基於 Boids 之三維蜜蜂模擬

彭煜博 陳品中 柯冠宇 王派軒 陳忠義
國立中山大學
b0730400{02, 10, 11, 12, 29}@student.nsysu.edu.tw

摘要

人工生命（artificial life）作為發展已久之領域，其目標旨在使用簡單規則模擬出自然界複雜的生物行為。本文介紹之蜜蜂模擬模型，啟發自人工生命中知名的 Boids，透過對其規則之增修，以及引入適當的資料結構，使我們能夠在合理的運算開銷上，近似地模擬出現實生活中蜂群的增減與環境的互動。

1 前言

蜜蜂[1][2]為一種具高度社會分工的昆蟲，一隻蜜蜂本身沒有對事物的思考能力，全憑本能行事，但當許多蜜蜂群聚在一起後，卻能展現出集體智慧（swarm intelligence）。通常來說，蜜蜂之社會分工可分為女王蜂、雄蜂和工蜂等，其中女王蜂與工蜂皆為雌蜂，唯工蜂無生殖能力。

對於女王蜂，其職責在於生育，確保整體蜜蜂之數量能穩定提升，故女王蜂終生以蜂王乳為食，使其壽命能夠長達三至五年，保證蜂群不會因女王蜂的存亡受到影響；至於雄蜂，其存在之意義僅為與女王蜂交配，不參與生產與防禦。

工蜂是蜂群內數量最龐大的，負責確保整個蜂巢的正常運作。工蜂的任務眾多，舉凡採蜜、築巢、育幼和防禦等。

成群的工蜂在外出採蜜前，會先派出數隻「偵查蜂」尋找蜜源，在偵查蜂找到合適的採蜜地點後，會帶著花蜜回巢，並透過「蜂舞」告知蜜源位置，引導其他工蜂前往採蜜。

對於蜂群而言，除去人類對環境的破壞外，最大的敵人莫過於虎頭蜂。虎頭蜂體型可達一般蜜蜂的五倍大，具有強壯的大顎和能夠使用無數次的毒針，並以蜜蜂蜂巢中的幼蟲為食，故對蜂群的生存是致命的。當蜂巢附近出現虎頭蜂時，工蜂會捨命向其衝去，利用震動產生熱能，以求將入侵的虎頭蜂殺死，保衛蜂巢。

蜂群相當適合作為人工生命的題目，透過研究單一蜜蜂的行為並簡化，我們能將其與 Boids 的概念相結合，在簡單的規則下，建立蜂群的模型，並進行模擬。

2 相關研究

Boids[3]是由 Craig Reynolds 於 SIGGRAPH '87 上提出的人工生命模型，該字由 bird 與字根 oid 所組成，意即類鳥的物體。Craig 的主要貢獻在於透過三個簡單的規則，使電腦能夠模擬出大量需要體現出集體行為的情境，如電影內出現的魚群鳥群等[4]。

三個規則分別為分離（separation）、對齊（alignment）與聚合（cohesion）。

分離的目的是避免與鄰近的個體相互碰撞，當個體越靠近鳥群，所受的分離力度就越大，而對齊的目的是使自身飛行方向與鄰近個體一致，至於聚合會讓飛行的方向朝鄰近群體的質心移動，使自己向群體靠近。

