中華民國第 65 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學科(一)

032814

以 Scratch 為模型規劃最短逃生路線

學校名稱： 臺東縣立寶桑國民中學

作者：	指導老師：
國一 黃義捷	邱偉誌
國一 張維德	楊惠如
國一 高隆興	

關鍵詞： 逃生路線、曼哈頓距離、Dijkstra 演算法

摘要

本研究針對超市等室內空間中之逃生路線進行規劃，並利用 Scratch 建立模擬模型，結合曼哈頓距離與 Dijkstra 演算法進行最短路徑分析。研究旨在開發一套具效率的逃生路線規劃工具，以供空間設計與安全規劃參考。

壹、前言

一、研究動機

最近，台灣發生了一起嚴重的火災事故，地點位於某商業大樓。事故的起因是金屬構件焊接作業時產生的高溫火花，透過樓層間的開放空間掉落至地下室，引燃存放的大量易燃塗料與溶劑。由於火勢蔓延迅速，當員工發現時，已無法及時逃生，導致多人受傷，甚至有人不幸喪生。

此事件凸顯建築空間中逃生路徑設計之重要性，當火災發生時，能否在最短時間內抵達安全區域，常為生死存亡之關鍵。建築物內的逃生動線規劃，應考量火勢擴散的可能性，確保人員能夠在最短時間內撤離至安全區域。因此，本研究將專注於最佳逃生路線設計，即分析火災發生時，不同樓層的人員應如何選擇最短且最安全的逃生路徑，減少受困風險。

本研究旨在探索更有效的逃生路線設計方式，提升建築物的安全性，減少災害造成的傷亡，讓人們在危急時刻能有更高的生存機會。

二、研究目的

本研究在探討運用Scratch解決平面最短路徑問題，並比較不同距離計算方式、移動方式及路徑規劃方式對於最短路徑取得的影響，依此訂定下列研究目的與研究問題：

(一)距離計算方式的比較：比較曼哈頓距離及歐幾里得距離——配合電腦的平面座標及網格系統，比較曼哈頓距離及歐幾里得距離對於最短路徑規劃時間與距離差異。

(二)移動方式的比較：比較四向度與八向度移動方式對於路徑規劃的距離影響及時間差異。

(三)路徑規劃方式的比較：比較貪婪法及Dijkstra演算法對於最短路徑取得的影響。

(四)實踐於生活中的探討：分析Scratch取得最短路徑對於解決實際生活的可行性與局限性。

三、文獻回顧

(一)曼哈頓距離(Manhattan Distance): 曼哈頓距離基於水平與垂直的距離，而非對角線的距離。這種計算方式類似於在城市中行走，必須沿著街道（橫向或縱向）行駛，無法斜著走。因此，曼哈頓距離是兩個點在座標平面上的 x 與 y 座標差值的絕對值之和。具體而言，假設兩點的座標分別為 (x_1, y_1) 與 (x_2, y_2)，則曼哈頓距離可用公式表示為：

$D=|x1-x2|+|y1-y2|$

(二)歐幾里得距離（Euclidean Distance）：歐幾里得距離是用來衡量平面上兩點之間直線距離的一種基本方法。其名稱源自古希臘數學家歐幾里得，為現代幾何學中最常使用的距離計算方式。

在二維平面中，若有兩點 $A(x_1, y_1)$ 與 $B(x_2, y_2)$，其歐幾里得距離的計算公式如下：

$$d = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

此公式表示的是兩點之間的直線距離，類似於在地圖上以尺量測兩個地點之間的最短距離。

(三)貪婪法：貪婪法的核心思想為在每一步的選擇上，皆挑選「當下看來最有利」的選項，也就是每次選擇目前路徑中與起點距離最短的節點進行擴展，期望透過這種逐步選擇最佳的方式，達成整體最佳解。以最短路徑問題來說，貪婪法在起點選定後，演算法會從起點出發，尋找與起點的鄰近節點，並將該節點納入考慮範圍，再選擇最有利的下一步(距離終點最短的節點)，依序拓展至尚未造訪且距離起點最短的節點，直至抵達終點。

(四)Dijkstra演算法(Dijkstra's algorithm): Dijkstra 演算法是一種用來計算單一起點到其他所有節點最短路徑的方法，適用於邊的權重為非負數的加權圖。它的基本原理是利用「貪婪策略」，每次都選擇目前距離起點最近的節點進行處理，並嘗試更新鄰近節點的最短距離。演算法一開始，會將所有節點的距離設為無限大，表示還不知道怎麼到達，只有起點的距離設為 0。然後建立一個「未處理節點集合」，用來追蹤哪些節點還沒有被確認最短距離。每一步，從這個集合中選出距離起點最短的節點，接著檢查它的所有鄰居，看是否可以透過它走更短的路。如果可以，就更新那個鄰居的最短距離。處理完後，把該節點標記為「已處理」，表示它的最短距離已確定。這個過程會重複，直到所有節點都被處理，或所有可到達的節點的距離都確定。因為邊的權重不能是負數，所以每次選出來的最短距離都一定正確，不會被之後的路徑更新影響。

貳、研究設備與器材

一、硬體：電腦（安裝 Scratch 3.0）

二、軟體：Scratch 3.0（搭配自訂擴展畫筆功能）

三、其他工具：模擬用網格地圖模板（以手繪或列印方式製作，輔助場景建構）

參、研究過程與方法

一、研究架構圖

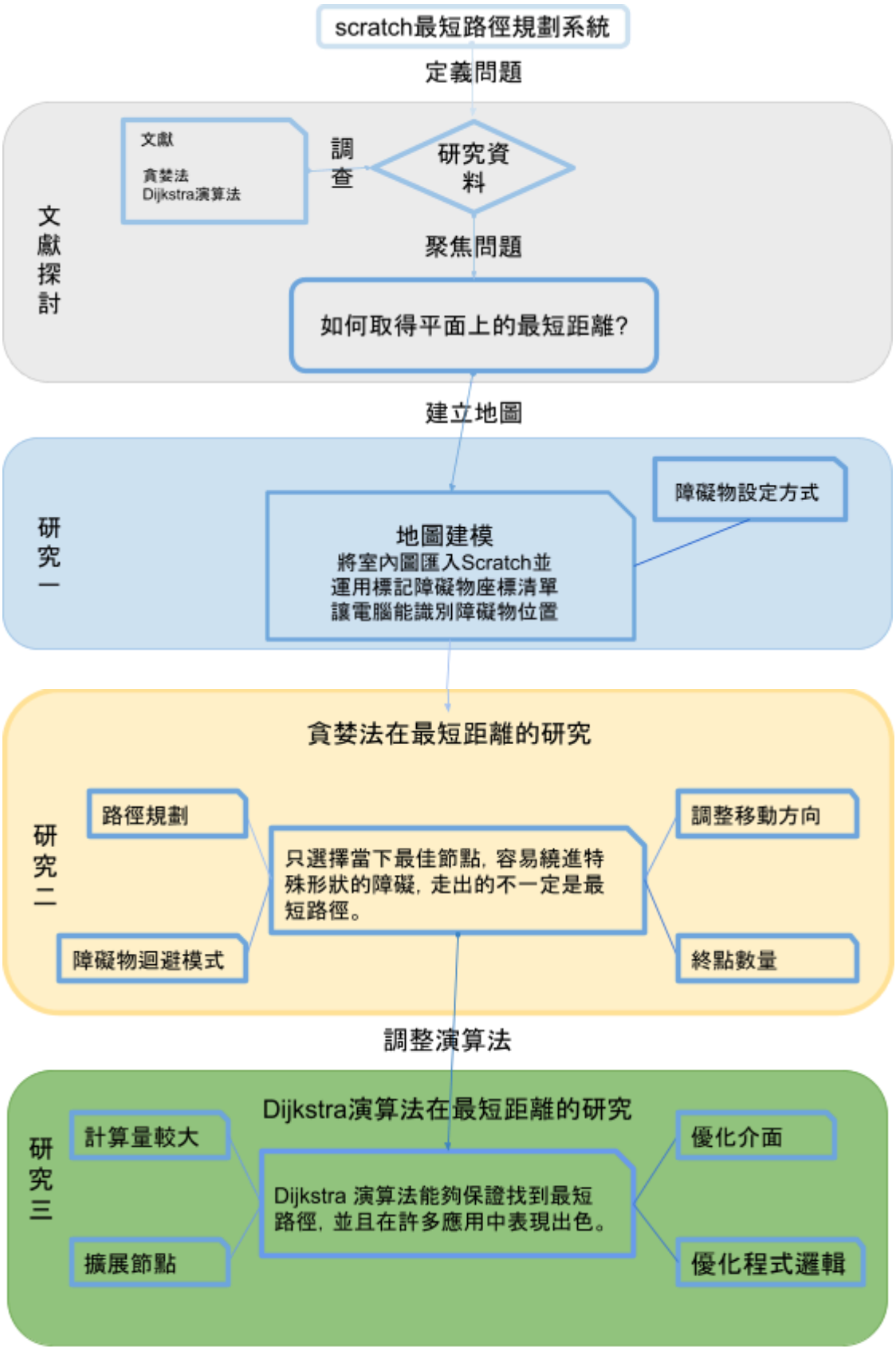


圖 3-1 研究架構圖
(圖片由第一作者與第二作者共同繪製)

二、建立地圖

為了解決賣場可能遇到的逃生路徑問題，本研究參考網路上公開之某知名賣場與便利商店之平面圖，並根據 Scratch 畫面限制進行尺寸調整與室內空間建模，其製作流程如下。

(一)地圖建模：將室內環境劃分為網格，考量到Scratch運作效率與畫面限制，長為38格、寬為28格，每個格子代表一個節點。

(二)標記節點：運用Scratch製作的小工具，於圖形介面中手動標記起點、終點(複數)、障礙物，其中起點為地圖中的任意點，終點為地圖中的出口位置，障礙物為所有無法通行的地方。

