

## 第五章 虛擬實境場景中的碰撞偵測

一般在虛擬實境的場景中，要使物件進行移動或轉動等改變其空間位置的動態操作，這時 model 與 model 間的空間位置與幾何外型就難免會發生干涉的情形。如圖 5-1 所示，為兩台車重疊在一起。



圖 5-1 物件干涉情形

而就真實世界來說，物體並不會有這種情形產生，所以如果虛擬實境的場景發生如此現象，就會予人有跟一般認知世界不同的奇怪感，在真實世界中物與物接觸應會發生碰撞的反應而其碰撞的反應會因物體未發生碰撞前的狀況之不同而隨之不同，虛擬實境要能讓人有真實的感覺，當然必須盡可能地把真實環境的種種特性模擬出來。其中，這種「碰撞」的效應就是模擬人在真實世界中的各種行為時非常重要的課題。特別是在需要動態模擬的虛擬場景裡，例如飛行模擬、汽車駕駛模擬、虛擬組裝乃至於時下許多 3D 遊戲等，碰撞的「偵測」不但是必須的也是最基本的。唯有能掌握碰撞的發生與否，才能更進一步的模擬物體碰撞後的物理變化情形，例

如可能會發出聲音，或是移動路徑會改變甚至是物體產生變形等。  
所以「碰撞偵測」在虛擬實境的應用上可說是居關鍵的地位。

回到 VRML 架構，VRML 對碰撞的支援度只有虛擬人物 (Avatar) 與虛擬世界中之物體有碰撞偵測的機制，但在場景內物體與物體之間，並沒有任何碰撞偵測的機制，這是其在虛擬實境的應用上，最為人詬病的地方。以本研究為例，虛擬人物為放置在車內駕駛座上，然而會發生碰撞的地方，乃是駕駛者所駕駛汽車之外體，而非虛擬人物本身，故 VRML 所提供對碰撞的支援，對本研究而言並不適當，故需使用其他方法，來偵測碰撞情形的產生與否。

本研究有包含車流之開發，如有車流即必須討論駕駛者車輛與車流發生碰撞情形的產生。故將在以下之小節，討論 VRML 場景中的碰撞偵測。

## 5.1 關於在 VRML 中移動與轉動的行為

前面有談到，碰撞的發生一定是在物件「移動」或「轉動」的時候。並且因為我們所進行的移動與轉動操作，都是根據 VRML 所規定的格式來運作。所以要探討 VRML 的碰撞問題，最首要的課題就是能完全掌握 VRML 中各物件移動與轉動的模式與特性，才能使各種碰撞概念可以順利在 VRML 中實行。

在 VRML 的檔案裡，關於任何物件移動(translation)與轉動(rotation)的資訊都放在該物件的 Transform 節點(Node)裡，其撰寫格式如下：

```
Transform {  
    translation    x y z  
    rotation       a b c  $\theta$   
}
```

其中 translation 之後所接的  $x, y, z$  , 代表空間中的座標值（以向量  $T$  表示）；而 rotation 之後接的  $a, b, c$  , 則是旋轉軸的向量表示法（以向量  $R$  表示），並且  $\theta$  即為以該旋轉軸  $R$  所旋轉的徑度量（遵守右手螺旋定則，逆時針為正）。另外，VRML 接受物件用多階層的 Transform 來描述，如下所示：

```

Transform {
    translation    T
    rotation       R  $\theta$ 
    children Transform {
        translation    T
        rotation       R  $\theta$ 
    }
}

```

只要在內層的 Transform 前頭加上 children 的文字即可，而此種階層式架構的運作方式會產生全域座標系(Global Coordinate System)與區域座標系(Local Coordinate System)的不一致。一開始全域座標與區域座標為一樣的，但經過第一層的 translation，區域座標系原點移動到第一層的 translation 所給定之值之世界座標後，區域座標系再繞第一層的 rotation 所給定之軸旋轉，而其 rotation 之值乃是對經第一層的 translation 移動後之區域座標系而言，然而由於向量在空間中可平移，且經第一層的 translation 移動後之區域座標系並未經旋轉之過程，故其 rotation 值對世界座標系和區域座標系而言皆代表同一方向，第二層的 translation 值則是對經第一層的 translation 和 rotation 後之區域座標系而言，同理第二層的 rotation 值則是對經第一層的 translation 和 rotation 後再經第二層的 translation 後之區域座標系而言。