透過分別調整三個規則對於鄰近個體的判定以及個體的受力強度，可以使個體產生不一樣的行為模式。

3 方法

對於蜜蜂的模擬，我們將其行為簡化後分成四個部分，分別為蜜蜂、食物（花蜜）、捕食者（大黃蜂）以及環境，而各個部分都有自己的規則，將規則細分確定後，即以程式實現。

3.1 蜜蜂之規則

3.1.1 生命（life）

當生命數值為 0 時，表蜜蜂死亡，初值設定應大於 0。

3.1.2 位置（position）

位置為蜜蜂在空間中之座標，其初值應設定為蜂巢所在之位置，位置更新時應分別加上速度的相應分量。

3.1.3 速度（velocity）

速度為蜜蜂下一次位置更新的參考數值，其值（非分量）應恆為一常數，速度之更新需參考規則 3.1.4 至 3.1.9。

3.1.4 分離（separation）

分離即為 Boids 的 separation 規則，會使自身遠離一定範圍內之蜜蜂。

3.1.5 對齊（alignment）

對齊即為 Boids 的 alignment 規則，會使自身前進之方向對齊一定範圍內蜜蜂的前進方向。

3.1.6 聚合（cohesion）

聚合即為 Boids 的 cohesion 規則，會使自身靠近一定範圍內之蜜蜂。

3.1.7 探測（sense）

探測可分為兩個部分，在沒有獲取食物時，會探尋一定空間內有無食物存在，並朝食物前進；在獲取食物時，會朝蜂巢位置前進。

3.1.8 漫遊（wander）

漫遊會使蜜蜂隨機調整自身前進方向，其目的是使模擬蜜蜂之路徑更加接近現實情況

3.1.9 攻擊（attacking）

攻擊會偵測一定範圍內有無敵人出現，若有則向其前進，此規則使蜜蜂能夠保護蜂巢。

3.2 食物之規則

3.2.1 生命（life）

當生命數值為 0 時，表食物被蜜蜂採集完畢，初值設定應大於 0。

3.2.2 位置（position）

位置為食物在空間中之座標。

3.3 捕食者之規則

3.3.1 生命（life）

當生命數值為 0 時，表捕食者死亡，初值設定應大於 0。

3.3.2 位置（position）

位置為捕食者在空間中之座標，位置更新時應分別加上速度的相應分量。

3.3.3 速度（velocity）

速度為捕食者下一次位置更新的參考數值，其值（非分量）應恆為一常數，速度之更新需參考規則 3.3.4 和 3.3.5。

3.3.4 聚合（cohesion）

與規則 3.1.6 之行為相同，但目的不同，雖然都是尋找一定空間內之蜜蜂，並朝其前進，但此處之目的是讓捕食者顯現出捕食的行為。

3.3.5 漫遊（wander）

漫遊會使捕食者隨機調整自身前進方向。

3.4 環境之規則

3.4.1 蜜蜂的生成

在一開始需生成至少一隻蜜蜂，每當蜜蜂採集回來的總食物量到達一定值，會消耗一定量的食物生成一隻蜜蜂。

3.4.2 食物的生成

在一開始會生成一定數量的食物，當食物總量未達上限時，機率性補充食物。

3.4.3 捕食者的生成

每當蜜蜂總量達到一定數量，且未達捕食者總量上限時生成一隻捕食者，且每次捕食者生成需耗費一定時間。

3.4.4 壽命

每當蜜蜂或捕食者更新位置時，減少一點生命。

3.4.5 食物採集

每次更新位置時，若蜜蜂足夠接近食物，該食物之生命減少一點，且蜜蜂取得食物。

3.4.6 捕食

每次更新位置時，若蜜蜂足夠接近捕食者，蜜蜂對捕食者造成一定量的傷害；經過一定次數的位置更新後，若捕食者足夠接近蜜蜂，則蜜蜂的生命為 0。

3.4.7 蜜蜂的移除

當蜜蜂生命不大於 0 即被判定為死亡，需要移除。

3.4.8 食物的移除

當食物生命不大於 0 即被判定為死亡，需要移除。

3.4.9 捕食者的移除

當捕食者生命不大於 0 即被判定為死亡，需要移除。

3.5 程式實現

我們使用 JavaScript 作為本次模擬程式實現所使用之語言，並使用 p5.js 進行圖形繪製，整體程式被分成七個文件，以下逐一說明。

