新竹市第 41 屆中小學科學展覽會

作品說明書

科　　別：生活與應用科學科(一)(機電與資訊)

組　　別：國中組

作品名稱：Don’t Touch Me! 以微觀的粒子碰撞視角探討流行疾病傳染模型

關 鍵 詞：COVID-19，電腦模擬，傳染模型

編　　號：

摘要

本實驗是以「有限群體在有限空間中，透過彼此接觸傳遞病毒」為基礎，以 Python 撰寫以粒子碰撞模擬的程式，從微觀的角度以 SEIR 模型的原理，於傳播機率、感染期長短、個體大小(社交距離)三個面相，探討流行疾病傳播的機制的差異。除了能得到相似於宏觀角度常微分方程的結果之外，也可提供宏觀所看不到的細節。

若將都市的空間切分成許多彼此不相往來的小區域，在較嚴苛的條件下，即使該區域有染疫者，也不一定會將病毒擴散至整個群體。其次即使是疫情在群體中擴散，也有一定比例的人不受到感染。此外也發，現即使相同條件下的傳播狀況也存在著明顯的差異，這差異大致歸類為兩種類型，一個是周期短、峰值高的猛快型，另外一種是週期很長、緩慢進展、峰值低的緩和型，每一組的數據基本上都可以用這兩種子類型組合而成。

壹、研究動機

近年來新冠病毒正在世界各地流行，對人類社會造成很的影響，由於病毒致命且尚無有效疫苗與藥物，因此若是能了解傳染的嚴重性與疾病的 lifecycle time，對於防疫的資源安排至關重要。目前相關研究大多以宏觀角度用微分方程作為模擬的依據，由於常微分方程式並不在國中數學範圍內，因此並沒有在科展國中組看到做此研究的報告。前一段時間學校鼓勵學生參加名為 Code-Combat 撰寫程式的遊戲競賽，期望透過對戰遊戲機制以習得 Python。競賽的勝負是以遊戲中生命值消長作為依據，這和流行病傳染模型的參數變化很類似，於是我們就想試試，是否能以微觀的角度佐以電腦模擬，探討新冠病毒的傳播狀況。

貳、研究目的

1.用微觀角度，透過類似遊戲生命值消長的機制，探討 COVID-19 等流行病的傳染病模型。

2.運用自製的環境程式，從微觀角度觀察整個傳染的過程，並與宏觀結果和真實現況作分析比較。

3.透過微觀的模擬，嘗試將宏觀方法無法探討的因素，像是社交距離等空間變因，加入本研究中。

參、研究設備與器材

電腦、紙、筆

肆、研究過程

一、文獻探討

一般來說流行病的傳染模型研究，大部分的學者都是用常微分方程式來探討結果的，透過S、E、I、R等狀態，藉由傳染係數的相關參數設定，探討疾病傳播的速率，預測高峰期發生時間、確診比例等重要數據，以供醫護相關人員安排防疫措施與醫療資源。

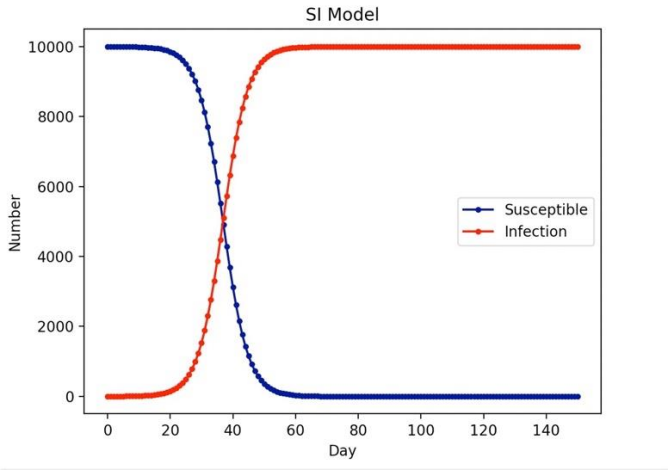
常見的傳染病模型按照具體的傳染病的特點可分為 SI、SIS、SIR、SIRS、SEIR 模型五種不等。其中 SEIR 的現實含義如下：

- **S (Susceptible)易感者**：
  指缺乏免疫能力健康人，與感染者接觸後容易受到感染。
- **E (Exposed)，暴露者**
  可用於存在潛伏期的傳染病，沒有發病但具有傳染性
- **I (Infectious)，患病者：**
  指有傳染性的病人，可以傳播給 S，將其變為 E 或 I。新冠肺炎的 Omicron 型 只會在前三分之一的時間具有傳染性
- **R (Recovered)，康復者：**
  指病癒後具有免疫力的人，如是終身免疫性傳染病，則不可被重新變為 S 、E 或 I ，如果免疫期有限，就可以重新變為 S 類，進而被感染。

以下介紹下列四種模型：

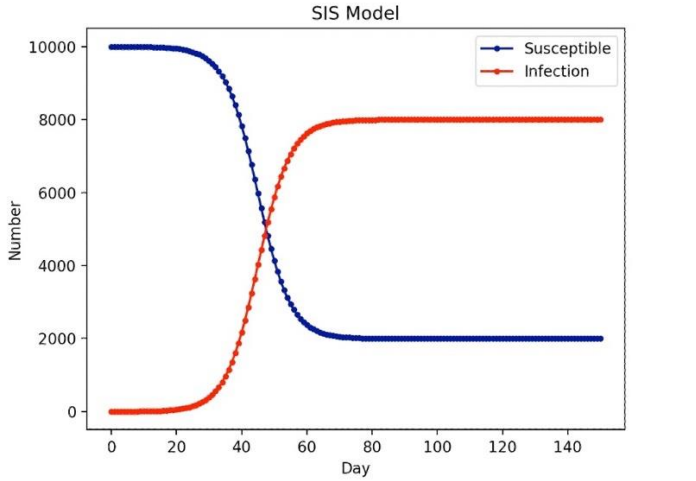
(一)、SI 模型：

此模型只考慮了易感者(Susceptible)和感染者(Infection)，並且感染者不能夠恢復。易感者和感染者人數變化如下圖所示：



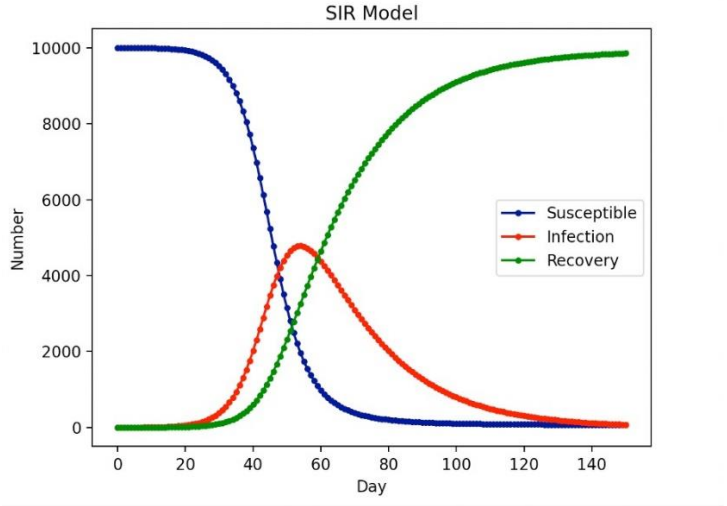
(二)、SIS 模型：

傳染病無免疫性，病人在治癒成健康的人後仍可能再次感染。易感者和感染者人數變化如下圖所示：



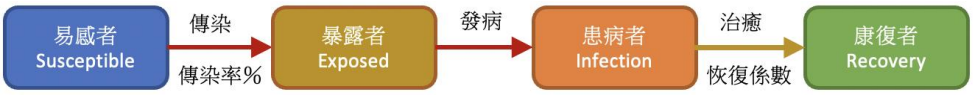
(三)、SIR 模型：

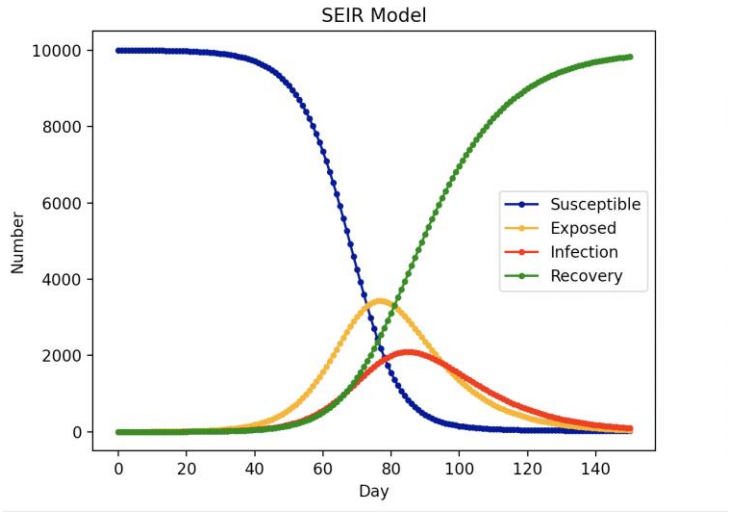
傳染病具有免疫性，再受感染一段時間後病人治癒獲得終身免疫力。易感者和感染者人數變化如下圖所示：



(四)、SEIR 模型：

受感染後會進入所謂的"潛伏期"，不會發病但具有傳染性，一段時間後進入發病期





二、環境變因

在本實驗以「**有限空間中，固定數量的球體彼此接觸**」模擬真實世界的傳染病傳播行為，因此必須考慮**實驗空間**中相關設定與**真實世界**的關聯性，以下就四個面向進行討論：

1. 個體與行為：

本實驗以球體代表人，人步行的距離約與自身的寬度相當，為簡化實驗參數因此不考慮個體之間的差異，在此以 0.5m 代表一個人的寬度，每次步行的距離是 0.5m。依照上述條件，以**實驗空間 10 pixels 等同於實體空間 0.5m** 計算，實驗裡的球體半徑 r=5 pixels，每顆球每次移動 10 pixels(與球體直徑相同)。

2. 空間大小：

空間相關參數，若是以中學生自身所處的環境考慮，依「國民小學及國民中學設施設備基準」(修正日期：民國 108 年 7 月 24 日，發文字號：臺教授國部字第 1080056287B 號 令) [13]，其中有關國民中學校地面積規定，十二班以下，都市計畫區外學校校地面積不得少於二公頃(20,000 平方公尺)計算，每個班級依法應享有 1667 平方公尺的面積，這意味著班級裡的每個學生理當享有約為 1600m2=40mx40m 的活動範圍。依照前項所述，10pixels=0.5m 比例換算，實驗中的個體活動範圍可合理設為 800pixels x 800pixels 大小空間。

3. 群體與時間：

a. 每人每天接觸人數 12.43 人：依據中研院論文所提供，一個人每天平均的接觸人數為 12.43 人。

b. 實驗空間中的個體數量 100 顆球體： 依相關研究報告顯示，每個人每天有機會觸及的人數是有限的，從數位、數十位，乃至 100 人以上不等，這跟從宏觀角度需要考慮群體母數上千上萬以具備統計意義不同。如果以自己的中學校園來說明此變因，目前學校每班有 25 位學生，考量到有可能會接觸到隔壁兩班的同學(25+2×25)，還有下課時去上廁所、福利社、操場會接觸到的人數(大約一個班級的人)，所以推估活動範圍內總共有機會接觸到 25 人 x4，也就是約當 100 位學生。

c. 實驗中的時間校準: 100 rounds=1 day ： 依據每人每天平均碰撞次數 12.43 次的數據，本實驗中的球體碰撞每 300 輪左右會達到全體 100 顆球平均碰撞 12.43 次的結果，考慮人類外出活動時間只占一天中的三分之一，也就是 8 小時，因此實驗中每 100 輪(rounds)即代表真實世界一天。

