第三章

系統總覽

在介紹了運動計畫及虛擬角色設計的相關研究之後，接下來將對本論文所提出的運動模組進行初步的介紹。在這一章裡，3.1 節是問題描述，我們將會詳細說明所要處理的問題。之後的 3.2 節則是系統概述，將對這個運動模組的系統架構進行簡要的介紹。包括我們提出這種架構所採用的設計構想，以及系統中各個執行流程的功能及運作說明。

3.1. 問題描述

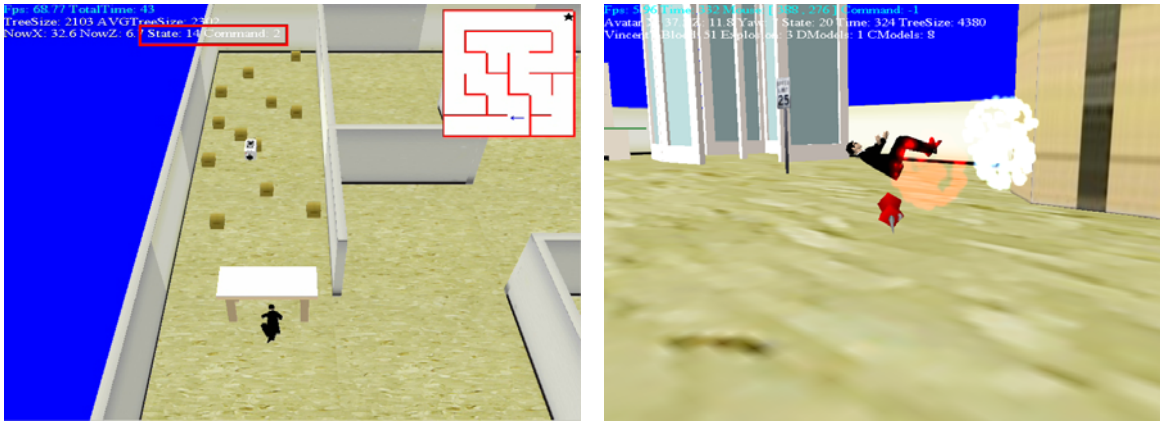
3.1.1. 整體問題描述

本論文的目的是要在 3D 虛擬環境裡，設計出一個可以和使用者進行互動的智慧型角色運動模組。在這個運動模組裡，使用者不需要給角色很多複雜的動作要求指示，而是只需要輸入一些簡單的操作命令組合，此運動模組便能根據輸入的角色動作資訊及目前的場景狀況，即時規劃出一連串符合使用者要求的角色組態。例如在一般的 3D 角色操作系統裡，使用者只須要下達前進、後退、往右和往左等各種相關的命令，系統便能從角色的可能動作裡，搜尋出跳躍、蹲下或側翻等能符合使用者操作要求且又能閃避過場景中障礙物的角色動作。

3.1.2. 場景定義

- **工作空間**

在運動計畫的問題中，場景通常會視應用的需求而被定義為 2D 或 3D 的工作空間。例如在一般的人群模擬研究中[11][20]，由於計算量非常龐大，程式設計者為了增加工作的執行效率，場景都會採用 2D 的工作空間來定義。而在抓取物體的研究裡[19]，因為此應用要求比較高的碰撞精確度，所以場景通常會採用 3D 的工作空間來定義。在我們的研究裡，為了讓角色能有著比較細緻的反應能力，我們因此採用 3D 的工作空間來定義場景。這種定義方式讓我們所設計出來的角色，可以用跳躍、蹲下或側翻等驚險的動作來閃避場景中的障礙物(如圖 3.1)。



