# 國立交通大學

## 多媒體工程研究所

## 碩 士 論 文

跨平台手機遊戲開發框架

在 Symbian 手機之研究

The Study and Development of Symbian for

Cross-Platform Mobile Game Framework

研 究 生：高偉傑

指導教授：林正中 教授

**中 華 民 國 九 十 七 年 七 月**

# 跨平台手機遊戲開發框架
# 在 Symbian 手機之研究

研究生：高偉傑　　指導教授：林正中

國立交通大學 資訊工程學系

摘要

本論文發展一套跨平台的手機遊戲發展框架，在 C\C++ 系統上讓使用者只需開發維護一份程式，卻也能像 Java 平台一樣讓應用程式執行於不同平台。此外還提供以開發遊戲爲目的的高階開發環境，尤其是在網路方面，統合各種通訊協定於同一界面。

# The Study and Development of Symbian for Cross-Platform Mobile Game Framework

Student: Wejai Kao Advisor：Cheng-Chung Lin

Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract

In this thesis, we develop a cross-platform mobile game framework. The game developer just need to maintain one source code on C/C++, but the developed application can runs on different platform like Java. Beside, we provide a high level develop enviroument for game. Especially, we have unified different protocol for network connecting.

# 誌謝

首先誠摯的感謝指導教授吳毅成博士及林正中博士，兩位老師悉心的教導使我得以一窺遊戲領域的深奧，不時的討論並指點我正確的方向，使我在這些年中獲益匪淺。老師對學問的嚴謹更是我輩學習的典範。

感謝汪益賢學長不厭其煩的指出我研究中的缺失、指點研究方向，且總能在我迷惘時爲我解惑。感謝許珉嘉學長的參與，在你的協助與不斷相互討論、共同研究下，才能激發出此篇論文。也感謝吳秉儒同學的幫忙，恭喜我們順利走過這兩年。實驗室的李柏甫學弟當然也不能忘記，你開發的工具真的幫上不少忙。

最後，謹以此文獻給我摯愛的雙親。

# 目錄

摘要.....i

誌謝.....iii

目錄.....iv

表目錄.....v

圖目錄.....vi

第一章、 緒論.....1

第二章、背景說明.....3

第三章、框架的設計.....17

第四章、實做及相關問題探討.....38

第五章、結論與未來發展.....42

附錄.....45

參考文獻.....49

# 表目錄

表3-1、類別列表-Application.....34

表3-2、類別列表-State ... .....35

表3-3、類別列表-Rule .....35

表3-4、類別列表-GUI .....36

表3-5、類別列表-Network.....36

# 圖目錄


圖2-1、 Symbian官方軟體架構(Nokia).....6
圖2-2、Symbian程式觀點軟體架構.....6
圖2-3、Symbian軟體架構~View.....7
圖2-4、Symbian初始化流程.....8
圖2-5、ECDS概念示意圖.....13
圖2-6、ECDS程式概觀.....13
圖3-1、系統架構圖.....17
圖3-2、事件驅動程式設計模式.....18
圖3-3、WinCE事件驅動程式設計模式.....18
圖3-4、Symbian事件驅動程式設計模式.....19
圖3-5、CCPK事件驅動程式設計模式.....20
圖3-6、模組關聯圖.....21
圖3-7、GUI元件與event示意圖.....22
圖3-8、資源轉換工具.....24
圖3-9、Network模組.....24
圖3-10、一般遊戲流程.....26
圖3-11、遊戲流程-Stage 0.....26
圖3-12、遊戲流程-Stage 1.....27
圖3-13、遊戲流程-Stage 2.....28
圖3-14、遊戲流程-Stage 3.....28
圖3-15、Class Diagram.....29
圖3-16、Network模組.....31
圖3-17、AI模組 in Network.....32
圖3-18、網路事件驅動流程.....33
圖3-19、AI模組模擬網路示意圖.....33
圖4-1、GUI Button特例.....39
圖4-2、Connect 6 at Symbian.....40
圖4-3、Big2 at Symbian.....41

# 第一章、緒論

本章會說明行動裝置遊戲的發展現況，並點出跨平台開發時會遇到的問題。最後，會說明本論文的目標與論文大綱。

## 1.1 行動裝置遊戲的發展現況

目前多媒體娛樂已經越來越成爲主流，在行動裝置普及的現代，手機附加功能〔包含下載遊戲或其他加值服務〕自然佔有一席之地。根據 2007 台灣數位內容產業年鑒[5]，美日各國的手機遊戲市場近幾年穩定上升。而在台灣，行動電話附加服務只佔總營運 10%，還有待提升。

市面上的手機產品在下載遊戲這方面的服務，大都是支援 Java 遊戲。雖然目前以 Java 遊戲下載爲大宗，但在各種不同的行動裝置下，仍存在不同作業系統原生的遊戲軟體，例如採用 WinCE 的 Smart Phone 裝置。

## 1.2 跨平台開發

手機下載遊戲已成流行。不僅僅是手機，PDA 等可攜式平台也十

分普及。但是當想要開發行動裝置的遊戲時，必須學習不同平台的開發方式，且彼此不共通。此外當平台上有新的技術時，開發者可能已經離開，而維護者必須重新研究前人所留下的程式。所以我們要解決以上問題。

## 1.3 研究目標

本論文最主要的目的是要跨越平台間的鴻溝，同一份標準適用於不同項目，這些目標又可細分爲以下幾點。

- 開發出能對應多種平台的開發框架：

  我們將框架名稱訂爲 CYC Cross-Platform Kit，簡稱 CCPK。目前對應的有 Symbian、WinCE[6]。

- 高階遊戲開發流程：

  以開發遊戲爲目的，簡化並完整提供開發一款桌面遊戲所需要的 API，並建立詳細說明文件與範例，讓最初階使用者也可輕鬆使用。

- 共通網路界面：

  針對網路連線，提供最簡單的模組，讓使用者可直接使用，而不用去管後端如何實做，並且保留平台擴展空間，設計單一簡單化的規格，讓後續平台開發者能輕易上手。

- 建構遊戲範例，提供各種樣板：

  像是棋類模型、撲克牌模型、麻將模型等。

- 三層次使用者概念：

  一般使用者：運用系統提供的 API 撰寫遊戲。其主要工作在於設計遊戲邏輯與畫面表現。

  新增連線功能：使用者繼承系統所提供的 Network 類別，自行設計新的連線方式。

  平台開發者：使用者必須於新平台上實做所有 API 提供的函式。

## 1.4 論文大綱

本論文主要是發展一套跨平台的手機遊戲發展平台，共有五大章節。第一章說明行動裝置遊戲的發展現況，並點出跨平台開發時會遇到的問題。第二章先介紹目前常見的行動裝置作業系統，並做分析比較。再來會簡介過去類似的研究，最後會以遊戲開發的角度，提出設計目標。第三章是本論文的重點，將會詳細介紹框架的設計。第四章會述說系統的開發原則與實做環境，並展示範例遊戲。第五章說明結論與未來發展。

