手機遊戲發展平台

研究生：陳智文　　指導教授：吳毅成

國立交通大學 資訊工程學系

摘要

隨著時代的潮流，手機遊戲的發展愈來愈蓬勃。為了市場的需求及競爭，各種不同類型的手機遊戲不斷的被設計開發出來，而在這眾多的手機遊戲裡面，同一類型的手機遊戲的情境及架構都非常的類似，也因此我們希望能夠建立一個手機遊戲發展平台，讓遊戲開發者能夠更快速、更方便的開發手機遊戲。而本論文的手機遊戲發展平台主要著重在棋盤(Board Games)和撲克牌(Card Games)，這些類型的遊戲。

除此之外，由於手機上的記憶體容量通常很小，因此如何縮小手機遊戲的程式碼大小，也將是本論文的另一個重點。最後，由於手機及遊戲開發技術的限制，在手機上要進行除錯(debug)，是相當不容易的，本論文也將會探討一些關於在手機上除錯的有用方法及好用的類別工具。希冀此研究成果能夠對於手機遊戲領域有一定的貢獻。

A Mobile Game Development Platform

Student: Chih-Wen Chen Advisor: Yi-Cheng Wu

Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract

With the rapid development of mobile games, more and more mobile games have been designed. For the same class of mobile games, the scenarios and design architectures are usually similar. Hence, we hope to set up a mobile game development platform for game designers to develop mobile games easier and faster. In this thesis, the mobile game development platform focuses mainly on the board games and card games.

Besides, in the mobile devices, since the memory sizes for programs are usually small, so that reducing the code size is another important issue in this thesis. Finally, it is very difficult to debug the mobile code due to different constraints imposed by various mobile devices. This thesis will also discuss some guidelines and utilities for debugging. It is believed that our research is helpful for game designers to research and develop similar mobile games.

誌謝

首先，要感謝的人是我的指導教授，吳毅成博士，在我寫這篇論文的期間，給我相當多的指導，也進行了相當多的討論，由於他的寶貴意見及指正，這篇論文才能夠順利的完成。

接下來要感謝的是博士班的徐健智、汪益賢、林秉宏等學長，這篇論文的許多設計和架構都是和他們一起不斷的討論、修正而完成的。同時，也要感謝實驗室的同學林義欽、朱俊欣、陳家齊，和所有的學弟妹們在我念研究所這段時間內所給予的幫助及鼓勵，也因為有他們的陪伴，讓我的生活充滿了歡樂。

最後，要感謝的是我的父母，在我求學生涯上所給我的幫助及照顧，不管是否遇到挫折，不論成功或失敗，總是默默的支持我，給我最大的信心、動力。謹以此論文，獻給我最摯愛的家人。

目錄

摘要..... i

誌謝..... iii

目錄..... iv

表目錄..... vi

圖目錄..... vii

第一章、緒論..... 1

1.1 手機遊戲的發展趨勢 1

1.2 手機遊戲的開發技術 3

1.3 手機的限制 5

1.4 論文大綱 7

第二章、背景介紹..... 8

2.1 J2ME 8

2.1.1 CLDC..... 11

2.1.2 MIDP..... 13

2.2 開發手機遊戲的問題 14

2.3 研究動機 15

2.4 JAVA手機遊戲分類 16

2.5 手機棋類、牌類遊戲的共通情境(Scenario) 17

第三章、平台的設計..... 19

3.1 遊戲的階層架構 19

3.2 平台的模組 20

3.2.1 核心模組 (Core Module)..... 21

3.2.2 圖形模組 (GUI Module)..... 24

3.2.3 儲存模組 (Storage Module)..... 30

3.2.4 網路模組 (Network Module)..... 33

3.2.5 工具模組 (Utility Module)..... 37
第四章、實作及相關問題探討..... 41
4.1 實作環境 41
4.2 縮小程式碼 42
4.2.1 縮短名稱..... 43
4.2.2 合併類別及移除介面 44
4.2.3 移除多餘的程式碼..... 46
4.2.4 移除網路功能 47
4.2.5 綜合使用 48
4.2.6 其它..... 49
4.2.6.1 合併圖形 49
4.2.6.2 多使用函式庫所提供的類別..... 50
4.3 除錯 50
4.3.1 手機上除錯的困難..... 50
4.3.2 使用模擬器..... 51
4.3.3 除錯的注意事項..... 52
4.3.4 除錯的工具類別..... 52
第五章、結論與未來發展..... 55
參考文獻..... 56

表目錄

表格一、JAVA2 版本 J2EE、J2SE 及 J2ME 的比較 9
表格二、手機模擬器列表..... 41
表格三、縮小程式碼-使用混淆器..... 44
表格四、縮小程式碼-合併類別及移除介面..... 45
表格五、縮小程式碼-移除多餘的程式碼..... 46
表格六、移除多餘程式碼檢查的項目..... 47
表格七、縮小程式碼-移除網路功能..... 47
表格八、縮小程式碼-綜合使用..... 48

圖目錄

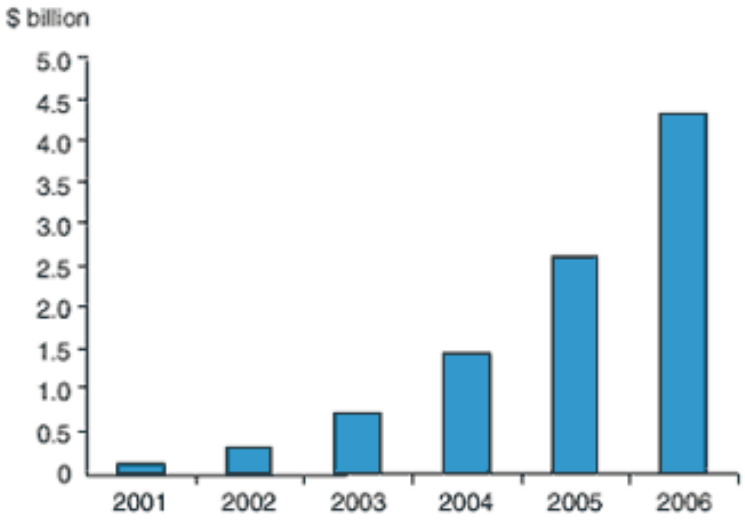
圖一、全球行動遊戲市場規模之成長趨勢（2001-2006）..... 1
圖二、全球行動遊戲市場之區域分佈（2001 及 2006）..... 2
圖三、J2EE、J2SE、J2ME 核心函式庫類別的關係 9
圖四、JAVA 各平台的架構圖 10
圖五、棋類和牌類遊戲的共通情境圖..... 17
圖六、遊戲的階層架構圖..... 19
圖七、平台各模組之間的關係圖..... 20
圖八、核心模組的類別圖..... 21
圖九、核心模組的 BASECLILET 類別函式對照遊戲情境圖..... 23
圖十、五子棋的遊戲說明選單..... 25
圖十一、遊戲說明選單的 HELPNODE 類別..... 26
圖十二、五子棋的遊戲主選單..... 27
圖十三、遊戲主選單的類別圖..... 27
圖十四、遊戲 GUI 的介面..... 28
圖十五、DEVICE-DEPENDENT GUI 介面 29
圖十六、遊戲的載入畫面和顯示分數畫面..... 30
圖十七、儲存模組的類別圖..... 31
圖十八、網路模組的類別圖..... 33
圖十九、切割圖形..... 38
圖二十、程式使用 PROGUARD 縮短名稱的結果 43
圖二十一、除錯的工具類別及使用畫面..... 53

第一章、緒論

本章會說明目前手機遊戲的發展現況及趨勢，並介紹相關的手機遊戲開發技術及手機上的一些限制。最後，會說明本論文的論文大綱。

1.1 手機遊戲的發展趨勢

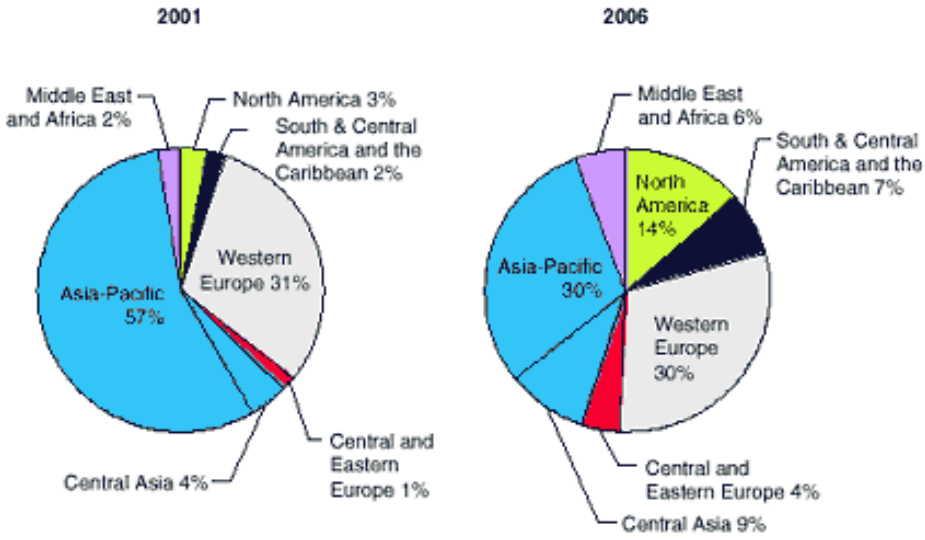
隨著通訊產業不斷的發展，「無線」與「行動」的觀念愈來愈大眾化，所以手機已經成為現代人連絡與溝通最重要的工具之一，「人手一機」的情況也愈來愈普及。隨著手機的重要性提升，使得有更多的廠商加入手機硬體的研發，也因此手機的運算能力大幅的提升，顯示的畫面從約 80*50 到 120*160 上下，色彩也從黑白變成彩色，整個手機的環境也愈來愈適合行動遊戲的發展，所以逐漸有愈來愈多的遊戲廠商開發出不同類型的手機遊戲，來娛樂大眾和消磨時間。



圖一、全球行動遊戲市場規模之成長趨勢 (2001-2006)
（資料來源：Wireless Playing to Win，OVUM 2001 年發表[1]）

如上圖一，OVUM 在西元 2001 年 10 月所發表的一篇關於無線遊

戲的報告資料[1]指出，在西元 2001 年全球行動遊戲的市場約是一億二仟四百多萬美元，而根據他們的統計資料估計，顯示之後的每一年行動遊戲市場幾乎呈現倍數成長，而在西元 2006 年約達到四十四億美元之多。



圖二、全球行動遊戲市場之區域分佈 (2001 及 2006)
（資料來源：Wireless Playing to Win，OVUM 2001 年發表[1]）

另外，我們來看看行動遊戲市場之區域分佈，如上圖二，同樣是 OVUM 在西元 2001 年 10 月所發表的一篇關於無線遊戲的報告資料[1]指出，在西元 2001 年時，由於日本和韓國積極的發展行動遊戲，也因此當時的行動遊戲市場光是亞洲的太平洋區就佔了 57%，若再加上中亞部份的 4%，整個亞洲就佔了全球的 61%，若換算成金額大約是七仟伍百多萬美元。而日本、韓國及西歐的發展也帶動了全球行動遊戲的發展，也因此 OVUM 估計在西元 2006 年時，全球行動遊戲的市場，會均衡的發展，所以亞洲、歐洲和美洲約各佔了 30%左右。從西元 2001 年及 2006 年的估計值比較，我們可以看出，雖然亞洲的市場，從 61%降到了 39%，但換算成資金來看的話，卻是從七仟伍百多萬美元升到了十七億一仟六百多萬美元，整整成長了二仟二百多倍。基本上，這個數字並不誇張，以下載一個手機行動遊戲約二美元來說，十七億一千六百多萬美元相當於下載了八億五仟八百萬人次的手機行動遊

戲，以亞洲將近二十億的人口看來，在西元 2006 手機更普及的時代，這似乎並不是個不可能的任務。

1.2 手機遊戲的開發技術

目前制定第三代移動通信標準的國際性組織(3GPP)，定義了行動應用執行環境技術之規範(MExE)[2]，在此規格書中有三項建議的開發技術：

- WAP(Wireless Application Protocol)：
  是一種無線應用軟體協定，主要是為無線終端裝置提供無線通訊與服務。一個 WAP 系統，主要包含二種元素，一種是 WML 語言，另一種是用來負責轉換 WML 語言的 WAP Gateway[3]。

- Personal JAVA：
  是早期昇陽(Sun)公司為了能夠在資源有限的設備上執行 JAVA 應用程式所建立的一種 JAVA 版本，它算是 JAVA 的過渡性規格，而它的 API 為 JDK1.1 的子集合。

- J2ME CLDC/MIDP：
  也是昇陽(Sun)公司為了能夠在資源有限的設備上執行 JAVA 應用程式所建立的一種 JAVA 版本，但它所需要的硬體支援較低，並且核心函式庫也是特別經過精簡化的。目前 J2ME 的應用很廣，包括網路電話、車用電腦、手機、PDA 等[4][5]。

基本上，J2ME 與 WAP 是二項互不衝突的開發技術，但有幾個重要的原因，使得 J2ME 在開發手機遊戲上比 WAP 更受歡迎。首先，WAP 使用 WML 語言，且也支援使用 WMLS 來做基本的運算，並提供圖片、圖示、表格、程式控制功能鍵與變數[3]。但是，WML 語言與 WMLS 每

