中華民國第 60 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學(一)科

032807

〝拐彎抹角〞 --- 擦窗車最佳清潔路徑的運算

學校名稱：臺中市私立弘文高級中學(附設國中)

作者：	指導老師：
國一　莊貽竣 國一　謝昀佑 國一　廖柏甯	賴玉艷

關鍵詞：平面坐標、極坐標、擦窗車

摘要

每到學校打掃時間，我們常看到同學擦不到高處的窗戶或不想擦，因而聯想到從事清潔玻璃的工人，因 PM2.5 的夢魘未改善，又得搭乘吊車、冒生命危險擦洗大樓窗戶，加上曾在網路上看到自動擦窗車，所以想改善這些問題。我們以資訊課學的 Scratch3.0 編程、運算擦窗面積及路徑長度，設計三大類擦窗路徑，再以 mblock 轉換成 Python 語法、以 Excel 作數據分析。

研究結果為：「直線來回法」乾淨度最佳；「六角形法」最省水、也省電，節能效果最佳；「雨刷法」最省電，擦窗效能最佳。未來，希望能應用在日常生活中，代替人類進行玻璃清潔的工作，無論高低或大面積的窗戶，都能讓擦窗車在〝拐彎抹角〞時，達到最佳的清潔效能。

壹、研究動機

在學校，每到打掃時間，常看到身型矮小的同學擦不到上方的窗戶或打掃不積極的同學不想擦窗戶，因而聯想到我們的居住環境，都市大樓林立，PM2.5 的夢魘至今仍揮之不去，因此，玻璃的清潔工作與日俱增。這項工作目前大都依賴清潔工人搭乘吊車、冒生命危險替居民服務。但是，墜樓事件頻傳，如何藉助一些擦窗神器，不用爬窗到外面，也能把玻璃窗戶擦洗乾淨，是本研究的發想動機。加上在瀏覽網頁時，曾經看過網路上有賣自動洗窗機。因此，我們想要研發節能擦窗車，利用資訊課所學的 Scratch3.0 編程及運算它的最佳清潔路徑，希望在〝拐彎抹角〞時，達到我們的研究目標！

照片 1-1 玻璃機器人(來源：youtube)

貳、研究目的

(一)探討「一筆畫法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

(二)探討「平面坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

(三)探討「極坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

叁、名詞解釋及研究過程

一、名詞解釋

(一) 一個好的路徑規劃演算法應該為機器爭取最少的運作時間、盡量減少機器傷害，並且同時達到最大目標效果，例如：省水、省電…。

1. 循序掃描法：連續實心圖形有最佳表現，但遇到不連續圖形或空心圖形效率會大幅下降。上下型及左右型填充率為所有演算法中最低，但斜向掃描卻是所有演算法中最高。上下及左右型演算法在角度統計的資料中，有最佳表現。斜向掃描演算法產生最多畫筆狀態改變次數，推測其與填充密度相關。(朱雁丞、柯昱任、謝佳峻，2008)。移動方向的估測(motion estimate)、相鄰場的內插(inter interpolation)、相同場的內插(inter interpolation)和移動補償(motion compensation)實現移動補償解交錯(motion compensation)。(維基百科，2017)。
2. 重複邊緣：重複進行邊緣搜尋逐步深入圖形內部，以達到填滿效果。(朱雁丞、柯昱任、謝佳峻，2008)。
3. 深度優先搜尋法：深度優先搜尋是圖論中的經典演算法，利用深度優先搜尋演算法可以產生目標圖的相應拓撲排序表，利用拓撲排序表可以方便的解決很多相關的圖論問題，如最大路徑問題等等。(維基百科，最後修訂於 2019 年 6 月 11 日)。
4. 廣度優先搜尋法：是一種圖形(graph)搜索演算法。從圖的某一節點(vertex, node)開始走訪，接著走訪此一節點所有相鄰且未拜訪過的節點，由走訪過的節點繼續進行先廣後深的搜尋。以樹(tree)來說即把同一深度(level)的節點走訪完，再繼續向下一個深度搜尋，直到找到目的節點或遍尋全部節點。(網路資料，2019 年 10 月 26 日節錄)。

(二) 本研究採用之路徑規劃演算法：

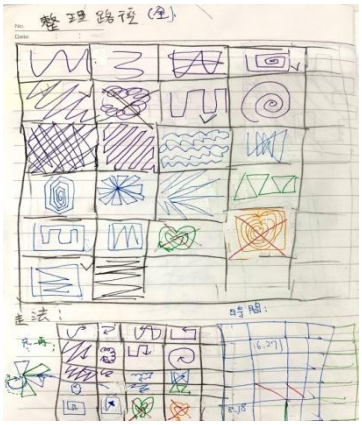
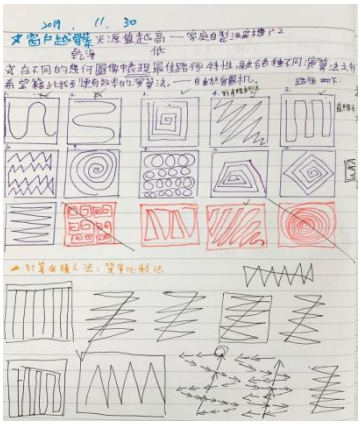
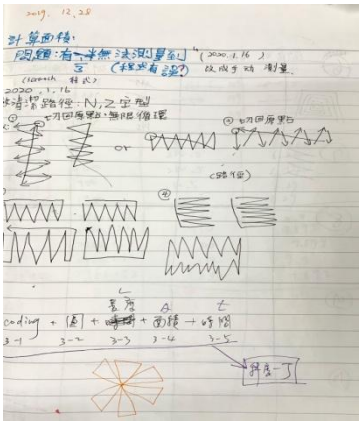
1. 「一筆畫法」：一筆畫問題（Eulerian graph）是圖論中一個著名的問題。一筆畫問題起源於柯尼斯堡七橋問題。數學家歐拉在他 1736 年發表的論文《柯尼斯堡的七橋》中不僅解決了七橋問題，也提出了一筆畫定理，順帶解決了一筆畫問題。一般認為，歐拉的研究是圖論的開端。(維基百科，最後修訂於 2019 年 10 月 19 日)。

2. 「平面坐標法」：也稱為直角坐標系，是最常用到的一種坐標系。是法國數學家勒內·笛卡兒在 1637 年發表的《方法論》附錄中提到的。在平面上，選定二條互相垂直的線為坐標軸，任一點距坐標軸的有號距離為另一軸的坐標，這就是二維的笛卡兒坐標系，一般會選一條指向右方水平線稱為 x 軸，再選一條指向上方的垂直線稱為 y 軸，此兩坐標軸設定方式稱為「右手坐標系」。(維基百科，最後修訂於 2019 年 10 月 19 日)。
3. 「極坐標法」：極坐標系（英語：Polar coordinate system）是一個二維坐標系統。該坐標系統中任意位置可由一個夾角和一段相對原點—極點的距離來表示。極坐標系的應用領域十分廣泛，包括數學、物理、工程、航海、航空以及機器人領域。在兩點間的關係用夾角和距離很容易表示時，極坐標系便顯得尤為有用；而在平面直角坐標系中，這樣的關係就只能使用三角函數來表示。對於很多類型的曲線，極坐標方程式是最簡單的表達形式，甚至對於某些曲線來說，只有極坐標方程式能夠表示。(維基百科，最後修訂於 2019 年 8 月 9 日)。

(三) 本研究所設計的路徑手繪圖初稿及實際編程的路徑名稱與說明：

1. 本研究初始預設人類可能的擦窗路徑手繪圖：

表 3-1 本研究所設計的路徑手繪圖初稿

人類可能的擦窗路徑手繪圖	選擇可編程的路徑
	
討論擦窗行為與邏輯運算思為關係	面積運算方法及說明書內容討論
	

2. 實際編程的路徑名稱與說明：

表 3-2 本研究所設計編程的路徑名稱與說明

演算法		擦窗法		說明
1	一筆畫法	1A	直線來回法	定位到左上角，移動到左下角。往右轉 90 度，再往上轉 90 度。
		1B	對角線來回法	定位到左上角，每次轉約 25 度上下來回移動。
		1C	邊界螺旋法	沿邊界碰壁時轉 90 度且起始邊界坐標減少車子的寬度，轉到邊界=或<1/2 邊界為止。
		1D	邊界斜線切入法	先定位到左上角，沿邊界碰壁時轉 90 度，重複來回三次。
		1E	縱坐標來回法	先定位到右下角，再移動到左上角，再斜線切入往右下約 75 度角，重複執行 3 次。最後再移動到右上角。
		1F	梯形螺旋法	先定位到右下角，斜線切入往右上約 75 度角，再往右移動 50 點，斜線切入往右下約 75 度角，再往左移動 60 點。再重複兩次，每次移動值減 10 點。
2	平面坐標法	2A	六角形迴圈法	定位到中心然後移動 100 點，在右轉 60 度，重複三次呈現梯形，再以每轉 20 度，重複進行 17 次。
		2B	T 形法	運用 T 字型的方式，X 軸移動 65 點後垂直往下，再回到往下移動的那點，最後以重覆執行來擴大面積。
		2C	田字法	先定位到中心點上方，再以 135 度角移動到點左方，再以同樣方式移動到，再來回到原點，最後在中間畫出一個十字來擴大面積。
		2D	六角形法	先畫一個六角形，然後以 90 度角把梯形切一半，再來滑行到梯形終點，最後從終點滑行到梯形的兩個角，就會在梯形中央呈現一個十字。
		2E	米字法	先定位到左上角，然後滑行到左下角，右下角和右上角，然後回到原點，再來已 45 度角切入右下，然後往內移動 44 點後再繞上去左上，再來已 45 度角切入右下，再繞到右上，最後在中間畫一個王字來擴大面積。
3	極坐標法	3A	箭靶(扇形 1)	畫完兩個同心圓再轉 30 度，接著在兩個同心圓中畫十字。
		3B	同心圓(扇形 2)	由中心繞出同心圓，直到車子上下碰觸邊界為止。
		3C	螺旋(扇形 3)	由固定一點繞出圓形螺旋，直到車子＜或=邊界為止。
		3D	雨刷(扇形 4)	由一支雨刷刷過固定 180 度面積，直到車子碰觸邊界為止。
		3E	雙雨刷(扇形 5)	兩支雨刷同時刷過固定 180 度面積，直到車子＜或=邊界為止。

二、研究過程及設備

(一) 探討「一筆畫法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

假設：一筆畫法路徑不同會影響擦窗的乾淨度、節能程度與擦窗效能

控制變因：刷頭大小、窗戶總面積、估算面積程式、估算路徑總長度程式

操縱變因：擦窗路徑

應變變因：覆蓋面積、路徑總長度

利用 Scratch3.0 以「一筆畫法」繪製 6 種擦窗路徑，估算覆蓋面積及路徑總長度，紀錄程式運算結果，進行排序比較，並以 mblock 轉換成 Python 相對應的程式語法。

(二) 探討「平面坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

假設：平面坐標法路徑不同會影響擦窗的乾淨度、節能程度與擦窗效能

控制變因：刷頭大小、窗戶總面積、估算面積程式、估算路徑總長度程式

操縱變因：擦窗路徑

應變變因：覆蓋面積、路徑總長度

利用 Scratch3.0 以「平面坐標法」繪製 5 種擦窗路徑，估算覆蓋面積及路徑總長度，紀錄程式運算結果，進行排序比較，並以 mblock 轉換成 Python 相對應的程式語法。

(三) 探討「極坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

假設：極坐標法路徑不同會影響擦窗的乾淨度、節能程度與擦窗效能

控制變因：刷頭大小、窗戶總面積、估算面積程式、估算路徑總長度的程式

操縱變因：擦窗路徑

應變變因：覆蓋面積、路徑總長度

利用 Scratch3.0 以「極坐標法」繪製 5 種擦窗路徑，估算覆蓋面積及路徑總長度，紀錄程式運算結果，進行排序比較，並以 mblock 轉換成 Python 相對應的程式語法。

(四) 研究設備與軟體：ASUS 筆記型電腦、mBot、Scratch3.0、mblock 編程軟體、Excel 軟體。先選擇可編程的擦窗路徑，利用 Scratch3.0 編程，再將檔案匯入 mblock 編程軟體轉換 Python 程式語言，並以 Excel 軟體進行數據處理與繪製雷達圖，作為分析比較依據。

(五) 擦窗路徑的各項數據來源：面積估算是利用自製計算程式(小方格)來運算，一個小方格是 20*20，一個普通刷頭大小是 7 格小方格，刷頭面積為 20*20*7(2800)；而扇形刷頭大小是 14 格小方格，所以刷頭面積為 20*20*14(5600)，再將路徑長度(l)*刷頭面積就會等於擦窗面積(A)；而覆蓋面積(B)為程式執行後，覆蓋格數*小方格面積 20*20；重疊面積(C)為擦窗面積(A)減覆蓋面積(B)而得；擦窗效能則為覆蓋面積(B)與路徑長度(l)的比值。各種面積及擦窗效能的計算公式整理如下：

1. **路徑長度(l) =利用坐標定位估算路徑總長**
2. **一般刷頭 ⇒ 擦窗面積(A) =路徑長度(l)*2800/20**
   扇形刷頭 ⇒ 擦窗面積(A) =路徑長度(l)*5600/20
3. **覆蓋面積(B) = 覆蓋格數*400**
4. **擦窗面積(A)－覆蓋面積(B) =重疊面積(C)**
5. **擦窗效能 = 覆蓋面積(B)/路徑長度(l)**

(六) 刷頭設計

本研究擦窗車的刷頭設計分為兩種，分別為一般刷頭(圓形)與扇形刷頭(棒形)，一般刷頭適用於一筆畫法、平面坐標法的擦窗路徑；而扇形刷頭則適用於極坐標法中的雨刷路徑。

表 3-3 擦窗車刷頭-一般刷頭

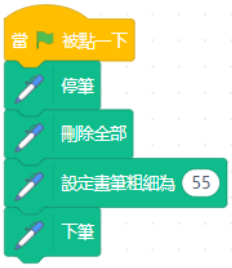
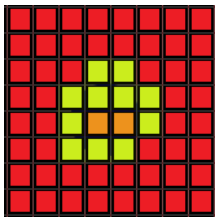
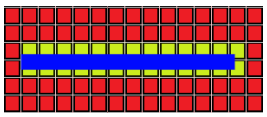
	路徑控制設計	Python 程式語法	路徑圖	刷頭面積堆疊圖
一般刷頭		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.penup() 6 sprite.clear() 7 sprite.pensize(55) 8 sprite.pendown()		20*20*7(2800) 

