國立臺灣師範大學
資訊教育研究所碩士論文

指導教授：林美娟 博士

Alice 程式設計初學者之錯誤類型分析

An Analysis of Novice Alice Programmers' Error Types

研究生：林冠宇　撰

中華民國一百零二年二月

摘要

Alice 程式設計初學者之錯誤類型分析

林冠宇

本研究旨在探討程式設計初學者於實作 Alice 程式專案時所遭遇之困難，並將其進行分析與歸類。研究實施採用質性研究之一對一診斷性訪談，參與者為臺北市某高級中學一年級的學生共 22 名。研究者設計了涵蓋不同概念、難度漸進的 9 個題目供學生實作，並由訪談者全程觀察學生進行解題的過程，且於學生遭遇解題困難時，依照「引導式提問」、「間接提示」、「直接指導」的漸進順序給予提示，以了解學生在各項程式設計概念的學習困難類型。研究結果歸納出學生於內建方法、時序控制結構、變數、選擇結構、重複結構、陣列、自訂方法及參數中常見的錯誤類型，將其分類為連結、定位、使用以及其他四類，並針對學生實作程式專案時所見之情形，提出使用 Alice 進行程式設計教學時所應注意之事項及相關建議。

關鍵字：Alice 程式設計、錯誤類型、診斷性訪談

Abstract

An Analysis of Novice Alice Programmers' Error Types

Lin Kuan-Yu

This study aims to investigate the difficulties encountered by novice Alice programmers and categorize the types of errors they make. The researcher conducted one-on-one clinical interviews with 22 freshmen at a senior high school. The participants were offered nine Alice projects with increasing levels of difficulty, each requiring the use of a different set of programming constructs. Each participant was observed closely as s/he worked on the given projects. Whenever an error occurred in his/her program or when s/he did not know how to proceed, the observer would intervene by asking a sequence of questions with increasing specificity, ranging from *prompts*, *hints*, to *provides*, in order to determine the cause of the impasse. This study summarizes the common errors for each of the Alice programming constructs that the participants used in their projects and categorized the errors into four types: *connection* errors, *location* errors, *usage* errors and *math-and-other* errors. In view of these error types, this study also gave suggestions for teaching Alice programming.

Keywords：Alice programming、Error types、Clinical interview

誌謝

歷經了四年半，中間夾雜了實習和工作，我總算是要從研究所畢業了。

首先要感謝的是我的指導教授美娟老師，除了在研究上的悉心指導外，更要感謝老師願意包容我的任性和脫線的個性。其次要感謝口試委員吳正己老師與王佩瑜老師對於論文提出許多精闢的見解與建議，使我的論文更加完善。

從規劃到實際進行實驗的過程中，感謝秋良學姊、慧君學姐和文慧學姊提供他們的專業知識和經驗，協助我順利完成整個實驗。在整理研究資料及撰寫論文的時候，感謝舜尹學姐提供我許多寶貴的經驗跟建議，協助我克服許多困難。還有實驗室的夥伴宗緣學長、宜岑學姊、信育、曉薇、文良、怡雯、孝齊、傑安，特別是和我一起拚這一個學期畢業的胤廷的鼓勵、打氣和吐槽，讓我在能夠獲得堅持下去的動力。

最後要感謝父母的不斷催促跟日常的關心，不僅讓我能夠沒有後顧之憂地在研究領域中鑽研知識；也在我想要放棄時給予我鼓勵，促使我回到校園來完成這本應早該完成的學業。

冠宇 謹誌

中華民國 102 年 2 月

目錄

附表目錄

附圖目錄

第一章　緒論

第一節 研究背景

根據教育部於民國100年所修正訂頒的「普通高級中學課程綱要」，高中資訊科技概論課程欲達成的目標之一為培養學生邏輯思維及運用電腦解決問題之能力，因此課程中「電腦與問題解決」單元之授課重點應為引導學生進行問題分析、設計解題步驟、及說明解題策略，並透過基礎程式設計與學生日常生活、學習等相關之實例進行連結，以激發學生學習電腦解題的興趣。

程式設計的過程中，學生除須掌握所使用的程式語言之語法架構及程式設計技巧外，同時需要具備一般性的問題解決能力，甚至還可能需應用其他領域的知識（Brooks, 1983）；因此對初學者而言，學習程式設計並非是一件簡單的事。許多研究均發現學生在進行程式設計的過程中經常遭遇到困難，包括對於程式語言的語法結構及程式概念具有迷思概念、解題方法無法順利轉化為實體程式、無法構思足以解決問題的演算法、缺乏除錯能力等（Samurçay, 1985; Bonar & Soloway, 1983; Spohrer & Soloway, 1986a; Perkins & Martin, 1986; Ebrahimi, 1994; Lahtinen, Ala-Mutka, & Järvinen, 2005）。這些困難發生的原因不僅和學生對相關知識的缺漏有關，也受到學生程式設計風格的影響。Bishop-Clark（1992）以及Lane與VanLehn（2003）皆發現程式設計初學者在面對問題時，通常選擇直接進入編寫程式的階段，並於編寫程式過程中才同時思考如何解決問題，導致初學者因為無法同時處理程式語法及問題解決兩種不同層面的知識，而造成錯誤的發生。

另一方面，程式語言本身設計上的差異也在學生學習程式設計時扮演了非常重要的角色，除了直接反映於學生學習語法及語意的難易度之外，更間接影響學生將解題方法化為實際程式碼時的表現。Ahmanzadeh、Elliman與Higgins（2005）發現學生在使用語法結構較為簡單的程式語言編寫程式時除了能降低語法錯誤

量以外，也因為不需要花過多的心力在處理語法錯誤上，更能夠專心於處理程式目標及流程，使得邏輯錯誤量亦同時降低。因此，教師可考慮使用操作介面或環境較為簡單，或能動態顯示演算執行過程（algorithm animation）的程式語言、劇本語言（script language）或模擬軟體作為學生學習程式設計時的工具。除此之外，也可以考慮使用可程式化機器人系統、物件導向式語言、函數式語言（functional languages）、網頁程式設計語言或交談式繪圖語言等多元化且生動活潑的電腦解題工具進行教學。Butler與Morgan（2007）認為若是使用這類能清楚地提供資訊回饋的教學方式，可以有效地幫助學生學習程式設計。

因此，許多具備上述特色的程式語言被運用於初學者程式設計教學中。其中 Alice 為美國卡內基美隆大學所研發之 3D 程式設計環境，使用者可以透過建構 3D 動畫或遊戲來學習入門的物件導向程式設計概念。同時 Alice 提供使用者拖放式（drag-and-drop）操作介面，可使學生完全避免語法錯誤（syntax error），編寫出絕對可以編譯執行的程式碼。這兩個特色使得 Alice 符合前段所述「操作介面或環境較為簡單」，且「能動態顯示演算執行過程」的程式語言之一，被認為適合用於大學前教導學生程式設計知識的入門語言。

第二節 研究目的

本研究以台北市某高級中學一年級學生為研究對象，並以 Alice 2.2 版為教學工具，採質性研究之一對一診斷式訪談研究方法，觀察並協助學生實作程式專案。研究者試圖從學生在實作 Alice 程式專案時所發生的錯誤中，歸納及呈現程式設計初學者在各基礎程式設計概念中所可能遭遇的困難狀況及類型供教學者作為設計教學方法的參考。本研究之具體目標如下：

一、瞭解學生在實作 Alice 程式專案時所遭遇的困難。

二、分析學生在各項程式設計概念中常見的錯誤類型。

三、提出 Alice 教學上的建議。

第二章　文獻探討

根據第一章所提出的研究問題，本章蒐集程式設計教學的相關文獻並做整理和探討，包含程式設計教學、Alice 程式設計軟體及診斷式訪談。

第一節 程式設計教學

一、程式設計學習

Brooks（1983）認為程式設計是一個將無形的問題轉化為有形的程式碼的過程，除了需要組合程式語言本身的語法及結構，還有可能會使用其他領域的知識方可完成。因此初學者在學習程式時，最重要的是需要學會理解整個程式，包含程式的功能、程式碼內容、使用目的以及難易程度的差別。這個能力不僅能實際用於撰寫程式的過程中，亦有助於程式設計者進行除錯、測試以及閱讀其他人的程式碼。

Linn 與 Dalbey（1985）提出程式設計理想的「認知鏈結」，認為程式設計的學習成果可以透過下列三方面進行評量：(1)程式語言特性；(2)程式設計技能；(3)一般性的問題解決能力。一般而言，在入門的課程中學生所學習的程式設計知識多屬於基本程式語言特性的理解及設計技能的範疇，鮮少觸及問題解決技巧的領域。

在程式開發過程方面，Shneiderman（1980）將程式設計解題的過程對應至 Polya（1966）所提出的問題解決四階段：(1)理解問題；(2)設計解題策略；(3)執行策略進行解題；(4)回顧解答，並修改為理解問題需求、規劃與設計、編寫程式和除錯（引自 Bishop-Clark, 1992）：

1. 理解問題需求：理解程式所要完成的工作、呈現的功能。
2. 規劃與設計：針對問題需求找出解決方法。

3. 編寫程式：將所找出的解決方法透過程式語言實作。

4. 除錯：檢視程式執行成果，並進行修正。

Soloway（1986）認為學習程式設計的重點除了精熟目標程式語言的知識，更需要學習問題解決的能力。教師除了程式語言的語法及語意之外，還要進一步的教導要教導程式概念的作用原理、說明、使用目標、計畫、程式論述的規則和計畫組成的方法等問題解決能力。如果能有效的協助學生將他們的想法轉化為實際的程式，那程式設計就不單單是一個專業技能，而能成為一個解決問題的工具。

綜上所述，學生在學習程式設計時，除了必須針對程式語言學習語法結構及規則以外，更重要的是要夠從中培養出分析問題、構思解題策略、透過程式設計技巧將解題策略實作、並根據測試結果進行調整及除錯的能力。

二、程式設計的學習困難

程式設計一直被認為是難以學習精熟的學科。Jenkins（2002）認為學生在學習程式設計的過程中容易產生困難，是由於程式設計這門學科具備以下特性：

1. 需要多種技能：程式設計的範圍包含語法、語意、程式結構、寫作風格等不同的面向。課本跟課堂通常只有教導語法、語意的部分，其他程式設計技能必須透過程式設計的過程來學習
2. 具備多個程序：學生必須先自己的想法轉換為有效、可行的演算法，再將演算法轉化為程式碼，在這些不同的領域轉換的過程中，發生任何微小的錯誤都有可能導致學生無法順利完成程式。
3. 所使用之程式語言：目前大多數課堂都是選用業界常見的程式語言，而非適合教學的程式語言。當學生必須花費大量心力於理解和處理語法時，往往難以分心關注問題解決的技巧。
4. 教學新穎性：對很多學生而言，程式設計是過去從未接觸過的學科領域。學習的過程必須兼顧指令語法的記憶及解題方法的知識理解兩種不同面向的能

力，和其他的科目不盡相同。

5. 難以引發興趣：程式語法課程通常十分枯燥，且基礎程式概念程式範例亦較為單調，難以引起學生興趣。
6. 外在觀感不佳：「程式設計」普遍被認為是困難的技術，而「程式設計師」也常給人書呆子的印象。學生可能會因為這些認知而產生排斥感。
7. 課程步調：由於授課時間受制於學制，使得課程進度必須不斷推進，造成學生若是基本概念有所缺漏，將難以跟上後續進度。

由於在程式設計的過程中不僅需要針對目標問題發展出對應的問題解決方法，還必須使用「程式語言」這個工具將其實作出來，因此 Spohrer 與 Soloway（1986b）將學生常犯的錯誤統整為程式語言結構以及計畫組織兩大類型。其中程式語言結構相關的程式錯誤是由於對程式語法的誤解所產生，除了基本的語法錯誤以外，初學者在編寫程式的過程中，往往會使用較為熟悉的自然語言去解釋程式碼（Bonar & Soloway, 1983），進而導致如邏輯判斷中的「且」及「或」混淆、重複結構的終止條件設定錯誤等邏輯錯誤的發生；Putnam、Sleeman、Baxter 與 Kuspa（1989）發現學生在面對未完全理解的程式語法時，亦會使用其他領域的知識進行程式碼的解讀，進而導致迷思概念及錯誤的發生，例如將 BASIC 程式語言中等號（代入）以一般數學的使用方式解釋為「等於」。SamurÇay（1985）更發現即使核心概念相同，學生在將其他領域的知識轉化為可執行的程式碼的過程中仍有困難。以重複結構 loop 為例，學生在數學中所學到的代數、等號以及方程式的知識並不足以協助他們在實作程式題目時建構變數、代入及迴圈概念。

Perkins 與 Martin（1986）認為學生在犯下程式錯誤的原因，並不完全是因為學生缺乏所需的知識，而可能是有部分缺漏或誤解。他們將其稱之為脆弱知識，並分為以下四種：

1　局部知識：學生僅擁有該知識的局部片段。

2　惰性知識：學生擁有該知識，但無法在需要時取用。

3　錯置知識：學生擁有該知識，但使用在錯誤的地方。

4　混用知識：學生將不同的知識混雜在一起。

另一方面，計畫組織型的錯誤多發生於學生將不同的程式片段整合為完整程式的過程中，因為缺乏整體性的規劃使得程式有所缺漏無法正常運行，如未進行邊界錯誤的阻擋。Spohrer、Soloway 與 Pope（1985）將計畫組織目標程式時所需進行規劃設計的計畫元件（plan components）分為：(1)輸入；(2)輸出；(3)初始值；(4)資料更新；(5)防護措施；(6)語法；(7)整體計畫。並將計畫組織時會發生的錯誤分為以下四種類型：

1　缺乏（Missing）：未進行計畫元件的實作。

2　異常（Malformed）：計畫元件未實作完全或有錯誤。

3　偽造（Spurious）：實作了不屬於計畫中的元件。

4　錯位（Misplaced）：計畫元件有實作，但位置錯誤。

Spohrer、Soloway 與 Pope（1985）；Kopec、Yarmish 與 Cheung（2007）及 Kopec 與 Yarmish（2007）實際統計分析學生在 Pascal 及 C 程式語言中的程式錯誤發現，學生最常犯的程式組織錯誤多屬於異常及缺乏兩類。Kopec 與 Yarmish（2007）更進一步提出若是能清楚呈現程式題目將有助於學生理解題意及規劃解題計畫。

Ebrahimi（1994）發現程式語言結構以及計畫組織這兩種類型的錯誤發生量具有高度相關，意即若學生常犯下語言結構類的程式錯誤，他同時也較容易犯計畫組織型的錯誤。

另外有部分的迷思概念是源自於學生對電腦的能力具有不切實際的幻想，Pea（1986）將這些與特定程式語法無關的錯誤觀念分為以下三種：

1. 排比型（Parallelism bug）：認為程式可以同時執行多段程式碼。
2. 意向型（Intentionality bug）：認為電腦會在執行過程中具有自動完成的能力。

3. 心之所向型（Egocentrism bug）：認為電腦會自動補足程式設計者遺漏的部分，或可以正確的將寫錯的序數修正。

Lahtinen、Ala-Mutka 與 Järvinen（2005）針對 559 名學生和 34 名教師所進行的問卷調查統計結果顯示學生在進行程式設計時最容易遭遇到的困難有(1)設計程式以處理特定任務；(2)將程式依所需的功能化分為數個程序；(3)從自己的程式中找出錯誤。在程式概念方面，學生較難理解的概念包括遞迴（recursion）、指標與參照（pointers and reference）、抽象資料型態（abstract data type）及錯誤處理（error handling）。Ragonis 與 Ben-Ari（2005）及 Chen、Cheng 與 Lin（2012）統整學生使用 Java 學習物件導向程式設計時常見的迷思及迷失概念，發現學生之所以無法順利解答程式題目，是由於(1)無法記住程式語言語法規定，如無法正確定義靜態資料成員及常數、建立物件、迴圈、方法的呼叫；(2)缺乏對程式碼實際運作情形的理解，特別是 for-each 及 for 迴圈的控制及運作流程，以及方法的參數回傳方式；(3)不了解物件導向概念，如靜態資料成員、方法重載、建構式及常數與變數的差異等，進而導致未進行宣告、呼叫或誤用；(4)對程式概念未完全理解，如將建構式與一般方法混淆、資料定義位置錯誤等。

除此之外，由於程式設計教學的過程中教師及教科書皆提供學生「絕對正確」的程式碼範例，因此造成許多學生在遭遇程式錯誤時缺乏解決錯誤的能力。Ahmanzadeh、Elliman 與 Higgins（2005）指出，一個好的程式設計者，未必同時是一個好的除錯者，學生在除錯時需要掌握的能力包含(1)對程式的執行目標；(2)目前程式的實際狀況；(3)錯誤內容；(4)針對錯誤的處理方法。其中最為重要的是對程式現況的理解能力，學生不僅需要能夠讀懂程式碼，並且還要能夠透過輸入、輸出等資訊判斷程式運作情形，藉此找出錯誤進行處理。Vainio 與 Sajaniemi（2007）發現學生在追蹤（tracing）程式碼中的困難包含難以追蹤變數值的變化、程式結構及函式的混淆、無法有效的使用其他程式表示方式及無法提升至抽象思考層面，究其原因除了學生缺乏對程式語言語法結構及程式概念上的專業知識，

還牽扯到程式呈現的方式及相關輔助工具（如流程圖）的使用。教師應該要協助學生學習及掌握除錯技巧，以增加學生的自信及學習動機。

綜上所述，學生在學習程式設計時，從各語言專屬的語法結構，到共通的問題解決層面都有可能遭遇學習困難。教學者應透過學生所遭遇到的困難，分析理解學生在知識上的誤解或缺失，並規劃合適的教學方法，以期培養學生在程式設計能力的提升。

第二節 Alice 程式設計

一、Alice 程式語言與特色

Alice 為美國卡內基美隆大學所研發之 3D 程式設計環境，它讓使用者可以輕鬆的創造和操控一個 3D 的虛擬世界，並用它來述說故事、製作互動式遊戲或是動畫影片（http://www.alice.org/）。Brown（2009）認為，作為一個入門程式語言，Alice 具有以下優勢：

1. 情境式設計環境：

   在 Alice 中，每一個程式即為一個 3D 的虛擬世界，設計者透過編寫程式來操控、改變這個世界中的事物。Alice 提供了豐富的物件及控制物件動作的方法及函式，透過這些，程式設計者可以輕鬆的製作出逼真的 3D 世界。

2. 3D 動畫呈現：

   當 Alice 程式執行時，虛擬世界便會依照程式碼產生動作和變化，並以動畫的形式呈現執行結果。程式設計者可以立即清楚地從動畫觀察到程式的執行狀況，同時可對應至編寫的程式碼。Ward（2009）認為透過動畫的輔助，可以將部分抽象的程式概念轉化為實際可以觀察到的現象，有助於學生學習程式設計概念、進行知識保留及享受程式設計的過程。

3. 拖放式程式設計環境：

   Alice 使用視覺化程式設計環境（visual programming environment），將程式的

各種指令、語法及結構，作成模組化的標籤，程式設計者僅需透過「拖曳－放置標籤」即可編寫程式。透過這種程式編寫方式，學習者除了可以避免語法錯誤（syntax error），更可以將心力集中於了解語意和解決問題的方法（Ko, 2004; Kelleher, Cosgrove, Culyba, Forlines, Pratt, & Pausch, 2002）。

這些優點使得學生在寫程式的過程中，發展出良好的程式設計技巧，如先行規劃演算法在上機實作、願意嘗試及挑戰錯誤、編寫專用函式等（Cooper, Dann, & Pausch, 2003）。

二、Alice 程式設計教學相關研究

很多初學者之所以無法順利解題，是由於他們在寫程式的過程中無法將所學到的程式設計概念對應到所寫的程式碼（Dann, Cooper, & Pausch, 2000）。由於 Alice 讓學生可以看到程式執行的過程，學生得以將程式碼對應至實際執行的結果，因此可以使學生更容易將發生的問題描述清楚、拆解題目及思考方法處理拆解後的問題，進而發展出演算法式的思考模式，增進學生問題處理的能力（Cooper, Dann, & Pausch, 2000a）。而「編寫程式即為製作動畫」的這個特性，不僅降低學生對於程式的排斥感，也增加學生學習的動機。Adams（2007）更發現，Alice 的 3D 動畫呈現使得學生在遭遇程式邏輯錯誤時，會覺得動畫中錯誤的演出看起來很有趣或好笑，進而降低學生的挫折感。

在動畫製作上的高完成度，使得 Alice 可以將許多抽象的程式概念透過動畫，以具體的形式呈現，因此相當適合作為教導程式設計概念的入門程式語言。Zaccone、Cooper 與 Dann（2003）於大學一年級的資訊科學選修課程中，透過 Alice 來教導學生物件導向的程式設計概念，課程內容包含物件、物件方法、參數、繼承及判斷條件等概念。超過 2/3 的學生可以順利的理解課堂中所學習到的程式概念，並實作出 Alice 程式。林恬忻（2007）發現在克服英文介面的問題後，國中的學生也可以順利的使用 Alice 編寫程式及學習程式概念。除此之外， Alice

獨有的視覺化物件，如視覺化陣列、視覺化串列等，使得 Alice 比起其他類似的程式語言（如 Scratch、Greenfoot）更能清楚呈現進階的演算法及程式概念（Ward, 2009）。

與傳統的文字式程式語言（如 C++、Java）相比，學生透過 Alice 程式語言來學習程式設計概念除了更容易理解程式概念，而且在寫程式的過程中，對教師協助的依賴性相對較低，學生可以用更快的速度完成程式作業，將多餘的時間和心力花費在解決問題及除錯上。在課堂上的良好表現也使得使用 Alice 的資訊科學課程的通過率、學習成績都較使用傳統文字語言的課堂為優（Sykes, 2007; Brown, 2008; Mullins, Whitfield & Conlon, 2009; Wang et al, 2009）。

但是 Alice 並非被認為是沒有缺點的，Alice 程式本身有著系統不穩定、資源佔用過多等缺點。而其所使用的拖放式程式設計環境，雖然能有效降低程式編寫的難度，但也有著拖放步驟過於繁鎖不夠直觀，以及不易轉換至目前業界慣用的文字介面程式語言的問題存在。但若以課堂使用的來看，這些問題大多可以透過課程設計和教師授課方式加以避免（Powers, Ecott, & Hirshfield, 2007; Brown, 2008; Ward, 2009; Wang et al, 2009）。

第三章　研究方法

第一節 研究方法

本研究旨在探討 Alice 程式設計初學者實際進行程式設計時，對於不同的程式設計概念所面臨的困難與問題類型，所使用的研究方法為一對一的診斷性訪談（clinical interview，亦譯為臨床晤談）。根據謝如山（2004）所譯的書中所述，診斷性訪談是由皮亞傑（Jean Piaget）所發展出來的研究方法。其目的為透過訪談者與受訪者的互動過程中，訪談者的引導提問及觀察，能夠深入瞭解受試者的想法、認知過程、心智如何作用及問題解決概念等。

診斷性訪談可視為一種半結構化的訪談方式，實施的步驟為：訪談者事先根據欲瞭解的主題訂定一些相關的題目，詢問受試者這些題目並觀察其回應，然後根據這些回應提出更進一步的問題，以釐清受試者的認知內容及思考方式。診斷性訪談的主要特色如下（引自謝如山，2004，p. 34）：

- 活動一開始的標準化
- 將活動所圍繞的主題以具體物表示
- 「你是如何做的？」或是「為何這麼做」的問題
- 立刻解讀受訪者的回應
- 現場擬定假設並加以測試。
- 即興發問的自由

診斷性訪談被應用於許多學習領域上，而本研究欲透過診斷性訪談以深入瞭解學生在實作 Alice 程式專案實的思考方式，期望找出學生在使用 Alice 學習程式概念時容易產生的錯誤類型及其成因。

第二節 實施程序

本研究共進行三天，分為六個時段，每個時段為 2.5 個小時，單一時段至多有 4 名參與研究之學生同時進行解題，每位學生均配置一位訪談者坐於其身旁觀察其解題過程。單一時段之實施程序如圖 3.1。

