國立交通大學

理學院科技與數位學習學程

碩士論文

以 Scratch 學習程式設計
及其與學習者認知風格的關連性

Learning Programming with Scratch
and Its Relationship with Learners' Cognitive Styles

研 究 生：陳冠岑

指導教授：孫春在　教授

中 華 民 國　一〇一　年 六 月

以 Scratch 學習程式設計及其與學習者認知風格的關連性

Learning Programming with Scratch and Its Relationship with Learners' Cognitive Styles

研 究 生：陳冠岑　　Student：Kuang-Tsen Chen

指導教授：孫春在 教授　　Advisor：Dr. Chuen-Tsai Sun

國立交通大學

理學院科技與數位學習學程

碩士論文

A Thesis

Submitted to Degree Program of E-Learning

College of Science

National Chiao Tung University

In partial Fulfillment of the Requirements

for the Degree of

Master

In

Degree Program of E-Learning

June 2012

Hsinchu, Taiwan

中華民國一〇一年六月

以 Scratch 學習程式設計及其與學習者認知風格的關連性

學生：陳冠岑　　指導教授：孫春在　教授

國立交通大學理學院科技與數位學習學程碩士班

摘　要

Scratch 軟體的視覺化及拼圖式設計之介面，增加了程式設計的易學性，程式分享網站提供學習者知識累積及共創的平台，藉由現成範例的拆解、組合、重構和微調，初學者從仿效中學習程式概念，增加學習的自主和彈性。

本研究依學習者認知風格偏好的不同，將國中九年級受試學生分為場地獨立和場地依賴二組，以學生自學過程中的程式設計策略和解題行為來觀察其學習歷程，並以學習任務來檢測學習成效，探討個別差異在 Scratch 學習程式設計之影響。

根據本研究結果，得到下列結論：

一、不同認知風格學生，在 Scratch 學習程式設計，其設計策略沒有顯著差異，可能與範例式學習只需修改部分程式有關，所以採用的策略較不明顯，但場地獨立偏好之學習者較傾向 Bottom-Up 方式。

二、場地獨立偏好之學生，傾向使用解構的解題行為進行程式設計，於拆解和重組次數上較多，能從中瞭解程式概念。

三、在 Scratch 學習程式設計，不會因為認知風格偏好的差異而產生學習成效之不同。

關鍵字：認知風格、Scratch、學習程式設計

The Effects of Cognitive Styles on Program Learning in Scratch .

Student:Kuang-Tsen Chen Advisor: Dr. Chuen-Tsai Sun

Degree Program of E-Learning

National Chiao Tung University

ABSTRACT

Scratch, a visual and block-based programming language software, makes programming easy to learn. Websites of Scratch program sharing provide further knowledge and a discussion platform for students. With deconstructing, combination, reconstruction and fine tuning, beginners learn the concepts of programming by imitating and recombination. This approach encourages autonomy and flexibility in learning programming.

In this study, we collect data from 75 junior high school students for research by convenience sampling, and use Group Figure-Embedded Test (GFET) to measure the tendency of their cognitive style. We observe the learning process of the students in terms of their design strategy and program-solving behavior, evaluate their learning achievements based on a design tasks, as well as investigate the individual differences.

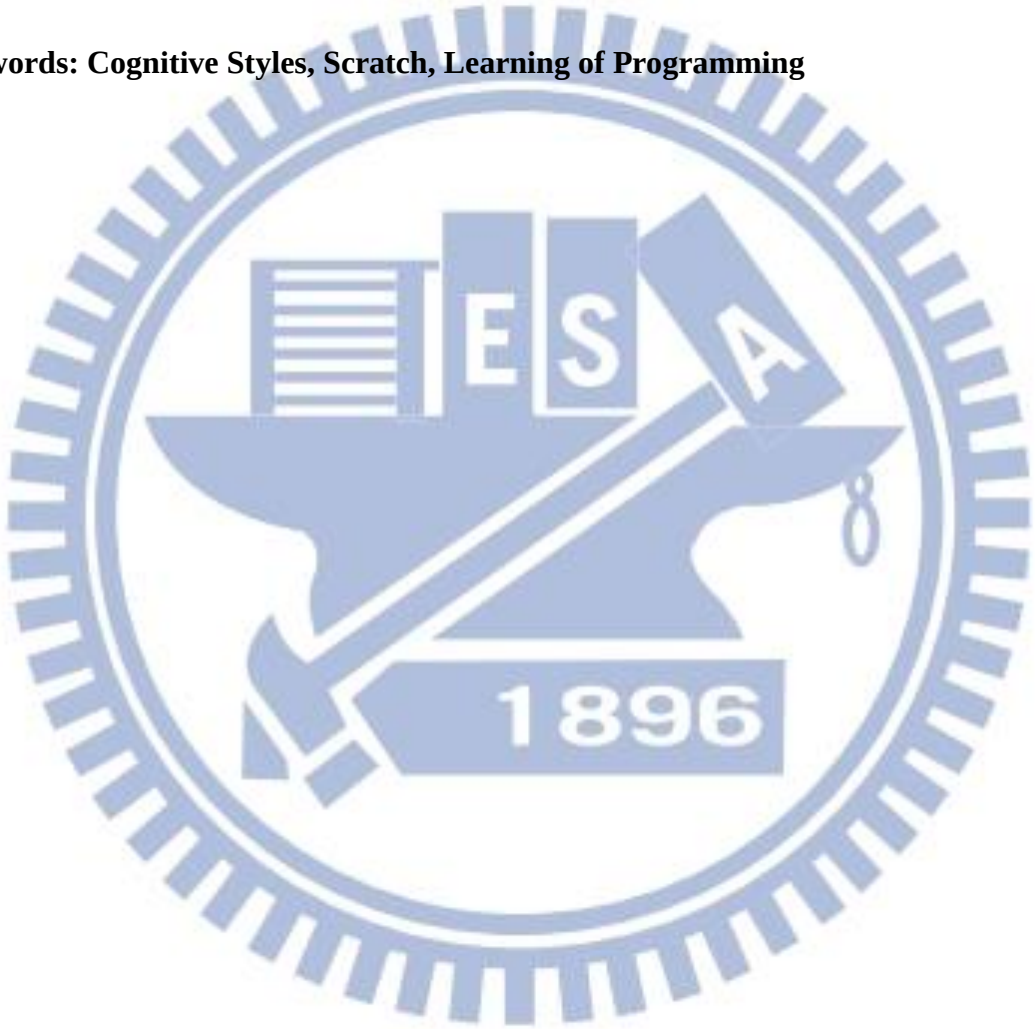
Based on the result of experiment, our primary findings are as follows:

a. Different cognitive styles show no relationship with design strategy. This may result from the fact that the students need only to modify others' program. However, the learners with Field Dependence tendency usually use the Bottom-Up approach.

b. The students with Field Dependence tendency express problem-solving behavior of deconstruction in programming, show higher frequency of breaking down modules and recombination, and find out more concepts of programming.

c. Cognitive Styles have no relationship with learning achievements in program design of Scratch.

Keywords: Cognitive Styles, Scratch, Learning of Programming



誌　謝

終於結束台北－新竹兩地奔波的日子，雖然有點辛苦，卻也在這兩年的研究所生涯中學到了許多,除了教材製作能力的提昇,更重要的是個人想法的延展和多方向的探索，從研究中發現問題、探究和證明，培養判斷和邏輯思考的能力。

最感謝的是孫春在老師不厭其煩且正向肯定的教導方式，每次的 meeting 都能激發出另一種見解和看法，從論文題目的挑選開始，與老師的討論便獲益良多，在研究過程中的問題和困難，老師都能一語道破，讓雜亂無章的思緒都變得有脈絡了起來。

也感謝實驗室的學長姐給予的幫助，讓實驗的流程和統計分析都能順利完成。實驗室伙伴們的互相鼓勵和分享，成為能夠繼續堅持下去的動力，能跟你們在同一個實驗室做研究，真的很榮幸。

撰寫論文的過程歷經喪父之痛，感謝研究所同學、同事和朋友們的安慰和幫忙，媽媽、哥哥以及家人們的關心和體諒是我能放心做研究的定心丸，希望這份成就能與大家分享，也獻給一直期望我能完成研究所學業的爸爸。

最後要特別謝謝一直在我背後默默支持、鼓勵的育瑋，在我論文寫得心煩意亂的時候、在我從新竹回來暈車難過的時候、在我實驗遇到瓶頸的時候，都能包容和陪伴我，真的很謝謝你！

冠岑

2012.06

目　錄

表目錄

圖目錄

第一章 緒論

本研究的目的在探討於 Scratch 軟體下，不同類型的認知風格學生，如何影響程式設計的學習歷程以及學習成效。本章共分以下五節：第一節為研究背景與動機；第二節為研究目的；第三節為研究問題；第四節為相關的重要名詞定義；第五節則為研究範圍與限制。

1.1 研究背景與動機

Scratch以視覺化程式設計介面幫助抽象概念的具體化，利用表徵模式減少學生在程式設計時受限的語法，拼圖式設計使得操作容易且避免迷思於語意或語法偵錯上，讓學習者能將程式設計時的注意力專注在問題解決的邏輯推理上，且動態呈現方式增加學生學習的興趣，符合數位原住民處理多樣訊息、圖像偏好及喜歡主動探索和互動式學習等特性(Prensky, 2001)。許多研究者(如Funkhouser,1993)認為，適當的程式設計介面，視覺化或物件化的環境，會對初學者的學習產生影響，並能將程式的基本觀念類化至其它程式語言或相關領域。一項以Scratch為補救教學工具的實驗中也發現，對於程式設計學習成效、降低學習焦慮以及維持學習動機上都能有所幫助(何昱穎、張智凱、劉寶鈞，2010)。

Scratch讓程式以積木方式堆疊而成，於是教師可能迷思於各個程式區塊的使用時機及用途，從流程控制、角色如何進行動作及行為開始解說，讓學生根據教師的專家型解題經驗來組合程式。但是，Scratch除了以結構方式呈現程式，更重要的是能透過現成範例，以觀察、拆解、模仿和重組的學習歷程，形成學習者為中心的重構模式，學習者不需要先了解各個程式碼所代表的意涵，從範例看到結果的呈現、於執行中發現撰寫規則，推論出各組件的功能，修改某部分程式後再反覆觀察其差異，於做中學瞭解程式架構和行為模式，達到具體、立即性的效果，藉由主動探索來自我經驗出新的知識，進而對程式邏輯產生體悟。許梅君(2008)利用演練範例的呈現方式對高二學生進行程式設

計教學,結果發現與程式語言學習成效、程式設計解決能力以及學習態度都有正面影響，證明範例式的學習可在演練過程中形成解決問題的基模。程式的模仿並非抄襲或不創新，反而讓學習者有了目標且確定程式的可行性，加上因為可以重製的優點，使得學習者免於失敗的挫折，增加了重複操作練習和改造程式的信心。

Scratch提供程式設計者一個共享平台，如同Web2.0能夠主動傳遞學習經驗與回饋機制，學習者可以主導掌握進度，既是知識的提供者也是接收者，資訊的累積增加了程式設計的創意，在集體創作中刺激個體思考，使得學習達到更好的溝通、互動和成果(林曉薇，2010)。Driver和Oldham(1986)認為，學習者透過觀察、評鑑、概念的衝突和重組、反省和分享等,於學習歷程中建構知識,在經驗交流的過程中提升參與感和認同感，衍生出合作學習和新知識形成的環境，也能促使自學能力的養成。Scratch簡單、直覺式的視覺化介面讓程式具備高自我說明性，且年齡層相近之程式設計者間相較於教師所提供之範例，更符合Vygotsky的近側發展區理論，高可讀性和理解性讓程式被閱讀和修改的次數提高，培養從現成程式中擷取所需，形成程式設計的能力。由別人分享的程式進行修改與再分享的模式也符合Lawrence Lessig提出的創用CC(Creativity Commons)概念，個人著作經過合理授權改作後，集合眾人知識而形成更完整的作品，讓程式設計隨著不同創作者的經驗成長而進化。

認知風格是影響程式設計學習的重要關鍵(R. Mancy and N. Reid., 2004)。個體處理訊息的習慣是穩定不易改變的，此認知風格偏好的差異，會使學習者結構性知識有明顯不同，直接影響資訊擷取和問題處理的方式，使得在 Scratch 學習程式設計時，仿效、拆解及修改的過程有所差異，而分享和回饋機制隨不同個體訊息的交互作用，可能使得相同結果下，程式設計策略及解題行為的不同，進而對學習成效造成影響。不適當的學習情境可能會讓學習者無法掌握教學目標及運用學習策略，甚至可能造成認知混淆，阻礙了學習。因此，瞭解不同學習者在認知風格的個別差異，以及分析可能的操作歷程和學習成效，可做為教師的教學模式預測(Riding & Cheema,1991)。

認知風格的類型依不同學者各有所論述，過去對於 Scratch 和認知風格之研究，發現視覺導向之學習者在作品的構圖上明顯優於文字導向之學習者(張素芬，2010)，但非

視覺型學習風格學生的學習投入平均高於視覺型學習風格學生(何昱穎、許靜坤、王一成、林育伶，2011)。本研究以學習者於 Scratch 範例程式進行拆解和重組、採用的設計策略等學習歷程進行分析，學習者必須要能從複雜的情境中將主要概念抽離出來，並進行組織與重構，此模仿、類比式學習與場域相關，因此認知風格採用場地獨立(Field Dependence, FD)與場地依賴(Field Independence, FI)的分類方式，且在教育領域研究中，也以 Witkin 所提出的認知風格為最常使用之分類方式。

因此，本研究試圖找出學習者學習程式設計時，不同人格特質學生在 Scratch 學習程式設計的個別化差異，並希望藉由模仿、共享、解構、重組的特性，瞭解範例式學習對於程式設計學習是否有益，觀察初學者在 Scratch 自學過程中的設計策略和解題行為，提供教師設計適切之教學教材及學習策略，加倍學習效率和成效。

1.2 研究目的

本研究主要目的是藉由 Scratch 進行程式設計學習，解決抽象概念對學生造成認知負荷的增加，以共享和仿做模式讓初學者在學習程式設計時，免於不知如何下手的挫折感，並藉由自學方式來觀察學生學習成效。此外，為了解此教學情境是否適合不同學習者的需求，利用團體藏圖測驗進行施測，將學生依認知風格分成場地獨立與場地依賴共二種類型，探討不同人格特質的學習者，於 Scratch 學習程式設計，其訊息處理的過程，與對學習成就之影響，提供教師適性化的教學策略。

基於上述理由，本研究的目的如下：

一、瞭解對初學者以 Scratch 進行自學是否有助益。

二、瞭解不同認知風格學習者，於 Scratch 學習程式設計時，藉由模仿和共享的特性，其學習歷程及學習成效為何。

三、提供教師教學設計及教學模式預測。

1.3 研究問題

依據上述研究目的，訂定研究問題如下：

一、使用 Scratch 軟體學習程式設計時，不同認知風格學習者，在設計策略上是否有差異？

二、使用 Scratch 軟體學習程式設計時，不同認知風格學習者，在解題行為上是否有差異？

三、使用 Scratch 軟體學習程式設計時，不同認知風格學習者，在學習成效上是否有差異？

1.4 名詞定義

為避免本研究中使用的詞彙與意義產生混淆，將研究中使用的相關重要名詞，解釋說明其概念性和操作型定義，分述如下：

一、認知風格：

個人在知覺、記憶、思考和問題解決時的慣性偏好處理方式(Allport, 1937)。本研究根據 Witkin 等人發展的團體藏圖測驗（Group Embedded Figures Test, GEFT）對受測者進行測驗，以區別其認知型態，分成場地獨立型和場地依賴型。

二、學習歷程：

學習者在學習過程中，認知處理的過程，而非學習結果的產物(Lewalter, 2003)。本研究使用螢幕錄製軟體，透過學習者個別使用 Scratch 軟體進行程式設計時所產生的操作行為，加上問卷調查來獲得學習者的設計策略和解題行為等資料，以此作為學習歷程量化分析之依據。

三、學習成效：

學生在完成學習活動後，能符合學習目標，達到學習進展的成果表現。本研究讓學生從 Scratch 提供的分享平台下載範例進行修改，以學習任務的完成度進行檢測，任務共分為十個評分細項，學生完成一項即得一分，總共十分。

1.5 研究範圍與限制

本研究以某國中九年級學生為研究對象，研究對象共三班，有效樣本 75 人，目的是探討學生在 Scratch 學習程式設計的學習歷程和學習成效。受限於樣本數，可能無法代表所有當今九年級學生的認知歷程和學習表現，因此不宜過度推論。

使用 Scratch 軟體來學習，並不包括其它程式設計軟體，對於各項程式設計介面與功能的差異，本研究不加以討論。

研究中所探討之認知風格，以場地獨立／場地依賴做分類，全程使用螢幕錄製工具來記錄學習者的學習歷程，並以問卷調查輔助量化資料，然而以量化方式來分析學生操作 Scratch 軟體時的操作過程，只能判斷學習者是否有出現該行為，無法知道學習者在不同程式碼間拖曳時，其認知轉換過程。

第二章 文獻探討

本研究是以 Scratch 進行程式設計為出發點，探討不同認知風格的種類，對受試者於學習程式設計的歷程和學習成效之影響。本章分別對 Scratch 程式設計環境以及認知風格進行整理歸納，以三節探討相關的文獻。第一節探討程式設計的相關議題、第二節介紹 Scratch 的特性、第三節定義認知風格及其相關研究。

2.1 程式設計

2.1.1 程式設計的重要性

程式設計課程可以發展出有價值的生活技能。資訊科技的普遍性發展，改變生活型態及教育環境，在科技時代生長的數位原住民，習慣同時處理多樣訊息，偏好視覺圖像介面，喜歡主動探索和互動式學習，除了熟悉使用數位媒體外，也應該設計、創造和發明，才能真正形成資訊科技知識的能力(Prensky, 2001)。程式設計應是學生必備的基本技能，透過反覆思考來培養解決問題和高層次思考的能力，尤其對於邏輯推理的解題方式有助於運用在其它學科及生活應用上。許多學者(如 Shafto，1986)指出，程式設計課程讓學生在數學和科學的學習上是有幫助的，證明清楚有條理的分析、確實執行驗證及偵錯，能類化至其它學科的學習。

建構主義強調藉由實作來達成學習目的，在設計程式的過程中，學習者除了必須具備的邏輯概念，為了完成作品，也必須在製作的過程學習相關的技能或知識，甚至是學習到解題策略或培養創造力等，於是學習成為一種主動知識的建構而非被動的吸收，學習型態也由傳統教師為主導的方式轉變為以學生為中心的學習模式。Kafai 認為，設計能將個人想法具體實現，且在設計的過程能產生學習動機，而設計也能讓學習者將情意和認知相互結合，即使學習者未能完成設計的成品，在操作與練習的過程中已經達到學習效果，因此，程式設計課程正是具備高度學習經驗的重要課題。

2.1.2 程式設計的工具

學習者所使用的程式設計開發軟體，其操作上可能因不同學習者的認知型態而產生不同的影響，使得程式設計變得複雜和困難。在程式設計問題解決過程中，如果能提供適切的程式設計工具，能幫助學習者容易把思緒完整確實表達在程式設計上，達到較佳的學習效果。