表 3-4 擦窗車刷頭-扇形刷頭

	路徑控制設計	Python 程式語法	路徑圖	刷頭面積堆疊圖
扇形刷頭		Python 1 from mblock import even 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.show() 6 sprite.direction = 7 sprite.x = -9 8 sprite.y = 149 9 sprite.clear()		20*20*14(5600) 

(七) 本研究中，各種擦窗路徑模擬進行擦拭的窗戶面積為平面坐標系統之方格，窗戶總面積設定為：24*18 =432 格。而模擬執行擦窗程式的機器為 mBot。

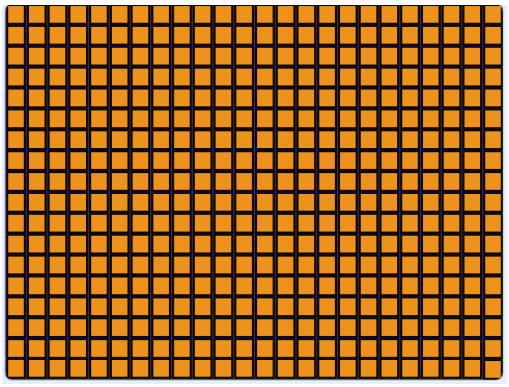


圖 3-1 窗戶總面積方格

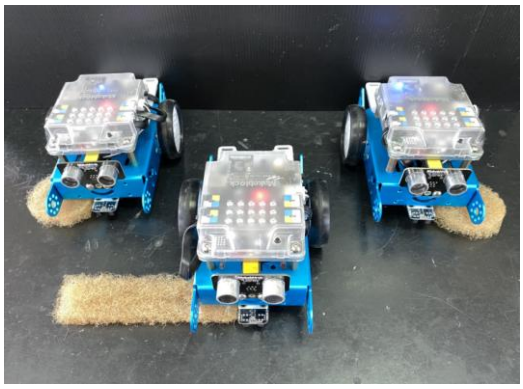
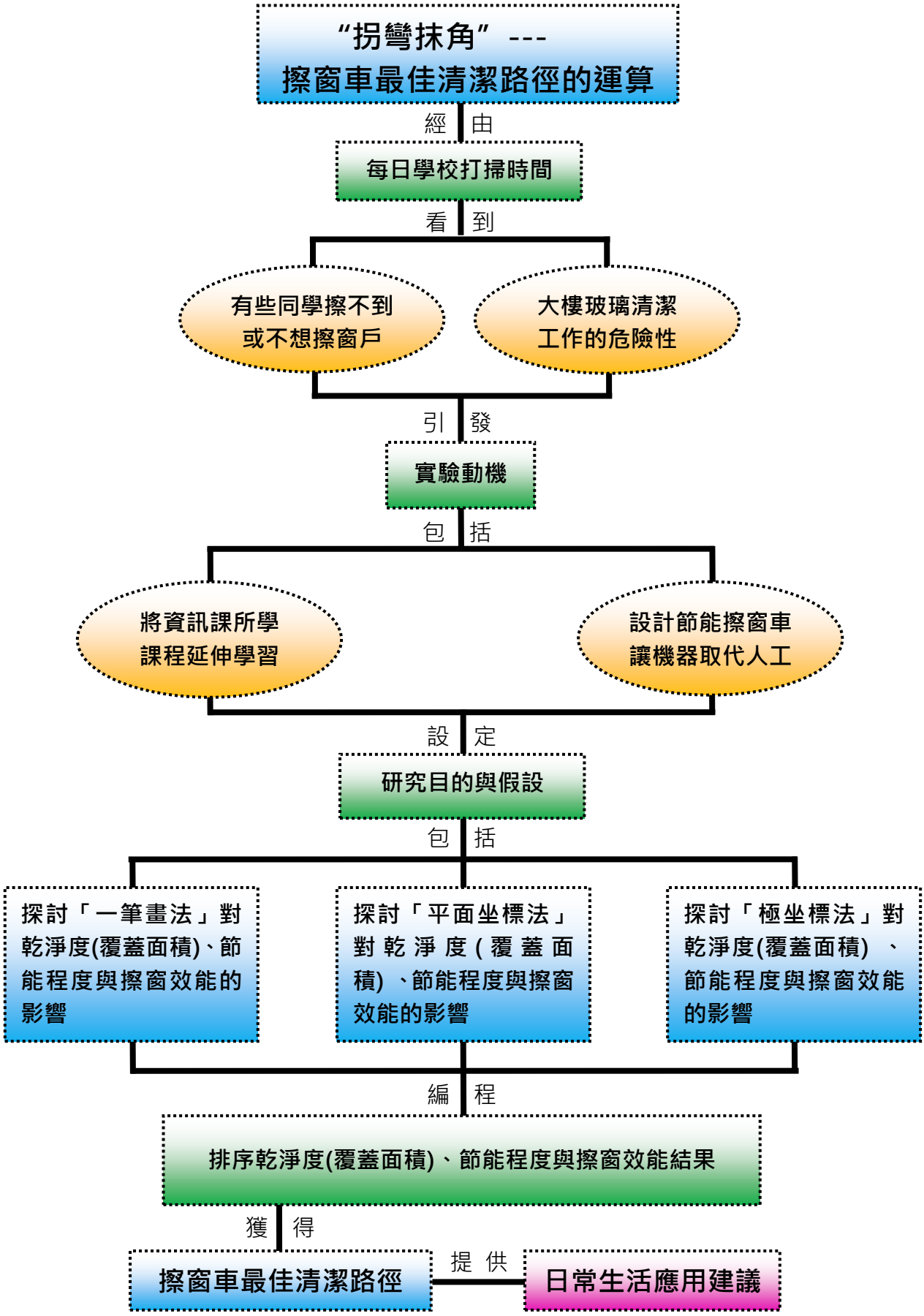


圖 3-2mBot

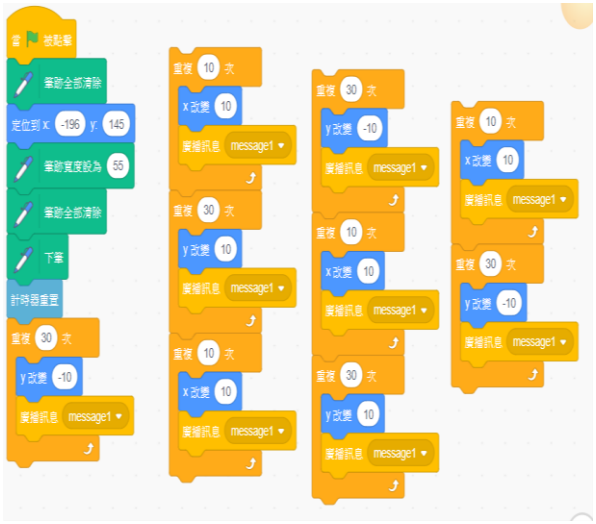
三、研究流程圖


“拐彎抹角” ---
擦窗車最佳清潔路徑的運算
經由
每日學校打掃時間
看到
有些同學擦不到
或不想擦窗戶
大樓玻璃清潔
工作的危險性
引發
實驗動機
包括
將資訊課所學
課程延伸學習
設計節能擦窗車
讓機器取代人工
設定
研究目的與假設
包括
探討「一筆畫法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響
探討「平面坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響
探討「極坐標法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響
編程
排序乾淨度(覆蓋面積)、節能程度與擦窗效能結果
獲得
擦窗車最佳清潔路徑
提供
日常生活應用建議

肆、結果與討論

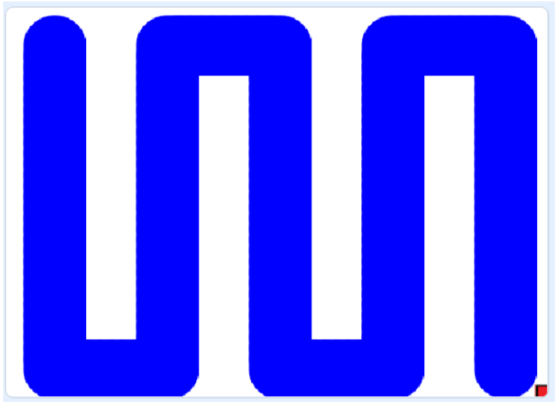
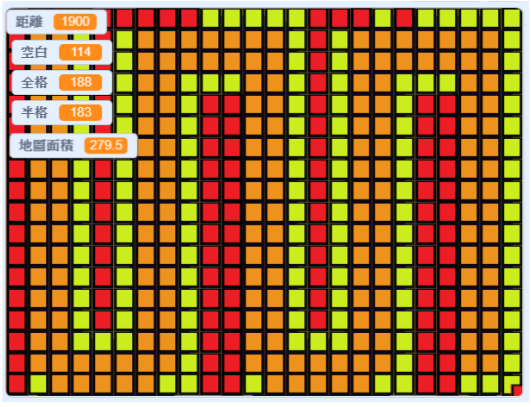
一、探討「一筆畫法」對乾淨度(覆蓋面積)、節能程度與擦窗效能的影響。

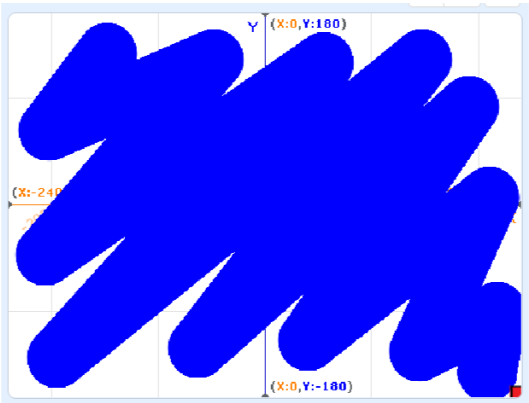
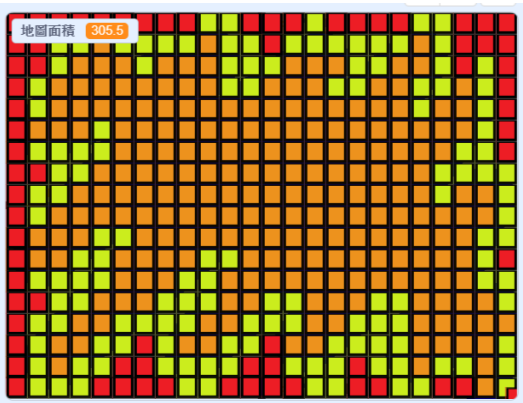
(一) 各種擦窗法之路徑編程、對應的 Python 程式語法、面積與長度計算圖

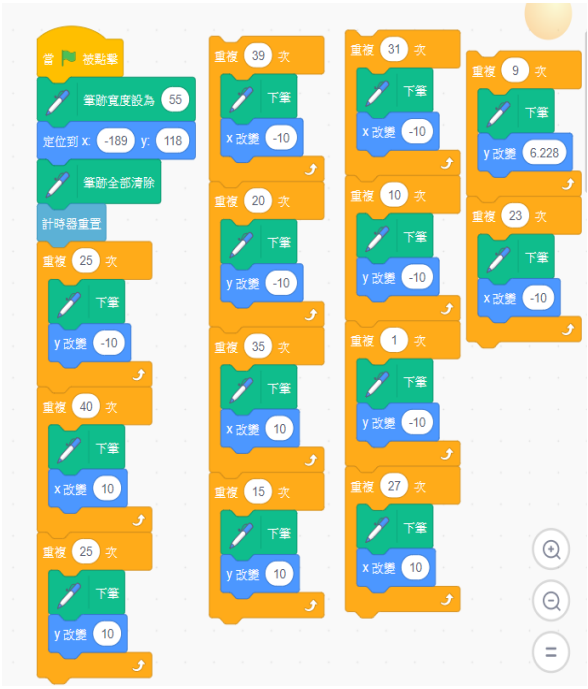
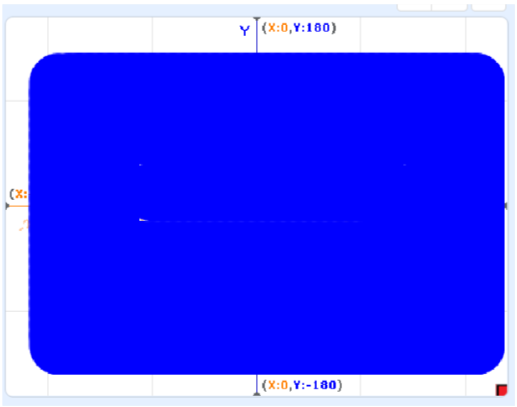
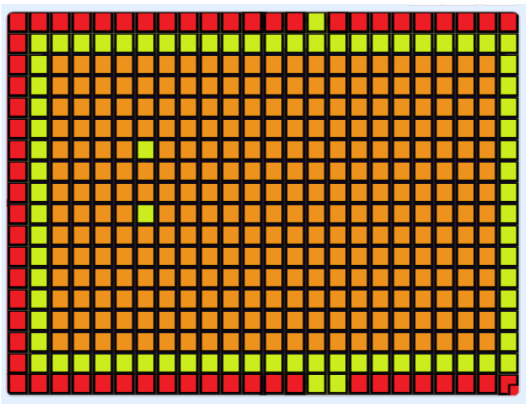
<table>
<tr><td>1A</td><td>路徑控制設計</td><td>對應的 Python 程式語法</td></tr>
<tr><td rowspan="3">直線來回法</td><td rowspan="2"> 

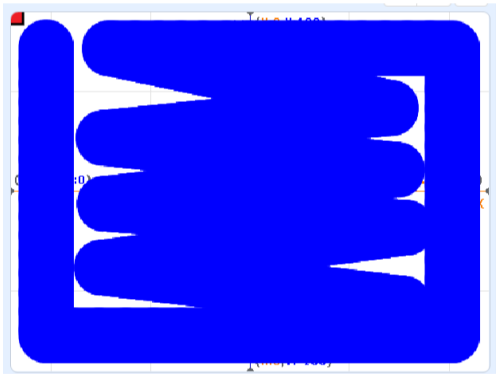
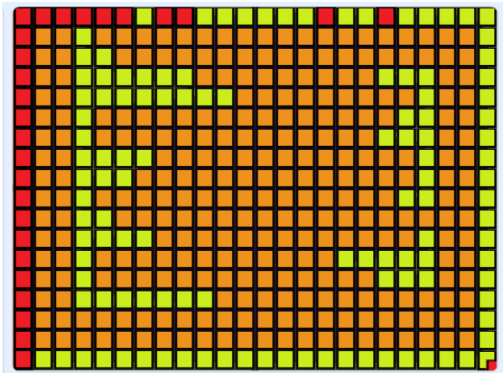
直線來回法擦窗車模擬實測影片:</td><td rowspan="2">Python

