壹、課程說明

單元名稱	程式設計專題：Alice 實作
單元摘要	Alice 程式設計、物件導向程式設計、3D 動畫程式設計。
設計者	王鼎中(臺北市立建國中學)
學習目標	1. 了解 Alice 程式開發環境。 2. 認識物件導向程式設計的基本觀念。 3. 瞭解變數、運算式及內建函數等程式基本組成要件。 4. 熟習循序、選擇及重複等三種程式設計的基本結構 5. 透過作品展示與分享，展現同學們的創意。
課綱範圍	5.電腦與問題解決➡5.4電腦解題實作➡5.4.1電腦解題工具介紹 5.4.2解題實作
教學節數	8 節（400 分鐘）
先備知識	視窗環境操作、網路資料瀏覽與下載、基本邏輯概念。
評量方法	1. 簡單口頭問答 2. 課堂觀察 3. 作業練習與實作：可於課程進行中，請同學以教材中的範例進行練習，並嘗試完成牛刀小試所提供的例題，以檢視同學對課程內容的吸收程度。 4. 作品觀摩： (1)於每單元課程結束前，請同學以牛刀小試的練習結果進行展示與說明 (2)於課程結束時，安排專題製作的時間並請同學展示專題製作的成果
參考資源	1. 自編教材 2. Wanda Dann, Steve Cooper and Randy Pausch, Learning to Program with Alice, Prentice Hall, 2006 3. Alice 官方網站 http://www.alice.org

貳、教學活動計畫（敘述活動細節、教學工具、教學策略等）

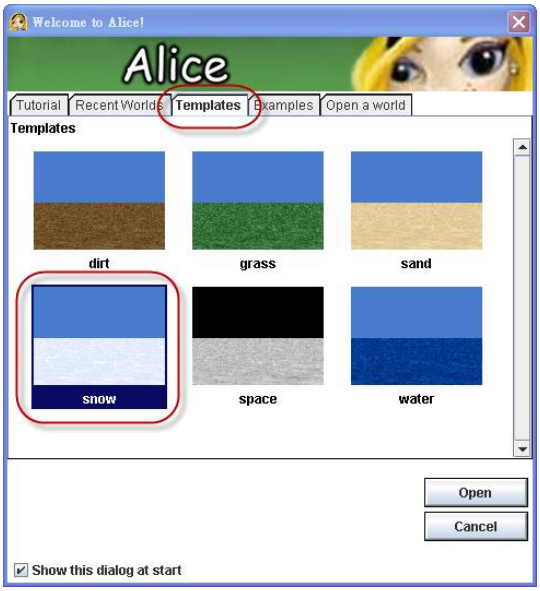
<table>
<tr><th>教學活動</th><th>時間</th><th>說　明</th></tr>
<tr><td>一、Alice 簡介</td><td>1 節</td><td>
1-1 什麼是程式設計（programming）
　　說明程式語言的用途與目的，並介紹機器語言、組合語言與高階語言等不同類型的程式語言。

1-2 什麼是 Alice
　　介紹 Alice 程式開發環境的發展歷程，並介紹 Alice 程式開發環境的各項特色。

（展示）展示溜冰者在冰上滑冰的程式，初步瞭解以 Alice 撰寫程式的成果，以引起同學的學習動機。

1-3 如何取得 Alice

1. 在 Alice 官方網站（http://www.alice.org）的 downloads 下，找到 Alice2.2 版
2. 將下載的檔案解壓縮，建議將檔案解壓縮到 C:\ 或 D:\
3. 開啟 alice2.2 資料夾，執行 Alice.exe
4. 當出現 Alice 的歡迎畫面（Welcome to Alice!），即可開始使用

1-4 Alice 操作介面介紹
　　1. 在歡迎畫面的對話方塊中，選取樣版（Template）頁面中的 grass Template


</td></tr>
</table>

2. 如此便可建立一個由藍天和雪地組成的初始場景，並開啟 Alice 的程式編寫環境

Alice 的程式編寫環境除上方的功能表列外，約可分為五個區塊，分別說明如下：

1. 物件區

位於操作介面的左上方部分，以樹狀結構的方式顯示虛擬世界中的所有物件，各個物件尚可展開顯示其子物件

2. 物件資訊區

位於操作介面的左下方部分，視窗中顯示所選取物件的各項資訊，共分為屬性頁面（properties tab）、方法頁面（method tab）以及函數頁面（function tab），分別顯示所選取物件的相關屬性、可使用的方法與函數等。

3. 場景區

位於操作介面的中間上方部分，主要作為程式設計時配置場景之用，所顯示的畫面即為 Alice 程式的起始場景，當程式開始執行時，便是由此場景開始呈現，在此視窗中，設計者可以規劃場景、加入物件、調整攝影機以獲取最佳的呈現畫面等。

4. 事件區

位於操作介面的右上方部分，使用者可在此設定當某個事件發生時（如：當滑鼠按了某個物件、當鍵盤按下了某個鍵），所要回應執行的方法（method）。

5. 程式編輯區

位於操作介面的右下方部分，使用者可透過拖曳的方式，將程式相關的結構、變

數及敘述等拖曳進來，組成所需的程式。

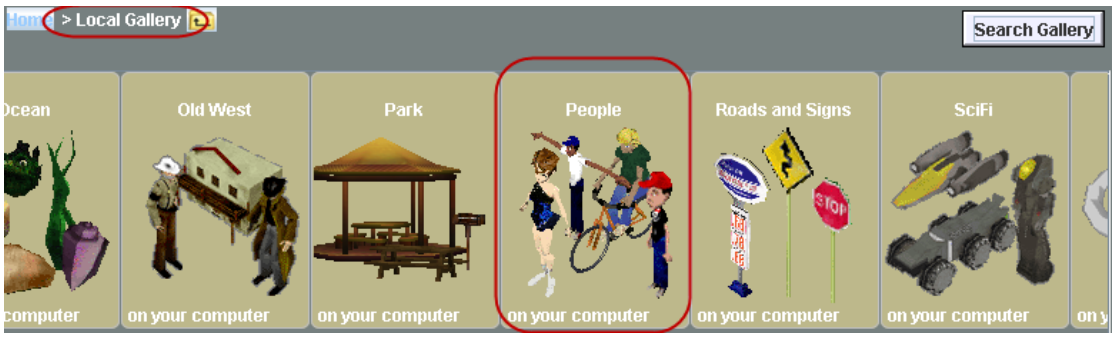
1-5　Alice 的 3D 虛擬世界

介紹 Alice 3D 虛擬世界 (virtual world)的概念，並介紹三度空間中各個座標軸的方向，以及 Alice3D 虛擬世界所提供的六個預設場景：雪地(snow)、泥地(dirt)、草地（grass）、沙地（sand）、太空（space）及水中世界（water）。

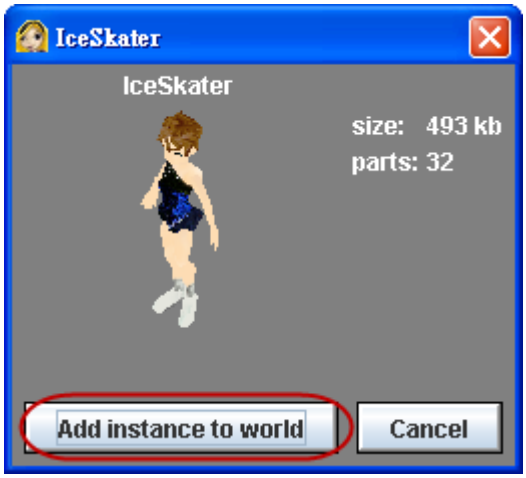
1-6 物件與虛擬世界的關係

Alice 提供了許多預先建置好的 3D 物件，可點選場景區右下方的

按鈕來加入物件。



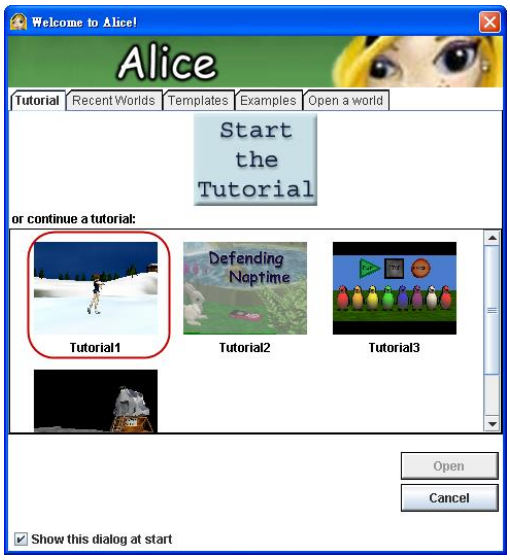
點選其中的溜冰者（iceskater），並按下 Add instance to world 即可完成加入物件的動作。