(三)再將已經標記完所有障礙物節點的程式轉入至清單內，後續可更方便的生成已經設置完畢地圖的障礙物。

三、貪婪法在最短距離的研究

(一)移動方向:

本研究初期程式設計中，移動方向僅限於四個基本方向（上、下、左、右）。為提升路徑規劃的靈活性，後續新增對角線移動功能，使移動方向擴展為八向(上、下、左、右、右上、右下、左上、左下)。同時，為確保對角線移動的合理性與通行性，亦調整相關邏輯：在執行斜對角移動（例如右上）時，系統會同步檢查與該方向相鄰的兩個正交方向（此例中為右方與上方），若兩者皆無法通行時，禁止進行該對角線方向的移動。

(二)路徑規劃:

本研究在實作貪婪法演算法時，所使用的距離評估方式為曼哈頓距離。此種距離計算方式適用於僅允許上下與左右移動的情境，藉由計算目前節點與目標節點在水平方向與垂直方向上的距離總和，作為節點之間的估計距離。由於貪婪法在選擇擴展節點時，僅考量與目標的預估距離，不考慮實際累積路徑的成本，因此選用何種距離計算方式，將直接影響搜尋的效率與結果的合理性。曼哈頓距離的優點是計算快速且直觀，特別適合用於初期僅支援四向移動的版本。隨著本研究後續新增對角線移動，使移動方式擴展為八向，亦同步評估曼哈頓距離是否仍為合適的啟發策略，或需改以更貼近實際移動成本

的距離評估方式，如對角距離或其他改良方式，以提升演算法在不同移動模式下的準確性與效率。

(三)圖形化操作介面:

本研究初期版本中，程式中障礙物、起點與終點皆為預先設定，使用者無法於執行階段進行動態調整。為提升系統的彈性與操作便利性，後續透過程式邏輯與介面的改良，將障礙物、起點及終點的位置設定，全面轉換為圖形化操作介面。使用者可透過滑鼠點選進行障礙物的新增與刪除，並設定起點及一個或多個終點，以提升模擬靈活性。

此一改進大幅提升整體互動性與模擬靈活性，使系統能更直觀地反映使用者需求，也更便於進行不同情境下的實驗與測試。圖形化的擺放方式有助於模擬真實應用中如動態環境變化等複雜場景，並為後續進一步探討多終點搜尋、避障策略與使用者介面優化等研究方向奠定良好基礎。

(四)Scratch程序於貪婪法的執行流程:

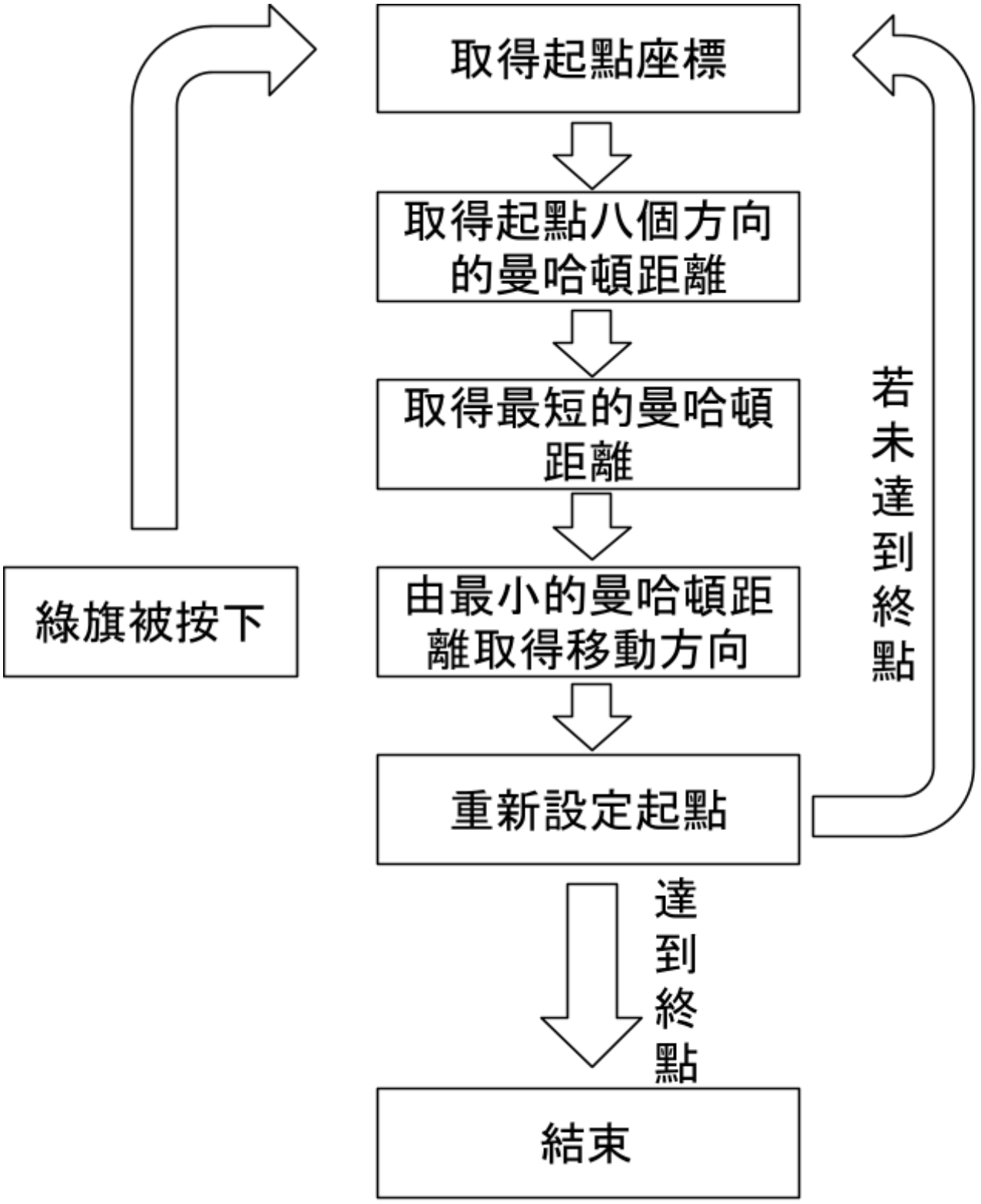


圖 3-2 Scratch程序於貪婪法的執行流程
(圖片由第二作者繪製)

四、Dijkstra演算法在最短距離的研究

因貪婪法在部分場景的路徑規劃較為費時，經與指導教授討論後，教授建議改用 Dijkstra 演算法在最短路徑規劃上較為合適。在實際測試後發現，Dijkstra 演算法在整體路徑規劃上，與貪婪法相比在相同起點與終點的最短路徑規劃所需移動距離更短，雖然付出了較多的規劃時間與電腦效能，但仍在可以接受的範圍內。因此，本研究決定以 Dijkstra 演算法替換貪婪法，作為最短路徑規劃的方式。

(一)演算法調整:

Dijkstra演算法在進行路徑搜尋時，會同時考量起點到目前節點的實際路徑長度，並依照目前累積的移動成本逐步擴展節點，直到抵達目標位置。由於每一步都以目前總花費最小的節點為優先處理，最終可保證找到起點至終點的最短路徑。

與僅參照最短距離的貪婪法相比，Dijkstra 演算法在搜尋過程中雖然可能需處理更多節點，計算量相對較大，但能夠提供更高的準確性與穩定性，特別是在障礙物密集或地圖結構複雜的情境下更具優勢。

(二)圖形化介面:

本研究於後期擴大地圖的解析度與規格，搭配Dijkstra演算法使系統能處理更大範圍的節點分布。地圖中的節點皆依固定間距建立網格，並在畫面上完整呈現每一個節點的位置，以提升視覺化效果與使用者對路徑搜尋過程的理解。

(三)優化程式邏輯:

為配合地圖規模的擴張與節點數量的增加，本研究亦全面重構程式邏輯，針對原有架構中不必要的變數與重複性程式段落進行優化，移除冗餘程序，簡化資料結構，並提升演算法的執行效率。此外，也調整節點的儲存與處理方式，使系統在節點數倍增的情況下，仍可維持良好的效能與穩定性。

透過本階段的優化與擴充，不僅提升了系統處理複雜地圖的能力，也使整體程式結構更具延展性與可維護性，有助於未來整合更多進階功能，如動態障礙處理、多重路徑規劃與演算法效能比較等應用場景。

(四)Scratch程序於Dijkstra演算法的執行流程:

