四、轉譯器的設計及實作

4.1 系統架構

本轉譯系統由 ParseMap、ParseAdlTree、ParseScnTree、TransObj2VRML 及 TransAction2VRML 等五大模組所組成。圖 16 顯示轉譯系統模組架構與其配合系統之相關示意圖。圖形最上方虛線圓角區塊為「3D 教材樣板編輯雛形系統」所輸出的 3D 教材描述檔，在此處我們以 F.ini、FScn#.adl 及 FScn#.ini 等檔名來代表。右方圓柱形區域代表本系統所使用到的媒體檔類型。中間的大方形區塊代表轉譯器，其中之五小塊長方形區塊代表轉譯器的五大模組。

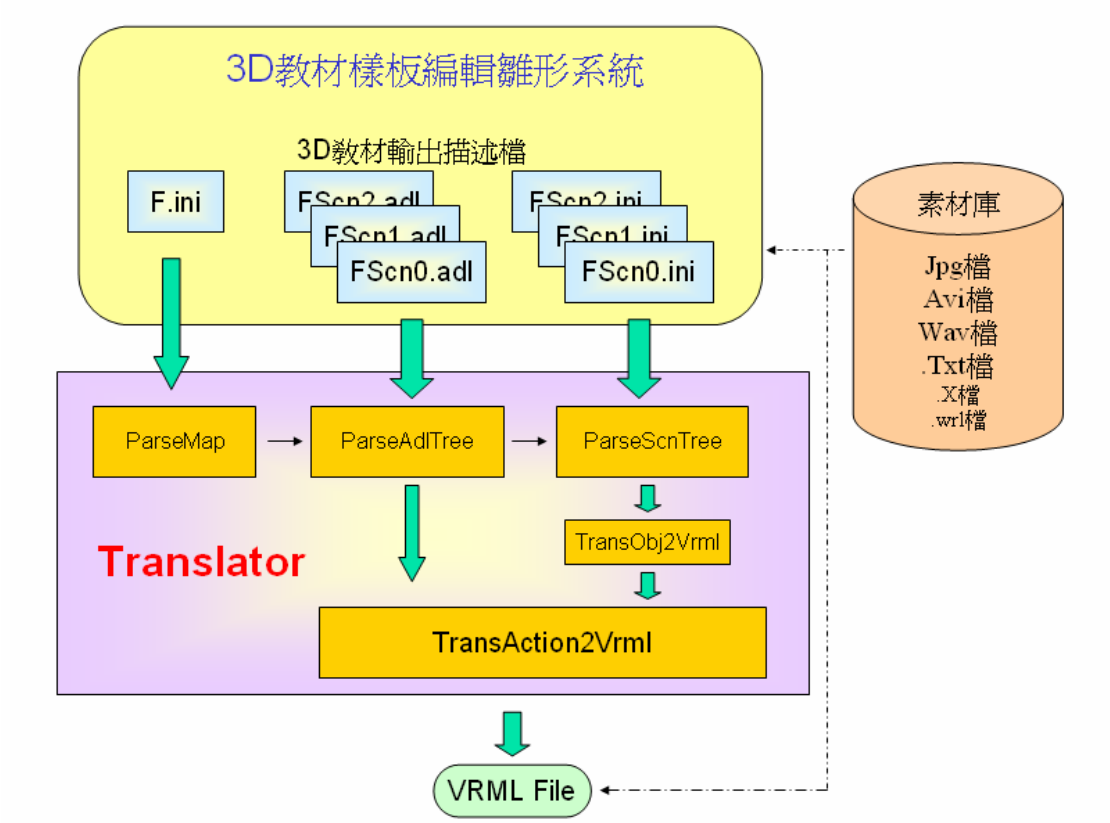

圖 16 轉譯系統模組架構

轉譯器的五大模組功能分別為：

1. ParseMap 模組：取得 3D 教材各畫面的名稱及劇情檔檔名。
2. PaseAdlTree 模組：讀入各畫面的劇情及各素材屬性記錄檔的檔名，並建立教材劇情樹狀結構。

3. PaseScnTree 模組：讀入各畫面的素材屬性記錄，並建立素材屬性樹狀結構。
4. TransObj2VRML 模組：將各素材轉成 VRML 節點。
5. TransAction2VRML 模組：加入 VRML 動作指令並產生 VRML 檔案。

轉譯器的運作流程如下：

1. 首先讀入 F.ini 檔，由 ParseMap 模組取得 3D 教材各畫面的名稱及劇情檔檔名。在此處劇情檔檔名為 FScn0.adl、FScn1.adl、FScn2.adl…….。
2. PaseAdlTree 模組依據 FScn0.adl、FScn1.adl……等檔名，讀入各畫面的劇情及取得各素材屬性記錄檔的檔名(FScn0.ini、FScn1.ini…)，並建立教材劇情樹狀結構。
3. PaseScnTree 模組依據 FScn0.ini、FScn1.ini…..等檔名讀入各畫面的素材屬性記錄，並建立素材屬性樹狀結構。
4. TransObj2VRML 模組依據素材屬性樹狀結構將各素材轉成 VRML 節點。
5. TransAction2VRML 模組探訪教材劇情樹狀結構，按照 VRML 語法加入動作指令，結合各素材之 VRML 節點，輸出 VRML 檔案。

