逢 甲 大 學

資訊工程學系專題報告

簡易3D多人線上遊戲系統

學　　生：詹欽賢（四乙）
　　許志信（四乙）
　　何明政（四乙）
　　鄧欣宗（四乙）

指導教授：楊東麟

中華民國九十二年十一月

逢甲大學
資訊工程學系
專題報告

簡易3D多人線上遊戲系統

92

目　錄

圖 目 錄

表 目 錄

摘　要

現在市面上充斥著許許多多令人目不暇給的遊戲軟體，據統計2003年7月份到10月份間，共上市了106套電腦遊戲，電腦玩家們可能不僅喜歡玩遊戲，更對遊戲的製作充滿好奇與熱忱。

我們政府提出了『兩兆雙星』計畫，其中，雙星產業之一的『數位內容』更包含電子遊戲、音樂、動畫、及網路服務等領域，政府期盼能由二〇〇一年產值達新台幣一三三四億，到二〇〇六年時，產值再拉高到新台幣三七〇〇億元。因此，製作遊戲軟體的利基不言可喻。所以我們想把對於網路線上遊戲的愛好，轉為學習相關技術和遊戲的設計，利用專題實作的機會實際建置一個簡易的3D多人線上遊戲，從整體系統的架構、規劃乃至於分工實作，最後整合成一個完整的系統。我們的實作成果和專題報告也可以給有相同喜好的人做參考。

本報告的各個章節，包含以下的內容：

- 第一章 導論
  簡述當初鑽研之動機及系統欲達成之功能。
- 第二章 需求分析
  在本章中，我們探討遊戲軟體產業未來的競爭力以及針對台灣市場走向及玩家實際心得之解析，最後並定義出系統的需求。
- 第三章 系統分析
  開發系統之前，須對開發工具及其原理有充分的了解和正確的選擇，並適當地使用系統分析方法來設計。
- 第四章 系統設計與規劃
  本章中，我們說明系統的整體架構，以及各個子系統的基本架構、相互關係、和運作流程。
- 第五章 系統開發與整合
  首先介紹我們分工的情形，再來，就是各個子系統實作的技術與方法。
- 第六章 心得與結論
  本系統特色、組員們的心得、與未來尚需改進的地方。

第一章 導 論

1.1 概述

本專題是以開發 3D 線上遊戲並以主從式連線模式進行的系統。玩家必須安裝完成用戶端的主程式，並經由網路連線到伺服端、登入成功後，才能進入遊戲。在遊戲世界中，玩家會依劇情來扮演主角，並藉由破解劇情任務、不斷地提升等級等方式，來進行遊戲。

1.2 研究動機

資訊時代來臨，凡事講求實事求是，於是乎，人們少了幻想的空間。在暢銷電影—魔戒中，大地不僅只被人類支配，它也存在許多不同的種族，其中，巫師擅長神秘的法術，精靈除了是幸運的象徵之外，亦有使用弓箭戰鬥捍衛家園的能力。除此，不僅人類有善惡之分，其它種族亦是如此，整個世界彷彿被塑造成光明與黑暗的對比。但這一切，在現實生活中根本不可能出現，但是，除了電影之外，遊戲帶給我們更耀眼的希望。

台灣政府提出了『兩兆雙星』[1]計畫，其中，雙星產業之一的『數位內容』包含電子遊戲、音樂、動畫、及網路服務等領域，政府期盼能由二〇〇一年產值達新台幣一三三四億(其中以多媒體電腦軟體及網路服務為主)，到二〇〇六年時，產業產值再拉高到新台幣三七〇〇億元。因此，製作遊戲軟體的利基不言可喻。

綜合上述原因，興趣與前途，便成了我們製作本專題的主要動機。

[1]見中華招商網：
http://investintaiwan.nat.gov.tw/moea-web/InvestOpps/Type/2-2/index-c.htm

1.3 研究目標

藉由本次專題實驗，我們希望能逐步地揭開遊戲的神秘面紗，實作一套具有下述功能之系統：

- 主從式系統可以多人同時上線；
- 2D 人物/3D 環境場景；
- 角色扮演劇情；
- 即時戰鬥計算；
- 即時交易處理；
- 即時聲光效果；
- 聊天系統。

第二章 需 求 分 析

2.1 遊戲軟體產業的探討

在我們開始製作遊戲之前，須對目前市面上的遊戲軟體種類具些許的認知，並對國內及外的市場走向有一定程度的了解，才能初步確定遊戲系統的定位。在介紹完遊戲軟體的種類及全球產業現況後，我們會針對台灣市場以及廠商作進一步的探討。

2.1.1 遊戲軟體種類[2]

全球遊戲軟體大致可以區分為大型遊戲機（Arcade）、電視遊戲器（game player）、單機版遊戲（PC game）及線上遊戲（On-line game）。電視遊戲機台所佔比重最高，約佔整體遊戲市場 5 成左右，目前市場主要競爭者為SONY之PS系列、微軟之XBOX及任天堂之Game Cube，目前以SONY之PS系列市佔率為最高。

2.1.2 全球產業現況[3]

就全球遊戲市場而言，不管是軟硬體，每年均呈正成長走勢，2001年更呈現 30%的高成長，主要原因為該年度 SONY 及微軟分別推出新機種 PS2 及 XBOX，使得該年度硬體銷售比重提升到 34%，導致整體市值提高，由於遊戲產業軟體銷售比重一向較硬體高，在 2001 年硬體比重大幅提高後，預估之後陸續推出之軟體，將使得硬體比重逐漸下滑，軟體比重將重回 8 成比重水準，而整體市場，在少了新硬體投入的動力，未來幾年整體產業成長將無法與 2001 年高成長相比，成長率將逐漸下滑，但整體產值仍是呈現正成長，主要成長動力來自軟體市場，預估軟體市場 2002 年仍有 16.5%之成長，因此遊戲軟體未來將成為帶動整體遊戲市場成長動力來源。

---

[2] 本節資料整理自http://intra.yuanta.com.tw/PagesA2/hot_issue/9108gam.html
[3] 本節資料整理自http://intra.yuanta.com.tw/PagesA2/hot_issue/9108gam.html

2.1.3 台灣市場規模[4]

台灣號稱硬體製造王國，尤其在 PC 相關產品上，因此國內 PC 普及化的程度相當高，再加上自 86 年起微軟 Windows 95 作業環境接近成熟之際，CPU、DRAM 等硬體產品價格大幅滑落，以致低價的電腦普及，使得 PC 多媒體功能提升，電腦遊戲畫面、聲光效果越顯精緻，及多元化，更成為國內電腦遊戲蓬勃發展的原因之一。

其次，由於台灣市場規模太小，易為電視遊樂器業者所忽視，加上電視遊樂器(TV Game)不論是硬體及軟體都由日、美商壟斷，且其開發成本高昂須給付高額權利金費用下，國人能切入遊戲機軟體市場是微乎其微。但台灣挾其為全球最大 PC 製造產地之便，提供了國人切入 PC 遊戲軟體相當好的契機，因此國人在 PC 遊戲軟體之研發製造，幾乎可以不假他人之手，自行研發出相當多元且高品質之軟體。且在 2001 年時，國內及全球經濟不景氣，但在國內個人相關應用軟體市場規模成長亦呈現趨緩的情況下，遊戲軟體產業，卻在遊戲軟體廠商的成功行銷模式、及產品不斷推陳出新的推波助瀾下逆勢成長。故台灣業者非常重視遊戲產業並以 PC 遊戲為主流。

近年來網路的快速發展，民眾對頻寬的需求日增。且隨著政府在開放固網事業後，加上寬頻硬體設備的進步、寬頻上網費用的降低，讓寬頻上網使用愈來愈普遍。另外，網路咖啡店的興起以及線上遊戲強調「社群」的觀念，在在吸引青年人結伴消費及上網玩遊戲，為台灣線上遊戲成長急速加溫。

而 Forester Research 的預估資料則指出，今年全球約有 1800 萬使用者加入線上遊戲的行列，因此線上遊戲便成為遊戲軟體業者下一波競爭的市場所在。

[4]本節內容整理自http://intra.yuanta.com.tw/PagesA2/hot_issue/9108gam.html及 http://www12.brinkster.com/losi/hot1.htm

2.1.4 國內廠商[5]

遊戲軟體產業的上中下游可概分為三大區塊，即研發、行銷與發行、及通路三部份。而國內上中下游超過 80 家業者爭奪有限市場，其中較具規模與知名度的有：研發見長的奧汀、宇峻、聖教士、昱泉、及大宇;以通路起家的智冠、第三波、華義、及華彩等;以行銷為主的遊戲橘子、松崗及英特衛等公司，另尚有國外業者如微軟、美商藝電、及光榮等。

由於台灣市場太小，但市場玩家卻相當多，因此為達經濟規模，部份規模較大的廠商均進行上下游整合，例如智冠、大宇、第三波、華義、及華彩陸續跨足研發、行銷、及通路領域，因此產業分工愈來愈不明確，競爭相當激烈。

即使國內從事遊戲軟體廠商近兩年來投入者眾，但由於軟體研發至成品上市時間耗日費時，短期間新進入市場者尚難取得一定市場領導地位，早期進入市場者如智冠、大宇、昱泉、華義、及橘子等均已取得一定市場地位，此外擁有通路也是產品是否能熱賣之主要關鍵，在綿密的行銷通路下，無論是自製或代理產品均可在其銷售網下達到最大行銷效果。

近年來由於線上遊戲持續發燒，使得單機板遊戲軟體逐漸萎縮，多家原本從事單機軟體廠商均紛紛推出線上遊戲軟體，以符合市場趨勢，由於韓國政府全力扶植線上遊戲產業，使該國線上遊戲產品無論在質與量上均具有國際水準，因此國內線上遊戲廠商初期大多以代理韓國產品為主，其中較熱門產品如天堂，即帶給遊戲橘子相當好的業績，目前為國內市佔率次高之線上遊戲。但由於代理產品均需給付約 3 成之權利金，故國內廠商也積極研發自製產品。

[5] 本節內容整理自http://www.nettrade.com.tw/stock/a/01-08-03a.htm及 http://intra.yuanta.com.tw/PagesA2/hot_issue/9108gam.html

由於線上遊戲可同時容納數千人上網對戰，同時亦可提供聊天室的新興遊戲模式，於是深深吸引國內玩家的流連及注目，也因此形成了一股社群的力量。此外，網路頻寬問題陸續改善及上網撥接費用下降的趨勢，亦有利於線上遊戲玩家人數之成長。在線上遊戲的風行下，台灣也循著南韓線上遊戲發展的模式，由線上遊戲帶動週邊的網咖業、ISP 業、網路設備業、及通路商（傳統通路及一般通路）的蓬勃發展。

2.2 網路遊戲

經過上一節的探討後，我們可以確定系統的定位是個人電腦上的遊戲軟體，且要透過網路連線來進行遊戲。因此，在本節中，我們對網路遊戲的背景及分類作詳細的介紹，以定出系統的網路連線模式。

2.2.1 背景

在對網路遊戲做進一步的介紹之前，我們必須對幾個相似的名詞有一基本認識。對於所有藉由電子型態呈現的遊戲，我們都可稱之為「電子遊戲」，其中包括了「電腦遊戲」、「電視遊樂器遊戲」、以及「大型電玩遊戲」等分類。最早的電子遊戲出現於 1970 年代初期，是由美國麻省理工學院所開發在早期電腦上執行的遊戲「太空大戰」（Space War）。

在 1972 年，一家叫做 Syzygy 的公司首先將電子遊戲製成商品販售，成為後來遊戲工業的先驅。此後，隨著家庭電視遊樂器的出現，以及個人電腦的普及，電子遊戲的發展也大致以這兩者為主要平台。而自從微軟推出了 Windows 作業系統後，其統一的發展介面大大有利於遊戲發展環境，再加上從 1995 年後開始的網際網路風潮，以及電腦 3D 圖形加速的技術，電腦遊戲市場正逐漸威脅到電視遊樂器的地位，其中又以近來盛行的網路遊戲為最。

就前述的電子遊戲發展歷程來看，網路遊戲可說是電腦遊戲的一類，指的是所有透過網路來進行的電子遊戲[6]。

2.2.2 型態分類[7]

網路遊戲並不是最近幾年的事，早在 WWW 尚未誕生之前，現在網路遊戲的前身 MUD（Multi User Dungeon）就已經存在於學術網路上了。廣義來說，目前台灣的網路遊戲可分為以下三類：

- 第一種是目前最流行的線上遊戲（online game）—

此類遊戲通常會有測試版讓玩家試玩，並持續 1~2 個月，在此之間是不收費的。玩家透過各種管道，如上遊戲網頁下載、免費索取遊戲光碟等，安裝完成用戶端主程式後便可連上遊戲伺服器與其他玩家共同進行遊戲。而在正式版推出後，遊戲公司便藉由會員收費制度，發售點數卡及月費卡，待玩家購買後上網註冊及儲值，才能在上線。因此，線上遊戲業者主要的收益來自遊戲玩家連線帳號的使用時數費，和單機版遊戲或區域網路型遊戲主要收益來源為販售遊戲套裝軟體的經營模式並不相同。例如「天堂」、「仙境傳說」、「無盡的任務」、以及「天翼之鏈」等國內目前知名遊戲，都是採取此種方式(圖 2-1)。

[6]本節內容整理自http://eteacher.edu.tw/game_1.htm#3
[7]本節內容整理自http://eteacher.edu.tw/game_1.htm#3

圖 2-1 仙境傳說點數卡及其週邊[8]

其實線上遊戲可以說是學術網路上相當風行的 MUD 的多媒體版，又可稱為多人線上角色扮演遊戲（Massive Multiperson Online Role Playing Game，MMORPG），如圖 2-2 為線上遊戲介面。即然名之為「角色扮演」，每個玩家必須選擇特定的性別、職業、種族、以及能力，在遊戲過程中玩家必須以完成指定任務或是消滅怪物等方式來提昇角色的能力。等級提昇所帶來的成就感，加上不同玩家的角色之間的互動，是此類遊戲最吸引人的地方。

[8] 圖片來源：仙境傳說官方網站http://ro.gameflier.com/main.asp

圖 2-2 線上遊戲介面[9]

- 第二種區域網路型遊戲（LAN Game）—

這類遊戲通常是透過區域網路來進行，除此之外，玩家也可以連上區域網路之外的伺服器，不過此伺服器擔任的是「中介」的工作，使用者在找到其他玩家之後，就可以離開伺服器進行連線遊戲。在網咖中相當流行的多人對戰遊戲，例如「戰慄時空」(Counter-Strike)(圖 2-3)與「世紀帝國」，就是此類遊戲的代表。以網咖中相當普遍的連線對戰遊戲「戰慄時空」為例，每個玩家可以選擇要扮演警察或是恐怖份子，然後分成兩個陣營對戰，場面相當逼真刺激，你可以與恐怖份子展開一場街巷槍戰，也可以埋伏在金融大樓內等待霹靂小組上勾，許多青少年還特地為了追求勝利而定期舉行團隊的戰術演練，俗稱「練功」。

[9] 圖片來源：黃麒榮老師之投影片『遊戲e觀念—網路遊戲製作與發展』，p11

圖 2-3 CS 戰鬥畫面

- 第三種是提供小品遊戲的遊戲網站（Web Game）—

例如「宏碁戲谷」，這類網站大多採免收費的方式，只要填寫基本資料加入會員，就可以享受如象棋、橋牌、及麻將等回合式的線上遊戲，其中以網路麻將最受歡迎。又如目前及時通訊軟體 MSN，也可提供點對點式的小型連線遊戲(圖 2-4，圖 2-5)。此類遊戲由於每次不需要持續投入太多的時間，而仍舊具有真實的互動競賽特性，因此深受許多上班族的喜愛。

圖 2-4 MSN 訊息傳遞介面

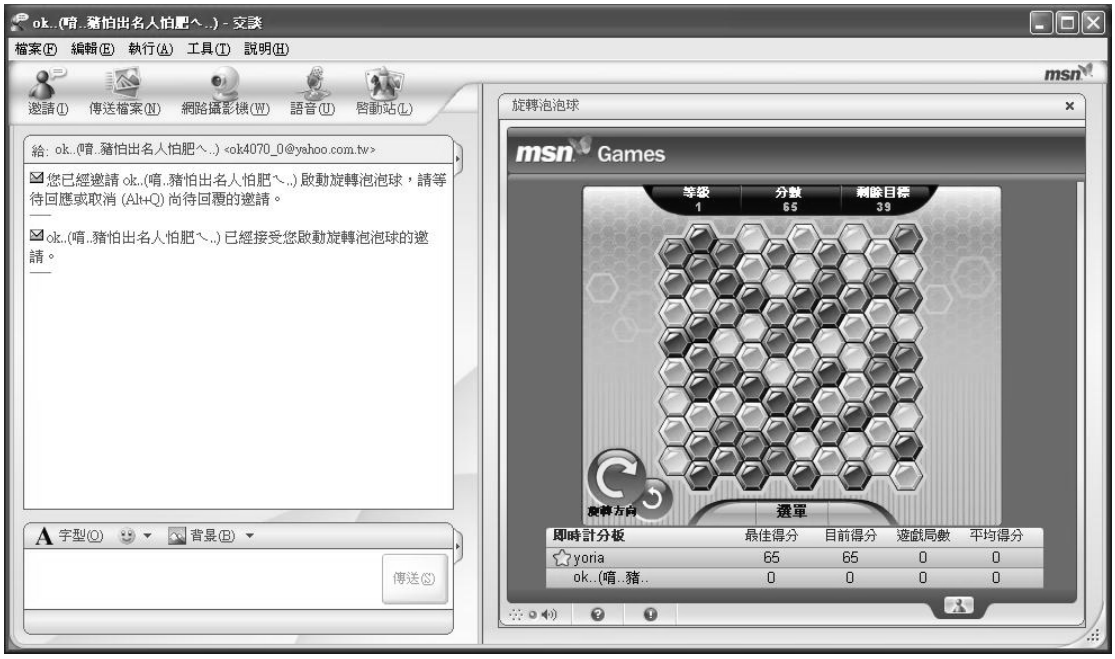

圖 2-5 MSN 連線遊戲介面

在這三類網路遊戲中，又以前兩類為青少年的最愛。線上遊戲講求的是角色扮演，玩家必須投入時間提昇人物屬性，因此往往必須與其他玩家合作，著重的是人際互動與培養角色的過程中所帶來的樂趣。區域網路型遊戲由於著重戰役式的對戰，不需要長時間的角色屬性提昇，因此主要提供的是緊湊刺激的聲光享受與宣洩效果。相較於過去單機版的電腦遊戲或是電視遊樂器，網路遊戲裡的角色是由所有參與遊戲的「活生生」的玩家所操縱，劇情並非既定的線性發展，也沒有特定的結局，不管是對話、情節、或劇情在線上玩家錯綜複雜的網絡互動下，都能夠有千變萬化的展現方式。

除此之外，線上遊戲亦受業者青睞。其除了可抑止盜版問題之外，更提供過去賣斷整套軟體所缺乏的成熟營運模式，也為網路將改造軟體「製造業」為「服務業」作了最佳詮釋。台灣線上遊戲業者主要的營收來自玩家的連線時數費，包括月費制及點數卡制，約佔營收的八成以上，其餘還有銷售使用者端程式軟體、向網咖業者收取授權費、週邊產品、及廣告等收入。因此線上遊戲除了可延長遊戲生命，創造週邊商機外，更可擴展消費層年齡，時數費更是穩定的收入來源。

2.3 系統需求

在系統定位明確後，我們要建立系統的需求。在本節中，我們首先對各類遊戲(以執行平台分類)在市場銷售的情形作了解，再來，便針對兩款線上遊戲的玩家，他們所提供的意見作分析，最後確定系統需求。

2.3.1 市場分析[10]

如圖 2-6 所示，截至 2002 年 12 月底為止，我國上網人口達 859 萬人，網際網路連網應用普及率為 38%。而在這之中，至少有多達 270 萬的網路玩家。

圖 2-6 上網人口

(資料來源：經濟部技術處&資策會)

根據台灣網路資訊中心(TWNIC)的調查，截至 2003 年第 2 季止，上網人口已達 1,175 萬人，普及率 57.23%，其中寬頻使用人數達 937

[10] 本節內容整理自黃麒榮老師之投影片『遊戲e觀念—網路遊戲製作與發展』，p3~15

萬人佔 7 成，方式為 ADSL（64.4%）、CableModem（5.48%），ADSL 已成為最主要的上網連線方式。

以下便針對各類遊戲市場加以介紹：

- 線上多人遊戲：天堂、仙境傳說、無境的任務。

七年前由 UO 開創，直到前年天堂累積會員百萬，再由去年的仙境傳說拉到最高峰，現在為所有平台獲利最高的遊戲，但由於跟隨廠商過多造成遊戲數量也過剩，以台灣的 270 萬的使用者前三名約占 7 成。而今年發表的遊戲約有 60 套。

- 多人付費對戰：瘋狂坦克、炸彈超人、泡泡龍。

去年由韓國引進的瘋狂坦克最高曾創下 15 萬會員，而今年初的炸彈超人也有 7 萬的數字。這類的遊戲通常生命週期都在 3 個月到半年之間，因為社群性與耐玩性都沒有線上人數多。以致於玩家的忠誠度無法持之以恆，但是有項優勢，就是不會跟線上遊戲衝突到，這就是付費對戰遊戲一直都有不錯成績的主因。

- PDA：40 萬 PDA 使用者(25 萬 PPC)。

Plam 與 PockPC 的主要差異是技術資源與容量，PPC 目前佔有率與開發環境都比 Plam 來的好，是不錯遊戲開發平台，可惜 PDA 拿來當娛樂消費的習性尚未打開，在台灣雖有過 20 萬的使用者，但遊戲的銷售量都再 2 千套以下，接連著行銷通路的意願也跟著低落。

- Mobile：2300 萬使用者(超過 3 萬 JAVA 彩色機)。

今年最熱莫過於手機的發展，無論硬體、服務、通訊都有大幅度的成長，台灣是手機持有率世界第一，超過 2000 萬的使用者是一塊

很大的市場，不過目前還停在語音通訊階段，加值服務正起步中。

2.3.2 一般玩家的需求分析

台灣的線上遊戲目前最受歡迎的莫過於仙境傳說跟天堂了。在此節中，我們將整理從全國知名遊戲網站-遊戲基地，這兩款線上遊戲的討論區中，所引用過來的資訊，並予以分析。

- 仙境傳說玩家

如圖 2-7，在這次的投票，我們可以看到其中第 4、3、10 項是玩家最關心的前三名。由以其中第 4 項『LAG 成為歷史』為最，而此現象，本組的組員(欣宗)當初也經歷過此種椎心之痛，因此，既然號稱多人線上遊戲，玩家最基本的要求便是連線品質。其次，便是玩家資料的絕對正確、安全，過去也有很多這樣的事件在新聞上發生，玩家常常因為帳號被入侵、盜用，而導致一些虛擬的寶物被盜用等侵害玩家遊戲財產的事件，這是非常不能被接受的。至於最後一項，外掛程式，事實上是遊戲本身內容安排不當所產生的副作用，而這些不當主要包括，等級很難提升、及寶物難以獲得等，而導致玩家使用外掛程式(利用程式運作來幫他們玩遊戲)，這些都會直接反應在討論區中玩家的抱怨文章上面。

玩家會願意付費及上網玩遊戲，有兩大基本要求，第一是系統品質，第二便是遊戲內容。系統品質強調的是連上伺服端的速度及遊戲過程中網路的流暢等；而遊戲的內容便是其趣味性、耐玩度甚至是新鮮感。而仙境傳說最叫人詬病的是前者，也就是它的硬體。舉例來說，早期，當系統(伺服端)停機維修過後，玩家要登入簡直比登天還難，往往要耗上一天來重複著打帳號、密碼的動作。但即便如此，仙境傳說的會員人數仍屢創新高，甚至後來還拉下盤據遊戲龍頭多年的天堂，成為第一名，可見其遊戲內容是絕對夠吸引人的。

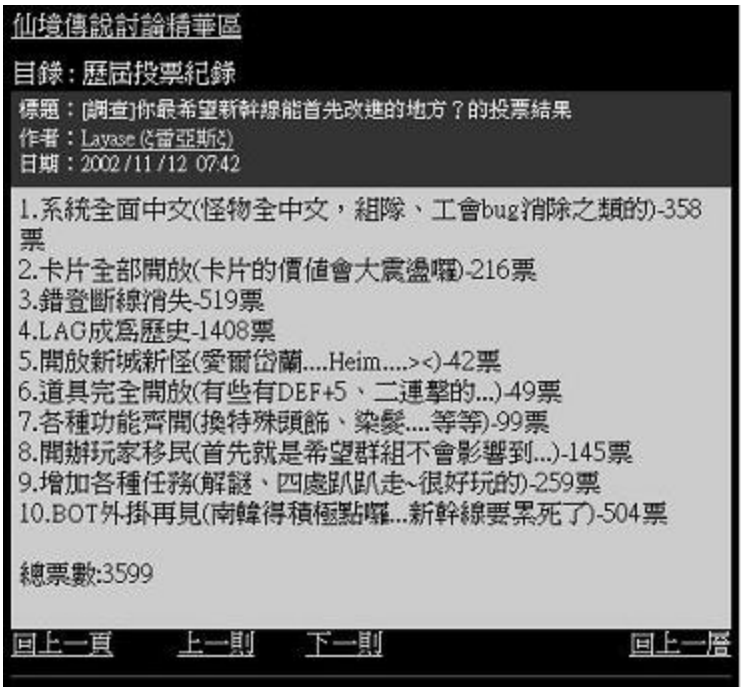

圖 2-7 仙境傳說投票紀錄(資料來源:www.gamebase.com.tw)

- 天堂系列玩家

在過去幾年，天堂一直是線上遊戲第一名，故玩家對於遊戲內容會失去較多的新鮮感，因此，在兩次的投票中我們可以發現，玩家最在意的便是遊戲改版後的功能及內容。也因此，天堂代理商(遊戲橘子)也決定要在明年引進全新的天堂 2 代，來取代舊有的遊戲系統。

而在圖 2-8 左邊的投票紀錄中，我們可以發現一個很好笑的選項『殺光所有白目』，其實，不只是在天堂甚至是其他線上遊戲，都會出現一種玩家，他們專門作一些惡劣的事，比如搶怪及亂殺人等，而這些人後來也就被稱為「小白」並成為遊戲中「過街老鼠」的角色。因此，在遊戲中必須要有管理者(Game Master，GM)來負責維持秩序，這些 GM 都是遊業者專職聘任的。

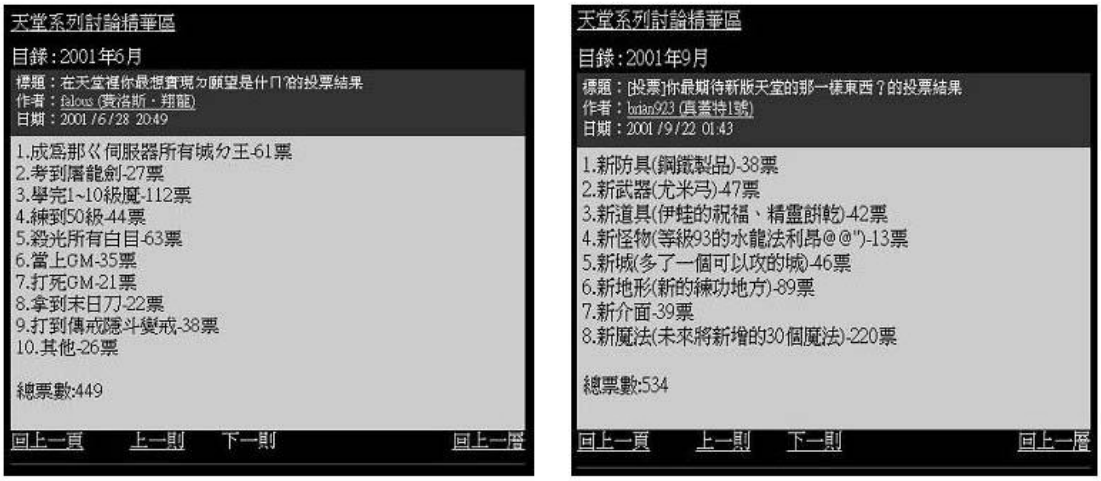

圖 2-8 天堂系列投票紀錄(資料來源:www.gamebase.com.tw)

2.3.3 建立系統需求

在市場分析一節中，我們發現到，網路普及率在未來幾年絕對有機會追上手機普及率(據調查，台灣目前的手機普及率已到達人手一機的情形)。因此，儘管國內現階段已充斥著許多線上遊戲，但多半來自國內大廠向國外的代理及引進，且在代理權利金高漲的情況下，國內大廠紛紛成立研發小組，準備由國人自行研發。

而在使用者需求分析一節中，我們更可以確定，線上遊戲受歡迎的程度是非常驚人的。在系統品質與遊戲內容兼顧的前提下，線上遊戲發展的潛力將會無與倫比。

因此，綜合第二章中各小節的介紹與探討，我們決定，系統要以打造一個多人連線的角色扮演遊戲為目標。在我們的遊戲世界中，每個人擁有自己的遊戲角色，藉由此角色可以與其他玩家產生互動，如透過聊天系統，與其他玩家聊天甚至相約探險打怪。而在遊戲畫面的表現上，則以 3D 環境來真實呈現場景、再輔以轉動視角、多角度的方式操作角色，以求達到無場景死角。而人物的建構則以 2D 的繪圖方式呈現。在遊戲內容方面，以劇情對話方式加上打怪系統(即時戰鬥的方式，搭配即時的音效播放)，創造出多元的遊戲故事。

將上述內容整理條列出來後，需求便確定如下：

- 即時動作 RPG 遊戲
- 即時聲光音效
- 2D 人物/3D 環境場景
- 主從式系統可以多人連線
- 聊天系統
- Windows 作業系統平台

第三章 系 統 分 析

3.1 系統分析方法論

不論是傳統的結構化系統分析(structured analysis)或是另一種目前非常盛行的物件導向分析與設計(object-oriented analysis and design)，在系統的開發過程中，都必須有一套可以遵循的方法。而每種方法論又各有許多變形，因此，在系統分析的過程吾人應該充分的了解各種不同方法的優缺點。

有些公司發展出自己的一套方法，或是修改軟體廠商及顧問所提供的方法。甚至大部份的資訊科技專家都認為沒有單一方法是所謂最佳的系統開發方法，也因此，我們在分析系統以便建構出系統初步模型時所採用的便是結構化分析跟物件導向式方法的結合。

在主從式系統間資料傳遞及處理我們使用 DFD(資料流程圖)來表現，而物件與物件之間(如玩家與怪物)的互動便以 UML 來表達出彼此的互動及關聯性。我們參考的資料整理節錄於附錄 A。

3.2 系統開發模式

硬體技術不斷成長，相對的，與之契合軟體的需求與實用性日益多元及複雜化，往往不是開過幾次會議便能通盤掌握。而隨著軟體開發的過程，process model 往往扮演重要的角色。它是軟體系統開發一連串的步驟及執行程序，當系統照一定程序開發時，它亦成為開發者重要的依據及規範。

許多專家經過實際經驗與研究分別定義出許多不同的模型，如：線性序列模型(linear sequential)、遞增模型(incremental model)、及正式方法模型(formal methods model)等各式各樣的模型。這些模型的理論，我們經整理後將其節錄在附錄 B。此外，在經

過一段時間研究及探討後，在下一段的內容中，我們提出在系統開發過程中該注意的事項。

3.2.1 3D 線上遊戲系統-開發過程模式分析

跟所有能在電腦系統上執行的軟體程式一樣，遊戲也是軟體的一種。但比較特別的是，遊戲的需求在一開始企劃專案時往往能很明確的表述及策劃出來，儘管現在市面上充斥著各式各樣的遊戲，但對遊戲程式撰寫者而言，較基本的需求及規格是相去不遠的。

儘管如此，遊戲開發廠商並不會因此公開他們的技術文件，這些都是他們的財產或者說是商業機密。因此，我們僅能在有限的參考資料及親身的經驗(玩遊戲)來將遊戲相關的需求及技術一層一層地抽絲剝繭開來。

基於這樣的原由，再加上上節對各模型定義的些許認知之後，我們主要以螺旋模型的理論再輔以原型。一開始我們採取原型模式，希望能在<第二章 需求分析>一章所述的工作一一完成後能盡快將軟體的原型定義出來，接下來便遵照著螺旋模型的技巧，以期在各個週期循序建構出越來越完整的系統。但這樣的方法卻會有某些嚴重的問題：

- 定義出軟體原型會相當耗時耗力，但卻是必須。
- 每一週期所需評估的風險相對增多，而這些風險隨著系統的不斷擴大往往會是由容易忽視的小細節所引起。
- 功能開發者兼使用者的我們可能會因技術上的不足而放棄了當初所定義的需求，也就是對需求作出讓步。

當然，問題或許不只這些，但上述確實是我們在開發的過程更須密切及謹慎注意的。

3.3 系統開發工具的探討

開發一套遊戲，並不需要特定的工具，只要有創意，記事本也一樣可以設計出以簡易的文字遊戲，但其能表現出來的形式有限。所以在選擇開發工具之前，須對本身產品的走向，要有明確的定位。在 3.3.1 節中，我們會介紹目前開發工具的類型，3.3.2 節，便針對系統需求，明確定義出，在開發過程中我們所使用的工具。

3.3.1 開發工具概觀

目前語言大至上分為三類：

- 第一類—描述性語言，如 Flash、JavaScript、或 VBScript。這一類最大的優點是容易撰寫，但效率慢。

- 第二類—半編譯、半直譯，如 Java 或 VB6。而 Java 最大的好處是，能達到跨平台的特性，但需透過虛擬機器架構在不同的硬體平台上，會影響到效能。

- 第三類—純編譯，如 C/C++或 Delphi。能與硬體直接做溝通，有效發揮效能，但是對硬體有依賴性。

一、Flash

Flash 是 Macromedia 公司所推出的軟體，專門用來設計互動式的多媒體動畫，用向量繪圖和流式播放技術，產生的 swf 檔，所佔空間很少，有助於在網際網路之間流傳。而 Flash 本身有 ActionScript 的語法，可以輕鬆地寫出互動式動畫，再加上創意，就是遊戲(圖 3-1)。

圖 3-1 Flash遊戲Rural Racer[11]

[11] 圖片來源：http://www.miniclip.com/Homepage.htm

二、Java

這是一種由美國昇陽電腦公司所研發出來的語言,可透過JVM(虛擬機器)達到跨平台的特性，而且 Java 具有純物件導向的性質，使得程式維護較為容易。另一方面 Java 主要是經由網際網路發展，所以在安全性上的設計，有多方面的考量，原則上 Java 程式是無法破壞系統。而 Java 在記憶體管理方面，也有強大的能力，可經由 JVM 幫助，可將不需要的記憶體空間自動釋放，即所謂的資源回收處理。

因為 Java 主要有著跨平台的特色，所以能在不同的硬體設備上開發遊戲。現在各大手機公司紛紛推出 Java 遊戲，如圖 3-2 便是由台灣大哥大所提供的 Java 遊戲。

圖 3-2 Java手機遊戲[12]

另一個特色，在個人電腦的環境上，透過網頁開啟 Java 遊戲，不需要另外安裝，就可以盡情玩樂。Java 遊戲也可以做到多人連線，享受多人同樂，如 iGame 遊戲網間，就提供多種遊戲，包括單人或是多人。如圖 3-3，便是他們大老二遊戲的大廳，玩家可以透過此處，跟其他玩家接觸，每一桌湊足四個人就可以進入遊戲如圖 3-4。

[12] 圖片來源：http://www.gamecool.com.tw/softstar/tccjava/index.htm

圖 3-3 遊戲大廳[13]

圖 3-4 遊戲畫面[14]

[13] 圖片來源：http://www.igame.com.tw/

[14] 圖片來源：http://www.igame.com.tw/

三、Visual C++

這是由微軟公司研發出來的開發工具，保有 c 語言精簡的語法及高效能的執行能力，並加入物件導向的設計理念，提供了更有彈性，管理性的設計環境。加上 c/c++早已盛行多年，擁有無數強大的函式庫，如 STL、DirectX、及 OpenGL 提供給程式員使用。而且現在的個人電腦還是以微軟的 Windows 作業系統佔有率較高，因為以上種種原因，各大遊戲公司多半以 VC++開發大型遊戲軟體，VC++更是被評為遊戲開發中最佳的開發工具。而 VC++所開發出來的遊戲也非常多樣化。

3.3.2 系統開發工具

在 2.3.3 節中曾明確定義出以下的系統需求：

- 即時動作 RPG 遊戲
- 即時聲光音效
- 2D 人物/3D 環境場景
- 主從式系統可以多人連線
- 聊天系統
- Windows 作業系統平台

以下就針對這些需求分析，決定在開發的過程中，我們使用到的工具。

所謂即時性(作業)，就是要能立即對使用者所下達的命令做出反應，所以首先要以效能來做優先考量。再來就是遊戲所牽扯到的硬體控制介面，如鍵盤跟滑鼠的輸入。而在影像及音效的輸出方面，對於硬體的介面，更要有完整的支援，因此，在影像方面需要 3D 繪圖 API，目前兩大主流為 OpenGL 及 Direct3D。另一方面，就是要能寫出網路程式，以建立主對從的傳送方式。最後，在聊天系統方面，在於字串輸入的處理上，要能做到中及英文輸入。

一、效能方面

在一開始就已經對程式語言做簡單地介紹，因為不需做到跨平台，就沒必要犧牲效能，所以純編譯式的程式語言(C/C++及 Delphi 等)為最佳選擇。以 C/C++和 Delphi 兩種程式語言來做考量，C/C++與 Delphi 的語言發展至今，在彼此之間效能上的差異，已相差不多了，所以兩者在書架上或是網路上都有不少遊戲程式寫作技術相關的資料。就單以效能，來說這兩套語言都是不錯的選擇。

二、硬體介面控制方面

因為是要在 Windows 作業系統上開發，加上 Windows 本身就提供 Windows API 應用程式介面，以事件觸發、訊息傳送的方式，來傳達週邊設備的狀態，這樣的方式對一般的應用軟體來說是足夠的；但以遊戲來說，其需要做出即時的回應，還稍嫌不足。因為周邊設備所觸發的訊息會進入訊息佇列中等待處理，當系統忙於其他行程的時候，將會造成訊息的延遲。

所幸在這一方面，微軟有發展出 DirectX，其元件包含 Direct Graphics、Direct Audio、DirectPlay、DirectInput、DirectShow、及 DirectSetup 等許多的軟體介面可直接與硬體層溝通，而這些幾乎都具備開發遊戲所需要的元件。且 DirectX 內部結構是由硬體抽象層(HAL)，和硬體模擬層(HEL)組成，這對程式設計師做硬體控制的時候，硬體抽像層會自動針對不同的硬體將功能實做出來。就算硬體有不支援的功能，也可以透過硬體模擬層，模擬出類似的功能。這對於開發遊戲者來說，可以大大減少研發時間，並能更專注於遊戲內容的程式設計。並且，DirectX 所實作出來的效能，更受到大家的肯定。

大部分 Windows 上的遊戲都有使用到 DirectX，所以 DirectX 成為這次遊戲開發的主要工具之一。

三、3D 繪畫技術方面

目前的遊戲種類大部分都標榜著 3D 遊戲的口號進軍市場，以吸引更多玩家的青睞。這似乎說明著 3D 畫面的遊戲，快成為現今遊戲的基本條件。或許有些人不這麼認同，他們認為 2D 畫面一樣可以表現出豐富好玩的遊戲。這一點我們也是不否認，只是觀察現在的市場反應，受歡迎的遊戲當中，多數為 3D 遊戲。這都是因為現在 3D 繪畫技術不斷地推陳出新，表現出令人讚嘆效果所帶來的效益。

目前，兩大主流技術，分為 OpenGL 和 Direct3D。前者為 OpenGL Architecture Review Board (簡稱 ARB)委員會共同定制標準；後者由 Microsoft(原本是 ARB 委員其中一員)跳出來自行開發。兩者實作出來的效果也都有高水準的表現，個別擁有支持者。像是 ID software 生產的知名遊戲 Quake3，就是以 OpenGL 技術所開發出來的；而 Epic MegaGames 出產的 Unreal Tournament 則是以 Direct3D 技術開發，當然也有同時支援 OpenGL 跟 Direct3D 的遊戲。因此，以一個遊戲程式設計師來說，最好是學會這兩種技術。

在抱著想學會這兩種技術的前提之下，首先選擇了 OpenGL 為入門語言，主要的原因是因為 OpenGL 採用 C 語言風格來當介面程式，以呼叫函式的方式就能輕易撰寫。再加上 Direct3D 是構築於 COM(Common Object Model)上的套件，對於還沒接觸過 COM 的人來說，這將是一個學習阻礙。而且 OpenGL 為公開原始碼，這對想了解實做原理的人，有很大的幫助。

最後原因則是，OpenGL 是由 ARB 共同定下的標準，也就是可以跨平台，以後可以在其他的作業系統下開發 3D 遊戲。

四、網路方面

Windows 自己有提供一套網路程式的應用程式介面 winsock，但和 DirectPlay 比較之下，DirectPlay 提供更方便的介面，不用再撰寫較低層 sock 服務的程式，DirectPlay 已經寫到內部，所以可以更專注於撰寫伺服端與用戶端之間遊戲資料傳送的設計。

五、總結

根據上述一至四項的分析，可以整理出這次的開發工具，是以 OpenGL 為 3D 繪圖介面。鍵盤、滑鼠、音效、和網路則分別使用 DirectX 的套件 DirectInput、DirectAudio 和 DirectPlay。其中，OpenGL 與

DirectX 的相關技術介紹，我們整理節錄於附錄 C。

在開發語言的選擇上，基於 OpenGL 和 DirectX 都是以 C/C++語言為介面程式，而且 C/C++語言撰寫上，自由度大，效能高，又為我們擅常的語言，所以就以 C/C++為開發語言，另外，我們是以 Visual C++.Net 作為軟體開發平台。

最後，由於系統是以視窗的方式呈現，因此，我們必須對 Windows 視窗程式設計有些許的了解，在下一小節，我們便會說明一些 Windows 視窗程式設計的基本概念。

3.3.3 Windows視窗程式[15]

要在Windows上開發視窗程式,必須要先對Windows程式的事件驅動特性，包括訊息的產生、獲得、分派、判斷、及處理等，有一定的了解，才能對於整個視窗程式的架構有明確的認知。

視窗程式設計跟DOS模式下的程式設計，有幾點不一樣的地方：

- 視窗程式的進入點為WinMain，一般C程式的進入點為Main。

- 視窗程式是架構在事件驅動的方式之下，等到有事件發生時，系統會將事件放入所謂的訊息佇列，並通知它去處理。而DOS模式下則是靠中斷系統，提供一些中斷服務。

- 視窗程式，就如其名，是以視窗來表現出程式行為，所以必須跟系統呼叫一個視窗資源，經由 WndProc()(視窗函式)，提供給系統callback，DOS下本身就提供一個顯示平台，不需要再另外提供，所以程式行為上比較直觀，所寫的程式行為都會一一呈現，不會有系統內部的行為。

以下便針對Windows視窗程式，做簡單的介紹：

一、以訊息為基礎，以事件驅動之。

Windows程式的進行依靠外部發生的事件來驅動。換句話說，程式持續等待(利用一個while迴路)，等待任何可能的輸入，然後做判斷，再做適當的處理。上述的「輸入」是由作業系統捕捉到之後，以訊息形式(一種資料結構)進入程式之中。

作業系統如何捕捉週邊設備(如鍵盤和滑鼠)所發生的事件呢？

[15] 本節內容整理自侯俊傑之書”深入淺出MFC程式設計”p7-8及p25-26

是藉由 USER 模組掌管各個週邊的驅動程式，它們各有偵測迴路。如果把應用程式獲得的各種「輸入」分類，可以分為由硬體裝置所產生的訊息(如滑鼠移動或鍵盤被按下)，放在系統佇列(system queue)中。以及由 Windows 系統或其它 Windows 程式傳送過來的訊息，放在程式佇列(application queue)中。

若以應用程式的眼光來看，訊息就是訊息，來自哪裡或放在哪裡其實並沒有太大區別，反正程式呼叫 GetMessage API 就取得一個訊息，程式的生命靠它來推動。所有的 GUI 系統，包括 UNIX 的 X Windows 以及 OS/2 的 Presentation Manager，就像這樣，是以訊息為基礎的事件驅動系統。每一個 Windows 程式都應該有一個迴路如下：

```
MSG msg;
While(GetMessage(&msg,NULL,NULL,NULL)){
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}//以上出現的都是 Windows API 函式
```

訊息，也就是上面出現的 MSG 結構，其實是 Windows 內定的一種資料格式：

```
/*Queued message structure*/
typedef struct tagMSG
{
    HWND        hwnd;
    UINT        message <- -WM_xxxx
    WPARAM wParam; (例如 WM_MOUSEMOVE，WM_SIZE，
                    WM_LBUTTONDOWN…)
    LPARAM      lParam;
    DWORD       time;
    POINT       pt;
```

}MSG;

接受並處理訊息的主角就是視窗，每一個視窗都應該有一個函式負責處理訊息，程式員必須負責設計這個所謂的「視窗函式」(window procedure，或稱為 window function)。如果視窗獲得一個訊息，這個視窗函式必須判斷訊息的類別，決定處理的方式。

以上就是 Windows 程式設計最重要的觀念。至於視窗的產生與顯示、十分簡單，有專門的 API 函式負責。稍後我們就會看到 Windows 程式如何把這個訊息的取得、分派、處理動作表現出來。而從圖 3-5 可得知 Win32 應用程式的本體與作業系統之間的關係。

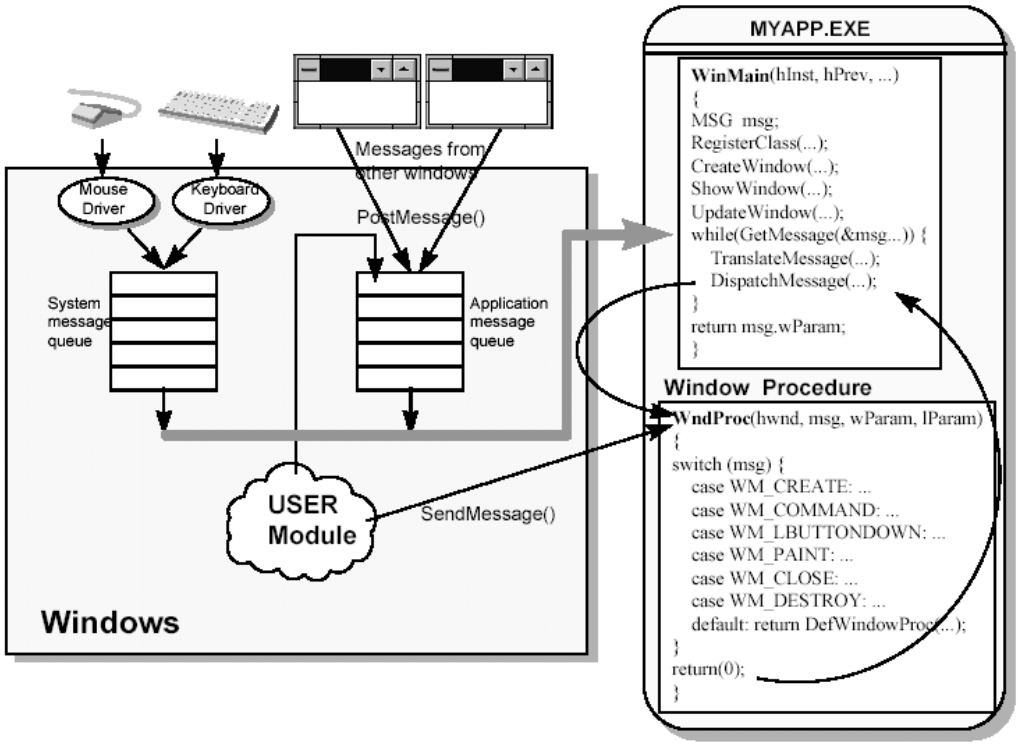

圖 3-5 Windows 程式的本體與作業系統之間的關係

二、Windows 程式的生與死

對 Windows 訊息種類以及發生時機的透徹了解，正是程式設計的關鍵，現在，我以視窗的誕生和死亡來說明訊息的發生與傳遞，以及

應用程式的興起與結束。如圖 3-6、圖 3-7 所示。

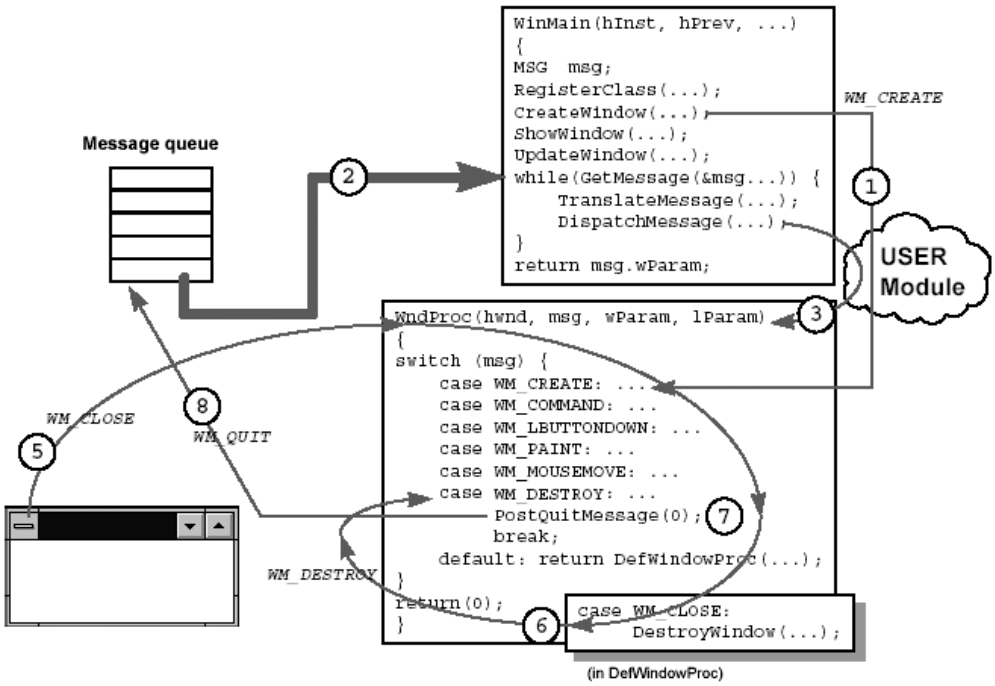

圖 3-6 視窗的生命週期（細詳說明請看圖 3-7）

1. 程式初始化過程中呼叫 *CreateWindow*，爲程式建立了一個視窗，做爲程式的螢幕舞台。*CreateWindow* 產生視窗之後會送出 *WM_CREATE* 直接給視窗函式，後者於是可以在此時機做些初始化動作（例如配置記憶體、開檔、讀初始資料...）。
2. 程式活著的過程中，不斷以 *GetMessage* 從訊息貯列中抓取訊息。如果這個訊息是 *WM_QUIT*，*GetMessage* 會傳回 0 而結束 *while* 迴路，進而結束整個程式。
3. *DispatchMessage* 透過 Windows USER 模組的協助與監督，把訊息分派至視窗函式。訊息將在該處被判別並處理。
4. 程式不斷進行 **2.** 和 **3.** 的動作。
5. 當使用者按下系統功能表中的 Close 命令項，系統送出 *WM_CLOSE*。通常程式的視窗函式不攔截此訊息，於是 *DefWindowProc* 處理它。
6. *DefWindowProc* 收到 *WM_CLOSE* 後，呼叫 *DestroyWindow* 把視窗清除。*DestroyWindow* 本身又會送出 *WM_DESTROY*。
7. 程式對 *WM_DESTROY* 的標準反應是呼叫 *PostQuitMessage*。
8. *PostQuitMessage* 沒什麼其他動作，就只送出 *WM_QUIT* 訊息，準備讓訊息迴路中的 *GetMessage* 取得，如步驟 **2**，結束訊息迴路。

圖 3-7 視窗的生命週期（細詳說明請看圖 3-6）

3.4 軟硬體支援

3.4.1 軟硬體需求

本系統的操作環境要求如表 3-1 所示。

表 3-1 軟硬體需求對照表

執行環境	基本配備	建議配備
作業系統	Windows2000/XP(繁體中文版)	
CPU	P III 500	P III 733
記憶體容量	128MB	
硬碟空間	50MB	
顯示卡	支援 Direct X8.0 以上、OpenGL 之 16MB 顯示卡	32MB 以上
音效卡	支援 Direct Sound 的音效卡	
操作設備	滑鼠(必備)、鍵盤	

3.4.2 測試所需硬體

在開發的過程，我們需要建立一個網路連線平台，來實際測試本系統階段性任務的完成與否。且本系統的目的之一是以提供一個平台，讓多人能藉由網路同時上線並進行角色扮演之遊戲為主，故分散式處理的結果必須正確且有效。藉由下圖的架構，我們得以測試連線處理的結果是否(如預期)正確。

圖 3-8 左端之伺服器(Server)，是整個 3D 遊戲最核心的元件。透過網路，它會依照設定檔來建構出遊戲世界中所有的物件跟設定值，再來就等待用戶端(Client)連線進來。當用戶端的玩家連線登入成功之後，伺服將會依照帳號來傳送玩家的角色資料，用戶端接收後再依照資料建構出玩家所在的遊戲世界，包括附近的其他玩家角色、怪物、NPC(電腦控制角色)等都會一併傳送給用戶。之後用戶端便一直等待直到玩家有新的動作出現，再依玩家所下的訊息處理完後，便將訊息傳給伺服端處理，而處理過後的結果再回傳給玩家(用戶端)，完成彼此之間的互動。同樣的，這些互動會一一傳送給其他玩家知道，以建構出一個虛擬的世界。