個檔案的限制是 1.4KB，這對開發遊戲來說，限制非常大。因此，WAP 並不適合用來開發較複雜的遊戲。其次，WAP 並不支援 HTTP、TCP/IP、UDP 等協定，故無法直接跟遠端的伺服器連線，而需透過一個 WAP Gateway 來完成這件事，且 WAP Gateway 還得負責做中間的協定轉換工作，這對於遊戲要進行訊息交換，或存取資料，都是很不方便的事情。

Personal JAVA 因為它所需要的硬體支援較高，如中央處理器的速度和記憶體的需求，所以並不適合，同時也沒有人用它來開發手機遊戲。

J2ME 提供了 HTTP、TCP 等協定，讓 JAVA 程式可以輕易做到(1)直接跟遠端的伺服器連線(2)訊息交換(3)存取資料。除此之外，J2ME 所開發的應用程式或遊戲，可以透過 GPRS 或無線網路直接下載到手機端，且執行的時候，不再需要透過網路執行，這對手機用戶端而言，是非常方便的事情。另外，J2ME 也有許多的功能[4][5]，例如：

- 平台的獨立性和可攜性：
  使得手機應用程式只要寫一次，就可以在任何有 Java 虛擬機器的手機上執行，這點也是眾多手機大廠如 Nokia、Sony Ericsson Motorola、所非常喜愛使用 J2ME 來開發應用程式的重要原因。

- 安全性：
  JAVA 程式無法使用指標，並且會自動對記憶體進行管理，應用程式便無法隨便的存取記憶體，因此不會因為執行 JAVA 程式而造成手機當機或資料遺失。

- 豐富的函式庫類別：
  J2ME 提供了豐富的函式庫類別，讓我們可以快速的開發應用程式。

◆ 動態下載應用程式：

可透過 GPRS 或其它的無線網路下載應用程式到手機上自動安裝並執行。

由於 J2ME 的許多優點，目前大部份手機遊戲業者，都選擇用 J2ME 來開發手機遊戲。本論文也因此以 J2ME 來開發我們的手機遊戲平台。

1.3 手機的限制

從上一節的介紹我們可以知道，雖然 J2ME 確實提供了不少功能讓我們可以更方便、快速的開發手機遊戲，但手機畢竟是一種小型的行動裝置，它的運算能力和顯示能力，跟個人電腦(PC)比起來，仍然相去甚遠，底下我們列出手機的一些限制，而這些限制都將進一步的影響到手機遊戲的設計和開發：

◆ 有限的記憶體(RAM)容量：

在成本及手機空間的考慮下，目前市面上手機的記憶體容量，大致上都介於 160KB 到 2MB 之間，例如，Motorola T720 約有 200KB 或 Sony Ericsson T610 約有 300 至 400KB 等等。而若以 J2ME/CLDC (下一章將會介紹)來開發遊戲的話，由於硬體上的最小需求，至少要有 128KB 要保留給虛擬機器和類別函式庫，32KB 要保留給應用程式執行。另外，由於記憶體的限制，應用程式的大小也連帶受到限制，例如，一隻只有 200KB 記憶體的程式，我們總不可能開發 220KB 大小的應用程式。

◆ 有限的計算能力：

目前大部份手機上的處理器(CPU)的速度，大致上都介於 70 到 90MHz 之間，再好一點的也大概是 100MHz。這跟個人電腦上的處理器速度動不動就幾百、幾千 MHz 比起來，相當的慢。也因為處

理器速度的影響，計算能力明顯的不足，當應用程式需要大量的計算時，執行的速度就會變慢，例如，當遊戲需要執行動畫時。

◆ 有限的電池容量：

目前市面上的手機，待機時間短則幾天，長則可達一、二個禮拜之久，但若經常執行應用程式，不斷地使用處理器和記憶體，則耗電量將會非常快，甚至一天就沒電了。

◆ 有限的畫面大小和解析度：

由於先天上的限制，大家希望手機能夠盡量小一點，以免帶起來不方便，這當然也影響到螢幕所佔的比例。目前大部份的手機螢幕解析度都在 90*120 像素至 120*160 像素之間。例如，Motorola T720 的螢幕解析度為 120*160 像素或 Sony Ericsson T610 的為 128*128 像素等等。而畫面的大小，也進而影響到應用程式的畫面設計。通常，我們會希望遊戲能夠同時顯示更多或夠多的遊戲資訊在螢幕上，例如，遊戲的分數、規則、玩家的名稱等等，但當畫面大小不足，有些資訊必須取捨，被迫只能佔更小的空間或去掉，這對設計遊戲而言，是非常不方便的。

◆ 有限的使用者輸入能力：

在手機上按鍵不像個人電腦的鍵盤一樣多，也沒有滑鼠可以使用，要操作遊戲，只能透過手機的按鍵。如何利用這些按鍵，設計出讓遊戲玩家覺得方便又容易操作的介面，這也是遊戲設計者所必須要考量的一個地方。

由上述的說明，我們知道，在手機裝置的限制很多，因此，我們在設計時，要將這些因素也考慮進去。

1.4 論文大綱

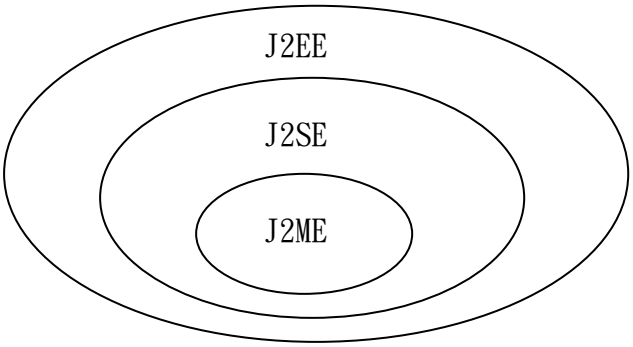
本論文主要是設計一個手機遊戲的開發平台，共有五大章節。第一章說明手機遊戲的發展趨勢、開發技術及手機的限制。第二章說明相關的背景，包括介紹 J2ME CLDC/MIDP 及在語言和虛擬機器上的一些限制，及在手機上開發遊戲的問題，並說明研究動機，最後介紹 JAVA 手機遊戲分類，以及棋類和牌類遊戲的共同情境。第三章是本論文的重點，會詳細說明我們整個手機遊戲架構的設計。第四章是實作，對說明實作的環境，還有如何縮小程式碼大小及在手機上進行除錯。第五章說明結論與未來發展。

第二章、背景介紹

本章將會說明用來開發消費性電子產品應用程式的J2ME(JAVA 2 Micro Edition)平台，及介紹J2ME平台的架構，包括在其上的Configuration和Profile的概念。另外，我們會分析一下在手機上開發遊戲的問題及困難點，並說明我們的研究動機。最後，會說明目前市面上用J2ME所開發的手機遊戲的分類，及本論文所著重的棋類、牌類的遊戲情境。

2.1 J2ME

JAVA是昇陽(Sun)公司於西元1991年所開發的一套程式語言，而當時他們的計畫，是希望能讓JAVA所開發出來的程式能在不同的平台上執行，例如，Windows、Linux、FreeBSD等等。而JAVA發展到JAVA 2之後，因為不同需求，分成了四個不同的版本，其中三個目前應用較廣泛的平台分別是J2EE、J2SE及J2ME。而這三個平台的核心函式庫類別的關係[9]如圖三所示。J2EE的核心函式庫支援最多，其次是J2SE，最後是J2ME。而各平台所支援的類別，除了核心函式庫之外，再加上一些額外的API，例如，J2EE還包含有支援Servlet、JSP等API，J2SE則包含有支援使用者介面的AWT、Swing及網路功能等API，而J2ME部份，我們之後會做更詳細的說明。



圖三、J2EE、J2SE、J2ME 核心函式庫類別的關係

另外,各個平台之間的應用及比較關係[10],則如下表格一所示。

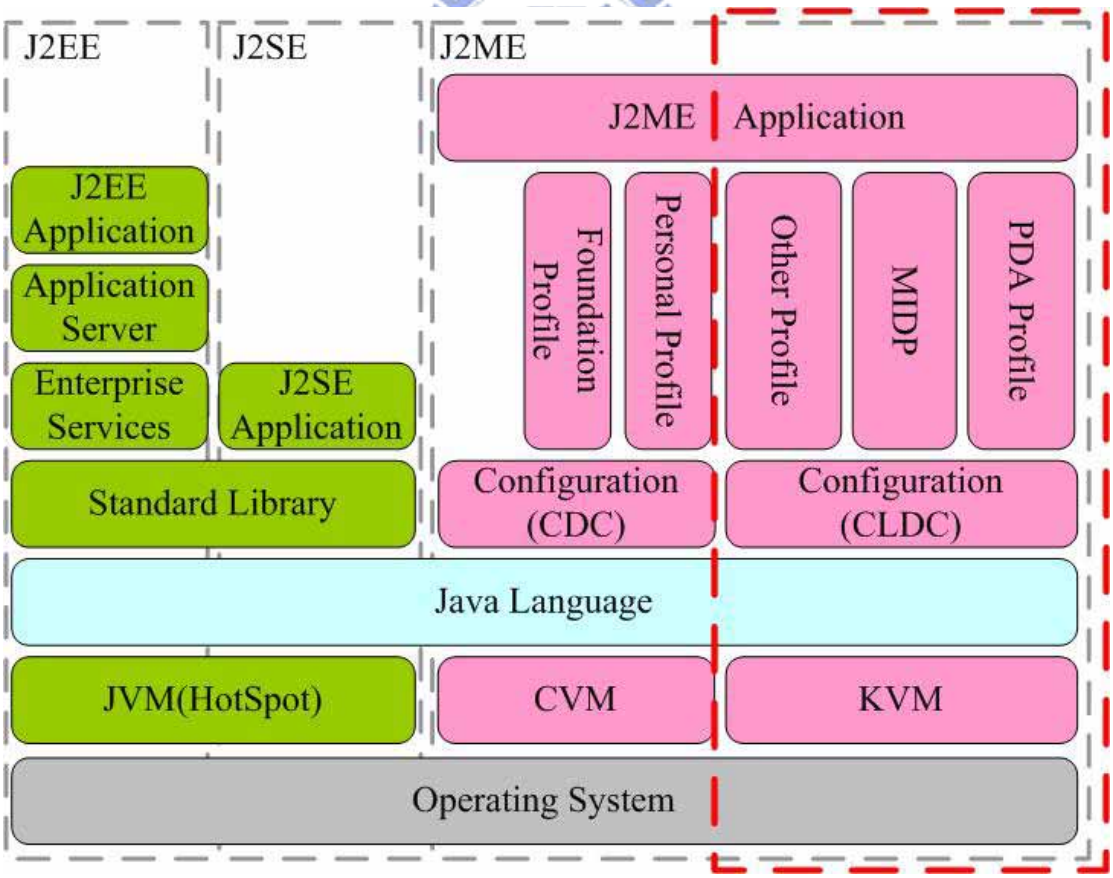
表格一、JAVA2 版本 J2EE、J2SE 及 J2ME 的比較

平台	定位	說明
J2EE(企業版)	伺服器端的應用	最主要用來開發企業級的伺服器端應用程式，例如，用 JDBC 開發資料庫存取相關的應用，用 JAVA Mail 開發郵件收發相關的應用，用 JSP 及 Servlet 開發網頁服務或電子商務應用等等。
J2SE(標準版)	個人電腦及用戶端的應用	最主要用來開發一般的用戶端應用程式，例如，網路連線遊戲、文書編輯工具等等。
J2ME(精簡版)	消費性電子產品的應用	利用精簡化過的核心函式庫來開發消費性電子產品上的一些應用程式，例如，手機上的遊戲、PDA 上的觀看股市行情軟體等等。

各平台之間的架構，則如下圖四所示。一般而言，JAVA 的各個平台由 JAVA 虛擬機器、JAVA 語言及 JAVA 類別函式庫所組成，但昇陽(Sun)公司在 J2ME 平台加入了 Configuration 和 Profile 這二個截然不同的新概念，最主要的原因是消費性電子產品或小型手持裝置，它們的硬體差異性很大，昇陽公司認為對於各種具有不同硬體特性的電子產品或小型手持裝置，應該要具有不同的設計規格[4][5][9]。因此，Configuration 主要根據硬體的記憶體容量、處理器的速度、連結網路的能力、使用者輸入輸出的能力來將這些不同規格的電子產品做個區分。目前 Configuration 主要分為二種，一種是 CLDC

(Connected, Limited Device Configuration)及 CDC (Connected Device Configuration)，顧名思義，CDC 不管在記憶體容量、處理器速度、連網能力的支援度要求都比 CLDC 要來的高。其中，手機即屬於 CLDC 所定義的這類型的裝置之一，因此我們開發手機遊戲時，所使用的 Configuration，即是 CLDC。除此之外，Configuration 還定義了對 JAVA 語言、JAVA 虛擬機器及 JAVA 類別函式庫的支援程度，這部份等到我們之後詳細介紹 CLDC 時，會再做進一步的說明。

另外，Profile 的話，我們可以將它想做是一組 API 的集合，對屬於同一組 Configuration 概念的硬體裝置，如手機和 PDA，他們會有不同的應用。所以針對手機，就提供一組 Profile，專門用來給手機開發應用程式，而這組 Profile 的 API 內容就跟手機的特性非常相關；針對 PDA，就提供另一組 Profile，而這組 Profile 的 API 內容就跟 PDA 的特性非常相關，專門用來給 PDA 開發應用程式。



