第四章

場景製作

在製作飛行場景來與駕駛員產生互動上，我們仰賴場景製作的軟體來開發飛行虛擬實境場景，並使用程式設計的方式來建構擬真的環境。如此一來，虛擬實境場景便能結合電腦繪圖、立體音效與各種感知介面使駕駛員達到沉浸虛擬世界的效果。市面上的場景製作軟體種類繁多，這裡僅列出幾個較著名的商用軟體加以比較，其軟體規格列於表 4.1 之中[6,7,23]。

Superscape VRT：

是由英國 Superscape(超景)公司所研發的一套專業級的虛擬實境軟體。是一套具有和 C 語言類似的程序式個體控制語言，具有虛擬個體行爲控制、以及與玩家之間互動關係之能力。支援許多種虛擬實境的硬體周邊、可以提供比較優美的畫面、即時性互動等的功能。

WorldToolKit (WTK)：

由美國 Sense8 公司所研發的一套虛擬實境軟體，能提供一個高效率、即時 3D 圖形以及跨平臺的發展環境，其中，函式庫是由一千個以上的 C 語言函式(Function)所構成，並且所有的函式用物件導向觀念寫成，共形成了 20 幾個類別(Classes)。另外，並包含網路連線相關函式，適用多台 PC 及 UNIX 工作站連線溝通。

Division DVS：

DVS 爲英國 Division 公司所發展的系統軟體，而 dVise 爲其編寫虛擬場景的工具，dVise 是一套在 Silicon Graphics 工作站上執行的虛擬實境軟體，此套軟體的特色是玩家當某些事件發生後，可設定採取哪些動作來回應此一事件的發生。另外，dVise 也提供了簡易的行爲控制能力以加強行爲控制方面的不足。

函式庫	Superscape VRT	WorldToolKit	Division DVS
工作平臺	PC(Windows)	PC(Windows) , Sun SGI IRIX , Linux , HP DEC , PPC	PC(Windows) , Sun SGI IRIX , DEC
支援程式語言	自訂	C/C++	C/C++
支援物件模型	DXF、VRML2.0	3DS、DXF、NFF、VRL	3DS、DXF、NFF、OBJ
底層函式庫	Direct3、OpenGL	Direct3D、OpenGL	Direct3D、OpenGL
網路功能	Ethernet	Ethernet、分散式處理	Ethernet、分散式處理
3D 音效	有	有	有
優 點	1.視覺式整合編環境 2.行為控制語言	1.適用範圍廣、文件齊全 2.支援物件格式較多樣 3.採用多層次精細度模型	1.採用多層次精細度模型 2.在程式中使用 API 加以定義
缺 點	1.無法採用多層次精細度模型	1.特殊效果較少 2.不支援多重顯示	1.價格昂貴

表 4.1　Superscape VRT、WTK 與 Division DVS 之比較

在列舉的幾個商用軟體之中，由於 WTK 功能較 Superscape VRT 完備，色彩顯示也沒有像 Superscape VRT 受限於 256 色，視景能更逼真，且 WTK 價格較專業級的 Division DVS 更爲低廉，目前學術界使用 WTK 來進行研究的機構相當多；另一方面 WTK 必須搭配微軟 Visual C++ 4.02 版本以上來撰寫程式，與本研究開發環境一致。在經過各方面的評估之後，則選用 Sense8 的 WTK 作爲飛行虛擬實境場景的開發工具。

由於 WTK 等於是 C 語言的延伸應用，故具有即時性且跨平台的相容特性。另一方面 WTK 軟體也是物件導向的軟體，且函式(Function)眾多，每個函式可針對不同的功能需求應用之。而在 WTK 所有的函式裡，我們又可以分類成 20 幾個重要的類別 (Classes)，其類別分列如下：

- Universe：此一類別包含所有 WTK 之物件；
- Scene Graphic：建構 VR 場景樹狀結構的基礎類別；
- Moveable Nodes：此一類別包含位置和方面資訊；
- Geometry：此一類別爲組織和更動任何模擬基礎材料的 3D 物件；
- Polygons：此一類別提供使用者所看到物件景色的形狀；
- Materials：此一類別給與 3D 物件外觀(如亮度和顏色特質)；
- 3D Text：此一類別以 3D text strings 爲主，爲簡易的 Geometry 應用；
- Textures：此一類別爲負責地形質材貼圖；
- Tasks：此一類別可讓使用者自訂任務工作；
- Lights：此一類別主要負責光線效果；
- Sensors：此一類別負責 Input 裝置；
- Paths：此一類別負責錄取物件在 3D 空間中之移動路徑；
- Motion Links：此一類別負責與指定動作連結；
- Viewpoints：此一類別負責代表觀測位置；
- Windows：此一類別負責產生多視窗模擬；
- Adding User Interface (UI) Objects：此一類別負責新輸入介面驅動；
- Drawing Functions：負責將 2D 的繪圖程序嵌入 WTK 的應用中；
- Sound：此一類別負責音效處理；
- Networking：此一類別負責網路之間的連接；
- Serial Ports：此一類別能設定溝通 WTK 的 Serial Port；
- Portability：此一類別能使 WTK 同時執行於不同平台架構；
- Math Library：此一類別負責處理管理位置與方向的資料並運算處理之。

由以上眾多類別我們可以看出 WTK 的功能很完整，幾乎囊括需要建構虛擬實境場景的所有功能。除此之外，WTK 也和專業場景繪製軟體 Division DVS 一樣支援多層次精細度模型(LOD)物件，更能顯其實用性。以下我們便進一步介紹 WTK 場景架設的概念，並實際經由程式的撰寫架設飛行場景。

4.1 場景架設概念

WTK 所提供的場景結構爲一個有等級體系的樹狀結構圖，我們稱之爲場景圖解(Scene Graph)的樹狀架構，如圖 4.1 所示 ，其樹狀架構全都由一個一個節點(node)所構成，上層的爲母節點(Parent node)，連接在其下的稱之爲子節點(Child node)。最原始也是一開始以及最後結束的節點稱爲根節點(Root node)，而燈光節點(Light node)以及霧節點 (Fog node) 將會使往後的節點產生光和霧的效果，動作節點(Xfrom node)則會使後面所有的節點都按照它所描寫的動作移動或轉動，物件節點(Geom node)則是放置 3D 物件的節點。

而當 WTK 在描繪整個 Scene Graph 時，其執行節點順序是由樹狀圖的上到下，由左到右，且順序中先執行到的節點會影響其後面節點的變化，其執行順序如圖 4.2 所示(其流程順序照數字由小到大進行，節點順序由 A→B→C→D→E→F)。

