01

用ActionScript 3.0開始認識Flash CS3

Flash是應用在網路上非常流行且高互動性的多媒體技術，由於擁有向量圖像體積小的優點，而且Flash Player也很小巧精緻，很快的有趣的Flash動畫透過設計師的創意紅遍了整個網際網路。雖然很多人都對Flash可以做精美的網路動畫並不陌生，但是實際上Flash不僅如此，只要搭配ActionScript 3.0就可以搖身一變成為多媒體或商業服務的展示平台。在本章中，我們不只要介紹如何在Flash CS3中撰寫ActionScript 3.0，還要讓你知道如何正確的架構專案，了解一般人常犯的錯誤，這將會讓你在Flash之路上有一個好的開始與未來。

學習目標：

閱讀完本章你將會學習到：

- ◆ 認識Flash與ActionScript 3.0
- ◆ 透過Flash CS3使用ActionScript
- ◆ 撰寫ActionScript常犯的錯誤
- ◆ 用ActionScript控制時間影軸

探討主題：

1. Flash與ActionScript
2. Flash專案設計常見的錯誤

1-1 Flash與ActionScript

Flash是目前全世界最流行且市佔率最高的網路技術，經過Adobe的調查，Flash Plyaer在瀏覽器的散佈率高達98%。也就是說，若你用Flash來製作出一個動畫，全世界將會有98%以上的人可以觀看到Flash的效果，這遠比其它的網路技術更容易讓使用者去接受與觀賞，這一切都要歸功於Flash有趣、迷你且跨平台的特性。

Flash在當初設計的時候是用來製作有趣的網路動畫工具，並搭配簡單的ActionScript來操作單純的動畫行為。但是隨著時代的變遷與需求的改變，Flash已經不單單是一個動畫製作工具而已，在這幾年的轉變已經遠遠的超過當初我們所認識的Flash動畫，它已經發展為RIA(Rich Internet Application)與Web 2.0的解決方案了。

YouTube就是一個善用Flash技術的例子，你可以透過Flash來播放與分享各式各樣的影音，而且使用者不需額外再安裝Flash Player以外的撥放器，就能及時的參與互動，並且使用伺服器與資料庫的資源。不僅如此，甚至連國際知名大廠如Yahoo、SAP、Oracle等，也紛紛採用Flash的技術應用於人機介面的開發，這也證實Flash可以發揮其潛能於各個平台與各行各業中。

Flash的構成很簡單，其中包含了「美術設計」與「互動設計」兩大部分，美術設計使用Flash設計工具製作精美的動畫或圖片，讓使用者可以賞心悅目的觀看畫面效果的呈現，製作完的成品可以儲存在fla檔中。互動設計則是描述畫面上的元件該如何與使用者互動，它是一種邏輯的組合，簡稱為程式。在Flash中是用ActionScript來完成，撰寫完的ActionScript可以直接儲存在fla檔或as檔中。當這兩大部分結合在一起時，就可以發揮Flash最大的效果，完成唱作俱佳的互動多媒體平台。

專案完成後，就可以經過Flash的編譯成為SWF檔，這個SWF檔是以中介碼(Bytecode)的方式存在，最後再經由Flash Player來進行執行的動作，如圖1-1所示，這樣的好處就在於不論你是哪種平台，例如手機、遊戲機、PC、MAC、Linux等等，只要有Flash Player就可以觀看到Flash所製作出來的效果。

這個觀念與JAVA是非常相似的，JAVA也會先編譯成JAVA中介碼(JAVA Bytecode)，並交由JVM(Java Virtual Machine)在不同的作業系統上執行，好處就在於設計師只要編寫一次，在不用或少量的修改下，就可以快速的移植到不同的執行環境工作。

圖1-1：Flash的構成

說到ActionScript這套程式語言，為了要讓Flash發揮更強大的功能，從開始至今已度過了3個版本。由一開始單純的ActionScript 1.0，後來又加入物件導向與更多的功能成為ActionScript 2.0，在2007年又再度強化了物件導向與事件流成為ActionScript 3.0。在演進的過程中，除了功能上的改良之外，在執行的效率上更是有卓越的表現。

請參閱圖1-2，ActionScript 2.0執行效率要比ActionScript 1.0要快上6倍，而ActionScript 3.0執行的效率又比ActionScript 2.0快上10倍，有了這些效率上的改良，才能讓Flash完成更多不可能的視覺與互動任務。

圖1-2：ActionScript的演進

要讓Flash擁有互動的能力，千萬不能錯過ActionScript，因為所有的互動邏輯都是由它來完成的，但是要撰寫ActionScript要用何種工具呢？其實Flash本身就是一個開發與編譯的工具，接下來讓我們來初識Flash的動作面板，如此一來才能善用Flash的功能來撰寫互動的程式。

