前言

為誰而寫

簡單地說，本書是專為沒有紮實的力學和物理背景，卻負責要在遊戲中加入物理模擬的電腦遊戲設計師所設計的。

身為遊戲設計師或電腦玩家的讀者，也許曾經見過以「超擬真」或「寫實」為號召的產品。同時，自己或公司的行銷部門，也許正在考慮如何在產品中添加物理的擬真度。或是想要作一些全新的嘗試，因此必須去探索真實的物理學。但唯一的問題是在大學時期時，可能考完期末考之後就將物理教科書丟入湖中，而且之後未曾再接觸此領域。也許還買了一套相當酷的物理引擎，但是卻不了解其中運作的原理，或是執行模擬時此引擎的影響。或者負責調校其他人所寫的物理學程式碼時，卻不清楚其運作的原理。有以上這些情況的人都適合閱讀本書。

當然，可以從網際網路、專業期刊和雜誌來獲得相關資料與指引，讓遊戲能加入物理的擬真度，也可以重新閱讀舊的物理教科書。但或許你會發現這些資料可能太過廣泛而不適合直接應用於遊戲，或是內容過於進階而需要搜尋其他資料來補強基礎知識。本書將蒐集所有需要的資料，讓作為遊戲設計師的人能夠有一個好的出發點，以便能在遊戲中加入物理擬真度來豐富遊戲內容。

本書並不只是針對各式各樣的問題而提供不同的範例程式而已，因為網路上充斥著太多這類的範例程式（當然也有一些非常好的程式）。本書除了對某些問題提供解答之外，還希望讓讀者對於相關領域能有徹底而基本的瞭解，進而對其他問題發展出自己的解決方案。因此將會詳細解說運用在遊戲開發的物理學原理，除了範例程式外，也會提供手寫的計算範例來補充說明。

讀者的先備知識

本書不會假設讀者都是物理專家，但仍然希望非物理或工程學系的讀者，至少要有大專程度的基礎物理學知識。但是前幾章都會複習與遊戲物理相關的主題，所以對物理學已經印象不深的話也沒關係。

但是本書希望讀者精通三角函數、向量和矩陣數學，相關資料可參考附錄。同時也希望具備有大專程度的基本微積分知識，包括對於顯函數（explicit function）的積分與微分。數值積分與微分又是另一個領域，本書稍後幾章將會詳細介紹這些技巧。

力學

當本書還處於構思階段時，許多人若聽到「擬真物理學」或「即時模擬」這些名詞，通常會立刻想到飛行模擬程式。頂尖的飛行模擬當然與本書有關，然而許多不同類型的遊戲和特定遊戲的元素，也都受益於物理的擬真度。

想像以下這個範例：設計下一個轟動的射擊遊戲，有第一人稱的 3D 畫面、華麗的貼圖和令人震撼的音效，但卻缺少某個元素：擬真度。具體地說，希望藉由挑戰玩家的射擊技術讓遊戲感覺更真實，欲達成這個目標就要將某些因素列入考慮，例如與目標之間的距離、風速和方向、槍口速度等等。但其實不需要捏造這些元素，而是根據物理原理來模擬這些元素的物理特性。任職於 MathEngine Plc 的 Gary Powell 說：「虛擬世界的影像和令人沉迷的體驗、以精密的多邊形仔細建立的模型、複雜的貼圖和先進的光源投影，當物體開始移動和互動時通常是令人震驚的[1]」。Havok.com 的 CEO — Steven Collins 博士則說「這完全是互動性和吸引力」[2]。我們認為這些人都找對了方向。為什麼要耗費這麼多時間和精力，只為了讓遊戲世界看起來夠真實，但不再多花一點時間讓遊戲中的動作更真實呢？

---

1 撰寫本書第一版時，Gary Powell 任職於 MathEngine Plc。其產品包括 Dynamics Toolkit 2 和 Collision Toolkit 1，可以處理單一或多重物體的動力學。現在該公司的名稱已改為 CM Labs。

2 撰寫本書第一版時，Collins 博士是 Havok.com 的 CEO。這間公司的技術可以處理剛體、柔體、布料、流體和粒子的動力學。而 Intel 在 2005 年買下此公司。

以下是一些可利用物理學增加擬真度的遊戲元素例子：

- 火箭和飛彈的飛行軌跡，包含燃料消耗的影響。
- 物體（像是撞球）之間的碰撞。
- 大型物體（像是星球和戰鬥要塞）之間的萬有引力效應。
- 賽車急轉彎時的穩定性。
- 船或浮於水面的交通工具的動力學。
- 棒球被球棒打擊出去的飛行路徑。
- 被丟進帽子的紙牌的飛行路徑。

這絕不會是個完整的清單，僅是提供一些概念正確的範例。遊戲中有許多彈跳、飛行、翻滾、滑翔、或非靜止的物體，可用相當逼真的方式模擬出來，得以創造吸引人且可信的遊戲內容。

