# 將模擬系統移轉至視覺化程式語言環境之建置對照—以 StarLogo TNG 為建置工具

**林均恒 洪文麟 張智凱**
**國立臺南大學 數位學習科技學系**
**chihkai@mail.nutn.edu.tw**

## 摘要

模擬學習(simulation-based learning)與「從設計中學習」理論皆提出，學習者在自行進行操作或設計的過程中，可高度融入學習情境，以個人方式累積經驗，將有助於所學知識的靈活運用。基於以上觀點，NetLogo 豐富而完整的模型庫可視為良好的學習教材，其中每個模擬系統都被清楚地分類且橫跨多個領域，但由於 NetLogo 裡的模擬系統是以文字型程式語言(text-based programming)進行編寫，對於程式能力不精熟的學習者，需花費時間在克服程式識讀與編寫的問題。為了降低學習者程式能力的門檻，本研究選擇以具視覺化程式編寫環境的 StarLogo TNG 作為模擬系統建置工具，將 NetLogo 中的模型進行移轉，並整理出移轉工作可依循的對照及轉換方針，搭配實際移轉範例做說明，期望藉此提供於 NetLogo 中遭遇程式問題的學習者另一個建置模擬系統的管道。

**關鍵詞**：模擬學習、模擬系統、NetLogo、StarLogo TNG、模型移轉

## Abstract

In light with two theories of simulation-based learning and learning by design, learners are expected to enjoyably participate in learning task and developed a high order thinking skills to apply the prior knowledge they acquired to solve the problems. Based on the above concept, NetLogo which provides rich models library could be seen as a great learning materials. There're many simulations being specifically classified and the types include variety of fields. However, learners are required to dedicate themselves to master programming because NetLogo is a platform of text-based programming that low proficiency programmer needs to spend lots of time editing and debugging code. In order to lower the threshold of programming, the research adopted a visual programming language StarLogo TNG to transfer NetLogo models and listed principles and guidelines for transferring. Through a demonstration from a transferring example, the research offered an alternative for low-proficiency learners of programming to establish a simulation-based learning system.

**Keywords**: simulation-based learning, simulation, NetLogo, StarLogo TNG, simulation transferring.

## 1. 前言

目前視覺化程式語言相較於一般的程式語言在編寫上雖然較缺乏彈性，但其程式的排列及編輯方式能夠讓即使不是電腦相關科系的學習者亦可輕易理解程式內容與結構。因此在以學習為主要目標的基礎上，選擇以視覺化程式語言做為模擬系統建置的環境，且在模擬系統建置的過程，將欲模擬的現象進行分析，並轉化為抽象概念，嘗試以程式語言將其完整地表達出來，接著運作模擬系統，不斷地進行測試、修正符合真實情況，在每一次的程式修改中，都將更清楚了解程式編寫的結構、規則以及實際問題抽象化的技巧，重複地面對問題、解決問題，將進行處理的經驗內化，進而應用於生活中，達到融會貫通的學習境界。

## 2. 文獻探討

### 2.1 模擬學習(Simulation-Based Learning)

根據體驗式學習理論(experiential learning theory)，藉由實作、思考並以自身經驗的方式進行學習將可以得到較好的學習效果[1]，學習者是主動地、完全地投入在學習的過程中[2]，以自己的學習方法吸收經驗。在 Kolb 的體驗學習圈(experiential learning cycle)中將學習過程分成四階段：具體經驗、反思性觀察、抽象概念化、主動進行實驗[3]。

而模擬系統的建置與運用可分為以下六步驟[4]：分析特定系統、發展概念雛形、將雛型轉為程式架構、完成初步模擬系統、進行實驗驗證模擬系統、進行模擬以解決問題或找出原因。對照後可發現模擬系統建置與運用的部分過程恰能與體驗學習圈的四個階段相互呼應。

學習者由實際情況進行觀察分析，再依據分析的結果發展出模擬的概念雛型，經歷了具體經驗與反思性觀察兩階段；接著將其中概念抽象化，設法以程式指令表達，進而完成模擬系統，進入了抽象概念化階段；且在模擬系統中，學習者可以調整變因，運行模擬系統，即為主動進行實驗；由不同的