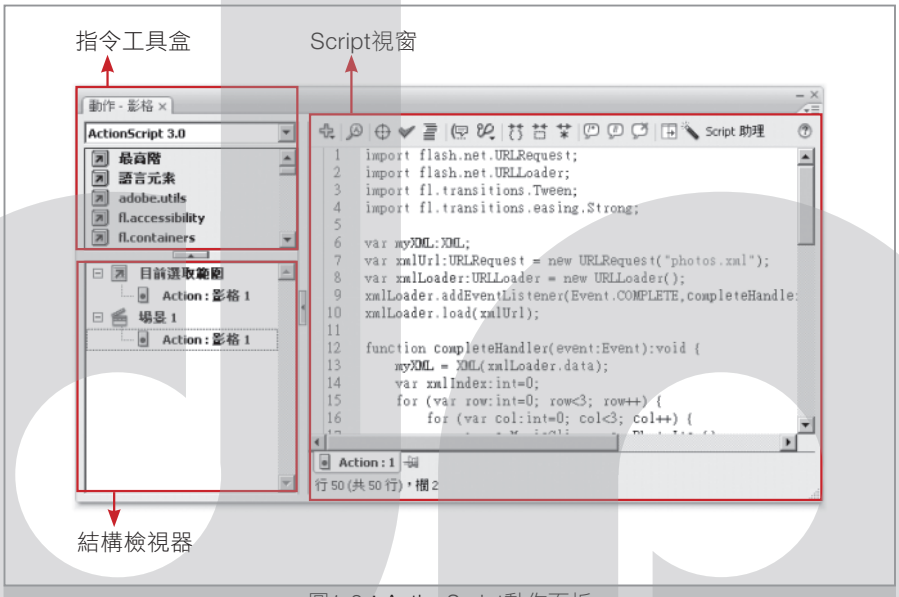

圖1-3：ActionScript動作面板

動作面板是在Flash中撰寫ActionScript的開發工具，主要包含了兩大區塊。圖1-3右邊的區塊為[Script視窗]，簡單的來說，就是讓你輸入ActionScript程式碼的地方，左邊則分為[指令工具盒]與[結構檢視器]，上方的[指令工具盒]視窗中將會列出了所有ActionScript指令功能參考，分門別類的依照特性將指令排放好，若需要使用，只需由選單中選擇需要的項目即可；下方的[結構檢視器]則是方便使用者在面臨較大的Flash專案時，還能迅速的找出我們曾經在哪些影格或是影片片段寫過的程式，它也可以顯示目前正在編輯程式的對象，避免我們寫錯對象。

如圖1-4，在[Script視窗]最上方的一排則為[工具列]，是用來幫助我們撰寫ActionScript的一些工具，每一種都有它獨特的功能：

圖1-4：動作工具列

接下來讓我們認識一下，動作工具列上的每個按鈕是代表什麼用意，在撰寫ActionScript的時候又能帶來什麼樣的好處。

在Script中增加新的項目

按下[在Script中增加新的項目]的按鈕後，會出現圖1-5的多層次的選單，可以由選單中選擇想要的程式碼指令，它會依照類型或是套件(Package)來排列。若你想找某個套件中的指令，但是卻又記不太起來正確的指令名稱，那這個功能將可以讓你用點選的方法來加入指令，加入後的指令將會出現在[Script視窗]中。

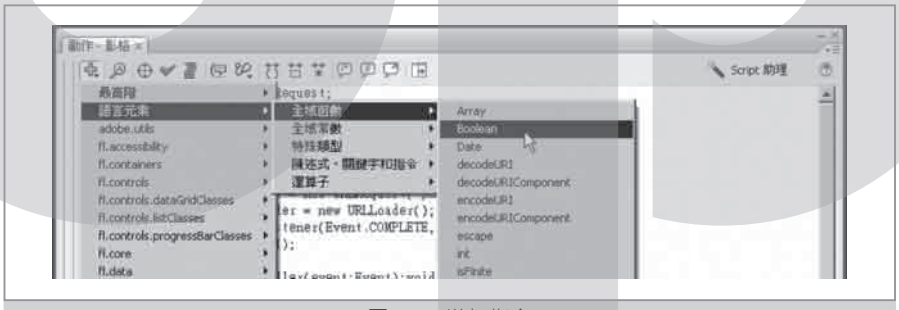

圖1-5：增加指令

尋找

按下[尋找]按鈕後，會在目前的程式碼中做搜尋的動作，它也同時具備取代的功能。如圖1-6所示，用[尋找項目]可以找到在程式碼中符合的程式區段，而[取代為]則可用來尋找程式碼中字串的位置，以及將字串取代為輸入的字串內容，這與WORD中的尋找取代功能是相仿的。

圖1-6：尋找與取代

插入目標路徑

按下[插入目標路徑]後，將跳出插入目標路徑視窗，點選其中的元件再按確定後，即可於[Script視窗]中插入該元件的路徑表示語法。如果在程式碼中你不想自己輸入目標路徑，就可以使用該功能，其路徑又分為[絕對]與[相對]兩種表示方法，[相對]選項選取時，其路徑表示法將會以this來表現相對路徑，反之若核選[絕對]選項，則會用root來表現絕對路徑，關於this和root的表示法，接下來的章節會再多加解釋。

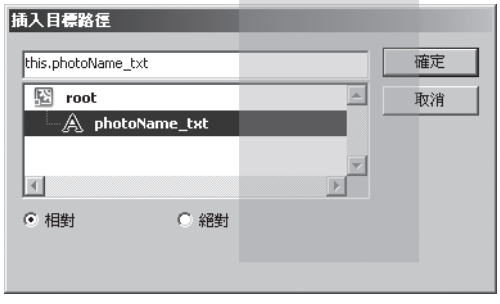

圖1-7：目標路徑

檢查語法

按下[檢查語法]按鈕後，可幫你檢查所撰寫的ActionScript是否正確，看看是否有用錯指令或打錯字的情況。這個功能相當方便，若你撰寫好一段程式，可以先檢查是否有語法上的錯誤，若沒有錯誤，會顯現圖1-8的視窗，提示你所撰寫語法沒有錯誤。

圖1-8：無錯誤訊息

反之，若是程式語法有誤，則會出現如圖1-9的錯誤訊息，提示你所撰寫語法有錯誤存在，並按下[確定]按鈕得知更進一步的訊息。

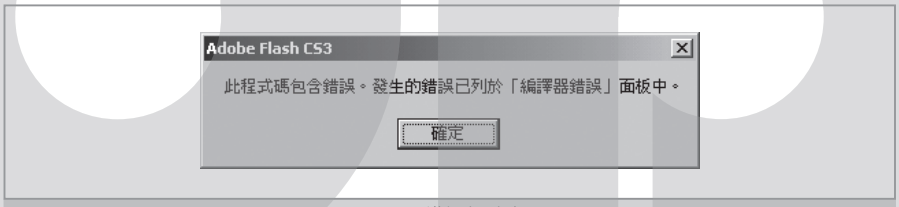

圖1-9：錯誤訊息提示

如圖1-10的提示，這時應該回到Script視窗中，按照[編譯器錯誤]的提示來進行修改，檢查程式碼功能僅可檢查語法上的錯誤，若是邏輯上的錯誤則無法檢測出來。

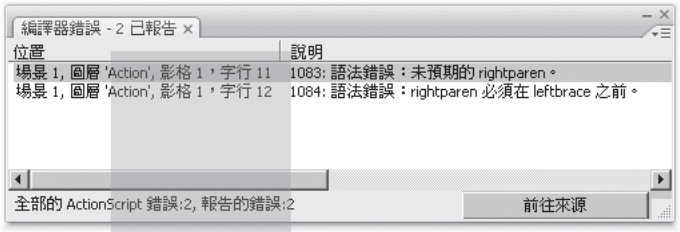

圖1-10：編譯器錯誤

自動格式化

按下[自動格式化]按鈕後，將可以幫你將程式碼自動做斷行與縮排格式處理，如圖1-11是一個未經過處理的原始碼，執行該功能後，程式碼將會重新整理如圖1-12的狀態，讓程式在編排上會較為容易閱讀。