至於要如何達到這樣的真實性呢？當然就要透過物理學原理，這時就要回到本節標題：力學。物理學是個範圍極廣的科學，包含許多不同卻相關的領域。最常使用在擬真遊戲內容的主題是力學，也就是「真實的物理學」所代表的意義。

根據定義，力學是物體在靜止或運動時研究作用於物體上的力的影響。因此力學可以分成靜力學與動力學，前者著重於物體在靜止狀態時的研究，而後者則著重於物體運動狀態的研究。物理學其中一個最古老且最常被研究的領域 — 力學的起源，可追溯到 2000 年前亞里斯多德的時代。此領域另一個更早的論述為「力學的問題」，但是這個作品的起源無法考證。雖然某些早期的研究讓物理現象變成魔術領域，但是這些偉大的想法例如伽利略、克卜勒、尤拉、拉格朗治、達朗貝爾、牛頓、愛因斯坦等等不及備載，使現今對物理學的了解能發展到這種程度，讓我們能見識到許多驚人的先進技術。

因為希望能讓遊戲生動活潑且栩栩如生，所以本書將著重在運動中的物體，並將鑽研此領域的力學。在力學領域中，還有許多特定主題值得研究。像是運動學只著重在物體的運動，而不是作用在物體上的力。動力學同時注重物體的運動和作用在物體上並影響物體運動的力。本書將仔細探討這兩個主題。

數位物理學

本書第一版專注於力學問題，但自其出版開始到現在也經過幾十年了，因此需要將遊戲物理的定義擴大，並將數位物理學涵蓋進來，例如智慧型手機和其獨特的使用者互動體驗相關的物理技術。當像是 Wii、PlayStation、X Box 和智慧型手機等平台愈廣泛流行時，開發者必須瞭解為這些平台帶來新鮮遊戲體驗的週邊輸入或感測技術。但這不應該視為一個負擔，而該看成是一個增加使用者和自己遊戲互動的絕佳機會。

章節編排

物理擬真在遊戲界並不是新玩意。事實上，許多銷售的遊戲都以物理引擎作為號召。許多 3D 模型和動畫軟體也有內建的物理引擎，能寫實畫出特定動作。雜誌也就自然而然時常討論各種以物理為基礎的遊戲。同樣在另一個層次，對於剛體（rigid body）[3] 即時模擬的研究也已經有好幾年了，技術期刊也充滿了這方面的論文。此主題從多重且相連的剛體模擬到布料的模擬，範圍非常廣大。然而正如之前所述，這些都是吸引人且有價值的主題，但對於遊戲開發者而言，要直接使用卻是受到限制。因為遊戲開發者必須要對這些領域的力學有完整地認識，也要學習其他資源的基礎。況且，其中大多重點主要在解運動方程式的數學上，並未真正處理作用在模擬中的物體或系統上的力。

在 Animats 工作的 John Nagle 認為對他而言，研發以物理為基礎的模擬遊戲時，最困難的部分是研發數值穩定且健全的程式碼[4]。Gary Powell 對此發表回應，他說：要最小化與產品穩定度和真實行為相關的參數調整是最困難的挑戰之一。對於處理物體運動的數學來說，要使其速度和穩定兼具確實是模擬器最困難的部分。最重要的是，保持模擬程式在初始及進行中作用力的完整性和正確性也是如此。

---

3 剛體正式的定義為粒子系統組成的物體，其粒子能彼此保持固定的距離而不會有相對的縮放或旋轉。即使力學的範疇也包括處理柔體甚至是向水般的流體，但這裡只會討論剛體。

4 撰寫本書第一版時，John Nagle 是 Falling Bodies 的開發者，Falling Bodies 是應用在 Softimage 3D 軟體上的動力學嵌入模組。

