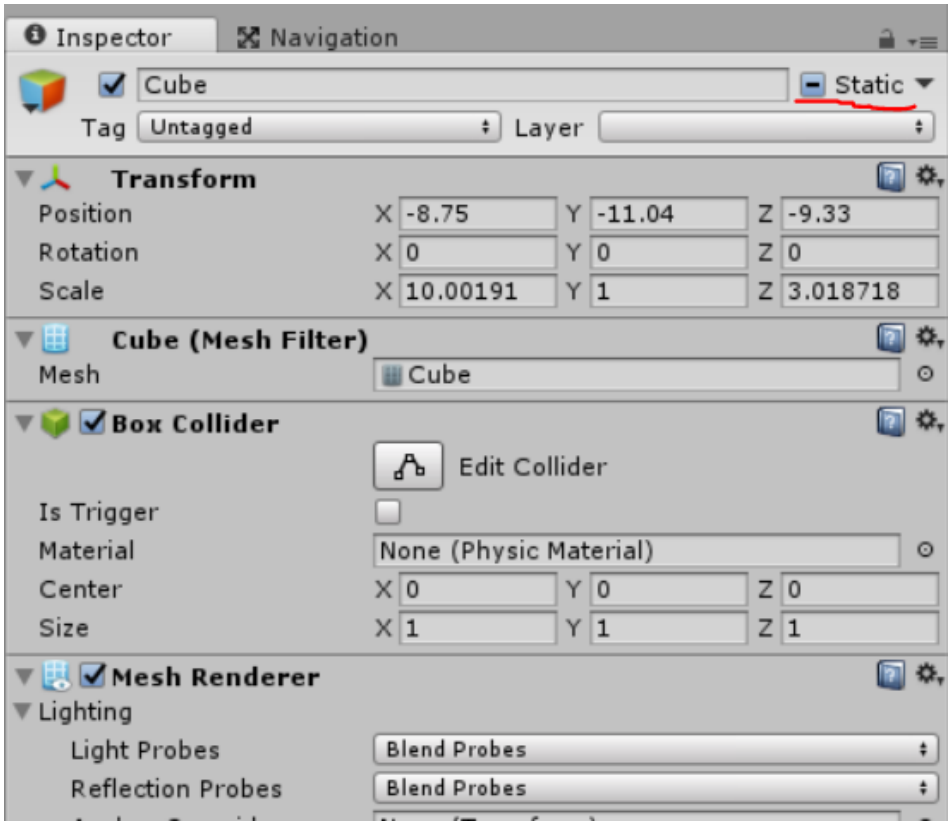

圖 58、修改 static

3. 在 Window 選單中找到 Navigation，將障礙物的四個 cube 的 Navigation static 打勾。

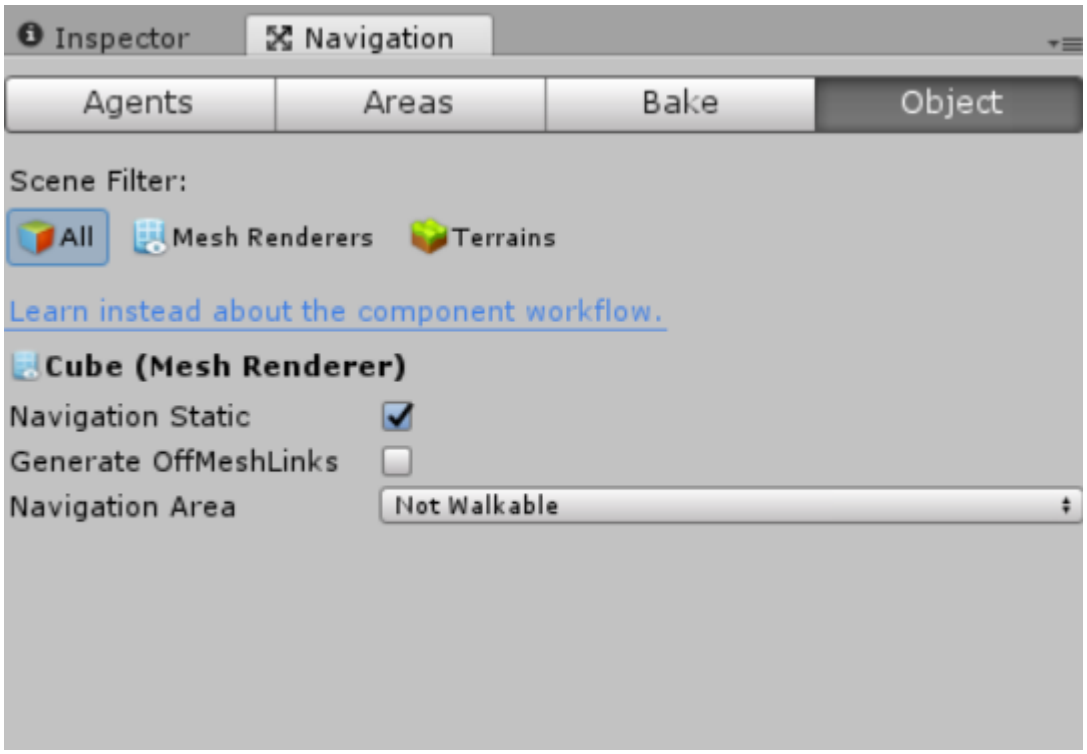

圖 59、修改 static

4. 一樣在 Navigation，選擇 Bake，在底下還有 Bake，點擊 bake，就可以生成導引網格。