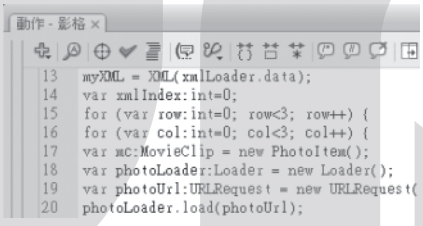

圖1-11：整理前的程式碼

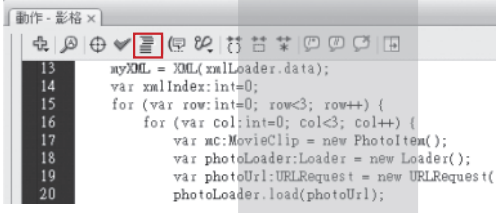

圖1-12：整理後的程式碼

顯示程式碼提示

在使用方法時，若是忘記裡面的屬性該如何填寫，可以在Script視窗中先用滑鼠點到想要提示的項目，再點選[顯示程式碼提示]按鈕，將會在該位置顯示程式碼的提示，如圖1-13，設計師就可以靠著提示來撰寫程式，而不用辛苦的翻Help來對照使用，是一個非常實用的功能。

圖1-13：程式碼提示

除錯項目

對[除錯項目]按鈕點選左鍵，可以從選單中選擇[切換中斷點]功能。如圖1-14所示，該功能可以為動作視窗中的程式碼加上中斷點標記，當使用者按下[Ctrl+Shift+Enter]啟用除錯模式時，程式會暫時中止在該中斷點處，並顯示目前程式中變數與屬性的內容，對於邏輯上的除錯相當有幫助。

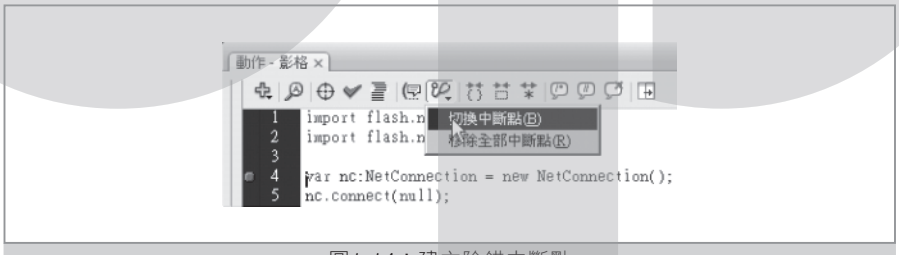

圖1-14：建立除錯中斷點

收合括弧之間

按下[收合括弧之間]的按鈕後，於Script視窗中被選到的程式碼，將會以括弧來做收合範圍，如圖1-15，自動將for迴圈括弧內的程式碼收合起來。這將可以讓你在閱讀程式碼時，先將看完或不重要的程式碼收起來，在閱讀上會清爽許多。

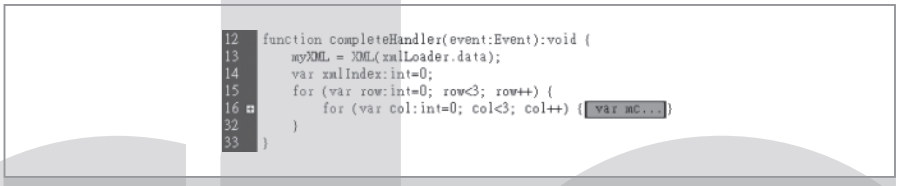

圖1-15：收合括弧之間的程式碼

收合選取範圍

按下[收合選取範圍]的按鈕後，於Script視窗中被選取到的程式碼，將會以選取的範圍來做收合範圍，如圖1-16，自動將選取範圍內的程式碼收合起來，這將可以讓你在閱讀程式碼時，做更彈性的收合動作，該功能與[收合括弧之間]是相同的，只不過是收合參考者不同罷了。

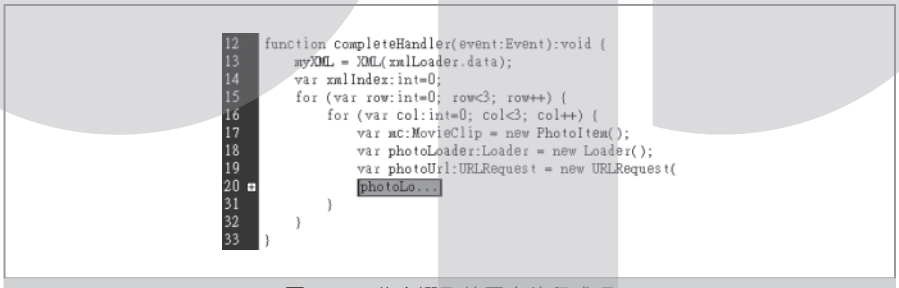

圖1-16：收合選取範圍中的程式碼

全部展開

按下[全部展開]按鈕，可以讓之前被收合過的程式區段再度的展開成原始的狀態，如圖1-17即為展開的結果。若是不想用該功能，也可以透過左邊程式碼編號區域的[+]或[-]圖示，來進行展開與收合的動作。

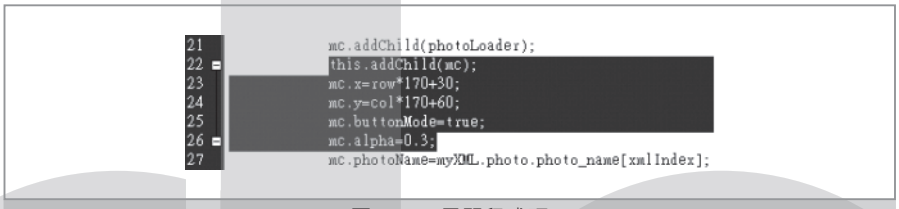

圖1-17：展開程式碼

套用區域註解

註解是指程式中產生不編譯的文字區段，所以註解可以用來描述程式的內容解說或是相關資訊的註記，使用前可先用滑鼠選取一段區塊，再按[套用區域註解]按鈕，可以看到如圖1-18的程式碼前後被加上「/*」及「*/」。被這兩個符號之間的程式碼將不會執行，套用區域註解功能適合在要標示一大塊或多行的註解時使用。