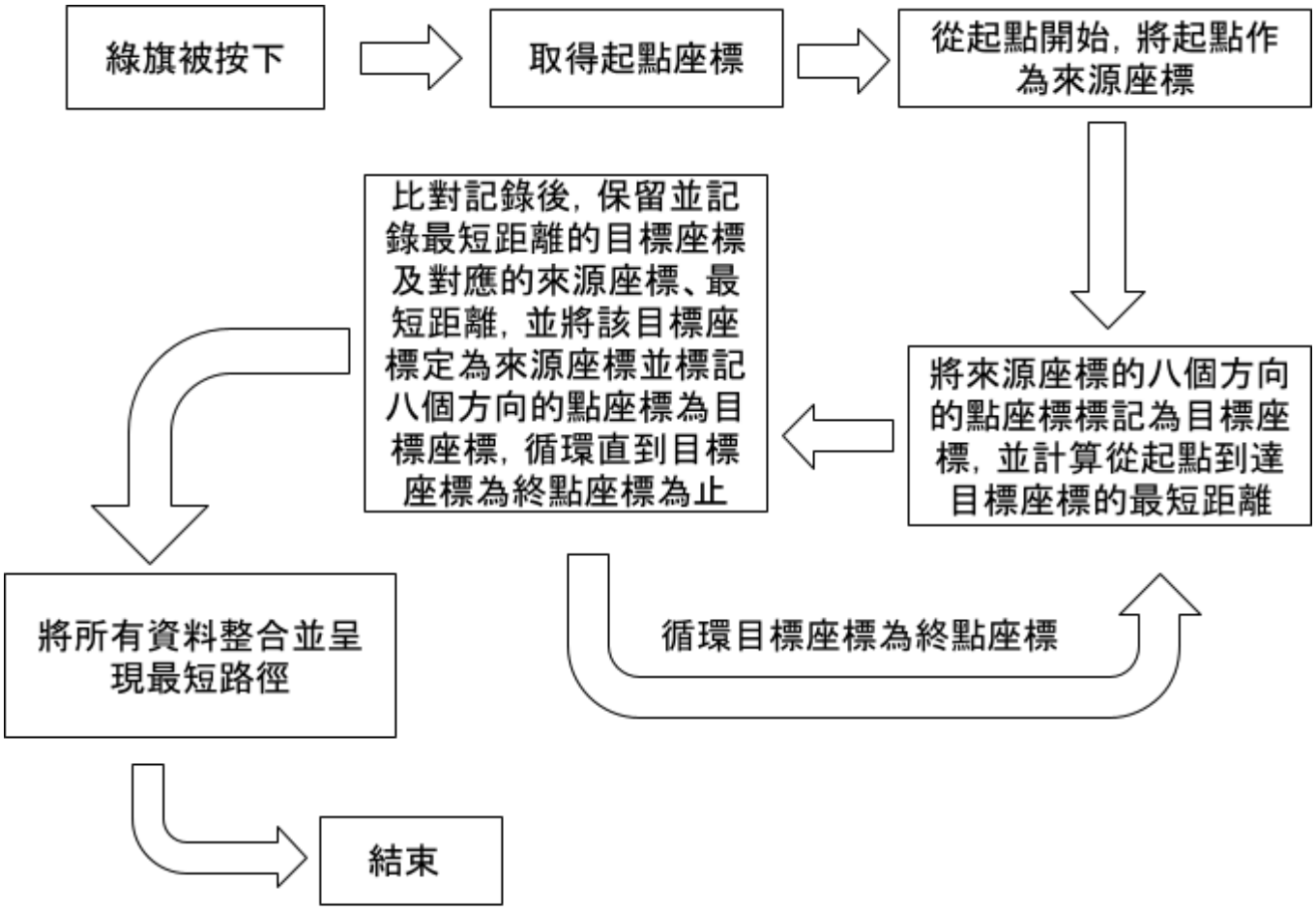


圖 3-3 Scratch程序於Dijkstra演算法的執行流程
(圖片由第二作者繪製)

肆、研究結果

一、貪婪法在最短距離的成品展示

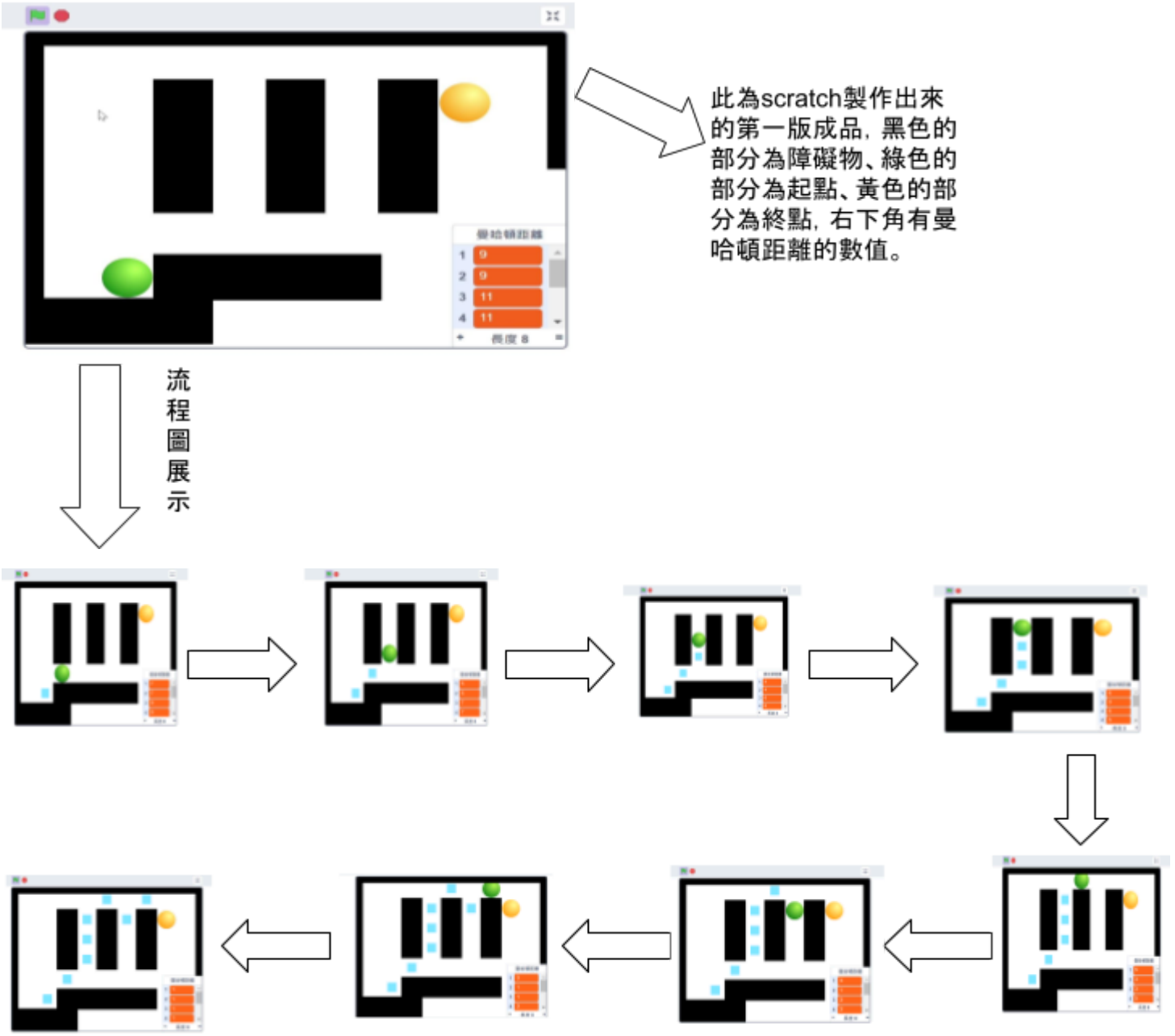


圖 4-1 貪婪法在最短距離的成品展示
(圖片由第二作者截圖)

當綠球（起點）與黃球（終點）重疊時，代表演算法已成功完成路徑搜尋。

二、貪婪法在最短距離的內部程式碼

(一)角色1=起點

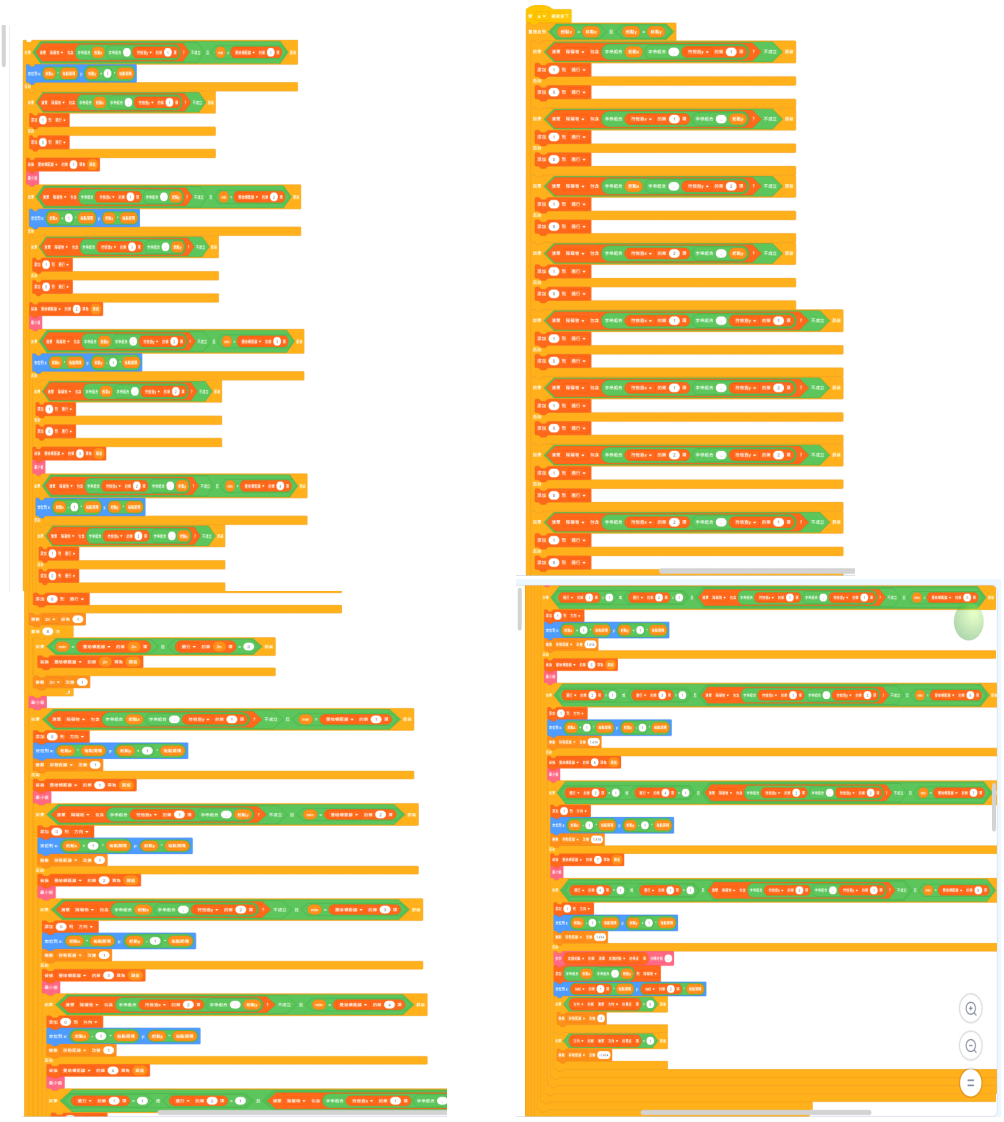


圖 4-2 貪婪法在最短距離的內部程式碼(角色1=起點)
(圖片由第二作者截圖)