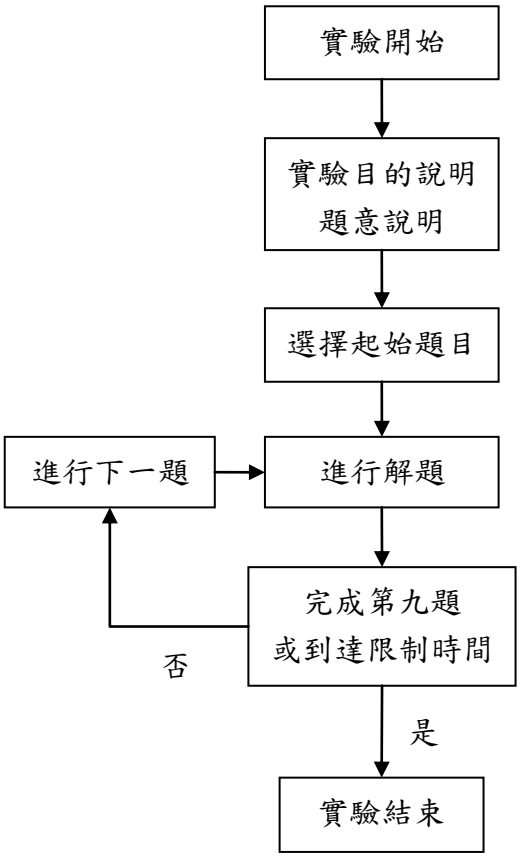


圖 3.1 實驗實施程序

在學生正式進行解題之前，研究者先對其說明實驗目的、設置訪談者的用意，並搭配題目動畫影片，說明各題題意。學生可依照自己的能力選擇九題中任何一題開始解題，並依序完成該題及後續的題目，直到完成最後一題，或是限定時間 150 分鐘用完為止。學生於實作任何一題時，可反覆觀察研究者錄製的影片，以了解題目要求。

在每位參與學生身邊均配置一位訪談者，其功能除了實際觀察參與者使用

Alice 進行解題的過程，記錄其於解題過程中所遭遇的困難與迷思概念之外，同時擔任該生的引導者，於該生解題陷入膠著時視情況介入，適時提供相關的引導提問或操作指導，助其順利解決問題。以下分別說明訪談者所提供之協助形式及特性。

當學生初次遭遇問題時，訪談者先使用引導式提問。此種提問屬於解題策略層面的一般性問題，適用於所有題目，例如：「你打算如何解決這個問題？」、「你要不要再看一次影片以確認這一題應有的執行結果？」、「可以告訴我這一段程式碼是在做什麼嗎？」、「你知道ＸＸ指令的功能是什麼嗎？」皆屬引導式提問的範疇。

若引導式提問無法協助學生順利解題，訪談者會根據學生所遭遇的困難提供進一步的提示，試圖將學生導向正確的解題方向。這種類型的提示我們稱之為「間接提示」，例如「你知道有哪個控制結構是用來讓動作同時執行的嗎？」、「這個重複結構的判斷條件該怎麼寫才能讓迴圈重複執行 3 次？」、「使用者輸入的數值該怎麼做才能將它保留起來？」、「resize 這個方法的作用是改變這個物件的大小，後面的值是它的縮放倍率，這裡的倍率該填多少呢？」等等。

若訪談者給予學生若干引導提示後仍無法協助其正確解題，訪談者將進行「直接指導」，意即將所需使用之程式指令或修改方法直接告知參與者，或進行示範操作以幫助學生完成解題，例如「將控制結構 loop 拖曳至程式中，並設定後面的重複次數為 3」、「將設定變數值的那一行程式碼移到第一行」等等。訪談者於提供直接指導時，會針對該指令或作法進行概念解說，幫助學生理解相關概念，以利學生順利進行後續的解題。

在一般情形下，訪談者將會依照「引導式提問」、「間接提示」、「直接指導」的順序在學生遭遇困難時進行協助，但訪談者亦得視學生的個別狀況，直接使用「間接提示」、甚至「直接指導」以協助學生克服解題困難。

表 3.1 診斷性訪談測驗範例提示（第 1 題）

觀察者：　　被觀察學生：　　開始時間：　　結束時間：

Q1、在場景中建立一隻兔子，在執行階段讓兔子做以下動作：

1　兔子說「Jump 1.」，持續 2 秒。

2　讓兔子作以下動作（2.1 與 2.2 同時發生）

2.1　兔子 2 秒內由白色變成其它顏色（例 brown）

2.2　兔子 2 秒內完成跳躍的動作（2.2.1 及 2.2.2 兩個動作依序發生）

2.2.1　兔子 1 秒內往上跳 1meter

2.2.2　兔子 1 秒內往下跳 1meter

3　兔子說「Jump 2.」，持續 2 秒。

4　讓兔子作以下動作（4.1 與 4.2 同時發生）

4.1　兔子 2 秒內變成另外一種顏色

4.2　兔子 2 秒內完成跳躍的動作（4.2.1 及 4.2.2 兩個動作依序發生）

4.2.1　兔子 1 秒內往上跳 1meter

4.2.2　兔子 1 秒內往下跳 1meter

- **本題建議使用的提示內容**

1. 引導式提問：請說出這一題要求你做什麼?
2. 引導式提問：請觀察那些動作先後發生？那些動作同時發生？
3. 間接提示：什麼樣的控制結構可以讓兩個以上的動作同時發生？
4. 間接提示：什麼樣的控制結構可以讓兩個以上的動作依序發生？
5. 間接提示：如何選取適當的指令讓人物可以跳躍？
6. 間接提示：如何選取適當的指令讓人物可以說話？
7. 間接提示：如何選取適當的指令讓人物可以改變顏色？
8. 直接指導：若經過以上提示後，學生仍無法解決問題，就直接告訴學生該用什麼指令

- **本題實際使用的提示內容**

序號	型態	提示內容
1	直接使用以上第 ____ 個提示 □引導式提問 □間接提示 □直接指導	經過提示後，是否有解決問題□是 □否
2	直接使用以上第 ____ 個提示 □引導式提問 □間接提示 □直接指導	經過提示後，是否有解決問題□是 □否

第三節 參與者

本研究以臺北市 N 高中一年級的 22 名學生為研究對象，其中包含男生 10 人、女生 12 人，這些學生皆為自願參加。這 22 名學生來自三個不同的班級，他們在一年級的資訊概論課程中皆已學過 Alice 程式設計的基本知識與實作技巧。這三個班級的資訊概論科授課教師以及授課內容都相同，同時因該校學生之基測成績相近，且學校採取常態編班，因此學生的先備知識與能力分布可視為近似。

在訪談者的部分，本次研究中共有四人擔任訪談者，其中兩位為高中、職電腦教師；另一位任教於大專資訊管理系，教授程式設計課程，皆具有豐富的教學經驗；最後一位訪談者即研究者本人。以上四人皆對於程式設計的技巧與實作，以及 Alice 的操作有著相當程度的精熟。另外，研究者於實驗前召開會議向訪談者說明實驗方法、實驗工具、提示類別及範例，以確保訪談者介入學生解題過程以及觀察記錄上的一致性。表 3.1 為實驗前會議中，研究者所提供之第一題實作題目內容及該題範例提示。

第四節 資料收集工具

本研究設計了 9 題實作題目供研究對象進行解題以收集研究資料，同時以軟、硬體記錄研究對象的解題過程，以下分別詳述之。

一、實作題目：

研究者根據學生過去的學習經驗與電腦課程內容，列舉出初學 Alice 程式設計者應學會的程式設計概念，包含物件、內建方法（built-in methods）、運算式、內建函式、變數、選擇結構、重複結構、陣列、自訂方法（user-defined methods）和參數，再針對這些概念設計了 9 個複雜度漸進的 Alice 程式設計題目，並由三位現任高中職電腦科教師進行審閱，再根據其意見進行修正與調整。

每個實作題目均包含複數的程式概念，題號越大的題目所包含的程式設計概念越多，並且涵蓋之前的題目所使用的概念。如此安排的目的為若是學生依照題

號順序進行實作的話，每一題基本上都只需要增加一個程式概念，方便研究者觀察學生於個別概念中所犯的錯誤；若是學生直接自較複雜的題目開始實作，亦可觀察到他在之前題目中所包含的程式概念中犯錯的情形。

表 3.2 題目與程式觀念對應表

程式構句（Programming Constructs）	第一題	第二題	第三題	第四題	第五題	第六題	第七題	第八題	第九題
物件	V	V	V	V	V	V	V	V	V
內建方法	V	V	V	V	V	V	V	V	V
時序控制結構	V	V	V	V	V	V	V	V	V
運算式及內建函式		V	V	V	V	V	V	V	V
變數			V	V	V	V	V	V	V
選擇結構				V	V	V	V	V	V
重複結構（固定次數）					V	V	V	V	V
重複結構（不定次數）						V	V	V	V
陣列							V	V	V
自訂方法								V	V
參數									V

基於 Alice 適於製作動畫的特性，上述題目之輸出結果均為動畫，因此我們並未提供文字敘述的題目，而是將各題的執行結果錄製成影片，由參與研究的學生於解題前播放，藉由觀察影片了解各題的要求。參與者必須寫出完整的 Alice 程式，以輸出各題所要求的結果。以下分別描述各題所需呈現之執行結果與該題所欲測試之程式設計核心概念。

第一題：物件、內建方法及時序控制結構

本題（圖 3.2）要求程式設計者在場景中建立一個任意物件，於程式執行時改變它的顏色，並使其進行跳躍的動作。換言之，程式設計者必須能宣告物件、修改物件屬性（properties）以調整物件狀態，並使用方法（methods）使物件進行

動作。此外，本題亦須使用控制結構 Do in order 和 Do together 來控制物件動作發生的時序。

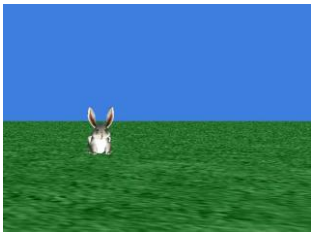
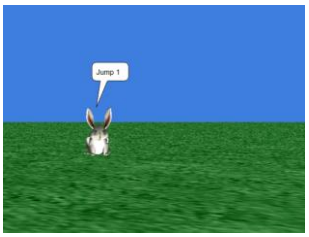
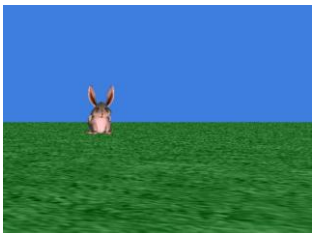
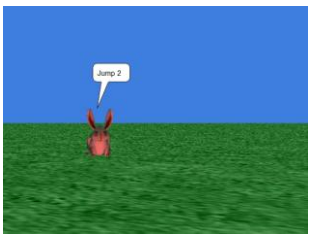
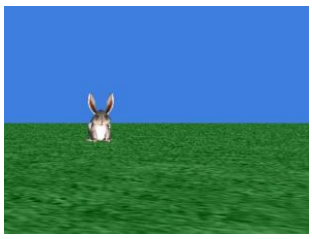
<table>
<tr><td colspan="2">在場景中建立一個任意物件。</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依序執行以下動作</td></tr>
<tr><td>1　物件說「Jump 1.」。</td><td>2　讓物件同時執行動作 2.1 與 2.2。
2.1　物件 2 秒內變成其它顏色。
2.2　物件 2 秒內完成跳躍的動作。</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>3　物件說「Jump 2.」。</td><td>4　讓物件同時執行動作 4.1 與 4.2。
4.1　物件 2 秒內再次變成其它顏色。
4.2　物件 2 秒內完成跳躍的動作。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.2 第一題程式執行示意圖

第二題至第六題為一連續的題組，以內建方法 resize 為主題，每題依序加入不同的程式設計概念。

第二題：內建函式

本題（圖 3.3）要求程式設計者在場景中建立任兩個身高互異的物件，於程式執行時，改變矮物件的大小，使其與高物件同高。程式設計者必須使用物件的

函式（functions）進行數學運算，並將其設定為物件內建方法所接收的參數。此題依然需要使用第一題中運用內建方法與控制結構的技巧。

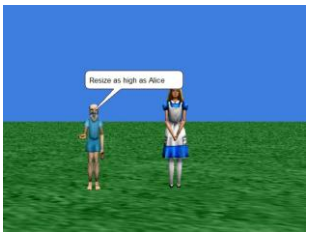
<table>
<tr><td colspan="2">在場景中建立兩個身高相異的物件（以下以身高較高的物件為物件 1，較矮的為物件 2）</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1　物件 2 同時執行動作 1.1 與 1.2：
1.1　物件 2 說「Resize as high as (物件 1 之名稱).」。
1.2　透過物件內建方法 Resize 來進行物件大小的改變，使得物件 2 放大至與物件 1 相同身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.3 第二題程式執行示意圖

第三題：變數

本題（圖 3.4）要求程式設計者在場景中建立任兩個身高互異的物件，於程式執行時，改變矮物件的大小，使其與高物件同高後，再恢復為原本身高。延續前一題所使用的方法，本題新增之程式設計概念為變數（variable）的宣告與使用。由於矮物件增高至與高物件同高後，必須再恢復為原身高，程式設計者必須使用一個變數來儲存矮物件原本的身高，才能順利將矮物件縮小為原來的身高。

第四題：選擇結構

本題（圖 3.5）要求程式設計者在場景中建立任兩個身高互異的物件，於程式執行時，先要求使用者輸入一個數字，並根據使用者輸入的數字進行動作。若使用者輸入的數字為正數，則改變矮物件的大小，使其與高物件同高後，再恢復為原本身高；若使用者輸入的數字為負數，則改變高物件的大小，使其與矮物件同高後，再恢復為原本身高。本題加入選擇結構（if/else）的使用，程式設計者必須能理解判斷條件的設定方法，並能依條件判斷結果執行相對應的動作，方能完成此題。

第五題：重複結構（固定次數）

本題（圖 3.6）要求程式設計者在場景中建立任兩個身高互異的物件，於程式執行時，先要求使用者輸入一個數字，並根據使用者輸入的數字進行動作。若使用者輸入的數字為正數，則重複改變矮物件的大小三次，使其與高物件同高後，再恢復為原本身高；若使用者輸入的數字為負數，則重複改變高物件的大小三次，使其與矮物件同高後，再恢復為原本身高。本題必須使用重複結構（loop）進行實作。若程式設計者並未使用 loop，而是以複製目標動作程式碼三次作為解題方法，訪談者將介入提問，以確保程式設計者使用重複結構進行解題。

第六題：重複結構（不定次數）

本題（圖 3.7）與第五題相似，差別僅在於重複執行的次數乃是由使用者於程式執行時輸入之數字所決定；換言之，本題著重於透過變數來控制重複結構的執行次數。程式設計者必須正確處理變數並運用於重複結構之中，程式方可正確執行。

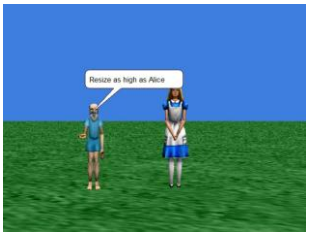
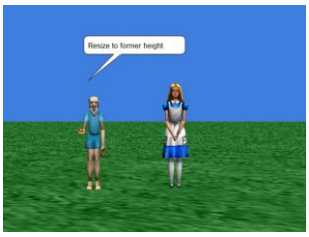
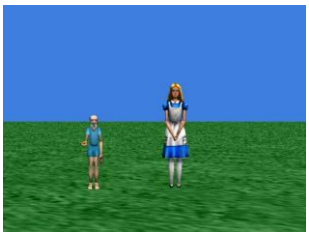
<table>
<tr><td colspan="2">在場景中建立兩個身高相異的物件（以下以身高較高的物件為物件 1，較矮的為物件 2）</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1. 物件 2 同時執行動作 1.1 與 1.2
1.1 物件 2 說「Resize as high as (物件 1 之名稱).」。
1.2 透過物件內建方法 Resize 來進行物件大小的改變，使得物件 2 放大至與物件 1 相同身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">2. 物件 2 同時執行動作 2.1 與 2.2
2.1 物件 2 說「Resize to former height.」。
2.2 透過物件內建方法 Resize 來進行物件大小的改變，使得物件 2 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.4 第三題程式執行示意圖

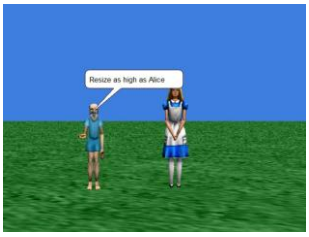
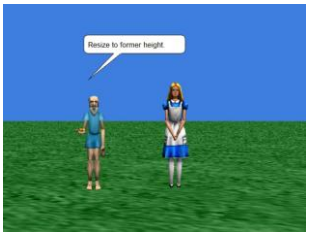
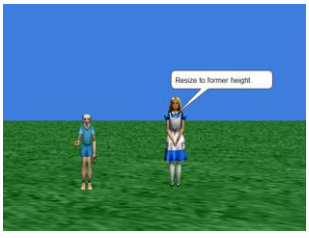
<table>
<tr><td colspan="2">在場景中建立兩個身高相異的物件（以下以身高較高的物件為物件 1，較矮的為物件 2）</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1. 要求使用者輸入一個數字 N。</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>2. 若 N > 0，則依序執行動作 2.1 與 2.2。
2.1 物件 2 同時執行動作 2.1.1 與 2.1.2。
2.1.1 物件 2 說「Resize as high as (物件 1 之名稱).」。
2.1.2 改變物件 2 的大小，使得物件 2 的放大至與物件 1 相同身高。</td><td>2.2 物件 2 同時執行動作 2.2.1 與 2.2.2：
2.2.1 物件 2 說「Resize to former height.」。
2.2.2 改變物件 2 的大小，使得物件 2 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>3. 若 N < 0，則依序執行動作 3.1 與 3.2。
3.1 物件 1 同時執行動作 3.1.1 與 3.1.2。
3.1.1 物件 1 說「Resize as high as (物件 2 之名稱).」。
3.1.2 改變物件 1 的大小，使得物件 1 的縮小至與物件 2 相同身高。</td><td>3.2 物件 1 同時執行動作 3.2.1 與 3.2.2：
3.2.1 物件 1 說「Resize to former height.」。
3.2.2 改變物件 1 的大小，使得物件 1 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.5 第四題程式執行示意圖

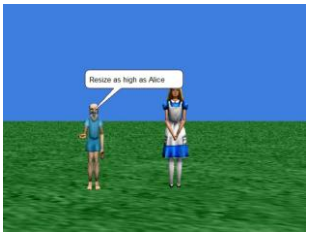
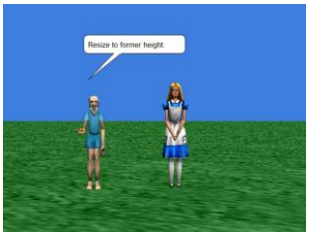
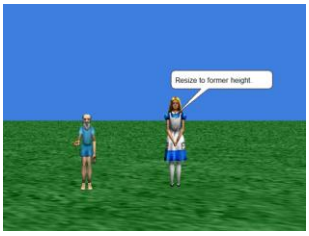
<table>
<tr><td colspan="2">在場景中建立兩個身高相異的物件（以下以身高較高的物件為物件 1，較矮的為物件 2）</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1. 要求使用者輸入一個數字 N。</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>2. 若 N > 0，則重複執行動作 2.1 與 2.2 三次。
2.1 物件 2 同時執行動作 2.1.1 與 2.1.2。
2.1.1 物件 2 說「Resize as high as (物件 1 之名稱).」。
2.1.2 改變物件 2 的大小，使得物件 2 的放大至與物件 1 相同身高。</td><td>2.2 物件 2 同時執行動作 2.2.1 與 2.2.2：
2.2.1 物件 2 說「Resize to former height.」。
2.2.2 改變物件 2 的大小，使得物件 2 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>3. 若 N < 0，則重複執行動作 3.1 與 3.2 三次。
3.1 物件 1 同時執行動作 3.1.1 與 3.1.2。
3.1.1 物件 1 說「Resize as high as (物件 2 之名稱).」。
3.1.2 改變物件 1 的大小，使得物件 1 的縮小至與物件 2 相同身高。</td><td>3.2 物件 1 同時執行動作 3.2.1 與 3.2.2：
3.2.1 物件 1 說「Resize to former height.」。
3.2.2 改變物件 1 的大小，使得物件 1 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.6 第五題程式執行示意圖

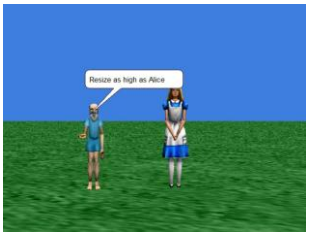
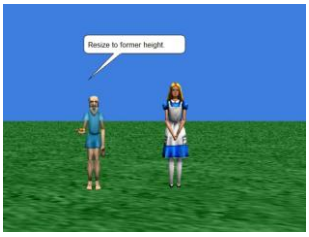
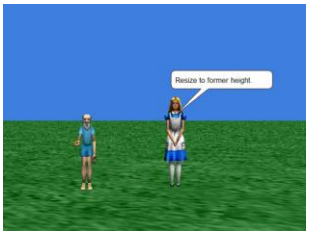
<table>
<tr><td colspan="2">在場景中建立兩個身高相異的物件（以下以身高較高的物件為物件 1，較矮的為物件 2）</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1. 要求使用者輸入一個數字 N。</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>2. 若 N > 0，則重複執行動作 2.1 與 2.2 N 次。
2.1 物件 2 同時執行動作 2.1.1 與 2.1.2。
2.1.1 物件 2 說「Resize as high as (物件 1 之名稱).」。
2.1.2 改變物件 2 的大小，使得物件 2 的放大至與物件 1 相同身高。</td><td>2.2 物件 2 同時執行動作 2.2.1 與 2.2.2：
2.2.1 物件 2 說「Resize to former height.」。
2.2.2 改變物件 2 的大小，使得物件 2 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>3. 若 N < 0，則重複執行動作 3.1 與 3.2 |N|次。
3.1 物件 1 同時執行動作 3.1.1 與 3.1.2。
3.1.1 物件 1 說「Resize as high as (物件 2 之名稱).」。
3.1.2 改變物件 1 的大小，使得物件 1 的縮小至與物件 2 相同身高。</td><td>3.2 物件 1 同時執行動作 3.2.1 與 3.2.2：
3.2.1 物件 1 說「Resize to former height.」。
3.2.2 改變物件 1 的大小，使得物件 1 恢復原來的身高。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.7 第六題程式執行示意圖

<table>
<tr><td colspan="2">在場景中建立視覺化陣列（ArrayVisualization），並在上面放置任意五個身高互異的物件</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">按下執行後，依以下順序執行動作</td></tr>
<tr><td colspan="2">1　要求使用者輸入一個數字 N。</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>2　若 N > 0，則依序使陣列上的每個物件同時執行動作 2.1、2.2。
2.1　透過物件內建方法 Resize 來進行物件大小的改變，將該物件的身高調成和最左邊的物件一樣高。
2.2　讓該物件右轉 1/4 圈。</td><td>3　若 N < 0，則依序使陣列上的每個物件同時執行動作 3.1、3.2。
3.1　透過物件內建方法 Resize 來進行物件大小的改變，將該物件的身高調成和最右邊的物件一樣高。
3.2　讓該物件左轉 1/4 圈。</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

圖 3.8 第七題至第九題程式執行示意圖

第七題至第九題的程式執行結果皆相同，僅使用方法不同。因此在每題開始前，訪談者將直接說明該題所需使用之程式設計概念，確保程式設計者使用符合要求之概念進行解題。

第七題：陣列

本題（圖 3.8）要求程式設計者在場景中建立五個身高互異的物件，並將其放置於視覺化陣列中。程式執行時，先要求使用者輸入一個數字，並根據使用者輸入的數字進行動作。若使用者輸入的數字為正數，則陣列中所有物件依序改變