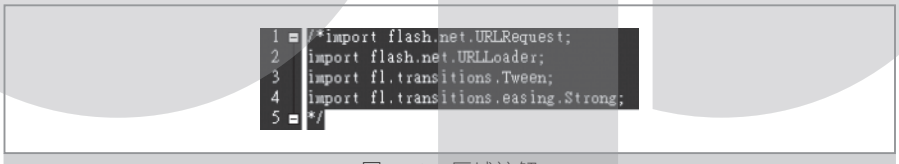

圖1-18：區域註解

套用字行註解

同套用區域註解功能，套用字行註解也是註解的一個方式，它是以「//」符號來代表該行為註解內容，一次僅可註解一行的程式，若需要註解多行則需要每行都加註「//」符號。使用前可先用滑鼠選取一段區塊，再按[套用字行註解]按鈕，可以看到所有被選取的程式碼前面都被加上「//」符號，如圖1-19所示，套用字行註解功能適合在要標示單行或少數行的註解時使用。

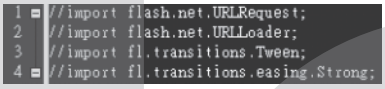

圖1-19：字行註解

移除註解

移除註解顧名思義就是將註解過的區域給移除註解符號，不論是區域註解或是字行註解都能透過這個方法來做，使用前可先用滑鼠選取一段已被註解的程式區塊，再按[移除註解]按鈕，就可以移除註解符號，這對取消註解而言是一個非常方便的功能。

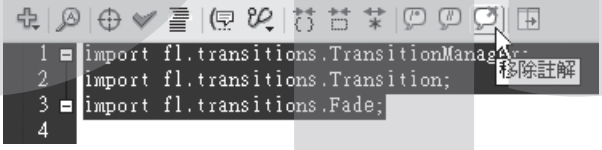

圖1-20：移除註解

顯示/隱藏工具箱

如圖1-21，在撰寫程式時若是覺得[Script視窗]太過於擁擠，想要加大編輯視窗，就可以藉由[顯示/隱藏工具箱]按鈕，將[指令工具盒]與[結構檢視器]兩個功能視窗隱藏或顯示，讓程式在撰寫上更加的方便。

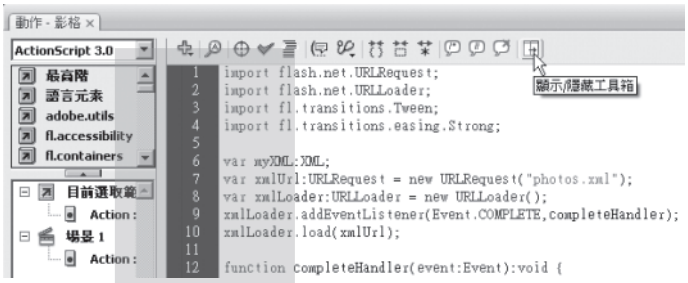

圖1-21：工具箱隱藏與顯示

在動作面板的最右側還有一個下拉選單，如圖1-22，若點選該下拉符號，將會出現關於Script視窗中的一些細部設定與所有工具列功能，也可以藉由[偏好設定]選項來自訂出你所習慣的程式撰寫環境。

圖1-22：Script視窗細部功能

點選[偏好設定]功能後，將可以看到如圖1-23的偏好設定視窗，你可以由左側的設定選項中調整對應的功能設定，例如可以由[類別→ActionScript]來設定是否要做程式碼提示，或是程式的字型與大小，甚至是標示的顏色都可以自己調整。

一般我們會建議將ActionScript中的[字體]改為Tahoma，並將字體大小設定為12，這樣將會讓字顯示的更清楚，當然你也可以調整成自己喜好的樣式。

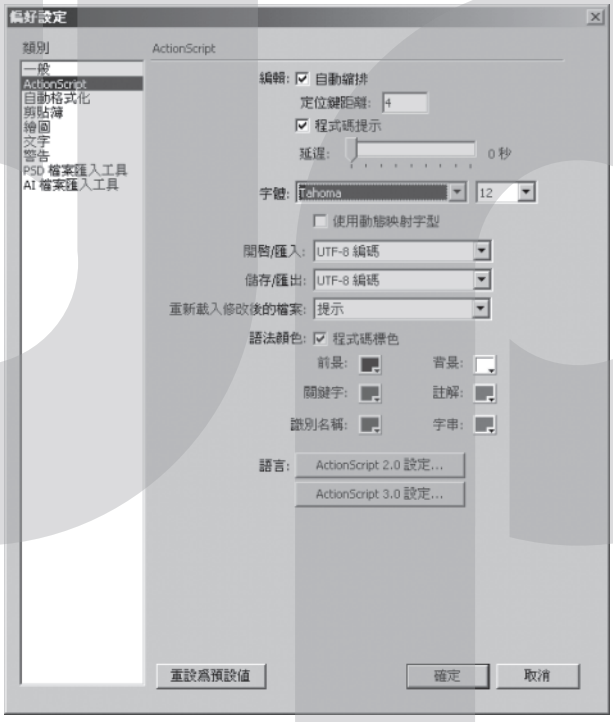

圖1-23：偏好設定視窗

1-2 Flash專案設計常見的錯誤

曾經有人向我反應Flash雖然是一個很棒的軟體，但是卻擁有難以維護的問題，其實這種說法並不是這麼的正確，因為是否好維護並不是以哪一種程式語言或開發軟體來決定，而是由撰寫程式的人來決定，若是用不當的方式來做專案的佈局，不要說難維護，甚至連要把專案給結掉也不是件容易的事。一個好的程式不在於寫了多少行程式，或是用了多少種語法，而是如何將專案用最短的時間保留最大的彈性。在這個章節中，讓我們來探討一般在Flash專案設計中常見的不當方法，並了解這會造成什麼問題，並以此借鏡。