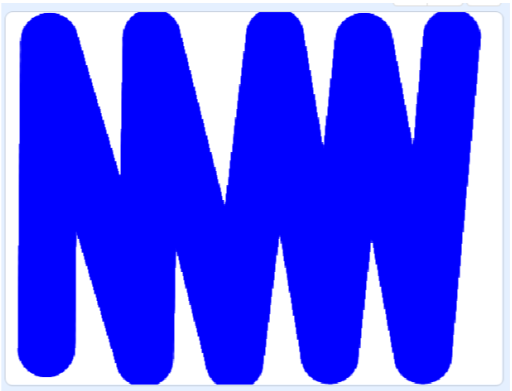
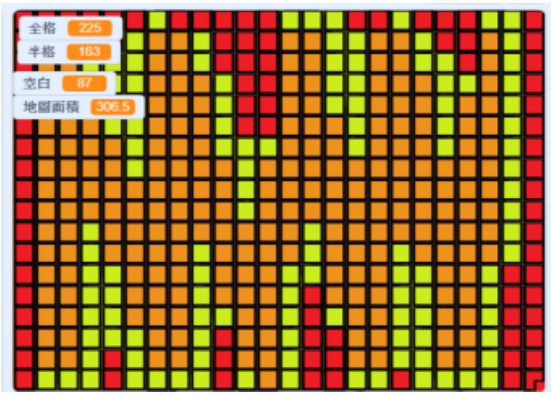
```
from mblock import event

@event.greenflag
def on_greenflag():
    # 程式
    sprite.clear()
    sprite.x = -196
    sprite.y = 145
    sprite.pensize(55)
    sprite.clear()
    sprite.reset_timer()
    for count in range(30):
        sprite.pendown()
        sprite.y = sprite.y + -10

    for count2 in range(10):
        sprite.x = sprite.x + 10

    for count3 in range(30):
        sprite.pendown()
        sprite.y = sprite.y + 10

    for count4 in range(10):
        sprite.x = sprite.x + 10

    for count5 in range(30):
        sprite.pendown()
        sprite.y = sprite.y + -10

    for count6 in range(10):
        sprite.x = sprite.x + 10


    for count7 in range(30):
        sprite.pendown()
        sprite.y = sprite.y + 10

    for count8 in range(10):
        sprite.x = sprite.x + 10

    for count9 in range(30):
        sprite.pendown()
        sprite.y = sprite.y + -10

    sprite.say(sprite.timer, 2)
```

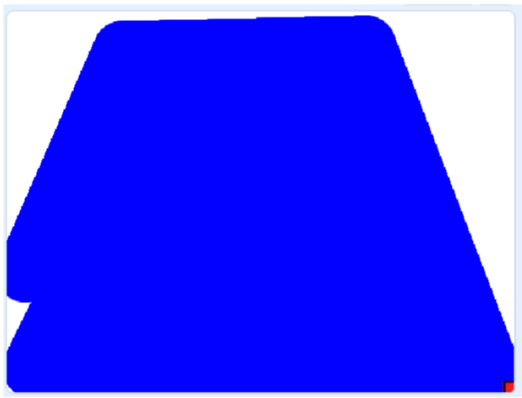
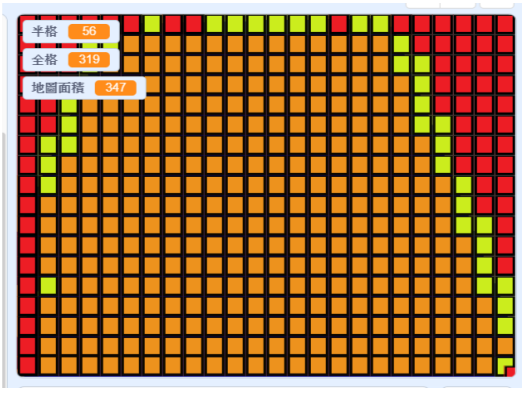
</td></tr>
<tr><td>路徑圖</td><td>清潔面堆疊圖</td></tr>
<tr><td>  </td><td> 
</td></tr>
</table>

1B	路徑控制設計	對應的 Python 程式語法
對角線來回法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.penup() 6 sprite.clear() 7 sprite.pensize(55) 8 sprite.reset_timer() 9 sprite.pendown() 10 sprite.x = -147 11 sprite.y = 143 12 sprite.clear() 13 sprite.glide(-202, 70, 0.5) 14 sprite.glide(-47, 137, 0.5) 15 sprite.glide(-205, -47, 0.5) 16 sprite.glide(56, 134, 0.5) 17 sprite.glide(-194, -143, 0.5) 18 sprite.glide(148, 137, 0.5) 19 sprite.glide(-62, -134, 0.5) 20 sprite.glide(192, 93, 0.5) 21 sprite.glide(41, -127, 0.5) 22 sprite.glide(211, 8, 0.5) 23 sprite.glide(145, -137, 0.5) 24 sprite.glide(218, -100, 0.5) 25 sprite.glide(209, -156, 0.5) 26 27 28 29 30 sprite.broadcast(str('message1'))
	路徑圖	清潔面堆疊圖
		

1C	路徑控制設計	對應的 Python 程式語法
邊界螺旋法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.pensize(55) 6 sprite.x = -189 7 sprite.y = 118 8 sprite.clear() 9 sprite.reset_timer() 10 for count in range(25): 11 sprite.pendown() 12 sprite.y = sprite.y + -10 13 14 for count2 in range(40): 15 sprite.pendown() 16 sprite.x = sprite.x + 10 17 18 for count3 in range(25): 19 sprite.pendown() 20 sprite.y = sprite.y + 10 21 22 for count4 in range(39): 23 sprite.pendown() 24 sprite.x = sprite.x + -10 25 26 for count5 in range(23): 27 sprite.pendown() 28 sprite.y = sprite.y + -10 29 30 for count6 in range(35): 31 sprite.pendown() 32 sprite.x = sprite.x + 10 33 34 for count7 in range(20): 35 sprite.pendown() 36 sprite.y = sprite.y + 10 37 38 for count8 in range(30): 39 sprite.pendown() 40 sprite.x = sprite.x + -10 41 42 for count9 in range(17): 43 sprite.pendown() 44 sprite.y = sprite.y + -10 45 46 for count10 in range(25): 47 sprite.pendown() 48 sprite.x = sprite.x + 10 49 50 for count11 in range(13): 51 sprite.pendown() 52 sprite.y = sprite.y + 10 53 54 for count12 in range(20): 55 sprite.pendown() 56 sprite.x = sprite.x + -10 57 58 for count13 in range(9): 59 sprite.pendown() 60 sprite.y = sprite.y + -10 61 62 for count14 in range(15): 53 sprite.pendown() 54 sprite.x = sprite.x + 10 55 56 for count15 in range(5): 57 sprite.pendown() 58 sprite.y = sprite.y + 10 59 70 for count16 in range(11): 71 sprite.pendown() 72 sprite.x = sprite.x + -10 73 74 sprite.say(sprite.timer, 5)
	路徑圖	清潔面堆疊圖
		

1D	路徑控制設計	對應的 Python 程式語法
邊界斜線切入法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.hide() 6 sprite.penup() 7 sprite.clear() 8 sprite.pensize(55) 9 sprite.x = -204 10 sprite.y = 145 11 sprite.pendown() 12 for count in range(29): 13 sprite.y = sprite.y + -10 14 15 for count2 in range(41): 16 sprite.x = sprite.x + 10 17 18 for count3 in range(29): 19 sprite.y = sprite.y + 10 20 21 for count4 in range(35): 22 sprite.x = sprite.x + -10 23 24 sprite.glide(140, 73, 1) 25 sprite.glide(-138, 46, 1) 26 sprite.glide(138, -13, 1) 27 sprite.glide(-132, -67, 1) 28 sprite.glide(145, -87, 1) 29
	路徑圖	清潔面堆疊圖
		

1E	路徑控制設計	對應的 Python 程式語法
縱坐標來回法		Python 1 from mblock import event 2 3 @event.keypressed('1') 4 def on_keypressed(): 5 sprite.hide() 6 7 8 @event.keypressed('5') 9 def on_keypressed1(): 10 sprite.hide() 11 12 13 @event.keypressed('2') 14 def on_keypressed2(): 15 pass 16 17 @event.keypressed('3') 18 def on_keypressed3(): 19 sprite.hide() 20 21 22 @event.greenflag 23 def on_greenflag(): 24 sprite.show() 25 sprite.penup() 26 sprite.clear() 27 sprite.x = -200 28 sprite.y = -145 29 sprite.pensize(55) 30 sprite.pendown() 31 sprite.glide(-197, 152, 1) 32 sprite.glide(-105, -155, 1) 33 sprite.glide(-98, 156, 1) 34 sprite.glide(-18, -158, 1) 35 sprite.glide(21, 157, 1) 36 sprite.glide(74, -152, 1) 37 sprite.glide(107, 152, 1) 38 sprite.glide(164, -152, 1) 39 sprite.glide(192, 156, 1) 40 sprite.hide() 41 42 43 @event.keypressed('4') 44 def on_keypressed4(): 45 sprite.hide() 46 47 48 49 50 51 52 53 54 55 sprite.broadcast(str('message1'))
	路徑圖	清潔面堆疊圖
		

1F	路徑控制設計	對應的 Python 程式語法
梯形螺旋法		Python 1 from mblock import event 2 import time 3 4 @event.greenflag 5 def on_greenflag(): 6 time.sleep(0.5) 7 sprite.show() 8 sprite.penup() 9 sprite.clear() 10 sprite.pencolor('#0300ff') 11 sprite.pensize(55) 12 sprite.x = -222 13 sprite.y = -67 14 sprite.pendown() 15 sprite.reset_timer() 16 sprite.glide(-130, 143, 1) 17 sprite.glide(101, 148, 1) 18 sprite.glide(218, -157, 1) 19 sprite.glide(-216, -157, 1) 20 sprite.glide(-90, 99, 1) 21 sprite.glide(76, 100, 1) 22 sprite.glide(161, -116, 1) 23 sprite.glide(-164, -102, 1) 24 sprite.glide(-67, 50, 1) 25 sprite.glide(44, 54, 1) 26 sprite.glide(130, -64, 1) 27 sprite.glide(-109, -67, 1) 28 sprite.glide(-41, 5, 1) 29 sprite.glide(56, 5, 1) 30 sprite.glide(79, -14, 1) 31 sprite.glide(-22, -35, 1)
	路徑圖	清潔面堆疊圖
		

(二) 各種擦窗法路徑之乾淨度與節能度的測試結果與省水、省電及擦窗效能討論

1. 「一筆畫法」各種擦窗法路徑之乾淨度與節能度的測試結果

表 4-1 擦窗法對長方形窗戶之乾淨度與節能度的影響

「一筆畫法」擦窗法		擦窗面積 (A) 省水	覆蓋面積 (B) 乾淨度	重疊面積 (C)	路徑長度 (l) 節能度	擦窗效能 $=\frac{\text{覆蓋面積(B)}}{\text{路徑長度(l)}}$	擦窗效能排序
1A	**直線來回法**	**266000**	**111800**	**154200**	**1900**	**58.84**	**1**
1B	對角線來回法	434280	121600	312680	3102	39.20	6
1C	邊界螺旋法	415240	133400	281840	2966	44.98	2
1D	邊界斜線切入法	479360	**144000**	335360	3424	42.06	4
1E	縱坐標來回法	392980	123200	269780	2807	43.89	3
1F	梯形螺旋法	452760	135800	316960	3234	41.99	5

****擦窗效能排序分數越少代表擦窗效能越高

2. 各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與雷達圖分析

表 4-2 各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
1A	**1**	**6**	**6**	**6**	**6**
1B	2	3	3	3	1
1C	4	4	4	4	5
1D	**6**	1	1	1	3
1E	3	5	5	5	4
1F	5	2	2	2	2

****分數越大代表擦窗向度表現越好

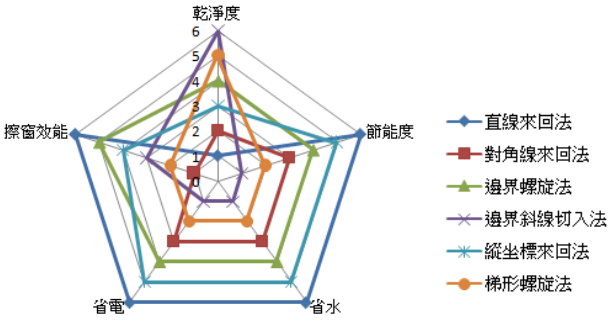


圖 4-1「一筆畫法」擦窗向度雷達圖

(三) 討論

1. 由表 4-1「一筆畫法」的六種擦窗法結果分析，擦窗**乾淨度為〝邊界斜線切入法〞最佳**，而**節能度以〝直線來回法〞最佳**，擦窗效能總排序為：

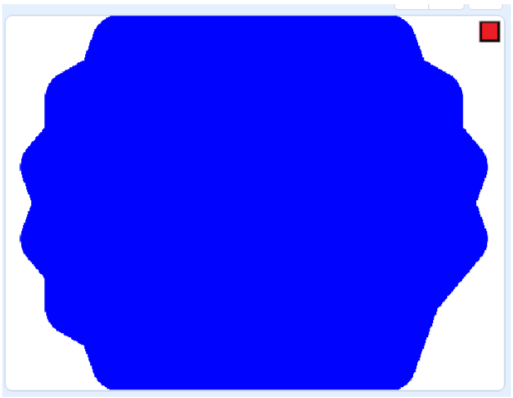
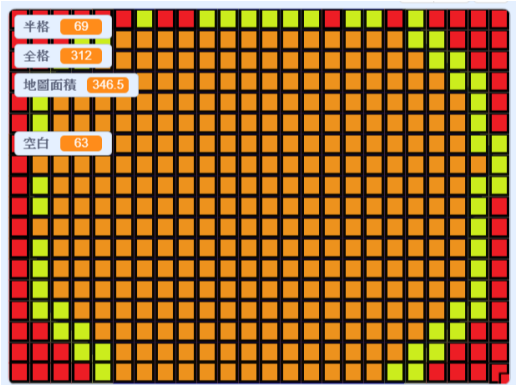
   直線來回法>邊界螺旋法>縱坐標來回法>邊界斜線切入法>梯形螺旋法>>對角線來回法