# 第二章、背景說明

本章會先介紹目前常見的行動裝置作業系統，並做分析比較。再來會簡介過去類似的研究，最後會以遊戲開發的角度，提出我們的設計。

## 2.1 WinCE 平台介紹

WinCE 是微軟作業系統中最小的一個版本，但麻雀雖小、五臟俱全。輕量級的設計，仍舊支援像是 32 位元定址、應用程式間的記憶體保護、虛擬記憶體動態分配、局部和單獨的堆疊空間等。此外他也是完全的多工多執行緒作業系統。

WinCE 最大的優勢在於和 Windows 一脈相承，雖然核心與 Windows (ME, XP)不同， 但是提供與 Win32 API 高相容性的 API。而且支援多種處理器，如：X86、ARM、SHx、MIPS 等。還支援 Windows 常用的技術，如：ActiveX、COM、MFC、ATL、STL。

不過有些優點也造成問題，這些技術隨版本更動而快速變化，使得維護相對困難。

更詳細的內容請參照同實驗室的學長許珉嘉所撰寫的[6]這篇論文。

## 2.2 Symbian 平台介紹

Symbian OS 是專門爲手機裝置而設計的操作系統，包含相關函式庫、UI 架構、共用工具等，由 Symbian Ltd. 開發維護。以 Symbian 作業系統爲基礎的手機用戶介面有 UIQ、諾基亞的 S60、S80、S90、NTT DoCoMo 的 FOMA 等。

Symbian 是以 EPOC 爲基礎，架構與許多桌上型作業系統相似。支援先佔式多工 (pre-emptive multitasking)、多執行緒 (thread)、記憶體保護 (memory protection)等技術。

因爲是專門爲可攜式裝置所設計，在一些情況下會有更高的效能或更佳的表現，譬如可以在有限的資源下，長時間執行。此外 Symbian 還設計了許多機制，像是節省記憶體之機制(ex:堆疊清除管理)、延長電池使用時間之機制。Symbian 本身架構使用 event-driven。這些技術讓 Symbian 變得非常專業，因此學習曲線較陡，初學者不易上手。

### 2.2.1 Symbian 架構

下圖是官方(Nokia)的架構圖[14]，我們以更清楚的方式來解釋。

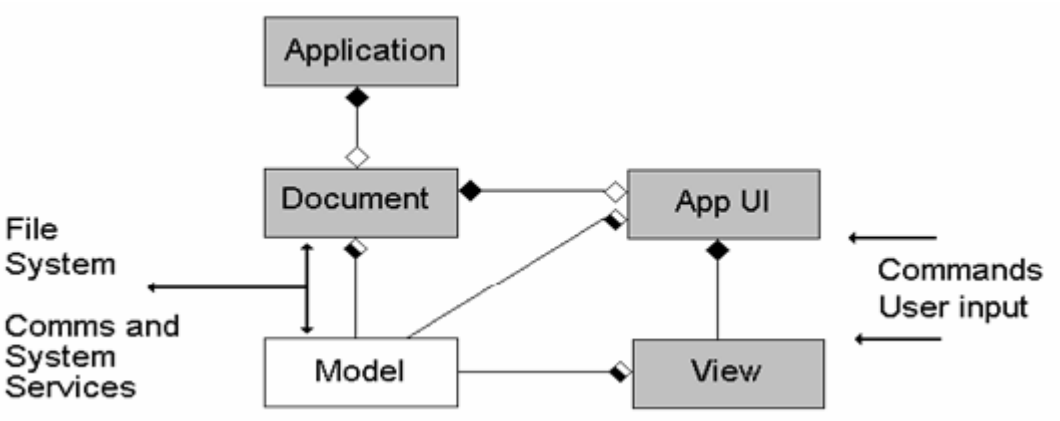



圖 2-1、Symbian 官方軟體架構(Reference:Nokia)

下圖 2-2 展示開發者角度的 Symbian 整體架構：

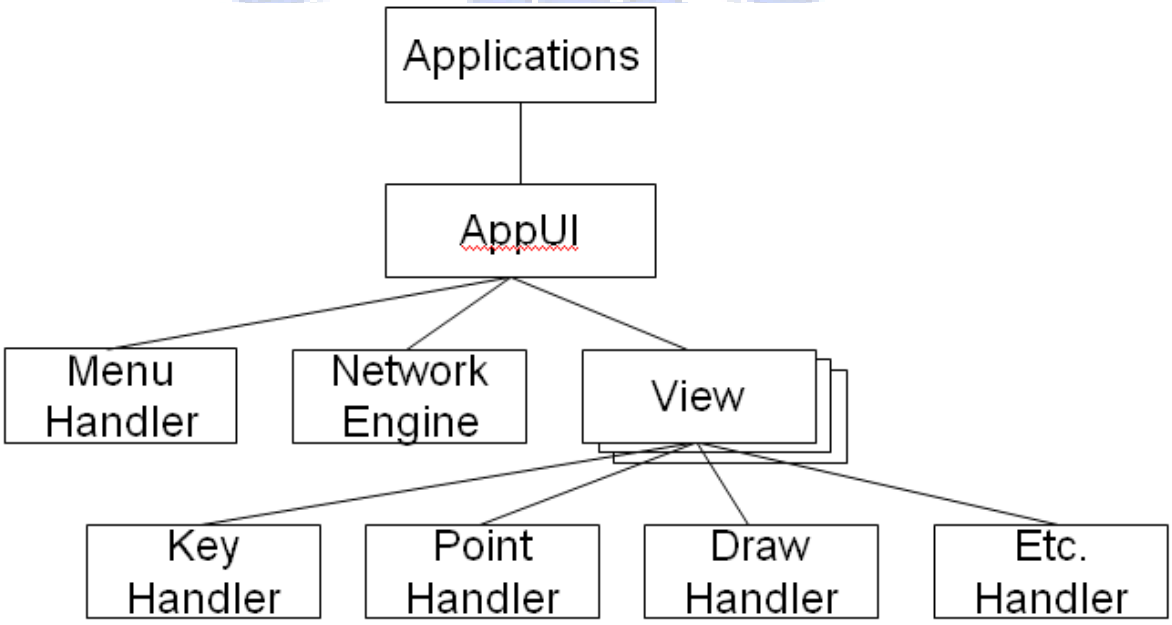



圖 2-2、Symbian 程式觀點軟體架構

- Application:

  程式的整體架構，包含 Application、Document(圖 2-1)等，一般開發者並不會更改此部份內容。

- AppUI:

  專門處理 User Interface 的類別，還負責 Menu command 事件的分派。在這個類別裡，包含了 View classes 與網路模組。

- View:

  專門處理畫面表現以及相關的事件分派，在一個 application 中可能包含多個 View。圖 2-2 展示多個 View 的情形。

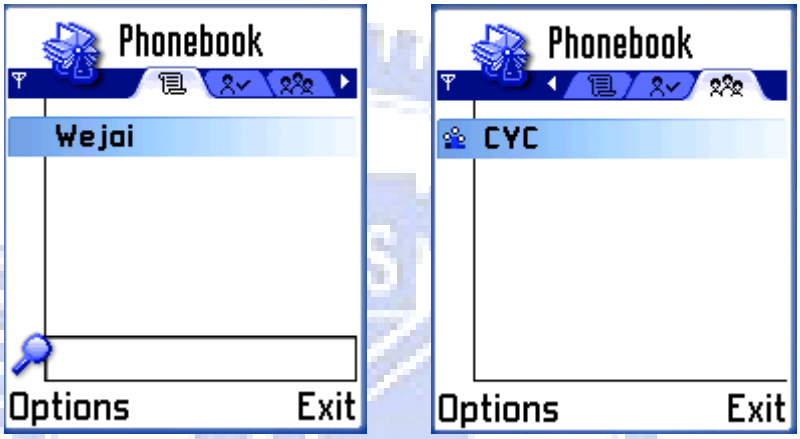



圖 2-3、Symbian 軟體架構~View

- Network Module:

  處理網路連線事宜，舉凡連線、登入界面，一般都是由使用者自行撰寫，其實在 Symbian 架構中並非必要的。

- Handlers:

  因爲 Symbian 架構封裝得比較緊密，一些事件的接收函式已先定義好，要使用的話必須直接繼承覆寫。在 View 下有提供 Key、Point、Draw 等和螢幕畫面有關的事件函式，而 Menu 事件則是由 AppUi 負責。另外 Netwok 模組的事件要用另一種事件的接收方式(Active Object)。

此圖展示 Symbian Start-up 流程[15]：

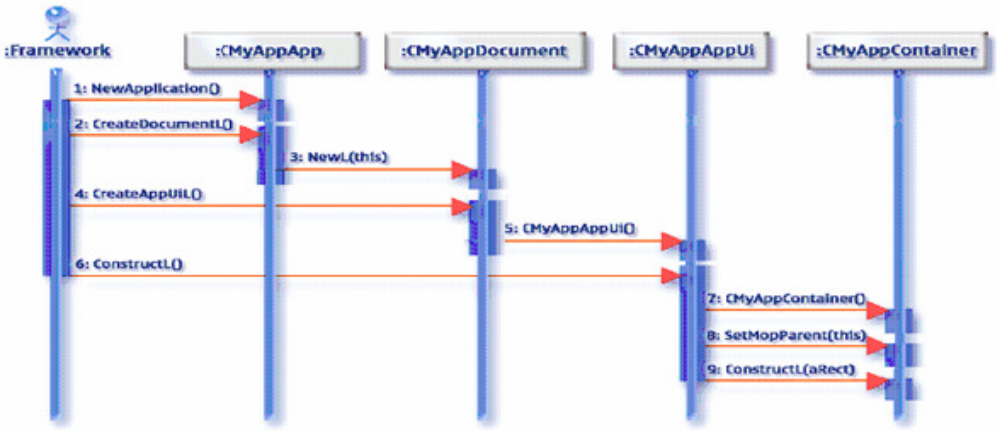



圖 2-4、Symbian 初始化流程(Reference:Nokia)

接下來會一步步介紹每一元件的初始化步驟。

這段程式碼爲系統的真正進入執行點：

```
// Entry Point for applications:
GLDEF_C Tint E32Main()
{
   return EikStart::RunApplication(NewApplication) ;
}
```

這段程式碼爲Application類別的起始點：

```
// NewApplication() contructs MyApp
LOCAL_C CApaApplication* NewApplication()
{
```

```
return    new MyApplication();
}
```

建構出 Application 元件後，由 Application 創造 Document：

```
CApaDocument* MyApp::CreateDocumentL()
{
   return MyDocument::NewL( *this );
}
```

由 Document 生出 App UI：

```
CEikAppUi* MyDocument::CreateAppUiL()
{
   return new (ELeave) MyAppUi;
}
```

最後由 App UI 生出其他的元件，像是 View、Network Engine 等，這裡只展示 View 的情況：

```
void MyUi::ConstructL()
{
   BaseConstructL();
   MyView = MyView::NewL(ClientRect());
   MyView->SetMopParent(this);
    AddToStackL(MyView);
}
```

### 2.2.2 如何使用 Symbian 的事件驅動模式

Symbian 的 event-driven 方式，以繼承、覆寫官方提供的元件爲主。只要按照規則寫好，當事件發生時自然會得到呼叫，而不用管系統底層是如何掌控事件的發生與通知。以下這段程式碼[16]，簡單地述說要如何去接收 Key 這事件。

首先在 AppUI 類別的初始化中，必須先使用 AddToStackL 註冊是由哪個 View 接收 Key 事件，再來覆寫 View 的事件處理函式 OfferKeyEventL。由呼叫 OfferKeyEventL 傳進來的參數，就可以判斷是哪個 Key 被按下。

```
void MyAppUi::ConstructL()
{
    BaseConstructL();
    iView= MyView::NewL();
    AddToStackL(iView);
 }
```

這是MyView的定義，必須覆寫CCoeControl定義出的OfferKeyEventL函式。

```
Class MyView :  public CCoeControl
{
Public:
    TKeyResponse OfferKeyEventL(……);
    void HandlePointerEventL(……);
```

```
    void Draw(……);
    ……
}
```

這是覆寫的內容，可以由傳進來的參數判斷是哪個鍵(EKeyxxArrow)被按下。

```
MyView :: OfferKeyEventL(TKeyEvent& aKeyEvent, TEventCode aType)
{
    switch(aKeyEvent.iCode)
    {
        case EKeyUpArrow:
        case EKeyDownArrow:
        ……
    }
}
```