圖四、JAVA 各平台的架構圖

目前針對手機應用而設計的 Profile，最有名也最受歡迎的就是 MIDP (Mobile Information Device Profile)[12]。MIDP 提供了不少便於開發應用程式的 API，例如，應用程式生命週期的管理、使用者介面、資料儲存、網路連結、事件處理等等。在此，我們先讓大家對 MIDP 有個初步的概念，之後我們會做更詳盡的介紹。

以上所提到的各種 Configuration 和 Profile 是經由一個叫做 JCP (JAVA Community Process)的程序所制定的，而這個程序是由一群廠商所共同領導的，當 Community 的成員提出一個尚未被制定且有關 Configuration 或 Profile 的提案時，這個提案會被先命名為一個 JSR (JAVA Specification Request)，當這些領導廠商經過不斷的開會與討論而通過這個提案時，這個提案就會給予一個編號，成為一個正式的規格。例如，CLDC1.0 的 JSR 編號是 JSR-30，MIDP1.0 的 JSR 編號是 JSR-37 等等。除此之外，J2ME 也提供了平台擴充的能力，廠商可以根據自家手機的特性，自行制定適合手機特性的 API (Profile)，例如，為了讓手機可以顯示特殊的聲光或音效，廠商自行開發相關可以顯示特殊聲光或音效的 API，讓應用程式使用等等。但使用為手機專門量身訂做的 API 有一個缺點，就是當應用程式要移到另一隻手機執行時，可能會因為沒有該特殊的 API 而無法執行。

2.1.1 CLDC

我們先前有提到過，Configuration 除了定義對硬體的支援程度之外，還定義了對 JAVA 語言、JAVA 虛擬機器及 JAVA 類別函式庫的支援程度。CLDC 目前共二套規格，分別為 CLDC1.0 及 CLDC1.1，而 CLDC1.1 是從 CLDC1.0 所延伸而來的。以下列出在 CLDC1.0 規格裡，對 JAVA 語言和 JAVA 虛擬機器所規範的一些限制[4][5][11]：

- 不支援浮點數：
  由於在 CLDC1.0 制定的當時，大部份手機的處理器支援有限，無法處理浮點數的表示和運算，故 CLDC1.0 乾脆就不把浮點數列入考量，也就是無法使用 float、double 等資料型態。

- 不支援 JNI (JAVA Native Interface)：
  由於安全性上的考量，虛擬機器限制不能存取 native function，而且對 JAVA 而言，實作 JNI 是非常耗費記憶體的，何況手機的記憶體容量原本就不大，在這些考量下，捨棄對 JNI 的支援。

- 不支援使用者定義的類別載入器：
  基本上，也由於安全性上的考量，應用程式的下載和管理應用由虛擬機器所預先提供的類別載入器來進行，而使用者所定義的類別載入器不能覆蓋或修改它。

- 不支援 Reflection：
  雖然 Reflection 可以讓我們動態的建立物件、呼叫函式，或觀看類別、物件、函式的資訊，但虛擬機器並不支援。

- 不支援 Thread Groups 和 Daemon Threads：
  雖然有支援 Multi-Thread，但不支援 Thread Groups 及 Daemon Threads，若要對一群 Thread 進行管理，要自己實作程式來完成。

- 不支援 Finalization：
  在 JAVA 語言中，當某一物件已經沒有被參考時，要被進行資源回收，而在回收前，會呼叫該物件的 finalize()方法，但 CLDC1.0 不支援 Object 的 finalize()方法。

- 不支援弱參考(Weak Reference)：
  所謂的弱參考就是在程式執行時，希望知道某個物件是否已經被

回收，但又不想影響這個物件被回收的過程，便可利用弱參考來達成這件事。使用弱參考，我們可以檢查物件是否被回收，或我們可以命令它進行回收。

◆ 有限的錯誤處理能力：
CLDC1.0 只提供部份的錯誤處理，分別是 Error、OutOfMemoryError 及 VirtualMachineError。其它像是 IllegalAccessError 或 StackOverflowError 等等，已經沒有支援。

而 CLDC1.1 則是向前相容於 CLDC1.0，並增加了對浮點數的支援、錯誤處理及對弱參考的支援等等。另外，類別函式庫方面，是 J2SE 的一個子集合，有不少需要使用大量記憶體的類別被捨棄，例如，被用來切割子串的工具類別 StringTokenizer 等等。

2.1.2 MIDP

我們之前也提到過，MIDP[4][5][12]是目前廣泛被用來開發手機應用程式的一套 Profile。而 MIDP 目前也有二套規格，分別是 MIDP1.0 及 MIDP2.0，而 MIDP2.0 也是從 MIDP1.0 所延伸而來。

在 MIDP1.0 中的專屬套件大概可分為五大類[12][13]：(1)使用者介面，(2)繪圖，(3)應用程式的生命週期管理，(4)資料儲存，(5)網路連結。其它則包含精簡核心函式庫的系統類別、資料型態類別、集合類別、工具類別、日曆與時間類別、輸入輸出類別、國際化類別、錯誤類別、例外類別等等。而 MIDP2.0 除了 MIDP1.0 所提到的套件及類別外，另外還有新增幾類的套件：(1)遊戲圖像管理，(2)聲音播放管理，(3)行動交易與資訊交換。其它則包含，在網路連結套件裡面

新增通訊協定類別，例如 ServerSocket、Socket、Datagram 等等。

目前大部份的 JAVA 手機上的 Profile 幾乎都是 MIDP。而各家手機廠商所自行定義的 Profile 也不少，較有名的如日本 NTT DoCoMo 公司所自行設計的 DoJa Profile。當然，使用自行定義的 Profile 可以支援較多的功能，但相對的，所開發出來的應用程式也只能使用在具有該 Profile 的手機上執行。

2.2 開發手機遊戲的問題

由於我們先前所介紹手機的限制以及 J2ME 平台的一些限制，我們知道在手機上開發遊戲，會遇到許多不同的限制，如下：

- 記憶體：
  由於手機上的記憶體較小，遊戲的大小當然是愈小愈好，也因此一些不必要或不需要的程式碼要盡可能的去掉，可縮小程式碼的方法，都要盡量採用。此外，在執行時，會消耗大量記憶體的函式庫類別，也要盡量少用，例如，Vector 等等。

- 網路支援能力：
  由於 JAVA 版本的關係，導致有的手機有支援使用 Socket，而有的手機並沒有，因此，如何在有支援 Socket 跟沒支援 Socket 之間做個適當的處理，也是我們要注意的。

- 不同手機有不同限制：
  例如，不同手機的螢幕大小通常不同，如何根據螢幕大小而調整遊戲畫面及顯示資訊的位置，是相當重要的。另外，有些手機對圖檔的大小有限制，有些手機對遊戲的大小有限制，有些手機對類別方法的使用有限制，這也都是我們必須要注意的事。

◆ 各家手機廠商的KVM實作不同：

雖然KVM是一個標準化，可是各家手機廠商所實作出來的KVM有些差異，且會多加上什麼限制，我們也不曉得。所以，在手機上要進行除錯，並不簡單。通常我們手機遊戲在使用手機模擬器開發的時候，由於模擬器並沒有辦法實際且完整的模擬在手機上的狀況，再加上手機上所使用的作業系統和虛擬機器，都和在使用模擬器模擬時有很大的差距，也因此很容易發生在模擬器執行程式沒有問題，但下載到手機上卻出現一堆問題的情況，且手機並不會好心的告訴我們發生了什麼問題，通常會使用「應用程式發生錯誤」來帶過。所以，在手機上進行除錯，變成一個很重要的工作，這也是我們在設計平台時，所必須要考量的。

上述的這些問題而言，對我們想要在不同的手機上開發應用程式，都是非常關鍵而且重要的。

2.3 研究動機

基於開發手機遊戲的一些問題，我們希望對於同一款遊戲而言，不必為了不同手機而開發太多不同的版本。再加上目前手機遊戲的趨勢，各種不同類型的手機遊戲不斷的被開發，各家手機遊戲的廠商競爭非常的激烈，我們希望能夠設計出一個手機遊戲的開發平台，而此開發平台具備下列三項元素：

◆ 一個手機遊戲程式設計的架構：

這個架構主要是希望能讓遊戲開發者能更方便、快速的在不同的手機上開發手機遊戲，並且支援讓不同的遊戲玩家透過網路對玩遊戲。而此架構主要是設計給遊戲開發者用來開發棋盤遊戲(Board Game)或牌類遊戲(Card Game)為主[6][7][8]，例如，圍

棋、五子棋、橋牌或大老二等等。

◆ 縮小程式碼的方針和工具：

由於手機的記憶體的限制，程式碼的大小變的相當重要，可能僅僅差個 1、2KB 就無法下載到手機上執行。也因此，縮小程式碼變成開發應用程式中所不可或缺的一個環節，工具則是輔助我們縮小程式碼的一個利器。

◆ 除錯的方針和工具類別：

在手機上不論是下載、安裝或執行應用程式都有可能發生錯誤。例如，Motorola T720 並不支援 Clone 函式，而在下載應用程式要安裝時，卻僅僅顯示「無法安裝」的訊息，或應用程式的某個地方發生例外(Exception)，但卻僅僅顯示「應用程式錯誤」的訊息。由上述例子可知，要在手機上進行除錯，是一項不容易且極為不方便的事情，也因此，除錯也是開發應用程式中一項重要的指標，好的除錯方法和工具類別可以縮短開發應用程式的時間，並讓遊戲能盡量減少錯誤的產生。

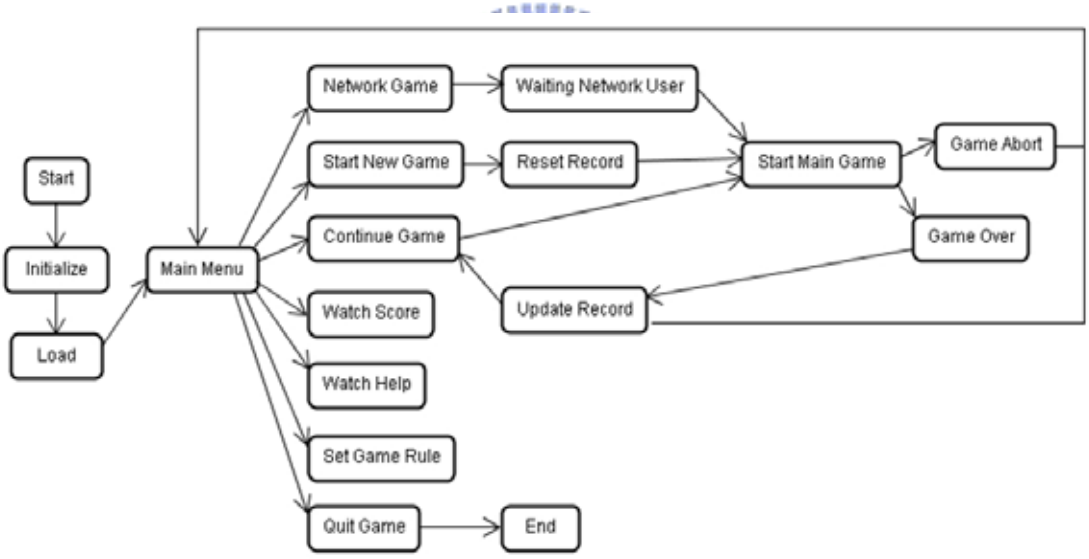
2.4 JAVA 手機遊戲分類

到目前為止，手機遊戲在全球各地的發展，可說是又多又雜。而目前國內幾個較大型的手機遊戲網站，分別是 iGame 手機遊戲館、sina 手機遊戲網及大宇資宇手機遊戲網，就分析他們所提供的手機遊戲而言，大致上可分為幾個不同的種類：(1)益智類，例如，彈力磚塊、賓果遊戲、大富翁等等，(2)趣味類，例如，魔法氣泡、俄羅斯方塊、撈金魚等等，(3)角色扮演類，例如，仙劍客棧、仙劍舞俠傳等等，(4)射擊類，例如，飛鏢快手、細菌入侵、Lucky Shooter 等等，(5)動作冒險類，例如，行動賽車手、滑板小子等等，(6)棋類、牌類，例如，五子棋、大老二、十六張麻將等等。

而在這眾多不同的手機遊戲分類當中，較受手機遊戲玩家所歡迎的就是棋類、牌類遊戲，而此類型遊戲也將是本論文所著重的地方。也就是說，本論文所設計的一個手機遊戲發展平台，即是一個主要用來開發棋類、牌類遊戲的發展平台。

2.5 手機棋類、牌類遊戲的共通情境(Scenario)

根據我們觀察目前市面上棋類和牌類遊戲的共通情境，大致上即如下圖五所示：



圖五、棋類和牌類遊戲的共通情境圖

首先，遊戲被選擇並且執行之後，會進入初始化(Initialize)的階段，而由於手機的處理器較慢，初始化通常需要一段時間，為了不讓使用者以為遊戲當掉或中止，因此在初始化時，會同時進入載入(Loading)的階段，告知使用者目前遊戲初始化的進度，而在初始化的階段，遊戲的分數等記錄，會從手機的資料庫載入到遊戲當中。而在初始化結束之後，遊戲會進入主選單(Main Menu)。