4. 傳染相關參數

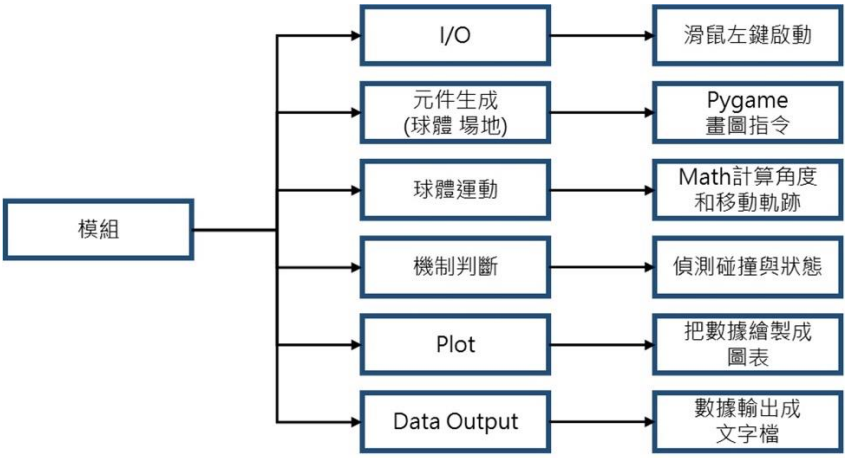
因為 R_0=感染機率×感染天數×接觸人數，已知目前流行的病株 Omicrom R_0 為 9-10 (R_0 值之相關說明請見附錄二)，而且研究報告[12]中第 44 頁指出：每人平均接觸人數為 12.43 次，感染天數為 7 天，感染機率就會是 10.3%-11.4%，故本研究取 11%(0.11)當作基準感染率係數。

三、實驗用環境與機制程式撰寫

以下是程式的運作原理、判別機制、以及所使用的程式模組

(1) 程式模組

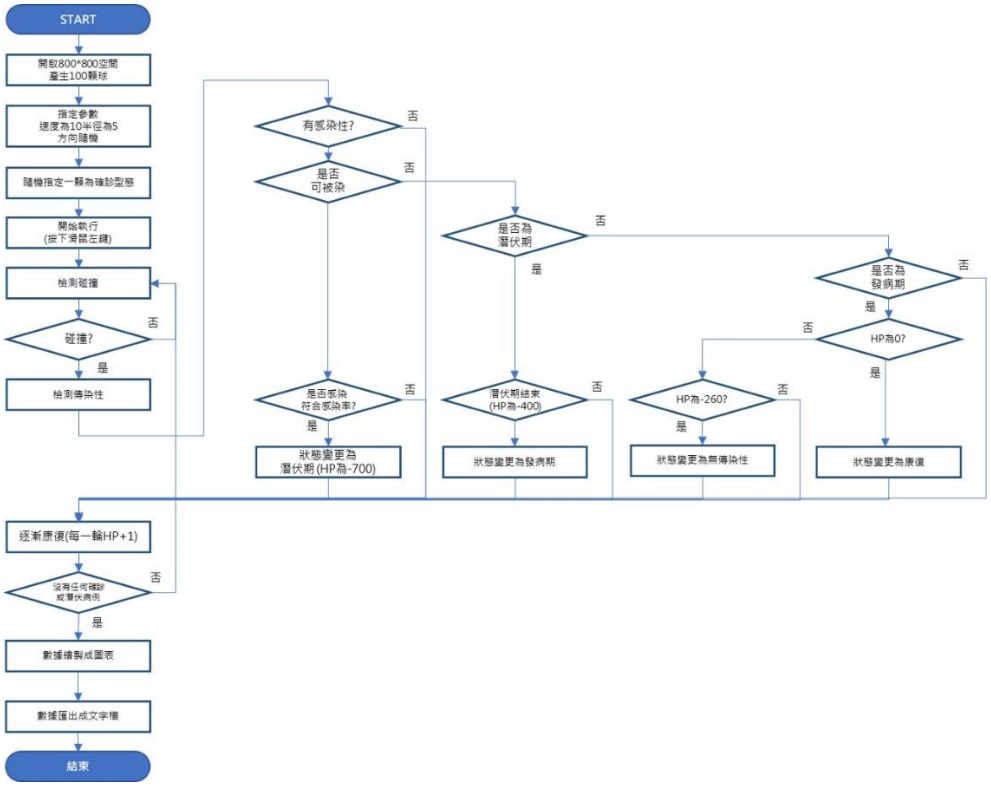
本實驗用模擬城市主要由六個模組所構成，並透過 Python 以及相關 library，像是 Pygame、Matplotlib 等完成相關功能。個功能模組請參見下圖：



- I/O 模組：啟動與中斷等控制元件模組。
- 元件生成模組：舉凡城市裡所有物件，像是球體、場地等，透過 Pygame library 生成佈建。
- 球體運動模組：計算每個球體在空間中的移動速度、方向角度等機制。
- 機制判斷模組：偵測場中每個球體倆倆是否碰撞，以及依當下生命值賦予易染者、潛伏期、發病期、恢復期等四種狀態。並予以統計紀錄。
- Plot 模組：將整個過程依照 rounds timeline 繪製 SEIR 四種狀態數量的變化。
- Data output 模組：依照 rounds timeline 將整個過程 SEIR 四種狀態數量的變化數值輸出為.txt 黨，作為後續數值分析使用。

(2) 程式運作流程

實驗用程式的流程圖如下：



(3) 模組與判斷機制

a. 元件生成：Class Ball

i. 定義每顆球的狀態，大小顏色與機制

```python
class Ball(pygame.sprite.Sprite):
    dx=0
    dy=0
    x=0
    y=0
    direction=0
    speed=0
    health=10
    def __init__(self,name,sp,srx,sry,radium,color,infected):
        pygame.sprite.Sprite. __init__(self)
        self.speed=sp
        self.x=srx
        self.y=sry
        self.infected=infected
        self.color=color
        print("a new ball is",name)
        self.image=pygame.Surface([radium*2,radium*2])
        self.image.fill((255,255,255))
        pygame.draw.circle(self.image,color, (radium,radium),radium,0)
        self.rect=self.image.get_rect()
        self.rect.center=(srx,sry)
        self.direction=random.randint(0,360)

    def update(self):
        radian=math.radians(self.direction)
        self.dx=self.speed*math.cos(radian)
        self.dy=-self.speed*math.sin(radian)
        self.x+=self.dx
        self.y+=self.dy
        self.rect.x=self.x
        self.rect.y=self.y
        if self.rect.left <= 0:
            self.bounceleft()
        if self.rect.right >= screen.get_width():
            self.bounceright()
        elif(self.rect.top <= 0):  #到達上邊界
            self.rect.top = 0
            self.bounceup()
        if(self.rect.bottom >= screen.get_height()):  #到達下邊界出界
            self.rect.bottom=800
            self.bounceup()
        else:
            return False

    def changeColor(self,color,radium):
        color=((color))
        self.image=pygame.Surface([radium*2,radium*2])
        self.image.fill((255,255,255))
        pygame.draw.circle(self.image,color,(radium,radium),radium,0)
    def bounceup(self):
        self.direction=360-self.direction
    def bouncelr(self):
        self.direction=self.direction-180
    def bounceleft(self):
        self.direction=180 + (360-self.direction)
    def bounceright(self):
        self.direction=180-self.direction
```

ii. 自動生成 1 顆染疫+99 顆健康球，並且各自帶有 ID，以便後續追蹤控制每一顆球的行為以及健康狀態。

```python
for i in range(1,2):
    x_coordinates=random.randrange(20,780)
    y_coordinates=random.randrange(20,780)
    locals()['ballname_'+str(i)]=Ball(i,10,x_coordinates,y_coordinates,10,(255,0,0),-400)
    allsprite.add(locals()['ballname_'+str(i)])

for i in range(2,101):
    x_coordinates=random.randrange(20,780)
    y_coordinates=random.randrange(20,780)
    locals()['ballname_'+str(i)]=Ball(i,10, x_coordinates, y_coordinates, 5, (0,0,255),0)
    allsprite.add(locals()['ballname_'+str(i)])
```

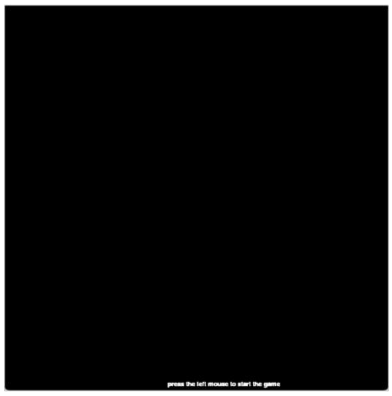
b. 機制判斷

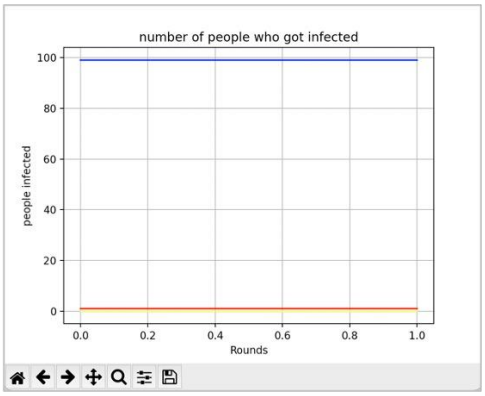
檢測碰撞，機率，狀態，並且改變球體顏色和數值：

每顆球都有一個生命值參數 hp，若是 hp>1 則為健康狀態(Susceptible/Blue)，顯示為藍色，若是 hp<0 則為染疫狀態。因為是 SEIR 模型，因此染疫狀態有兩種，分別是潛伏期(Exposed/greed)以及感染期(Infected/red)，以潛伏期 300rounds/感染期 400rounds 為例，當健康球體被感染時，hp=-(300+400)，並呈現綠色(潛伏期)，隨後每回合 hp+1，當 hp>-300 時便進入感染期(Infected)，球體顏色由綠轉紅，隨後每回合 hp+1，當 hp>-1 時便進入健康期，球體恢復為藍色。

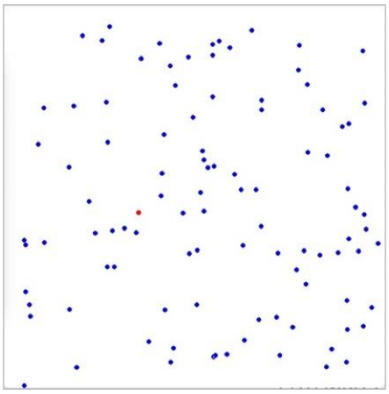
(4) 模擬用程式各階段工作與執行畫面

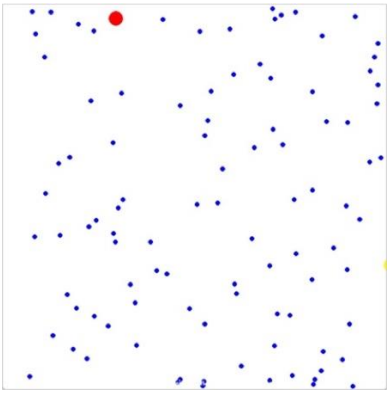
a. 階段零-準備：創建(1) 800x800 空間，按下滑鼠左鍵即開始(下方左圖)。(2)呈現Ｓ、Ｅ、Ｉ、Ｒ四組數據變化狀態圖表(下方右圖)。



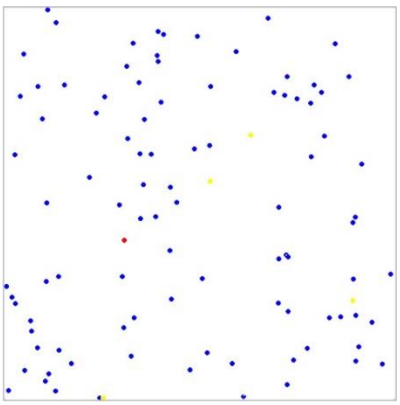


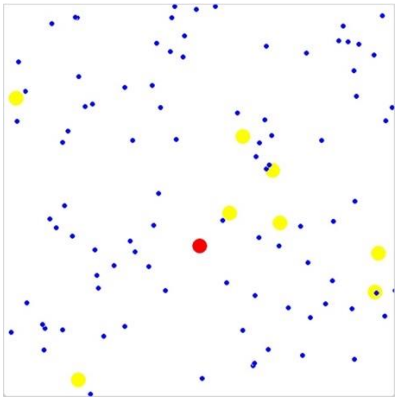
b. 階段一-啟始：創建位置與運行方向隨機指派的 100 顆球(藍色)，並隨機指定一顆為感染狀態(紅色)(下方左圖)。(有些實驗條件被感染球體半徑會變大/下方右圖)