這種階層的描述方式，主要可以具有兩種功用，第一是群組的效果。即一個 children 之後可以接續兩個或更多的 Transform，而因

為這些 Transform 都屬於同一層，所以均同被上一層的 Transform 所控制。第二個功能是產生全域座標系(Global Coordinate System)與區域座標系(Local Coordinate System)的分別。意即只有最外層的 Transform 等同於在全域座標系運作，其它第二層以下的 Transform 全部屬於區域座標系(Local Coordinate System)。

要偵測物體與物體是否有碰撞情形產生，通常需要得知物體之位置，然而如使用階層式架構，則需經矩陣運算後才能得知物體在世界座標系下之位置。一般我們在數學裡表達物體中的旋轉情形，是使用所謂的旋轉矩陣。其中對 x, y, z 軸的旋轉矩陣  $R_x$ ,  $R_y$ ,  $R_z$  分別為：

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \quad (5.1)$$

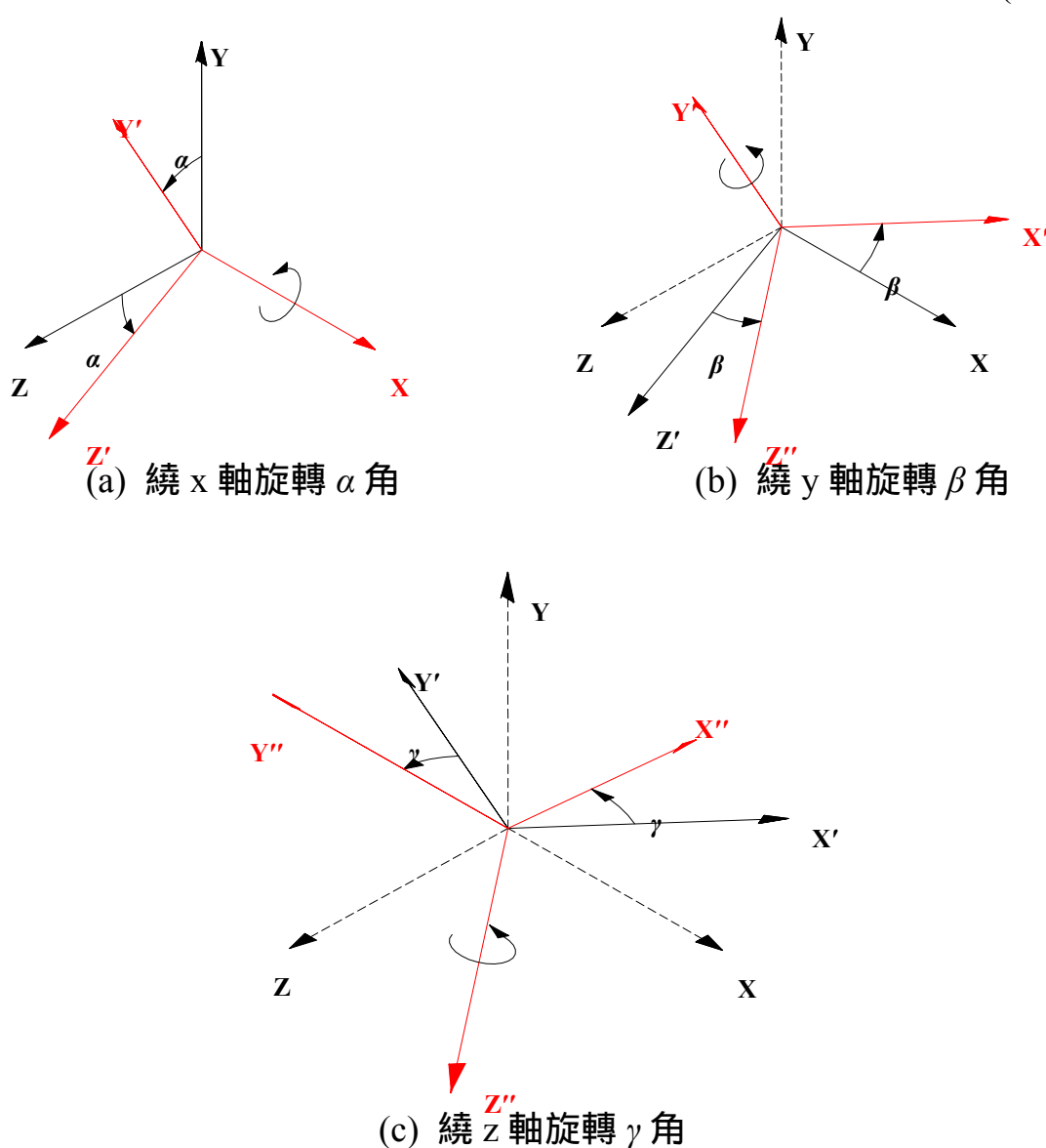
$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad (5.2)$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.3)$$