遊戲在進入主選單(Main Menu)之後，會有不少的選單讓玩家選擇，例如，網路對戰、開啟新局、繼續上次、看累積分數、看遊戲說明、設定遊戲的規則、離開遊戲等等。當玩家選擇網路對戰時，遊戲會進入等待狀態並顯示等待畫面，一旦遊戲伺服器(Game Server)確定有其它玩家加入網路對戰時，便會通知手機用戶端，然後開始與其它玩家進行對戰；當玩家選擇開啟新局時，會先清除之前的遊戲記錄，然後開始玩遊戲；當玩家選擇繼續上次時，便以上次遊戲儲存的記錄，直接開始玩遊戲。

在玩遊戲的同時，玩家可以選擇中止遊戲回到主選擇。當遊戲的每局結束之後，會把新的遊戲記錄存到手機的資料庫，接著玩家可以選擇繼續玩遊戲或回到主選單。

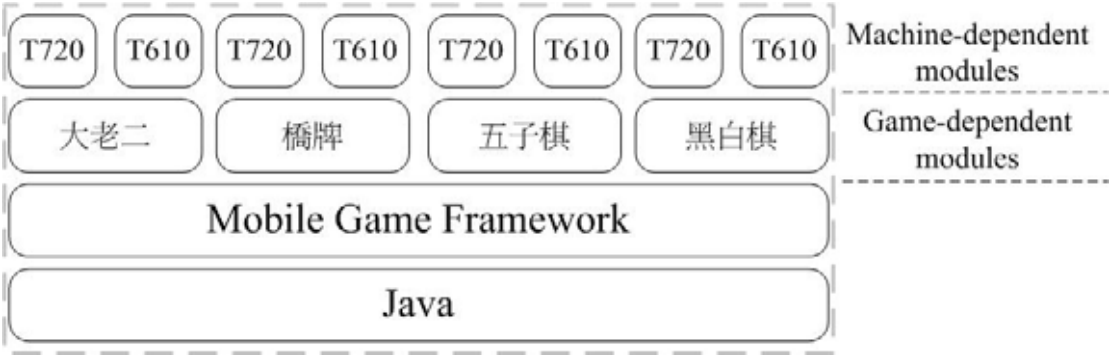
而本論文的手機遊戲發展平台，也會盡量根據這樣的情境去設計及調整。

第三章、平台的設計

用一個已經開發好的手機遊戲平台來設計遊戲的好處就是，平台已存在一些既定的流程，遊戲只需覆寫一些平台中的函式，就可以照著平台規範好的流程執行。也因為遊戲只需覆寫一些函式，並撰寫遊戲的邏輯，所以遊戲開發者可省下不少的開發時間。本章將會詳細介紹平台的設計，並舉例子來加以解釋使用的方法。

3.1 遊戲的階層架構

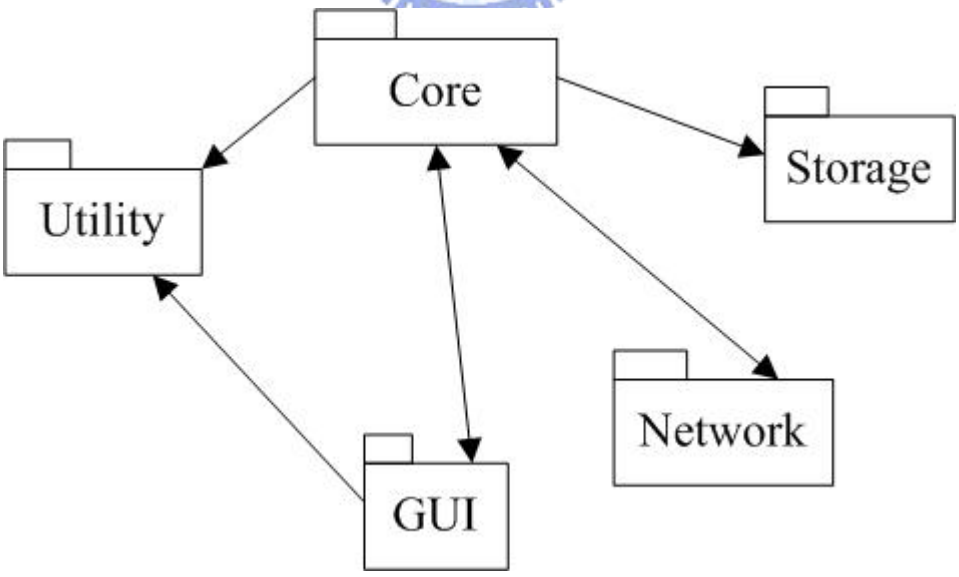
在介紹我們平台的各個模組之前，我們先來看一下遊戲程式的階層架構。如下圖六所示，最下面的階層即是 Java 平台，也就是我們先前在第二章所介紹用來在手機上開發遊戲的 J2ME 平台。在 J2ME 平台之上的 Mobile Game Framework 即是我們的手機遊戲平台。有了手機遊戲平台之後，各種不同的棋類或牌類遊戲，例如，大老二、橋牌、五子棋、黑白棋、麻將等等，即可利用這個平台在手機上快速的開發。而在整個階層架構的最上層，即是跟機器特性相關而另外開發出來的模組，例如，不同手機的螢幕大小不同，遊戲畫面的位置或使用的圖形必定有所不同，所以需要額外的模組去控制這些差異。



圖六、遊戲的階層架構圖

3.2 平台的模組

我們先來做個簡單的分析，假設今天我們要開發一個遊戲的話，會包含那些東西：(1)遊戲會有遊戲的邏輯和規則，例如，棋步走法是否正確或判定勝負的規則等，(2)遊戲會有畫面，用來顯示所要提供給玩家的相關資訊，例如，棋盤或棋子資訊等，(3)遊戲若要儲存相關資訊，例如，分數、勝、敗、和等，會有儲存的功能，(4)遊戲若要支援網路連線的話，會有網路功能，以便和網路上其它的玩家進行對戰。所以，根據這樣子的分析和概念，我們平台設計共分成了五個不同的模組，分別是核心模組(Core)、圖形模組(GUI)、儲存模組(Storage)、網路模組(Network)及工具模組(Utility)。其中，關於儲存模組的部份，由於手機上有提供儲存裝置，所以我們目前只考慮將資料存在手機端，暫時不考慮將資料儲存在遠端的資料庫伺服器。



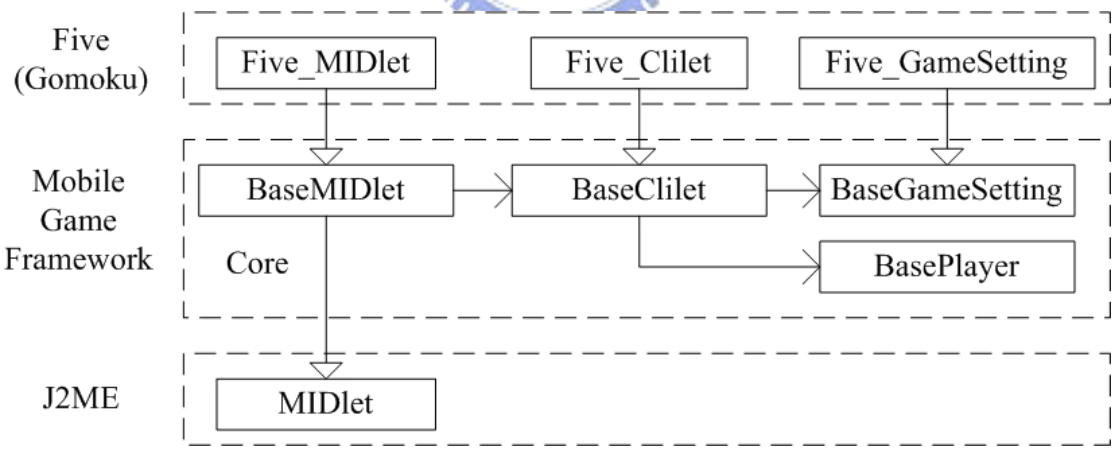
圖七、平台各模組之間的關係圖

接下來的各小節，我們將會對這些模組分別做更詳細的介紹。同

時，為了讓大家更容易了解我們模組的設計和使用方法，我們將會使用五子棋遊戲來當作說明的例子。另外，為了介紹方便，之後我們本文中所用到的「底層」字眼即是指我們的手機遊戲平台，而「上層」字眼即是指 Game-dependent 及 Module-dependent 階層，如圖六所示。

3.2.1 核心模組 (Core Module)

核心模組是整個遊戲架構的中心點，主要做的事情為：(1)負責整個遊戲的起始流程。(2)遊戲的邏輯及規則、存放遊戲的狀態等資訊。(3)跟遊戲特性相關的規則設定。(4)儲存玩家資訊。此模組包含四個主要的類別，分別是 BaseMIDlet、BaseClilet、BaseGameSetting 及 BasePlayer。圖八為核心模組的類別關係及以五子棋(Five)為例的繼承關係。



圖八、核心模組的類別圖

3.2.1.1 BaseMIDlet 類別

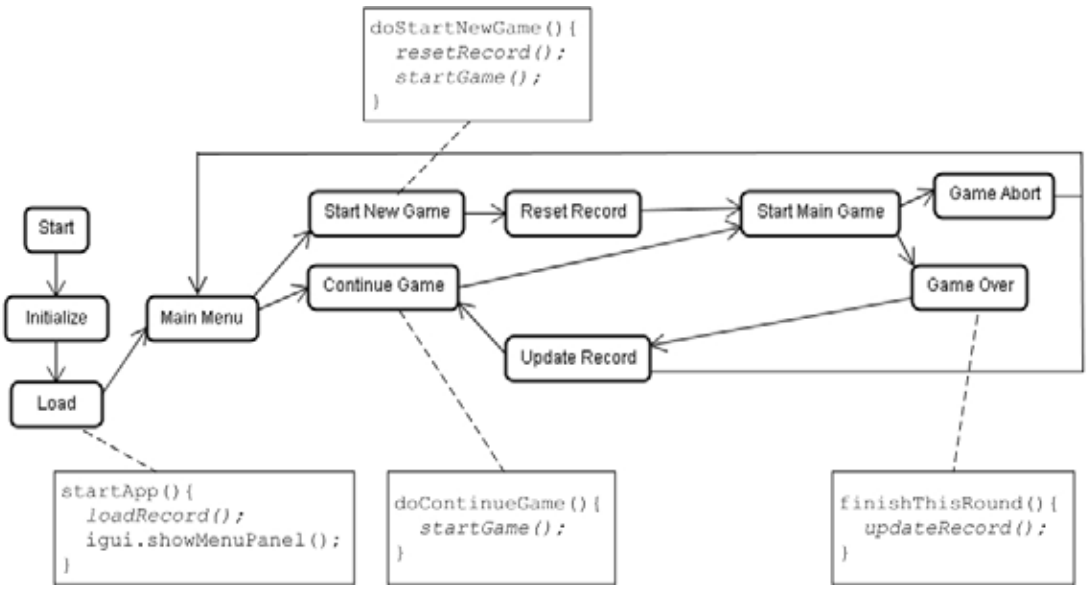
此類別是整個遊戲的起始點，它的角色就如同一隻應用程式的主程式，或如同一隻 Applet。此類別之所以存在的原因是，MIDP 規定應用程式的起始點是一隻 MIDlet，並提供一些需被覆寫的 abstarct 函式，以便對應用程式的生命週期進行管理[4][5][12]。

此類別最主要的任務是做初始化的動作，把遊戲中會用到的物件建立起來。但是由於 J2ME 並不支援 Reflection 的功能，所以沒辦法在 BaseMIDlet 中根據類別名稱把物件建立起來。所以，在此類別中，我們提供一個 init()函式，交由上層進行覆寫，在 init()函式中把物件建立起來。以五子棋為例，如圖八所示，由 Five_MIDlet 繼承 BaseMIDlet，並覆寫 init()函式。

3.2.1.2 BaseClilet 類別

此類別是整個遊戲的核心，遊戲的邏輯及規則，還有遊戲的狀態、資訊，都存放在此。但由於各種棋類、牌類的邏輯及規則都不同，且遊戲的狀態、資訊，例如棋盤的大小、棋盤上有什麼棋子、棋子的資訊等等，也都不同，所以上層必須提供一個類別繼承此類別，並且撰寫自己的邏輯及規則等。以五子棋為例，如圖八所示，由 Five_Clilet 繼承 BaseClilet。

我們在 2.3 節曾經介紹過手機棋類和牌類遊戲的共同情境。由於我們的平台也是照這樣的情境去設計，所以，在 BaseClilet 類別中，存在一些函式，而這些函式就是我們平台所提供的既定流程。



圖九、核心模組的 BaseClilet 類別函式對照遊戲情境圖

如上圖九，在遊戲做初始化及載入後，呼叫 startApp()函式，將遊戲儲存在手機資料庫的資料載入遊戲中，並顯示主選單；在開始新局時，呼叫 doStartNewGame()函式，清除遊戲的紀錄，並開始遊戲；在繼續上次遊戲時，呼叫 doContinueGame()函式，直接開始遊戲；以及在遊戲結束後，呼叫 finishThisRound()函式，更新遊戲的資料到手機資料庫中等等。在這些流程的各個點上，我們提供幾個需由上層去覆寫的函式，分別是 startGame()、loadRecord()、resetRecord()、updateRecord()。其中，startGame()即是底層用來通知上層遊戲要開始了，而另外幾個函式，則是因為各種遊戲所儲存的資料不一定，例如，有的遊戲儲存分數、勝、敗、和，有的遊戲則只儲存回合、分數，所以，要載入哪些紀錄，重新設定哪些紀錄，以及更新哪些紀錄，是由上層的遊戲決定的。