稍後在本書就可以看到，作用力會控制模擬程式中物體的行為，所以若想讓物體有真實行為，就要正確模擬出作用力。

本書的編排方式主要依據對於力學的必要認知，以及真實世界中會作用於特定物體上的作用力。主要可分成四大部分，每一部分都以前一部分為基礎。

第一部分：力學基礎

第一章到第六章皆為複習力學的基礎。

第一章：基本概念

本章只是暖身，內容包含一些最基礎的定理，可作為其他章節的參考。將會提到的主題包括質量和質心、牛頓運動定律、慣性、單位與度量和向量。

第二章：運動學

本章主題有線性速度與角速度、加速度、動量以及 2D 和 3D 環境中粒子與剛體的一般性運動。

第三章：作用力

本章包括力與力矩的原理，可以作為由運動學的範疇跨入動力學範疇的橋樑。一般種類的作用力都會被討論，包括阻力、力場、和壓力。

第四章：動力學

結合前兩章的元素，介紹運動學的主題和解釋動力學與運動學之間的差異。更進一步介紹在 2D 及 3D 環境中，粒子與剛體的動力學。

第五章：碰撞

本章包含粒子與剛體的碰撞反應，也就是說明兩物體在相互碰撞後會發生什麼事。

第六章：拋體

說明一個簡單的拋體落在地上的物理觀念，作為後續章節相關模擬的基礎。

第二部分：剛體動力學

第七章到第十四章在介紹即時模擬（real-time simulation）。

第七章：即時模擬

介紹即時模擬，以及其核心技術-數值積分函式。會談到不同的方法，以及相關的穩定性和參數調整。

第八章：粒子

在進入剛體模擬器之前，本章會說明如何實作粒子模擬器。然後會於下一章擴充納入剛體的部分。

第九章：2D 剛體模擬器

本章將前一章的粒子模擬器擴充成剛體模擬器，新增旋轉和處理轉動慣量的部分。

第十章：碰撞反應實作

本章將碰撞偵測和反應結合起來，以便在 2D 模擬器中實作及時偵測的功能。

第十一章：3D 剛體模擬器的旋轉

本章將說明如何在 3D 中處理剛體模擬器的旋轉，包含處理慣性張量，並將 2D 模擬器延伸為 3D 模擬器。

第十二章：3D 剛體模擬器

本章模擬器含有多個不相連的物體，因此需要一點技巧處理多個物體的碰撞。另外也會討論到模擬器的穩定性和真實性。

第十三章：物體的連接

說明如何將剛體連結成一個物體，用來模擬像是人體、會被炸開的的汽車還有許多其他遊戲中的物體。同時也會討論到多種連接器的類型。

第十四章：物理引擎

本章討論到汽車性能方面的問題，包括空氣動阻力、滾動阻力、滑行距離和道路邊坡。

第三部分：建造物理模型

討論真實世界的問題，從第十五到第十九章。

第十五章：飛機

本章著重的是飛機的元素，包括推進器的力量、阻力、幾何形狀、質量和最重要的升力。

第十六章：船舶和船

討論排水型船艦的基本元素，包括浮力、穩定度、體積、阻力和速度。

第十七章：汽車和氣墊船

討論到汽車性能方面的問題，包括空氣動阻力、滾動阻力、滑行距離和道路邊坡。此外也有討論氣墊船。氣墊船同時有汽車和船的特性。本章將討論讓氣墊船變成獨特的交通工具的特性，涵蓋主題有盤旋飛行、空氣靜升力和方向控制。

第十八章：槍枝與爆炸

本章討論的是槍枝的物理觀念，包含威力大小、後座力和拋體飛行。另外也會討論砲彈擊中物體時的爆炸現象。

第十九章：運動

本章討論的是球類運動的物理觀念，像是棒球、高爾夫球和網球。涵蓋內容除了拋體運動之外，還包含了棒球的投球、打擊、球棒與球的撞擊、高爾夫揮桿、高爾夫球桿與球的撞擊以及網球拍的揮動和網球拍與球的撞擊。

第四部份：數位物理學

這部分會解釋加速度計、觸控螢幕、GPS 和其他相關技巧的物理意義，並說明如何應用於遊戲中。範圍是從第二十章到第二十六章。

第二十章：觸控螢幕

觸控螢幕為行動裝置的遊戲提供一個虛擬的觸控介面，像是在 iPhone 上的應用。本章將會解釋觸控螢幕的物理觀念，以及如何應用於遊戲中，尤其是透過手勢來互動的遊戲。

第二十一章：加速度計

加速度計目前已廣泛應用於行動裝置和遊戲控制器上，讓人能和遊戲中的元素互動。本章將解釋加速度計如何運作及其提供的資料，且這樣的資料該如何應用在遊戲中產生互動效果。談到的主題包含整合加速度計的資料來產生速度、位移和旋轉等。

第二十二章：更換遊戲地點

目前行動裝置普遍有 GPS 功能，本章將解釋 GPS 背後的物理觀念，包含相對論效應。再者，也會解釋 GPS 資料，並示範如何使用 GPS 資料融入到遊戲的互動中。例如會說明在其他操控中，如何區分 GPS 資料去產生速度和加速度。

第二十三章：壓力感測器和負荷元

壓力感測裝置讓玩家能夠與遊戲中的元素互動，例如 Wii 平衡板使用壓力感測器，使玩家能與 Wii Fit 遊戲互動。本章會說明壓力感測器的物理觀念，包含產生的資料以及如何在遊戲中操控資料產生互動效果。

第二十四章：3D 顯示器

Sony 的 PlayStation Move 和微軟的 Kinect 使用光學追蹤系統偵測玩家遊戲控制器或手勢的移動。本章將解釋光學追蹤背後的物理原理，並說明如何將此技術應用在遊戲中。

第二十五章：光學追蹤

當電視和行動遊戲裝置競相採用 3D 顯示時，許多相關的技術應運而生。藉由瞭解需要眼鏡的立體顯示技術和無需眼鏡的自動立體顯示技術，以及令人期待的全像式和體積式立體投影技術，開發者可以適當將這些技術引用到遊戲當中。