以程式設計的開發環境介面做分類，分為文字式與視覺化，以下就二種類別分別說明：

一、文字式程式設計：

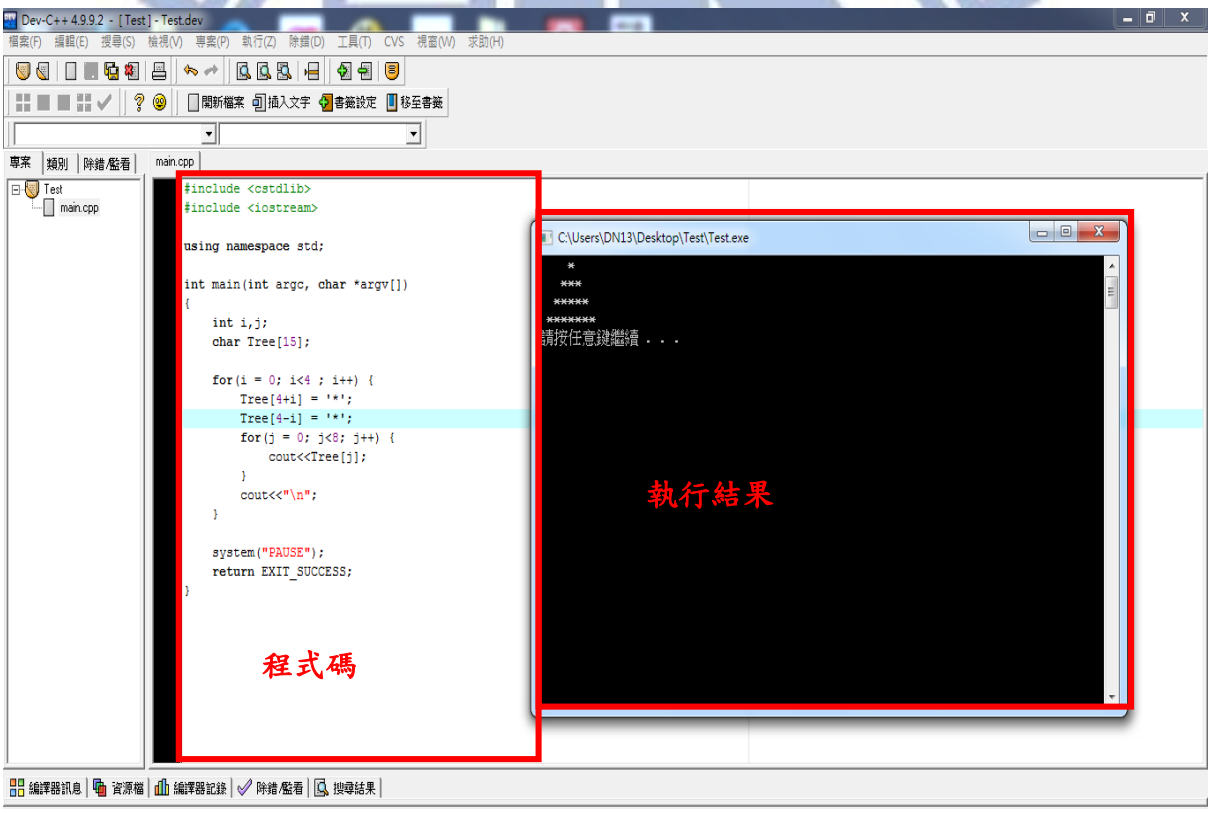


圖 1 文字式程式設計介面

文字式程式設計介面分為程式碼撰寫區和執行結果區，程式碼由使用者依照各個程式語言的規則自行輸入，沒有既定的模組和圖形化可供使用，使用者在撰寫過程中若出

現錯誤，只能一行一行找出問題所在，不論是在程式設計或結果呈現上都比較抽象不具體，畫面缺乏生動和互動性(如圖 1)。

學習者必須以文字的描述讓程式產生類別及物件，程式的編輯以由上而下的方式，不需受限於已經建置好的物件設定或動作判斷，控制性和延展性較強，沒有預先的參考程式模組，學習者必須具有各個程式功能的先備知識，適合專家型學習者，例如 C、C++、JAVA……等。程式執行結果缺少色彩與互動，不易讓學習者產生興趣，缺少具體且生動的學習情境，可能使得學習目標不夠明確。

程式的撰寫需要較多的邏輯性，通常需要事先規劃程式流程及架構，對於程式的進階設計提供比較彈性的空間，但容易產生語法或語意錯誤，且不同程式語言之間，即使表達相同意思，語法卻有所不同，例如 C 語言中的「prinft」代表列印結果，但在 C++語言中為「cout」，而 JAVA 語言則是使用「System.out.print」，學習者可能花費過多時間在偵錯上，尤其初學者在不同程式語言間的轉換可能產生困難或混淆，邏輯思考因此受到阻礙。

目前文字式的程式語言，例如 C、C++、Java 等，雖然程式撰寫功能較強，但龐大的程式結構並非為教學而設計。對於初學者而言，並不需要知道太多繁瑣的細節，以精簡的指令來建構出主要程式的流程才能使學習者專心於程式的設計(Kurtz, 1981)。

二、視覺化程式設計：

將問題視覺化或物件化，可以使得問題變得具體，將能更有效掌握程式設計的架構，促進學習的思考與解題，尤其對於初學者能夠降低學習程式設計的門檻(Funkhouser, 1993)，以圖像式或特定指令模組，讓程式的組成變得簡單，雖然沒有提供完整和進階的程式設計功能，但具備初學者可能使用的程式基本語法，重點放在程式的邏輯培養。目前幾個較常見於中小學程式設計教學之軟體如下:

1. LOGO

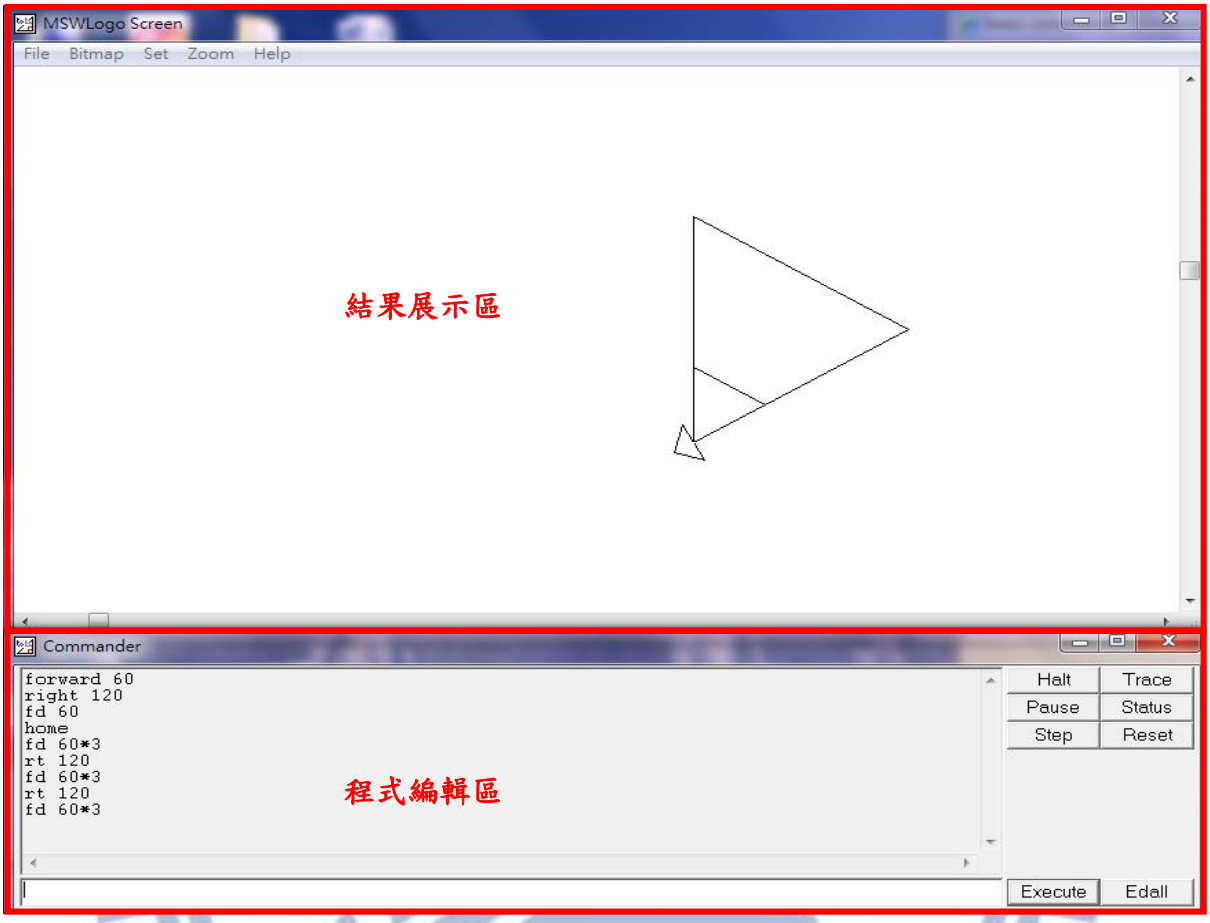


圖 2 LOGO 程式設計介面

於 1967 年以 Jean Piaget 的建構理論為基礎，由 Massachusetts Institute of Technology 的 Seymour Papert 教授所研發，使用既定的指令來控制小海龜行進方向，透過簡單的指令就能讓小海龜所畫出的圖形複雜化，並能計算和發出聲音，適合不熟悉程式語法的學習者，且能透過小組互動來增加同儕學習效果(Webb, 1984)。

Logo 語言的設計目的是為了透過程式設計讓數學學習具體、操作化，對於幾何圖形、算術、統計或空間概念皆能有所幫助，但提供之程式語法較少，學習者無法使用太多程式碼讓小海龜有更多的行為表現，且在角色的提供上也不足。

LOGO 程式設計介面分為二個區塊:在最下方的程式編輯區輸入基本指令來指揮海龜前進(FD)、後退(BK)、右轉(RT)、左轉(LT)……等動作，結果會立即呈現(如圖 2)，因此，學習者必須記憶基本語法，並對每個語法的功能和使用方法仍需有基本概念。

KTurtle則是簡易版的LOGO程式設計介面(如圖3)，為了讓學童更早接觸程式，且藉由此軟體學習基礎的數學幾何觀念，以直觀的語法，簡單的錯誤訊息畫面，並結合繪圖，同樣以小海龜的行走做為程式執行的依據。

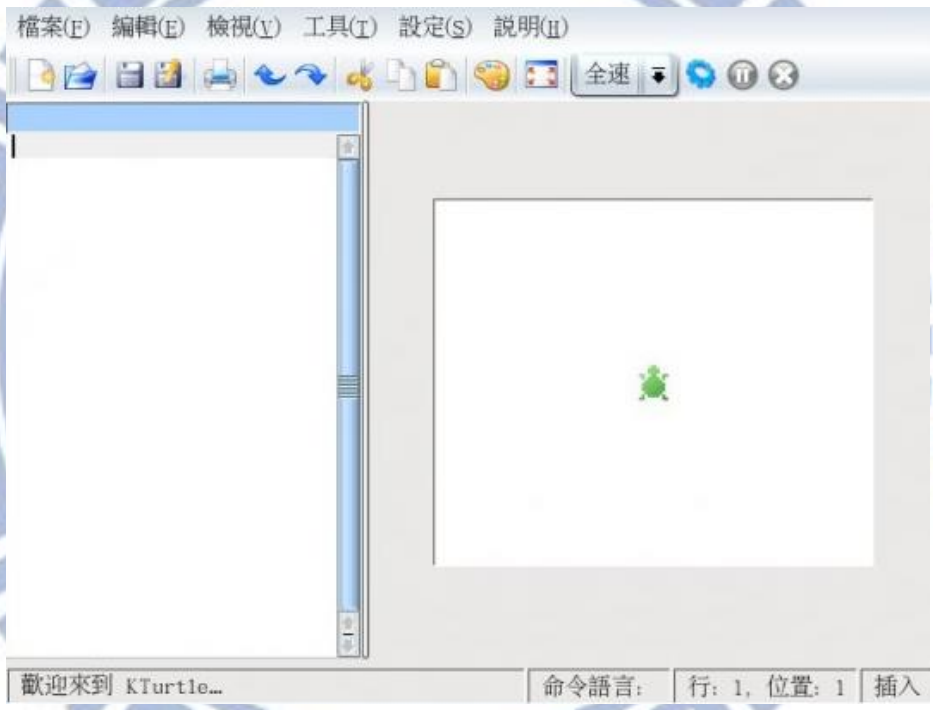


圖 3 KTurtle 程式設計介面

對於用 Logo 程式語言進行學習的研究，整理如表 1，雖然對於可使用的指令有限，但透過不斷的思考、分析問題及解題策略，仍能增加學生學習興趣及增加創造思考力，尤其對於數學解題有明顯提昇:

表 1 使用 Logo 進行學習之相關研究

研究者	*研究結果*
Clements & Gullo(1984)	Logo 能增加學生流暢力、獨創力的創造思考力。
Mayer & Fay(1987)	學習 Logo 語言可以增進空間認知能力。
Many, Lockard, Abrams, & Friker(1988)	學習 Logo 程式語言的學生在問題解決與思考的技能優於沒有學習過的學生。
Clements(1991)	Logo 實驗組在創造思考力圖形測驗上明顯增加。
黃文聖(2001)	Logo 學習環境提供學生主動思考的情境，且可以幫助學習數學並增進解題能力。
許宏彰(2005)	以 Logo 進行學習明顯提昇了學生的學習興趣。

2. Alice

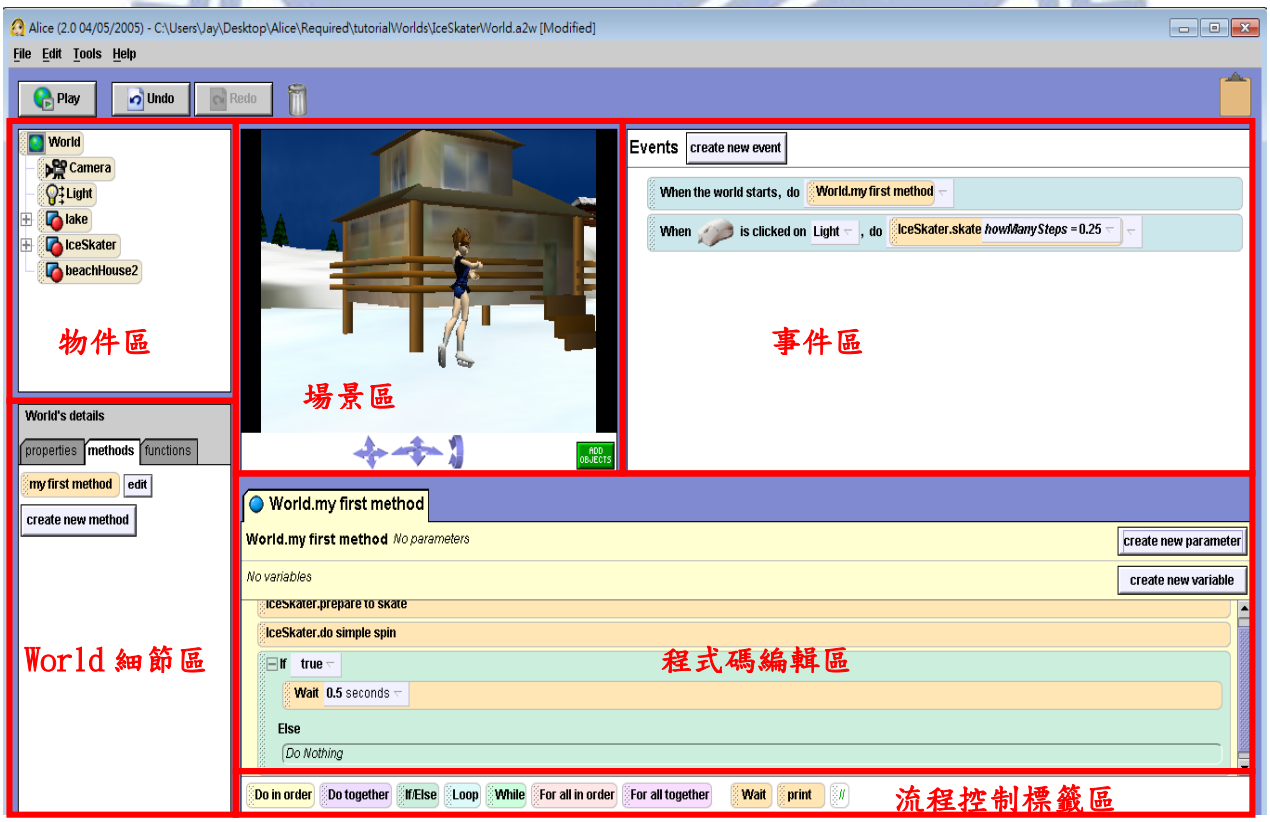


圖 4 Alice 程式設計介面

由 Carnegie Mellon University 所研發，是 3D 虛擬實境軟體，讓學習者像是導演一樣，創作屬於自己的動畫故事，其 3D 物件類似 Java 物件導向的概念，配合拖曳程式碼以及下拉式選單來設定參數，且程式語言類似英文文法，學習較容易又能避免產生語法錯誤。提供了語法切換功能，當學生對物件導向概念有了基本認識後，可以進一步學習較為抽象的 C++或 JAVA 程式。

Alice 利用說故事的方式讓學習者在 3D 環境中隨著各個角色的定位，自然的學會程式設計，對於初學者或女性學習者能引起其學習動機（Carnegie Mellon University, 2006）。但目前 Alice 只有英文版，對國內學習者而言，較不夠親和力，其設計介面包含六大區塊：物件區、場景區、事件區、World 細節區、程式碼編輯區和流程控制標籤區(如圖 4)。因功能介面較多，對於初學者仍需花費時間熟悉畫面的操作。

使用 Alice 進行教學的相關研究整理如表 2，以動畫呈現方式，將程式從物件概念開始建立，對於學生在程式設計的學習態度及成效能有明顯提昇。

表 2 使用 Alice 進行教學之相關研究

研究者	*研究結果*
王鼎中，丘聖光，林淑玲，梅文慧，林美娟(2009)	Alice 能有效提昇學生在程式設計概念的建立。
朱君怡(2010)	Alice 動畫式程式設計能啟發學生之創意思考能力，並改善學生的電腦態度，增進程式設計課程的學習成效。
林輝鐸，莊桓綺，陳怡琴，羅珮妤，鄭郁蝶(2010)	使用 Alice 進行程式設計教學，相較於 VB 程式設計，所花費的時間較少，而學習成效較高，學習興趣也相對提昇。