(a) 蹲下穿越桌子　　(b) 後翻閃過飛彈

圖 3.1：角色閃避場景中的障礙物

- **場景的可變動性**

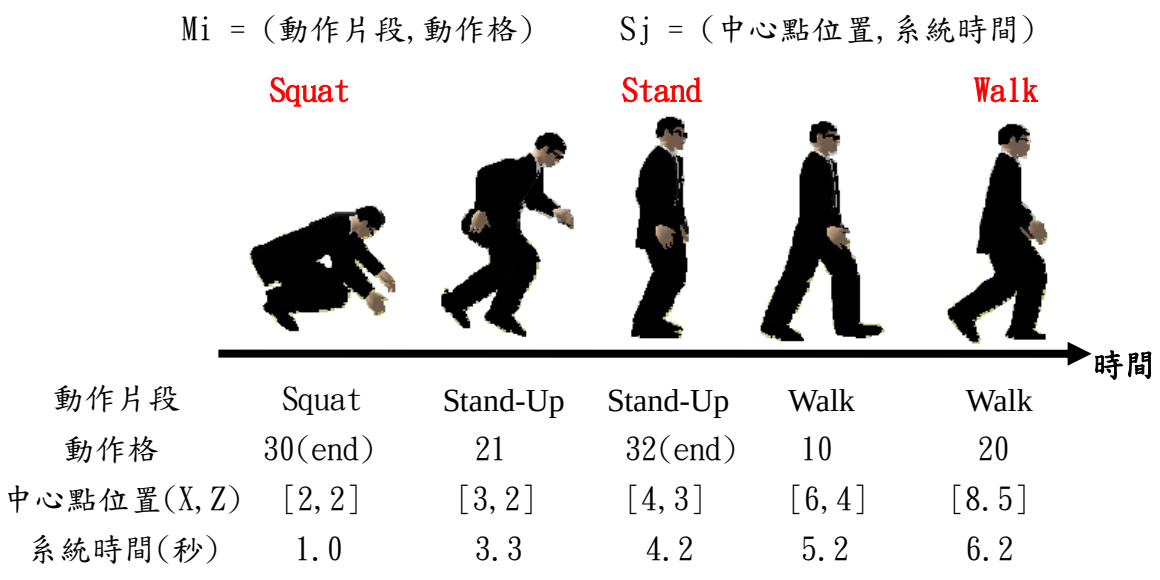
假設障礙物的位置皆為已知，在我們所設計的智慧型角色運動模組裡，為了增加系統應用的彈性，我們允許場景中的某些障礙物可以因為角色動作和使用者命令的影響而產生變動。當我們在定義場景資訊時，我們會將場景分成靜態和動態兩個部分來定義。靜態的部分是一些不會受到角色動作和使用者命令所影響的場景資訊，包含了地形高度

變化、無法破壞的牆壁及無法移動的障礙物等。動態的部分則是一些會受到角色動作和使用者命令所影響的場景資訊，包括了使用者所發射的飛彈、可以被破壞的牆壁及可以被角色移動的障礙物等。這種分類方式讓我們可以對靜態和動態場景採取不同的處理方式，藉此提高處理的效率。

3.1.3. 角色組態定義

在我們所設計的智慧型角色運動模組裡，虛擬角色的組態是以(Mi,Sj)來表示。Mi 是角色所正在進行的動作，因為本研究的角色動作是取自 Mocap 資料，而這些 Mocap 資料被儲存成由不同動作片段所組成的動作圖，所以我們用動作片段及動作格兩個參數 (m,f)來表示動作的畫格。另外，Sj 是角色在所處環境裡的狀態，它的定義方法則會因應用的不同而有所不同。在一般的 3D 射擊遊戲應用裡，我們可以是用角色的位置、剩餘子彈數及受傷狀態等各種跟相關的參數來表示 Sj。

圖 3.2是個簡單的角色組態範例，這個範例是以角色中心點(Root)所處的位置及當時的系統時間來表示 Sj，其中的橫軸代表角色前進方向的位置，縱軸代表高度。這裡的範例列出了角色從 Squat 動作轉換到 Walk 動作的一連串角色組態變化。



動作片段	Squat	Stand-Up	Stand-Up	Walk	Walk
動作格	30(end)	21	32(end)	10	20
中心點位置(X,Z)	[2,2]	[3,2]	[4,3]	[6,4]	[8.5]
系統時間(秒)	1.0	3.3	4.2	5.2	6.2

圖 3.2：角色組態的範例

3.1.4. 使用者的輸入

在使用者的輸入部分，我們是以鍵盤、滑鼠及控制手把等各種常見的低自由度操作裝置來讓使用者進行輸入，並只提供有限的命令組合給使用者進行選擇。例如在一般的第三人稱 3D 角色操作系統裡，系統只會提供前進、後退、往右和往左等四種有限的命令組合來讓使用者操作場景中的虛擬角色。而在第一人稱的 3D 射擊遊戲裡，使用者則是只能用上下左右四個鍵來控制視角的移動，並以滑鼠點擊螢幕來發射飛彈。這裡我們希望能提供給使用者一個簡單的操作介面，這個操作介面可以讓使用者輕鬆的跟虛擬場景中的角色進行互動。且場景中的虛擬角色也能隨著狀況的不同，對使用者的輸入做出相對合適的反應動作，而不是直接根據輸入的命令來找出相對映的相同動作。例如在 3D 角色操作系統的應用裡，當使用者下達前進命令時，虛擬角色可能會因為前方有障礙物而選擇往前跳躍、停止或迴避的動作，並不會固定選擇往前走的動作。而在 3D 射擊遊戲裡，當虛擬角色扮演玩家的敵人時，系統更會進一步根據使用者的視角範圍、可能的攻擊指令及可供閃躲的障礙物等相關資訊，來規劃出各種不同的閃避動作。