範例1-1：影片與場景關係

在Flash中常見的有三種元件(Symbol)，分別是影片片段(MovieClip)、按鈕(Button)和圖像(Graphic)，當然若是使用ActionScript，會有更多種的元件類型可以使用。在這些元件中，其中影片片段的功能最多也最為強大，按鈕可以與使用者增加互動性，而圖像則只侷限在動畫表現，所以在開發專案時用的最多的就是影片片段，因此我們常會將焦點放在影片片段身上。

Flash中的root，本身是整個SWF結構中最上層的元件，它可以包含其它的影片片段、按鈕或圖像。其它的影片片段也具有相同的特性，可以層層的包含各種元件，一個影片片段元件可以用多個圖層、影格組成，而每個圖層和影格內又可以包含其它的元件，因此可以完成相當複雜的影片片段結構，也才能表現出各種奇形怪狀的創意。

請打開範例檔案[AS3Sample\CH01\1-1.fla]，可以看到如圖1-24的人偶，這個人偶就是一個影片片段元件，其中也包含了很多的影格與其它元件，藉由影格與元件的組成，當執行的時候才能做出走路的動畫效果出來。

圖1-24：影片片段元件的構成

說到動畫效果，在Flash中，時間軸的觀念是比較複雜的，因為除了我們看到場景上面的時間軸之外，其實每個影片片段(MovieClip)也都擁有各自獨立的時間軸，如果我們能善用這些時間軸的特性，就可以做出更複雜豐富的動畫或是效果。

在這個範例中我們設定了兩組按鈕控制項，分別是針對[主角人偶]與[場景]來製作的，可以透過ActionScript(先嘗試用ActionScript 1.0的語法)來做影片播放動作的設定，例如以下的ActionScript 1.0程式碼，就能幫助我們在滑鼠按下以後，協助播放時間軸。

```
on(Press){
    滑鼠按下的執行動作…
}
```

表1-1即是ActionScript中常用來播放或是停止時間軸的內建方法，同時也可以套用在影片片段上。若想要藉由互動來控制時間軸，你就可以參考該表來加以設計出你想要的效果。

方法	說明
stop( )	停止在目前所播放的影格。
play( )	從目前的影格開始播放。
gotoAndStop(影格,[場景名])	跳到所指定的影格並停止播放，場景名是選填項目，若不填寫代表為使用中的場景。
gotoAndPlay(影格,[場景名])	跳到所指定的影格並開始播放，場景名是選填項目，若不填寫代表為使用中的場景。
nextFrame	跳至下一個影格並停止。
prevFrame	跳至上一個影格並停止。
nextScene( )	跳到下一個場景，若是沒有該場景，則跳回本場景的第一格。
prevScene( )	跳到上一個場景，若是沒有該場景，則跳回本場景的第一格。

表1-1：時間軸控制方法

步驟1 ≫

打開範例1-1.fla，人偶的動畫已經事先製作好了，所以當執行時會表現出原地踏步的感覺。現在要完成的是讓人偶在3秒內由畫面的右側走到左側，由於Flash的影格速率是12.0 fps，因此在放置人偶的圖層person上第36個影格，如圖1-25，用滑鼠右鍵從選單中點選[插入關鍵影格]的功能，代表在第3秒處可以紀錄一個動畫的改變。

圖1-25：增加關鍵影格

步驟2 ≫

如圖1-26，我們在第1個影格上設置人偶於場景右側且為原來大小，而第36個影格上重新設置位置為場景左側，並縮小其原比例，製造出由近而遠的視覺感受。

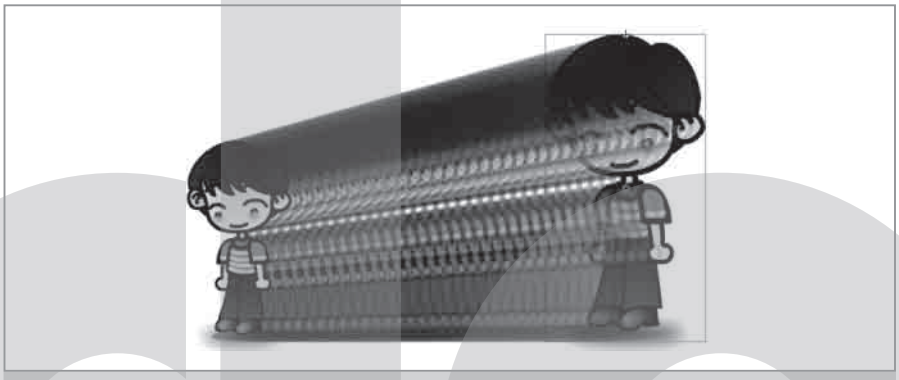

圖 1-26：配置動畫

步驟3 ≫

如圖1-27，在配置好位置的person圖層時間軸上，在[屬性面板]中將[補間動畫]的設定調成[移動]，代表要做出移動補間動畫的效果。

圖 1-27：設定補間動畫

步驟4 ≫

如圖1-28，因為目前放置人偶的person圖層，時間軸製作到第36個影格，若其它圖層沒有同樣延伸至36影格，將會看不到其它背景的效果，因此將所有圖層全部延伸到第36個影格處後，點擊滑鼠右鍵從選單中使用[插入影格]功能，將影格補足至36格，方可在動畫撥放過程中看到所有的元件與動作。

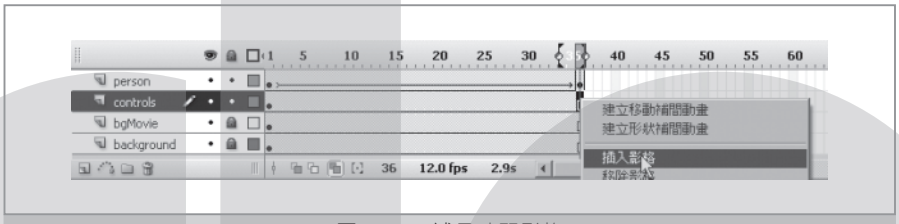

圖 1-28：補足時間影格

其執行結果如圖1-29所示，播放時人偶會做出由場景右側慢慢縮小至畫面左側的移動補間動畫，這是人偶在場景上的動畫表現。但是人偶自己本身內部也有走路的動畫，兩個動畫效果加起來，就像是人偶用走路的方式，由畫面右側走到左側，且越走越遠的感覺，還會不停的循環播放。