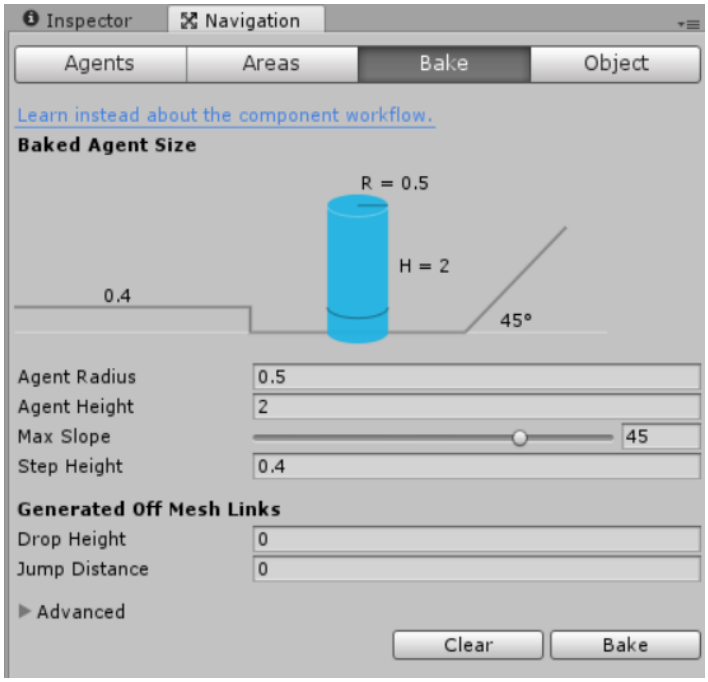

圖 60、Bake 網格

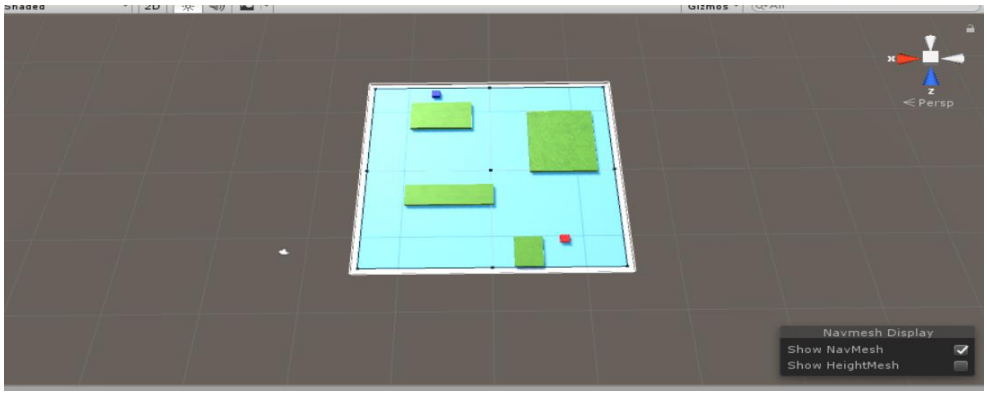

圖 61、成功生成網格

5. 在終點的 cube 掛載 A star 路徑演算法的腳本，點擊播放鍵，即可看到結果。

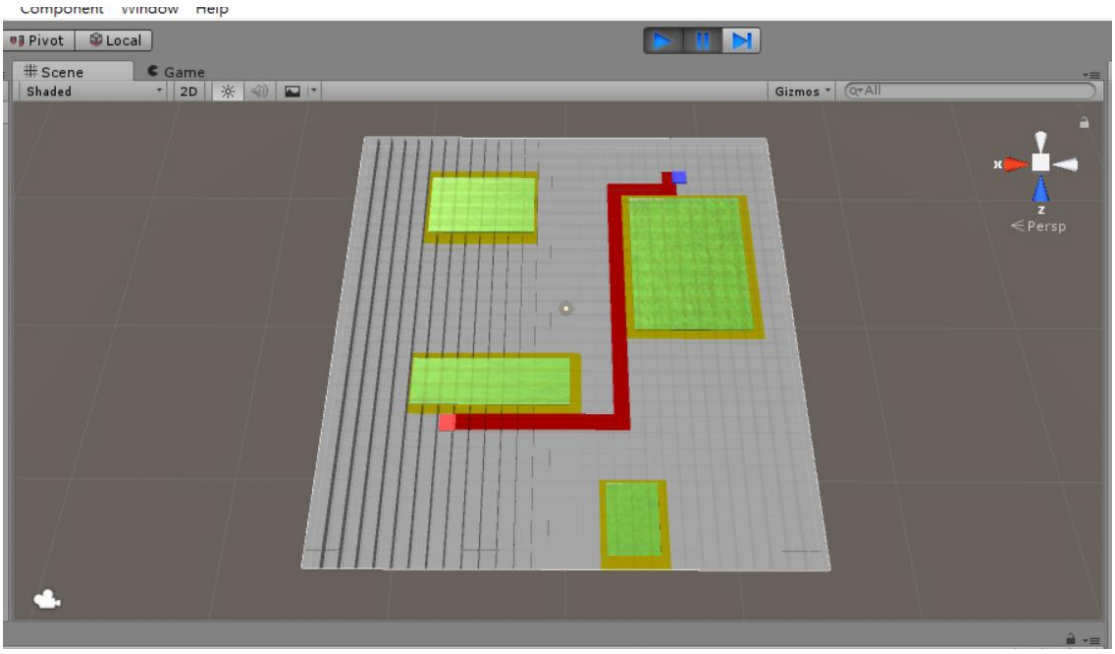

圖 62、路徑展示

綠色的是障礙物，淺紅色的 cube 為終點，藍色的 cube 為起點，深紅色的線為導引路徑。

6. 學習 Unity 可能會遇到的問題：Unity 報錯時，可以查看 console 裡面的 Error pause，它會告訴你的錯誤訊息，想要查看的更詳細，可以點選 open Editor log，通常會報錯，問題都會出現在腳本上，請先編譯通過再點播放，通過播放可以知道是否到達想要的效果。