3. Scratch

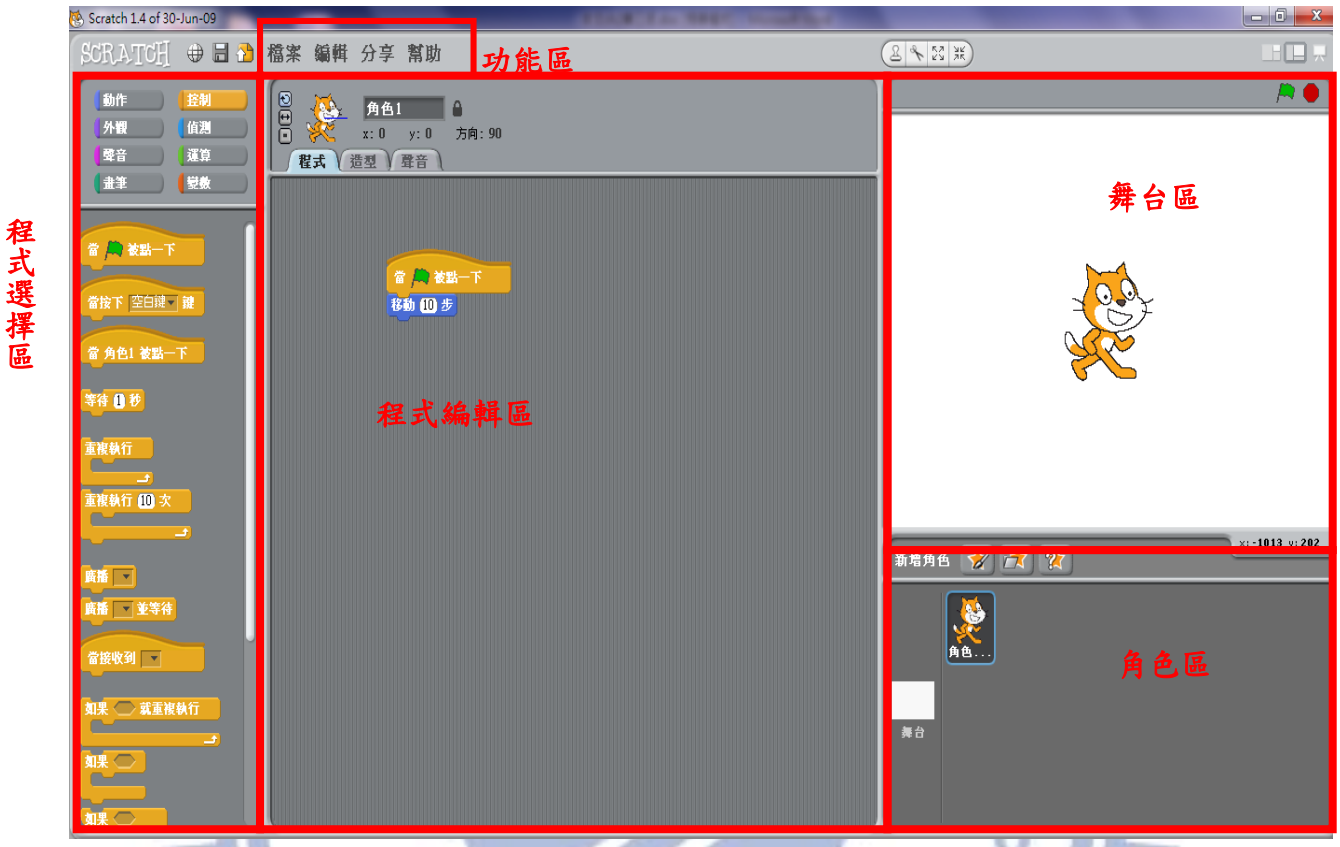


圖 5 Scratch 程式設計介面

由 Massachusetts Institute of Technology 所研發，是一套適合八歲以上的學習者學習程式設計的軟體，可以創造互動故事、動畫、遊戲、音樂和藝術等。學習者在 Scratch 中，只要用滑鼠拖曳拼圖式的結構圖案，只有適合的語法才能正確組織在一起，因此，不需要考慮程式語法，只要專注在流程結構以及問題邏輯就可以了，適合用來進行初次的程式設計課程，且有助於之後學習其它物件導向語言的程式設計(Malan & Leitner, 2007)。

Scratch 是參考 LOGO 語言而設計，因此同時具備了 LOGO 能夠繪圖以及發出聲音的功能，簡單及中文化介面幫助初學者更容易學習。其開發環境分為五大區塊(如圖 5)：

(1)角色區：Scratch 內建許多有趣的角色提供學習者使用，也可藉由簡單的繪圖功能自行繪製或修改現有的角色外觀，或是直接從檔案匯入圖片，角色製作的彈性很高(如圖 6)。

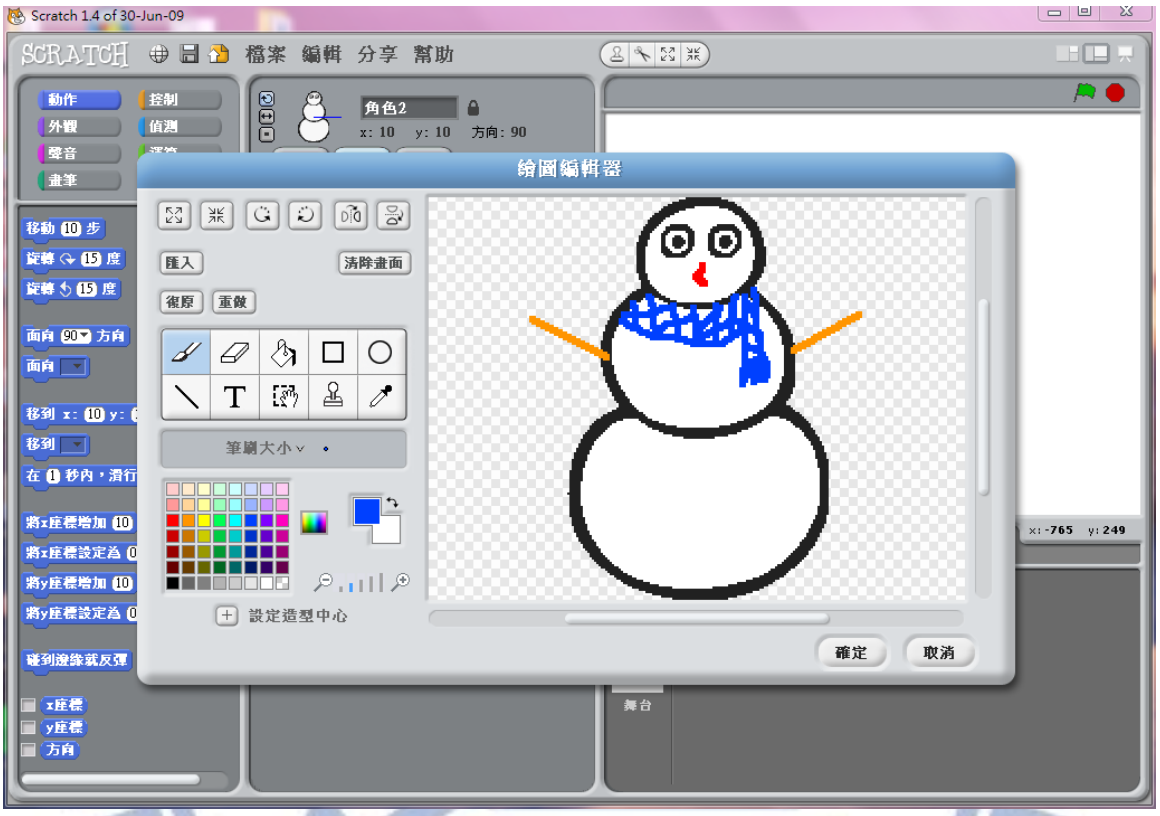


圖 6 Scratch 角色繪圖編輯器

(2)程式選擇區：利用不同顏色來區分程式類別，分為控制、動作、偵測、外觀、運算、聲音、變數以及畫筆(如圖 7)，各程式的使用時機和功能如表 3。學習者不需要自行撰寫程式，直接從程式選擇區挑選適合的程式碼即可，在程式組合時也不用擔心程式語法錯誤，拼圖式的外形讓符合的程式才能結合在一起。

圖 7 Scratch 程式類別

表 3 Scratch 程式類別及功能

程式類別	功能說明
控制	用來驅動角色，可以滑鼠、鍵盤或廣播方式進行角色控制 程式流程的控制，例如「重複執行」、「如果／否則」
動作	讓角色進行移動或旋轉，提供座標軸來設定角色位置
外觀	角色造型、大小及特效
偵測	是否進行滑鼠／鍵盤輸入、角色位置以及角色碰撞等， 偵測的目的是要提供下一個步驟執行的依據
聲音	每個角色可匯入內建的音效或由使用者自行錄製聲音
運算	提供＋、－、×、÷及大小判斷等運算式
畫筆	可設定畫筆大小、顏色及亮度等
變數	提供變數及列表，可做進階程式設計

(3)程式編輯區:從程式選擇區挑選的程式碼,使用滑鼠拖曳的方式拉到程式編輯區,並利用拼圖式的外形將程式組合起來，避免程式語法錯誤。參數的設定則是利用鍵盤直接輸入或是下拉式選單來挑選，簡單、直覺又易用。先將每個角色的造型及聲音匯入後，就可以用程式進行控制。

(4)舞台區:放置所有角色,只要按下綠色旗子就能讓各個角色依照所挑選的程式進行動作，讓程式的結果以動態方式呈現。

(5)功能區:Scratch 本身也提供了「分享」功能，學習者可以藉由分享平台搜尋特定主題，觀看並下載他人程式，進行修改與增加新功能後，再上傳自己的作品，以「想像」、「程式」和「分享」為此分享網站的設立目的，希望藉由集體創作和分享、回饋等機制，達到更好的學習效果(如圖 8)。

圖 8 Scratch 分享平台

國內使用 Scratch 進行教學實驗的結果中，不論在問題解決能力、邏輯推理能力、創造力和學習態度上，大部分都能呈現正向相關(如表 4)，證明了 Scratch 適用於程式設計學習的適切性，且在眾多視覺化程式設計軟體中，以 Scratch 為目前最廣為接受並運用於教學現場。

表 4 與 Scratch 教學相關之研究

研究者	研究結果
楊書銘(2007)	學生對於學習 Scratch 認為是有趣的活動，不會感到焦慮與排斥，且 Scratch 課程對於學童之「猜測原因」、整體問題解決能力、「開放性」和「變通力」創造力等有顯著提升。
王麒富(2009)	多數學生對 Scratch 科技接受模式的滿意度高，學習態度正向，且問題解決能力有所進步
蕭信輝(2010)	Scratch 教學課程對學習者整體科學過程技能、科學問題解決能力有顯著影響。
蔡孟憲（2010）	Scratch 程式設計課程對幾何概念有顯著提升

2.1.3 範例式學習

程式的基本設計流程分為四個階段：分析問題、流程設計、程式撰寫、偵錯與修正(如圖 9)。傳統的程式設計課程，會從規劃程式流程開始，但因個體認知型態的不同，可能使得解題流程產生差異，甚至於分析題型上已造成學習阻礙，利用流程圖雖然可以幫助學習者了解程式的運作，卻無法有效幫助初學者寫出程式，學習者受挫於流程圖的繪製，或者對流程了解但仍無法具體著手進行程式設計，於是對程式設計產生恐懼(Winslow, 1996)。

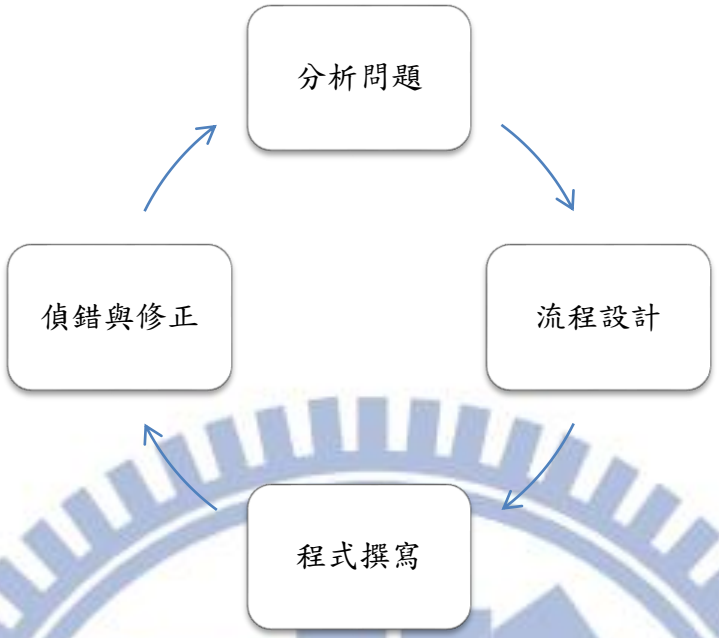


圖 9 程式的基本設計流程圖

為了讓初學者能快速進入程式撰寫階段，教學者會先將各個語法的使用時機與撰寫方式進行完整的介紹，但程式概念抽象的本質不易理解，使得初學者反而受限於程式語法或專注在程式偵錯上，無法確實將思考邏輯清楚呈現，且不同程式語言間的語法轉換也可能讓學習者產生混淆，於是教學者花費過多時間於程式語法的講解，而學習者卻無法真正學習到解決問題的能力(Costelloe, 2004)。

Bandura 的觀察學習理論，認為人與生俱有經由觀察來獲取訊息的能力，且能移轉至下次面對類似問題時的解決能力，或是可以經由他人錯誤的經驗來吸取教訓，避免發生同樣的錯誤(Bandura, 1973)。範例式學習便是讓學習者從現成程式範例去觀察各個程式語法的出現時機和運用方式，以專家的解題流程來讓新手經驗出解決問題的步驟，對於背景知識缺乏的初學者而言，範例的模仿觀察能比傳統方式更快速達到學習目的，且在解決相似問題時的錯誤也較少(Sweller & Cooper, 1985)。

2.1.4 程式設計策略

程式設計依據解題程序的不同可歸納為 Top-Down(由上而下)以及 Bottom-Up(由下而上)兩種常見的設計策略，分別說明如下：

1. Top-Down(由上而下)：

Top-Down 先是將整體問題分層規劃成較簡單的子問題，直到原來的複雜問題已經可以由可理解的小問題來取代，例如先將主程式分成 A、B、C 三個場景，再將 A 場景分解成 A1、A2、A3 三個角色，B 場景分解成 B1、B2 二個角色，而每個角色又分別有自己的動作行為，例如 A1 角色有 A11 動作，而 A2 角色有 A21 及 A22 動作……等等（如圖 10）。

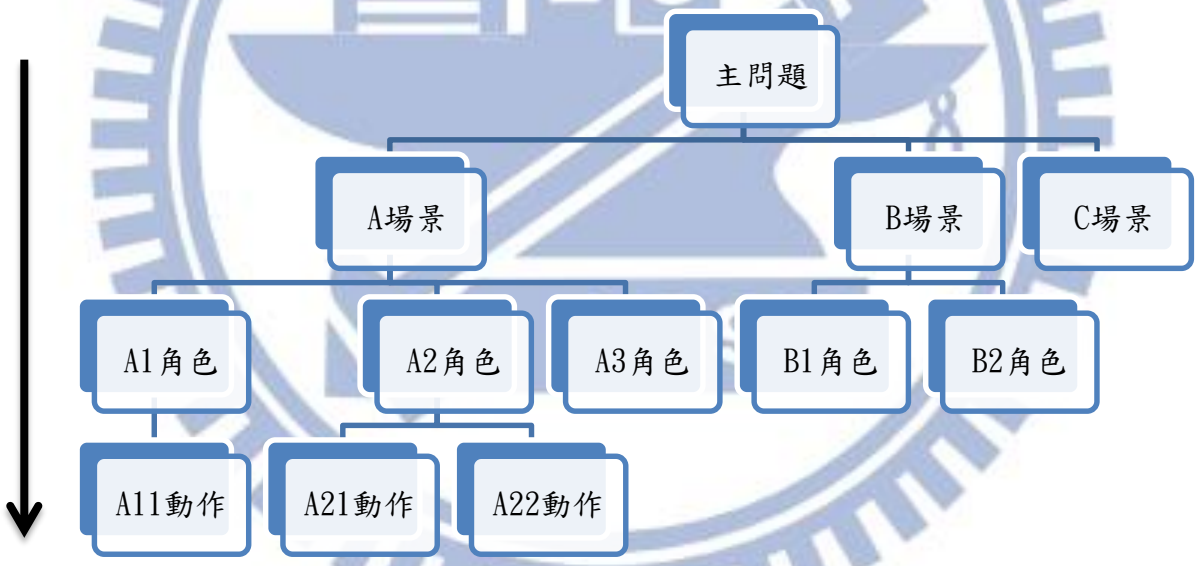


圖 10 Top-Down 設計策略

Top-Down 設計策略也可以稱做「模組化(Module)」設計，先將程式可能會常用到的主要功能規劃設計後，再根據各個角色的需求，將相似的部分重複使用再進行微調，易於分化問題，也可組合起來解決不同且更複雜的問題。

2.Bottom-Up(由下而上)：

Bottom-Up 則是依據題目要求，先考慮較底層的項目，找出相似的區塊，以相同方法按部就班的解決，再將各個項目組合成整體問題，例如先設計需要的角色 A、B、C、D，再將角色間的關係組合成場景 X 和 Y，最後完成主程式(如圖 11)。

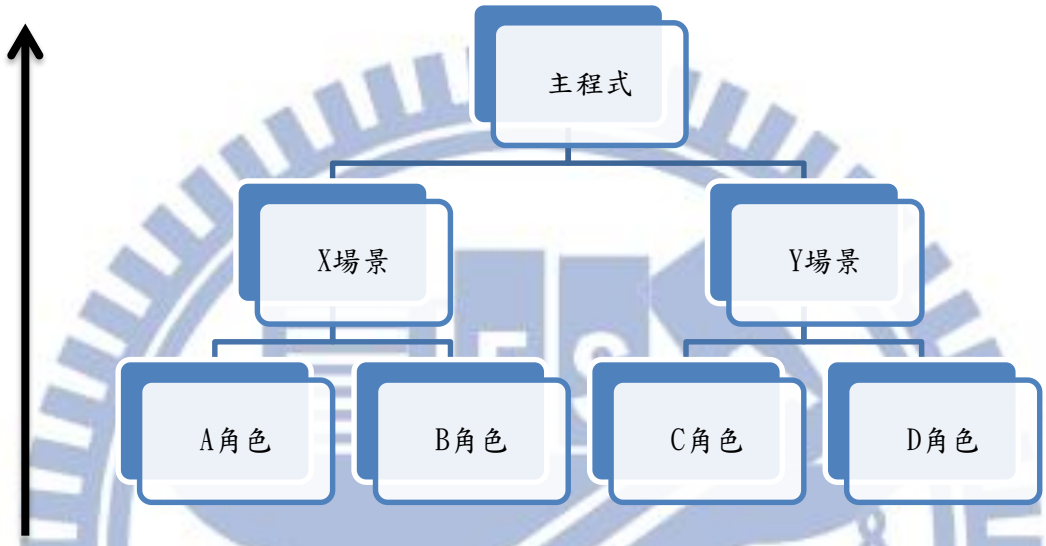


圖 11 Bottom-Up 設計策略

其中,Top-Down 被認為是較有效的設計策略,因能將問題分解成較容易解決的區塊,使得程式設計能由簡入繁。而此兩種策略之特色及優缺點比較如下(如表 5):

表 5 Top-Down 與 Bottom-Up 設計策略比較表

設計策略	Top-Down	Bottom-Up
特色	先設計程式的主要功能，再分層規劃其次要功能。	先完成細部的問題,再逐次組合成完整的問題
優點	1. 延緩考慮細節問題 2. 可依據題目要求來驗證程式主功能,再由主功能去檢測次要功能是否正常,且主功能被測試的機會最多 3. 若不符題目要求之功能可即早發現問題 4. 當程式規模過大時,有助於減少程式設計整合問題	1. 可即早對各個細項的子問題評估是否可行 2. 最基本的問題能夠徹底解決 3. 若錯誤發生在低層問題，可即早發現並解決
缺點	1.必須先完成最高層的問題後，才能知道低層次的項目是否可行 2.若無法正確分析出程式架構，可能造成整體程式設計方向錯誤	1.同樣程式結構重複出現的機率較高，可能造成程式時間複雜度較差 2.必須將區塊完全組合後才能看到整個程式全貌

2.2 Scratch 的特色

2.2.1 視覺、拼圖式程式設計介面

Scratch 的開發環境，從程式設計開始到結束皆以視覺方式呈現，包含了程式可用模組、程式撰寫方式、角色物件，以及程式結果執行畫面等，不同於一般的視覺化程式設計介面，比較如表 6。