4.2 VRML 文件檔的結構簡介

在談到如何轉譯成 VRML 前，我們需要先介紹 VRML 文件檔的結構。VRML 文件檔的架構分為三個部分[3][4][5]，如圖 17 所示：

(1)Header 區塊：

第一行一定要寫成”#VRML V2.0 utf8” 這樣的格式，用來宣告這個 VRML 文件的版本及所採用的字元集，目前所使用的都是 V2.0 的版本，字元集所採用的是國際 utf8 字元集合。如果想要記錄程式的標題及版權、作者、完成日期等資訊，則可以使用用 WorldInfo{ }這個節點來做記錄，或者使用註解符號#，將說明訊息寫在#之後。

(2)場景物件描述區塊：

場景及物件節點的屬性描述，如物件的種類、大小、位置、顏色、材質及所引用的物件模型種類等。

(3)物件動作描述區塊：

本區域描述物件的動作類型及順序，以 ROUTE….TO….指令串起各物件節點間的輸入及輸出欄位。

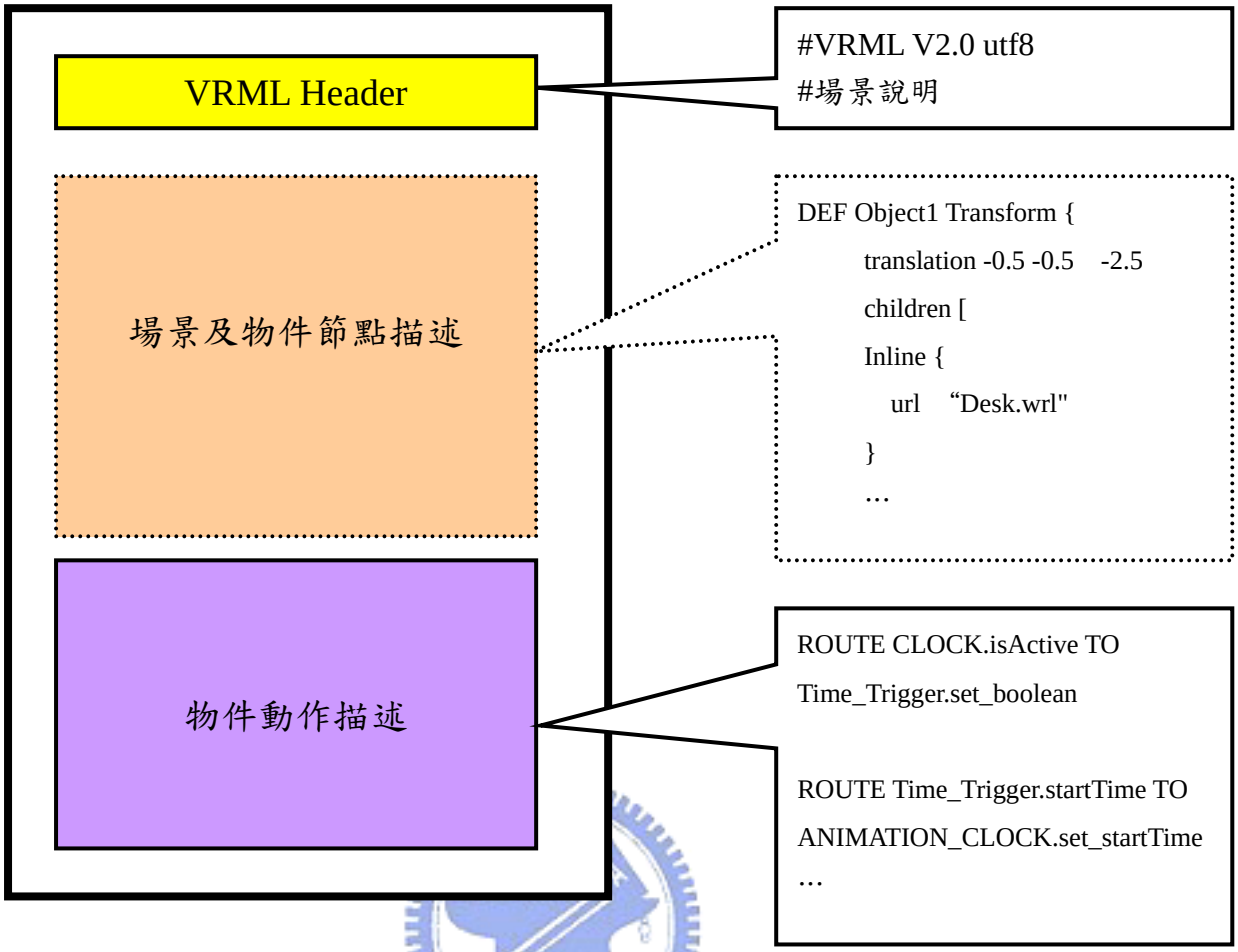

圖 17 VRML 文件檔的結構

4.3 3D 教材描述檔的檔案結構

本研究所設計的轉譯器是把 3D 教材樣版編輯雛形系統所得到的 3D 教材描述檔轉譯成 VRML，因此必須先瞭解該系統所編好的教材有那些檔案及其結構。3D 教材樣版編輯雛形系統的輸出檔有三種：

(1) F.ini：本檔為索引檔，一份教材只有一個檔，記錄著 3D 教材各畫面的名稱，及執行畫面動作順序的劇情檔檔名。一份 3D 教材有幾個畫面都須由這個檔案查詢。此處之 F 為 FileName 之簡寫，代表著某個檔案名稱，在以下(2)(3)的說明中也是這個意思，而實際的檔名則要依使用者編輯 3D 教材存檔時所用的檔名而定。

(2) FScn#.adl：這種檔案為動作順序檔，記錄著教材各畫面素材的動作順序及動作類型，也就是所謂的劇情。一份教材可能有好幾個畫面，所以就有好幾個劇情檔，此處 FScn#.adl 中之＃代表從 0 開始的數字，所以一