不過，到此為止我們只做出了單純的動畫而已，接下來要完成互動的介面，讓使用者可以透過ActionScript來控制時間軸的播放或停止。

圖 1-29：人偶走路效果

步驟5 ≫

在ActionScript 1.0的語法中，若是要控制按鈕，就必須將語法寫在該元件上，如圖1-30，先點選代表場景的play的按鈕，再打開動作視窗。請注意看，這時標籤上應該會顯示[動作-按鈕]，代表目前程式寫作的對象是按鈕，這也是常常大家會犯的錯誤之一，並描述其互動的程式碼，如果為[動作-影格]則代表有誤。

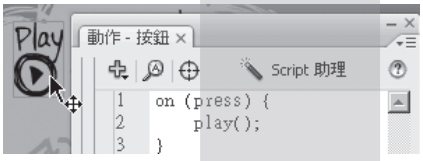

圖1-30：增加場景播放動作

```
01 on (press) {
02   play();
03 }
```

程式碼：範例1-1-1

01行，用on(Press)來描述當該元件被按下(Press)時，將會執行01~03行大括弧內的程式碼。

02行，使用play這個方法來播放時間軸，若是play前沒有加應用的對象，代表要控制的為場景上的時間軸，也可將程式寫為root.play( )，其效果是相同的。

步驟6 ≫

同步驟5將控制語法寫在元件上，如圖1-31，先點選代表場景的stop的按鈕，再打開動作視窗描述互動的程式碼。

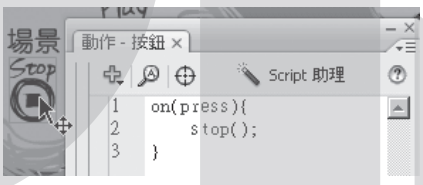

圖1-31：增加場景停止動作

```
01 on (press) {
02    stop();
03 }
```

程式碼：範例1-1-2

第01行，用on(Press)來描述當該元件被按下(Press)時，將會執行01~03行大括弧內的程式碼。

02行，使用stop這個方法來停止時間軸，若是stop前沒有加應用的對象，則代表要控制的為場景上的時間軸，也可將程式寫為root.stop( )。

步驟7 ≫

完成了場景上的時間軸控制，接下來要控制人偶的時間軸。因為等一下要執行的play和stop對象為人偶而非場景，所以需要先將人偶取個實體名稱，這樣在ActionScript中才會方便呼叫使用。如圖1-32，回到第一個影格上，先點選人偶這個影片片段元件，並在[屬性面板]中將實體名稱設為「boy」。

圖1-32：命名人偶實體名稱

步驟8 ≫

如圖1-33，先點選代表人偶主角的play的按鈕，再打開動作視窗描述滑鼠點擊的互動程式碼。

圖1-33：增加人偶播放動作

```
01 on (press) {
02     boy.play();
03 }
```

程式碼：範例1-1-3

01行，用on(Press)來描述當該元件被按下(Press)時，將會執行01~03行大括弧內的程式碼。

02行，使用play這個方法來播放時間軸，由於這時人偶為要操控的對象，所以必須在play方法前加上應用的對象，故寫成boy.play( )來播放人偶上的時間軸。

步驟9 ≫

如圖1-34，先點選代表人偶主角的stop的按鈕，再打開動作視窗描述互動的程式碼。

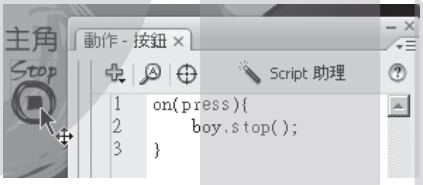

圖1-34：增加人偶停止動作

```
01 on (press) {
02     boy.stop();
03 }
```

程式碼：範例1-1-4

01行，用on(Press)來描述當該元件被按下(Press)時，將會執行01~03行大括弧內的程式碼。

02行，使用play這個方法來停止時間軸，由於這時人偶為要操控的對象，所以必須在stop方法前加上應用的對象，故寫成boy.stop( )來停止人偶上的時間軸。

撰寫完範例1-1以後執行，其動畫呈現結果跟之前的是一模一樣的，只不過這次可以用按鈕來操作停止或播放時間軸。首先請先按下所屬場景的stop按鈕，可以看到人偶停止由右往左移動，但是原地踏步的動作卻沒有停止，如圖1-35所示，很明顯的當stop這個方法執行的時候，只有停止了場景上的時間軸，並不會去影響人偶(boy)這個影片片段中的時間軸。由此可知，每個元件內的時間軸是相互獨立的，如果再按下所屬場景的play按鈕，人偶就會繼續往左移動，恢復原來的樣子。

圖1-35：操作場景時間軸

換個方法再來試一次，這次要來控制人偶主角的時間軸，所以當所屬主角的stop按鈕被按下，這時人偶的踏步動作會立即的被凍結住，但是整個人偶依然會向畫面左邊移動，變成像是溜冰的滑稽動作，如圖1-36所示。這是因為我們只停止了人偶身上的時間軸，但是場景上的時間軸依然繼續進行，再度證明了時間軸在元件中是相互獨立的，點選所屬人偶的play按鈕，人偶會恢復踏步的動作。從範例1-1的驗證中得知，若想要整個停止住畫面，光是停止場景的時間軸是不夠的，還必須將每個元件的時間軸都全部停止，才能達成這個目的。

圖1-36：操作人偶時間軸

除了時間軸的觀念外，範例1-1也告訴我們用這種方法來設計專案是非常不好的開始，尤其是想要完成一個較為複雜的互動專案，這將會造成維護上非常大的困擾，原因就在於程式碼太過散亂。如圖1-37，在程式中為了要控制4個按鈕的動作，所以分別在按鈕元件上撰寫了on(Press)的程式碼，如果專案有問題或是要調整時，必須要一個個的點入元件，才能觀看或修改程式碼。

這是一件非常辛苦且沒有效率的事情，想想目前範例1-1還只是一個超級單純的程式，我們就覺得不容易管理了，那麼別說在複雜的互動中，可能會有幾十個可供互動的元件，要如何在幾十個元件中找尋錯誤或需要修改的程式碼，就猶如大海撈針一般的痛苦。

雖然有非常多的Flash設計師可能都在使用這種方法在設計互動，但是這的確不是一個好的設計架構。基本上我們會希望能將程式碼集中在一起，才會方便我們修改跟維護專案內容。