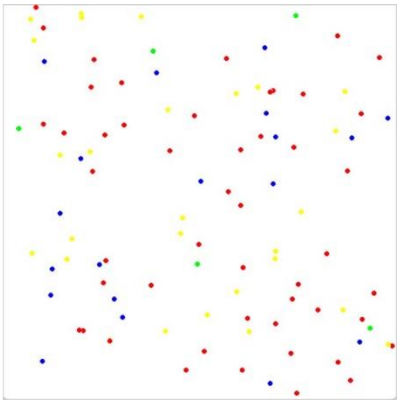
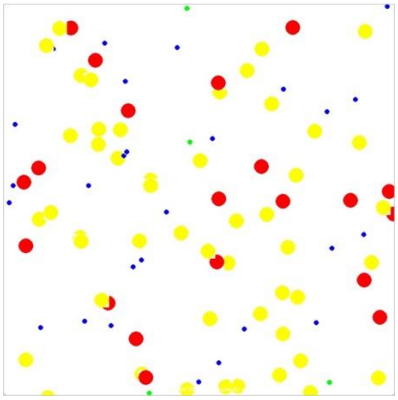


c. 階段二-傳播狀態Ｉ：被感染者進入無症狀的潛伏期(黃色)，並開始擴大感染(下方左圖)。(有些實驗條件被感染球體半徑會變大/下方右圖)

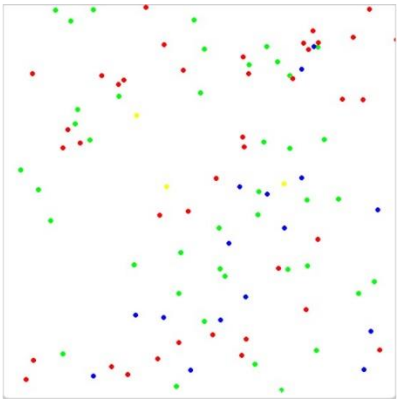
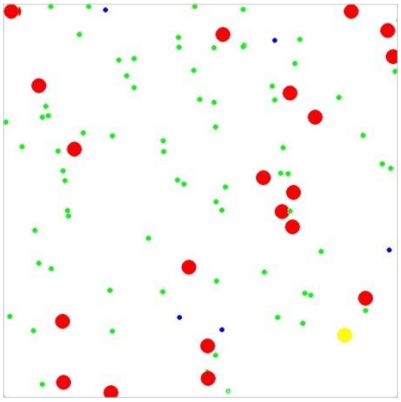




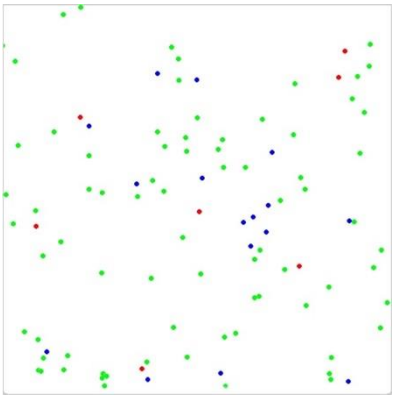
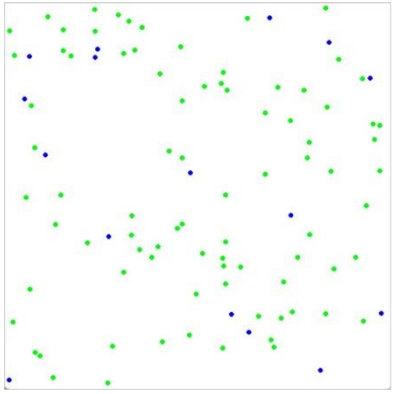
d. 階段三-傳播狀態 II：潛伏期(黃色)開始進入有症狀發病期(紅色)，被感染數量迅速飆升(下方左圖)。(有些實驗條件被感染球體半徑會變大/下方右圖)

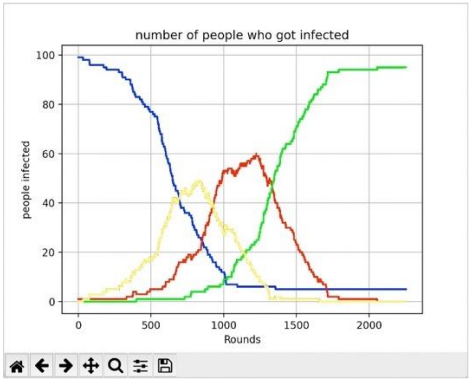
e. 階段四-恢復期：走完發病週期的個體恢復健康(綠色)

f. 階段五-結束：當空間中已不再有具備傳染性的個體，整個傳播週期結束。

g. 階段六-輸出：(1) 模擬結束(下方左圖)，畫出依時間軸 S、E、I、R 四個狀態的變化(下方右圖)。(2) 輸出數據並存為.txt 檔。



四、研究方法

考慮 Omicron 有無症狀感染者(潛伏期)，染疫後有三個月免疫期，且復陽率不高(低於 5%)，因此本研究採用 **SEIR 模型**，參數設定為 **100 顆球體**，位置與方向隨機產生於 **800x800 的空間**裡，透過**相互碰撞**模擬感染。

一、 實驗變項：

1. **接觸傳染機率(8%,11%,14%,17%)** – 本實驗的基本傳染律定為 11%，若傳染率增減，對於全體中感染的分佈與速率有什麼影響？
2. **傳染時間(3+ 4/3、3+4 天)** – 新型病株傳染的時間越來越短，這對於群體中的感染狀況有什麼影響？
3. **球體半徑(r=5,10,15)** – 疫情爆發後，隨著病株快速演化，從體液的接觸傳染變成飛沫與空氣傳染，因此有社交距離 1.5m 的健康準則，透過染疫後球體變化(從原本 r=5 變成 10 or 15)，藉由微觀的模擬觀察對群體內感染狀況有什麼影響與變化。

二、 實驗規劃：

實驗編號共三碼，第一碼為半徑，第二碼為感染期，第三碼為感染率。

(一)、實驗一(t001)：半徑為 5，傳染期為潛伏期加上發病期的前 1/3，感染率為 11%

(二)、實驗二(t002)：半徑為 5，傳染期為潛伏期加上發病期的前 1/3，感染率為 8%

(三)、實驗三(t003)：半徑為 5，傳染期為潛伏期加上發病期的前 1/3，感染率為 14%

(四)、實驗四(t004)：半徑為 5，傳染期為潛伏期加上發病期的前 1/3，感染率為 17%

(五)、實驗五(t011)：半徑為 5，傳染期為潛伏期加上發病期共 7 天，感染率為 11%

(六)、實驗六(t012)：半徑為 5，傳染期為潛伏期加上發病期共 7 天，感染率為 8%

(七)、實驗七(t013)：半徑為 5，傳染期為潛伏期加上發病期共 7 天，感染率為 14%

(八)、實驗八(t014)：半徑為 5，傳染期為潛伏期加上發病期共 7 天，感染率為 17%

(九)、實驗九(t101)：半徑為 15，傳染期為潛伏期加上發病期的前 1/3，感染率為 11%

(十)、實驗十(t102)：半徑為 15，傳染期為潛伏期加上發病期的前 1/3，感染率為 8%

(十一)、實驗十一(t103)：半徑為 15，傳染期為潛伏期加上發病期的前 1/3，感染率為 14%

(十二)、實驗十二(t014)：半徑為 15，傳染期為潛伏期加上發病期的前 1/3，感染率為 17%

(十三)、實驗十三(t201)：半徑為 10，傳染期為潛伏期加上發病期的前 1/3，感染率為 11%

(十四)、實驗十四(t202)：半徑為 10，傳染期為潛伏期加上發病期的前 1/3，感染率為 8%

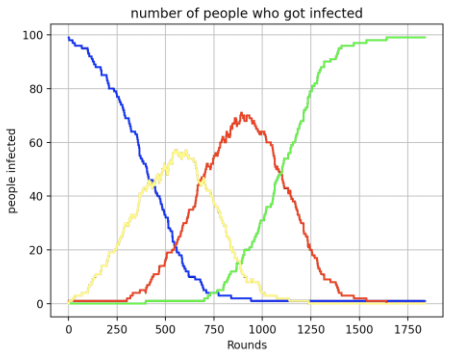
(十五)、實驗十五(t203)：半徑為 10，傳染期為潛伏期加上發病期的前 1/3，感染率為 14%

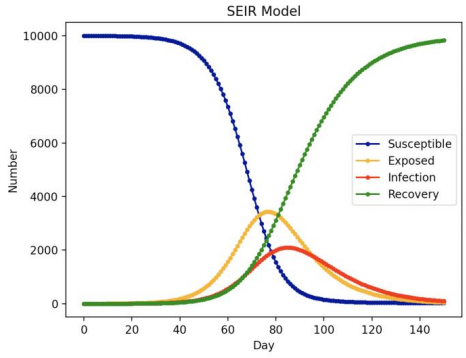
(十六)、實驗十六(t204)：半徑為 10，傳染期為潛伏期加上發病期的前 1/3，感染率為 17%

伍、實驗結果

一、微觀與宏觀方法結果的比較

微觀粒子碰撞的實驗結果圖形(左圖)與宏觀微分方程式的結果(右圖)類似，以有限空間與粒子的碰撞模擬傳播行為，可以得到看似與微分方程式相同的結果。





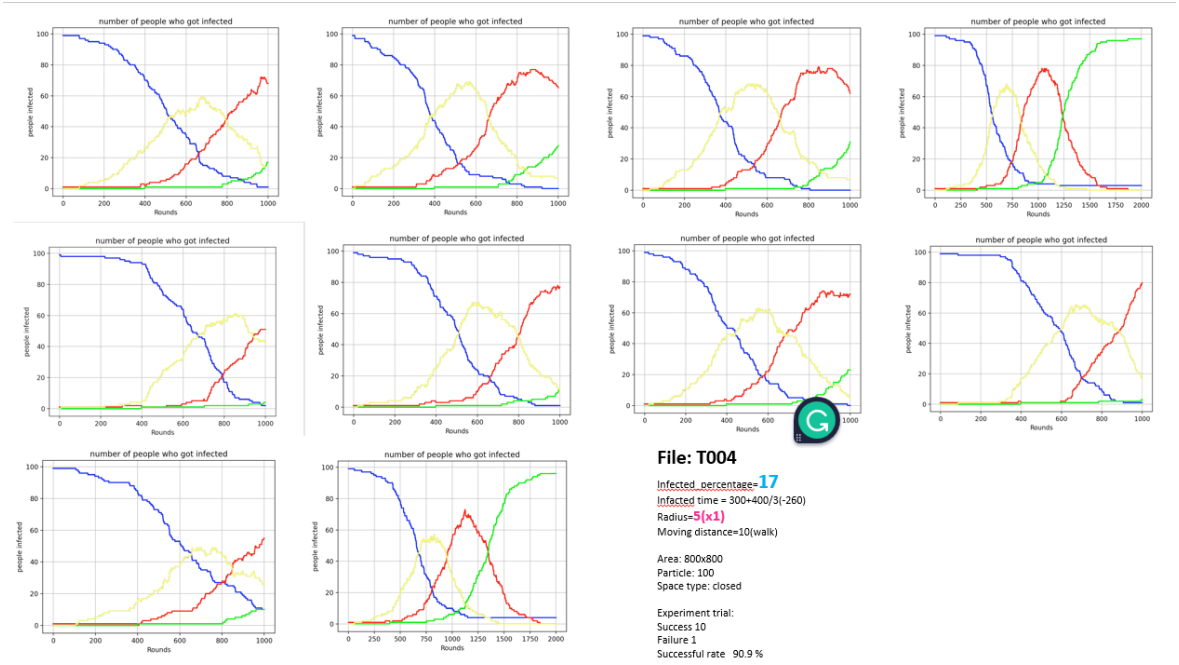
二、實驗結果

本實驗共有三種條件組合，4 個組合系列 T00x,T10x,T20x,以及 T01x，詳細條件如下：

a. T00x 系列：r=5, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。
b. T10x 系列：r=10, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。
c. T20x 系列：r=15, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。
d. T01x 系列：r=5, 感染機率 8%,11%,14,17%，感染期 3+4 天。