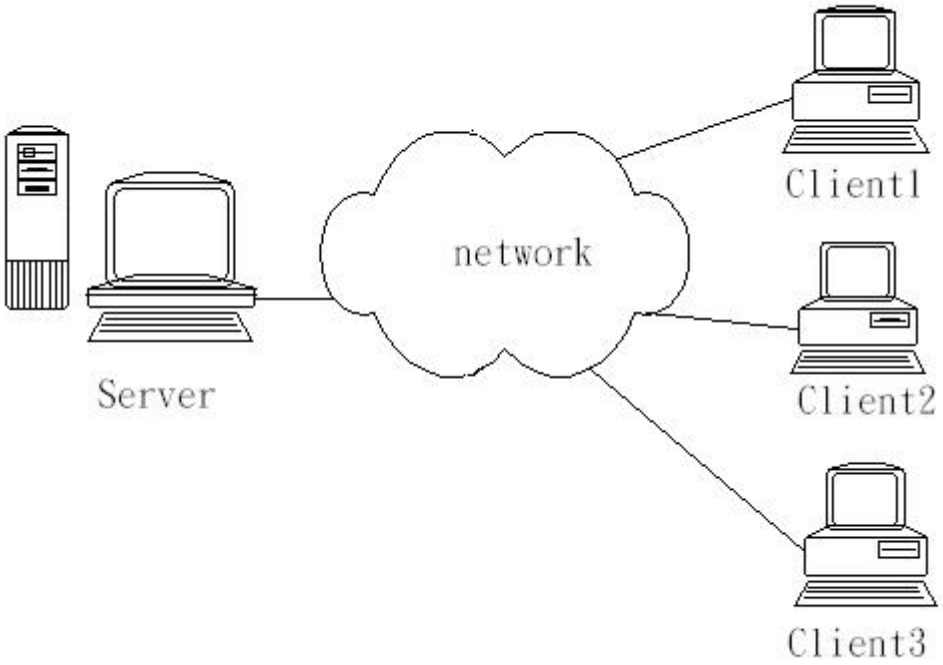

圖 3-8 主從式系統硬體架構

第四章 系 統 設 計 與 規 劃

在系統開發的過程，系統設計與規劃的好壞，會影響到往後系統整體效能的表現。本章利用 DFD 圖的特性，說明一個系統內部在處理工作時，是如何將輸入的資料轉換成有用的資訊，且在這設計過程中明白地指出系統做了那些事，而不去考慮如何做的邏輯問題，這樣較能尋找出系統的最佳化。

4.1 系統架構

我們的遊戲系統基本上分為兩大子系統，一為用戶端系統，另一個為伺服端系統，這兩大子系統以主從式的網路系統為基礎互相溝通。伺服端集中管理所有用戶端的行為，每個用戶端間，透過網路，必須經由伺服端系統的同意才能進行互相溝通。

這種方式可以確保資料的安全性以及遊戲的公平性，使得玩家不能透過修改器改變本機上的資料數值，即便改了，伺服端在下次更新資料的時後，還是以其原本的資料做為更新。但現在越來越多的電腦駭客，以各種方式進行修改，如以解讀封包格式，再自己撰改封包內容送到伺服端，達到做弊的目的，而封包的加密及解密又是一門堅深的學問，此系統中暫時不對封包加密及解密有進一步的設計。如圖 4-1 所示為此系統的網路架構圖。

本遊戲系統中的用戶端系統偏重於如何將伺服端所傳送的資訊，以圖形化的方式清楚地表達出來，並提供方便的介面給使用者傳達命令給伺服端，而伺服器則偏重於處理用戶端所傳過來的指令運算，如圖 4-2 將說明更詳細的系統架構圖。如以資料流程圖(DFD，Data Flow Diagram) 來表明這個系統的資料流向，首先第一步是畫出這個系統的全景圖，來表示系統的界限，如圖 4-3 所示，這個系統的外部實體有四個：玩家、管理者、顯示器、和喇叭。玩家輸入指令，而管理者則將設定給遊戲系統，遊戲系統將會把結果呈現到顯示器和喇叭

上。除此之外，圖 4-4 將全景圖中圖 0 展開為更多的細節，說明遊戲系統內是由兩大系統，用戶端和伺服端組成，彼此互相傳送訊息。

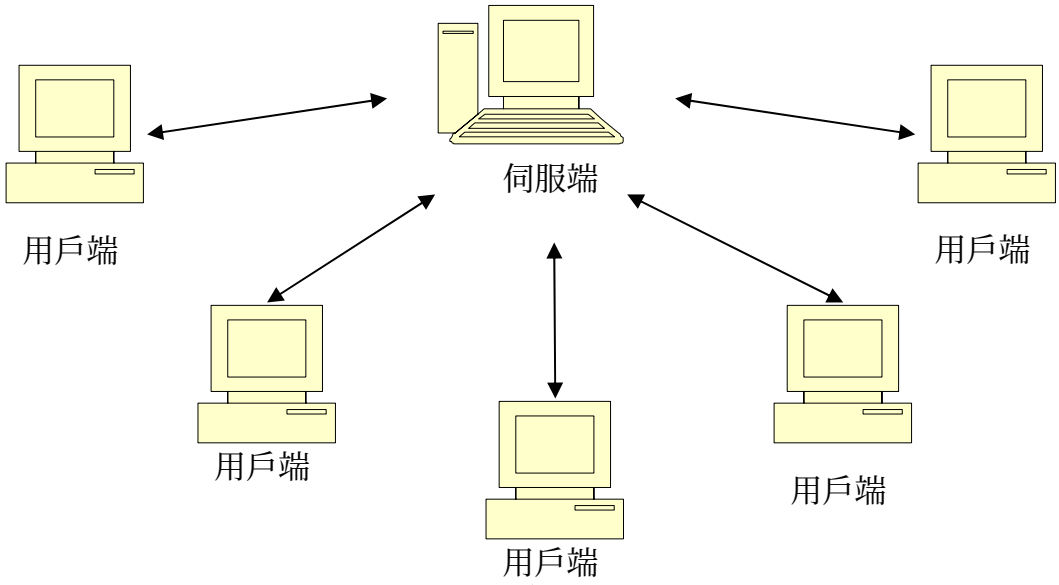

圖 4-1 主從式系統網路架構圖

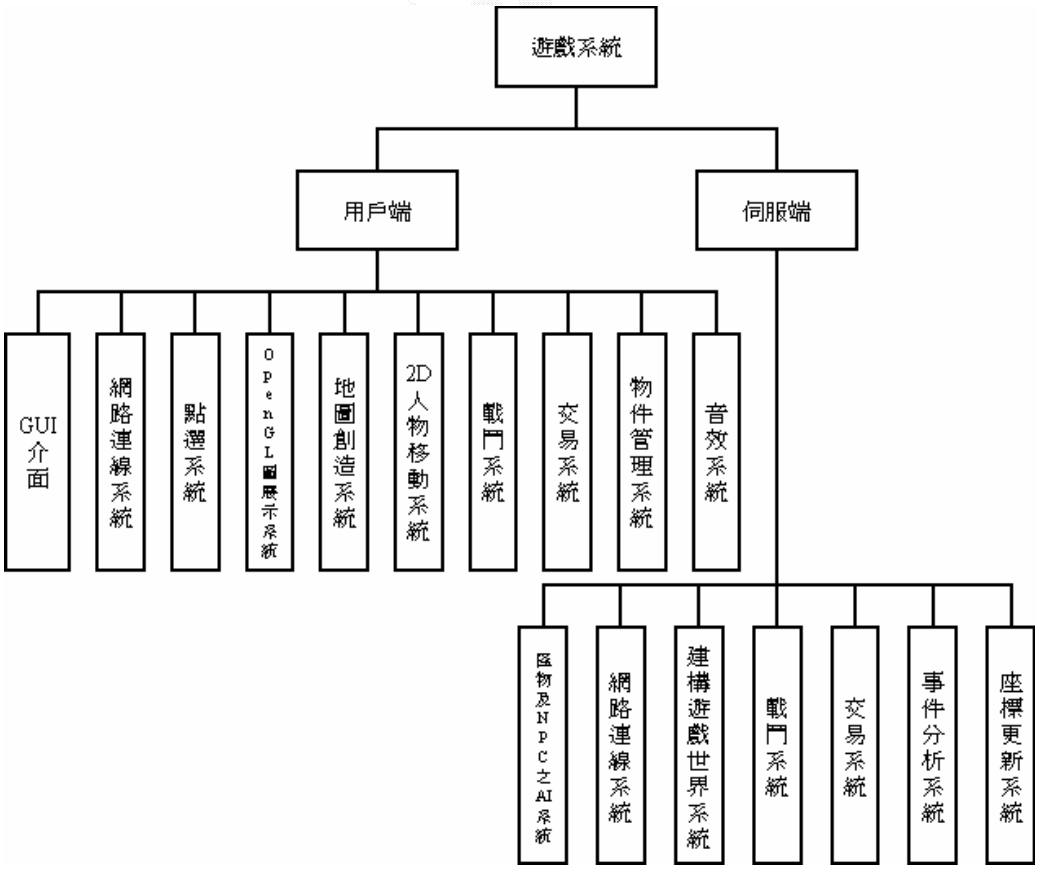

圖 4-2 系統架構圖

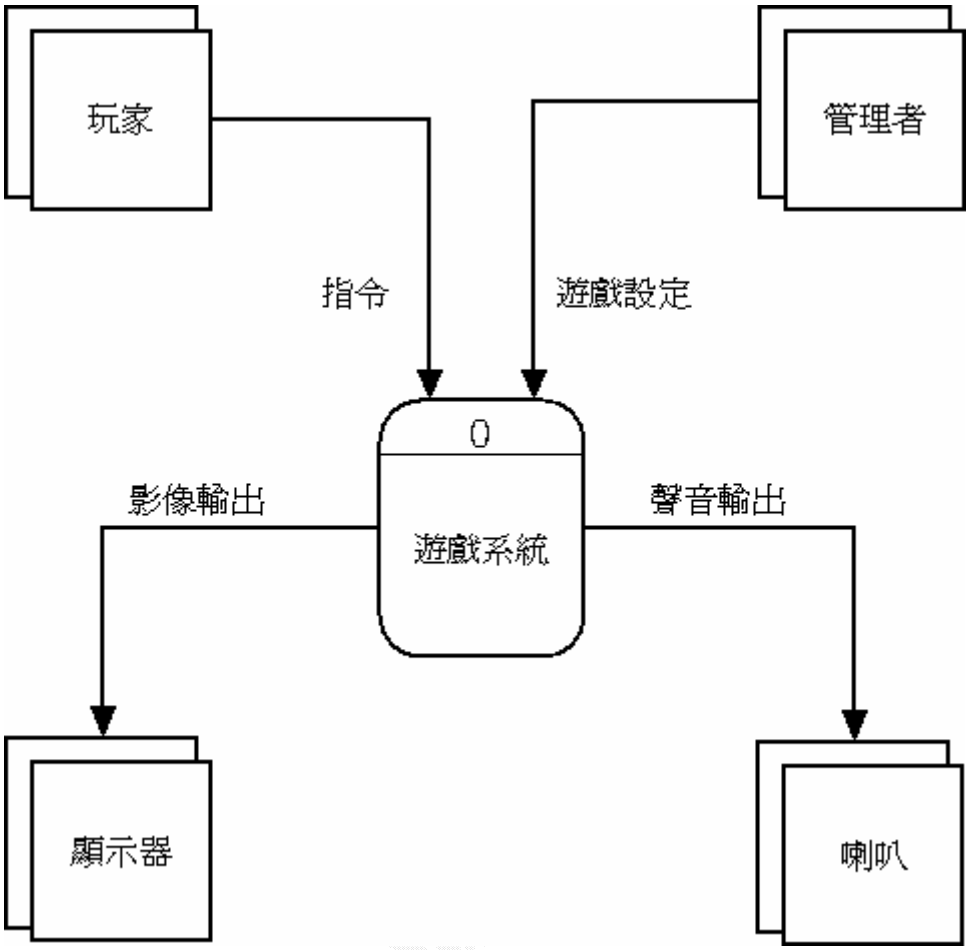

圖 4-3 遊戲系統的 DFD 全景圖

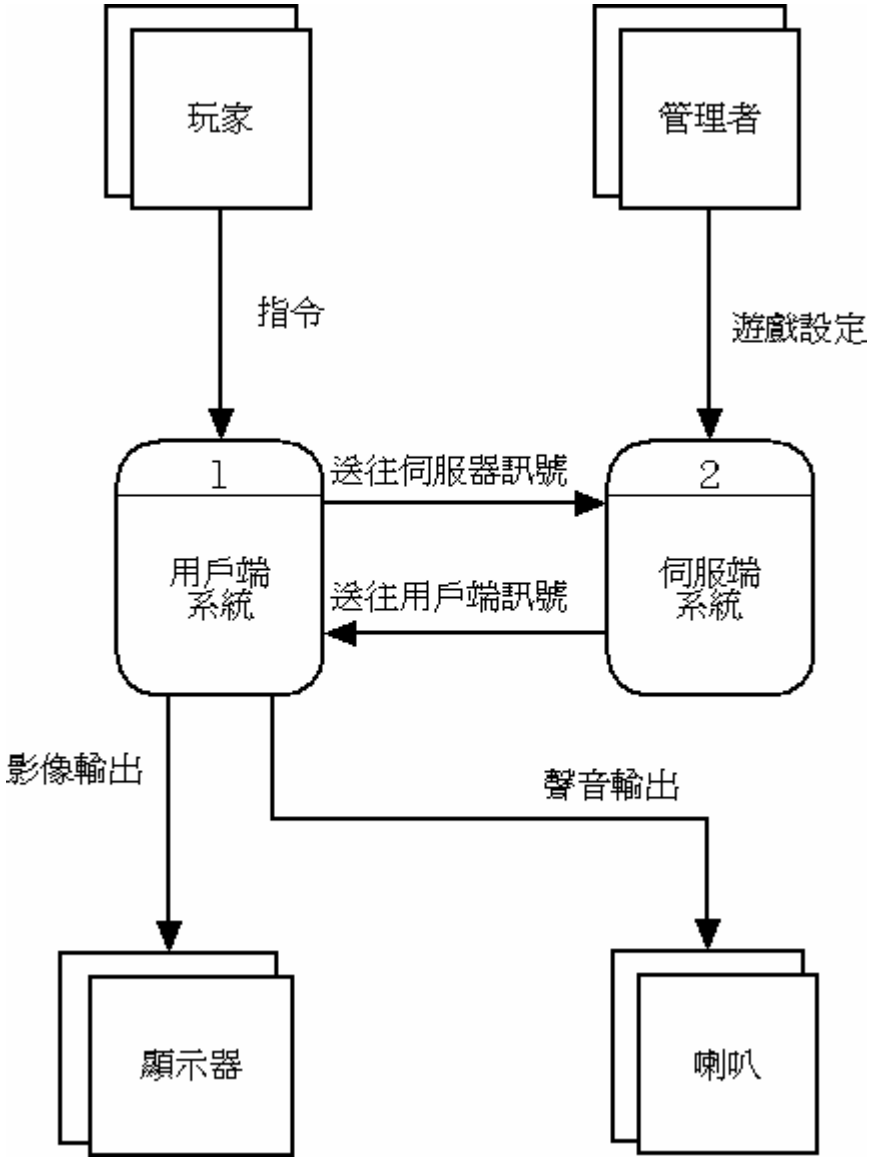

圖 4-4 遊戲系統的 DFD 圖 0

4.2 子系統架構—用戶端

用戶端系統，主要是一個介面程式，把伺服端傳回來的資料以3D 繪圖方式呈現出來，使用者再經由視窗畫面下達命令給伺服端。所以用戶端系統主要的系統技術在於將資料圖形化和配上音效，如伺服端傳回戰鬥的訊息回來，用戶端必須依照動作編號將連序戰鬥動作的圖片逐一顯示出來，和發出戰鬥的聲音，讓玩家了解到已經發動攻擊，再加以做出反應傳給伺服端。所以用戶端的聲光畫面的表現會直接影響到玩家對這個遊戲的評價。

圖 4-5 為用戶端的細部架構圖，在這個系統中最重要的就是OpenGL 技術的部分，當用戶端接收到封包的時後，透過網路連線系統會把每一個封包內容解讀成不同的命令指示，之後再傳達到對應的系統中，如戰鬥系統、物件管理系統、交易系統、和 2D 人物移動系統中。而在這各個系統中會把封包內所傳達的資訊，更新到物件資料上。而這些有關物件資料的數值，會傳給 GUI 介面系統，其主要是將更新後的數值能表現給玩家知道最近的情形，如角色的血量是否達個下限，是，就該做出逃跑或補血等應變行為；在另一個地圖創造系統中，它會從物件資料中取得玩家角色的位置，載入那個位置的周遭環境的資料。之後 OpenGL 圖展示系統就會從物件資料及 GUI 介面內的視窗資料和地圖資料取得畫面上各個部分的資料數值，處理之後轉換成影像交給螢幕顯示出。然而玩家就能從螢幕上了解到目前的狀況，透過滑鼠和鍵盤輸入指令，經由 GUI 介面系統，解析之後，判斷出各個指令的意義，將其包裝成封包傳送到伺服端。如圖 4-6 為此系統內部工作系統詳細的資料流動情形。以下各小節將會再針點各個工作系統做介紹。

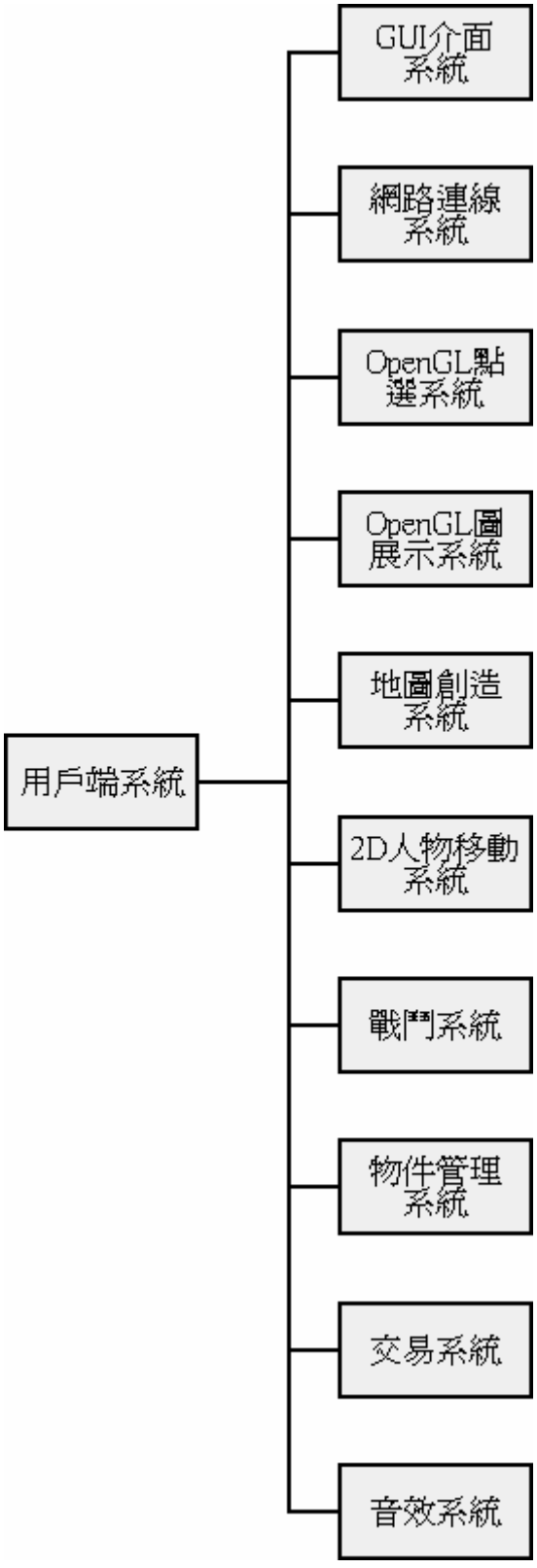

圖 4-5 用戶端系統架構圖

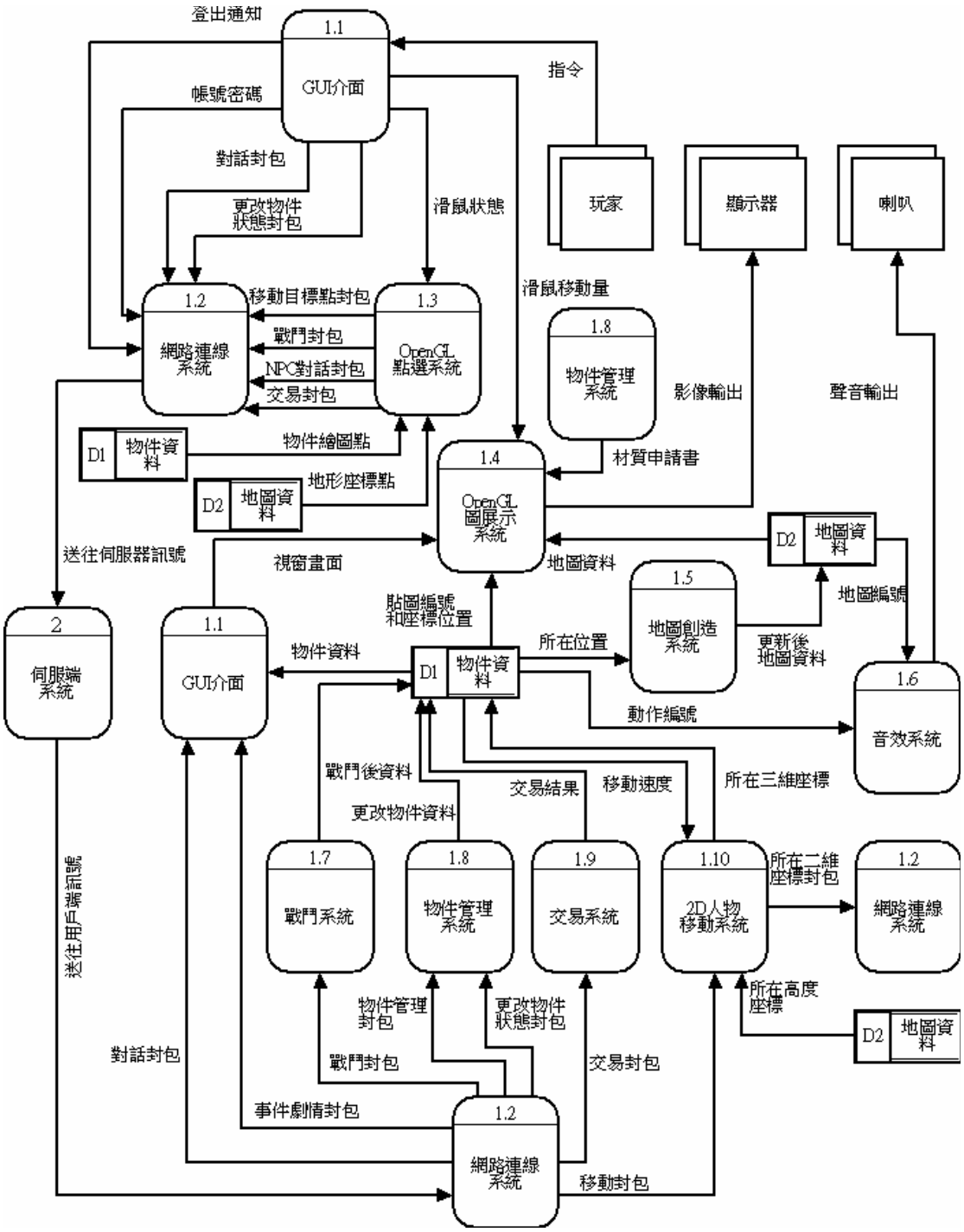

圖 4-6 遊戲系統的 DFD 圖 1-用戶端系統

4.2.1 GUI 介面系統

使用者圖形介面系統，是使用者第一個接觸到的系統。其架構圖如圖 4-7 所示，此系統提供一個方便操作的圖形視窗介面，藉由此介面，可幫助使用者閱讀，玩家能以滑鼠鍵盤輸入指令進行點選及拖曳等動作，來操作及控制遊戲系統，並做出資料輸入(如，登入資料或對話內容)，視窗系統就會將這些動作，加入封包中交給網路連線系統傳送，詳細情形如圖 4-8 所示。做得好的話，能讓第一次使用的使用者，在圖形的表達下，直觀的了解此圖形所代表的使用方法跟意義，當然，這也是我們最大的目標。

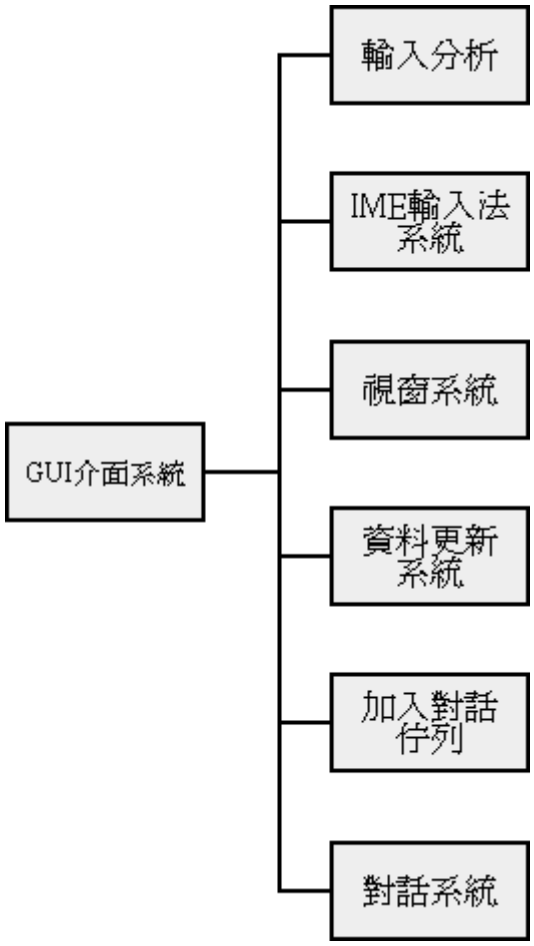

圖 4-7 GUI 介面系統架構圖

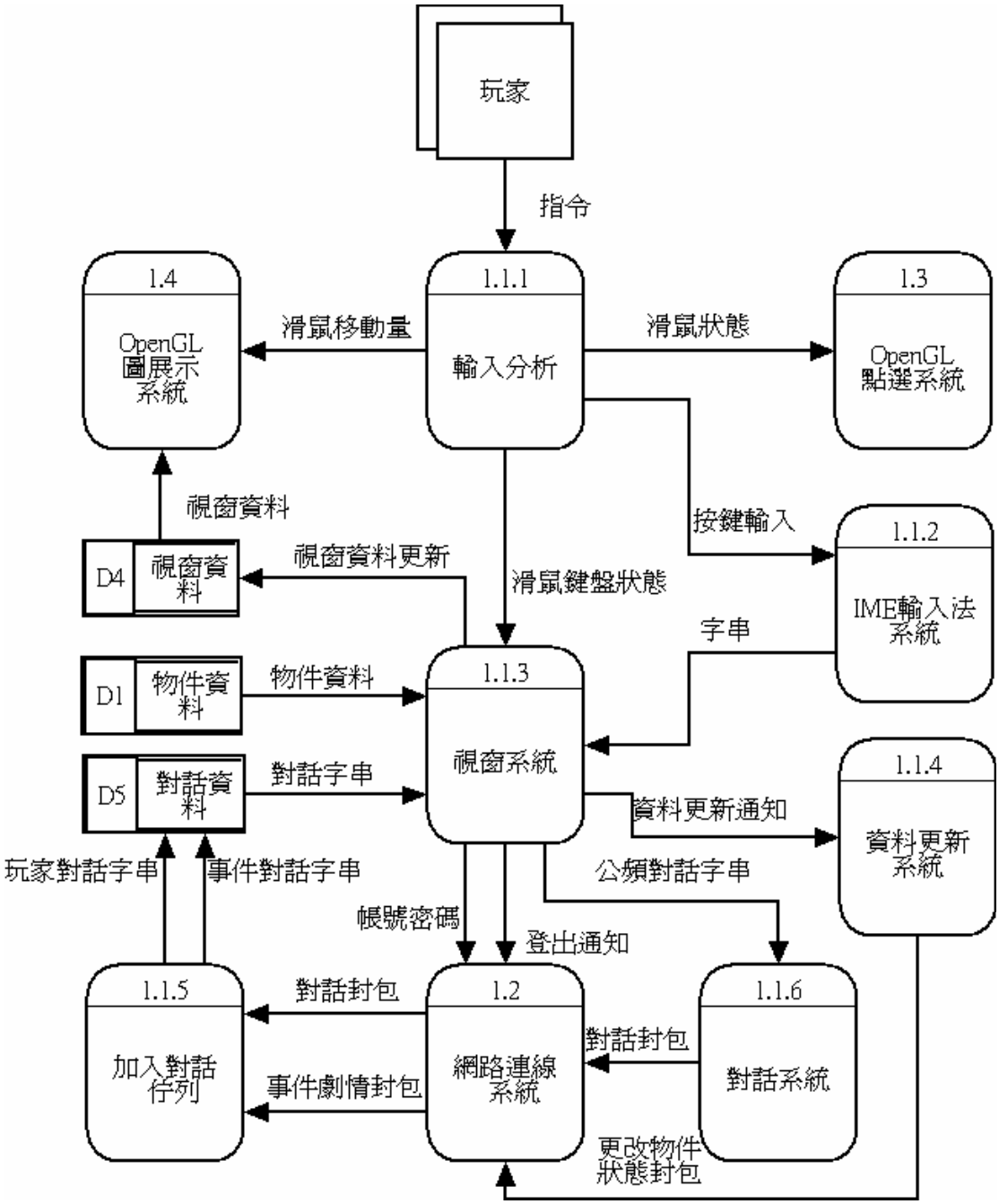

圖 4-8 用戶端系統中的”GUI 介面”的細節

4.2.2 網路連線系統

網路連線系統是搭起用戶端跟伺服端之間的橋樑，其架構圖如上圖 4-9 所示。傳送端將指令訊息封裝成封包資料形態，透過網路線傳送到接收端；接收端則將收到的封包進行分析，並把封包上各式各樣的資訊分配給各別對應的系統處理。圖 4-10 為本系統內部更詳細的資料流情形。

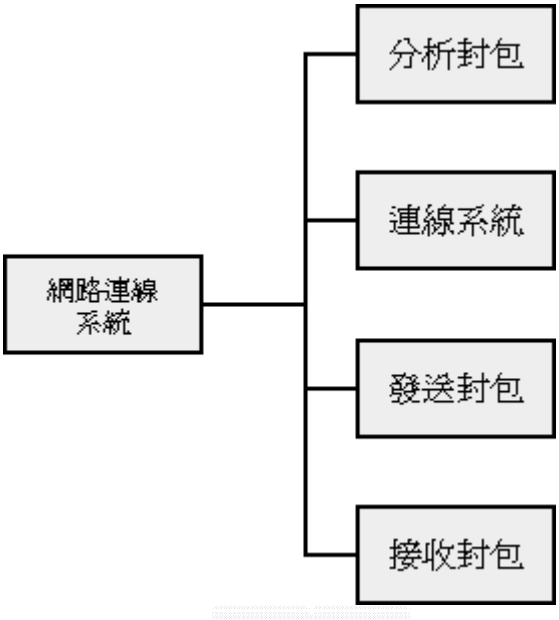

圖 4-9 網路連線系統架構圖

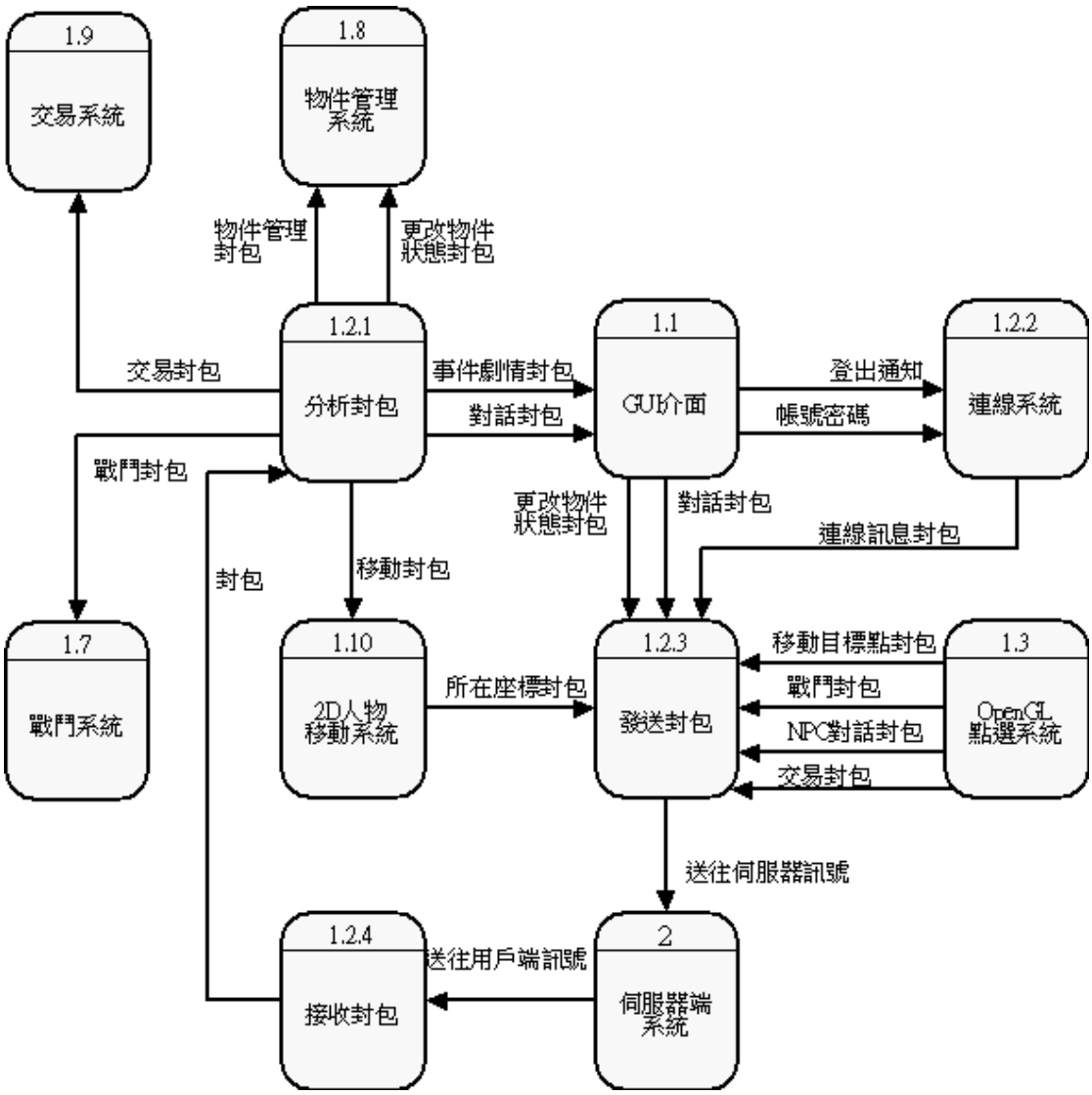

圖 4-10 用戶端系統中的”網路連線系統”的細節

4.2.3 OpenGL 點選系統

圖 4-11 為 OpenGL 點選系統的架構圖，點選系統必須對玩家點選之對象，包括地表座標，作出判斷。點選時會依照點選的對象不同而做出不同的處理動作，如：點選怪物由 2D 物件點選模式來判斷出，那一隻怪物被點選，並對它進行攻擊、點選 NPC 則同樣透過 2D 物件點選模式，來進行對話或進行買賣；點選地表則是透過另一個 3D 地圖點選模式，判斷出所點選的座標，進行移動的動作。而這其中的差異必須由動作分析來解讀後並發出動作訊息。圖 4-12 為本系統內部更詳細的資料流情形。

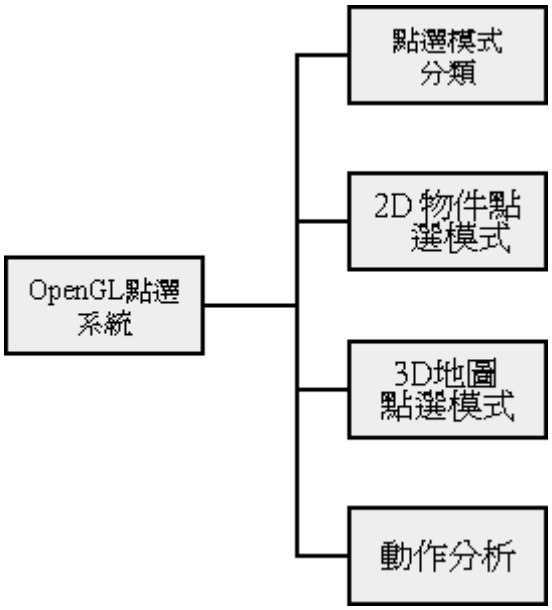

圖 4-11 OpenGL 點選系統架構圖

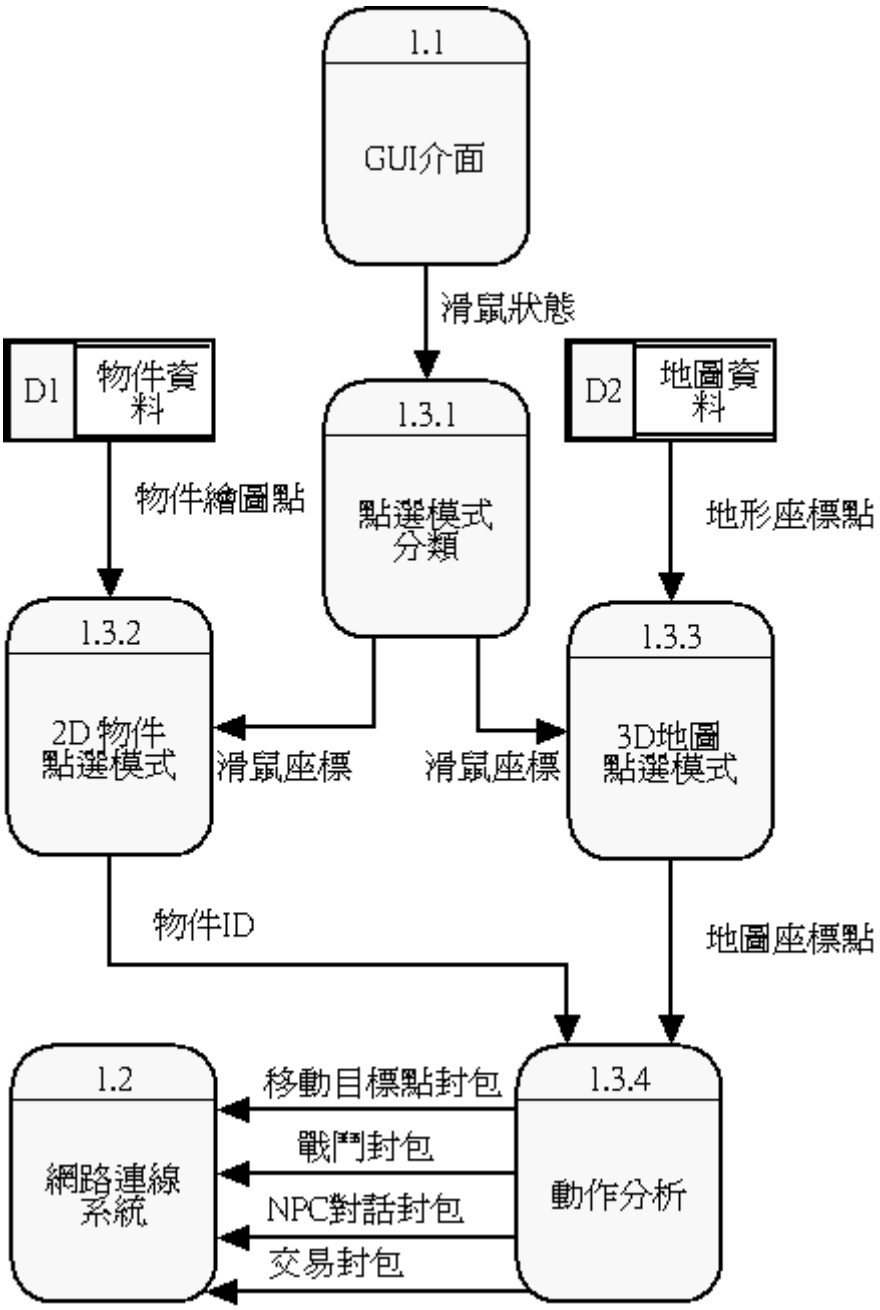

圖 4-12 用戶端系統中的”點選系統”的細節

4.2.4 OpenGL 圖展示系統

簡單來說，此系統會依照我們所設定的顯示圖資料，透過 OpenGL 介面函式，把畫面顯示到螢幕上。整個系統，包括 2D 人物貼圖動畫、3D 地形建模和視窗圖形介面繪製。圖 4-13 為此系統的架構圖。

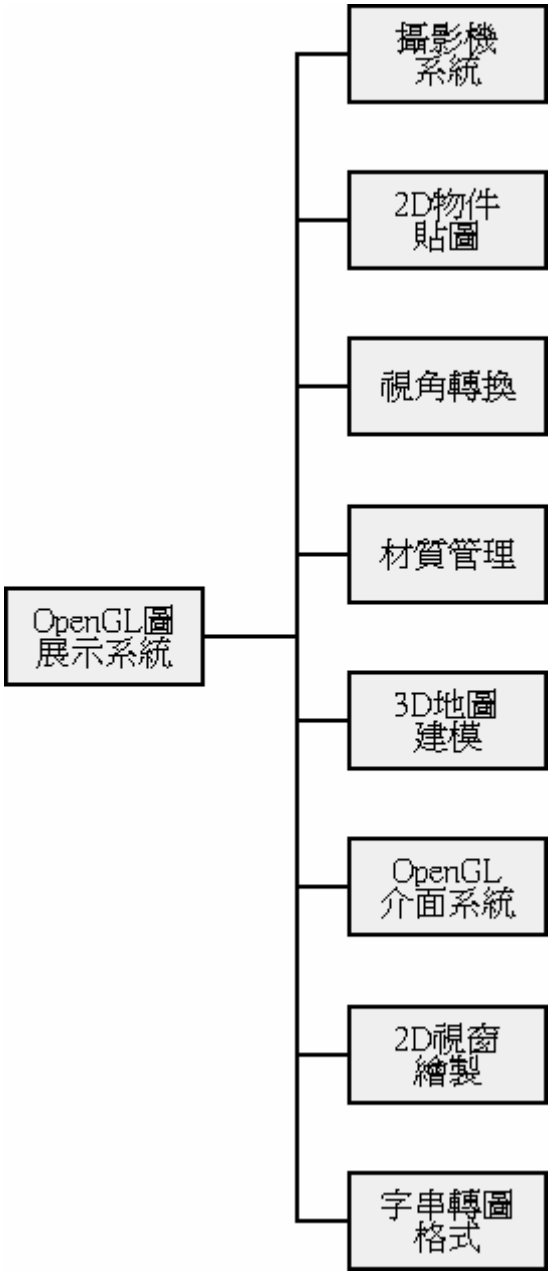

圖 4-13 OpenGL 圖展示系統架構圖

圖 4-14 為 OpenGL 圖展示系統的細節，主要有三大子系統 2D 物件貼圖、2D 視窗繪製、和 3D 地圖創造建模。2D 貼圖如何產生動畫？先將連續動作的圖一張一張繪製好，一起放進記憶體，再依照預先定義好的順序顯示出，利用眼睛的視覺暫留，讓觀看者有物體會動的錯覺。上述提到記憶體空間，有可能是顯示卡記憶體或系統記憶體，OpenGL 會偵測顯示卡記憶體大小後，再放進適當的圖量，不夠的先用系統記憶體暫存。因此，顯示卡記憶體大小會明顯的影響遊戲畫面的流暢。

視窗圖形繪製，這是屬於另一個 2D 繪畫模式，把視窗圖形畫在最上層。統稱遊戲視窗系統，有別於 Windows 提供的視窗介面，這是由程式設計者自行設計，可依照需求擴充設計，本系統提供的視窗介面有對話視窗、玩家狀態視窗、登入視窗、裝備視窗、和交易視窗等。

3D 地形建模會依我們指定的座標填入一連串三角形（OpenGL 的 3D 元件以三角形組成），連續的三角形蜿蜒高低表現出地形變化，而地表的處理會讓影像看起來更真實；地表通常要經過 2 層處理：第一層先將地圖基本顏色攤開貼上，第二層將紋路放上。

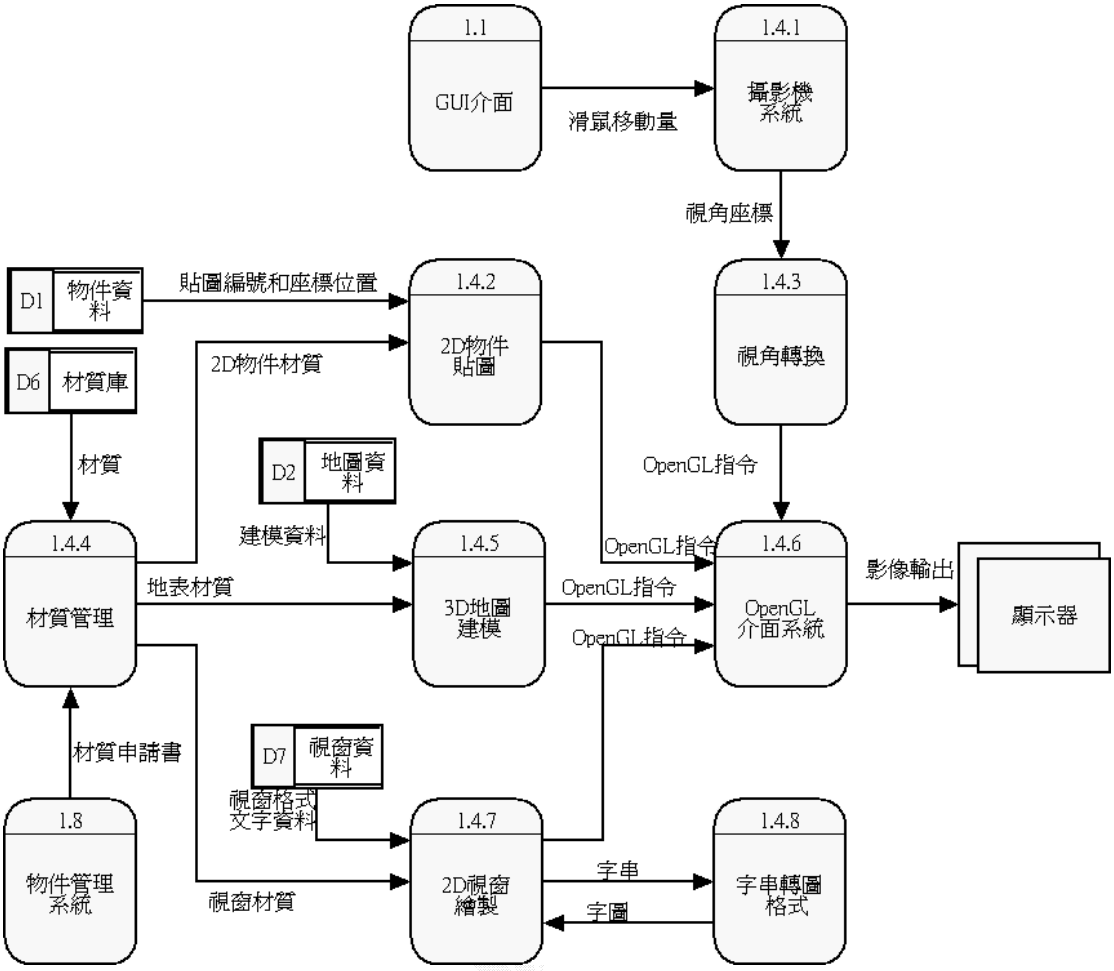

圖 4-14 用戶端系統中的”OpenGL 圖展示系統”的細節

4.2.5 地圖創造系統

地圖創造系統中只有三個子系統，如圖 4-15 所示，依照地圖設計者所製作的設定檔，本系統會載入所需的 3D 物件以及地形高低座標值。系統所載入的 3D 物件檔案格式是『.3DS』，市面上，有很多套的 3D 繪圖軟體都能支援這樣的模型格式，如 3ds Max、Maya、和 SketchUp 等。所以在設計 3D 物件上可以使用現成的軟體。

而在地形產生器方面，地形的產生是依照 raw 圖檔所設定的地形產生。raw 圖檔的特性就是，圖檔沒有外加圖檔格式的檔頭，以數值由小到大表示黑到白，地形的高低就能依顏色的深淺變化設定，而在這方面的細節將在第五章介紹。當地形跟 3D 物件的資料都讀入之後，就會依照玩家的座標位置，畫出可視範圍內的地形和地表上的 3D 物件，此系統的資料流情形如圖 4-16 所示。這樣可提高繪圖效能，而且可創造出較大的地圖場景，給玩家有更寬廣的感覺。

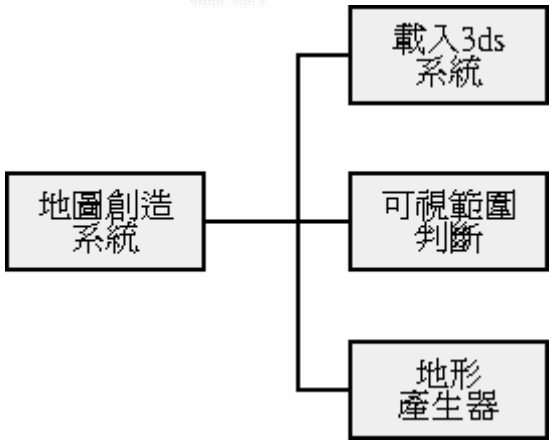

圖 4-15 地圖創造系統架構圖

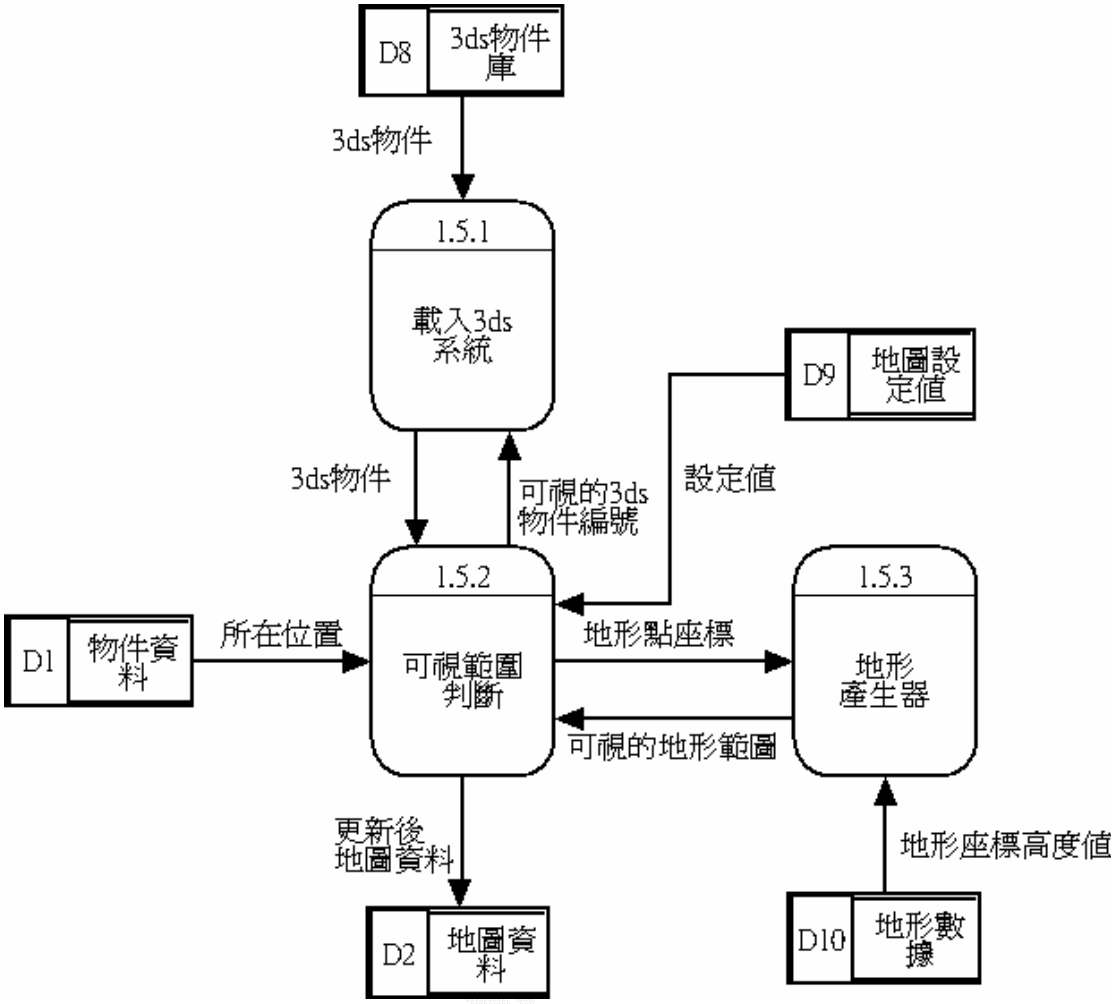

圖 4-16 用戶端系統中的”地圖創造系統”的細節

4.2.6 2D 人物移動系統

2D 人物移動系統其架構圖如圖 4-17，這是對包括所有玩家、怪物、和 NPC 的移動，作處理。人物的移動會依照事先設定的移動速度，再乘上時間間距，就是移動的距離。以這種方式運作，有一個優點，就是當程式並不是以等速的方式進行時，難免會吃資源，而造成每秒的繪圖張數跳動很大，可能從一秒六七十張跳到三四十張，將近是減少一半的張數，這時，也不會影響到移動的速度。

以計算最短路徑來說，一方面，可以方便玩家控制角色移動到指定位置而不需要一步一步移動；而另一方面，伺服端傳送移動訊息給用戶端時，可以只傳送目地作標點，然後用戶端再依照相同的最短路徑公式去算出移動過程的座標點，這樣用戶端所看到的移動的軌跡是一樣的，這樣可大幅度減少傳送封包的數量，只要傳目地點的座標封包就好，不用把每一個經過的點都一併傳送，而且也會因為行徑的過程是由用戶端的最短路徑公式算出來的，反而移動起來會比每一步都要由伺服端來傳送的順暢，其細節如圖 4-18 所示。