圖1-37：架構示意圖

既然範例1-1用ActionScript 1.0會造成程式碼散亂的問題，那麼改用ActionScript 2.0好了，至少他可以寫在某一個關鍵影格上。當使用ActionScript 2.0來開發互動時，就不能像ActionScript 1.0一樣直接寫在元件中，你必需寫在時間軸的關鍵影格或是外部的as檔上。

以剛剛按鈕被按下的動作為例，在ActionScript 2.0的語法要寫成：

```
元件.onPress=function(){
    滑鼠按下的執行動作…
}
```

除了事件的描述方法不同之外，時間軸的控制方法在ActionScript 1.0、2.0、3.0中並沒有什麼太大的差異。在範例1-2中，我們要用ActionScript 2.0來實現動動腦遊戲這個簡單的互動，並來探討其中有哪些大家會常犯的錯誤。

範例1-2：幼兒動動腦遊戲

請打開範例檔案[AS3Sample\CH01\1-2.fla]，在本範例中要完成一個對答的遊戲，使用者可以藉由點選畫面上的答案來作答，並且會告知答題後的對錯。這是一個非常簡單的互動程式，但是卻可以很清楚的發現其缺失與不好之處。

步驟1 ≫

如圖1-38，有三個答案的影片片段元件放置在場景上。影片片段也可以用來描述點擊的動作，所以當然可以用來取代按鈕來進行互動，不過由於ActionScript 2.0不能直接在元件中寫互動程式，所以必須將3個影片片段分別取上實體名稱，才能讓程式指定為互動的對象。由左至右的實體名稱分別為answerA_mc、answerB_mc與answerC_mc。

圖1-38：答題按鈕

步驟2 ≫

為了要能完成作答，所以我們利用時間軸的特性。如圖1-39所示，所有的圖層都只設立1個關鍵影格，除了在speak圖層上才另外分成3個關鍵影格，這是為了讓每個影格上的畫面都不一樣，而第1個影格是顯示問題提示作答的畫面。

圖1-39：答題的畫面

步驟3 ≫

如圖1-40，在第2個影格上將原來的問題提示與作答按鈕刪去，並在畫面上加上「真聰明，答對啦～」的字樣，用來當作問題答對時該顯示出的畫面。

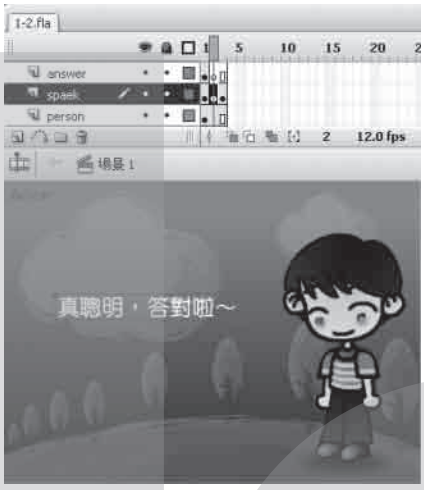

圖1-40：答對的畫面

如圖1-41，在第3個影格上將原來的問題提示與作答按鈕刪去，並在畫面上加上「答錯了，再試試～」的字樣，用來當作問題答錯時該顯示出的畫面。

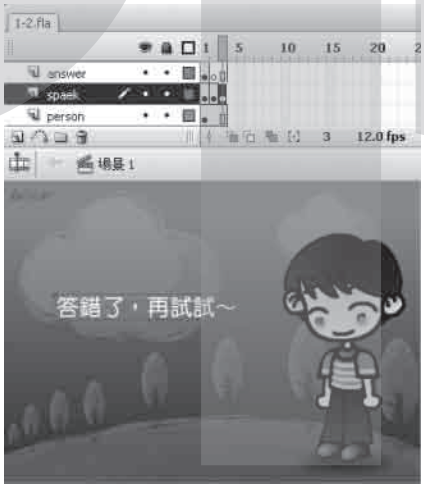

圖1-41：答錯的畫面

其答題的邏輯可以參考圖1-42，透過簡單的gotoAndStop指令，就能控制時間軸停在第幾格來呈現答對與否，若答對則要停在第2個影格，否則若答錯則跳到第3個影格。

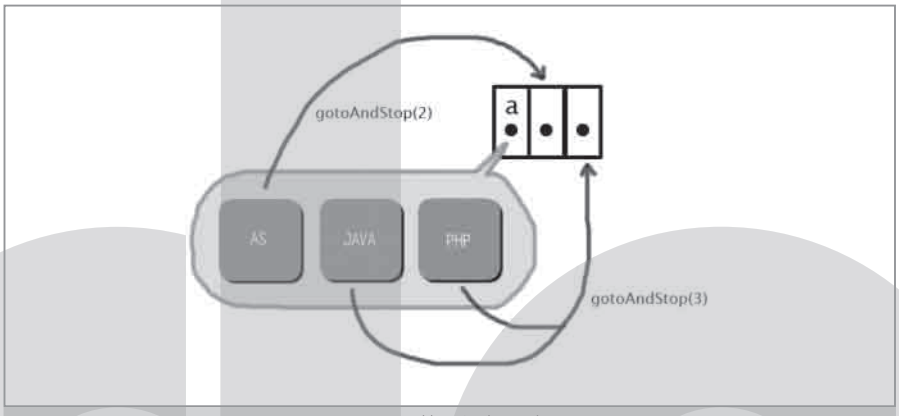

圖1-42：答題邏輯示意圖

步驟4 ≫

為了將程式碼集中在一起，如圖1-43，產生一個新的圖層名為「Action」，上面不放置任何元件，只單單用來撰寫程式碼，這樣可以讓程式碼單純化，否則若是元件跟程式碼共用同一個關鍵影格或是圖層，將來在選取時就很容易選錯。