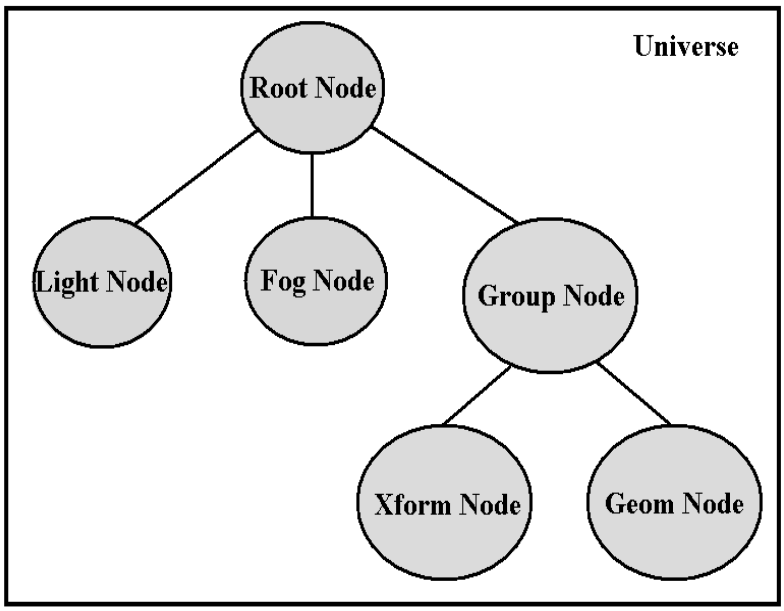


圖 4.1 場景圖解(Scene Graph)樹狀結構圖[41]

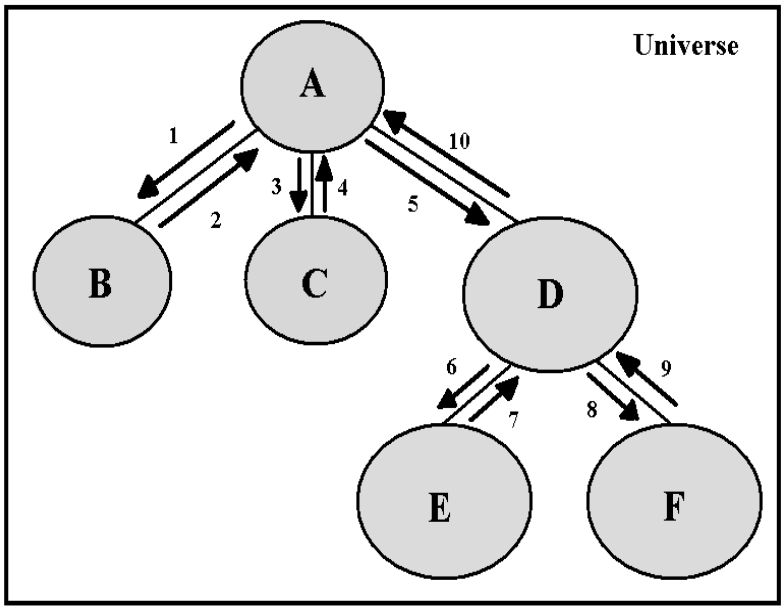


圖 4.2　WTK 描繪場景架構順序圖

由於我們先前說明節點之間的關係時,曾經談到前面節點的設定會影響到後面的節點，然而在實際架設場景時複雜度絕對超過圖 4.1 的結構，其不同 Group node 之間甚至可能不允許其動作互相干擾，所以在實際架設樹狀結構圖時我們必須使用其他功能的節點來處理複雜一點的架構。

我們以所設計樹狀結構圖爲例子來更進一步說明其節點之間的關係，其結構如圖 4.3 所示，根節點(Root node)、燈光節點(Light node)以及霧節點(Fog node)與先前說明一致，接下來地形的 Group node 底下有分海洋物件節點(Geom node)與天空物件節點(Geom node)兩個地形物件，但在天空物件節點之上有多定義一個分離節點(Sep node)，而設置分離節點之主要目的就是爲了隔離天空的動作節點(Xform node)使其不對另一個 Group node 的物件動作造成影響。在此實例中，天空的部份動作(例如雲層的移動動態)只會在地形 Group node 裡改變，天空的動態並不會影響到另一個飛機 Group node 內的動作，直接說明就是天空的動態無法影響飛機姿態的改變，而飛機姿態的改變是直接受到飛機動作節點(Xform node)的影響，其樹狀結構的執行順序如圖 4.4 所示。