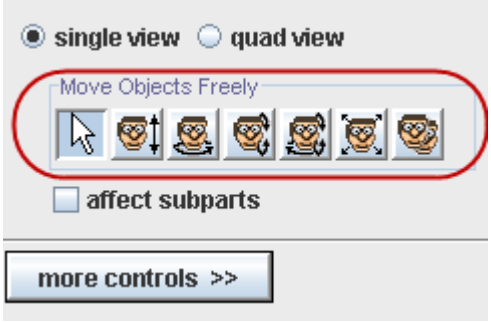
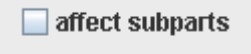


Alice 的物件都是一個 3D 物件，具有高度、寬度及深度三個維度，及前、後、左、右、上、下六個方向。

此外，每個物件都有一個中心點，也有一個相對於虛擬世界中心的相對位置。

牛刀小試：

		為使雪地溜冰的場景更加豐富有趣，請在雪地中加入一個湖景的布景物件，該物件存放在 local gallery → environments → lake。 1-7 調整攝影機獲取所需的場景 透過場景區下方的運鏡按鈕，調整攝影機角度與位置，以獲取最佳的場景視覺效果。 1-8 以 tutorial 來熟習 Alice 開發環境的操作 Alice 共提供了四個導覽範例，請以 tutorial 1 來熟習 Alice 的開發環境，在 tutorial 1 中共有 40 個步驟，請依照說明依序完成所有步驟。 
二、物件導向程式設計簡介	1節	2-1 什麼是物件導向程式設計 介紹什麼是物件導向程式設計（object oriented programming），物件導向程式設計的基本概念，並比較物件導向程式設計與傳統程式設計的差異。 2-2 Alice 的物件 介紹 Alice 開發環境所提供的各種物件、物件藍本與物件實體的關係，並藉由操作介紹物件係由許多子物件所組成的概念。 2-2-1 加入所需物件至虛擬世界中 在接下來的程式中我們會用到溜冰者（People→IceSkater）、雪地湖景佈景物件（Environments→Lake）、三角錐（Shapes→Cone）、圓形（Shapes→Circle）等物件，請分別將這些物件加入到虛擬世界中。 2-2-2 利用物件操作面版安置物件

		將物件加入虛擬世界後，我們可以透過滑鼠拖曳的方式，將物件擺放到期望的位置上，此外，亦可透過右上方物件控制區的協助來調整物件的位置、大小及方向，並可進行物件的複製。   若要對子物件進行調整，則需勾選 affect subparts 選項 控制物件自由移動 控制物件上下移動 控制物件水平旋轉 控制物件垂直旋轉 控制物件垂直及水平旋轉 控制物件放大與縮小 複製物件：點選按鈕後，再點選要複製的物件，即會複製出另一物件。 牛刀小試： 1. 請複製出另外兩個三角錐（共三個三角錐） 2. 請如下圖所示，調整物件的擺放位置

註：若要對物件作更精準的調整，則可透過三視圖的模式來進行，只要點選物件控制區右上方的 quad view 按鈕即可進入。

2-3 Alice 物件的屬性（property）

藉由改變三角錐的顏色、透明度、承載者及是否顯示等屬性，來介紹物件屬性的概念，以及更改物件屬性的方法，並藉由操作的過程，觀察並瞭解物件屬性的改變與物件外觀及特性的關係。

2-3-1 顏色（color）

在左上角的物件區中點選 cone，先按滑鼠右鍵選 rename 先將該物件更名為 redcone，接著在左下方的物件資訊區點選 properties 即會出現 redcone 物件的所有屬性，點選 color 屬性的右方向下三角形並選擇 red 即會發現畫面中的三角錐以變成紅色的了，請以前述方法分別將 cone2 更改為 bluecone，並將顏色設定為 blue，將 cone3 更改為 yellowcone，並將顏色設定為 yellow。

牛刀小試：

請嘗試將畫面中的圓形物件 circle，更名為 hole 並將顏色設定為 black。

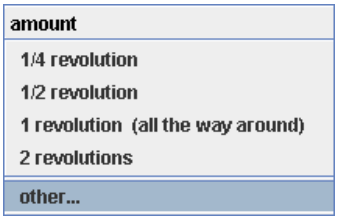
2-3-2 透明度（opacity）

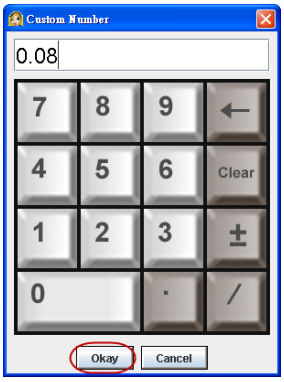
剛剛設定完成的 hole 顏色太深，若能將顏色調淡些，將能與背景更為融合，此時可透過調整透明度（opacity）屬性來達成，點選 opacity 屬性的右方向下三角形並選擇 0.4(40%)即可。Opacity 屬性在程式執行時相當好用，若我們在程式執行時，以程式指令改變 Opacity 屬性為 0，並將該指令的執行時間（duration）設定為 5，便可達到讓物件逐漸消失的效果。

2-3-3 承載者（vehicle）

承載者屬性在製作動畫程式時相當重要，透過這個屬性的設定可以將兩個物件的相對位置關係結合起來，舉例來說：若我們要讓溜冰者伸手將三角錐舉起，通常的作

		法是在溜冰者伸手後，同時讓手及三角錐往相同的方向移動，模擬出舉起東西的動作，但透過 vehicle 屬性，我們可以在溜冰者伸手後，將三角錐的 vehicle 屬性內容設定為溜冰者的手，如此一來三角錐的承載者變成是溜冰者的手，只要溜冰者的手移動，三角錐就會跟著進行相對位置的移動，真正的達到舉起東西的動作。 2-3-4 是否顯示（isShowing） isShowing 屬性用於決定物件是否顯示於場景畫面中，當屬性值設定為 false 則不顯示，設定為 true 則為顯示，當我們希望某個物件在起始場景中隱藏，直到程式執行的過程中才顯示時，我們可以在起始場景中，將物件的 isShowing 屬性設定為 false，並在程式執行時，利用程式指令改變 isShowing 的屬性值為 true，如此便可達到所要的效果。 2-4 Alice 物件的方法（method） 先前我們可以透過改變屬性的方式，來改變物件的特徵與外觀，但若要讓物件做出某些動作或改變姿態，則要透過物件的方法來達成，Alice 對每一個物件都提供了一些基本的方法，如：移動（move）、旋轉（turn）、側轉（roll）、改變大小（resize）等，以下我們以溜冰者物件來介紹物件的方法。 2-4-1 移動（move） 假若我們要讓溜冰者向前移動一公尺，則我們可以在左上角物件區中的 iceSkater 物件上，按滑鼠右鍵開啟快顯功能表，接著點選 methods → move → forward → 1 meter，如此便可以讓溜冰者向前移動一公尺。 2-4-2 旋轉（turn） 接著我們可以用 turn 方法讓溜冰者轉身，在快顯功能表中，點選 methods → turn → right → 1/4 revolution，即可讓溜冰者向右轉身 1/4 圈，以上介紹的這些方法除了在物件上可用，在物件的子物件同樣亦具有這些方法可使用，如果我們要讓溜冰者轉頭，此時便要點選物件區中 iceSkater 前的＋號，將 iceSkater 的子物件展開，一層一層展開後，找到 head 子物件後，在其上按滑鼠右鍵開啟快顯功能表，點選 methods → turn → right → 1/4 revolution，即可讓溜冰者向右轉頭 1/4 圈。 2-4-3 側轉（roll） 若我們要讓溜冰者的頭向左傾 0.08 圈，則可以在 head 子物件上，按下滑鼠右鍵開啟快顯功能表，點選 methods → roll → left → 0.08 revolution，由於選單中並無 0.08 revolution 個選項，可點選 other 選項自行設定數值，在數字鍵盤中打入 0.08 後按 okey 即可。