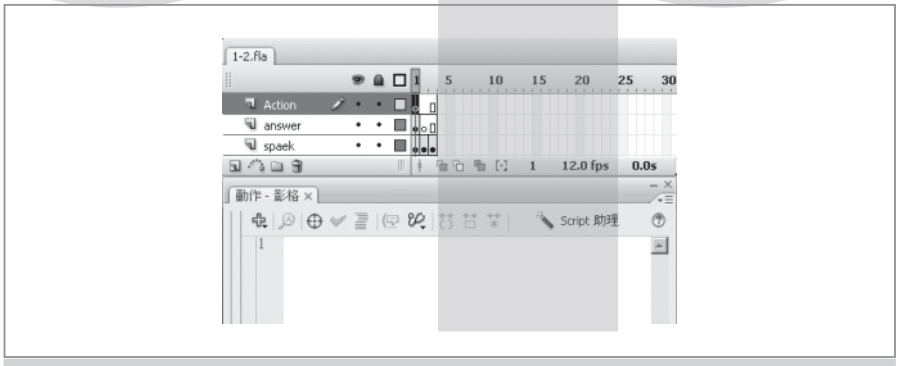

圖1-43：在Action圖層上匯集程式碼

步驟5 ≫

依照之前所設計的劇情，將互動的結果用ActionScript 2.0描述出來：

```
01 stop();
02 answerA_mc.onPress = function() {
03     gotoAndStop(2);
04 };
05 answerB_mc.onPress = function() {
06     gotoAndStop(3);
07 };
08 answerC_mc.onPress = function() {
09     gotoAndStop(3);
10 };
```

程式碼：範例1-2

01行，先將場景停下來，否則動畫會循環播放，使用者就會不停的看到1~3影格的內容，不僅是穿幫了，使用者也無法順利操作答題的介面。

02~04行，將answerA_mc元件註冊滑鼠按下的動作，當使用者用滑鼠點擊元件時就會自動的被執行第03行的程式碼，將影片影格跳至第2個影格並停止播放，以顯示答題正確的畫面。

05~07行，將answerB_mc元件註冊滑鼠按下的動作，當使用者用滑鼠點擊元件時就會自動的被執行第06行的程式碼，將影片影格跳至第3個影格並停止播放，以顯示答題錯誤的畫面。

08~10行，將answerC_mc元件註冊滑鼠按下的動作，當使用者用滑鼠點擊元件時就會自動的被執行第09行的程式碼，將影片影格跳至第3個影格並停止播放，以顯示答題錯誤的畫面。

執行範例1-2的結果如圖1-44所示，使用者可以藉由點選下方3個答題的影片片段來做答題的動作，點擊答題按鈕後即可顯示回答正確與否，是一個非常簡單的互動程式。

圖1-44：幼兒動動腦遊戲

雖然範例1-2已經改用ActionScript 2.0來撰寫程式，並且將程式碼匯集在一個關鍵影格上，但是我們還是要說，這種方法也並不是一個好的專案架構，為什麼這麼說呢？這種寫法存在了兩個都是跟時間軸有關的問題。第一個問題是當程式碼寫在關鍵影格上時，就會有先後執行順序上的問題，簡單的來說，就是當時間影格播放時，先播放的關鍵影格上若有程式碼，就會先執行，但是在範例1-2中，我們用gotoAndStop這樣的指令，就可以命令影格的播放要跳至哪個影格繼續，這種行為很容易讓程式執行的順序亂掉，尤其是一個複雜的互動程式如圖1-45，你將會無法控制程式碼而造成各種千奇百怪的錯誤。

第二個問題在於複雜度高的互動程式若使用這種專案架構，勢必會產生一堆又長又多的時間軸，無可避免的就必須在時間軸上面多增加幾個關鍵影格，來放置ActionScript與控制影格播放，這樣一來又等於走了回頭路，因為程式碼還是不能集中在一起，反而是散佈在時間影格上，並沒有比ActionScript 1.0的方法好到哪去。雖然很多Flash設計師也都在用這種架構在編寫專案，但是時間軸混著程式碼一起編寫，確實是一個非常糟糕的方法，隨著專案的成長，會製造出更多的問題出來。

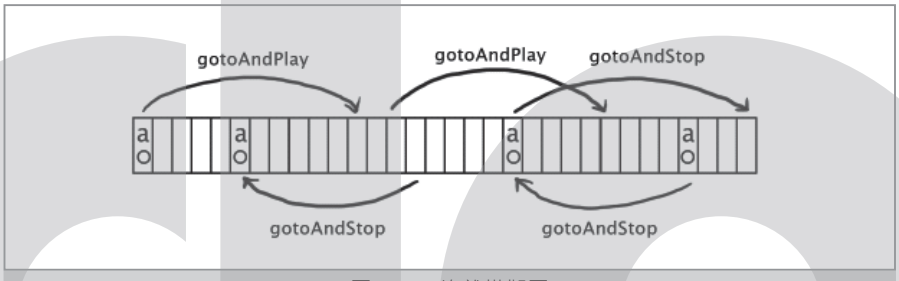

圖1-45：複雜模擬圖

問與答

既然時間軸會製造這麼多的麻煩，是否代表在Flash中，從此以後再也不應該用時間軸這個東西？

其實並不盡然，若是在Flash中不使用時間軸，那就消滅了當初Flash動畫的優勢了。

我們所說的是不讓時間軸跟程式碼混在一起，因為時間軸可能會造成程式碼執行的不定數，有可能會亂跳或是重複執行，若要使用時間軸，也請先包裝成影片片段，這樣在撰寫程式的時候就不會受到時間軸的影響。

總結

Flash與ActionScript是相輔相成缺一不可的好伙伴。Flash可以提供動畫設計與繪製向量圖檔的工具，製作精美的網路動畫，而ActionScript是支援Flash的程式語言，使用ActionScript可以用來產生各種與使用者互動的效果，編譯完成後會成為SWF檔，只要透過Flash Player就可以在不同的平台上執行。

要在Flash中撰寫ActionScript程式，可以使用動作視窗中的各項功能，善用這些工具能讓程式撰寫更加的便利。在開發Flash專案時避免使用不良的設計架構，否則將會造成未來專案維護及修改上的困難，設計常見的錯誤有兩大類型：

- 在元件中撰寫ActionScript程式碼

  這將會造成程式碼散佈在許多的元件中，設計師會難以找到之前所寫入的程式，而變得難以維護。

- 混合使用時間軸與ActionScript程式碼

  混著時間軸使用的程式碼，可能會因為時間軸播放的順序而造成程式執行順序的不同，很容易造成難以預測的錯誤，且當專案複雜時，還是會造成程式碼散佈在許多關鍵影格上。

有好的設計架構，才能夠讓專案順利的撰寫下去，也才能避開許多潛藏性的地雷。因此，有正確的觀念永遠是最重要的事情。