份教材如果有三個畫面，就有三種劇情，就有 FScn0.adl 、FScn1.adl 及 FScn2.adl 三個檔案。

(3) FScn#.ini：這種檔案為素材屬性記錄檔，記錄著教材各畫面素材的屬性，如素材的擺放位置、大小、素材來源等。一份教材如果有好幾個畫面，就會有幾個素材屬性記錄檔；此處 FScn#.ini 中之＃代表從 0 開始的數字，所以一份教材如果有三個畫面，就有 FScn0.ini、FScn1.ini 及 FScn2.ini 三個檔案。

4.3.1 F.ini 檔案分析

F.ini 檔以[Registry_Begin]之 tag 開頭，以[Registry_End]之 tag 結束，中間以[Map0]、[Map1]、[Map2]…….之 tag 區隔各畫面的名稱及劇情檔檔名等資訊，圖 18 為記錄著有兩個畫面名稱及劇情檔檔名資訊的檔案內容示例：

```
[Registry_Begin]

    [Map0]
    File = "C:\Documents and Settings\USER\桌面\FScn0.adl"
    Name = “第一頁"

    [Map1]
    File = "C:\Documents and Settings\USER\桌面\FScn1.adl"
    Name = “第二頁"

[Registry_End]
```

圖 18　F.ini 檔案內容範例

4.3.2 FScn#.adl 檔案分析

FScn#.adl 檔分成「場景資訊」及「物件動作描述」二個區段。「場景資訊」區段記錄著「素材屬性記錄檔」的檔名，以[SceneInfo]及[SceneInfo_End]之 tag 做為開頭及結尾。「物件動作描述」區段，以 [EventDescription] 及 [EventDescription_End]之 tag 做為開頭及結尾，中間以 SceneBegin：{….}區塊記錄開場時動畫劇情；以 Actor 物件名稱：{……..}區塊記錄某物件被按下時的動

畫劇情。各物件所要展示的動畫項目以「物件名.動作名稱{.... }」或「物件名.AniPlay(自訂動畫名稱,動畫展示次數值,展示位置,展示速度值) { ..}」來表示。

在動畫展示的順序方面，如果某物件Ｎ的「物件Ｎ.動作名稱{.. }」包含在另一個「物件Ｏ.動作名稱{ ..}」後面的大括弧{.....}中，則代表物件Ｏ的動作先執行後才執行物件Ｎ的動作。圖 19 為一個記錄著開場動畫及新演員 3 被按下後驅動新動畫 0 動作之檔案內容示例：

```
[EventDescription]
        SceneBegin : {
                新演員 4.AniPlay( 新動畫 1, Loop, None, 1 ){
                        新演員 3.AniPlay( 新動畫 0, Loop, None, 1 ){
                                新演員 7.AniPlay( 新動畫 0, Loop, None, 1 ){
                                }
                        }
                }
        Actor 新演員 3 : {
                新演員 3.AniPlay( 新動畫 0, Loop, None, 1 ){
                }
        }
[EventDescription_End]
```

圖 19　FScn#.adl 檔案內容範例

4.3.3　FScn#.ini 檔案分析

FScn#.ini 檔將每個素材各存成一個區塊，各以該[素材名稱]之 tag 做開頭及兼做各素材區塊之區隔。素材區塊中則以英文簡寫代名儲存各素材的屬性。各屬性的代表意義舉例說明如下：

1. objType：多媒體素材的種類，如 13 代表圖形，16 代表影音。
2. objName：多媒體素材在場景中的名稱。
3. atrPosition：多媒體素材在三維空間中的位置。
4. atrOrientation：多媒體素材在三維空間中所面對的方向。
5. atrScale：多媒體素材在三維空間中體積大小。
6. filFileName：多媒體素材來源檔案名稱。
7. visColor：可視多媒體素材的顏色。
8. txtFontface：文字多媒體素材其字型。
9. txtFontsize：文字多媒體素材的字體大小。

10. txtFontcolor：文字多媒體素材的字體顏色。
11. txtText：文字多媒體素材的文字內容。

圖 20 為記錄著某個物件的屬性檔案內容示例：

```
[新演員 3]
objType = 13
objName = "新演員 3"
atrPosition = "(-2.31738 2.0403 0 )"
atrOrientation = "(0 1 0 -0 -0 -1 )"
atrScale = "(3 3 3 )"
filFileName = "c:\論文\MyProject\ModelGallery\素材\圖片
                \GlyImg0001.bmp"
新動畫 0 = (
( T: 0    P: -2.30016 2.2186 0    Q: 0 1 0 -4.37114e-08    S: 3 3 3 )
( T: 5    P: -2.37197 2.57412 0    Q: 0 1 0 -4.37114e-08    S: 3 3 3 ))
```

圖 20　FScn#.ini 檔內容示例

4.4 系統模組功能介紹

4.4.1　ParseMap 模組

ParseMap 模組的功能為將 F.ini 檔的內容解析成如圖 21 所示的樹狀結構。各節點分別記錄著教材各畫面的劇情檔檔名或各畫面的名稱。ParseAdlTree 模組只要探尋這個樹狀結構，就可以依序取出各頁的劇情檔檔名及各頁的名稱