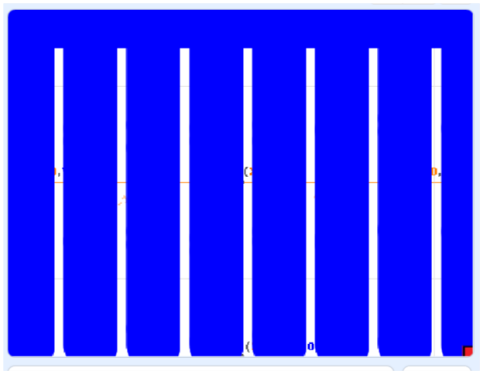
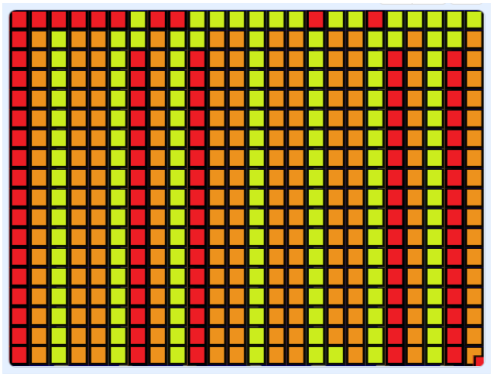
2. 由表 4-2、圖 4-1 可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，**〝直線來回法〞**各向度積分面積最大，它的重疊面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能，在日常生活中的應用，建議使用於設置較低、大面積的窗戶。

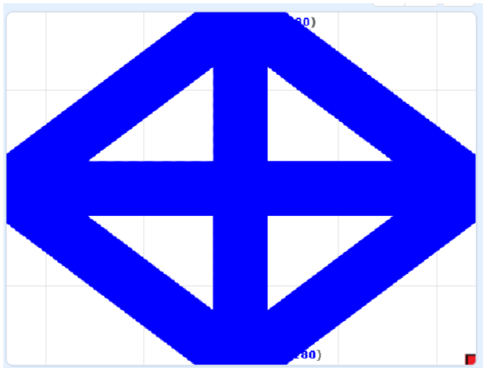
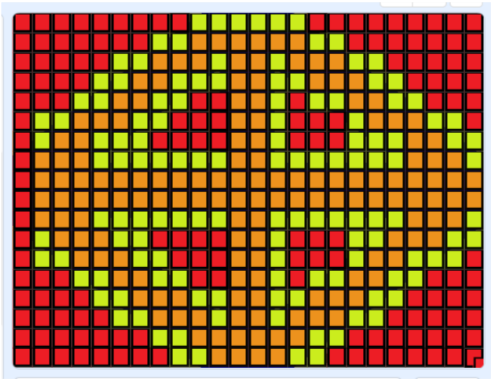
二、探討「平面坐標法」對乾淨度(覆蓋面積) 、節能程度與擦窗效能的影響。

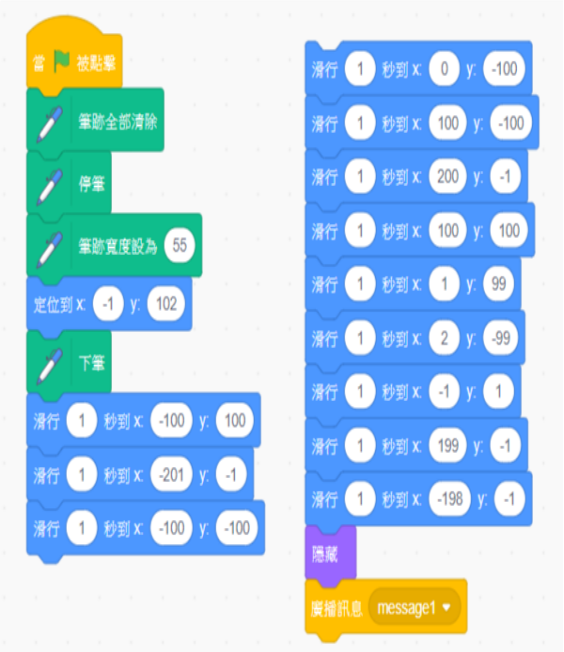
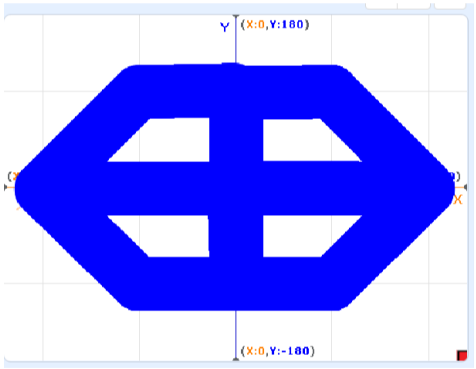
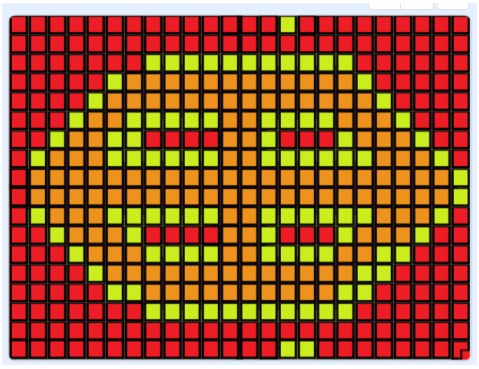
(一) 各種擦窗法之路徑編程、對應的 Python 程式語法、面積與長度計算圖

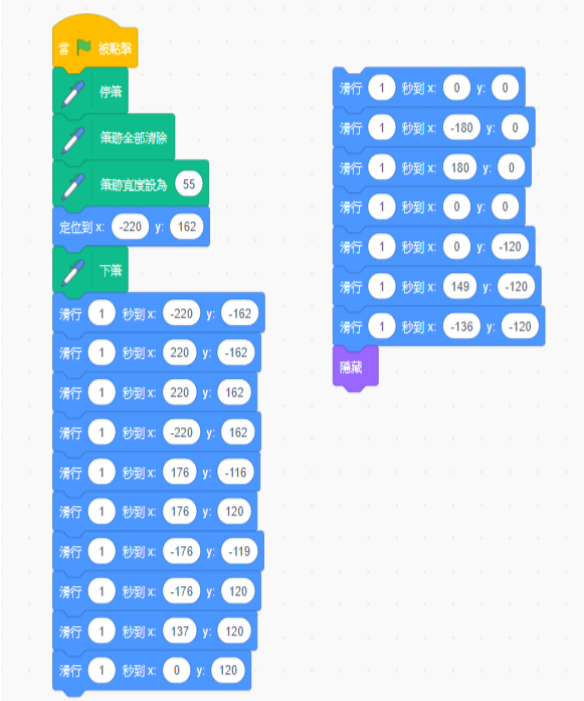
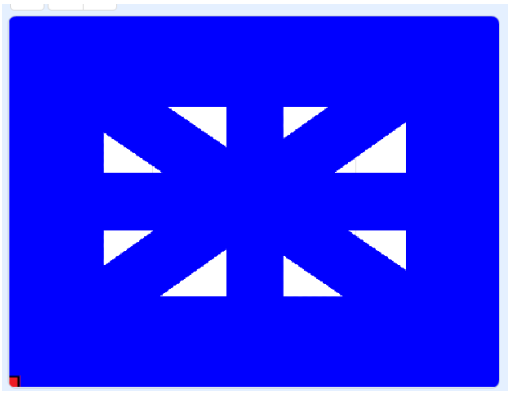
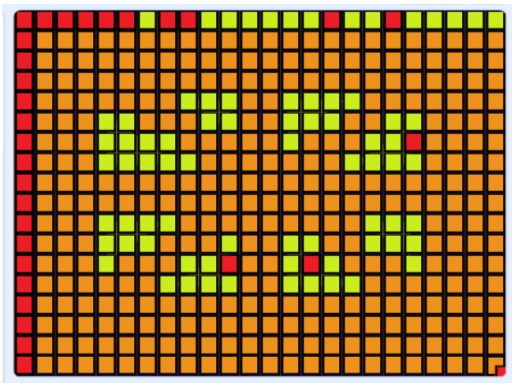
<table>
<tr><td>2A</td><td>路徑控制設計</td><td>對應的 Python 程式語法</td></tr>
<tr><td rowspan="4">六角形迴圈法</td><td rowspan="2"> 
</td><td rowspan="2">Python

```
from mblock import event

@event.greenflag
def on_greenflag():
    sprite.pencolor('#0004ff')
    sprite.pensize(55)
    sprite.penup()
    sprite.clear()
    sprite.x = 0
    sprite.y = 0
    for count2 in range(17):
        for count in range(10):
            sprite.pendown()
            sprite.forward(100)
            sprite.right(60)
            sprite.forward(100)
            sprite.right(60)
            sprite.forward(100)
            sprite.right(60)
            sprite.penup()

        sprite.right(20)


sprite.broadcast(str('message1'))

sprite.hide()
```
</td></tr>
<tr><td>路徑圖</td><td>清潔面堆疊圖</td></tr>
<tr><td>  </td><td> 
</td></tr>
</table>

2B	路徑控制設計	對應的 Python 程式語法
T 形法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.clear() 6 sprite.penup() 7 sprite.pendown() 8 sprite.pensize(55) 9 sprite.x = -219 10 sprite.y = 168 11 sprite.glide(-219, -168, sprite.get_variable('速率')) 12 sprite.glide(-219, 168, sprite.get_variable('速率')) 13 sprite.glide(-154, 168, sprite.get_variable('速率')) 14 sprite.glide(-154, -168, sprite.get_variable('速率')) 15 sprite.glide(-154, 168, sprite.get_variable('速率')) 16 sprite.glide(-89, 168, sprite.get_variable('速率')) 17 sprite.glide(-89, -168, sprite.get_variable('速率')) 18 sprite.glide(-89, 168, sprite.get_variable('速率')) 19 sprite.glide(-24, 168, sprite.get_variable('速率')) 20 sprite.glide(-24, -168, sprite.get_variable('速率')) 21 sprite.glide(-24, 168, sprite.get_variable('速率')) 22 sprite.glide(41, 168, sprite.get_variable('速率')) 23 sprite.glide(41, -168, sprite.get_variable('速率')) 24 sprite.glide(41, 168, sprite.get_variable('速率')) 25 sprite.glide(106, 168, sprite.get_variable('速率')) 26 sprite.glide(106, -168, sprite.get_variable('速率')) 27 sprite.glide(106, 168, sprite.get_variable('速率')) 28 sprite.glide(171, 168, sprite.get_variable('速率')) 29 sprite.glide(171, -168, sprite.get_variable('速率')) 30 sprite.glide(171, 168, sprite.get_variable('速率')) 31 sprite.glide(236, 168, sprite.get_variable('速率')) 32 sprite.glide(236, -168, sprite.get_variable('速率')) 33
	路徑圖	清潔面堆疊圖
		

2C	路徑控制設計	對應的 Python 程式語法
田字法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.penup() 6 sprite.clear() 7 sprite.pensize(55) 8 sprite.x = -220 9 sprite.y = 162 10 sprite.pendown() 11 sprite.glide(-220, -162, 1) 12 sprite.glide(220, -162, 1) 13 sprite.glide(220, 162, 1) 14 sprite.glide(-220, 162, 1) 15 sprite.glide(176, -116, 1) 16 sprite.glide(176, 120, 1) 17 sprite.glide(-176, -119, 1) 18 sprite.glide(-176, 120, 1) 19 sprite.glide(137, 120, 1) 20 sprite.glide(0, 120, 1) 21 sprite.glide(0, 0, 1) 22 sprite.glide(-180, 0, 1) 23 sprite.glide(180, 0, 1) 24 sprite.glide(0, 0, 1) 25 sprite.glide(0, -120, 1) 26 sprite.glide(149, -120, 1) 27 sprite.glide(-136, -120, 1) 28
	路徑圖	清潔面堆疊圖
		

2D	路徑控制設計	對應的 Python 程式語法
六角形法	 六角形法擦窗車模擬實測影片： 	Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.clear() 6 sprite.penup() 7 sprite.pensize(55) 8 sprite.x = -1 9 sprite.y = 102 10 sprite.pendown() 11 sprite.glide(-100, 100, 1) 12 sprite.glide(-201, -1, 1) 13 sprite.glide(-100, -100, 1) 14 sprite.glide(0, -100, 1) 15 sprite.glide(100, -100, 1) 16 sprite.glide(200, -1, 1) 17 sprite.glide(100, 100, 1) 18 sprite.glide(1, 99, 1) 19 sprite.glide(2, -99, 1) 20 sprite.glide(-1, 1, 1) 21 sprite.glide(199, -1, 1) 22 sprite.glide(-198, -1, 1) 23 sprite.hide() 24 25 26 27 28 sprite.broadcast(str('message1')) 29
	路徑圖	清潔面堆疊圖
		

2E	路徑控制設計	對應的 Python 程式語法
米字法		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.penup() 6 sprite.clear() 7 sprite.pensize(55) 8 sprite.x = -220 9 sprite.y = 162 10 sprite.pendown() 11 sprite.glide(-220, -162, 1) 12 sprite.glide(220, -162, 1) 13 sprite.glide(220, 162, 1) 14 sprite.glide(-220, 162, 1) 15 sprite.glide(176, -116, 1) 16 sprite.glide(176, 120, 1) 17 sprite.glide(-176, -119, 1) 18 sprite.glide(-176, 120, 1) 19 sprite.glide(137, 120, 1) 20 sprite.glide(0, 120, 1) 21 sprite.glide(0, 0, 1) 22 sprite.glide(-180, 0, 1) 23 sprite.glide(180, 0, 1) 24 sprite.glide(0, 0, 1) 25 sprite.glide(0, -120, 1) 26 sprite.glide(149, -120, 1) 27 sprite.glide(-136, -120, 1) 28
	路徑圖	清潔面堆疊圖
		

(二) 各種擦窗法路徑之乾淨度與節能度的測試結果與省水、省電及擦窗效能討論

1. 「平面坐標法」各種擦窗法路徑之乾淨度與節能度的測試結果

表 4-3 擦窗法對長方形窗戶之乾淨度與節能度的影響

「平面坐標法」擦窗法		擦窗面積 (A) 省水	覆蓋面積 (B) 乾淨度	重疊面積 (C)	路徑長度 (l) 節能度	擦窗效能 $=\frac{\text{覆蓋面積(B)}}{\text{路徑長度(l)}}$	擦窗效能排序
2A	六角形迴圈法	1428000	138600	1289400	10200	13.59	5
2B	T 形法	832300	127200	705100	5945	21.40	4
2C	田字法	344400	89200	255200	2460	36.26	2
2D	**六角形法**	**260400**	**76600**	**183800**	**1860**	**41.18**	**1**
2E	米字法	665840	**151600**	514240	4756	31.88	3

****擦窗效能排序分數越少代表擦窗效能越高

2. 各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與雷達圖分析

表 4-4 各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
2A	4	1	1	1	1
2B	3	2	2	2	2
2C	2	4	4	4	4
2D	**1**	**5**	**5**	**5**	**5**
2E	**5**	3	3	3	3