表 6 視覺化程式設計介面比較一覽表

	程式可用模組	程式撰寫方式	角色物件	程式執行結果
Scratch	視覺化	視覺化	視覺化	視覺化
Alice			視覺化	視覺化
Logo			視覺化	視覺化

拼圖式的設計，讓學習者免於程式語法偵錯的困擾，且容易將邏輯思緒直覺的表達呈現，中文和圖像式的程式碼，讓程式撰寫變得簡單易用，不需要特別繪製流程圖，即可以很清楚的將程式執行順序以拖曳方式規劃出來。Scratch 內建豐富的物件，並提供簡單的繪圖工具，學習者不需浪費時間在角色的美工設計上，而能專注在程式動作的撰寫，其原理如同訓練演員技能，只要該角色匯入造型(如圖 12)和聲音(如圖 13)後，就可以藉由程式的控制來完成外觀和音效的呈現方式(如圖 14)。舞台區的設計讓程式執行變得生動有趣，學習者可於各個角色的行為動作來檢測程式邏輯概念的正確性，不同於傳統的程式設計軟體，必須在抽象的語法文字中找尋問題所在。

圖 12 角色匯入造型

圖 13 角色匯入聲音

圖 14 組合程式以進行角色控制

2.2.2 拆解、重組程式解題方式

積木式及拼圖式的設計，讓程式的堆疊或銜接變得容易，只要將各個程式碼區塊進行組合，即可將其關係構成一個整體。Piaget 於 1960 年代提出，結構具有以下三種特徵：

一、整體性：結構中的各個部分，其實是以一定的規則進行組合而形成一個整體。

二、轉換性：只要依照一定的規則，此整體不會因為順序的更動而改變結構本身。

三、自律性：組成結構的各個部分可劃分為一個個的整體，不受外在因素影響。

但程式結構並非固定不變，隨著不同程式碼的差別組成，結構也會跟著改變，Scratch 除了具備合併程式模組的功能外，更重要的是可以透過現成範例進行拆解程式區塊再進行重組，學習者可單獨針對某個物件撰寫程式後再組合起來，或者先組織程式架構再分段設計，適合不論是由上而下(Top-Down)或由下而上(Bottom-Up)的設計策略。

程式結構的變化更有彈性和創意的發揮空間，學習者能在拆解的過程中嘗試出各個程式碼的功能和配對方式，在觀看現成範例時也能直接進行複製或修改。

Jacques Derrida 於 1967 年提出解構理論，融合 Nietzsche 的反傳統思想，以 Heidegger 的「毀壞」概念出發，認為結構主義之穩定、單一意義是不存在的。以解構理論為基礎的教學策略，將教學分為四個歷程階段:解構、思考、批判與再建構(如圖 15)。在解構階段，學習者發揮「增殖」和「繁衍」的功效，以拆解的過程將各個程式模組深入探究，自行經驗出其程式碼代表意涵(Adams, 1996)，在學生自我分析討論時，教師不需加以指正或引導，反而讓學生於拆解的過程中逐漸釐清真正的意義;思考階段讓學習者從反思的過程中建置出更符合需求的結構，具備改造力;批判階段則是針對其差異與變動提出合理的質疑和評斷;最後於重構階段將被拆解的元素重新尋找連結，並延伸成新的詮譯。

圖 15　解構理論學習歷程

2.2.3 範例式學習－觀察、模仿

Scratch 提供 Web2.0 網站(如圖 16)，只要是使用 Scratch 軟體開發的學習者，皆可以將自己的設計作品上傳至網站上，並分享給其他學習者，透過觀察他人的範例，從程式執行結果來推導程式設計結構，當面對類似問題時，可以依據之前觀察所得的訊息，作為新行為的指引，此模仿學習可達到良好的學習成效(Bandura, 1973)。

圖 16 Scratch 分享網站

Scratch 自由軟體的特性加上程式創作的分享機制，讓學習者可下載和合理範圍內使用他人作品，符合創用 CC 的概念(如圖 17)，只要在作品上加上原著作者姓名、非商業用途，並以相同方式分享，即可對該程式進行再改造。而提供範例之程式創作者遍佈各個年齡層，學習者很容易可以挑選與自己年紀相符的程式作品，提高程式的可讀性與創作的信心，藉由與其他高能力或更有經驗的同儕互動下，所達到的超越性發展（如圖 18)(Vygotsky, 1978)。

圖 17 創用 CC 標註

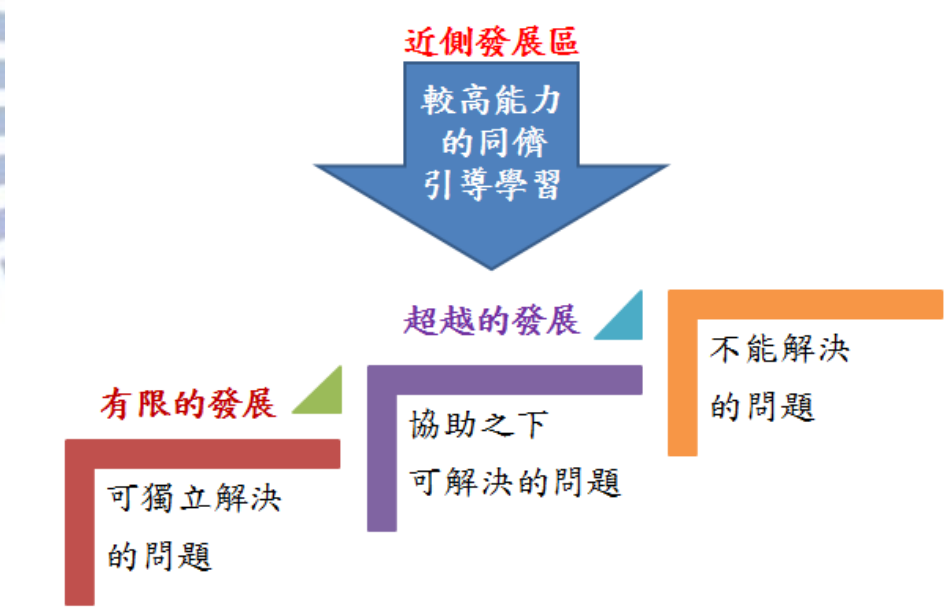


圖 18 近側發展區

2.2.4 自主學習

學習者藉由討論功能來延展所學，成為另一種合作學習的方式(如圖 18)。Scratch 讓程式設計不只侷限在個人的邏輯思維，反而藉由群體知識的力量，讓程式設計的學習擁有強大的自主和自學性，學生在嘗試和錯誤中學習，形成以學生為中心的學習模式。相較於傳統程式設計教學模式，增加了學生學習的彈性、個人化學習以及回饋等(如圖 19)。圖像式的程式設計增加了可讀性，且可重複下載的特性讓學習者不必擔心原始程式遭到破壞，反而能在拆解和組合的過程中推理出新的思維。

圖 19 Scratch 作品評論區

Jeroen, Merrienboer, and Krammer(1987)歸納教導新手進行程式設計最常用的三種教學策略並進行分組實驗，發現閱讀式學習明顯優於專家式以及螺旋式。Scratch便是具備了閱讀式學習的特性，讓程式設計的開發不必從無到有，而能從已經完成的範例程式中找出程式撰寫規則，從中修改並於嘗試和錯誤中訓練邏輯思考能力，也培養學習

者勇於挑戰的能力，並能容易的將想法付諸實行。Scratch與學生興趣或經驗結合，不需任何先備知識，也不必深入探討程式語法，很容易就可以執行，學生可嘗試使用各種不同的組合順序，並馬上看到執行結果(Resnick, et al., 2009)。

Scratch讓學習者可以透過分享的範例程式先看到結果的呈現，有目標和明確可行的信心後，進行程式的拆解和重組，在嘗試不同組合方式中自動產生程式概念，不需要教師特別教學，即可進行自主性學習，符合其「想像」、「程式」、「分享」的設計目的(如圖20)。

圖 20 Scratch 設計目的

2.3 認知風格

2.3.1 認知風格定義

認知風格是指個體在進行有關思考、理解和訊息處理等行為時，所具備的習慣和特質，是恆常、穩定的，不因時間或環境而有重大轉變，可以經由個體學習的過程當中觀察(R. Riding & Cheema, 1991)。Kelly(1955)認為，個體就像電腦一樣，針對外在環境或事件會有不同的訊息接收、儲存和處理特色，當面對問題時，個體會主動解釋並建構一套理論或假設以進行預測和控制。當個體受到外界刺激時，會經由視、聽、思考等過程，將內在歷程轉化為外顯行為，此轉換過程與認知構造有關且呈一致性(Neisser, 1967)。Messick(1976)提出，認知風格是個體偏好組織、處理訊息與經驗的方式，是一種人格特質，無好壞之分，且穩定不易改變，會影響到個體於日常生活的所有行為。

認知風格依不同學者提出的看法而有不同定義，但共通點皆是認為與個人處理訊息有關、且是呈現一致不易改變的，整理如表 7。

表 7 認知風格定義

Allport	1937	能評鑑出不同的人格或行為類型
Kuhlen	1968	個體面對認知工作或學習情境時，所採取的應對方式
Witkin	1976	個人收集和組織訊息的方式
Messick	1976	是知覺、思考、問題解決和記憶的貫性模式
Goldstein and Blackman	1978	面對環境刺激時，個體特有的風格
Wardell and Royce	1978	整理外在世界或內在刺激的一種自我控制之心理結構
Kuchinskas	1979	對環境反應、行為或適應的方式
Curry	1983	個人認知偏好不因時間或環境而有重大改變
張春興	1989	個體在認知活動中，表現在性格上的差異
Riding	1991	人的認知偏好其實是個體習慣或擅長處理訊息的方式，若給予相符的學習環境，可減低處理的負擔
Jonassen and Grabowski	1993	面對外界環境的刺激時，個體會因認知特性的影響而產生不同反應

2.3.2 認知風格分類

初學者在程式設計的解題歷程上，會依據學習者本身的認知處理習慣進行資訊的擷取，此歷程往往成為程式是否設計成功的重要關鍵，但卻不易被初學者所察覺。對於認知風格類型的區分，因學者研究目的、所持觀點或分析層面的不同，而有多達二十幾種類別。以下就常見的認知風格分類方式說明如下：

一、場地獨立（Field Independent）與場地依賴（Field Dependent）

場地獨立／場地依賴型的研究始於 Witkin 和 Goodenough(1977)對視空間知覺(Perception of Visual Space)的研究，最初是用於軍事需要。他們設計了三種有關心理分化的實驗，利用軀體調整測驗(Body-Adjustment Test,BAT)、桿框測驗(Rod andFrame Test,RFT)或旋轉屋測驗(Rotating Room Test,RRT)進行測量個體間在知覺方面受到周圍場地影響的情形，至 1950 年，又以「藏圖測驗」(Embedded Figures Test，EFT)驗證場地獨立性理論。測驗工具說明如表 8：

表 8 認知風格測驗工具

測驗名稱	測驗方式	說明
軀體調整測驗(BAT)	在一個可控制傾斜度的小實驗室中，受試者坐在一張可左右傾斜的椅子上，當實驗室傾斜時，受試者須調整自己身體位置使成垂直	瞭解個體是由傾斜的實驗室來調整身體位置或憑自我知覺來調整。
桿框測驗(RFT)	在黑暗的實驗室中放置會發光的亮框和亮桿，受試者須調整方形框中傾斜的桿子，使其成垂直位置	瞭解個體是參照方形框來調整桿子，或是以自己的感覺來調整成垂直位置。
旋轉屋測驗(RRT)	在一個可在圓形軌道上旋轉的小屋中，受試者坐在一張可調整傾斜角度的椅子上，當小屋旋轉時，由於離心力和地心引力的同時作用使小屋產生傾斜的感覺，受試者須調整椅子角度使其與地面垂直	瞭解個體是以小屋或自身感覺為參照點來調整成垂直位置。
藏圖測驗(EFT)	個體必須在錯綜複雜的圖形中辨識出藏在圖中的簡單圖形	瞭解個體的知覺辨析能力

BAT、RFT 及 RRT 測驗皆是用於瞭解個體在空間中，對垂直方位的知覺，均是要求受試者處理組織場域中的某個部分，用以觀察個體受場域影響的程度，雖然可用於測量個體認知風格屬於場地獨立或場地依賴型態，但受試方式不易，因此後來發展出 EFT 的紙筆測驗方式，此四種測驗方式基本結構相同，在 BAT、RFT 及 RRT 中所測得的位置愈正確，則在 EFT 上所花費時間愈少，表示個體知覺較容易從情境中分離出重要的元素，認知風格偏向場地獨立取向。

Witkin 等人認為，人會因所處環境的影響而使得知覺產生差異，於 1971 年以 EFT 為基礎編製出團體嵌圖測驗（Group Embedded Figures Test, GEFT），其差別在於 GEFT 是用於團體施測，此種測量方式也是較為被廣泛使用的。測驗共有 25 題，將簡單的幾何圖形隱藏在複雜的圖形中，讓圖形不易辨試，受試者測驗得分是依據找出簡單圖形的時間，所費時間愈少則得分愈高，表示受試者能不受複雜圖形所影響，洞見目標的能力較強，屬於場地獨立型（Witkin et al., 1977)。

場地獨立型的學習者，能從複雜背景中以各種方式分化出物件，知覺判斷較少受到外在環境干擾，專注不易分心，且學習目標明確，不會因學習情境的改變而對學習過程產生影響，會依據內在的參照架構來工作，較能夠獨立完成作品。分析能力較強，能夠從複雜的題目中將重要問題抽離出來，對於缺乏結構性的資訊可以重構和統整組織。場地獨立型的學習者則會因學習工具或學習環境的刺激而使得學習策略改變，習慣用整體的觀點來獲得概念，不容易發現細微的差異，適合較具結構性的教學方式，以清楚不籠統的教法來達到較好的學習效果(如表 9)。

表 9 場地獨立／場地依賴特質分析表

場地獨立	場地依賴
參考內在架構產生明確的學習目標	需要外在刺激或教師詳細說明教學目標
能從複雜的環境脈絡中分化出重要的物件	將背景和物件視為一整體，不易發現相似的差異
主動積極、專心	較被動，容易受干擾
能自主性學習	需要固定的學習方式或外部指引，否則容易產生迷失
分析、統整、組織能力佳	以整體觀點來獲得概念
能獨立完成作品，不需給予太多規定	結構性的教學方式
喜歡發現式學習	喜歡引導式學習
喜歡嘗試新的工作，不需要教師協助	缺少冒險精神

研究指出，認知風格類型偏向場地獨立取向的學生，在程式設計的學習成效高於場地依賴取向的學生(Catherine,1992)。羅芝芸（1999）在認知風格、情緒智力與問題解決能力關係的研究中也發現，問題解決能力會因不同的認知風格而有所差異，但此差異也顯示，並非場地獨立偏好的學生，其問題解決能力絕對優於場地依賴取向的學生。Clark & Yinger(1979)在研究認知風格與重建結構的關係時發現，認知風格偏向場地獨立者的學習者，在發現學習材料之隱藏脈絡的能力優於場地依賴者很多。游朝煌(1995)指出大學生在程式設計學科學習上，場地獨立者其學習成效優於場地依賴者。

二、視覺導向（Field Independent）與語文導向（Field Dependent）

Richardson 依學習者處理訊息的偏好程度，分成視覺導向和語文導向(Richardson, 1977)。人類的訊息處理方式，是以圖像與文字兩種不同的認知子系統進行，圖像系統

主要以整體、並行、整合的方式來處理和組織訊息；文字則是以個別、循序的、語法的方式。

表 10 視覺／語文導向特質分析表

視覺導向	語文導向
圖像式的學習情境較能引起學習興趣	透過文字獲取訊息
喜歡拼圖式遊戲	喜歡文字性質的遊戲
想像力較豐富，喜歡具體圖像	對複雜的語意符號理解力佳，運用文字能力強
對抽象概念較不易理解	訊息處理時，偏好文字表徵方式
較主觀，自我導向	較客觀，任務導向

學習歷程會因學習者在處理及組織訊息時的認知風格偏好而受影響(Paivio, 1991)。視覺導向和語文導向特質由研究者自行整理分析如表 10。視覺導向認知風格偏好，擅於理解與處理視覺化訊息，喜歡經由具像的實體來獲取知識，較為主觀、自我導向，幻想的情境清晰，能靈活的操弄與轉化影像訊息；語文導向認知風格偏好，則是在運用文字的能力較流暢，能輕易理解複雜的語意訊息，較客觀、任務導向，在語意符號的轉化能力較強(Jonassen & Grabowski, 1993)。

Cronbach & Snow(1977)認為，依照學生的認知風格類型，給予相對應的教學教材，能讓學習者產生不同的學習成效，例如提供圖片或影片讓視覺偏好的學習者能更流暢的擷取訊息。張素芬(2010)將學生分成視覺與文字導向兩種認知風格，進行動畫與程式遊戲作品的同儕互評，研究發現，在認知風格中屬於視覺導向的學習者，產出作品的構圖層面上，較語文導向的學生更具優勢，若能依學生的特性，提供合適的畫面結構與組成的學習，可以達到適性化教學的目的。

三、文字－圖像(Verbal－Imagery,V-I)和整體－分析(Wholist－Analytic,W－A)

Riding(1991)歸納所有認知風格類別，認為認知風格只分成兩種類別，以得分高低來判定其歸類，未能完整呈現其偏好程度，因此以「文字－圖像」向度和「整體－分析」向度來分類(如圖 21)。「文字－圖像」向度代表個體在訊息接收的喜好方式，偏好圖像表徵方式的學習者，其心像能力較強，需要具體的圖像來幫助學習。「整體－分析」向度則是代表個體訊息處理的方式，也是訊息組織的程度，分析型的學習者，不易受外在事物影響，較能獨立思考，並習慣將訊息以條列方式做歸納分析，偏好以推理方式來理解問題，傾向使用熟悉的舊有解題方法。而整體型的學習者則是習慣將事物看成一個整體做思考，採同化原則，對於解題順序較不重視，靠直覺快速下決策，但也容易誤判，喜歡尋求不同的解題方式，偏好開放式的問題以及具創造性的工作(Rayner & Riding, 1998)。

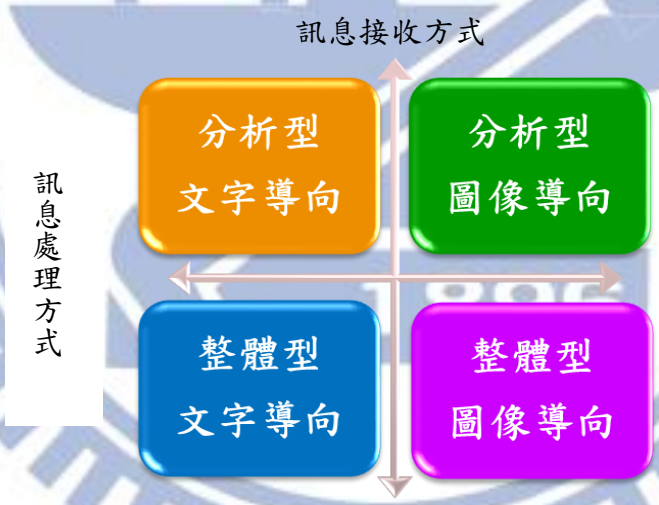


圖 21 CSA 認知風格分類

CSA 認知風格量表使用電腦量測，以反應時間來推論其認知風格，計分客觀準確。其「文字－圖像」向度的折半信度為 0.36，在「整體－分析」向度上的折半信度為 0.69(Peterson, Deary, & Austein,2001)，且此兩項向度間的相關係數為-0.01，與智力、性別等無相關，表示具有良好結構效度。但對臺灣地區的受測者而言，仍存在文化