物件大小與最左邊的人同高，並右轉 90 度；若使用者輸入的數字為負數，則陣列中所有物件依序改變物件大小與最右邊的人同高，並左轉 90 度。本題要求程式設計者正確地宣告視覺化陣列，並理解物件在陣列中位置與陣列索引值的對應關係，方能順利對目標物件進行動作。

第八題：自訂方法

本題的程式執行結果如第七題（圖 3.8），但要求程式設計者必須於程式中宣告兩個自訂方法（user-defined methods），分別包裝輸入數字為正數或負數的兩種對應動作。程式設計者必須理解如何宣告自訂方法，並於程式呼叫使用該方法。

第九題：參數

本題的程式執行結果如第七題（圖 3.8），但程式設計者於此題中須將程式包裝成一個自訂方法，並透過傳遞參數的方式，將使用者輸入的數字傳入自訂方法中，以作為選擇結構判斷的依據。本題的重點在於如何宣告、傳遞與使用參數。

二、資料收集工具：

本次實驗採用螢幕錄製軟體 PowerCam（http://www.powercam.com.tw）將參與者的解題過程以錄影的形式進行記錄，同時運用軟體功能，搭配外接麥克風將參與者與訪談者的問答及對話內容同步記錄。

第五節 資料處理與分析

研究者將實驗過程中所錄製之影片進行逐字稿之謄錄，並根據訪談者與學生在解題過程中的對話內容與行為，分析並歸類學生在各題中所遭遇之解題困難，及訪談者所提供之協助。

第四章　結果與討論

本章共分為三節：第一節探討學生答題過程中所遭遇的困難，以及訪談者引導及協助學生解題的情形；第二節整理訪談過程中發現的學生迷思概念；第三節為綜合討論。本章所引用的逐字稿中，皆以 T 代表訪談者，S 代表學生。

第一節 學生答題情形與困難點分析

本節說明研究者將螢幕錄影與對話錄音，在排除檔案損壞及錄音不清的資料並轉換為逐字稿後，共取得男生 7 人、女生 6 人，合計 13 份資料中統整分析而得之各題答題情形，包含學生答題過程中所遭遇的困難，以及訪談者如何提供協助使學生能順利完成解題。

第一題：物件、內建方法及時序控制結構

本題主要目標為測試學生使用 Alice 程式語言的基本能力，需要學生正確地宣告物件，並透過使用物件內建的方法（method）和屬性（property）來操作該物件，以及透過不同的時序控制結構來控制物件進行動作的順序。本題程式碼範例如圖 4.1，此程式讓一個物件（如程式範例中的 rabbit）依序執行說話、改變顏色及跳躍的動作，其中變色與跳躍動作須同時進行。

在本次實驗中，13 名學生在第一題中所發生的錯誤類型如表 4.1 所示。

1. 新增動作物件

所有學生在新增動作所需的物件時，皆可順利完成「按下 Add Object 按鈕」、「選擇並新增物件」這一連串新增動作物件的操作。

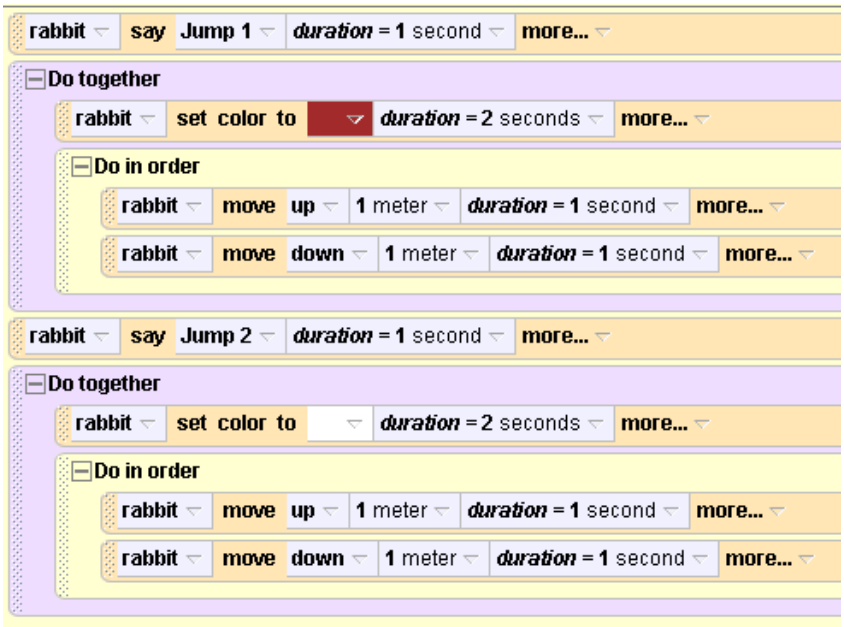


圖 4.1 程式碼範例－物件、內建方法及時序控制結構

表 4.1 學生遭遇困難以及訪談者給予協助情形－物件、內建方法及時序控制結構

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
新增動作物件	所有學生皆能正確作答			
控制物件動作	無法使用 move 實作物件跳躍動作	2	學生反問確認操作是否正確	1
			提示跳躍動作是由移動上及移動下構成	1
	無法自屬性欄中拖曳 color 進行顏色設定	7	提示應搜尋物件顏色	2
			告知物件顏色位於屬性區	5
動作時序控制	未取用同時控制結構	2	提示有動作需同時執行	2
	未取用循序控制結構	8	學生反問確認操作是否正確	1
			提示應使用特定控制結構	1
			告知應使用 do in order 控制結構	6
	無法正確組合同時及循序控制結構	9	詳細告知動作順序	9
	未進行動作時間控制	10	提示學生應設定時間	10

2. 控制物件動作

在使用物件方法（method）及屬性（property）來控制物件進行動作時，所有學生皆能理解並正確使用內建方法 say 來實作說話動作。但在實作物件跳躍動作時，有 2 名學生無法立刻反應需使用兩個移動方法（move）來組合出跳躍動作。其中 1 名學生無法確定自己的動作是否正確，向訪談者反問進行確認：

S01_Q01

S：（完成新增物件 bunny）

S：老師，是 move 嗎？

T：你先看第一個動作是什麼？

S ：（開啟 Q1 題目影片）

S：要說話。

S：（將 bunny say 拖曳至主程式區，並設定說話內容為 jump1）

S：跳是這個吧？（指 bunny move）

T：對啊。

另外 1 名學生則需訪談者更進一步，提示跳躍動作是由 move up 及 move down 兩個動作構成：

S02_Q01

T：他的第一個動作是什麼，你要不要再看影片確認一下？

S：（開啟 Q1 題目影片）

S：（將物件區 rabbit 拖曳至主程式區，並在選單選擇 say 後設定說話內容為 jump1）

S：（將左上方 rabbit 拖曳至主程式區，並在選單選擇 move 後，在 move 設定選單上猶疑）

S：跳的話是這個嗎？

T：跳的動作是不是先上再下？

S：（選取 up 0.5 meter，完成 rabbit move 設定）

T：那個高度你自己設。

S：（複製 rabbit move up 0.5 meter 並修改為 rabbit move down 0.5 meter）

約半數的學生（7 名）不知道改變物件顏色的做法為從物件屬性區（properties）中拖曳顏色屬性至程式中，或是直接將物件拖曳至程式後，於選單中選擇 set color to 選項便可進行改變顏色設定。其中有 2 名學生在訪談者提示需尋找物件顏色之後，可以找到並進行顏色設定，例如：

S11_Q01

S：老師，變色怎麼做？

T：你覺得顏色會是什麼東西？會是動作嗎？

S：他應該是一個 function。

T：顏色的英文是什麼？

S：Color。

T：那就去找 color 。你現在選的是 head，你要選整支兔子。

S：（點選左上方物件區 rabbit）

T：這是動作，methods 都是動作

S：（點選 properties 標籤）

S：（將 color 拖曳進主程式區，並設定為 bunny set color to red）

另外 5 名學生在訪談者明確告知物件顏色位於屬性區後，方能找到並拖曳顏色屬性至程式中進行設定，例如：

S12_Q01

T：兔子有顏色，它會放在哪裡？

S：（右鍵點擊物件區 bunny，瀏覽選單）

S：兔子的顏色應該在…

S：（右鍵點擊主畫面中 bunny2，瀏覽選單後，點擊 bunny2 upper body 選擇兔子的上半身）

T：像身高是兔子的一部分，顏色也是兔子的一部分，所以它會放在哪裡？

S：他會放在哪裡…

T：如果先不要管變色的話，你覺得兔子的身高、位置、顏色，會放在哪裡？

S：儲存檔案裡。

T：就是這邊（指 properties），這邊有兔子的顏色

S：（將 color 拖曳進主程式區，並設定為 bunny2 upper body set color to red）

3. 動作時序控制

在本題中，需要使用同時控制結構 do together 及循序控制結構 do in order 來控制物件動作的發生順序，以下先分別討論學生取用此兩種控制結構所遭遇的困難，再討論學生在組合兩者完成題目時所遭遇的困難。

有 2 名學生在操作過程中，未注意到有特定動作需要同時執行，但在訪談者提示後，便可正確取用同時控制結構 do together，例如：

S12_Q01

T：你覺得這樣跟影片一樣嗎？

S：讓我比對一下。

S：（開啟 Q1 題目影片）

T：你覺得一樣嗎？

S：嗯。

T：你做的是兔子跳到半空中之後，再變顏色。

S：喔，他是邊跳邊變。

S：（將 Do together 拖曳進主程式中，並將兔子跳上及變色的程式碼拖曳至 Do together 中）

在使用同時控制結構 do together 時，有部分學生（8 名）會將跳躍（move up 及 move down）及改變顏色（set color）的程式碼全部放置於同一個 do together 結構中，以致因為向上移動及向下移動兩個動作彼此抵銷，使得物件無法產生跳躍動作，僅改變顏色（如圖 4.2 所示）。也有學生僅將向上移動及改變顏色程式碼放置於同一個 do together 結構中，使得物件顏色改變，卻未與整個跳躍動作同時執行結束（如圖 4.3 所示）。這些學生會發生這樣的錯誤，乃是因為未取用循序控制結構 do in order 來協助控制物件處理順序。

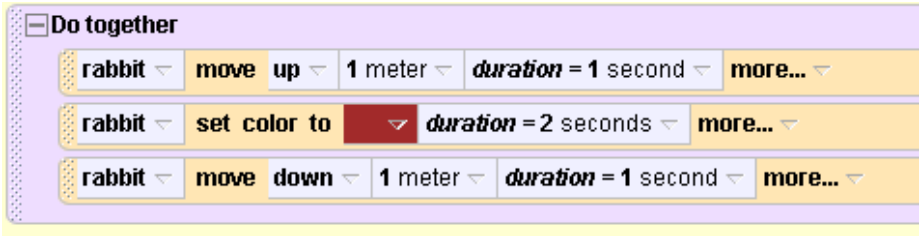


圖 4.2 程式碼範例－時序控制錯誤情形 1

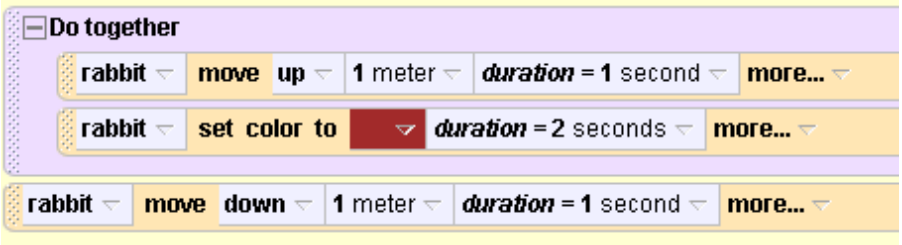


圖 4.3 程式碼範例－時序控制錯誤情形 2

其中有 1 名學生在比對執行結果及題目影片後向訪談者反問確認是否需要取用控制結構 do in order：

S04_Q01

T：它是整隻變紅以後才跳下來嗎？你再看一下影片。

S：（開啟 Q1 影片）

S：是要用這個（指 do in order）嗎？

S：（將 do in order 移入 do together 中）

另有 1 名學生由訪談者提示應取用控制結構中的其中一個後，猜測應選用 do in order，訪談者於學生取用後向學生講解 do in order 的用途：

S11_Q01

S：（按下 PLAY 進行測試，發現 bunny 跳上同時變色完再跳下）

S：（打開 Q1 題目影片）

T：在下面這些控制結構裡面…有沒有哪些你學過的控制結構你現在還沒用到的？

S：（將 do in order 拖曳至主程式區）

T：好，這是什麼？

S：Do in order。

T：Do in order 是什麼?

S：不太記得，他應該跟這個(指 do together)差不多吧？

T：Together 是…

S：一起。

T：In order 就是依序。

S：依序。

其餘 6 名學生則需要訪談者直接告知需使用 do in order 控制結構來處理依序執行的動作，例如：

S09_Q01

T：你想想看，跳上跟跳下要依序發生還是同時發生？

S：依序。

T：依序，對。那下面有哪一個控制結構是依序的？

S：（尋找結構控制列約 30 秒）

T：就是 do in order。

S：那不是「另外做」嗎？

即便在順利取用兩種時序控制結構後，仍有 9 名學生因無法順利將其組合，而無法完成目標動作，這些學生皆由訪談者詳細講解物件動作，並告知兩種時序控制結構的組合方式後方能解題，例如：

S06_Q01

T：你想想看這邊變顏色這個動作應該跟誰同步？

S：跳上跟跳下。

T：就是跳上跟跳下，變顏色要跟這兩個同步。那跳上跟跳下這兩個動作要怎麼樣？

S：要有先後。

T：又要有先後，那你想想看這要怎麼拉？

S：（將 do together 拖曳至主程式區，並 do in order 移入 do together）

S：（按下 PLAY 進行測試，發現兔子仍依序執行跳上、變色、跳下的動作）

T：你這樣這（do in order）裡頭三個（rabbit move up、rabbit set color、rabbit move down）還是循序不是嗎？

S：（將 do in order 移出 do together 後刪除 do together）

S：（將 do together 拖曳至 do in order 中）

T：你是 do together 裡頭應該要有 do in order，do together 裡頭先一個變顏色的動作…

S：（將 do together 移出 do in order）

T：然後再用一個 do in order 分別做…

S：（將 do in order 中的 rabbit set color to red 移入 do together 中）

S：（將 do in order 移入 do together）

完成上述動作後，大多數的學生（10 名）未能注意到變色動作與跳躍動作並非同時執行完畢，需要更進一步修改動作執行時間加以控制，需要訪談者提示學生進行指令執行時間的設定，例如：

S12_Q01

T：還差一點點，你仔細觀察一下他變色的情況。

S：（開啟 Q1 題目影片）

S：漸層。

T：就是他是他跳下來才變色完成。

S：（調整物件 cat 的大小和位置）

S：（按下 PLAY 進行測試）

S：跳下來才變完…時間嗎？

T：嗯。

S：時間…

S：（點擊 cat set color 的 more 選單，將 duration 設成 0.5 sec）

S：（按下 PLAY 進行測試，發現貓還沒跳到最頂就變完顏色）

S：（將 setcat set color 的 duration 調整為 0.25 sec）

S：（按下 PLAY 進行測試，發現貓還沒跳到最頂就變完顏色）

S：（將 setcat set color 的 duration 調整為 2 sec）

S：（按下 PLAY 進行測試，發現貓變顏色過程與整個跳躍動作同步）

第二題：內建函式

本題需要學生在宣告兩個身高互異的物件後，使用物件內建方法 resize 來改變其中較矮的物件大小，使其放大至與較高物件同樣高度。學生必須了解方法 resize 的功能，並且取用物件內建的函式 height（身高）用於倍率計算，進行正確的縮放倍率設定，方可完成本題。本題程式碼範例如圖 4.4，此程式讓一較矮物件（如程式範例中的 plato）透過方法 resize 來改變大小為和較高物件（如程式範例中的 aliceLiddell）同樣高度，同時在放大過程中說話。

本題共有 13 名學生作答，學生答題所遭遇到的困難及訪談者給予的協助統整於表 4.2 中。

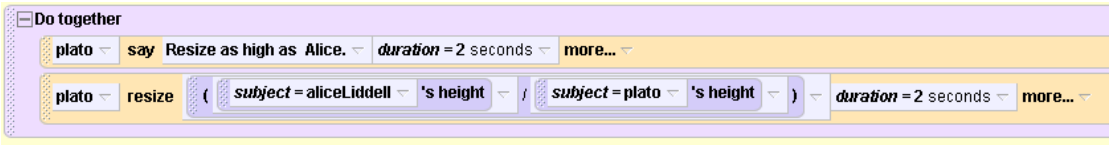


圖 4.4 程式碼範例－內建函式

1. 取用物件方法 resize

參與實驗的學生中，有 6 名學生不知道必需取用方法 resize 來改變物件大小。其中 2 名學生在訪談者詢問改變大小是什麼動作以後，即可自行找到並選用方法 resize，例如：

S07_Q02

T：你現在要做的動作是什麼？

S：讓它變得跟它一樣大。

T：變大是什麼動作？

S：這個(指向方法 resize)。

T：嗯。

另外 4 名學生則需要訪談者直接告知需要取用方法 resize 以進行縮放的動作，例如：

S05_Q02

S：（將 dancer stand up 拖曳至主程式區，並點選 more 檢視選項，但未做任何設定）

S：（按下 PLAY 進行測試，發現無任何改變）

S：這一行(dancer stand up)是什麼啊？

T：就是站起來阿，所以你知道有什麼指令可以用來做改變大小的嗎？

S：（刪除 dancer stand up）

S：（尋找 method 及 function 約 20 秒）

表 4.2 學生遭遇困難以及訪談者給予協助情形－內建函式

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
使用方法 resize 使目標物件縮放	不知道如何取用方法 resize 實作改變大小	6	提問「改變大小」是什麼動作	2
			告知須使用方法 resize	4
	不知道方法 resize 後方數字意義	13	提問是否知道 resize 用法	2
			告知 resize 後方數字為比例或倍數	11
	無法正確計算身高比例	11	提問兩物件比例如何計算	2
			舉例說明比例計算方式	7
			告知正確比例計算方式	2
	無法正確取用物件身高函式用於比例計算式	9	學生反問確認操作是否正確	2
			提問物件身高應如何取得	4
			告知物件身高函式位置	3
動作遺漏或時序錯誤	說話動作缺漏	4	提示觀看影片補齊缺漏動作	4
	未使特定動作同時執行	1	提示觀看影片有動作須同時執行	1

S：不知道。

T：OK，那這邊往上拉，有個 resize。這個可以用來改變大小，它的功能是把東西變成…

S：（將 dancer resize 拖曳至主程式，但未設定縮放倍率）

T：…幾倍大？

2. 理解方法 resize 的使用方式

所有學生在使用方法 resize 進行操作時，皆因不了解方法 resize 後方所接受參數為縮放倍率，而無法順利完成目標動作。其中 2 名學生在訪談者詢問時可正確回答後方數字代表倍數，例如：

S09_Q02

S：（將 joey's resize 拖曳至主程式區，將縮放倍率設為 2）

S：是要用他的高度嗎？

S：（拖曳 joey's height）

T：你想想看，他要跟企鵝一樣高。

S：（放棄拖曳的 joey's height）

S：（點擊左上方物件區 penguin）

S：（拖曳 penguin height）

T：他要跟企鵝一樣高。你想想看 resize 後面這個數字代表的意義是什麼，比如說 2，2 是…

S：他的倍數

T：對，倍數。那所以你 resize 後面的數字應該怎麼算？

其餘 11 名學生則皆由訪談者說明方法 resize 後方數字所代表的意義為比例或倍數，例如：

S07_Q02

S：（按下 PLAY 進行測試，發現 Euripides 放大至比 Alice 大）

S：嗯?

T：好，我們在來看一下 resize 這個動作，你回到 method，把 resize 拉出來看一下。

S：拉出來看一下？

S：（將 Euripides resize 拖曳至主程式區）

T：好，放開，你看一下他（指向放開產生的選單）…twice as big 就是兩倍大，所以說你現在給他的值是多少倍大。如果照你現在程式這樣寫的話，就是告訴它要放大到 Alice 身高的這麼多倍。

S：（取消用來解釋的 Euripides resize）

T：你現在是要把他放大，用 Alice 的身高當作倍數，放大那麼多倍。但是它是要放這麼多

倍嗎？

S：不是。

T：那它應該要放多少倍？

3. 縮放倍率之計算

13 名學生中，在知道方法 resize 後方數字為縮放倍率後，有 11 名仍因無法正確計算兩物件身高比例而使執行結果發生錯誤。其中 2 名學生於訪談者提問比例如何計算後即可拖曳正確比例計算式，例如：

S01_Q02

T：你現在要做什麼動作？

S：跟它(Alice)一樣高。

T：跟它一樣高，那這個比例要怎麼算?

S：（點擊 resize 後方清單，將 Mana resize 設定為 Mana resize aliceLiddell's height / 0.5）

T：要除以誰？

S：它(指 Mana)。

有 7 名學生則在訪談者舉例說明比例關係後，能順利進行計算，例如：

S04_Q02

T：你是不是要計算它比它大幾倍？

S：嗯。

T：那要怎麼算？

S：不知道。

T：好，那我問你 5 是 3 的幾倍大？

S：5/3 倍。

T：是 5 去…

S：除以 3。

T：除以 3 嘛，所以現在是要變成 Alice 的身高是…

S：它(Alice)去除以它(Plato)。

其餘 2 名學生由訪談者直接告知比例計算方式：

S06_Q02

T：你先想想看這個縮放的比例要怎麼算？

S：（在 pj resize howMuch = object only 的 more 清單中尋找）

S：（修改 pj resize howMuch = object only 為 pj resize howMuch = object and parts）

T：其實就是 Alice 的身高除以小孩的身高啊！

4. 物件身高函式的取用

在知道應使用物件身高比例作為縮放倍率後，有 9 名學生無法順利取用物件的身高函式用於比例計算。其中 2 名學生對於自己的選擇有所疑問，向訪談者提問確認，例如：

S12_Q02

S：（瀏覽物件 aliceLiddell 的函式庫）

S：Depth 是什麼？

T：深度，就是你的這個（比畫身體厚度）

S：高度是 height?

T：對啊！高度是 height。

另外 4 名學生在訪談者詢問物件身高如何取得後，即可自物件函式庫中拖曳物件身高用於運算，例如：

S06_Q02

T：你知道怎麼樣取小孩的身高嗎?