****分數越大代表擦窗向度表現越好

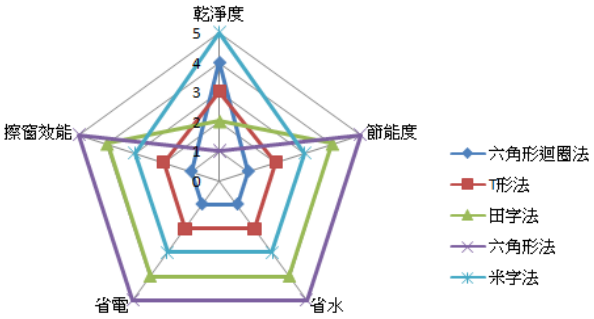


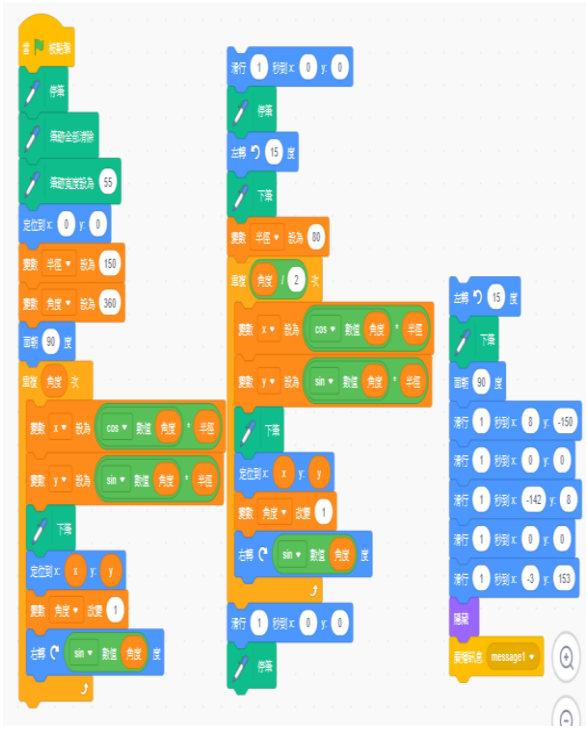
圖 4-2「平面坐標法」擦窗向度雷達圖

(三) 討論

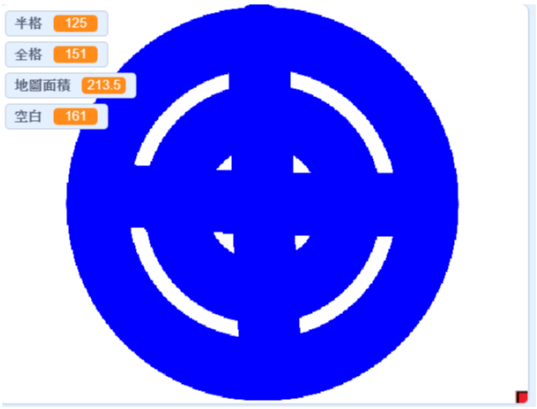
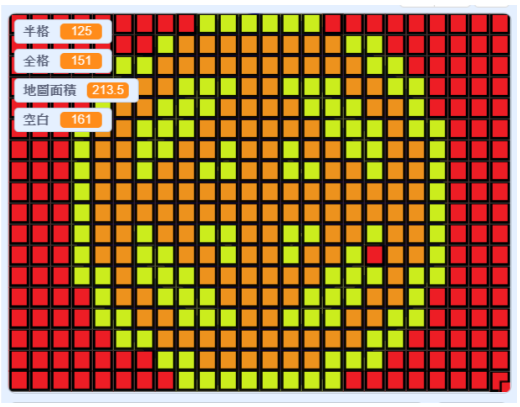
1. 由表 4-3「平面坐標法」的五種擦窗法結果分析，其中擦窗**乾淨度為〝米字法〞最佳**，而**節能度以〝六角形法〞最佳**，擦窗效能排序為：
   六角形法>田字法>米字法>T 形法>六角形迴圈法
2. 由表 4-4、圖 4-2 可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，其中**〝六角形法〞**各向度積分面積最大，它的重疊面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能。而相較「一筆畫法」而言，「平面坐標法」中的六角形法的節能度更佳，在日常生活中的應用，建議使用於設置較高的窗戶。

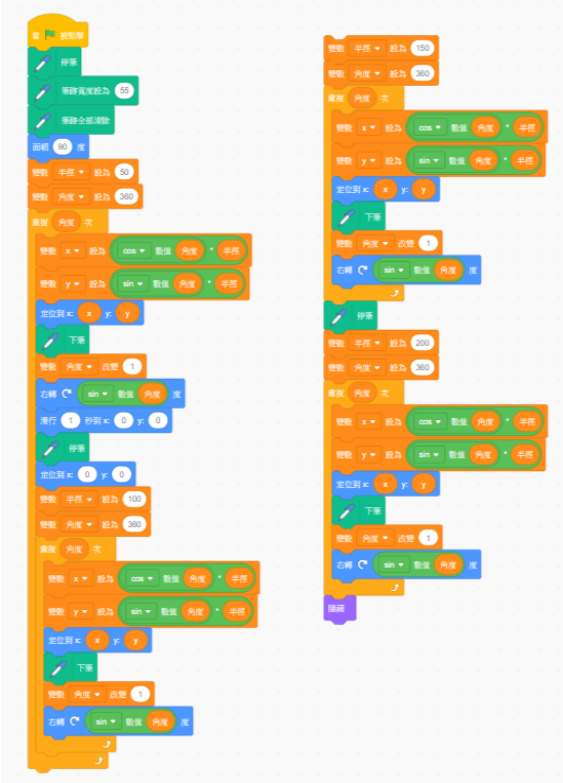
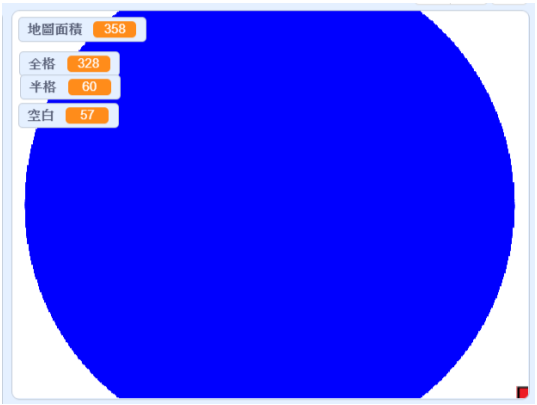
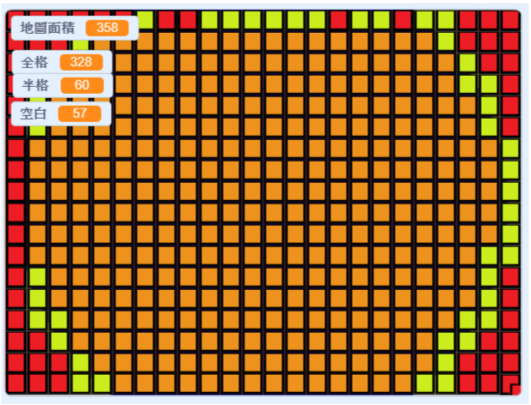
三、探討「極坐標法」對乾淨度(覆蓋面積) 、節能程度與擦窗效能的影響。

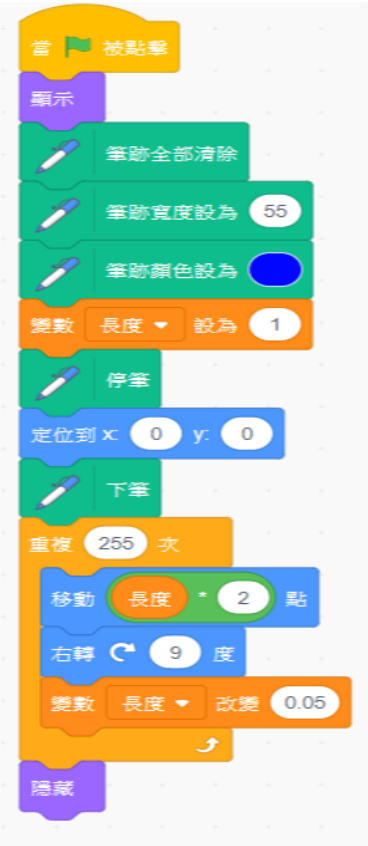
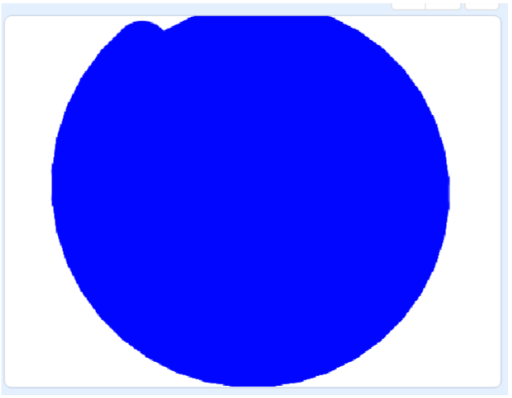
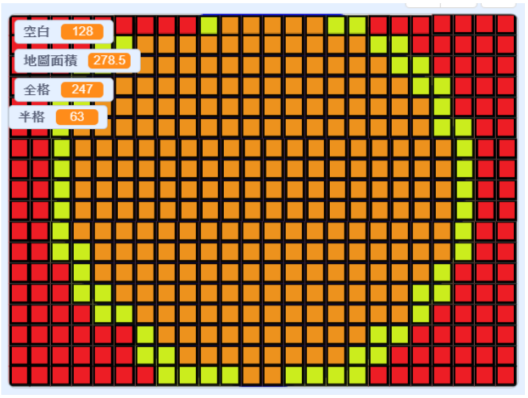
(一) 各種擦窗法之路徑編程、對應的 Python 程式語法、面積與長度計算圖

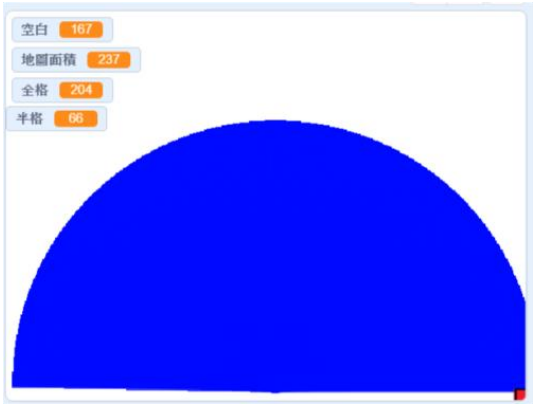
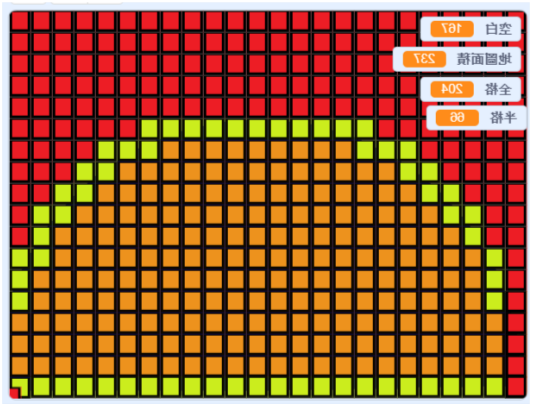
<table>
<tr><td>3A</td><td>路徑控制設計</td><td>對應的 Python 程式語法</td></tr>
<tr><td rowspan="3">箭靶(扇形 1)</td><td>  </td><td>

```python
from mblock import event
import math

_E5_8D_8A_E5_BE_91 = 0
_E8_A7_92_E5_BA_A6 = 0
y = 0

@event.greenflag
def on_greenflag():
    global _E5_8D_8A_E5_BE_91, _E8_A7_92_E5_BA_A6, y
    sprite.penup()
    sprite.clear()
    sprite.pensize(55)
    sprite.x = 0
    sprite.y = 0
    _E5_8D_8A_E5_BE_91 = 150
    _E8_A7_92_E5_BA_A6 = 360
    sprite.direction = 90
    for count in range(int(_E8_A7_92_E5_BA_A6)):
        sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91)
        y = math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91
        sprite.pendown()
        sprite.x = sprite.get_variable('x')
        sprite.y = y
        _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1
        sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi))

    sprite.glide(0, 0, 1)
    sprite.penup()
    sprite.left(15)
    sprite.pendown()
    _E5_8D_8A_E5_BE_91 = 80
    for count2 in range(int(_E8_A7_92_E5_BA_A6 / 2)):
        sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91)
        y = math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91
        sprite.pendown()
        sprite.x = sprite.get_variable('x')
        sprite.y = y
        _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1
        sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi))

    sprite.glide(0, 0, 1)
    sprite.penup()
    sprite.left(15)
    sprite.pendown()
    sprite.direction = 90
    sprite.glide(8, -150, 1)
    sprite.glide(0, 0, 1)
    sprite.glide(-142, 8, 1)
    sprite.glide(0, 0, 1)
    sprite.glide(-3, 153, 1)
    sprite.hide()
    sprite.broadcast(str('message1'))
```

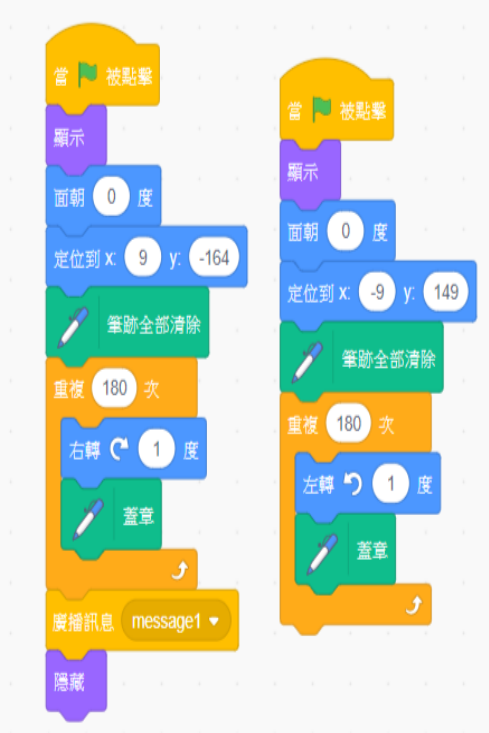
</td></tr>
<tr><td>路徑圖</td><td>清潔面堆疊圖</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