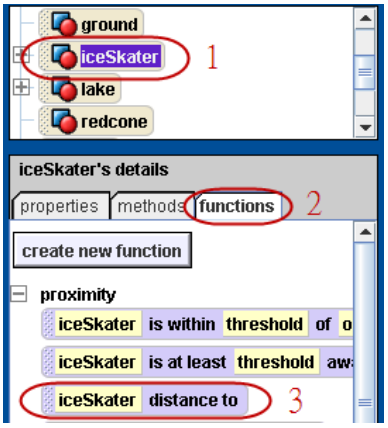
2-4-4 改變大小（resize）

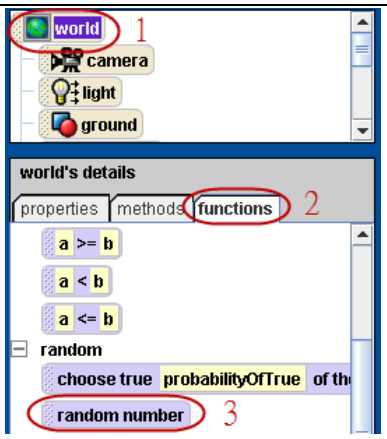
接下來我們來幫畫面作點卡通效果，如果我們可以讓溜冰者的身體保持原狀，但讓頭部放大三倍，應該會相當有卡通的效果，我們同樣在 head 子物件上，按下滑鼠右鍵開啟快顯功能表，點選 methods → resize 並在前述介紹的數字鍵盤中設定為 3 倍即可。

除前述介紹的四個常用方法外，一般物件通常還具備有以下方法，包含：說話（say）、思考（think）、播放聲音（play sound）、移動到某物件所在地（move to）、朝某物件移動（move toward）、遠離某物件（move awayfrom）、將物件的方向設定與某物件相同（orient to）、面向某物件（turn to face）、指向某物件（point at）、設定物件的姿態（set pose）等，請嘗試設定並觀察其結果，以便後續程式設計時使用。

2-5 Alice 的內建函數

介紹內建函數的概念與意義，並說明物件型內建函數與通用型內建函數的差異與使用方法。

<table>
<tr>
<td></td>
<td></td>
<td>


物件型函數


通用型函數

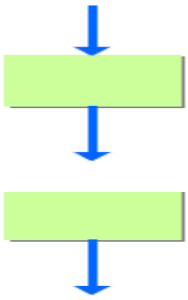
由於 Alice 的內建函數只能在程式執行階段使用，目前我們尚處於程式設計階段，因此，在此只進行概念的介紹，待後續程式設計時再進行操作說明與練習。

2-6 Alice 的事件
介紹事件觸發程式設計的概念，以及 Alice 事件的設定方法，嘗試讓使用者在按下→鍵時，即可看到溜冰者向右轉動了一圈，若再按一次，則會再轉一圈，。

牛刀小試：
請以前述方式為程式設定一個新的觸發事件，使得當使用者按下鍵盤←鍵時，溜冰者會向左轉一圈。
</td>
</tr>
<tr>
<td>三、循序結構</td>
<td>1節</td>
<td>
3-1 如何撰寫一個程式
介紹編寫一個程式的基本流程，包括：構思、腳本、編寫程式碼、執行以及修正的歷程。

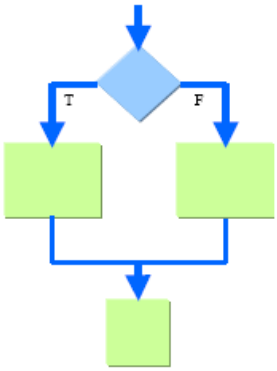
（展示）展示溜冰者在溜冰場上進行滑冰表演，結束時做出 ending pose 的程式，使同學瞭解本單元的學習目標，並引起同學的學習動機。

3-2 程式的基本結構
（1）循序結構(Sequence)
程式由上而下，第一行指令敘述執行完後接著執行第二行，如此依序一行一行的執行。
</td>
</tr>
</table>



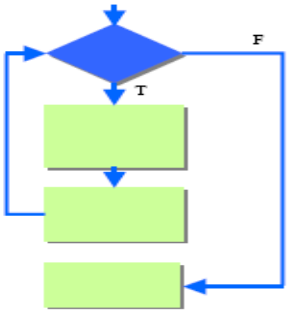
(2) 選擇結構(Selection)

程式流程進入判斷的菱形符號後，會判斷測試條件是否成立。然後，依據判斷的結果選擇程式的流向。



(3) 重複結構(Iteration)

搭配條件判斷的機制，部分程式片段可重複執行多次，直到某測試條件發生為止，程式重複執行部分即構成迴圈。



3-3 以循序結構完成你的第一個 Alice 程式

初始場景：

在溜冰場中有一位溜冰者，以及一個紅色的三角錐

動作腳本：

溜冰者向後抬起左腳

溜冰者向前滑動 2 公尺

溜冰者以自己為圓心向右旋轉 1 又 1/4 圈

<table>
<tr>
<td></td>
<td></td>
<td>
溜冰者擺出上身前傾的滑冰姿勢
溜冰者以紅色三角錐為圓心向左滑冰一圈
溜冰者擺出表演結束的姿勢 ending pose
攝影機以溜冰者為圓心向右旋轉一圈

參考程式碼：
<pre>
world.my first method ( )
    No variables
    iceSkater.leftLeg turn forward 0.25 revolutions
    iceSkater move forward 2 meters
    iceSkater turn right 1.25 revolutions
    iceSkater set pose iceSkater.pose4
    iceSkater turn left 1 revolution asSeenBy = cone
    iceSkater set pose iceSkater.pose2
    camera turn right 1 revolution asSeenBy = iceSkater
</pre>
3-4　Alice 特有的 do together 結構
1. 將程式編輯區下方的 do together 方塊，拖曳至第一行程式之前
2. 分別將第一行及第二行敘述拖曳進 do together 結構中
3. 完成程式修改並進行測試

3-5　儲存與分享
介紹透過 save world 將前述完成的程式儲存起來，並可透過 Export Video（匯出影片）將程式結果匯出成動畫影片與他人分享。
</td>
</tr>
<tr>
<td>四、內建函數、變數及算術運算式</td>
<td>1節</td>
<td>
程式構想：
假定我們希望不論紅色三角錐放置在何處，溜冰者都能溜到紅色三角錐前，繞行三角錐後並溜行回到原點。

（展示）展示溜冰者溜到紅色三角錐前，繞行三角錐後並溜行回到原點的程式，使同學瞭解本單元的學習目標，並引起同學的學習動機。

初始場景：
在溜冰場中有一位溜冰者，以及一個紅色的三角錐
</td>
</tr>
</table>

動作腳本：

溜冰者轉向紅色三角錐

溜冰者向後抬起左腳

溜冰者向前滑動（溜冰者與三角錐的距離）

溜冰者向右轉 1/4 圈(90 度)

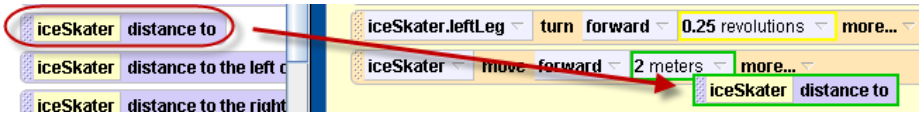
溜冰者繞行三角錐一圈

溜冰者向右轉 1/4 圈(90 度)，朝向起始點

溜冰者向前滑動（溜冰者與三角錐的距離）

4-1 內建函數

1. 溜冰者轉向紅色三角錐
2. 溜冰者向後抬起左腳
3. 溜冰者向前滑動（溜冰者與三角錐的距離）
   (1) 讓溜冰者向前滑動 2 公尺
   (2) 以（溜冰者與三角錐的距離）取代原先的 2 公尺
   透過 Alice 提供的內建函數 distance to 來取得（溜冰者與三角錐的距離）。




```
iceSkater move forward ( iceSkater distance to cone )
```

4-2 算術運算式

為避免讓溜冰者直接踩在三角錐上的狀況，可藉由算術運算式的協助調整出適當的距離。

1. 點選 distance to 右方的向下三角形，選取 math 進行算術運算的設定，並選擇 － 1。
2. 完成將溜冰者與三角錐距離減 1 公尺的運算

```
iceSkater move forward ( ( ( iceSkater distance to cone ) - 1 ) )
```