S：（將左上方 pj 拖曳至 pj resize(pj.opacity / pj.opacity)的分母部分）

T：你現在要選的是小孩子的身高，所以是物件資訊區。要選小孩還是 Alice？

S：小孩。

S：（點擊左上物件區 pj）

T：對，然後呢？

S：（將 pj height 拖曳至縮放倍率分子）

其餘 3 名學生需訪談者直接告知物件身高函式的取得方式，例如：

S11_Q02

T：你一個、一個看那個函數裡面有哪些東西可以讓你用的。

S：（尋找 Plato 函式庫約 15 秒）

T：身高的英文叫什麼？

S：身高？沒學過。

T：沒學過嗎？那你看一下這個是 size 嘛，size 裡面有什麼？

S：體重(指 width)

T：不是，這個是寬度，那這個（指向 Plato's height）就是身高。

S：這個。

S： (將 Plato's height 拖曳至 Plato resize 縮放倍率中)

5. 動作遺漏或時序錯誤

在完成物件縮放動作後，有 4 名學生未實作物件說話動作即認為程式完成。在訪談者提示觀看影片確認是否完全完成後，4 名學生皆順利補上動作，例如：

S05_Q02

S：（按下 PALY 進行測試，發現 dancer 放大至跟 Alice 同高）

S：這樣嗎？

T：對啊，那你要不要檢查一下影片看你還有沒有漏什麼？

S：（開啟 Q2 題目影片）

S：（將 do together 拖曳至主程式區）

S：（將 dancer say 和 dancer resize 程式碼移入 do together 中）

另 1 名學生有實作物件說話動作，但未將其與縮放動作同步。該生於訪談者提示有動作需要同時執行後，加入控制結構 do together 完成程式，例如：

S01_Q02

T：你再回去看 QUICKTIME 的影片。

S：（開啟 Q2 題目影片）

T：它有沒有邊說邊放大？

S：有。

T：那這兩個動作要同時發生怎麼做？

S：（將 do together 拖曳至主程式區）

S：（將 Mana say 和 Mana resize 程式碼移入 do together 中）

<u>第三題：變數</u>

本題目標是為了測驗學生是否了解變數的宣告方式及使用方法。本題延續第二題的程式，在較矮物件放大至與較高物件同高之後，再將它縮回原本的大小。具體而言，學生需要宣告一個變數以儲存較矮物件改變前的身高，方可於第二次 resize 中順利將其縮為原始身高。本題程式碼範例如圖 4.5，此程式讓一較矮物件（如程式範例中的的 plato）透過方法 resize 增高至與較高物件（如程式範例中的 aliceLiddell）同樣高度，並同時進行說話動作；完成後再復原，並同時說話。

本題所有學生均有作答，其中 1 位學生不需訪談者給予任何提示即可正確作答，其餘 12 名學生答題所遭遇到的困難及訪談者給予的協助統整於表 4.3 中。

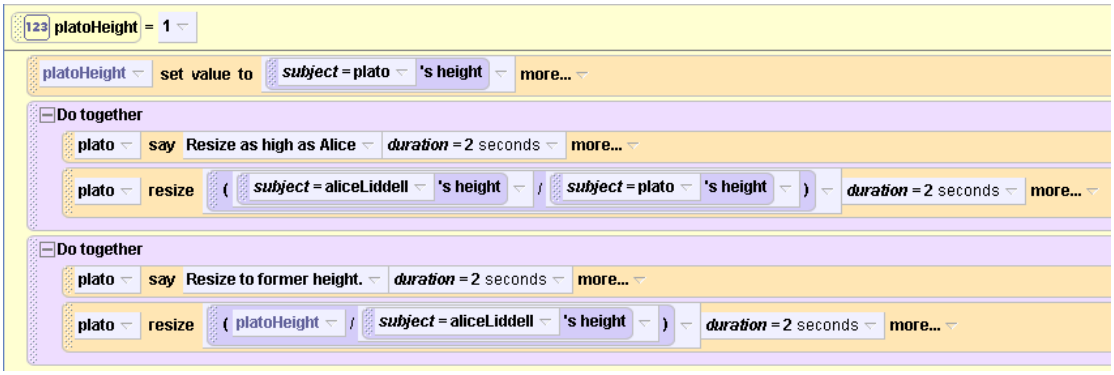


圖 4.5 程式碼範例－變數

表 4.3 學生遭遇困難以及訪談者給予協助情形－變數

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
新增變數	不知道必需使用特定物件（變數）儲存原始身高	12	學生反問確認操作是否正確	1
			提問如何儲存原身高	2
			告知需使用變數來儲存原始身高或比例	9
	不知如何宣告變數	10	學生反問確認操作是否正確	3
			告知變數宣告方式	7
使用變數儲存資料	未設定變數值	10	詢問要將何者指定給變數	2
			告知設定變數值方式及儲存對象	8
	設定變數值程式碼位置錯誤	5	告知應於改變前設定	5
使用變數進行計算	未將變數使用於縮放倍率計算	7	詢問應如何計算	1
			詢問應使用何者進行計算	6

1. 新增變數

在 Alice 中，新增變數的方式為按下程式編輯區右上方的「create new variable」按鈕，如圖 4.6 框線所示，在按下「create new variable」按鈕後會出現如圖 4.7 的變數設定畫面，可在此進行變數名稱、類別等相關設定，設定完成後按下 OK 按鈕即可完成變數宣告。

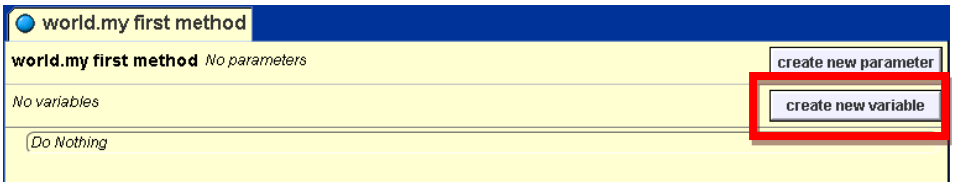


圖 4.6 變數宣告按鈕位置

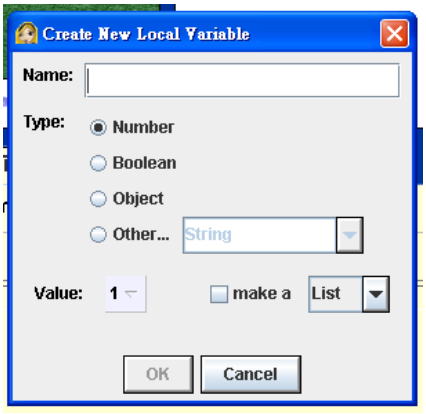


圖 4.7 變數宣告選單

在 12 名遭遇困難的學生中，其中 1 名學生在發現第二次使用 resize 時，無法直接使用身高函式作為比例，順利將目標物件縮回原大小後，向訪談者確認是否需要使用變數儲存，例如：

S09_Q03

S：這邊需要用變數存嗎？

T：這邊的確需要用到變數。那為什麼需要用到變數？

S：因為它這邊會變掉（指第一次 resize）

另有 3 名學生在訪談者解釋未縮回原身高的原因後，被詢問應如何記錄原始身高，可順利回答，例如：

S07_Q03

S：（按下 PLAY 進行測試，發現放大後未縮回）

T：他為什麼沒有變回來？因為你前面的動作是不是已經讓他跟 Alice 一樣高了？

S：喔！所以他們兩個相除是一樣的。

T：對啊！他們兩個是一樣高的，所以他們除出來就是 1 嘛！

S：嗯。

T：那這個時候怎麼辦呢？

T：柏拉圖的身高會變。

S：Alice 呢？

T：Alice 不會變，柏拉圖的身高會變，那我現在要讓它回來的話，我是不是要先想辦法把它原來的身高記起來，或者是它原來跟 Alice 的比例記起來。那你現在想想看，要怎麼樣記綠它原來的身高？

S：是要用…（指 create new variable）是這個吧？

其餘 9 名學生則需訪談者直接告知須宣告一個變數以儲存原始身高或比例，例如：

S02_Q03

T：你要怎麼記得它原來的身高？

S：用它（Mana）除以它（Alice），因為本來是它（Alice）除以它（Mana），那就反過來變成它除以它。

T：把它改變之後它身高就不是原來的值，你要怎麼記得它原來的值？

S：（思考約 10 秒）

S：不知道。

T：你要用一個變數

在知道需要儲存原始身高之後，仍有 10 名學生無法順利新增變數，其中有 3 名學生在操作後向訪談者反問確認動作是否正確，例如：

S01_Q03

T：你要宣告一個變數

S：（點擊 create new variable 按鈕）

S：這個嗎？

T：對。

其餘 7 名學生則由訪談者直接告知變數宣告方式：

S_Q03

T：你剛剛按的那個的下面。

S：這個？（指 create new variable 按鈕）

T：這個叫什麼？

S：新的儲存。

T：Variable 叫做變數。

S：新的變數。

T：對。所以我們要用變數。

S：（點下 create new variable 按鈕）

T：變數是用來儲存我們要存的東西。你要先為你的變數取一個名字。

S：（點擊變數類別）

T：喔，你覺得身高會是什麼呢？

S：數字。

S：（點擊類別 number）

T：數字嘛，OK。那你要取一個名字。

S：（輸入變數名稱 H1，完成變數新增）

2. 使用變數儲存資料

變數宣告後，須將其拖曳至主程式中進行變數值的設定。有 11 名學生無法順利完成指定變數值的操作，其中 2 名在訪談者詢問應將何者儲存之後，即可完成操作，例如：

S01_Q03

T：你想想看，這個 high 是要做什麼的？

S：儲存他的身高。

T：要怎麼樣？

S：儲存他的身高。

T：對，那要怎麼樣把原來的身高儲存給 high？

S：（試圖將 Mana’s height 拖曳至變數 high 的預設值，但無法放入）

S：（點開變數 high 的預設值選單，瀏覽後關閉）

S：（試圖將 mana’s height 直接拖曳至程式中，但未放置便取消）

S：（將變數 high 拖曳至程式中兩個 do together 間，並設定值為 1，再將 mana’s height 拖曳取代 1）

其餘 9 名學生則需訪談者告知設定變數值方式及儲存對象：

S06_Q03

T：你要用它來儲存小孩的身高，所以要指定什麼值給它？

S：小孩的身高。

T：對。

S：（試圖將 pj's height 拖曳至變數 z 的預設值，但無法放入）

T：這邊不能指定，那是不是應該把它拉下來，用 set value 去指定？

S：（將變數 z 拖曳至程式第一行，設定為 z set value to 0.5）

S：（拖曳 pj's height 取代 0.5，修改為 z set value to pj's height）

在設定變數值的操作中，有 5 名學生誤將該指令放置於物件身高改變之後。這 5 名學生皆由訪談者告知應於目標物件改變大小前儲存物件身高，例如：

S13_Q03

T：你在這邊做第一次的縮放，對不對？那你在第一次縮放之前要先把它的身高指定給它（變數 hi）啊！

S：嗯。

3. 使用變數進行計算

完成變數宣告及定值之後，有 7 名學生無法將變數使用於運算中。其中 1 名學生在訪談者詢問比例計算方式後，可正確回答並將變數用於計算：

S06_Q03

T：這邊恢復原來的，所以應該是什麼比例？

S：小孩原來的比例。

T：這個比例應該怎麼算？

S：Alice 的身高除以小孩的身高。

T：剛剛放大的時候是 Alice 除以小孩的身高，那現在呢？要縮回來應該是什麼？

S：小孩除以 Alice。

T：對，那小孩除以 Alice 怎麼做？

S：（將第二次 resize 縮放倍率修改為變數 z/aliceLiddell's height）

另外 6 名學生則需訪談者提示取用原來的身高後，始可正確拖曳變數用於計算中，例如：

S01_Q03

T：你這裡的高度，是小孩原來的高度還是已經放大的高度？

S：放大的高度。

T：對，那你要怎麼用原來的高度？

S：（將第二次 Mana resize 縮放倍率調整為變數 high / aliceLiddell's height）

第四題：選擇結構

本題目標為測驗學生能否正確取用選擇結構 if/else、正確設定判斷條件、並將目標程式碼置於正確區塊。具體而言，學生需要使用函式 ask user for a number 向使用者要求輸入一個數字，並以使用者所輸入的數字來控制執行縮放動作的物件。本題程式碼範例如圖 4.8，此程式將接受使用者輸入的數字，若使用者輸入為正數，則將較矮物件（如程式範例中的 plato）變大與較高物件（如程式範例中的 aliceLiddell）等高後再縮回原身高；若使用者輸入為負數，則將較高物件縮小與較矮物件等高後再放大回原身高。

本題有 1 名學生不需要訪談者提供協助即可順利完成程式，其餘 12 名學生答題所遭遇到的困難及訪談者給予的協助統整於表 4.4 中。

1. 取用控制結構 if/else

作答的 13 名學生中，有 3 名不知道必需取用 if/else 來處理兩種不同的情況。其中 1 名學生在訪談者提示「如果」這個關鍵字後，即了解需取用 if/else：

S04_Q04

S：這個如果是要負數，我們要設什麼？

T：你看一下影片你再告訴我。

S：（開啟 Q4 題目影片）S：就是，如果是正數，它就變，就是小的那隻要變大…

T：好，再重複一次。

S：如果是正的

T：「如果是」

S：（將 if/else 結構移入程式中）

另 2 名學生在訪談者詢問應如何進行判斷動作後，可順利拖曳 if/else 控制結

構至程式中，例如：

S05_Q04

T：你打算怎麼去判斷是誰要變大變小？

S：（將 if/else 拖曳至程式碼最後，並設判斷條件為 true）我覺得應該是用這個。

在將控制結構 if/else 拖曳至程式的過程中，有 1 名學生發生放錯位置的情形，在訪談者告知正確位置後改正：

S11_Q04

T：你先問了它一個數字對不對？

S：對。

T：問完之後呢 ？才能去判斷它是…

S：正還是負。

T：對，所以你要問完之後才能判斷。所以一定是在問完之後做。

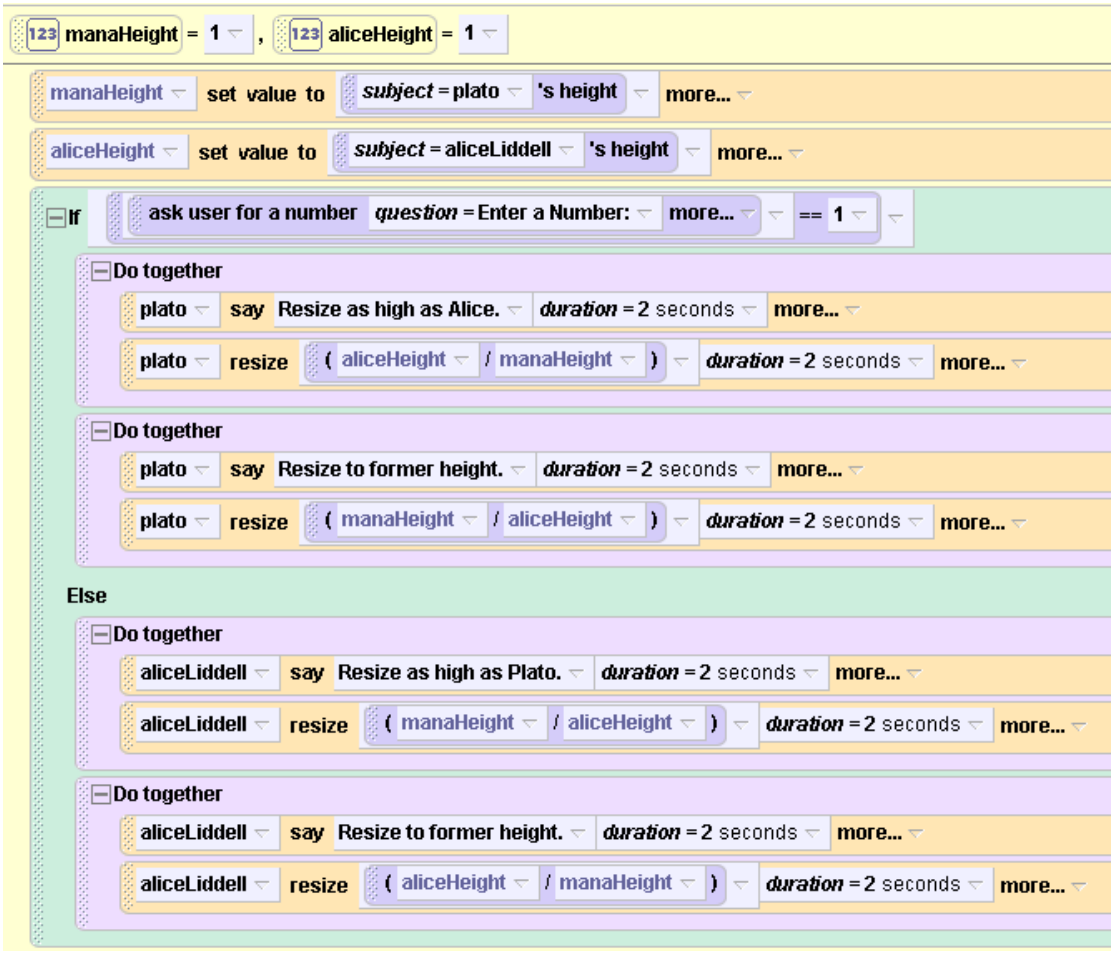


圖 4.8 程式碼範例－選擇結構

表 4.4 學生遭遇困難以及訪談者給予協助情形－選擇結構

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
取用控制結構 if/else	不知道必需使用控制結構 if/else	3	提示關鍵字"如果"	1
			提問如何進行判斷	2
	If/else 程式碼放錯位置	1	告知程式碼應放置於向使用者要求輸入數值之後	1
設定 if/else 判斷條件	未宣告變數以儲存輸入值*	8(9)	提問第一步動作後學生反問確認操作是否正確	1
			告知應儲存輸入值	2
			告知應宣告一個變數儲存	5
	無法正確取用函式 ask user for a number	9	詢問是否知道函式位置	1
			提示此動作與特定人物無關	3
			告知尋找物件 world 函式庫	5
	不知如何設定判斷條件	1	告知設定判斷條件位置	1
	不知道正負數的判斷方法	3	詢問正負數定義	2
			告知正負數定義	1
	無法取得比較判斷式	2	學生反問確認操作是否正確	1
			告知判斷函式 a > b 位置	1
	無法將函式 ask user for a number 用於判斷式	3	詢問何者為輸入值	3
Else 中動作編寫	不知道必需使用變數暫存儲存原身高	3	提示身高已經改變	3
	設定變數值之程式碼位置錯誤	3	告知應於改變物件大小的程式碼前設定	3
	身高比例計算錯誤	4	詢問縮放倍率	1
			告知比例計算錯誤	3
於 if/else 中放置對應程式碼	無法於 if/else 中分別放置正確程式碼	3	詢問兩種情況下應執行的動作為何	3

註：在本題中宣告變數以儲存數字並非必要動作，因此在學生人數欄的括號中註明本題中如此實作的學生數

S：問完之後做。

T：對啊！所以你剛剛拉 if 沒有錯，但是你放錯地方了，要問完之後。

S：（將 if/else 拖曳至程式碼最後，並設判斷條件為 true）

2. 設定 if/else 的判斷條件

在本題中，有部分學生（9 名）使用變數以儲存使用者輸入的數值。此一動作在完成本題中並非必要，因為函式 ask user for a number（要求使用者輸入一個數值）可直接用於 if/else 的判斷條件中。有 8 名學生在宣告變數的過程中遇到困難，其中 1 名學生在訪談者詢問學生第一步需執行動作後即可順利宣告變數及回答其用途：

S06_Q04

T：所以第一步要做的是什麼呢？

S：（點擊 if/else 判斷條件選單，瀏覽後取消）

S：要先創造（指 create new variable）

T：你說要先建立變數是不是？

S：嗯。

S：（點擊 create new variable）

T：這個變數的用途是什麼？

S：數字，就是看輸入正數還是負數。

另有 2 名學生在訪談者告知需儲存輸入值後，便知必需宣告一個變數，例如：

S07_Q04

T：你要讓它輸入一個數字，還是一樣要把這個數字記錄下來。所以你一樣要有一個東西去記錄你輸入的數字。

S：這個？（指向 create new variable）

T：嗯。

其餘與 5 名學生則是由訪談者告知需使用變數儲存使用者輸入的數字，例如：

S01_Q04

T：你剛剛那樣做（拖曳函式 ask user for a number）是對的。那個方程式為什麼拉不進來，是因為讀回來的數字需要指定給一個變數來儲存。

S：（思考約 30 秒）

T：那變數應該怎麼產生？

S：按這個（指 create new variable）

T：嗯。

另有 1 名學生直接將 if/else 判斷結構及物件動作放置於新增的方法(method)中，透過參數來傳遞使用者輸入的數值。

大部分的學生（9 名）不知道如何取用函式 ask user for a number 來實作「向使用者要求輸入數字」的動作。其中 1 名學生在訪談者詢問是否知道該函式的所在位置後可順利找到：

S05_Q04

S：缺這個（指 Q2 題目影片中的 ask a number 視窗），這要如何設定？

T：你知道它在哪嗎？

S：(點擊物件區中的 world)

S：(點擊 function 標籤)

S：應該在 world 裡面。

S：(尋找 20 秒後將函式 ask user for a number 拖曳至程式中尋找可以放置的位置)

另有 3 名學生在訪談者提示「向使用者要求輸入一個數字」這個動作與特定人物無關以後，即能夠找到函式 ask user for a number 的位置，例如：

S13_Q04

T：一樣，你這邊要讀一個數字。那這個動作跟特定人物有關嗎？

S：嗯…

T：沒有，所以左上角物件資訊區要選什麼？

S：(點擊左上物件區 world) 應該還是 world。

T：對。

S：(尋找 20 秒後將函式 ask user for a number 拖曳至 if 判斷條件)

其餘 5 名學生則需訪談者告知 ask user for a number 函式位於物件 world 的函式庫中，方可順利找到，例如：

S04_Q04

T：接下來呢？怎麼樣輸入數字？

S：我不會輸入數字。

T：你不會。

S：我不會輸入數字。

T：那你可以選物件中的 world。

S：物件？

T：物件。

S：（點擊物件區中的 world）

T：的 function。

S：（點擊 function 標籤）

T：在這裡面，你覺得哪一個會是？

S：這個（指向 ask user for a number）

順利取用函式 ask user for a number 之後，有 1 名學生不知如何設定 if/else 的判斷條件，而將函式 ask user for a number 拖曳至錯誤位置：

S07_Q04

T：你要把它（變數）拉到哪裡？你剛剛的動作沒有錯，但是你拉錯地方了。

S：（試圖將變數 ask 拖曳至 if 中）

T：我再講一次好了，「如果它是正數」，所以它一定是放在如果…

S：後面。

T：你再拉一次。

S：（將變數 ask 拖曳至 if 中）這邊？

T：「如果後面」是什麼？

S：（將變數 ask 拖曳至 if 判斷條件處）

在設定判斷條件時，需判斷輸入值是否大於（或小於）零來決定它是否為正數，並依判斷結果進行相對應的動作。有 3 名學生無法理解此概念，其中 2 名學生在訪談者詢問正負數定義後，可理解判斷條件應如何設定，例如：

S01_Q04

T：正數是他（Mana）變大，負數是他（Alice）變小。那你怎麼判斷正負？

S：正負…

T：對。

S：這個（指向 world 函式庫中 either a or b, or both）

T：你看一個數值是正數或負數的定義是什麼？

S：大小，大於零小於零。

另 1 名學生則需訪談者告知正負數定義：

S03_Q04

T：什麼情況是小孩動，什麼情況是 Alice 動？

S：輸入數字的正負。

T：對。

T：那一個數字的正負怎麼判斷，跟誰比？

S：（思考約 10 秒）

T：我們不是都跟 0 比嗎？比 0 大就是正數。所以這邊條件式要怎麼改？

S：大於 0。

有 2 名學生在取得比較函式上發生困難，其中 1 名學生在找到後反問訪談者確認是否正確：

S03_Q04

S：（將 if / else 拖曳至主程式區，並設定判斷條件為 true）

S：（指向函式 a > b）這有可能負的嗎？

T：對，因為它條件式的形式就是這樣。

S：（拖曳函式 a > b 至 if/else 判斷條件中，並設定為變數 mana1 > 2）

另外 1 名學生需要訪談者直接告知比較函式的取得方式：

S02_Q04

T：你有沒有辦法判斷大小？

T：你按 world，選 function。大於。

S：（點擊物件區 world，點擊函式庫，拖曳函式 a > b 至 if 判斷條件，並設定為 0.25 > 0.25）

在設定判斷條件時，有 3 名學生未能使用函式 ask user for a number 或儲存其數值之變數於判斷條件中。這些學生在訪談者詢問何者為輸入值後，皆可正確設定判斷條件，如下：

S02_Q04

T：那誰大於零？

S：數字。

T：對，那數字在哪裡？

S：這裡（指向 question set value to ask user for a number）

T：那個是指定的動作，數字是存在哪個變數裡頭？

S：這裡（指向變數 question）

T：對。

S：（修改判斷條件為變數 question > 0）

2. else 動作編寫

本題 else 中需執行的動作基本上與第三題相同，差異僅在於動作物件為較高物件需縮小至與較矮物件同高，再恢復成原本身高。在編寫過程中，有 3 名學生使用改變後的身高進行比例計算，而未使用變數儲存其原始身高，以致無法完成復原身高的動作。在訪談者提示身高已經改變後，這 3 名學生皆理解需使用變數，例如：