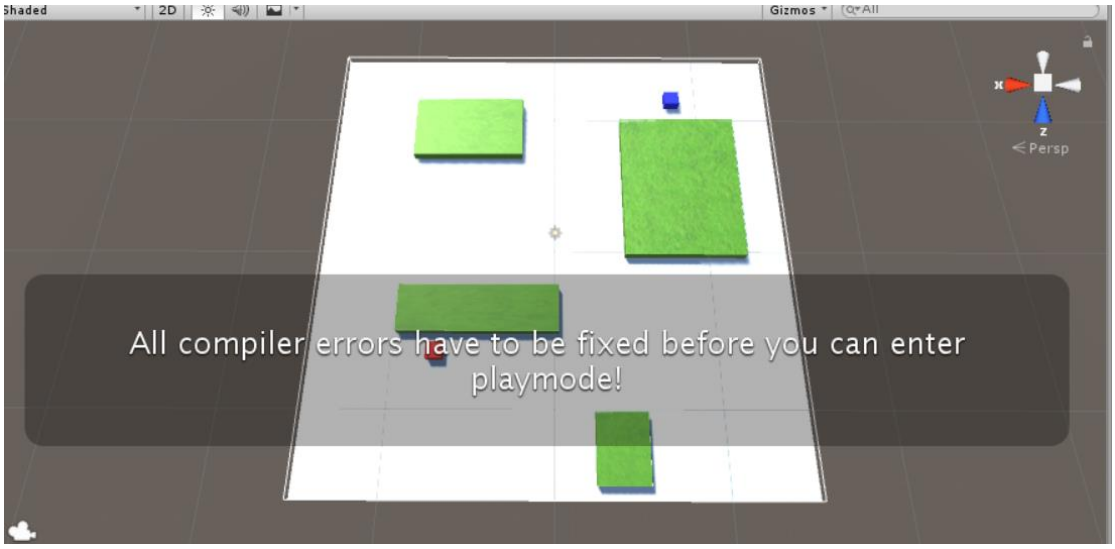

圖 63、錯誤提示

總結：

值得一提的是：目前看到的網格是 30*30，計算速度很快，但要開到 400 以上，甚至 1000，計算速度之慢，會達五分鐘以上，影響計算速度的除了網格大小，還有硬體設備：CPU、GPU，可以想象如果是在移動設備上使用，計算速度會更慢，那使用者根本就不會想要使用，加上多人向系統提出路徑規劃的需求，系統會無法負荷， 這就導致了我們之後停止了對 Unity 的研究，轉而研究 Android Studio 原生平台。