變因組合觀察到模擬的可能結果，同時亦檢視模擬系統是否合理，再對模擬系統的設計進行微調，回到了具體經驗與反思性觀察的階段；最後參考模擬結果對實際情況做出相應行為，此乃為學習效果的呈現。經歷這些過程使得學習者充分將接收訊息消化，更能將所學融會貫通。

### 2.2 從設計中學習(Learning by Design)

為使學習者有能力去融會貫通學習內容並進行應用，在過去有研究對學習者日常行為模式的產生進行相關資料的蒐集，發現在處理生活中的各項事務時，第一時間通常會從以往的記憶中找尋類似事件，根據經驗著手進行思考解決方式，即為案例式推理(Case-Based Reasoning)的基本原理[5]，同時也是問題導向學習(Problem-Based Learning)理論的目的[6]。

「設計」是提供實際經驗的過程，需要學習者完全地投入思考，其中會使用到分析問題、蒐集資料、訂定方法、評估可行性等技巧，最後找出最好的解決方法[7]，Gee 認為設計能促使學習行為的發生[8]。「從設計中學習」理論依據以上所述，希望學習者在找出問題答案的過程中，運用「設計」發展出解決方法。

案例式推理與問題導向學習的概念融合在「從設計中學習」的理論中呈現[9]，以學習者的角度出發，引導學習者主動、積極進行學習活動，Kolodner、Gray 以及 Fasse 所提出 LBD's Cycle[10]的活動包含兩個部分：設計／再設計(design/redesign)與研究&探討(investigate & explore)。

設計部分由瞭解挑戰、運用科學知識進行規劃設計、驗證、建構與測試、分析與解釋、展示與表達想法，這幾個程序進行循環，而研究部份的循環則是由清楚明白疑問、做出假設、設計調查、指導調查、分析結果、海報展示所組成。兩個循環間的關係為：從設計的結果得到知識與面臨問題而進行研究，再將研究的成果實際應用進行規劃設計。在 LBD's Cycle 中可見，「設計」在學習行為中扮演著重要的角色，與研究的過程相輔相成。

「設計」的動作可以使學習者全心投入學習的狀態中，而在此同時，學習者也能藉由「設計」的行為，以自己的方式累積個人經驗，使學習者能靈活運用所學的成果，擺脫傳統教學的單一性(即由教師主導的單一教學方式)，達到更佳的學習效果。

### 2.3NetLogo 與 StarLogo TNG 之比較

先前所提到的模擬學習理論表示，由模擬系統的建置進行模擬現象的學習能有較深刻的學習經驗。因此模型庫中具有眾多模擬系統且橫跨多個領域[11]的 NetLogo 便被納入考量中，NetLogo 是以 Java 為基底所開發，由美國西北大學 The Center for Connected Learning(CCL) and Computer-Based Modeling 所開發的模擬建置環境，以 Logo 語言進行編寫，多被使用於建置須長時間觀察或結果不可逆的模擬系統[12]，使用者可依自己的需求在環境中建立模型進行模擬。

由 MIT Scheller Teacher Education Program 所開發的 StarLogo TNG 在程式編輯方式與 NetLogo 全然不同，改以視覺化程式語言進行模擬系統的程式編寫，發展概念乃基於以下兩點：(1)降低使用者建立模擬系統的門檻，以及(2)增加模擬場景的生動感[13]。各式指令以不同顏色及形狀的程式塊(programming blocks)替代，運用拼圖拼接、拖拉的方式來編輯組合，爾後在其程式編輯視窗(Workplace)中依背景分類區塊進行擺放，使得即使不精通程式編輯的人也能輕易進行程式編寫，而模擬系統則以 3D 方式呈現，除了符合現今視覺傳達的趨勢，亦給予模擬建置者一個機會更精確地呈現模擬系統，地面可進行高低起伏的設計，生動的場景呈現更能吸引目光，提升使用者的興趣；學習者能利用可從任意角度檢視的鏡頭，更清楚地進行觀察。