S07_Q04

T：他（Alice）變不回來了。

S：變大。

T：你先思考一下為什麼。Alice 變小了，所以現在 Alice 的身高跟…

S：他（Plato）一樣高。

T：跟他一樣高。

S：喔！所以 Alice 也要去設定一個東西

T：對。

S：（點擊 create new variable，新增變數 height2）

S：（將變數 height2 拖曳至程式第一行並 set value to aliceLiddell’s height）

另外 3 名學生將設定變數值的程式碼放錯位置，在訪談者告知後進行修正，例如：

S06_Q04

T：你想想看，你在這邊保存身高有意義嗎？來到這邊他們的身高已經變了，那你要儲存的是原來的身高，還是變完之後的身高？

S：原來的。

T：原來的，所以呢？

S：要往前移

S：（將 x set value to pj’s height 及 c set value to aliceLiddell’s height 移至程式最開始處）

在計算縮放倍率的算式中，有 3 名學生發生計算錯誤的情況，其中 1 人在訪

談者詢問算式內容後順利解決：

S06_Q04

S：（將變數 x 拖曳至程式中，並 set value to pj's height）

S：（將 pj resize 拖曳至 if 中，設定縮放倍率為 x）

S：（按下 PLAY，輸入 3 進行測試，發現 pj 不會回復原身高）

T：你想想看，你這邊用的 x 的值是多少？

S：（將 if 中縮回的 resize 縮放倍率調為變數 x / aliceLiddell's height）

另有 2 名學生在訪談者告知縮放倍率有誤之後，能夠修正錯誤，例如：

S04_Q04

S：（按下 RESTART 進行測試，輸入-3 發現 Alice 放大至與 Plato 等高又變大）

T：怎麼會變大？

T：你這裡是不是寫錯了？

S：這裡？是兩個寫反了嗎？

T：你覺得呢？

S：（將 Alice 縮回的縮放倍率改為 Plato's height / aliceLiddell's height）

4. 在 if 及 else 分別放置對應程式碼

多數學生能將正確的執行步驟分別放入 if 及 else 後方，但仍有 3 名學生須經訪談者提醒後，方能正確設定，例如：

S04_Q04

T：大於零要做什麼？

S：這個（指向原 Q3 程式碼）

T：所以就是放大再縮小。

S：（將原 Q3 程式碼拖曳至 if 後）

S：然後…

T：否則呢？

S：就是再做那個，他變小。

第五題：重複結構（固定次數）

本題目標為測驗學生能否正確取用重複結構 loop 來使目標動作重複執行。具體而言，本題要求學生拖曳重複結構 loop 至正確位置、設定重複次數，並將目標

動作移入 loop 之中。本題程式碼範例如圖 4.9，此程式將接受使用者輸入的數字，若使用者輸入為正數，則執行「將較矮物件（如程式範例中的 plato）變大與較高物件（如程式範例中的 aliceLiddell）等高後再縮回原身高」此動作三次；若使用者輸入為負數，則執行「將較高物件縮小與較矮物件等高後再放大回原身高」動作三次。

本題所有學生皆有作答，其中 8 名學生在不需訪談者給予任何提示即可正確完成，其餘 5 名學生答題所遭遇的困難以及訪談者給予的協助統整於表 4.5。

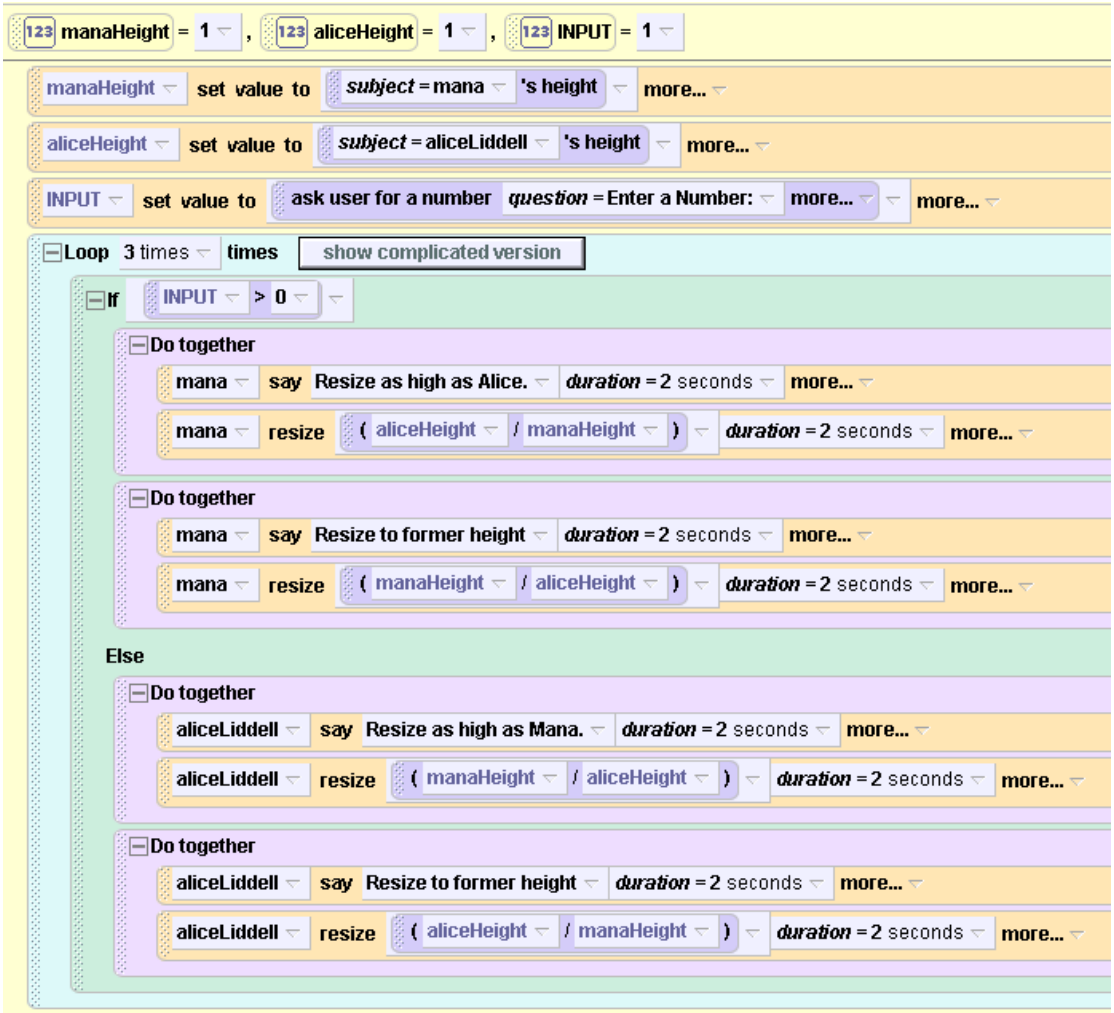


圖 4.9 程式碼範例－重複結構（固定次數）

表 4.5 學生遭遇困難以及訪談者給予協助情形－重複結構（固定次數）

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
取用重複結構 loop	不知道需使用重複結構 loop	4	學生反問確認操作是否正確	2
			提問與前一題（Q4）差異	1
			提示物件動作次數為三次	1
設定 loop 重複次數	重複次數設定錯誤	1	詢問重複次數為何	1
於 loop 放置正確動作	無法於 loop 中放置正確程式碼	1	詢問哪個動作需重複	1

1. 取用重複結構 loop

在確認本題題意之後，有 4 名學生不知道需取用重複結構實作「重複特定動作數次」的功能。其中 2 名學生無法確定自己選擇是否正確，例如：

S02_Q05

T：這一題要你做什麼？

S：它會一直做。

T：做幾次？

S：三次。

S：所以是要用這個嗎？（指控制結構列的 loop）

T：試看看。

有 1 名學生誤以為本題須修改控制結構 if/else 的判斷條件，在訪談者詢問本題與前一題（Q4）差異後，順利完成本題：

S05_Q05

S：那這個（指 if 判斷條件）要改嗎？就是改成等於零或是…

T：不是。我問你這一題跟上一題有什麼差別？

S：上一題就是輸入正數的話就是小的會變大，然後輸入負值的話它會變小。

T：這一題呢？

S：這一題的話就是看正負號，正號的話它會變大再縮小，然後如果是負號的話它會變小再變回來。

T：所以這一題跟上一題的差異是在？

S：次數。

S：所以只要動…

S:（將 if 判斷條件中的函式 ask for a number > 函式 random number 改回函式 ask for a number > 0

S：（將 loop 拖曳至 if 中，設定重複次數為 2）

S：（開啟 loop 的 complete version，調整重複次數為 3）

另有 1 名學生則在訪談者告知物件動作次數為三次之後，即可正確取用重複結構 loop 進行操作：

S02_Q05

T：它要做三次，你現在只做一次而已

S：（拖曳 loop 至程式中，並設定重複次數為 3）

2. 設定 loop 重複次數

在本題中，物件動作重複次數為一固定值 3，學生在設定迴圈重複次數時，僅需指定常數 3 即可。有 1 名學生將重複次數設定錯誤，經訪談者提示後修正：

S03_Q05

S：（拖曳 loop 至 if 中，並設定重複次數為 4 次）

S：（點開 loop 的 complete version）

T：這邊次數是固定的，還是變動的？

S：固定的。

T：對，固定的。那是幾次？

S：3 次。

T：對。

S：（修改 loop 重複次數為 3 次）

3. 於 loop 中放置正確動作

學生在取用重複結構 loop 後，需將欲重複執行的動作程式碼拖曳至 loop 中。在所有學生中，有 10 名學生是在控制結構 if 及 else 中分別新增一個重複結構 loop 來進行解題。另外 3 名學生則是新增一個重複結構 loop，並將整個 if/else 移入，這兩種方法皆能順利達成本題解題目標。

有 1 名學生在取用重複結構後，並未將目標動作移入 loop 中，由訪談者提問哪個動作需重複：

S11_Q05

S：（按下 PLAY 進行測試，輸入正數後發現 Plato 僅做一次縮放）

T：你要讓誰做三次？

S：（拖曳原 if 中程式碼至 if 中 loop 裡，但未放開）

T：你還沒放進去啊！

S：（放開拖曳物件將其移入 loop 中，並將其他 if 中程式碼移入 loop）

第六題：重複結構（不定次數）

本題目標為測驗學生能否透過函式及變數來控制迴圈重複次數。具體而言，學生在本題中需將使用者輸入的數值儲存於變數之中，並將其視不同狀況加以適當處理後，用以控制迴圈執行次數。其餘動作則與第五題相同。本題程式碼範例如圖 4.10。

本題學生作答所遭遇的困難以及訪談者給予的協助統整於表 4.6。

1. 宣告變數以儲存使用者輸入數值

在未使用變數儲存使用者輸入數字的 3 名學生中，有 1 名學生在訪談者提示後才理解需宣告變數，例如：

S06_Q06

T：你在哪一行程式讀數字的？

S：這裡（指 if 判斷條件中的函式 ask user for a number）

T：好，那在這裡讀完之後有存下來嗎？還是讀完只是拿來判斷？

S：拿來判斷。

T：對啊！

S：所以要存起來（指向 create new variable）

T：對，要存起來。

S：（按下 create new variable，新增變數 v）

S：（將變數 v 拖曳至程式第三行，並 set value to 1）

S：（試著將 if 判斷式中的函式 ask user for a number 直接拖曳至 v set value to 1 但失敗）

S：（從 world 函式庫中將函式 ask user for a number 拖曳至 v set value to 1 取代 1）

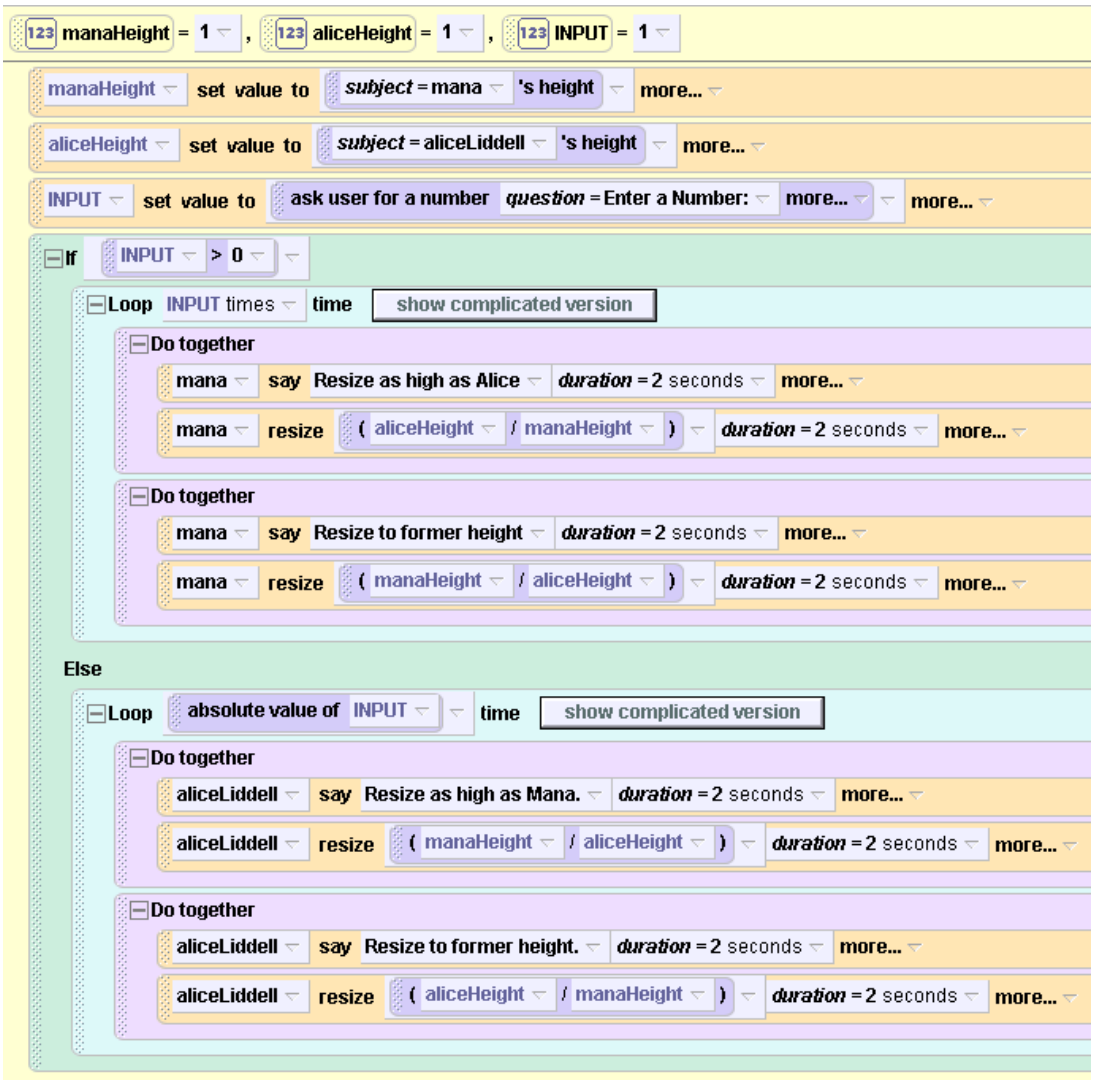


圖 4.10 程式碼範例－重複結構（不定次數）

另 2 名學生則是由訪談者直接告知需要宣告變數：

S13_Q6

S：數字…

T：你這裡有沒有把那個輸入的數字存起來？

S：(指向 if 判斷條件) 在這邊。

T：對阿，那有沒有存？還是只是拿來判斷？

S：判斷。

T：對，那有沒有存？

S：好像沒有？

T：對啊！那你要怎麼存？

S：嗯？

T：你要宣告一個變數來存啊！

S：宣告變數。

T：對，宣告變數剛剛不是做過了。

S：（點擊 create new variable）

S：宣告變數，可以直接…

T：你需要另外一個進行儲存。

S：（完成新增變數 hi2）

S：所以，再拉一個過來。

S：（將變數 hi2 拖曳至 if 中，出現選單，選擇 set value to，設定為變數 hi）

T：你這個變數 hi2 是要做什麼用？不就是要儲存讀進來的數字嗎？

S：這不算嗎？

T：這樣不算阿，你 hi 的值是 1 啊！你這邊是把 hi 的值指定給 hi2。hi2 這個變數是要做什麼的？

S：（將 ask user for a number 拖曳至變數 hi2 set value to 變數 hi 取代變數 hi）

表 4.6 學生遭遇困難以及訪談者給予協助情形－重複結構（不定次數）

解題目標	遭遇到困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
宣告變數以儲存讀入數字*	未正確宣告一個變數以儲存使用者輸入的數字	3(3)	提示應儲存輸入值	1
			告知應儲存輸入值	2
	變數設定值程式碼位置錯誤	2(3)	提示需於進行判斷前進行設定	2
	未將變數使用於判斷條件	2(3)	告知判斷條件中「要求使用者輸入數字」動作重複	1
			詢問應使用何者判斷	1
迴圈次數控制	無法使用變數作為迴圈次數	4	詢問變數儲存內容及用途	1
			告知需使用變數以保存輸入值	3
	無法將負數的輸入值轉為迴圈執行次數	13	學生反問確認操作是否正確	1
			提問如何將負數轉為正數	2
			告知如何將負數轉為正數	10

註：在本題中宣告變數以儲存數字為必要動作，但在 Q4 中已有 8 名學生完成此解題目標，因此在學生人數欄的括號中註明本題中實作的學生數。

宣告變數時，有 2 名學生將變數接收使用者讀入數字的程式碼放置於判斷控制結構之後。這 2 名學生在訪談者提示程式執行順序之後進行修正，例如：

S05_Q06

T：所以一開始你先拿到了一個數字，放到 n 裡，然後你再拿那個 n 的數字去判斷，他是不是大於零或小於零，對吧？

S：嗯。

T：所以？

S：所以是要拉過來這裡。

S：（將 n set value to ask user for a number 移至程式第一行）

完成變數宣告後，有 2 名學生未將該變數取代原先選擇結構判斷條件中的函式 ask user for a number，導致在執行過程中出現兩次「向使用者要求輸入數字」的動作。其中 1 名學生在訪談者提示後完成修改：

S06_Q06

T：這裡讀了一次，下面的 if 有沒有再讀了一次？

S：所以…

S：（將變數 v 拖曳取代 if 判斷條件中的函式 ask user for a number）

另有 1 名學生在訪談者詢問應使用何者作判斷後，理解需使用變數於判斷條件中：

S05_Q06

T：你用誰來判斷？

S：hi2

T：對啊！

S：（將 if 判斷條件調整為 if hi2 > 0）

2. 迴圈次數控制

13 名學生中，有 4 名學生無法將儲存輸入值的變數用於控制迴圈重複執行次數，其中 1 名學生在訪談者詢問變數儲存內容及用途後，便能正確設定：

S13_Q6

T：你想想看這個 hi2 要扮演的角色是什麼？

S：這個（變數 hi2）等於這個（變數 hi）

T：不是，剛剛 hi 是用來暫存身高嘛，那現在的 hi2 他用來暫存什麼東西？

S：輸入的數字。

T：對，輸入的數字。那輸入的數字在我們這邊要用來控制什麼？

S：次數。

T：對，次數。所以 hi2 要用在哪裡？

S：（將 if 中 loop 的重複次數設為變數 hi2）

T：對。

其餘 3 名學生則需使用者明確告知需使用保存輸入值的變數作為重複次數，例如：

S06_Q06

T：Alice 動，那要動幾次？比如說他輸入了負三，難道要動負三次嗎？

S：（點開 if 判斷條件選單，瀏覽未做任何設定）

T：我們先看讀到數字大於零的情況…

S：（按下 play 進行測試，輸入 2 進行測試，因為未使用 loop，僅縮放一次）

T：要做 v 次嘛，那怎麼樣做 v 次？

S ：（將 loop 移入 if 中，並設定重複次數為變數 v）

所有學生在設定迴圈重複次數時，皆直接使用儲存使用者輸入值的變數作為迴圈重複次數。導致當使用者輸入數值為負數時，會因為迴圈重複次數無法執行負數次，而使得目標物件不進行任何動作。正確的解決方法之一乃是透過數學運算或是絕對值函式（absolute value of A）將變數值自負數轉為正數。其中 1 名學生在觀看執行結果後，向訪談者提問確認修改目標：

S09_Q06

S：（按下 PLAY 進行測試，輸入 2 測試成功）

S：（按下 RESTART 進行測試，輸入-2 發現不會動）

S：這裡錯？（指向 else 中 loop 重複次數）

T：對。

S：（將 else 中的 loop 次數從變數 number 修改為變數 number*(-1)）(在這裡的文字敘述中需要加上括號，否則會看不懂)

另有 2 名學生在訪談者詢問如何將負數轉為正數後，自行找出方法：

S13_Q06

T：這個（變數 hi2）是負的，比如說你輸入負四，那這邊要做四次。要做四次的話這邊（重複次數）應該要是四。你想想看負四怎麼變成四？

S：math…乘以…

S：（將 else 中 loop 次數從變數 hi2 修改為變數 hi2 * (-1)）

S：乘以負一。

其餘 10 名學生則需訪談者告知變數轉正的計算方式或絕對值函式所在位置，例如：

S06_Q06

T：（變數）question 是負的，要怎麼變正的？

S：（搜尋函式庫）

T：負數怎麼轉成正數？

S：不知道。

T：這邊有兩個方法，一個是取絕對值，一個是用零去減它。

S：（將 else 中 loop 次數從變數 question 修改為 0 -變數 question）

第七題：陣列

視覺化陣列（ArrayVisualization）為 Alice 的陣列結構之一，其特色為將陣列結構化為實體的站台（如圖 4.11 所示），並可將場景中的物件指定為陣列中的元素。被指定為元素的物件會移動到陣列的站台上（如圖 4.12 所示）。如同一般的陣列，程式設計者可以透過索引去控制其所對應之物件。

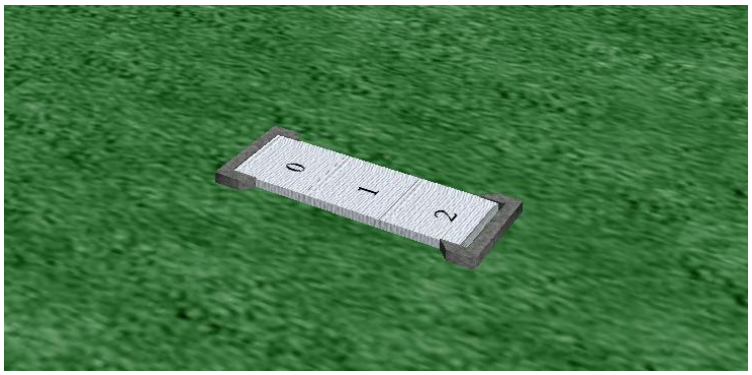


圖 4.11 視覺化陣列示意圖

圖 4.12 視覺化陣列及其元素示意圖

本題需要學生正確宣告及設定視覺化陣列，並在完成陣列宣告後，透過陣列中的元素（elements）來控制對應物件的動作。本題程式碼範例如圖 4.13，此程式將接受使用者輸入的數字，若使用者輸入正數，則使陣列上的物件依序執行「同時向左轉 90 度及縮放至與陣列中第一個物件等高」的動作；若使用者輸入負數，則使陣列上的物件依序執行「同時向右轉 90 度及縮放至與陣列中最後一個物件等高」的動作。

本題學生作答所遭遇的困難以及訪談者給予的協助統整於表 4.7。

1. 視覺化陣列的宣告與設定

本題的第一個動作為宣告一個視覺化陣列（ArrayVisualization）及數個物件，並將物件指定於視覺化陣列上中。視覺化陣列的宣告方式為在新增物件時選擇物件類別 Visualizations 中的 ArrayVisualization；當宣告後出現如圖 4.14 的陣列內容設定畫面時，可按下「New Item」按鈕來增加陣列長度，以及指定物件至陣列上。設定完成後按下 OK 按鈕即可完成視覺化陣列的宣告。若在宣告視覺化陣列時未進行上述設定，亦可以在宣告完成後，點擊視覺化陣列的屬性欄（properties）中的 element（元素）按鈕，便會出現同樣的設定畫面。