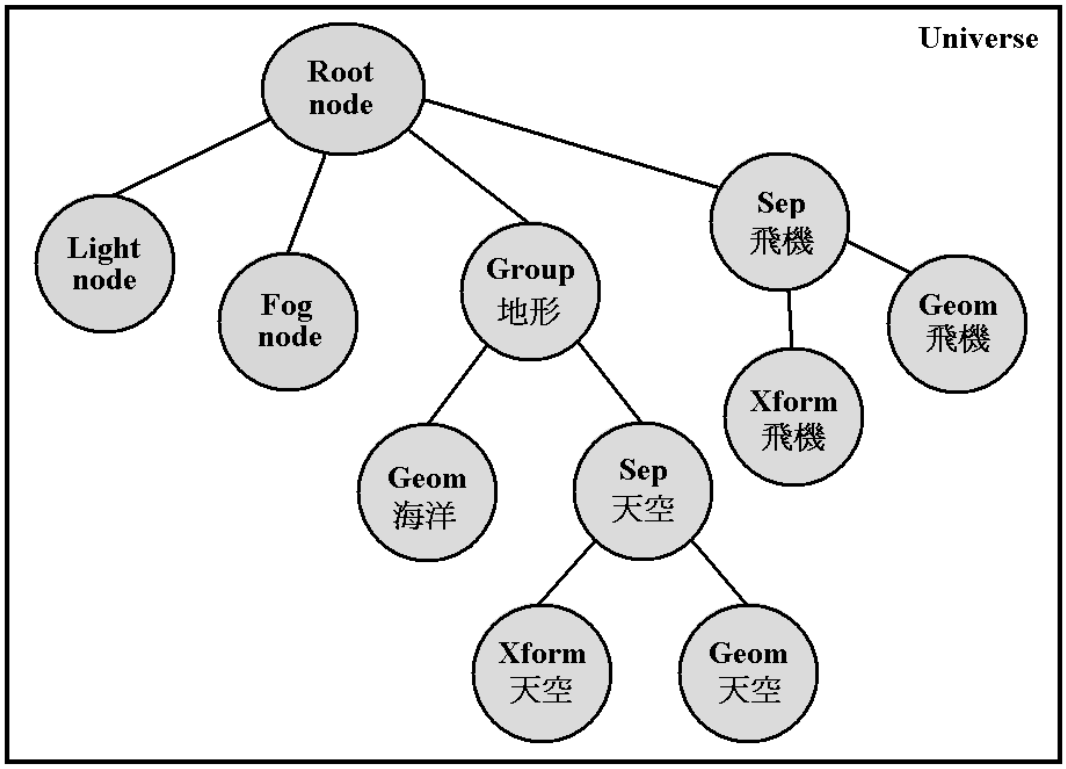


圖 4.3　飛行場景 Scene Graph 樹狀圖

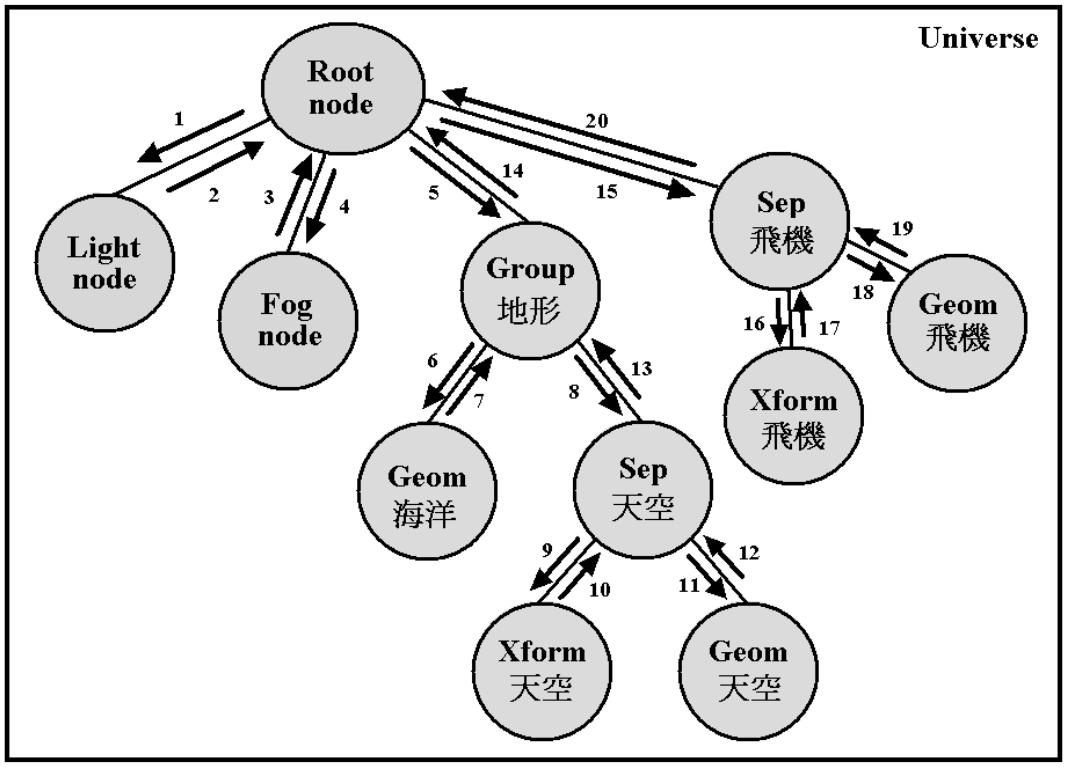


圖 4.4　飛行場景描繪架構順序圖

模擬迴圈是應用 WTK 製作場景的中心要點，模擬的每個物件項目皆在這個 Universe 中發生。WTK 所造成的 3D 動畫效果是用很多畫面(Frame)堆疊而成的，而每一個畫面皆有其內定的處理流程，當所有的處理流程在呼叫 *WTuniverse_go* 時則進入迴圈；當處理流程在呼叫 *WTuniverse_stop* 時則離開迴圈。另外，你也可以使用另一個函式 *WTuniverse_go1* 讓程式自動的跑一次迴圈後離開回圈，此 WTK 模擬迴圈流程圖如圖 4.5 所示[41]：

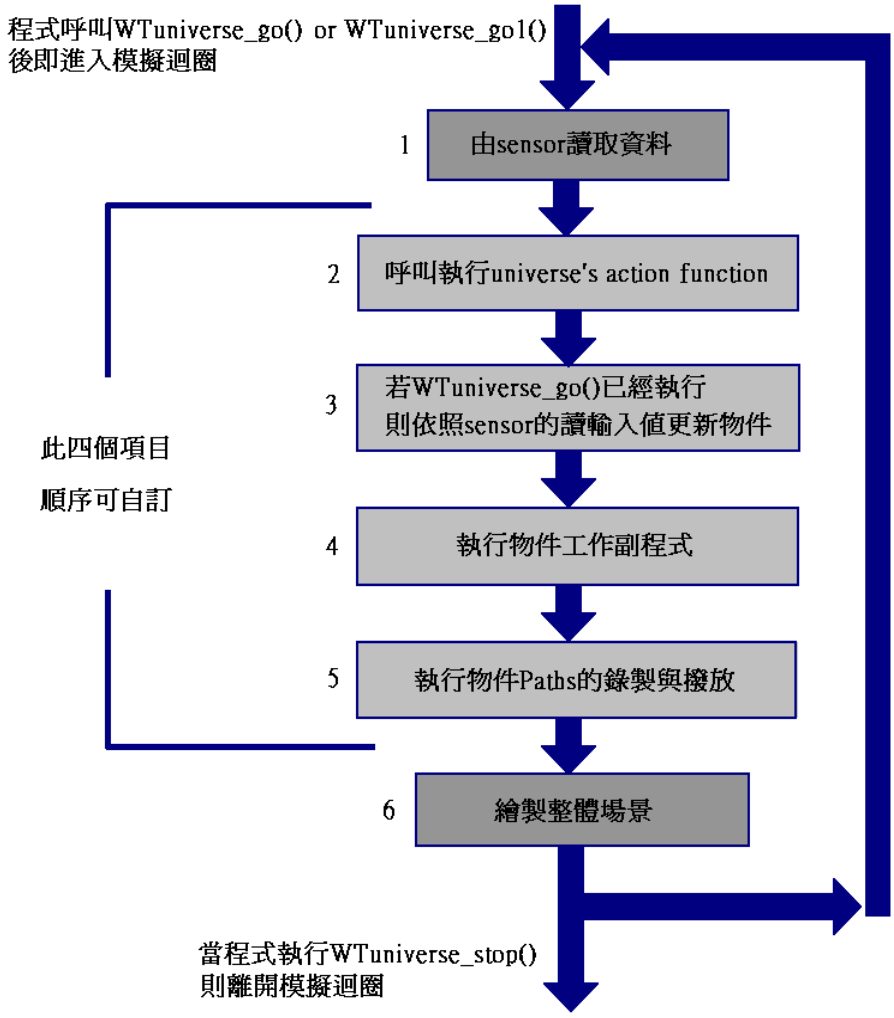


圖 4.5　WTK 模擬迴路流程圖

由上圖我們可發現，其中有四項(2、3、4、5)流程順序是可以變更的，程式設計者可以使用WTK裡面的函式(*WTuniverse_seteventorder*)更改其流程順序。

4.2 場景設計

在開發虛擬實境場景時，首先要繪製場景的地形，以此研究主題來說也就是適合飛行的航空測試場景，本研究的場景以 3D 模型物件與一般貼圖技巧來實現，在建構場景和幾何實體時，所選擇實體 polygon 數目的多寡不但會影響實體表面的外觀而且還會直接影響到場景執行的流暢度，例如秒格數(Frame Rate)的數值等。WTK 所建構的場景支援多物件檔案格式，如 DXF、OBJ、3DS、NFF、FLT…等的格式，而本研究建模方面以 NFF 搭配 3DS 格式來開發場景模組。

4.2.1 場景主程式流程

本研究所設計之飛行場景的主程式流程如圖 4.6 所示，而在執行主程式流程圖之架構前，我們必須引入 WTK 的標頭檔，這樣才可啓用 WTK 的函式庫，往後所有場景程式撰寫皆需使用到 WTK 函式庫。

在主程式流程中，我們分別規劃兩組副程式來處理主程式之重要程序，一是環境初始化，二是場景圖解樹狀結構設定。環境初始化在本流程之中我們以副程式 INITIAL_SCENE( )來負責，而場景圖解樹狀結構設定在本流程之中我們另外以副程式 SCENE_GRAPH( )來負責，最後則由主程式分別呼叫之以完成場景流覽程序。關於處理環境初始化的 INITIAL_SCENE( )副程式之流程圖如圖 4.7 所示，以下爲副程式流程之重點說明。