3.5.1 bee.js

本檔案包含類別 Bee，該類別為蜜蜂規則之實現，擁有五個成員變數，七個私有（private）方法和四個公有（public）方法。

A. 變數 life

用於紀錄蜜蜂之生命，初值定義於 global.js。

B. 變數 position

用於紀錄蜜蜂之位置，初值被設定為蜂巢之位置。

C. 變數 velocity

用於紀錄蜜蜂之速度，各個分量之初值隨機設置。

D. 變數 color

用於紀錄蜜蜂繪製時的顏色，初值定義於 global.js。

E. 變數 gotFood

用於紀錄蜜蜂有無獲取食物，初值設為 false。

F. 私有方法 _searchNeighbors

用來在給定的蜜蜂群中，尋找在一定範圍和在一定角度內的其他蜜蜂，該範圍與角度定義於 global.js 中。

G. 私有方法 _separate

用來在給定的蜜蜂群中，計算出遠離所需的速度。

H. 私有方法 _align

用來在給定的蜜蜂群中，計算出對齊其他蜜蜂所需的速度。

I. 私有方法 _cohere

用來在給定的蜜蜂群中，計算出接近其他蜜蜂所需的速度。

J. 私有方法 _sense

a. 若 gotFood 為 false，在給定食物位置的情況下，根據一距離限制（定義於 global.js），決定在該限制下，向最近食物的前進速度。

b. 若 godFood 為 true，獲得向蜂巢前進的速度。

K. 私有方法 _wander

取得一隨機的前進速度。

L. 私有方法 _attack

用來在給定的捕食者群中，根據一距離限制（定義於 global.js），決定在該限制下，向最近捕食者的前進速度。

M. 公有方法 constructor

建構子，用於初始化五個成員變數。

N. 公有方法 update

用於更新蜜蜂的速度與位置。每次更新時，該方法會先調用私有方法 F.，取得

周圍的蜜蜂群，再分別調用私有方法 G. 到 L.，並將個別結果與對應權數（定義於 global.js）相乘後，一同累加得出新的速度，最後根據新的速度更新位置，蜜蜂的位置不會超出設定的範圍。

O. 公有方法 draw

以一球體繪畫出蜜蜂在空間中的位置。

P. 公有方法 isDead

回傳蜜蜂是否死亡。

3.5.2 food.js

本檔案包含類別 Food，該類別為食物規則之實現，擁有三個成員變數和四個公有方法。

A. 變數 life

用於紀錄食物之生命，初值定義於 global.js。

B. 變數 position

用於紀錄食物之位置，位置採隨機產生，唯其高度值分布於 0.25 到 0.4 倍設定高度。

C. 變數 color

用於紀錄食物繪製時的顏色，初值定義於 global.js

D. 公有方法 constructor

建構子，用於初始化三個成員變數。

E. 公有方法 draw

以一球體繪畫出食物在空間中的位置，並會在食物到地板間以一圓柱體作為植物之莖部。

F. 公有方法 isDead

回傳食物是否耗盡。

3.5.3 global.js

本檔案包含大量預先定義之常數與變數，用來控制模擬條件。

3.5.4 index.html

本檔案用來啟動整個程式。

3.5.5 octree.js

本檔案透過類別 Cuboid、OctreeNode 和 Octree 來實現八元樹（octree）[5]，藉由預先將所有點（如蜜蜂）加入至八元樹中，再透過搜索八元樹來獲取一點在一定範圍內之鄰居，相較於直接的暴力解法有著更佳的表現。

在本模擬中有大量操作需要搜尋一點在一定距離內之鄰居，引入八元樹能夠減少搜尋的開銷，達到改善模擬效能的目標。

表一、暴力法與八元樹之時間複雜度比較

操作	暴力法	八元樹
樹的建立	-	$O(nlog_8 n)$
每次搜尋	$O(n)$	$O(log_8 n)$
整體耗費	$O(n^2)$	$O(nlog_8 n)$