3.2.1.3 BaseGameSetting 類別

此類別主要用來存放跟遊戲特性相關的規則設定，例如五子棋有

國際規、日規、簡易規等規則，或是電腦 AI 強度有強、中、弱等不同級別。而此類別中最主要存在一個型態為 int 的變數，讓我們用來做這些設定。以剛剛所提的例子為例，我們可以假定變數的第 0 個 bit 代表國際規，第 1 個 bit 代表日規，第 2 個 bit 代表簡易規，第 3 個 bit 代表 AI 強，第 4 個 bit 代表 AI 中，第 5 個 bit 代表 AI 弱，這樣的話，如果此變數被設定為 9，即是第 0 個和第 3 個 bit 是 1，表示遊戲是使用國際規，且電腦 AI 是中。由於各種遊戲有不同的特性或規則，所以上層必須繼承此類別，並妥善運用此類別所提供的變數來做設定。以五子棋為例，如圖八所示，由 Five_GameSetting 繼承 BaseGameSetting。

我們把這個類別獨立出來的原因是，各種類型的遊戲都有不同的遊戲設定，且我們不希望將所有的功能全部擠在 BaseClilet 類別裡。

3.2.1.4 BasePlayer 類別

此類別主要用來存放跟玩家相關的一些資訊，例如，玩家的名字、暱稱、分數、勝、敗、和等等。上層可以透過 BaseClilet 類別取得此類別的物件參考，並加以使用，如圖八所示。

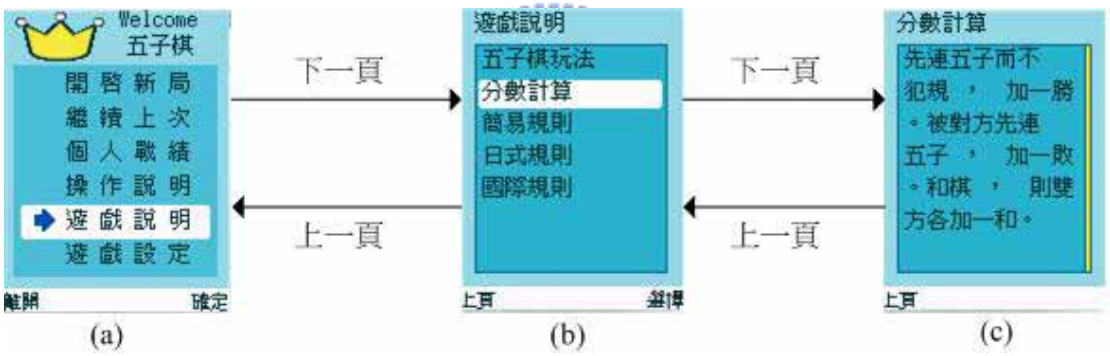
3.2.2 圖形模組 (GUI Module)

圖形模組主要跟遊戲畫面顯示相關。由於我們平台，是依照 2.3 節曾經介紹過的手機棋類和牌類遊戲的共同情境去設計，所以我們提供幾個在這些情境下設計遊戲會用到的畫面，方便遊戲開發者使用，例如，遊戲的主選單、遊戲的說明選單、遊戲的初始載入畫面、遊戲

的顯示分數畫面等等。除此之外，還包含核心模組和圖形模組之間的介面，以及因手機裝置特性不同而有的介面。

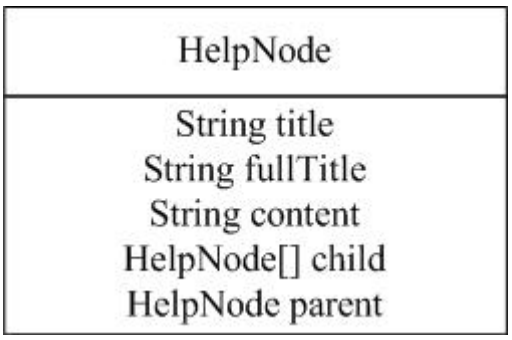
3.2.2.1 HelpMenu 類別(遊戲的說明選單)

此類別是用來支援顯示遊戲的說明選單。如圖十所示，這是五子棋的遊戲說明選單，共有五項說明，分別是五子棋玩法、分數計算、簡易規則、日式規則、國際規則等。當然，不同的遊戲有不同的遊戲說明，這就要看上層的遊戲開發者所要列出來的項目而決定。



圖十、五子棋的遊戲說明選單

遊戲的說明選單是一個階層式的架構，為了讓上層的遊戲開發者能輕易的建立出這樣的選單，我們將遊戲說明選單建構成樹狀的架構。樹的每個節點，包含五個參數，如圖十一所示。其中，title 是用來顯示選單的標頭資訊，如圖十(b)中的「遊戲說明」；為了彈性，fullTitle 是附加在標頭資訊後面的文字；content 則是在樹葉節點時，用來顯示說明內容的文字，如圖十(c)中的「先連五子…」這段文字；child 是用來連到下一頁的一群子節點；parent 則是父節點，用來返回上一頁。



圖十一、遊戲說明選單的 HelpNode 類別

以五子棋的說明選單為例，我們可把圖十的(b)看成是根節點，這個根節點的 title 是「遊戲說明」，沒有 fullTitle、content 及 parent，但有五個子節點，而這些子節點的 title 分別是五子棋玩法、分數計算、簡易規則、日式規則及國際規則。對分數計算這個樹葉節點而言，title 是「分數計算」，content 是「先連五子…」這段文字，如圖十的(c)所示。

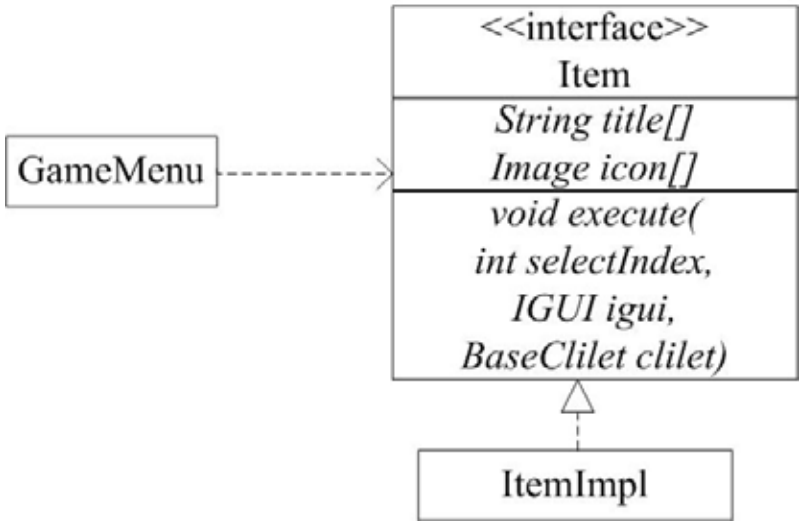
另外，HelpNode 類別也提供 setTitle()、addChild()等函式，來建立遊戲說明選單。因此，上層的遊戲開發者，只要使用 HelpNode 類別，就可以方便地建立遊戲說明選單。

3.2.2.2 GameMenu 類別(遊戲的主選單)

此類別是用來支援顯示遊戲的主選單，當遊戲起始並且載入完成之後，會進入遊戲主選單，讓玩家選擇接下來的動作，例如，要開啟新局或觀看個人戰績等等。如圖十二的五子棋遊戲主選單，包含六個選項，分別是開啟新局、繼續上次、個人戰績、操作說明、遊戲說明及遊戲設定。不過，由於各種遊戲的主選單選項可能都不同，因此我們在設計時，必須考慮讓上層遊戲設計者能自行決定這些選項以及他們所對應的動作等等。

圖十二、五子棋的遊戲主選單

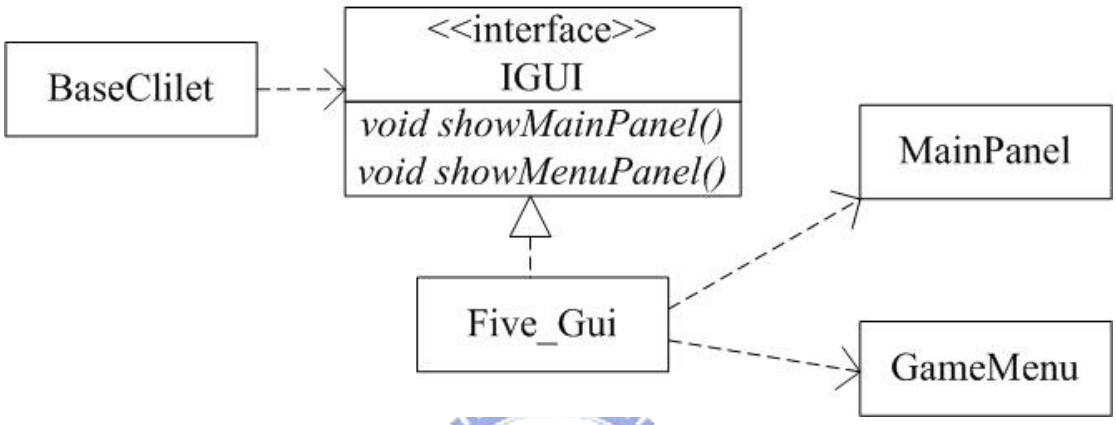
此類別可設定畫面上方所要顯示的圖形及標題，如圖十二中的皇冠圖形及 Welcome、五子棋標頭。最重要的是，可自行設定選單中的選項，也就是說這些選項都是可插入的，由上層的遊戲開發者自行決定主選單會有那些項目。而每一個項目，都包含項目的名稱、小圖示及相對應要執行的動作。這些可插入的選項是透過一個 Item 介面來完成，如圖十三所示。這個介面包含了名稱的陣列及小圖示的陣列，還有一個相對應執行的函式，其中，這個函式有一個索引值 selectIndex 是用來決定當玩家選擇那一個選項時所要執行的動作，例如，若第一個選項是開啟新局，則索引值 0 代表執行開啟新局的動作等等。上層的遊戲開發者必須實作這個介面，來定義這些選項。



圖十三、遊戲主選單的類別圖

3.2.2.3 IGUI 介面(Game GUI Interface)

由於遊戲中存在著遊戲的主選單及遊戲的主畫面，且不同遊戲的主選單及主畫面都不同，所以我們必須提供一個 GUI 的介面，讓不同的遊戲去實作這個介面，決定遊戲所要顯示的主選單及主畫面。

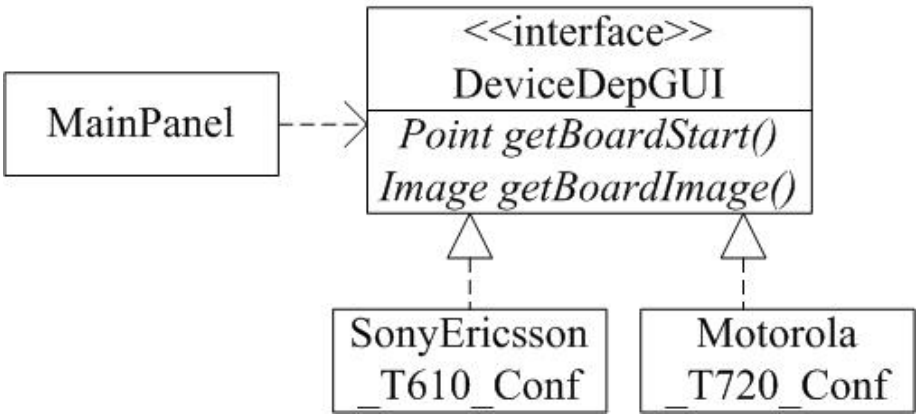


圖十四、遊戲 GUI 的介面

如上圖十四所示，以五子棋為例子，當底層要通知上層的遊戲顯示主畫面或主選單時，只需要呼叫 IGUI 介面，而真正的實作，由 Five_Gui 去決定。

3.2.2.4 Device-dependent GUI 介面

由於不同手機裝置的螢幕大小不一樣，所以遊戲畫面中元件的位置或圖形可能會做調整。為了讓同一種遊戲的不同畫面能較方便的在不同的手機上顯示，我們建議上層的遊戲開發者定義一個 DeviceDepGUI 介面，這個介面定義許多遊戲畫面中的位置和顯示的圖形。然後，再由實作些介面的類別去決定真正要顯示的位置和圖形。



圖十五、Device-dependent GUI 介面

如上圖十五所示，以五子棋為例，當畫面要顯示在 Motorola T720 的手機時，就定義一個 Motorola_T720_Conf 的類別，來實作這個介面；當畫面要顯示在 Sony Ericsson T610 的手機時，就定義一個 SonyEricsson_T610_Conf 的類別，來實作這個介面。由實作的類別去決定畫面中元件所要顯示的真正位置和圖形，例如，棋盤的位置、棋盤的圖形、棋子的相對位置、棋子的圖形等等。

3.2.2.5 其它

除上述所介紹的元素之外，圖形模組還包含了遊戲的載入畫面及顯示分數畫面。但是，由於這二個畫面比較簡單，在此我們只做簡單的介紹。

遊戲的載入畫面，就是遊戲在做初始化動化時，玩家所會看到的畫面，通常我們會看到畫面有一條 bar 在跑，從 0%到 100%，其中 100%代表載入完成。而之所以需要這個畫面的原因是，因為手機上的處理器速度較慢，記憶體也較少，遊戲初始化的動作會比較慢，為了不讓使用者以為遊戲當機或發生問題，所以使用這個載入畫面讓玩家知道目前遊戲的載入進度。