第二十六章：音效

音效對於遊戲來說相當重要，因為能讓玩家有身歷其境的效果，然而至今仍未有遊戲物理相關的書籍提到這一塊。本章著重於音效的物理觀念，包含音效速度以及都卜勒效應。也會提及音效物理為何常在遊戲中被忽略，例如模擬外太空的爆炸時。

附錄 A：向量運算

本附錄介紹如何實作 C++ 的類別，此類別包含所有在 2D 和 3D 模擬程式中會用到的向量運算。

附錄 B：矩陣運算

本附錄實作一個類別，包含所有需要用來處理 3x3 矩陣的運算。

附錄 C：四元數運算

本附錄實作一個類別，此類別包含所有在 3D 剛體模擬程式中處理四元數時會用到的運算。

第一部分著重在牛頓的力學基礎中，像是運動學和動力學。運動學處理物體的移動，會談論到線性以及角速度與加速度。動力學則處理作用力和產生的移動。第一部分只是當作第二部分(包含剛體動力學)的入門知識。若讀者對於力學已有基礎概念，可以略過這部分。

第二部分著重於剛體動力學以及開發單一或多個物體的模擬。內容包含了數值積分、粒子與剛體的即時模擬以及剛體的連接。一般來說，此部分觸及到大部分遊戲設計師會考慮的物理引擎元素。

第三部分著重於物理模擬，主要目的是希望說明詳細的物理原理，以便開發物理模型時能適當將所需項目納入考量，並在不犧牲物理真實性的情況下，捨棄一些東西。這裡無法觸及到所有想模擬的東西，而是針對遊戲中常見的項目去模擬，像是飛機、船和球類運動等。

第四部份著重於廣義的數位物理學。這個主題與目前行動平台關係很深，像是智慧型手機（例如 iPhone），以及創新的遊戲平台（例如任天堂的 Wii）。內容主要包含了加速度計、觸控螢幕和 GPS，以及將這些元素應用到遊戲中的其他技術。這樣的主題其實並非多數遊戲開發者所認為的傳統遊戲物理學範疇，但必須知道這樣的技術在現代的行動裝置遊戲中扮演相當重要的角色。因此本書介紹了其背後的物理觀念，並希望讀者能應用於自己的遊戲中。

除了與即時模擬有關的資源外，在書後的參考書目中也提供關於力學、數學和其他技術主題（例如空氣動力學）有關的資料。

本書體裁

以下是本書使用的排版體裁：

定寬字

用來表示命令模式的電腦輸出、範例程式碼和鍵盤快速鍵。

斜體定寬字

用來表示在範例程式碼中的變數。

楷體字

用來介紹新名詞和表示 URL、變數、檔名和目錄、命令和副檔名。

粗體字

用來表示向量變數。

第二十章

觸控螢幕

隨著智慧型手機、平板和其他行動計算平台的快速發展，對於人與電腦的互動有著深遠的影響，因此無法否認現在已經進入到後 PC 的時代了。這些行動裝置不再使用傳統的滑鼠與鍵盤當作輸入，而是依賴觸控螢幕。本章即是要介紹幾種不同的觸控螢幕，包括如何運作及技術上的限制。請注意，此處也會延伸粒子模擬器的運用，以便和 iPhone 的電容式觸控螢幕結合。最終的產品會很類似滑鼠驅動的版本，但卻是個觸控驅動的模擬器。

本章主要著重在兩種常見的觸控螢幕技術：電阻式和電容式，不過接下來會簡單介紹許多不同類型的觸控技術。或許在不久的將來會出現更多奇特的裝置，尤其是在大型計算裝置的領域中。

觸控螢幕的類型

電阻式

電阻式觸控螢幕基本上是由許多微小的按鍵所組成的，有的甚至在一平方英吋中會有 4,096 × 4,096 個按鍵。所以雖然很像按鍵，卻又不是普通的按鍵。電阻式觸控螢幕至少由兩層導體所構成，其中間則是空氣間隙。當按壓螢幕時，空氣間隙就被關閉，上下兩層電路導通，按鍵就被按下。對此稍後會有更詳細的說明。

電容式

電容式觸控螢幕目前被廣泛應用於智慧型手機，對此稍後也會詳加說明。當手指的觸摸影響到螢幕玻璃下方的導線線路時，螢幕四個角落的電容會有變化，可因此求出觸控位置。此技術的限制是碰觸螢幕的物體必須能夠導電，例如手指。倘若戴上手套，就會無法感應到觸控，但這點可在手套上增加導電線來解決。