3.5.6 predator.js

本檔案包含類別 Predator，該類別為捕食者規則之實現，擁有四個成員變數、兩個私有方法和四個公有方法。

A. 變數 life

用於紀錄捕食者之生命，初值定義於 global.js。

B. 變數 position

用於紀錄捕食者之位置，初值被設定為蜂巢對角之位置。

C. 變數 velocity

用於紀錄捕食者之速度，各個分量之初值隨機設置。

D. 變數 color

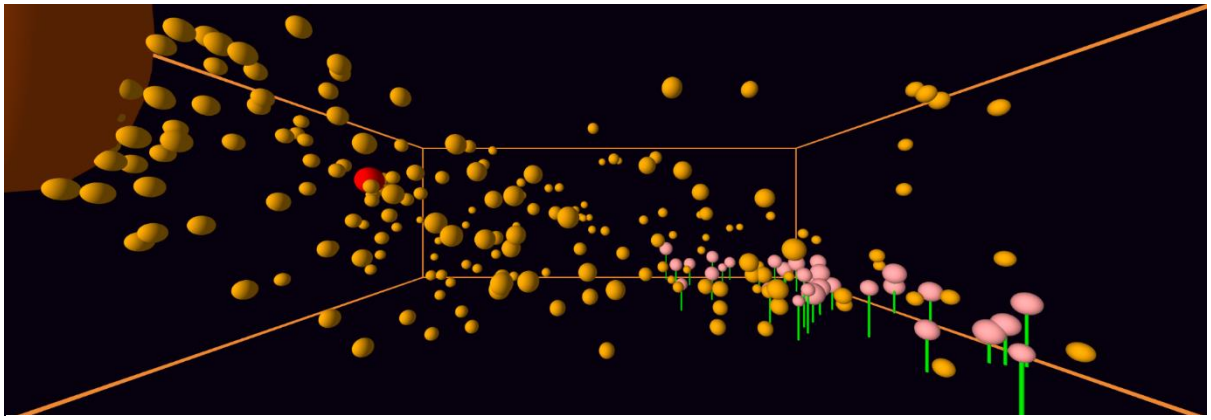
用於紀錄捕食者繪製時的顏色，初值定義於 global.js。

E. 私有方法 _wander

取得一隨機的前進速度。

F. 私有方法 _cohere

來在給定的蜜蜂群中，計算出接近蜜蜂所需的速度



圖一、程式模擬運行畫面

G. 公有方法 constructor

建構子，用於初始化四個成員變數。

H. 公有方法 update

用於更新捕食者的速度與位置。每次更新時，該方法會先調用私有方法 E.和 F.，將結果與相應權數（定義於 global.js）相乘後累加得出新的速度，最後根據新的速度更新位置，捕食者的位置不會超出設定的範圍

I. 公有方法 draw

以一球體繪畫出捕食者在空間中的位置。

J. 公有方法 isDead

回傳捕食者是否死亡。

3.5.7 sketch.js

本檔案負責進行繪圖和執行環境規則，環境規則之執行流程如下：

1. 判定蜜蜂是否攜帶食物回到巢穴
2. 執行規則 3.4.4
3. 執行規則 3.4.5
4. 執行規則 3.4.6
5. 執行規則 3.4.7
6. 執行規則 3.4.8
7. 執行規則 3.4.9
8. 執行規則 3.4.1
9. 執行規則 3.4.2
10. 執行規則 3.4.3
11. 更新各個蜜蜂和捕食者的位置

4 實驗

本程式模擬之蜜蜂可以視作生活在箱中世界，在此世界裡有食物重生機制和捕食者。

圖一背景的橘色框線即是箱子的邊，而左上角的橘色橢球表示蜂巢，圖中大量的黃色球體為蜜蜂，紅色的球體是正在攻擊蜜蜂的捕食者，至於粉色有著綠色桿的即是蜜蜂所要採集的食物。