(二)角色2=終點

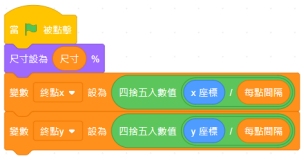


圖 4-3 貪婪法在最短距離的內部程式碼(角色2=終點)
(圖片由第二作者截圖)

(三)角色3=障礙物

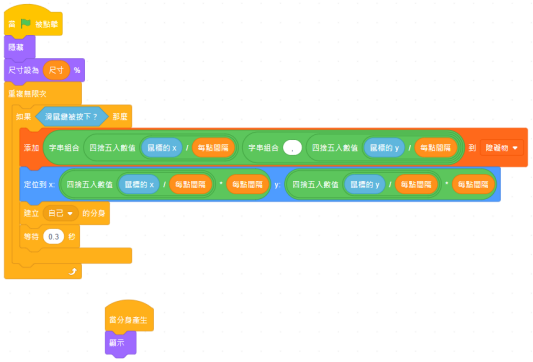


圖 4-4 貪婪法在最短距離的內部程式碼(角色3=障礙物)
(圖片由第二作者截圖)

(四)角色4=軌跡

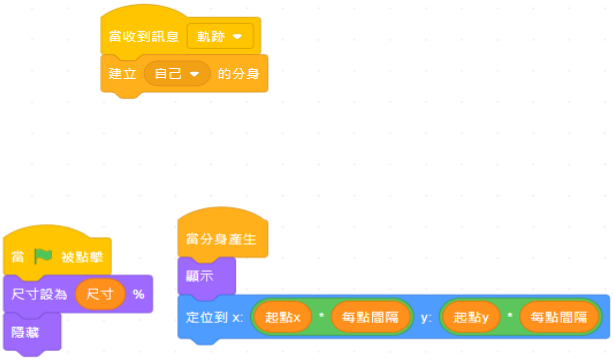


圖 4-5 貪婪法在最短距離的內部程式碼(角色4=軌跡)
(圖片由第二作者截圖)

三、Dijkstra演算法在最短距離成品展示

此為scratch製作出來的第二版成品，灰色的部分為障礙物、紅色的部分為起點、綠色的部分為終點，最短路徑的終點將會套上高亮的效果。

流程圖展示

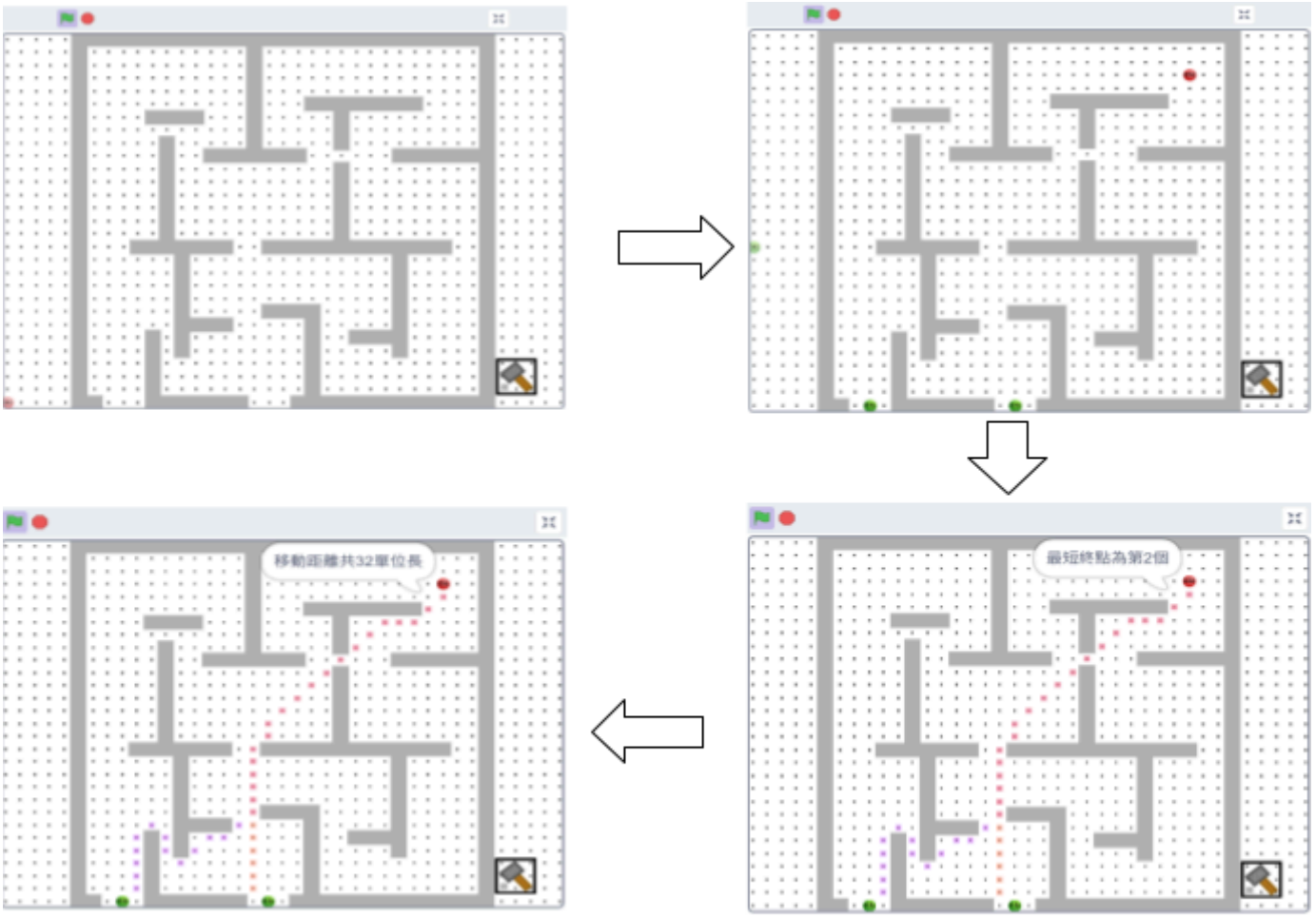


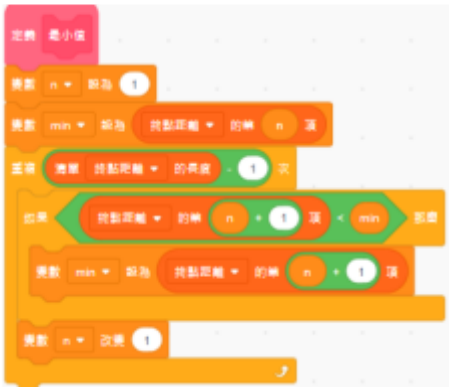
圖 4-6 Dijkstra演算法在最短距離成品展示

(圖片由第二作者截圖)

四、Dijkstra演算法在最短距離的內部程式碼(把常調用的程式定義為函式)

函式1(節點)

函式2(最小值)



函式3(拆字input分隔格符號s)
功能:把座標分解為x,y

圖 4-7 Dijkstra演算法在最短距離的內部程式碼(函式1、2、3)
(圖片由第二作者截圖)

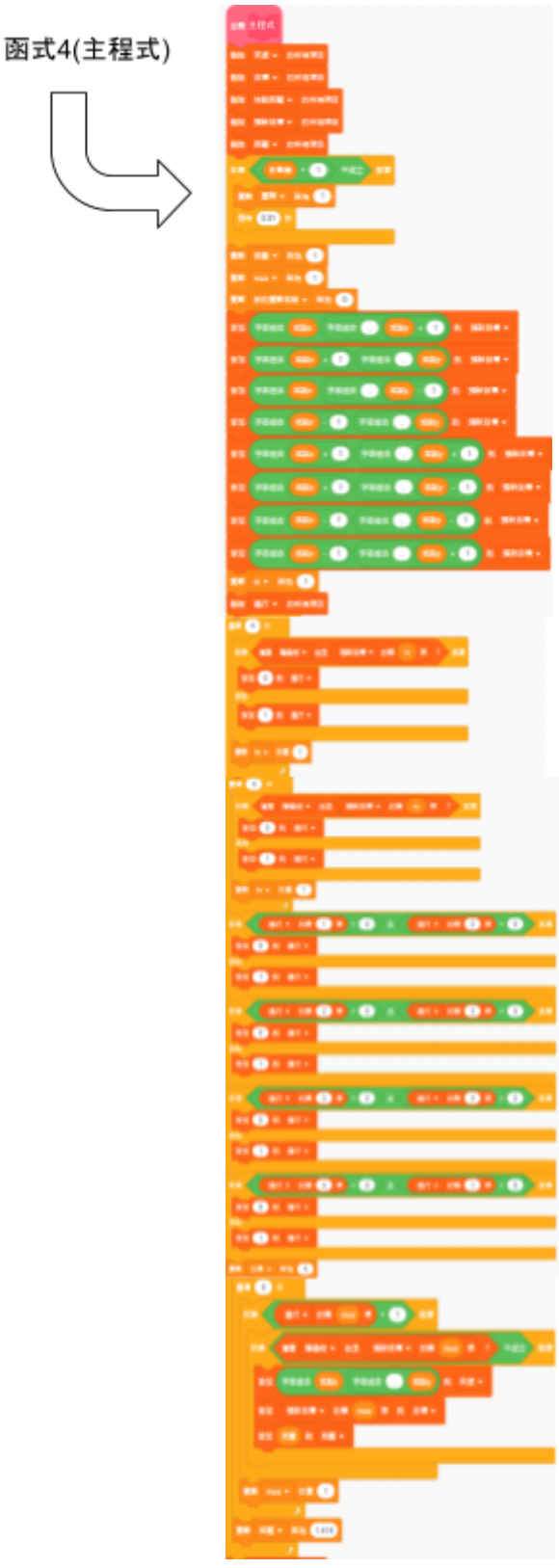


圖 4-8 Dijkstra演算法在最短距離的內部程式碼(函式4-1)
(圖片由第二作者截圖)