每個條件至少取十組有效的數據，若數據結果發散則視情況增加有效組數。實驗結果請見下圖示意，所有的數據以圖形紀錄，詳情請參見附錄三。

a. T00x 系列：r=5, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。



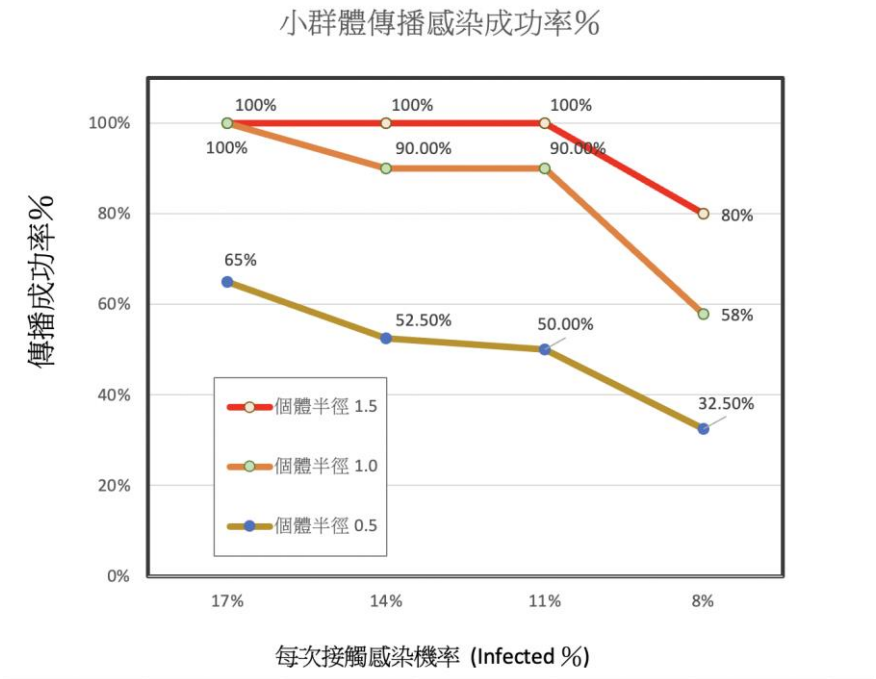
三、實驗分析與發現

本實驗是在有限區域有限人數中，放入一位起始染疫者，透過染疫者與他人接觸的過程逐步將疫情傳染給該群體的成員。過程中發現，不是每次群體出現一位感染者就會擴散至全體群組，甚至還有整個傳染期限內，沒有其他人受到感染的現象，相關結果說明如下：

a. 有限群體內的成功傳播率-染疫者在傳染期沒傳染給任何一個人(球)：

球體半徑的角度：相較於 r=10(T10x 組)、r=15(T20x 組)兩組，球體半徑 r=5(T00x 組)，傳播成功率有明顯的差異，在感染機率為 17%的條件時，當群體中出現一位染疫者，有 1/3 機率不會在群體中擴散疫情，傳播機率降為 8%時，擴散失敗的比例近一步擴大至 2/3。但是若球體半徑 r=10 以上時，成功機率都在 90%以上，但若傳播機率只有 8%，三組條
件均看到明顯的下降。

傳播成功率	個體半徑	個體半徑	個體半徑
接觸感染率 (Infected %)	0.5	1.0	1.5
17%	65%	100%	100%
14%	52.5%	90%	100%
11%	50%	90%	100%
8%	32.5%	58%	80%

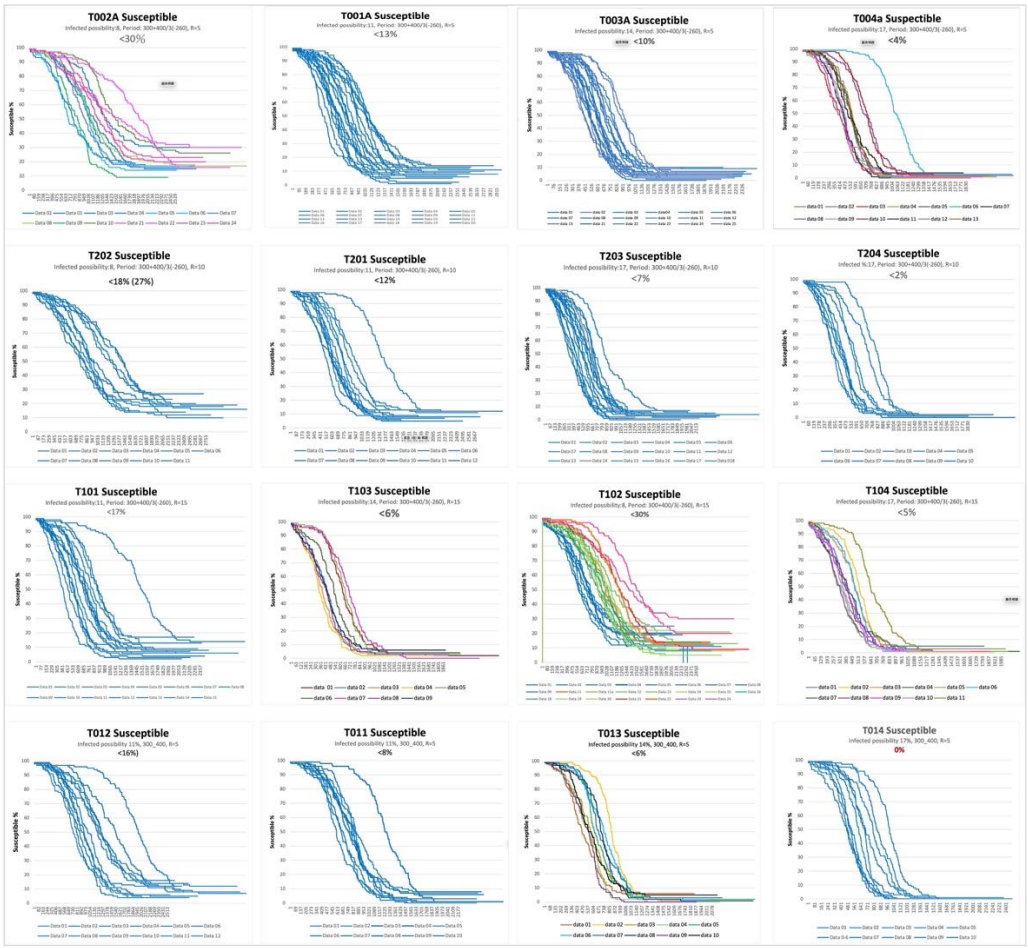


b. 「天選之人」的現象-未染疫者穿梭於大量染疫者之間未被感染：

實驗機制中，傳染發生的機制需要買足三個條件：碰撞事件、可被感染、感染機率，三者缺一不可。從實驗的 16 個條件中發現，除了 T014 感染期最長(3+4 天)、感染率最高(17%)這一組，所有實驗皆達到全員完全感染之外，其他的皆有幸運的漏網之魚，亦即我們一般稱之為天選之人永不染疫的狀況，本實驗並不考慮基因的變項，純粹是以機率為前提。詳細數據請見下方圖表。

從實驗數據發現，如果感染率降至 8%，即便該全體發生疫情擴散現象，仍有高達 30%的成員因機率而逃過一劫，若是合併考慮前一項因素，在特定條件下(T004)有 77.25%(67.5%+32.5%x0.3)的人不會感染。

Susceptible % /conditions	感染機率			
	8%	**11%**	**14%**	**17%**
半徑 r=5, 感染期:3 + 4/3	<30%	<13%	<10%	<4%
半徑 r=10, 感染期:3 + 4/3	<27%	<12%	<7%	<2%
半徑 r=15, 感染期:3 + 4/3	<30%	<17%	<6%	<5%
半徑 r=5, 感染期:3 + 4	<16%	<8%	<6%	**0%**

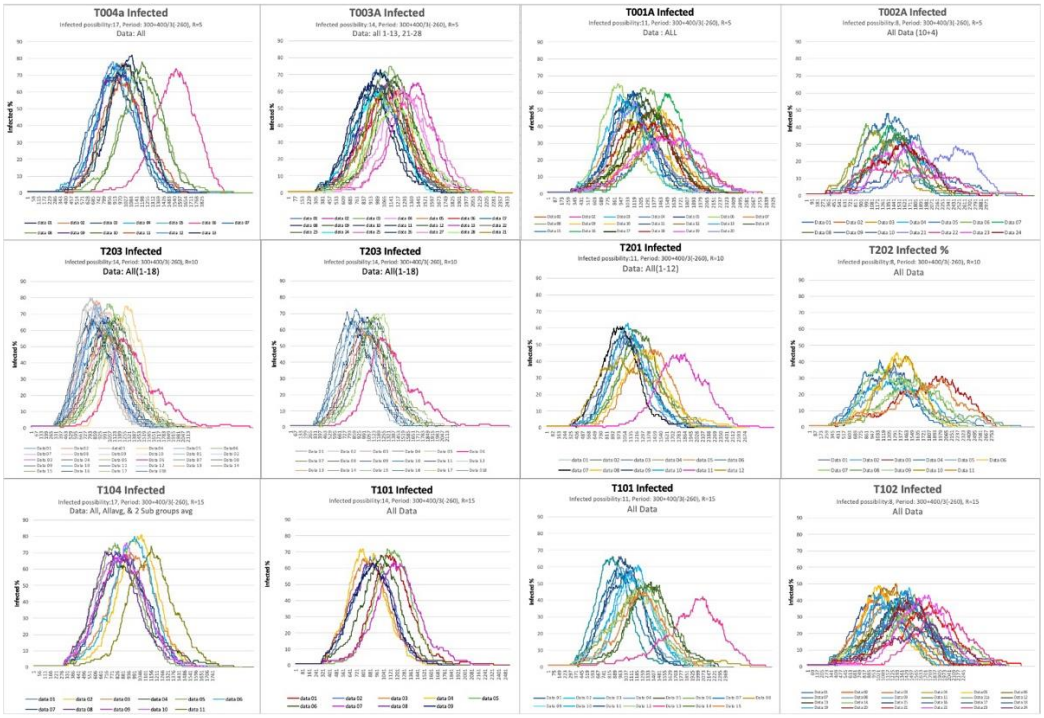


c. 宏觀是多種微觀行為的平均：

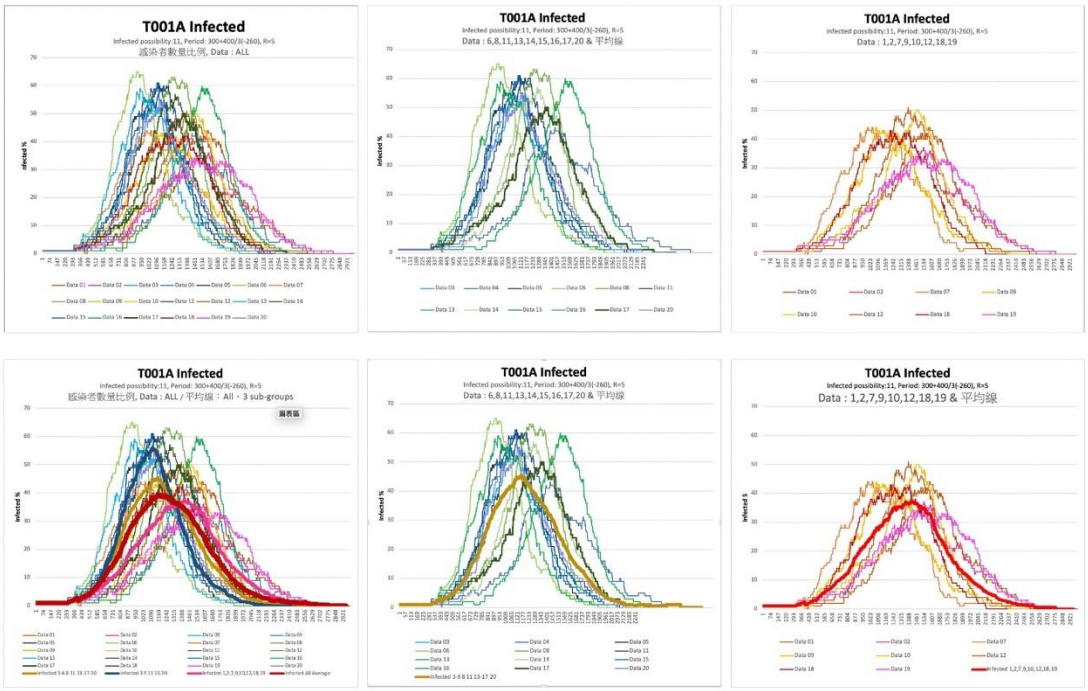
將各組的實驗結果統合一起，可以看到各組之間雖然可以看到差異，但是組間的數據本身也差異不小，甚至可以看到不同的波形、與持續時間