以上是以 Key 這個事件爲範例，其他事件的做法大同小異，只是對應函式不同。

## 2.3 Palm OS 平台介紹

Palm OS 是一個在行動裝置上使用的 Linux 平台，當年由 US Robotics 研製的專門用於其掌上電腦產品 Palm 的作業系統。過去曾被 IBM、Sony、Handspring 等廠商取得授權，使用在旗下產品中。[20]

Palm OS 以簡單易用爲大前提，運作需求的記憶體與處理器資源較小，速度也很快。相對的，它不支援多工處理，長遠發展受到限制。

目前 Palm OS 版權由 PalmSource 公司擁有，2005 年 9 月 9 日，PalmSource 被日本軟體開發商愛可信收購。[20]

## 2.4 過去研究探討

此節會介紹兩項過去曾有過的類似研究，並闡明其優缺點。

### 2.4.1 ECDS

ECDS，全名 Embedded Cross-Platform Development Solution，嵌入式跨平台開發解決方案，是由位於中國的武漢卓睿公司[1]所研發。此解決方案提供了一套跨平臺智慧手機應用程式開發庫，包含核心庫、聲音庫、UI 庫、網路庫、手機功能擴展庫等。支援的平台有 Symbian S60 ($1^{st}$/$2^{nd}$/$3^{rd}$)、SmartPhone 2003、Pocket PC 2003、Windows 2000/xp。

“一次代碼編寫，多個平台運行”，是此解決方案的中心概念。設計上引用平台/OS 抽象層的概念，隱藏底層系統及硬體平台差別，並提供一致且統一的應用開發介面。能免去了反復代碼編寫和多套代碼管理的不便。此外運用此方案，還可直接在 Windows 作業系統上進行 Debug。

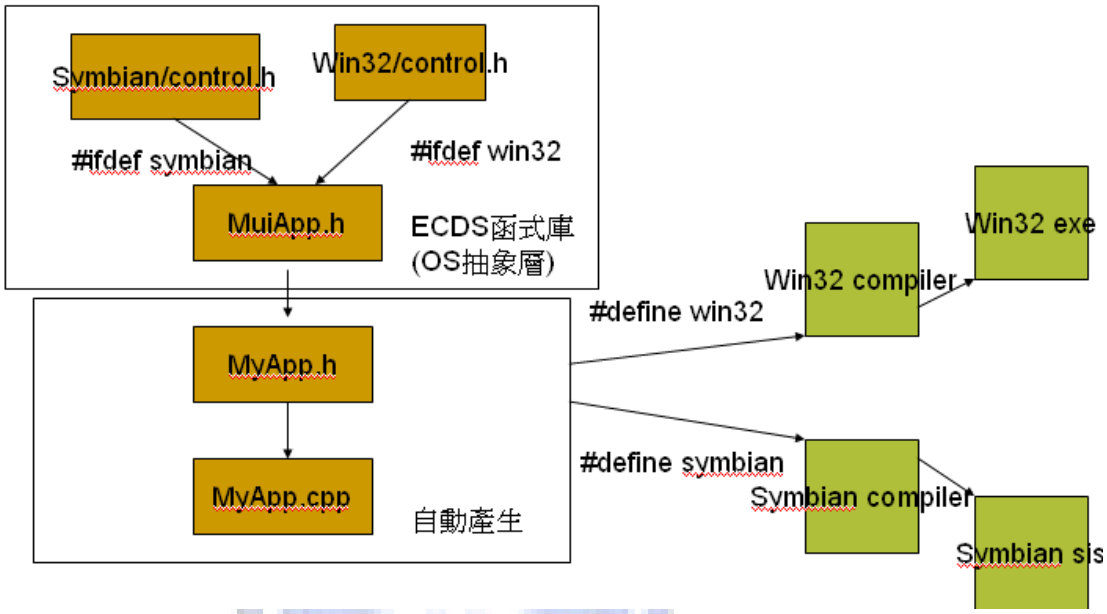



圖 2-5、ECDS 概念示意圖

圖 2-5 展示 ECDS 是如何切割出 OS 抽象層，根據平台不同使用不同的定義檔，在編譯時使用各自的版本。

```
#define INC_MUI_CONTROL
#include <mui/mui.h>
using namespace mui;
#include "../../src/testApp.h"
MuiControl g_MuiControl;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR pCmd, int iCmdShow)
{
    if (!g_MuiControl.Create())
    {
        return -1;
    }

    testApp *app;
    app = new testApp();
    MuiEnv::Static()->RunApp(app);

    delete app;
    return 0;
}
```

負責生出所有環境設定，包含command dispatch、call back function 設定

使用者實做的app

圖 2-6、ECDS 程式概觀

上圖展示 ECDS 以 Visual Studio 開發的概觀，這只是一開始的 main 函式，如果要使用到內部的各種函式庫，還要再去看更多文件。

但是 ECDS 仍有不少缺點，非公開的設計難以讓人了解其中原理。加上其意圖仿效 Java 的架構，造成整體功能機制複雜。除此之外，其設計的網路功能不夠完善，甚至系統本身仍有錯誤。不過最大的問題在於，此研究已結束，未再研發(2008 Jan)。

接下來簡介 ECDS 複雜的使用方法。圖形方面，ECDS 採取仿 Java 的機制，在畫圖之前必須呼叫許多函式。首先必須取得目標顯示幕的表面點陣圖，再得到螢幕大小〔這些都是 ECDS 自己定義的格式〕，然後使用 Graphics 物件，呼叫許多設定函式包含顏色、字型等，最後呼叫系統總管繪圖。 而讀取圖片還有更多步驟，除了要求使用者使用自己定義的圖片格式(cmp)外，在繪圖前還得指定各種參數條件。至於畫字部份，仍舊使用 Graphics 物件執行並且設定一堆規則。

UI 這部份，必須先了解各種不同元件所指定的參數，再去修改指定位置的函式後才能使用。這些動作都必須覆寫原本的類別。ECDS 還採用了 Slot 的概念，每個 Widget 元件與實體表現就像是插座與插頭間的關係。這些關聯必須使用 ECDS 規定的函式來處理，而每種元件的參數與使用時機又不盡相同，要完全熟悉使用需花費一些時間。

聲音方面也是使用 ECDS 特有的定義，在使用上沒什麼問題。但在網路方面就我所測試的結果，DNS 解析部分是正常的。但他們提供的 Socket 類別 InetSocket、BTSocket 根本沒有實作完成，因此我