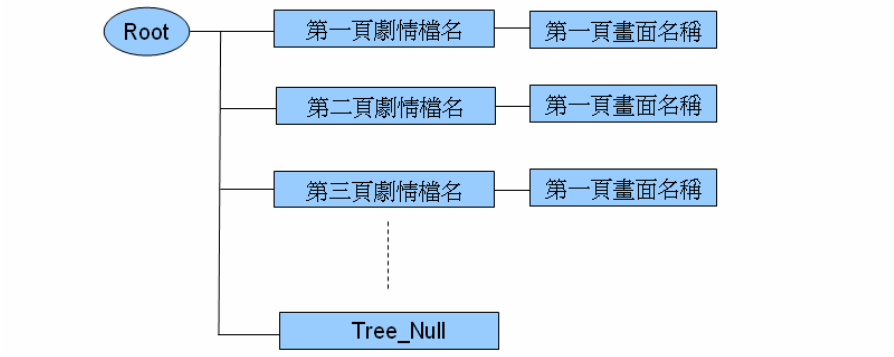

圖 21　F.ini 檔的樹狀結構圖

4.4.2 ParseAdlTree 模組

本模組的功能為將 FScn.adl 檔的內容解析成如圖 22 所示的樹狀結構。其中「場景」節點記錄著「素材屬性記錄檔」的檔名，ParseScnTree 模組可以從這個節點找到相對應的「素材屬性記錄檔」加以解析。「劇情」節點又分為「開場時」及「物件被按下」兩個子節點，分別記錄著「開場時」及「物件被按下時」各物件的動作描述。TransAction2VRML 模組只要探尋這個樹狀結構，就可以依照物件的動作描述加入相對應之 VRML 動作指令，讓物件在瀏覽器上產生正確的動作。

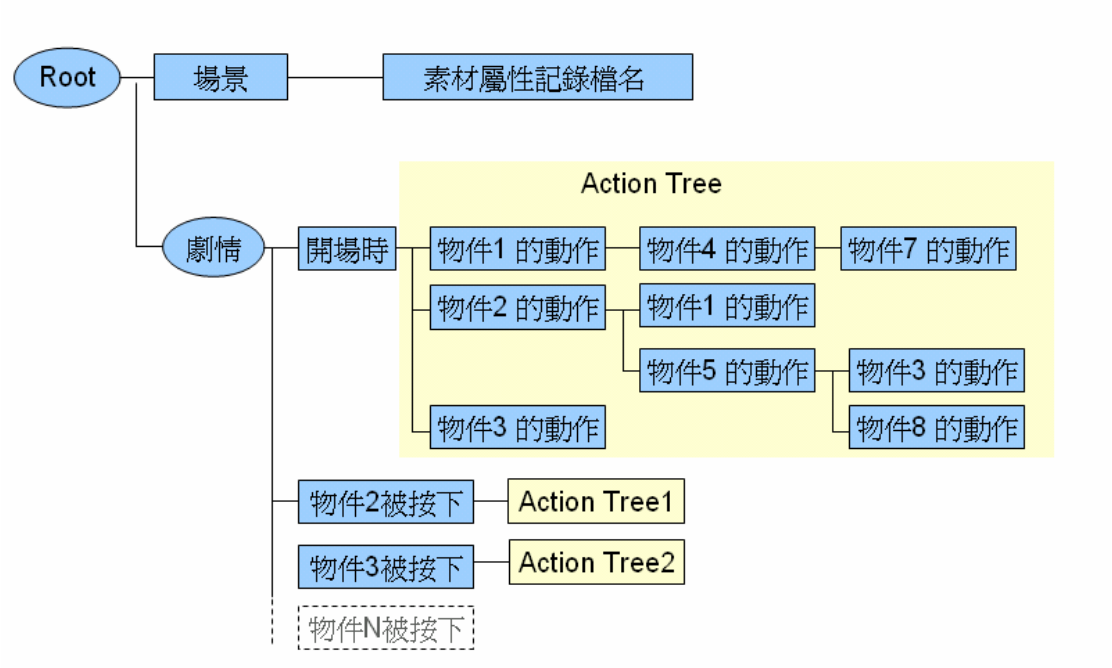

圖 22　FScn.adl 檔的樹狀結構

本模組樹狀結構的產生是用 lex,yacc[20]這兩個工具來解析 Fscn#.adl 檔而得到的，首先用 lex 將檔案中的字串切成 Token 送給 yacc，然後用 yacc 的訂出解析檔案的語法規則，以建構出圖 22 的樹狀結構。yacc 解析 Fscn#.adl 檔的語法規則如圖 23。

4.4.3 ParseScnTree 模組

本模組從 ParseAdlTree 取得 FScn#.ini 的檔名後，會將該檔的內容解析成如圖 24 所示的樹狀結構。在圖 24 中的每一個物件節點都是一個父節點，該物件

的屬性都是其下一層的子節點，惟一例外的是物件的自訂動畫屬性，因為資料的格式較多，所以需要更深一層的子節點。ParseScnTree 模組只要探尋這個樹狀結構，就可產生相對應之 VRML 節點。