圖 4-9 Dijkstra演算法在最短距離的內部程式碼(函式4-2)
(圖片由第二作者截圖)

圖 4-10 Dijkstra演算法在最短距離的內部程式碼(函式4-3)
(圖片由第二作者截圖)

伍、討論

一、優勢分析：曼哈頓距離簡化計算，尤其適合網格化地圖，提升系統即時性。

二、研究限制：

(一)網格化地圖無法精確反映弧形走廊或斜向移動。

(二)Scratch 的運算效能限制大規模場景應用。

三、改進方向：

(一)結合貪婪法的效能優勢與Dijkstra演算法的最佳路徑規劃，減少最短路徑取得的效能需求。

(二)運用S4A結合感測器數據（如 IoT 人流計數器）強化即時路徑規劃與回饋。

陸、結論

一、主要發現：

(一)結合曼哈頓距離，能有效平衡路徑長度與計算效率。

(二)動態障礙機制，提升系統實用性。

二、應用價值：

(一)可整合至學校防災演練系統，提供即時疏散指引。

(二)框架可擴展至商場、捷運站等公共空間。

三、第一版(貪婪法)及第二版(Dijkstra演算法)比較

版本	第一版(貪婪法)	第二版(Dijkstra演算法)
成品展示		
比較	第一版的貪婪法因為運算當下最佳下一步，會被直角地形卡住，把每個可能走過的路徑都排除，而每次移動都需要進行運算，效率不佳。	第二版的dijkstra演算法會先從起點向外搜尋節點，直至節點中包含終點座標，再從終點往回推，運算只需一次，更高效。
距離比較	距離59.8個單位長	距離32個單位長
對比總結	從本次比較可以看出，第二版採用改良版的 Dijkstra 演算法，在搜尋路徑時效率明顯優於第一版的貪婪法。雖然貪婪法在初期看似快速，但因為容易走入彎路並需多次修正，導致總距離較長，效率較低。相較之下，改良後的 Dijkstra 演算法能全面考慮所有可能路徑，並在一次搜尋中找到最短路徑，不僅提升準確性，也大幅縮短了移動距離。因此，在追求最短路徑與高效率的情況下，第二版演算法是更理想的選擇。	

表5-1 第一版(貪婪法)及第二版(Dijkstra演算法)比較
(表格由第一作者及第二作者共同繪製)

柒、參考文獻資料

一、scratch wiki

(https://en.scratch-wiki.info/)

二、曼哈頓距離

(https://en.wikipedia.org/wiki/Taxicab_geometry)

三、畢氏定理

(https://zh.wikipedia.org/zh-tw/%E5%8B%BE%E8%82%A1%E5%AE%9A%E7%90%86)

四、貪婪法

(https://zh.wikipedia.org/zh-tw/%E8%B4%AA%E5%BF%83%E7%AE%97%E6%B3%95)

五、Dijkstra演算法

(https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

【評語】032814

1. 本作品運用 Scratch 進行建模，並結合兩種演算法進行最短逃生路徑比較，展現出良好的操作性與可行性，具備一定的實務應用潛力。

2. 若能進一步將目前的理論與模擬結果延伸至實際場域進行測試，並蒐集更多相關數據，以探討是否存在其他影響因素，將有助於讓作品更加貼近真實情境，進一步提升其應用價值與說服力，值得持續發展。

作品海報

以Scratch為模型規劃最短逃生路線

摘要

本研究針對超市等室內空間中之逃生路線進行規劃，並利用 Scratch 建立模擬模型，結合曼哈頓距離與 Dijkstra 演算法進行最短路徑分析。研究旨在開發一套具效率的逃生路線規劃工具，以供空間設計與安全規劃參考。

壹、前言

一、研究動機

近期台灣某商業大樓發生嚴重火災，起因為焊接作業產生的高溫火花，掉落至地下室並引燃大量易燃塗料與溶劑。火勢迅速蔓延，導致多人受傷，甚至喪生。此事件突顯逃生路線設計的重要性。火災發生時，能否在最短時間內抵達安全區域，往往決定生死。建築物內的逃生動線，應考量火勢擴散情況，確保人員能迅速、安全撤離。