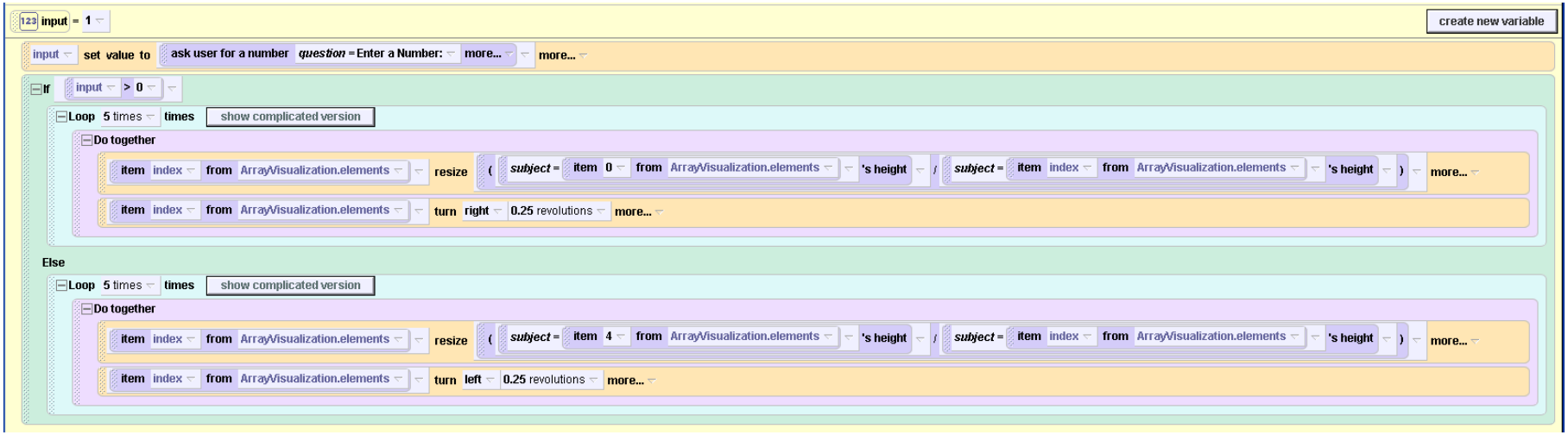


圖 4.13 程式碼範例－陣列

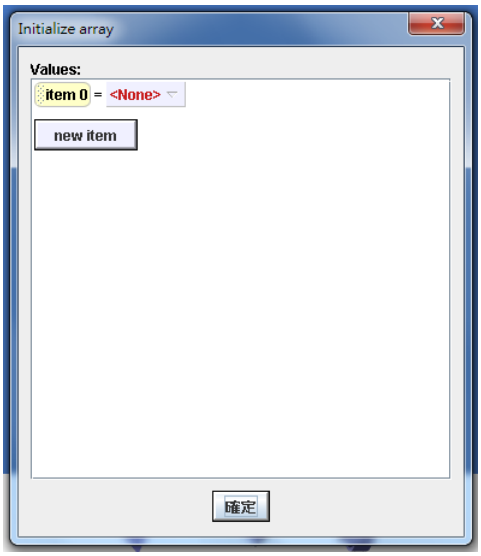


圖 4.14 視覺化陣列內容設定畫面

表 4.7 學生遭遇困難以及訪談者給予協助情形－陣列

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
視覺化陣列的宣告與設定	不知如何宣告視覺化陣列	7	告知宣告視覺化陣列的物件類別及位置	7
	不知如何將物件指定至視覺化陣列中	4	提示不可手動將場景中物件移至視覺化陣列上	2
			告知物件設定方法	2
取用控制結構 if/else 及判斷條件的設定	不知需使用控制結構 if/else	1	提示關鍵字「如果」	1
	未將函式 ask user for a number 指定給變數	3	提示需儲存輸入值	3
選取動作物件	無法順利取得視覺化陣列上之物件以進行動作	10	告知物件位於視覺化陣列中	2
			告知如何取得視覺化陣列中之物件	8
轉動物件	未使用內建方法 turn 實作轉動	2	告知需使用內建方法 turn	2
	轉動角度錯誤	3	提示轉動角度錯誤	1
			告知轉動角度	2

續表 4.7 學生遭遇困難以及訪談者給予協助情形－陣列

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
使用重複結構控制視覺化陣列上物件	不知需使用重複結構 loop	10	提示多人需做同樣動作	5
			告知需使用重複結構 loop	5
	重複次數設定錯誤	5	詢問重複次數為何	3
			講解 loop 內容	1
			告知重複次數應設定數值	1
	不知需透過迴圈索引（index）做為控制	9	學生反問確認操作是否正確	1
			提示動作物件非單一物件	2
			說明視覺化陣列索引與迴圈索引之間的關係	6
縮放物件	未使用內建方法 resize 實作縮放	2	詢問縮放指令為何	2
	resize 縮放比例計算錯誤	4	告知 resize 後方數字為比例或倍數	1
			詢問比例應如何計算	3
	不知如何取得視覺化陣列上物件身高做為比例計算	9	告知取得陣列上物件身高的方法	9

在 13 名學生中，有 7 名學生因為不知道視覺化陣列所處的類別及位置，無法順利進行陣列的宣告，這些學生皆需要訪談者告知視覺化陣列在新增物件類別中所處的位置，例如：

S06_Q07

S：（調整物件順序及位置）

S：版子是在哪一個類別中？

T：在 Visualization 裡，因為它是一個視覺化的東西。

S：（點擊類別 Visualization）

S：（點擊 arrayvisualization，取消後點擊 listvisualization）

T：是剛剛那一個，arrayvisualization。

S：（按下確認新增 listvisualization）

T：不是這一個，是 arrayvisualization。

S：（刪除 listvisualization）

S：（點擊 arrayvisualization）

在視覺化陣列的長度設定上，所有學生皆能順利完成。至於陣列內容的設定，則有 4 名學生無法順利將物件指定至陣列中。其中兩名學生誤認為僅需在場景中將物件手動移至視覺化陣列的板子上方即算完成，在訪談者告知不能如此操作後隨即找到正確的方法，例如：

S01_Q07

S：（點擊新增 ArrayVisualization，並按下 New Item 按鈕 5 次）

T：他一開始就會要求你指定值。那如果你沒有值的話，可以先關掉。

S：（點擊 OK 按鈕，完成新增 ArrayVisualization）

S：（新增人物物件 AliceLiddell、blueballerina、Bob、Euripides、Mana）

S：（將場景中人物物件移至 ArrayVisualization 上）

T：你如果想要指定，必須回到 ArrayVisualization 裡，不能直接這樣移。你現在透過指定值的方式，來把人物放上去。

S：（尋找後點擊 ArrayVisualization 的 properties 中的 element，出現設定視窗）

T：對。

S：（將人物物件指定至 ArrayVisualization 中）

S：（按下 DONE 完成場景建置）

另外 2 名學生則需要訪談者明確告知指定物件的方法，例如：

S06_Q07

T：你要把人物放上來是不是？

S：要寫程式讓他們站上來？

S：（點擊 DONE 完成場景建置）

T：不是，不能用拖拉的…

S：（點擊 Add Object 重新開啟場景編輯，未做任何操作又按下 DONE 結束場景建置）

T：…你要指定陣列的值。那怎麼去指定陣列的值呢？選陣列，選 ArrayVisualization，然後在 properties，在 elements 這裡，點下去編輯。

S：（點擊 element 並指定 item 內容）

2. 取用控制結構 if/else 及判斷條件的設定

在本題中，仍然需要使用者輸入一個數值作為判斷執行動作的依據。由於本題需要使用者重新進行程式的編寫，因此發生有 1 名學生忘記需使用控制結構 if/else 來進行判斷，需要訪談者提示關鍵字「如果」:

S11_Q07

T：我們輸入了一個數，是不是還是要判斷？如果是正的，向右轉，如果是負的，向左轉。我們先把這個部分做出來。

S：（思考約 10 秒）

T：你剛剛有做過，「如果」…

S：（將 if/else 拖曳至程式中，並設定判斷條件為 true）

而在判斷條件的設定上，有 3 名學生宣告變數用於判斷條件中，但卻未將變數值設定為函式 ask user for a number，導致程式執行時未能向使用者要求輸入數值。這些學生皆在訪談者提示需要儲存使用者輸入後發現並修正，例如：

S07_Q07

S：（按下 create new variable 新增變數 aaa）

S：（拖曳 if/else 進程式中，並將判斷條件設為 aaa > 0）

T：你還沒輸入。

S：不是這個(指變數 aaa)大於零然後…

T：可是你要先輸入，才能知道他是不是大於零。不然他永遠等於 1。

S：（按了幾次變數後方的選單，但未做修改）

T：我們的題目要求什麼？你要先把 aaa…

S：（將 ask for a number 拖曳至程式中但無位置可以放）

S：啊！這個(變數 aaa)要先拉進來。

S：（將變數 aaa 拖曳至程式開頭，並設定為 aaa set value to ask user for a number）

S：這樣。

3. 選取動作物件

本題需要針對視覺化陣列上的個別物件進行動作。選取陣列上物件的方法為點選視覺化陣列物件中的屬性區（properties），將 elements（元素）拖曳至程式中（如圖 4.15）。有 10 名學生無法順利完成上述操作。其中有 2 名學生在訪談者提示動作物件位於視覺化陣列中後便能順利進行取用，例如：

S02_Q07

T：視覺化陣列裡面的值要怎麼找？

S：（思考約 10 秒）

T：在 ArrayVisualization 裡。

S：（點擊左上方物件區 ArrayVisualization）

S：（拖曳函式 the value of ArrayVisualization [index]，但放棄）

S：（拖曳 element 取代 if 中動作物件，並索引設為 1）

其餘 8 名學生則需訪談者告知取得視覺化陣列中物件的方法，例如：

S03_Q07

T：你看，是整個 ArrayVisualization 去 turn 嗎？

S：不是。

T：是上面的元素嘛，那怎麼樣去選到上面的元素？

S：不知道。

T：選它。

S：（點選物件區 ArrayVisualization）

T：然後這裡有一個 properties，陣列的元素都在這裡（指 elements）。

4. 轉動物件

本題需要學生使用內建方法 turn 來實作物件轉動的動作。有 2 名學生不知道需要使用內建方法 turn，必須透過訪談者告知，例如：

S07_Q07

S：（將 aliceLiddell turn 移入 if，並設定 left…）

T：正數表示右轉。

S：（取消 turn 設定）

T：那…

S：他沒有 face to。

T：就是 turn 沒錯阿。

S：是 turn 還是 face to？

T：Turn。

S：（將 AliceLiddell turn 移入 if，並設定為 left）

在設定轉動角度時，有 1 名學生誤設為轉動 1/2 圈（180 度），在訪談者提示應設定為 1/4 圈（90 度）後進行修正：

S04_Q07

T：他右轉多少度？

S：（設定 iceSkater turn right 0.5 revolutions）

S：（將 index 拖曳至程式中，還未放開）

T：等一下，你先執行看看。

S：（按下 PLAY 進行測試，發現 iceSkater 一直轉）

T：你看他一次轉多少？是不是轉半圈？

S：喔，所以要 1/4。

S：（設定 iceSkater turn right 0.25 revolutions）

另有 2 名學生則是被訪談者直接告知應設定角度為 90 度：

S10_Q07

T：這裡操作還記不記得？

T：你先隨便抓一個人出來轉。

S：（拖曳 iceSkater turn 至 if 中的 loop 裡）

T：右轉。

S：（設定 iceSkater turn right 0.5 revolutions）

T：轉 90 度，你現在這是轉半圈喔！

S：（設定 iceSkater turn right 0.25 revolutions）

5. 縮放物件

本題依然需要使用內建方法 resize 來使目標物件進行縮放的動作。有 2 名學生未使用內建方法 resize 進行實作，需經訪談者提示 resize 所在的位置：

S03_Q07

S：然後放大。

T：嗯。

S：（學生思考約 15 秒）

T:什麼指令可以讓她放大,你已經作完轉向了。放大是在 properties 還是 method,還是 function 裡頭？

S：（點擊 function）

T：人物做動作的指令都放在 method 裡頭。

S：這裡(指向方法 arrayVisualization resize)

T：嗯。

在計算縮放倍率時，有 4 名學生發生計算錯誤或不知如何設定縮放倍率的情形。其中 1 名學生在指導者告知 resize 後方數字為比例或倍數後進行修改：

S13_Q07

S：（將縮放倍率設定為 joey's height）

S：（按下 PLAY 進行測試，輸入正數所有人依序執行「依序轉動及縮小一定比例」）

S：怎麼變小？

T：那個 resize 後面是比例。

S：所以…

S：（修改縮放比例為 jeoy's height / index）

另外 3 名學生則需要訪談者告知如何計算比例，例如：

S01_Q07

T：如果要跟 Alice 一樣高的話，這個比例要怎麼算？

S：（學生思考約 30 秒）

T：這個比例要怎麼算？

T：它們除以 Alice。那你要怎麼去拿到 Alice 的身高？

在計算比例的過程中，需要使用到視覺化陣列上物件的身高作為計算。在 Alice 程式語言中，無法直接選取視覺化陣列上物件的身高，必須先從其他任一物件中取得身高的函式（ＯＯ's height），再拖曳陣列元素加以取代。有 9 名學生無法順利完成，皆需訪談者告知做法，例如：

S09_Q07

T：你現在是要拉身高是不是？

S：嗯。

T：那身高我們是透過 function 去取得，所以你要先從 function 這邊拉身高過來，再去指定是誰的身高。

6. 使用重複結構控制視覺化陣列上的物件

本題可使用重複結構 loop 來控制陣列上所有物件依序進行動作，但是大部分（10 名）學生並無法將「有很多人需要執行同樣的動作」與使用重複結構進行連結。其中有 5 名學生在訪談者詢問「如果有更多的人要執行同樣的動作應如何處

理？」或是「重複執行需要使用什麼控制結構？」後，方可理解需要使用重複結構，例如：

S12_Q07

T：如果我要重複從第一個人做、第二個人做、第三個人、做然後第四個人做，有什麼指令是可以造成重複的效果？

S：重複…loop 嗎？

T：嗯。

S：loop，所以…

S：（將 loop 移至 if 中的 do in order 裡）

其餘 5 名學生則需訪談者告知需使用重複控制結構 loop，例如：

S01_Q07

T：所以 arrayvisualization 上面五個人都發生同樣的動作嘛，那同樣的動作發生五次，只是不同的人物做，這個要怎麼樣？

S：（學生思考約 20 秒）

T：應該是要用 loop，do in order 的話你要拉五個人的程式碼。

S：（拖曳 loop 至 if 中，並設定重複次數為無限次）

S：（調整 loop 重複次數為 5 次）

完成重複結構的取用後，有 5 名學生發生迴圈重複次數設定錯誤的情況。其中 3 名學生在訪談者詢問重複次數時，可正確回答並進行修正，例如：

S03_Q07

S：（拖曳 loop 至 if 中，並設定重複次數為 4 次）

S：嗯？

T：你想想看，這裡要幾次？

S：五次。

T：嗯

S：（調整重複次數為 5 次）

另有 1 名學生在設定時，誤將迴圈設定中「每次增加」當作「迴圈次數」進行設定，在訪談者講解後進行修正：

S12_Q07

S：（將迴圈重複次數設為 from 1 up to (but not including) 4 times incrementing by 5）

S：（按下 PLAY 進行測試，輸入正數，發現只有前兩個人轉）

T：你把英文完整唸出來告訴我它的意思是什麼。

S：從 1 到…But not including 5，by…這是什麼意思？

S：（將迴圈重複次數設為 from 1 up to (but not including) 4 times incrementing by 4）

S：（按下 PLAY 進行測試，輸入正數，發現只有前兩個人轉）

T：你一個字一個字跟我講是什麼意思。

S：從 1 至，不包含 5 次，by…

T：一次增加。

S：一次加…4！

T：你把整個唸一次。

S：所以是從 1 不包含 5，增加 4

S：（將迴圈重複次數設為 from 1 up to (but not including) 4 times incrementing by 1）

T：應該是從 1 開始，一直到 5，但是不包含 5，每次增加 1。你剛剛是每次增加 4，所以如果每次增加 4 的話，會變成…

S：5。

T：1，那就跳到 5。

S：就跳到 5。

T：對。

有 1 名學生則需訪談者告知重複次數，方能正確設定：

S02_Q07

T：為什麼會只有第一個人轉？

S：（刪除 if 跟 else 中單獨的轉身指令）

S：（按下 PLAY 進行測試，輸入 2，僅陣列中第二個人轉動）

T：跟 question 無關，都是要五個人。

S：（調整重複次數為 6）

T：不是六次。

S：（調整重複次數為 5）

因為陣列索引（index）及迴圈控制索引皆是從 0 開始，至 4 結束，所以應使用迴圈的索引來控制視覺化陣列上的物件，使其依序進行動作。有 9 名學生不知道需將陣列索引與迴圈索引做連動，其中 1 名學生向訪談者反問確認自己的操作是否正確：

S03_Q07

S：（拖曳 elements 移至 resize 動作物件，出現選單）

S：這裡是選這個（index）？

T：嗯。

S：（修改縮放倍率為 element[index] resize euripide’s height /av’s height）

另有 2 名學生在訪談者提示動作物件並非只有單一物件後，可正確修改元素索引值為 index，例如：

S11_Q07

T：接下來，我們現在就要想辦法變讓所有人都轉。

S：（將 element 拖曳取代動作對象，出現選單）

S：老師這是什麼？

T：你先隨便點一個，我再來解釋。

S：（點選 3，設定為 element[3] turn right 0.25）

T：好，我們現在看到的這個是什麼 item…

S：第三個…第四個人！

T：所以這樣寫是第四個人轉。照你這樣寫的方式就是全部都是第四個人轉。

S：（學生思考約 10 秒）

S：（將 index 拖曳取代 3，修改為 element[index] turn right 0.25）

T：對了！

其餘 6 名學生則需訪談者說明陣列與迴圈關係後，才知道需要使用迴圈索引 index 控制視覺化陣列物件，例如：

S06_Q07

T：這個迴圈會執行五次，那我第一次要讓 iceSkater 這個人轉，再來是 Alice、老人、PJ，最後是這個。那怎麼去控制不同的人轉呢？

S：（學生思考約 30 秒）

T：陣列，這幾個人在陣列上，他們的位置都不一樣。所以我可不可以用迴圈的索引來代表陣列上人物的位置，來指定這次要轉的人。比如說第一次值是 0，下一次是 1，那再下一次是 2…

S：那是要？

S：（將 element 拖曳至 loop，試圖取代 index，但因不是數字無法放入）

T：他這裡已經有 index 了。

S：所以…

T：我們每個人都要讓他右轉…

T：（將 iceSkater turn 移入 loop，並設定為 right 0.25 revolutions）

T：（將 AliceLiddell resize 移入 loop，設定縮放倍率為 0.5）

T：而且要跟 iceSkater 一樣高。那接下來我們每個人都要右轉…

T：（將 element 拖曳取代 iceskater turn 的 iceskater，出現設定物件索引選單）

T：想想看這裡要選什麼？

S：index。

T：對，那這樣就會第一次是 iceSkater，下一次是 Alice。

S：然後再來就是老人。

第八題：自訂方法（User-defined Methods）

本題目標為測試學生能否正確地宣告自訂方法，並於主方法中（my first method）中呼叫自訂方法，本題範例程式碼如圖 4.15、圖 4.16 及圖 4.17。此程式執行結果與第七題相同，差別僅在於需將判斷使用者輸入的數字後需執行的「使視覺化陣列上的物件依序執行同時向左轉 90 度及縮放至與陣列中第一個物件等高」及「使視覺化陣列上的物件依序執行同時向右轉 90 度及縮放至與陣列中最後一個物件等高」這兩個動作分別獨立為自訂方法。

本題學生作答所遭遇的困難以及訪談者給予的協助統整於表 4.8。

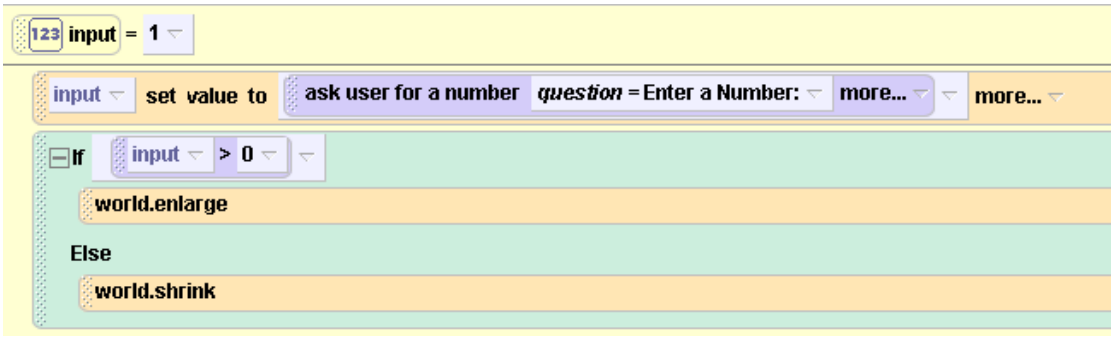


圖 4.15 程式碼範例－自訂方法（my first method）

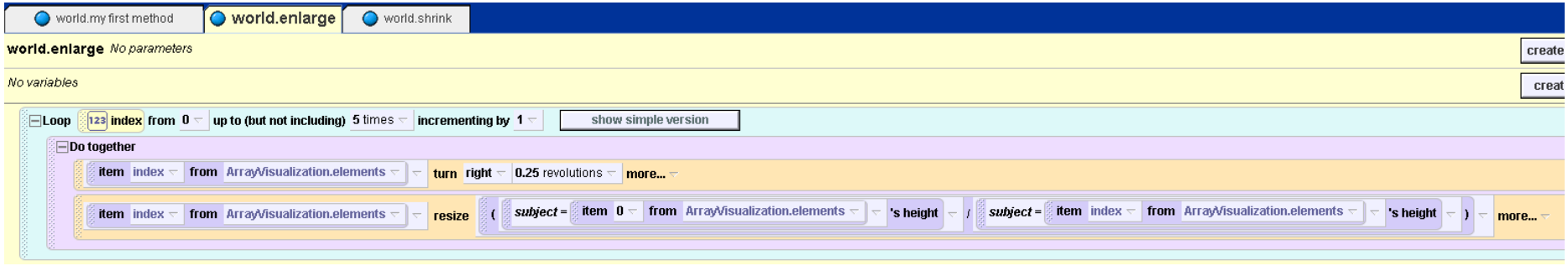


圖 4.16 程式碼範例－自訂方法（自訂方法 enlarge）

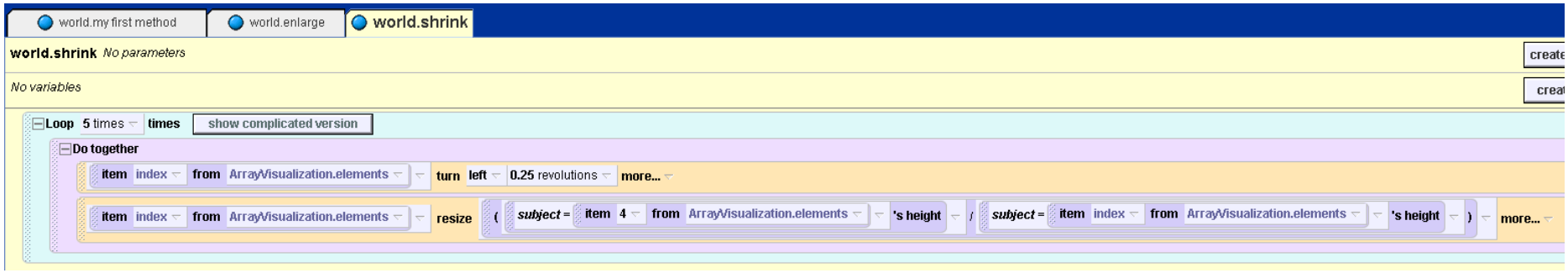


圖 4.17 程式碼範例－自訂方法（自訂方法 shrink）

表 4.8 學生遭遇困難以及訪談者給予協助情形－自訂方法

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
宣告及編寫自訂方法	不知如何宣告自訂方法	5	學生反問確認操作是否正確	1
			提示宣告錯誤並告知方法為 method	4
	無法於自訂方法中放置正確程式碼	4	學生反問確認操作是否正確	1
			告知方法中需執行的動作	2
			提示程式碼尚有缺漏	1
	未於自訂方法中補宣告變數	2	提示有程式碼中有缺漏（<NONE>）情況發生	2
呼叫自訂方法	未呼叫自訂方法	2	學生反問確認操作是否正確	1
			詢問如何使用自訂方法	1