上及語言上的差異，使得此種認知風格衡鑑方式在施測對象的適切性以及數據分析的方法上尚待商確。

在認知風格相關的分類中，一般教育研究上皆以「場地獨立／場地依賴」作為認知風格之分類標準，此種分類方式可以觀察出學習者如何在學習環境中，依據所找尋到的訊息來重新建構資訊(許麗玲, 2000)。根據近年來學者文獻研究中指出，場地獨立型的學習者在操作網路媒體的相關輔助工具的表現上較場地依賴型優，但在學習成效上並沒有造成顯著不同。不同的認知風格會影響學生的學習行為，為了使學生達到學習的效果，提供適合其認知風格教學方法、情境與教材組織，應該是學習的重要課題，因此教師在教導學生學習，應先了解學生的認知特質及風格，建立良好的溝通方式，採取有助於學生學習動機的策略，以達到有效的學習。因此場地依賴型的學習者是否可推論為容易受到學習環境影響，使其進行 Scratch 程式設計時的設計策略及解題行為產生差異，這也將成為本研究之目的之一。

第三章 研究方法與設計

Scratch 軟體利用拼圖式的語法組合成程式語言，簡單易用，且提供學習者可以在分享平台上直接下載別人的程式，經過現成範例的拆解、執行偵錯、修改部分程式、複製類似程式……等行為，增加某部份程式功能後再分享與回饋。本研究採用準實驗設計法，自變項為認知風格，依變項為學生學習歷程以及學習成效，探討在 Scratch 軟體學習情境下，學生的學習歷程和學習成效是否有差異。本章節針對研究架構、研究對象、研究工具、實驗設計及流程分別說明。

3.1 研究架構

研究活動在 Scratch 學習情境下進行，利用 Scratch 可拆解、分享及自學之特性，以學生認知風格為自變項，學生學習歷程及學習成效為依變項，探討不同認知風格學生其學習歷程和學習成效是否有差異，研究架構如圖 22。

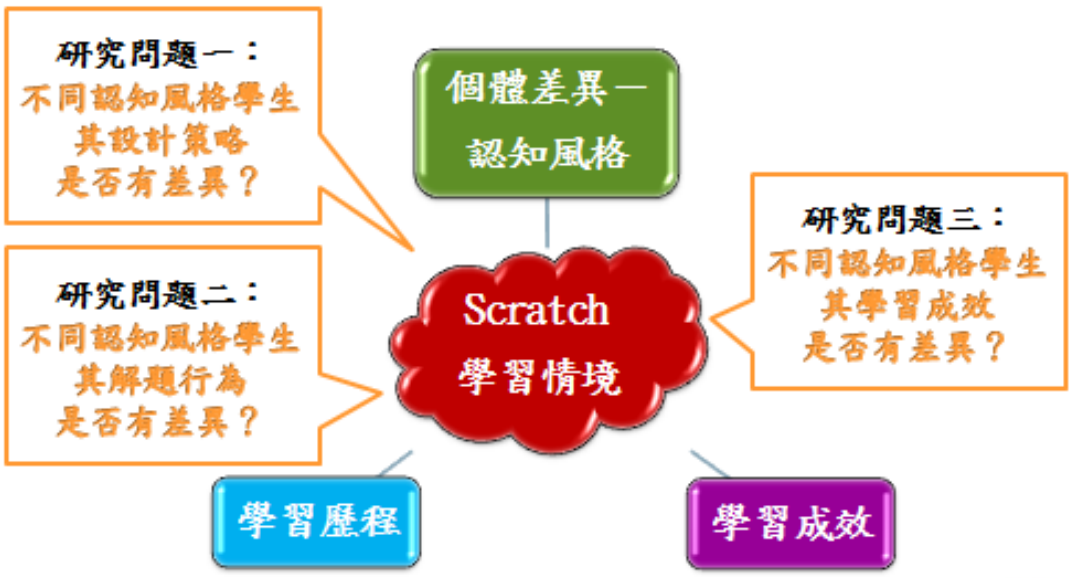


圖 22 研究架構圖

學習歷程以設計策略和解題行為為變項，其中，設計策略又分為 Top-Down(由上而下)和 Bottom-Up(由下而上)，而解題行為分為結構和解構；學習成效則以學習任務之得分為依據，各研究變項之關係如圖 23。

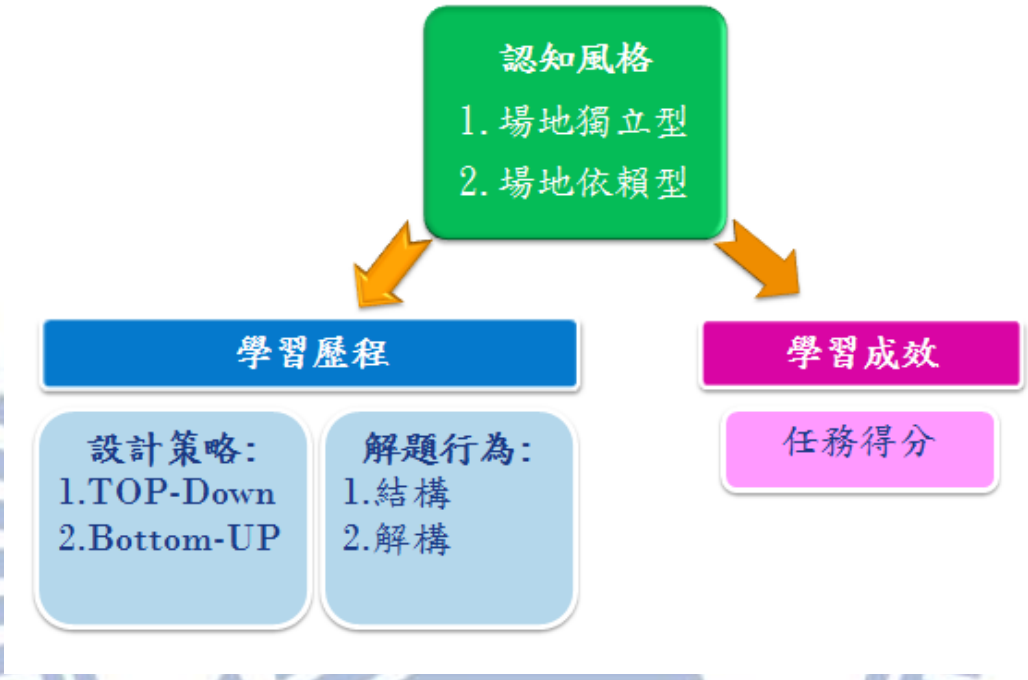


圖 23 研究變項關係圖

3.2 研究對象

研究樣本以某國中九年級學生，三班共 87 人進行實驗，扣除研究期間曾經請假，無法全程參與施測、螢幕錄製軟體異常或作品上傳失敗之人數，其中有效樣本為 75 人。先以團體嵌圖量表進行施測，將樣本分為場地獨立型與場地依賴型認知偏好兩組。為配合實際教學情況，不將研究對象做隨機分派處理，雖然分成兩組，但隨班進行研究活動。

國中以前的資訊課程並未有程式設計相關的介紹，在程式設計的先備知識較為缺乏且程度一致，僅能從 Scratch 程式設計軟體中取得相關資訊，較不受其它學科之影響，符合程式設計初學者的條件。

3.3 研究工具

本實驗使用之研究工具有 Scratch 軟體、認知風格量表、學習歷程記錄軟體、學習任務及問卷調查等。分述如下：

一、Scratch 軟體：

為自由軟體，由麻省理工學院(Massachusetts Institute of Technology, MIT)所開發的視覺化程式設計軟體，適合初次學習程式設計語言的新手，使用的版本為 1.4。選擇此軟體的原因，除了免費、中文版、視覺化介面和積木式結構設計，受測者只要拖曳滑鼠便能輕易將程式組合起來，易用性高，且 Scratch 本身所提供之社群網站(如圖 24)，讓學習者可以從觀摩、仿效來達到自學效果。研究期間由學生自行至 Scratch 分享網站註冊帳號，並進行搜尋和下載他人作品進行修改(如圖 25)。

圖 24 Scratch 網站

圖 25 下載他人作品進行修改

二、認知風格量表：

本研究採用吳裕益於民國 76 年修正 Witkin 所編製的團體嵌圖測驗（Group Figure-Embedded Test, GFET）進行施測，測驗題目分為三大部份，第一部分有七題的練習題，用來讓受測者熟悉題型，測驗時間二分鐘;第二部分和第三部分各為九題的複雜圖形，測驗時間各為五分鐘，受測者必須於規定時間內找出隱藏其中的簡單圖形，每答對一題得一分，總分 18 分，研究者依據測驗結果分成場地獨立及場地依賴偏好，分數越高代表場地獨立性越強。此量表為客觀易行的知覺測驗，可測得個體在認知及心理功能的分化傾向，並用以解釋個體的行為。

團體藏圖測驗初步的常模有效建立在一群東方的美術學院男女大學生上，且只適用於個體同樣來自於相同的團體，採用折半信度的斯比公式(Spearman-Brown formula)得出信度為.82，在性別間，差異雖小，但一般男性比女性傾向場地獨立性。而此測驗之

效度有三，與 EFT 的相關-.82、與 PRFT 的相關-.39、與 ABC(Degree of Body Articulation)的相關.71，其中有兩個數值因計分方式相反而為負數。

測驗結果屬於「場地獨立型」學習者，不易受到周圍環境之影響，學習較主動，在分析、統整及組織能力強，善於重建訊息以產生新的經驗，而「場地依賴型」學習者，不易發現相似的差異。

三、學習歷程記錄軟體：

學生的學習歷程，全程使用螢幕擷取軟體－SCREEN2EXE(如圖 26)，在學生進行程式設計課程前，由研究者先行安裝至學生使用的電腦上，並於實驗中由學生自行操作錄製。

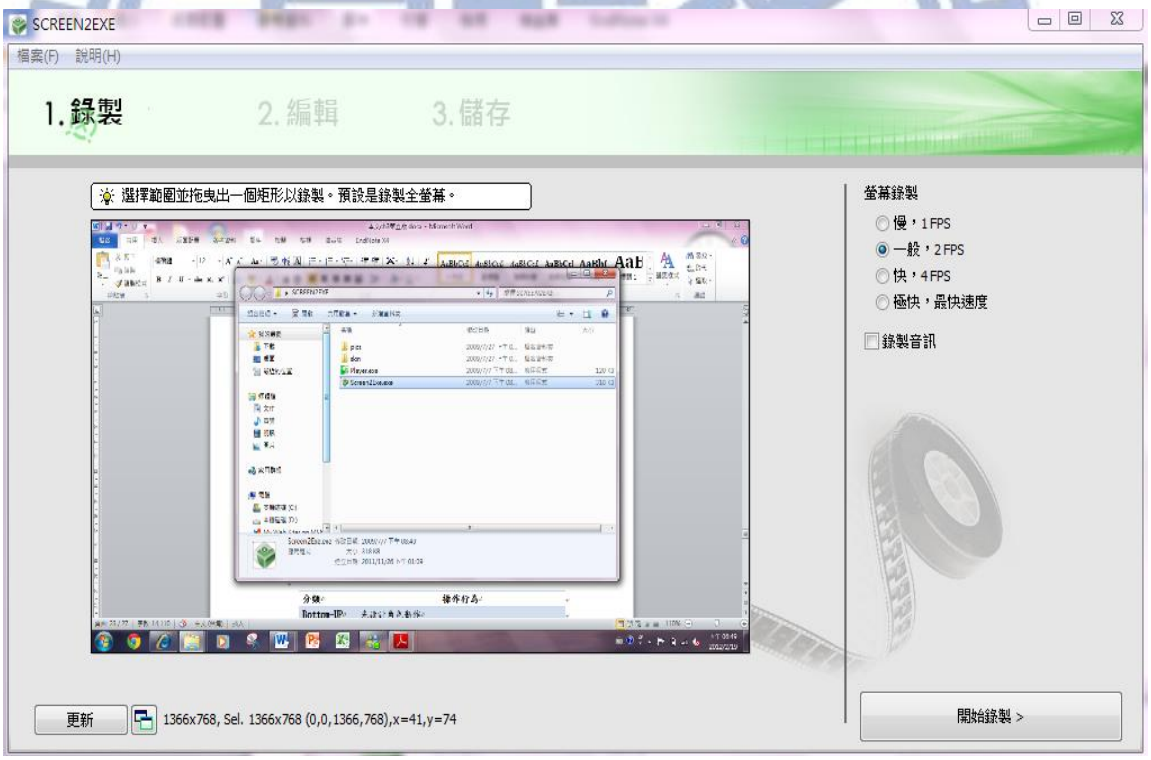


圖 26 螢幕錄製軟體

四、學習任務：

研究者設計一項程式設計學習任務為「搶救人質大作戰」，每項任務細分成十個評量細目，每完成一個評量細目即得一分，總分十分。

學習任務說明如下：

任務名稱：搶救人質大作戰

【玩法】

使用鍵盤的上、下、左、右鍵讓角色前進，當角色碰到人質表示搶救成功，過程當中，可能會有一些蝙蝠出現，若碰到蝙蝠就表示救援任務失敗。

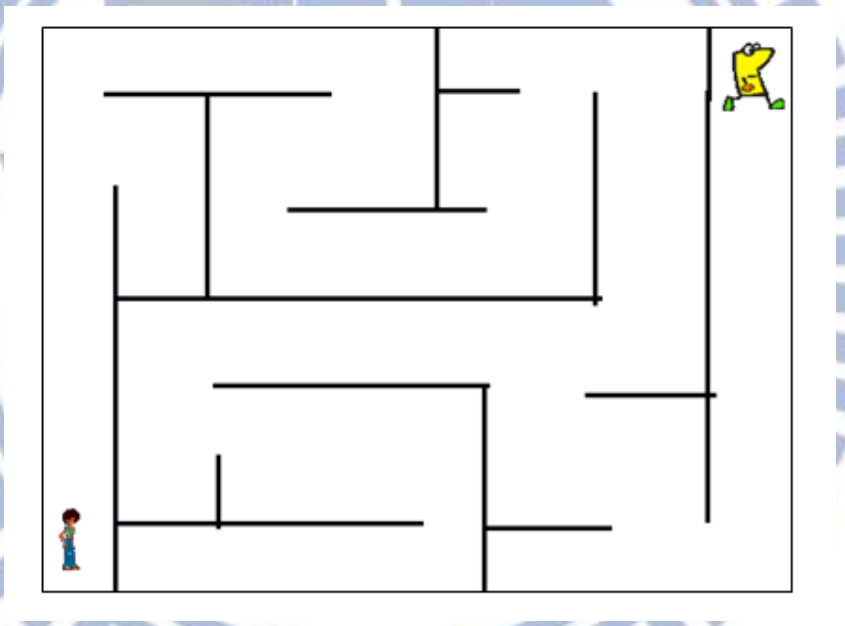


圖 27 學習任務畫面一

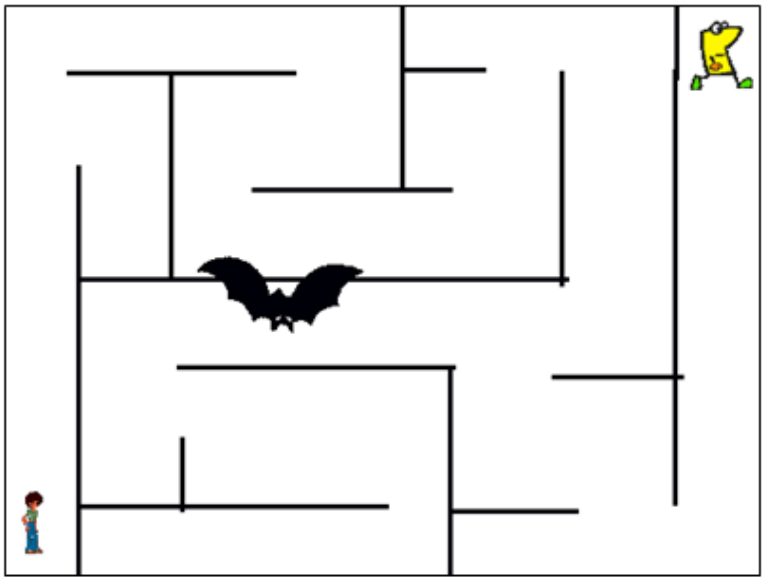


圖 28 學習任務畫面二

表 11 學習任務評分細項表

【評分項目】
1.能設計出迷宮背景
2.利用鍵盤的上、下、左、右鍵可以讓角色移動
3.角色只能在限制的白色區域中行走
4.蝙蝠可以任意飛翔
5.蝙蝠的翅膀可以振動
6.蝙蝠會突然出現或消失
7.會有兩隻以上的蝙蝠，且能達成 4、5、6 項要求－2 分
8.當角色碰到蝙蝠時會顯示任務失敗
9.當角色碰到人質時會顯示救援成功

五、問卷調查：

因此研究主要觀察學生在 Scratch 軟體的探索及自學等操作行為，實驗時間較長，除了使用螢幕錄製軟體外，為避免錄影檔分析時因主觀判斷的不同而產生數據上的差異，在學生完成作品後，以問卷調查之方式得到受測者操作歷程之量化依據，讓實驗結果更客觀，問卷調查分為二部分，第一部分為了解學習者的程式設計策略，分為 Top-Down 和 Bottom-Up 兩種，第二部分依據受試者的解題行為分為結構和解構兩種，資料分析方式說明如下：

1.程式設計策略：

研究者根據學者 Kafai 所提出的 Top-Down 與 Bottom-Up 設計策略之流程示意圖，將學習任務之設計流程假設如圖 29 所示：

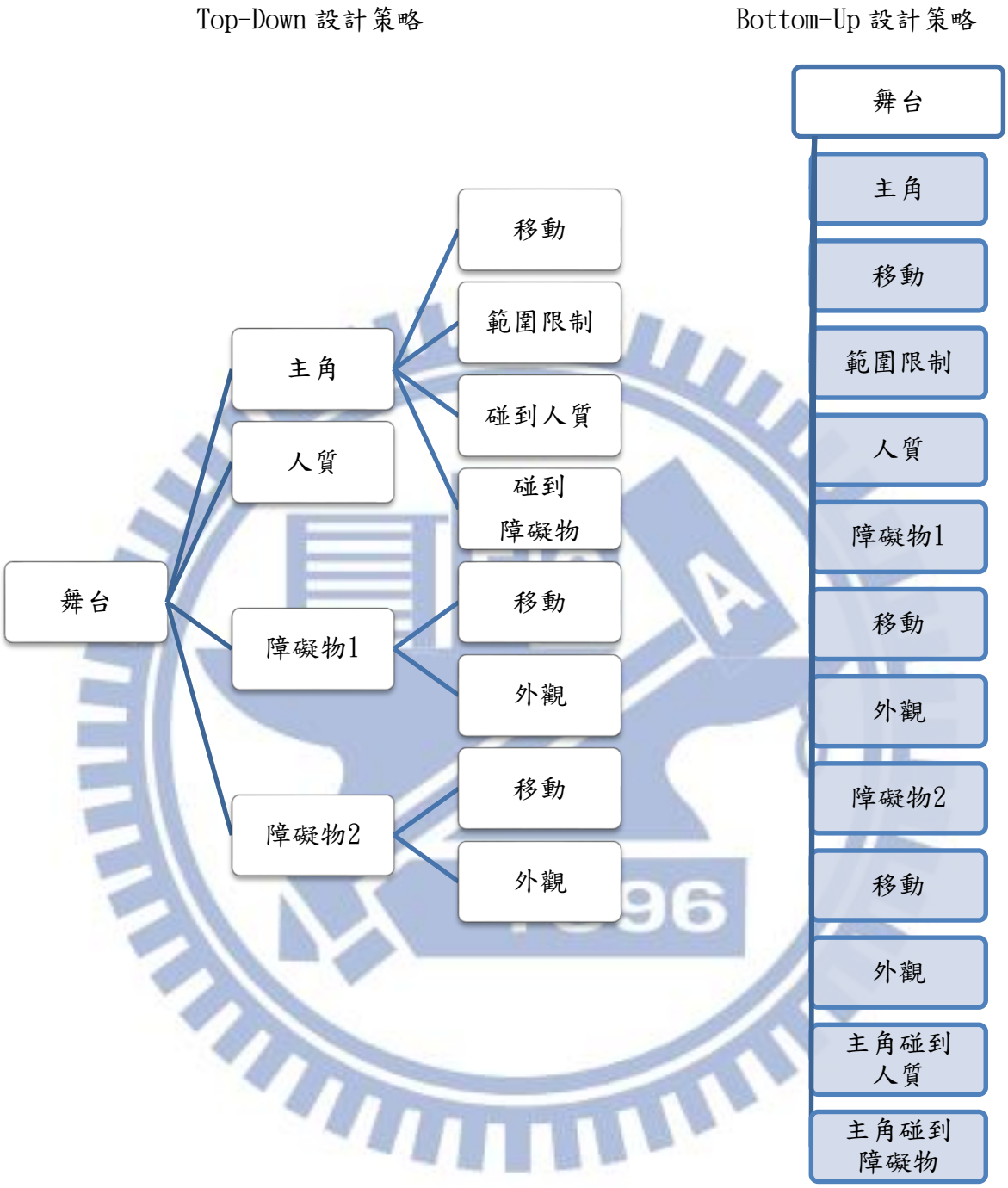


圖 29 學習任務之 Top-Down 與 Bottom-Up 設計流程示意圖

Top-Down 設計策略需要針對問題進行詳細的分析和規劃，將題目劃分成較小的子問題後再進行程式設計，可能符合認知風格中的場地獨立型學習者，在分析、統整及組織能力上較強，此設計策略的學習者在學習任務的解題上可能會先將所需背景、角色等物件都建置好後，先針對各個角色的主要功能進行程式設計，再修改微調次要功能，此外，對於相類似結構的程式也能夠以複製的方式進行設計，再局部修改細部結構，對於整體程式設計的主控性較強。