遊戲的顯示分數畫面，則包含幾個顯示分數的欄位，例如，顯示分數、勝、敗、和等。而這二種畫面的實際例子，如圖十六所示。

圖十六、遊戲的載入畫面和顯示分數畫面

3.2.3 儲存模組 (Storage Module)

儲存模組主要是用來幫助上層的遊戲開發者，將遊戲所需要儲存的資料儲存在手機的資料庫裡。

手機要儲存資料有幾個問題：(1)MIDP 並不提供 JDBC 的功能，也就是說，無法在手機上直接跟遠端的資料庫進行連線，所以我們若想要將資料儲存在遠端的資料庫的話，中間必須有一個 gateway 伺服器來做這件事，手機透過 socket 將資料傳給 gateway，再由 gateway 幫忙將資料儲存到遠端的資料庫，當然，這是非常不方便的。(2)MIDP 不提供讀寫檔案的功能，也就是說，我們無法將資料儲存在檔案裡面。

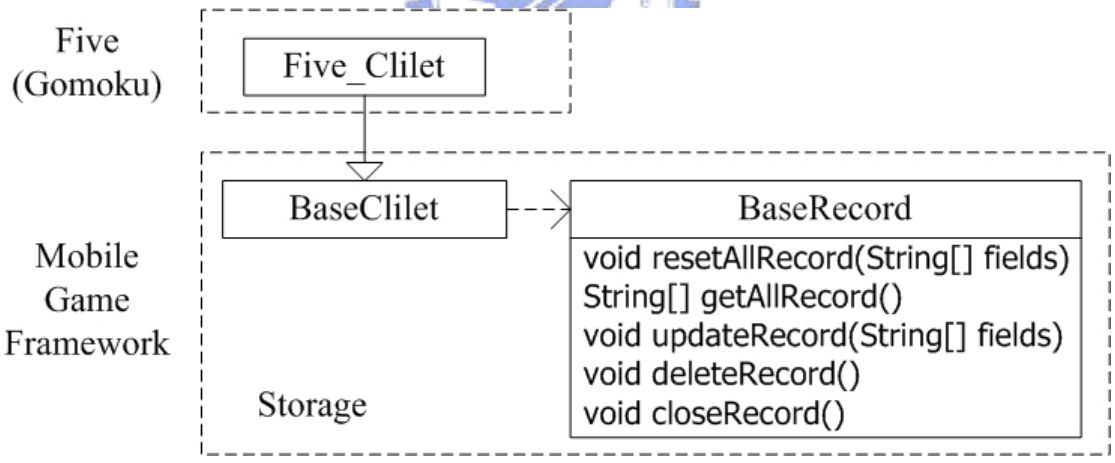
不過，MIDP 提供了記錄管理系統(Record Management System，簡稱 RMS)[4][5][12]，而這個系統可視為手機上的簡易資料庫，允許我們將資料儲存在手機上。RMS 擁有許多的資料表格，而每一個表格又可以儲存許多筆資料，我們可以指定名稱來開啟一個新的表格或舊的表格。但是， RMS 使用上並不方便，例如，要對一個表格進行

操作時，有下列幾個函式：

```
int addRecord(byte[] data, int offset, int numBytes)
byte[] getRecord(int recordId)
void deleteRecord(int recordId)
```

每一個表格可以儲存多筆的資料，而每一筆資料都有一個 recordID，來紀錄它在表格中被儲存的位置，以便下次再根據這個 recordID 取出這筆資料。另外，每一筆資料都以 byte 的型式儲存在資料庫中。所以，我們在對表格進行操作時，還得去記住 recordID，還得將資料轉成 byte 陣列，這對遊戲開發者來說，是一件不方便的事情。

為了讓遊戲開發者更容易使用 RMS，我們設計了一個 BaseRecord 類別，這個類別提供一些對資料庫表格進行操作的函式，讓使用者不必去管理 recordID，也不必將資料轉換成 byte 陣列。我們所提供的一些函式操作，及跟底層的關係，如圖十七所示。



圖十七、儲存模組的類別圖

另外，由於不同遊戲使用的資料庫表格名稱不一樣，所以，我們在 BaseClilet 類別裡提供了下面的函式，讓上層去覆寫：

```
String getRecordStoreName()
```

由上層去決定所要開啟的表格名稱，並透過 BaseClilet 來取得 BaseRecord 的物件參考。

我們曾經在核心模組的 BaseClilet 類別裡介紹幾個上層需要覆寫的函式，其中幾個跟儲存資料相關的函式，分別是 loadRecord()、resetRecord()、updateRecord()，在此，我們舉五子棋為例，以 updateRecord()來說明如何覆寫這些函式，及如何使用我們所提供的 BaseRecord 類別：

- 在 Five_Clilet 裡覆寫 getRecordStoreName()函式，並且回傳所要開啟的資料表格名稱，以便讓底層知道所要開啟的表格名稱：

```
/** 回傳要開啟的資料庫表格名稱 **/
public String getRecordStoreName(){
   return "Five_Record";
}
```

- 在 Five_Clilet 裡覆寫 updateRecord()函式，並且使用 BaseRecord 類別所提供的函式，將所要儲存的資料，儲存到資料庫的表格中：

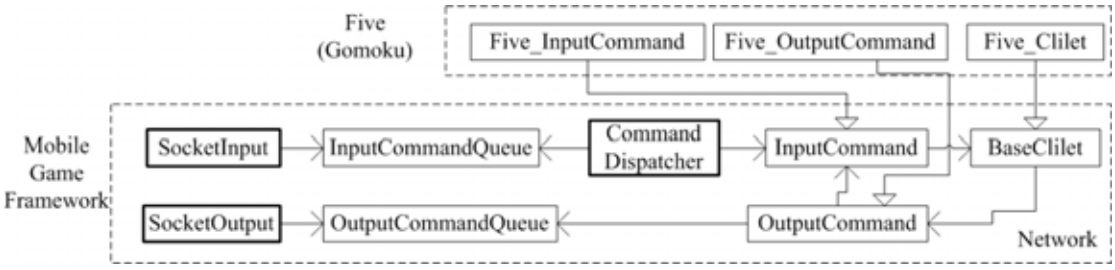
```
/** 儲存遊戲的規則、回合、勝、敗、和 **/
public void updateRecord(){
   String rule =
      Integer.toString(((Five_GameSetting)gs).getRule(
      ));
   String round = Integer.toString(this.round);
   BasePlayer player = this.getUser(getSelfSeatID());
   String win = Integer.toString(player.getWin());
   String draw = Integer.toString(player.getDraw());
   String lose = Integer.toString(player.getLose());
   String[] params = {rule,round,win,draw,lose};
   this.getRecord().updateRecord(params);
```

}

3.2.4 網路模組 (Network Module)

網路模組主要用來負責在手機和遊戲伺服器之間傳送命令。所謂的命令，以五子棋來說，就是在棋盤中的某個位置下了一顆子；以麻將來說，就是打了一張牌或碰一張牌等等。另外，由於遊戲必須一邊執行命令，一邊從網路接收或傳送命令，為了不影響遊戲的進行，我們將會採用 non-blocking 的設計方式[7][8]。而為了能夠達到 non-blocking 的效果，我們將會使用多執行緒 multi-thread 來做處理。

對於每一個遊戲的命令而言，都包含了命令的名稱，以及一個參數的陣列。由於網路模組所包含的類別相當的多，如圖十八，為了方便介紹，我們將分為二個不同的流程，來介紹我們的網路模組：一個是命令的接收流程，另一個是命令的傳送流程。



圖十八、網路模組的類別圖

3.2.4.1 遊戲命令的接收流程

首先，SocketInput 類別是一隻執行緒，它會不斷的從遊戲伺服

器端接收 byte 型態的資料，當它判斷接收的資料可以形成一個遊戲命令時，便利用 InputCommandQueue 所提供的函式，將遊戲命令丟到 InputCommandQueue。其中 InputCommandQueue 是一個佇列的資料結構，是用來負責存放一堆 Command，而 Command 類別則包含命令的名稱及一個參數的陣列。而 SocketInput 要如何判斷是否形成一個命令，我們是使用「|」字元來分隔命令，當 SocketInput 檢查到「|」字元時，就知道這是一個命令的結束符號了。

接著，CommandDispatcher 類別也是一隻執行緒，它會不斷的從 InputCommandQueue 裡去檢查是否有未執行的 Command，若有的話，就取出 Command，並呼叫 InputCommand 去執行，而在 InputCommand 裡有個重要函式：

```
void dispatchCommand(String cmd,String[] params)
```

其中 cmd 是命令的名稱，而 params 是命令所夾帶的參數。由於 J2ME 並不支援 Reflection 的功能，且各個遊戲的命令都不盡相同，所以我們沒有辦法在底層的 InputCommand 裡，根據函式名稱和參數，使用 Reflection 去動態的呼叫函式。因此，上層的遊戲必須覆寫這個函式，並根據命令的名稱來呼叫適當的函式執行。

我們使用五子棋的下子命令來說明上述介紹流程及一系列相關的執行函式。

◆ 在 CommandDispatcher 裡，負責不斷的檢查 InputCommandQueue 是否有 Command，若有則取出，並丟給 InputCommand 去執行，否則就 wait：

```
public void run(){
   while(true){
      Command cmd = inputCommandQueue.retrieval();
      if(cmd!=null){
```

```
            inputCommand.dispatchCommand(
                cmd.getCommandName(),cmd.getArgs());
        }else{
            wait();
        }
    }
}
```

◆ 在 Five_InputCommand 裡覆寫 dispatchCommand()函式，利用字串比對的方式檢查命令的類型，並且呼叫適當的函式執行：

```
/** 檢查命令的類型，並且呼叫適當的函式 **/

public void dispatchCommand(String cmd,String[] params){
    if(cmd.equals("ply_placePieceOnBoard")){
        ply_placePieceOnBoard(params);
    }else{
        super.dispatchCommand(cmd,params);
    }
}
/** 執行下子的動作 **/
public void ply_placePieceOnBoard(String[] params){
    ((FiveClilet)clilet).placePieceOnBoard(
                        Integer.parseInt(params[0]),
                        Integer.parseInt(params[1]));
}
```

3.2.4.2 遊戲命令的傳送流程

基本上，遊戲命令的傳送流程，剛好就跟接收流程相反。由於不

同的遊戲有不同的命令，所以上層必須繼承 OutputCommand 類別，撰寫自己的命令，並且使用 OutputCommand 類別所提供的函式：

```
void toAllClilets(String cmd,String[] params)
```

將遊戲的命令丟到 OutputCommandQueue，而 OutputCommandQueue 和 InputCommandQueue 是屬於同一類型的資料結構。

接著，SocketOutput 是一隻執行緒，它會不斷的去檢查 OutputCommandQueue 中是否有命令，若有的話就取出命令，並將資料以 byte 型態傳到遊戲的伺服器。

同樣的，我們使用五子棋的下子命令來說明上述介紹流程及一系列相關的執行函式。

◆ 在 Five_Clilet 裡，呼叫 FiveOutputCommand 對應的命令：

```
/** 五子棋的下子動作 **/

void requestPlacePieceOnBoard(int targetX, int targetY){
   ((FiveOutputCommand)outputCommand).requestPlacePiec
eOnBoard(targetX, targetY);
}
```

◆ 在 Five_OutputCommand 裡，利用底層所提供的函式，將命令傳給所有的玩家：

```
void requestPlacePieceOnBoard(int targetX, int targetY){
   String[] params = {Integer.toString(targetX),
                      Integer.toString(targetY)};
    //將命令傳給所有的玩家
   this.toAllClilets("ply_placePieceOnBoard",params);
}
```

◆ 在 OutputCommand 裡，將命令包起來，並丟到 OutputComamndQueue：

```
public void toAllClilets(String cmd,String[] params){
   outputCommandQueue.append(cmd,params);
}
```

3.2.5 工具模組 (Utility Module)

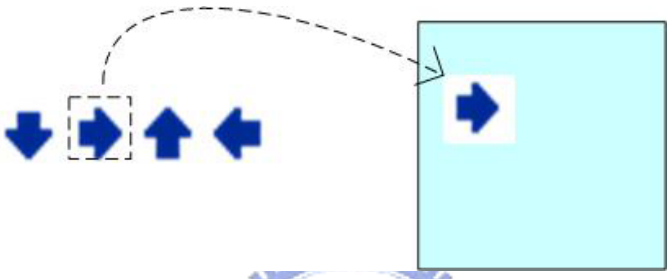
目前在我們的工具模組裡面，包含二個工具類別，一個是用來做切割圖形的，另一個是用來播放聲音的，下以我們會分別對這二個工具類別做詳細介紹。

3.2.5.1 切割圖形

由於每一個圖形檔案都需要一個 header 來紀錄圖形的資訊，所以太多數量的圖形檔案，會使得整個應用程式的檔案大小變大，所以，若能將圖形檔案內容相似的圖形合併成一張圖，即可省掉許多的 header 資訊，之後再使用程式去做切割圖形畫圖的動作，這樣即可以節省不少的圖形檔案的大小，例如，四張圖形內容分別為上、下、左、右的圖檔大小，會比一張圖形內容同時具有上、下、左、右等圖形的圖檔大小要來的大。

由於 MIDP 的 Graphics 類別支援比 J2SE 的 Graphics 類別來的少，所以沒有辦法直接使用 Graphics 類別所提供的 drawImage()函