無法測試網路連線功能。

### 2.4.2 MobileCore

一個 Open Source Library。特色就是簡單，僅支援部份功能。像是一些常用的設計都沒有包含，如：功能表、按鍵、文字方塊。支援的平台有：Pocket PC、Symbian、Win32 等。[9]

雖然此研究是以開放源碼的方式進行，但相關討論並不踴躍，自 2003-10-29 就未再更新。

## 2.5 遊戲開發

本節會以遊戲開發角度探討跨平台框架的設計理念，最後提出我們的研究目標。

### 2.5.1 遊戲開發背景

究竟開發一款網路桌面遊戲需要考慮那些要素，需要實做那些元件，那種系統架構比較有效率，何種遊戲流程概念比較合理。在徐健

智學長[2]這篇論文內已有深入討論，其理論基礎也被真實運用到網路遊戲平台。陳智文學長[4]這篇論文則提出，如果將環境拉到行動式平台，會要考慮那些問題，資源與測試方式有何不同。此外，市面上有沒有類似想法的產品，這些產品的表現與其優缺點。2.4 已有介紹。可以知道的是，目前並還沒有一個針對遊戲而設計的跨平台行動裝置開發機制。

### 2.5.2 開發平台選擇

或許很多人會問，明明大部分手機都支援 Java 遊戲下載，根據 Java 的特性，根本不會有跨平台的問題，那爲何我們選擇 C/C++？根據 Andreas Janecek 和 Helmut Hlavac 的這篇論文[7]，在動畫表現上.NET CF 會比 Java 有更好的效能，尤其是在低階機器上更爲明顯。此外部份行動裝置操作系統，必須另外進行 Java 的安裝。所以以 C/C++爲開發語言，直接對應各平台底層，是有其必要性的。

下一章就會來介紹 CCPK 的設計。

# 第三章、框架的設計

用一個已經開發好的手機遊戲框架來設計遊戲的好處就是，框架已存在一些既定的流程，遊戲只需覆寫一些框架中的函式，就可以照著框架規範好的流程執行。也因爲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。本章將會先針對 Event-Driven 的模式分析，再詳細介紹框架的設計。

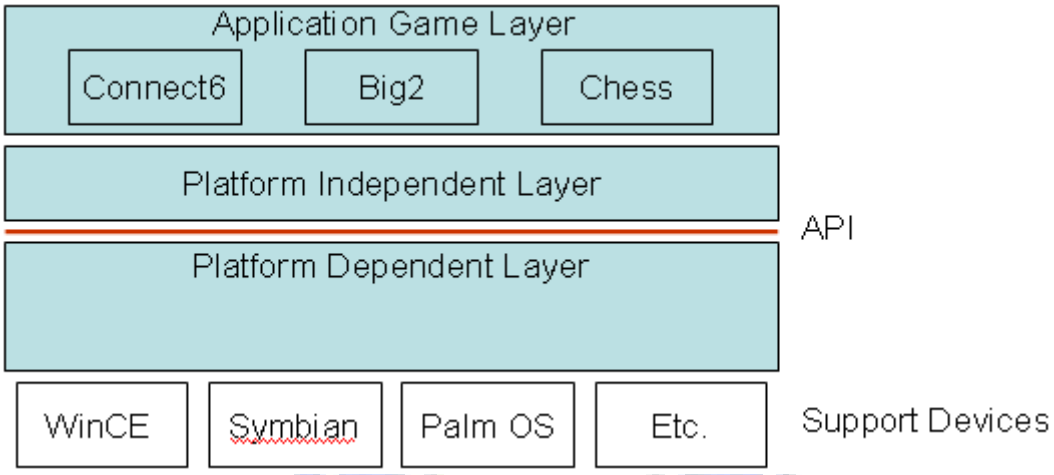



圖 3-1、系統架構圖

## 3.1 事件驅動模式(Event-Driven)

大部分的圖形化介面程式都是使用事件分派來處理訊息傳遞。根據[8]這篇論文指出，使用事件驅動(event-driven)的程式比使用執行緒(thread)的在開發上比較簡單，後續偵錯(debug)也比較容易。

圖 3-2 展示一般的事件驅動模式。在初始化的過程中，必須指定事件所對應的處理原則，之後程式開始運行，事件不斷地發生。一般來講，此時會有一個組列(queue)暫存這些事件的狀態，在一段時間或系統有空時，事件分派器會分析組列內的事件，將它交由相對應的函式或機制來處理。