NetLogo 與 StarLogo TNG 皆以 StarLogo 為基礎發展而來[14][15]，繼承 StarLogo 大部分的特性，NetLogo 承接了 StarLogo 的簡單語法[16]，程式編寫較具彈性，因多了跨平台之便利性，大幅降低了系統上的限制問題，使得大多數使用者選擇以 NetLogo 建置模擬系統，而 StarLogo TNG 則一改 StarLogo 原有的文字型程式語言(text-based programming)，轉而發展視覺化程式語言，以具有繽紛色彩的程式塊替代了文字型指令，但由於同承接自 StarLogo 系列且兩者之開發目的相同，因此在設計上仍有許多雷同之處，如：部分指令的名稱、使用介面的形式，以及皆提供多個 Agent 同時運作的功能，這些使得使用者在操作上只要熟悉其中一個環境，對另一個就不會感到陌生。

本研究以學習為主要考量，為使學習者即使不精通程式編輯也不受影響，視覺化程式語言具有的幾項屬性：具體性、直接性、清晰性和立即性[17]，以及不必費心思顧慮指令拼法與程式語法的編寫過程可以減少學習者在建立模擬系統時程式編輯上的困難，而當學習者進行模擬系統的觀察及程式識讀時，以 3D 方式呈現模擬系統再搭配色彩繽紛且按照分類排放的程式指令，不但能清楚表現模擬系統，亦可提升學習者的興趣。

有鑑於以上原因，本研究選擇 StarLogo TNG 做為模擬系統建置的工具，但發現目前在 StarLogo TNG 的模型庫中可供使用的模擬系統範例仍不足，多需使用者自行建置，在模型庫豐富的 NetLogo 則因為文字型程式編寫方式而有可能產生編寫與識讀上困擾的問題(表 1)，為此本研究決定從 NetLogo 移轉模型至 StarLogo TNG，並於建置的經驗中為使用者整理出規則，供使用者於模擬系統建置過程中參考使用。

**表 1 Starlogo TNG 輔助解決 Netlogo 使用上之問題**

| Netlogo 使用上所遭遇的問題 | 問題描述 | Starlogo TNG 可提供的輔助 |
|---|---|---|
| 需要具有程式語法的先備觀念以及留心程式的拼字細節 | Netlogo 仍是以文字型程式語言(text-based programming)進行程式編寫，使用者需花時間了解語法結構，並且注意指令拼法，以避免小小的拼字錯誤造成整個模擬系統無法運行。 | 以視覺化的程式塊為最小單位，且不同類型的程式塊有特定的接口形狀，符合形狀的連結才可接上，不必費心於注意程式結構及拼字上的錯誤。 |
| 程式碼識讀不易 | Netlogo 的程式編寫十分彈性，但程式碼若未經整理並進行註解，則在識讀上易造成混亂，對程式初學者更是一大障礙。 | Starlogo TNG 的程式編輯視窗(Workplace)有針對各個 agent 及運行階段進行區域劃分，編寫者可依此排列程式，識讀者可依此理解程式。 |
| 必須自行創建與連結模擬系統的控制項(control) | Netlogo 中需在介面上自行產生控制項，再進行程式碼的連結設定，若連結錯誤指令將影響系統運行結果。 | 於程式編寫中自動產生相應控制項。 |

## 3. 移轉動機與方針

NetLogo 與 StarLogo TNG 皆可將環境分成兩個區塊：程式編寫及模擬呈現，程式編寫區塊即學習者主要學習程式概念的部分，亦為影響模擬系統可否順利運作的核心，StarLogo TNG 中此區域在左側設有程式指令的分類，點擊後可出現各分類下的程式指令；在模擬呈現區塊中學習者可以看到自己編寫出來的模擬系統，藉由調整控制項，觀察運行結果及數據結果來輔助對於程式部分的理解。以下將針對這兩部分，分別解釋指令移轉對照與控制介面轉換方法，最後輔以流浪狗範例進行解釋說明。