組間的數據大致上可以分為 2 類子集合，一種是發生時間早，峰值高且波形較瘦，屬於快猛型，另一種是發生時程長，峰值低且波形矮胖的溫和型。從這個數據來看，若將真實世界切分成許多 800x800 的 cell，在不考慮各個 cell 之間互動的條件下，每個 cell 裡疫情的發展狀況差異可以很巨大，如果直接將所有的數據平均，這些細節都會消失。不過從這裡的數據可以發現，若是在疫情爆發時將全市切割成幾個小的區域，彼此不相往來，如此該區域突然出現一位染疫者，似乎也有機會免於疫情擴散至全體的局面，安然度過疫情。(分區風險控制)

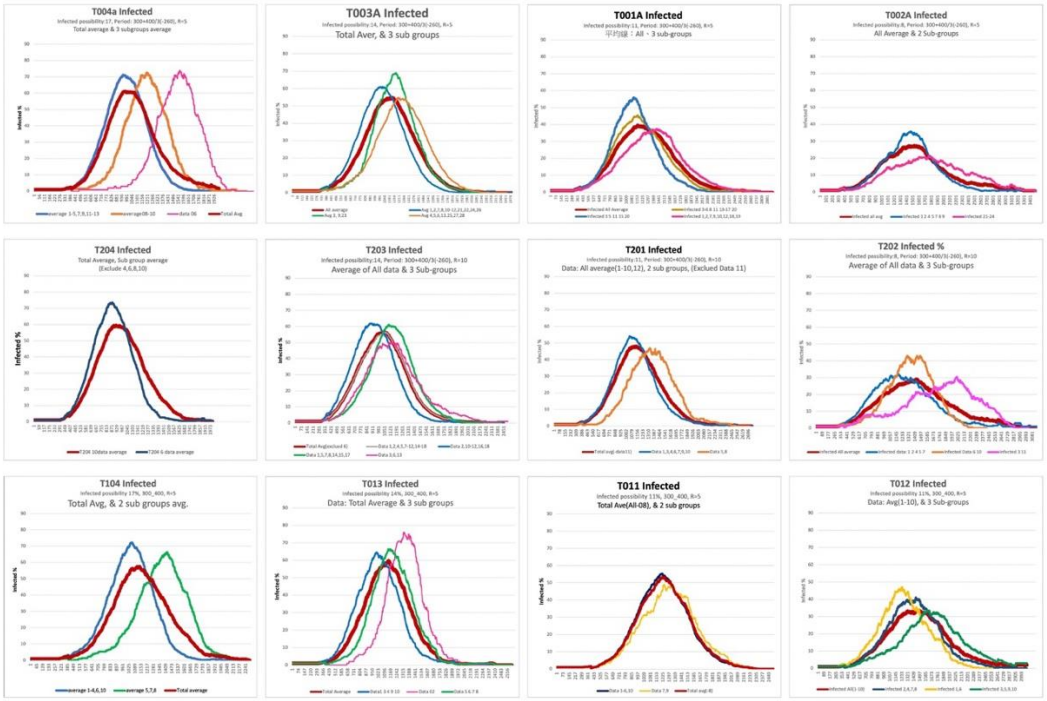
T00x、T10x、T20x 數據請參閱下方圖表。



下圖是以標準組 T00a 為例，可以看到依照各個波的峰值與發生的時間，大致分為兩類。左邊上圖是 T001 所有數據的疊加，中、又兩圖則是依照波的形狀分為兩群，下方的三張圖則是將各組平均後的波線疊加在原始數據上。

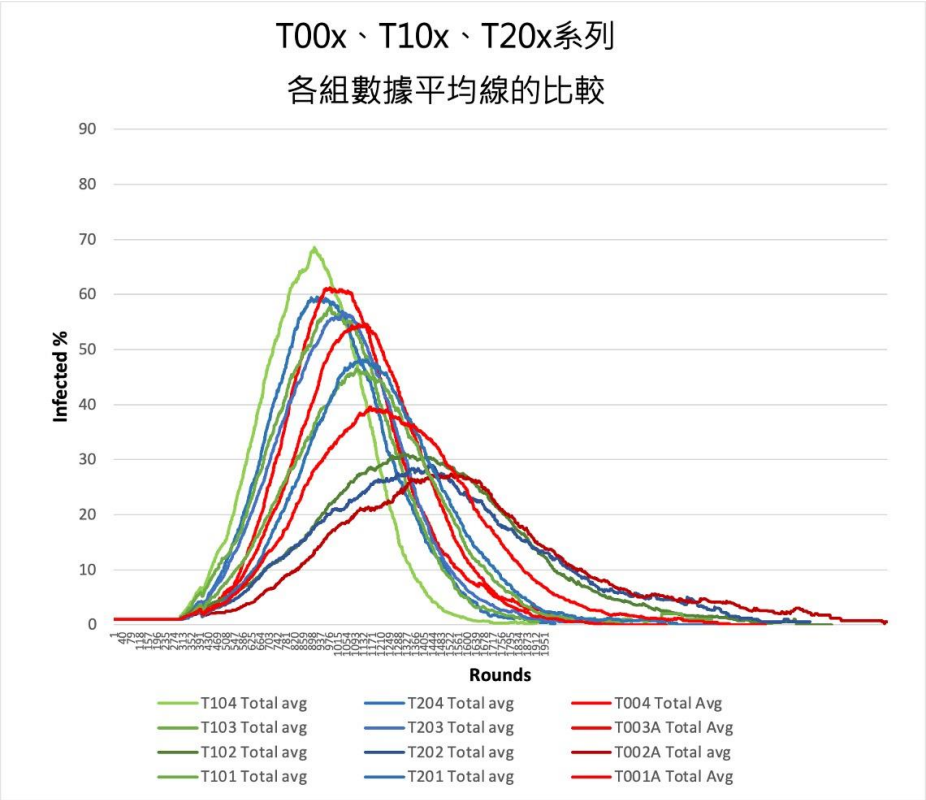


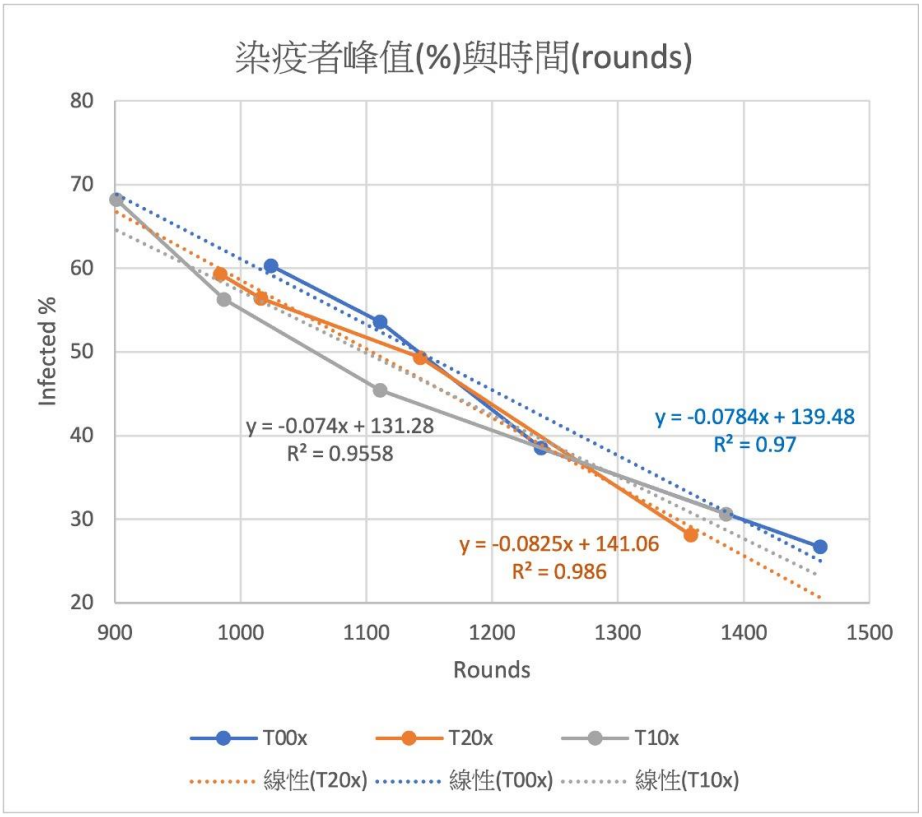
依上述的分析，試著將 T00x~T20x 共 12 組數據，依波型分佈與型態，拆解成 2-3 個不等的子群及，請參見下圖：



d. 以球體半徑變化模擬社交距離(r=5/10/15)對疫情變化的影響：

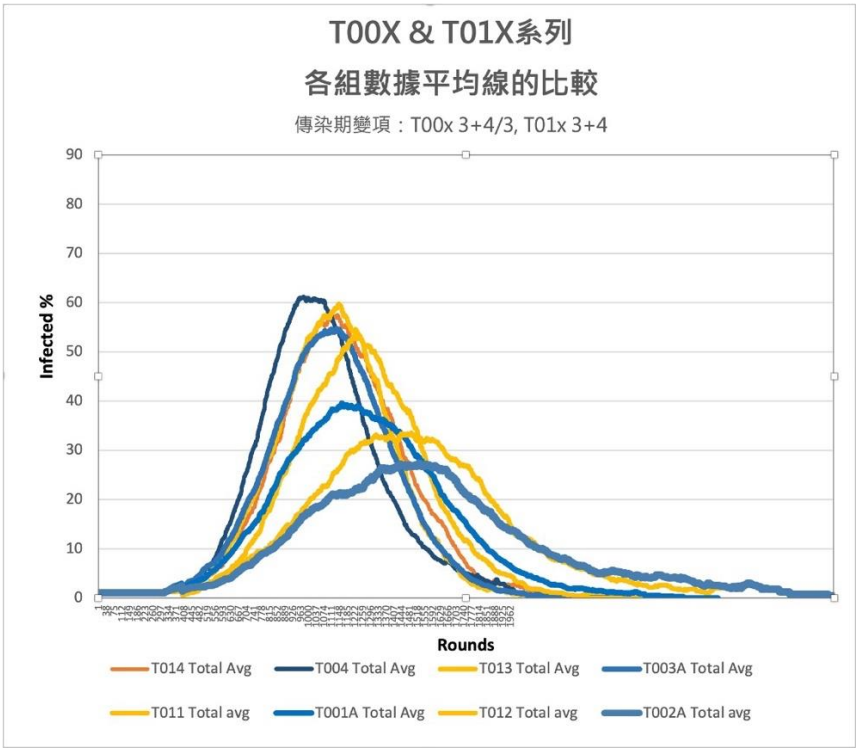
從感染高峰發生的時間與強度來看，在球體半徑 r、碰撞人數、傳染天數等變數 fixed 的情況下，各個條件的峰值大體在 $y=-0.078x + 140$, $R^2=0.97$ ~ $y=-0.074x + 131.28$, $R^2=0.95585$ 這個維度區間上變化，此時即便是改變球體半徑 r，縱使稍有偏離，但也各個條件的峰值也在相仿的維度上變化（請參見下圖）。這意味著球體半徑 r 的變化可能不是一個獨立變項，可以透過感染機率的增減達到類似的結果。