1. 宣告及編寫自訂方法

因為本題的執行成果與第七題完全相同，所以訪談者皆於學生開始編寫程式時，即說明本題需要使用自訂方法來分別處理兩個動作。在 Alice 中，宣告自訂方法的作法為按下位於方法庫中的「create new method」按鈕，如圖 4.18 紅色框線所示，在按下「create new method」按鈕後，會出現自訂方法名稱的設定視窗，輸入自訂方法名稱後，按下 OK 按鈕即可完成宣告。

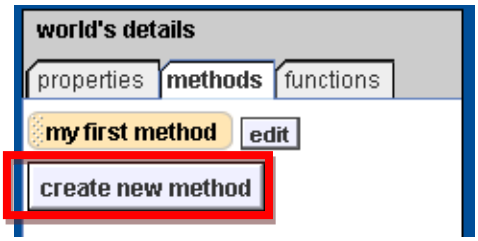


圖 4.18 自訂方法宣告按鈕位置

有 6 名學生無法順利完成自訂方法的宣告，其中有 1 名學生無法確定自己的操作是否正確，需向訪談者確認：

S03_Q08

T：我們本來是大於零執行這段，小於零執行這段。那現在這兩段要分別定義成一個 method，然後再從 my first method 去呼叫。等於你要定義兩個 method，一個是負責做縮放加右轉；另外一個 method 是做縮放加左轉。

S：（點擊 create new method）

S：是按這個？

T：嗯。

另外 4 名學生則是進行參數或自訂函式的宣告，經訪談者告知宣告錯誤，並提示關鍵字 method 後，方可順利找到「create new method」按鈕進行宣告，例如：

S04_Q08

T：做成一個 method，自訂 method。

S：（點擊 create new parameter）

S：這個嗎？

T：不是。

S：（關閉 create new parameter 設定視窗）

T：自訂 method。

S：（點擊 create new method）

S：是這個嗎？

S：（按下 cancel 關閉 create new method 設定視窗）

S：（點擊左上方物件區 world，點擊 create new method，並輸入名稱 turn 完成宣告）

在完成自訂方法的宣告後，有 4 名學生發生不知道如何設定自訂方法中內容的情況。其中 1 名學生在與訪談者確認自己的操作後，可順利完成：

S03_Q08

S：然後這裡面（自訂方法 turnright）要放這個（指 if 中的 loop）。

T：對，那個 method 的功能就是這邊這個 loop 做的功能。

S：有辦法直接複製嗎？

T：你要試試看。

S：（透過剪貼板複製 if 中的 loop 至自訂方法 turnright 中）

另有 2 名學生在訪談者再次說明方法中需執行的動作後，方能順利將正確的程式碼複製至自訂方法中，例如：

S13_Q08

T：你現在要做什麼動作？

S：把這個全部都移到這邊。

T：如果你把它移到那邊，那你怎麼呼叫它？

S：嗯…我再想想。

S：（刪除自訂方法 hello 中程式碼）

T:我們現在是要把他的動作分別定義在不同的方法裡頭,然後再從 first method 裡頭去呼叫。

S：（透過剪貼板複製 if 中 loop 至自訂方法 hello 中）

有 1 名學生則是在複製的過程中有所遺漏,需要訪談者提示程式碼尚有缺漏:

S04_Q08

S：（將 if 中 loop 裡的 do together 複製至方法 turn 中貼上，但部分變數出現<NONE>）

S：還有嗎？

T：只有這一塊嗎？應該不只。

S：把 loop 整個…

S：（複製整個 loop 但失敗）

S：那這個要先刪掉。

S：（清空自訂方法 turn）

S：（透過剪貼板複製 if 中 loop 至自訂方法 turn 中）

在複製程式碼至自訂方法的過程中，若是被複製的程式碼有使用到變數，在貼上時原本變數的欄位會變為<NONE>（無內容物）的狀態。有 2 名學生在複製的過程中發生此情況，並於訪談者提示有缺漏後，可理解並進行正確的修正，例如：

S07_Q08

T：有東西不見了。

S：這個(指 my first method 中的變數 resize)，這個東西不見了。

T：所以你要在另一邊…

S：創造。

S：（在自訂方法 right 中新增變數 resize 並用以取代<NONE>）

2. 呼叫自訂方法

在呼叫自訂方法時，有 1 名學生向訪談者詢問確認自己的想法是否正確，在得到確認後可順利進行自訂方法的呼叫：

S04_Q08

S：所以我可以把這個(my first method 中 if 裡的 loop)刪掉換成這一個(指自訂方法 turn)？

T：對。

S：（刪除 my first method 中 if 裡的 loop 部分，拖曳方法 turn 至 if 中）

另外有 1 名學生則是在呼叫自訂方法時，誤以為需要使用控制結構 print（印出），經訪談者詢問如何呼叫自訂方法後，找出正確的做法：

S05_Q08

T：你知道要去哪裡用你剛剛寫好的那兩個自訂方法嗎？

S：這裡。

S：（點開 world 的方法庫，指向自訂方法 turnRight 及 turnLeft）

T：那你知道怎麼用嗎？

S：（將自訂方法 turnRight 拖曳至 if 中，自訂方法 turnLeft 拖曳至 else 中）

第九題：參數

本題目標為測試學生能否於自訂方法中宣告參數，用以接收來自呼叫端的數值進行相關處理。具體而言，學生必須於自訂方法中新增一個參數，這個參數的功能是接收來自 my first method 中的使用者輸入數值，作為自訂方法中判斷應執行動作的依據。本題範例程式碼如圖 4.19 及圖 4.20，此程式執行結果與第七題相同，但其中必須使用參數進行資料的傳遞與接收。

本題有 1 名學生在實作第八題時，即在未透過訪談者協助的情況下使用參數進行資料傳遞，其餘 12 名學生作答所遭遇的困難以及訪談者給予的協助統整於表 4.9。

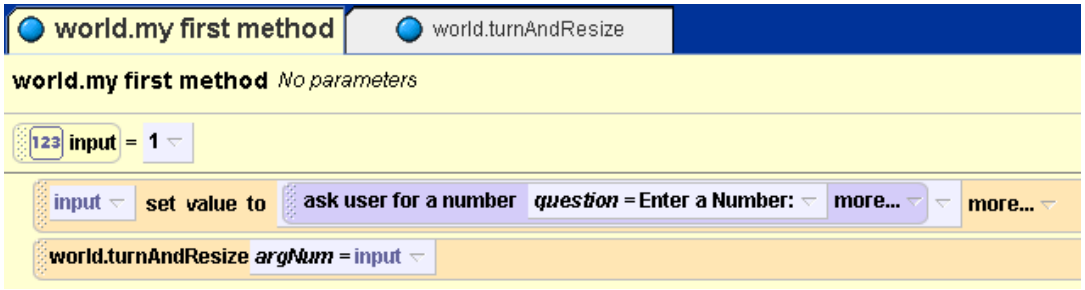


圖 4.19 程式碼範例－參數（my first method）

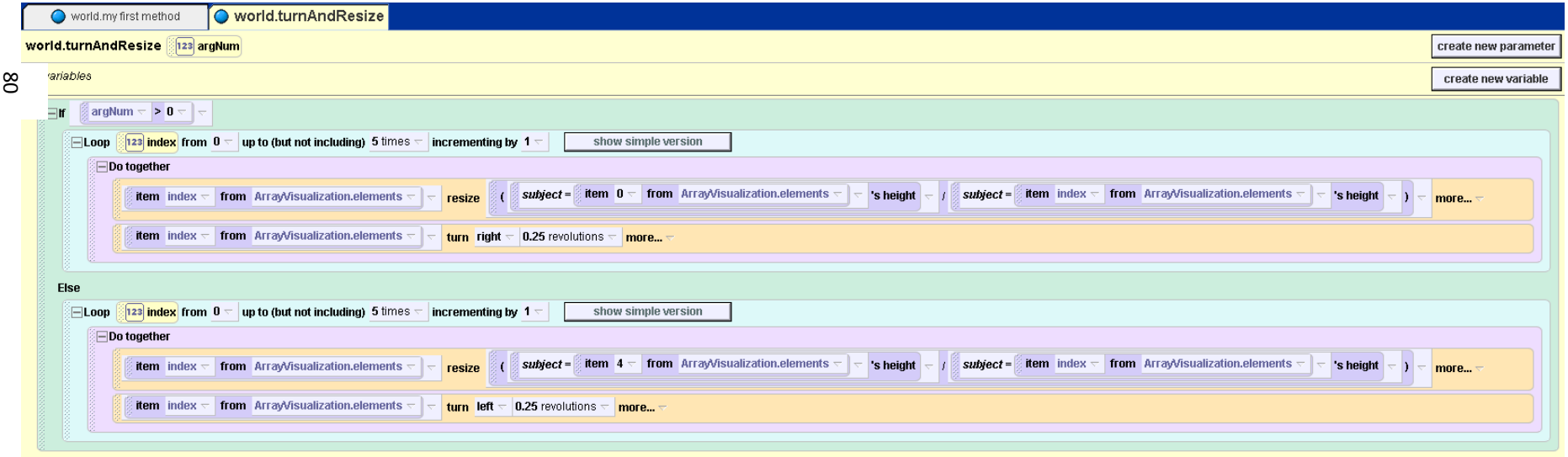


圖 4.20 程式碼範例－參數（自訂方法 turnAndResize）

表 4.9 學生遭遇困難以及訪談者給予協助情形－參數

解題目標	遭遇的困難	遭遇此困難的學生數	訪談者給予的協助	得到此協助後順利解決問題的學生數
自訂方法新增或修改	未於自訂方法中使用選擇結構 if/else	4	提示應於方法中使用 if/else	1
			告知應於方法中使用 if/else	3
於自訂方法中新增及使用參數	不知道如何宣告參數	8	學生反問確認操作是否正確	1
			告知宣告錯誤	2
			告知宣告參數方式	5
	未於正確方法中宣告參數	3	提示宣告位置錯誤	2
			告知應於自訂方法內新增	1
	未將參數用於自訂方法中	5	學生反問確認操作是否正確	1
			提示學生檢查程式碼	1
			告知應使用的位置	3
呼叫自訂方法	未呼叫自訂方法	4	提示需呼叫自訂方法	4
	未正確指定傳遞值	8	詢問傳遞值為何	5
			告知應傳遞值	3

1. 自訂方法新增或修改

實作本題的過程中，需要將使用者輸入的數值傳遞至自訂方法中進行判斷，因此必須新增或修改原有的自訂方法，並在其中使用選擇結構 if/else 實作判斷的動作。有 4 名學生在修改過程中未將選擇結構 if/else 移入自訂方法中，其中 1 名學生在訪談者提示需進行判斷後，向訪談者反問確認是否需要使用 if/else：

S06_Q09

T：你現在定義了一個參數，那就表示說我們等一下呼叫（自訂方法）zzz 的時候會傳一個值給它，這個值就用 a（參數名）來代表。所以你要去判斷說我傳進來的值有沒有大於零，有的話我就要做…

S：向右轉。

T：對，或者我就向左轉。一個情況是做左，一個是做右。那要怎麼寫？

S：（點至自訂方法 zzz）

S：所以我要在這邊用 if 嗎？

T：對啊！

其餘 3 名學生則需訪談者告知需使用 if/else，例如：

S13_Q09

T：我是根據他的值來決定要做什麼動作，不是根據他的值來決定要做幾次。

S：喔！

S：（將參數 ri 拖曳至 loop 中，但未放開）

T：這邊有沒有 if/ else 敘述？

S：有耶。

T：對啊！

S：（將 if /else 拖曳至 loop 中）

2. 於自訂方法中新增及使用參數

在 Alice 中，新增參數的方式為按下程式編輯區右上方的「create new parameter」按鈕，如圖 4.21 框線所示。按下該按鈕後會出現與宣告變數時類似的參數設定畫面，可在此進行參數名稱、類別等相關設定的修改，設定完成後按下 OK 按鈕即可完成參數宣告。

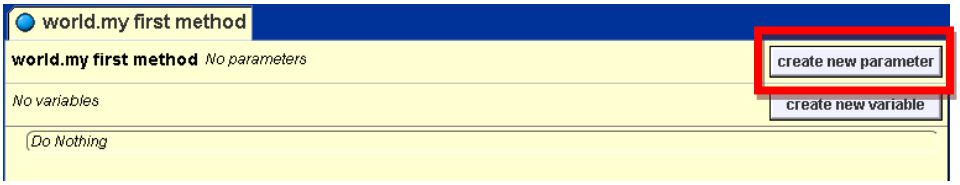


圖 4.21 參數宣告按鈕位置

8 名遭遇困難的學生中，有 1 名學生在訪談者告知需要新增參數後，向訪談者確認自己的選擇是否正確：

S04_Q09

T：你要在這裡面（自訂方法 turn）設置一個參數。

S：這個嗎？ (指 create new parameter)

S：（點擊 create new parameter，新增參數 aaa）

有 2 名學生在新增參數時點擊錯誤，誤選了新增變數按鈕「create new variable」，經訪談者告知錯誤後可正確按下「create new parameter」按鈕，例如：

S06_Q09

S：（點擊 create new variable）

T：那是建立變數。

S：那是這個。

S：（點擊 create new parameter）

T：嗯。

其餘 5 名學生則需訪談者告知參數的宣告方法，例如：

S07_Q09

T：還記得什麼是參數嗎？不記得，那我們先點（自訂方法）askm，你剛剛一直想點一個東西，可是我都沒有讓你點的，就是右邊有一個 create…

S：這個喔？

S：（點擊 create new parameter）

T：對，參數的意思就是說，比如我們告訴電腦 sin90 度，那他就會告訴我 sin90 度是 1/2。那我要先告訴他 90 度他才會告訴我 1/2，所以 90 度就是我給他的參數。所以這是同樣的意思，我在呼叫這個 method 的時候，他會給我一個參考的數值讓我知道要做什麼事情。

S：嗯。

T：好，那你就給那個參數一個名字吧！

S：（輸入參數名稱 number，完成新增參數）

宣告參數時，有 3 名學生於錯誤的方法中進行宣告。其中 2 名學生在訪談者提示宣告位置錯誤後，即自行修正錯誤，例如：

S02_Q09

T：你怎麼宣告在這裡？

S：（刪除 my first method 中宣告的參數 dir）

S：（在方法 turn 中點擊 create new parameter，新增方向參數 dir）

另有 1 名學生則需訪談者告知應於何處宣告參數：

S04_Q09

T：我現在希望你能夠把這裡面(my first method)的 aaa 傳給他，所以我們要用到一個參數。

S：這個嗎？（指控制結構列中的 print）

T：不是，參數，你現在要到 turn 裡面設參數，world turn。

S：（點擊至自訂方法 turn）

T：你要在這裡面設置一個參數。

宣告完參數之後，有部分（5 名）學生未將參數使用於自訂方法中，其中有 1 名學生向訪談者提問確認參數用途：

S03_Q09

S：所以是要表示這個數字(參數 a)是不是大於零？

T：對，就是要用它來判斷。

S：（拖曳參數 a 至 if 判斷條件中，並設為 a > 0）

另有 1 名學生在訪談者提示後找到缺漏部分，並使用參數補上：

S04_Q09

T：你現在輸入負的變小，輸入正的呢？

S：都變小。

T：我們先回過頭來看一下 right 這裡

S：（點至自訂方法 right，未發現錯誤）

T：那 trunleft。

S：（點至自訂方法 trunleft，未發現錯誤）

T： world trun 看一下

S：（點至自訂方法 turn，發現有判斷條件為<NONE> > 0）

S：啊！這邊！(指<NONE>)

S：（拖曳參數 bbb 取代<NONE>）

其餘 3 名學生則需要訪談者告知參數應使用的位置，例如：

S07_Q09

T：現在這個 if 判斷條件不知道。那我希望是由主程式告訴你說 if 這裡面要放什麼東西

S：所以是這裡…

S：（試圖將自訂方法 askm 拖曳取代 if 判斷條件中<NONE>但失敗）

T：不是，我們要把 number 放進來。

S：（拖曳參數 number 取代判斷條件中<NONE>，修改為 number > 0）

3. 呼叫自訂方法

有 4 名學生在實作本題時，新增了額外的自訂方法來接受 my first method 中

的使用者輸入數值及進行判斷，但卻忘記需要於 my first method 中進行呼叫，需要訪談者予以提示，例如：

S11_Q09

T：這一段（my first metod 中的 if）以經複製了，所以可以刪掉對不對？

S：(my first metod 中的 if)

T：接下來呢，我們這一段寫在哪裡？剛剛我們寫的那個…

S：這裡（指自訂方法 m3）

T：對。

S：(將自訂方法 m3 拖曳至 my first metod)

有 8 名學生未於 my first method 中呼叫自訂方法時給予正確的傳遞值，其中 5 名學生在訪談者提示後自行修正，例如：

S03_Q09

S：(拖曳方法 turnright 至 my first metod 中第一行，瀏覽指定傳遞值選單)

S：呼叫這個嗎？

T：呼叫剛剛定義的那個方法，那要傳誰給它，傳什麼值給它？我們剛剛定義了一個參數，那表示說我們呼叫的時候要傳一個值給它。那這邊傳什麼值？

S：(選擇傳遞變數 question， question 為接收使用者輸入數值之變數)

S：這個。

T：對。

其餘 3 名學生則需訪談者直接告知所需傳遞者為何，例如：

S04_Q09

S：(將自訂方法 turn 拖曳至 my first method 中，出現指定傳遞值選單)

S：先隨便設？

T：不是，你要把你輸入的數傳給它，你要把這邊的 aaa 傳給它的 bbb。

S：(設定傳遞值為 1)

S：所以要把 aaa…所以還要再多設一個？

T：把它拉進去。

S：拉進去就好？

S：(將變數 aaa 拖曳取代傳遞值 1)

第二節 錯誤類型分析

本節將學生於各個程式設計概念中所犯下的錯誤狀況進行類別及原因的分類，並統整各程式概念中常見的錯誤類型。

一、錯誤類型

本研究根據「學生在撰寫程式時的不同階段」將錯誤狀況進行分類，分別為連結（connection）、定位（location）、使用（usage）和數學與其他（math and others）四類，以下將詳細敘述各錯誤類型定義及可能發生原因。

1. 連結錯誤：

此類錯誤為無法將題目影片中的動作或欲實作的功能與目標程式概念進行對應，常見的發生形式為「未使用／不知道需要ＸＸ」或「不知道怎麼做ＸＸ」。此類錯誤發生的原因除了學生缺乏對目標程式概念的理解外，亦有可能是由於題目無法清楚的進行呈現導致。

2. 定位錯誤：

此一分類是由於 Alice 程式的程式編寫環境為拖放式（drag-and-drop）而導致的錯誤狀況，常見的發生形式為「無法找到拖曳目標／點擊目標按鈕」。可能的發生原因包含 Alice 本身介面設計不良及學生缺乏宣告或取用的相關知識。

3. 使用錯誤：

此類錯誤泛指所有學生在使用目標程式構句時所發生的錯誤，常見的發生形式為「內容／條件設定錯誤」、「程式碼位置錯誤」等。發生原因為學生對於目標程式構句之使用方法缺乏或不熟悉。

4. 數學與其他錯誤：

其他與程式概念無直接關聯的錯誤皆歸類於此類別中，以本實驗觀察到的狀況而言，其中之一為學生在完成正確的操作後未主動進行測試，而是直

接向訪談員反問進行確認；另一種則是數學應用相關的錯誤，意即學生無法順利使用數學進行相關的處理，如無法順利計算倍率、負數轉正的操作。

二、各程式概念之錯誤類型發生狀況

此處將分別針對學生在實作各程式概念時所犯下的錯誤類型進行統計並進行相關探討。

1. 內建方法

本實驗在各實作題目中要求學生使用不同的內建方法來操作物件，使其做出目標動作，所使用的內建方法包含 say（說話）、move（移動）、set color to（改變顏色）、resize（縮放大小）以及 turn（轉向）。由於不同的內建方法之使用方法複雜程度有所差異，進而導致所發生錯誤類型亦有不同，發生情形統整於表 4.10。

表 4.10 錯誤類別及原因－內建方法

實作對象	連結錯誤	定位錯誤	使用錯誤
方法 say	V		
方法 move	V		
方法 set color to		V	
方法 resize		V	V
方法 turn	V		V

方法 say 在第一題中學生首次使用時，並未發生任何錯誤情形。但在後續題目中，有部分學生並未進行說話動作的撰寫，但在訪談員提示後即可順利補上，推測應為學生忽略或忘記進行實作所導致。

在方法 move 的使用上，由於在題目中所使用的方式為組合兩個移動動作成為一跳躍動作，無法實作的學生多半認為 Alice 已內建「跳躍」方法，故持續的在物件方法庫中尋找，需要訪談員進行說明方可順利取用方法 move 完成實作目

標。

在方法 set color to 的使用上，有 7 名學生無法順利取得此內建方法，其原因為方法 set color to 的取得方法和其他一般內建方法不同，是自物件的屬性中進行拖曳，而非自方法庫中拖曳。因此研究者認為此一情況的發生是由於 Alice 介面設計不良所導致。

在使用方法 resize（改變大小）時，有 6 名學生需要訪談員告知方能順利找到方法 resize 來實作縮放動作，但方法 resize 之取用方法與其他內建方法並無差異，因此推測學生可能是由於英文不熟悉導致無法找到。另一方面，所有學生皆誤以為 resize 的功能為「縮放至指定數值的大小」，而非 「縮放為指定倍率的大小」，雖然其中有 2 名學生在觀察執行成果或程式碼後可以看出 resize 所接收之數值為縮放倍率，但其餘 11 名學生無法從程式執行結果推敲出該數值之意涵。值得注意的是，當使用者將內建方法 resize 拖曳至程式中使用時，會出現如圖 4.22 的預設選單，其中的選項皆有英文說明，因此推測學生們在取用方法時，並未仔細觀看選項內容便直接進行操作。

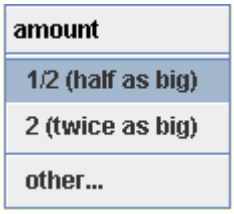


圖 4.22 resize 設定選單

在使用方法 turn（旋轉）來實作旋轉動作時，有 2 名學生認為需要使用方法 turn face to（轉面向目標）來進行實作，而非 turn。研究者認為此一錯誤情況同樣來自於學生對於方法功能的誤解。而在使用時學生也因為未仔細閱讀而將 1/2 round（半圈）誤認為轉身 90 度。

2. 時序控制

在本次實驗中所使用的時序控制結構包含 do in order（依序執行）及 do together（同時執行）來進行物件動作順序的控制，以及設定指令時間控制動作時間長短，錯誤發生情形統整於表 4.11。

表 4.11 錯誤類別及原因－時序控制

實作對象	連結錯誤	定位錯誤	使用錯誤
Do together	V		V
Do in order	V	V	V
指令時間控制	V		

在使用同時控制結構時，學生多是因為未注意到有動作需同時進行而未進行取用，但是只要訪談者進行相關的提示，學生即可順利的取用及進行使用，表示學生對於同時結構的取用有著一定程度的理解與熟悉。