四、 研究成果

I. 使用者流程:

由理工大樓 C 棟 4 樓到共教大樓(鏡心書院)A 棟 1 樓師培中心主任辦公室最為示範。

1.點開下拉式選單

i. 選擇大範圍：點選共教大樓(鏡心書院)A 棟

ii. 選擇樓層：根據資料庫帶入該大樓的樓層數選擇樓層，點選 1 樓

iii. 選擇地點：最後資料庫帶入大樓該樓層所擁有的所有教室，選擇師培中心主任辦公室

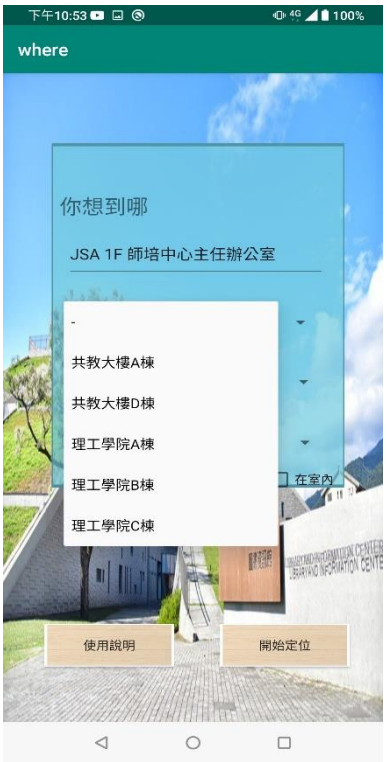

圖 64、選擇大樓

。

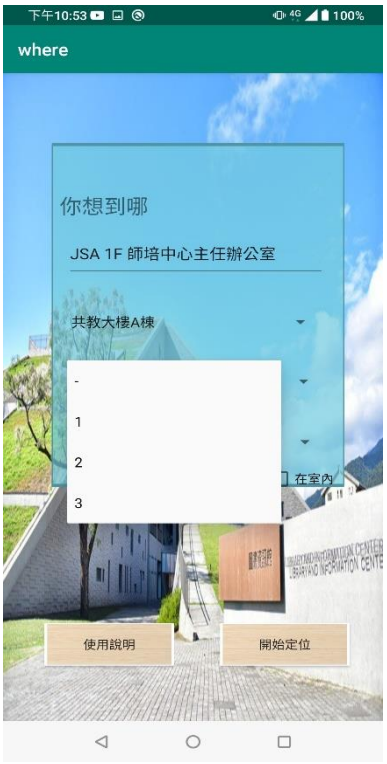

圖 65、選擇樓層

。

圖 66、選擇地點

2.顯示畫面如下圖。如果是在室內，記得點選在室內，接著就可以點擊開始定位，APP 就會開始規劃路線。

圖 67、顯示畫面

3.我們使用紅色的線條代表導引路徑，而藍色的原點代表使用者目前所在的位置，跟隨路線一直走，即可到達目的地。當走到樓梯時，APP 會提示訊息：請下至一樓。

圖 68、理工大樓 C 棟 4 樓

4.跟隨路線至理工大樓 C 棟 1 樓。可以按樓下地圖，即可切換到 1 樓地圖。

圖 69、理工大樓 C 棟 1 樓

5.即將到達室外時，APP 會提示請按啟用室內外定位，也就是關閉室內定位。這時，使用者可以按下個區域按鈕，切換室外地圖，跟隨路線，到達目的地。

圖 70、室外地圖

6.藍色原點代表使用者目前的位置，跟隨紅色導引路線，到達目的地。地圖可以隨手勢可以放大縮小，旋轉，平移，方便使用者查看地圖。注意事項：下圖為模擬器示意圖，實際效果跟下圖一樣，除了點擊啟用室內定位後，實機上 APP 提示訊息會消失。切換室外地圖，記得點選關閉室內定位，否則會出現閃退 BUG，如果使用者偏移原本路線過多，請按下重新定位按鈕，即可重新定位，APP 也會重新規劃新的路線，幫助使用者到達目的地。

圖 71、放大後的室外地圖

7.在即將到達目的地時，記得按啟用室內定位按鈕，以幫助 APP 確認使用者位置。地圖經過反黑處理(不需要的座標點都設為 false),這樣可以達到提高計算速度的結果。室外地圖經過美化，不是只有黑白色之後，像是有河流的地方也能幫助使用者知道為何這條路不能走。導引畫面下方的電梯按鈕和首頁的使用說明按鈕暫時沒法使用，之後會補上此按鈕的功能。

圖 72、共教大樓(鏡心書院)AB 棟 1 樓

五、 結論

心得：

通過這次做專題，讓我們得到了很多光是上課無法獲得的實作經驗，有些不自己實作看看的事情，光是看網路上的資料，覺得很簡單，實際做起來才發現問題一大堆，就像是這次 Unity 的 3D 建模模型轉 2D 平面座標點的問題，程式寫好之後，run 完之後才發現，實際得出的結果有偏差，有些地方的座標點會出現斜著出現的情況，嘗試過幾遍才發現，是比例不對的原因，因為我們 3D 建模除了理工大樓要縮小 0.4 以外，其他的都是 1 比 1，我們會對建模的比例進行縮小，是因為如果網格開太大(超過 800*800)，在 Unity 上，計算速度會變慢，後來轉為在 Android Studio 原生平台上做，修改了資料結構後，計算速度得到了很大的提升，讓我對於程式 input 的方法產生了很大的疑問，雖然在室內地圖表現良好，但在室外地圖表現差強人意，後來在查找資料的過程中，發現 Android Studio 有內建的功能可以對顏色進行判別，也就是說，或許不需要建模，就可以達到一樣的結果，我們可以將障礙物弄成黑色，可走的地方弄成白色，這樣對於地圖的顯示就會有很大的幫助，讓初次使用 APP 的使用者不需要花太多時間在猜想地圖所顯示的地方。在選擇程式 input 的方法上，或許可以從 Unity 方面上突破，找到更好的方法。