Bottom-Up 設計策略則是著重在各項細部元件的設計上，待所有元件完整製成後再組合起來，與認知風格中的場地依賴型學習者可能相符，不容易發現細微的差異，因此在學習任務的解題上，因無教師的指引，可能無法將重要部分抽離出來，只能根據題目線索進行程式設計，程式的結構較瑣碎雜亂，對於相似的角色也無法以複製再微調的方式進行修改。

因學習者可能不只採用單一策略，研究者根據自身教學經驗及前導實驗，歸納出表 12 之操作行為，出現次數越多則屬於該項設計策略。

表 12 設計策略分類表

分類	操作行為
Bottom-UP	第一步，先設計角色動作
TOP-Down	第一步，先規劃場景，並將所需角色置入
Bottom-UP	個別完成每個角色所有功能
TOP-Down	先規劃設計每個角色的主要功能，再修改微調次要功能
Bottom-UP	每個角色分別獨立完成
TOP-Down	能利用複製功能完成相同性質的角色

2. 解題行為：

範例式學習不只是模仿現有程式，而是能從他人的程式結構中經驗出新的知識，Scratch 可拆解和重組的特性正是提供範例式學習一個良好的自學情境，從觀察範例程式後進行程式拆解，有助於瞭解各個程式碼間的關係和使用時機。

認知風格偏好場地獨立型的學習者，能從複雜的環境脈絡中分化出重要的物件，可能符合解構解題行為，能參考多個範例並依據學習任務之要求重組成新的程式結構，於拆解的測試過程瞭解各程式碼代表意涵，而重複區塊的運用更是代表對於程式整體結構的統整和歸納。

認知風格偏好場地依賴型的學習者，習慣將事物看成一體，不容易將角色和背景進行分離，也不易察覺各個角色間的關係或相似性，因此其解題行為可能偏向結構，只參考一個範例程式就直接進行修改，對於範例程式上不符學習任務要求之行為無法完全進行刪減或重整，較容易部分的完成題目規定，因此程式結構可能也較為紊亂。

結構和解構的解題行為定義如表 13：

表 13 解題行為分類表

分類	操作行為
解構	參考多個範例程式，將需要的部分擷取出來進行組合
結構	只參考一個範例程式，直接在上面進行修改
解構	利用重複區塊幫助減少程式結構
結構	使用多個初始控制模組，將各個功能分開設計
解構	將程式碼拆解、測試，重新組合以符合題目要求
結構	檢查角色是否符合題目要求，增加／刪減部分功能

3.4 實驗設計

本研究採用準實驗研究法，主要步驟可分為下列四個階段：

一、施測階段：

對所有樣本施測認知風格量表，施測時間約十五分鐘。依不同認知風格類型，挑選出場地獨立／場地依賴兩組。因國中以前並未上過程式設計課程，且受測學生皆無接觸 Scratch 軟體的經驗，因此兩組受試者皆為初學者。

二、準備階段：

每位受測者必須至 Scratch 網站註冊帳號，方便實驗時可以下載他人範例程式，並練習操作使用螢幕錄製軟體來記錄整個學習歷程。為了讓學生熟悉 Scratch 軟體，使用一節課的時間進行 Scratch 基本概念教學，並讓受測者練習及熟悉學習環境。

三、實驗階段：

實驗時間為四節課，根據研究者所提出的學習任務，進行程式設計的實際操作，實驗期間由受試者自行進行範例式學習，教師不加以引導或回答問題，同時以螢幕擷取軟體將學習歷程記錄成影像檔。

四、資料分析階段：

使用問卷調查方式分析受測者之設計策略及解題行為，並透過螢幕錄製軟體，將學習過程中的操作行為量化，此外，研究者對學習任務進行評分，作品完成度越高，得分越高，表示學習成效越好。使用統計軟體 SPSS 18 進行統計分析的工作，以了解不同認知型態受試者在 Scratch 軟體下，其學習歷程及學習成果是否有差異。

實驗流程如圖 30，各階段使用工具及取得資料如表 14 所示。

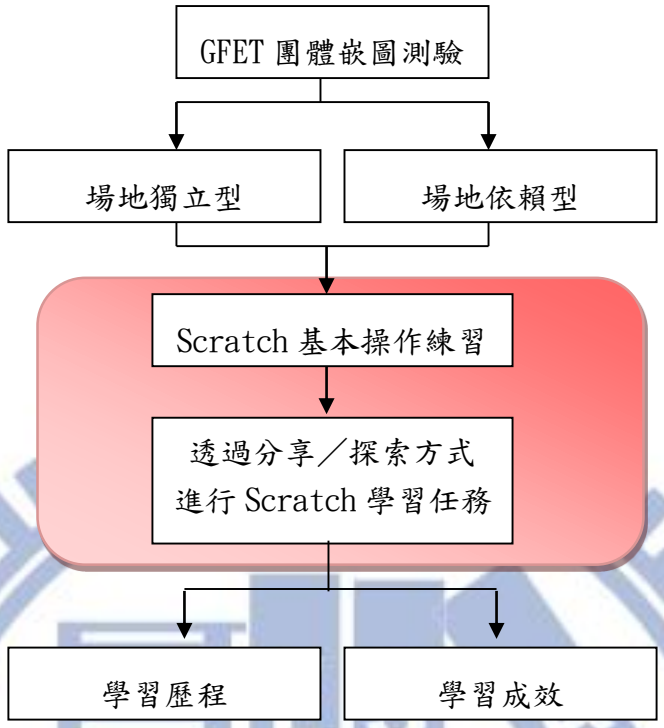


圖 30 實驗流程圖

表 14 實驗各階段使用工具及取得資料

量測項目	**量測工具**	**取得資料**
認知風格偏好	團體嵌圖測驗	場地獨立、場地依賴
學習歷程	螢幕錄製軟體	設計策略(TOP-Down、Bottom-UP)
	問卷調查	解題行為(結構、解構)
學習成效	學習任務	學習任務得分

第四章 資料分析

4.1 描述性統計分析

本研究以團體嵌圖測驗作為認知風格量表，施測結果之平均數為 9.53、標準差為 4.104、中位數為 10，大致呈常態分佈(如圖 31)。採用吳靜吉建議以中位數做為區分之依據，將受試者分成場地獨立及場地依賴二種認知風格偏好，為配合實驗人數之平均，因此以團體嵌圖測驗分數 10 分以上(含 10 分)為場地獨立型、10 分以下為場地依賴型，在有效樣本 75 位中，屬於場地獨立型共計 41 位、場地依賴型共計 34 位。

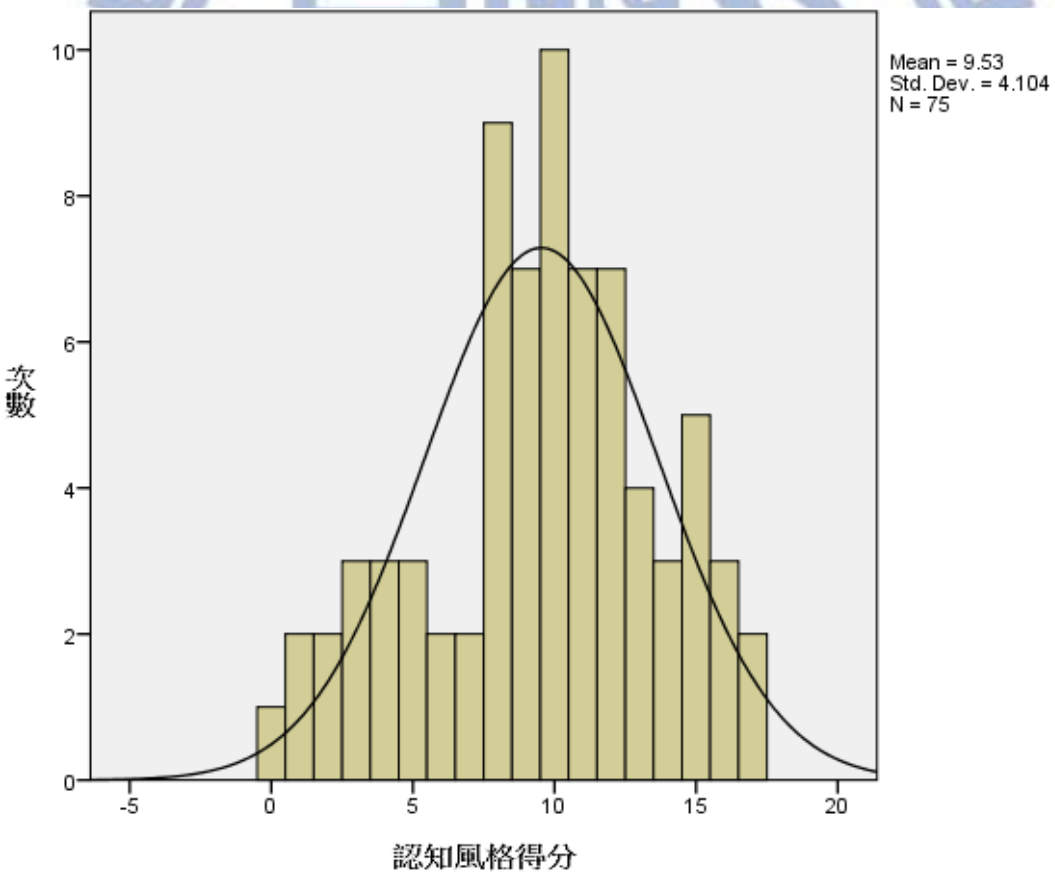


圖 31 認知風格量表得分之次數分配圖

研究者以學習任務得分作為學習成效的評分依據，施測結果之平均數為 5.65、標準差為 2.648，中位數為 5.5，其中，最高分為 10 分(滿分)，而最低分為 0 分(如圖 32)。

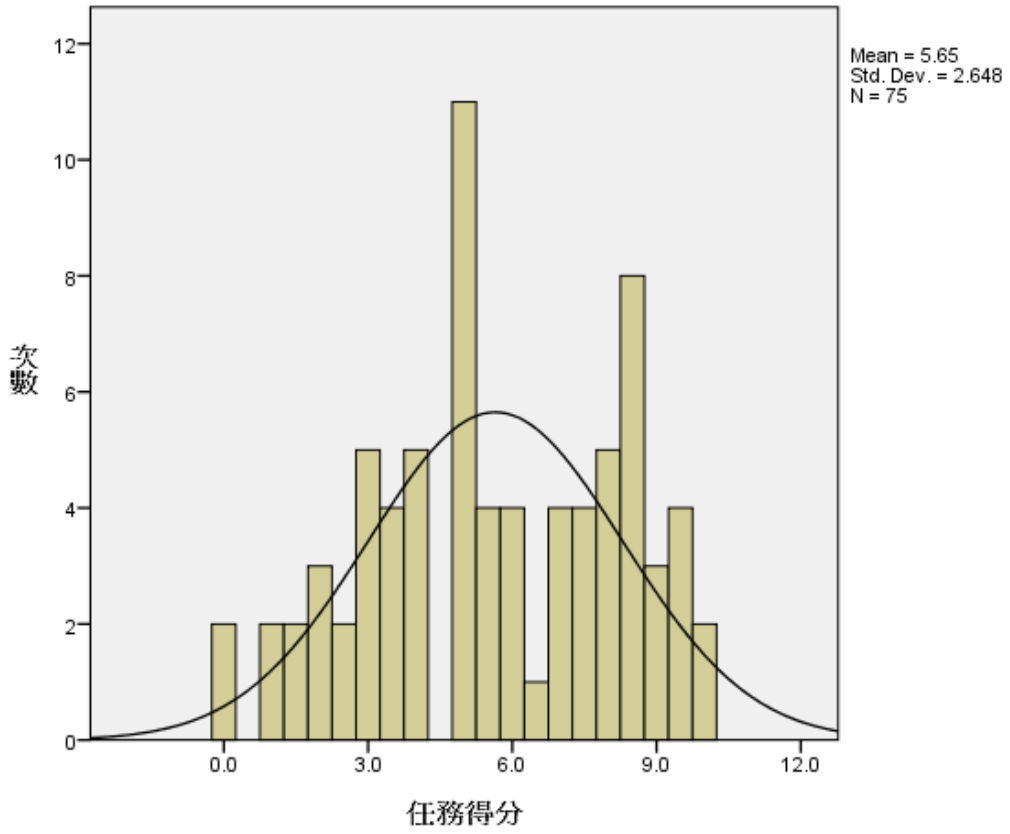


圖 32 學習成效得分之次數分配圖

4.2 使用 Scratch 軟體學習程式設計時，不同認知風格學習者，在設計策略上是否有差異性？

本項分析以「卡方檢定」分析認知風格與使用 Top-Down、Bottom-Up 兩種設計策略的差異性，自變項與依變項皆為名義尺度。分析結果如表 15 所示，場地獨立與場地依賴的設計策略未達顯著，$x^2 = .056$，$p = .812(> .05)$，表示兩組在設計策略上不具有差異性。

表 15 場地獨立與場地依賴在設計策略之卡方考驗摘要分析表

			設計策略		x^2
			TOP-DOWN	Bottom-UP	
認知風格	場地獨立	個數	23	18	.056
		期望個數	23.5	17.5	
		在認知風格分類之內的%	56.1%	43.9%	
		在設計策略之內的%	53.5%	56.3%	
	場地依賴	個數	20	14	
		期望個數	19.5	14.5	
		在認知風格分類之內的%	58.8%	41.2%	
		在設計策略之內的%	46.5%	43.8%	

進一步以「卡方檢定」分析認知風格在主角的設計策略上，使用 Top-Down、Bottom-Up 兩種設計策略的差異性，自變項與依變項皆為名義尺度。分析結果如表 16 所示，場地獨立與場地依賴的設計策略達顯著差異，$x^2 = 6.954$，$p = .008(< .01)$，表示兩組在主角的設計策略上具有差異性。

表 16 不同認知風格在主角設計策略之卡方考驗摘要分析表

			主角設計策略		x^2
			TOP-DOWN	Bottom-UP	
認知風格	場地獨立	個數	14	27	6.954
		期望個數	19.7	21.3	＊＊
		在認知風格分類之內的%	34.1%	65.9%	
		在主角設計策略之內的%	38.9%	69.2%	
	場地依賴	個數	22	12	
		期望個數	16.3	17.7	
		在認知風格分類之內的%	64.7%	35.3%	
		在主角設計策略之內的%	61.1%	30.8%	

＊＊ p < .01

由表 17 可知，Phi(φ)係數=-.305(<0)，顯示兩個變項的關聯具有相反的變動方向，表示認知風格傾向場地獨立的受試者，其主角的設計策略偏向 Bottom-Up，而場地依賴傾向的受試者，其主角的設計策略偏向 Top-Down。