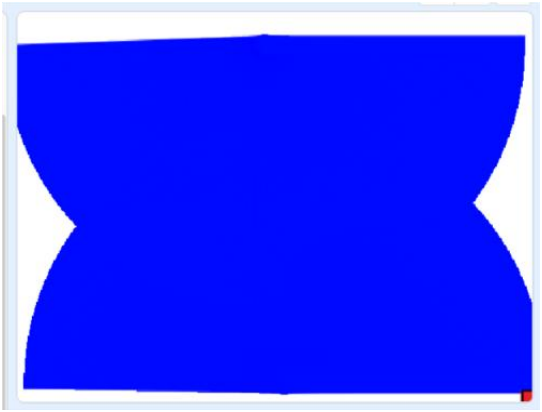
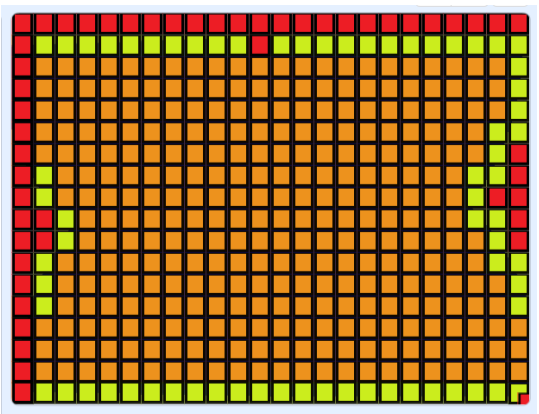
3B	路徑控制設計	對應的 Python 程式語法
同心圓(扇形 2)		Python 1 from mblock import event 2 import math 3 4 _E5_8D_8A_E5_BE_91 = 0 5 _E8_A7_92_E5_BA_A6 = 0 6 y = 0 7 8 @event.greenflag 9 def on_greenflag(): 10 global _E5_8D_8A_E5_BE_91, _E8_A7_92_E5_BA_A6, y 11 sprite.penup() 12 sprite.pensize(55) 13 sprite.clear() 14 sprite.direction = 90 15 _E5_8D_8A_E5_BE_91 = 50 16 _E8_A7_92_E5_BA_A6 = 360 17 for count in range(int(_E8_A7_92_E5_BA_A6)): 18 sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91) 19 y = math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91 20 sprite.x = sprite.get_variable('x') 21 sprite.y = y 22 sprite.pendown() 23 _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1 24 sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi)) 25 26 sprite.glide(0, 0, 1) 27 sprite.penup() 28 sprite.x = 0 29 sprite.y = 0 30 _E5_8D_8A_E5_BE_91 = 100 31 _E8_A7_92_E5_BA_A6 = 360 32 for count2 in range(int(_E8_A7_92_E5_BA_A6)): 33 sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91) 34 y = math.sin(E8 A7 92 E5 BA A6 / 180.0 * math.pi) * E5 8D 8A E5 BE 91 35 sprite.x = sprite.get_variable('x') 36 sprite.y = y 37 sprite.pendown() 38 _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1 39 sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi)) 40 41 _E5_8D_8A_E5_BE_91 = 150 42 _E8_A7_92_E5_BA_A6 = 360 43 for count3 in range(int(_E8_A7_92_E5_BA_A6)): 44 sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91) 45 y = math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91 46 sprite.x = sprite.get_variable('x') 47 sprite.y = y 48 sprite.pendown() 49 _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1 50 sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi)) 51 52 sprite.penup() 53 _E5_8D_8A_E5_BE_91 = 200 54 _E8_A7_92_E5_BA_A6 = 360 55 for count4 in range(int(_E8_A7_92_E5_BA_A6)): 56 sprite.set_variable('x', math.cos(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91) 57 y = math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi) * _E5_8D_8A_E5_BE_91 58 sprite.x = sprite.get_variable('x') 59 sprite.y = y 60 sprite.pendown() 61 _E8_A7_92_E5_BA_A6 = _E8_A7_92_E5_BA_A6 + 1 62 sprite.right(math.sin(_E8_A7_92_E5_BA_A6 / 180.0 * math.pi)) 63 64 sprite.hide() 65
	路徑圖	清潔面堆疊圖
		

3C	路徑控制設計	對應的 Python 程式語法
螺旋(扇形 3)		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.show() 6 sprite.clear() 7 sprite.pensize(55) 8 sprite.pencolor('#0006ff') 9 sprite.set_variable('長度', 1) 10 sprite.penup() 11 sprite.x = 0 12 sprite.y = 0 13 sprite.pendown() 14 for count in range(255): 15 sprite.forward(sprite.get_variable('長度') * 2) 16 sprite.right(9) 17 v = sprite.get_variable('長度') 18 sprite.set_variable('長度', v + 0.05) 19 20 sprite.hide() 21
	路徑圖	清潔面堆疊圖
		

3D	路徑控制設計	對應的 Python 程式語法
雨刷(扇形 4)		Python 1 from mblock import event 2 3 @event.greenflag 4 def on_greenflag(): 5 sprite.show() 6 sprite.direction = 0 7 sprite.x = 9 8 sprite.y = -164 9 sprite.clear() 10 for count in range(180): 11 sprite.right(1) 12 sprite.stamp() 13 14 sprite.broadcast(str('message1')) 15 sprite.hide() 16
	雨刷法擦窗車模擬實測影片: 	
	路徑圖	清潔面堆疊圖
		

3E	路徑控制設計	對應的 Python 程式語法

雙雨刷(扇形 5)		Python
		`from mblock import event`
		
		`@event.greenflag`
		`def on_greenflag():`
		`  sprite.show()`
		`  sprite.direction = 0`
		`  sprite.x = 9`
		`  sprite.y = -164`
		`  sprite.clear()`
		`  for count in range(180):`
		`  sprite.right(1)`
		`  sprite.stamp()`
		
		`  sprite.broadcast(str('message1'))`
		`  sprite.hide()`
		
		Python
		`from mblock import event`
		
		`@event.greenflag`
		`def on_greenflag():`
		`  sprite.show()`
		`  sprite.direction = 0`
		`  sprite.x = 9`
		`  sprite.y = -164`
		`  sprite.clear()`
		`  for count in range(180):`
		`  sprite.right(1)`
		`  sprite.stamp()`
		
		`  sprite.broadcast(str('message1'))`
		`  sprite.hide()`

	路徑圖	清潔面堆疊圖
		

(二) 各種擦窗法路徑之乾淨度與節能度的測試結果與省水、省電及擦窗效能討論

1. 「極坐標法」各種擦窗法路徑之乾淨度與節能度的測試結果

表 4-5 擦窗法對長方形窗戶之乾淨度與節能度的影響

「極坐標法」擦窗法		擦窗面積 (A) 省水	覆蓋面積 (B) 乾淨度	重疊面積 (C)	路徑長度 (l) 節能度	擦窗效能 $=\frac{\text{覆蓋面積(B)}}{\text{路徑長度(l)}}$	擦窗效能排序
3A	箭靶(扇形 1)	338016	85400	252616	2414.4	35.37	4
3B	同心圓(扇形 2)	460600	**143200**	317400	3290	43.53	3
3C	螺旋(扇形 3)	524440	110400	414040	3746	29.47	5
3D	**雨刷(扇形 4)**	**320736**	**94800**	**225936**	**1233.6**	**76.85**	**1**
3E	雙雨刷(扇形 5)	641472	141800	499672	2467.2	57.47	2

****擦窗效能排序分數越少代表擦窗效能越高

2. 各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與雷達圖分析

表 4-6 各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
3A	1	4	4	4	2
3B	**5**	2	3	2	3
3C	3	1	2	1	1
3D	**2**	**5**	**5**	**5**	**5**
3E	4	3	1	3	4

****分數越大代表擦窗向度表現越好

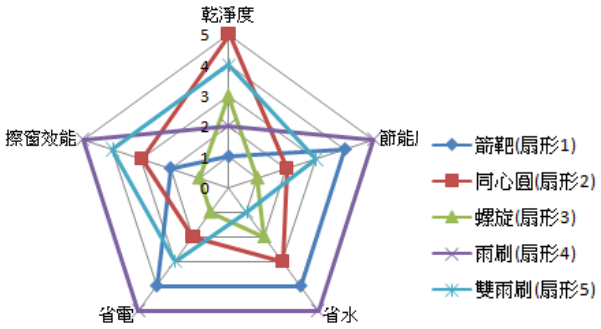


圖 4-3「極坐標法」擦窗向度雷達圖

(三) 討論

1. 由表 4-5「極坐標法」的五種擦窗法結果分析，其中擦窗**乾淨度為〝同心圓〞最佳**，而**節能度以〝雨刷〞最佳**，擦窗效能排序為：

   雨刷 > 雙雨刷 > 同心圓 > 箭靶 > 螺旋

2. 由表 4-6、圖 4-3 可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，其中**〝雨刷〞**各向度積分面積最大，因為這個擦窗路徑非常簡單，它的重疊的面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能；而相較於「一筆畫法」、「平面坐標法」而言，雨刷路徑的乾淨度(清潔面積大)最佳，在日常生活中的應用，建議使用於大面積的窗戶。

伍、結論與未來展望

我們所編程、設計三大類節能擦窗車路徑，**「一筆畫法」因路徑不重複，所以清潔路徑最長、清潔面積大，乾淨度最佳；「平面坐標法」因清潔面積小，較省水、也省電，比其他兩類擦窗路徑節能；而「極坐標法」因清潔路徑最短，較省電且擦窗效能最佳。**三大類擦窗法綜合比較結果，如表 5-1。我們將三大類擦窗法中擦窗效能最佳的路徑，進一步做五個擦窗向度的積分比較，並繪製雷達圖，如表 5-2、圖 5-1，希望未來能應用在日常生活中的玻璃清潔工作，所以我們提供建議適用範圍如下：

一、「一筆畫法」中的〝直線來回法〞乾淨度最佳，可適用於設置較低、面積大的窗戶。

二、「平面坐標法」中的〝六角形法〞最省水、也省電，可適用於設置較高的窗戶。

三、「極坐標法」中的〝雨刷〞較省電且擦窗效能最佳，可適用於較大面積的窗戶。

表 5-1 三大類節能擦窗車五個擦窗向度比較

擦窗向度	一筆畫法(直線來回法)	平面坐標法(六角形法)	極坐標法(雨刷法)
乾淨度	O	╳	Δ
節能度	╳	Δ	O
省水	Δ	O	╳
省電	╳	Δ	O
擦窗車效能	Δ	╳	O
可應用的實例	較低、面積大的窗戶	較高的窗戶	大面積窗戶

****O-佳，Δ-尚可，╳-差

表 5-2 三大類節能擦窗車五個擦窗向度積分表

擦窗向度	一筆畫法(直線來回法)	平面坐標法(六角形法)	極坐標法(雨刷法)
乾淨度	5 勝	1	3
節能度	1	3	5 勝
省水	3	5 勝	1
省電	1	3	5 勝
擦窗效能	3	1	5 勝

****5-佳，3-尚可，1-差---分數越高，擦窗向度表現越好

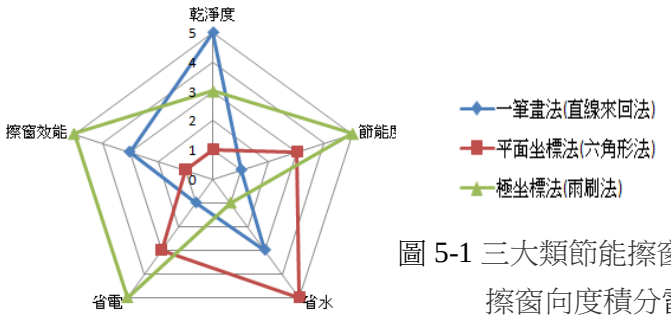


圖 5-1 三大類節能擦窗車五個擦窗向度積分雷達圖

未來的研究方向：我們希望再編程更多的擦窗路徑，並增加髒污感測、洗窗乾淨程度及洗窗原理的改良…等，提供人類應用於各種擦窗情境的需求，實際做出最速、最節能的擦窗車，讓機器充分取代人工，還能呈現省水、省電的功能，不管在教室、家裡、大樓〝拐彎抹角〞時，達到最佳經濟效益。

陸、參考資料及其他

一、吳亞靜(2008)家庭自製洗窗機 SUPER!・中華民國第四十八屆中小學科學展覽會作品說明書・2019 年 10 月 15 日，取自 https://activity.ntsec.gov.tw/activity/race-1/48/elementary/080816.pdf

二、朱雁丞、柯昱任、謝佳峻(2008)神來之手---自動繪圖機設計與探討・中華民國第 57 屆中小學科學展覽會作品說明書・2019 年 10 月 15 日，取自 https://activity.ntsec.gov.tw/activity/race-1/57/pdf/052508.pdf

三、【老實開箱】不只可以擦玻璃♥可高空吸附的全自動擦窗機器人(BOBO 玻娃三代)・2019 年 10 月 26 日，取自 https://cvlifesty.com/life/%E3%80%90%E8%80%81%E5%AF%A6%E9%96%8B%E7%AE%B1%E3%80%91%E5%85%A8%E8%87%AA%E5%8B%95%E6%93%A6%E7%AA%97%E6%A9%9F%E5%99%A8%E4%BA%BAbobo%E7%8E%BB%E5%A8%83%E4%B8%89%E4%BB%A3/

四、台灣科學教育館(2018) 邏輯運算思維・科學研習・2019 年 10 月 26 日，取自 https://activity.ntsec.gov.tw/activity/ssm/57_5/images/publication.pdf

五、深度優先搜尋法・維基百科・2019 年 10 月 26 日，取自 https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2

六、廣度優先搜尋法・2019 年 10 月 26 日，取自 http://simonsays-tw.com/web/DFS-BFS/BreadthFirstSearch.html

七、一筆畫法・維基百科・2019 年 12 月 29 日，取自 https://zh.wikipedia.org/wiki/%E4%B8%80%E7%AC%94%E7%94%BB%E9%97%AE%E9%A2%98#%E6%9C%89%E5%90%91%E5%9B%BE%E7%9A%84%E4%B8%80%E7%AC%94%E7%94%BB

八、平面坐標法・維基百科・2019 年 12 月 29 日，取自 https://zh.wikipedia.org/wiki/%E5%9D%90%E6%A8%99%E7%B3%BB

九、極坐標法・維基百科・2019 年 12 月 28 日，取自 https://zh.wikipedia.org/wiki/%E6%9E%81%E5%9D%90%E6%A0%87%E7%B3%BB

十、Scratch3.0・取自 https://scratch.mit.edu/projects/370527304/editor

十一、mblock・取自 https://ide.mblock.cc/#/

【評語】032807

由真實問題引發實驗動機，實作一個擦窗機器人，對於各種可能路徑之效能進行詳細探討，具相當創意與實驗精神。模擬機器人在地面上實驗，而非實際之窗戶，與實際狀況會有落差，實用性亦較無法驗證，若改以掃地機器人或割草機為測試標的可能更為實際。另外，作品在實驗變因控制的嚴謹程度還可以再精進，例如，各式路徑有不同參數，同一類型不同參數之路徑在省水度及乾淨度上亦可能有不同表現與權衡，可以進行更多實驗與討論。