3.1.5. 角色的運動計畫

在運動計畫的部分，我們的運動模組是以動作片段為單位來進行角色動作的規劃。系統每次會規劃出一個動作片段的連續角色組態，並在這個動作片段進行到結尾時，才繼續從設計者事先輸入的動作圖裡選擇出下一個可接續的動作片段。例如在圖 3.2的例子裡，系統會在角色組態進行到 Squat 動作片段的結尾時，才從動作圖裡選出 Stand-Up 動作片段為下一個接續的動作。之後當 Stand-Up 動作片段進行到結尾時，再繼續選擇 Walk 動作片段為下一個接續動作，如此重複下去。這種動作規劃方式能有效的降低問題的複雜度，並維持不錯的動作反應精細度。而藉由這種以動作片段為單位來進行角色動作規劃的方式，我們也能將所需的計算分布在動作片段的各個動作格裡，並動態的調整計算量的分布，藉此達到使用時間預算(Time Budget)的效果。

3.2. 系統概述

3.2.1. 設計構想及系統架構

在系統架構的設計上，我們主要是根據分析出來的人類動作反應模式來進行設計。在現實生活中，當我們在進行一個動作時，我們往往也還會繼續思考接下來幾個可能的動作，並隨著環境狀態的改變，不停的改變著我們的思考結果。這樣的動作反應模式，能將人類思考動作的時間分散在不同的時段裡，不會因為思考時間過度集中而導致動作的停頓，且所反應出來的動作也能夠隨著時間的緊迫性而有所不同，藉此達到即時反應的效果。另外，在能夠預知未來動作的假設情況下，因為選錯動作而導致無法繼續進行下個動作的情形也能相對減少，使得我們的反應動作有著很高的成功率。

根據以上的構想，我們設計出執行流程如圖 3.3的系統架構。如圖所示，角色會在限定時間內進行思考，並在思考時限到達後檢查是否必須進行下一個反應動作。如果需要進行下一個反應動作，角色便直接根據之前的思考結果來決定要選擇什麼樣的反應動作，否則便跳到下個步驟去更新場景，並繼續進行思考。而在角色互動的部份，這個系統裡的使用者命令會影響到場景資訊的變動，場景資訊的變動又會影響到角色的思考結果，如此便可以間接的讓角色根據使用者的命令來進行反應，藉此達到互動的效果。

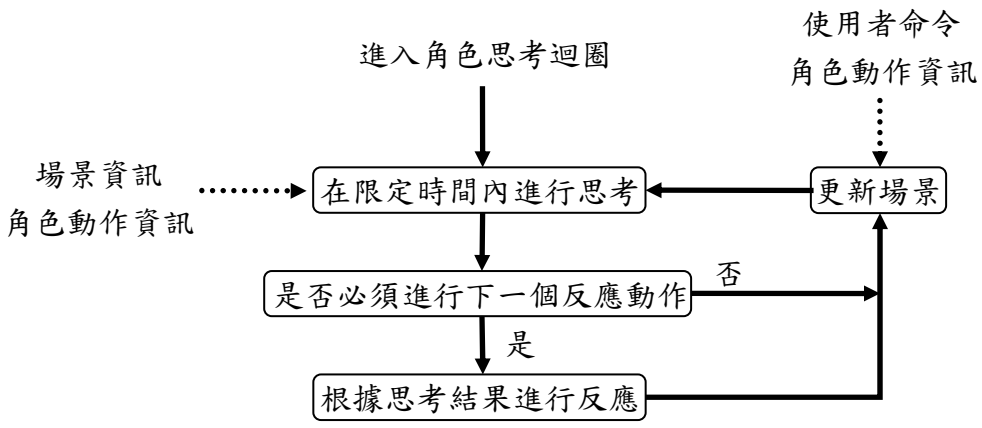


圖 3.3：系統流程的示意圖

圖 3.4是本論文所提之系統流程圖。如圖所示，我們將系統劃分成三部分：第一部分為角色動作預測，目的是在角色選擇下個動作前，讓系統盡量預測角色的未來可能動作及所處環境狀態。例如在十字路口的場景中，當角色決定是否要穿越馬路時，它會先檢查附近是否有車子存在，穿越馬路的途中是否會被這些車子撞到等可能狀況。第二部分為角色動作選擇，目的是分析預測的結果，來判斷角色必須進行的反應動作。例如在