以下介紹一些我們在不同參數設定下所觀察到的情況。由於可供調節的參數眾多，先給定一些重要參數的設定，若之後的設定沒有特別提到，皆按照表二之設定，表二未列出之參數，則按程式本身設定。

表二、重要參數之設定

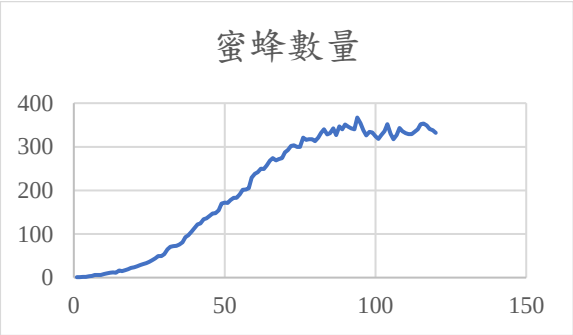
參數名	數值
FOOD_REGEN_PROB	0.01
VEL_LIMIT	8
SEP_MULTIPLIER	0.07
ALI_MULTIPLIER	0.07
COH_MULTIPLIER	0.07
SEN_MULTIPLIER	0.1
WAN_MULTIPLIER	0.25
ATK_MULTIPLIER	0.8
NUM_BEE	1
NUM_FOOD	80
NUM_ PREDATOR	10
LIFE_BEE	1500

4.1 邏輯（logistic）型增長

將 NUM_PREDATOR 設定為 0，FOOD_REGEN_PROB 設定為 1，LIFE_BEE 設定為 800。

以上參數表示不會有獵食者生成，箱中食物永遠不會耗盡。此時蜜蜂數量對時間的曲線可以視為邏輯函數，並能觀察到蜜蜂量與食物量達到平衡。

表三、蜜蜂數量對時間的函數（120 秒）

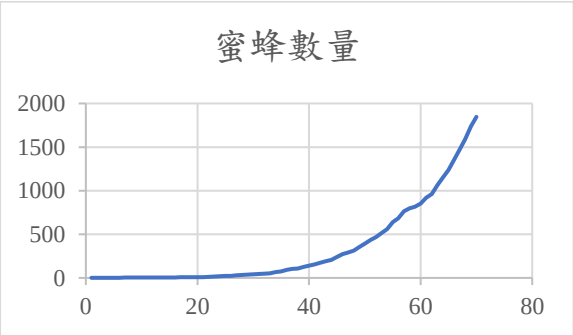


4.2 指數型增長

將 LIFE_BEE 設為 10000，NUM_FOOD 設為 50，其餘設定與邏輯型增長相同。

由於蜜蜂壽命相當長，且食物不虞匱乏，蜜蜂總量呈指數型增長，表四因蜜蜂數量過大，運算吃力，僅紀錄至 70 秒。

表四、蜜蜂數量對時間的函數（70 秒）



4.3 天體運動

將 NUM_FOODS 設為 1，VEL_LIMIT 設為 15，SEP_MULITIPLIER、ALI_MULTIPLIER、COH_MULITIPLIER 和 ATK_MULITIPLIER 設為 0，SEN_MULTIPLIER 設為 0.15 和 WAN_MULITIPLIER 設為 0.01。

以上設置會使蜜蜂靠近食物時，會有相當高的機會圍繞食物進行圓周運動，類似於月球繞著地球轉，若蜜蜂攝取到食物，可以類比為彗星撞地球，與天體的互動相似。另外，當多個蜜蜂繞行食物時，看起來會與電子圍繞原子核旋轉一樣。

4.4 分子運動

將 ALI_MULTIPLIER、COH_MULTIPLIER、SEN_MULTIPLIER 和 ATK_MULTIPLIER 設為 0，SEP_MULTIPLIER、WAN_MULTIPLIER 設為 1，NUM_BEE 設為 500，LIFE_BEE 設為 10000，NUM_PREDATOR 設為 0。