1. INITIAL_SCENE( )副程式開始執行；
2. 初始化整個場景，WTuniverse_new( )函式必須寫在整個程式最開頭位置，故在主程式中先呼叫 INITIAL_SCENE( )副程式。其有兩個參數需設定，一個爲 DISPLAY，另一個爲 WINDOW，皆用內定值；
3. 對模擬迴圈內的事件定義其行爲，使用函式爲 WTuniverse_setactions( )；

4. 視點重置，使用函式爲 WTwindow_zoomviewpoint( )；
5. 初始 3D 座標位置，通常爲設定受控物件之初始座標；
6. 設定音效，包括背景音效、事件音效等，使用到的函式有 WTsounddevice_open( )、WTsounddevice_setparam( )、WTsounddevice_setlistener( )、WTsound_load( )、WTsound_play( )等；
7. 在 Universe 之中抓取整個系統的 Root Node，而 Root node 與其底下場景樹狀結構設定皆在副程式 SCENE_GRAPH( )之中描寫，其使用到的函式爲 WTuniverse_getrootnodes( )；
8. 開啓鍵盤功能與設定滑鼠之靈敏度；
9. 結束 INITIAL_SCENE( )副程式。

關於處理場景圖解樹狀結構設定的 SCENE_GRAPH( )副程式，其程式撰寫之流程順序便直接按照圖 4.4 所示，以 Root Node 爲場景樹狀結構之初始節點，並依照場景更新的順序設定燈光、霧、地形與飛機在樹狀結構之位置與相關設定，以完成整體樹狀圖的架構。例如本研究對於"霧"之效果設計如圖 4.8 所示，除了宣告 Root Node 底下有存在霧節點之外，還必須對其霧顯現的模式與範圍作深入設定，在此本研究設定霧之呈現模式爲 Exponential Squared 的形式，其特徵爲視點越深感受霧之濃度以指數倍率增加，其霧化效果如圖 4.8(b) 所示；另外飛機物件設定也如同一般節點，除了宣告 Root Node 底下有存在飛機節點之外，還必須對其飛機動作的描述作深入設定，例如配合相關的飛機運動方程式與氣動力的描述使其動作擬真，最後再使用分離節點隔離，使其他節點之動作無法影響飛機節點的運作，其本研究所使用之飛機 3D 物件如圖 4.9 所示，其中 4.9(a) 爲完整 3D 物件模型而 4.9(b) 爲物件局部放大圖。

而主程式之後半段經函式 WTuniverse_go( )的執行則進入場景模擬迴圈流程，此時場景便顯現以及持續更新前景，直到執行 WTuniverse_stop 才會離開主程式。