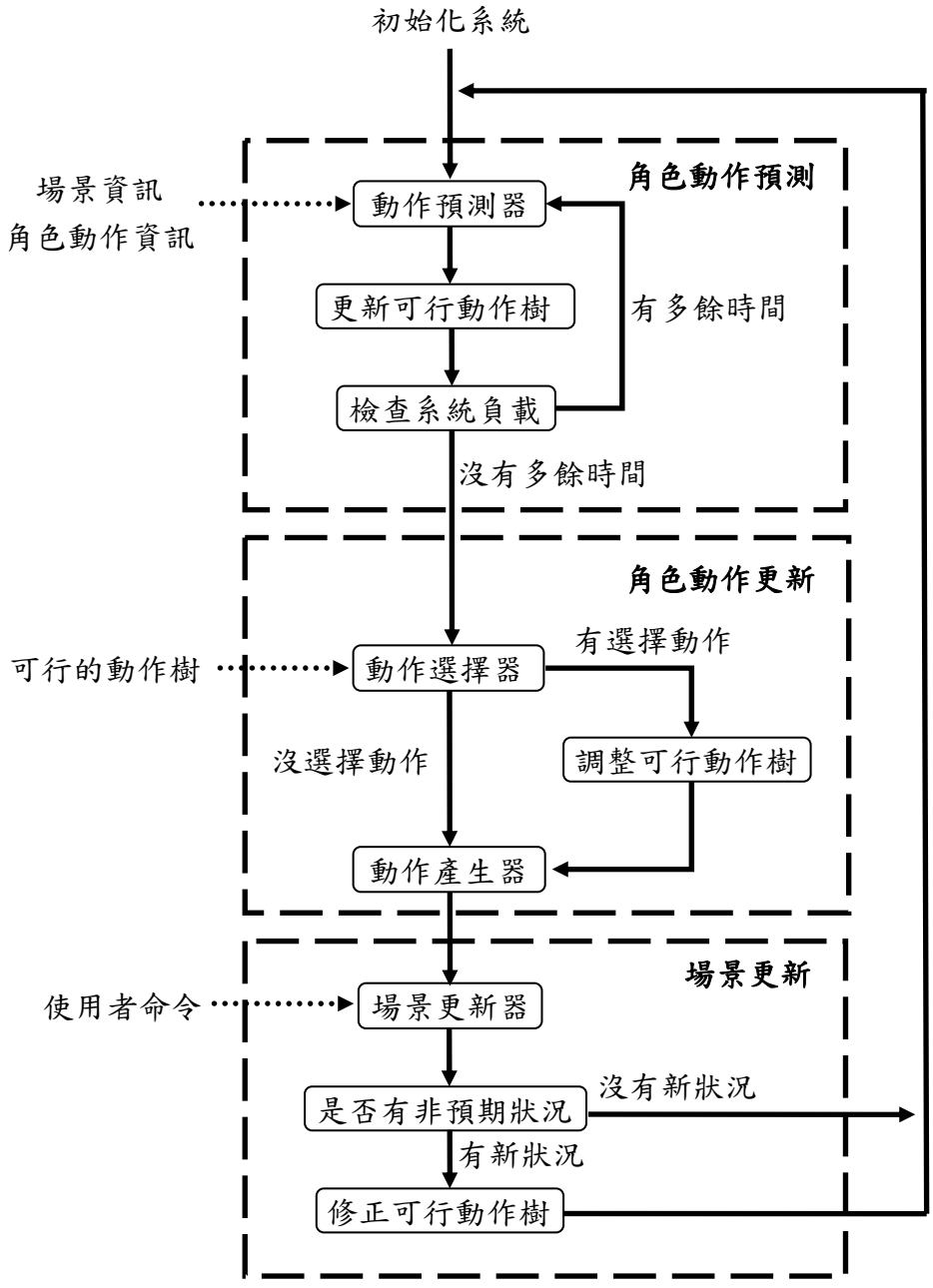


圖 3.4：系統流程圖

上述例子裡，當角色的思考結果是穿越馬路途中不會被車子給撞到時，角色便會根據這個結果來做出穿越馬路的動作選擇。第三部分為場景更新，目的是根據使用者命令來改變場景，並根據變動結果來適時修正預測內容。例如當角色決定穿越馬路，並已在穿越的過程時，附近卻突然有輛車子出現。角色此時必須立刻修正思考結果，重新預測角色在穿越的過程裡是否會被這輛車子撞到，並在之後做出是要快速通過或後退的決定。

3.2.2. 角色動作預測

此步驟的目的是要預測角色的未來可能動作及所處環境狀態。在我們系統裡，角色是以動作片段為單位來進行反應，所以如何根據角色目前的動作片段來推測出角色未來的動作片段，並計算出這些動作片段會對場景造成的影響，便是我們這裡想得到的輸出結果。在這部分，系統需給予兩個輸入參數：角色動作資訊及場景地形資訊。角色動作資訊包括了角色的動作圖(Motion Graph)及骨架模型資訊；其中動作圖定義了角色的反應機制，系統可以藉此來預測角色未來的反應動作；骨架模型資訊則是用來檢查角色和場景的碰撞。場景地形資訊包含了地形高度變化、障礙物的位置座標、幾何資訊及移動速度等，此參數會隨著應用而有所不同，我們可以根據這個參數來預測場景的變動。

在輸入了這兩個參數後，系統便能藉由動作預測器來預測出角色及場景的未來可能狀態，並將這些預測結果儲存在可行動作樹(Feasible Motion Tree)裡。可行動作樹是我們在這篇論文裡所提出來的一種新架構，這種架構會將系統的預測結果儲存成樹狀資料結構，有著容易更新及修正的優點。之後程式會確認系統的負載狀態，檢查是否已超過角色思考的時限。如果沒超過時限，便繼續進行思考，否則就進入角色動作更新步驟。

3.2.3. 角色動作更新