紅外線和光學成像式

紅外線觸控螢幕使用紅外線 LED 和光感應器陣列去偵測和解析是否有物體遮斷了某個 LED 光感應器路徑上的紅外線訊號。其使用的是行掃描的技術且是個很堅固的設計。

光學成像式是很新的觸控螢幕技術，主要優點是擴充性極佳。透過影像裝置和光源，經由解析物體產生出的任何陰影，感測出螢幕上所發生觸控的位置。

其他奇特技術：色散信號和表面音波式

當然還有其他奇特的觸控螢幕技術，但此處不會詳加說明。3M 公司有個系統能偵測觸控對於玻璃所產生的機械能。每個感測器感知到震動的能量，即可決定出觸控位置。

另外還有表面音波式，其會偵測螢幕表面上超音波的變化。

逐步解析物理觀念

電阻式觸控螢幕

電阻式觸控螢幕屬於被動式技術，因為其觸控偵測不需要碰觸螢幕的物體的參與。電阻式觸控螢幕在被非導體（例如筆或戴手套的手指）按壓時也能有所反應，這一點是其優於主動式技術（例如：電容式觸控）的地方。在過去，電阻式觸控螢幕只限於接受單一輸入，此處要討論的即是此類型，但其實也可以設計成同時支援多個輸入，亦即支援多點觸控。

一維電阻式觸控感測器

為了讓人更能融入討論的議題內，首先從一維觸控螢幕開始說起。假設我們已經建立如圖 20-1 所示的機器。

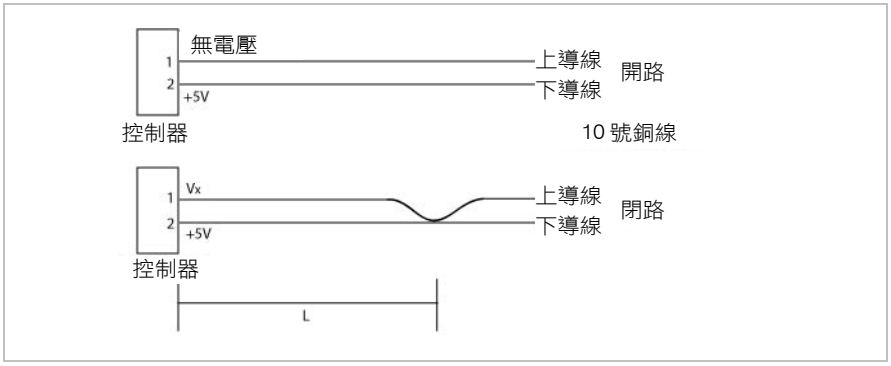

圖 20-1：線性電阻式觸控感測器

此處的感測器有兩個狀態：開路（*Open circuit*）和閉路（*Closed circuit*）。在開路狀態時，控制器會提供 5V 訊號給接腳 2，然後等待接腳 1 傳回的電壓。若沒有觸控發生，則線路之間不會有所接觸，呈現開路狀態，接腳 1 就沒有任何電壓。反之，當觸控發生時，會呈現閉路狀態，接腳 1 會有電壓出現。所以判斷接腳 1 的電壓，可以偵測是否有發生觸控。

這種只看電壓是否存在，而不管其值的感測器稱作數位感測器，只能偵測兩種狀態：開或關（1 或 0）。目前為止所擁有的只是一個簡易的按鍵，還不是觸控螢幕。進一步來說，當按下按鍵時，除了觸發一個事件外，我們還希望能從按壓的位置開始，沿著線路 *L* 輸入一個值。

若要這樣做的話，控制器必須耐心等待接腳 1 的電壓。當接腳 1 有電時，開關（上下線路交錯處）會產生「ON」信號，然後經由排線傳給控制器處理，進一步計算施壓處的座標位置，亦即標記為「*V*x」之處。此即其名稱的由來。電流、電壓和電阻在歐姆定律中都有相關，公式如下：

$$V = IR$$

其中 *V* 是電壓，*I* 是電流，*R* 是電阻。若想不起來這些名詞的物理定義沒有關係，因為現在比較重要的是他們彼此間的關聯性。以此處的電路為例，電流 *I* 會是一個常數。

因為控制器會測量接腳 1 的電壓值，所以可以求出電阻值：

$$R = V/I$$

在電流 I 是常數且電壓 V_x 已知的情況下，可以將電壓 V 設為（5 - V_x）來求出電路的電阻值：

$$R = (5 - V_X) / I \quad (1)$$

因為電阻（導線）必須有電壓變化，所以兩個值必須有差異。每個導體有其內部電阻，因此可透過測試來決定每單位長度的電阻值是多少歐姆，再依此求出總電阻值：

$$R = 2rL$$

其中，r 是前述的每單位長度歐姆值，R 是電路的總電阻。請注意，此處將 L 乘以 2 是考慮到控制器與接觸點間的來回。若將前述公式中的 R 以此替換，變成：

$$2rL = (5 - V_X) / I$$

則最終為：

$$L = (5 - V_X) / (2rI)$$

其中 L 是唯一的未知數。為了方便說明，假設測量到的電壓是 4.95V 且使用 24 號銅線。由標準電機工程的書可以知道每公尺是 0.08422 歐姆。假設將恆流源設定為 50 毫安培：

$$L = (5 - 4.95V) / ((2)(.08422 \text{ ohms/meter})(.05A))$$
$$L = \text{離控制器 } 5.9 \text{ 公尺}$$