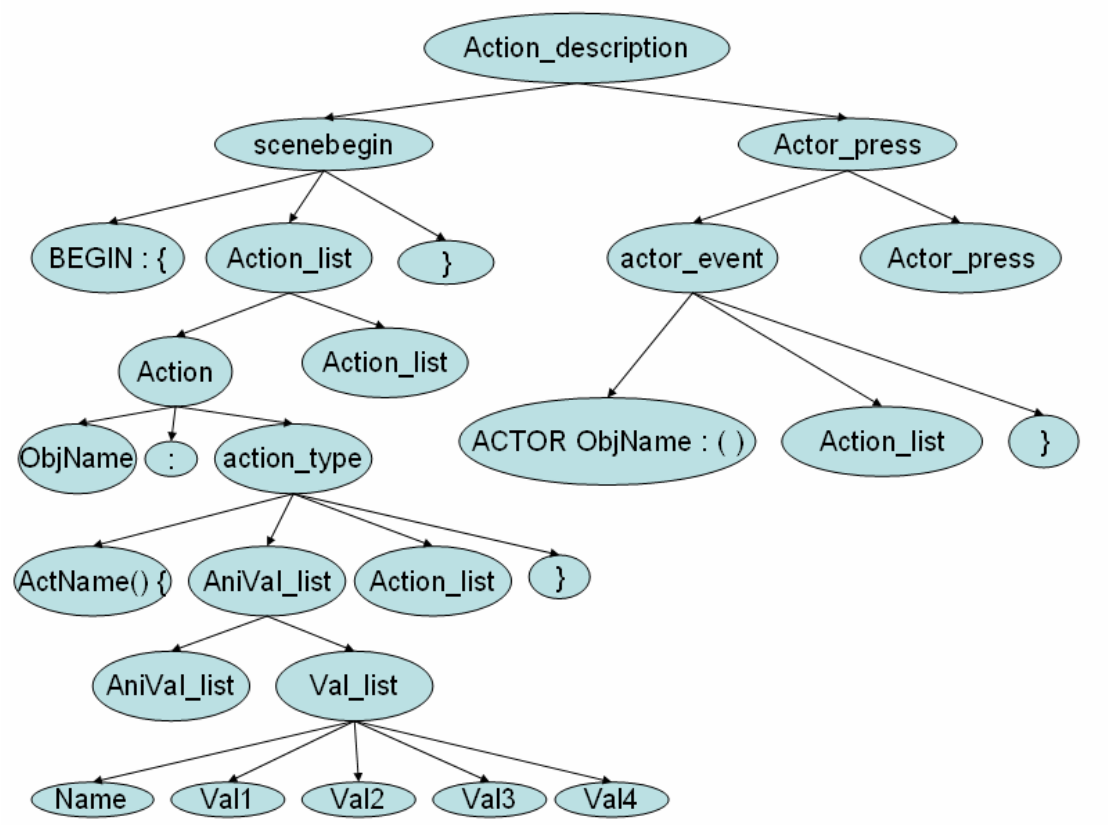

圖 23　使用 lex & yacc 解析檔案的語法規則樹狀圖

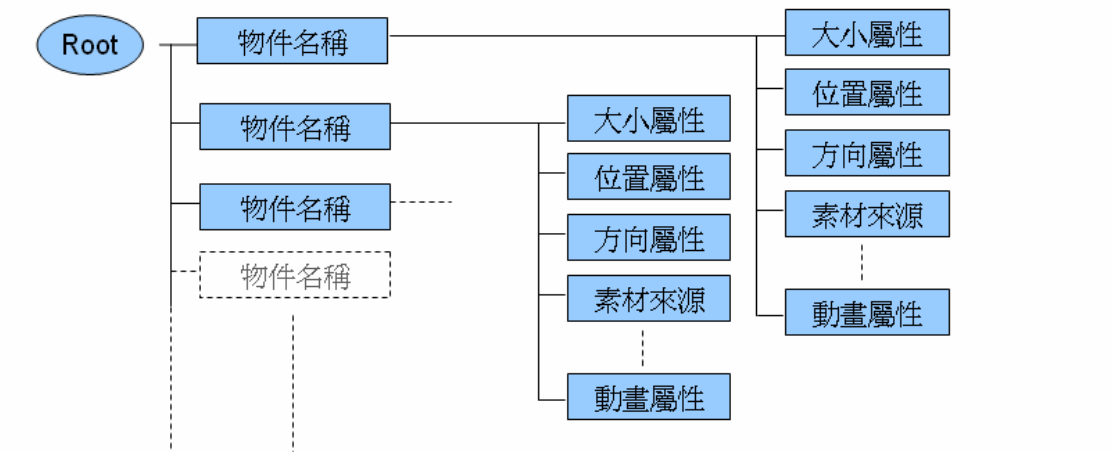

圖 24　FScn.ini 檔的樹狀結構

4.4.4 TransObj2VRML 模組

本模組的功能為將 ParseScnTree 模組所建立的物件屬性樹狀結構轉譯成為 VRML 的物件節點，如圖 25 所示是一個物件屬性樹狀結構轉成 VRML 物件節點的示例：