表 17 認知風格與解題行為之對稱性量數表

	數值	顯著性近似值
Phi 值	-.305	.008＊＊

此研究結果與研究假設相反，進一步觀察錄影檔進行分析，發現認知風格偏好場地獨立型之學習者，因是參考現成範例進行部分修改，並非以全新的程式進行設計，花費

較多時間觀察學習任務與範例程式的差異(如圖33)，並修改各角色之細部功能(如圖34)，此種學習方式較符合Bottom-Up之設計策略。

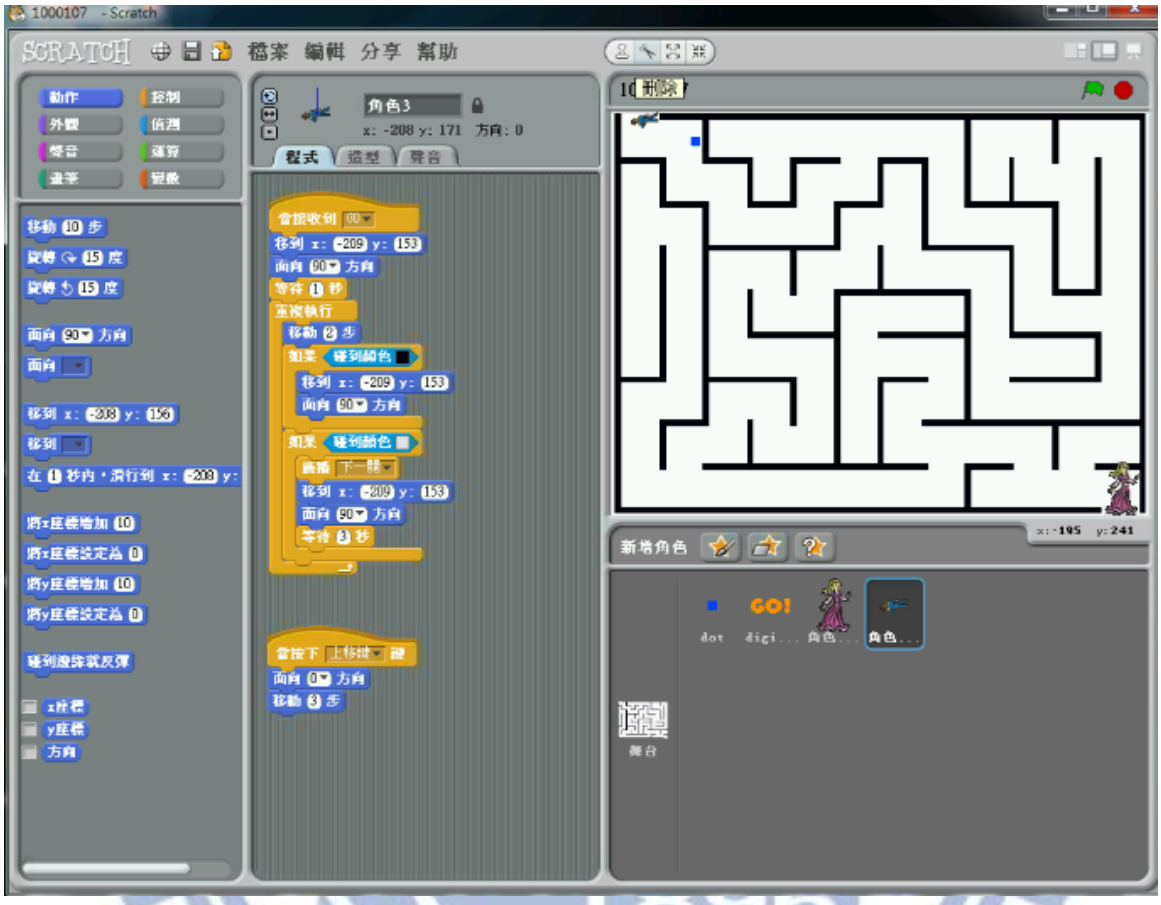


圖 33 Bottom-Up 之設計策略

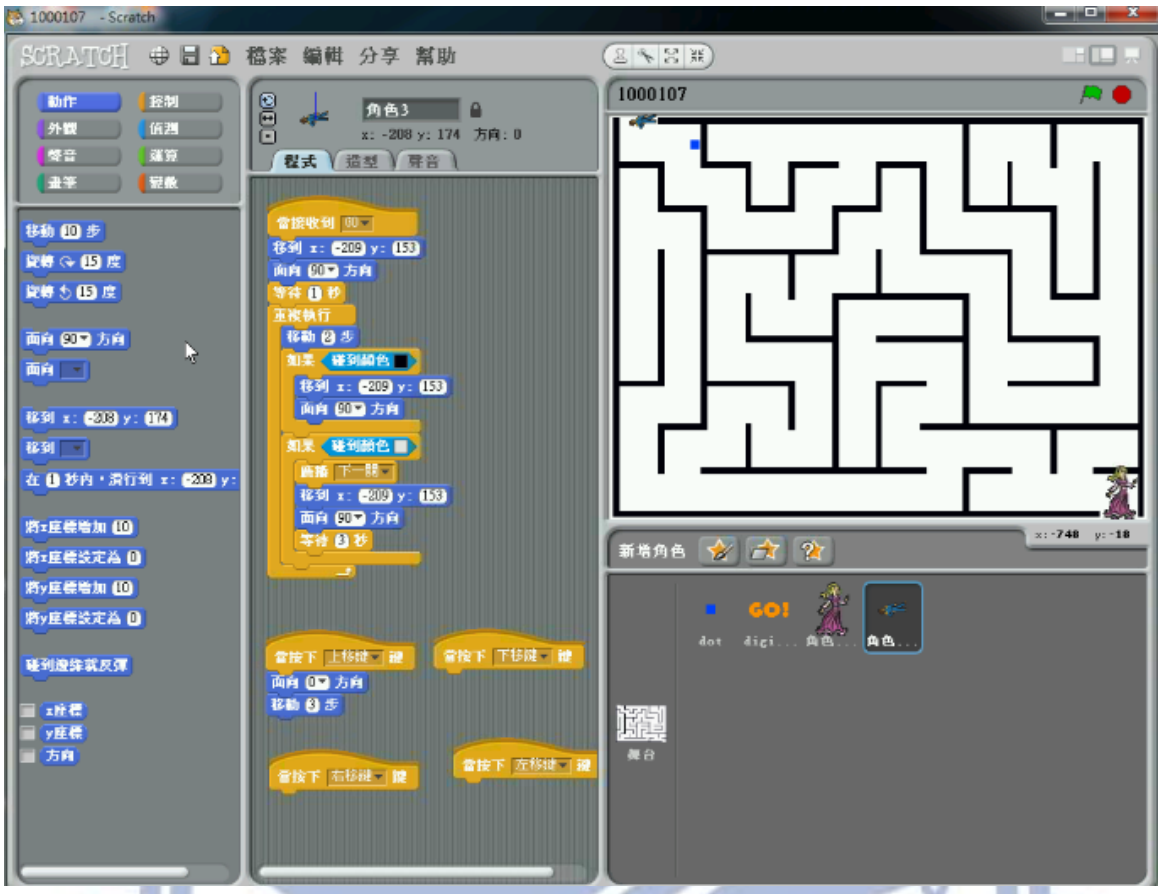


圖 34 Bottom-Up 之設計策略

認知風格偏好場地依賴型之學習者，根據學習任務之要求完成部分，於解題時遇到無法解決的步驟先跳過，因此並無完整、獨立的完成單一角色，但能先將角色主要功能設計出來，例如先將程式所需背景及各個角色準備好(如圖35、圖36)，再針對程式碼進行修改(如圖37)，於是產生Top-Down的設計策略。

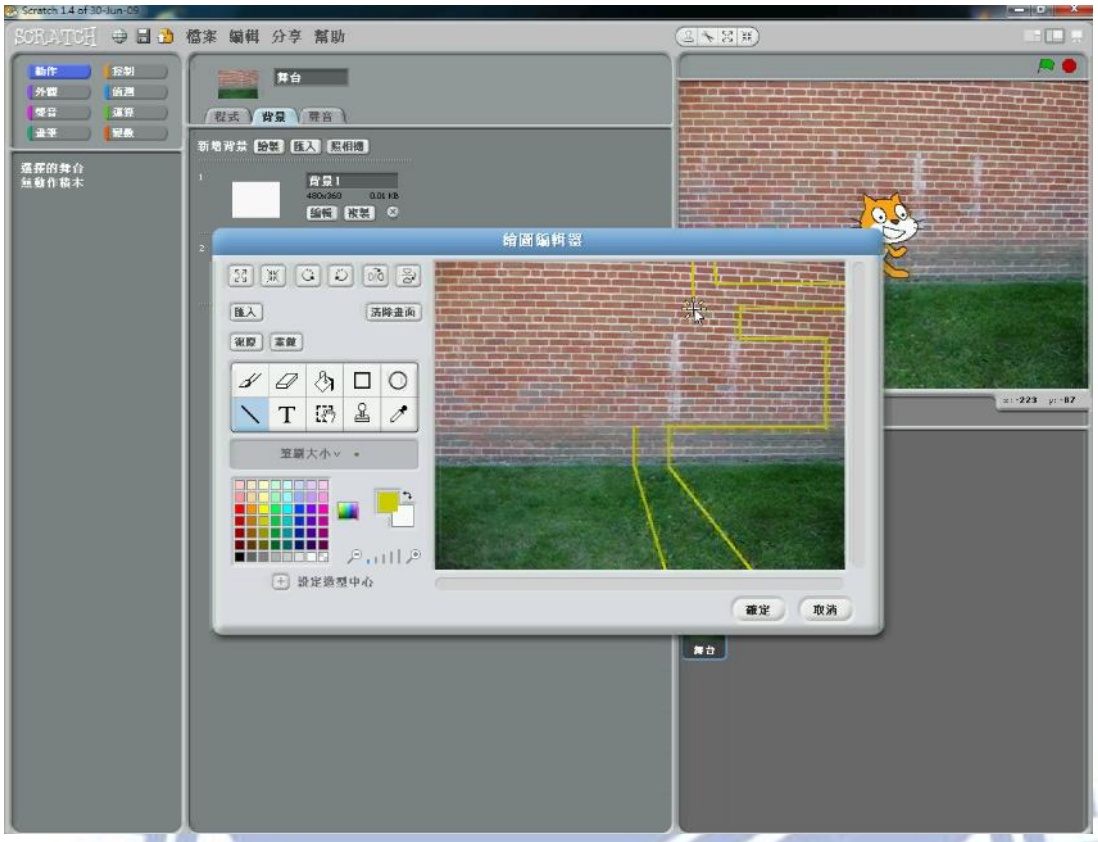


圖 35 TOP-Down 設計策略

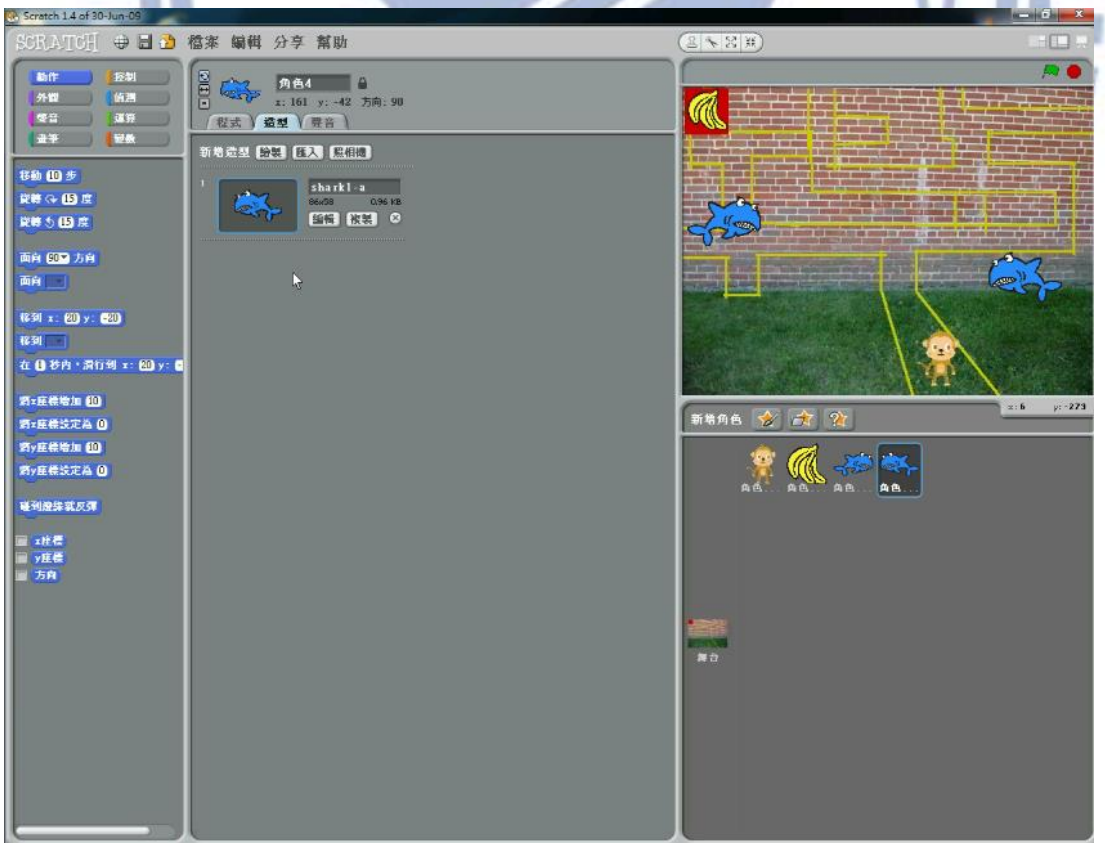


圖 36 TOP-Down 設計策略

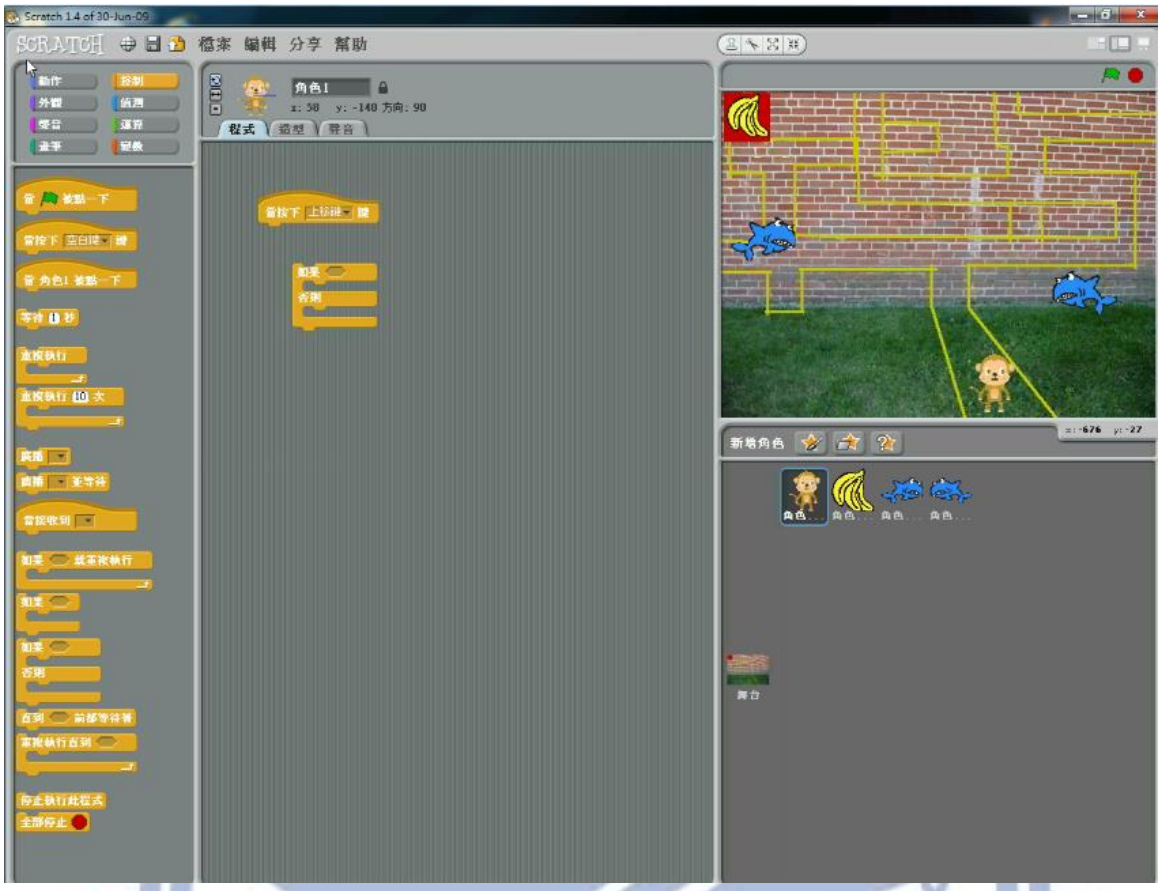


圖 37 TOP-Down 設計策略

可知以範例式進行程式設計學習，不論採用Bottom-Up或Top-Down的設計策略，學習者已可從現成範例觀察程式流程，從專家之解題歷程經驗出程式概念，因此，教師可在此進行程式碼較詳細的解說和差異性的分析上，讓學習者更清楚各個程式碼所具備的功能和代表意涵，教師扮演協助指引的角色。

4.3 使用 Scratch 軟體學習程式設計時，不同認知風格學習者，在解題行為上是否有差異性？

本項分析以「卡方檢定」分析認知風格與使用結構、解構兩種解題行為的差異性，自變項與依變項皆為名義尺度。分析結果如表 18 所示，場地獨立與場地依賴的解題行為達顯著，$x^2 = 6.009$，$p = .014(< .05)$，表示兩組在解題行為上具有差異性。

表 18 不同認知風格在解題行為之卡方考驗摘要分析表

			解題行為		x^2
			解構	結構	
認知風格	場地獨立	個數	21	20	6.009
		期望個數	15.9	25.1	*
		在認知風格分類之內的%	51.2%	48.8%	
		在解題行為之內的%	72.4%	43.5%	
	場地依賴	個數	8	26	
		期望個數	13.1	20.9	
		在認知風格分類之內的%	23.5%	76.5%	
		在解題行為之內的%	27.6%	56.5%	

* $p < .05$

由表 19 可知，Phi(φ)係數=.283(>0)，顯示兩個變項的關聯具有相同的變動方向，表示認知風格傾向場地獨立的受試者，其解題行為偏向解構，而場地依賴傾向的受試者，其解題行為偏向結構。

表 19 認知風格與解題行為之對稱性量數表

	數值	顯著性近似值
Phi 值	.283	.014

場地獨立型偏好的學習者，在進行範例式學習時擁有較強的主動性，觀察範例程式時，能夠將現有程式進行拆解與重組，在程式設計的過程中瞭解各個程式碼之關聯性，有助於日後面對類似問題之解決能力，如圖 38 為受試者對於角色只能在限制範圍內行走進行測試，發現只要碰到白色就會回到原點，為了確定這項假設是否成立，受試者會將程式碼拆解出來進行驗證，而後針對學習任務之要求，下載多個相似範例程式進行觀察，如圖 39 範例的角色可以自由移動，於是受試者將所需部分程式碼自行重組，如圖 40。

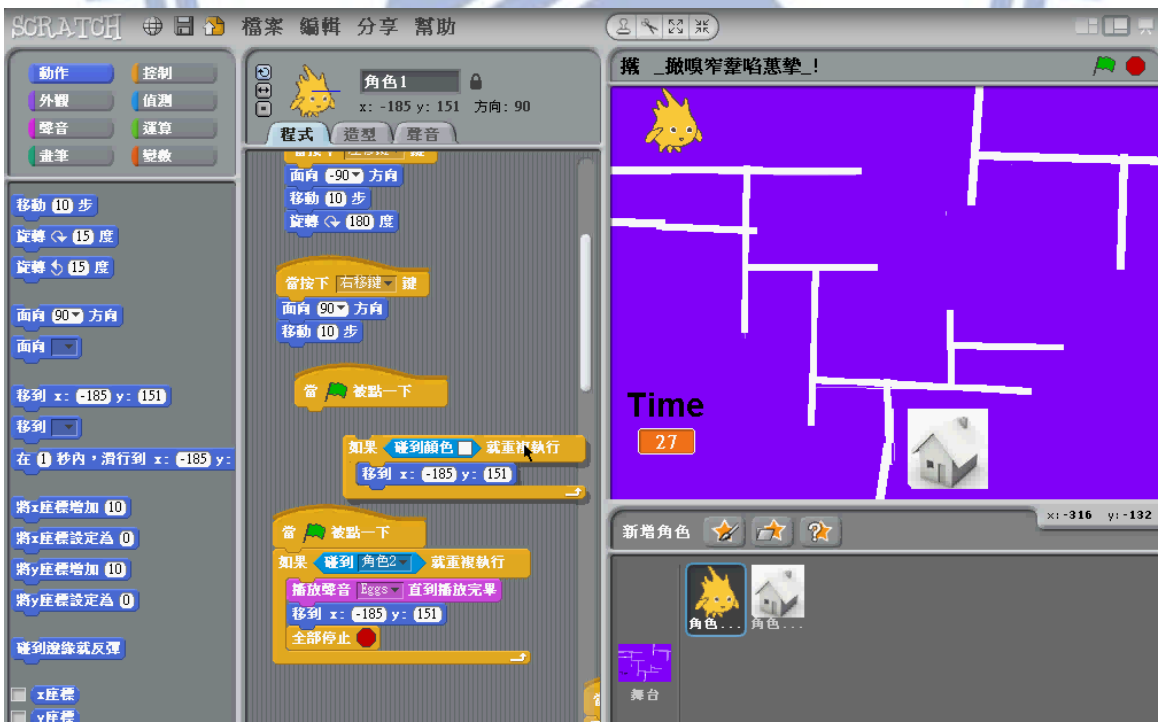


圖 38 場地獨立型學習者之拆解行為

圖 39　場地獨立型學習者參考多個相似範例

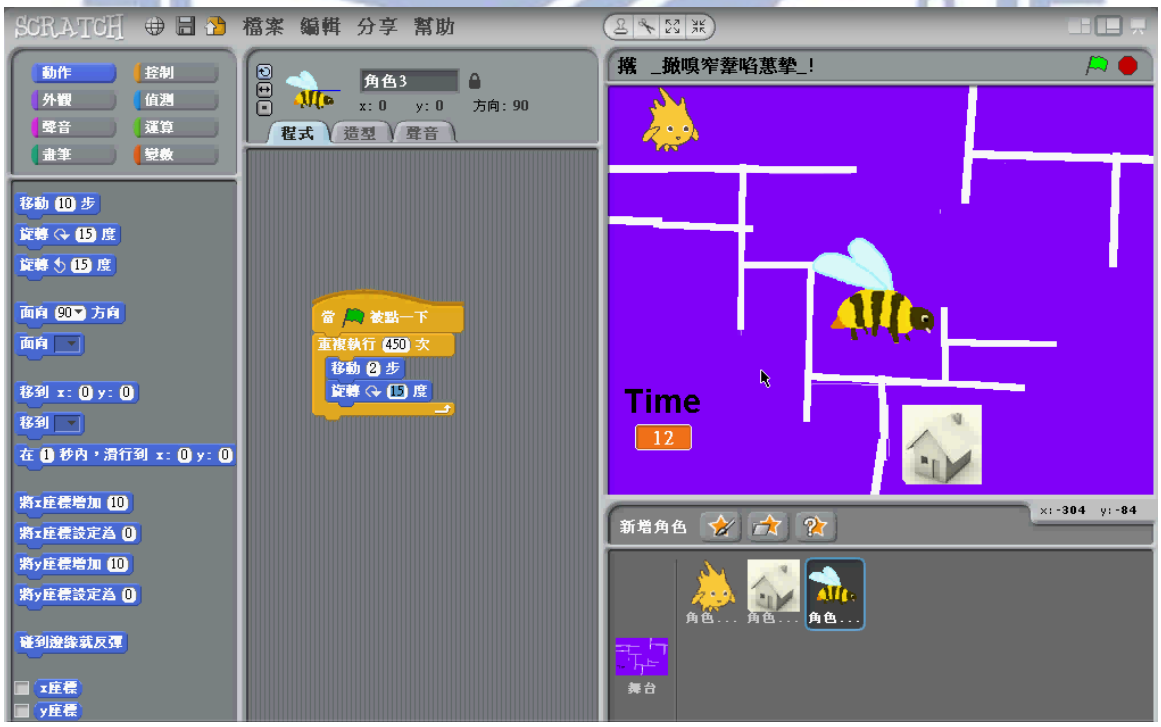


圖 40　場地獨立型學習者參考範例進行程式重組

場地獨立型之學習者，可以透過範例程式找出控制角色的程式流程，再依據學習任務之要求進行程式拆解及重組，如圖 36，受試者發現範例程式之主角一旦超出限制範圍便會停止所有程式，於是開始進行拆解測試，發現與顏色有關，因此將原本的「停止執行此程式」移除後，重新改寫條件(如圖 41、圖 42)。