4-3 變數

接著，先以上個單元介紹的方法完成接下來的腳本動作

1. 溜冰者向右轉 1/4 圈(90 度)
2. 溜冰者繞行三角錐一圈
3. 溜冰者向右轉 1/4 圈(90 度)，朝向起始點

腳本中的最後一個動作是要溜冰者向前滑動（溜冰者與三角錐的距離），此時，我們很直覺的想到用先前的方法，利用 distance to 函數來完成，因此，我們完成了如下的程式

```
iceSkater move forward（（（ iceSkater distance to cone ）- 1 ））
```

但程式執行結果並不如我們的預期，溜冰者指向前移動了約 1 公尺，這究竟是什麼原因呢？你想到了嗎？對了，因為溜冰者的位置已經改變，所以沒有辦法以 distance to 函數取得原先的數值。

要如何解決以上的問題呢？如果我們能在溜冰者移動前，就將溜冰者與三角錐的距離，也就是溜冰者即將移動的距離保存下來，最後再以此保存下來的距離，來讓溜冰者回到原點即可，此時，程式設計中的變數正可派上用場。

4-3-1 什麼是變數

介紹什麼是變數、變數與主記憶體間的關係，以及變數的特性。

4-3-2 變數的宣告

1. 點選程式設計區右上角的 create new variable ，新增一個變數
2. 設定變數名稱為 move_dist、資料型態為 Number 以及變數初始值為 1
3. 完成 move_dist 變數的宣告。

4-3-3 將數值指定給變數

1. 嘗試將數值 2 指定給變數 move_dist
   (1) 將 move_dist 變數拖曳至溜冰者 move forward 指令敘述之前
   (2) 將 move_dist 變數的內容設定為 2（set value to）
2. 將內建函數 distance to 所產生的數值，指定給變數 move_dist
   (1) 將 distance to 函數拖曳至 move_dist 變數後的 2 上
   (2) 將 move_dist 變數的內容設定 distance to 的函數運算結果

```
move_dist set value to（ iceSkater distance to cone ）
```

4-3-4 使用變數進行運算

1. 拖曳變數 move_dist，分別取代溜冰者 move forward 程式敘述中的 distance to 函數
2. 如此即完成所需的程式

```
world.my first method（ ）
   move_dist = 1
      iceSkater point at cone
```

```
iceSkater.leftLeg turn forward 0.25 revolutions
move_dist set value to ( iceSkater distance to cone )
iceSkater move forward ( ( move_dist - 1 ) )
iceSkater turn right 0.25 revolutions
iceSkater turn left 1 revolution  asSeenBy = cone
iceSkater turn right 0.25 revolutions
iceSkater move forward ( ( move_dist - 1 ) )
```

3. 請任意移動紅色三角錐後，觀察程式是否能正確完成所希望的動作

五、選擇結構 | **2節**

5-1 什麼是選擇結構

介紹什麼是選擇結構，並舉例說明選擇結構的用途及其對程式運作的影響。

（展示）展示溜冰者溜到冰湖破洞上並掉進洞裡的程式，使同學瞭解本單元的學習目標，並引起同學的學習動機。

5-2 Alice 選擇結構語法

IF 條件判斷式

條件式為真(True)時的程式敘述區塊

Else

條件式為假(False)時的程式敘述區塊

5-3 條件判斷式

5-3-1 關係運算式

關係運算式的語法結構如下：

A 運算元　關係運算子　B 運算元

例如：　小明的身高　＞　172 公分

關係運算子包含有下列類型，分別說明如下：

關係運算子	說明
= =	左邊的值***等於***右邊的值
! =	左邊的值***不等於***右邊的值
＞	左邊的值***大於***右邊的值
＞ =	左邊的值***大於或等於***右邊的值

<	左邊的值***小於***右邊的值
<=	左邊的值***小於或等於***右邊的值

實作練習：

以下我們就以溜冰者是不是會掉進冰湖面破洞的例子，來介紹選擇結構以及條件判斷式。

初始場景：

在溜冰場中有一位溜冰者，以及一個黑色的冰湖面破洞

（註：冰湖面破洞在此是以一個圓形(Shapes→Circle)來替代，請在 Circle 物件的屬性（properies）中將 color 設定為黑色 black）