本研究聚焦於最佳逃生路線設計，分析火災發生時，不同樓層人員如何選擇最短且最安全的路徑，以降低受困風險。

二、研究目的

本研究運用 Scratch 解決平面最短路徑問題，探討以下四項重點：

（一）距離計算方式比較：分析曼哈頓距離與歐幾里得距離在路徑長度與計算時間上的差異。

（二）移動方式比較：比較四向與八向移動對最短路徑的影響。

（三）演算法比較：比較貪婪法與 Dijkstra 演算法在路徑規劃上的表現。

（四）生活應用探討：評估 Scratch 在實際情境中解決最短路徑問題的可行性與限制。

貳、研究設備及器材

硬體：電腦（安裝 Scratch 3.0）

軟體：Scratch 3.0（搭配自訂擴展畫筆功能）

其他工具：模擬用網格地圖模板（以手繪或列印方式製作，輔助場景建構）

參、研究過程

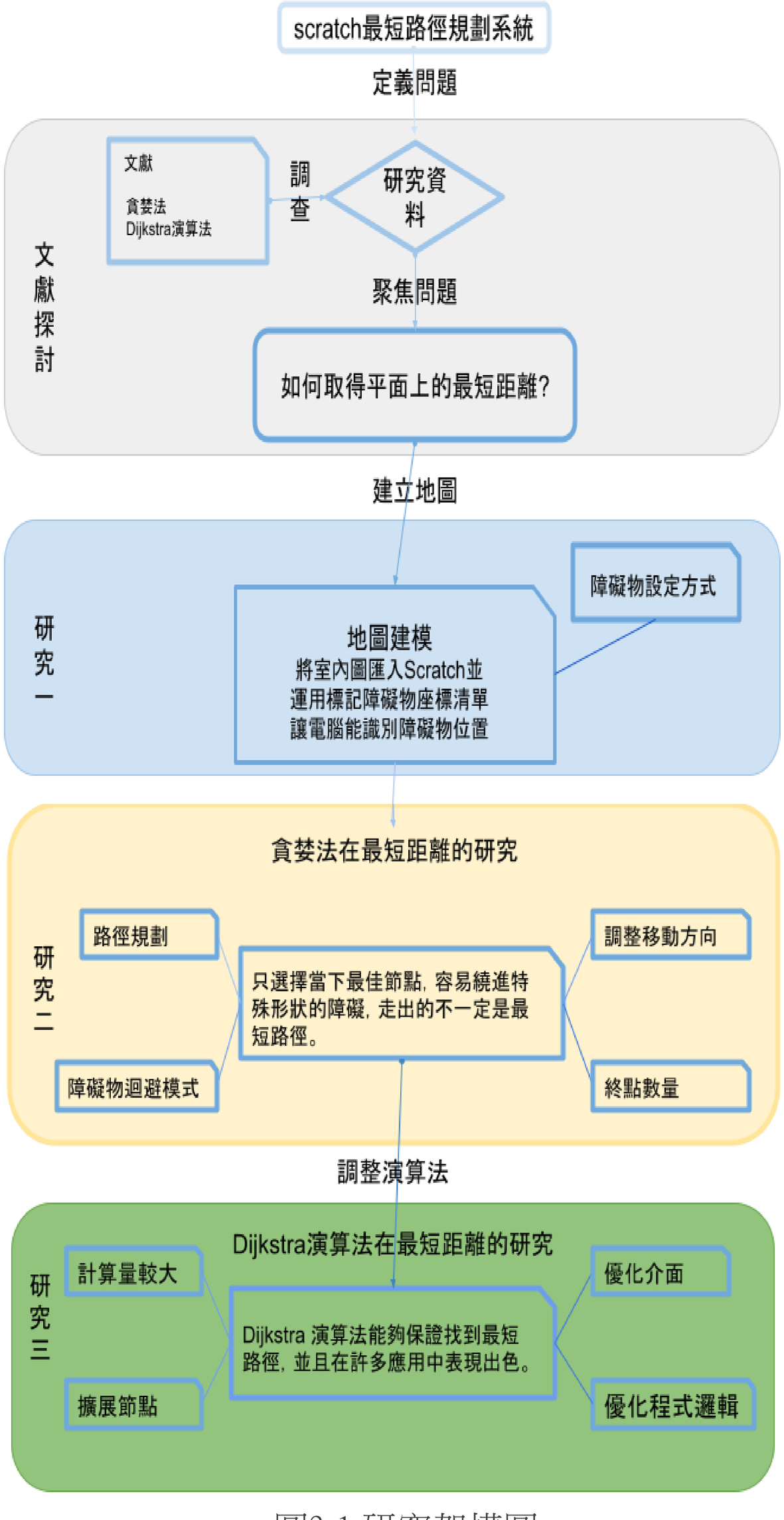


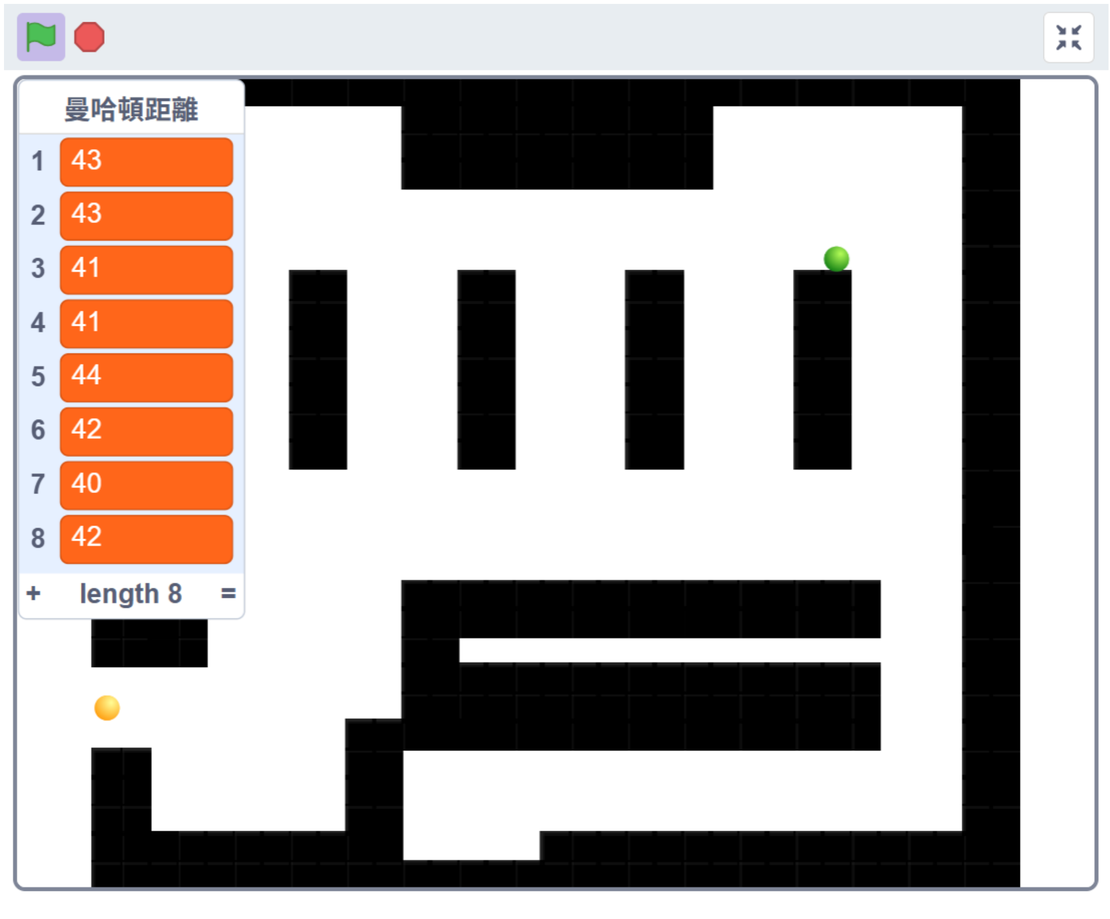
圖3-1 研究架構圖
(圖片由第一作者與第二作者共同繪製)

肆、研究結果

一、貪婪法在最短距離的成品展示

(一)、成品展示：

此為scratch製作出來的第一版成品，黑色的部分為障礙物、綠色的部分為起點、黃色的部分為終點，左上角有八個方向的曼哈頓距離的數值。



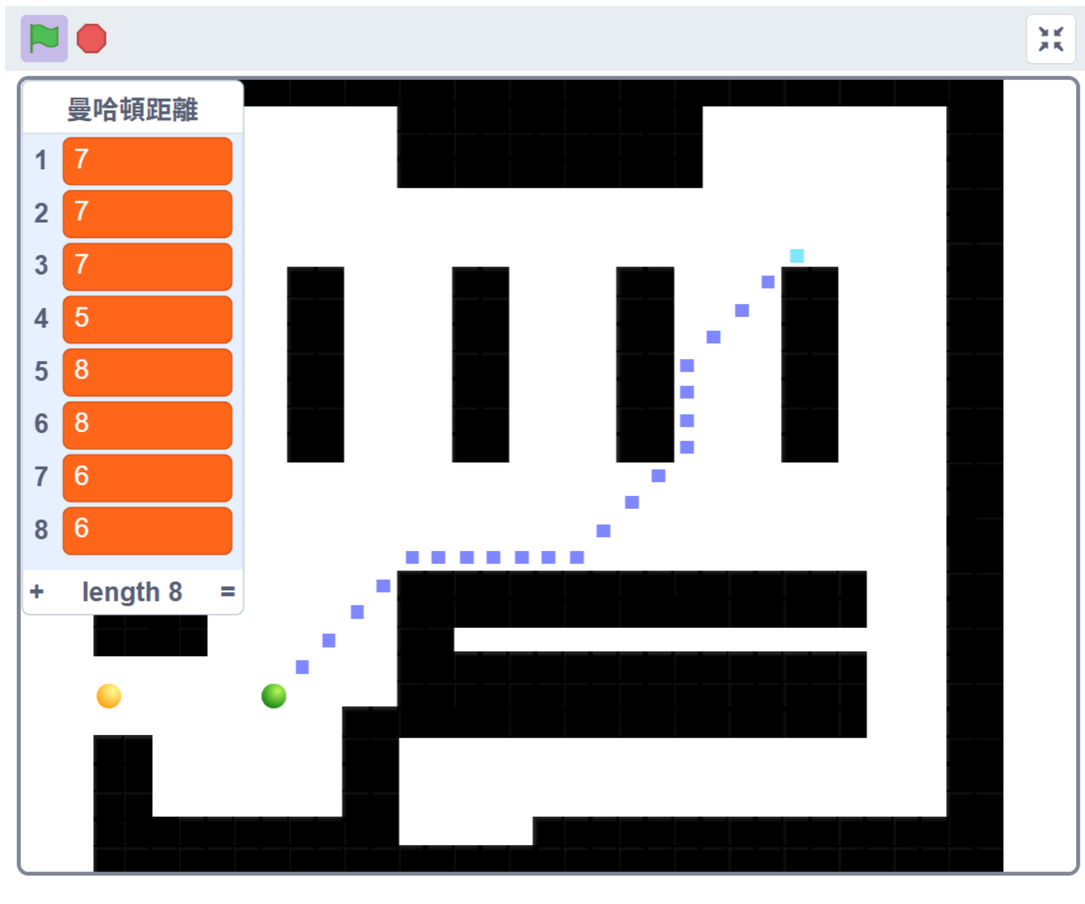


圖3-2 貪婪法路徑 圖1
(圖片由第二作者截圖)

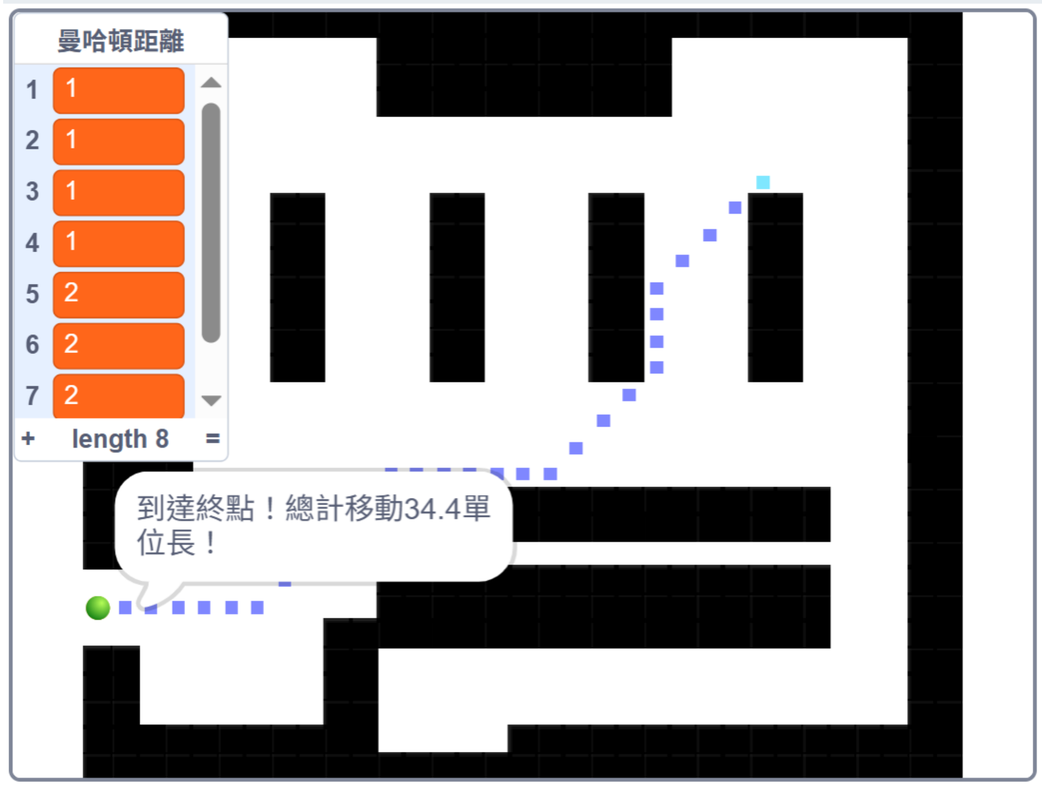


圖3-3 貪婪法路徑 圖2
(圖片由第二作者截圖)