當思考結束後，系統會判斷角色所正在進行的動作是否已經到達動作片段的結尾。如果還沒到結尾，角色會繼續進行下一個動作格，否則便找出下一個可行的動作片段，

並以這個動作片段的開頭來當角色的下一個動作格。其中在選擇動作片段的部分，系統會將之前所計算出來的可行的動作樹輸入到動作選擇器裡，並根據裡面的預測結果來計算出最適合角色的反應動作，並在之後根據所選擇的動作反應來調整可行的動作樹。而這裡之所以要調整可行的動作樹，是因為當角色選擇了一個新的反應動作後，之前所預測的部分結果可能會因此而失效。例如當角色目前的動作是向前走時，系統預測它接下來的可能動作會是繼續往前走、往右轉、往左轉或是停下來。但在角色選擇了停下來的動作後，往前走、往右轉和往左轉的相關預測已經不算是角色的未來可能狀況之一了，所以我們必須將它們從預測結果裡移除。

在我們的系統裡，角色的動作狀態是以參數化的方式進行儲存。這裡的動作產生器的主要功能便是將輸入的角色動作參數，轉換成實際場景裡的角色動作動畫。例如當角色目前的動作參數為(Motion 3, Frame 20)時，動作產生器必須從動作片段 3 的第 20 個動作格裡取出動作資料，並將其套用到角色的人體模型中。這部分有許多動畫處理的相關功能，包括動作擷取資料的讀取及動作參數的混合等。

3.2.4. 場景更新

角色選擇完反應的動作後，系統必須根據各種環境變數來改變場景的狀態，並在之後檢查場景裡是否有非預期的新狀況產生。如果沒有新狀況產生，就繼續進行下一個步驟，否則便必須根據這些新狀況來調整可行的動作樹。這裡的非預期狀況指的是之前在進行角色動作預測時，系統所沒有考慮到的場景變數。在 3D 射擊遊戲的例子裡，它指的便是使用者所新發射出來的飛彈。這裡之所以修正可行的動作樹的原因，是因為系統在進行角色動作預測時，它並不會將使用者新命令所會對場景造成的影響加入預測範圍中。而是當這些新命令對場景產生新的狀況時，才用修正的方式來將其納入考量中。這種方式可以減少許多不必要的計算，藉此提升我們程式的執行效率。