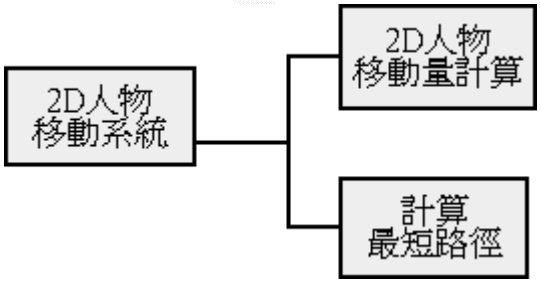

圖 4-17 2D 人物移動系統架構圖

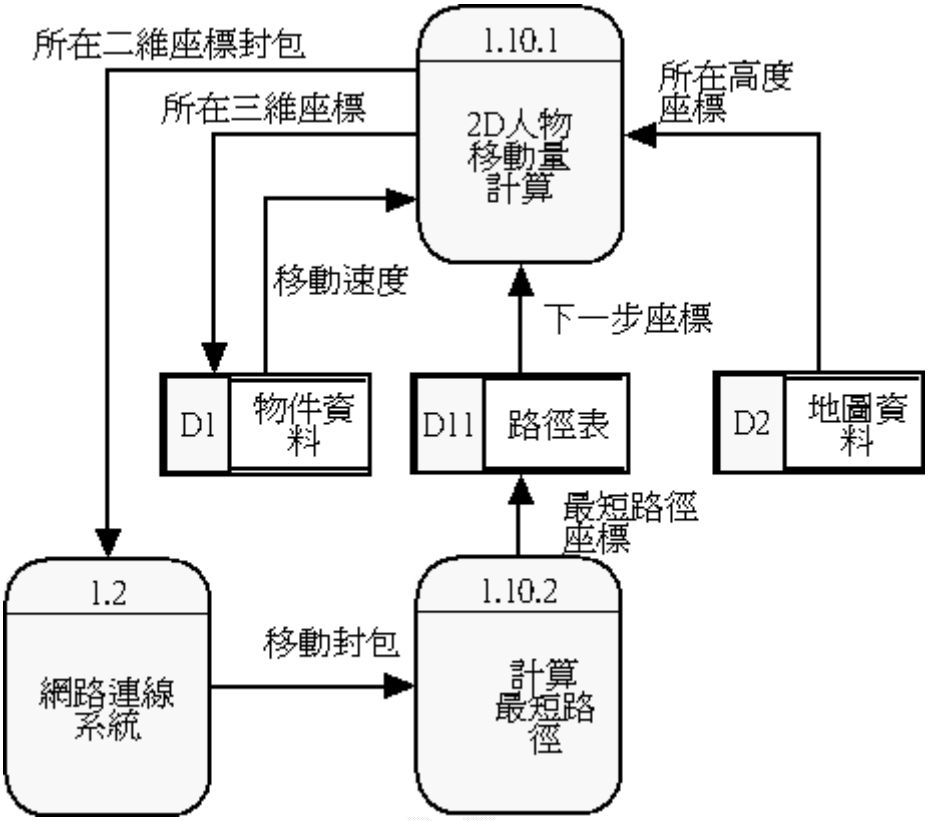

圖 4-18 用戶端系統中的”2D 人物移動系統”的細節

4.2.7 戰鬥系統

在用戶端這邊的戰鬥系統，比伺服端的戰鬥系統來的簡單一點，因為複雜的攻擊計算方面是在伺服端處理計算，用戶端這邊負責的是將計算後的結果，更新到物件資料上，並依資料做出不同的反應。如玩家的血量已經到達 0 時，玩家的狀態會變成死亡，此時，貼圖就會貼出玩家倒下的動作。

4.2.8 物件管理系統

物件管理系統所管理的對象，包括玩家、怪物、和 NPC 等物件。其功能有：新增、刪除、更新物件資料、和快速搜尋。其中，物件管理系統的搜尋速度將會影響到程式的效能。在整個遊戲進行的過程中，搜尋的動作是最頻繁的，如每次伺服端所傳回的玩家移動位置，都必須正確尋找出那一位玩家的資料位置，對它下達移動的指令，所以有指定要對誰做出怎樣動作的時後，就必須搜尋一次，為了能快速搜尋，資料的儲存方式是以二元樹的資料結構型態，二分法來搜尋。

4.2.9 音效系統

音效系統掌管聲音輸出，遊戲中的背景音樂或是各類事件的音效都由它統一管理。背景音樂主要是讀取 MIDI 檔格式的音樂，MIDI 檔的最大優點就是所佔的記憶體容量小，一首兩三分鐘的歌通常只要十幾 K 的大小，且其音源的聲音由音效卡模擬出來，很適合拿來做遊戲音樂。

而音效方面是播放 WAV 檔，WAV 檔可以保留最完整的聲音，可說是原音重現，像是遊戲中揮刀的聲音、爆炸的聲音等。這是 MIDI 檔所無法表現出來的聲音效果，但 WAV 很佔空間，所以都是一小段的聲音效果。

4.3 子系統架構—伺服端

伺服端系統其架構圖如圖 4-19 所示，是負責建構出整個遊戲世界，並且賦予怪物及 NPC 有自己的行為能力之系統。透過此系統可產生和玩家互動之效果，如怪物與玩家之間的戰鬥，NPC 與玩家之間的對話，以及玩家跟玩家之間地溝通，都將透過伺服端各類子系統之運算協調。

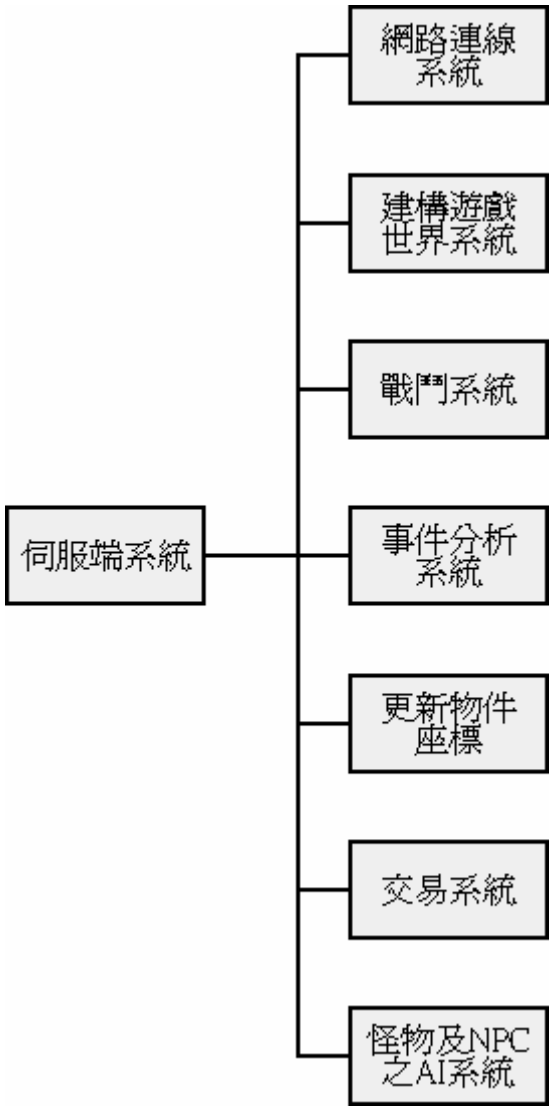

圖 4-19 伺服端系統架構圖

圖 4-20 為伺服端的資料流情形，當伺服端接收到各個用戶端的指令封包後，將立即計算並更新物件狀態，隨即把結果傳回給用戶端，所以伺服端可以說是最重要的中樞系統。一旦伺服端發生問題，一切都將無法運作。以下各小節則再針對伺服端系統內部各系統做進一步的說明。

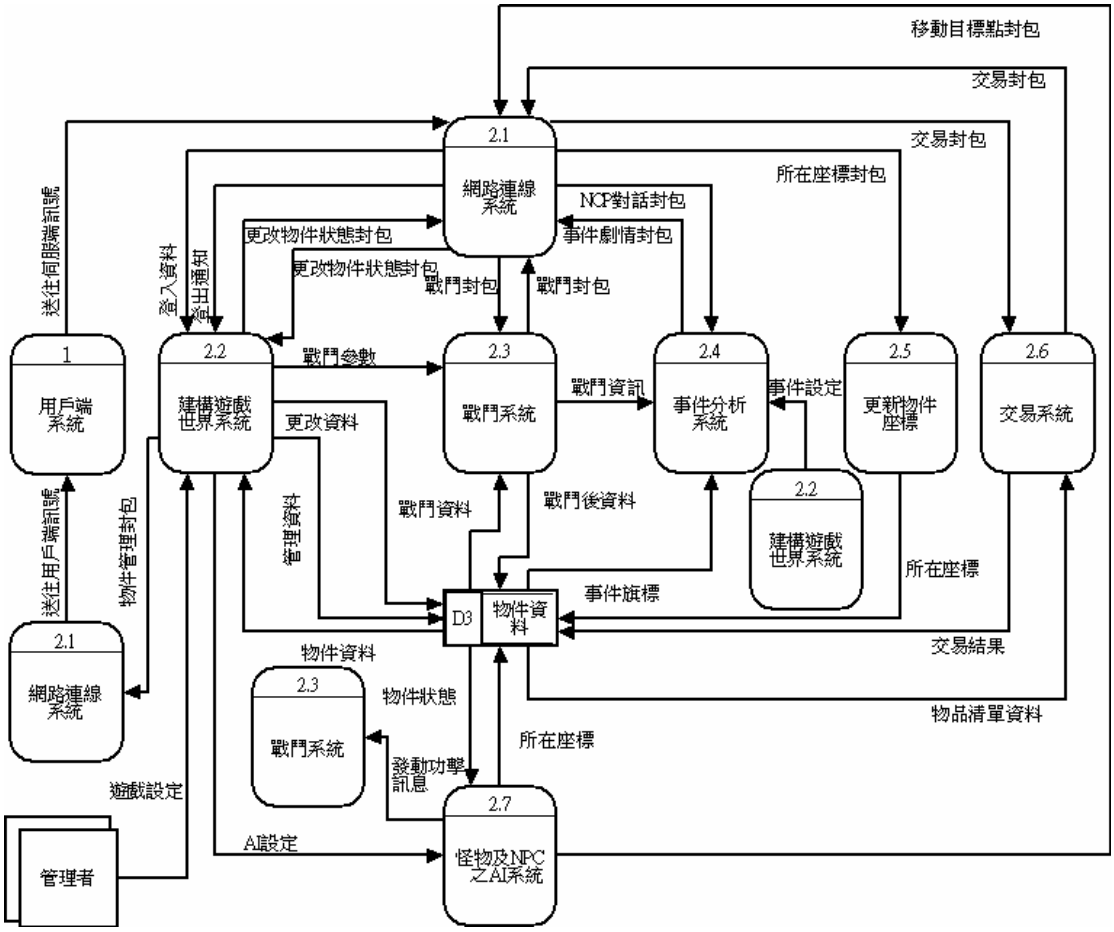

圖 4-20 遊戲系統的 DFD 圖 2-伺服端系統

4.3.1 網路連線系統

伺服端的網路連線系統和用戶端的網路連線系統有差不多的功用，其架構圖如圖 4-21 所示，多了封包轉送的功能，都是負責伺服端跟用戶端之間的溝通。但在伺服端，有幾個不一樣的功能，以發送封包而言，伺服端是和所有用戶端連線，但傳送封包的對象，並不需要傳送給全部的玩家。就像是傳送移動封包，在 A 地圖上的怪物移動方式，在 B 地圖上的玩家並看不到，所以，就沒必要傳送這類的封包給位於 B 地圖上的玩家。

而解決這方面的問題，我們將玩家加以分類成族群(group)，傳送封包的對象就是以族群來分發傳送，並藉此減少傳送封包的流量。此系統內部資料流細節如圖 4-22 所示。

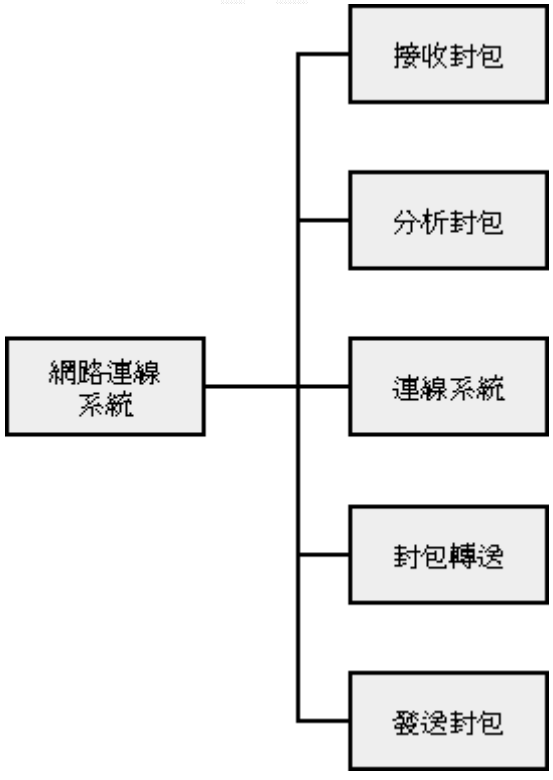

圖 4-21 網路連線系統架構圖

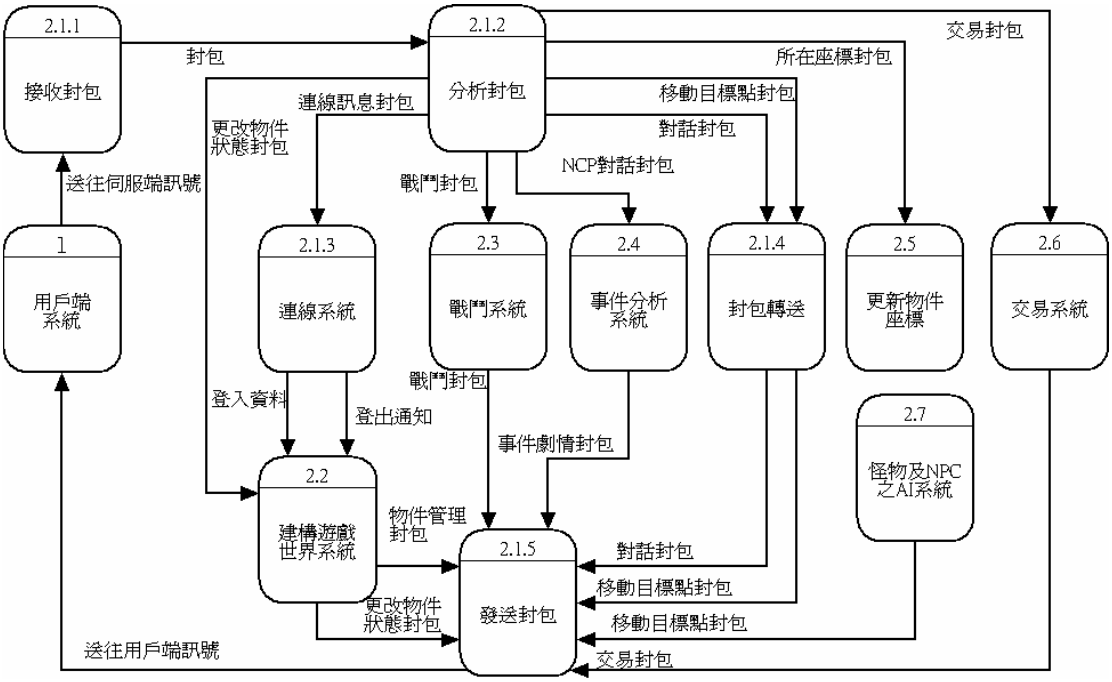

圖 4-22 伺服器系統中的”網路連線系統”的細節

4.3.2 遊戲世界建構系統

這個系統的架構圖如圖 4-23 所示，此系統會依照管理者的設定值，建構出一個真正的遊戲世界，而系統內部情形則如圖 4-24 所示。設定值包括，那些怪物或 NPC 該在那個區域出現，一隻怪物有多少血量、攻擊力、及防禦力等數值，而在某個區域中又會有多少數量的怪物，這些資料，物件管理系統都要一一記錄，如果怪物不幸戰亡，也會馬上再生出一隻，以維持原本所設定的數量值。

而在帳號檢查這個部分，會檢查玩家的帳號密碼，以確定是否為本人登入遊戲。如果玩家備份資料顯示，已經有之前玩過的進度，則將會到備份系統去讀取出最新存檔，使玩家接續玩下去；如果是第一次登入，將會依照初始設定，給於一開始的出生點和屬性，正式加入遊戲的行列。

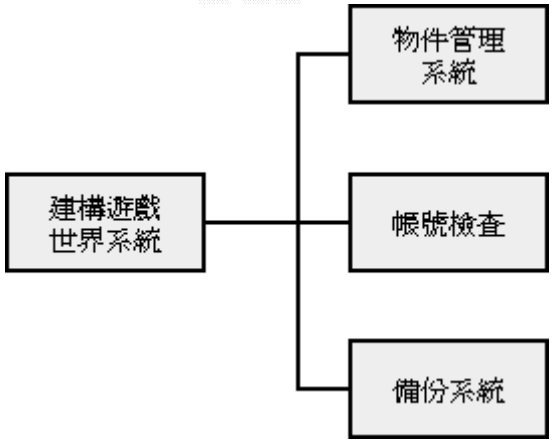

圖 4-23 『建構遊戲世界』之系統架構圖

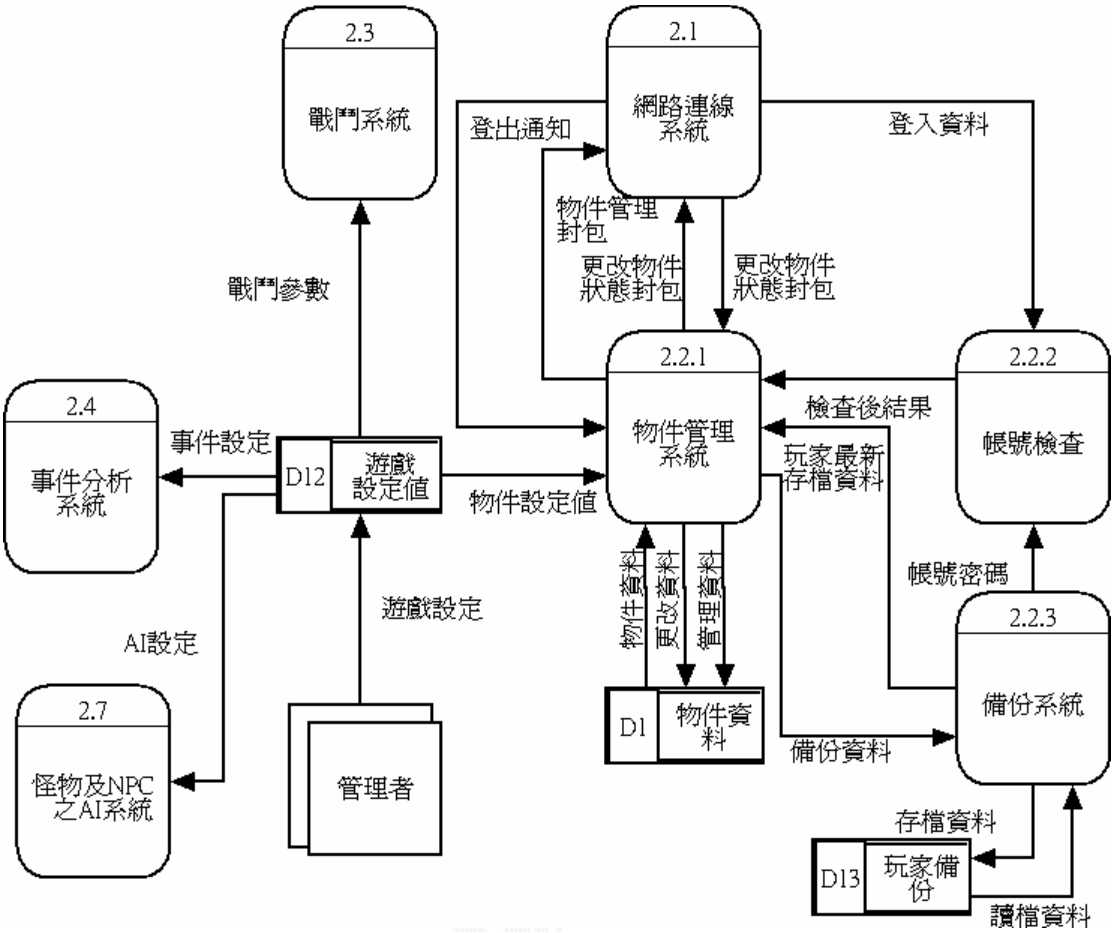

圖 4-24 伺服器系統中的”建構遊戲世界系統”的細節

4.3.3 戰鬥系統

戰鬥系統是劇情表現上重要的一環，比如說，玩家角色必須透過戰鬥來增加他的等級能力，或則某某村民受到攻擊時，必須伸出援手，幫他擊退壞人，以此展開一連串的冒險旅行，戰鬥會帶來刺激的感覺讓遊戲有更多的吸引力。而這個戰鬥系統會處理玩家與怪物之間攻擊所造成的傷害計算以及所獲得的經練值(用來計算能力升級的準則)。

4.3.4 事件分析系統

一連串事件的發生也就是劇情的表現。換句話說，劇情是由各種大大小小的事件連串在一起。因此，管理者可以設定一個故事檔，說明著種種觸發事件的條件，如一個簡單的例子：只要玩家跟某位 NPC 對話就算觸發事件，此時，系統就會到設定檔中找出 NPC 該講的那一段劇情對白，將它顯示到螢幕上，此時，玩家就能依照劇情內容前往下一個地點。同樣地打死一隻怪也能觸發事件，所以就靠這種方式來設計出豐富的劇情內容出來。

4.3.5 更新物件座標

這個部分的想法是蠻單純的，因為在之前有提過，玩家的移動方式是靠傳送目標點的方式，但實際上，玩家走到那一個座標點是不確定的，因此，就由玩家傳送出他所在位置座標，伺服端就會以此座標當作各種判斷的準則，如是否進入怪物攻擊範圍內的判斷。

4.3.6 怪物及 NPC 之 AI 系統

圖 4-25 架構圖中所示本系統是由兩個子系統組成，其中一個是行為判斷系統，怪物和 NPC 如何活動將由這個系統控制，像是何時該走動、走到哪裡，受到攻擊該如何反擊等。而怪物又可分為主動怪或是被動怪，主動怪是看到玩家進入到他的可視範圍內，就會主動追玩家到攻擊範圍內進行攻擊，被動怪就比較單純一點只有當玩家攻擊怪物的時後才會進行反擊，一旦怪物進入戰鬥狀態時後，當玩家逃跑的時後，就會猛列地追擊，直到超過所設定的區域範圍才會停止。而 NPC 是較屬於定點的方式移動，避免 NPC 隨便亂移動，造成玩家找不到重要的 NPC 進行劇情對話，或是進行交易。

另一個系統為計算最短路徑，它將從行為判斷系統中所得到怪或 NPC 下一個移動座標點，加以計算它們的最短移動路徑，一方面這可以減少行為判斷系統要去計算每下一步的位置；另一方面，在傳送移動封包的時後，只要傳送目標點，用戶端以同樣的公式就能算出相同的移動路徑。

此系統內部較詳細的資料流如圖 4-26 所示

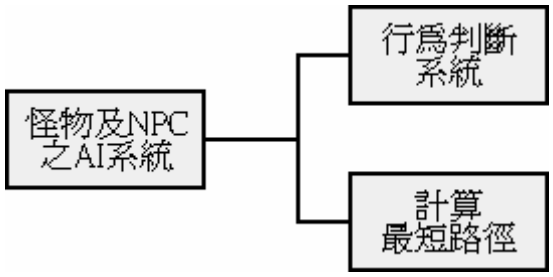

圖 4-25 怪物及 NPC 之 AI 系統架構圖

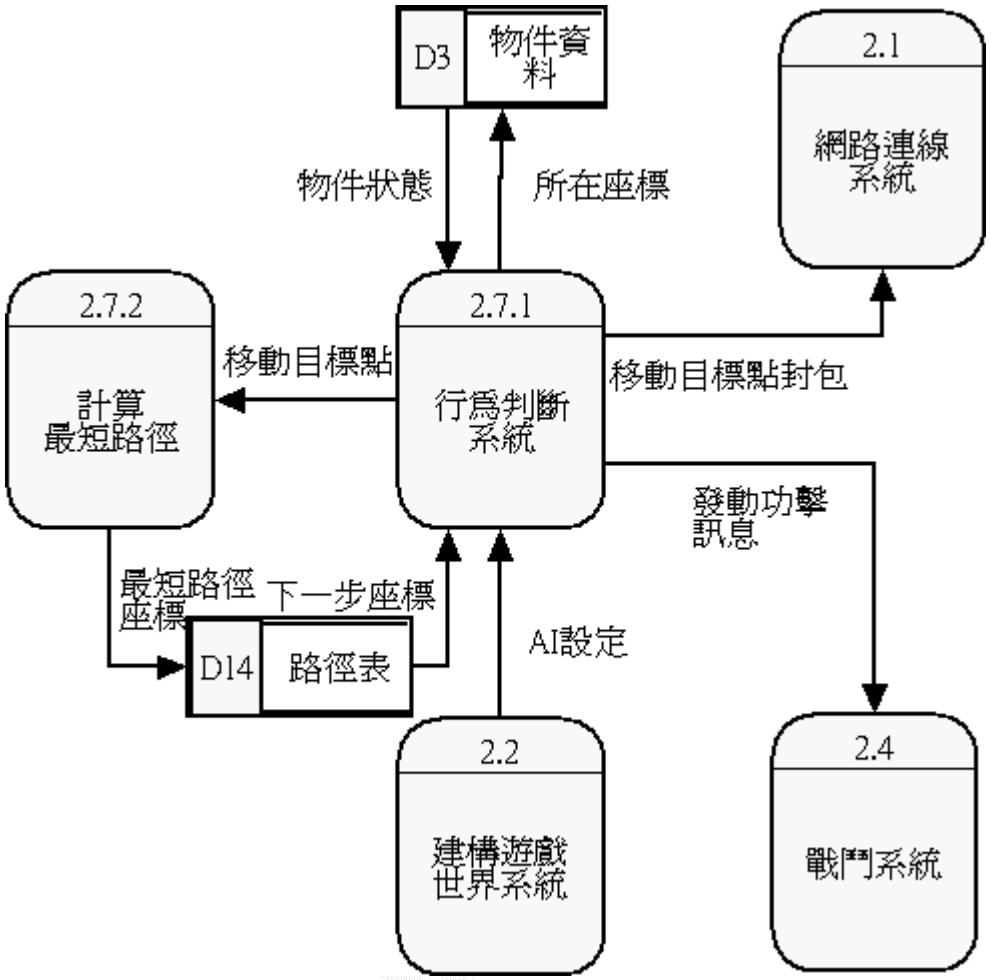

圖 4-26 伺服器系統中的”怪物及 NPC 之 AI 系統”的細節

第五章 系 統 開 發 與 整 合

上一章說明整個系統的架構，我們對子系統的細節也有了進一步的規劃。接著，本章的主要目的就是說明我們在製作子系統時，所用到的技術及方法。在 5.1 節會提到在開發過程中，我們所使用的軟體以及分工的情形，隨後的各小節便是子系統的實作和整合部份。

5.1 開發前的準備工作

開發前需要把系統做好分析，決定好軟體工具，再針對各別所專長的能力分工。以下會先說明一下業界是怎麼分工的，再依我們的狀況定出小組分組。

5.1.1 軟體的準備

程式開發—Visual C.NET

- API—Direct X 8.0 SDK
- 2D 影像處理—Adobe Photoshop 7.0
- 3D 繪圖—Sketchup

5.1.2 工作分配

本節內容主要是介紹我們分工的情形，在這之前，會先說明一下遊戲業界其遊戲開發小組的分工情形，以作為我們分工的依據。

一、業界分工情形

一、程式

- 客戶端、伺服端、資料庫、開發工具
- 付費機制、後端管理工具、GM 系統等

二、企劃

- 執行企劃：數值調整、內容設計
- 文編企劃：遊戲腳本、劇情製作
- 助理企劃：資料收集、品管測試

三、美術

- 平面美術：插畫繪製、造型設定、場景設定、影像處理、後製處理
- 3D 美術：模型製作、動態編輯、材質繪製、動畫製作、特效處理

四、音樂

- 音樂取得、音效編輯

二、本組分工情形

通常一個遊戲的品質主要由分工能力與經驗所控制，接著影響的才是時間因素，而人力支援又與開發時間略成反比。很顯然地，要在短期開發出品質優良的作品並不是不可能的事，但其中的變數在於小組人員的素養與創作經驗上，而整體事前規劃與細部分工則是推動開發小組的原動力。

但是我們缺乏經驗且人員有限，在此前提下，要如上一小節介紹之業界分工方式來分工，是非常困難的；且在實作的過程中，往往需要不停的開會分析，才能決定出具體的方向，此舉便浪費許多時間與體力。而在人員有限的情況下，往往一個人要身兼數職，且某些地方，如美術、音樂創作，我們可能就要對品質作出極大的犧牲或讓步。

表 5-1 便是我們的分工情形。

表 5-1 專題分工表

	明政	志信	欣宗	欽賢
戰鬥系統			o	
視窗系統		o		
創造地圖系統	o			
音效系統				o
物件管理系統		o	o	
OpenGL 圖展示系統	o	o	o	o
網路連線系統				o
OpenGL 點選系統		o		o
對話系統				o
交易系統			o	
2D 人物移動系統			o	o
事件系統		o		
怪物及 NPC 之 AI 系統			o	o
執行企劃(數值調整)	o	o	o	o
文編企劃(遊戲腳本、劇情製作)	o			
平面美術(場景設定)	o			
3D 美術(模型製作、材質繪製)	o			o
3D 美術(動態編輯)	o		o	o

5.2 地圖編輯系統

此系統用來編輯遊戲裡的地圖場景，也就是遊戲進行的舞台。系統所負責的範圍非常廣：包括地形高低設定、地表貼圖、地圖上 3D 物件之製作、擺設、特殊旗標、及部份劇情發展等，最後所完成的地圖檔案會經由一個特定格式的文件檔，套入遊戲中使用。5.2.1 節便簡短地介紹整個系統的運作流程，往後各小節再依照流程一一詳細介紹裡面的功能如何運作。

5.2.1 地圖製作的流程

一、依劇情繪出地圖草稿圖

- 決定劇情所需的地圖草稿
- 繪出地形高度圖：俯視整張地圖，將不同高低的地形用不同深淺的顏色表現出來。
- 設定 3d 物件：依草圖決定 3d 物件的數量以及所在座標。
- 設定可走與不可走的位置：決定哪些位置可以通過、哪些不行。
- 特殊旗標：設定哪些座標會觸發特殊事件。

二、製作地表

- 製作地形骨架：程式依圖檔顏色深淺的程度，製作出地形高低起伏。
- 地表貼圖：將顏色圖檔以及材質圖檔結合，貼附在地表。

三、3D 物件製作

- 3D 物件繪製：3D 物件製作方式，利用 SketchUp 軟體。
- OpenGL 讀 3ds 檔案原理：介紹 3ds 檔案的內部結構，然後依程式碼說明系統讀取 3ds 檔案的方法。

四、特殊旗標

- 地面座標設定：地面座標的判定方式。

- 特殊旗標：特殊旗標所在的座標，以及其判斷方法。

五、整合

- 製作地圖資訊文件檔。
- 檢查視覺效果與修改。
- 架構地圖之間的關聯。

由於整個遊戲系統的架構非常複雜，因此往往某一功能並不是完全由某一系統開發出來，而是多個系統整合的結果。以上說明的許多功能也多少跟其他的程式有所關聯，但因為功能比較偏向於地圖的製作，所以全部包含在這個章節中。

5.2.2 依劇情繪出地圖草稿圖

一、決定劇情所需的地圖草稿

製作地圖是整個遊戲進行很重要的一部份，不管遊戲正在進行什麼動作，都會發生在地圖之上。比如，戰鬥的發生、角色的移動、NPC的移動、以及玩家行走時程式所決定的路徑等等，全部都需要經由地圖這個舞台來整合。另外地圖上 3D 物件、建築的安排還有場景的設定等背景功能，也全部整合在地圖編輯系統之中。由於編輯一張完整的地圖需要考慮到如此多的因素，所以在一開始的草稿以及初步分析時就必須很謹慎，要多方面考量之後再開始製作，以免到後來發生完成的地圖不適合遊戲進行的狀況。

依照遊戲劇情的需要，首先，構思出一個劇情要上演的舞台。考慮到遊戲故事的背景、風格以及要表達給玩家的感覺，先以手工繪出地圖的大略草圖，如圖 5-1，經由組員討論（尤其是編劇負責人的意見最為重要）之後，不斷的修改直到大家都滿意為止。修改到一定的程度、或是地圖的複雜度太高難以用手繪來表示時，可以使用簡易的 3D 物件作為參考，如圖 5-2，不但可以直接的表現出地形的高低起伏，也可以粗略的標出 3D 物件的位置安排，以方便組員更快並且更明瞭的想像出地圖完成後的樣子，並且能夠提供給地圖製作者更好的意見。

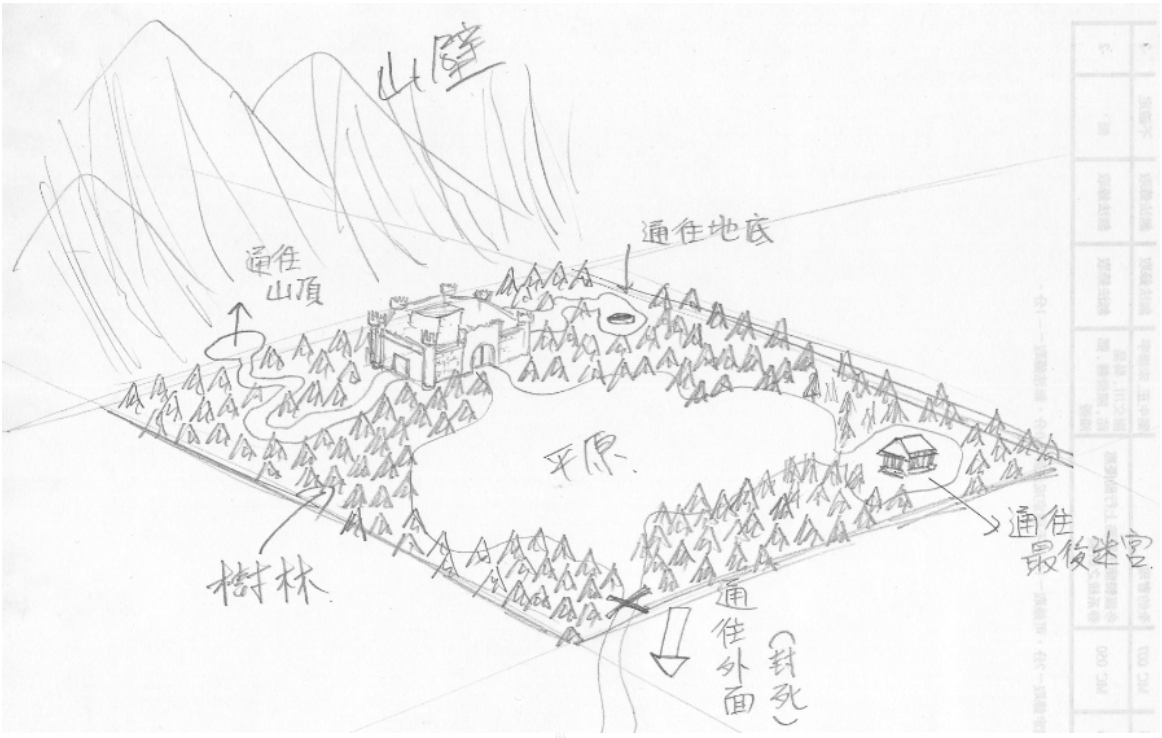

圖 5-1 地形草圖（手繪）

另外，在繪製地形草圖時，必須注意到下面幾點：

- 盡量誇大需要表現的部份，因為這是組員之間藉以討論的重點。
- 大略的塗上地圖所要呈現的色彩（但不必很詳細），讓人能看出整張地圖的感覺。
- 要盡量按照比例的做出高低起伏，但是可以忽略小幅度的斜坡或是小山丘之類的地形。
- 由於是用來參考用的，所以力求簡潔但不失真。不管色彩、風格、及地形高低等等，都要讓人能夠一目瞭然。

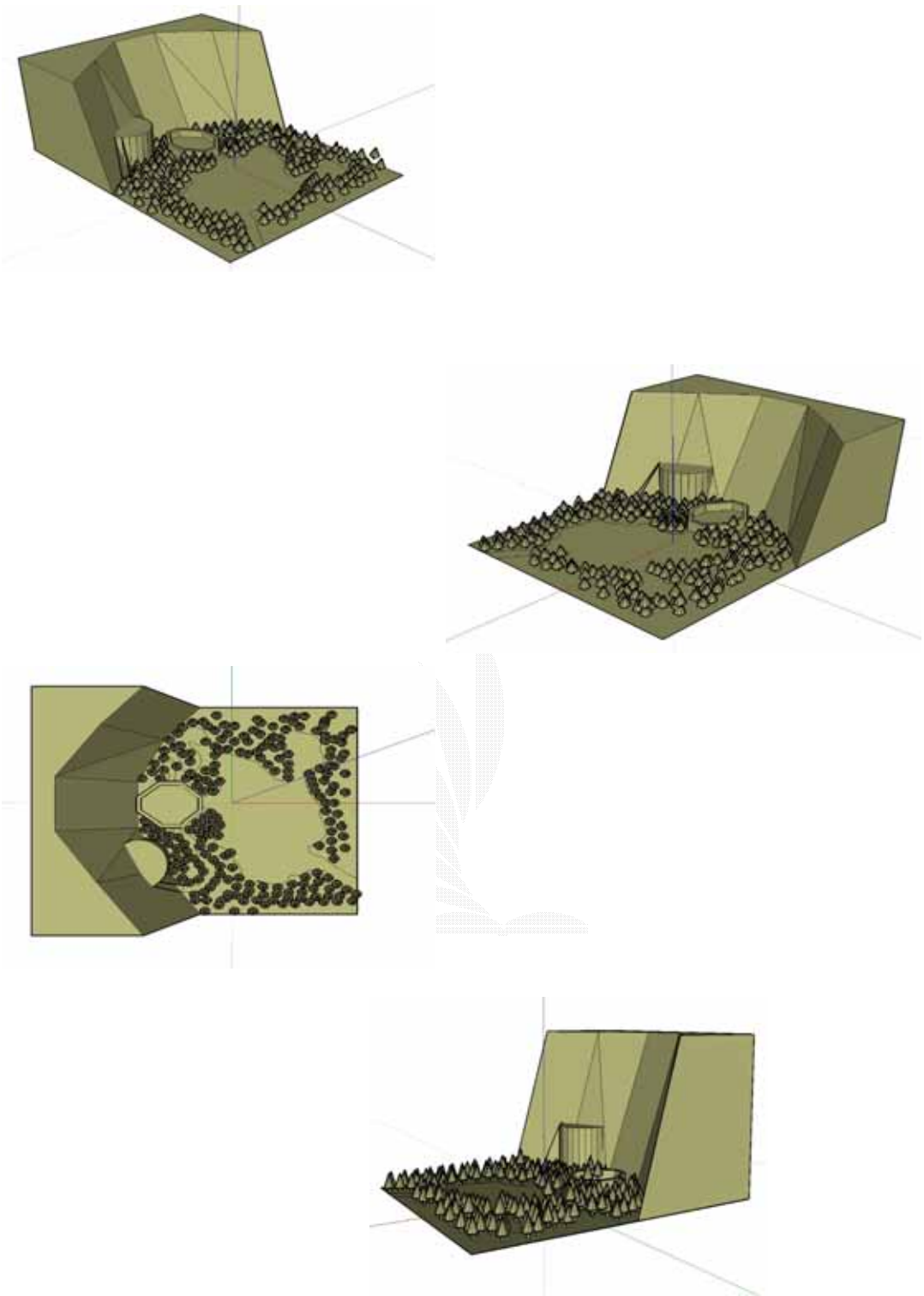

圖 5-2 輔助用 3D 地形模型（以 SketchUp 軟體製作）

二、繪出地形高度圖

依照地表高低的不同設定，製作一張「.raw」圖檔，利用單一顏色的深淺不同，經過程式的運算後，會自動產生出一個有高低起伏的地表，亮度為 0 的部份是整張地圖最低的地方，愈亮的部分就會產生愈高的地形，範圍高低可在 0~255 之間自由變化，是整張地圖的最基本的一層，之後的其他物件都是架構在地形圖之上。

「.raw」檔的特色，就是它的內部資料沒有所謂的標頭檔，或是其它記錄任何屬性的資料，而且顏色也一概只有單色（黑白），從檔案的左上角第一個像素開始記錄，每一個像素只有一個資料：深淺數值，由 0 到 255 表示不同的深淺。由於「.raw」檔的架構如此特別而簡單，讓我們可以利用這個方法很方便的製作出我們所要的地形。下圖便是「.raw」檔的範例。

圖 5-3 用來計算地形高低的「.raw」檔

三、設定 3D 物件

先依照劇情的需要，分別列出要排在地圖上的 3D 立體物件，可以是建築、花草樹木、及特別的岩石等，全部都在要考量的範圍內。然後把會用在地圖上的所有 3D 物件列成一張表格，內容須包括物件的名稱、數量以及簡易的說明，以便在 3D 物件製作的時候可以一目了然。表 5-2 便是我們遊戲中會用到的 3D 物件列表。

表 5-2 3D 物件列表

物件名稱	數量	說明
Tower	10	城牆間用來瞭望的高塔
Wall	100	城牆
Main_Gate	1	城鎮大門
West_Gate	1	城鎮西側門
East_Gate	1	城鎮東側門
House_1	2	民房-1
House_2	4	民房-2
House_3	3	民房-3
House_4	2	民房-4
Weapon_Store	1	武器店
Item_Store	1	道具店
Hotel	1	旅館
Tree_1	60	平原上的樹-1
Tree_2	60	平原上的樹-2
Tree_Snow	20	積雪的樹（用於山頂）
God_Temple_1	1	神廟-1（城鎮內）
God_Temple_2	1	神廟-2（森林中）
Well	1	古井（地底世界入口）
Transport	4	傳送點
…		
…		

四、設定可走與不可走的位置

地圖上多數的地方都是玩家或是怪物可以行走的，但是有一些特殊的位置座標沒有辦法通過，例如坡度超過一定角度以上的斜坡、有3D物件所在的地點(例如長樹的地方或是城牆、及建築的所在地等)，這些設為不可走的部份會把自身的座標屬性存入陣列之中，其最主要的功用就是計算移動路徑的程式在運作時，會自動把這些不可走的點列入考量，來計算出人物移動的最短路徑。

五、設定特殊旗標

當玩家走到某一特定的地點(或某一特定範圍)的時候會觸發事件。這項功能在遊戲劇情運作中，佔很重要的一環，往後跟劇情系統所提供的功能做連結之後，可以選擇多種觸發條件以及觸發結果，來達成故事所要表達的特殊效果。例如，跟某一位NPC說話之後就可以得到開啟某些門的鑰匙、走到某一地點時石門會自動打開、或是移動到傳送點上的時候會被送到另外一張地圖等(不過其中當然只有跟地圖有關的旗標、需要在這個系統裡設定)。

5.2.3 製作地表

一、製作地形骨架

製作好地形高度圖的「.raw」檔之後，由 cMAP 類別下的 Loadmap() 函式呼叫 LoadRawFile()後，LoadRawFile()會將 raw 檔案中的資料讀入記憶體，並且回傳陣列給 Loadmap()。Loadmap()將陣列的內容做分析後，將檔案中每一個對應點的明亮度數值讀出來，經由簡短的程式碼將數值放大十倍後存入陣列 m_TexturePool 中。

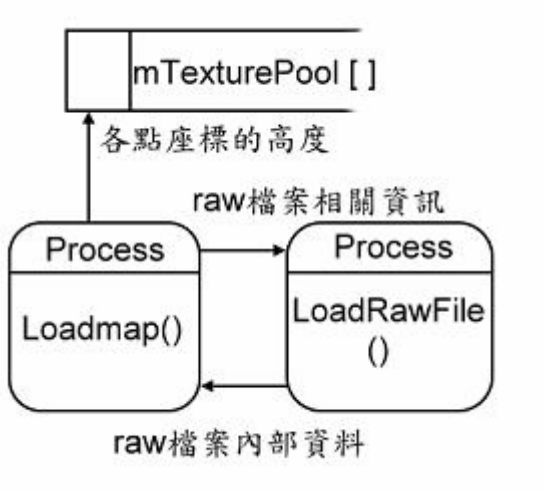

圖 5-4 讀入 raw 檔的步驟

讀入所有的座標高度之後，再利用 OpenGL 的繪圖功能，把整個地圖的骨架建構起來。首先是原本已經存好的平面座標 (X,Y)，之後再把剛剛完成的高度座標 (Z) 跟平面座標，合成一個點的完整立體座標 (X,Y,Z)。程式中所有的面都是由三角型組合而成，也就是三個點的座標會先構成最基本的面，之後再由許多的三角形面組成不同的地形。如圖 5-5、5-6。

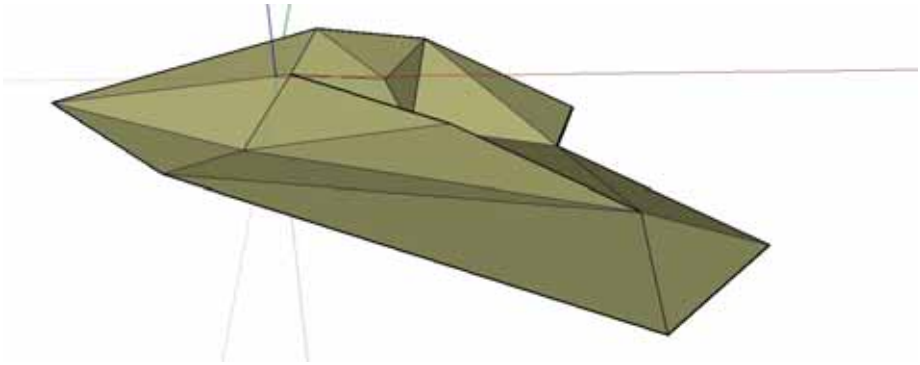

圖 5-5 由三角形構成的地表

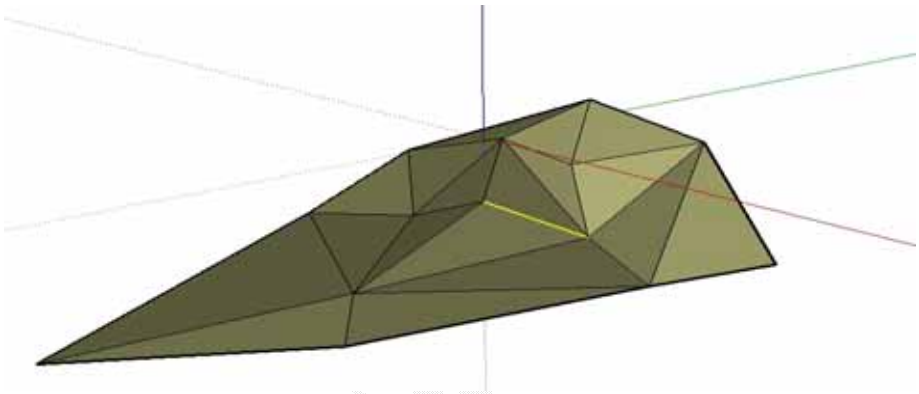

圖 5-6 由三角形構成的地表

另外 OpenGL 之中有一個很重要的功能，可以在繪製平面三角形的時候大量減少記憶體的使用。一般陣列的儲存方法，是將每個點的座標一一存起來，然後每讀出三個點的座標之後再組合成一個三角形。但這種方法其實浪費掉許多不必要的記憶體空間，因為每一個三角形的三個點座標是跟前一個三角形的座標重複的，沒有必要再把它當作另一組數據存起來。在 OpenGL 中，只要按照特定的步驟把點的座標按順序存入陣列中，那麼在繪圖的時後，程式會自動判斷座標間的關係，把新讀入的一組座標，跟前兩組座標合成一組新的平面。這樣不只能夠更快的繪出圖形，更有效的節省了將近 2/3 的記憶體資源。

圖 5-7 中，實心的圓點代表最近一次讀入的點座標，空心的圓點代表已經讀完的資料。由上圖可知一般的方式儲存資料，畫出三個完整的三角形，需要用到 9 個單位的繪圖時間以及 27 個記憶體位置，而且其中有大半的資料是重複的。這麼做不僅沒有節省記憶體空間，而且沒有辦法加速 3D 繪圖的速度，後者更是製作遊戲時最忌諱的情況之一。

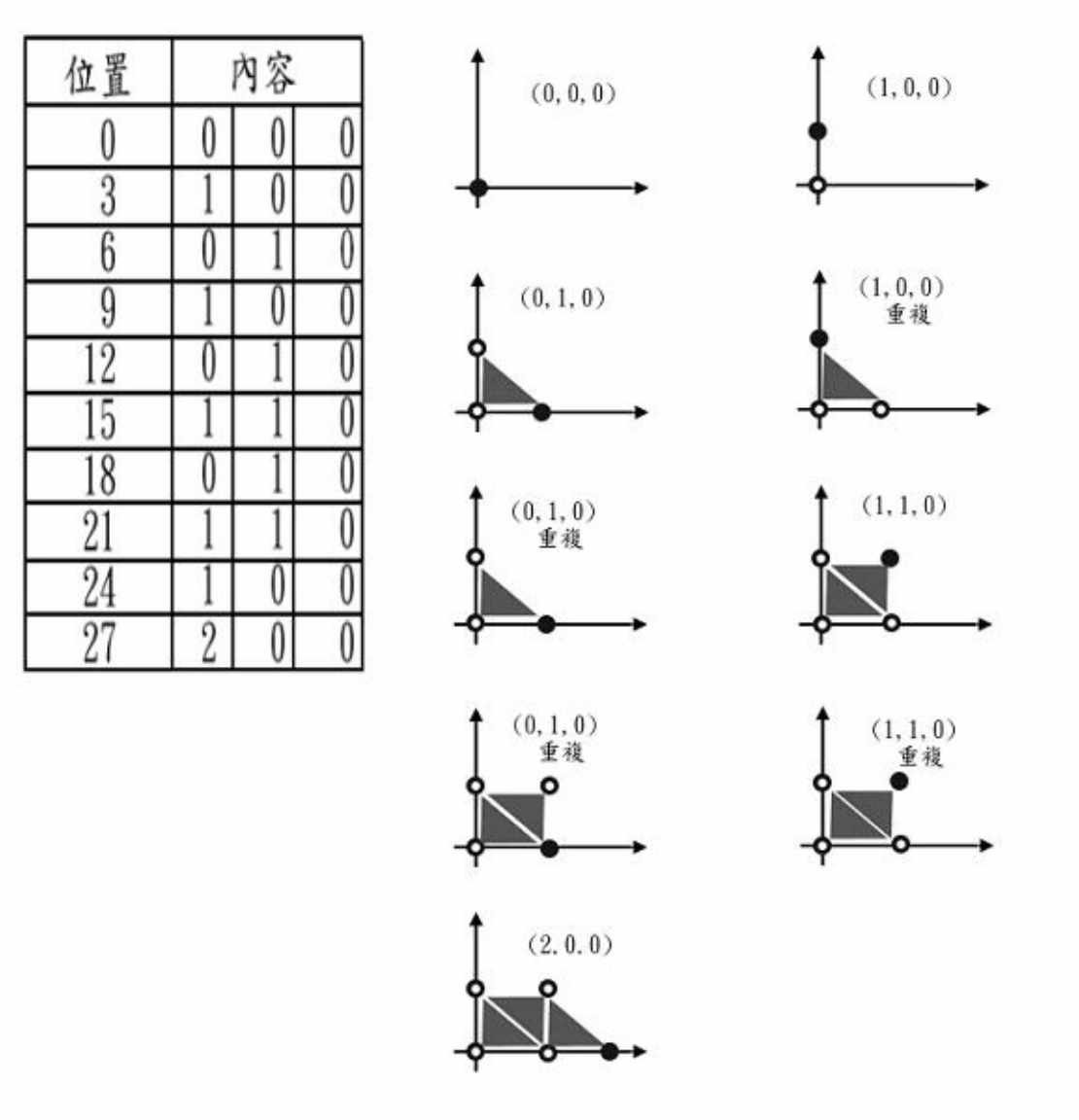

圖 5-7 一般程式的陣列讀取法

如同大家所知，OpenGL 中除了有可以讓硬體加速的函式之外，還提供許多方便運算以及節省記憶體的運算函式。上面稍微提到的三

角型加速繪圖法，就是節省記憶體並加速繪圖的函式之一。m_TexturePool 陣列會將內部的資料處理過後，縮減記憶體的使用量，把重複的點座標資料清除。如此一來就可以照著 OpenGL 的演算法直接讀入下一個點的座標後，跟仍然保留在記憶體內的前兩組資料運算，直接在螢幕上畫出下一個三角形。圖 5-8 為 OpenGL 的陣列讀取方法。

由圖 5-8 可以發現，同樣是畫出三個三角形，OpenGL 的陣列讀取方法只需要 5 個單位的繪圖時間以及 12 個記憶體位置，而且越往後畫下去差距會越來越大，到達一般陣列讀取方式的 1/3 左右。這樣一來不但可以讓遊戲進行的更順暢，還可以減少相當大的記憶體空間，對一個遊戲程式來講，實在是不可多得的一個優點。

位置	內容		
0	0	0	0
3	1	0	0
6	0	1	0
9	1	1	0
12	2	0	0

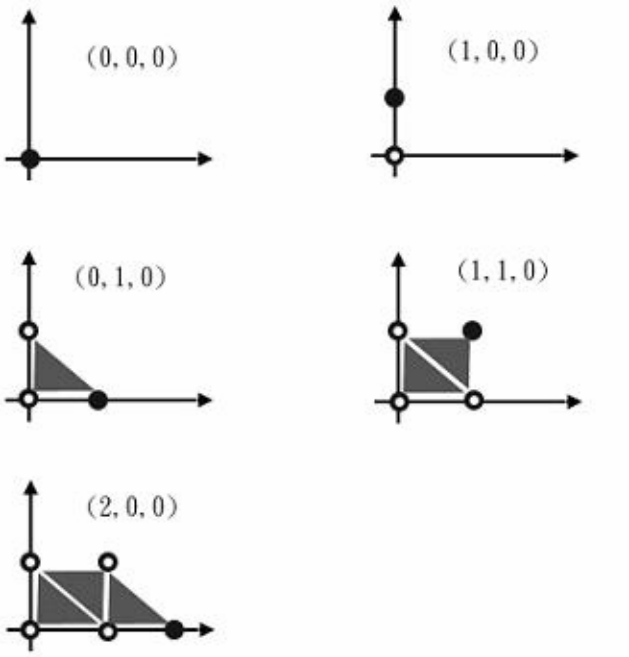

圖 5-8 OpenGL 的陣列讀取法

二、地表貼圖

當畫面上所有的三角形都畫完了之後，算是完成了地形的基本骨架。如圖 5-9 所示，大致上可以看出整個地形的高低起伏，還可以看

到許許多多的三角形是如何把地形拼湊起來的。但是這樣當然根本不能算是完成，除了把每一個面畫出來之外，還要把所需要的圖片貼上去才行。

圖 5-9 部分地圖的透明骨架

在貼圖系統中，我們採用了雙重貼圖的技術。所謂雙重貼圖，就是視我們的需求，把貼圖拆成幾個層次來個別動作。而在我們地表貼圖的系統中，把整個貼圖的工作拆為兩個層次：「地面色彩分佈圖」跟「地面材質貼圖」。