動作腳本：

溜冰者轉向冰湖面破洞

溜冰者向前移動 3 meters

「如果」(溜冰者與冰湖面破洞的距離) < 1 meter

溜冰者向下移動 10 meters

溜冰者發出掉到水中的聲音

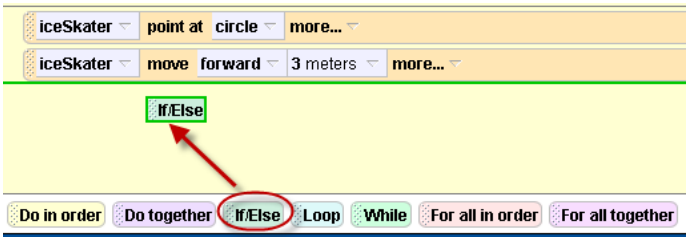
「否則」

溜冰者擺出表演結束的姿勢（set pose to pose2）

程式編寫：

1. 溜冰者轉向冰湖面破洞
2. 溜冰者向前移動 3 meters
3. 完成選擇結構的基本結構設定

(1)拖曳 If /Else 結構



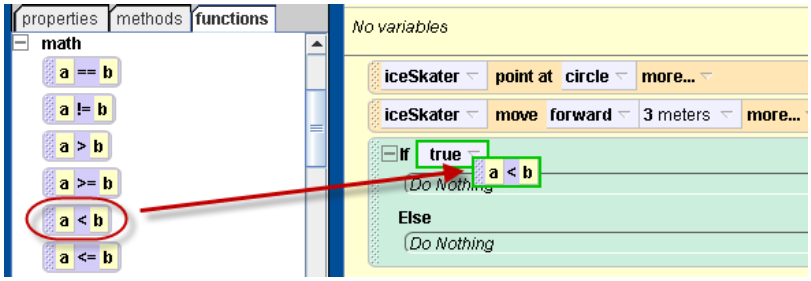
(2)設定條件判斷式：(溜冰者與冰湖面破洞的距離) < 1 meter

a. 拖曳（關係運算式<）

(a)點選物件視窗中的 world

(b)點選物件資訊區的 functions

(c)將 math 類別中的 (a<b) 拖曳至程式編輯區中，並取代 If 後的 true



(d)暫時設定左方運算元為 2，右方運算元為 1

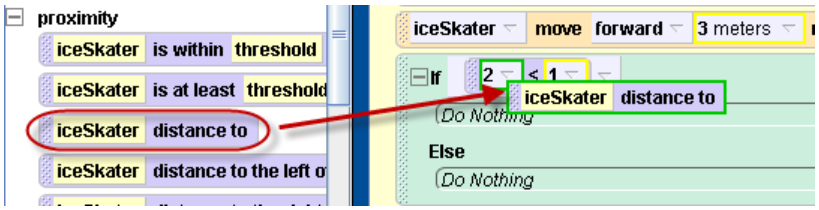
(e)完成（關係運算式<）的設定

b. 設定左方運算元（溜冰者與冰湖面破洞的距離）

(a)點選物件區中的 iceSkater 物件

(b)點選物件資訊區的 functions

(c)將 proximity 類別中的 distance to 拖曳至程式編輯區中，並取代 If 後的 2



(d)設定 distance to 的目標為 circle，完成設定

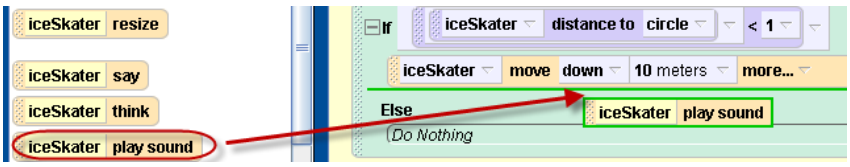
c. 設定右方運算元（1 meter）

此例中不需重新設定，若要重新設定，設定方式與左方運算元相同。

(3)設定條件判斷式為真(True)時的程式區塊

a. 溜冰者向下移動 10 meters

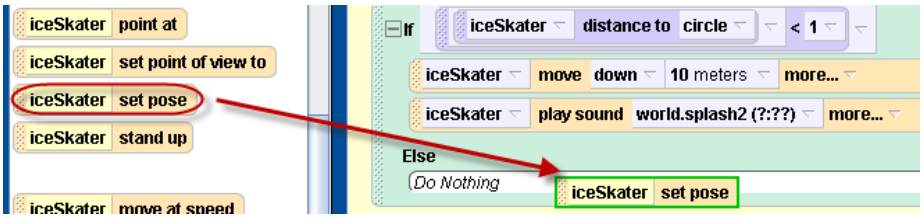
b. 溜冰者發出掉到水中的聲音，並選擇 splash 聲音



(4)設定條件判斷式為假(False)時的程式區塊

動作：溜冰者擺出表演結束的姿勢（set pose to pose2）

a. 拖曳 set pose 至 else 下方的 Do Nothing 中



b. 設定 pose 為 pose2

(5)完成的程式如下

```
world.my first method ( )
    No variables
    iceSkater point at circle
    iceSkater move forward 3 meters
    If ( ( iceSkater distance to circle ) < 1 )
        iceSkater move down 10 meters
        iceSkater play sound world.splash2 (?:??)
    Else
        iceSkater set pose iceSkater.pose2
```

4. 程式測試

5-3-2 邏輯運算式

邏輯運算式的語法結構如下：

A 條件式　邏輯運算子　B 條件式

例如：　小明的身高＞172 公分　AND　小明的體重＜50 公斤

邏輯運算子包含有下列類型，分別說明如下：

		邏輯運算子		
條件式 A	條件式 B	A And B	A Or B	Not A
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

實作練習：

接續上一節的例子，若將冰湖面破洞的顏色也列入考慮的條件，則判斷式中就有兩個需考量的條件，因此需要運用邏輯運算式來達成。

動作腳本：

溜冰者轉向冰湖面破洞

溜冰者向前移動 3 meters

「如果」(溜冰者與冰湖面破洞的距離) < 1 meter 而且 冰湖面破洞是黑色

溜冰者向下移動 10 meters

溜冰者發出掉到水中的聲音

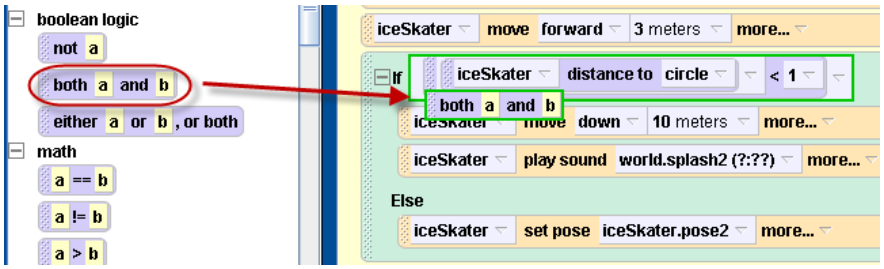
「否則」

溜冰者擺出表演結束的姿勢 (set pose to pose2)

程式編寫：

這個例子與上一節例子的差異僅在於判斷式的內容，因此，請先載入上一節的程式，以下僅介紹邏輯判斷式的設定方式。

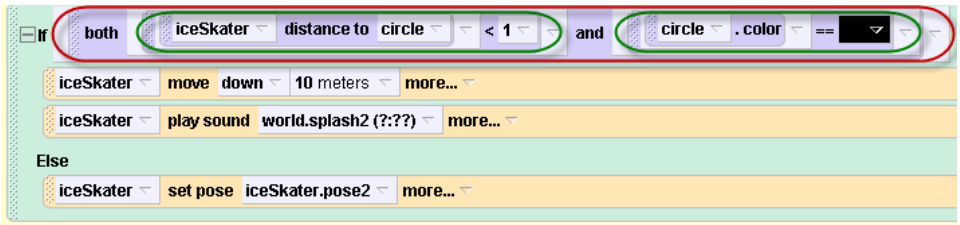
1. 拖曳 (邏輯運算式 AND)，附加在原判斷式之上



2. 在原條件式之外增加一個邏輯運算式，並將 b 條件式設定為 true

3. 拖曳 circle 物件的 color 屬性，取代 b 條件式(true)

4. 選擇 color 為黑色(color = = black)



5. 程式測試

完成的程式碼如下：

```
world.my first method ( )
```

```
No variables
  iceSkater point at circle
  iceSkater move forward 3 meters
  If ( both ( ( iceSkater distance to circle ) < 1 ) and ( circle . color == [redacted] ) )
      iceSkater move down 10 meters
      iceSkater play sound world.splash2 (?:??)
  Else
      iceSkater set pose iceSkater.pose2
```

5-4 巢狀選擇結構

介紹什麼是巢狀選擇結構，巢狀選擇結構的使用時機以及

5-4-1 巢狀選擇結構的語法

IF 條件判斷式 A

IF 條件判斷式 B

條件式 A 為真(True)且條件式 B 為真(True)時的程式敘述區塊

Else

條件式 A 為真(True)且條件式 B 為假(False)時的程式敘述區塊

Else

IF 條件判斷式 B

條件式 A 為假(False)且條件式 B 為真(True)時的程式敘述區塊

Else

條件式 A 為假(False)且條件式 B 為假(False)時的程式敘述區塊

5-4-2 Alice 巢狀選擇結構的實作

以終極密碼遊戲的部分程式（只能猜一次）來介紹與練習巢狀選擇結構。

動作腳本：

存入一個數字作為密碼

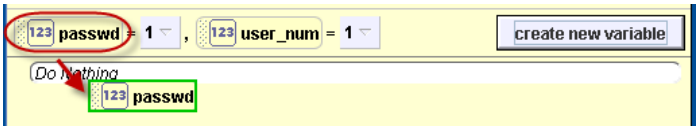
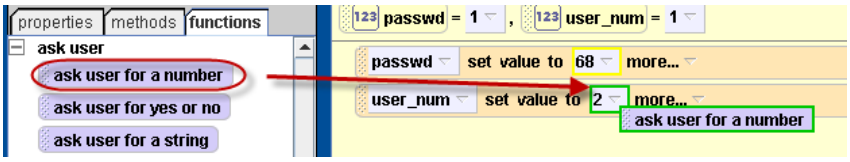
要求使用者輸入一個數字

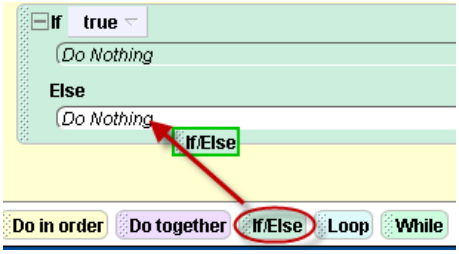
「如果」(使用者猜的數字）等於（密碼）

印出（猜中了 Bingo）

「否則」

「如果」(使用者猜的數字）大於（密碼）

		印出（太大了） 「否則」 印出（太小了） 初始場景： 請任意選擇一個場景。 程式編寫： 1. 宣告兩個變數並將初始值設定為 1，一為 passwd（數值型態 number）用以儲存密碼，另一為 user_num（數值型態 number）用以儲存使用者猜的數字。 2. 將一個數字(68)存入 passwd 中作為密碼 (1) 將 passwd 變數拖曳至程式區塊中  (2) 利用數字輸入鍵盤視窗，將 68 輸入 3. 要求使用者輸入一個數字（使用 world 物件中的 ask user for a number 函數） (1)將 user_num 變數拖曳至程式區塊中的第二行位置，並設定值為 2 (2)點選物件區的 world 物件 (3)點選物件資訊區中的 functions (4)a. 將 ask user 類別中的 ask user for a number 拖曳至程式編輯區中，並取代 user_num set value to 後的 2  b. 選擇問題提示語為（Enter a Number:） 4. 建立巢狀選擇結構 a. 建立外層選擇結構 將 If / Else 方塊拖曳至程式區塊中的第三行位置 b. 建立內層選擇結構 將 If / Else 方塊拖曳至外層選擇結構中的 Else 區塊中



c. 完成巢狀選擇結構

5. 設定外層選擇結構的判斷式（(使用者猜的數字）等於（密碼））

(1)拖曳（關係運算式<）

1. 點選物件視窗中的 world
2. 點選物件資訊區的 functions
3. 將 math 類別中的（a= =b）拖曳至程式編輯區中，並取代外層 If 後的 true