### 3.1 指令對照

StarLogo TNG 與 NetLogo 大多數的指令名稱相同，此與兩者之發展皆參考自同一軟體 StarLogo 有關，指令的命名顯現了簡短、直覺的特點，主要是希望使用者不必在理解指令的過程中花費太多心力。StarLogo TNG 中可進行相似對照的指令分類主要有 Setup & Run、Movement、Logic、Trait 與 Math。

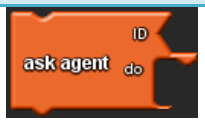
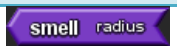
在 Setup & Run 分類中的指令會直接對模擬呈現介面(Spaceland)造成改變，包含按鈕與圖表的產生，這部分也是 StarLogo TNG 的特點之一，特別規劃程式塊供使用者進行模擬系統的初始條件設定(Setup)與運行過程設計(Run)，此分類中只有與清除物件相關的指令能進行相似對照，例如：clear all(clear-all) 、 clear patches(clear-patches) ； 而 Movement 分類中的指令主要在使 Agent 產生位置移動，除去 NetLogo 中沒有的三維空間之上下移動指令，其餘基本的指令 forward(forward)、towards(towardsxy x y)以及 set heading(set heading)等皆有相似指令可進行呼應。

Logic 分類中的指令以邏輯判斷式為主，為程序間進行連接的重要指令，大多數的程式塊都能找到相似的指令，譬如：if(if)、and(and)、true(true)、hatch(hatch)，少部分程式塊即使無相似對應，仍能由其他程式塊組合而成生相同的效果，例如 StarLogo TNG 的 switch 指令於 NetLogo 中無相似對應，而其功能為判斷多個性質相同的不同條件，依條件分別進行相應的動作，此即為結合多個 if 指令的結果。

Trait 分類，顧名思義即為對各項 agent 的特性做調整的指令集，拿掉對於 agent 高度(altitude)與場景轉換(level)的設定，其他指令幾乎都能相應，至於 Math 分類，其中包含所有數學相關的指令，例如：數字、運算子與各類常見之數學符號(sqrt、abs、log 等)，每一個程式塊都能找到近乎完全相同的指令。

少部分的指令因視覺化程式塊單位內容的設計，使得程式指令名稱產生差異，再加上 StarLogo TNG 在模擬系統的呈現上較 NetLogo 多了一個維度的呈現方式，甚至還可以自由調整鏡頭，從不同角度呈現模擬結果，因此 StarLogo TNG 的指令還多了 Terrain 分類與 Control 分類，Control 內部的指令可用來控制鏡頭，而 Terrain 分類裡主要是控制地形的起伏，另有與地面相關之判斷式以及場景轉換動作(level)的指令。表 2 內容即針對部分相異但仍可以進行對照的指令做整理。

表 2 部分相異指令名稱對照範例

| 相異 | |
|---|---|
| **StarLogo TNG** | NetLogo |
| **Setup & Run** | |
|  | set xy random-xcor random-ycor |
|  | to *setup*<br>*cmds*<br>end |
| **Procedure** | |
|  | to *name*<br>*cmds*<br>end |
| **Terrain** | |
|  | pcolor |
| **Logic** | |
|  | ask *agent* [*cmd*] |
| **OtherAgents** | |
|  | in-radius *num* |
| **Colors** | |
|  | one-of base-colors |

### 3.2 模擬系統之控制介面轉換

除了程式的編寫外，模擬系統的呈現與互動亦是本研究重點之一，在 StarLogo TNG 中，各個模擬系統的控制項會經由程式的編寫相應產生，而於 NetLogo 中則須自行於介面上創建，再與程式指令進行連結。NetLogo 常用的模擬控制項目(control)有：Button、Slider、Switch、Chooser、Monitor 與 Plot。以下將針對 NetLogo 各控制項列出使 StarLogo TNG 產生相應項之方法。

Button 通常應用在模擬系統的初始設定與運行，在 StarLogo TNG 中選擇 setup、forever、run、run once 皆能自動於介面出現按鈕，可於程式編輯時變更按鈕名稱，運行部分的按鈕在運行其間會發亮，而當 forever 按鈕發亮時只須再點選一次即可結束其運行的狀態。