- 地面色彩分佈圖

就是指地面所要呈現的顏色。首先參考劇情之後畫出一張符合劇情風格的地表色彩分佈圖，利用不同的顏色可以先大致上地區分出不同的地形，並且可以用較淺的顏色來表示道路，讓玩家不至於迷路。如圖 5-10。

- 地面材質貼圖

地面除了要有顏色之外，還需要有一張材質貼圖。材質貼圖可以表現出地表細部的質感，並且做出模擬實物效果的畫面（例如：岩石、草地、及石磚路面等）。

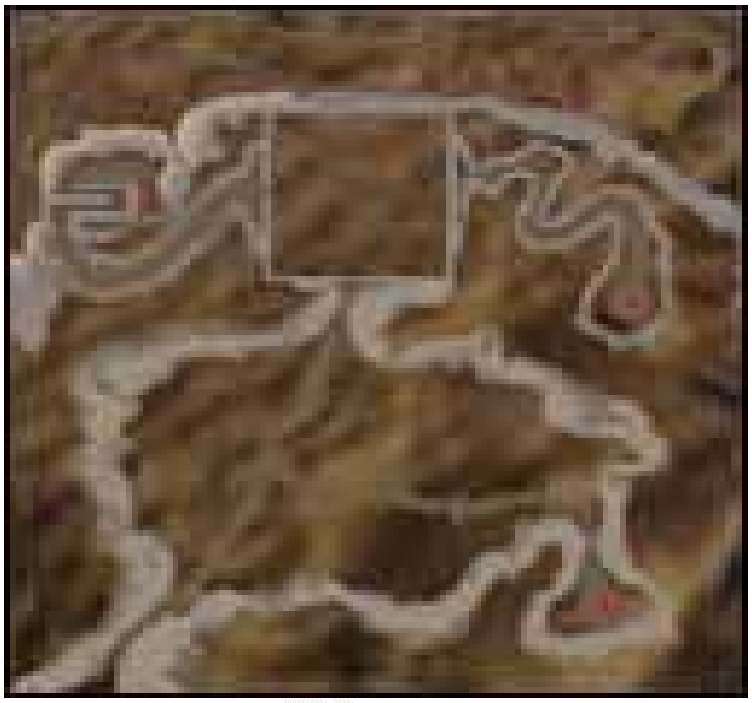

圖 5-10 地面色彩分佈圖

由圖 5-10 可以很清楚的看出地形上大致的分佈，還有道路的部份也用灰色表現的很明顯，可以帶領玩家前往會發生劇情事件的關鍵地點。不過如果地面只有這樣單以顏色來表現的話，會顯的過於單調。整張圖看起來的感覺雖然還好，但是真的進行遊戲時由於畫面會放的很大，地面會看起來只有一大塊一樣的顏色，如圖 5-11。

圖 5-11 只用一張地面色彩分布圖時，地面的效果

因此為了刻畫細部的地表，我們使用了另一張材質貼圖，如圖 5-12。將兩張圖重疊，貼在剛剛製作好的地形骨架上，就可以展現出很棒的地面效果。如圖 5-13。

圖 5-12 地表材質貼圖

兩種材質不同的是，色彩分佈圖是一整張圖剛好貼滿整個地圖骨架；而材質貼圖是以小範圍重複貼圖，利用不斷貼上一樣的材質貼圖而佈滿整張地圖。因為不是放大之後再貼，所以材質貼圖可以展現出很細微的地面特徵，如此一來整張地圖就能夠非常的完整展現出來。

圖 5-13 貼上材質之後的地面效果

如此一來，地表的製作算是完成了。下一步要在地面上安插原本預定的 3D 物件，如此一來就可以塑造更完美的地圖。

5.2.4 3D 物件製作

一、3D 物件繪製

使用 SketchUp 軟體編輯。此軟體多半是運用在建物方面，因為它用來編輯建築物之類的 3D 物件時，非常方便。而這也是我們選擇使用 SketchUp 而不是使用知名的 3D 編輯軟體(如 3D studio Max、Strata、及 Maya 等)的原因。相較起來，SketchUp 的編輯程序實在簡便許多，省下了學習軟體所花費的時間以及編輯 3D 物件的時間。但由於 SketchUp 軟體比較不常見，以下便介紹編輯 3D 物件的流程、常遇到的問題、以及解決方法。

根據先前列出的 3D 物件列表，決定出要製作的 3D 物件。決定好之後先以手工繪出物件的初步模型，不必在意畫的詳不詳細，重點著重於整體的架構表現，以便於實際製作的時候，能夠很清楚的知道整個物件的結構，不僅在製作的過程中能夠得心應手，也會節省不少製作的時間。如圖 5-14 的神廟初步模型草稿圖，所代表的便是我們遊戲世界中會出現的 3D 物件之一。

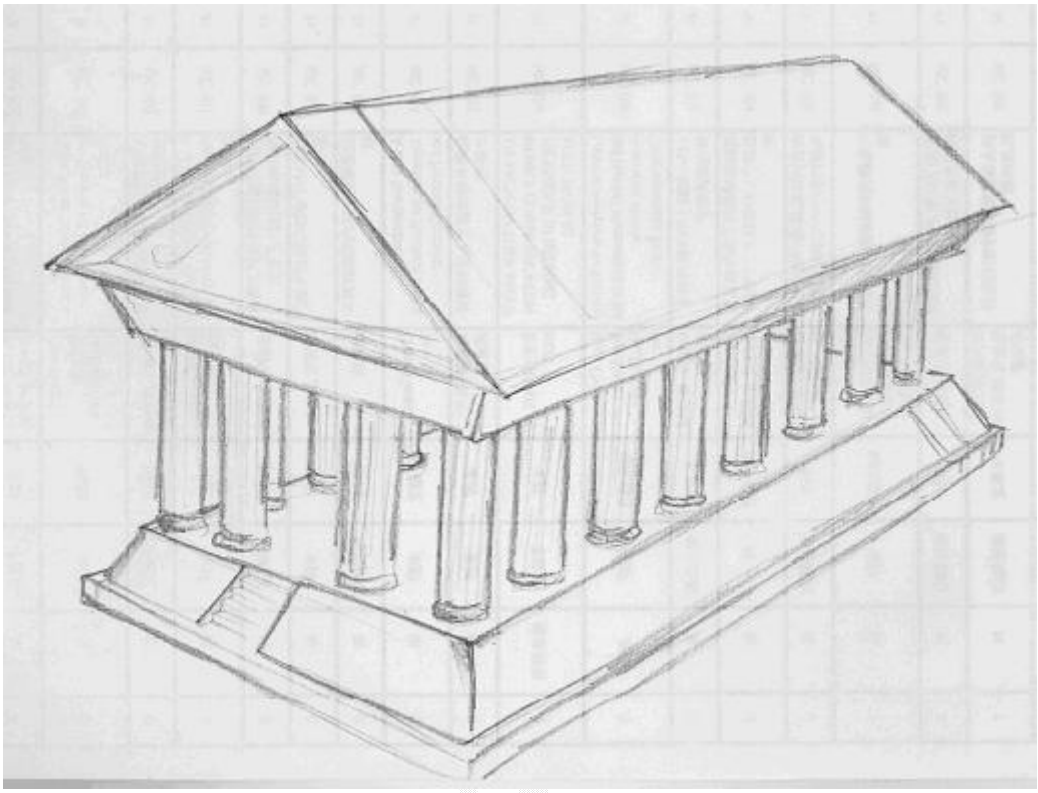

圖 5-14 3D 物件的手繪草稿（神廟-1）

完成草稿圖後，下一步就是依照比例設定出物件的大小（高度座標不用考慮，我們所要決定的大小是在於物件的平面面積佔了幾格）。由於之後要設定可走不可走的座標，因此物件的大小要盡量跟標準格子對齊，如圖 5-15，此舉除了可走不可走的座標好設之外，也方便地圖的編排。

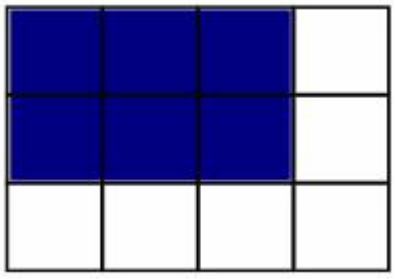

圖 5-15 正確的 3D 物件大小

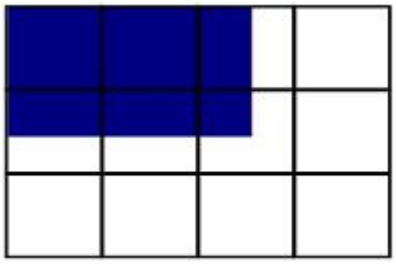

圖 5-16 錯誤的 3D 物件大小

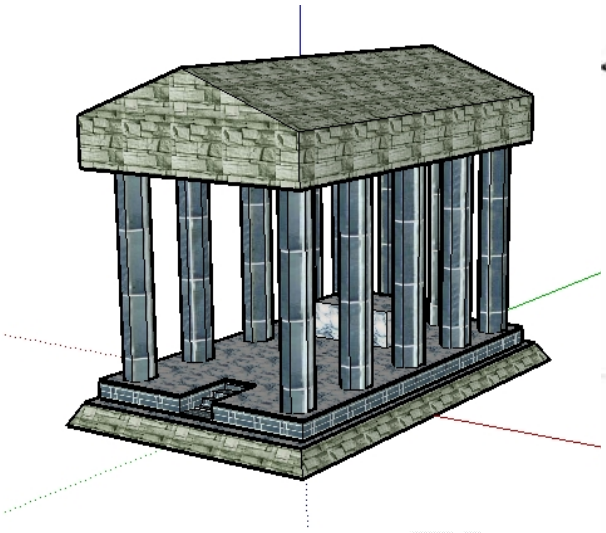

圖 5-17 整個物件

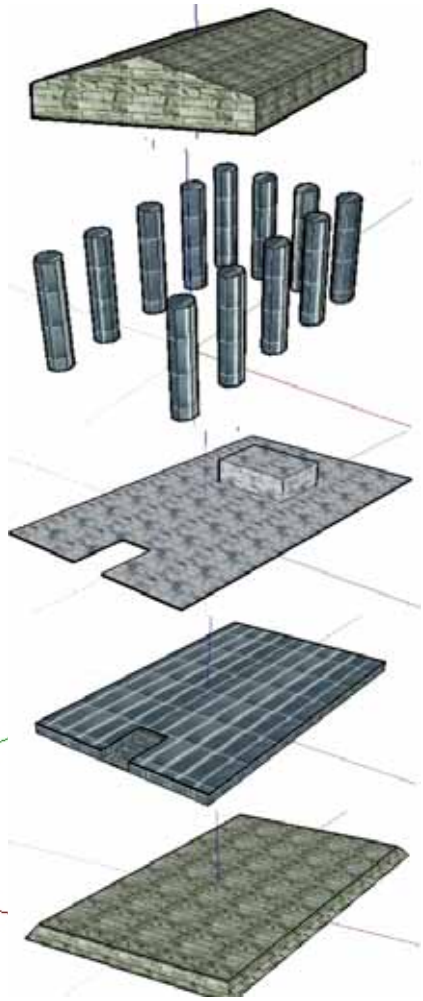

圖 5-18 各個元件的分解圖

另外還要注意的是，SketchUp 軟體製作出來的 3ds 檔，有一項很麻煩的限制，那就是單一物件中各個元件只能貼一種圖，且只要是觸碰在一起的不同元件就會被判斷成是單一元件。

因此解決方法就是，如果我們要在同樣一個物件中貼上不同材質的貼圖，就必須讓各個元件很靠近，但是絕對不能夠碰在一起。如下圖 5-19、5-20，製作的要訣就是讓各元件之間保持著 1 個單位像素的距離，這麼一來才可以在地圖上顯示出我們想要的效果。這一個步驟雖然讓我們在編輯的時候大傷腦筋，但是比起前面所說過的其它軟體帶給我們的困擾，我們還是選擇 SketchUp，因為它的編輯程序實在是太簡單了。

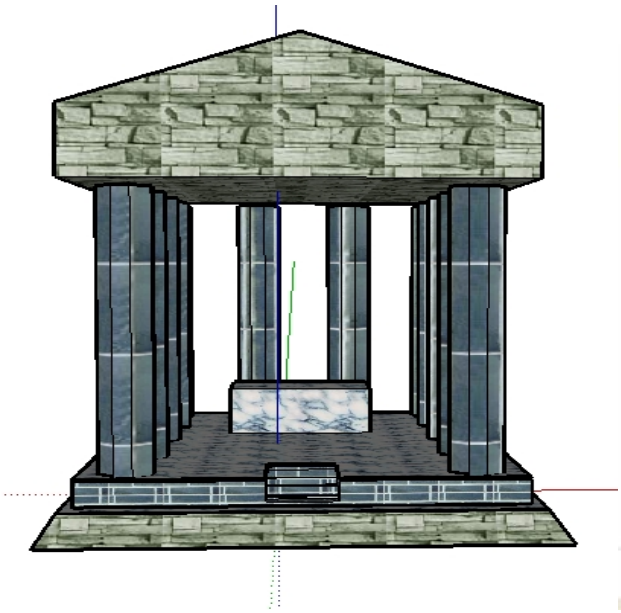

圖 5-19 物件外觀

圖 5-20 元件之間間隔的特寫

製作完成的 3D 物件，利用 SketchUp 的 Import 功能，把原本的「.skp」檔（SketchUp 專屬檔案格式）輸出成「.3ds」檔，然後利用系統中 cLoad3ds 類別中的函式多次分析，解讀成 OpenGL 能夠顯示的格式，再顯示在地圖上。

為什麼要特地將.skp 檔案輸出成.3ds 檔之後才進行解析呢? 理由是因為.3ds 檔是目前市面上最廣為接受的 3D 物件檔案格式，並且相容於多種不同的 3D 編輯軟體。而且其檔案架構的特性簡潔明瞭，捨去許多不必要的繁雜屬性資料，簡單的紀錄不同的點、線、面、及貼圖的關係，除去了 3D 物件檔案最大的缺點之一，也就是檔案格是太過龐大的問題；再者，這樣一來也使的我們的系統有更好的擴充性，不只是能夠解讀 SketchUp 所製作出來的 3D 物件.skp 檔，而是能夠處理幾乎所有 3D 編輯軟體都能夠輸出的.3ds 檔。

由圖 5-21 及圖 5-22 可發現，一個 skp 檔轉成 3ds 檔的格式之後，大小相差了幾乎 20 倍，但是解出來的資料仍然可以繪出很完整的 3D 模型。這就是 3ds 檔這麼受歡迎的原因：結構簡潔及檔案小。

圖 5-21 skp 檔案的大小

圖 5-22 3ds 檔案的大小

二、OpenGL 讀 3ds 檔原理

完成一個 3ds 檔之後，我們製作 3D 物件最後的工作，就是要利用程式把它顯示在遊戲畫面上。由於 OpenGL（其實 DirectX 也是）之中並沒有直接讀取 3ds 檔案的功能，因此我們要先自己理解 3ds 檔案的內部架構後，再設計出一套函式把 3ds 檔轉換成 OpenGL 可以讀取的型態。

這實在是一件大工程，首先先使用 UltraEdit 文件編輯程式打開一個 3ds 檔，如圖 5-23，這樣我們就可以仔細研究檔案的內部結構。

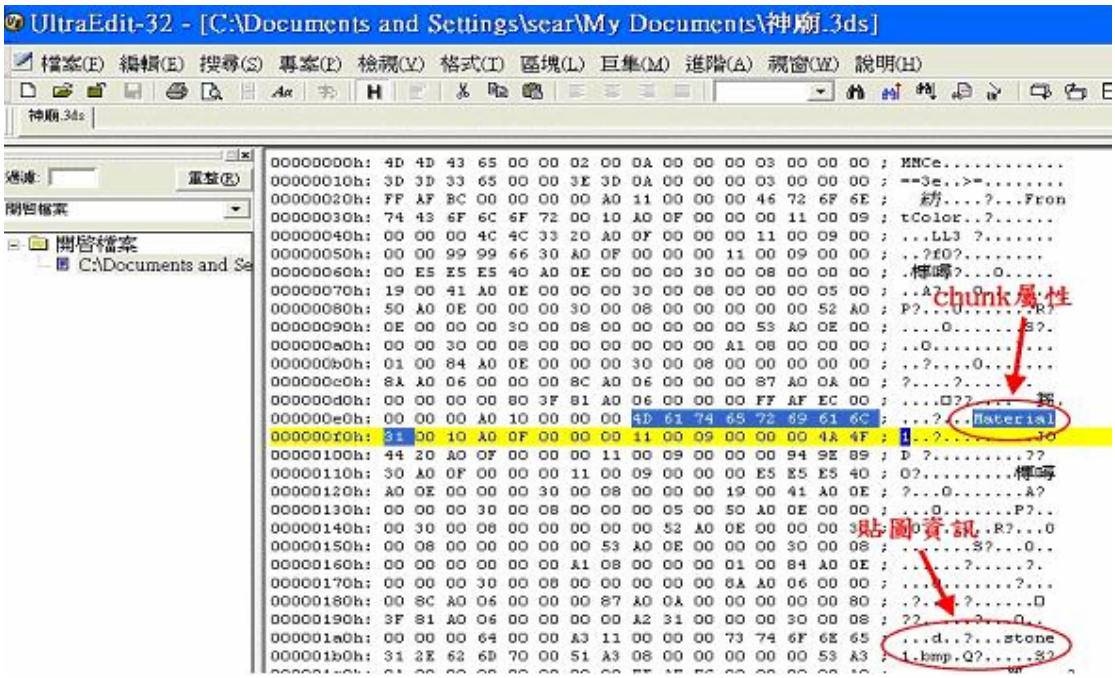

圖 5-23 3ds 檔的內容

3ds 檔案內最主要是用所謂的「chunk（典譯通譯：區塊）」來儲存每一筆資料。在整個 chunk 的最開頭顯示這一個 chunk 的屬性，一共有以下幾種：

- vertices—頂點，也就是代表 3ds 檔案中分佈的線條，由這些線條的資料才能夠繪出所有的平面。
- Face—平面，表示各個由線條組成的三角形平面，由許多的平面才能夠組合成立體的 3d 物件。（但是檔案中並沒有所謂「立體物件」的 chunk 類型，因為其實立體物件就是由平面集合而成的；但是平面在繪圖的觀點來看並不是用線條組成的，所以分開成兩種 chunk 方式儲存）
- Material—材質貼圖，就是指某一個特定平面上所要覆蓋的圖片，簡單的說就是替這個 3ds 檔案上色，程式會依照其 chunk 的內容指定位置尋找到所要的貼圖檔案所在，然後用其所指定的方式把圖片貼滿整個平面。

緊接在 chunk 屬性後面的一大串資料，就是這一個 chunk 的內容，依照其屬性的不同，資料所代表的意義也就不一樣。詳細的分類

如表 5-3 所示：

表 5-3 不同屬性的 Chunk 後面所跟的資料內容

Chunk 屬性	Chunk 內部資料
vertices	向量的 ID、 起始點座標、 結束點座標、 線條的顏色（單色）
face	平面的 ID、 組成向量的 ID、 頂點座標、 正反面的設定（系統中只會顯示正面的顏色，反面的話會看不見而成透明狀）
Material	貼圖的 ID、 所要貼圖平面的 ID、 貼圖圖檔的位置、 貼圖透明度、 貼圖的方式（將在下面做說明）

所謂的貼圖方式，是指 3ds 檔可以將要貼的圖片壓縮變形，將不同的部份以不同的比例貼在平面上。這些 3ds 檔的內部資料是一群的數值，代表要將整張圖分成幾份，之後還有一串座標數值表示這些分割點要分布在平面的哪些地方，如果平均分佈就會如圖 5-24 所示；如果集中在左半邊比較集中的話，會就如圖 5-25 所示。

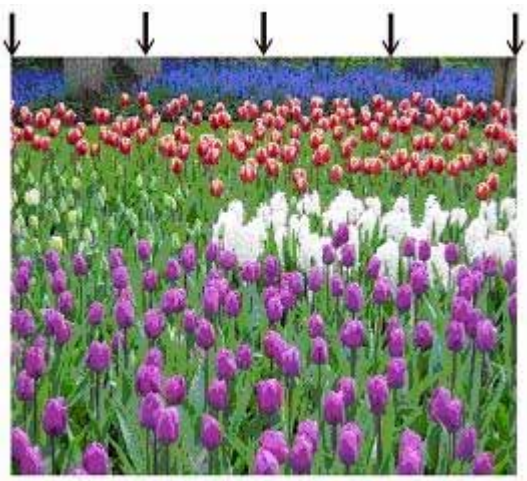

圖 5-24 平均分割的貼圖

圖 5-25 不平均分割的貼圖

整個 3ds 物件就是經由這一個個的 chunk 組合串接起來，合成一個完整的 3D 物件。而我們的系統則是利用分析 3ds 檔裡面的每一筆 chunk，並依照屬性分類放入不同的陣列中後，再對每一個陣列作不同的繪圖處理。

在 cLoad3DS 類別（讀入 3D 物件的 class）裡，處理個各個屬性 chunk 的函式如表 5-4 所示：

表 5-4 cLoad3DS 類別中的函式

cLoad3DS	
cLoad3DS ( )	初使化、建構子
cImport3DS ( )	開始讀入 3ds 檔的動作
cleanUp ( )	清除記憶體空間
ProcessNextChunk ( )	將下一筆 chunk 讀入記憶體裡
ReadChunk ( )	開始讀入 chunk 的動作
ProcessNextObjectChunk ( )	要讀入 chunk 之前，先判斷 chunk 內部資料的正確性與否
ReadVertices ( )	處理向量 chunk 的函式
ReadVertiexIndices ( )	處理平面 chunk 的函式
ReadObjectMaterial ( )	處理貼圖 chunk 的函式
ProcessNextMaterialChunk( )	讀取下一筆貼圖的相關資訊

由系統中繪圖系統裡的 LoadFile()函式呼叫 cImport3DS()之後，就會開始整個讀取 3ds 檔的動作。完成之後就會各有一組完整的資料，用 OpenGL 能夠存取的模式存在三個記錄 chunk 的陣列之中。然後就可以靠著 OpenGL 內建的繪圖函式把我們製作完成的 3d 物件一個個放在地圖上，如此一來就算是完成了載入 3d 物件的大工程。

5.2.5 特殊旗標

一、可走不可走的設定

前面已經介紹過「設定地面座標可否行走」的用意，現在詳細說明實作的方法。首先要用到先前已經做好的地形高度圖如圖 5-26，要利用到.raw 檔的特別性質（內部只有記錄像素的亮度數值）。

原理是先將地形高度圖的內部資料讀入陣列中，然後經由程式判斷每個相鄰像素之間的數值大小（也就是代表相鄰間隔的地形高度差），若是超過一定的範圍之外，就表示這兩點在地形上的相對高度太大，也就是坡度太陡。若是發生這種情形，則程式會將這些點之間的數值轉成「0」，其餘的則設為「1」，然後輸出成另一個新的.raw 圖檔，作為地圖判斷可否行走的參考依據。

因此，在完成之後的這張可否行走的設定圖中，只看的見黑跟白兩種顏色，如圖 5-27 所示。地圖系統在讀入這張設定圖的時候，會自動把白色的部份設定為不可行走，以限制玩家角色、電腦操縱角色可移動的範圍，使得遊戲進行時不會產生不合理的畫面（比如說人走進樹幹裡）。

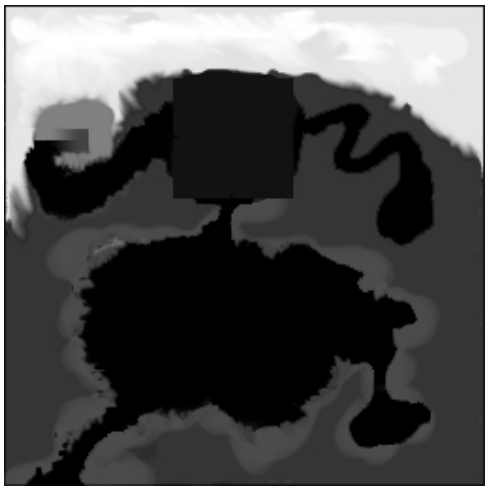

圖 5-26 地形高度圖

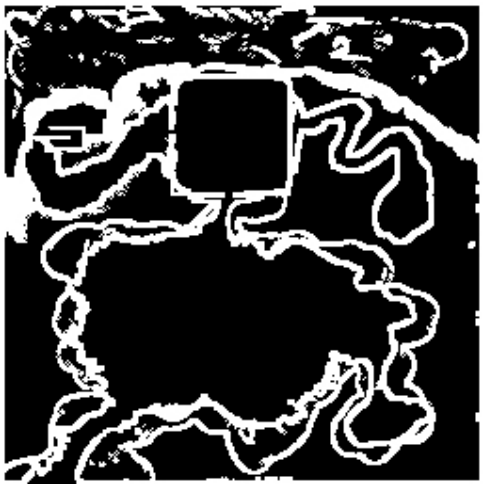

圖 5-27 由地形高度圖產生出的可否行走設定圖

另外一點要注意的是，產生「可否行走設定圖」的程式並不整合在系統中，也不會跟任何其他的程式有關聯。我們必須手動去生產設定圖，原因有二：

1)此程式在遊戲運作的途中完全沒有任何的作用，只有在一開始編輯地圖的時候會用到。

2)編輯出來的設定檔不一定可以馬上使用，有必要時還需要經過手動更改。例如若是地形當中有一座高台，依劇情需要玩家們必須走到高台上去，但是如果直接使用產生出來的設定檔，那麼這座高台很可能是到不了的。所以我們還要經由手動更改的方式，把高台的部份塗黑，才能夠讓遊戲順利進行。如果將此程式整合在系統中，不但沒有必要，而且還會讓手動更改的動作難以進行。

二、傳送點設定

其實地圖之間傳送的動作，是屬於劇情系統的管轄範圍。但是基本的設定，如傳送點座標位置及傳送點呈現出來的外觀，都是要在地圖編輯系統裡作設定。首先利用 3D 物件來顯示傳送點的外觀，讓玩家可以很清楚的看到傳送點所在的位置，如圖 5-28 所示。

圖 5-28 傳送點的外觀

安插好 3D 物件之後，同時把傳送點所在的座標紀錄下來，一旦觸發條件發生(玩家走到傳送點的座標)，就會呼叫劇情系統來處理，以引發觸發事件後的結果。由圖 5-29 及圖 5-30 就能明顯看出，進入傳點前和進入傳點後，傳送到另一張地圖動作。

圖 5-29 進入傳送點之前

圖 5-30 進入傳送點後，被傳送到另外一張地圖

遊戲中的整個世界，就是靠著一張張的小地圖拼湊起來的；而整個故事劇情的進行，也是靠著一張張地圖的轉換，來實現故事的流程。再者伺服端及用戶端之間的封包資料量，是以地圖為單位劃分，因此，地圖若是劃分的愈細，對整體的傳輸速率會有很大的幫助，但是劃分的太細又會影響遊戲進行的流暢感，所以如何規畫地圖的劃分也是不可忽視的一個重點，在一開始的地圖規劃時就要很仔細的考慮。

5.2.6 整合

所有的基本設定都完成之後，就可以把它們全部拼成一張完整的地圖了。我們在此使用一個收集地圖所有相關資訊的文件檔，如圖5-31，把該有的元件、資料全部紀錄在其中，之後系統只要從這個文件檔中獲得編輯地圖所需的資訊，就可以把一張完整的地圖呈現出來。

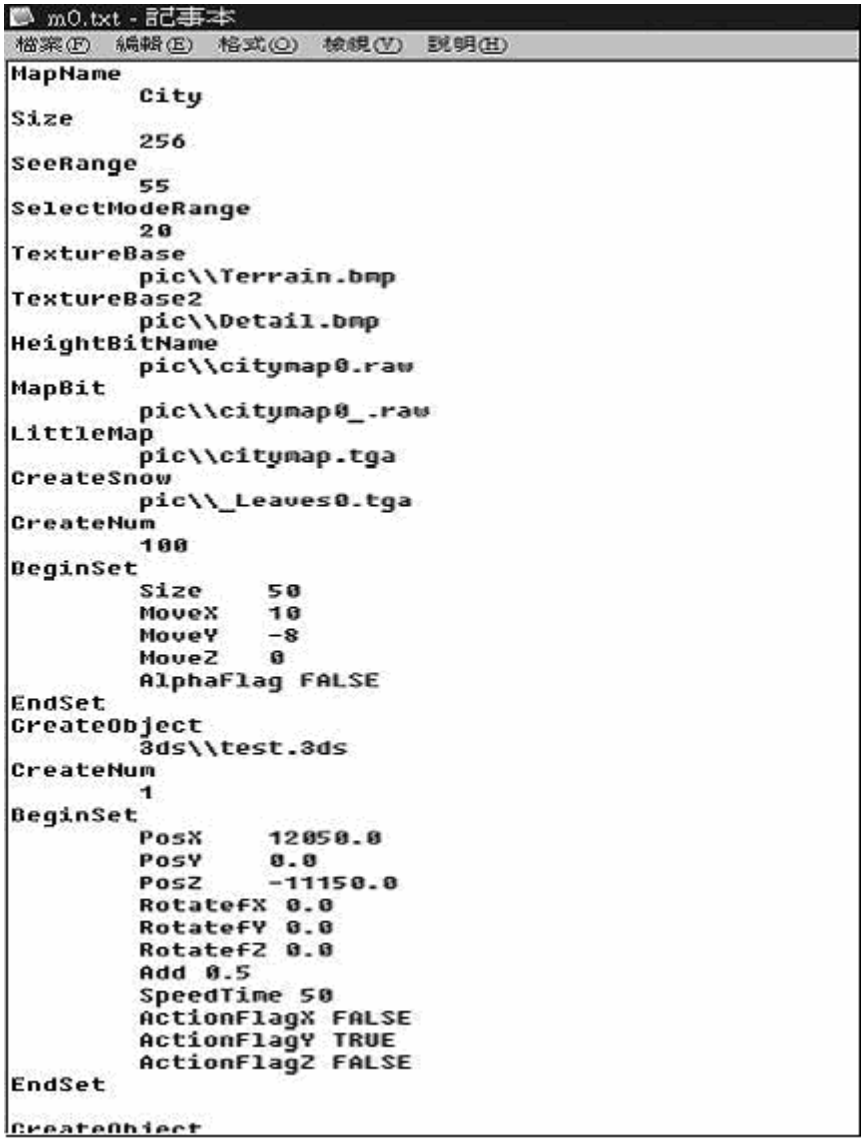

圖 5-31 地圖資訊文件檔

各欄位資訊分別為：

- Size

代表這張地圖的大小，256 就表示這張地圖地大小是 256*256 格的正方形。

- SeeRange

表示可視範圍。我們設定系統在繪圖的時候，只會畫出以主角為中心的某一個範圍內的畫面。這樣可以大量減少繪圖晶片記憶體的使用量，使的繪圖系統只需處理一部分的畫面，而不是任何時間都必須

處理整張地圖的所有顯示資訊。

- SelectModeRange

設定滑鼠可以點選到的移動最大範圍。雖然可以擴張到無限大，但是過大的可選取範圍，會讓系統在運算移動路徑的時候大大的增加工作量。因此可選取範圍的設定也必須恰到好處，有時候也需要照著劇情的需要調整此一數值的大小：例如在伸手不見五指的山洞中時，移動範圍可能只有 2~3 格。

- TextureBase

地面色彩分布圖。

- TextureBase2

地表材質貼圖，跟地面色彩分布圖重疊貼在地表上。

- HeightBitName

地形高度圖，紀錄地形高低的.raw 圖檔。

- MapBit

紀錄可走不可走的設定圖檔。

- LittleMap

顯示在畫面上的小地圖視窗，讓玩家知道現在自己在地圖中的位置。

- CreateSnow

氣候效果貼圖。下面的屬性分別是：

1. CreateNum：貼圖數量
2. BeginSet：開始設置
3. Size：貼圖大小
4. MoveX：X 方向位移
5. MoveY：Y 方向位移

6. MoveZ：Z 方向位移
7. AlphaFlag：文字貼圖旗標
8. EndSet：結束設置

- CreateObject

創造 3d 物件。下面的屬性分別是：

1. CreateNum：貼圖數量
2. BeginSet：開始設置
3. Size：貼圖大小
4. PosX：X 方向位置
5. PosY：Y 方向位置
6. PosZ：Z 方向位置
7. RotatefX：X 方向旋轉速度
8. RotatefY：Y 方向旋轉速度
9. RotatefZ：Z 方向旋轉速度
10. Add：旋轉位移量
11. SpeedTime：旋轉更新頻率
12. ActionFlagX：X 方向旋轉功能開啟旗標
13. ActionFlagY：Y 方向旋轉功能開啟旗標
14. ActionFlagZ：Z 方向旋轉功能開啟旗標
15. EndSet：結束設置

以上是地圖資訊文件檔的基本形式。整個地圖編輯系統最後最重要的動作，就是讀入地圖資訊文件檔，然後把我們精心設計的地圖完整的呈現在螢幕上。所有的地圖資訊、元件大致可分類為三個階層，如圖 5-32 所示：

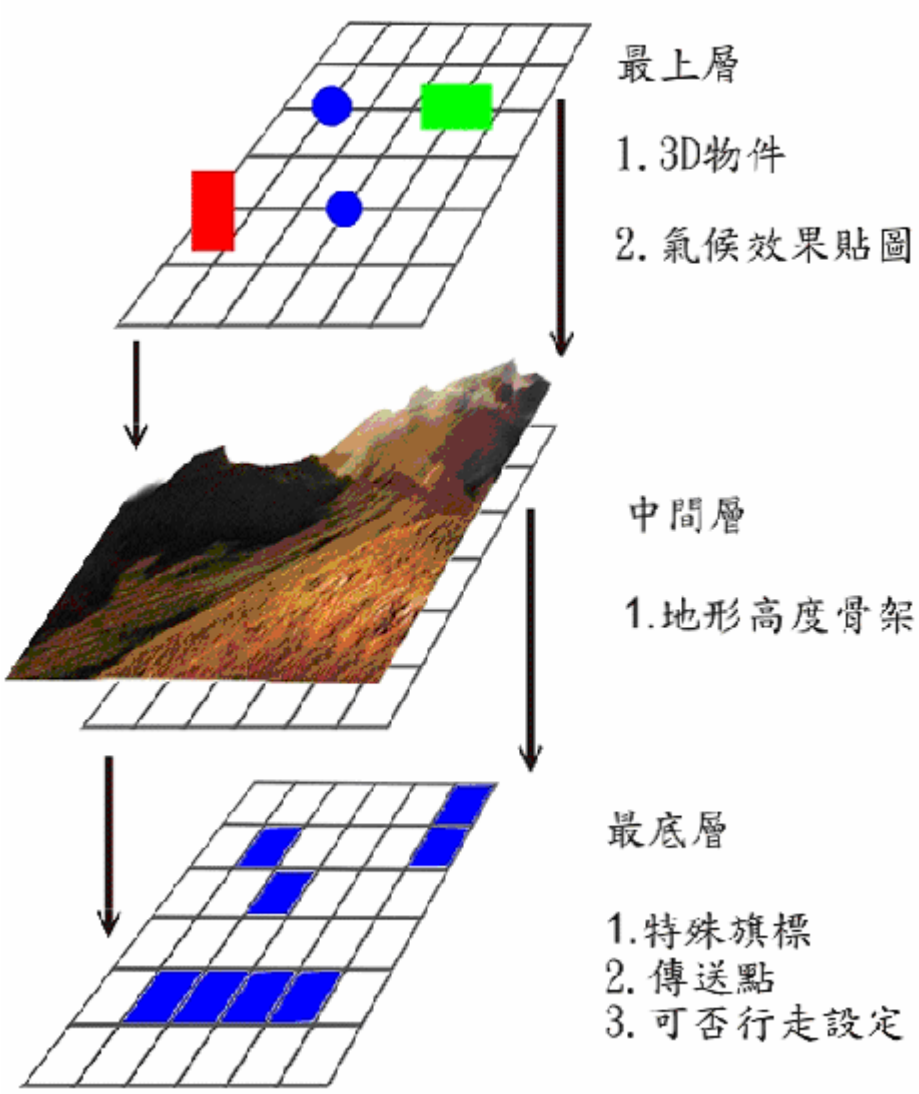

圖 5-32 地圖元件的三個層次

所有的元件全都合在一起之後，一張地圖就可以算是初步的大功告成。不過完成地圖之後還有一步很重要的動作，那就是先自我測試，看看地圖有沒有什麼玩起來不流暢的地方：比如說從某一點到另一點的距離太長，但卻又是劇情的必經之路等。

測試後確定沒有問題，便完成了一張完整的地圖，可再著手設計下一張地圖。等到所有的地圖皆完成後，再將它們用傳點全部串連起來，就可以構成整個遊戲世界上演的舞台，讓玩家們可以在裡面自由的活動及冒險了。

5.3 2D 人物動作貼圖

人物動作貼圖技巧，就類似動畫電影的技術一樣，連續播放動作連貫圖，如圖 5-33 所示，那是怪物往左走動的八張動作連貫圖，利用人的視覺暫留，就能產生流暢的走路動作。而光一個走路的動作有八個方向，每個方向又有八個分解動作所以總共就有 8×8＝64 張，然後就看這個怪總共會有幾個動作，如有站立、走路、攻擊、和倒下，再乘上動作數量，但站立的動作是例外的情況，站立只是完全靜止站著，所以只要 8 個方向的站立圖就夠了，這樣算出來怪物基本圖量就需 8+3×64=200 張。

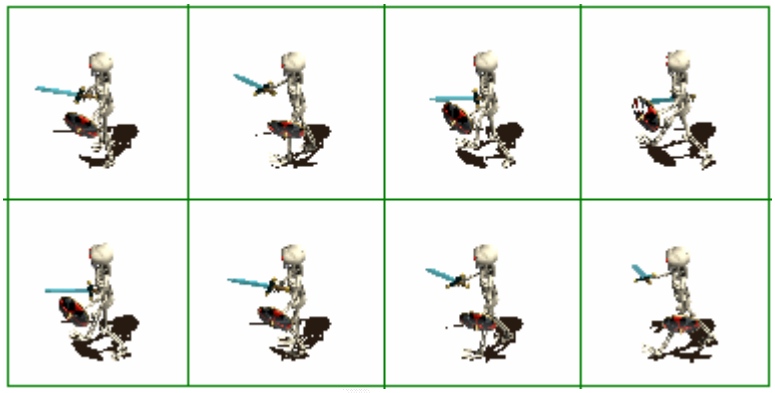

圖 5-33 8 個分解動作

貼圖的技術，就程式方面來說，我們使用 OpenGL 中的 glBegin 和 glEnd 這兩個成對的函式來指定多組座標點來組合 OpenGL 中的元件，OpenGL 的元件形態有很多種，如 GL_POINTS(用來畫點)、GL_LINES(用來畫線)、及 GL_TRIANGLES(用來畫三角形)等，基於現在的顯示卡都有對畫三角形元件來加速，所以我以兩個三角形組成一個正方形當框架來貼圖。

OpenGL 三角形的元件有一個進階的形態 GL_TRIANGLES_STRIP 可以用更少個點來畫出兩個三角形。以下為這兩者的差別，GL_TRIANGLES 必須指定六個才能畫出兩個三角形，而 GL_TRIANGLES_STRIP 的前三個點為第一個三角形，當再指定第四個點的時後，第四個就會跟前兩個點組成第二個三角形，這樣就明顯地

少了二個點，當三角形的個數一多，所少的點數也是很可觀的。

有了框架之後，就可以指定貼圖材質的範圍，而 glTexCoord2f 的函式就是用來指定。glTexCoord2f(s,t)有兩個參數 s 為材質水平座標；t 為材質垂直座標，其一張材質圖的範圍值以 0~1 為表示

做完前面的準備之後，再來就只要用 glBindTexture 的函式來指定該貼那一張圖，將上述的連續動作圖片依序貼出來，就完成了。

接下來，就會發現一個問題，之前有說明過一種怪物的動作圖的材質基本上就會有 200 張，如果一個地圖上 10 隻怪物同時出現，那就必須載入 2000 張，這樣所佔的記憶體空間，實在太大了，這就需要一個材質管理系統來掌管這些材質，讓同一種怪物只要載入一次圖庫就行了，其他的就共同。所以每當物件管理系統創造一隻怪物的時後，就必須先寫一張材質申請表給材質管理系統，材質管理系統就會先尋找之前是否已經申請過了，申請過就不會再載入，反之就載入。最後都在申請書個數上加一，整個材質管理系統流程圖如圖 5-34 所示。

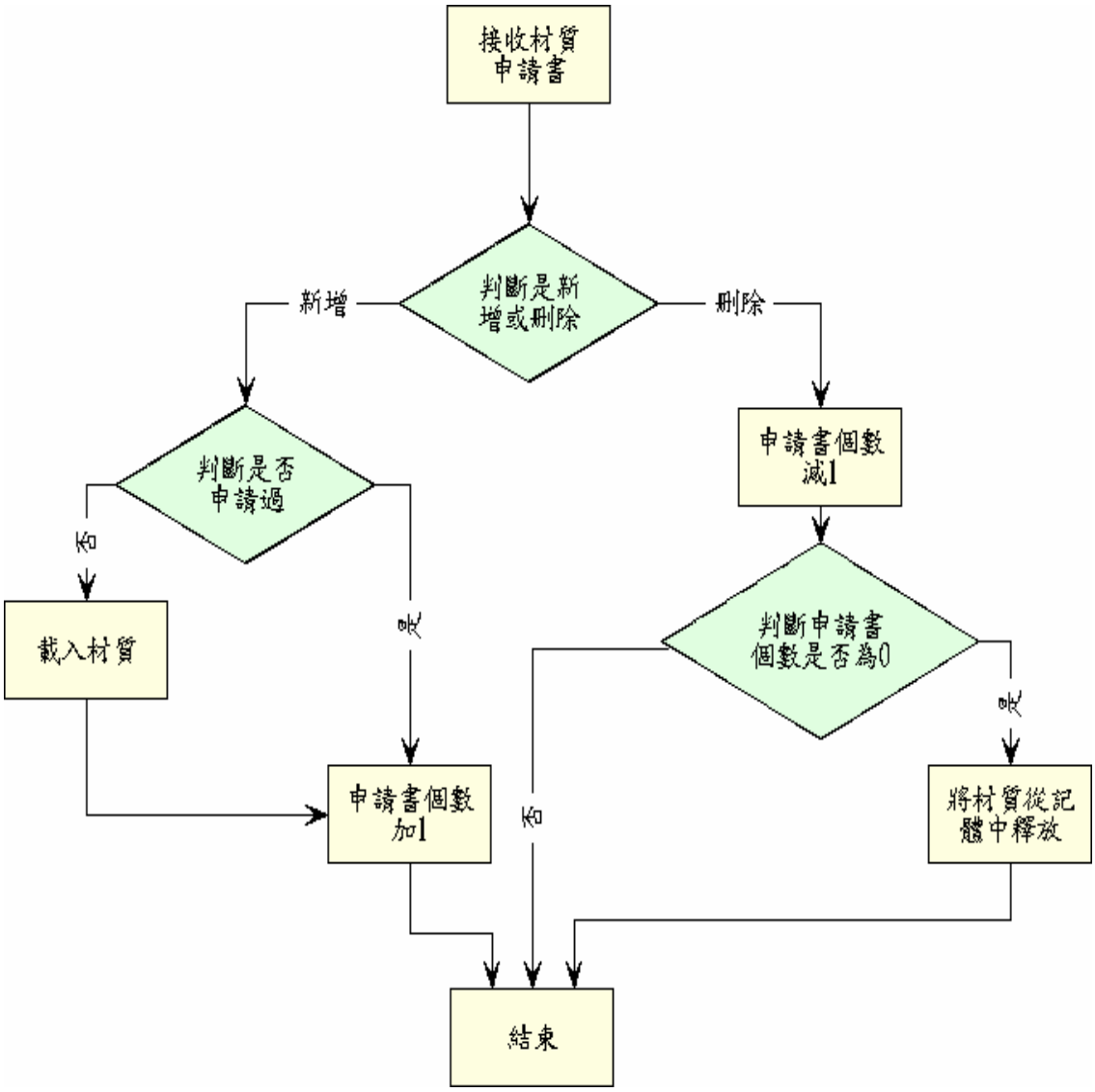

圖 5-34 材質管理系統流程圖

5.4 點選系統

點選系統重點是在於 3D 的環境上點選物件，可分為兩個部分。一個是地圖座標的點選，另一個是地圖上 2D 物件的點選。兩個都是運用同一種原理，但會有不太一樣的方式來實做。在 OpenGL 本身有提供一個功能 Selection，它能接收滑鼠的座標點，就能對應到場景的物件。而 Selection 實際上是一種繪圖模式，但是在這個模式下，不會將像素複製到 frame buffer(這個暫存區是為了在更換畫面時，將新的畫面存到這個暫存區，再直接對應到螢幕上，這樣就不會閃爍)，反而會被紀錄到 selection buffer 中。

為了能判斷出，使用者所點選的物件是什麼，就必須先為物件命名，以便進行 selection buffer 點選的時後，被分辨出來。所以命名方式將是整個點選系統的重點。處理物件點選的流程，可以抽象化的以下列步驟表示：

step1: 點選事件發生
step2: 進入 select mode
step3: 為每個物件設立編號
step4: 畫物件 （只需畫出可被點選的物件即可）
step5: 離開 select mode，並取得被選中物件的相關資訊
step6: 對被點選之物件做適當處理

點選的事件發生的情形，是當滑鼠按下左鍵的時後，就觸發了。雖然一般人再做點選的時後，按下左鍵的時間間距大概是 0.1 秒，但是以流暢的遊戲速度來說，FPS=60 張，跑一次迴圈只要 0.016667, 在按下左鍵的當中就跑了 6 次，所以選點事件的判斷間距會以 0.1 秒為單位，這樣可以降低進入點選模式的機率。

而地圖座標點選的物件命名方式，將以最直觀的方法以座標點來命名，以下以 4x4 的地圖大小為例，把地圖分割成 4x4 的小方塊，每一個方塊為一個物件，而每一個方塊就以它所在的座標位置命名。如

圖 5-35 所示。

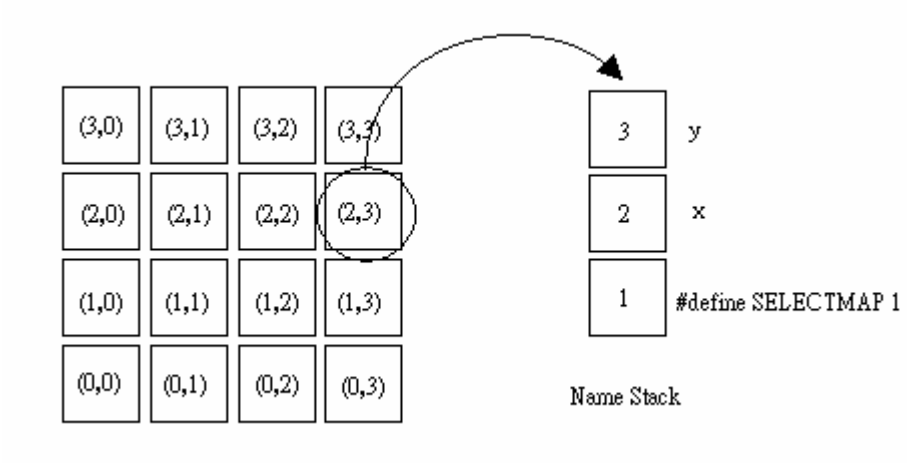

圖 5-35 物件命名方式—座標點命名

點選的命名，要經過 OpenGL 的 glPushName 函式將名字放入 Name Stack，首先要放入選點的種類，如地圖點選點就放 SELECTEDMAP，再來就是 x 座標，y 座標，這樣就完成地圖的命名。

2D 物件的命名，就比較簡單一點了，2D 物件的都是畫在方形的框架上，所以框架就是一個點選物件，而命名就直接用 2D 物件的 ID，如圖 5-36 所示，以後進行點選判斷的時後就方便多了。

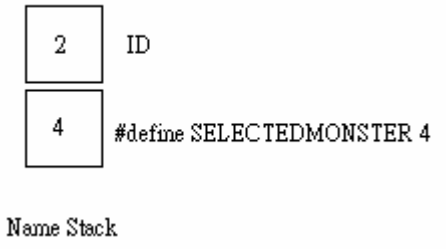

圖 5-36 2D 物件命名結構

命名之後，當點選事件觸發，就呼叫 glRenderMode(GL_SELECT) 進入點選繪圖模式，如同前面所說的，點選模式並不是真的把像素畫到 frame buffer，而是畫到 selection buffer 中，如果有選中物件就以上述的命名方法存到所指定的陣列裡，用 glSelectBuffer(BuffSize, selectBuff);這個函式去指定陣列。所以讀取陣列的值就知道點選到什麼，再經由動作分析系統，判斷出這要做出什麼動作，如玩家移動或是攻擊。

5.5 遊戲世界中各類角色的設定與行為能力

所有會出現在遊戲中的角色統稱為電腦角色，主要有以下三大類：

一、Person Character

玩家控制的角色，亦即透過用戶端的連線、登入後，玩家可依照其喜愛來選擇遊戲世界中，所要扮演的角色職業。以下介紹皆由玩家表示 PC(Person Character)。

二、Not Person Character

非玩家控制的角色，其行為全由伺服端負責處理。包括，螢幕上影像的呈現以及程式內部的資料處理。

三、Monster Character

怪物角色，其設計與 NPC 一樣，只是它們加入更多人工智慧，像是攻擊玩家的能力。較特別的是，除了玩家 PK(互相攻擊)外，怪物也是玩家可以攻擊的主要對像。也因為玩家須透過殺死怪物來取得經驗值、寶物、甚至完成任務等，所以遊戲設計者必須賦予怪物生命力，並妥善安排。以下介紹皆由怪物表示 MC。

以下各小節便依序介紹這些角色是如何設定、且其具有哪些行為能力。

5.5.1 玩家設定

一、基本狀態

1.玩家角色的能力

角色需要特殊的能力，才能影響遊戲進行時所發生特定動作的結果，像是戰鬥，如果角色揮動一把劍，它刺中目標的機會有多少，而且這把劍會造成多少的傷害？

在我們的遊戲設定中是以數值的方式來代表能力，如圖 5-37 為基本狀態類別圖，數值越高代表角色在這項能力上的表現越好。除此之外，使用數值表示能力也會讓計算更容易。舉例來說，若力量值的可能範圍是從 0(非常脆弱)到 999(超級英雄)，則一般人的力量值可能是 100 到 150 之間，則它的力量只能打開一扇門而不足以搬起一塊巨大岩石。

圖 5-37 基本狀態類別圖

以下便是我們遊戲中，角色所具有的各種能力：

- Strength(力量)：影響人物的攻擊力。
- Magic(魔法智慧)：影響人物的魔法攻擊力和魔法防禦力。也影響升級時，其 MP 提升的多寡與恢復 MP 的速度。
- Agilities(敏捷)：影響閃躲率和命中率的數值，也可能影響到攻擊速度。
- Mental(體力)：影響人物的防禦力。也影響到升級的時候，HP 提升的多寡與恢復 HP 的速度。
- Basic_ATK(基本攻擊力)：徒手攻擊、不使用任何武器時的攻擊力。
- Basic_DEF(基本防禦力)：天生的防禦能力、不穿戴任何防具。
- Attack(總攻擊力)：角色可造成傷害之總值。
- Defense(總防禦力)：角色受保護之程度即能減少之傷害總值。
- Basic_ToHit(基本命中率)：在不佩帶任何裝備下的數值。
- Basic_AGI(基本閃躲率)：在不佩帶任何裝備下的數值。
- ToHit(總命中率)：角色在攻擊過程中能夠命中目標的能力數值。
- Agility(總閃躲率)：角色被攻擊時能夠避開避免被傷害的數值。

2.玩家角色的屬性

角色的屬性與能力頗為類似，只是屬性定義的是角色的不同層面。例如：玩家的健康狀況就是一種屬性，它的變化根據的是角色身上受傷的狀況，這些受傷可以治癒，由此也會增加角色的健康程度。在我們的遊戲中預設了四個屬性，如下：

- HP：玩家健康狀況，數值越高表這個角色可以承受更多的傷害。
- MP：玩家魔力，施魔法所需減少之數值。
- Level：玩家基本等級。
- EXP：玩家升級所需經驗值。

3.對照表

在決定玩家與玩家間的差異性時,能力加屬性是最基本的判斷方式。隨著等級越高則玩家角色會越來越強壯、習得更多的戰鬥技巧、及可以使用更特別的武器等。下表便是玩家升級時,增加某項能力數值時所相對增強的項目對照表(表 5-5)。

表 5-5 能力效益對照表

增加之能力	效益/影響
Strength+1	Attack+1
Magic+1	MATK+1、MDEF+1
Magic+2	MP+1
Agilities+1	ToHit+1、Agilit+1
Mental+1	HP+2、Defense+1

二、玩家角色的動作

遊戲中的每個角色,都有一組可以執行的關聯動作,有了這些動作之後,相關的動畫才能在螢幕上播放。例如,揮動武器、發射魔法或與其他玩家聊天等,這些都是動作。

另外,遊戲中的每種動作也都會產生影響。走路的動作會讓角色移位,攻擊的動作則會揮動武器並決定被擊中的對象。在程式內部會產生一連串相對應的處理程序;而在外部,玩家會看到代表這項動作的圖形動畫效果。