作品海報

壹、研究動機與摘要

每到了學校的打掃時間，我們常看到同學擦不到高處的窗戶或不想擦窗戶，因而聯想到從事清潔玻璃的工人，因PM2.5的夢魘未改善，又得搭乘吊車、冒生命危險擦洗大樓窗戶，再加上曾在網路上看到自動擦窗車，所以我們想設計節能擦窗車來改善這些問題。我們利用資訊課學到Scratch3.0，利用它編程、運算擦窗面積及路徑長度，模擬設計了三大類擦窗路徑，並利用mblock轉換成Python程式語法、Excel作數據分析。

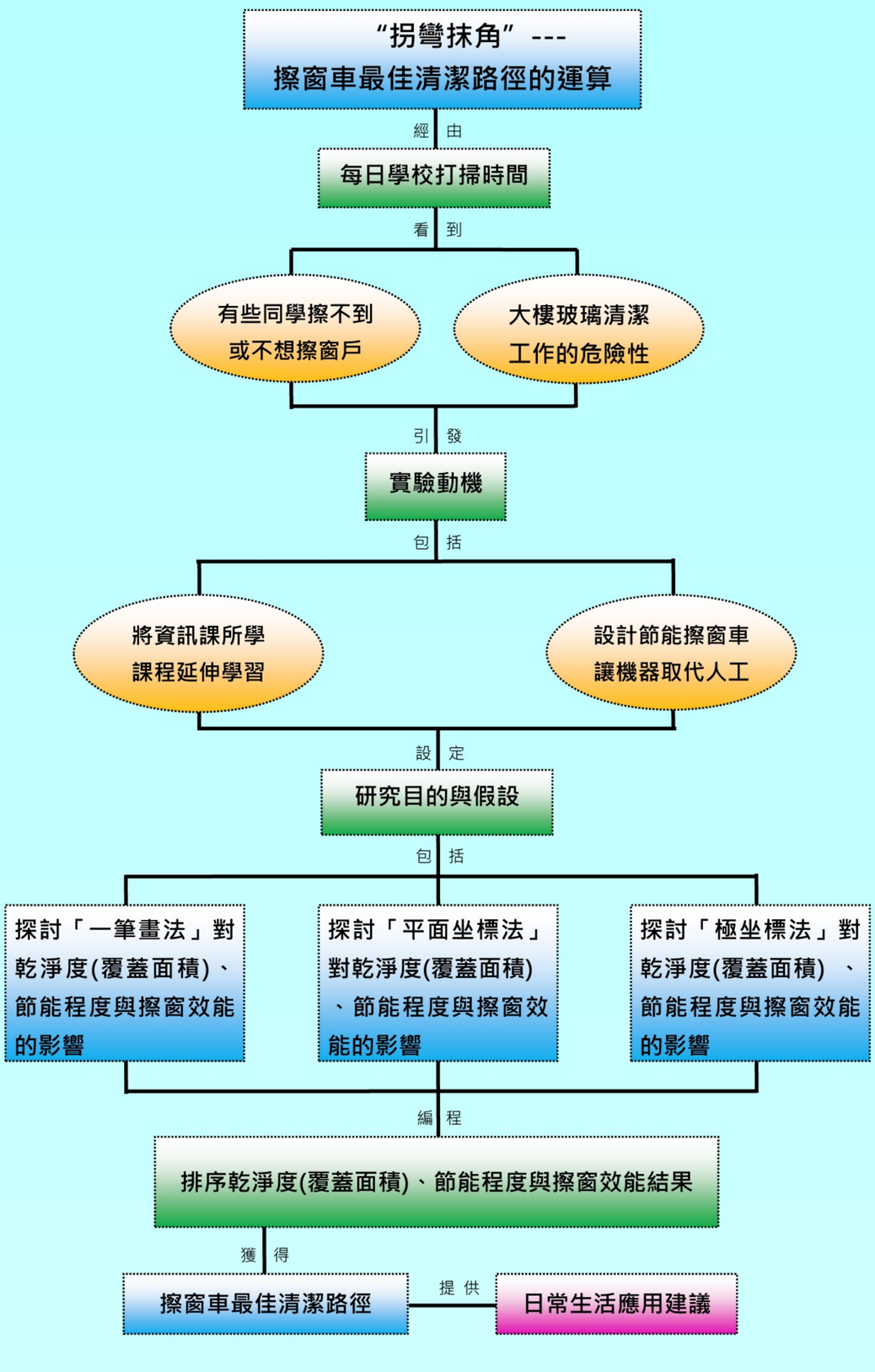
研究結果為：「直線來回法」的乾淨度最佳；「六角形法」最省水、也省電，節能效果最佳；「雨刷法」最省電，擦窗效能最佳。未來，我們希望能應用在日常生活中，代替人類進行玻璃清潔的工作，無論高、低或大面積的窗戶，都能讓擦窗車在“拐彎抹角”時，達到最佳的清潔效能。

貳、研究目的

一、探討「一筆畫法」對乾淨度(覆蓋面積) 、節能程度與擦窗效能的影響
二、探討「平面坐標法」對乾淨度(覆蓋面積) 、節能程度與擦窗效能的影響
三、探討「極坐標法」對乾淨度(覆蓋面積) 、節能程度與擦窗效能的影響

參、研究方法

一、研究流程



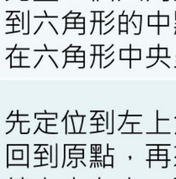
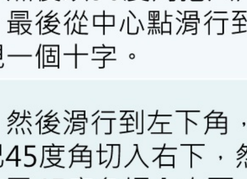
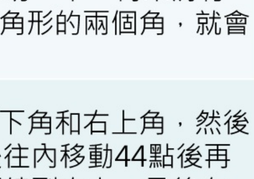
二、擦窗路徑

演算法	擦窗法	說 明
一筆畫法	1A 直線來回法	定位到左上角，移動到左下角。往右轉90度，再往上轉90度。
	1B 對角線來回法	定位到左上角，每次轉約25度上下來回移動。
	1C 邊界螺旋法	沿邊界碰壁時轉90度且起始邊界坐標減少車子的寬度，轉到邊界=或<1/2邊界為止。
	1D 邊界斜線切入法	先定位到左上角，沿邊界碰壁時轉90度，重複來回三次。
	1E 縱坐標來回法	先定位到右下角，再移動到左上角，再斜線切入往右下約75度角，重複執行3次。最後再移動到右上角。
	1F 梯形螺旋法	先定位到右下角，斜線切入往右上約75度角，再往右移動50點，斜線切入往右下約75度角，再往左移動60點。再重複兩次，每次移動值減10點。
平面坐標法	2A 六角形迴圈法	定位到中心然後移動100點，在右轉60度，重複三次呈現梯形再以每轉20度，重複進行17次。
	2B T形法	運用T字型的方式，X軸移動65點後垂直往下，再回到往下移動的那點，最後以重覆執行來擴大面積。
	2C 田字法	先定位到中心點上方，再以135度角移動到點左方，再以同樣方式移動到，再來回到原點，最後在中間畫出一個十字來擴大面積。
	2D 六角形法	先畫一個六角形，然後以90度角把六角形切一半，再來滑行到六角形的中點，最後從中心點滑行到六角形的兩個角，就會在六角形中央呈現一個十字。
	2E 米字法	先定位到左上角，然後滑行到左下角，右下角和右上角，然後回到原點，再來已45度角切入右下，然後往內移動44點後再繞上去左上，再來已45度角切入右下，再繞到右上，最後在中間畫一個王字來擴大面積。
極坐標法	3A 箭靶(扇形1)	畫完兩個同心圓再轉30度，接著在兩個同心圓中畫十字。
	3B 同心圓(扇形2)	由中心繞出同心圓，直到車子上下碰觸邊界為止。
	3C 螺旋(扇形3)	由固定一點繞出圓形螺旋，直到車子 < 或=邊界為止。
	3D 雨刷(扇形4)	由一支雨刷刷過固定180度面積，直到車子碰觸邊界為止。
	3E 雙雨刷(扇形5)	兩支雨刷同時刷過固定180度面積，直到車子 < 或=邊界為止。

三、刷頭設計與擦窗路徑的各項數據來源

1.本研究擦窗車的刷頭設計分為兩種，分別為一般刷頭(圓形)與扇形刷頭(棒形)，一般刷頭適用於一筆畫法、平面坐標法的擦窗路徑；而扇形刷頭則適用於極坐標法中的雨刷路徑。

一般刷頭	刷頭程式	Python	圖 示	刷頭面積

扇形刷頭	刷頭程式	Python	圖 示	刷頭面積
				

2. 擦窗路徑的各項數據來源

項 目	說 明	項 目	說 明
長形窗戶面積	24*18 =432格	扇形刷頭 ⇒ 擦窗面積(A)	路徑長度(l)*5600/20
一般刷頭面積	20*20*7(2800)	覆蓋面積(B)	覆蓋格數*400
扇形刷頭面積	20*20*14(5600)	重疊面積(C)	擦窗面積(A) - 覆蓋面積(B)
路徑長度(l)	利用坐標定位估算路徑總長	擦窗效能	覆蓋面積(B)/路徑長度(l)
一般刷頭 ⇒ 擦窗面積(A)	路徑長度(l)*2800/20		

肆、結果與討論

一、一筆畫法

▲ 各種擦窗法之路徑編程、對應的Python程式語法、面積與長度計算圖

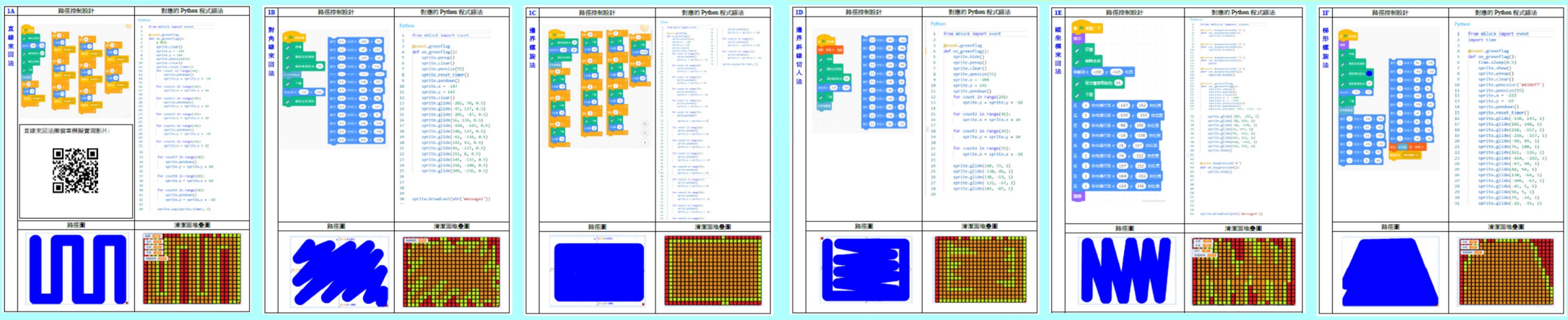


表 4-1擦窗法對長方形窗戶之乾淨度與節能度的影響

「一筆畫法」擦窗法		擦窗面積(A) 省水	覆蓋面積(B) 乾淨度	重疊面積(C)	路徑長度(l) 節能度	擦窗效能=覆蓋面積(B)/路徑長度(l)	擦窗效能排序
1A	直線來回法	266000	111800	154200	1900	58.84	1
1B	對角線來回法	434280	121600	312680	3102	39.20	6
1C	邊界螺旋法	415240	133400	281840	2966	44.98	2
1D	邊界斜線切入法	479360	**144000**	335360	3424	42.06	4
1E	縱坐標來回法	392980	123200	269780	2807	43.89	3
1F	梯形螺旋法	452760	135800	316960	3234	41.99	5

表 4-2各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
1A	1	6	6	6	6
1B	2	3	3	3	1
1C	4	4	4	4	5
1D	6	1	1	1	3
1E	3	5	5	5	4
1F	5	2	2	2	2

1. 由表4-1「一筆畫法」的六種擦窗法結果分析，擦窗**乾淨度為“邊界斜線切入法”最佳**，而**節能度以“直線來回法”最佳**，擦窗效能總排序為：
   直線來回法>邊界螺旋法>縱坐標來回法>邊界斜線切入法>梯形螺旋法>>對角線來回法
2. 由表4-2、圖4-1可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，〝**直線來回法**〞各向度積分面積最大，它的重疊面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能，在日常生活中的應用，建議使用於設置較低、大面積的窗戶。

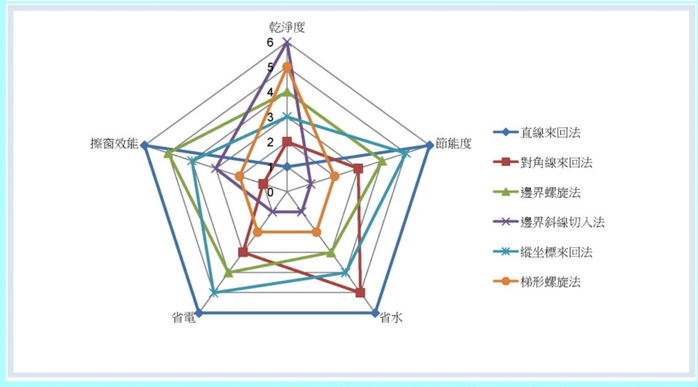


圖4-1「一筆畫法」-擦窗向度雷達圖

二、平面坐標法

▲ 各種擦窗法之路徑編程、對應的Python程式語法、面積與長度計算圖

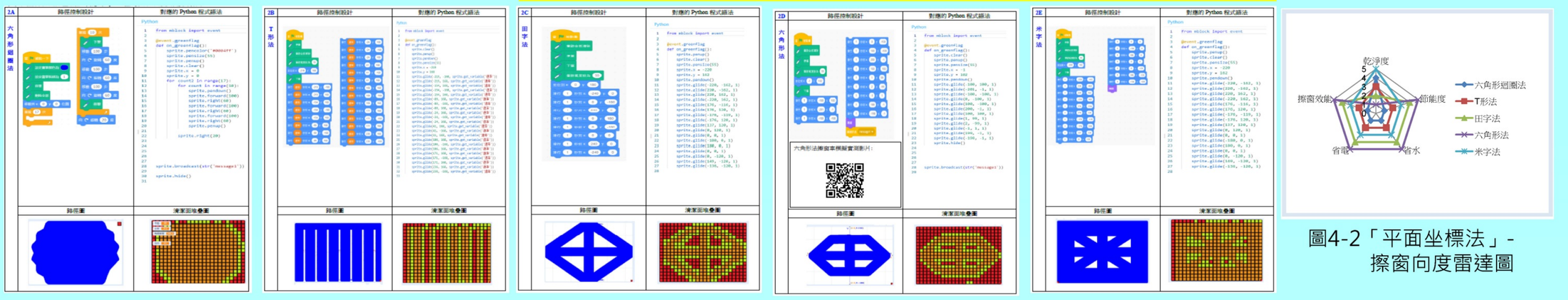


圖4-2「平面坐標法」-擦窗向度雷達圖

表 4-3擦窗法對長方形窗戶之乾淨度與節能度的影響

「平面坐標法」擦窗法		擦窗面積(A) 省水	覆蓋面積(B) 乾淨度	重疊面積(C)	路徑長度(l) 節能度	擦窗效能=覆蓋面積(B)/路徑長度(l)	擦窗效能排序
2A	六角形迴圈法	1428000	138600	1289400	10200	13.59	5
2B	T形法	832300	127200	705100	5945	21.40	4
2C	田字法	344400	89200	255200	2460	36.26	2
2D	六角形法	260400	76600	183800	1860	41.18	1
2E	米字法	665840	**151600**	514240	4756	31.88	3