4. 暫時設定左方運算元為 2，右方運算元為 1

5. 完成（關係運算式= =）的設定

(2)設定左方運算元（使用者猜的數字）

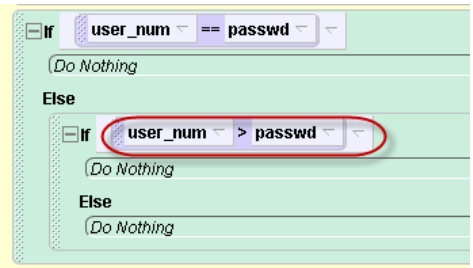
拖曳 user_num 變數至外層判斷式上並取代 2，完成設定

(3)設定右方運算元（1 meter）

拖曳 passwd 變數至外層判斷式上並取代 1，完成設定

6. 設定內層選擇結構的判斷式（(使用者猜的數字）大於（密碼））

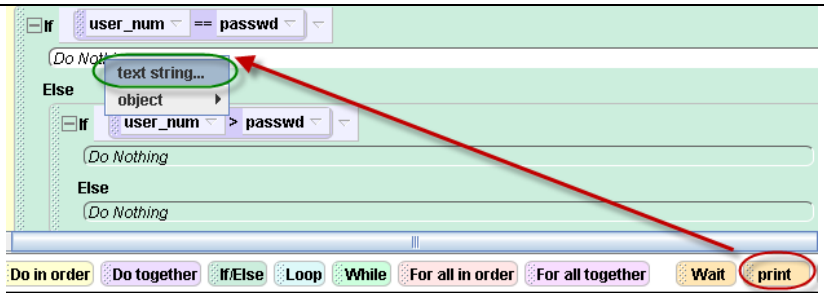
內層選擇結構的判斷式編寫方式與外層判斷式相同，請依前述方式編寫完成如下程式：

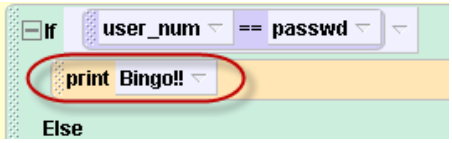


7. 編寫選擇結構內的程式區段

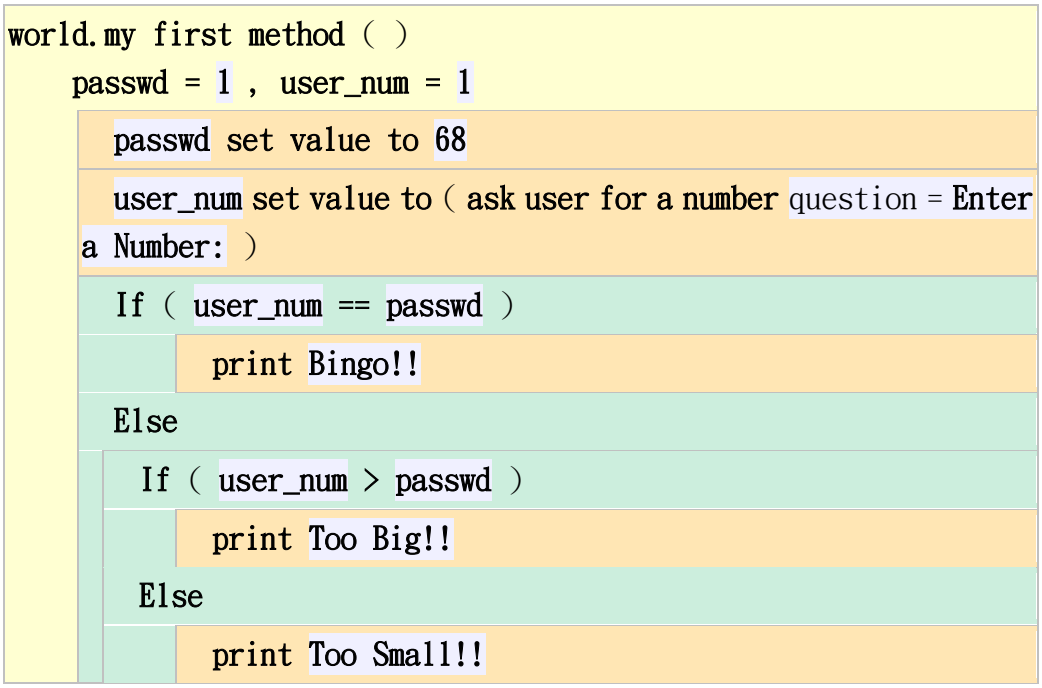
a. 印出（猜中了 Bingo）

拖曳 print 方塊至外層 If 下的程式區塊

<table>
<tr>
<td></td>
<td></td>
<td>


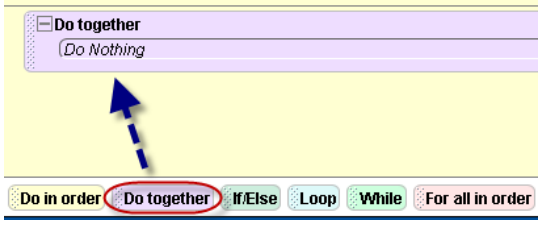
設定列印的字串為 Bingo!!


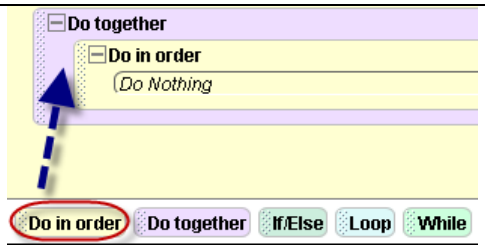
b. 印出（太大了）
拖曳 print 方塊至內層 If 下的程式區塊，設定列印的字串為 Too Big!!

c. 印出（太小了）
拖曳 print 方塊至內層 Else 下的程式區塊，設定列印的字串為 Too Small!!
8. 完成的程式碼如下：


</td>
</tr>
<tr>
<td>六、重複結構</td>
<td>2 節</td>
<td>6-1 什麼是重複結構
介紹什麼是重複結構、重複結構的類型，並以實例說明重複結構的用途與編寫方式。

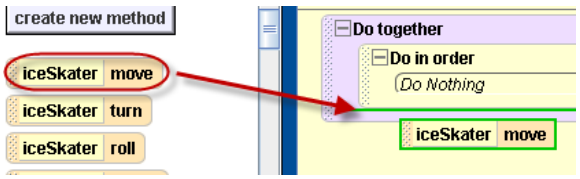
（展示）展示溜冰者重複進行流暢滑冰動作的程式，使同學瞭解本單元的學習目標，</td>
</tr>
</table>

		並引起同學的學習動機。 6-2 計數式重複結構 說明什麼是計數式重複結構。 6-2-1 溜冰者往前滑動的程式 在先前的例子裡，溜冰者只能很生硬的往前移動，是不是覺得很可惜呢？以下我們就來介紹讓溜冰者往前滑動的程式。 初始場景： 有一位溜冰者在溜冰場的左側邊緣，面朝溜冰場內部。 動作腳本： 為了讓溜冰者滑動的動作較為流暢，我們將滑動的動作分解為腳部動作與向前移動兩個部分，且最後這兩個部分必須能同步執行，才能達成在腳向後滑動的同時身體向前移動的動作。 同步執行以下 2 動作： 1. 循序執行以下動作（腳部動作）： 溜冰者右腳向後擺動 0.08 圈 溜冰者右腳收回（向前擺動 0.08 圈） 溜冰者左腳向後擺動 0.08 圈 溜冰者左腳收回（向前擺動 0.08 圈） 2. 溜冰者身體向前移動 1.5 公尺 程式編寫： 1. 建立同步執行的程式結構 拖曳 do together 方塊進入程式編輯區  2. 安置需同步執行的兩項動作 (1)建置腳部動作的循序執行區塊