以下便是在我們的遊戲中角色會出現的各項動作:

1.閒置

當角色靜止不動時，它們便處於閒置的狀態。主要有以下三種情形：

- 當玩家登入成功後未再下新的動作；
- 玩家離開螢幕前但並未真正下線；
- 玩家進行其它動作如聊天、交易，但並未移動。

此時，代表玩家的角色便會做出一些圖形動畫效果，如雙手來回擺動、及身體左右搖擺等等之類的動作，來表示閒置狀態。

2.移動

行走、飛翔、及奔跑，這些都是讓角色在遊戲世界中移動的方式。而在我們的遊戲世界裡，每個角色的移動方式主要是行走。

3.對話

角色間需要對話！角色間的對話也是角色扮演遊戲中的一個主要部份，在我們的遊戲世界中，主要有兩種對話模式：玩家與玩家間的對話及玩家與 NPC 間的對話。而藉由玩家跟玩家間的對話，能讓彼此直接產生互動以達特殊氣氛及效果。除此之外，藉由 NPC 與玩家的對話，遊戲設計者能充分將其理念表達出來，亦即使玩家能充分享受劇情。

4.戰鬥

在戰鬥的過程中，眩目的圖形表現與特殊效果是玩家所能直接感受到的。不過，在每次戰鬥的背後，都有一組規則(戰鬥規則)，用來決定角色在揮動武器及使用魔法等攻擊動作之後的狀態。之前曾經介紹角色的能力及屬性，這些便是用來表現戰鬥結果最好的素材。例如：玩家的 HP 一開始是 200，但經過一場廝殺後可能只剩 150，此時玩家角色的資料 HP 便會改變。而當 HP 降到 0，角色便會死亡，此時玩家所看到的就應該是角色從站立到不支倒地的連續貼圖。

除此之外，在角色扮演遊戲中，角色可以增長它們的經驗。在它們贏得的每次戰鬥，都會增加它們的能力。角色擁有所謂的經驗點數與經驗等級，每次戰鬥都會增加角色的經驗點數，在特定的經驗點數範圍內，角色的經驗等級也會隨著提升。當等級提升的時候，角色會獲得利益，這通常指的是力量、魔法、與技能的增加。在本遊戲中，經驗值的來源主要是殺死怪物。

5.撿丟物品

當怪物被殺死後，玩家通常會從他們身上獲得金錢與寶物的獎賞，而這些會在地圖上出現的物品就是玩家的角色所能撿起的。藉由滑鼠點選，我們可以輕易的把物品撿起並放到背包中，但這之中往往會有些許限制。就像，背包的空間必須足夠及第一個打怪的玩家會有第一優先權來撿寶等。

既然能撿，相對地，玩家也能將角色身上不需要的物品丟到地圖上，除了那些被限制成不能丟棄的特殊物品之外，連金錢亦能丟棄。而你不要的東西，別的玩家也能撿起。除非真的很普通到沒人想要，則在經過一段固定時間後，物品會從地圖上消失。

6.買賣物品

新手玩家要怎麼玩呢？一般遊戲會提供他一些最基本的裝備，然後玩家便得依 NPC 的引導，到城外去找較弱的怪物下手，以賺取金錢跟經驗值。而有了足夠的金錢後，便能跟城裡的商人進行交易。到了遊戲中期，玩家除了跟商人外，也能跟其他玩家交易，在這之間更多了互動的快樂。

7.使用物品

經由交易或路邊撿起所得來的物品要幹麻？當然，就是要使用。物品又可分為裝備、補給物跟特殊物品等，藉由這些不同屬性物品的使用，會帶來一些不同的遊戲效果。

8.登入

若玩家是第一次登入系統，則須先經過註冊的程序，來將其資料建檔。若玩家並非第一次，則當他每次進入遊戲時，伺服器都應該依其資料(通常是帳號密碼)，正確地將玩家的角色資料、上一次登出前的狀態等，傳送到用戶端，並完整的呈現出來。

9. 登出

事實上，用戶端不論是正常或不正常地結束連線，都稱為登出。當玩家登出後，系統都應該將其資料、狀態，完整備份起來，以供下次登入初始化使用。

經由上述的定義後，我們便知道當玩家發生各種動作時，在伺服端系統內部需建構戰鬥、交易、及倉庫等子系統，以處理各自的行為。而這些子系統會在往後的小節見到。

三、最短路徑計算

為了讓玩家、怪物、及 NPC 在移動時，有一定的路徑(座標的一連串集合)可循，我們需要作最短路徑之計算。

如圖 5-38 所示，在計算最短路徑時會有以下五道程序：

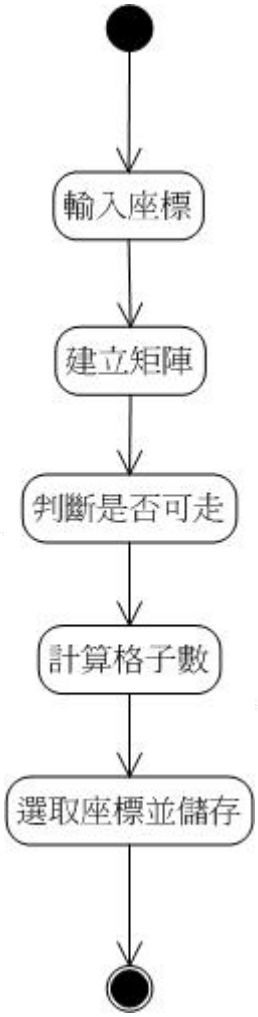

圖 5-38 最短路徑計算程序

1 輸入座標

輸入角色目前所處位置座標以及滑鼠所點選之位置座標。

2 建立矩陣

建立跟地圖相對應大小之矩陣。

3 判斷是否可走

將剛建立完成之矩陣內所有元素初值化，程式流程如圖 5-40。

4 計算格子數

以起始座標為基準點，將矩陣內所有元素與基準點間的距離算出，程式流程如圖 5-41。

5 選取座標並儲存

從目的地座標開始選取所要移動的路徑。

其中，3→4→5 之範例如圖 5-39 所示。

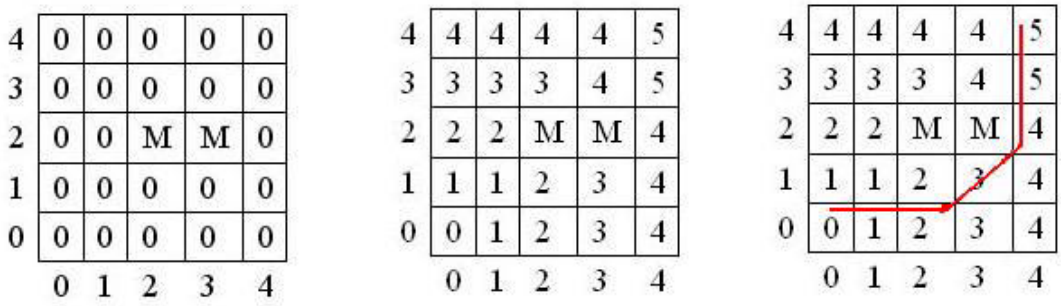

圖 5-39 範例(3→4→5)

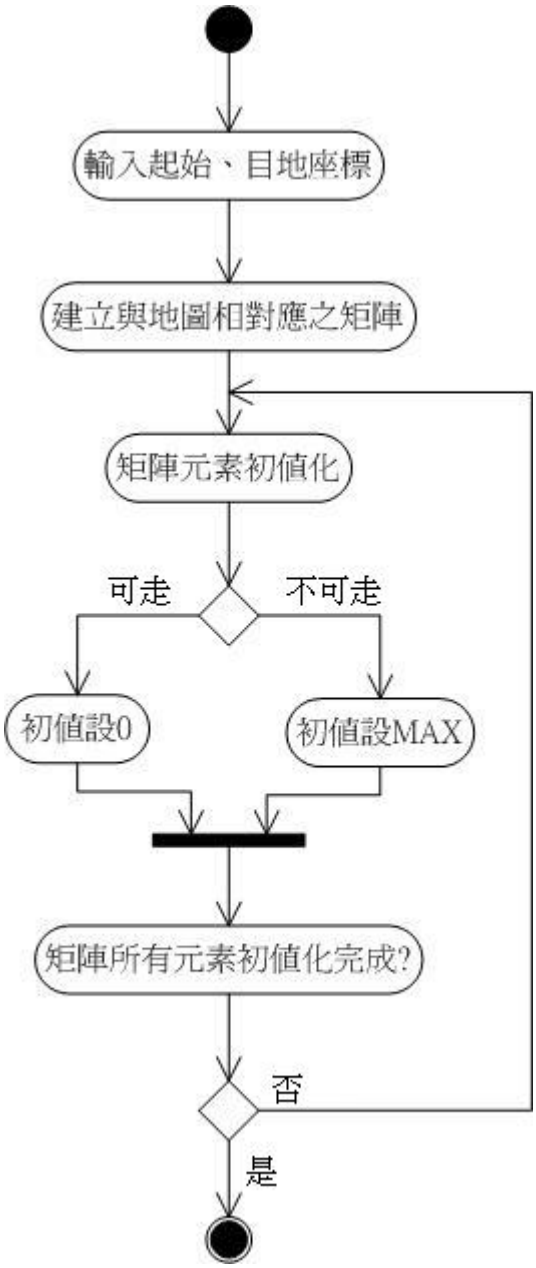

圖 5-40 可走不可走設定程序

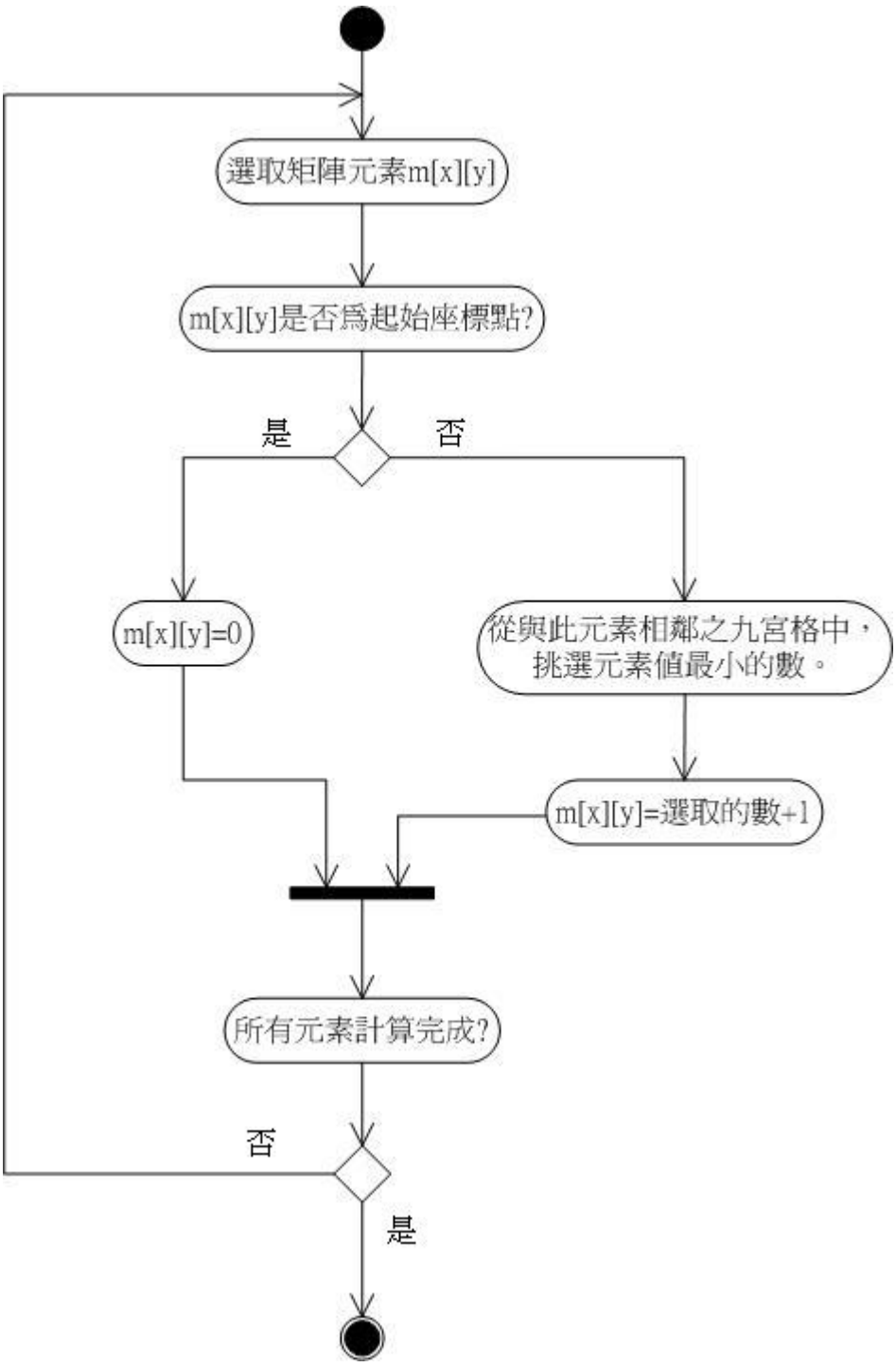

圖 5-41 格子數計算程序

5.5.2 怪物設定

在戰鬥的過程中，除了玩家之外，最主要的就是怪物，因此，怪物的設定一點都不能馬虎。怪物間強弱的差距及在地圖上區域性的安放等，這些動作不但關係到整個遊戲的流暢度，也關係到遊戲的平衡感。太過簡單往往會令玩家覺得無趣，太過困難又會使人打退堂鼓。

產生怪物物件時，可透過其建構式(參考圖 5-42)來設定其基本狀態。介紹如下：

- 攜帶金錢 怪物被玩家擊敗後，可能會掉出來的金錢數量(玩家能獲得的)。
- 攻擊力 怪物能傷害玩家的基本數值。
- 防禦力 怪物躲避玩家傷害的基本數值。
- 血量 健康狀況。
- 命中率 是否容易命中玩家。
- 閃躲率 是否容易躲過玩家的攻擊。
- 經驗值 怪物受到傷害後，玩家所能獲得的經驗值。
- 掉寶機率 機率高低會影響是否掉寶(玩家能獲得的)。
- 會掉的寶物 每隻怪物身上基本都會攜帶一件物品(寶物)，至於會不會掉出來，則要視掉寶機率以及掉寶公式計算過後的結果來判斷。

圖 5-42 怪物類別

經過以上初始設定，在搭配怪物提供的行為函式(5.5.3 節)，便能限定相同類型(種族)的怪物在地圖上某些特定的區域出沒。而玩家也能隨著任務、等級或所處地圖位置的不同，來跟不同的怪物纏鬥以獲得不同的結果。

5.5.3 NPC+怪物的移動控制

由於 NPC 及怪物都不是由玩家直接控制，但卻是遊戲中跟玩家互動最重要的兩大部份。NPC 如村民、城門守衛、商人…等，常常扮演引導玩家進行遊戲的角色，更是遊戲設計者與玩家之間訊息傳遞最重要的橋樑；而怪物所扮演的角色就非常明顯，不再贅述。因此，所有跟 NPC 及怪物相關的控制作業都必須由程式開發者負責撰寫，在伺服端執行運作及管理。

NPC 及怪物的共同部分便是移動，有以下三種基本模式：

一、靜止不動：放置某個角色並使它面對預設的方向靜止不動。

二、漫無目的地的在固定區域(座標範圍)內游走

程式流程如圖 5-43 所示。先藉由設定 NPC 或怪物物件裡的行為函式 LocMoving()開始，此函式共需要 4 個變數來設定範圍，分別代表最小、最大的 x 座標以及最小、最大的 y 座標。當範圍設定好後，便亂數產生一組(x,y)座標，並跟一開始設定的座標作比對，以檢查是否超出可移動之範圍。若比對的結果是超出範圍，則回到亂數產生座標的程序在重新往下作(此時怪物是不會移動的)。反之，若比對結果仍在範圍內，則計算最短路徑並將結果存下以作為接下來逐步移動的依據。

移動的過程中，若無發生其它事件(如：被玩家攻擊)便會一直往前走，待到達目的地座標後，便會再回到亂數產生座標的動作再繼續下去。但移動期間，若遭受玩家攻擊，則怪物便會停下腳步，予以回擊。

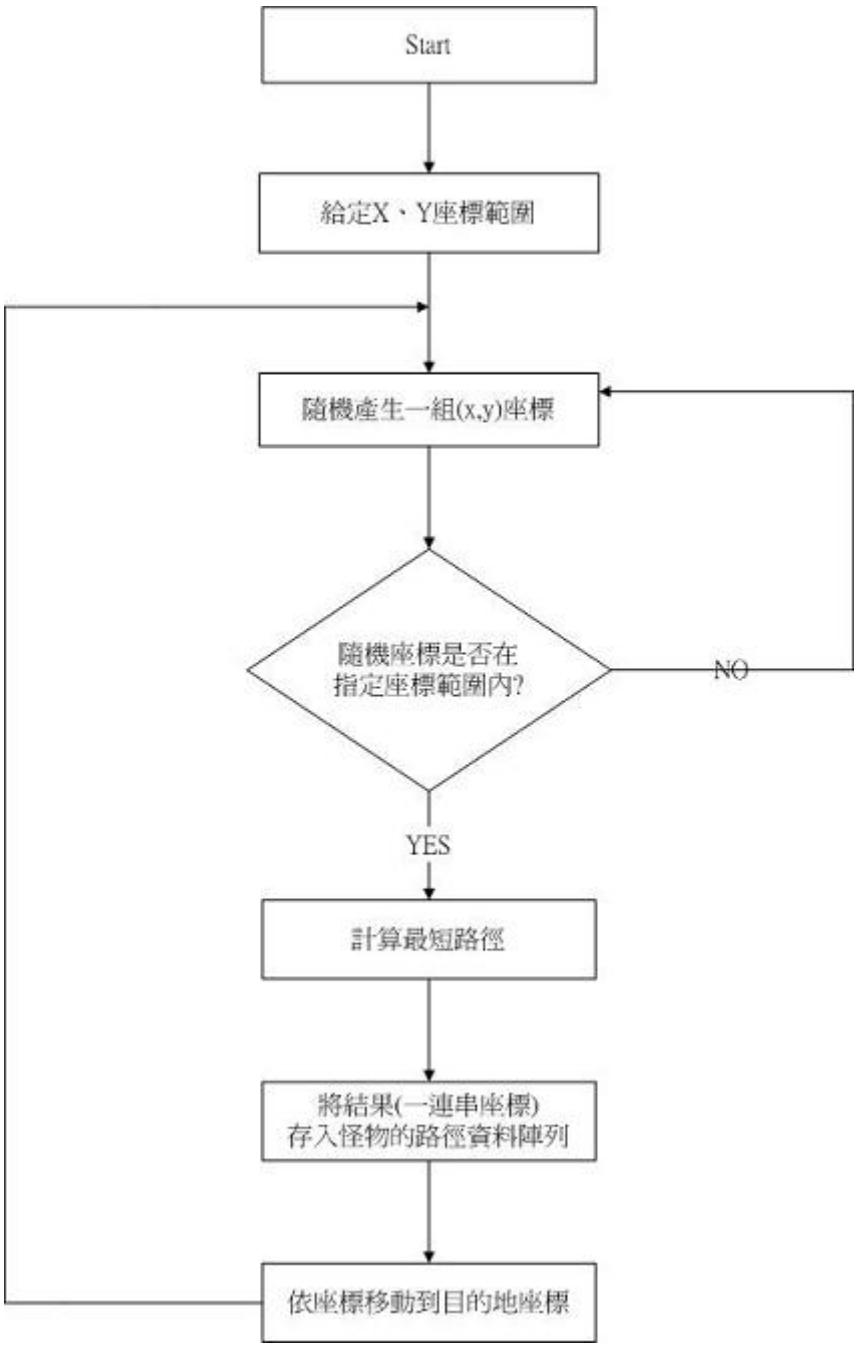

圖 5-43 隨機游走程序圖

三、在特定的路線上走動

雖然 NPC 及怪物並沒有那麼聰明，知道該在地圖中的哪些路徑上移動，不過我們可以透過函式來指定它們所要行走的路線。這些路線是一連串座標的集合，我們必須保證當在它們抵達某一座標後，才能再往下一組座標前進。且一但到達指定的最後一組座標，它們需回頭

走向起始座標再重複相同路線。主要有以下兩種模式：

1)在兩座標間來回走動

透過NPC或怪物物件裡行為函式Setpath()的呼叫，我們可以指定一條路線來讓NPC或怪物在上面來回走動。此函式需要4個變數，(Xstart，Ystart)、(Xfinal，Yfinal)分別代表一組起始座標及一組目的地座標。

當設定好之後，程式內部就會呼叫計算最短路徑的函式予以計算，並將結果放在NPC或怪物的資料欄位walkroutine[]內。如此主程式便可依照陣列內的座標逐步移動。在移動的過程中，需將下一步座標與目的地座標作比對，若不是目的地座標則繼續走。若是，則走到下一步之後便要開始往回走。程式流程如圖5-44所示。

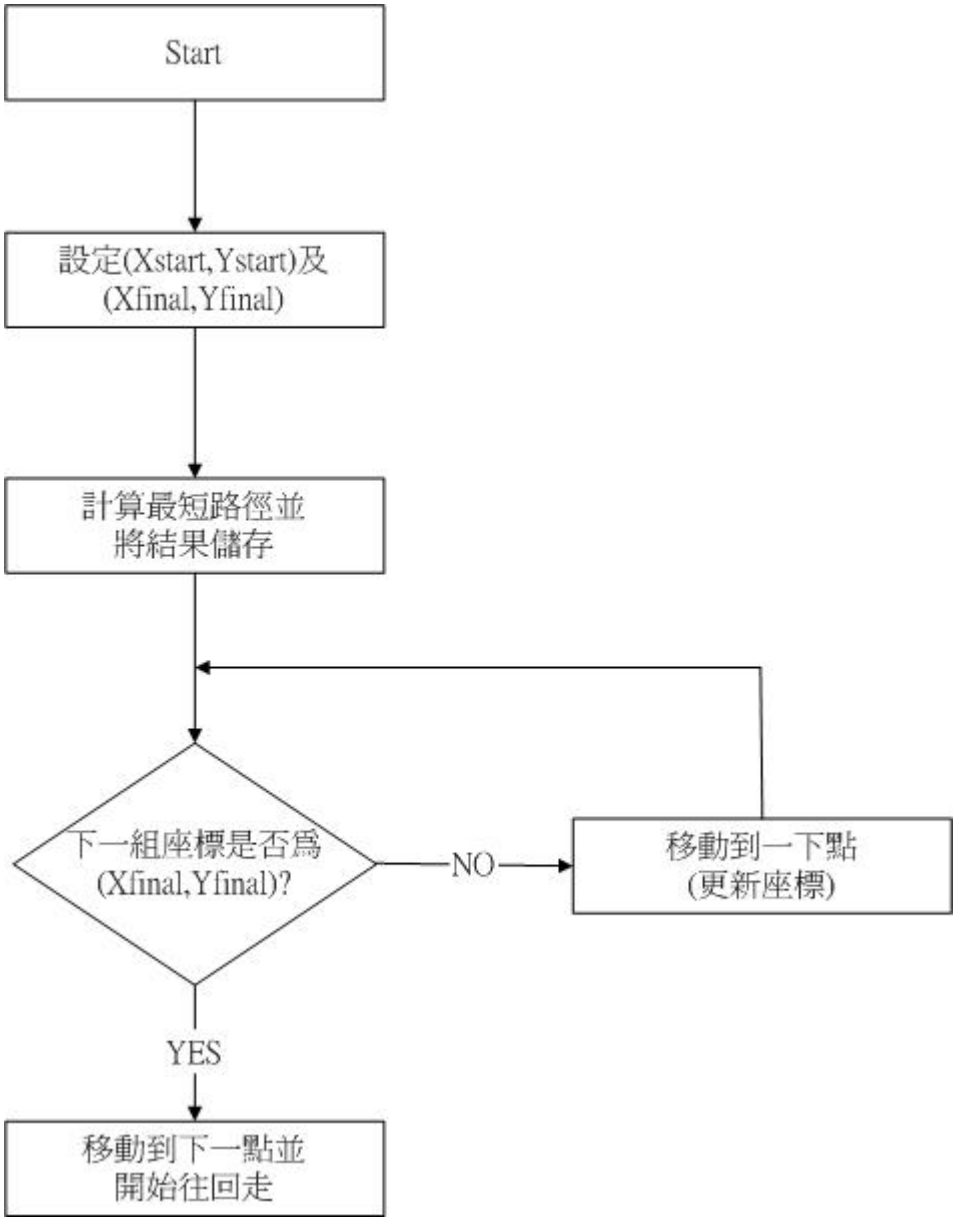

圖 5-44 兩座標間來回走動程序圖

2)在給定的數個座標間繞圈圈

事實上，這個控制模式是前者的應用。在圖 5-45 中，NPC 可以繞這四個圓點來依序移動，每兩個圓點的座標就是 Setpath()的參數值。甚至，我們也可讓 NPC 繞著整間屋子移動，也是可行的。

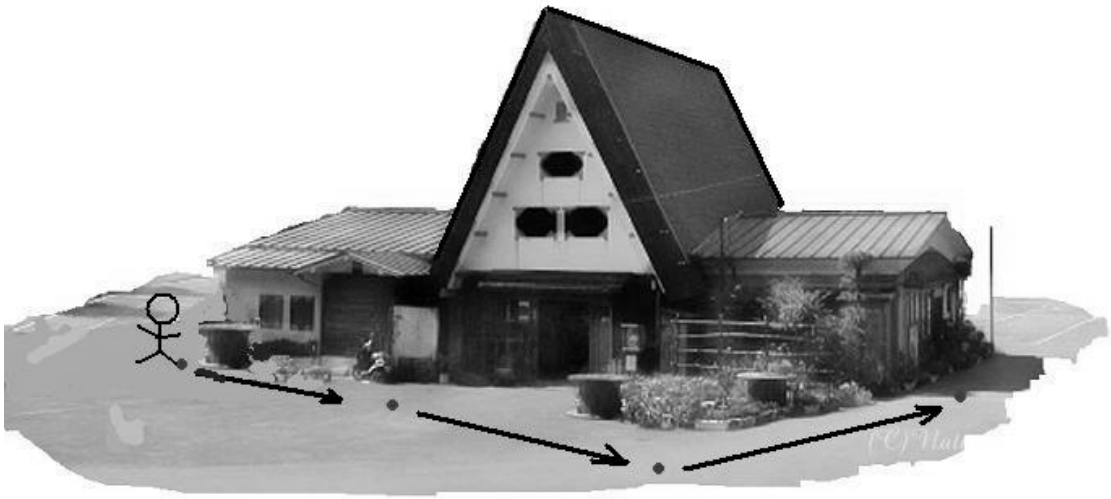

圖 5-45 繞圈圈範例

5.6 戰鬥系統

遊戲會根據一套基本的規則，來處理所有與戰鬥相關的事宜。因此，當遊戲設計者擬定好原則後，程式撰寫者便依規格來撰寫。本系統中，所有跟戰鬥相關的一連串處理步驟如以下幾節所示。

5.6.1 攻擊

在攻擊的過程中，玩家或許有使用武器、或許沒有(徒手攻擊)，但這都沒關係，在程式內部處理看起來都一樣，只是攻擊數值有所差距罷了。當玩家對另一個角色揮動武器時，這個動作會觸發一項程序，判斷它是否擊中目標。在這之中，便牽扯到攻擊者的命中能力及被攻擊者的閃躲能力。而須遵守的原則是，能力數值越高則命中或閃躲的機會便越大。

怪物被玩家攻擊時會引起特定的行為函式 beAttack()，在此函式中主要有兩個判斷依據：

一、 HitFlag = (玩家命中率 >= rand()%100)?1:0;
二、 DodgeFlag = (怪物閃躲率 >= rand()%100)?1:0;

其中，HitFlag 表玩家是否命中的旗標，它的值可能是 0(False)或 1(True)。當第一個判斷程序結束並確定玩家有命中之後，才會再進入第二個判斷程序繼續處理，否則就不做任何動作。若玩家命中(第一個程序判斷完成)，此時並還沒確定怪物已經受到傷害，接下來會進入到第二個判斷程序，其中，DodgeFlag 表怪物是否閃避的旗標，它的值亦可能是 0(False)或 1(True)。當 DodgeFlag 旗標的值也為 1(True)，這時，我們才可說怪物即將受到玩家所帶來的傷害，並進入傷害計算的階段。

相對的，當玩家被怪物攻擊時也會引起玩家物件內相同函式的運作。差別是，玩家的命中、閃躲能力會因外在的因素，如等級提升、佩帶裝備等，而有所提升。但同一種類的怪物，它們的能力一開始就設定好，不會再有所變動。

玩家的總命中率及總閃躲率可透過 cToHit()、cAgility()函式的呼叫來計算。其中，

- 總命中率=基本命中率+佩帶裝備(如項鍊)所附加之數值。
- 總閃躲率=基本閃躲率+防具附加之數值。
- 基本命中率=玩家 Agilities 的數值(隨點數增加而增加)。
- 基本閃躲率=玩家 Agilities 的數值(隨點數增加而增加)。

5.6.2 傷害計算

當判斷結果表示攻擊已經命中目標的時候，則要計算傷害的程度有多少。計算的規則是：攻擊者的總攻擊力再減掉被攻擊者的總防禦力，如此得到的數值便是被攻擊者的損血量。玩家的總攻擊力及總防禦力可透過 cAttack()、cDefense()函式的呼叫來計算，其中，

- 總攻擊力=基本攻擊力(徒手)+武器攻擊力+使用攻擊性物品。
- 總防禦力=基本防禦力+防具防禦力。
- 基本攻擊力=Strength+(Agilities/5)*2。
- 基本防禦力=玩家 Mental 的數值。

5.6.3 經驗值取得與升級判斷

因為我們的系統是以多人連線為主，故在打怪練等級的過程中，不一定是一個玩家獨自攻擊一隻怪，有時候可能是兩個人(或更多)來聯手出擊。除此之外，有時候更會出現搶怪的情形，因此，誰是第一個攻擊者便格外重要。在怪物的資料欄位有一項 FirstBeATK_Id，

此欄位會紀錄著第一個攻擊它的玩家的編號。若此時發生搶怪的情形時，第一個攻擊它的玩家就會得到多一點的經驗值。

透過怪物的 cExpRate()函式，我們可以把經驗值比例先計算出來，再依照攻擊者所給予的傷害程度來將經驗值分配給他。其中，

Exp_Rate=(sCharDef.EXP)/(sCharDef.HP)
(經驗值比例=怪物擁有經驗值/怪物血量)

因此，根據損血量的多寡再乘上經驗值比例，便可換算成玩家實際上獲得的經驗值，而此經驗值更是玩家的角色是否能升級的主要依據。如表 5-6 所示，當我的角色等級為 2 時，想升到等級 3 則必須再取得 5 經驗值。當玩家獲得經驗值後，在 GetEXP()函式內，會先把玩家升級所需經驗值扣掉新獲得的經驗值(更新玩家資料)，然後再呼叫函式 cLvUp()來判斷升級與否。

表 5-6 前 30 級升級所需經驗值對照表

LV	EXP	LV	EXP	LV	EXP	LV	EXP	LV	EXP
1	0	2	2	3	5	4	20	5	30
6	85	7	120	8	170	9	220	10	300
11	568	12	630	13	697	14	756	15	900
16	1345	17	1615	18	1975	19	2200	20	2535
21	3000	22	3545	23	4120	24	4670	25	5250
26	6200	27	7125	28	8020	29	8975	30	9900

怪物每被玩家攻擊一次，就會檢查它的 HP(健康狀況)扣掉損血量後是否降到零(或以下)。若尚未降到零，則表怪物仍有還擊能力，此時程式內部只要將經驗值計算完並傳給玩家更新資料即可。反之，則表怪物已經死亡，此時在螢幕上除了把代表怪物死亡的連續貼圖展示出來之外，在程式內部計算完經驗值之後，怪物物件會呼叫 Droping()函式，來決定是否會掉寶物，其中共有五種情形，掉武器、

掉防具、掉飾品、掉金錢或什麼都不掉。

5.6.4 提升基本能力

戰鬥除了能獲得寶物及金錢之外，另一項更吸引人的就是升級。升級就像是小孩逐步長大一樣，體格不但越來越健壯、所會的技能也越來越多。如表 5-7 所示，玩家的角色每升一個等級便會獲得相對的點數，來讓他們選擇要加在哪一方面。

而遊戲中提供力量(Strength)、魔法(Magic)、敏捷(Agilities)、及體力(Mental)四大類來讓玩家選擇。在程式內部，他們分別藉由玩家物件函式 cAddStrength()、cAddMagic()、cAddAgilities()、及 cAddMental()來處理，增加某項能力(數值)會帶來的效益如之前表 5-5 所示。

為什麼要提供這項功能呢？在我們的遊戲世界裡，有戰士、法師、及弓箭手等許多角色，能讓玩家盡情地享受角色扮演遊戲的樂趣。如果你所選擇的角色未來是要當個力大無窮的戰士，那力量(Strength)這項數值對你而言可能就比其它能力來的重要;若你想當個法師,為了加強你所施放魔法的攻擊力時,適當的增加魔法(Magic)數值，可能就是必須的了。

表 5-7 升級獲得點數對照表

LV	得到的點數	LV	得到的點數	LV	得到的點數	LV	得到的點數
01~05	3	06~10	4	11~15	5	16~20	6
21~25	7	26~30	8	31~35	9	36~40	10
41~45	11	46~50	12	51~55	13	56~60	14
61~65	15	66~70	16	71~75	17	76~80	18
81~85	19	86~90	20	91~95	21	96~	22

在處理戰鬥系統的內部行為，可由圖 5-46 明顯看出之間所需處理行為之間的關係。

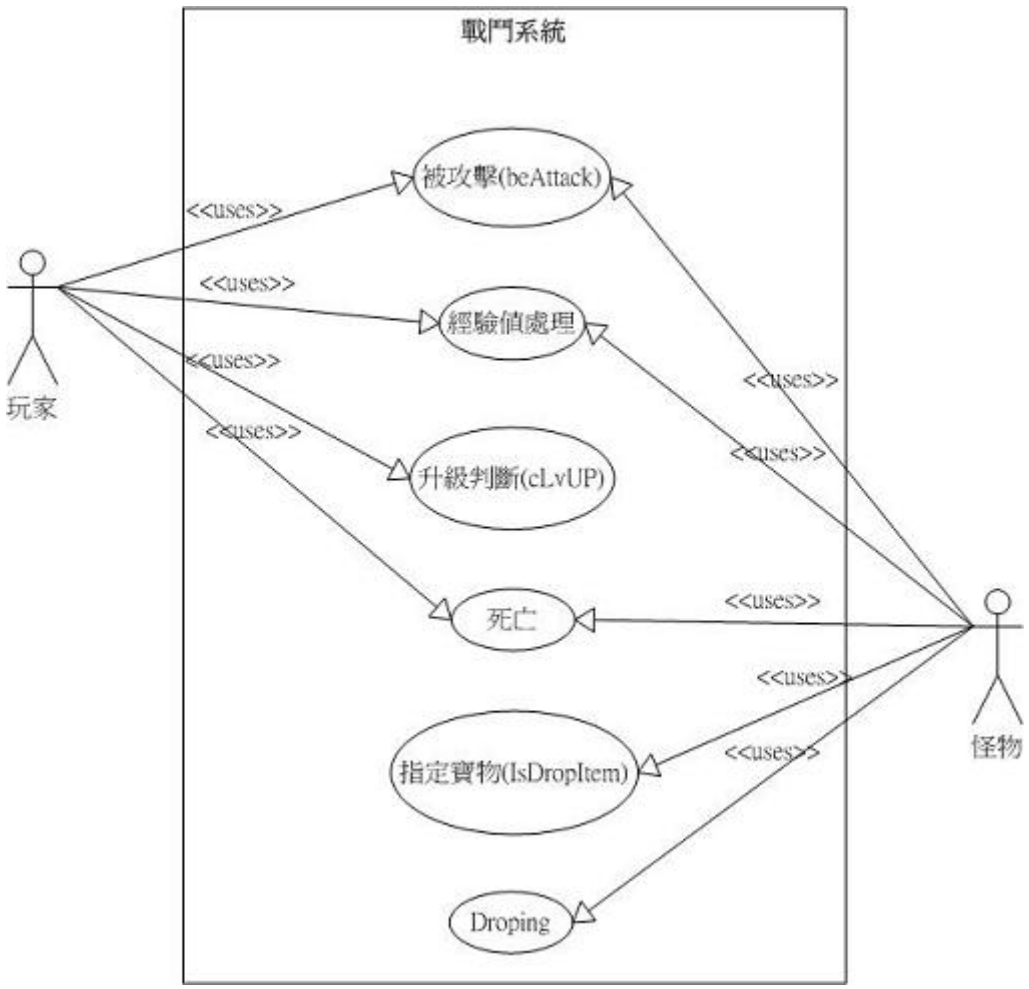

圖 5-46 戰鬥系統使用案例

5.7 交易系統

這個世界上充滿了各式各樣的東西，有大有小、有圓有扁，相同的，在我們的遊戲世界裡，也該有這些東西，並能讓玩家的角色實際佩帶使用。本節除了介紹交易系統外，還會介紹我們遊戲世界中的物品是如何定義產生，以及所有跟物品相關之動作，如玩家使用武器。

5.7.1 定義物品的方法

在遊戲的過程中，玩家一定會使用某些物品(或稱物件)，不管是打怪所得到或向商人買的。故當製作遊戲時，每個物品都須經過定義，也都需要有特定的用途。武器、盔甲、及飾品甚至治療用的物品等，所有會在遊戲中出現的物品(統稱)都要有適當的定義，它包含了形式與功能的說明。形式指的是外觀長相及給人的感覺，而功能則指它的功用，像錢可用來交易及刀劍可用來攻擊。

一、定義物品形式

遊戲中各個物品都有其專屬的 2D 貼圖。對玩家而言，這些貼圖便是用來表示物品最好的呈現方式。

二、定義物品功能

遊戲中的物品會有特殊的作用，以下便是我們遊戲世界裡物品的分類：

1)WEAPON（武器）

人物可以徒手造成傷害，依照遊戲中的設定，人物的力量(Strength)越大所能造成的傷害也越厲害。如果將某項武器交給這個角色的話，他所造成的傷害值亦會增加。而武器也有強弱之分，所以拿到更好更完美的武器，也是讓玩家持續上線進行遊戲的原因之一。

在我們的遊戲世界裡，會出現的武器分類大約有刀、劍、法杖及

弓類。當然，並不是所有的角色都能佩帶使用任何武器，其中也存在著某些限制，像是職業的限制，如角色是個法師，那他就應該要使用法杖來施放魔法，如果硬要配刀帶劍，那麼他可能連一隻等級比他低的小妖怪都砍不死了。除了職業之外，武器可能還有大小重量的限制、使用等級的限制等，就看設計者如何設計。

另外，武器也可因對付怪物種類的不同，而歸類為不同的群組。對於特定的對象，某些武器比較容易造成它們的傷害，例如，在對付雪地裡的怪物時，使用具有火屬性的劍似乎比普通的劍有利。這些都是在設計武器時，應該考慮的地方。

2)SHIELD（防具）

在戰鬥的過程，防禦力的高低往往是決定勝利者的重要依據之一。就像武器一樣，防具也有使用上的限制。另外，除了衣服，遊戲中還會出現帽子、頭飾、及褲子等，這些也都是防具類。

3)DIAMOND（飾品）

如項鍊及戒指等，這些就是飾品。飾品有其獨特且特殊的功能，有些佩帶了之後會提升命中率、有些則會提升閃躲率。在別的線上遊戲中(如 RO)，飾品帶上之後，會在螢幕上顯示出不一樣的造型，讓玩家享受化裝舞會中爭奇鬥豔的快感，這些也是線上遊戲吸引人的地方。

4)FOOD（食物）

如其名，我們將遊戲中使用一次就會消失的物品統稱為食物。依其功能又可細分為治療類(如紅色藥水)、攻擊類(如弓箭)、及貨幣類(如金錢)。

5)SPECIALLY（特殊用途的物品）

在遊戲劇情進行時，玩家可能要完成某些任務才能再開啟下一個支線劇情，而這個某項任務可能是打敗大魔王並取回他身上所佩帶的

寶劍，並交給特定的警衛後才能完成。則此任務中，最終要取回的寶劍便是遊戲世界中的特殊物品。特殊物品有其特定的限制，它可能是無法使用或無法販賣的。

遊戲引擎會根據每個物品的分類(吾人所定義的)，決定它的作用。在我們的遊戲中要產生一個物品(如一把劍)，我們可以透過『sItem Cutter ("卡特短劍","以前有位卡特的旅行者為了方便而自製的武器。攻+40",0,All,WEAPON,40,200,YES,YES,NO,NO,1);』中的宣告來產生，

其中資料欄位依序代表：

- Name：物品名稱。
- Description：物品相關敘述、能力介紹。
- Id：獨一無二之編號。
- JClass：能使用的職業。遊戲中分為 Swordmans(戰士)、Magicians(法師)、Archers(弓箭手)、及 All(全部)四類。
- Category：物品分類(增加能力之種類)。遊戲中分為 WEAPON(武器)、SHIELD(防具)、DIAMOND(飾品)、FOOD_HP(補 HP 類)、FOOD_MP(補 MP 類)、FOOD_ALL(補全部)、FOOD_ATK(攻擊類)、SPECIALLY(特殊物品)。
- Value：物品附加數值。
- Price：物品能賣的金錢。
- SellAble：表示『能賣給商人嗎?』的旗標。
- CanDrop：表示『能丟掉嗎?』的旗標。
- UseOnce：表示『是否只能使用一次？』的旗標。
- Using：表示『是否在被角色使用中?』的旗標。
- Amount：表角色身上攜帶此物品的數量。

物品再有了上述屬性後，便能由如圖 5-47 物品類別，衍生出各式各樣的種類，並藉由遊戲引擎來賦予它們實用價值。

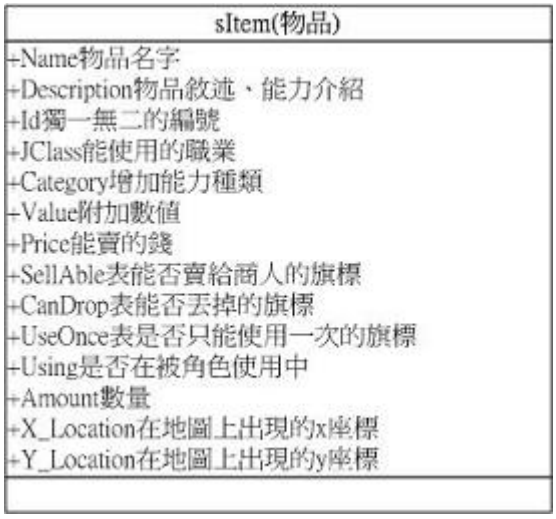

圖 5-47 物品類別

5.7.2 與物品相關之動作

在遊戲中，角色與物品的相關動作有，撿起、丟棄、使用、與不使用，分別介紹如下，

一、從地圖上撿起物品

在地圖上玩家能看得到的東西就能撿起，主要有兩種，金錢跟物品，兩者在程式內部是相同的資料型態。但較不一樣的是，玩家身上通常都會有錢，因此，在處理過程中，只要增加其金額就好。程式流程如圖 5-48 所示

二、將身上的物品丟到地圖上

丟物品的想跟賣東西很像，只是玩家只會失去而不會獲得金額。程式流程如圖 5-49 所示。

三、使用物品

物品又分為消耗性與非消耗性兩種。前者指的是使用過一次後，便會消失的物品，就像玩家喝的紅色藥水一樣，一灌藥水可能只能恢復玩家 60 點 HP，但喝完後就沒有了；而後者，像是武器、防具、及裝飾品都是此類，它們不會平白消失，除非玩家主動將它們賣掉或丟棄。而使用物品，無論是消耗或非消耗，皆透過呼叫玩家的 UseItem() 函式來達成，

- 使用非消耗性物品，程式流程圖如圖 5-50 所示。
- 使用消耗性物品，程式流程圖如圖 5-51 所示。

四、卸下使用中物品

這裡的意思是將使用中的非消耗性物品卸下，舉例來說，當我把現在持有的武器拿下來時，則角色的攻擊力也要扣除武器所提供的附加數值，而這些動作便是在最後一道程序『更新相關資料』時扣除。