表 4-4各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
2A	4	1	1	1	1
2B	3	2	2	2	2
2C	2	4	4	4	4
2D	1	5	5	5	5
2E	5	3	3	3	3

1. 由表4-3「平面坐標法」的五種擦窗法結果分析，其中擦窗**乾淨度為“米字法”最佳**，而**節能度以“六角形法”最佳**，擦窗效能排序為：**六角形法>田字法>米字法>T形法>六角形迴圈法**
2. 由表4-4、圖4-2可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，其中**“六角形法”**各向度積分面積最大，它的重疊面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能。而相較「一筆畫法」而言，「平面坐標法」中的六角形法的節能度更佳，在日常生活中的應用，建議使用於設置較高的窗戶。

三、極坐標法

▲ 各種擦窗法之路徑編程、對應的Python程式語法、面積與長度計算圖

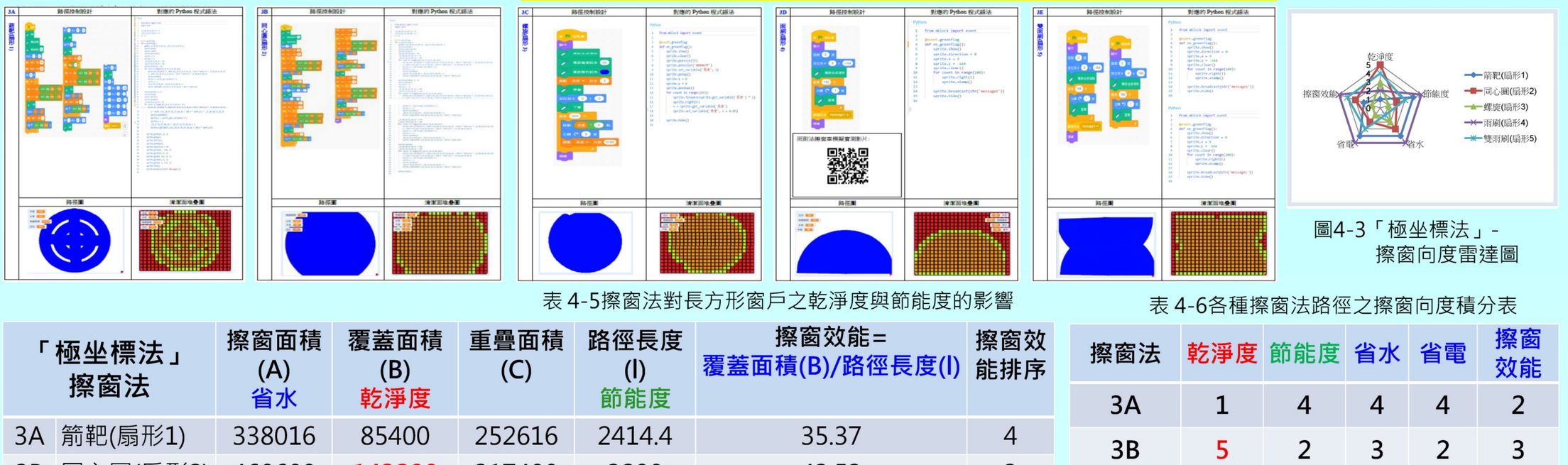


圖4-3「極坐標法」-擦窗向度雷達圖

表 4-5擦窗法對長方形窗戶之乾淨度與節能度的影響

「極坐標法」擦窗法		擦窗面積(A) 省水	覆蓋面積(B) 乾淨度	重疊面積(C)	路徑長度(l) 節能度	擦窗效能=覆蓋面積(B)/路徑長度(l)	擦窗效能排序
3A	箭靶(扇形1)	338016	85400	252616	2414.4	35.37	4
3B	同心圓(扇形2)	460600	**143200**	317400	3290	43.53	3
3C	螺旋(扇形3)	524440	110400	414040	3746	29.47	5
3D	雨刷(扇形4)	320736	94800	225936	1233.6	76.85	1
3E	雙雨刷(扇形5)	641472	141800	499672	2467.2	57.47	2

表 4-6各種擦窗法路徑之擦窗向度積分表

擦窗法	乾淨度	節能度	省水	省電	擦窗效能
3A	1	4	4	4	2
3B	5	2	3	2	3
3C	3	1	2	1	1
3D	2	5	5	5	5
3E	4	3	1	3	4

1. 由表4-5「極坐標法」的五種擦窗法結果分析，其中擦窗**乾淨度為“同心圓”最佳**，而**節能度以“雨刷”最佳**，擦窗效能排序為：
   雨刷 > 雙雨刷 > 同心圓 > 箭靶 > 螺旋
2. 由表4-6、圖4-3可得知各種擦窗法路徑之乾淨度、節能度、省水、省電及擦窗效能積分比較與分析，其中**“雨刷”**各向度積分面積最大，因為這個擦窗路徑非常簡單，它的重疊的面積較小且路徑長度較短，**擦窗效能最佳**，可同時達到省水、省電的功能；而相較於「一筆畫法」、「平面坐標法」而言，雨刷路徑的乾淨度(清潔面積大)最佳，在日常生活中的應用，建議使用於大面積的窗戶。

伍、結論與未來展望

我們所編程、設計三大類節能擦窗車路徑，綜合比較結果，如表5-1。「一筆畫法」因路徑不重複，所以清潔路徑最長、清潔面積大，乾淨度最佳；「平面坐標法」因清潔面積小，較省水、也省電，比其他兩類擦窗路徑節能；而「極坐標法」因清潔路徑最短最省電，且擦窗效能最佳。我們將三大類擦窗法中擦窗效能最佳的路徑，進一步做五個擦窗向度的積分比較，並繪製雷達圖，如表5-2、圖5-1，希望未來能應用在日常生活中的玻璃清潔工作，所以我們提供建議適用範圍如下：

一、「一筆畫法」中的“直線來回法”乾淨度最佳，適用於較低、大面積的窗戶。
二、「平面坐標法」中的“六角形法”最省水、也省電，適用於設置較高的窗戶。
三、「極坐標法」中的“雨刷”較省電且擦窗效能最佳，適用於較大面積的窗戶。

表5-1三大類節能擦窗車表五個擦窗向度積分表

擦窗法	直線來回法	六角形法	雨刷
乾淨度	5 勝	1	3
節能度	1	3	5 勝
省水	3	5 勝	1
省電	1	3	5 勝
擦窗車效能	3	1	5 勝

****5-佳，3-尚可，1-差---分數越高，擦窗向度表現越好

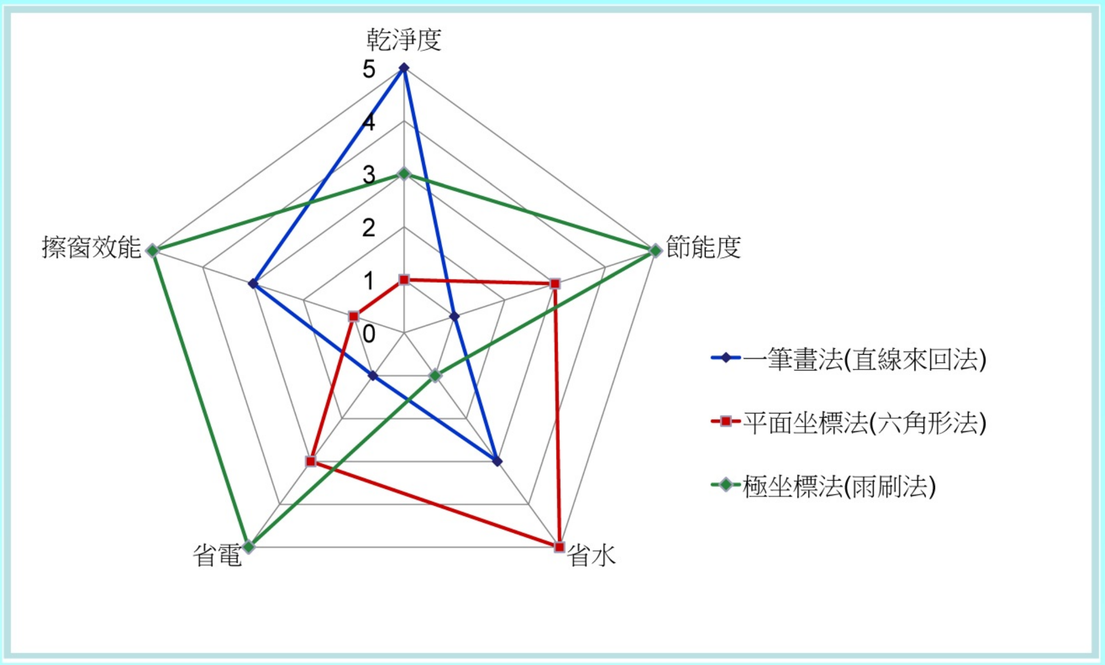


圖5-1三大類節能擦窗車五個擦窗向度積分雷達圖

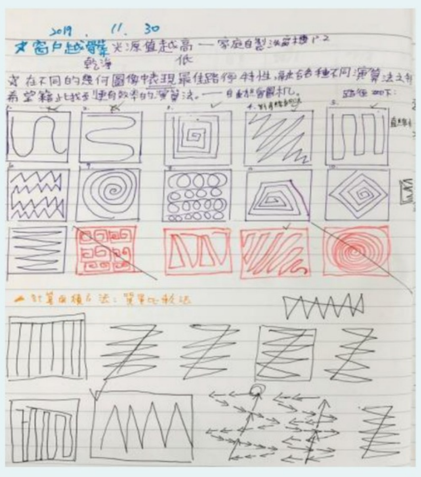
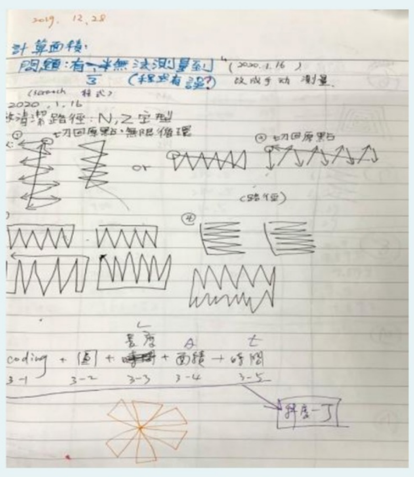
表5-2三大類節能擦窗車表五個擦窗向度比較

擦窗法	直線來回法	六角形法	雨刷
乾淨度	○	×	Δ
節能度	×	Δ	○
省水	Δ	○	×
省電	×	Δ	○
擦窗車效能	Δ	×	○
建議適用窗戶種類	較低、大面積窗戶	較高的窗戶	大面積窗戶

****○-佳，Δ-尚可，×-差

未來的研究方向：我們希望再編程更多的擦窗路徑，並增加髒污感測、洗窗乾淨程度及洗窗原理的改良...等，提供人類應用於各種擦窗情境的需求，實際做出最速、最節能的擦窗車，讓機器充分取代人工，還能呈現省水、省電的功能，不管在教室、家裡、大樓“拐彎抹角”時，達到最佳經濟效益。

陸、研究發想手繪圖

人類可能的擦窗路徑手繪圖	選擇可編程的路徑	討論擦窗行為與邏輯運算思維關係	面積運算方法及說明書內容討論
			

柒、參考文獻

一、吳亞靜(2008)家庭自製洗窗機SUPER!·中華民國第四十八屆中小學科學展覽會作品說明書·2019年10月15日，取自 https://activity.ntsec.gov.tw/activity/race-1/48/elementary/080816.pdf
二、朱雁丞、柯昱任、謝佳峻(2008)神來之手---自動繪圖機設計與探討·中華民國第57屆中小學科學展覽會作品說明書·2019年10月15日，取自 https://activity.ntsec.gov.tw/activity/race-1/57/pdf/052508.pdf
三、【老實開箱】不只可以擦玻璃♥可高空吸附的全自動擦窗機器人(BOBO玻娃三代)·2019年10月26日，取自 https://cvlifesty.com/life/%E3%80%90%E8%80%81%E5%AF%A6%E9%96%8B%E7%AE%B1%E3%80%91%E5%85%A8%E8%87%AA%E5%8B%95%E6%93%A6%E7%AA%97%E6%A9%9F%E5%99%A8%E4%BA%BAbobo%E7%8E%BB%E5%A8%83%E4%B8%89%E4%BB%A3/
四、台灣科學教育館(2018) 邏輯運算思維·科學研習·2019年10月26日，取自 https://activity.ntsec.gov.tw/activity/ssm/57_5/images/publication.pdf
五、深度優先搜尋法·維基百科·2019年10月26日，取自 https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2
六、廣度優先搜尋法·2019年10月26日，取自 http://simonsays-tw.com/web/DFS-BFS/BreadthFirstSearch.html
七、一筆畫法·維基百科·2019年12月29日，取自 https://zh.wikipedia.org/wiki/%E4%B8%80%E7%AC%94%E7%94%BB%E9%97%AE%E9%A2%98#%E6%9C%89%E5%90%91%E5%9B%BE%E7%9A%84%E4%B8%80%E7%AC%94%E7%94%BB
八、平面坐標法·維基百科·2019年12月29日，取自 https://zh.wikipedia.org/wiki/%E5%9D%90%E6%A8%99%E7%B3%BB
九、極坐標法·維基百科·2019年12月28日，取自 https://zh.wikipedia.org/wiki/%E6%9E%81%E5%9D%90%E6%A0%87%E7%B3%BB
十、Scratch3.0·取自 https://scratch.mit.edu/projects/370527304/editor
十一、mblock·取自 https://ide.mblock.cc/#/