二、Dijkstra在最短距離的成品展示

(一)、標記網格：

由於第一版的終點、起點的設置在背景並沒有明確的節點，我們在第二版以小點的方式表示28*38的網格。

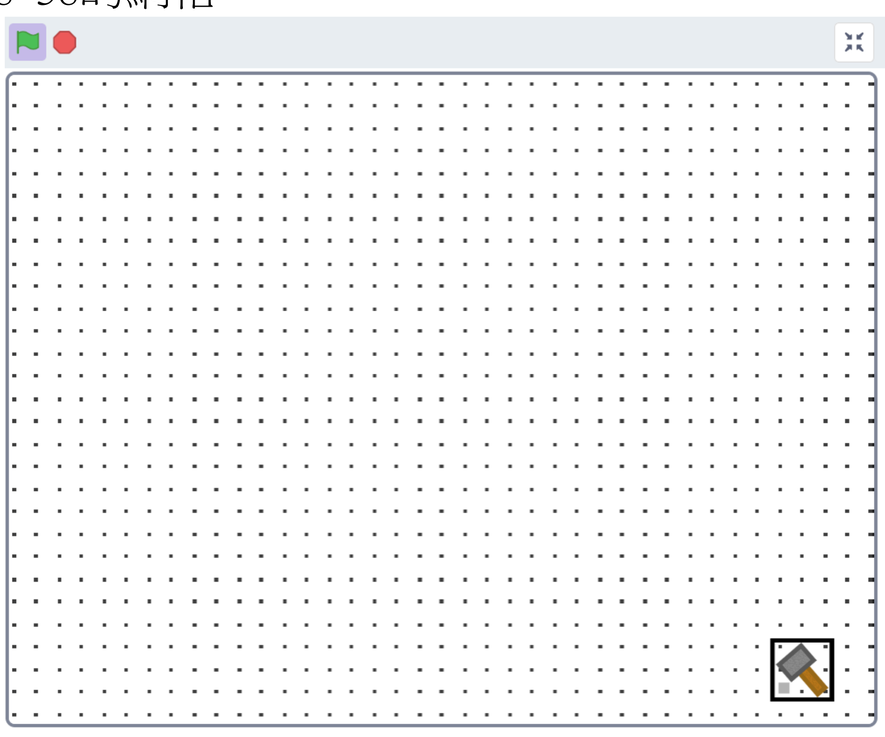


圖3-4 網格地圖呈現
(圖片由第一作者截圖)

(二)、成品展示：

此為Scratch製作出來的第二版成品，灰色的部分為障礙物、紅色的部分為起點、綠色的部分為終點，最短路徑的終點將會套上高亮的效果。

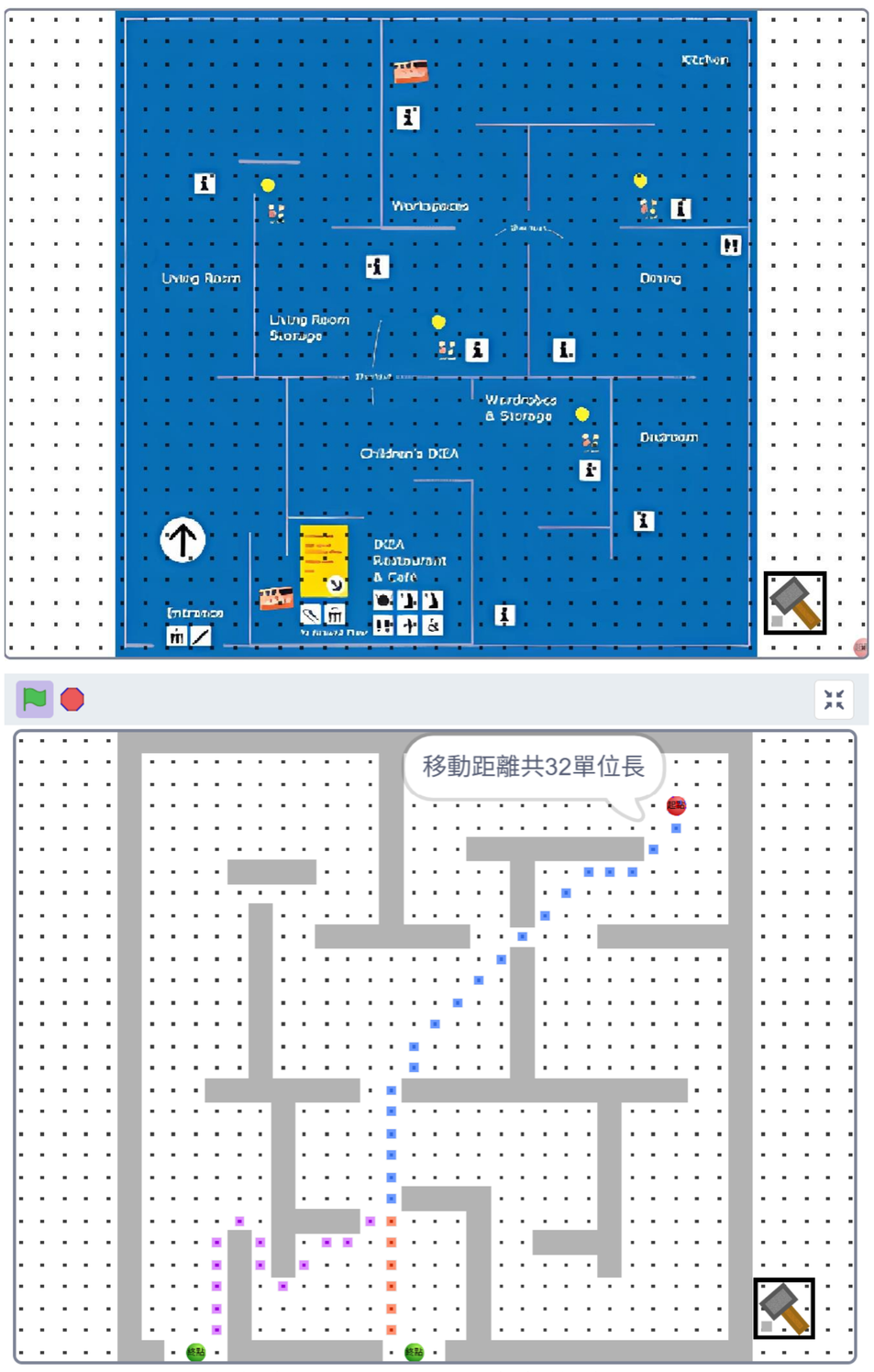


圖3-5 Dijkstra路徑圖1
(圖片由第二作者截圖)

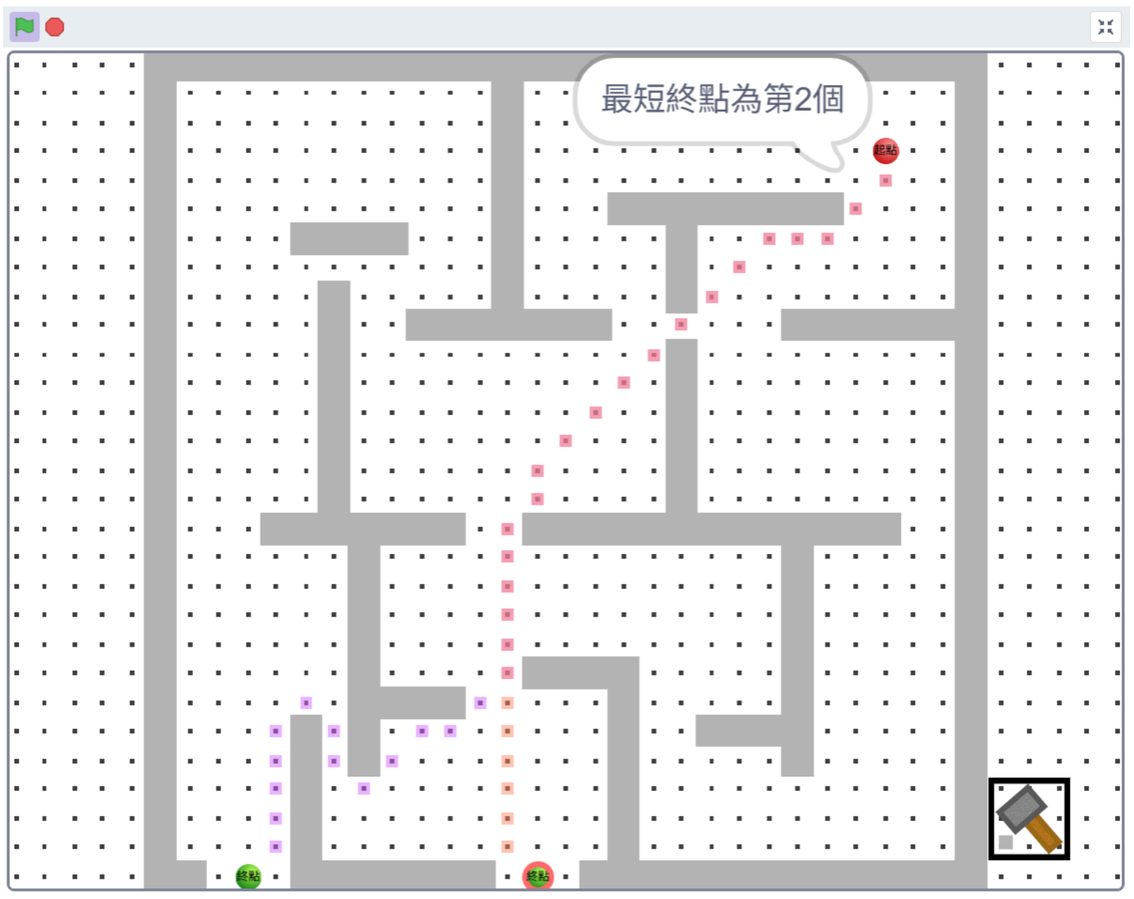


圖3-6 Dijkstra路徑圖2
(圖片由第二作者截圖)