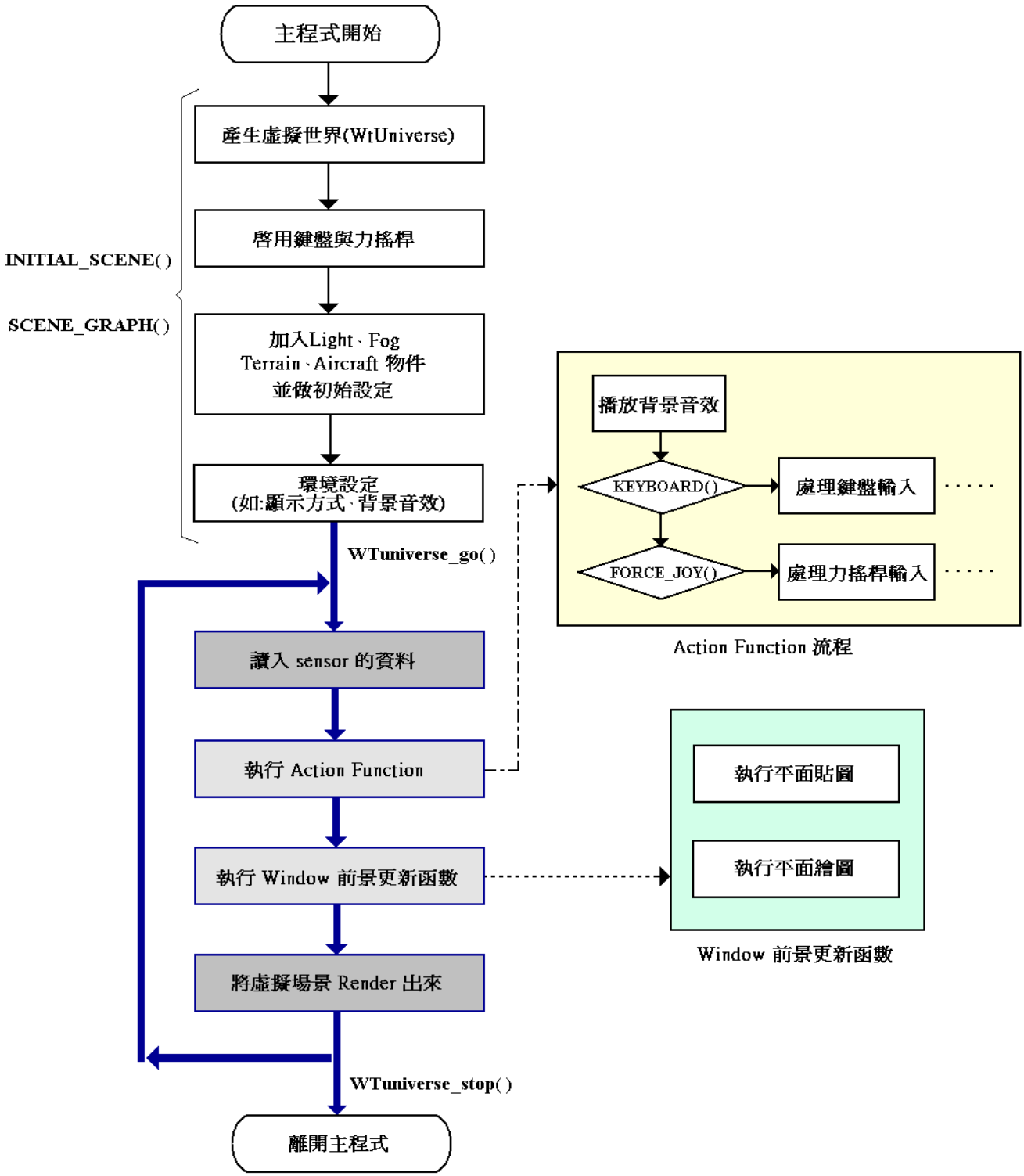


圖 4.6　飛行場景主程式流程圖

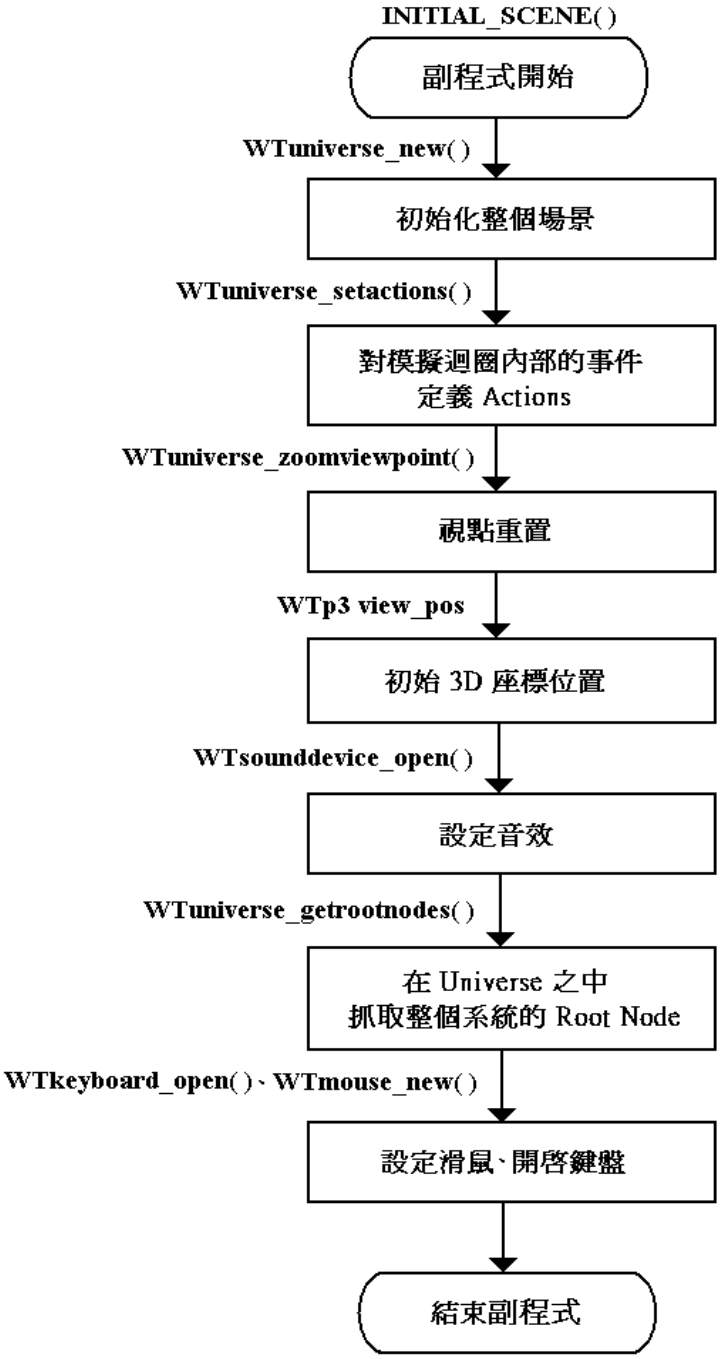
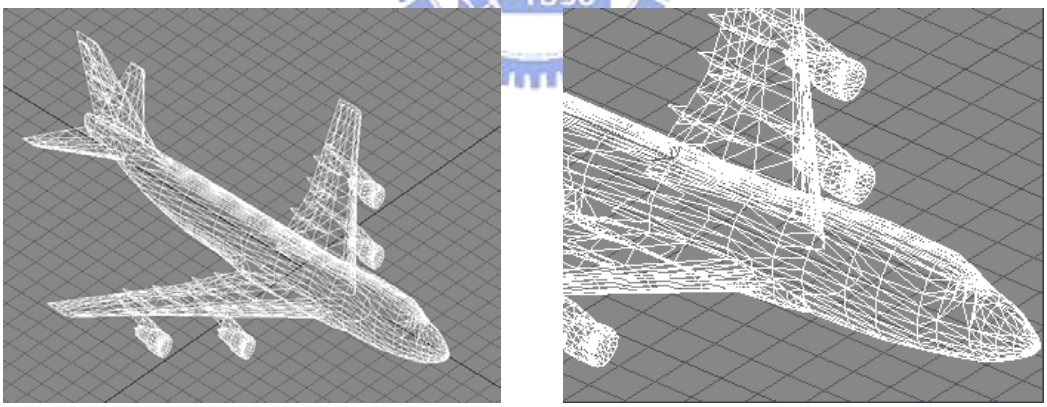


圖 4.7　主程式初始化流程圖

(a)未加入霧化處理之場景　　(b)加入霧化處理後之場景

圖 4.8　場景霧化處理比較：(a)未加入霧化處理之場景圖和(b)加入霧化處理後之場景圖。



(a)**Boeing 747** 之 **3D** 物件　　(b)前圖物件之局部放大圖

圖 4.9　**3D** 飛機物件模組：(a)**Boeing 747** 之 **3D** 物件圖和 (b)前圖物件之局部放大圖。

4.2.2 鍵盤功能與力搖桿操作設定

由於在主程式模擬迴圈之中場景需要經由飛行動態的運算才能不斷的更新其畫面資料，而將訊息送入飛行動態區塊的裝置即爲鍵盤以及力搖桿，其鍵盤與力搖桿之控制程式流程如圖 4.10 所示，說明如下。