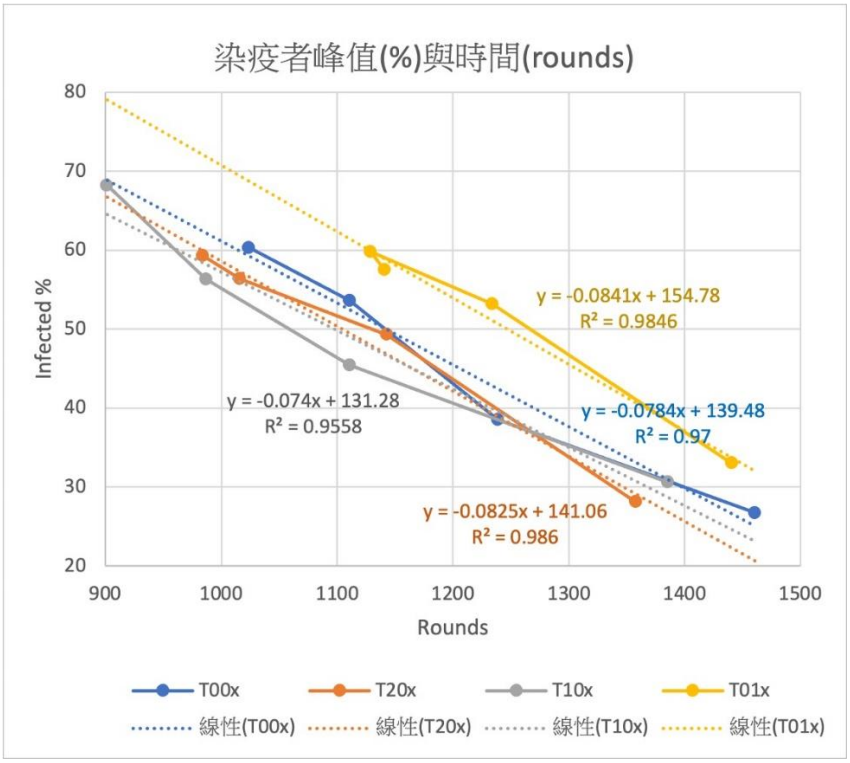

T00x、T10x、T20x系列
各組數據平均線的比較
Infected %
Rounds
T104 Total avg
T204 Total avg
T004 Total Avg
T103 Total avg
T203 Total avg
T003A Total Avg
T102 Total avg
T202 Total avg
T002A Total avg
T101 Total avg
T201 Total avg
T001A Total Avg


染疫者峰值(%)與時間(rounds)
Infected %
Rounds
y = -0.074x + 131.28
R² = 0.9558
y = -0.0784x + 139.48
R² = 0.97
y = -0.0825x + 141.06
R² = 0.986
T00x
T20x
T10x
線性(T20x)
線性(T00x)
線性(T10x)

e. 傳染期時間長度變化峰值線的影響：

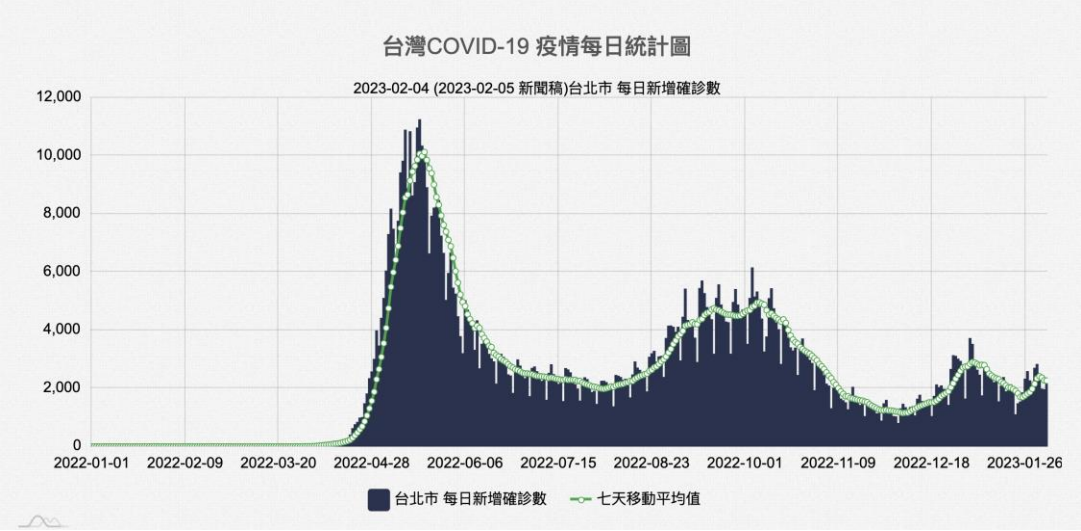
延續高峰線的討論，若是考慮傳染時間長度的變項，高峰線是否還是在同樣的位置上呢？採用相同的方式，將 T01x 系列的數據取平均值後，將峰值的相關數據與其他三組做比較，此時高鋒線並沒有與其他三組重合，而是向右上偏移，請參閱下圖。從 R_0 的公式可以理解，傳染天數與感染機率是不同的獨立變項，因此當參數內容改變，高峰線也會隨之不同。



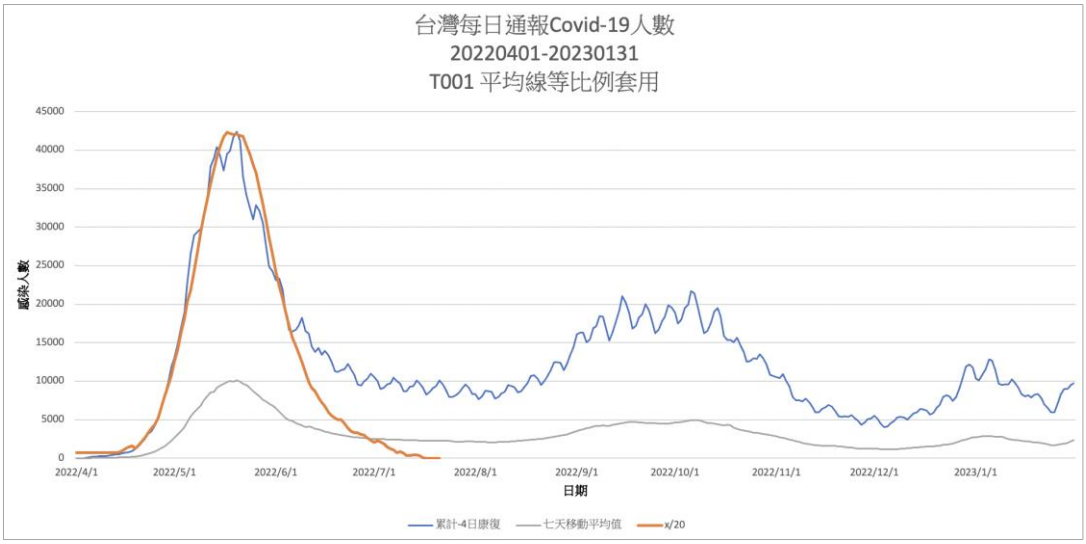


四、與真實疫情數據的比較

疫情爆發至今已經三年，過程中衛福部美日公告染疫人數的相關資訊，本實驗於實驗開始即每天記錄疫情變化(請參閱附件)。隨著疫情的變化，不同病株的傳染行為也不太相同。下圖是國家高速網路計算中心維護的 Covid-19 全球疫情地圖網站[13]中，擷取 2022/04/01-2023/01/31 期間的染疫人數變化，這段期間只要流行病珠為 Omicron，期 R_0 約為 10~11，潛伏期約為 3.4 天，發病期約 4 天：



實驗組 T001 的條件為 感染率 11%、接觸人數 12.42 人、感染其為 3+4 天，換算 R_0 為 9.57，與 Omicron 相當。將 T001 的平均值依比例調整峰值大小，並且以 20rounds 為 1 day，可以完美符合真實數據(請見下圖)。



陸、結論

本實驗是以「有限群體在有限空間中，透過彼此接觸傳遞病毒」為基礎，從微觀的角度以 SEIR 模型的原理看流行疾病傳播的機制，除了能得到相似於宏觀角度常微分方程的結果之外，還可以看到許多沒有想到的細節:

1. 在較嚴苛的條件下，即使有一位染疫者進到群體中，也不一定會將病毒傳播到群體裡，其次，即使是疫情在群體中擴散開來，也會有一定比例的人不會受到感染，在有些條件下一個群組內甚至高達 30%的人平安無事。
2. 在不同的條件下，的確可以看到各組之間的差異，但除此之外組內也存在著明顯的差異。經過分析組內差異大致可以分為兩類，一個是發生周期短、峰值高的猛快型，另外一種是週期很長、緩慢進展、峰值低的緩和型，每一組的數據基本上都可以用這兩種子類型組合而成。
3. 從上述結果來看，若是在疫情爆發流行時，若是能夠將都市的空間切分成許多彼此不相往來的小型生活區域，的確會有一定比例的區域能夠幸運躲過疫情的擴散，甚至透過感染機率的控制，降低區域內感染的人數比例。
4. 以半徑大小變化來模擬社交距離的空間變量，從實驗的結果的確是有差別，但若是比對染液者峰值的強度與發生時間，會發現半徑大小變化的三組數據，幾乎都是在同一個趨勢線上，代表這個空間變量並非完全獨立變項，而是可以透過調整感染機率大小作為替代。
5. 相較於空間變量，傳染期時間長度改變時，可以看到染疫者高峰值的趨勢線跟對照組並不在同一條趨勢線上，而是往右上方平移。

以有限區域粒子碰撞的微觀角度檢視流行性疾病傳染的過程，的確可以看到宏觀角度看不到的細節，並且提供給防疫相關人員制定政策的參考。

柒、參考資料

[1] SIR 模型：讓我們用「南部力學」模擬傳染病
https://www.thenewslens.com/article/135910

[2] 數學建模常用算法---傳染病模型(二)SIS 模型
https://zhuanlan.zhihu.com/p/142017716

[3] [Pygame] 繪製矩形的 Rect 簡單介紹
https://clay-atlas.com/blog/2021/09/10/pygame-cn-rect/

[4] 傳染病 SIR 模型與機率模型介紹與簡易模擬
https://tigercosmos.xyz/post/2020/03/sir/

[5] 傳染病的數學模型分析, 金門地區第 60 屆科學展覽, 2020
https://science.km.edu.tw/storage/media/2206/5e8980c6e63d2.pdf

[6] 衛生福利部疾病管制署(https://www.cdc.gov.tw/)

[7] 新冠肺炎的社會衝擊—跨國分析比較 | COVID-19 的人文社會省思(ascdc.tw)

[8] 啟新健康世界-專業健康檢查/醫學健康促進 (ch.com.tw)
https://www.ch.com.tw/index.aspx?sv=ch_epaper&chapter=epaper20210603

[9] 新冠肺炎》確診後幾天才會失去傳染力？, 風傳媒網站
https://www.storm.mg/lifestyle/4367876

[10] 歐美主要國家肺炎治癒率, 財經 M 平方網站
https://www.macromicro.me/charts/19684/us-euro-covid-recovered-ratio

[11] **行政院衛生署疾病管制局 99 年度科技研究發展計畫**
"類流感散佈的相關社會混合型態:接觸日誌的建構與應用"

[12] 法規名稱：國民小學及國民中學設施設備基準,公發布日：民國 91 年 06 月 10 日,修正日期：民國 108 年 07 月 24 日，發文字號：臺教授國部字第 1080056287B 號令。https://edu.law.moe.gov.tw/LawContent.aspx?id=GL000314

[13]Covid-19 全球疫情地圖網站，國家高速網路計算中心
https://covid-19.nchc.org.tw/index.php

附錄一 ：R_0 值說明

R_0 值(Basic Reproduction Number)全名為「基本傳染數或基本再生數」，是流行病學評估傳染病控制成效的重要指標。當病毒進入無抵抗力的族群中，也就是在所有人都無免疫力的情況下，平均一名確診者從染疫到康復或死亡期間，能傳染給幾個人的數值。R_0 值越大並非代表病毒越毒，而是病毒的繁殖速度越快，人與人之間的傳染力越強，疫情的傳播力就越難控制。美國 CDC 分析，COVID-19 的 R_0 值在 4～7.1，也就是一名感染者可傳給 4～7 人；以英國變種病毒(Alpha)為例，R_0 值超過 5，因此傳染力相當強。

R_0 值會變動的，當人與人之間的社交距離越親密、病毒傳染力越強或是病毒持續的病程越長，都會讓 R_0 值上升。R_0 值若超過 10 以上，代表疫情傳播相當嚴峻。當多數人落實戴口罩、勤洗手及保持適當社交距離等防疫，R_0 值就會下降。一旦 R_0 值小於 1，表示一人平均傳染不到一人，疾病就難以繼續流行。因此，R_0 值高低可判斷 COVID-19 是否容易控制。