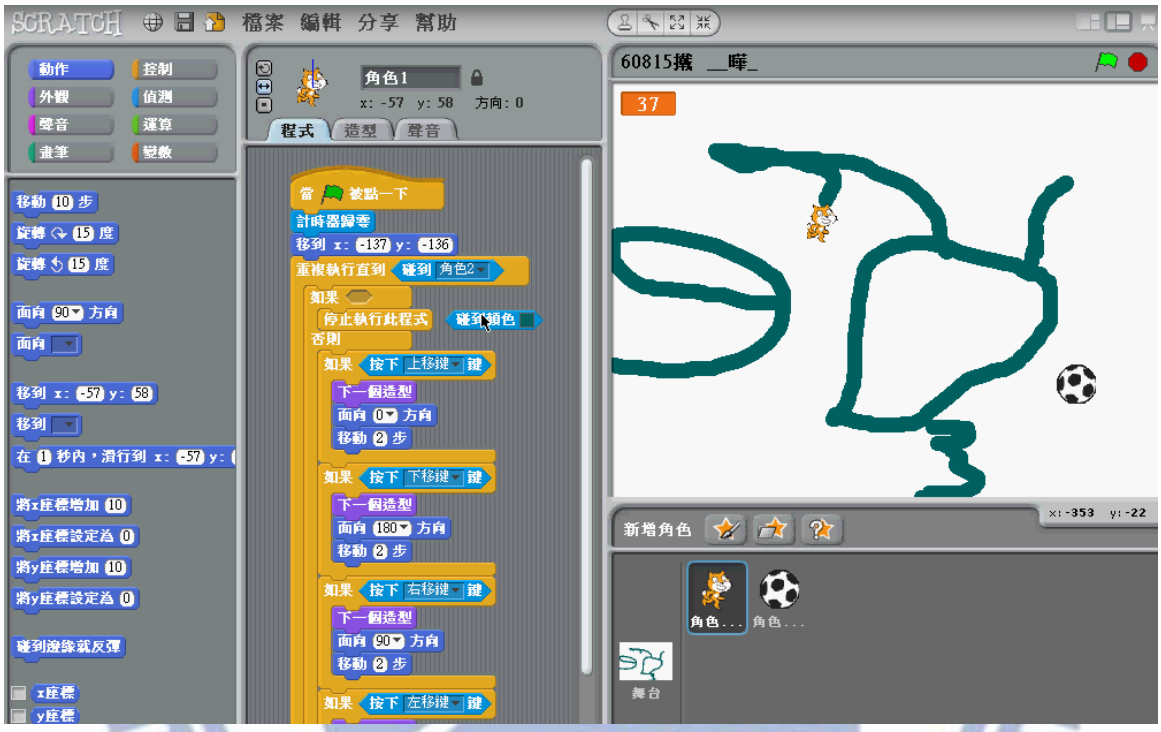


圖 41 場地獨立型學習者之拆解過程

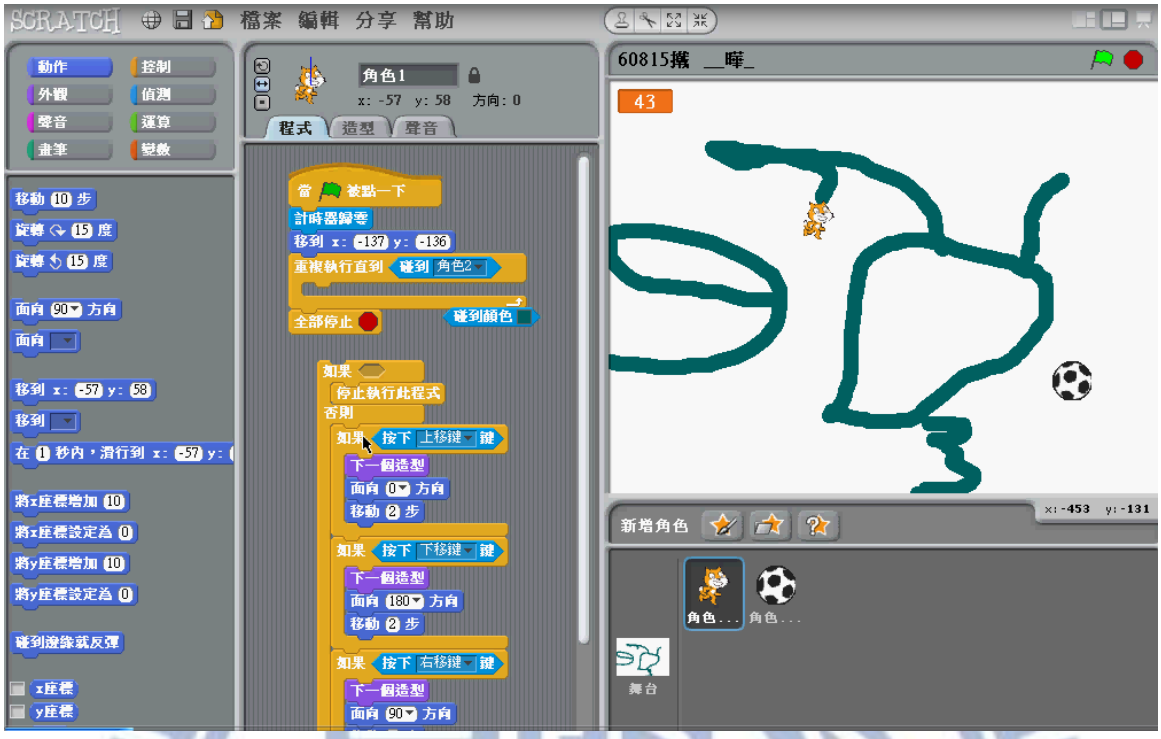


圖 42 場地獨立型學習者之拆解過程

圖 43 場地獨立型學習者之重組過程

而場地依賴型偏好的學習者，因容易受到學習場域之影響，無法將程式各種功能組合抽離出來，對於相似問題或程式結構無法判斷其差異性(如圖 44)，受試者雖然發現範例程式並非使用鍵盤控制角色方向，但並未因此對程式進行拆解，而雖然拆解過程較少，但仍能依題目要求，一步一步將所需功能結構出來，或者直接將範例程式之結構完整的進行複製(如圖 45)。

圖 44 場地依賴型學習者之解題行為

圖 45 場地依賴型學習者之解題行為

以Scratch進行範例式學習時，若能配合學習者之認知風格，給予不同的學習任務，有助於學習者更快達到學習效果。場地獨立型之學習者適合目標導向的學習任務，即能依照現成範例進行拆解和重組，於此過程中學習新知，而場地依賴型之學習者適合階段性目標的學習任務，學習者必須有較明確的具體目標，即可依此要求於範例中自行嘗試組合成新的知識。

4.4 使用 Scratch 軟體學習程式設計時，不同認知風格學習者，其學習成效是否有差異性？

本項分析以「t 檢定」分析認知風格在學習成效的差異性，自變項為名義尺度、依變項為連續尺度。分析結果如表 20 所示，由結果可發現兩組受試者在「任務得分」上未達顯著差異，從平均數來看，場地獨立和場地依賴學習者在得分上皆不相上下，可知認知風格偏好並不會對學習成效造成影響。

表 20 不同認知風格在學習任務得分之 t 檢定分析表

		個數	平均數	標準差	t值	顯著性
任務得分	場地獨立	41	5.378	2.5514	-.964	.338
	場地依賴	34	5.971	2.7632		

進一步分析受試者之操作錄影檔及作品完成檔，發現場地依賴型偏好之學習者，即使可能會產生多餘的行為和角色，仍能將程式依據學習任務完成，大部分是依據學習任務之要求，針對現成範例程式缺乏的行為進行程式增加，較少刪除的部分。而場地獨立型之學習者則是依照學習目標將現成範例程式進行拆解、仿做、重組以達成要求。

由此可知，不論是場地依賴或場地獨立型之學習者，只是解題過程不同，但學習成效並不因此產生差異。教師於此學習過程只是輔助的角色，若能於學生解題中，對於不同認知風格學習者，給予不同的教學策略，會達到更好的學習成效，例如場地依賴型的學習者比較需要結構性的教學和外部指引，適時的提供步驟化教學或分段式教材，而場地獨立型的學習者在達成目標時，可能需要教師提供多元、相似的程式範例以觀察其差異。

第五章 結論與建議

Scratch 提供模組化程式及 web 分享網站，讓學習者能夠藉由下載他人範例後進行拆解及重組，於此學習歷程中瞭解程式意涵，達到自我學習之目的。

本研究經由實驗發現，於 Scratch 學習程式設計時，場地獨立型之學習者偏好使用 Bottom-Up 之設計策略，解題行為傾向解構方式，而場地依賴型之學習者偏好使用 Top-Down 之設計策略，解題行為傾向結構方式，顯示出認知風格的不同確實會使得程式設計的策略和解題行為產生差異。

對於不同認知風格之學習者而言，學習歷程有異但學習成效卻相同，整體來說，Scratch 以範例式自學方式學習，只要讓學習者有充分時間進行拆解、組合，的確有助於自動產生程式概念，於做中學經驗出程式流程、於較有能力的同儕高手中學習出程式經驗和知識。

教師於 Scratch 自學中扮演輔助角色，協助引導學生解題，可根據學習者認知風格的差異，提供不同的學習任務及運用不同的教學策略，產生引導自學的教學模式，讓學習者自行透過設計遊戲來學習。

本研究以量化分析學習者之設計策略及解題行為，建議未來研究可以配合質化研究方法，對於學習者思考歷程進行更進一步的探討，以及學習者在進行拆解過程中的訊息轉換和學習成效之關聯性做研究。

參考文獻

中文文獻

王鼎中、丘聖光、林淑玲、梅文慧、林美娟(2009)。創新程式設計課程與教學模式之研發。論文發表於中華民國第二十四屆科學教育學術研討會，台北市。

朱君怡(2010)。動畫式教學策略對學生創意能力與電腦態度之影響：以物件導向程式設計為例。國立中央大學資訊工程研究所碩士論文。

何昱穎、張智凱、劉寶鈞(2010)。程式設計課程之學習焦慮降低與學習動機維持－以 Scratch 為補救教學工具。*數位學習科技期刊*，*2*(1)，11-32。

何昱穎、許靜坤、王一成、林育伶(2011)。視覺化程式語言之遊戲式數位學習活動設計與評估。資訊教育學門與科學教育學門 99 年度專題計畫聯合成果討論會議。

林輝鐸，莊桓綺，陳怡琴，羅珮妤，鄭郁蝶(2010)。3D 動畫學習環境－以電腦程式語言學習對比學習時間與學習成效之研究。全球商業經營管理學報。頁 45-54。

林曉薇(2010)。維新或危機？－論 Web2.0 及其對教育研究之意涵。*臺北市立教育大學學報*，*41*(1)，171-198。

許宏彰(2005)。國小學童 LOGO 語言程式設計思維歷程之研究。國立臺北教育大學數學教育研究所碩士論文。

許麗玲(2000)。認知風格在虛擬實境遠距學習遷移之影響。國立高雄師範大學工業科技教育學系碩士論文。

許梅君(2008)。不同演練範例呈現對高職生程式設計學習之影響。國立臺灣師範大學資訊教育學系在職進修碩士班碩士論文。

黃文聖（2001）。國小學童在 Logo 學習環境中數學學習與解決之研究。國立新竹師範學院數理研究所碩士論文。

游朝煌(1995)。大學學生空間能力、邏輯思考能力、不同補充教學策略及相關因素對程式設計學習成效影響之研究。國立彰化師範大學工業教育學系碩士論文。

張素芬(2010)。視覺程式語言 Scratch 適性化教學之研究。論文發表於新竹教育大學舉辦之 2010 電腦與網路科技在教育上的應用研討會，新竹縣。

羅芝芸（1998）。兒童認知風格、情緒智力與問題解決能力之相關研究。國立高雄師範大學教育系碩士班碩士論文。

英文文獻

Adams, L. S. (1996). Semiotics2: Structuralism and post-structuralism. *The* methodologies of art（pp.162-178）. Colorado: Westview Press.

Allport, G. W. (1937). *Personality: A psychological interpretation*. New York: Holt.

Bandura, A. (1973). Aggression: A social learning analysis. Englewood Cliffs, N.J.:Prentice-Hall.

Carnegie Mellon University (2006). Press release: Carnegie Mellon collaborates with EA to revolutionize and reinvigorate computer science education in the US. Retrieved December 18, 2007, from http://www.alice.org/simsannounce.html

Catherine, B. C. (1992). Field Independence and programming achievement: A Meta-Analysis. (ERIC Document Reproduction No. ED 348983).

Clark, C., & Yinger, R. (1979). Research on teacher planning, a progress report. Journal of Curriculum Studies, 11 (11), 175-7.

Clements, D.H. (1991) Enhancement of Creativity in Computer Environments,American Educational Research Journal, 28, pp. 173-187.

Clements, D.H. & Gullo, D.F. (1984). Effects of Computer Programming on Young Children's Cognition, *Journal of Educational Psychology,* 76, pp. 1051-1058.

Costelloe, E. (2004). *Teaching Programming The State of the Art*. CRITE Technical Report.

Cronbach,L.J. & Snow,R.E.(1977). Aptitudes and instructional method. New York: Irvington.

Curry, J. A.(1983). On the formation of polar continental air, J. Atmos. Sci., 40, 2279–2292.

Driver, R.& Oldham, V. (1986). A constructivist approach to curriculum development in science. *Studies in Science Education*,*13*,105-122.

Funkhouser, C. P. (1993). The influence of problem-solving software on student attitudes about Mathematics. *Journal of Research on Computing in Education*, *25*(3), 339-346.

Goldstein, K. M., & Blackman, S. (1978). Cognitive style: Five approaches and research. New York: Wiley.

Jeroen J. G, V, Merrienboer & Hein P. M. Krammer(1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science, 16*, 251-285.

Jonassen, D. H. (1993, April). Effects of semantic network versus production rule representation on structural knowledge. Paper presented at the annual conference of the American Educational Research Association, Atlanta, GA.

Kelly, G. A. (1955). The psychology of personal constructs (Vols.1–2). New York: Norton.

Kuchinskas, G. (1979). Whose cognitive style makes the difference? *Educational Leadership*, 36(4), 269-271.

Kurtz, T. E. (1981). BASIC in ACM, History of Programming Languages.

Kuhlen, R. G. (1968). Developmental changes in motivation during the adult years. In B. L.

Neugarten (Ed.), Middle age and aging (pp. 115-1 36). Chicago: University of Chicago Press.

Lewalter, D. (2003). Cognitive strategies for learning from static and dynamic visuals. *Learning and Instruction, 13*, 177-189.

Malan, D. J., & Leitner, H. H. (2007). *Scratch for Budding Computer Scientists.* Paper presented at the Proceedings of the 38th SIGCSE technical symposium on Computer science education.

Many, W. A., Lockard, J., Abrams, P. D., & Friker, W.(1988). The effect of learning to program in Logo on reasoning skills of junior high school students. Journal of Computing Research., 4(2), 203-213.

Mayer, R. E. & Fay, A. L. (1987). A chain of cognitive changes with learning to program in logo. *Journal of Educational Psychology*, 79, 269-279.

Messick, S. (1976). Individuality in Learning. San Franciso, CA: Jossey-Bass, 4-33.

Neisser, U.(1967). Cognitive psychology. New York: Appleton-Century-Crofts.

Paivio, A. (1991). Dual coding theory: Retrospect and current status. Canadian journal of psychology, 45(3), 255-287.

Peterson, E. R., Deary, I. J., & Austin, E. A. (2001). The structure and reliability of Riding's congnitive style analysis test. *Conference 2001 of the European Learning Styles Information Network, University of Glamorgan*, UK.

Prensky, M. (2001). Digital Natives, Digital Immigrants. *On the Horizon*, *9*(5), 1-6.

Rayner & Riding.(1998).Cognitive Styles and Learning Styles: Understanding Style Differences in Learning and Behavior. London: David Fulton Publishers,19-36.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009, Nov.). Scratch－Programming for Everyone. *Communications of the ACM, 52*, 60-67.

R. Mancy and N. Reid.(2004). Aspects of cognitive style and programming. *In E. Dunican and T. Green, editors, Proc. of the 16th Workshop of the Psycology of Programming Interest Group*, 1–9.

Richardson, A. (1977) Verbaliser-Visualiser: A Cognitive Style Dimension, Journal of Mental Imagery, 1, pp. 109-126.

Riding, R.J., & Cheema, I. (1991). Cognitive styles - An overview and integration. *Educational Psychology, 11*, 193-215.

Shafto, S. A. S. (1986), Programming for learning in mathematics and science, *ACM SIGCSE Bulletin, 18*(1), 296-302.

Sweller, J.. & Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. Cognition and Instruction, 2, 59-89.

Vygotsky, L.S. (1978). Mind in a society. Cambridge: Harvard University of Chicago Press.

Wardell, D. & Royce, J. R. (1978). Relationships between cognitive and temperament traits and the concept of "style". Journal of Multivariate Experimental Personality and Clinical Psychology, 1, 244–266.

Webb, N. M. (1984). Microcomputer learning in small groups: Cognitive requirements and group processes. *Journal of Educational Psychology*, 76(*6*), 1076-1088.

Winslow, L. E. (1996). Programming pedagogy: A psychological overview. SIGCSE Bulletin, 28, 17-22.

Witkin,H.A., Moore,C.A., Goodenough,D.R. & Cox,P.W.(1977). Field dependent and field-independent cognitive styles and their educational implications. *Review of Educational Research*, 47, 1-64.