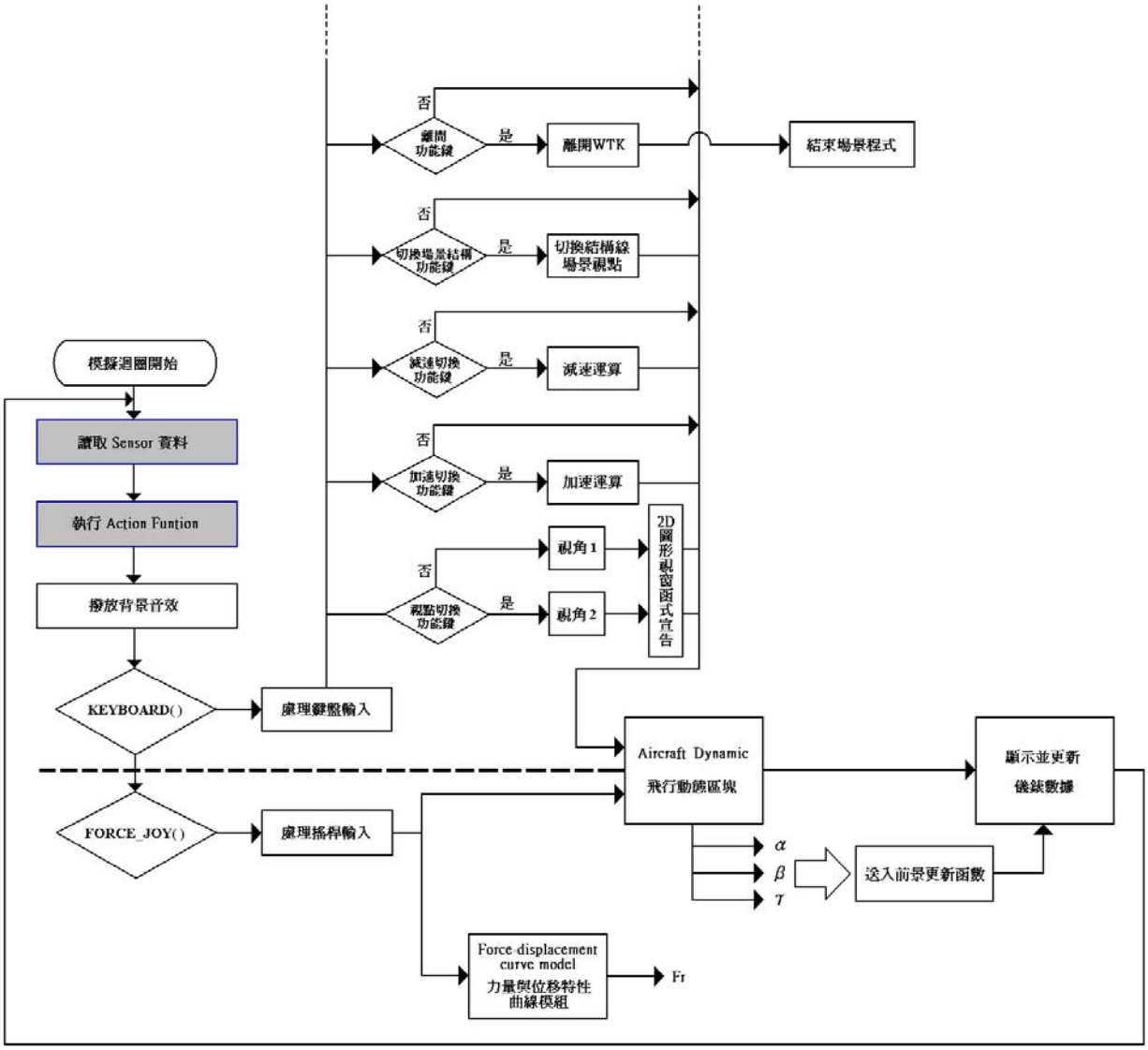


圖 4.10　鍵盤控制與力搖桿控制程式流程圖

首先說明力搖桿控制副程式流程，程式一開始先處理搖桿的輸入訊號，即爲力回饋端所送出之位置命令，經由場景端接收之後便將其位置命令分別送入力量-位移特性曲線運算區塊以及飛行動態區塊之中作進一步處理。然而經過力量-位移特性曲線區塊運算後即爲力感模型之力量命令(Fr)，而經由飛行動態區塊運算所得到之資訊即爲飛機在場景中所改變姿態之訊息，最後並將其訊息送入前景更新函數之中進而改變飛行儀錶之圖形介面與顯示數值。

在空間描述方面，由於在場景之中空間座標定義與一般座標方向稍有不同，故在描述其受控物件初始座標位置與操控動作時必須將此情況考慮進去。而場景座標定義如圖 4.11 所示，操作者面對螢幕之左右方向爲 X 軸，面對螢幕之上下方向爲 Y 軸，而面對與遠離螢幕之方向定義爲 Z 軸。

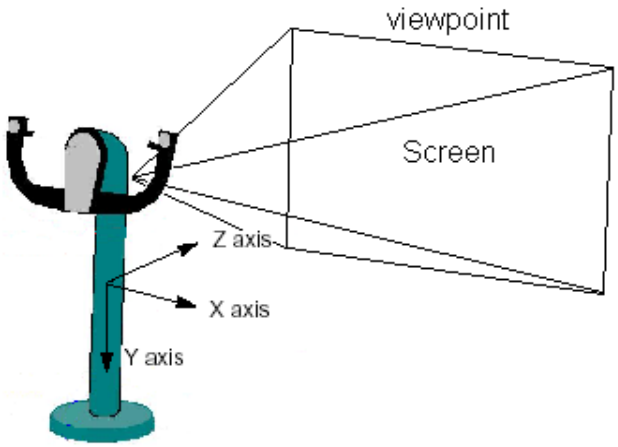


圖 4.11　力回饋搖桿在 WTK 的 3D 座標定義

而在飛機運動撰寫過程中，比較需要注意的是飛機的動作必須以飛行力學的角度撰寫。因爲在飛行力學之中，Lateral (側向運動)就已經包括了 Roll 和 Yaw 的動作；但在空氣動力學之中對飛機氣動力的穩定性討論就把 Lateral (roll) 和 Directional (yaw) 分開來討論。故我們要描寫側向飛行的動作，是因飛機的運動慣性而產生的 inertial coupling 動作，如圖 4.12 所示，當飛機做左測向與右測向飛行時便已包含了 Roll 與 Yaw 的耦合動作。左側向飛行與右側向飛行之後在本論文中我們統稱爲滾轉運動，其飛機之運動方程式與氣動力描述請參考附錄一。

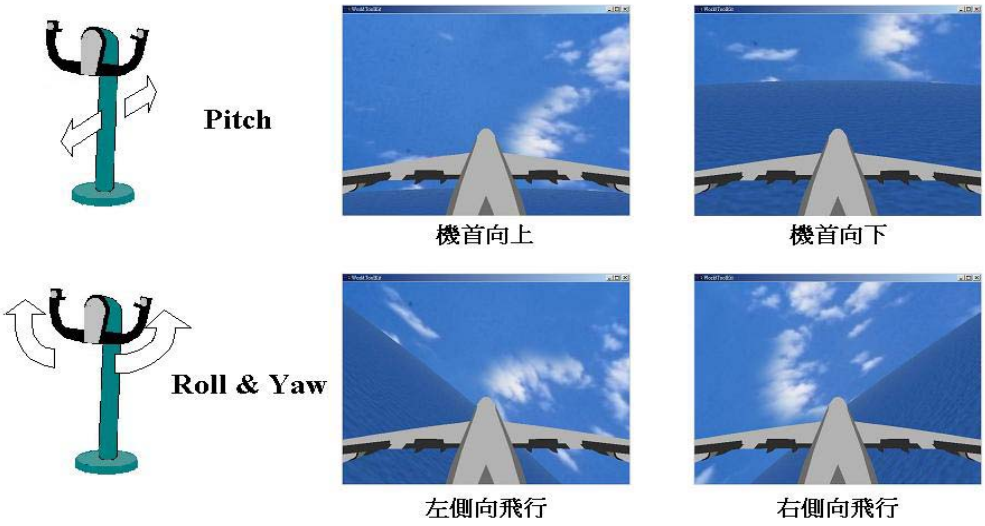
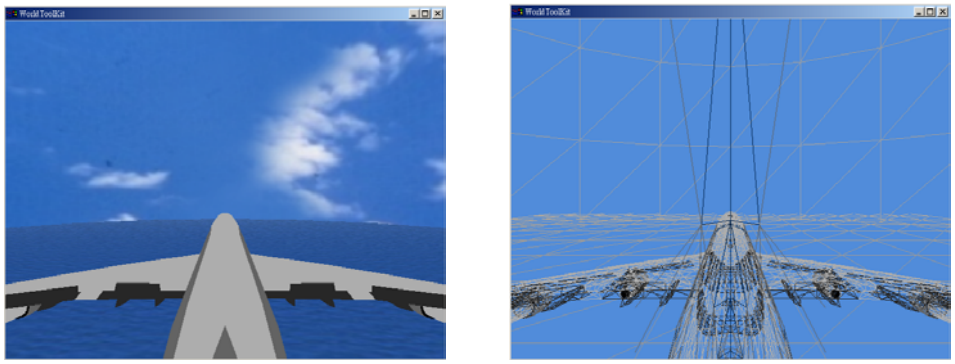