未來展望：

在室內定位我們嘗試了數種方法，但因為地形複雜，且理工大樓電腦設備居多，無法避免多徑路徑的影響造成的誤差造成任何跟據距離或傳輸時間差的定位方式都無法使用，所以最後我們選擇使用訊號紋比對法來進行比對，但與網路上的大多數使用機器學習的方式來進行比對的方法，我們透過斜率比較的方法雖然能簡單地找出我們現在的位置，但精準度跟閱讀的文獻上比較依舊有差距，未來的話可能從以下方面進行改善。

機器學習：

根據看到的文獻訊號紋比對法大部分皆以降躁及透過機器學習的方式來進行訊號紋的比對，這次因為研究不足則以平均斜率的方式來比較，未來希望透過 KNN 演算法來進行位置的預測。

資料庫改善：

現在資料採集為行走時每 4 秒測量一次，但僅完成理工學院 c 棟資料已經有數萬筆，若將全校每棟建築測量完畢，可能會因資料筆數過多造成定位計算時間過長，未來將改成先以 GPS 定位，確認在哪棟學院範圍內再進行資料庫查詢，將判斷哪棟建築的工作交給 GPS，以減少不必要的訊號比對的部分，或評估將計算部分交給伺服器中間的以及中間傳輸延遲也可能將資料庫從本地端移出，改成線上的方式，這樣進行資料更新也會比較方便。

地圖改善：

現在的地圖各樓梯所占的面積只有 2 格，當初在設計時沒考慮到樓梯的測量，所以導致樓梯的測量面積嚴重不足，所以可能在樓梯中間訊號準度特別糟糕，新版地圖會將樓梯的面積加入，並重新測量。

資料採集：

目前資料採集部分缺乏便利性，需要先自行找出起點、終點座標，再輸入自 app 再開始採集，這步驟費時費力，未來希望改成以點擊觸碰地圖選定終點起點座標，以及增加刪減資料庫的介面，以增加後續維護性。

六、 參考文獻

[1]如何計算距離
https://codertw.com/android-%E9%96%8B%E7%99%BC/343009/
[2]用感測器計算位移公式
https://www.2cm.com.tw/2cm/zh-tw/tech/72152B828EEA4406B5E10ECAF8462D5D
[3]使用最小偏差法及蜂群法之無線室內定位精度改進探討與比較-----梁昭隆
https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi?o=dnclcdr&s=id=%22104THU00394019%22.&searchmode=basic
[4]到達時間差
https://zh.wikipedia.org/wiki/%E5%88%B0%E8%BE%BE%E6%97%B6%E9%97%B4%E5%B7%AE
[5]室內無線定位方法之開發與驗證
https://ir.nctu.edu.tw/bitstream/11536/46802/1/359301.pdf
[6]應用分群方法於指紋比對分析之室內定位演算法---游晉彰
https://hdl.handle.net/11296/f4cxq9
[7]傳感器輔助的 Wi-Fi 指紋室內定位方法
https://read01.com/3DJoLg.html
[8]A star 演算法
https://en.wikipedia.org/wiki/A*_search_algorithm
[9] SQLite 維基百科，自由的百科全書
https://zh.wikipedia.org/wiki/SQLite
[10]Owens, Michael. Chapter 4: SQL. (編) Gilmore, Jason; Thomas, Keir. The Definitive Guide to SQLite. D. Richard Hipp (foreword), Preston Hagar (technical reviewer). Apress. 2006: 133 [30 December 2014]. ISBN 978-1-59059-673-9.
[101Most Widely Deployed SQL Database Estimates. Sqlite.org. [May 11, 2011].
[11] Patrick Lester 和 Myopic Rhino 發佈的 A star 演算法初學者教學
https://www.gamedev.net/reference/articles/article2003.asp