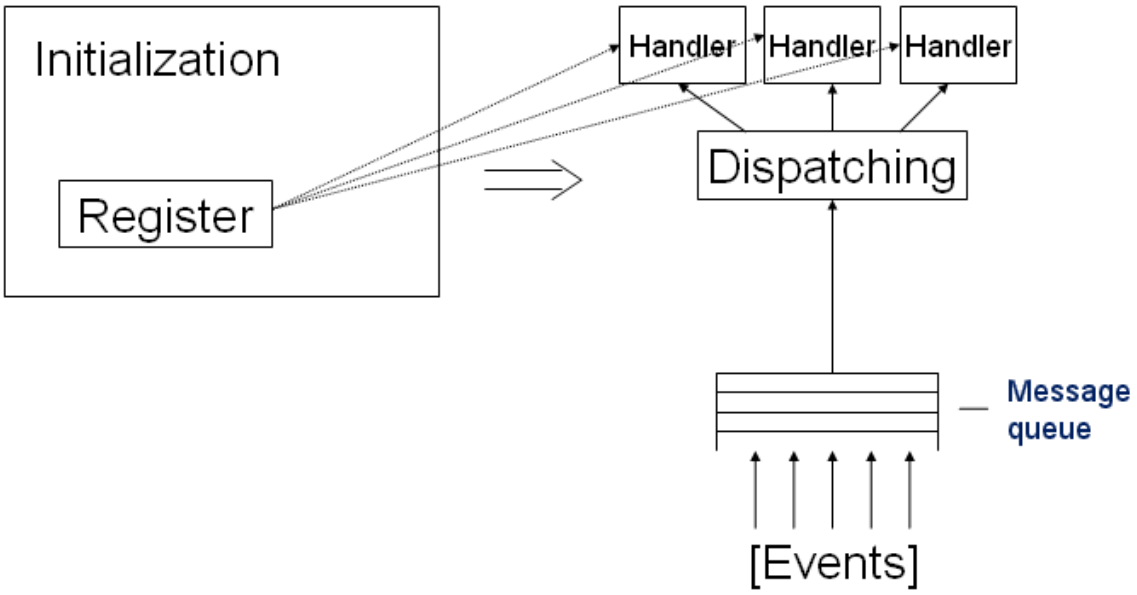



圖 3-2、事件驅動程式設計模式

下圖展示在 WinCE 中事件驅動的模式。

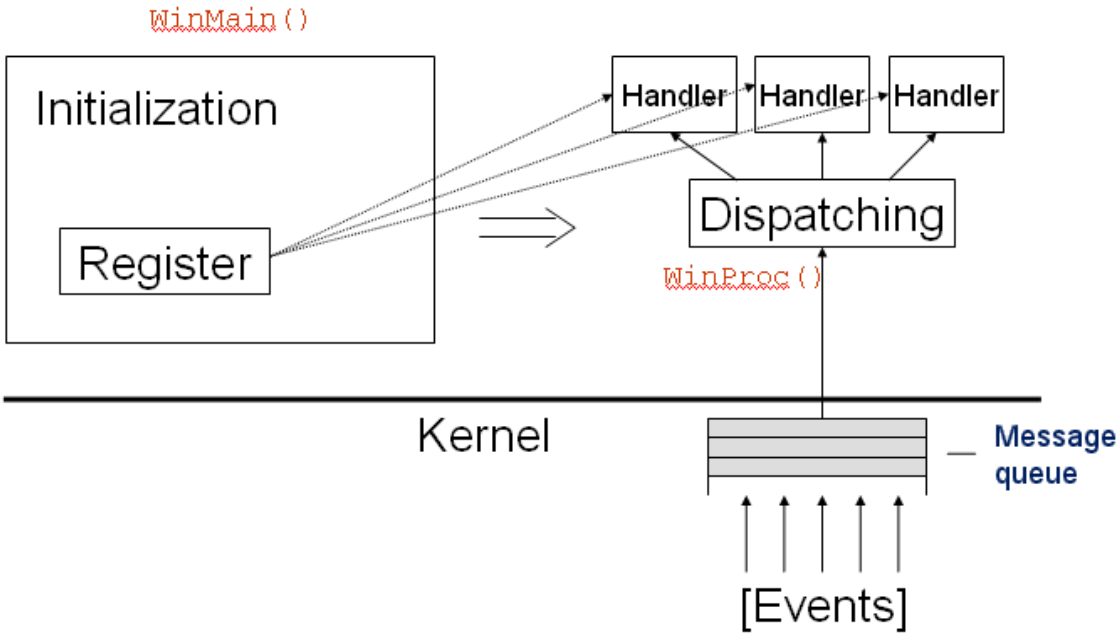



圖 3-3、WinCE 事件驅動程式設計模式

一般 WinCE 的程式進入點爲 WinMain()，在系統核心(Kernel)端會有個組列儲存事件，然後使用者在 Winproc()內將事件拿出，分配給相對應的函式。