Slider 用於調整輸入數值，StarLogo TNG 中有相應程式塊 slider，接上程式塊 shared number，便可於控制面板產生 Slider，程式塊 shared number 的命名即為控制面板上 Slider 的名稱，同時亦會產生一組能使此變數供其他程序使用的程式塊，另外，Slider 的數值範圍可直接於控制面板上進行設定。

當決定某項情況是否發生或是否納入考慮時可使用 Switch，在 StarLogo TNG 中可利用 run once 結合 set shared boolean 造成相同效果，要產生 set shared boolean 前必須先宣告一個 shared boolean，宣告方法為拉出一個 shared boolean 的程式塊放置在任意區塊，最後對 run once 程式塊重新進行命名，以點選按鈕的方式對預先設定的布林值做改變，來達到與 NetLogo 的 Switch 一樣的結果。

Chooser 可藉由下拉選單設定不同情況，於 StarLogo TNG 中並無發現相應的程式塊，它的移轉方式與 Switch 十分相似，只是把 set shared boolean 改成執行 procedure，procedure 內為該情況下之條件設定，這樣的作法即為把 Chooser 中的各種情況分成多個按鈕，藉由按鈕的點選造成與選取動作相同的效果。

Monitor 與 Plot 為數據呈現的部分，Monitor 會產生顯示數值的方塊於控制介面，有相應程式塊 monitor，Plot 則產生數值動態變化之圖表，亦有相應程式塊 line graph、bar graph，皆與內容形式為數值之程式塊結合即可，其中 monitor 只能接一個程式塊，而 line graph 及 bar graph 可與多個數值結合。

### 3.3 移轉範例

為更清楚解釋指令對照與控制介面的轉換，此節將以流浪狗模擬系統[18]的移轉為範例，對照其於 NetLogo 與 StarLogo TNG 環境下的程式碼以及產生之結果(圖 1 與圖 7)，並對上兩節所整理出來的移轉方法進行對應與說明。

圖 1 NetLogo 流浪狗問題模擬系統

建置一個模擬系統，首先要對其模擬環境進行設定，在 StarLogo TNG 中，特別為系統設置部分設計了 setup 程式塊(如表 5 之內容所示)，而在執行部分另有 forever、run、run once，這些程式塊會自動於模擬呈現介面(Spaceland)產生相應的控制項，供使用者控制模擬系統，此部分可參考 3.2 的 Button。

NetLogo 中部分程式指令會被 StarLogo TNG 進行改寫或結合成一單位的程式塊，如圖 2 的對照所示，將左邊 NetLogo 的程式指令(設定 X 與 Y 座標為任意 X 座標、Y 座標)簡短地以 scatter 代表，這部分可參考表 2 的相異指令名稱對照表。

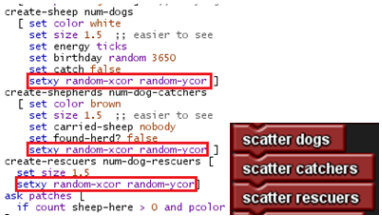



**圖 2 隨意散布 agent 的程式寫法對照**

完成了模擬系統設定，接下來要對系統的事件進行設計。經觀察發現，NetLogo 中程式架構主要以函式(function)組合而成，由 to 後接函式名稱(圖 3 中函式名稱為 wiggle)為一個函式的開頭，最後以 end 結束函式內容，需要運用該函式時，在執行函式中寫入函式名稱即可。


**圖 3 wiggle 函式對照**

我們可以根據事件，將內容分成多個函式，圖 3 的 wiggle 函式可使物件進行左右擺動的行為，在 StarLogo TNG 中，可以使用 procedure 這個程式塊進行函式編寫，每個 procedure 會自動產生如圖 3 右上角的程式塊，可與執行部分的程式塊連結，讓 procedure 中的內容得以運行。