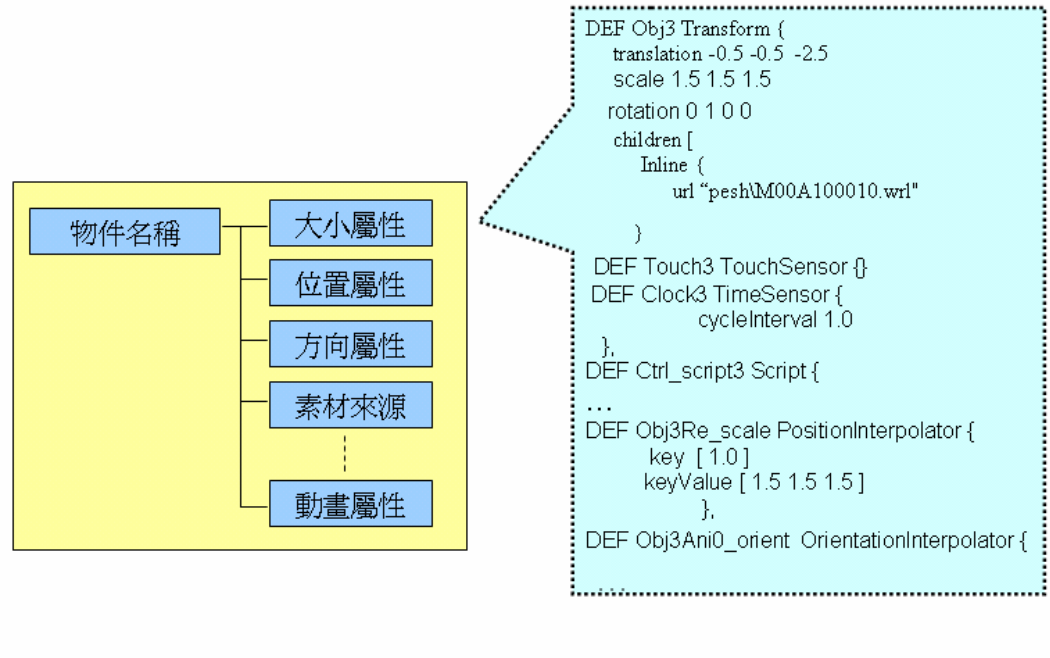

圖 25　一個物件屬性樹狀結構轉成 VRML 物件節點的示例

為了達成轉譯的過程，我們在程式中按照物件的屬性訂定了一個名叫 NodeInfor 的 structure，然後探尋圖 24 所示的物件屬性樹狀結構，先將物件都轉成 NodeInfor 型態的 structure ，然後再依照 VRML 的語法，為每個 NodeInfor 加入 TimeSensor、TouchSensor 及 Script 節點後，先以一個字串變數存起來，等到 TransAction2VRML 模組時再一併寫入附檔名為.wrl 的檔案中。在每個物件中加入 TimeSensor、TouchSensor 及 Script 節點是為了將來能控制物件動作之用。

4.4.5 TransAction2VRML 模組

本模組係用來探尋由 ParseAdlTree 模組所建立的圖 22 的樹狀結構，將物件的動作順序轉化成 VRML 的動作指令。

本模組的程式探尋邏輯是「先縱向再橫向」探訪節點。我們以圖 26 來做說明，開始時先從第一層縱向探訪節點，當探訪到「物件 3 的動作」節點時，因

為已到達縱向的尾端，所以會往橫向探訪，如圖 26 中之紅色標號 1 的路徑所示。在此處因為橫向也到達尾端了，所以返回到第一層倒數第二個節點「物件 2 的動作」處，重複「先縱向再橫向探訪節點」動作邏輯，因此產生圖 26 中之紅色標號 2 之第二條的探訪路徑，紅色標號 3、4、5、6 之探訪順序乃依此原則產生。

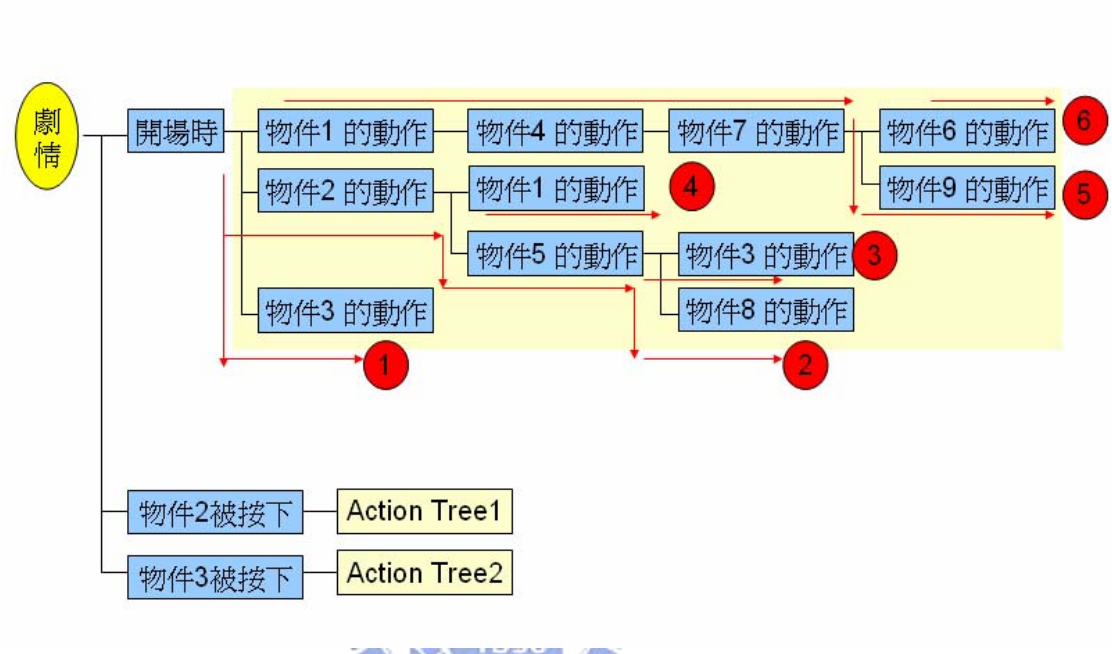

圖 26　劇情樹狀結構的動作邏輯

在探訪動作順序樹狀結構的過程中，我們就依據動作順序樹狀結構中各節點所描述的動作類型，把相對應的 VRML 的動作控制指令加入。

VRML 提供了控制時間的 TimeSensor 節點，一開場就動作及順序性的動作就由 TimeSensor 節點中的 isActive 及 clock 屬性驅動。