- 卸下非消耗性物品，程式流程圖如圖 5-52 所示。

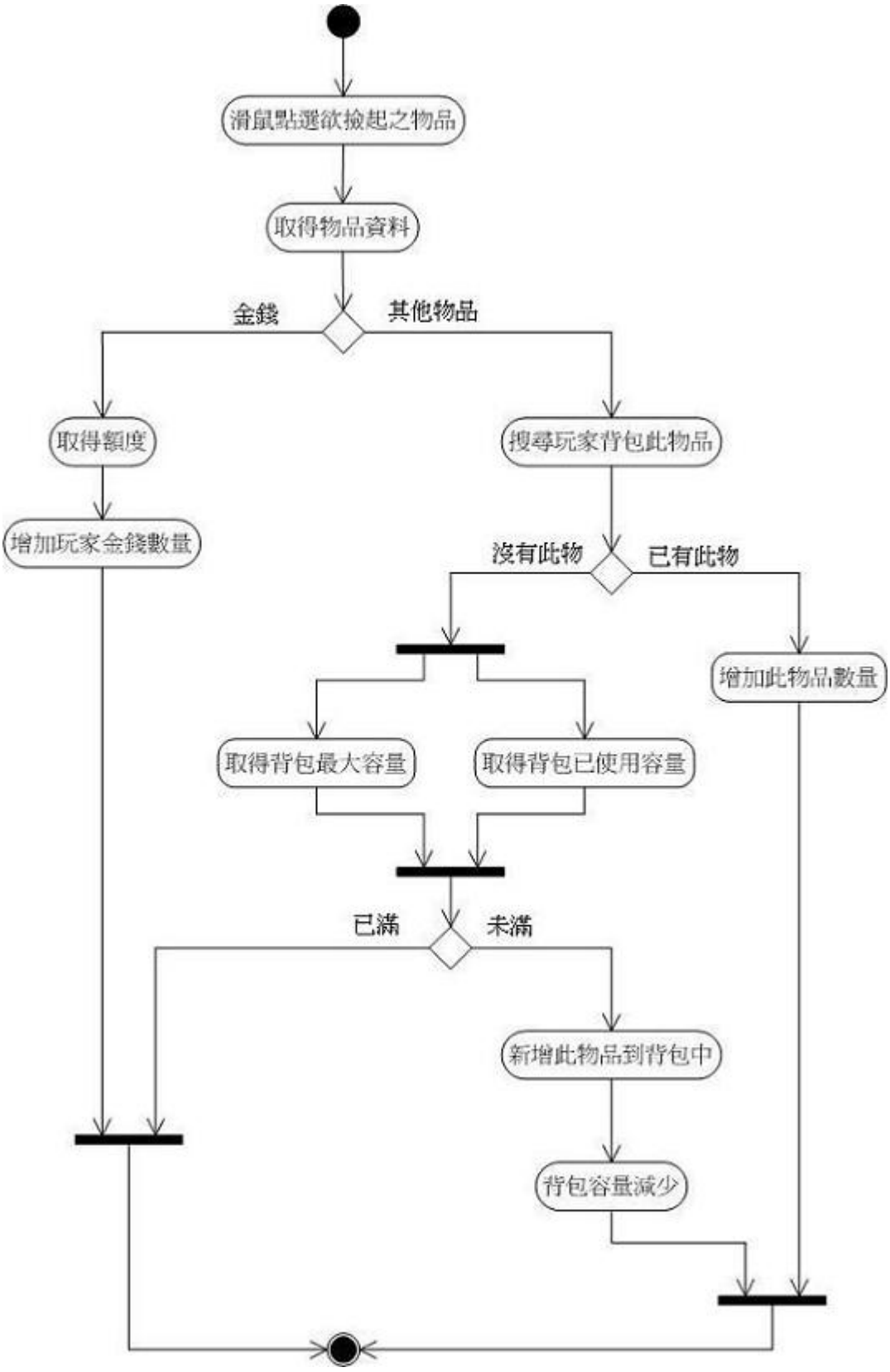

圖 5-48 玩家從地圖上撿起物品之活動圖

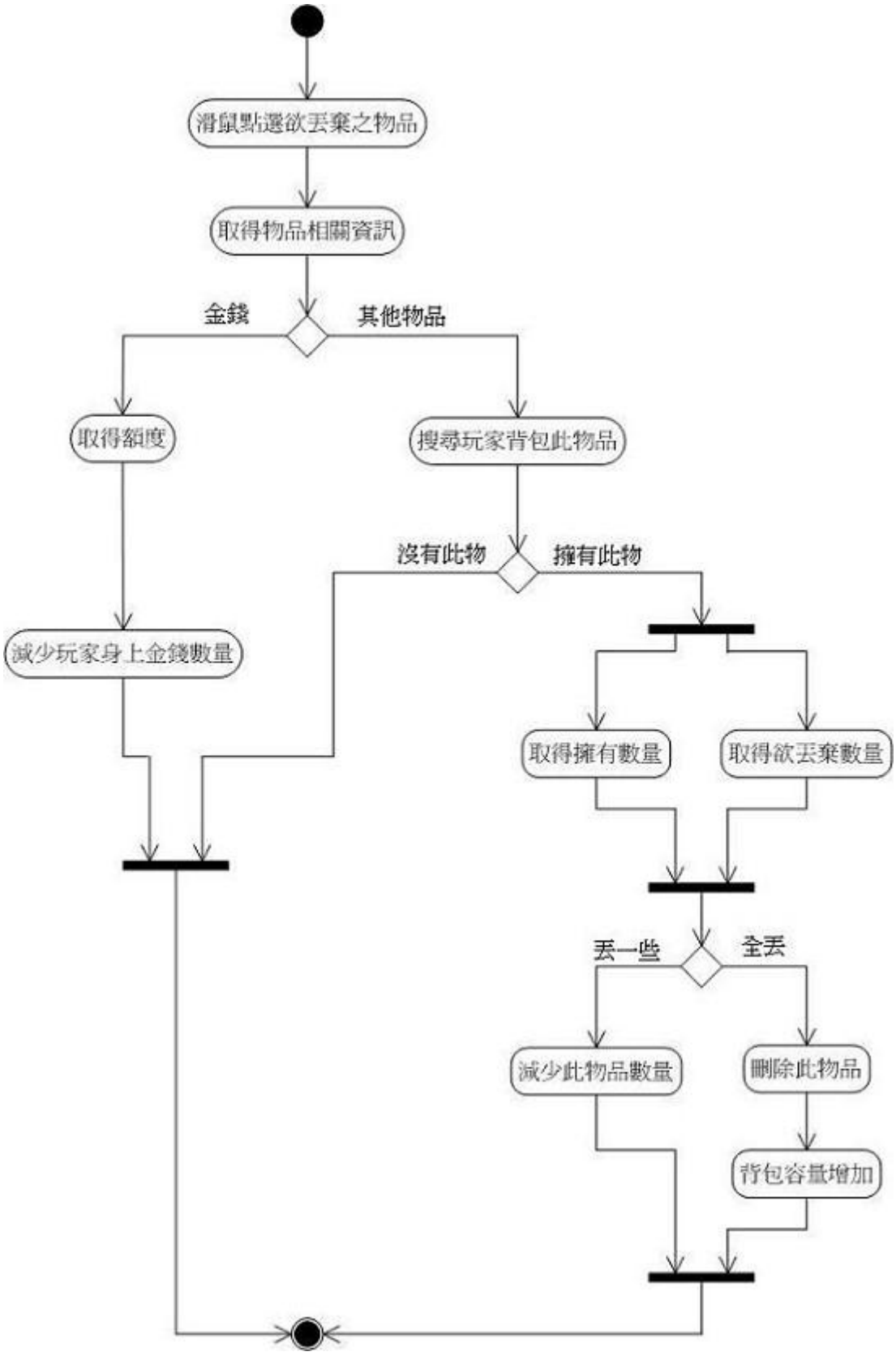

圖 5-49 玩家將身上的物品丟到地圖上之活動圖

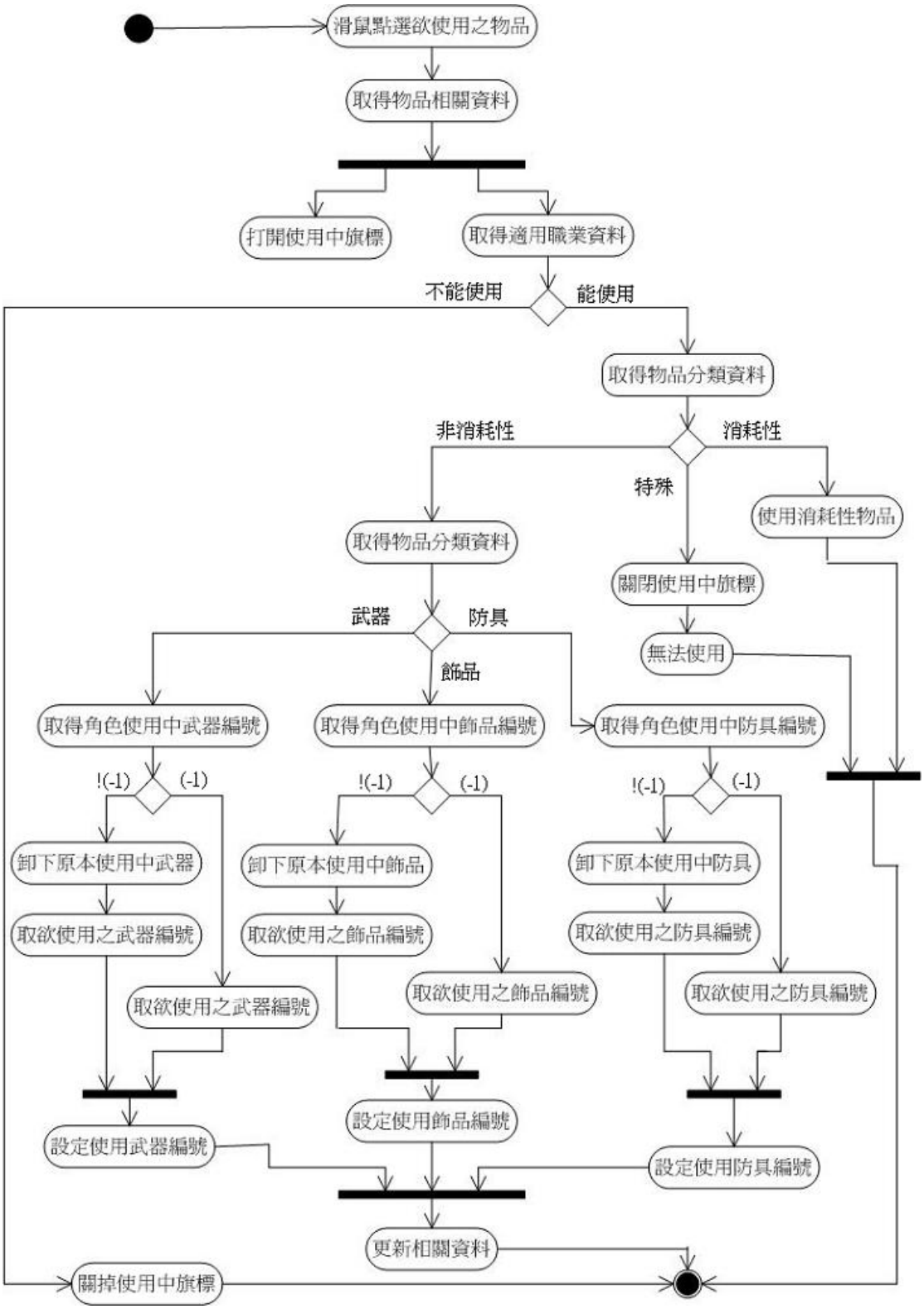

圖 5-50 玩家使用非消耗性物品之活動圖

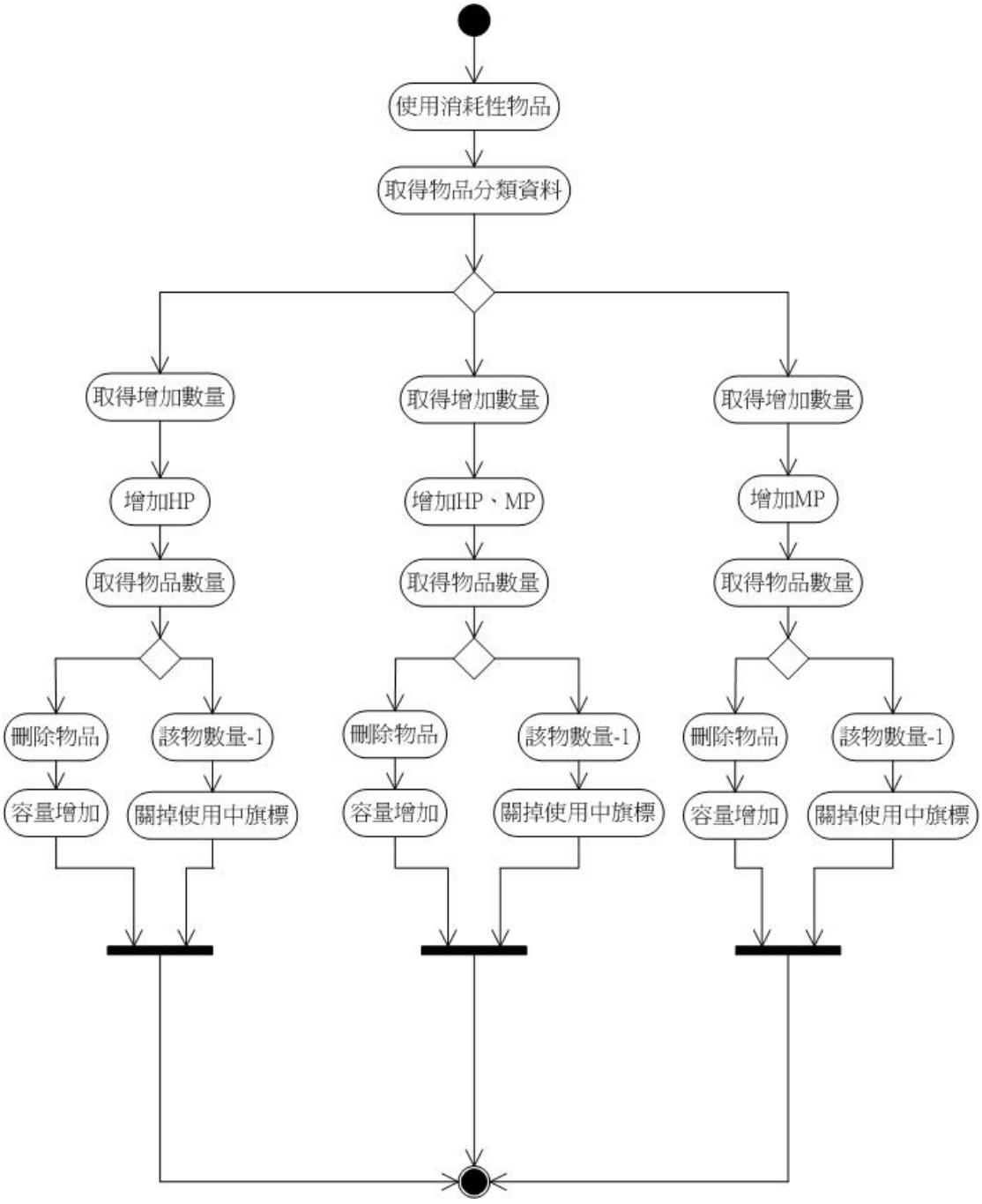

圖 5-51 玩家使用(消耗性)物品之活動圖

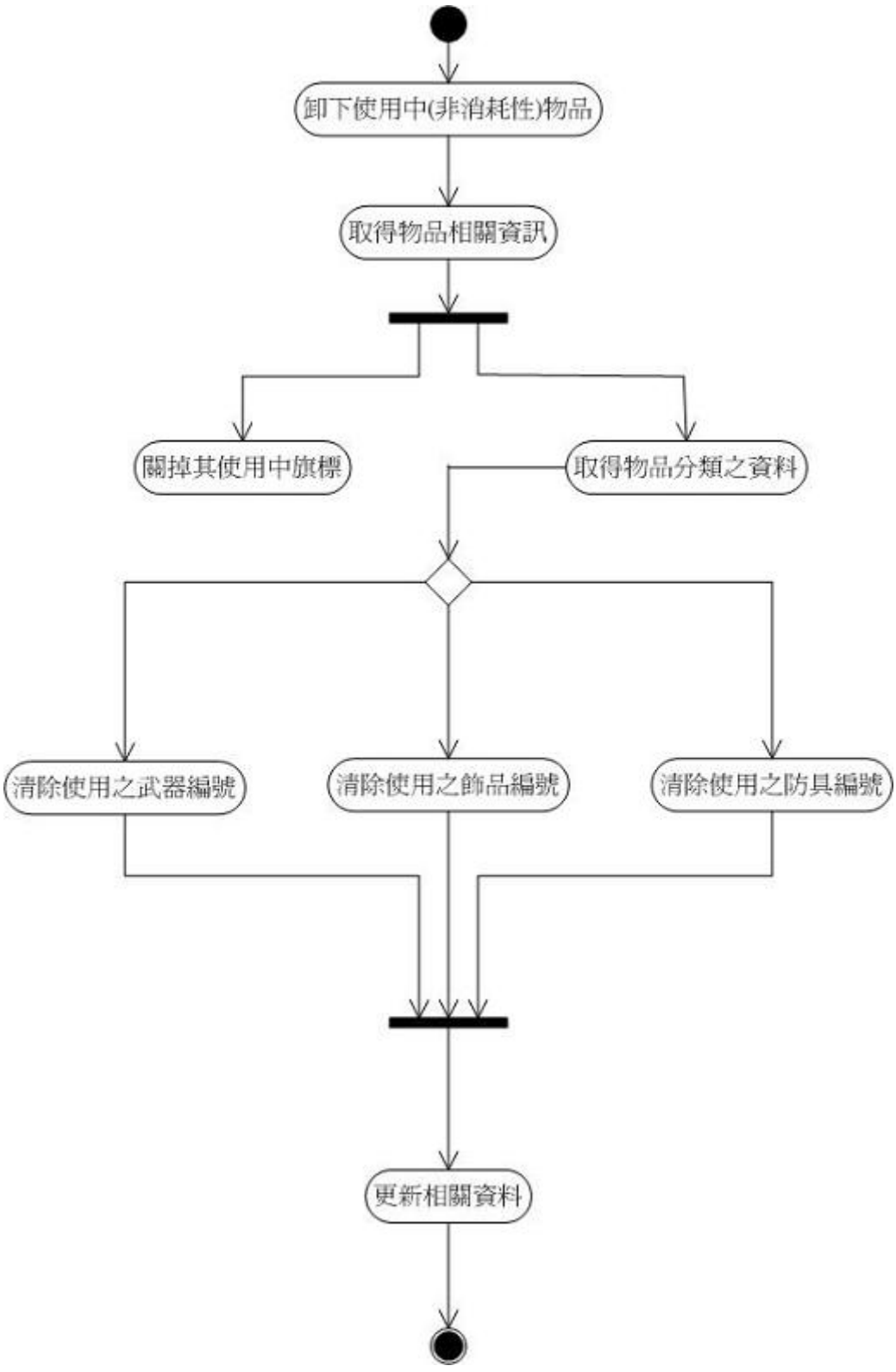

圖 5-52 玩家卸下(非消耗性)物品活動圖

5.7.3 倉庫與商人

遊戲中會出現的物品相當多，那是否有相關機制來管理這些物件呢？是的，倉庫的概念便是由此產生。在我們的遊戲世界中，會出現的物品約有 50~60 種，而這些物品的相關資訊，便通通存放在倉庫裡，當遊戲開始進行時，遊戲引擎便會依照各物品的資料，一一產生相對應的物件。

物件產生後，還要經過一連串的程序，才能讓玩家獲得並使用。而在這之間扮演訊息溝通的角色，就是管理者。倉庫有倉庫管理者，而商人跟玩家則分別有他們的背包管理者。透過這些共通的管理者介面，我們可以很輕易的將物品交予它們管理。而商人或玩家想獲得這些物品，在程式內部，就必須跟他們各自的管理者提出申請，再藉由管理者與管理者之間互相溝通及交流。以下便對管理者物件予以介紹。

一、背包管理者

在背包管理者(packagemanager)類別中，首先會提供一個名為背包(容器)的資料結構，在我們的程式裡將它稱作 sItemSet。如圖 5-53 所示，它內部的結構實際上是一個平衡二元樹(balanced binary tree)。當有元素欲插入時，它會依照程式設計者所撰寫的排序規則來一一插入；當有元素欲刪除時，它會依照程式設計者所設計的關鍵字(資料)搜尋，待尋得後，便將整個物件刪除。

因此，當玩家得到新的物品或商人欲販賣新的物品時，這些代表物品的物件都會透過各自的管理者，將其作排序後並一一放入背包中；當玩家將某些物品丟棄、使用完或賣給其它角色時，同樣地，管理者也會將這些物品從背包中徹底清除。

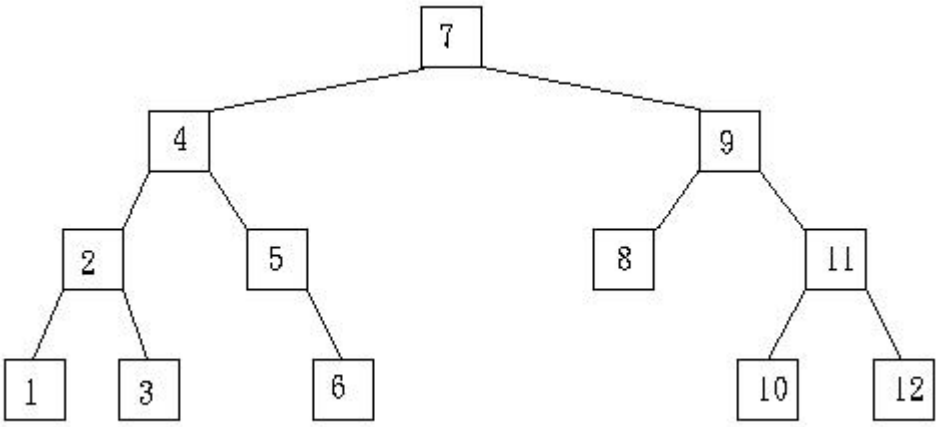

圖 5-53 Sets 的內部資料結構

但是，背包所能安裝的物品數量也不能毫無限制。倉庫須儲存所有會在遊戲中出現的物品的相關資訊，而玩家的角色或許只能攜帶 10 幾樣物品，因此，倉庫管理者的背包容量就遠比玩家角色的背包容量要大的許多。於是，在此類別中也提供了 cMaxCapacity()函式，以設定背包的最大容量。對玩家而言，倘若我的背包容量是 10，那當我欲撿起第 11 樣物品時，可能就撿不起來，而眼睜睜看著寶物被其他玩家撿走了。

另外，背包管理者(packagemanager)類別如圖 5-54 所示，其中提供的操作行為有：

- AddItem()，新增物品。這裡是把物品(物件)當成參數傳入，管理者便會依照其(category，id)來排序，其中 category 表物品分類，id 表編號。

- SearchItem()，背包有其特定的排序規則(由程式設計者提供)，在我們的系統裡是以物品的 category 及物品的 id 來排序。在做排序時會先由物品的 category 來做比較，若此數值不相等，則有小到大排入；若此數值相等，便比較 id。因此，在設計物品物件時，我們必須保證其 id 是獨一無二的。當玩家向背包管理者

提出搜尋某特定物品時，便須以(category，id)當作參數傳入。而當管理者找到該物件時，便會回傳該物件在記憶體中的位址，以供存取。

- DelItem()，刪除物品。這裡需要兩個參數來表示所要刪除的物件，category 以及 id。在程序內部會先呼叫 SearchItem()函式，待找到物件後，在予以刪除。

- GiveMeItem()，這個函式較特別，屬於 GetNewItem()的內部函式。它實際上會宣告一個全新的物品物件，再依傳入的(category，id)參數，尋得所要的物品物件後，將物件相關資料全設給之前宣告的新物件。之後，在把此物品(新物件)交由呼叫者處理。

- GetNewItem()，這個行為函式需要三個參數，category、id 及取得對象(屬背包管理者類別所衍生物件)。這個函式的呼叫，主要是在設定商人欲販賣的東西與玩家的角色欲向商人購買東西時。其中，商人的取得對象是倉庫管理者，而玩家的取得對象則是商人的背包管理者。在函式內部會呼叫取得對象 GiveMeItem()函式，因此在取得物品之前，會先搜尋取得對象的背包中是否有你所指定的東西。若無，則回傳 False，若有便傳回指向該物件的指標，以作資料存取。接下來的動作便如 GiveMeItem()中所介紹的一般。
- begin()，傳回指向此資料結構的第一個元素的迭代器。

- end()，傳回指向此資料結構的最後一個元素的下一記憶體位置。

- ClearAll()，將背包中的物件全部刪除。

玩家與 NPC 背包管理者之間的行為關係如圖 5-55 所示。

PackageManager(背包管理者)
#MaxCapacity背包最大容量 #Capacity背包目前已用容量 -sItemSet背包
+cMaxCapacity設定背包最大容量() +SCapacity取得目前已使用之容量數() +AddItem新增物品() +SearchItem尋找某物品() +GiveMeItem() +GetNewItem向特定對象取得物品() +DelItem刪除某物品() +ClearAll清除全部物品() +begin() +end()

圖 5-54 背包管理者類別

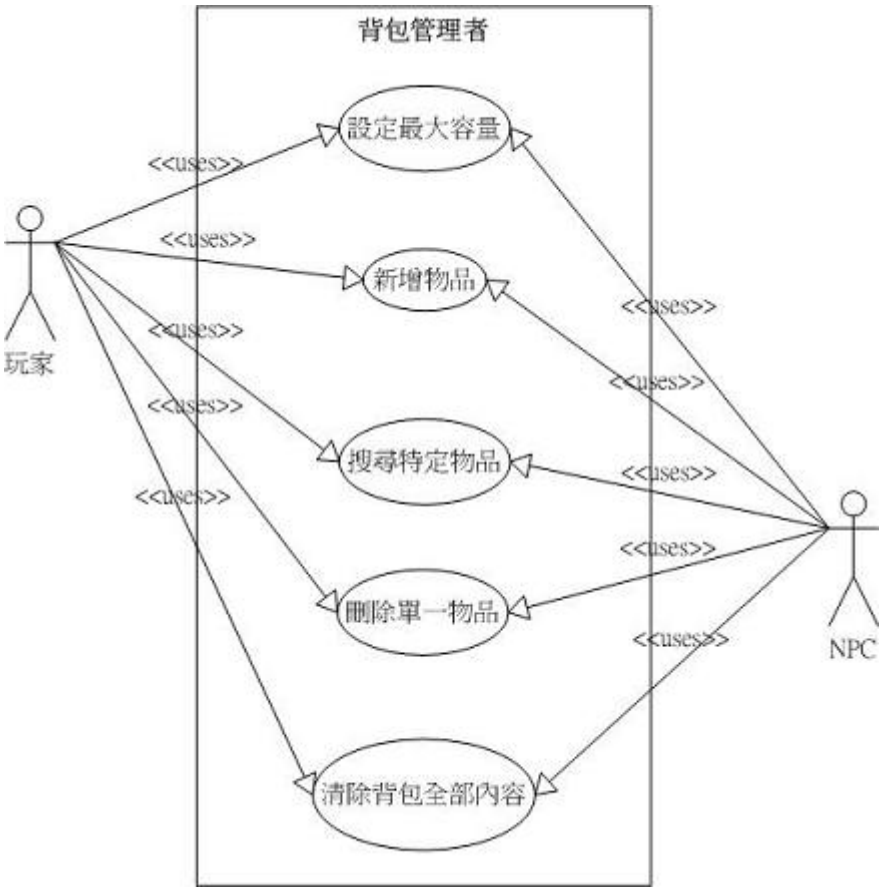

圖 5-55 背包管理者

在介紹完背包管理者(packagemanager)類別後，再來看看倉庫及商人是如何產生的。

二、倉庫的產生

遊戲中所有會出現的物品，經過遊戲開發者依遊戲風格、走向、及甚至平衡度等多方考量及評估之後，才設計出來。而這些物品的相關資料，包括其名稱及敘述等，都會被統一紀錄在一張清單中。當遊戲啟動時，遊戲引擎便會依照清單內容產生物品，再交由倉庫管理者放到其背包中作有效的管理。

三、指定商人販賣物

NPC(非玩家控制角色)主要有兩大類，一類專門負責劇情的傳遞。當玩家第一次接觸遊戲，除了從使用手冊上學到基本遊戲規則以外，在遊戲中，就是靠 NPC 來引導。NPC 可能會藉由對話的方式，來告訴玩家該怎麼打怪升級及任務該怎麼啟動甚至劇情進行到哪邊等，而其角色亦非常多元化，如閒逛中的村民、城門的守衛、及城堡中的公主…等都是。

除此之外，第二類的 NPC 就是商人。在一般的遊戲中，不同種類的物品，要在特定類型的商店中才能交易。像是武器店及衣飾店等，因此，商人又可分為武器商人、防具商人、或葯草商人等等。這樣的分類，可為玩家帶來便利，使他們容易記住要跟誰購買自己的所需物。但若以程式撰寫者的觀點來看，其內部的運作是相似的。

創造一個商人的程式流程便如圖 5-56 所示，其中須特別注意的是，在指定欲販賣物品的程序時，須呼叫其類別中的操作行為 GetNewItem()，而在這個函式中，第三個參數只能夠是倉庫管理者。這個規定也呼應之前所說的，所有物品的資訊都會存放在倉庫中，因此，商人也要跟其倉庫管理者申請才能擁有並販賣。

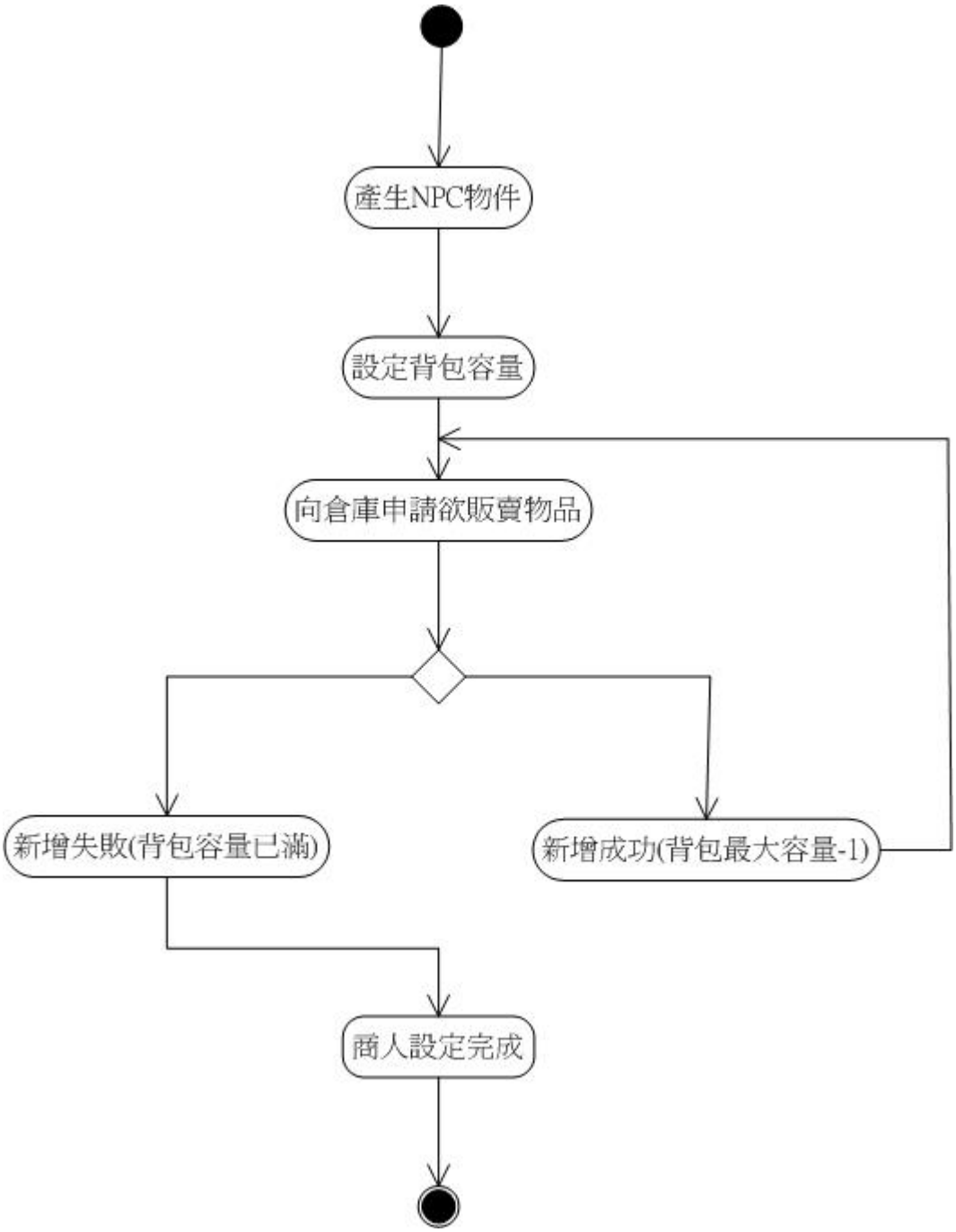

圖 5-56 商人產生與指定販賣物活動圖

5.7.4 交易處理

在遊戲中出現的物品，只要它們是可以販賣的，都可以被拿來交易。在本系統中，交易的對象間有兩種，分別是玩家與玩家、玩家與商人之間。

當玩家欲跟商人進行交易時，需先顯示該商人身上擁有物品的清單，上面會列出商人所有販賣物品及其單價。通常，不管玩家要購買哪項物品，只要是商人擁有的且玩家金錢足夠，都不會發生庫存不足的現象。

而在玩家跟玩家進行交易時，須注意買方跟賣方間資料的更新。就如，玩家甲欲賣物品 A 給玩家乙，當買賣雙方藉由視窗進行溝通及確認之後，此時，玩家甲的背包內就該少了該物品 A 而財產(金錢)也比販賣前有交易金額量的增加；玩家乙除了背包內多了物品 A 之外，其財產也比販賣前有交易金額量的減少，至此，此筆交易才算真正的成功。

以上介紹的都是玩家購買的狀況。而在販賣的情形下，玩家、商人與玩家、玩家之間，其程式內部處理，大致上是一樣的。僅有的差別是，玩家賣給商人的價錢會低於原價，本系統是打六折；而賣給玩家，則可由買賣雙方去協調價錢，所以可能會比直接賣給商人划算。

交易有四種模式，分別為：

一、玩家向商人購買物品

活動圖如圖 5-57 所示。玩家向商人購買物品時，須先搜尋玩家背包，看其是否原本就擁有此物品。因為玩家所能攜帶的物品數量有限，若原本就擁有此項物品，則可直接增加此物品數量；但若沒有，則其背包剩餘容量至少要為 1，玩家方能購買此物。

假設背包容量確定沒問題後，便要看玩家身上是否有足夠的金錢，來付給商人。當一切程序都沒問題後，玩家便會發現其身上多了某項物品，且其身上金錢亦少了一定的額度。這些動作會在伺服端處理，完成後再將結果傳回。

二、玩家將物品賣給商人

活動圖如圖 5-58 所示。當玩家身上有不要的物品，或為了買某項物品要湊錢時，便能把不要的賣給商人，但價格會比當初購買時低上一些。販賣物品時，玩家需要輸入數量，這時程式便會比較販賣數量及原本擁有數量，看是否全賣。若是，再增加玩家金錢後須將此物品從背包中刪除，且增加其容量；若不是，則直接減少數量、付錢給玩家即可。

三、玩家向玩家購買、販賣物品

活動圖如圖 5-59 及圖 5-60 所示。玩家向玩家購買物品時，通常會講好價錢，但是，有些玩家或許忘了自己身上是否還放的下、錢是否夠，因此，程式還是要做檢查。程式流程如圖所示。而賣東西的那一方，也得確定是否有把東西帶在身上，且要輸入正確的數量。

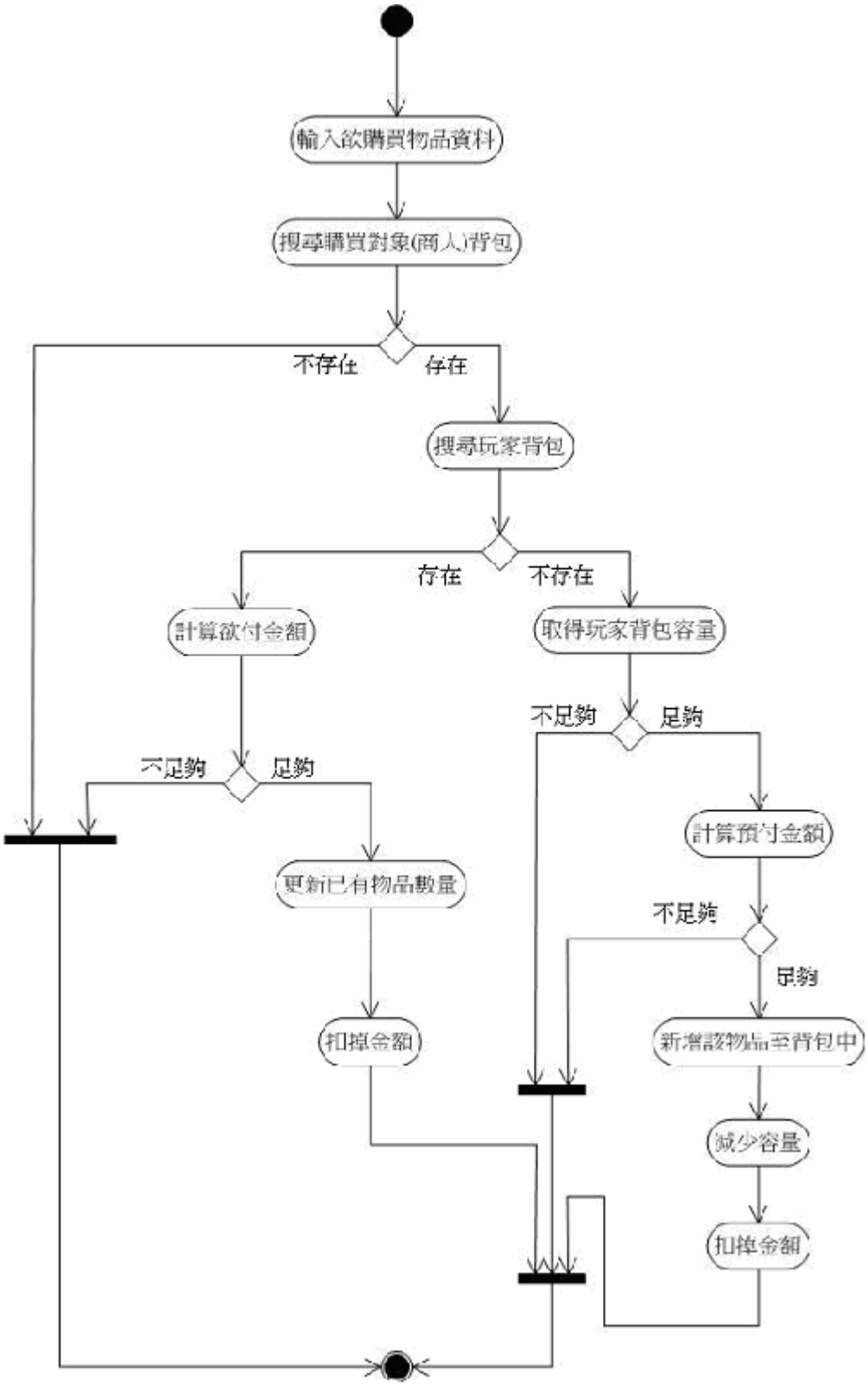

圖 5-57 玩家向商人購買物品之活動圖

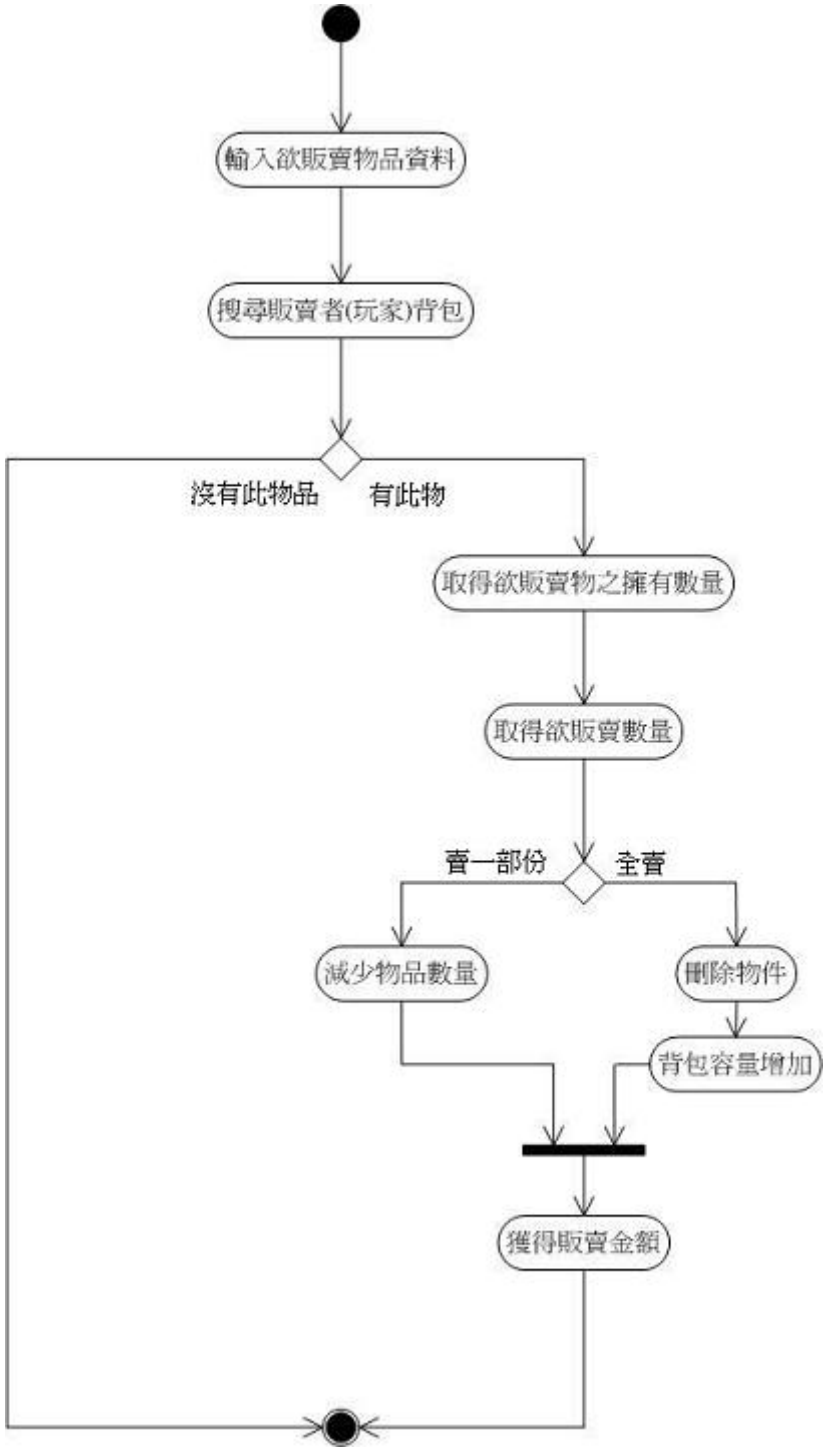

圖 5-58 玩家將物品賣給商人之活動圖

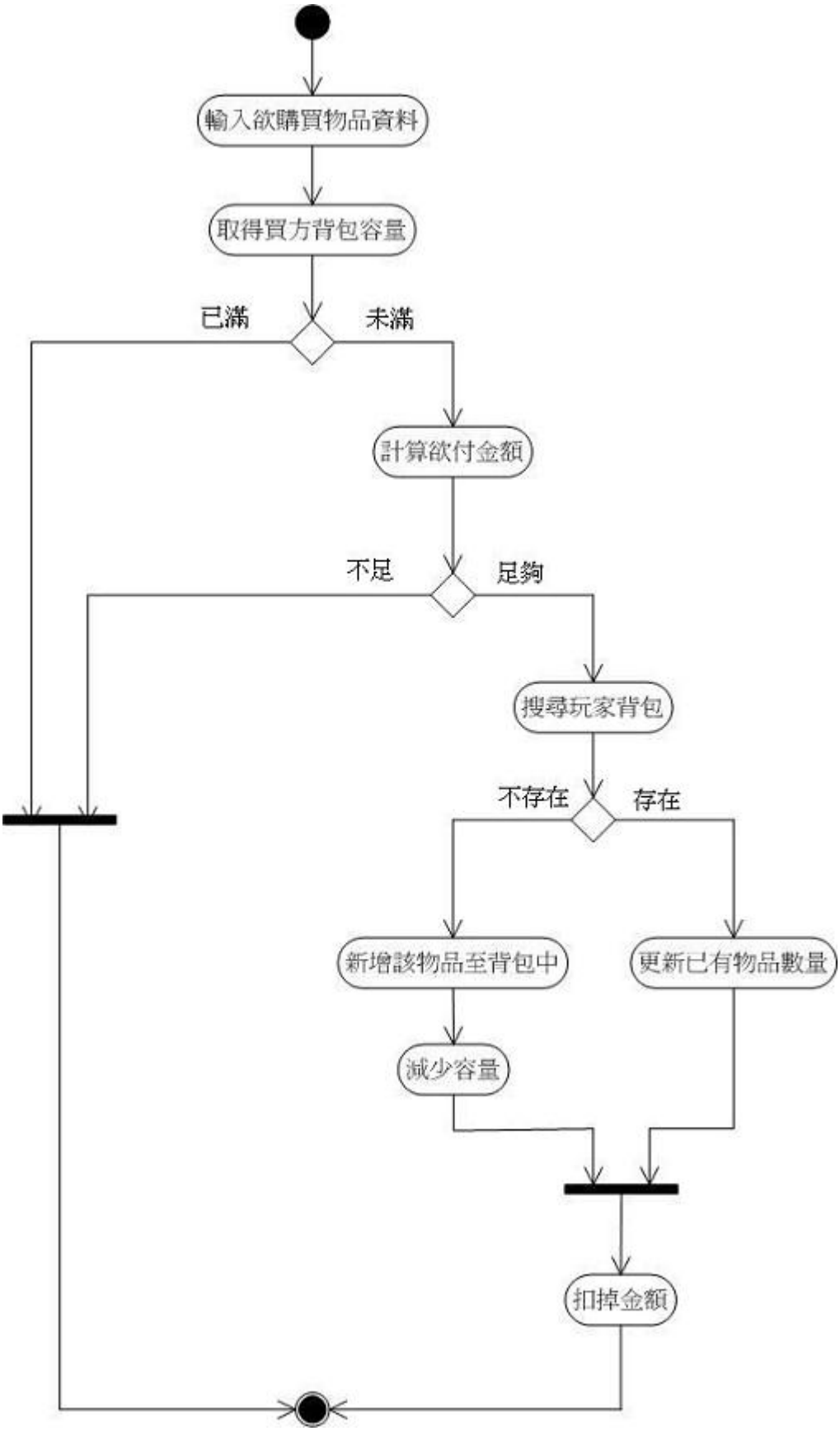

圖 5-59 玩家向玩家購買物品之活動圖

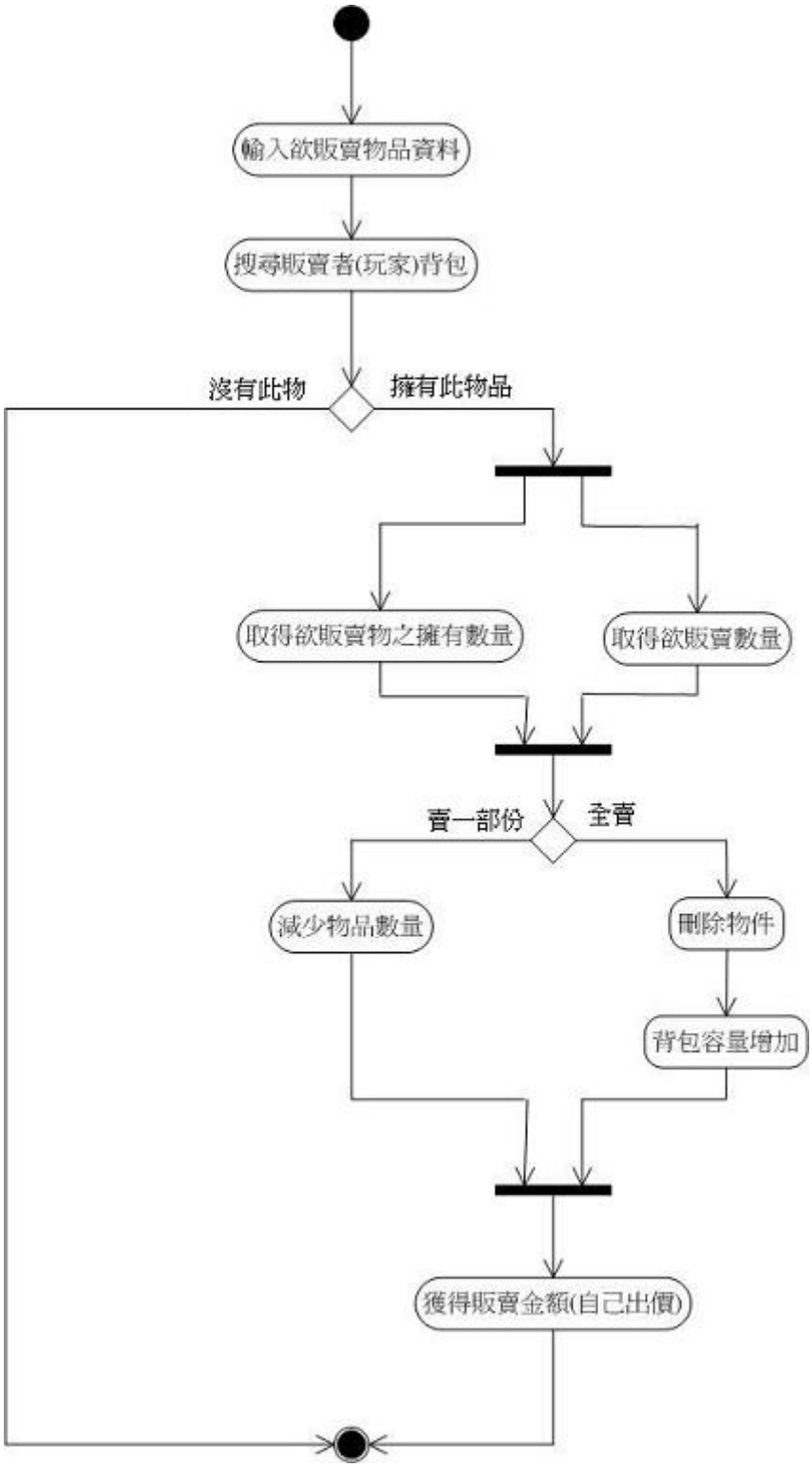

圖 5-60 玩家將物品賣給玩家之活動圖

5.8 物件工廠

物件工廠能統一管理物件，並提供主程式新增、刪除、及搜尋物件的功能，而且將物件之間共通的資料共用一份，這樣能達到，減少動態物件的重複性資料，以節省記憶體空間。

5.8.1 基本概念

在遊戲進行中，會產生相當多「類似」的物件，其中有些資料是相同重覆的。當這些物件越來越多時，資料的重覆性問題就越來越嚴重，對用戶端而言可能還可忍受，因為用戶端只需處理自己及部分從伺服端傳送的資料，但在 伺服端則必需處理所有連線進來的用戶端資料，對伺服端而言，這些重覆性資料會是一個極大的負擔，於是一個統籌管理物件以減少及共享重覆性資料的「物件工廠」概念便產生了。

5.8.2 共享性資料概念

在遊戲中，物件與物件之間的資料中，有些是重覆的，而有些則不相同，因此，如果將那些重覆的資料分離出來，亦即，在那些擁有部分相同資料的物件間，將相同的部分抽離出來並「共享」，這樣將可減少不必要的空間浪費。

在圖 5-61 中，將單一物件的資料拆成二個部分，一個部分是具有獨立性質，另一個部分則是具有共享性質，不過這樣的做法，在類別的設計上也需做點改變，得將原先的類別變成兩部分，如圖 5-62 及圖 5-63 所示。

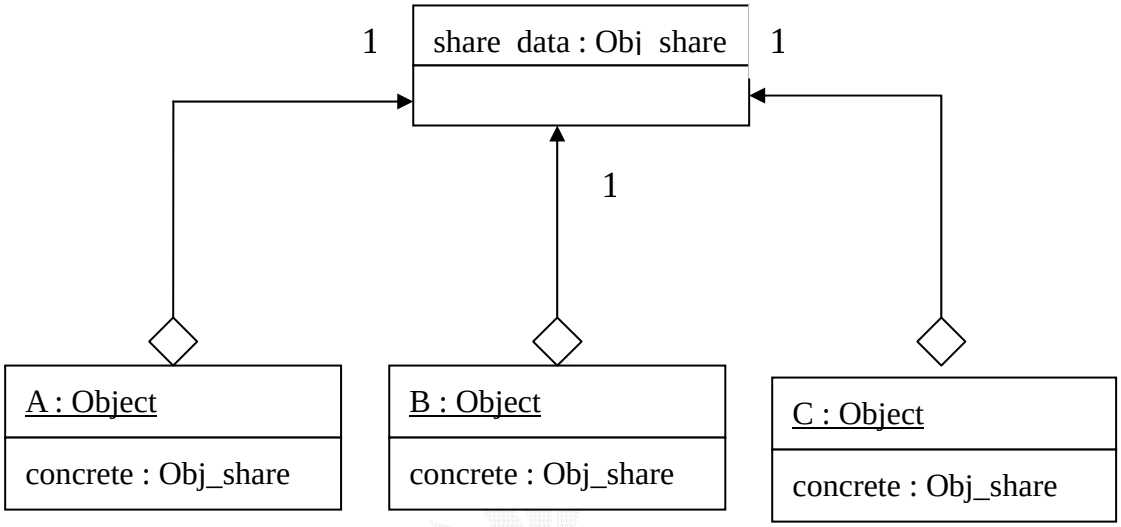

圖 5-61 物件共享資料關係圖

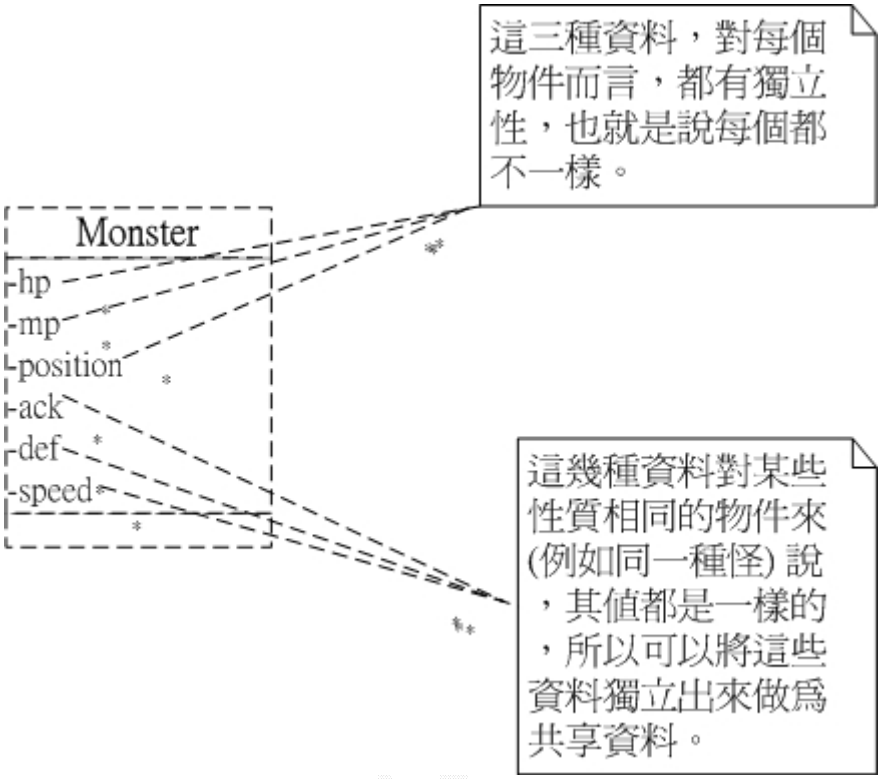

圖 5-62 資料未分離時之情況

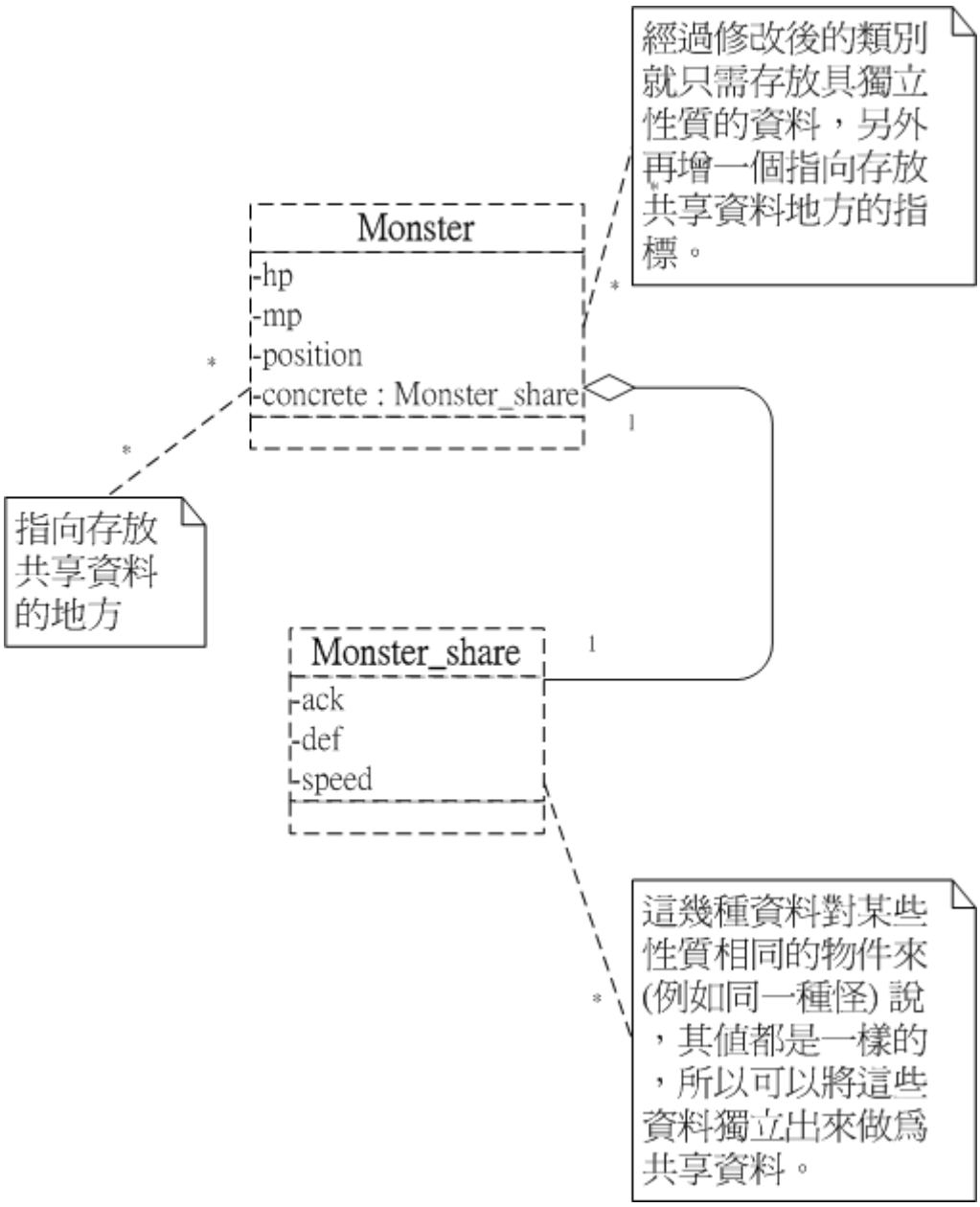

圖 5-63 資料採取共享機制之情況

如此一來，以同一種怪來說，他們的 ack（攻擊力）、def（防禦力）、speed（速度）都是一樣的，所以將他們獨立出來共享，Monster 物件內部以 concrete 指標指向共享資料，當需要使用到這些共享資料時，就透過 concrete 指標來捉取。不過這裡有一點要注意的是，共享資料的內容不應該被任何 Monster 物件所更改，因為既然是「大家共有」的資料，但又被當被「私有」資料更改，這樣就產生了矛盾，所以除非特殊情況（例如這個更改是針對群體的），否則個別物件不應該對任何共享資料做更改的動作。

5.8.3 工廠實作原理

圖 5-64、5-65、及 5-66 為伺服端的運作架構。在本系統中，伺服端可以產生許多不一樣的遊戲世界，如圖中的 game1、game2、及 game3 等，每個遊戲世界內會有它們各自的怪物資料、NPC 資料、及玩家資料等，以應付不同的風格劇情走向。比如說，對怪物的資料而言，會有一個怪物工廠為每個遊戲各自配置一條專屬生產線來放置該遊戲世界所有怪物的資料。