一般 R_0 值用最近七天的確診個案統計

附錄二：五到七月中每天本土病例境外移入與死亡人數

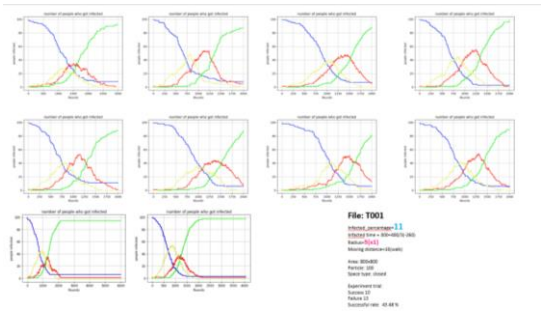
	日期	確診	死亡	本土	境外移入	五月平均		60870	死亡率
日	5 月 1 日	17088	3	16936	152	第一週平均		28490	0.000176
一	5 月 2 日	17858	3	17801	57	第二週平均		55315	0.000168
二	5 月 3 日	23139	5	23102	37	第三週平均		77499	0.000216
三	5 月 4 日	28487	5	28420	67	第四週平均		82177	0.000176
四	5 月 5 日	30106	5	30035	71	第五週平均(包含六月)		75344	0.000166
五	5 月 6 日	36213	10	36168	45	第六周平均		71337	0.000276
六	5 月 7 日	46536	11	46377	159	第七周平均		57587	0.000236
日	5 月 8 日	44361	12	44294	67	第八周平均		47063	0.000271
一	5 月 9 日	40304	12	40263	41	六月平均		57857	0.000298
二	5 月 10 日	50828	12	50780	48	第九周平均		37775	0.000236
三	5 月 11 日	57216	8	57188	28	第十周平均		30924	0.00014
四	5 月 12 日	65446	17	65385	61	第十一周平均		26639	0.00026
五	5 月 13 日	65011	41	64972	39	第十二週平均		24076	0.000631
六	5 月 14 日	64041	40	63964	77	七月平均		26327	0.000625
日	5 月 15 日	68769	19	68732	37				0.000276
一	5 月 16 日	61754	29	61697	57				0.00047
二	5 月 17 日	65833	38	65794	39				0.000577
三	5 月 18 日	85356	41	85310	46				0.00048
四	5 月 19 日	90378	59	90331	47				0.000653
五	5 月 20 日	85761	49	85720	41				0.000571
六	5 月 21 日	84639	59	84587	52				0.000697
日	5 月 22 日	79487	53	79441	46				0.000667
一	5 月 23 日	66283	40	66247	36				0.000603
二	5 月 24 日	82435	42	82363	72				0.000509
三	5 月 25 日	89389	76	89352	37				0.00085
四	5 月 26 日	81907	104	81852	55				0.00127
五	5 月 27 日	94855	126	94808	47				0.001328
六	5 月 28 日	80881	127	80835	46				0.00157
日	5 月 29 日	76605	145	76539	66				0.001893
一	5 月 30 日	60103	109	60042	61				0.001814
二	5 月 31 日	80705	90	80656	49				0.001115
三	6 月 1 日	88296	122	88247	46				0.001382
四	6 月 2 日	76986	144	76967	19				0.00187
五	6 月 3 日	76564	142	76517	47				0.001855

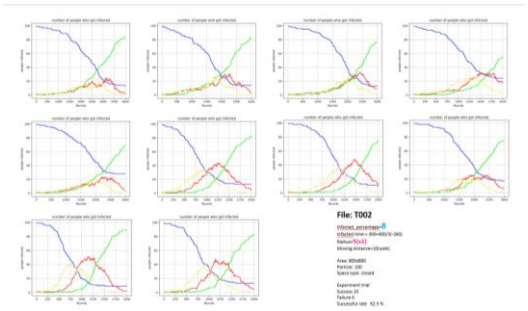
六	6 月 4 日	68151	152	68118	33				0.00223
日	6 月 5 日	62110	124	62080	30				0.001996
一	6 月 6 日	53023	151	52992	31				0.002848
二	6 月 7 日	83027	124	82973	54				0.001493
三	6 月 8 日	80223	159	80195	28				0.001982
四	6 月 9 日	72967	211	72921	46				0.002892
五	6 月 10 日	68347	213	68311	36				0.003116
六	6 月 11 日	79663	211	79598	65				0.002649
日	6 月 12 日	50657	163	50643	14				0.003218
一	6 月 13 日	45110	109	45081	29				0.002416
二	6 月 14 日	66189	123	66119	70				0.001858
三	6 月 15 日	68965	143	68939	26				0.002074
四	6 月 16 日	63221	168	63170	51				0.002657
五	6 月 17 日	55261	154	55187	74				0.002787
六	6 月 18 日	53707	181	53643	64				0.00337
日	6 月 19 日	50636	172	50561	75				0.003397
一	6 月 20 日	35632	144	35596	36				0.004041
二	6 月 21 日	56404	115	56339	65				0.002039
三	6 月 22 日	52246	171	52213	33				0.003273
四	6 月 23 日	48356	166	48283	73				0.003433
五	6 月 24 日	45767	152	45678	89				0.003321
六	6 月 25 日	40398	151	40293	105				0.003738
日	6 月 26 日	39646	134	39586	60				0.00338
一	6 月 27 日	28580	91	28489	91				0.003184
二	6 月 28 日	44428	103	44379	49				0.002318
三	6 月 29 日	42204	85	42112	92				0.002014
四	6 月 30 日	38942	118	38846	96				0.00303
五	7 月 1 日	35800	121	35699	101				0.00338
六	7 月 2 日	34827	96	34748	79				0.002756
日	7 月 3 日	32681	88	32567	114				0.002693
一	7 月 4 日	23118	69	23045	73				0.002985
二	7 月 5 日	36015	103	35914	101				0.00286
三	7 月 6 日	34577	95	34499	78				0.002747
四	7 月 7 日	31462	105	31364	98				0.003337
五	7 月 8 日	30477	131	30314	163				0.004298
六	7 月 9 日	28135	94	28028	107				0.003341
日	7 月 10 日	27844	71	27708	136				0.00255

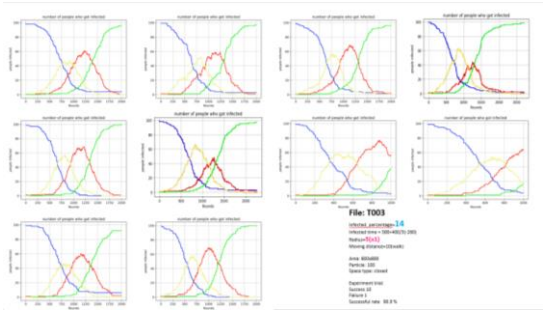
一	7 月 11 日	19131	96	19051	80				0.005018
二	7 月 12 日	31297	60	31152	145				0.001917
三	7 月 13 日	29911	49	29849	62				0.001638
四	7 月 14 日	27684	89	27597	87				0.003215
五	7 月 15 日	25310	72	25223	87				0.002845
六	7 月 16 日	25296	72	25251	45				0.002846
日	7 月 17 日	24325	73	24196	129				0.003001
一	7 月 18 日	18003	48	17549	454				0.002666
二	7 月 19 日	27146	37	26779	367				0.001363
三	7 月 20 日	27196	57	26943	253				0.002096
四	7 月 21 日	25171	74	24907	264				0.00294
五	7 月 22 日	24093	86	23813	280				0.00357
六	7 月 23 日	22595	53	22347	248				0.002346
日	7 月 24 日	21737	65	21460	277				0.00299
一	7 月 25 日	17127	53	16921	206				0.003095
二	7 月 26 日	25071	37	24790	281				0.001476
三	7 月 27 日	25692	28	25521	171				0.00109
四	7 月 28 日	23971	62	23822	149				0.002586
五	7 月 29 日	23289	57	22983	306				0.002448
六	7 月 30 日	21501	60	21273	228				0.002791
日	7 月 31 日	21069	34	20824	245				0.001614