式，來做切割圖形的動作。但是，MIDP 的 Graphics 類別提供了一個 setClip()函式，用來設定螢幕上的可視區域，讓我們間接的達到切割圖形的目的[4][5][12]。所以，我們運用了此函式，再加上一些畫圖的方式，實作了一個用來切割圖形的 ClipImage 類別。而使用此類別來切割圖形的概念，如圖十九所示，將一個含有上、下、左、右箭頭的圖形，做切割之後，將向右箭頭的圖形，畫在螢幕中的可視區域。除此之外，我們所提供的 ClipImage 類別，可以同時切割水平方向及垂直方向。



圖十九、切割圖形

◆ 使用方式(以圖十九為例，所以請對照圖十九)：

```
Image arrow; //含有上、下、左、右箭頭的圖形
ClipImage CArrow;  //切割後的圖形
/** 將箭頭的圖形，做好初始化及切割的動作 **/
private void initBoard(){
   screenWidth = getWidth();//取得畫面大小的寬
   screenHeight = getHeight();//取得畫面大小的高
   try{
       arrow = Image.createImage("/arrow.png");
   } catch(Exception e){}
   //水平方向切成 4 張，垂直方向切成 1 張
   CArrow = new ClipImage(arrow, 4, 1,screenWidth,
                                       screenHeight);
}
/** 將圖形中的第二個箭頭畫到螢幕的(20,30)位置 **/
```

```
private void drawSecondArrow(Graphics g){
   CArrow.draw(g,20,30,2);
}
```

3.2.5.2 播放聲音

目前，不同版本的MIDP支援播放聲音的方式都不同[4][5][12]。在MIDP 1.0版本裡面，只支援五種預設的播放音效，分別是INFO、WARNING、ERROR、ALARM及CONFIRMATION。這些音效的功能各不相同，例如，WARNING是用來提示使用者，他所下的操作指令可能具有潛在的危險；ERROR是用來提示使用者，這個操作指令是錯誤的；…等等。不過，這些聲音是MIDP 1.0的API所下的定義，當然我們也可以任意使用它，而不考慮是在什麼情況下使用。

到了MIDP 2.0之後，多了用來支援播放不用類型聲音的API，例如，支援Wave、Au、Mp3、Midi、Tone sequences等等不同的聲音格式。除了支援的格式外，MIDP 2.0播放聲音的方式，也跟MIDP 1.0完全不同。在MIDP 2.0裡，要開啟聲音檔案的串流，並且指定聲音檔案的格式才能播放。

為了同時能夠支援MIDP 1.0及MIDP 2.0的播放方式，並且讓上層的遊戲開發者省去開啟聲音檔案的串流、指定聲音檔案格式等等的工作，我們提供一個用來播放聲音的Sound類別，讓開發者可以更輕易的播放聲音。目前我們在Sound類別裡提供二個主要的API函式：

```
void playDefaultSound(int index)
void playSound(int index)
```

其中，第一個函式是用來播放MIDP 1.0的五種預設音效，index的值0到4分別代表ALARM、CONFIRMATION、ERROR、INFO及WARNING。

第二個函式是用來播放開發者所指定的聲音檔案。

◆ 使用方式，將聲音檔案建成一個陣列，並傳入 Sound 類別的建構子，即可利用它來播放聲音：

```
String[] name = {"/1.wav","/2.au","/3.mp3","/4.midi"};
Sound sound = new Sound(name,midlet);
sound.playDefaultSound(0);//播放 ALARM 聲音
sound.playDefaultSound(4);//播放 WARNING 聲音
sound.playSound(0); //播放 1.wav
sound.playSound(3); //播放 4.midi
```

第四章、實作及相關問題探討

本章將會說明我們的實作環境，包括我們所使用的模擬器、手機、程式編輯器及工具，還有實作的遊戲等等。除此之外，我們還會探討縮小程式碼的一些方法及輔助工具，還有在手機上除錯所要注意的事項及除錯的方法。最後，我們會展示我們所開發的五子棋遊戲的畫面。

4.1 實作環境

所謂「工欲善其事，必先利其器」，我們希望能更方便、輕鬆的開發我們的應用程式，當然少不了工具的輔助。以下就來介紹我們的利器，以及我們所實作的遊戲。

◆ 手機模擬器

目前 Sun 公司提供了支援 MIDP 1.0 及 MIDP 2.0 的手機模擬器。各家手機的廠商也分別以 Sun 公司所提供的手機模擬器為基礎，開發支援自家手機的模擬器，為的是能夠更真實的模擬自家手機的環境，讓應用程式開發者能更輕易的在他們的手機上開發應用程式。目前我所使用的模擬器列表如下：

表格二、手機模擬器列表

Sony Ericsson J2ME SDK v1.2.09 (MIDP 1.0)
Sony Ericsson J2ME SDK v2.0.0 beta (MIDP 2.0)
Motorola SDK v4.0 for J2ME (MIDP 1.0)
J2ME Wireless Toolkit v1.0.4.01 (MIDP 1.0)
J2ME Wireless Toolkit v2.0.01 (MIDP 2.0)

表格二中的各款模擬器，分別可在各公司專屬網站上的下載區下載[9][14] [15]。

◆ 手機實機

當手機遊戲開發完成之後，我們需要下載到手機上去執行，而我們所使用的手機分別分下列二種：

- Motorola T720
  螢幕大小是 120*160，記憶體是 200KB。
- Sony Ericsson T610
  螢幕大小是 128*128，記憶體是 300KB~400KB。

◆ 程式編輯器及工具

我們所使用來開發程式的編輯器及工具為別為下列二種：

- Microsoft Visual J++ 6.0
  微軟公司所開發的一套 JAVA 開發環境。
- IntelliJ IDEA 4.0
  JetBrains 公司所開發的一套 JAVA 開發環境[16]。特別的是，它還支援 Refactoring 及 Inspect code 等功能，這對我們在做程式碼的最佳化有相當大的幫忙，我們稍後會做更詳細的介紹。

◆ 實作的遊戲

我們套用我們的平台所開發的遊戲分別為:(1)五子棋,(2)麻將。

4.2 縮小程式碼

程式碼的最佳化，包含了程式碼大小的最佳化，以及效能的最佳化[17][18]。由於手機上的記憶體有限，甚至有的手機會限制應用程式的大小，所以，在手機上要開發應用程式的話，程式碼大小的最佳

化比起效能的最佳化還要重要。因此，我們將會詳細的探討關於程式碼大小最佳化的一些方法及輔助工具。

4.2.1 縮短名稱

我們在實作程式的過程當中，通常為了程式的易維護性及可讀性，我們會將程式中的封包、類別、函式、變數等名稱，用較易懂的名稱去命令，因此，這些名稱會變的比較長。倘若我們能將這些名稱的長度縮的較短，例如，只用一、二個字的長度來命名，我們將能節省下不少的程式碼大小。

目前市面上有不少的混淆器(Obfuscator)，例如，RetroGuard[19]及 ProGuard[20]等等。這些工具都可以 Plug-in 的方式，跟 Wireless Toolkit 結合使用。而這些所謂的混淆器，即是用來幫我們將程式中的變數、函式、類別等名稱縮短。舉一個簡單的例子，如圖二十所示，這是使用 ProGuard 混淆器所得到的結果，我們可以看到，程式中所有的類別、函式、變數等名稱，通通變成以一個字元為命名，比起原本大約十幾個字元的名稱來說，節省了相當多的字元。

```
Image arrowImage;
ClipImage clipArrowImage;
private void initBoard(){
  screenWidth = getWidth();
  screenHeight = getHeight();
  try{
    arrowImage =
      Image.createImage(
        "/arrow.png");
  } catch(Exception e){}
  clipArrowImage = new
    ClipImage(arrowImage, 4, 1,
      screenWidth, screenHeight);
}
```

After Obfuscation →

```
Image K;
b n;
private void j(){
  f = getWidth();
  e = getHeight();
  try{
    K = Image.createImage(
      "/arrow.png");
  } catch(Exception e){}
  n = new b(K, 4, 1, f, e);
}
```

圖二十、程式使用 ProGuard 縮短名稱的結果

當然，以圖二十的例子來看，大家可能還看不出整體的效果，所以，我們以五子棋和麻將整個應用程式的大小為例子，來試試看使用混淆器可以節省的比例。

表格三、縮小程式碼-使用混淆器

	原來程式碼大小	經過使用混淆器(ProGuard)	節省的大小	節省的比例
五子棋	99462 Bytes (98KB)	76502 Bytes (75KB)	22960 Bytes (23KB)	23.084%
麻將	137343 Bytes (135KB)	103107 Bytes (101KB)	33077 Bytes (33KB)	24.083%

如表格三所示，我們可以看到，五子棋和麻將的程式碼經過使用混淆器之後，分別節省了23.084%及24.083%，節省的比例可以說相當的大，也証實了使用混淆器的效果的確是非常明顯且有幫助的。

4.2.2 合併類別及移除介面

物件導向的程式設計傾向於使用更多的繼承和介面，但是使用更多的繼承和介面也代表著程式的類別會愈多，愈多的類別則程式碼的大小也會變大。因此，為了節省程式碼的大小，我們不得不做一些改變，希望能減少類別的數目或介面的數目，期望能縮小一些程式碼的大小。

合併繼承關係的這個主題，目前有不少的論文有研究[21][22]

[23]，但都不是使用 JAVA 語言為基礎。因此，我們使用 J2ME 平台所開發的程式碼，沒有辦法有一個完全自動化的工具，來幫我們完成合併類別的繼承關係。在 Refactoring[24]這個領域有二個方法，分別是 Pull Members Up 和 Push Members Down，來幫我們將程式中的成員從子類別移到父類別或從父類別移到子類別，進而達成合併類別的目的。

JetBrains 所開發的 IntelliJ IDEA 這套開發環境，有支援許多 Refactoring 的功能，包括我們剛剛所提到的 Pull Members Up 及 Push Members Down，我們可以選擇要將某一類別中的變數或函式，移到另一個類別中，在確定這樣的動作之後，IntelliJ IDEA 會給我們警告的訊息，確定我們是否要這樣做或這樣做有哪些潛在的危險。

同樣的，我們以五子棋和麻將為例，分別對程式中所存在的繼承關係進行合併，並且移除所有的介面使用。我們所得到的結果，如表格四所示，雖然節省的程式碼大小約僅 4KB，但是我們要知道，手機上的記憶體很小，差個 1、2KB 可能就無法下載，也因此將近 4KB 的節省大小，也算是小有貢獻的了。

表格四、縮小程式碼-合併類別及移除介面

	原來程式碼大小	合併類別及移除介面	節省的大小	節省的比例
五子棋	99462 Bytes (98KB)	95972 Bytes (94KB)	3490 Bytes (4KB)	3.509%
麻將	137343 Bytes (135KB)	133870 Bytes (131KB)	3473 Bytes (4KB)	2.529%

4.2.3 移除多餘的程式碼

我們在實作程式碼的時候，有時候我們所撰寫的函式、變數、宣告等的程式碼，沒有被用到，但也忘了要移除。久而久之，這類型的程式碼愈來愈多，累積在程式中，可能是不小的負擔，雖然不會對應用程式的執行造成任何的影響，但是卻會增加應用程式的大小，這對記憶體較小的手機來說，我們當然希望它們不要存在比較好。另外，我們所開發的平台，可能會針對不同類型的遊戲，開發了一些專門給這些不同類型的遊戲使用的程式碼，也因此，對某一類型的遊戲而言，他們不會用到專門給別的類型所使用的程式碼，而這些程式碼理所當然的，也變成了多餘的程式碼。

除此之外，由於我們使用模擬器開發應用程式時，也經常會印出一些用來 Debug 的訊息，而這些用來印出 Debug 訊息的程式碼，到了手機上，也都變成了多餘的程式碼，這也是必須要移除的一部份。

JetBrains 所開發的 IntelliJ IDEA 這套開發環境，也支援了 Inspect code 的功能，可以讓我們檢查整個應用程式的程式碼中，是否有那些程式是多餘的。

表格五、縮小程式碼-移除多餘的程式碼

	原來程式碼大小	移除多餘的程式碼	節省的大小	節省的比例
五子棋	99462 Bytes (98KB)	97230 Bytes (95KB)	2232 Bytes (3KB)	2.244%
麻將	137343 Bytes (135KB)	135101 Bytes (132KB)	2242 Bytes (3KB)	1.632%

表格五為五子棋和麻將，經過使用 IntelliJ IDEA 的 Inspect code 功能，移除多餘程式碼所得到的結果。而我們所用來檢查的項目如表格六所示：

表格六、移除多餘程式碼檢查的項目

Unused method parameters
Unused method return value
Empty method
Unused declaration
Unused assignment
Redundant type cast

這些檢查的項目，皆可在 Inspect code 的畫面中選擇勾選或是不勾選。

4.2.4 移除網路功能

在 MIDP 2.0 版本才有支援使用 Socket 連線，所以必須手機是使用 MIDP 2.0 版的才可以使用本平台的網路功能。並且，並不是每個遊戲都會用到網路功能，有的遊戲可能要開發成「單機版」，也就是讓玩家跟電腦的 AI 玩遊戲。因此，在不需要使用網路功能的情況下，將網路功能移除，也能讓我們應用程式的大小變小。

表格七、縮小程式碼-移除網路功能

	原來程式碼大小	移除網路功能	節省的大小	節省的比例

五子棋	99462 Bytes (98KB)	94460 Bytes (93KB)	5002 Bytes (5KB)	5.029%
麻將	137343 Bytes (135KB)	132341 Bytes (130KB)	5002 Bytes (5KB)	3.642%