其中  $\alpha, \beta, \gamma$  分別為  $R_x, R_y, R_z$  的旋轉角度。假設空間中某物體分別依序以 x, y, z 軸旋轉了  $\alpha, \beta, \gamma$  的角度，如圖 5.2 的(a), (b), (c)所示，其結果所得的旋轉矩陣  $R_{xyz}$  為：

$$\begin{aligned} \mathbf{R}_{xyz}(\alpha, \beta, \gamma) &= \mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha) \\ &= \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \end{aligned}$$

(5.4)

圖 5-2 對 x, y, z 軸旋轉  $\alpha, \beta, \gamma$  角度

須注意矩陣相乘的順序是從右到左。而若將(5.4)式乘開得：

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \begin{bmatrix} \cos \alpha \cos \beta \cos \gamma & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\ \sin \alpha \cos \beta & \sin \alpha \cos \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\ -\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma \end{bmatrix} \quad (5.5)$$

式(5.1)、(5.2)、(5.3)為對 x、y、z 軸旋轉的特定情形，如對任意軸旋轉則其旋轉矩陣  $\mathbf{R}_{\text{arbitrary}}$  如下：

$$R_{arbitrary}(\theta) = \begin{bmatrix} ka_x^2 + c & ka_xa_y - sa_z & ka_xa_z + sa_y \\ ka_xa_y + sa_z & ka_y^2 + c & ka_ya_z - sa_x \\ ka_xa_z - sa_y & ka_ya_z + sa_x & ka_z^2 + c \end{bmatrix} \quad (5.6)$$

其中 $[a_x, a_y, a_z]^T$  為沿著任意軸之單位向量， $c=\cos\theta$ ， $s=\sin\theta$ ， $k=1-\cos\theta$ ， $\theta$  為繞任意軸旋轉之旋轉角度。

## 5.2 VRML 中駕駛者車輛的控制

要得知駕駛者車輛是否與周圍環境發生碰撞情形，必先探討我們如何控制 VRML 中的駕駛者車輛，才能根據控制方式計算碰撞情形是否發生，我們使用類似如下之結構控制車輛的行為。

```
Transform{
  translation
  rotation
  children[
    Transform{
      rotation
      children[.....
    ]
  ]
}
```

第一層 transform 之 translation 控制車輛在虛擬世界的世界座標位置，第一層 rotation 則為車輛在有路面超高時，車輛的旋轉軸和旋轉角度，第二層 transform 之 rotation 控制車輛欲改變行進方向時所繞之旋轉軸及旋轉角度。當車輛欲改變行進的方向時，係由前輪偏轉成一角度，兩前輪的中垂線與後輪之輪軸延伸線相交於瞬時中心，車輛即以此點為旋轉中心作圓周運動，在本研究中則將旋轉中心設在車輛中心並設定速度需要高於某一速度才能旋轉，以達到近

似轉彎的效果。

現在分為在無路面超高和有路面超高之情形下，對駕駛者車輛的控制方式作一說明。

- 無路面超高

在本研究所設計的場景中，在無路面超高路段對駕駛者車輛的控制方式為將第一層 transform 之 translation 值設定為車輛在虛擬世界的世界座標位置，第一層 rotation 則設定為 0 1 0 0 即未旋轉，第二層 transform 之 rotation 值則將車輛欲改變行進方向時所繞之旋轉軸及旋轉角度設定給它，其值為 0 -1 0 ， 為繞 0 -1 0 旋轉之旋轉角度。