如同所見，每個材料的每公尺電阻值、提供的恆電流以及電壓感測電路的靈敏度，都必須適當調整以確保在適當維度，控制器才能偵測觸控事件。在電阻式觸控螢幕中，線路相當微小，因此每公尺電阻值較高，螢幕就能偵測較小的距離。

四線電阻式觸控螢幕

藉由一些修正，可將前述模型擴充為二維模型。在四線電阻式觸控螢幕中，基本結構有四層，且有四條引線，其中三條在任何時刻都會被使用。佈線方式如圖 20-2 所示。

其中含有 X 和 Y 引線的兩個矩形實際上會互相重疊，但在此處為了說明所以不重疊。

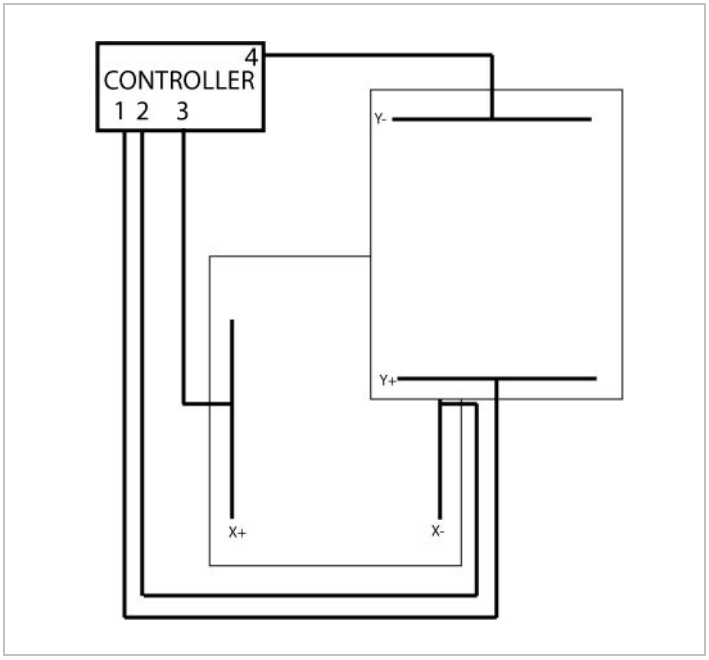

圖 20-2：四線電阻式觸控螢幕

現在其稱為「四線」的原因就很清楚了。然而，請記得在任何時刻只會有三條引線動作，基本結構如圖 20-3 所示。

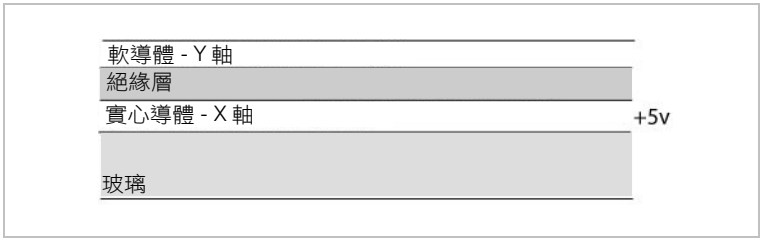

圖 20-3：四線電阻式觸控螢幕的結構

第一層是「軟導體」（flexible conductor），再來是「絕緣層」，然後是「實心導體」（solid conductor）。當手指觸摸螢幕時，絕緣的兩層導體就有了接觸。這些導體都是 *ITO*（錫氧化物）薄層，兩端都有導線，如圖 20-2 所示。

為了簡化其運作的說明，將三種可能狀態列在表 20-1 中。

表 20-1：四線電阻式觸控面板三種可能的狀態

動作	接腳 1	接腳 2	接腳 3	接腳 4
等待觸控偵測	開路	開路	數位輸入[上拉]	接地
讀取 X 位置	電壓偵測	接地	電壓源	開路
讀取 Y 位置	電壓源	開路	電壓偵測	接地

「電壓偵測」是表示晶片會偵測該接腳上的電壓，「電壓源」是表示接腳會提供電壓，「開路」則表示接腳未被使用。觸控事件發生的順序是接腳 1 和 2 呈現開路狀態，接腳 3 則設成數位數入，且有內部上拉電阻，因此會有電壓進入。當手指觸摸螢幕時，兩層導體有了接觸，接腳 3 會接地。當晶片控制器感應到接腳 3 的電壓下降，則會移動到表 20-1 中第二列的情況，去讀取 *X* 的位置。

要讀取 *X* 位置，下層導體由接腳 3 到接腳 2 呈現通電狀態，會形成電壓梯度。當觸控發生時，上層導體會接觸到下層已經通電的導體，而接腳 1 和上層導體相連接，會產生電壓通知晶片控制器。此電壓值取決於觸控發生在電壓梯度中何處，很類似前面線性觸控的例子。 一旦求出 *X* 的位置，晶片控制器就會移到下一列讀取 *Y* 的位置。