由於網路功能是由我們的平台所提供，因此不管所開發的遊戲是什麼，移除網路功能所節省的大小會是一樣的，如表格七所示。

4.2.5 綜合使用

前面幾小節所提到的一些關於縮小程式碼的方法，都有其各自的功效。當然，若我們能將這幾個方法結合起來使用，所節省的程式碼大小，將會是相當的可觀。

表格八、縮小程式碼-綜合使用

	原本程式碼大小	(a)經過使用混淆器	(b)合併類別及移除介面	(c)移除多餘程式碼	(d)移除網路功能	綜合使用(a)(b)(c)(d)	綜合節省大小	綜合節省比例
五子棋	99462 Bytes (98KB)	76502 Bytes (75KB)	95972 Bytes (94KB)	97230 Bytes (95KB)	94460 Bytes (93KB)	68494 Bytes (67KB)	30968 Bytes (31KB)	31.1%
麻將	137343 Bytes (135KB)	103107 Bytes (101KB)	133870 Bytes (131KB)	135101 Bytes (132KB)	132341 Bytes (130KB)	95619 Bytes (94KB)	41724 Bytes (41KB)	30.4%

表格八為使用各縮小程式碼方法的比較，包括綜合使用的結果。值的一提的是，綜合使用這些方法的結果，的確是讓我們節省了相當多的程式碼。五子棋和麻將分別節省了 31.1%和 30.4%，這相當於節省了將近原本程式碼的三分之一。

因此，有了這些用來縮小程式碼的方法，我們在開發應用程式或遊戲的時候，就能盡情的發揮實作能力，不怕程式碼中含有太多的繼承關係或介面，也不怕程式碼中的多餘程式碼過多，只要最後透過這些方法，我們就能縮小程式碼。

4.2.6 其它

除了上述所提到的方法之外，尚有一些值得我們注意，可以用來縮小應用程式大小的方法。

4.2.6.1 合併圖形

我們在 3.2.5 節曾經提到過，過多的圖形，會增加應用程式的大小，若能將同一類型的圖形合併成一張圖，然後再使用程式的技巧將圖切割使用，將能夠節省一些應用程式的大小。這在應用程式需要使用大量圖形的時候，尤其有用。

4.2.6.2 多使用函式庫所提供的類別

通常為了方便，我們會盡量開發一些好用的工具類別來使用，但無形中，這些工具類別，也經常是應用程式變大的一個原因，也因此我們建議開發人員要盡量使用函式庫所提供的類別。例如，MIDP 不支援 List 這種資料結構，若在不管效能的情況下，我們可以考慮使用另一種資料結構 Vector 來取代 List。

4.3 除錯

由於手機是小型裝置，礙於種種的硬體限制，再加上螢幕的解析度也比較小，顯示錯誤資訊的能力有限，要在手機上開發應用程式，除錯的工作實為一項不是很簡單的任務。接下來，就讓我們來看看，在手機上除錯的一些問題，及解決方法。最後，並根據我們實作的經驗及所遇到的問題，提供一些關於除錯的建議，以及用來輔助除錯的工具類別。

4.3.1 手機上除錯的困難

首先，手機裝置的畫面非常小，因此，手機並沒有一個像 JAVA Console 的除錯畫面，可以用來顯示除錯訊息。接著，手機在應用程式發生錯誤時，提示的錯誤訊息是非常有限的，例如，我們曾經發生下列幾個狀況：

- Motorola T720 並不支援使用 Object 類別的 Clone()函式，遊戲進行下載時，會顯示「無法安裝」。因此，我們必須檢視所有的程式碼，慢慢的進行除錯。可想而之，我們不知道問題的來源，且僅僅因為一行程式碼，就得花上很多的時間去進行除錯。

- Sony Ericsson T610 有圖檔大小的限制，也就是說它只支援圖檔大小在(255,255)解析度內的圖。若圖的大小超過這個範圍，就會顯示「應用程式發生錯誤」。當然，光是這樣的提示也是非常不足的，為了找出這個錯誤，我們也是花了相當多的時間在檢視程式碼的各個部份。

- 開發五子棋遊戲時，使用太多的 Vector，結果在 Sony Ericsson T610 執行時，發生「應用程式消耗過多記憶體」的錯誤訊息。當然，我們也是到最後才發現，原來是使用了太多的 Vector，於是便修改程式的寫法。

以上我們所介紹的曾經遇過的問題，可能只是一小部份，其它的問題還是需要更多的實作才能發現。不過，光由這些難解的問題，我們就知道，光是要找出問題可能就需要一些時間，再加上每隻手機的支援和限制又各不一樣，會發生什麼樣子的問題，甚至都難以預測。

4.3.2 使用模擬器

為了減少這些錯誤的發生，以及希望能在下載到手機執行前，就能將程式中所隱藏的 bug 除掉，我們當然要使用模擬器來開發應用程式。雖然說模擬器還是無法完全真正的模擬手機上的環境，不管是手機的作業系統、記憶體、處理器速度，連虛擬機器也一樣，但是，預

先使用模擬器來開發應用程式，至少能讓我們先找到一些隱藏在程式中的 bug。

有了模擬器後，我們在開發應用程式時，便能印出一些錯誤訊息或例外，顯示在模擬器的印訊息畫面，讓我們能較明確的知道程式發生錯誤的地方。在程式開發的階段，有個建議的除錯方法，就是使用一個除錯的參數，例子如下：

```
Boolean debug = true;
If(debug) System.out.println(“Print Something!”);
```

當 deubg 為 true 時，就印出訊息，否則，即不印出。這可以讓我們很方便的決定那些訊息是要印出，那些是不要。

4.3.3 除錯的注意事項

為了讓遊戲的開發者，在不同手機上開發遊戲時，能夠減少問題的發生。我們根據我們實作的一些經驗，提供一些準則，來做為參考：

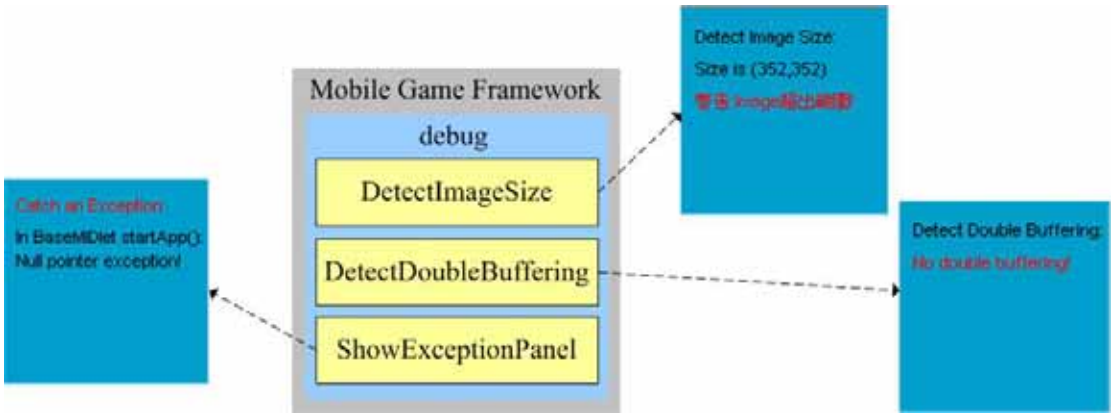
- 檢查應用程式的一些限制，例如，圖形大小的限制、Vector 使用的限制、應用程式大小的限制等等。
- 檢查類別函式庫函式的使用，例如，Object 的 Clone()函式。
- 檢查 GUI 是否需要自己做 Double buffering。

4.3.4 除錯的工具類別

為了讓遊戲的開發者方便除錯，我們也提供了一些用來除錯的工具類別：

- DetectImageSize
  自動偵測手機所支援的圖檔解析度大小。
- DetectDoubleBuffering
  自動偵測此手機是否支援使用 Double Buffering。
- ShowExceptionPanel
  將程式抓到的例外訊息，顯示在手機螢幕上。

以上這些工具類別都會將訊息顯示在螢幕上，如圖二十一所示。



圖二十一、除錯的工具類別及使用畫面

我們以 ShowExceptionPanel 為例，來說明它的使用方法，例子如下：

```
public void test(){
    try{
        //do something…
    }catch(Exception e){
        new ShowExceptionPanel(baseMidlet,”In Five_Clilet
                                test()”+e.toString());
    }
}
```

我們在程式中發生例外的地方，使用 ShowExceptionPanel，並將錯

誤訊息的參數傳進去，當例外真的發生的時候，螢幕就會顯示「In Five_clilet…」這段例外訊息，我們即可藉此知道例外發生的地方，並試著找出原因。

第五章、結論與未來發展

在本論文當中，我們設計了一個手機遊戲的發展平台，這個平台主要是用來發展棋類、牌類的手機遊戲。此發展平台包含三個主要部份。

- 一個手機遊戲程式設計的架構
  這個架構主要是希望能讓遊戲開發者能更方便、快速的在不同的手機上開發手機遊戲，並且支援讓不同的遊戲玩家透過網路對玩遊戲。

- 縮小程式碼的方針和輔助工具
  由於手機的記憶體的限制，程式碼的大小變的相當重要。因此，縮小程式碼變成開發應用程式中所不可或缺的一個環節，工具則是輔助我們縮小程式碼的一個利器。

- 除錯的方針和輔助的工具類別
  由於在手機上除錯非常麻煩，因此，好的除錯方法和工具類別可以縮短開發應用程式的時間，並讓遊戲能盡量減少錯誤的產生。

由於縮小程式碼時，我們所使用的工具，都是半自動化的，也就是在縮小程式碼的過程中，必須手動才能完成，因此，我們希望以後能夠提供一個全自動化的工具，將這些縮小程式碼的方法整合成一個工具。另外，由於除錯的一些方法及問題，我們必須從實作中才能發現，因此，未來我們也希望能夠將遊戲移植到更多款的手機上面，進而發現更多的除錯問題，提供給大家做參考。

參考文獻

[1] OVUM, "Wireless Gaming, Playing to Win", available from http://www.ovum.com, Mobile/Wireless, October 2001.

[2] 3GPP, "Mobile Execution Environment", available from http://www.3gpp.org, 2000.

[3] 吳卓俊、王獻志、吳美惠, WAP 網站設計入門手冊, McGraw-Hill, December 2001.

[4] 吳欣怡, J2ME MIDP 手機遊戲程式設計, 博碩出版社, February 2003.

[5] 微型爪哇人, Java 手機程式開發 J2ME-CLDC/MIDP, 學貫出版社, October 2002.

[6] 廖詩婷, 「無線與 WAP 遊戲發展平台」, 國立交通大學資訊工程學系, 碩士論文, June 2002.

[7] 徐健智, 「網際網路上桌上型遊戲之發展平台」, 國立交通大學資訊工程學系, 碩士論文, June 2000.

[8] 陳凌彬, 「網際網路遊戲發展平台之研究」, 國立交通大學資訊工程學系, 碩士論文, June 2001.

[9] Sun Microsystems, "Java 2 Platform, Micro Edition(J2ME)", available from http://java.sun.com/j2me/index.jsp, 2004.

[10] Sun Microsystems, "Java 2 Platform", available from

http://java.sun.com, 2004.

[11] JCP, "Connected, Limited Device Configuration", available from http://jcp.org/jsr/detail/30.jsp, 2004.

[12] JCP, "Mobile Information Device Profile", available from http://jcp.org/jsr/detail/118.jsp, 2004.

[13] Sun Microsystems, "J2ME Documentation", available from http://java.sun.com/j2me/docs/index.html, 2004.

[14] Sony Ericsson, "Java TM Support in Sony Ericsson Mobile Phones", available from http://www.sonyericsson.com, October 2003.

[15] Motorola, "Motorola SDK for the J2ME TM Platform Developer Edition v4.0", available from http://www.motorola.com, 2003.

[16] JetBrains, "IntelliJ IDEA IDE", available from http://www.jetbrains.com/idea/download/, 2004.

[17] Jack Shirazi, Java Performance Tuning, 2nd Edition, O' Reilly, January 2003.

[18] James Noble, Charles Weir, Duane Bibby, Small Memory Software: Patterns for Systems with Limited Memory, Addison Wesley, September 2000.

[19] Retrologic Systems, "RetroGuard", available from http://www.retrologic.com/retroguard-main.html, 2004.

[20] Eric Lafortune, ”ProGuard”, available from http://proguard.sourceforge.net, 2004.

[21] Danilo Beuche, Wolfgang Schröder-Preikschat, Olaf Spinczyk, Ute Spinczyk, “Streamlining Object-Oriented Software for Deeply Embedded Applications”, IEEE, Technology of Object-Oriented Languages and Systems (TOOLS 33), June 2000.

[22] Mario Friedrich, Holger Papajewski, Wolfgang Schröder-Preikschat, Olaf Spinczyk, Ute Spinczyk, “Efficient Object-Oriented Software with Design Patterns”, Generative and Component-Based Software Engineering (GCSE), Generative Approaches (79-90), 1999.

[23] Ivan R. Moore, “A Tool for Automatic Restructuring of Self Inheritance Hierarchies”, In TOOLS USA 1995 (TOOLS 17, pages 267-275) Prentice-Hall, 1995.

[24] Martin Fowler, Kent Beck, John Brant, William Opdyke, don Roberts, Refactoring: Improving the Design of Existing Code, Addison Wesley, June 1999.