- 有路面超高

車輛如位於有路面超高之路段，車輛應繞垂直向量與路面法向量之外積向量旋轉（見圖 5-3），故我們需得知路面法向量，才能算出車輛應繞何軸旋轉，同理在虛擬世界中，我們則需找出車輛位於哪一個 polygon 上，並由該 polygon 之點資料，算出該 polygon 之法向量  $N_p$ ，得知垂直向量  $y$  及 polygon 法向量  $N_p$  後，利用內積定義求其夾角  $\theta$ ，而  $\theta$  即為旋轉角度。

$$\theta = \cos^{-1} \frac{N_p \cdot y}{|N_p| |y|} \quad (5.7)$$

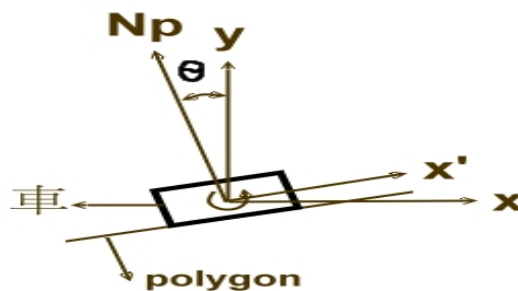


圖 5-3 車輛位於有路面超高路段示意圖

由前所述，我們將第一層 transform 之 translation 值設定為車輛在虛擬世界的世界座標位置，第一層 rotation 則將由  $N_p$  及  $y$  所外積出來的向量與經由(5.7)式算出的旋轉角度設定給它，第二層 transform 之 rotation 值則將車輛欲改變行進方向時所繞之旋轉軸及旋轉角度設定給它，其值為  $0 -1 0$ ，為繞  $0 -1 0$  旋轉之旋轉角度。

## 5.3 碰撞偵測技術

由於碰撞體之特性不同，需依不同特性使用不同的碰撞偵測技術，本研究所使用之碰撞偵測技術分可為兩部分：

### 1. 駕駛者車輛對車流之碰撞偵測

此部份技術為美國北卡大(The University of North Carolina at Chapel Hill.)之 S. Gottschalk 所發展出來的「碰撞函式庫」Rapid.lib (簡稱 Rapid)。Rapid 本身並非特別針對 VRML 的架構所撰寫，而僅是提供使用者在 C 語言的開發環境下，一個一般性的 3D model 碰撞偵測函式庫。

### 2. 駕駛者車輛對場景中不可侵入部分之碰撞偵測

為使用影像處理方法，將場景平面中不可侵入之部份，與可侵入之部份分別以 0, 1 標記之，當駕駛者車輛移動時，判斷駕駛者車輛所處位置之平面座標，是否落於場景平面中不可侵入之部份，來判斷碰撞情形是否產生。

#### 5.3.1 駕駛者車輛對車流之碰撞偵測

此部份使用美國北卡大(The University of North Carolina at Chapel Hill.)之 S. Gottschalk 所發展出來的 Rapid.lib 碰撞函式庫，



Rapid 與 VRML 之間，在資料格式上並不盡相同，我們在使用 Rapid 前，需將 VRML 的 3D model 資訊轉換成 Rapid 函式庫可接受的格式。表 5.1 中，我們將 Rapid 所需的物件資訊與 VRML 提供的物件資訊做一簡單的比較[28]：

表 5-1 Rapid 與 VRML 物件資訊比較

	Rapid 函式庫	VRML	比較
<b>位移資訊 ( translation )</b>	包含 x, y, z 座標之 $1 \times 3$ 矩陣	包含 x, y, z 座標之 $1 \times 3$ 矩陣	相同
<b>旋轉資訊 ( rotation )</b>	一般 $3 \times 3$ 旋轉矩陣	包含旋轉軸向量與旋轉角之 $1 \times 4$ 矩陣	需轉換
<b>縮放比例資訊 ( scale )</b>	固定比例之單一縮放比	x, y, z 三軸縮放比例	格式不合 ( 特殊情況可用 )
<b>物件幾何外型 資訊 ( geometry )</b>	3D model 之點資訊與 triangle 面資訊	3D model 之點資訊與 polygon 面資訊	VRML 面資訊需特別設定成三角面，且 EAI 不提供此資訊