(2)溜冰者身體向前移動 1.5 公尺

a. 將 iceSkater 物件的 move 方法拖曳至 Do together 結構中的 Do in order 結構下方



b. 以數字鍵盤視窗設定向前移動 1.5 公尺

3. 編寫腳部動作的四項分解動作

(1)溜冰者右腳向後擺動 0.08 圈

a. 點選物件區的溜冰者物件，並按前面的＋號展開物件

b. 點選 rightLeg 子物件

c. 拖曳 rightLeg 的 turn 方法至 do in order 結構中

d. 設定 turn forward 0.08 圈

以下(2)(3)(4)項動作請依序依動作(1)的設定方式完成。

(2)溜冰者右腳收回（向前擺動 0.08 圈）

(3)溜冰者左腳向後擺動 0.08 圈

(4)溜冰者左腳收回（向前擺動 0.08 圈）

4. 測試

5. 修改

請以指令敘述的 duration 設定來調整動作執行時間

（提示：同步執行結構中兩個動作的執行時間需相同）

6. 參考程式碼如下：

```
world.my first method（ ）
```

```
No variables
Do together
    Do in order
        iceSkater.rightLeg turn forward 0.08 revolutions duration = 1 second
        iceSkater.rightLeg turn backward 0.08 revolutions duration = 0.5 seconds
        iceSkater.leftLeg turn forward 0.08 revolutions duration = 1 second
        iceSkater.leftLeg turn backward 0.08 revolutions duration = 0.5 seconds
    iceSkater move forward 1.5 meters duration = 3 seconds
```

6-2-2 加上計數式重複結構（簡單版本）

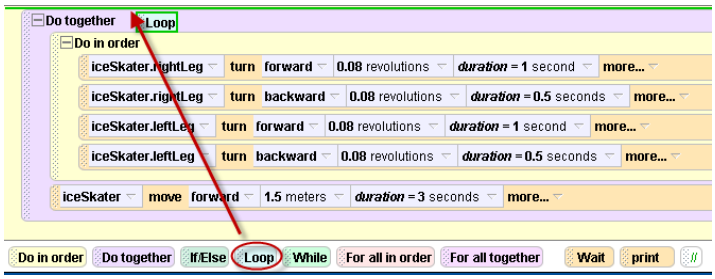
動作腳本：

重複執行以下動作 3 次：

溜冰者向前滑動的程式區段

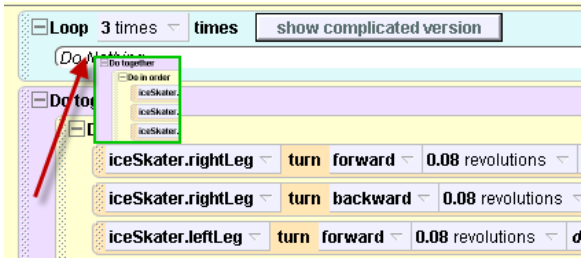
程式編寫：

1. 將程式編輯區下方的 Loop 方塊拖曳到程式區塊的第一行位置



2. 設定 Loop 結構重複執行 3 次

3. 將溜冰者向前滑動的程式區段（整個 do toghther 結構）拖曳至 Loop 結構中



4. 完成的程式碼如下：

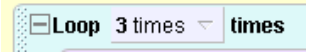
```
world.my first method ( )
    No variables
    Loop 3 times  times
        Do together
            Do in order
                iceSkater.rightLeg turn forward 0.08 revolutions  duration = 1 second
                iceSkater.rightLeg turn backward 0.08 revolutions  duration = 0.5 seconds
                iceSkater.leftLeg turn forward 0.08 revolutions  duration = 1 second
                iceSkater.leftLeg turn backward 0.08 revolutions  duration = 0.5 seconds
            iceSkater move forward 1.5 meters  duration = 3 seconds
```

6-2-3 計數式重複結構(複雜版本)

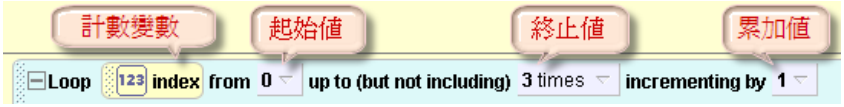
先前完成的程式中，我們可以利用計數式重複結構，很便利的讓程式區塊重複執行所設定的次數，但若希望在溜冰者每滑出一步前，都能指出現在滑出的是第幾步的話，簡單版本的計數式重複結構只能決定重複的總次數，顯然無法達成，因此，必須改用複雜版本的重複結構方能完成。那麼要如何啟用複雜版本呢？在先前的操作中，你應該已經發現在 Loop 結構中，有一個 show complicated version 的按鈕，點選該按鈕後即可開始使用。

首先，我們先來比較一下簡單版本和複雜版本的差異，在簡單版本中，只能設定重複結構執行的次數，而複雜版本則精細許多，可以設定重複結構的起始值、終止值以及累加值，在這個例子中，重複結構將由起始值 0 開始執行，每執行一次就累加 1，直到 2（最高到 3 但不包含 3）為止，亦即由 0、1 執行到 2 為止，一共也是執行 3 次。

簡單版本（simple version）



複雜版本（complicated version）



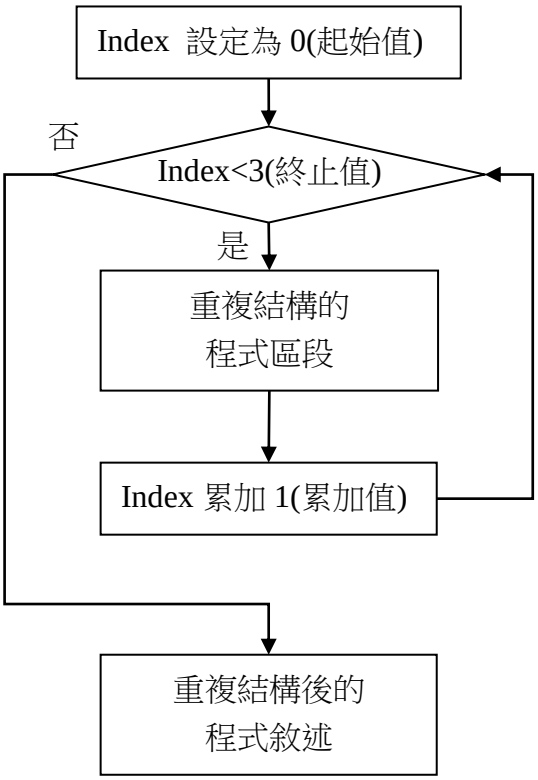
接著，我們以執行 3 次的例子，來介紹複雜版本的運作流程，讓同學們對於複雜版本的重複結構能有更深一層的認識。

1. 將起始值 0 指定給計數變數 index，在每一個計數式重複結構中都會搭配一個計數變數，作為重複結構執行時的計數器。

2. 判斷計數變數 index 是否仍小於 3，若是的話，則執行重複結構內的程式區段，反之，則跳出重複結構，執行後面的程式敘述。

3. 當執行完一次重複結構中的程式敘述後，便會將計數變數 index 累加 1

4. 跳到步驟 2 繼續進行判斷及後續處理。



起始場景：

接續前一單元

動作腳本：

重複執行以下動作：(起始值 0，終止值 2 (<3)，累加值 1，共會重複 3 次)

溜冰者說”Step”

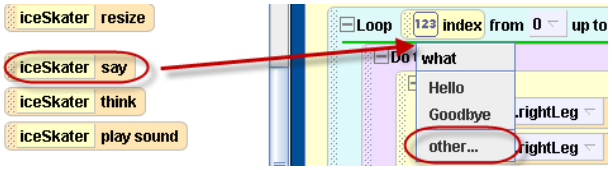
溜冰者說出目前的 index 值

溜冰者向前滑動的程式區段

程式編寫：

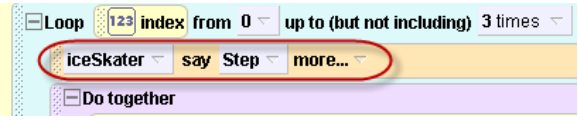
1. 點選 show complicated version 按鈕，啟用複雜版本的重複結構

2. 完成（溜冰者說”Step”）的指令敘述
   (1) 點選物件區的 iceSkater 物件
   (2) 點選物件資訊區的 say 方法
   (3) 將 say 方法拖曳至 Loop 程式區塊中的第一行位置