讀取 *Y* 位置的方式很類似，但過程相反。電壓源變成接腳 1，且和接腳 4 之間會產生電壓梯度，而接腳 3 會偵測電壓梯度上和距離相關的電壓值。當晶片控制器以每秒約 500 次的頻率不斷重複偵測 *X* 和 *Y* 的位置，則使用者就不會請注意螢幕其實並非同時感應到 *X* 和 *Y* 座標。

由於四線電阻式觸控螢幕是最簡單的二維觸控感測器，所以其耐久性有些問題。最大的缺點是因為每層導體之間需要絕緣層，所以至少有一層必須是軟導體。因為第一層導體不斷地彎曲，使得表層 ITO 塗層出現細小裂紋，導致非線性和精確度降低。其他的電阻式觸控螢幕則使用額外的導電層以消除對軟導體的需要來克服此問題。同時，這些處控螢幕也被調整為支援多點觸控。我們在稍後會討論電容式觸控的多點觸控運作方式。

電容式觸控螢幕

電容式觸控螢幕在玻璃螢幕鍍上一層透明導體，當手指或其他導體觸摸到螢幕時，靜

電場受到干擾，造成電容值的變化。若要瞭解電容式觸控螢幕如何運作，接著就要來大略了解一下電容。

電容可以簡單由兩個導體（通常是薄板）組成，且中間以絕緣物質隔開。若有電壓通過兩導體，就會產生電流，電荷量也會逐漸累積。一旦薄板兩端的電壓等於電源電壓，電流就會停止。電容量即是薄板間的電荷量。前面有提到電阻式觸控的一個問題就是第一層導體不斷地彎曲以導通線路造成的機械故障，但電容式觸控卻無此問題。一個電容可以動態地由任何兩個導體以絕緣體隔開組成。玻璃就是一種良好的絕緣體，所以很容易觀察到當手指接觸以玻璃隔開的導體時電容增加的現象。如此一來，手指或觸控筆不需要產生任何機械動作，仍然可以將變化傳給感測器以便判斷觸控的位置。

在行動裝置上，這樣的觸控方式可分為自電容型（*self capacitance*）和互電容型（*mutual capacitance*）。

自電容型

生活在乾燥冬天的人，有時會被靜電放電的現象給嚇到，因為人體就是個良好的電容，其值大約是 22 pF，這樣的特性稱為「人體電容效應」。自電容型觸控螢幕利用的物理特性是一個獨立導體的電荷量增加時，能夠提高一伏特的電位。當手指碰觸到觸控螢幕時，手指成為人體固有電容的導體，此時在玻璃另一側的感測器會偵測到電位上升。因為感測器是在玻璃（良好的絕緣體）的另一側，所以實際上不會有任何靜電現象，也就不會產生像觸摸到金屬車門時那種觸電感。這樣能產生相當強的訊號，但無法準確解決多點觸控的問題。因此常和接下來要討論的互電容型觸控技術結合在一起。

互電容型

互電容型觸控螢幕是由呈網格排列的多個獨立電容所構成，且探測電流會經過網格中的每一行或每一列。由於電容的充放電，使得系統可以偵測到每個獨立電容的值。正如剛才所討論的，人體是個良好的電容，因此靠近網格中的電容時會改變局部電場。被手指或其他導體觸碰到的電容會讀取到比一般情況還低的值。因為每個電容可以獨立進行掃描，所以能準確辨識觸控發生的位置，也就適用於多點觸控的情況。可以將

此系統想像成是為螢幕表面上的電容拍照。使用類似影像處理和邊緣偵測的演算法，此系統可計算觸控事件的程度。

程式範例

本書第八章粒子爆炸的範例即是使用觸控螢幕，而非滑鼠輸入。

針對 Cocoa touch Objective-C 事件的程式碼如下：

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch* touch = [[event touchesForView:self] anyObject];
    firstTouch = [touch locationInView:self];
    self.status = YES;
    [self trigger];
}
```

其中，firstTouch 是由標頭檔的 CGPoint firstTouch 所定義的。CGPoint 是一個 Cocoa touch 物件，儲存顯示視窗座標系統中的（*x*,*y*）座標。稍後即可在程式中使用 firstTouch.x 和 firstTouch.y 定位粒子爆炸。

雖然這與滑鼠驅動的程式很類似，但有個很大差別是，電腦一次只能認到一個滑鼠游標，但若使用觸控螢幕，可將程式修改為支援多點觸控。

多點觸控

在 iOS 中，必須先將自己 View 中預設值是 NO 的 multipleTouchEnabled 屬性改成 YES，以便處理多點觸控的事件。接著要建立一個類別去追蹤多個 TParticleExplosion 結構。接著就很簡單的依照每個觸控點的起始位置分別驅動爆炸。儲存多個觸控事件起始位置的 Objective C 程式碼如下所示：