由於汽車外型類似長方體，我們便以長方體包裹汽車來簡化碰撞偵測時，所需判斷之 polygon 數，並使用 CosmoWords2.0 將原本由四角面建構之長方體，分割成為由三角面建構之長方體，並處理縮放比例，Rapid 判斷是否發生碰撞為使用下列之函式來輸入資訊：

```
Collide(R1, T2, CollideModel_1, R2, T2, CollideModel_2,
ALL_CONTACTS)
```

其中 R1，R2 為待判斷兩物體之  $3 \times 3$  旋轉矩陣。T1，T2 兩物體之位置，CollideModel 為物體之三角面資訊，其中 R0，R1，T0，T1

需為在世界座標中之位置及對旋轉軸旋轉之旋轉矩陣，如 5.2 節中所述，我們在控制汽車位置時，皆以處在世界座標中之位置控制，所以問題不大，但車輛在有路面超高的路段下會有傾斜現象，且在加上汽車之左彎右彎，故須計算在世界座標下之旋轉軸的旋轉矩陣，根據車輛的旋轉情形我們將式(5.6)和式(5.2)相乘，如式(5.8)所示，並將所得之結果輸入 R1 或 R2 後，函式便會幫我們計算兩物體是否碰撞。

$$\begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} ka_x^2 + c & ka_x a_y - sa_z & ka_x a_z + sa_y \\ ka_x a_y + sa_z & ka_y^2 + c & ka_y a_z - sa_x \\ ka_x a_z - sa_y & ka_y a_z + sa_x & ka_z^2 + c \end{bmatrix} \quad (5.8)$$

### 5.3.2 駕駛者車輛對場景中不可侵入部分之碰撞偵測

如使用前述之 Rapid 函式庫，在彎道部分處理起來，將需耗費相當多之時間，將彎道不可侵入部分包裹，需用很多物體才能達成，故使用上相當不便，故利用影像處理之方法，將場景平面（即 VRML 預設座標系之 XZ 平面）中不可侵入部分設定為 0，可侵入部分設定為 1，當駕駛者車輛移動時，輸入駕駛者車輛位於場景中平面世界座標之位置（即 VRML 預設座標系之 XZ 平面之座標位置）以取得場景平面中用影像處理後之標記值，當傳回之標記值為 0 時，即碰撞情形產生，為 1 時，即碰撞情形未產生。其流程如下：

1. 在 3DS 之 TopView 視窗 render 一張由 TopView 視野觀看整個場景之上視圖。
2. 用影像處理軟體，將場景中可侵入部份塗白，不可侵入部份塗黑。
3. 撰寫影像處理程式，將影像黑色部份標記為 0，白色部份標記為

- 1。
4. 在虛擬場景中，找兩參考點，並記錄其平面座標值（即 VRML 預設座標系之 XZ 平面），並在由 1.所產生之影像中找到兩參考點之影像平面座標。
5. 由虛擬場景中兩參考點之平面座標值，與由 1.所產生之影像中找到兩參考點之影像平面座標，並找出其轉換關係。
6. 當駕駛者車輛移動時，輸入駕駛者車輛位於場景中平面世界座標之位置，並利用由 5.所得之轉換關係，找出駕駛者車輛之影像平面座標。
7. 檢查由 6.產生之影像平面座標之標記值，如為 0，則發生碰撞情形，如為 1 則未發生碰撞情形。

由於在本研究所設計之場景中，不可侵入部分乃是固定不變的，所以可以用影像處理方法，將不可侵入的部份標記起來，在駕駛者車輛對車流之碰撞偵測上則不適用，因車流為會移動的，所以不可能預先處理，故要處理會動之物體與駕駛者車輛之碰撞偵測，可用 rapid 碰撞偵測函式庫，而不會動之物體與駕駛者車輛之碰撞偵測，則可使用影像處理之方法預先標記之。