以上設置會使 500 隻蜜蜂在一定時間後均勻分布於箱子內，且彼此間保持一定距離，如同液態水中的分子運動。

4.5 完全彈性碰撞

將 ALI_MULTIPLIER、COH_MULTIPLIER、SEN_MULTIPLIER、WAN_MULTIPLIER 和 ATK_MULTIPLIER 設為 0，SEP_MULTIPLIER 設為 1，NUM_BEE 設為 50，LIFE_BEE 設為 10000，VEL_LIMIT 設為 20。

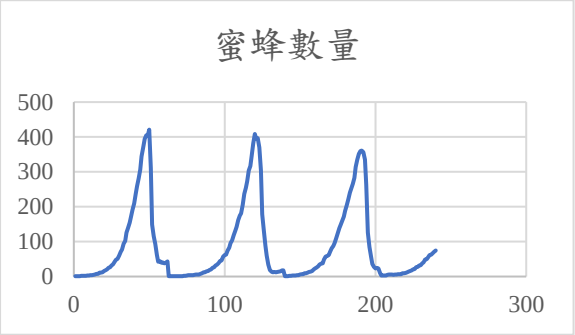
我們在此將蜜蜂視為球，以上設置會使箱內擁有 50 顆球，且球與球和球與箱子間的碰撞都為完全彈性碰撞，有如在箱子內打三維撞球一樣。

4.6 天敵攻擊

按程式原始參數運行，即是模擬蜜蜂與大自然的互動，由於加入掠食者，蜜蜂除了自然死亡外，又有遭捕食者攻擊的可能，生存環境相較於如 4.1 或 4.2 等理想條件下來的惡劣，蜜蜂數量起初能如指數般的增長，但在遇到捕食者的攻擊後，能夠採集的蜜蜂數量會相應變少，代表生成

的蜜蜂數量也會降低，而可供採集的食物量也不會如模擬開始時一樣豐富，故蜜蜂的生成速度也相應降低，因此在強敵侵襲的狀況下，整個蜂群能否活下來，就只能指望自身的運氣。

表五、蜜蜂數量對時間的函數（240 秒）



在表五中，蜜蜂總共遭到三次捕食者的攻擊，每次攻擊後蜂群都相當的幸運，恰好存活了個位數的蜜蜂，沒有導致蜂群的覆滅。實際上我們運行的相當多次的模擬，而其中的大多數在第一波攻擊就已全軍覆沒。

5 結論

本文由 1986 年提出的 Boids 模型開始，利用觀察蜂群，在基於 Boids 的概念上，將蜜蜂與大自然之互動簡化為四大部分，並各自建立規則，在透過程式實現後，得以藉由參數設定之不同，得到不同的模擬行為。

由於規則的可擴充性，可以透過新增規則來使模擬更加接近現實情況，並且目前參數之設定要透過手動更改 global.js 來達成，日後可以考慮透過滑動條或輸入框等方式，令使用者能更加容易地設定模擬條件。

參考資料

[1] Wikipedia contributors. (2020, October 8). Bee. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:25, November 7, 2020, from https://en.wikipedia.org/w/index.php?title=Bee&oldid=982529805

[2] *蜜蜂生態*. (n.d.). Retrieved 18:52, November 7, 2020, from https://sites.google.com/site/mifengdefengwo/mi-feng-sheng-tai-1

[3] Craig Reynolds. (2001, September 6). *Boids: Background and Update*. Retrieved 10:40, November 7, 2020, from https://www.red3d.com/cwr/boids/

[4] Wikipedia contributors. (2020, October 16). Boids. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:48, November 7, 2020, from https://en.wikipedia.org/w/index.php?title=Boids&oldid=983748782

[5] Wikipedia contributors. (2020, September 13). Octree. In *Wikipedia, The Free Encyclopedia*. Retrieved 15:33, November 7, 2020, from https://en.wikipedia.org/w/index.php?title=Octree&oldid=978159387