和同時控制結構相比，學生在定位和使用循序控制結構上都較為生疏。由於在第一題中是將同時與循序組合使用，有 9 名學生未將循序控制結構使用於同時結構中以完成實作目標。其中有 7 名學生在訪談者提示或告知「變色動作和整個跳躍動作須同時執行」後，直接將「向下移動」的程式碼移入原本僅包含「向上移動」及「變色」程式碼的 do together 中，使得「向上移動」及「向下移動」這兩個衝突動作置於同一個 do together 中（如圖 4.2），造成程式執行時目標物件僅變色未跳動。研究者推測學生可能誤認為 do together 中的程式碼可以將其中某些部分同時執行，其他部分依序執行。

在指令執行時間的控制上，學生同樣因為難以觀察的因素而未進行設定，但是只要訪談者進行相關的提示，學生即可順利的進行設定。

3. 內建函式

在本實驗中所使用的內建函式包含 height（身高）、ask user for a number（要

求使用者輸入數字）以及 a > b 或 a < b（比較），錯誤發生情形統整於表 4.12。。

表 4.12 錯誤類別及原因－內建函式

實作對象	連結錯誤	定位錯誤	使用錯誤
height	V		
ask user for a number		V	
a > b、a < b		V	

在實作題目中，物件身高是作為縮放倍率計算使用，由於學生在訪談員提示後即可順利進行使用，因此推測學生未進行取用主要是因為未意識到「縮放至與目標同高」所需的計算需使用身高而未取用。

另外要求使用者輸入數字及比較函式的位置因為其共通性，並未放置於物件函式庫中，而是位於世界（world）函式庫裡，導致學生在取用時較難以找到。

4. 變數

在變數概念中，連結、定位及使用三種類型的錯誤皆有發生，以下分別進行狀況說明及討論。

在第三題中首次使用變數時，幾乎所有的學生都不知道需要使用變數來進行資料的暫存以協助解題或計算。即使在第三題中已經使用變數儲存資料，在第四題中，仍有 3 名學生需要訪談者提示「目標身高已經改變」才進行變數宣告以暫存身高；第四題及第六題中，共計有 11 名學生仍需要訪談者提示或告知，才宣告變數儲存使用者輸入數值（ask user for a nubmer 函式），顯示大多數學生明顯缺乏「使用變數來暫時儲存數值」的概念。

而在定位方面，由於變數宣告按鈕與參數宣告按鈕十分接近，導致有 4 名學生將變數宣告與參數宣告混淆。其中有 2 名學生誤宣告為參數（parameter）；另外 2 名學生在宣告變數及宣告參數的按鈕之間游移，無法做決定。

在使用上，設定變數值時有 2 名學生認為可將欲進行儲存的函式直接拖曳至

參數預設值中進行指定。但在 Alice 中，數字變數的預設值僅能設定為常數，若要指定為特定函式，則須將變數拖曳至程式中進行設定。另一方面，在使用變數作為運算使用時，學生會發生的錯誤多為忘記進行使用，但在操作方面並無明顯問題。

5. 選擇結構

選擇結構中所發生的錯誤集中於連結與使用兩類，學生並未發生定位方面的錯誤。在第四題中，大多數學生皆能正確取用 if/else 選擇結構進行輸入數值的判斷，僅有 3 名學生需要訪談者提示，才理解需要使用 if/else。在使用上，僅有 1 名學生無法設定選擇結構的判斷條件，需要訪談者告知設定方法。

6. 重複結構

選擇結構中所發生的錯誤集中於連結與使用兩類，學生並未發生定位方面的錯誤。

在連結方面，學生對於重複概念的理解較為片面，在第五題首次取用 loop 重複執行特定動作時，大多數學生可以理解「重複執行 n 次」需用重複結構 loop 進行實作，僅有 4 名學生需要訪談者提示或告知；但在第七題中使用重複結構 loop 控制陣列上的物件依序執行相同的動作時，大部分（10 名）的學生並未意識到「不同人需要重覆執行相同的動作」同樣可以透過 loop 進行實作。根據此差異，研究者推測學生僅將 loop 視作「重複執行特定動作 n 次」的控制結構。

在使用方面，第五題中使用常數做為 loop 的重複次數時，僅有少數學生需要訪談者提示；但是在第六題使用變數做為重複次數時，所有學生皆使用儲存使用者輸入數值的變數做為迴圈次數。這個作法在使用者輸入數值為正數時，可使迴圈正常運作；但是當使用者輸入數值為負數時，會因為終止條件為負數使得迴圈內程式碼不會執行。所有的學生在解決此狀況時，都把焦點放在修改迴圈終止條件（如圖 4.23 框線處）中，並未考慮修改起始條件，或是迴圈的增量（increment）。

圖 4.23 loop 指令（複雜版本）

研究者推測學生未考慮其他修改方式的原因除了未完全理解重複結構 loop 中其他可修改項目的意義，另一個原因是因為 loop 在簡易版本中僅顯示重複次數（如圖 4.24 框線處），也就是複雜版本中的終止條件，使得學生直覺認定「迴圈終止條件」等於「重複次數」，而忽略了其他項目。

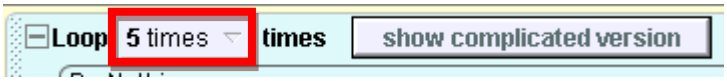


圖 4.24 loop 指令（簡易版本）

在第七題中，有 8 名學生無法順利取用迴圈索引 index 以控制陣列上物件進行動作。研究者推測學生並未完整了解 loop 的作用方式及迴圈索引 index 的功能。

7. 陣列

陣列中所發生的錯誤集中於定位與使用兩類，由於具有可以明顯觀察到的物件，因此學生並未發生連結方面的錯誤。

在定位錯誤方面，雖然與物件宣告的方式相同，但是學生卻無法順利的在宣告物件的資料夾中找到正確的類別進行宣告。而完成視覺化陣列宣告後，有 2 名學生誤以為僅需將場景中物件移動成「看起來像是在視覺化陣列」上即完成視覺化陣列中物件的設定。研究者認為此一情況的發生，是由於視覺化陣列在場景中的呈現方式（可參考圖 4.12）易於引起學生的誤會。

在使用上，學生發生的錯誤集中在取用陣列中元素作為動作物件時，陣列中物件的取用方式較為複雜，必須將視覺化陣列屬性庫中的元素進行拖曳後才能選

取，與學生慣用的自左上方物件區直接拖曳有所差異。另外有 4 名學生未進行陣列元素的選取，而是直接將視覺化陣列（ArrayVisualization）作為動作物件使用。研究者推測學生並未意識到視覺化陣列本身（即場景中站台，可參考圖 4.11）同樣是一個物件，而是認為視覺化陣列就是視覺化陣列上的物件。

8. 自訂方法

自訂方法中所發生的錯誤集中於定位與使用兩類。由於實作題目之呈現方式與前一題完全相同，因此訪談員於實作前即進行相關的說明，使得學生並未發生連結方面的錯誤。

在定位方面，學生將自定方法宣告與其他宣告混淆，其中有 2 名學生誤宣告為參數（parameter）；另外 1 名學生則是宣告為自訂函式（function）。

在使用上，學生在編寫自訂方法中程式碼時發生程式碼編寫缺漏以及未進行自定方法的呼叫兩種狀況，究其原因多為忽略或遺忘，僅需訪談員提示即能順利補齊。

9. 參數

參數中所發生的錯誤集中於定位與使用兩類。由於實作題目之呈現方式與前一題完全相同，因此訪談員於實作前即進行相關的說明，使得學生並未發生連結方面的錯誤。

在定位方面，有 8 名學生無法順利找到參數宣告按鈕進行宣告，其中有 2 名學生將參數宣告與變數宣告混淆；而在使用上，學生發生的錯誤主要為未將宣告完畢的參數使用於方法中導致程式碼有所缺漏。

第三節 綜合討論

在本次診斷性訪談測驗中，研究者根據觀察學生實作程式專案時所見的情形，提出以下幾項建議：

一、 程式概念

由於 Alice 的程式執行結果能以動畫形式呈現，故物件、物件內建方法及控制結構（if/else、loop、do together 及 do in order）皆可輕易化為程式執行動畫中可被學生觀察到的具體現象。對於這些可被具體觀察的項目，學生較能將其與對應的程式概念進行連結，訪談者通常僅需提示「需要執行ＯＯ動作」，學生便能發現自己所缺漏的部分。反之，諸如變數及參數這種屬於程式內部資料管理機制的程式概念，無法直接透過動畫具體呈現其功能，因此學生往往無法將這些概念與程式的執行過程進行連結。建議教師在講授此類概念時，除了強調其功能及使用方法外，另可搭配 print（列印）指令，將變數及參數值之變化情形列印出來，使其成為程式執行過程中可被觀察的對象。

大部分的學生對於程式概念的理解僅有單一面向，會認為「ＯＯ控制結構是專門用來執行ＸＸ動作」。以 loop 為例：許多學生認為重複結構 loop 只能針對特定物件動作重複執行，因此在控制陣列中不同的物件時，需要訪談者提示才知道可以使用 loop。建議教師在講授程式概念時，可搭配多樣化的例題進行講解，協助學生徹底理解該概念的應用範圍和使用時機。

二、 程式解題技巧

在實作程式專案時，學生往往在觀察完題目所需執行的流程及動作後，便直接開始進行解題，並未規劃程式流程或思考問題的解決方法，導致如身高縮放錯誤、無法順利設定判斷條件等邏輯錯誤。研究者推測此一狀況的發生是由於題目皆以動畫形式呈現，學生將動畫中的個別動作作為處理單元，認為依照動畫中物

件動作選取正確程式碼即可。建議教師應教導學生在面對任何程式問題時，宜先思考程式的整體流程、判斷條件、輸入－輸出之間的對應關係，如此將有助於避免許多邏輯錯誤的發生。

和傳統文字類的程式語言（如 Java 或 C++）相比， Alice 程式通常使用物件而非變數作為主要的操作對象，因此造成學生在面對問題時，較容易以尋找物件所包含之函式或方法的方式嘗試解題，而非將重要的資訊透過變數進行儲存及處理。建議教師特別注意協助學生建立變數概念。

三、 數學相關問題

學生在部分題目中所遭遇的困難並非單純因為程式設計技巧不足，而是由於學生無法正確處理題目中的數學計算或觀念。例如在第二題中，學生在得知方法 resize 的功能為縮放一定倍率後，仍有許多學生無法正確計算物件的比例。而在第四題使用選擇結構進行輸入數值正負的判斷時，亦有部分學生因為無法將「數字為正／負」的概念轉化為「大於／小於零」，而無法順利完成判斷條件的設定。在第六題中，大多數學生在未找到絕對值函式的情況下，也無法透過數學計算將「負數轉正」，導致迴圈無法順利執行。理論上一般高中生所具備的數學能力應能順利處理這些計算或設定，究其原因，應非學生真的欠缺相關的數學概念，而只是未曾想到必須將數學概念運用於程式解題中，因而一時無法將其以程式語法表現出來而已。教師應可於規劃 Alice 程式範例時，適當地加入一些運用簡單數學概念的題目，以幫助學生了解數學概念在程式中可能扮演的的角色。

四、 Alice 程式語言與程式設計介面的相關問題

研究者發現在實作 Alice 程式時，有數種情況是由於 Alice 程式語言本身設計導致學生較容易產生混淆的地方，分述於下。

1. **3D 物件建模**：Alice 中的 3D 物件在執行動作時，皆是以物件的中心點為動作基準點。但是各物件的中心點皆不盡相同，導致物件在執行動作時會因為中

心點位置的不同而產生動畫演出上的差異。以方法 reszie 為例，若物件中心點位於物件正中央時，放大會使其插入地面下，縮小時則會使其浮至半空中。這個情況除了增加觀察困難，也可能造成學生誤認自己的程式發生邏輯錯誤。教師除了在製作範例時應注意物件的選用之外，也應提醒學生在練習時，選取中心點位置相同的物件進行實作。

2. **物件屬性的修改**：一般物件的屬性區中包含該物件的顏色（color）、透明度（opacity）、外觀材質（skin texture）、物件位置（point of view）及裝填風格（filling style，物件呈現的樣子）等物件外觀方面的設定值。程式設計者若想在程式執行過程中變動這些外觀項目，必須將屬性區中的對應項目直接拖曳至程式中，而非透過物件內建的方法（method）。學生常會認為「物件的動作」皆是透過物件方法來實作，如第一題中使物件改變顏色，有不少學生在物件方法庫及函式庫中尋找未果，需由訪談者告知物件顏色位於物件的屬性區中方可正確拖曳。建議教師在教學時，需說明清楚物件屬性、方法及函式的意義及用途。
3. **視覺化陣列元素的使用**：在 Alice 程式語言中，雖然視覺化陣列中的元素多為物件，但卻無法直接取得個別元素的屬性、方法及函式。若需取用，必須先拖曳其他物件的屬性、方法及函式後，再使用陣列元素取代該物件。此操作並不直觀，往往為學生帶來很大的困擾。建議教師在讓學生實作視覺化陣列相關題目時，強調取用陣列元素屬性、方法及函式的步驟。
4. **自訂元件命名規範不夠嚴謹**：當使用者宣告變數、參數及自訂方法時，可使用數字、結構名稱（如 if、else）、甚至部分標點符號作為該元件的名稱。某些部分學生在編寫程式的過程中，因為將數字作為變數名稱使用於程式之中，導致錯誤的程式碼看起來語意通順而無法順利進行除錯，例如：圖 4.25 中 if 判斷式之判斷條件為「若使用者輸入數字大於變數 5」，圖 4.26 中判斷條件為「若使用者輸入數字大於常數 5」，在 Alice 顯示上兩者之間僅有微小的顏色差異，若不注意很容易造成誤解。除了建議 Alice 應適度添加命名規範外，也建

議教師在課程中強調良好命名習慣的重要性。

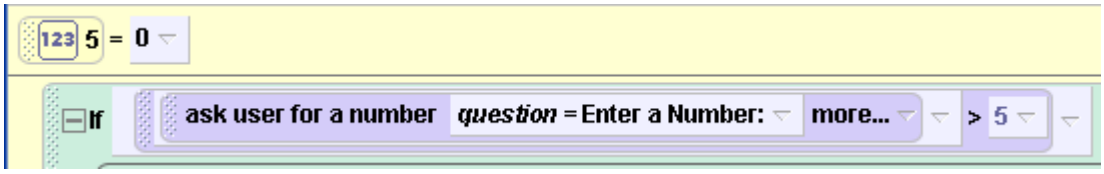


圖 4.25 變數名稱混淆範例（變數版本）

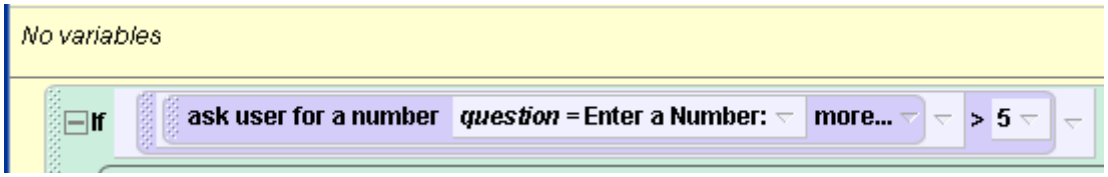


圖 4.26 變數名稱混淆範例（常數版本）

5. **視覺化陣列的宣告**：當使用者進行視覺化陣列的宣告時，必須點擊 new item 按鈕來增加陣列的長度。但是 Alice 僅提供增加長度的按鈕，並未提供減少長度的功能，因此若是使用者不小心多按了幾次按鈕使得陣列長度過長，他無法進行修改，必須刪除整個陣列重新宣告。建議 Alice 應盡速將此進行修正。

第五章　結論與建議

第一節 結論

本研究針對各個 Alice 程式設計概念專案的實作狀況進行審視，發現若是題目動畫可清楚呈現的概念（包含自訂方法、時序控制結構、選擇結構、重複結構及陣列），學生在編寫程式時較易聚焦於目標程式概念，且犯錯時也較能理解錯誤是出自程式碼中哪一個部分並進行除錯；相對的，學生在無法直接從題目動畫觀察到的程式概念（如變數、自訂方法及參數）中發生錯誤時，則容易陷入不知所措、胡亂嘗試的情形，但若能針對這些概念提供明確的功能說明及目的，則可有效避免這些情況的發生。本研究同時探討學生實作 Alice 程式專案時常見的程式錯誤狀況，並概分為以下四種：(1)連結錯誤：無法將解題想法對應至正確的程式概念或方法；(2)定位錯誤：無法順利的進行目標程式概念的取得；(3)使用錯誤：設定、操作上的錯誤；以及(4)其他錯誤：無程式概念無關的錯誤，如解題技巧或習慣。

建議程式設計教學者在使用 Alice 講授抽象且難以動畫化的概念時，可透過實際例子或情境模擬進行輔助，以降低概念的抽象程度。同時，教學者應給予學生多樣化的題目練習，以協助學生理解相關概念。此外，教學者應針對上述常見的錯誤情況修正教學方法或提醒同學注意，以期降低錯誤發生率，從而增進學生的學習成就與動機。

第二節 未來研究方向

由於本次實驗所使用的程式設計專案著重於探討學生在各基礎程式設計概念中所遭遇的困難，因此學生發生的錯誤僅止於 Linn & Dalbey（1985）所提到的程式認知連結中，對程式語言本身功能和語法的理解以及程式設計技巧的缺失。

建議未來的研究可以更進一步以較有經驗的學生為目標，要求他們實作較為複雜的程式概念，如排序或搜尋演算法題目，或需組合不同程式概念的應用題目，如：撲克牌洗發牌、老鼠走迷宮尋找出口等，藉此理解學生在模板（templates）轉移及程式流程規劃時所呈現的錯誤類型，以釐清學生在問題解決策略方面的學習困難。

参考文献

參考文獻

Adams, J.C. (2007). Alice, middle schoolers & the imaginary worlds camps. *ACM SIGCSE Bulletin. 39*(1), 307-311.

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin, 37*(3), 84-88.

Bishop-Clark, C. (1992). Protocol analysis of a novice programmer. *ACM SIGCSE Bulletin, 24*(3), 14-18.

Bonar, J., & Soloway, E. (1983). Uncovering principles of novice programming. *POPL '83 Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages,* 10-13.

Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies, 18*(6), 543-554.

Brown, P.H. (2008). Some field experience with Alice. *Journal of Computing Sciences in Colleges, 24*(2), 213-219.

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceedings of Ascilite, 2007*, 99-107.

Chen, C.-L., Cheng, S.-Y. & Lin, J. M.-C. (2012). A study of misconceptions and missing conceptions of novice Java programmers. *Proceedings of the 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering* (FECS'12, pp. 307-313)

Cooper, S., Dann, W., & Pausch, R. (2000a). Developing Algorithmic Thinking With Alice. In The Proceedings of the Information Systems Education Conference 2000, Philadelphia.

Cooper, S., Dann, W., & Pausch, R. (2000b). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching Objects-first In Introductory Computer Science. *ACM SIGCSE Bulletin, 35*(1), 191-195.

Dann, W., Cooper, S., & Pausch, R. (2000). Making the connection: programming with animated small world. *ACM SIGCSE Bulletin, 32*(3), 41-44.

Ebrahimi, A. (1994). Novice programmer errors: language constructs and plan composition. *International Journal of Human-Computer Studies, 41*(4), 457-480.

Fincher, S., Cooper, S., Kölling, M., & Maloney, J. (2010). Comparing Alice, Greenfoot & Scratch. *SIGCSE '10*, 192-193.

Good, J. (1999). VPLs and novice program comprehension: how do different languages compare? *Visual Languages, 1999*, 262-267.

Jenkins, T. (2002). On the difficulty of learning to program. *Proceedings of the 3rd LTSN-ICS Conference*, 53-58.

Kelleher, C., Cosgrove, D., Culyba, D., Forlines, C., Pratt, J., Pausch, R. (2002). ALICE2: programming without syntax errors. *User Interface Software and Technology*.

Ko, A.J. (2004). Designing a flexible and supportive direct-manipulation programming environment. *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, 277-278.

Kopec, D., Yarmish, G., & Cheung, P. (2007). A description and study of intermediate student programmer errors. *ACM SIGCSE Bulletin, 39*(2), 146-156.

Lahtinen, E., AlaMutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14-18.

Lane , H.C., & VanLehn, K. (2003). Coached program planning: Dialogue-based support for novice program design. *ACM SIGCSE Bulletin, 35*(1), 148-152.

Linn, M. C., & Dalbey, J. (1985). Cognitive consequences of programming instruction: Instruction, Access, and Ability. *Educational Psychologist, 20*(4), 191-206.

Mannila, L., Peltomaki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education, 16*(3), 211-227.

McIver, L. (2000). The effect of programming language on error rates of novice programmers. *12th Annual Workshop of Psychology of Programmers Interest Group (PPIG),* 181-192.

Mullins, P., Whitfield, D., Conlon, M. (2009). Using Alice 2.0 as a first language. *Journal of Computing Sciences in Colleges, 24*(3), 136-143.

Pea, R.D. (1986). Language-independent conceptual "bugs" in novice programming. Journal of. *Educational Computing Research 2*(1): 25–36.

Perkins, D.N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. *In Soloway E. and Iyengar S. (Eds.). Empirical Studies of Programmers. Norwood: Ablex Pub.*, 213–229.

Powers, K., Ecott, S., & Hieshfield, L.M. (2007). Through the Looking Glass: Teaching CS0 with Alice. *ACM SIGCSE Bulletin, 39*(1), 213-217.

Putnam, R. T., Sleeman, D., Baxter, J. A., & Kuspa, L. K. (1989). A summary of misconceptions of high school BASIC programmers. In E .Soloway and J. C. Spohrer (eds.), *Studying the Novice Programmers* (pp. 301-314). Hillsdale, NJ, Lawrence Erlbaum Associates.

Ragonis, N., & Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education, 15*(3), 203-221.

Samurç ay, R. (1985). Learning programming: An analysis of looping strategies used by beginning students. *For the Learning of Mathematics 5*(1), 37–43.

Soloway, E. (1986). Learning to program = Learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850-858.

Spohrer, J. G., & Soloway, E. (1986b). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM, 29*(7), 624-632.

Spohrer, J.C., & Soloway, E. (1986a). Analyzing the high frequency bugs in novice programs. *In Soloway E. and Iyengar S. (Eds.). Empirical Studies of Programmers. Norwood: Ablex Pub.*, 230–251.

Spohrer, J.C., Soloway, E., & Pope, E. (1985). A goal/plan analysis of buggy pascal programs. *Journal Human-Computer Interaction, 1*(2), 163-207.

Sykes, E.R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research, 36*(2), 223-244.

Vainio, V., & Sajaniemi, J. (2007). Factors in novice programmers' poor tracing skills. *ACM SIGCSE Bulletin - Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE'07), 39*(3), 236-240.

Wang, T-H., Mei, W-H., Lin, S-L., Chiu, S-K., & Lin, J.M-C. (2009). Teaching programming concepts to high school students with Alice. *Frontiers in Education Conference*, 1-6.

Ward, B. (2009). Using virtual world programming languages to teach computer science concepts.

Wiedenbeck, S. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. *International Journal of Human-Computer Studies, 51*(1), 71-87.

Yarmish, G., & Kopec, D. (2007). Revisiting Novice Programmer Errors. *ACM SIGCSE Bulletin, 39*(2), 131-137.

Zaccone, R., Cooper, S., & Dann, W. (2003). Using 3D Animation Programming in a Core Engineering Course Seminar. *Frontiers in Education, 2*, 14-17.

教育部（民 100），**普通高級中學必修科目「資訊科技概論」課程綱要**。臺北市：作者。

謝如山（譯）（2004）。**進入兒童心中的世界**（原作者：Ginsburh, H, P.）。臺北市：五南圖書。（原著出版年：1997）

林恬忻（2007）。**於國中實施 Alice 程式設計教學行動研究**。臺灣師範大學資訊教育學系碩士班學位論文，未出版，台北市。