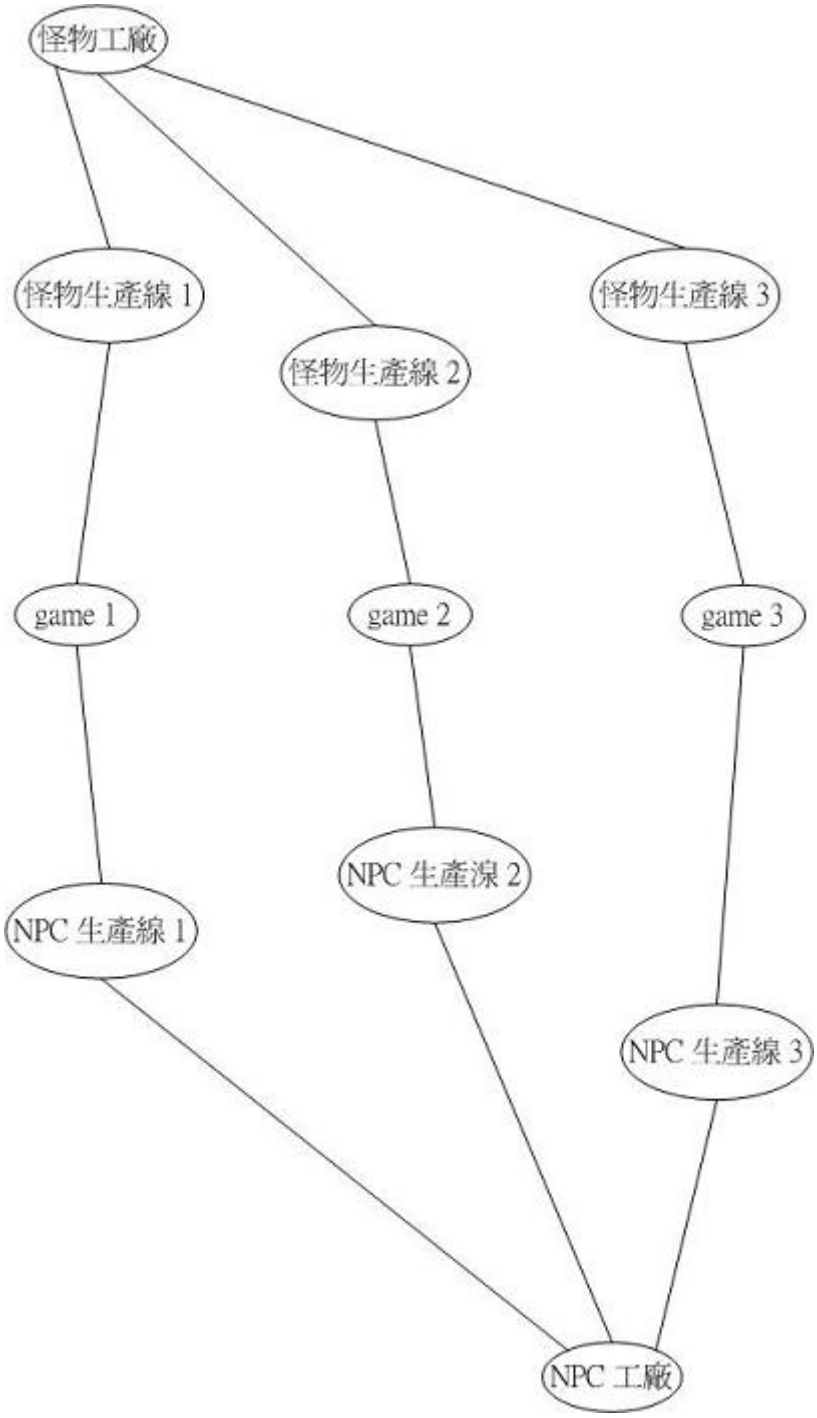

圖 5-64 伺服端工廠運作架構

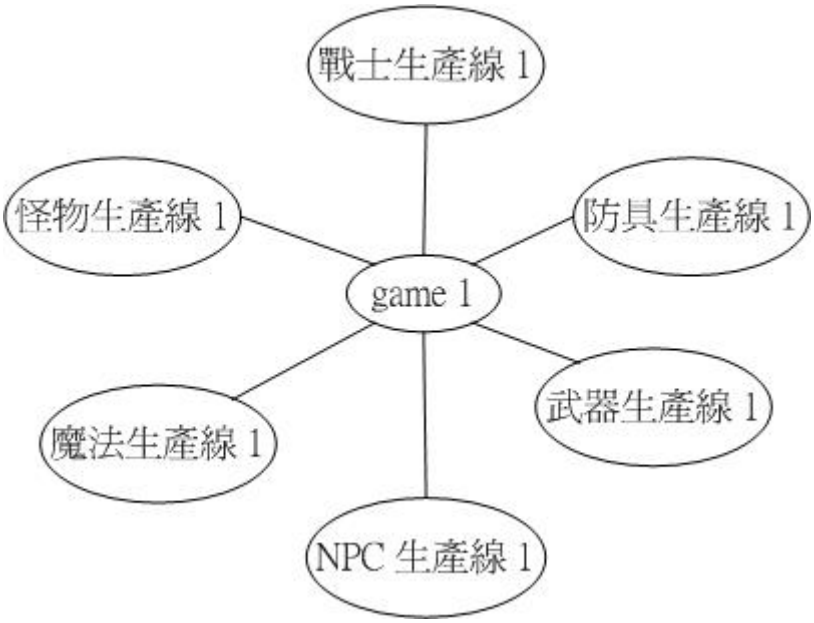

圖 5-65 game1 世界中負責生產的各類生產線

圖 5-66 game2 世界中負責生產的各類生產線

圖 5-67 中，需注意兩個地方：

一、伺服端需存有各個遊戲的物件資料。

比如玩家的編號、怪物的血量、及 NPC 的位置等。並且要維護這些資料的完整性，且加以妥善管理避免資料錯亂的發生，比如 game1 中的玩家資料存到 game2 的玩家資料。

二、用戶端這方，

關於像 NPC 的位置及怪物血量等，這些即時性且不需要由自己計算的資料，只需接受伺服端傳來的資料再處理就好。而用戶本身大部分所需儲存的資料是一些非動態更新的資料，像是圖檔、場景等。

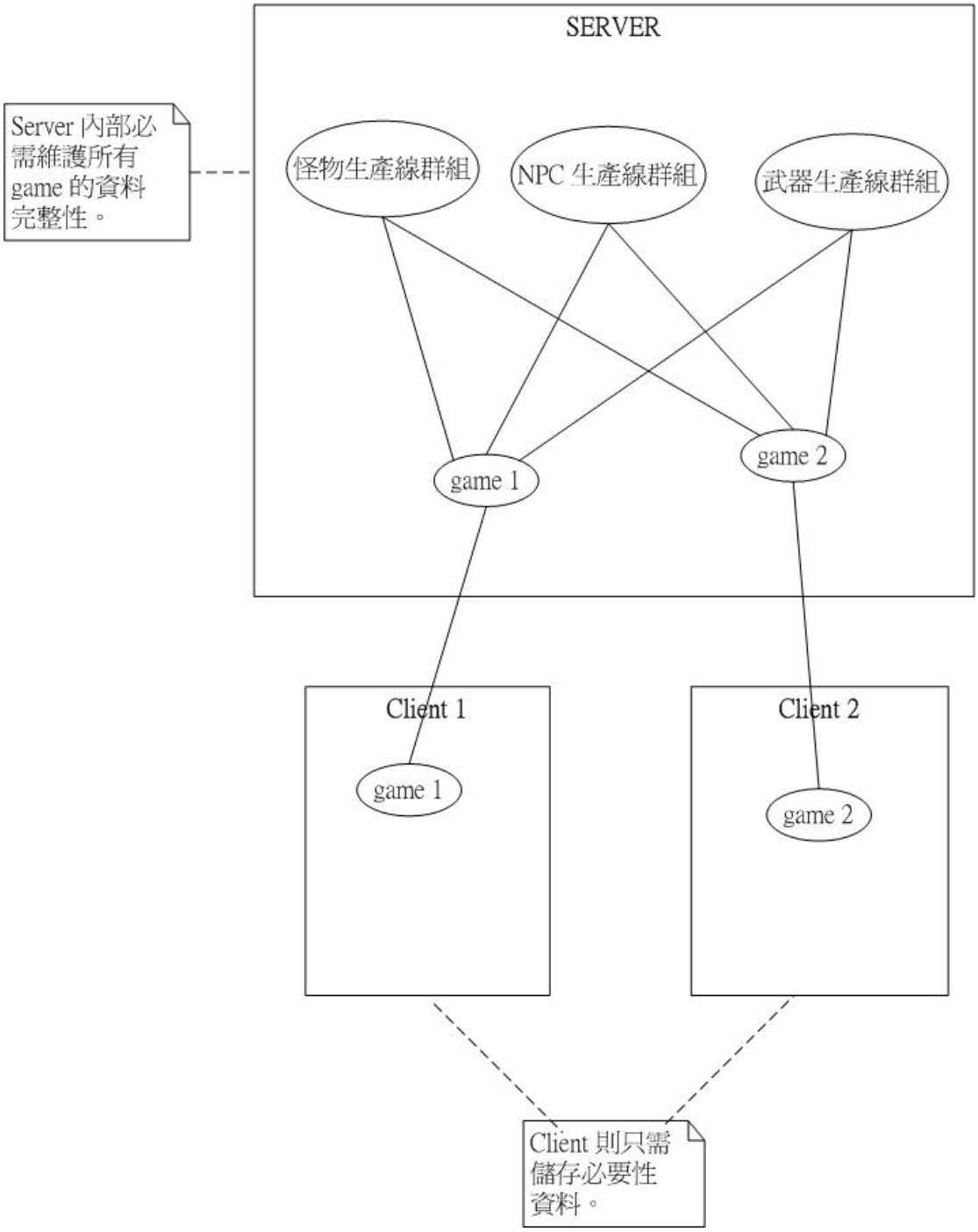

圖 5-67 伺服端、用戶端間資料的關係

5.8.4 物件生產過程

所有物件的資料部分會運用 5.8.2 節所述的概念，分成兩個部分，一個部分是基本共通的，另一部分則是自己特有的資料。比如說怪物生產線中所產生的怪物，它的資料像是 ack(攻擊力)及 def(防

御力)這些是共通的，而這些資料是由主程式利用工廠來設定(開發者需由工廠所提供的設定函式來塑造各種不同類型怪物的原型)。而其它資料像是 hp(血量)及 mp(魔力)等則由主程式利用生產線來設定(生產線亦會提供函式來供主程式使用)。圖 5-68 表某生產線中物件生產過程之活動圖。

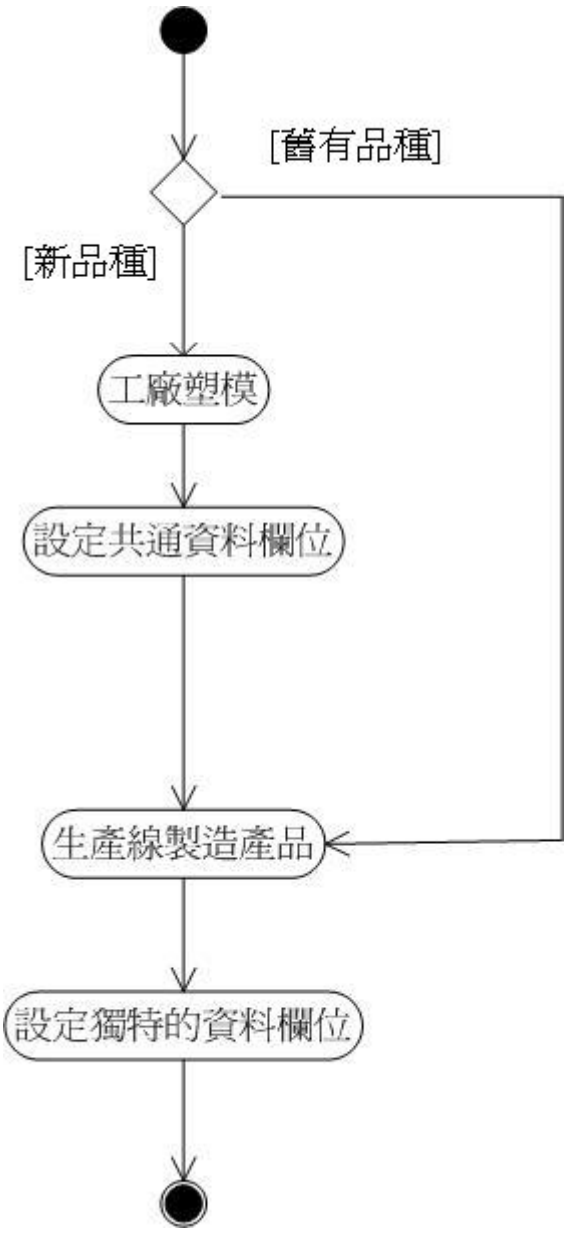

圖 5-68 某生產線中物件生產活動圖

5.9 視窗呈現系統

為了將遊戲內的某些資料有效的表達給玩家了解，於是得提供一些適當的介面，視窗，便是一項很好的工具。當玩家點選 NPC 後，可能會出現一段對話，這時就可出現一個「對話視窗」來顯示這段對話；或者當玩家想看自己遊戲內角色的裝備有那些時，就可開啟「裝備視窗」來查看，對玩家而言，這樣的表達方式會比較有親切感。

因為一般 Windows 的程式在畫面顯示的部分為單緩衝區的設計，但在 3D 的環境下，單緩衝區的設計會造成很嚴重的閃爍，無法直接在遊戲內直接使用 Windows 提供的視窗元件，因此，在畫面顯示的部分得自行處理。不過，平常 Windows 的視窗元件除了顯示外，還附帶地幫我們做很多處理及做一些預設動作，像是按扭按下後，按鈕圖示會有所改變，且會發送按鈕被按下的訊息給程式，但這些不能用之後，就全得自行模擬了，這也是最麻煩的地方。

5.9.1 遊戲中的視窗種類

我們的遊戲內主要有以下幾種視窗：

- 快捷列：方便玩家(開/關)某個特定功能視窗。如以下的狀態視窗。
- 狀態視窗：玩家的角色狀態視窗，會顯示一些玩家的血量及等級之類的資料。
- 物品視窗：玩家的物品資料視窗，會顯示目前玩家所擁有的裝備。
- 商店：當玩家對 NPC(買/賣)物品時，便會出現的視窗，其中會出現可供玩家購買的物品。
- 交易視窗：當玩家對 NPC 要進行交易時所出現的視窗，可讓玩家輸入一些交易數量等。

- 對話視窗：當玩家與 NPC 交談時會出現的視窗，會顯示對話內容。

5.9.2 視窗間的關係

視窗間的關係如圖 5-69 所示，有一個視窗管理者會負責管理所有的視窗（如 5.9.1 節中所定義的視窗），而每個視窗又有它們各自所屬的子視窗其關係如圖 5-70 所示，比如快捷列視窗中有四個子視窗(狀態欄、物品欄、技能、能力，如圖 5-71)，當玩家做出點選的動作後，點選的訊息會先傳遞給視窗管理者，視窗管理者會判斷它是否點中快捷列視窗，若是，則將此訊息交給快捷列視窗處理，而後，快捷列視窗則再依它內部的訊息處理函式去處理訊息，如果沒有點中子視窗的話，就不做反應，若點中，則會引起特定的處理。

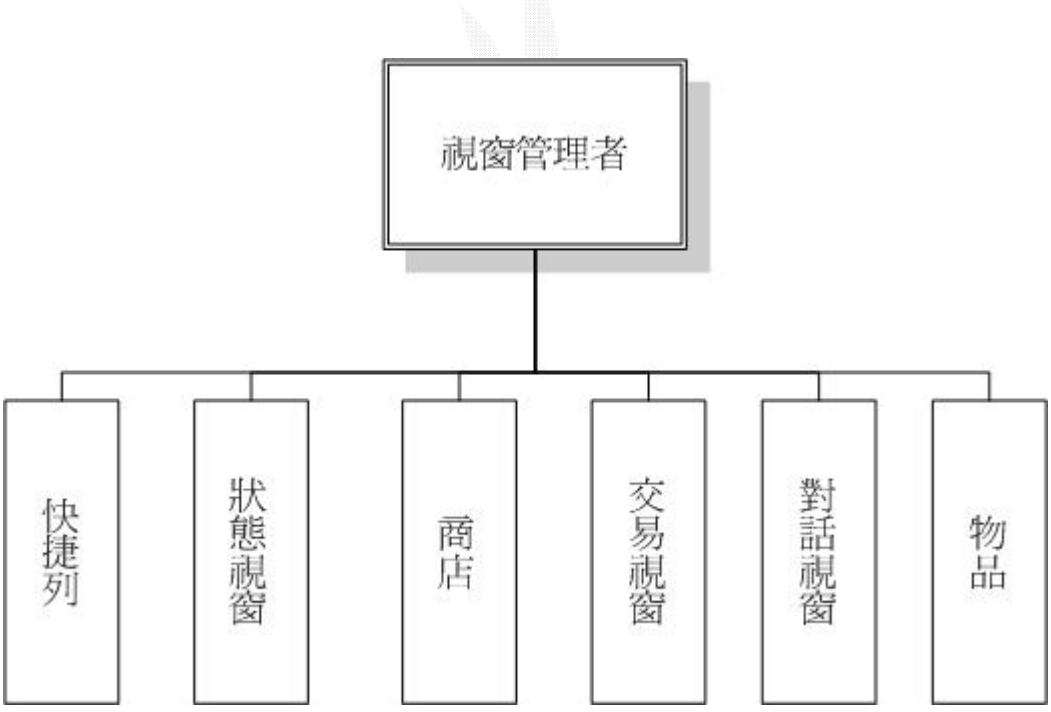

圖 5-69 視窗關係圖

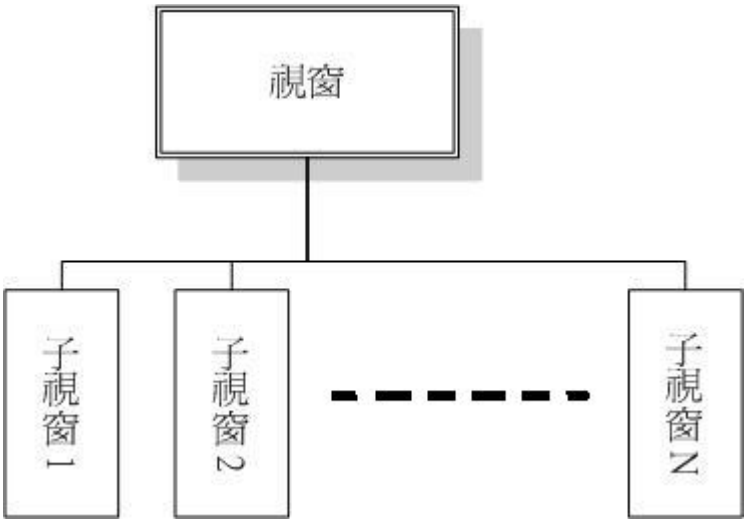

圖 5-70 父子視窗關係圖

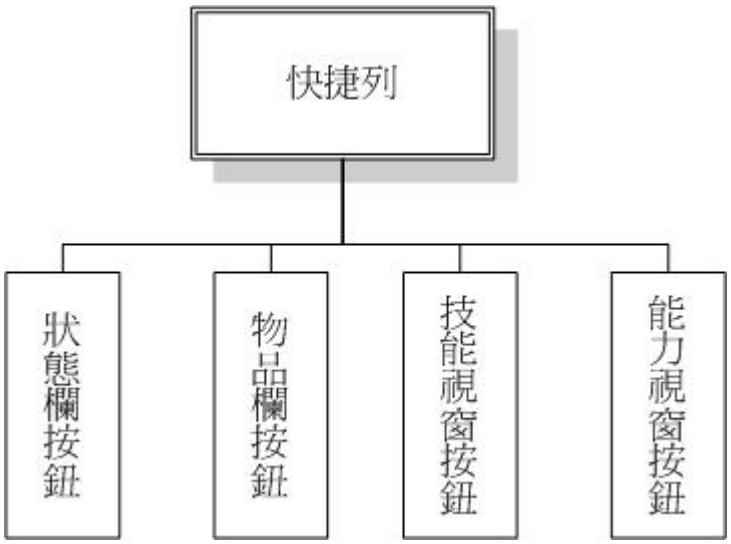

圖 5-71 父子視窗關係圖範例

5.9.3 視窗背後訊息處理的模式

玩家對滑鼠操作的不同，會產生不一樣的訊息，每種訊息在其背後會有一組相對應的連串處理程序，這樣的處理程序又可分為兩種，一種是與伺服端有關的，而另一種則是無關的。

一、與伺服端相關之處理程序

主要的有交易、升級、技能增加…等之點選，這些動作會影響到遊戲相關資料，需要由伺服端來計算更新。圖 5-72 則為玩家點選商品購買之活動圖，由圖中可以清楚看出整個過程是如何處理。

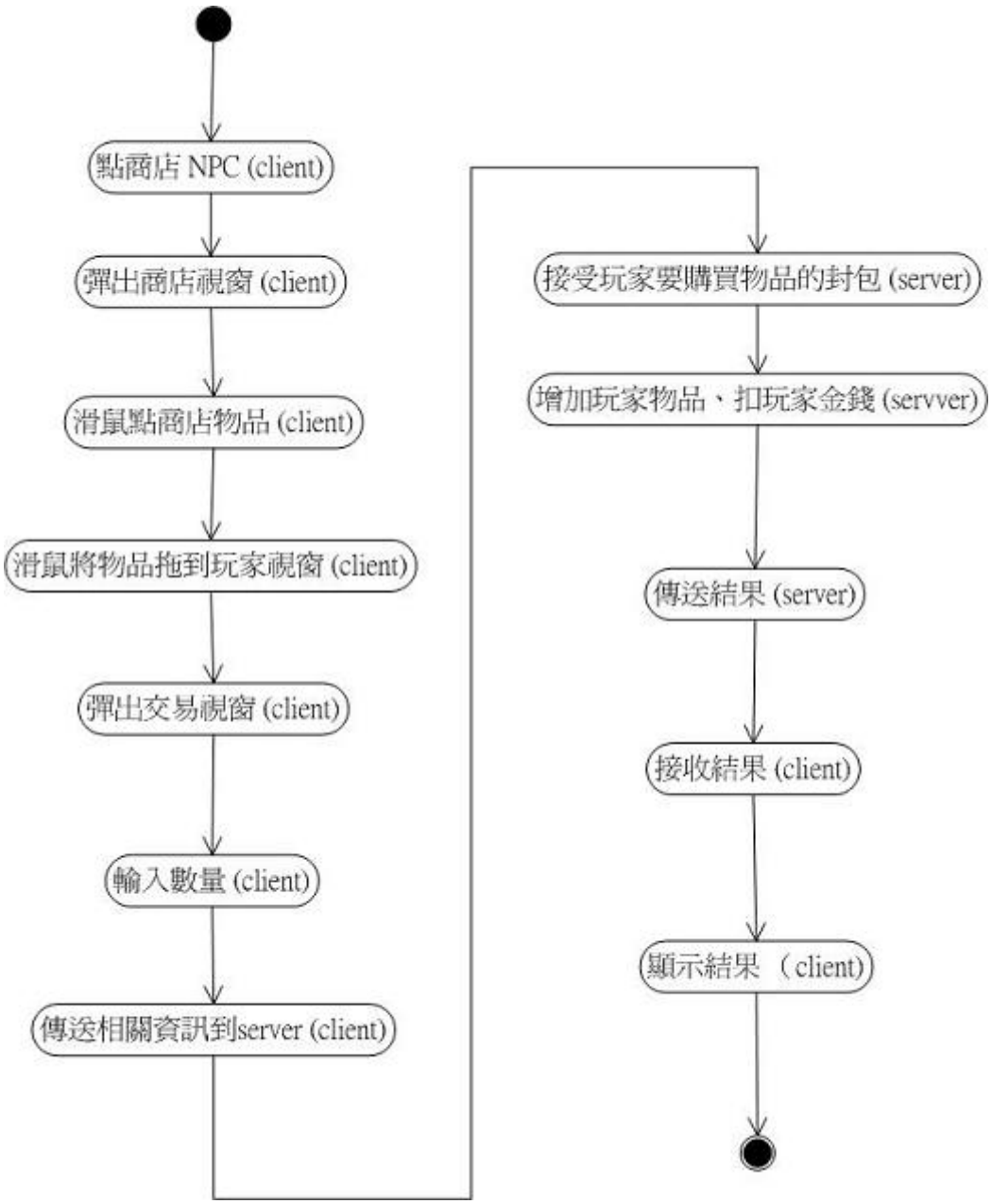

圖 5-72 玩家點選商品購買之活動圖

二、與伺服端無關之處理程序

主要的有按鈕及拖曳視窗等之點選，這些動作不會影響到遊戲相關資料，只會改變視窗的呈現。

5.10 網路連線系統

網路連線系統，是用 DirectPlay 架構出來的，而 DirectPlay 是以 COM 的方式開發出來，所以要先使用 CoCreateInstance 取得 DirectPlay API 介面，之後列舉出所有網路設備，選擇一個適合設備，這樣就完成第一步的準備工作。

最後，最重要的工作就是要如何設計編輯訊息封包，及管理訊息封包。訊息的設計方面，每個訊息封包會有一個標頭結構，這樣就很容易分配封包種類，如圖 5-73 所示。

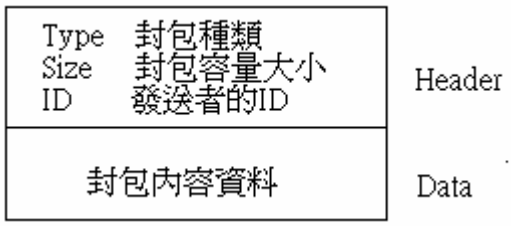

圖 5-73 封包標頭

網路封包的傳送接收是由另一個執行緒平行處理，如圖 5-74 所示，當接收到封包的時後，就將封包都放入佇列，以先進先出的方式處理每一個封包，運用 Header 指標和 Tail 指標控制資料儲存位置，資料寫入佇列時，Header 指標加一，如果超過最大值就回到 0；讀取資料的時後，Tail 減一，如果小於 0 就到最大值，當 Header 和 Tail 差距愈大的時後，表示尚未處理資料愈多，所以 Header+1=Tail 的情況發生的時後就是 full，而 Header=Tail 的時後，就是 empty。

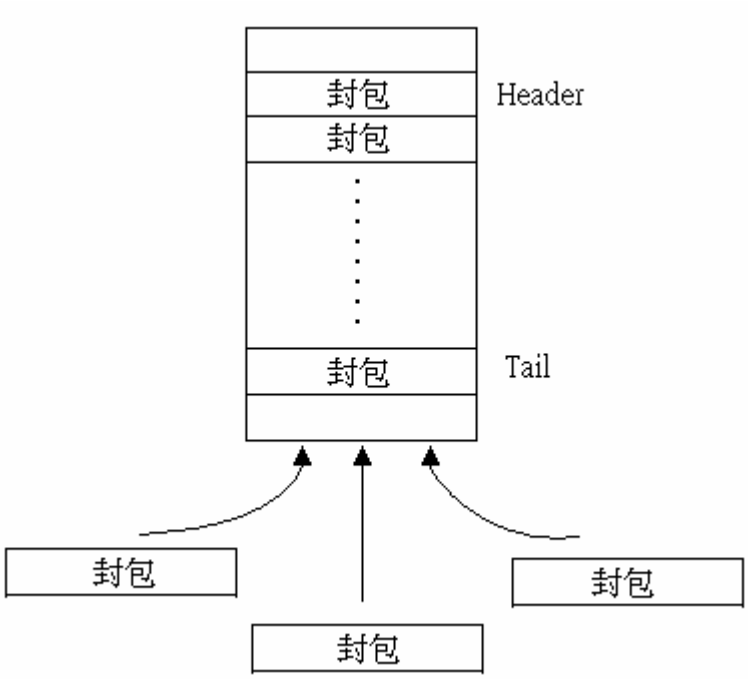

圖 5-74 封包佇列說明圖

在處理傳送過程中並不需要佇列，直接由低層的網路協定處理。另一面傳送封包的時後，指定傳送對象在用戶端跟伺服端上有一些不同的地方如圖 5-75 所示，用戶端傳送封包的對象只有伺服端一方，而伺服端要溝通的是所有的用戶端，但是有些訊息並不需要傳送給每一個用戶端，反而只要傳送給某些族群的用戶端就夠了，如移動封包和攻擊封包只要傳給同一個地圖上的玩家就行了，在這方面可以運用 DirectPlay 內 CreateGroup 的功能，把同地圖的玩家設為同一個族群，所以傳送封包的時後，對象只要指定 Group 的 ID，就會自動把訊息傳送給這族群的所有玩家，這樣就可以減少無謂的封包。

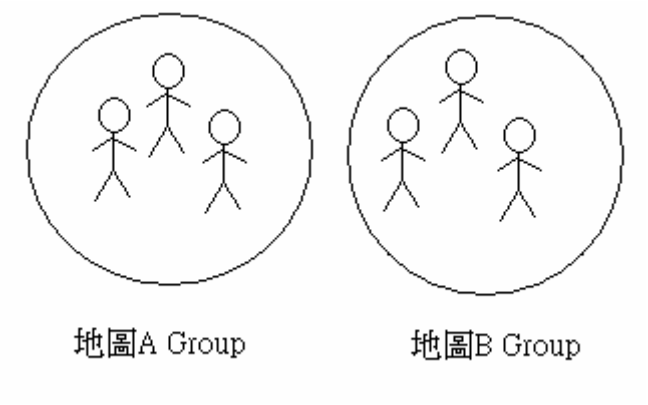

圖 5-75 不同地圖 Group 示意圖

5.11 劇情創造系統

提供給主程式一個介面，讓它可以建立、觸發劇情。使遊戲內有劇情的變化。讓遊戲有更豐富的內容來展現出遊戲的深度，如此一來就能吸引更多的玩家。

5.11.1 基本概念

為了讓遊戲開發者容易設計不一樣的劇情，且又不希望劇情寫死在程式裡，因此提供一套機制，讓遊戲開發者易於編寫、修改、及擴充劇情。圖 5-76 為此機制的主要組成元件。

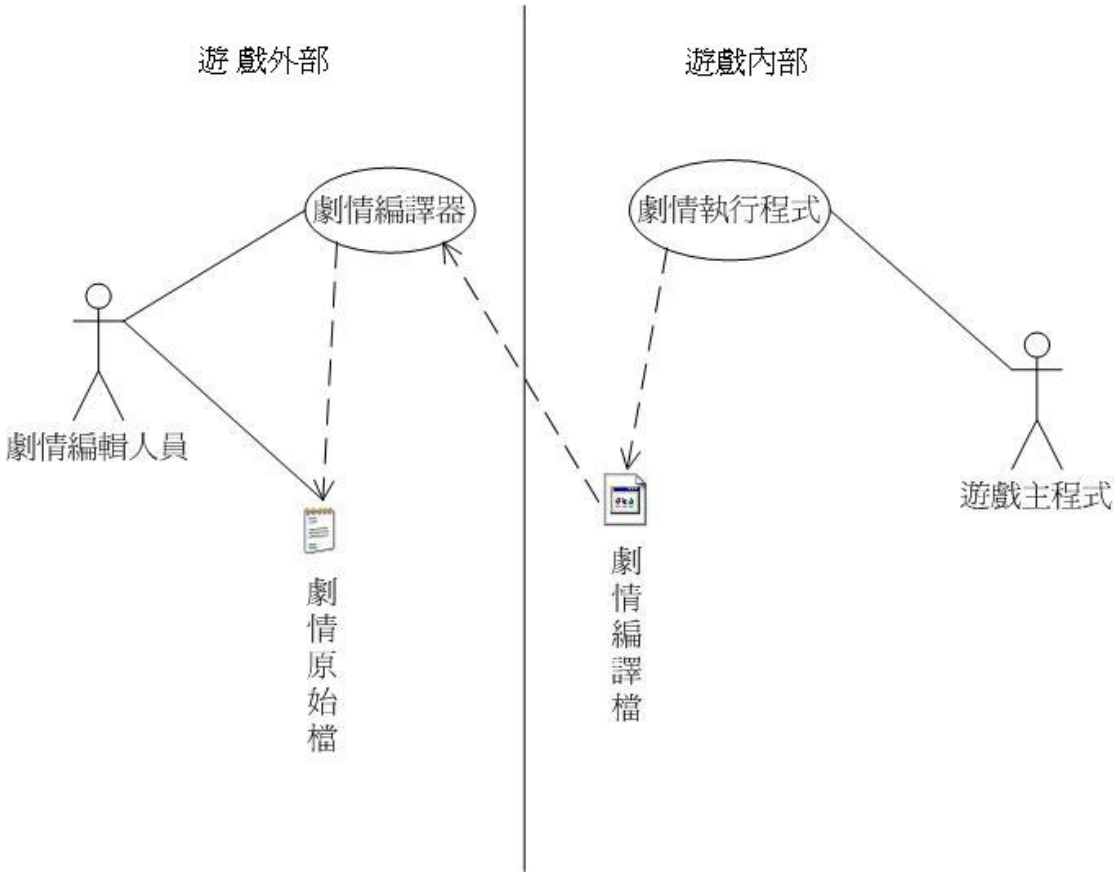

圖 5-76 劇情系統組成元件

一、劇情原始檔

存放未編譯的劇情內容，是給劇情編輯人員用的，是一種以編輯人員較能理解的語法所寫成，之後得經過劇情編譯器編譯過後，才能真正讓遊戲內的劇情執行程式所解讀。

二、劇情編譯器

用來將劇情原始檔編譯成可為劇情執行程式所能解讀的檔案格式。

三、劇情編譯檔

可被劇情執行程式接受的劇情檔案格式。

四、劇情執行程式

解讀劇情編譯檔，依其資訊來對遊戲內部做設定，並提供事件觸發的機制，依其劇情對遊戲內部做改變。

5.11.2 劇情是如何組成？

遊戲中的劇情是由一連串的事件所組成，這些事件可能包括「殺死怪物」及「撿起特定物品」等，玩家要進行某個劇情前，得「觸發」某些特定的條件(也是事件之一)，譬如在殺死魔王之前，得先跟村民談話，才能得知魔王的位置及長相等。以下便對『事件』與『觸發』作出解釋。

一、事件

每當玩家做出某個動作，或完成某件事，就可以稱做一個事件，像是與 NPC 談話是一個事件，滑鼠點選一個物品也是一個事件，不過這樣的行為太過廣泛及頻繁，有必要挑選出我們認為符合我們遊戲需要的即可，於是我們預先定義下列這些事件：

- TALK：與某 NPC 談話
- KILL：殺死某隻怪物
- CLICK：點選某項物品
- USE ：使用某項物品
- OPEN：打開某項物品
- CLOSE：關閉某項物品
- ENTER：進入某區域
- EXIT：離開某區域
- BUY ：跟 NPC 買某項物品
- SELL：跟 NPC 賣某項物品

所以，若不是(滿足)以上預設的動作，程式是不會認為它是事件的。

二、觸發

一旦某動作發生後，如果此動作為預設事件中的一種，接下來便會與劇情進行比對，看是否滿足劇情事件，倘若滿足，則稱此動作「觸發」了劇情事件。

5.11.3 劇情原始檔

存放劇情內容（一般的文字檔），是一種以編輯人員較能理解的語法所寫成。以下會看到原始檔之語法及範例(圖 5-77)。

一、原始檔語法

start ->
　(*root-expression*) *subnodes*
root-expression ->
　root { *actions* }
subnodes ->

*subnode-expression**

subnode-expression ->

subnode *self-id parent-node-id* { *events* }

(*finish-expression*)

finish-expression ->

finish { *actions* }

events ->

*event-expression**

event-expression ->

event target-category target-name (*number*) *semi-colon*

actions ->

*action-expression**

action-expression ->

command target-category target-name (*parameters*) *semi-colon*

event ->

talk | kill | click | use | open | close | enter | exit | buy | sell

command ->

say | move | follow | leave | changemap | set | show | hidden | moveto

target-category ->

player | npc | monster | item | weapon | shield | tool

target-name -> *string*

parameters -> *string**

self-id -> *number*

parent-node-id -> *number*

string ->

*alpha alpha** |

quote (*alpha* | *chinese-word*) (*alpha* | *chinese-word*)* *quote*

quote -> “

semi-colon -> ;
alpha -> [a-zA-Z_]
number -> *digit digit**
digit -> [0-9]
chiness-word -> Big5 碼中文字

二、範例

kill.story - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
root {
        set npc "三姑"  "你知道嗎? 昨天村長的女兒外出購買藥材。"
                        "經過五指山的山腳下，沒想到就沒回來了。";
        set npc "六婆"  "對阿，聽說五指山上最近來一個土霸王。"
                        "外號好像叫做「大呆」的樣子，不知道跟他有"
                        "沒有關係。";
        set npc "村長"  "請你救救我的女兒吧!";
}


subnode 1 0 {
        talk npc "三姑";
        talk npc "六婆";
        talk npc "村長";
}

subnode 2 1 {
        kill monster "大呆";
} finish {
        get item "村長之吻";
}
```

圖 5-77 劇情原始檔範例

5.11.4 劇情編譯器

這是獨立於遊戲程式之外的小程式，目的是用來將劇情原始檔(如圖 5-78)編譯成可為劇情執行程式所能解讀的檔案格式。

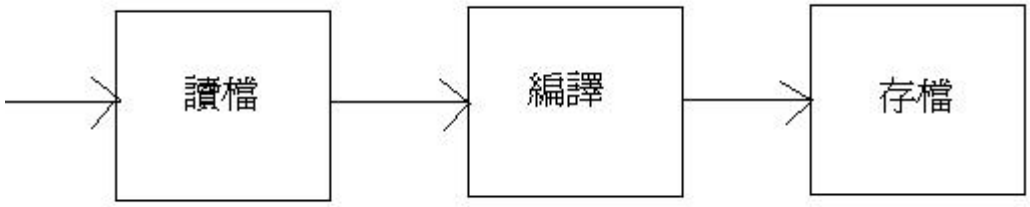

圖 5-78 編譯程序

5.11.5 劇情編譯檔

為可被劇情執行程式所接受的劇情格式檔 (binary 形式)。而為何不直接使用人類易讀的「劇情原始檔」(文字格式)，則是因為是文字格式雖然易讀，但要被程式所了解則需經過解譯，而如果本來決定的語法更改了，那解譯器也需要做修改，且語法在形成階段，修改幅度很大，便會造成解譯器的一再修改，這樣的話，負責整合整個遊戲的編程人員會非常困擾，為解決此問題，只好在原來的文字劇情檔及解譯器之間再多增加一層「中介檔案」，這樣只要這層中介檔案格式定義完善，那之後文字劇情檔語法的修改，便不需去修改遊戲內劇情解譯器的程式，這樣劇情程式的編程人員與主程式編程人員間溝通將可以部分簡化，只是如此一來需要額外增加劇情編譯器的編寫工作。

不過此中介檔案不管是文字形式或 binary 形式，都還是需要經過解譯的，雖然 binary 格式的檔案，直覺上似乎讓人感到難以擴充，但實際上並不會，在一些工作上甚至較為單純，像是不必再做「ASCII 的 '3' 轉換成實際的數字 3」這類工作，且此中介檔並不需被人所理解，更不需人工去編寫，所以最後決定採用 binary 形式。

此格式如下：

```
int : 4 bytes
char: 1 byte

STORY_FILE
|
+--COUNT          int
+--STORY          \ [COUNT]

STORY
|
+--MODEL_NAME
|  |
|  +--STRING     \
+--ROOT
|  |
|  +--COMMAND_COUNT  int
|  +--COMMAND    \ [COMMAND_COUNT]
|
+--NODE_COUNT     int
+--NODE              \ [NODE_COUNT]

NODE
|
+--NODE_CATEGORY int
+--PARENT_ID     int
+--SELF_ID       int
+--EVENT_COUNT       int
+--EVENT          \ [EVENT_COUNT]
+--COMMAND_COUNT int
+--COMMAND        \ [COMMAND_COUNT]
COMMAND
```

```
|
+--COMMAND_NAME      int
+--TARGET_CATEGORY   int
+--TARGET_NAME
|  |
|  +--STRING     \
|
+--ARGUMENT_COUNTint
+--ARGUMENT
   |
   +--STRING     \ [ARGUMENT_COUNT]

EVENT
|
+--EVENT_NAME    int
+--TARGET_CATEGORY  int
+--TARGET_NAME
|  |
|  +--STRING     \
|
+--TRIGGER_COUNT int
+--OWN_COUNT     int
+--OWN           \ [OWN_COUNT]