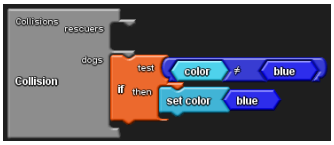



**圖 4 StarLogo TNG 中偵測碰撞的程式塊**

流浪狗問題模擬系統中有一事件為動保團體將流浪狗結紮並於原地放養，在此事件中會使用到 StarLogo TNG 的程式塊 Collision，碰撞偵測，動保人士一旦接觸到狗便會替狗結紮，圖 4 範例中將已結紮的流浪狗顏色設為藍色。程式塊 Collision 為前面所提到，NetLogo 程式指令被 StarLogo TNG 結合成一單位程式塊的例子。

在 NetLogo 中解決碰撞偵測問題時，會倚靠 patch 進行判斷，以指令 *agent*-here 編寫，對周遭 8 塊 patch 上的 agent 進行偵測，但是在 StarLogo TNG 裡，會依所建立的 agent 項目，自動倆倆產生一個 Collision 程式塊，使用者只需要選擇進行碰撞的對象，而後對碰撞所造成的結果進行設定即完成。


**圖 5 Slider 控制項之對照與 StarLogo TNG 中 Slider 程式塊**

除了程式部分，可與模擬系統進行互動的控制介面也是建置的重點，圖 5 下方紅色區塊為編輯 Slider 的程式塊，須與粉紅色的 shared number 程式塊結合才能出現對應的 slider 於控制面板上(圖 5 中)，可做數值變化的調整，面板上 slider 的命名將隨著粉紅色程式塊的名稱做變化。

相對於 StarLogo TNG 的自動產生控制項，NetLogo 必須要在控制面板上自行創建完 Slider(圖 5 上方)後，於其上按右鍵，叫出編輯視窗，輸入參數名稱以及設定數值範圍，須注意在程式碼編寫部分要對參數名稱進行全域變數宣告，否則便無法進行連結，以上可對照著 3.2 的 Slider 部分進行理解。

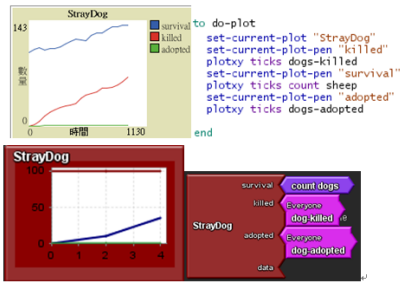



**圖 6 折線圖對照**

模擬系統中產出的各項數據亦是分析的素材，StarLogo TNG 在這方面亦有能自動產生相應圖表的程式塊，而 NetLogo 同樣需要再從編輯視窗中進行程式連結以及各數據呈現方式的設定。從圖 6 的呈現，StarLogo TNG 在程式碼的編寫上明顯地較 NetLogo 簡單、直覺。圖 6 的內容為流浪狗模擬系統之控制介面的折線圖，可參閱 3.2 的最後一段 Plot。

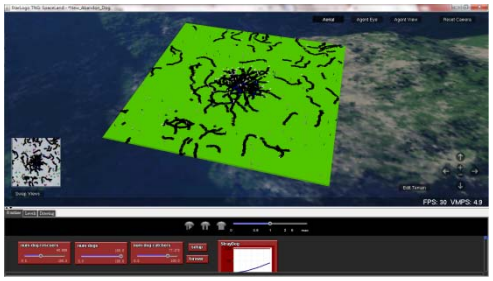


**圖 7 StarLogo TNG 流浪狗問題模擬系統**

由模擬環境設定開始，接著將模擬系統中的事件內容分割成一段段函式進行編寫，而後產生能影響系統的控制項以及模擬結果呈現的圖表。3.3 之內容以流浪狗問題模擬系統作為輔助，依據建置模擬系統的大致流程進行說明，希望可以幫助使用者了解 3.1 與 3.2 所整理之規則，在移轉的過程中更迅速、更輕鬆。