物件被按下才動作的感測則由 TimeSensor 節點中的 touchTime 屬性驅動。變換位置、方向、顏色等動作分別由 PositionInterpolator、OrientationInterpolator、ColorInterpolator 等動態節點中之 key/value 屬性值 set 給各物件的位置、方向、顏色等屬性來控制。

物件的動作觸發時機分為開場時及物件被按下時兩種狀況。底下我們分別說明這兩種狀況下的 VRML 動作控制指令的處理流程：

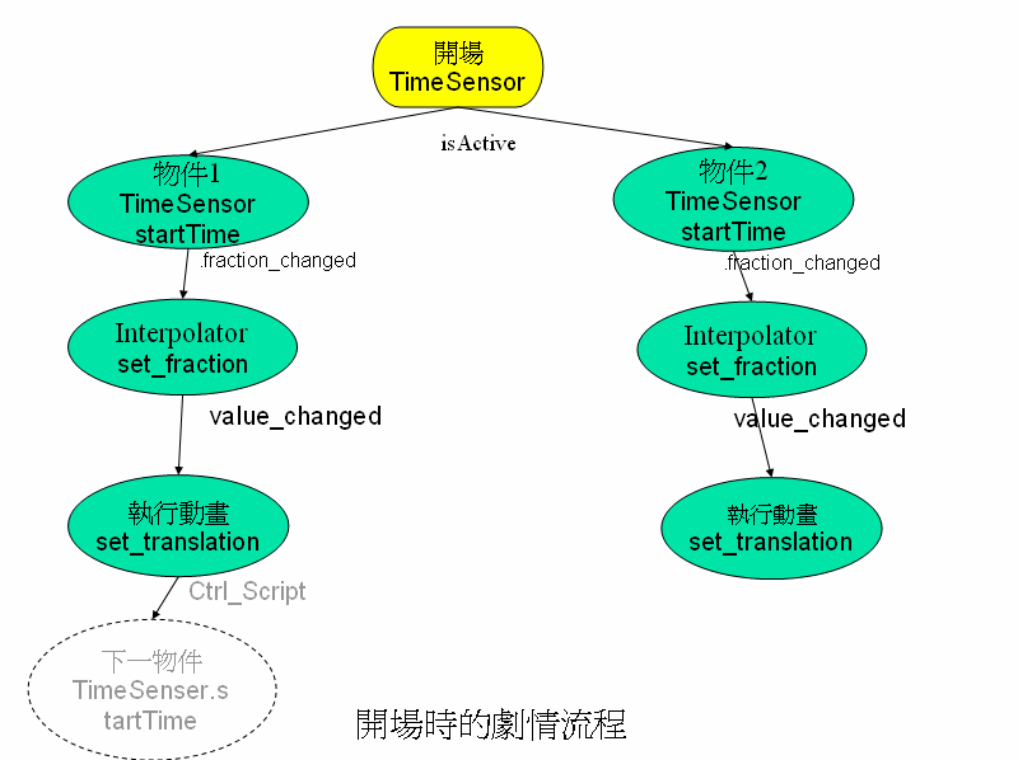

圖 27　開場時的 VRML 動作控制指令處理流程示意圖

1.開場時的 VRML 動作控制指令處理流程示意圖如圖 27 所示，說明如下：

(1) 瀏覽器會將物件的開場 TimeSensor 節點啟動，所以我們將 TimeSensor 節點的 isActive 屬性傳給即將動作物件的 TimeSensor 節點之 startTime 屬性。

(2) 即將動作的物件的 TimeSensor 之 startTime 屬性受到觸發後，TimeSensor 節點之 clock 屬性就會開始計時，所以會將 TimeSensor 之 fracton_changed 屬性傳給 Interpolator 節點之 set_fraction。Interpolator 之節點種類有 PositionInterpolator、OrientationInterpolator、ColorInterpolator…等 6 種，端視在動作順序樹狀結構中所描述的是那一種動作而定。改變位置的是 PositionInterpolator 節點，改變方向的是 OrientationInterpolator 節點，改變顏色的是 ColorInterpolator 節點。

TimeSensor 之 fracton_changed 屬性會傳送週期內的各個分割的時間點的值，Interpolator 節點以 set_fraction 屬性來接收各時間點值，然後由 Interpolator 節點中的 key/value 屬性值，以內插法計算出在目前的時間點上，某個屬性的值應該是多少，由 Interpolator 節點的 value_changed 把這個值設定給場景物件的大小、位置、方向等屬性，以改變物件目前的屬性值，物件就會產生動畫。

(3) 將 Interpolator 節點的 value_changed 傳給各物件的 set_translation 去改變物件的位置，或傳給 set_scale 去改變物件的大小，或傳給 set_rotate 使物件旋轉。至於是傳給那一個屬性，端視在動作順序樹狀結構中所描述的是那一種動作而定。

(4) 如果有下個物件要接續在本物件之後動作，則利用 JavaScript 在本物件中所建立之 ctrl_script 節點，傳送本物件的動作結束時間給下一物件的 TimeSensor，然後重複(2)至(4)的步驟。

2.「物件被按下時」的 VRML 動作控制指令處理流程說明如下：

(1) 「物件被按下時」的 VRML 動作控制指令處理流程和「開場時」的處理流程除了在觸發源不同外，其餘的處理情形都一樣。

(2) 當按下物件時，物件的 TouchSensor 會傳送 touchTime 給物件的 TimeSensor 的 startTime 屬性。

(3) 即將動作的物件的 TimeSensor 之 startTime 屬性會受到觸發，以下的處理流程均和「開場時」的處理步驟 2 至 4 相同，茲不再贅述。其 VRML 動作控制指令處理流程示意圖如圖 28 所示