```
- (void)storeTouchPoints:(NSSet *)touches{
    if ([touches count] > 0) {
        for (UITouch *touch in touches) {
            CGPoint *point = (CGPoint
            *)CFDictionaryGetValue(touchBeginPoints, touch);
            if (point == NULL) {
                point = (CGPoint *)malloc(sizeof(CGPoint));
                CFDictionarySetValue(touchBeginPoints, touch, point);
            }
            *point = [touch locationInView:view.superview];
        }
    }
```

其中，CFDictionaryRef 是個不可變的字典型態物件，允許複製物件和其值。此範例最後要考慮的是，因為同時模擬多個物理行為，所以可能需要降低時間間隔的頻率，讓動畫能順暢地播放出來。多點觸控會讓程式設計變得複雜，但物理部分卻是相當簡單的。實際上要如何處理多個事件可以參考自己所用程式語言的教學文件來開發。

其他考量

觸控螢幕一個主要的優點是，其佈局和觸發動作完全以軟體為基礎。亦即，如果某個按鍵與螢幕當前的佈局無關就可移除，並將多出的空間讓給其他相關的控制項使用。接下來我們會討論其他較不明顯的考量。

觸覺彈力回饋

沒有實體按鍵也就表示使用者必須完全依賴視覺來操作螢幕上所有的控制項。但若使用有實體按鍵的鍵盤，則可以藉由按鍵的實際位置或相關的觸覺和音頻回饋，以知道是否按下正確的按鍵。

這種幫助使用者和虛擬物件互動的技術稱為「觸覺彈力回饋」（*haptic feedback*）。最先將此技術導入到遊戲中的是像 *Motocross* 這樣的街機遊戲，當遊戲中發生衝擊時，手把會跟著震動。不過目前這已經是許多電玩遊戲的標準配備了能透過震動來通知使用者有某個事件發生。

在觸控螢幕的領域中，觸覺彈力回饋功能可以讓使用者知道成功按下按鍵或發生某個觸控事件。甚至有些觸控螢幕在被按壓時，整個螢幕都會震動。但這樣仍無法允許「盲打」，因為其僅是動態回應使用者的觸控，而無法為不同按鍵提供不同的觸控回饋效果。

遊戲中搭配觸控螢幕

因為觸控螢幕平面的特性且缺乏固有的觸覺彈力回饋，所以適合和遊戲中的一些控制項搭配來互動。若要在遊戲中建立一個逼真的鍵盤會需要不少物理模擬行為，因此很少遊戲會出現裡頭的角色坐著，並用標準鍵盤打字。

使用觸控螢幕來控制遊戲中物件的話，可以避免一些額外的物理模擬需求，同時又能維持真實性。若使用觸控技術的話，玩家可能需要脫下手套，以便使用電容式螢幕。最後，前面所提到的特殊觸控技術也都能讓遊戲設計更富有創意。例如，遊戲中低程度的爆炸可用於觸動這些表面音波式或偵測機械能的觸控螢幕。

與滑鼠輸入的差異

對於遊戲開發者來說，必須要考慮到觸控輸入遊戲和傳統的滑鼠／鍵盤輸入的差異。對於電視／電腦遊戲的開發者來說，滑鼠和鍵盤的結合帶來的速度以及精確度是其他輸入方式難以比擬的。因此，許多第一人稱的線上射擊遊戲會將觸控輸入和滑鼠／鍵盤輸入分開進行遊戲，避免讓後者的玩家享有不公平的優勢。在我們使用多個遊戲設備和行動裝置平台的觸控螢幕後，更是感覺滑鼠／鍵盤有顯著的優勢。

用手指觸摸螢幕形成的橢圓形接觸印痕是取決於使用者用哪根手指、按壓的力道以及手指方向。但使用者通常感知到的是位於橢圓形印痕的中心接觸點，所以必須做適度調整。這通常會由作業系統來計算出單一接觸點，再透過 API 傳給遊戲，但這樣做會犧牲準確性，也無法為特定使用者做校正。

觸控螢幕另一個固有的缺點就是需要觸摸螢幕。這表示當要控制遊戲中某個元素時，玩家的手勢必會擋住螢幕。所以若是玩第一人稱射擊遊戲，勢必很難跟使用滑鼠／鍵盤的玩家相互競爭。

再者，滑鼠游標懸停的功能不適用於觸控輸入。假設有一個遊戲，在使用者將滑鼠游標移到某個物體之上時並就會觸發某些事件，而且與用滑鼠點擊相同物體產生的事件不同。然而，若使用觸控輸入的話，因為該物體會被觸發螢幕的任何物體（例如手或觸控筆）給遮住，所以就算滑鼠游標懸停在該位置，使用者也看不到。

自訂手勢

最後，可能作為觸控輸入的遊戲還有使用自訂手勢。使用者能在螢幕上畫出一個形狀，然後程式會辨認出是某個手勢，再執行相對應處理。因為這比較偏圖形識別，而非物理學，所以不在這多著墨，但有興趣的讀者可以參考 Dan Saffer 所著的《*Designing Gestural Interfaces*》一書（由 O'Reilly 出版）。