## 4. 結論與未來展望

由模擬學習(Simulation-based Learning)與從設計中學習(Learning by Design)兩個理論可推知，學習者能藉由建置與使用模擬系統，在其中投入高度的專注力，進而達到將所學習之概念內化的效果；另一方面，有鑑於視覺化程式語言的易讀性以及程式編寫上的容易性，搭配著色彩繽紛的程式塊，還有立體的模擬呈現方式，可以增加使用者在學習上的樂趣。根據以上兩點，本研究將 NetLogo 中的模型移轉至 StarLogo TNG 中建置，並將移轉過程中所獲得之經驗進行整理，產生部分指令的對照範例以及模擬系統之控制項目的轉換方法，同時帶入流浪狗模擬系統範例，對移轉方式進行解釋，期望以上移轉規則能幫助其他曾經在 NetLogo 的使用中遭遇程式障礙的學習者，轉而以 StarLogo TNG 做為模擬系統建置工具，為他們提供一份可依循的模式，便於完成模擬系統的移轉，讓 StarLogo TNG 中的模擬系統可以日漸豐富，更進一步使得學習者在 StarLogo TNG 中進行更有效率與輕鬆的模擬學習。

## 致謝

本研究承蒙臺灣行政院國家科學委員會專題研究計畫贊助，計畫編號：NSC 100-2628-S-024-001-MY3，以及計劃贊助，計畫編號：NSC 101-2511-S-024-007-MY2。

## 參考文獻

[1] Fanning, R. M., & Gaba, D. M. (2007). The role of debriefing in simulation-based learning. *Simulation in Healthcare, 2*(2), 115-125.

[2] Horton, W. (2012). Games and Simulations: Learning by Playing and Pretending in Virtual Worlds. *E-Learning by Design, 2nd Edition*, 323-398.

[3] Kolb, D. A. (1984). *Experiential learning : Experience as the source of learning and development* (Vol. 1). Englewood Cliffs, NJ : Prentice-Hall.

[4] Landriscina, F. (2013). Simulation-Based Learning. In *Simulation and Learning*(pp. 99-146). Springer New York.

[5] 蕭碧茹、洪振方。2000。《案例式推理與科學教學》。 *Chinese Physics*, , 3.2： 57-74.

[6] 鄭宇樑。2006。《問題導向學習的課程與教學》。*致遠管理學院學報*，(1), 177-195.

[7] Kolodner, J. L. (1997). Educational implications of analogy : A view from case-based reasoning. *American Psychologist, 52*(1), 57-66.

[8] Gee, J. P. (2004). Learning by design: Games as learning machines.*Interactive Educational Multimedia*, (8), 15-23.

[9] 黃志豪。2006。《「從設計中學習」與網路學習社群對學習成效關係之研究》。臺南大學資訊教育研究所碩士論文。

[10] Kolodner, J. L., Gray, J., & Fasse, B. B. (2003). Promoting transfer through case-based reasoning : Rituals and practices in learning by design classrooms.*Cognitive Science Quarterly*,

*3*(2), 119-170.

[11] Tisue, S., & Wilensky, U. (2004, May). NetLogo : A simple environment for modeling complexity. In *International Conference on Complex Systems* (pp. 16-21).

[12] Sklar, E. (2007). NetLogo, a multi-agent simulation environment. *Artificial life,13*(3), 303-311.

[13] Wendel, D. J. (2006). *Design and editing 2.5-dimensional terrain in StarLogo TNG* (Doctoral dissertation, Massachusetts Institute of Technology).

[14] Thiele, J. C., Kurth, W., & Grimm, V. (2011). Agent-and individual-based Modelling with NetLogo : introduction and New NetLogo Extensions. *Die Grüne Reihe, 22*, 68-101.

[15] McCaffrey, C. (2006). StarLogo TNG : the convergence of graphical programming and text processing. *Massachusetts Institute of Technology Master's Thesis*.

[16] 朱江、伍聰。2005。《基於 Agent 的計算機建模平台的比較研究》。*系統工程學報*，20(2)，頁 160-166。

**[17]** Chang, S. K., et al. (1999).Future of visual languages. IEEE SYMP VISUAL LANG PROC,58-61.

**[18]** 張智凱、郭文鱗、王士瑋、李詩婷、林育伶、何昱穎 (2011)。建立探究學習的模擬環境：以流浪狗問題為例。第七屆數位內容國際學術研討會 (ICDC2011)，臺灣。