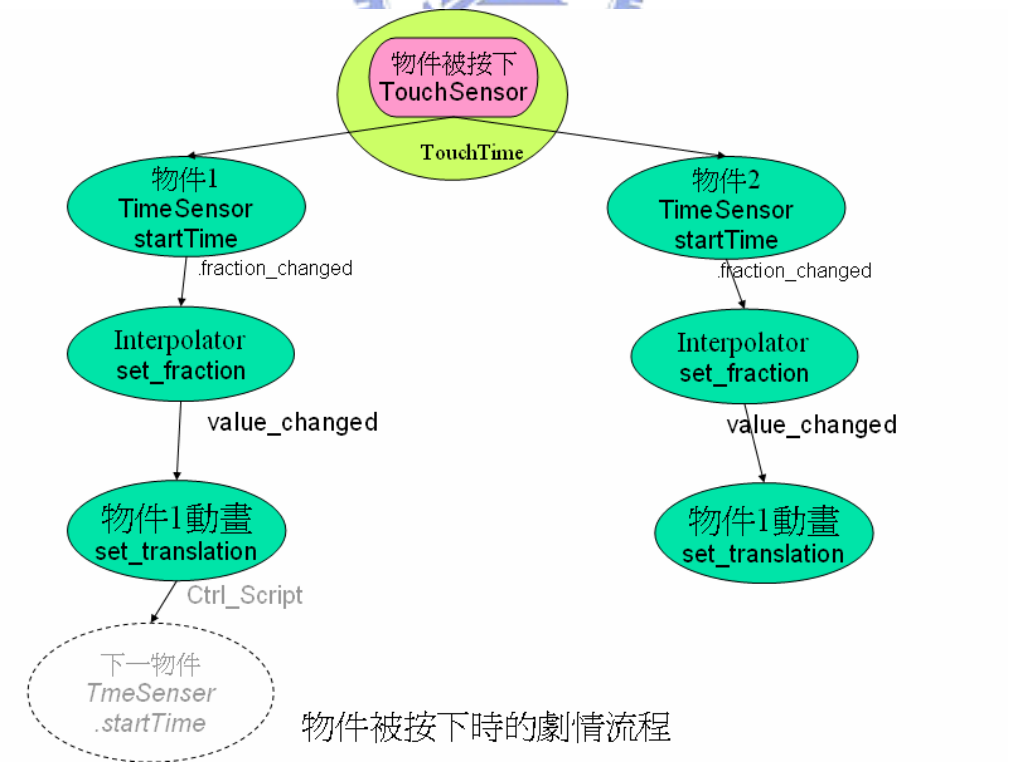

圖 28　物件被按下時的 VRML 動作控制指令處理流程示意圖

完成上述的步驟後，TransAction2VRML 會將轉換好的 VRML 動作控制指令碼及前面已轉換好的的 VRML 物件節點碼一起寫入到檔案中，圖 29 為 TransAction2VRML 模組所轉換好的 VRML 動作控制指令碼在 VRML 檔案中的寫入位置。

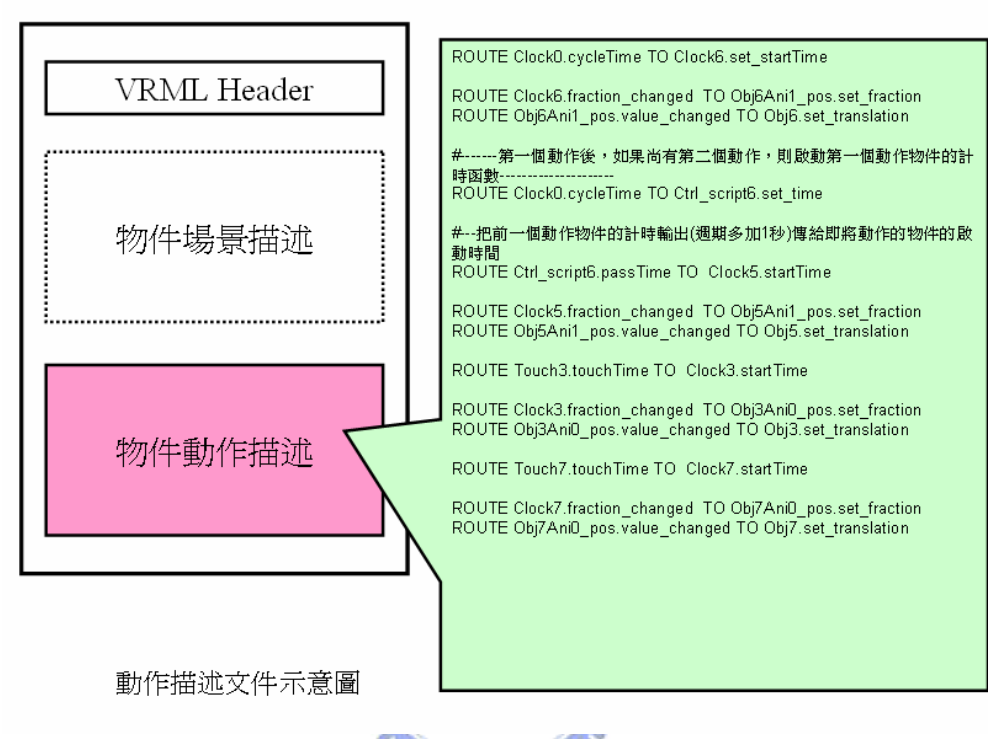

圖 29　轉譯後的動作指令在檔案中的寫入位置