三、研究成果

(一)、第一版研究成果：

貪婪法因優先移動至距離終點最近的相鄰節點，容易被直角障礙卡住，需要不斷的嘗試錯誤才能繞過障礙物，增加了許多額外的移動距離，雖然不用暫存太多的數據，但到達終點前所需移動的距離不見得是最短的。

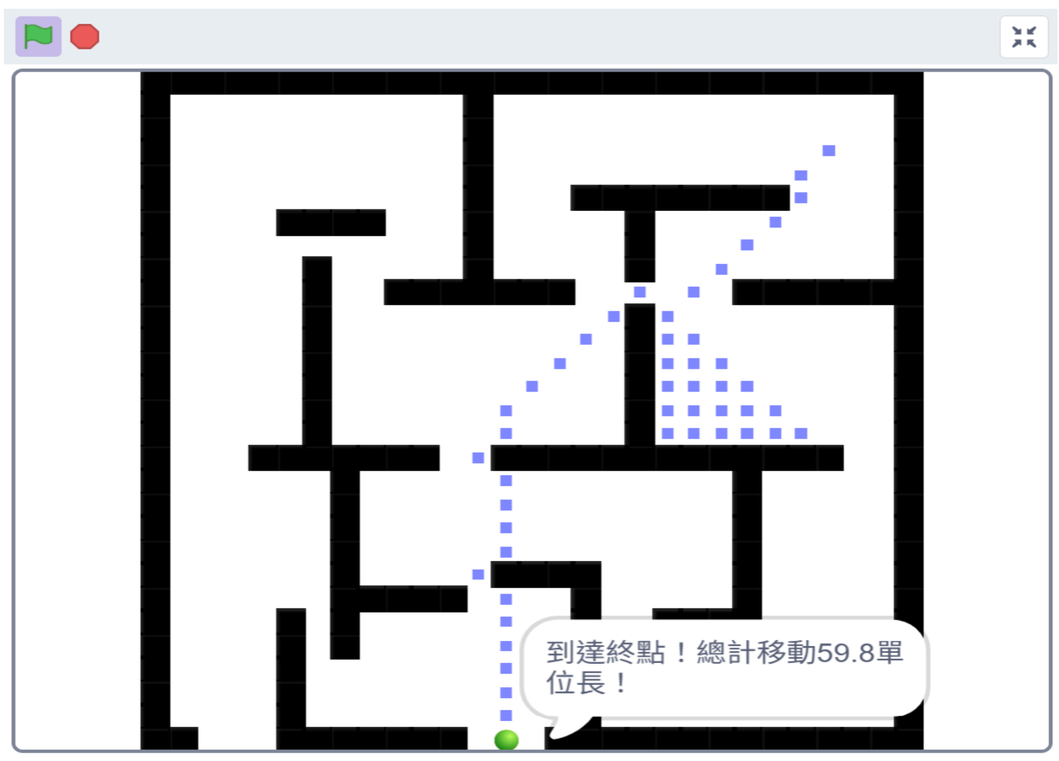


圖3-7 第一版研究結果
(圖片由第二作者截圖)

(二)、第二版研究成果：

Dijkstra演算法會先從起點向外搜尋可移動的節點並記錄每個節點的最短移動路徑，直至節點中包含終點座標，再從終點往回推論最短移動路徑，雖然計算量更大，要暫存的數據也更多，但求出的移動距離最短。

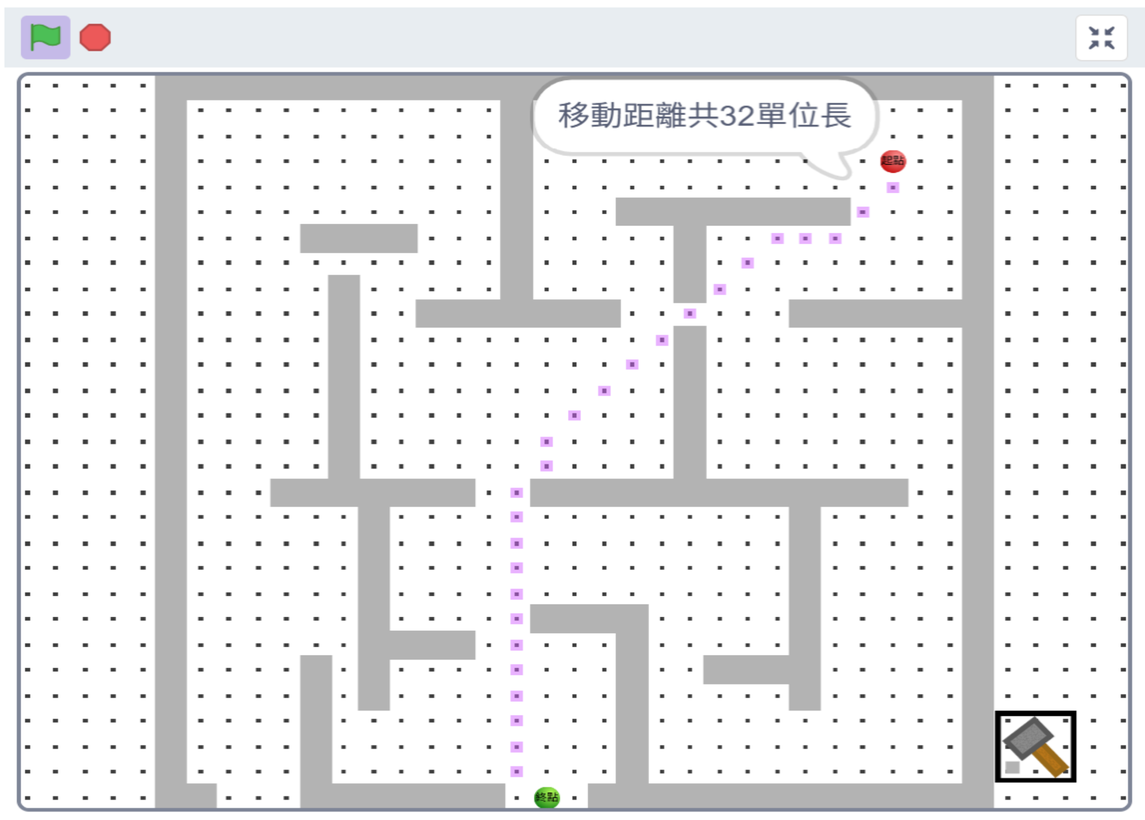


圖3-8 第二版研究成果
(圖片由第二作者截圖)

	貪婪法	Dijkstra演算法
距離比較	59.8個單位長	32個單位長
運行時間	運行五次 平均時間 5.082秒	運行五次 平均時間 0.528秒
紀錄數據	無關地圖大小，僅需保留八個方向的曼哈頓距離資料，共八筆資料。	地圖越大資料越多，以28*38的網格為例，會記錄每個節點的來源、目標、距離項目，三列各837筆資料。
版本特色	第一版的貪婪法每次都移動到距離終點最近的鄰近節點，容易被直角障礙卡住，卡住時雖然可以後退尋找出路，但是會嘗試所有鄰近節點直到找到可行路徑。	第二版的Dijkstra演算法會先從起點向外搜尋節點，直至終點，再從終點往回連結符合最短路徑的節點，可以得到絕對的最短路徑。

表3-1 研究結果對照 表1
(圖片由第一作者與第二作者共同繪製)

圖3-9 貪婪法迷宮路徑圖
(圖片由第二作者截圖)

圖3-10 Dijkstra迷宮路徑圖
(圖片由第二作者截圖)

	貪婪法	Dijkstra演算法
距離比較	266.5個單位長	121個單位長
運行時間	運行五次 平均時間 24.09秒	運行五次 平均時間 0.53秒
紀錄數據	無關地圖大小，僅需保留八個方向的曼哈頓距離資料，共八筆資料。	地圖越大資料越多，以28*38的網格為例，會記錄每個節點的來源、目標、距離項目，三列各834筆資料。
版本特色	第一版的貪婪法會在分歧點把每條可能的路徑走過，再以退步形式回到分歧點，重新選擇下一條路徑。	第二版的Dijkstra演算法會先從起點向外搜尋節點，直至終點，不會被死路影響。

表3-2 研究結果對照 表2
(圖片由第一作者與第二作者共同繪製)

肆、討論

一、優勢分析：曼哈頓距離簡化計算，尤其適合網格化地圖，提升系統即時性。
二、研究限制：

(一)網格化地圖無法精確反映弧形走廊或斜向移動。　　(二)Scratch 的運算效能限制大規模場景應用。

三、改進方向：

(一)結合貪婪法的效能優勢與Dijkstra演算法的最佳路徑規劃，減少最短路徑取得的效能需求。

(二)運用S4A結合感測器數據（如 IoT 人流計數器）強化即時路徑規劃與回饋。

伍、結論

一、主要發現：

(一)結合曼哈頓及Dijkstra演算法，能有效平衡路徑長度與計算效率。

(二)動態障礙機制，提升系統實用性。

(三)Scratch 具備直觀的視覺化操作介面，提升系統可用性與操作效率。

二、應用價值：

(一)可整合至學校防災演練系統，提供即時疏散指引。

(二)框架可擴展至商場、捷運站等公共空間。

陸、參考文獻資料

一、scratch wiki
(https://en.scratch-wiki.info/)

二、曼哈頓距離
(https://en.wikipedia.org/wiki/Taxicab_geometry)

三、畢氏定理
(https://en.wikipedia.org/wiki/Pythagorean_theorem)

四、貪婪法
(https://en.wikipedia.org/wiki/Greedy_algorithm)

五、Dijkstra演算法
(https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)