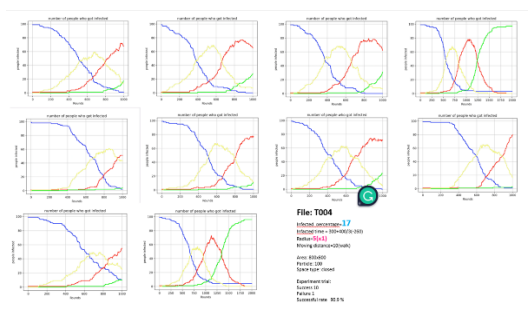
附錄三：實驗結果

a. T00x 系列：r=5, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。

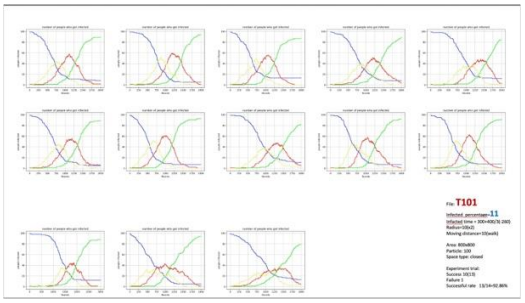


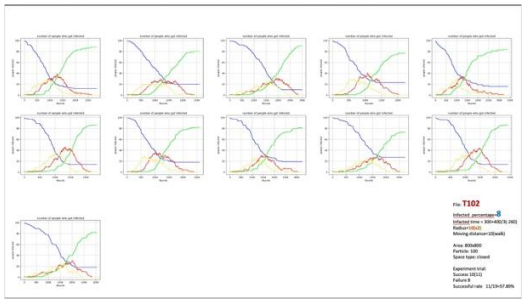


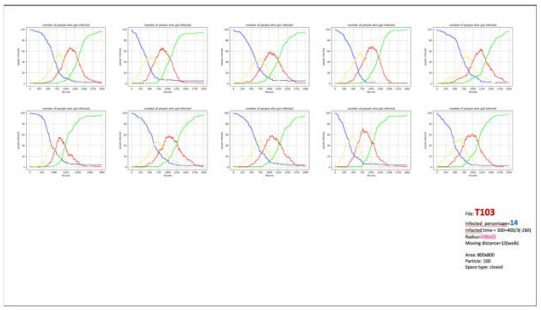


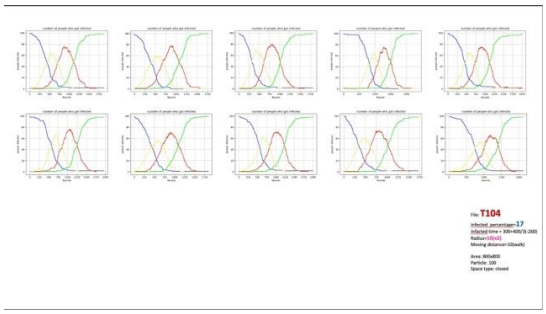


b. T10x 系列：r=10, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。

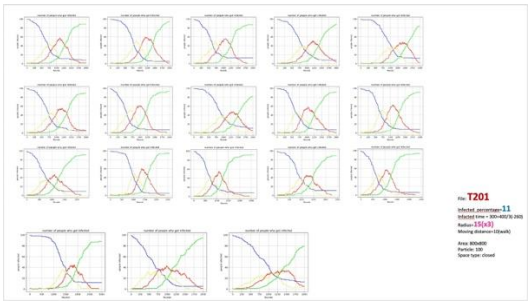


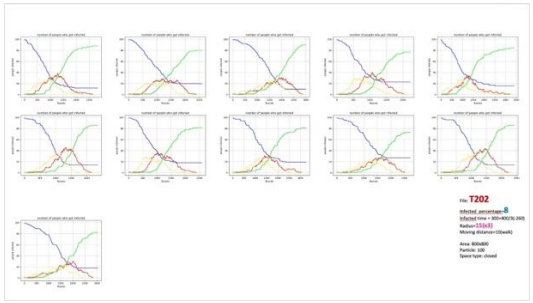


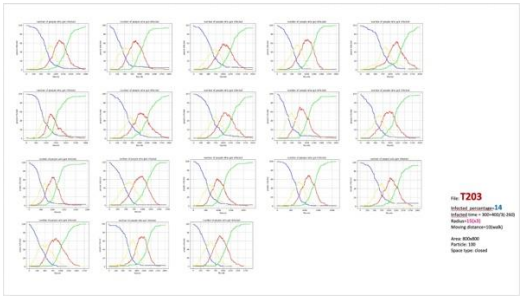


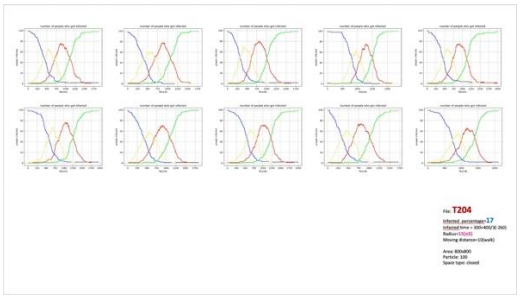


c.T20x 系列：r=15, 感染機率 8%,11%,14,17%，感染期 3+4/3 天。

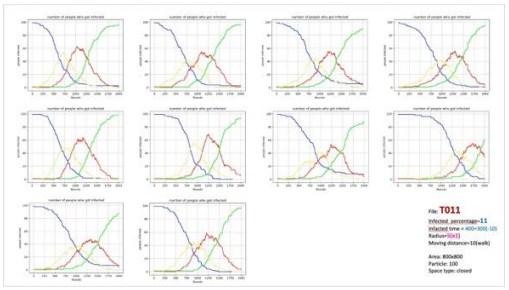


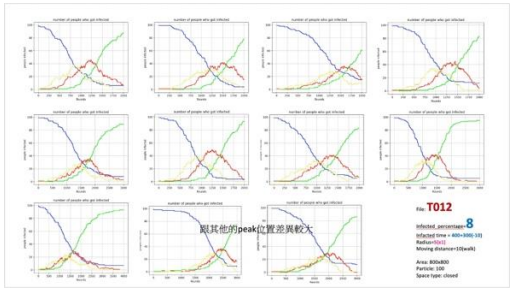


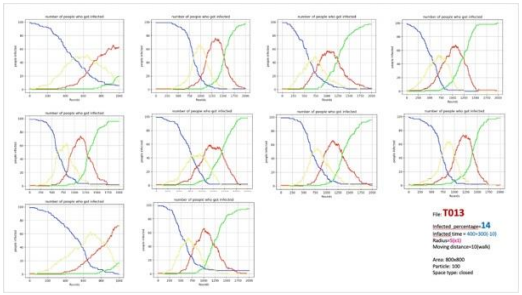


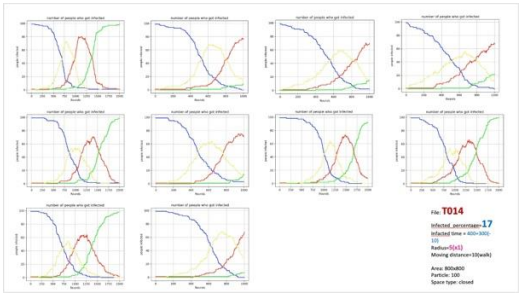


d.T01x 系列：r=5, 感染機率 8%,11%,14,17%，感染期 3+4 天。









附錄四：程式碼

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan  9 16:53:31 2023
@author: saber
"""

# -*- coding: utf-8 -*-
"""
Created on Thu Dec 23 20:40:47 2021
@author: saber
"""
import pygame
import random
import math
import time
from matplotlib import pyplot as plt

class Ball(pygame.sprite.Sprite):
    dx=0
    dy=0
    x=0
    y=0
    direction=0
    speed=0
    health=10
    def __init__(self,name,sp,srx,sry,radium,color,infected):
        pygame.sprite.Sprite. __init__(self)
        self.speed=sp
        self.x=srx
        self.y=sry
        self.infected=infected
        self.color=color
        print("a new ball is",name)
        self.image=pygame.Surface([radium*2,radium*2])
        self.image.fill((255,255,255))
        pygame.draw.circle(self.image,color, (radium,radium),radium,0)
```

```
        self.rect=self.image.get_rect()
        self.rect.center=(srx,sry)
        self.direction=random.randint(0,360)

    def update(self):
        radian=math.radians(self.direction)
        self.dx=self.speed*math.cos(radian)
        self.dy=-self.speed*math.sin(radian)
        self.x+=self.dx
        self.y+=self.dy
        self.rect.x=self.x
        self.rect.y=self.y

        #if(self.rect.left <= 0 or self.rect.right >= screen.get_width()-10):    #到達左右邊界
            #self.bouncelr()
        if self.rect.left <= 0:
            self.bounceleft()
        if self.rect.right >= screen.get_width():
            self.bounceright()
        elif(self.rect.top <= 0):    #到達上邊界
            self.rect.top = 0
            self.bounceup()
        if(self.rect.bottom >= screen.get_height()):    #到達下邊界出界
            self.rect.bottom=800
            self.bounceup()
        else:
            return False
    #def changeColorRed(self,color):
        #color=((255,0,0))
    #def changeColorGreen(self,color):

        #color=((0,255,0))
    def changeColor(self,color,radium):
        color=((color))
        self.image=pygame.Surface([radium*2,radium*2])
        self.image.fill((255,255,255))
        pygame.draw.circle(self.image,color,(radium,radium),radium,0)
```

```
        def bounceup(self):
            self.direction=360-self.direction

        def bouncelr(self):
            self.direction=self.direction-180
            #self.direction=360-self.direction
        def bounceleft(self):
            self.direction=180 + (360-self.direction)
        def bounceright(self):
            self.direction=180-self.direction
def gameover(message):
        global running
        text=font1.render(message,1,(255,0,255))
        screen.blit(text,(screen.get_width()/2-100,screen.get_height()/2-20))
        pygame.display.update()
        time.sleep(3)
        running=False

pygame.init()
score=0
font = pygame.font.SysFont("SimHei", 20)   #下方訊息字體
font1 = pygame.font.SysFont("SimHei", 32)   #結束程式訊息字體

screen = pygame.display.set_mode((800, 800))
pygame.display.set_caption("science fair stuff")
background = pygame.Surface(screen.get_size())
background = background.convert()#convert()建立副本，加快畫布在視窗顯示速度
background.fill((255,255,255))
allsprite = pygame.sprite.Group()   #建立全部角色群組
for i in range(1,2):
        x_coordinates=random.randrange(20,780)
        y_coordinates=random.randrange(20,780)
        locals()['ballname_'+str(i)]=Ball(i,10,x_coordinates,y_coordinates,10,(255,0,0),-400) #第一顆染疫的半徑
5,10,15 <<<<<<<<<<<<<<
        allsprite.add(locals()['ballname_'+str(i)])

for i in range(2,101):
        x_coordinates=random.randrange(20,780)
```

```
        y_coordinates=random.randrange(20,780)
        locals()['ballname_'+str(i)]=Ball(i,10, x_coordinates, y_coordinates, 5, (0,0,255),0)
        allsprite.add(locals()['ballname_'+str(i)])

#print(ballname_1)
#print(ballname_10)
#allsprite.add(ballname_[i])
#allsprite.add(ballname_2)
#ball11=Ball(5,225,300,10,(255,255,255))
balls=pygame.sprite.Group()
ballsinfected=pygame.sprite.Group()


#420 rounds per 3 days, 140rounds per day
allsprite.add(balls)
allsprite.add(ballsinfected)


clock = pygame.time.Clock()
f=open('sff.txt','a+')
f.write("Data: SEIR model, 800x800, 100people \n")
f.close()
msgstr = "press the left mouse to start the game"    #起始訊息
playing = False    #開始時球不會移動
running = True
totaltime=0
countdaysbounce=0 #everyone 12.43 times to caculate days
rounds=0
bouncetime=0
infectpeep=1
exposedpeep=0
exposed=[0]
timing=[0]
infecta=[1]
healthy=[99]
healthypeep=99
recoverypeep=0
recovery=[0]
roundtoend=0
```

```
#bouncetime 是總共撞了幾次 之後取平均要接近 12.43
while running:
    clock.tick(30)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    buttons = pygame.mouse.get_pressed()  #檢查滑鼠按鈕
    if buttons[0]:  #按滑鼠左鍵後球可移動
        playing = True
    if playing == True:  #遊戲進行中
        screen.blit(background, (0,0))  #清除繪圖視窗
        for i in range(1,101):
            fail=locals()['ballname_'+str(i)].update()

        for i in range(1,101):
            for j in range(i+1,101):
                collide=pygame.sprite.collide_rect(locals()['ballname_'+str(i)],locals()['ballname_'+str(j)])

                if collide==True:
                    bouncetime+=1
                    infect_percentage=random.randrange(0,100)


                    #R0=Infect_percentage(8,11,14,17) x  平均接觸人數(12.43) x  感染時間
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    if infect_percentage<=14: #if infect_percentage<=11:

                    #潛伏期三天，300x3=900
                        if locals()['ballname_'+str(i)].infected <= -260  and
locals()['ballname_'+str(j)].infected==0:
                            #infected <= -260

                            locals()['ballname_'+str(j)].changeColor((255,255,0),10)
#changeColor((255,255,0),5)  潛伏期的半徑變化  5/10/15<<<
                            locals()['ballname_'+str(j)].infected= -700  #4days +3.42days 潛伏期
                        #two days infected+1.5days exposed

                            exposedpeep+=1
```

```
                                    #infectpeep+=1
                                    healthypeep-=1


                                elif locals()['ballname_'+str(i)].infected==0 and    locals()['ballname_'+str(j)].infected
<= -260:
                                    #print("hihihihihi")
                                    locals()['ballname_'+str(i)].changeColor((255,255,0),10)
#changeColor((255,255,0),5)  潛伏期的半徑變化  5/10/15<<<

                                    #locals()['ballname_'+str(j)].changeColor((255,0,0))
                                    locals()['ballname_'+str(i)].infected= -700 #4 days 3days 潛伏期

                                    exposedpeep+=1
                                   #infectpeep+=1    #感染人數  紅色
                                    healthypeep-=1 #健康人數  藍色
            for i in range(1,101):
                if locals()['ballname_'+str(i)].infected<= -2:
                    locals()['ballname_'+str(i)].infected+=1
                if locals()['ballname_'+str(i)].infected == -400:

                    locals()['ballname_'+str(i)].changeColor((255,0,0),10) #changeColor((255,0,0),5)  潛伏期的半徑
變化  5/10/15<<<<<<<<<<<<<<<<<<<<<
                    infectpeep+=1
                    exposedpeep-=1

                if locals()['ballname_'+str(i)].infected == -1:

                    locals()['ballname_'+str(i)].infected += 11

                    locals()['ballname_'+str(i)].changeColor((0,255,0),5)
                    infectpeep-=1

                    recoverypeep+=1    #康復人數  綠色
            healthy.append(healthypeep)
            timing.append(rounds)
            recovery.append(recoverypeep)
            exposed.append(exposedpeep)
            #timing.append(totaltime)    改用一輪一輪計時
```

```
infecta.append(infectpeep)
lines=plt.plot(timing,infecta,timing,healthy,timing,recovery,timing,exposed)

plt.plot(timing,healthy,color=(0/255,0/255,255/255))
plt.plot(timing,infecta,color=(255/255,0/255,0/255))
plt.plot(timing,recovery,color=(0/255,255/255,0/255))
plt.plot(timing,exposed,color=(255/255,255/255,100/255))

plt.grid(True)
plt.title("number of people who got infected")
plt.ylabel("people infected")
plt.xlabel("Rounds")
#plt.setp(lines,marker="o")

#每 1000 個 ruonds 話一次圖表，以及最後結束時再畫一次圖表
a=5000 #原本是 1000，改成 5000 跳過 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
if rounds % a ==1:
        plt.show()

for i in range(1,101):
        fail=locals()['ballname_'+str(i)].update()

if infectpeep==0 and exposedpeep==0:
     roundtoend+=1
     print(int(roundtoend))
if roundtoend==200:
        gameover("The monitor is finished")
        plt.show()

rounds+=1
print("round:",rounds)

print(bouncetime)
averagebouncetime=int(bouncetime/100)
print(averagebouncetime)

rounds2=str(rounds)
```

```
            infectpeep2=str(infectpeep)
            healthypeep2=str(healthypeep)
            recoverypeep2=str(recoverypeep)
            exposedpeep2=str(exposedpeep)
            f2=open('sff.txt','a+')
            seq=["Rounds:",rounds2, ", Infected:",infectpeep2, ", Not-infected:",healthypeep2, ",
Recovered:",recoverypeep2, ",Exposed:",exposedpeep2, ", round-end() \n"]
            f2.writelines(seq)

            #f.write("Hello")
            #f.write(str(timing))
            #f.write(str(infecta))

            #f.write(str(healthy))
            #f.write(str(recovery))
            #f.write("round-end /n")
            allsprite.draw(screen)    #繪製所有角色
            #msgstr = "time:" + str(count)
        msg = font.render(msgstr, 1, (255,255,255))
        screen.blit(msg, (screen.get_width()/2-60,screen.get_height()-20))    #繪製訊息
        pygame.display.update()
pygame.quit()
```