圖 4.12　飛機物件於場景之姿態與動作

另外在鍵盤控制副程式流程方面，一開始則處理鍵盤之輸入訊號，如圖 4.10 之程式流程所示，我們可以在程式之中規劃使用哪些按鍵作爲切換控制並定義其相對應之事件發生情況。在本場景之中我們定義幾項事件切換功能，例如：場景視角選擇功能、飛機巡航加速功能、飛機巡航減速功能、結構線場景切換功能、離開 WTK 功能鍵等，我們只要在程式之中引用 WTK 的相關功能函式便可達到其目的。關於場景視角選擇之效果如圖 4.13 所示，我們能在鍵盤按鍵之中指定一視角切換功能，此功能能使操作者感受不同飛行視角；同樣的方式也能應用於結構線場景切換功能，其效果如圖 4.14 所示。

(a)座艙視角　　(b)飛機視角

圖 4.13　切換視角功能：(a)座艙視角圖和(b)飛機視角圖。



(a)一般真實場景　　(b)結構線場景

圖 4.14　結構線場景切換功能：(a)一般真實場景和(b)結構線場景圖

4.2.3　飛行儀錶設計

自從飛機發明至今，雖然儀表的數目、功能不斷的更新與增加，並且將其加入機艙內顯示，以便利飛行員得知飛機飛航的姿態與狀態，但是最爲重要的儀錶不外乎下面四種：

1.空速計：目前飛行速度多少，是否接近失速都是看此儀表。

2.高度計：目前飛行高度，是否有撞上障礙物的危險。

3.姿態儀：通常以圓球或是飛機正視簡圖代表，尤其是在無參考座標的空間時，經由姿態儀則可以得知目前飛行的姿態。

4.羅盤儀：指出目前飛行方向，如果飛行中並未另設導航參考指標，就只好靠羅盤指示方位了。

本研究場景的儀錶設計就以此四種基本的重要儀錶來作爲代表，其儀錶的程式設計流程如圖 4.15 所示，由於儀表畫面更新與數值變化是由搖桿與鍵盤將命令輸入飛行動態與前景更新函數之後所產生的更新值，再經過方程式運算處理後進而換算成儀表該轉的角度與該變化之位置，最後並配合座標移動的公式計算便能經由程式撰寫出適合本場景之儀錶，如圖 4.16 所示。