OWN
|
+--STRING        \
+--COUNT         int

STRING
|
```

```
+--COUNT          int
+--DATA               char [COUNT]
```

5.11.6 劇情執行程式

解讀劇情編譯檔，依其資訊來對遊戲內部做設定，並提供事件觸發的機制，依其劇情對遊戲內部做改變。程式的流程如圖 5-79 所示：

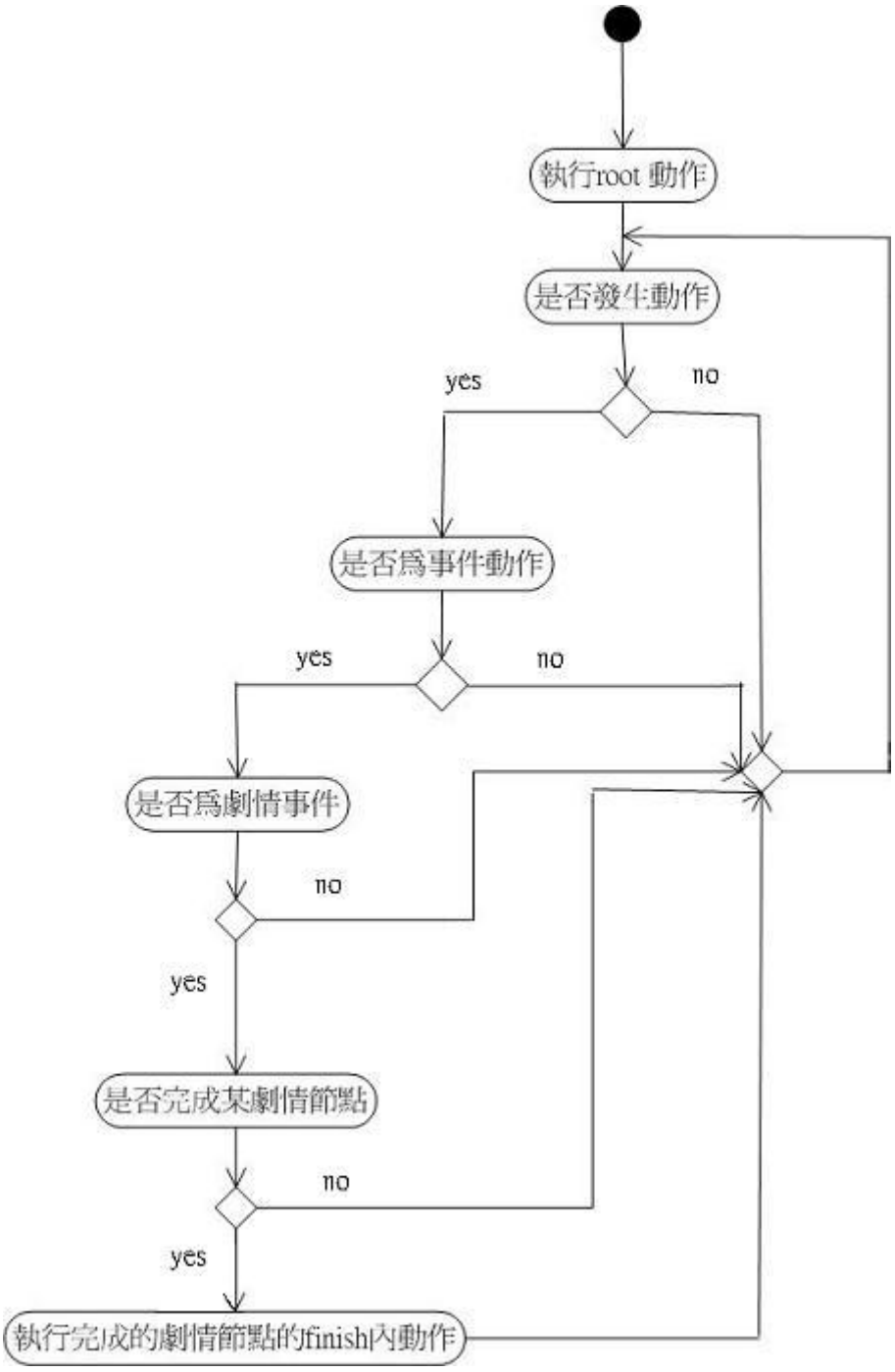

圖 5-79 劇情執行程序活動圖

第六章 心 得 與 結 論

6.1 系統特色與結論

我們根據系統需求，製作出本系統，能夠提供下列具有玩家和玩家之間以及和系統之間較多互動特色之功能：

1. 3D 場景環境
   以 3D 繪圖的方式呈現整個遊戲場景及 360 度全視角轉換，能夠完整觀覽四周場景而不會有所謂 2D 畫面的死角。輔以 3D 物件的擺設，讓場景有下雪及飄下落葉等之特效。
2. 多人連線
   數個玩家(用戶)同時上線，享受和其他玩家共同打怪及冒險的樂趣。
3. 線上聊天交友
   具備對話系統，可以透過對話系統來和朋友聊天或認識新朋友。亦可透過聊天系統來跟 NPC 對話，以開啟劇情任務。
4. 交易系統
   除了可將物品販賣給 NPC 外，亦可透過對話系統來跟其他玩家喊價並交易，可充分體驗當商人的樂趣。
5. 劇情系統
   讓玩家融入遊戲世界、充分享受劇情，以滿足玩家角色扮演的快感。
6. 戰鬥系統
   及時運算處理，確保資料處理快速及正確外，亦能調整升級的難易度，使玩家不會為了練功所苦。

6.2 個人心得

6.2.1 明政

我所負責的部份比較偏向於美工畫面、多媒體效果的顯示。記得當初研究了老半天的 3D 物件載入函式，當欽賢告訴我可以顯示出來的那一瞬間，我的心中的感覺真的是千言萬語也無法形容，不過最麻煩的工作卻隨後接踵而來，那就是要把 3D 物件一個個的做出來。前面說的要注意的一些小細節：比如說各個元件之間要留一點點縫隙，貼圖的時候要注意布滿整個物件等等，只要一點點沒有做好，就會導致整個顯示結果不正確。除了一開始的設計 3D 物件很有趣之外，這真的是一項漫長又枯燥的工作，一但過程中沒有做好（複雜的 3D 物件常常如此），顯示不正常，往往又要回頭一個面一個面的檢查貼圖有沒有設定好，那時真的是一想到就會瘋掉，甚至覺得整個物件再重做會比較快。以上這些是我所負責的系統裡，比較讓我印象深刻的部份，其他許多細小的地方就不再詳述。

另外當然要先感謝我們的指導老師：楊東麟老師。老師雖然可能不常接觸遊戲，但是還是願意帶領我們這一組，從零開始一步步往前走。我覺得老師給我們最大的幫助，就是給我們訂下一個前進的目標。開發的部分雖然老師只在旁提供給我們意見，不在系統上做任何協助，但是他引導著我們的進度往目標邁進，讓我們隨時隨地都有一個很明確的方向指引，這一點真的是我們背後的一股莫大助力。另外當然還要感謝我的組員們，感謝欣宗常在後面督促我們幾個，雖然有時後說話會比較直接，但我相信這對我們絕對是正面的幫助，另外欣宗還完成了許多項細部的系統，讓程式得以順利進行。欽賢負責的部份可以說是整個程式的 70%吧? 如果沒有欽賢，說真的我們這一組不會有開始。當我們還在研究架構要怎麼設定的時候，突然之間基本的系統就已經出來了? 這真的不是人辦的到的阿，真是怪物…。志信則負責了系統裡最重要的幾個功能，雖然有時候看他趕程式趕的焦頭爛額，我們大家都不禁爲他捏一把冷汗，但是最後都還是虛驚一場，那

真的是很有趣的一段日子。

經過了一年多的努力，整個專題總算是完美的劃下一個句點。雖然整個遊戲系統跟市面上的遊戲比起來，品質還是有一段不小的差距，但是我覺得我們已經盡力把它做到最好、最完整，光是這一點我覺得很值得自豪了。畢竟從一開始的理念、一步步規劃出系統的架構、不斷的新增功能、及修改畫面等，我們之間經歷過了無數次的討論、測試、再討論、及再測試，才慢慢的形成了今天大家所看到的這整個系統，裡面的每一個角落都是我們的心血結晶。記得我們每一次在小組討論的時候，桌上總是堆著到處散佈的紙，上面畫滿了我們討論時的筆記，現在回想起來，那段日子雖然辛苦，卻是我大學生涯裡最棒的時光。有時爲了某一個系統要怎樣運作，玩起來才會比較好玩，或者是畫面要怎麼顯示，看起來才會比較舒服之類的問題，爭論了好久；現在想一下那的確是很不錯的一段日子，畢竟大家都是真心的想要把遊戲作好的，我真的很高興能夠有這樣的一個經歷跟回憶，讓我學到了許多東西，也成長了不少。

6.2.2 志信

這次的專題是個很難得的經驗，為了一個遊戲的產生，所需要的技術，沒想到是如此的廣泛，每個不同的工作，都是個挑戰。而對所負責的工作越了解後，越發覺並不如想像的簡單，其中最讓人感到痛苦的，就是當工作陷入瓶頸時，那種說不出來的焦燥感，最讓人感到興奮的便是跨過這個瓶頸時的時刻，不過這種興奮通常持續不了太久，下個難題不久就出現了。在工作進行中，偶會發現一些東西原來前人已經有這些構想，甚至做的更好和想的更多，這時對這些前人的前瞻性及能力不禁感到敬佩。

一個專題並不只有寫程式，還包括企劃及系統分析等，在還沒接觸過前，總認為那些不過是簡單的紙上作業罷了，不過我想在經過這次之後，我再也不敢這麼想了，如果不是大家的合作幫忙，真不敢想

像現在的結果會是如何。而最後的成果雖然不如一開始時所想的那般遠大，但這過程是最寶貴的，這些經驗會讓自己在日後的學習上絕對有不小的幫助，也非常謝謝其它組員們在我遇到瓶頸時的幫助，讓我不至於中途放棄，真的相當感謝。

6.2.3 欽賢

程式部分我所負責是整個程式的整合規劃。在這之前所寫的程式都是一或兩百行的小程式，最多也只是一千多行，而且也不用太顧慮效能的問題，只要結果是對的，就沒多大的問題。但是遊戲的程式就必須考慮流暢度，以人類的視覺一般只要能做到一秒三十張，就不會有延遲的感覺，而這個數字就是我設計程式的標準，無形中也增加了不少的難度。而在整合的過程中，也是一堆大大小小的問題，比如說，組員們設計的程式都沒問題，但合到主程式上做整合的時後，卻出現奇怪的錯，常常會有記憶體資料錯誤的問題，而這種錯不是程式設計語法錯誤，是邏輯錯誤，這種錯很難去找出問題所在，也因此重寫了不少次，包括現在完成的程式，我也不敢保証不會突然掛掉，經過這次屬於較大型的程式設計經歷之後，覺得成長不少。

經過漫長的這一年，已經漸漸地接近尾聲了。回想過去的種種，對我們而言都是很好的回憶和經驗。從一開始我們決定題目的時後，大家都抱著想挑戰這個題目的想法，可是那時後對於遊戲這方面的經練驗可說是零，又擔心會做不出來，而一度想換題目。在那段混亂的時期，我們的指導老師，楊東麟老師那時剛好說出有一個遊戲的題目，問我們有沒有興趣，這真是一種巧合，大家就都很興奮決定了。而在那個時後，很感謝老師給我們很多的幫助，也常常鼓勵我們，而老師積極的態度更讓我們感動，總是盡快地想辦法解決了學生的煩惱和問題，老師這樣的付出，是我們做下去的動力，再次感謝老師。

再來就是關於整個遊戲跟報告的完成，在這過程中，因為有其他組員的互相幫忙討論，和共同努力渡過不少夜生活，這一切回想起來

都是很棒的，如果說沒有你們，也不會有今日的專題，這是屬於我們共同努力的成果。感謝大家。

6.2.4 欣宗

呼~終於告一個段落了！回想過去一年多以來的心情，著實是不好過，現在，終於如釋重負。如果說，成長必經的途徑是付出與努力，這我絕對同意。

從系統的分析、設計、分工、實作、整合、及到測試，每一個階段都有其不一樣的背景知識需要學習，很慶幸地，每一個階段我都參與且努力過。我們需要改善的地方還很多，一些系統應有的功能也並沒完全達成當初的計畫，有時往往因為技術上的困難或自己的懶惰，而對系統要求作出讓步，因此，我們還是有待磨練的。

在製作專題的這段過程中，讓我體驗最深刻的，除了專業技術，莫過於團隊合作了。由於每個人的素養、工作態度皆不相同，因此，在合作團隊中了解並扮演好自己的角色，對夥伴及工作進度而言，是最重要的。相信如此深刻的體認，對我往後的作事態度有絕對的幫助。

除此之外，我還要特別感謝我們的指導老師—楊東麟，一直以來，老師總是給于我們最大的包容跟最多的協助，並常鼓勵我們放手去作和不要怕失敗。而每當系統的進度有所延遲，或有困難解決不了時，老師也會試著了解問題，並引導我們解決不要放棄。

最後，我還要跟組員們說聲謝謝，大家辛苦了。沒有大家的參與付出，就沒有今天的成果，且不管最後結果如何，大家都可以為自己感到驕傲的。

6.3 未來展望

目前這個系統，離當初所規劃的完整方案還差一段距離，在我們所期望的系統中，希望能夠做到:

一、創造多個遊戲世界

我們希望能創造出多個遊戲世界來，每個遊戲世界可以是互無相關的，比如說，一個是中國神話故事，而另一個是希臘神話世界，兩者有其各自的劇情發展。但遊戲中金錢和特殊寶物是可以相通的，如能將在中國世界辛苦賺來的錢帶到希臘神話世界中使用。

二、更多的角色選擇

在人物角色設計上，能有更多的變化性，如可分男性、女性的角色，或則有職業上的差別。像是戰士，以力量取勝，最適合肉搏戰；魔法師使用自然的力量，進行遠距離方式攻擊；弓手使用弓箭進行遠距離攻擊，但他有敏傑的身手，進行閃躲。在設計上，有美工技術上的困難，因此，未來需加強在美工方面的相關繪圖技術、或相關人員。

三、劇情進行方式

這一方面，由於系統所提供的基本服務是多人連線，因此，未來將提供更完整的劇情，運用合作的原則，也就是玩家邀請其他人組隊以共同冒險，來增加遊戲的豐富性。如玩家可尋找各種不同屬性的角色，如戰士、魔法師，和弓手，互相合作產生多方面的攻擊模式，以打倒大魔王。

四、多功能的交易系統

在交易系統中，將會提供玩家開店的功能，和其他玩家進行交易活動，甚至可以透過網頁開店，不需要進入遊戲，就能知道買賣的情形，亦即能在任何有網路的地方也能經營自己的商店。

五、AI 人工智慧系統

AI 系統須再增加更多的行為反應，如加入一些野獸間互相打鬥的行為，或是加入生態觀念，如怪物也是需要靠繁殖的方式增加族群，怪與怪之間也會有競爭的行為、學習的能力等。在這方面 AI 系統所要改善的空間還很大，也是相當困難的部分。

六、連線品質方面

透過壓縮加密的方式，減少封包流量及安全性。

七、程式結構設計方面

整個遊戲架構上，缺少完整性的規劃，隱藏了不少的問題，而且也影響到遊戲效能的問題，所以未來將進行系統最佳化。

八、資料備份

目前是透過檔案系統，進行玩家資料的備份，未來將透過資料庫系統，提供更快速及方便的存取，而且這樣更能和網頁做結合，以進行多元化的運作方式。

九、遊戲編輯器

在遊戲編輯器這一方面，現在是透過設定檔來修改 3D 場景、人物屬性、及劇情等之設定，但仍必須直接改設定檔。所以需要一個遊戲編輯器(介面)，能夠立即看到遊戲畫面，輕易地佈置場景，和設定劇情。這個工具在未來是必要的。

總之，想當初我們在規劃整個遊戲系統時，是想改變現在線上遊戲的進行模式。現在進行的模式多半重點在於打怪、練功、和撿寶，和玩家之間也僅只於互相比較等級高低和所擁有的高級裝備。甚至還有玩家以真實貨幣來換取虛擬寶物，更為了虛擬寶物大打出手，這樣的價值觀在我們眼裡著實感到不可思議。

而在單機遊戲中，有一款人人樂道的 RPG 遊戲，仙劍奇俠傳，在

許多玩家心目中，它是國內經典 RPG 遊戲，其中，最吸引玩家的，就是它感人的故事劇情搭以背景音樂。和現在的線上遊戲相比，相差甚多。

如果能在豐富劇情加上線上多人連線，能和親朋好友共同來體驗遊戲所創造出來的世界，這才是我們最後的目的。但要達到這些目標我們還需要更多的時間和技術來完成，這次的專題只是我們的開頭及嘗試。有了這個開始，相信在未來我們有更多的信心來完成我們所規劃的最終版本。

參 考 資 料

[1]侯俊傑, ”深入淺出 MFC 程式設計”, 松崗電腦圖書資料股份有限公司, 96 年 10 月

[2][康吉爾]James L . Conger 原著/ 李明台 譯, ”Windows API Bible 中譯本”, 台北市:松崗, 1993

[3]Jim Adams, ”2D/3D RPG 角色扮演遊戲程式設計-使用 Direct X”, 博碩文化股份有限公司, 2003/02/21

[4]Jason Kolb 原著/ 劉登榮 編, ” DirectX 發展手冊”, 松格資訊有限公司, 1997/10

[5]Richard S. Wright, Jr. Michael Sweet 著/ 大新資訊 譯, ”OpenGL 超級手冊第二版”, 碁峰資訊股份有限公司, 2002/5

[6]Gamma, Johnson, Helm, Vissides 著/ 葉秉哲 譯, ”物件導向設計模式”, 培生, pp. 221-234, April 2001

[7]Charles Petzold 著/ 余虛學 譯, ” Programming Windows 程式開發設計指南 5th”, Microsoft Press/ 華彩軟體代理, 2001

[8]Nicolai M. Josuttis 著/ 侯捷, 孟岩 譯, ”C++標準函式庫—自修教本與參考工具”, 基峯資訊股份有限公司, 2002/6

[9]Gary B. Shelly, Thomas J. Cashman, Harry J. Rosenblatt 著/ 林國平, 吳宗杉 譯, ”System Analysis and Design系統分析與設計”, pp.169-232,東華書局, 2001年8月

[10]周斯畏 編著, ”物件導向系統分析與設計使用 UML 與 C++”, 全華科技圖書股份有限公司, 92 年 5 月

[11]Roger S. Pressman 著/ 金子葳, 洪秀朋 譯, ”軟體工程—實務專家作法,5th”, 儒林圖書公司, 2001 年 2 月

[12]遊戲中 2D 人物的動作圖由此站提供
http://www.reinerstileset.4players.de:1059/englisch.htm

[13]http://www.gametutorials.com

[14]http://cgd.pages.com.cn/cgd/index.html

[15]http://programmer.eforum2000.net/pc2020v5/

[16]http://nehe.gamedev.net/

[17]http://intra.yuanta.com.tw/PagesA2/hot_issue/9108gam.html
[18]http://eteacher.edu.tw/game_1.htm#3
[19]http://www.epochtimes.com/b5/3/8/26/n365126.htm
[20]http://www.nettrade.com.tw/stock/a/01-08-03a.htm
[21]http://www12.brinkster.com/losi/hot1.htm
[22]http://home.kimo.com.tw/money-clubs/stork099.htm
[23]http://investintaiwan.nat.gov.tw/moea-web/InvestOpps/Type/2-2/index-c.htm
[24]http://ro.gameflier.com/main.asp
[25]http://www.gamebase.com.tw
[26]黃麒榮 先生之『遊戲ｅ觀念—網路遊戲製作與發展』投影片

附 錄 A

建立資料處理模型-使用 DFD

一、資料流程圖

資料流程圖(DFD，Data Flow Diagram)表明資料如何流經一個資訊系統，但是卻不顯現出程式的邏輯或處理的步驟。DFD 是一種邏輯模型，展示系統做些什麼，而不展示他如何做這些事情。這種區別很重要，因為在這時如果著重於執行的問題，將會限制你尋找最佳的系統設計。

二、資料流程圖的符號

資料流程圖使用四種基本符號來代表處理工作、資料流、資料儲存、和外部實體。DFD 符號有一些不同的版本，但是他們全都被使用在同一個目的。有一種為 Gane&Sarson 符號集(Gane and Sarson symbol set)的流行版，另一種流行的符號集是 Yourdon 符號集(Yourdon symbol set)。

- 處理工作符號

一個處理工作(process)接受輸入資料而產生不同的內容及形式的輸出。以一個黑盒子(black box)的方式出現，其中處理工作的輸入、輸出，和一般功能可以看到，但是其內部細節則沒有顯示出來。黑盒子方法以一系統處理工作圖形來表示一個資訊系統。你可以展開每一個處理工作而顯示其系統，又不致使得系統的整體景像混雜凌亂。

- 資料流符號

資料流(data flow)是資料從資訊系的一個部分移動到另一個部

分的路線。DFD 中的一個資料流代表一項或多項資料。自發產生、黑洞，和灰洞都是在 DFD 邏輯上不可能存在的，因為每一個處理工作必須根據表示為流入的資料流的輸入資料來運作，並且產生以流出資料流表示出來的輸出資料。

- 資料儲存符號

資料儲存(data store 或 data repository)，用來在 DFD 中表示以後還有一個或多個處理工作需要用到某些資料，而系統必須將資料保存的情況。資料儲存的實體特微並不重要，因為你僅僅關心其邏輯模型。此外資料儲存的時間長度也不重要，它可以是幾秒鐘，例如在交易進行的一瞬間，它也可以是數個月，例如當資料累積供年終處理時。重要的是，某個處理工作需要在以後的某個時間存取這種資料。

- 外部實體符號

外部實體是提供資料給系統或接受系統輸出的一個人、部門、外部組識，或其他資訊系統。外部實體又稱端點(terminators)，因為他們是資料的來源或是最終的去處。

三、全景圖

建立一個 DFD 的第一步是畫出一個全景圖(context diagram)，是表示系統的界限及範圍一個最高階層的例子。為了畫出全景圖，你把代表整個資訊系統的單一個處理工作符號置於一張紙的中心，這個符號代表整個資料系統，而稱它為「處理工作 0」。然後圍繞著紙的周圍邊緣畫出所有的外部實體，並用資料流把這些實體與中心處理工作正確地連接起來。在全景圖中，你不顯示任何資料儲存，因為資料儲存在系統的內部。

四、圖 0

全景圖提供一個資訊系統最高階層的整體概觀，其中只有單獨一個像黑箱一般的處理工作符號。你可以產生 DFD 的圖 0 來展現更多的細節。是放大全景圖而列出一些主要處理工作、資料流、和資料儲存。

五、較低階層圖形

當需要一些較低階層的圖形，也被稱為子圖(child diagrams)，來表示細節時，對它們的分層和平衡是必須的。分層(leveling)是畫出一系列愈來愈詳細的圖形過程，直至達到所需要的詳細程度為止。平衡(balancing)則是保持這整系列圖形之間的一致性，包括輸入和輸出資料流、資料定義和處理工作說明。

建立物件模型-使用 UML

一、定義

UML 是一種視覺化的塑模語言(Modeling Language)及通用的(general)塑模語言，主要是由一些語義及表示法(符號或圖形)組成。UML 所描述的是真實系統的藍圖(blueprint)，讓系統設計者與使用者對系統架構有全面的認知，並可瀏覽系統的配置及維護等工作。它讓系統建構者從不同的觀點去建立系統的各種模型，以供不同階段或不同職務的系統建置人員使用。

二、目的

UML 經過完整的設計及規劃，能提供以下特質

- 提供標準化環境

  其最主要目的是提供一套共通的、單一的標準化環境。

- 簡單且易了解

  盡量以最簡單、容易明瞭且又可對系統作完整的描述，來建立系統模型，以達使用方便且易上手的目的。

- 核心化

  較易掌握問題之關鍵。當系統需求更新時，較易由改善若干的核心功能，而迅速的完成系統升級的目標。

三、適用範圍

UML 可以用來塑模系統，其應用範圍很廣，常見的有：資訊系統、分散式系統、技術系統(Technical System)、及嵌入式及時系統(Embedded Real-Time)等。其特色是在系統設計之初，便以概觀的方式來看系統的整個流程圖，先將最主要的大綱表示出來，讓設計者可以一眼就明瞭整個系統所有表達的概念，爾後再進一步的研究其中細

節，逐漸探索到系統的細項部份。UML 的模型觀點，主要取決不同使用者是用那種觀點來檢視系統的結果，以及每個觀點的設計考量為何。在不同的觀點中，使用不同的圖形來表示系統所要表達的意義。如下表所示。

表 A-1 UML 各個觀點所定義之圖形、功能

觀點(view)	**圖形(diagram)**	**功能及目的**
使用者模型觀點 (user model view)	使用案例圖 (use case diagram)	描述系統功能性的需求，找出 use case 與 actor，並藉定系統範圍。
結構模型觀點 (architecture model view)	類別圖 (class diagram)	表示物件類別之間的關聯性。
行為模型觀點 (behavioral model view)	循序圖 (sequential-diagram) 合作圖 (collaboration-diagram)	表示物件之間動態的溝通模式。
	狀態圖 (state diagram)	表示物件狀態的變化。
	活動圖 (activity diagram)	表示業務流程或一類型中單一操作的邏輯流程。
實作模型觀點 (implementation model view)	元件圖 (component diagram)	表模組與元件間的關聯。
環境模型觀點 (environment model view)	佈署圖 (deployment diagram)	表程序(process)和執行緒對應到主機或裝置的實際狀況。

以下便介紹開發過程中將常使用到的兩種圖形：類別圖及活動圖。

一、類別圖

類別圖是模型建構技術的重要核心，用來描述一個系統的靜態結構以及存在類別之間各樣的靜態關係(static relationships)。主要的靜態關係包括以下三種：

1)關聯(association)

關聯讓物件能夠彼此交流，是描述類別之間的一個具體關係。關聯的符號是以一直線表示，關聯的名稱則標示在直線上，例如學生與老師之間關聯性(圖 A-1)。

圖 A-1 學生與老師之間關聯性

2)父子型別(supertype/subtype)

在 UML 中以「空心箭號」表示繼承關係，以箭號指向父類別。父子類別的意義表示所有子類別的實例都是父類別的實例。子類別可以全部或是部份繼承父類別的結構及運作，例如人的繼承圖(圖 A-2)。

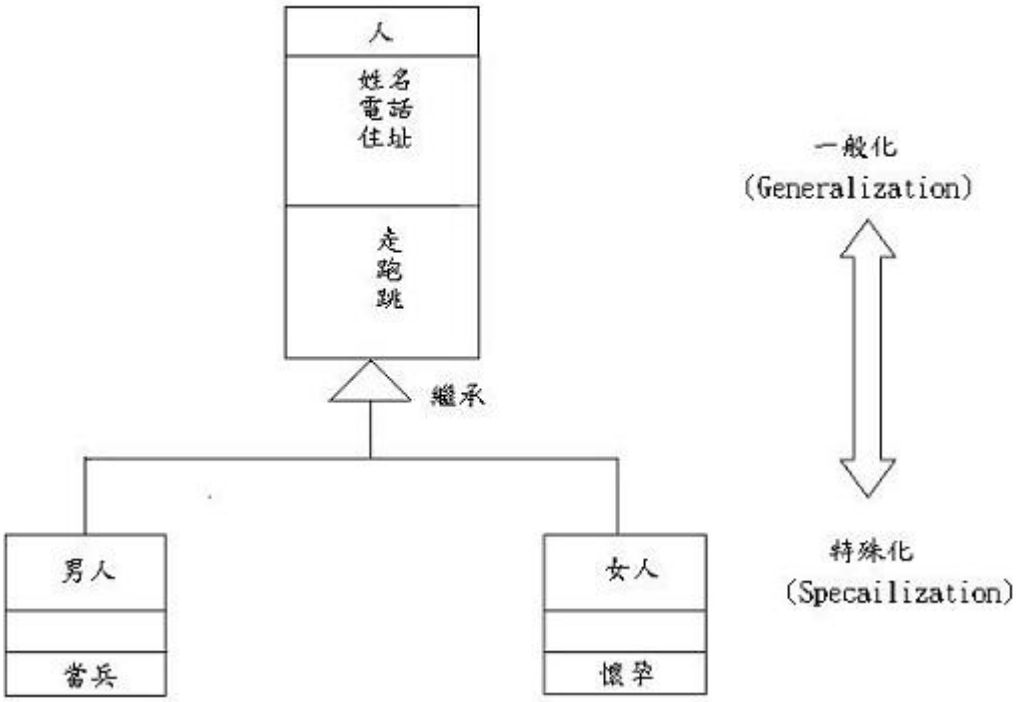

圖 A-2 人的繼承

3)合成(aggregation)

合成關係是指定某特定物件是由那些物件組成的，是表示物件間有「a-part-of」的關係(如 A-3)。

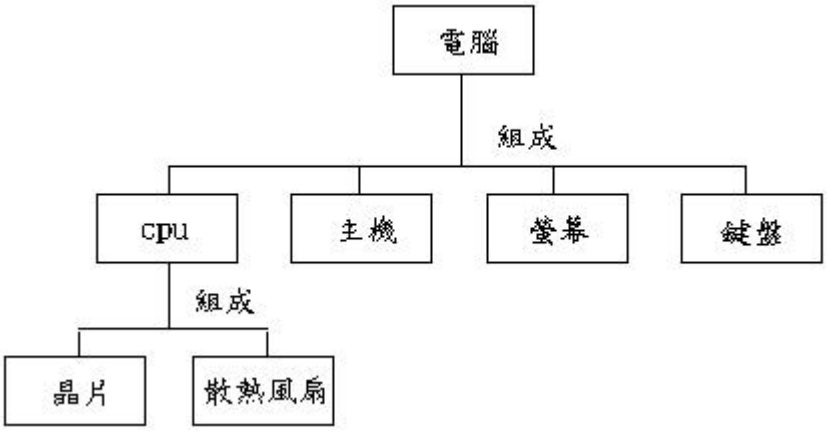

圖 A-3 電腦合成圖

類別圖著重在系統如何建立此結構，而不是它的行為，其主要是用來將一群有相同特質的概念或實體模組化。類別通常是以一個矩形

來表示，如圖 A-4 所示。

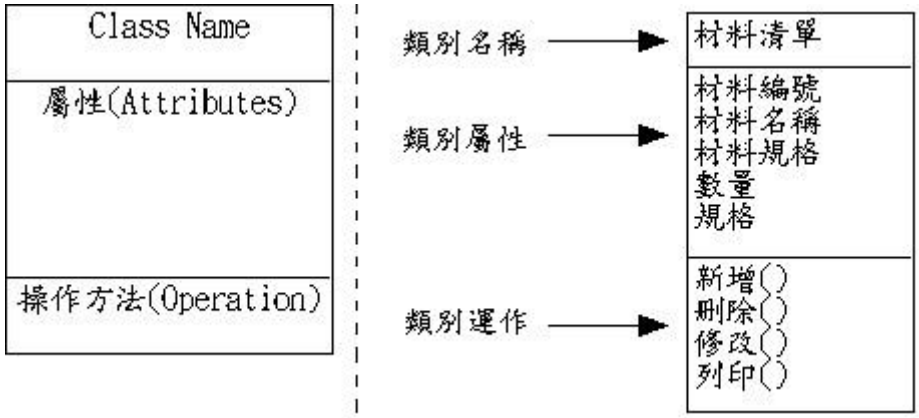

圖 A-4 類別架構圖與範例(材料清單類別)

二、活動圖

UML 中的活動圖是一個多工的流程圖，透過繪製活動圖不但可以塑模工作流程，同時可以做為將來實作工作流程的類別或方法。

活動圖主要是用來描述活動（activity）發生的順序，就觀念（sense）上來說，它有點像是流程圖（flow chart），不過它和流程圖最不同的地方就是：流程圖通常被限制只能作順序性的處理，然而活動圖卻允許平行（parallel）處理。而除了允許平行處理之外，它同時也支援條件式(conditional)行為。其組成元素如圖 A-5 所示：

編號	圖示	意義	編號	圖示	意義
(1)	name	活動	(2)	[條件b] [條件a]	分支
(3)		合併	(4)		分叉
(5)		會合	(6)	*	動態並行
(7)		行動流	(8)		物件流

圖 A-5 活動圖的元素

使用活動圖可達到以下的目的：

- 描述內部行為：利用圖形描述出運算的內部行為。
- 描述平行活動：能描述出平行的活動。
- 描述各類型活動。
- 呈現一般性功能：他們能夠將系統內部的一般性功能給呈現出來。
- 方便規範書撰寫：當你在撰寫規範書時會使用到活動圖。

由圖 A-6 可知，條件式的行為可藉由分支（branches）和合併（merges）來顯示；而平行行為則可藉由分叉（fork）和會合（join）來顯示。

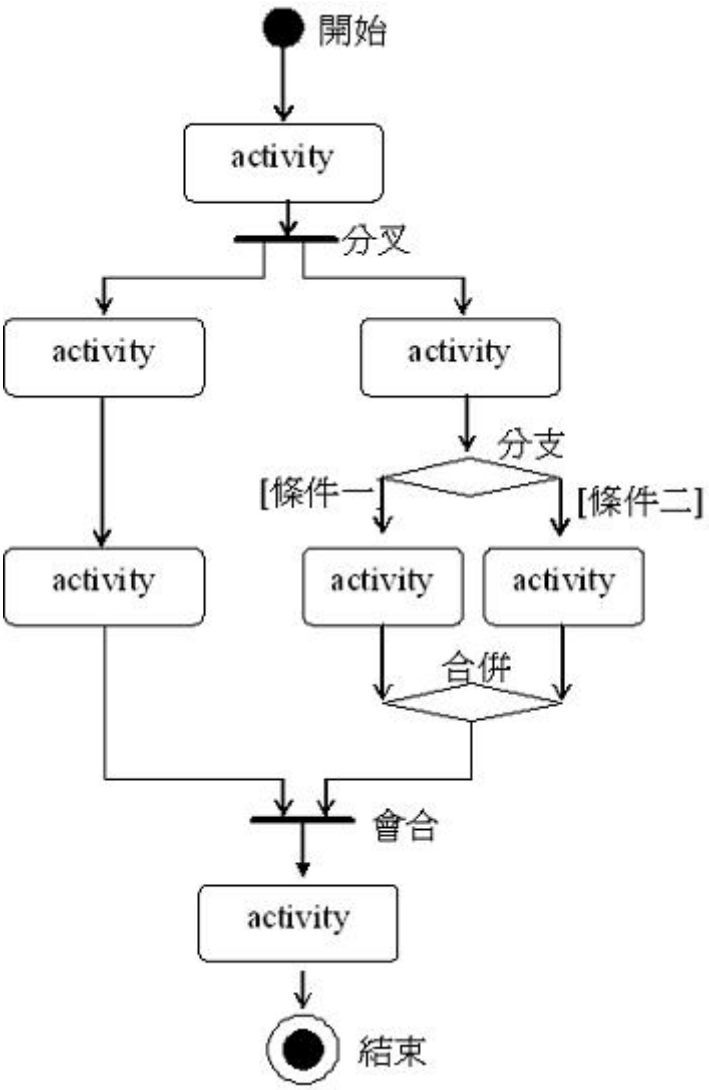

圖 A-6 活動圖架構

附 錄 B

系統開發模式的研究

一、瀑布模型(waterfall model)

瀑布模型(waterfall model)，亦即線性序列(linear sequential)模型(圖 B-1)，其為軟體開發建議了一個系統的、序列的方法，主要包含以下六階段：

1)調查階段(Survey)：又稱初步分析，以產出可行性報告(系統建議書)為目的。主要工作有：

- 定義系統任務及目標。
- 確認新系統的功能和需求分析。
- 確認系統限制級風險。
- 對所有可行方案進行可行性分析及成本效益分析，找出最佳方案。

2)需求分析(Requirement analysis)：此階段重點在於了解系統需要什麼，以產出軟體需求規範書為目的。主要工作有：

- 功能需求。
- 人機介面需求。
- 軟硬體需求。
- 系統配置需求。

3)軟體設計(Software design)
4)編碼(Coding)
5)測試(Testing)
6)維護(Maintenance)

其特性為：

- 整個生命週期有定義清楚的階段，且在每一階段完成後都會有嚴謹的文件產生。
- 階段轉移具循序性，即上一階段完成後才能進行下一階段的工作。
- 使用者只在調查、需求分析、及測試三個階段測試。

線性序列模型是使用最久也是在軟體工程中使用最廣泛的規範。但使用此種模型存在著以下問題：

- 某一階段工作無法如期完成時，將導致後續階段的所有停頓。
- 客戶經常難以將需求明確表現出來，這卻是此種模型所需的。
- 因以循序性的方式進行階段轉移，導致系統在沒開發完成前看不到結果。

因此它可能只適用於低風險、需求變動小又可清楚表達的專案。

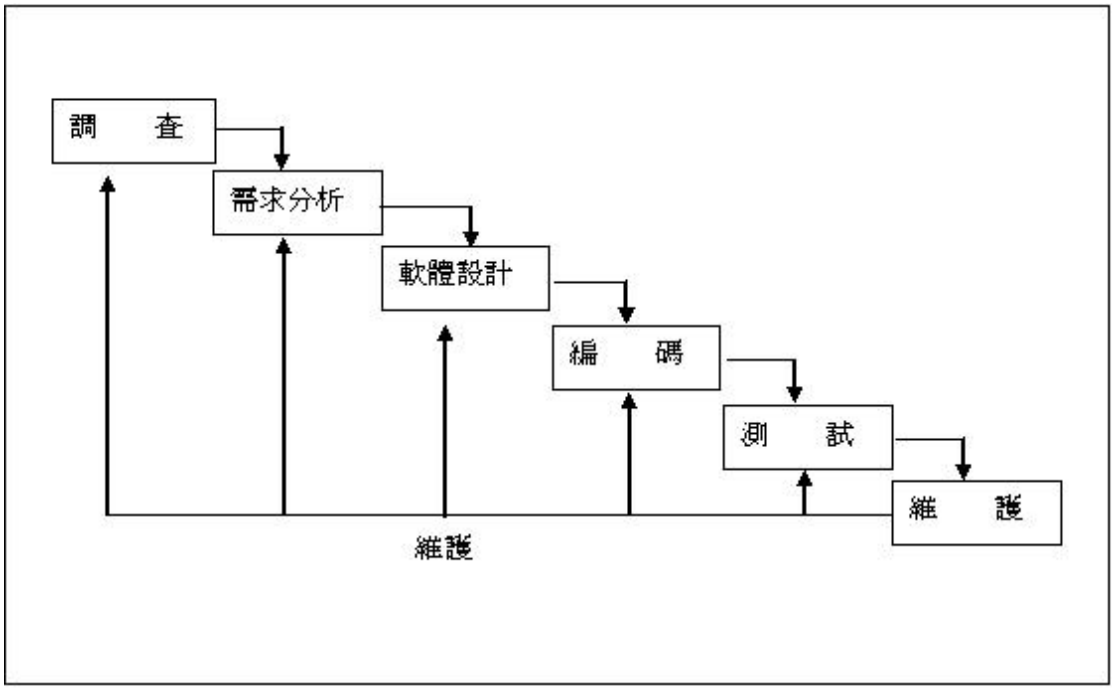

圖 B-1 瀑布模型

二、原型模型(prototyping model)

通常，客戶只會定義出一組軟體所要達成的目標，而不會確認詳細的輸入、處理與輸出需求。有時候，開發者可能不確定一個演算法的效率、作業系統的適用性或人機互動應有的型式等。在此情況下，原型模型便可能提供最好的方法。

原型模型開始於需求的蒐集。開發者和客戶會面，定義整個軟體目標，確認需求是否已明確了解，並描繪更進一步的定義。這時便產生了『快速設計』。快速設計的焦點放在軟體中客戶/使用者可以看到的部份，亦即導出了原型的架構。在理想狀況下，原型可視為一個確認軟體需求的機制。若建立了原型，開發者可以利用存在的程式區塊或應用工具以使工作中的程式更快速的產生出來(圖 B-2)。

原型可以被看成第一個系統，但它可能遇到以下問題：

- 因缺乏整體規劃、分析、與設計，靠著不斷快速修改原型，難以

顧及品質及長程的可維護性。

- 開發者為了使原型能快速工作而對實行部份作出讓步。結果可能使用了不合適的作業系統或程式語言，只因為它是熟悉且便於使用的。
- 使用者必須時時參與。

雖然會出現上述問題，但此模型仍是軟體工程中有效率的規範。關鍵在於快速的規劃、分析、與設計，並建立一個系統雛型，經由與使用者作不斷的評估與修正，可以辨別使用者的動態功能需求，或將工作雛型評估修正成最終軟體。

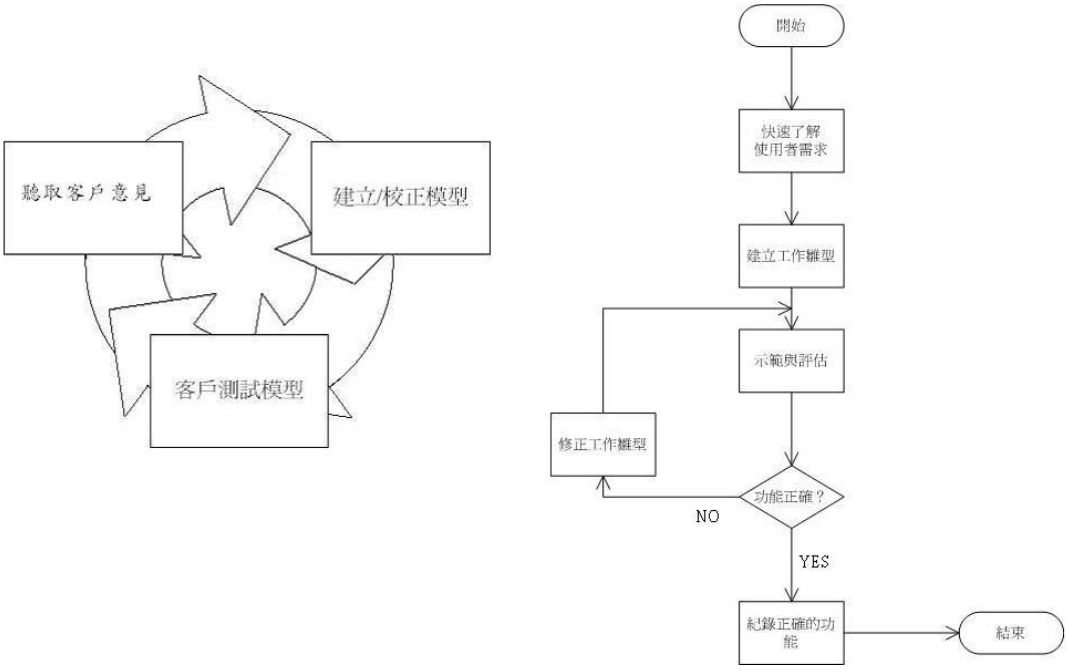

圖 B-2 原型理念及其工作流程

三、螺旋模型(spiral model)

螺旋模型利用原型方法的特點來降低風險，而同時又保持瀑布模型中步步為營的方法，也就是將原型的性質和瀑布模型的控制及系統觀點結合在一起的進化軟體模型。在早期的重複過程中，增量版可能是紙上模型或原型，但在較晚的重複過程中，則會逐漸建立出完整版的系統。

在進化過程開始時，開發小組開始於核心以順時針方向沿著螺旋型逐漸向外移動。當繞第一圈時，形成產品規格說明的開發，隨後被用來開發原型，並逐漸累進開發成更複雜的軟體版本、越繞向外步越接近實際系統。每一圈螺旋型週期的主要內容有三到六種任務：客戶溝通、計畫、風險分析、實作工程、建構及釋出、客戶評估(圖 B-3)。

螺旋模型是一種開發大量系統及軟體時很實際的方法，它的生命不是在交付軟體時即結束而是持續到軟體不再被使用為止。但跟其他模型一樣，它並非萬能。它可能難以說服客戶(尤其以合約的方式)這種進化的方法是可以控制的。它需要考慮風險評估的專門意見，若有一種風險被忽視便極可能導致整個專案的失敗。

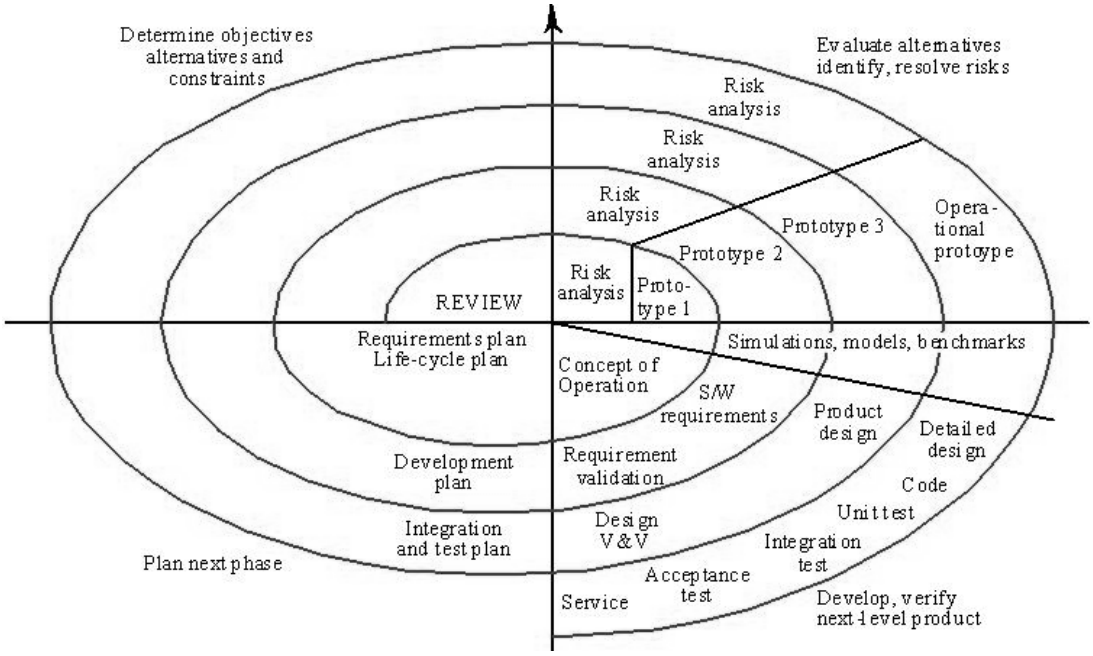

圖 B-3 螺旋模型

四、同步開發模型(concurrent development model)

在軟體開發週期中，如何有效地縮短開發時間是很重要的。同步模式主要是基於三個構想來達到時程縮短的目標：多個團隊同時開發、資訊同步、及整合性的管理系統(圖 B-4)。而其主要開發步驟為：

1)首先將每一版本的工作分成數個功能組，功能組是一個或多個功能的組合。

2)將功能組的工作分派給數個團隊平行開發。

3)當同一版本的功能組都完成了開發之後，便交給獨立的團隊進行整合、測試，開發團隊的人力則可進行下一版本的開發。

4)同理，當整合及測試團隊完成了一個版本的工作後，便可進行下一版本的整合和測試。

優點：開發時間的縮短可提高產品的競爭力

缺點：

- 緊湊的步驟及資訊溝通的頻繁，使得專案管理的複雜度大大提高。
- 人力成本也相對提高。
- 若沒有良好的工具及管理方法，則不易達成目標。

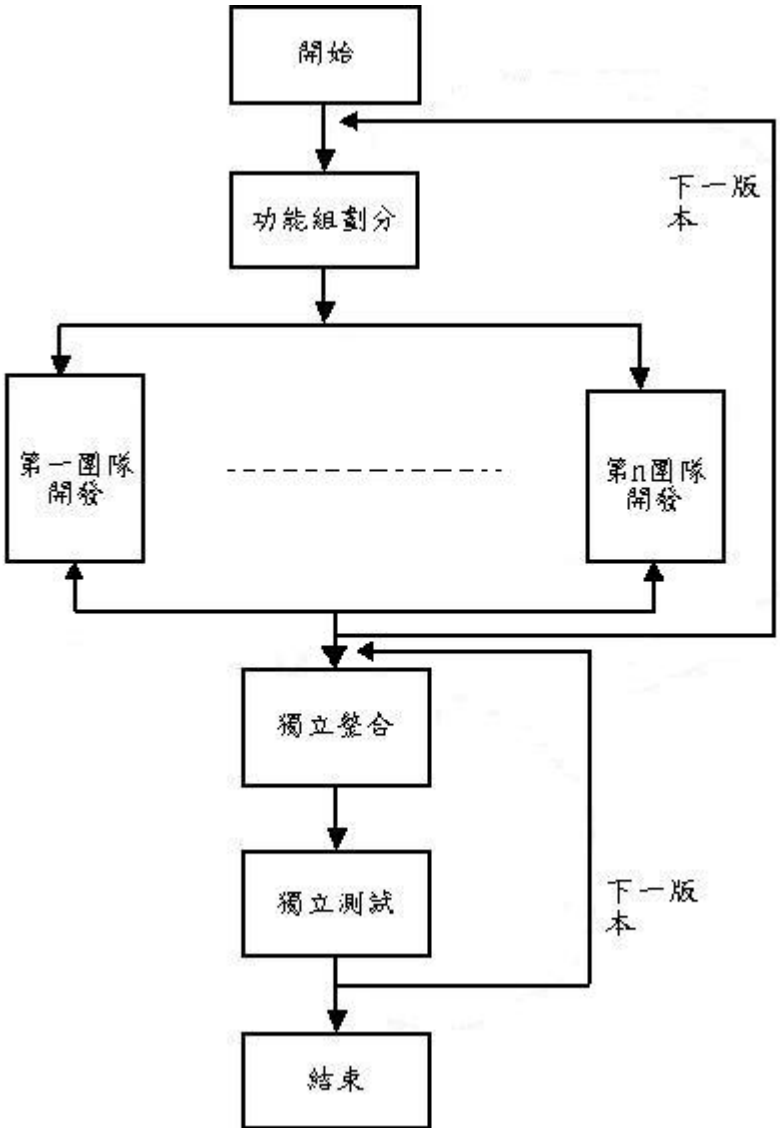

圖 B-4 同步模式執行程序

附 錄 C

DirectX 功能與元件之研究

一、DirectX 的由來

在 Windows3.1 之前，MS-DOS 一直是 AT 的主要作業系統。但是，從 Windows3.1 的 GUI 圖形介面登場以後，就開始有越來越多的個人電腦使用者開始使用 Windows3.1。不過，Windows3.1 的聲音處理一次只能播放一個音，圖形處理能力也很薄弱，是非常不適合電腦遊戲的作業系統。因此當 CD-ROM 問世的時候，Windows3.1 還是不被人使用，大家還是把電腦遊戲設計在 MS-DOS 底下。後來，MicroSoft 在 94 年推出了 WinG 和 WAVEMix；WinG 擁有較高速的圖形處理能力，而 WAVEMix 則是擁有同時播放八聲道的聲音檔。緊接著又出了能使得影像高速化的 WinTool，但是這些產品的實用性並不大，因此有人開始懷疑起 MicroSoft 的能力。

一直等到了 95 年，Windows95 的誕生產生了革命性的改變。電腦遊戲開始不一定要在 MS-DOS 環境下執行，同一時間，可以使電腦遊戲和多媒體在 Windows 平台上更快速執行的 DirectX 也誕生了。雖然 DirectX 的誕生是令人振奮的，不過中間的發展並不如想像中的快速，原因就是使用者太少。但隨著版本的升級，功能越來越強悍，支援的遊戲也就變多了。一直到現在，DirectX 早已成為製作遊戲的標準。

二、DirectX 的應用

微軟的 DirectX 軟體開發工具包(SDK)提供一套優秀的應用程式介面，以一致性的開發介面直接與電腦硬體進行溝通，開發出高效能的多媒體程式和電腦遊戲。總體來說 DirectX 提供了兩個主要的好處：

1)為軟體開發者提供硬體無關連性：

DirectX 的結構是由兩個驅動程式構成的，硬體抽象層(HAL)和硬體模擬層(HEL)，當 DirectX 建立時，同時會產生一張相容表，記錄了當前硬體系統支援的功能，在 DirectX 需要實作出某項功能時就查表，查得到就用 HAL 實作出來，反之則用 HEL 模擬出類似的功能。

2)為硬體開發者提供策略：

DirectX 的另外一個主要的目的是給硬體廠商提供開發策略。他們可以從高性能程式的開發者和獨立的硬體供應商(independent hardware vendors，IHVs)那裡得到回饋。所以，在 DirectX 程式員參考中有時可能會提供那些還不存在的硬體加速設備的技術細節。

三、DirectX 技術元件及其功能描述

1)DirectX Graphics — 集成 DirectDraw 和 Direct3D。

Microsoft DirectDraw 和 Microsoft Direct3D 併入了 DirectX Graphics 組件(圖 C-1)。DirectDraw 的主要任務在於 2D 繪圖功能；Direct3D 的主要任務是 3D 繪圖。API 已經進行了大幅度更新，現在更容易使用，並且支持最新的圖形硬件。最引人注目的新特性是支持可編程著色器（著色器是用著色語言編寫的一段代碼，著色語言是專為在可編程頂點流水線或可編程像素流水線中使用而設計的）。

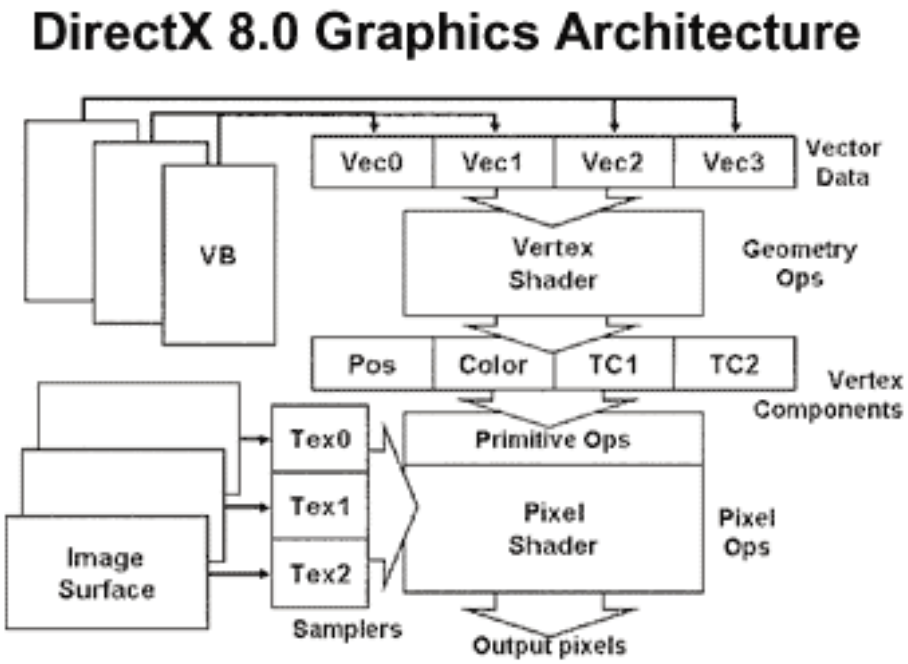

圖 C-1 DirectX 8.0 Graphics 系統結構

2)DirectX Audio — 集成 DirectMusic 和 DirectSound。

Microsoft DirectX 8.0 Audio 為播放集成的音樂和聲音效果(圖 C-2)提供了新體系結構。儘管仍然使用名稱 Microsoft DirectSound 和 Microsoft DirectMusic，但在它們之間已經沒有明顯的區別。WAV 文件或其他資源現在可以由 DirectMusic 加載器加載，通過 DirectMusic 演奏器進行播放，並用 MIDI 音響進行同步。

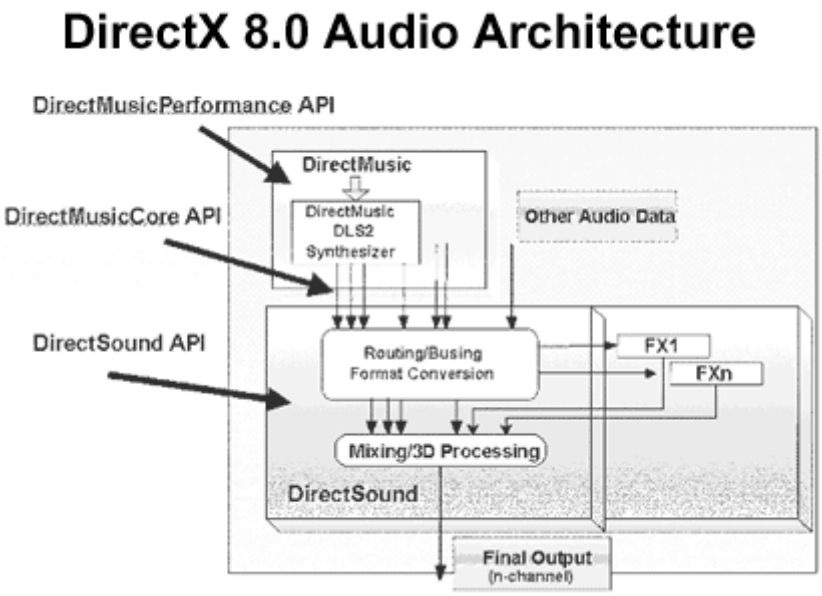

圖 C-2 DirectX 8.0 Audio 系統結構

3)DirectPlay

DirectPlay 是應用程序和通訊服務之間的高級軟體介面(圖C-3)。有了 DirectPlay，通過 Internet、調製解調器鏈接、或網絡來連接遊戲將非常簡單。DirectPlay 既提供了高級的傳輸層服務(例如，有保證或無保證的傳遞，慢速鏈接上的通訊扼殺，以及放棄連接檢測等)，也提供了會話層服務（包括玩家名稱表管理和點對點主機轉移）。以及提供與通訊服務提供程序無關的獨立性。Microsoft DirectPlay 組件進行了大幅度更新，其易用性和功能均得到改進，在可伸縮性和性能方面尤為顯著。此外，DirectPlay 現在還支援玩家之間的語音通訊。

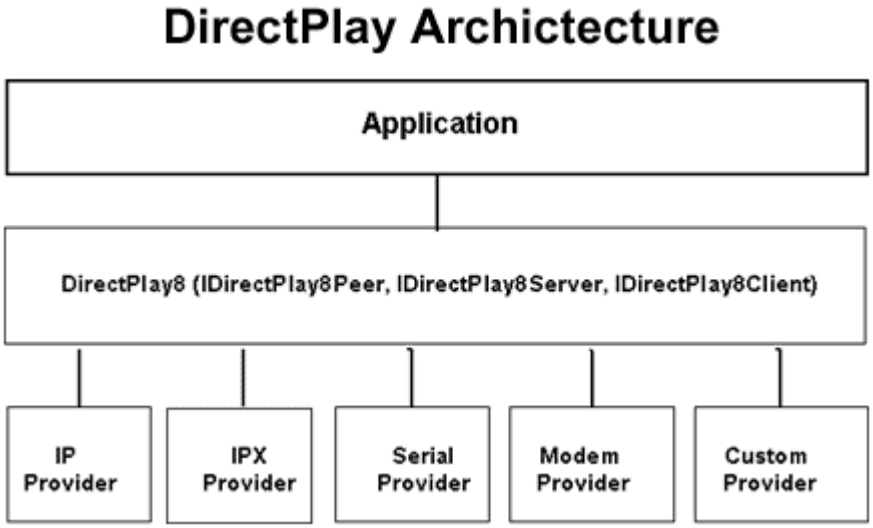

圖 C-3 DirectPlay 系統結構

4)DirectInput

DirectInput 為遊戲桿、頭盔、多鍵鼠標、以及力回饋設備等各種輸入設備提供了最先進的接口。通過直接與設備驅動程式配合，DirectInput 繞過了 Windows 訊息系統，提供了最佳性。DirectInput 引入了一個重要的新特性：操作映射。通過操作映射能夠在輸入操作和輸入設備之間建立連接。您只需輸入設備處理，而不必再依賴特定的設備對象。

5)DirectShow

DirectShow 原稱為 ActiveMovie，是 Microsoft 公司所制定的多媒體架構，可大幅提升前一種架構—媒體控制介面(MCI)的效能。由於 16 位元 MCI 在先天上擁有諸多限制，例如需要龐雜的單功能驅動程式，因此 DirectShow 便應運而生，以容納各類新興的多媒體硬體與技術。以 Microsoft 的組件物件模型(COM)為基礎的 MCI，原本有介面不一致的難題，在採用具備多線緒與多作業能力的 DirectShow 後，這些問題皆不復可見。

6)DirectSetup

可以使玩家只透過一個簡單的函數，就能夠完成 DirectX 驅動程式的全部安裝。

OpenGL 的功能與元件之研究

一、OpenGL 的由來

OpenGL 是 Open Graphic Library 的縮寫，即開放式圖形庫，是 Silicon Graphics 公司在其推出的 IRIS GL 圖形庫基礎上發展起來的，可以廣泛使用於個人電腦、工作站、和超級計算機上的開放式三維圖形應用和程式介面。適用於 Windows NT、Windows 95、Mac OS、OS／2、及 Unix 等平臺，具有非常強的可移植性。

OpenGL 現由業界著名的 OpenGL 體系結構評審委員會(ARB)主導。該委員會包括英特爾(Intel)、IBM、微軟(Microsoft)、DEC、康柏(Compaq)、SGI、Intergraph、Evans 和 Sutherland 等九個成員，主要負責評審 OpenGL 的功能擴展和制定相關的技術規範。

由於 OpenGL 開發的很早，因此性能非常完善，包含了大量功能強大的圖形函數，可以實現從渲染一個點，到採用 Phong 光源、MIP 紋理映射、及消除鋸齒等復雜操作的全部 3D 處理方案。因此，遊戲領域應用的雖然只有它的子集，就已經比 DirectX 有更好的發揮圖形的潛力。

微軟曾經多次試圖排擠 OpenGL 在遊戲領域的地位，但始終未能如願以償。反而越來越多的遊戲公司(包括 Id 制作公司在內)聯名要求微軟在其操作系統中加入對 OpenGL 的支持。這導致了微軟最終的妥協—在其與 SGI 聯合發表的一份聲明中，表示將就 OpenGL 在 Windows 中的應用與後者進行更廣泛的合作，在以後的 DirectX 體系中，OpenGL 將會獲得和 D3D 一樣的重視程度。他們將聯合在一起，在一種新的且命名為 DDl(Dev6ceDiver Kit)架構中出現。一種專門為 Windows 作過優化，叫做 OpenGL ICD(Instalallable Client Driver)的驅動將會使大多數 3D 卡更容易為 OpenGL 所識別，從而實現更廣的適用範圍。

二、OpenGL 應用功能

1)幾何建模

OpenGL 能夠繪製點、線、和多邊形。應用這些基本的形體，我們可以構造出幾乎所有的三維模型。OpenGL 通常用模型的多邊形的頂點來描述三維模型。它所定義的點、線、及多邊形等圖元與一般的定義不太一樣並存在一定的差別。對程式員來說，能否理解二者之間的差別十分重要。

一種差別源於基於計算機計算的限制。OpenGL 中所有浮點計算精度有限，故點、線、及多邊形的座標值存在一定的誤差。另一種差別源於位圖顯示的限制。以這種方式顯示圖形，最小的顯示圖元是一個像素，儘管每個像素寬度很小，但它們仍然比數學上所定義的點或

線寬大得多。當用 OpenGL 進行計算時，雖然是用一系列浮點值定義點串，但每個點仍然是用單個像素顯示。

2)座標轉換

在建立了三維景物模型後，就需要用 OpenGL 描述如何觀察所建立的三維模型。觀察三維模型是通過一系列的座標轉換進行的。模型的座標轉換在使觀察者能夠在視點位置觀察與視點相適應的三維模型景觀。在整個三維模型的觀察過程中，投影轉換的類型決定觀察三維模型的觀察方式，不同的投影轉換得到的三維模型的景像也是不同的。最後的視窗轉換則對模型的景像進行裁剪縮放，即決定整個三維模型在螢幕上的圖像。

3)顏色模式的指定

OpenGL 應用了一些專門的函數來指定三維模型的顏色。程式員可以選擇二個顏色模式，即 RGBA 模式和顏色表模式。在 RGBA 模式中，顏色直接由 RGB 值來指定；在顏色表模式中，顏色值則由顏色表中的一個顏色索引值來指定。程式員還可以選擇平面著色和光滑著色二種著色方式對整個三維景觀進行著色。

4)光照應用

用 OpenGL 繪製的三維模型必須加上光照才能更加與客觀物體相似。OpenGL 提供了管理四種光（輻射光、環境光、鏡面光、和漫反射光）的方法，另外還可以指定模型表面的反射特性。

5)圖像效果增強

OpenGL 提供了一系列的增強三維景觀的圖像效果的函數，這些函數通過反走樣、混合、和霧化來增強圖像的效果。反走樣用於改善

圖像中線段圖形的鋸齒而更平滑，混合用於處理模型的半透明效果，霧使得影像從視點到遠處逐漸褪色，更接近於真實。

6)位圖和圖像處理

OpenGL 還提供了專門對位圖和圖像進行操作的函數。

7)紋理映射

三維景物因缺少景物的具體細節而顯得不夠真實，為了更加逼真地表現三維景物，OpenGL 提供了紋理映射的功能。OpenGL 提供的一系列紋理映射函數使得開發者可以十分方便地把真實圖像貼到景物的多邊形上，從而可以在視窗內繪製逼真的三維景觀。

8)即時動畫

為了獲得平滑的動畫效果，需要先在內存中生成下一幅圖像，然後把已經生成的圖像從內存拷貝到屏幕上，這就是 OpenGL 的雙緩充技術（double buffer)。OpenGL 提供了雙緩充技術的一系列函數。

9)互動式圖形

目前有許多圖形應用需要人機介面，OpenGL 提供了方便的三維圖形人機介面，用戶可以選擇修改三維景觀中的物體。

三、Windows NT 下 OpenGL 的結構

OpenGL 的作用機制是用戶端（client）/伺服器（sever）機制，即用戶（用 OpenGL 繪製景物的應用程序)向伺服器（即 OpenGL 核心）發佈 OpenGL 命令，伺服器會解釋這些命令。大多數情況下，用戶和伺服器在同一機器上運行。

正是 OpenGL 的這種用戶/伺服器機制，使得 OpenGL 可以十分方便地在網路環境下使用。因此 Windows NT 下的 OpenGL 是網路透明的。正像 Windows 的圖形設備介面（GDI）把圖形函式庫封裝在一個動態鏈結庫（Windows NT 的 GDI32.DLL）內一樣，OpenGL 圖形庫也被封裝在一個動態鏈結庫內（OPENGL32.DLL)。受客戶應用程式呼叫的 OpenGL 函式都先在 OPENGL32.DLL 中處理，然後傳給伺服器 WINSRV.DLL。OpenGL 的命令再次得到處理並且直接傳給 Win32 的設備驅動介面（Device Drive Interface，DDI)，這樣就能把經過處理的圖形命令送給視頻顯示驅動程式。圖 C-4 簡要說明這個過程：

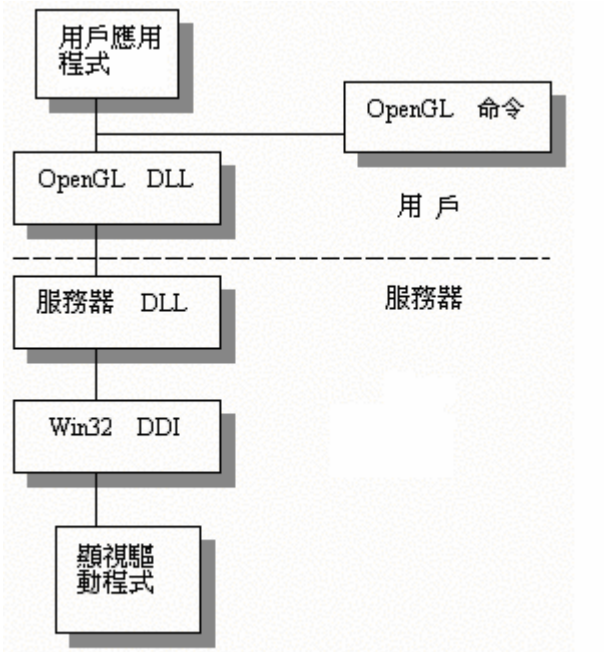

圖 C-4 OpenGL 在 Windows NT 下運行機制

在 3D 圖形加速卡的 GLINT 圖形加速晶片的加速支援下，二個附加的驅動程式被加入這個過程中。一個 OpenGL 可安裝用戶驅動程式（Installable Client Driver，ICD）被加在用戶這一端，一個硬體指定 DDI（Hardware-specific DDI）被安裝在伺服器端(C-5)，這個驅動程式與 Wind32 DDI 是同一級別的。

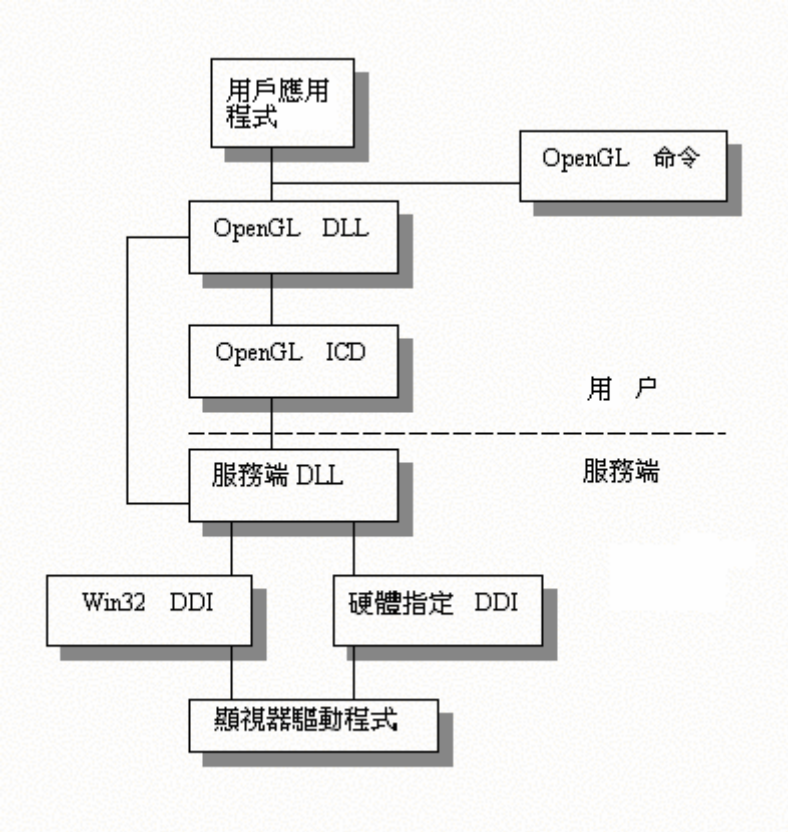

圖 C-5 在 3D 圖形加速下 OpenGL 運行機制