[12]取得三軸加速度計量值
https://www.youtube.com/watch?v=lYoubWBIkJ8
[13]Android studio 計時器功能
https://www.youtube.com/watch?v=U--E3tpfLhE
[14]Android studio 靠加速度感測器實現計步器
https://www.youtube.com/watch?v=V6qanRM67w8
[15]Android studio 步數感測器使用
https://www.youtube.com/watch?v=CNGMWnmldaU
[16]每人平均每步距離
https://zhidao.baidu.com/question/525229984242428365.html
[17]室內實時定位技術介紹以及定位原理介紹 - 每日頭條
https://kknews.cc/zh-tw/tech/g9zbe3e.html
[18]你以為網路慢只是 WiFi 信號不好，原來背後這麼多學問！
https://www.bnext.com.tw/article/40699/bn-2016-08-24-142137-178
[19]無線射頻辨識 - 維基百科，自由的百科全書
https://zh.wikipedia.org/wiki/%E5%B0%84%E9%A2%91%E8%AF%86%E5%88%AB
[20]Getting Started With Blender 2.8
https://youtube.com/playlist?list=PL32001WLw6WHTeo7Khqg0k1E5l1iPJc7e [21]How to Create 3D Terrain with Google Maps and Blender!
https://www.youtube.com/watch?v=Mj7Z1P2hUWk&feature=emb_title
[22]Blender: How to Download and install BLENDER GIS in BLENDER
https://youtu.be/46GCo5od61I
[23]A* pathfinding for beginners
https://www.gamedev.net/reference/articles/article2003.asp
[24]Google ARcore develop
https://developers.google.com/ar/develop
[25]RFID 種類
https://zh.wikipedia.org/wiki/%E5%B0%84%E9%A2%91%E8%AF%86%E5%88%AB

圖片出處：

1. https://youtu.be/46GCo5od61I (圖 9、使用 Google Map 選取區域&圖 10、使用 GIS 功能產生建築物模型)

2.https://zh.wikipedia.org/wiki/%E5%85%AD%E8%87%AA%E7%94%B1%E5%BA%A6 (圖 11、六自由度)

3. https://www.itread01.com/content/1544396044.html(圖 12、環境理解)

4.https://m.oursky.com/arkit-%E9%96%8B%E7%99%BC%E5%AF%A6%E6%88%B0-%E8%98%8B%E6%9E%9C%E6%8F%90%E4%BE%9B-%E6%A0%B8%E5%BD%88%E7%B4%9A-%E5%B7%A5%E5%85%B7-%E8%AE%93-ar-%E9%96%8B%E7%99%BC%E7%B0%A1%E5%96%AE%E5%A5%BD%E4%B8%8A%E6%89%8B-b091f4627efe(圖 13、光線評估)

5. https://developers.google.com/ar/develop/java/quickstart (圖 14、Google 提供範例)

6. https://developers.google.com/ar/develop/java/augmented-images (圖 15、識別&增強圖像)

7.https://developers.google.com/ar/develop/java/cloud-anchors/overview-android (圖 16、雲錨點)

8.https://kiosk-dot-codelabs-site.appspot.com/codelabs/sceneform-intro/index.html?index=..%2F..devfest18#0 (圖 17、Sceneform)

9. https://www.bsetec.com/blog/location-based-augmented-reality/ (圖 18、AR Location-Based)

10. https://sites.google.com/site/rfiddianzibiaoqian/shou-ye-2 (圖 19、RFID)