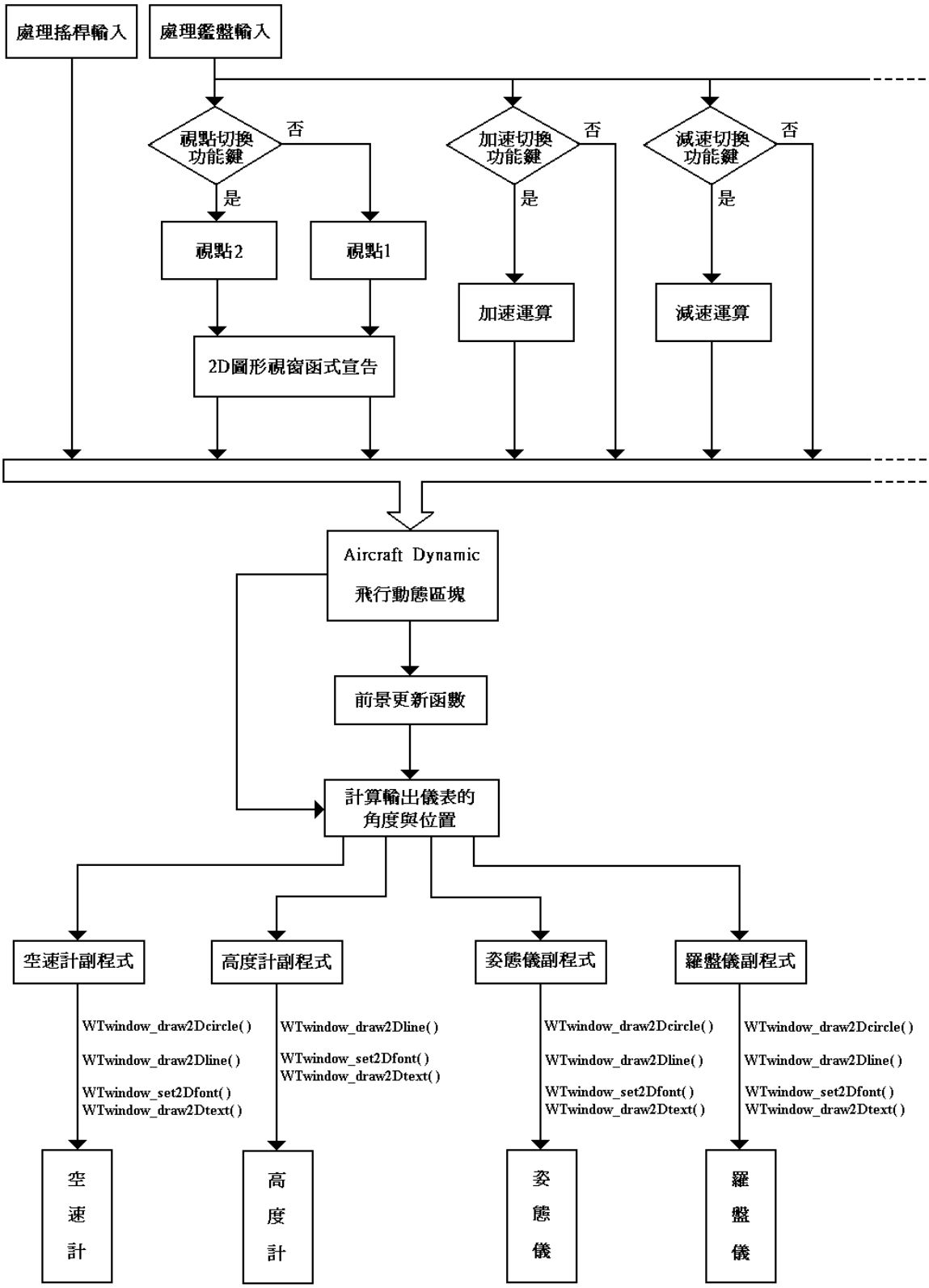


圖4.15　儀錶程式設計流程圖

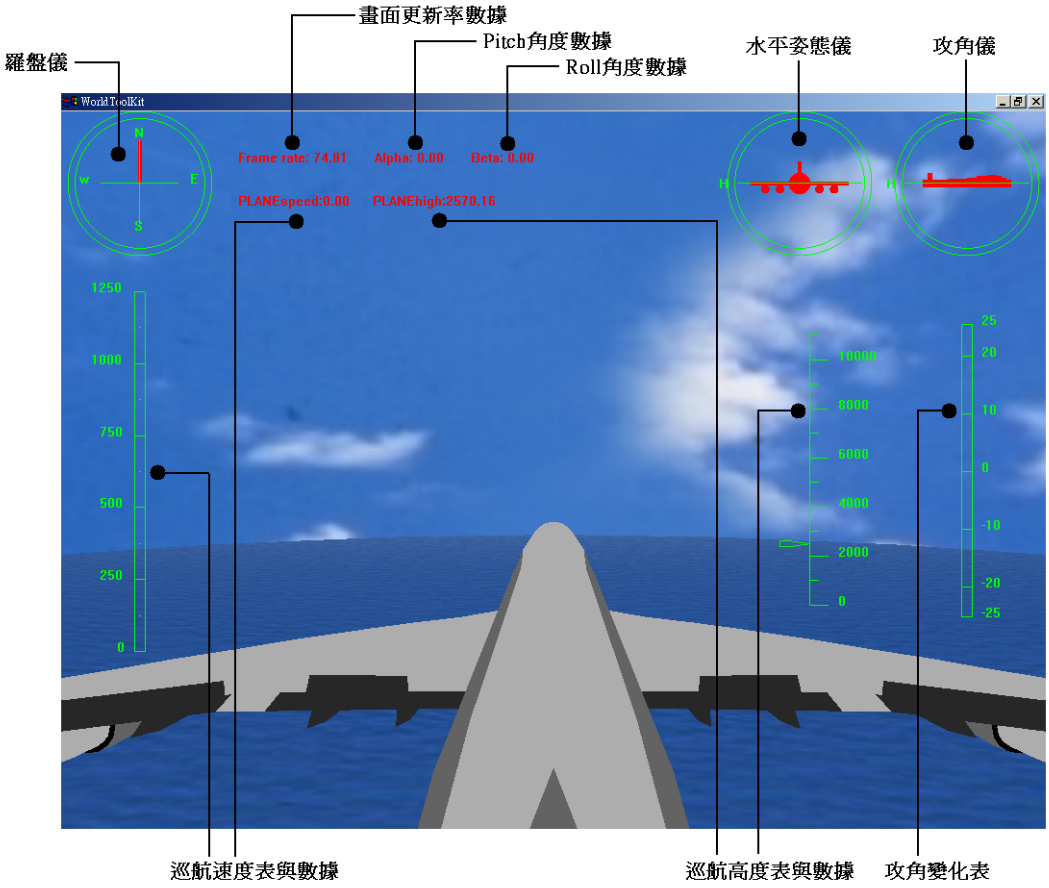
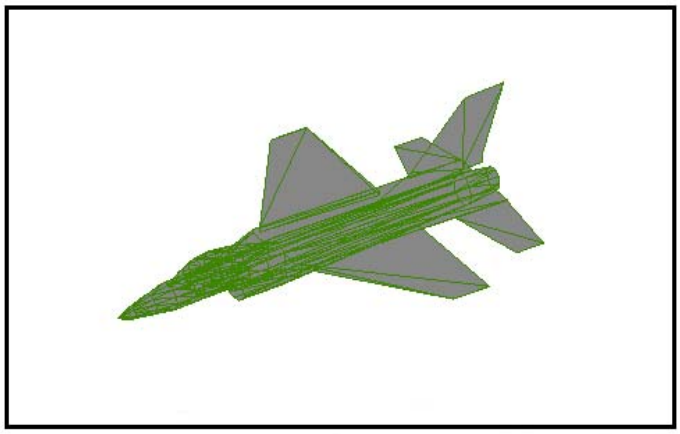


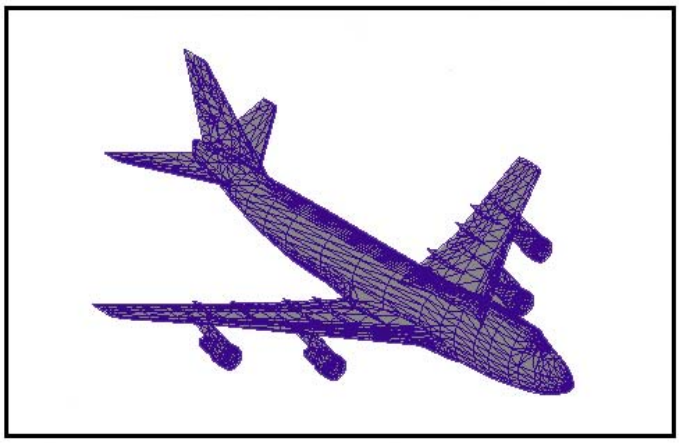
圖 4.16　飛行場景儀表功能圖

4.3　場景複雜度與動態畫面品質

一般在虛擬場景之中，若是要使場景的真實度更加逼真(也就是場景複雜度越高)，則必須要用較多的 polygons 數來使其物件更加美化真實。舉例來說，讓我們來比較兩架不同飛機的 3D 模型，其複雜度也隨著不同的 polygons 數而有所不同，一般來說 polygons 數越多之模型其物件細緻度越高[7]，如圖 4.17 所示。



(a) F-16 (總共 592 Polygons)



(b) 747-400 (總共 5706 Polygons)

圖 4.17　飛機之 3D 物件比較圖：(a)低複雜度模型(F-16 共 592 polygons)和 (b)高複雜度模型(Boeing-747 共 5706 polygons)。

在圖 4.17 (a) 的 F-16 物件只有 592 polygons，很明顯的可以看出此模型比較簡單且不細緻，對於細部的描述因 polygons 數目不高則無法呈現，反觀圖 4.17(b) 的 Boeing 747-400 物件總共有 5706 polygons，除了外觀方面一眼就能分辨更爲真實外，連細部部份的修飾也比 F-16 更爲明顯。但是當 polygons 越多真實性越高時，則代表物件本身的 frame rate 也會比較低。一旦將 polygons 較多的物件匯入場景之中時，VR 模擬系統的性能也會隨之變差。

本研究環境的場景複雜度與 frame rate 之關係如圖 4.18 所示，其量測的方式爲漸次增加場景畫面之 polygons 數目，並且紀錄其所對應 frame rate 平均值，之後再將其變化值畫成曲線表示。而我們可從圖中得知，當場景複雜度提高時其 frame rate 數也隨之減少；反之，當場景複雜度降低時其 frame rate 數也隨之增加，故其兩者之關係成一條反比曲線。

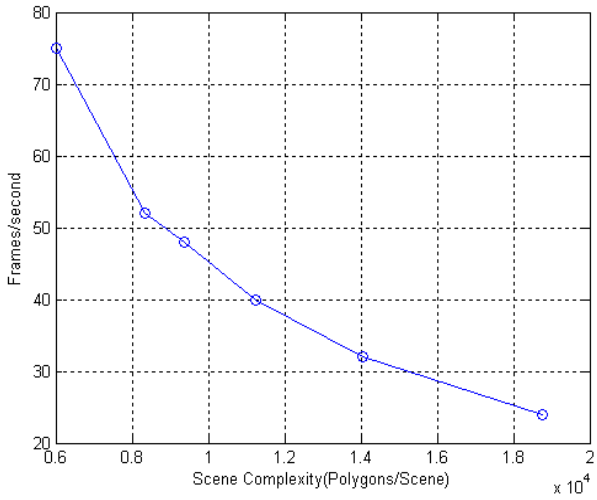


圖 4.18　場景複雜度與 frame rate 關係曲線圖

故爲了使所架構的飛行場景能維持住最佳的更新頻率且不造成 PC 負載過大而導致 frame rate 降低，除了選用 3D 加速卡來處理視訊外，並將場景分爲畫面較單純的天空模組與複雜度較多的地形模組，並分開運用策略處理。例如以少變化的天空模組佔據大部份之場景畫面，再以小部份的場景空間貼上地形並製造飛行的距離感，如此一來，雖然本研究採用的 3D 飛機物件是屬於較複雜之模型(Boeing-747 模型)，但由於我們使用策略將場景複雜度限制在一定的上限值，所以場景之 frame rate 也能如期地維持在 30 以上。