   (4) 在 other 選項的 Enter a string 視窗中，輸入 Step

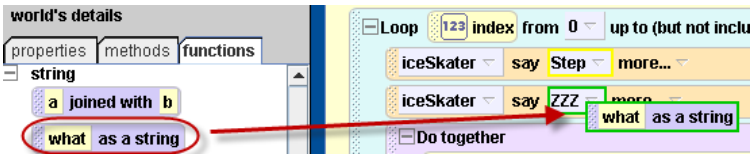
   (5) 完成如下的指令敘述



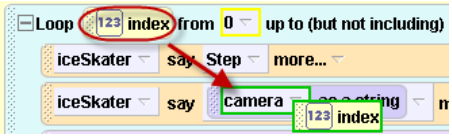
3. 完成（溜冰者說出目前的 index 值）的指令敘述
   (1) 拖曳 iceSkater 物件的 say 方法至 Loop 程式區塊的第二行位置，並任意設定 iceSkater 要說的話（如：ZZZ）
   (2) 將 Loop 結構的 index 變數拖曳進第二行的位置上，取代 ZZZ

此時你會發現 ZZZ 並沒有出現綠色方框，因此無法取代 ZZZ，原因在於兩者的資料型態不同，index 變數前面有 123 符號，表示其為數值型態的變數，而”ZZZ”則是字串型態(string)，若要以 index 變數取代 ZZZ，則需先進行資料型態的轉換，將 index 變數的內容轉換為字串型態，我們可以使用 world 物件中的(what) as a string 函數來處理。

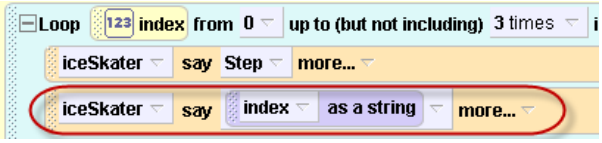
a. 拖曳 world 物件的(what) as a string 函數(function)至 Loop 中的第二行的位置上，取代 ZZZ ，並先將 what 任意設定為一個物件(如：camera)



b. 拖曳 Loop 結構的 index 變數進第二行的位置上，取代 what



c. 完成如下的指令敘述



在此行敘述中，index 變數中所存放的計數值，會先透過 what as a string 函數將數值轉換為字串型態，再將轉換完的字串交由 iceSkater 的 say 方法讓溜冰者說出來。

4. 測試與修改

請嘗試將起始值、終止值或累加值分別進行修改，並在執行時仔細觀察修改的結果，並比較各次修改之間的差異，以充分瞭解計數式重複結構的運作方式與用法。

完成的程式碼：

```
world.my first method ( )
    No variables
    Loop index from 0 up to (but not including) 3 times  incrementing by 1
        iceSkater say Step
        iceSkater say ( index as a string )
        Do together
            Do in order
                iceSkater.rightLeg turn forward 0.08 revolutions  duration = 1 second
                iceSkater.rightLeg turn backward 0.08 revolutions  duration = 0.5 seconds
                iceSkater.leftLeg turn forward 0.08 revolutions  duration = 1 second
                iceSkater.leftLeg turn backward 0.08 revolutions  duration = 0.5 seconds
            iceSkater move forward 1.5 meters  duration = 3 seconds
```

6-3 條件式重複結構

介紹什麼是條件式重複結構及條件式重複結構的程式編寫方式。

程式構想：

希望不論紅色三角錐放置在何處，溜冰者都可以滑動到三角錐前才停下來。

動作腳本：

（當（溜冰者與三角錐的距離）大於 2 公尺時）重複執行以下動作：

溜冰者向前滑動 1.5 公尺的程式區段

程式編寫：

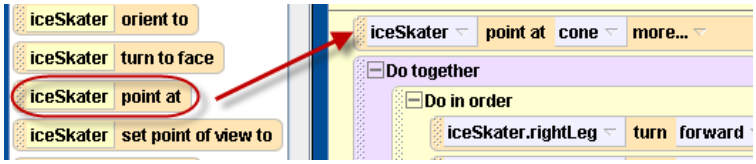
1. 完成如 6-2-1 之起始場景，並增加一個紅色三角錐

2. 完成溜冰者往前滑動的程式區塊

請參考 6-2-1 的說明完成如下程式片段

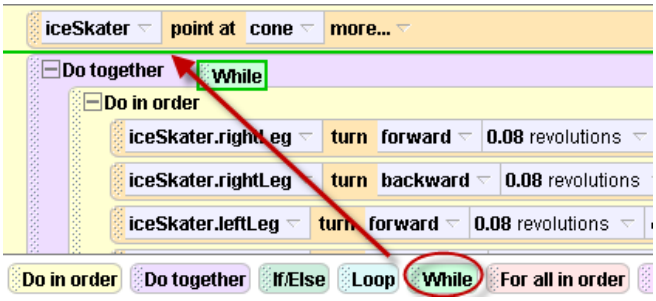
3. 先讓溜冰者朝向紅色三角錐

拖曳 iceSkater 的 point at 方法至程式最開頭



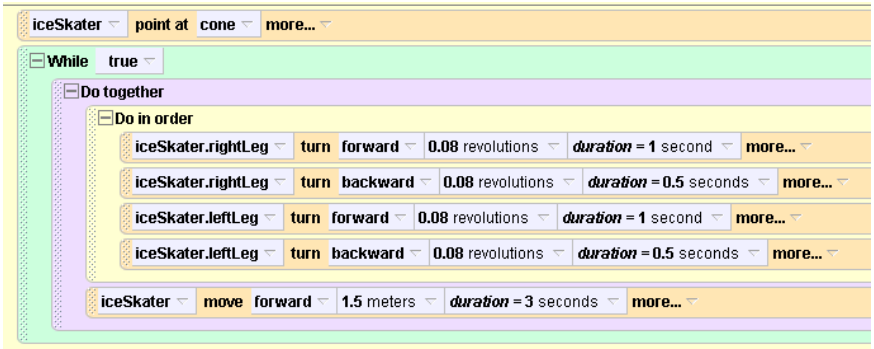
4. 新增 while 重複結構

(1) 拖曳程式編輯區下方的 while 方塊至第一行程式碼的下方



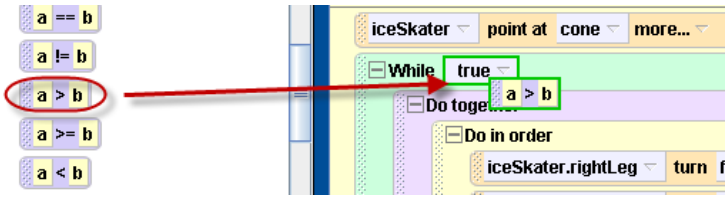
(2) 設定條件為 true（即當條件為真時執行重複結構中的程式）

(3) 將（溜冰者往前滑動的程式區塊）拖曳至 while 結構中



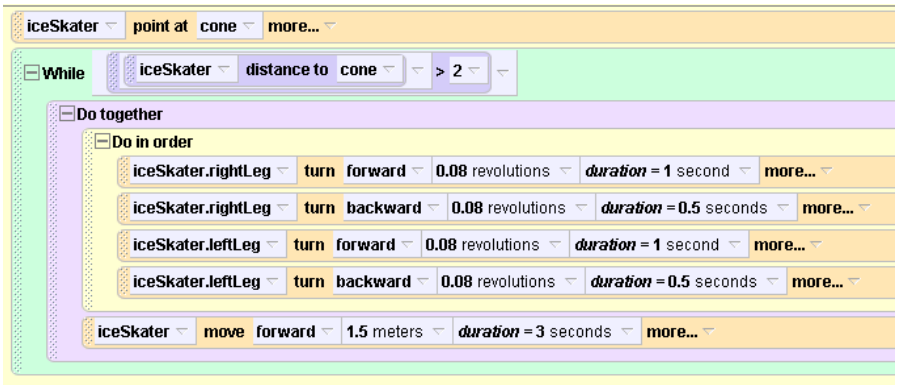
5. 設定 while 重複結構中的條件

(1) 拖曳 world 物件 functions 下的（關係運算 a>b）至程式區中，取代 while 後的 true，並設定 a 為 1 及 b 為 2



(2) 拖曳 iceSkater 物件之 distance to 函數至程式區中，取代關係運算式中的 1，並設定目標 object 為 cone

完成的程式碼如下：



執行與測試：

1. 請任意調整溜冰者或三角錐的位置，並觀察執行結果
2. 請嘗試修改 while 後的條件判斷式，並觀察執行結果