圖 3-3 是 Symbian 的情況。

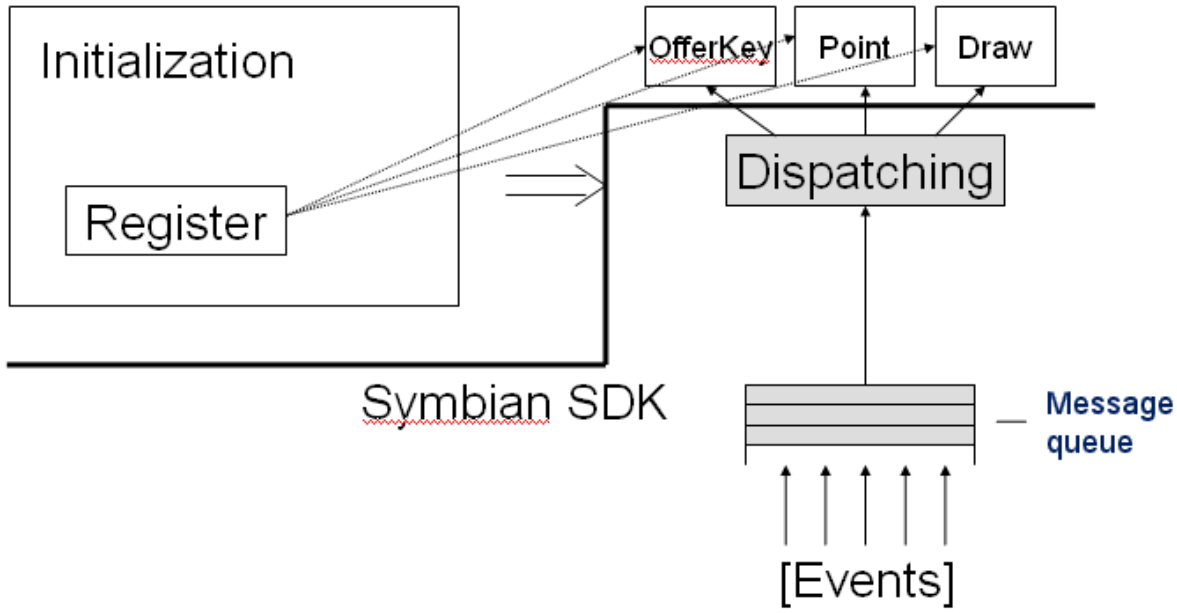



圖 3-4、Symbian 事件驅動程式設計模式

與 WinCE 不同，Symbian 結構包裝的比較完整，連事件分派都在系統底層完成。使用者需覆寫已經定義好的函式，當事件來臨時會直接轉呼叫這些函式。

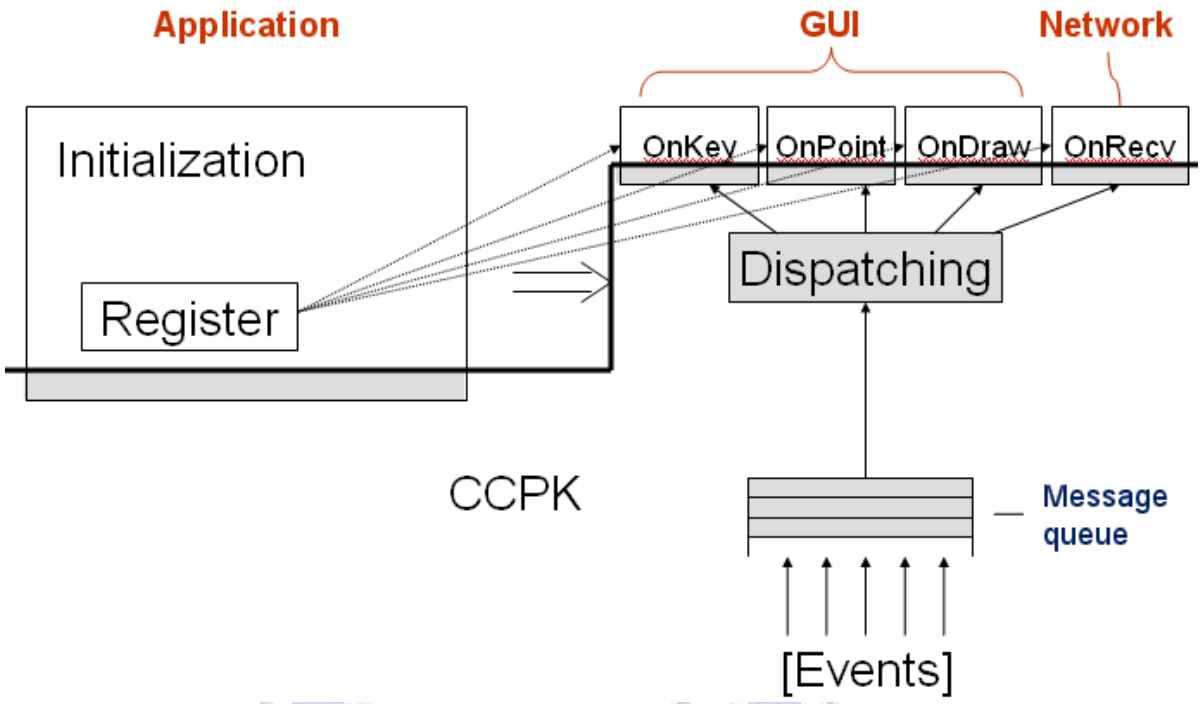



圖 3-5、CCPK 事件驅動程式設計模式

此圖爲我們 CCPK 的示意圖。爲了對應包裝比較完整的 Symbian，事件驅動模式會和 Symbian 比較類似。同樣地提供一些基礎函式供使用者覆寫。在這張圖展示出的三個模組：Application、GUI、Network，會在下一節作詳細介紹。

## 3.2 CCPK 跨平台設計(Design of CCPK)

CCPK 有三大模組：Application 模組(包含類別：CCPKApp and CYCEnv)、GUI 模組(包含類別：CCPKFrameWnd, CYCLabel, CYCButton, CYCGraphicContext and CYCImage)、Network 模組(包含類別：CCPKNetwork)。此節更詳細的設計理念請參照許珉嘉學長[6]

這篇論文。

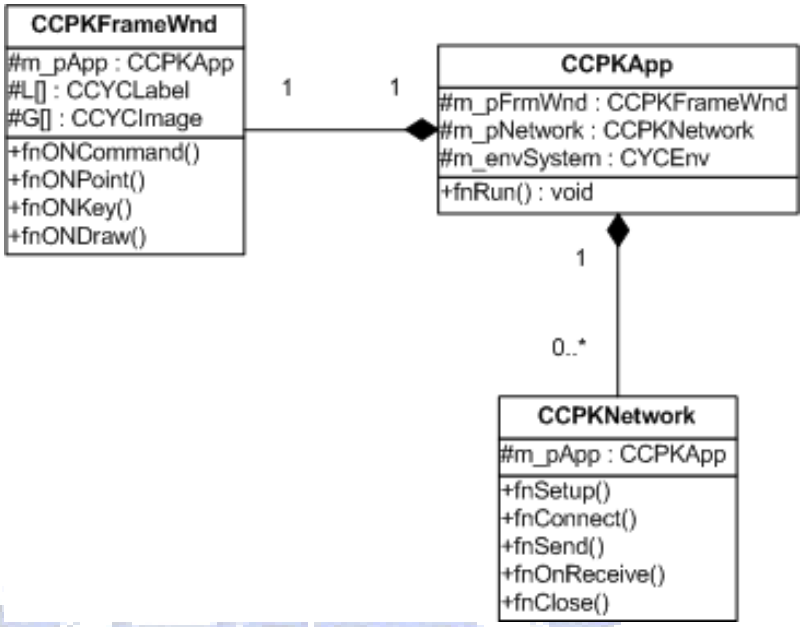



圖 3-6、模組關聯圖

- Application 模組：

  Application 模組是程式的中樞，負責初始化外還要設定其他兩個模組，且處理模組間的溝通。從另一個角度來看，Application 是整體應用程式的頭。除此之外 Application 還要擷取系統資訊(由 CYCEnv 負責)像是記憶體使用狀態、螢幕大小等，有時一些與系統底層的設定也要在此完成。CCPKApp 定義了一個重要函式 fnRun，需要使用者覆寫。這個函式就是系統的進入點，大部分的初始化都是在這個時候執行。

- GUI 模組：

  首先先展示目前有的元件與支援的事件。元件有：Frame Window、Label、Button、Menu、Image 等。事件有：Key、Point、Button/Menu events (called Command Events in CCPK)、

Draw 等。

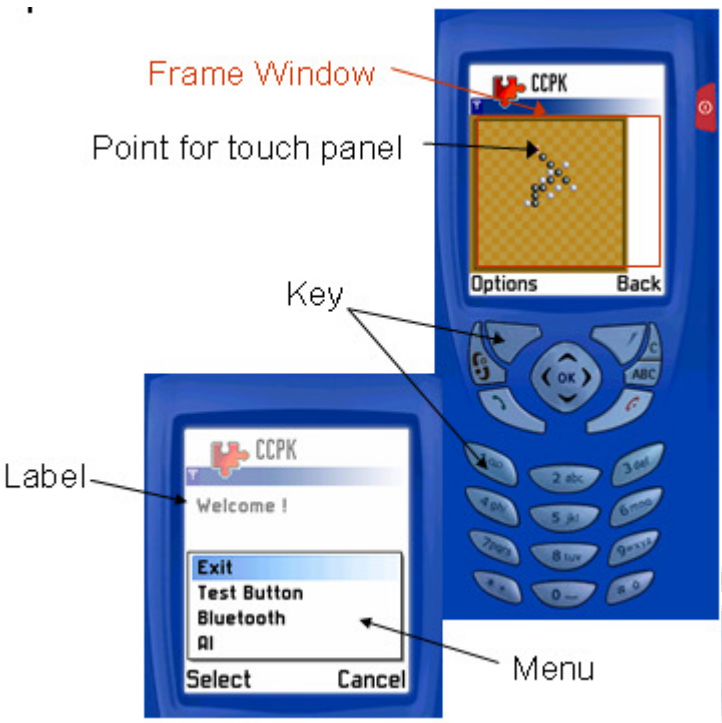



圖 3-7、GUI 元件與　event 示意圖

GUI 模組包含很多東西，粗略分有 Widget 和 Graphic 兩類，以下先介紹 Widget 類。

- Widget- CCPKFrameWnd:

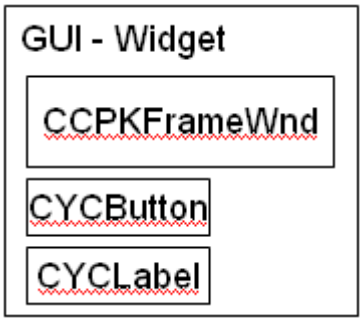
  主畫面表現，代表螢幕所見區域，對應各種事件(Draw, Key, Command, Point)，許多事件函式在這裡要被覆寫。

- Widget-CYCButton:

  按鈕物件，提供使用者點擊。

- Widget-Label:

  文字表現區域。



以下介紹 Graphic 類：

- Graphic- CYCGraphicContext:

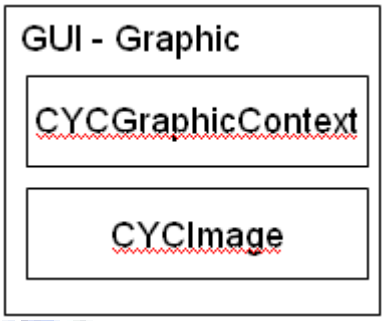
  畫布，管理螢幕顯示的物件，具有基本的繪圖能力(點、線、面)，可將影像區塊展現於其上。

- Graphic- CYCImage:

  影像格式，讀取影像檔，

  利用畫布將影像表現出來。

  目前 WinCE 支援: BMP/PNG/JPG、Symbian 支援: BMP。




在論文一開始有提過，不同平台都有各自的資源定義，而且這些格式是無法在程式內部避開的。因此我們發展了一個工具，能夠使用一個共有的格式，將兩平台的資源檔生出。尤其是在 Button/Menu 的客制化方面，必須先使用這個工具才能在程式中找到 Button/Menu 的定義。附帶一提，WinCE 的資源檔爲 Resource.h、Data.rc，Symbian 的資源檔爲 Resource.hrh、Data.loc。


圖 3-8、資源轉換工具

- Network 模組：

  Network 模組封裝底層 Network 複雜的實作，提供簡便的函式，利用事件驅動方式通知 Application 已收到網路資料。目前我們的實做支援 TCP/IP 以及 Bluetooth。

  以下簡介網路介面的各個函式定義：

  - 網路連接設定：

    設定網路連線，舉凡創造網路內部元件、呼叫預設函式等，使用函式名：fnSetup()。

  - 執行網路連接：

    真正執行連線及其之後的動作，使用函式名：fnConnect()。

  - 資料傳遞-送：

    將資料送到網路端，使用函式名：fnSend(char *pBuf, int

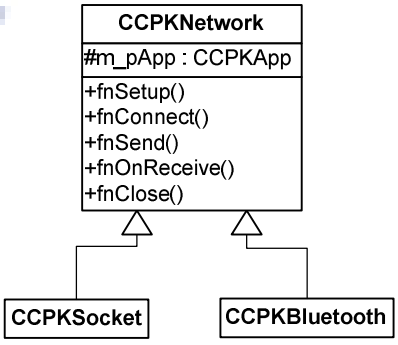



圖 3-9、Network 模組

len)。

- 資料傳遞-收：

  以事件驅動方式接收網路訊息，使用函式名：fnOnReceive(char *pBuf, int nLen)。

- 結束網路連線：

  結束連線時呼叫，斷開離線，使用函式名：fnClose()。

## 3.2 獨立遊戲層設計(Design of Independent Layer)

此節會詳細介紹在 Independent 層的設計。此層主要以遊戲開發爲目的，提供高階的 API 讓遊戲開發者使用與覆寫。首先會述說一套一般遊戲的進行流程，並會講解每一階段所要完成的工作。再來會說明在 Dependent 層的基礎上，如何擴充原有的基礎函式與功能，達到簡易開發的目的。最後會列出目前所有已設計出的類別與相對應的函式。

### 3.2.1 一般遊戲流程

我們將遊戲流程分爲四個階段：

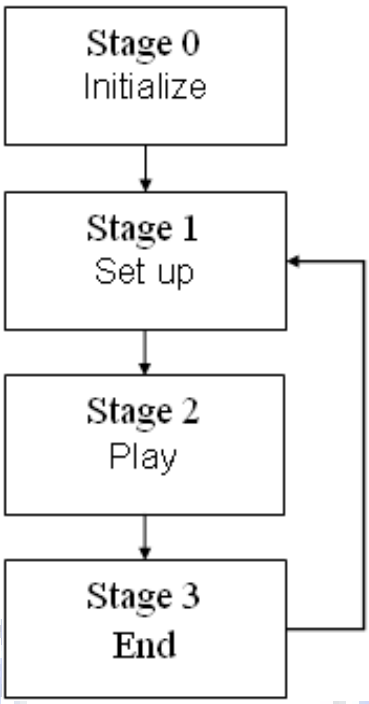



圖 3-10、一般遊戲流程

- 階段 0 初始化：

  當程式開始執行時，什麼物件都沒有。第一件事就是要辨明現在使用的平台，根據平台的不同接下來產生的物件也有所差異。在此階段產生的元件主要有 Application、GUI 等。

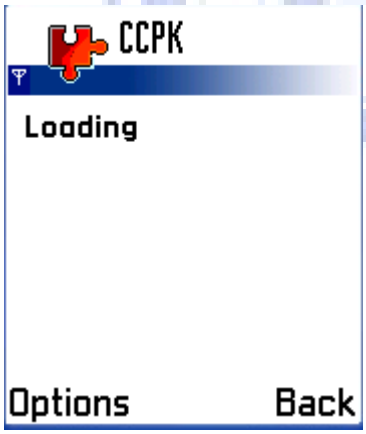



圖 3-11、遊戲流程-Stage 0

- 階段 1 設定：

  這時已真正進入遊戲本體。螢幕上要展示歡迎畫面，程式內部也要去檢索系統資源情況，還要提供使用者介面(User Interface)

給使用者選擇設定，像是：輸入用戶名或是選擇連線方式。在使用者選擇連線方式後，就要進入連線登入畫面。等連線動作完成，就要開始進行遊戲了。

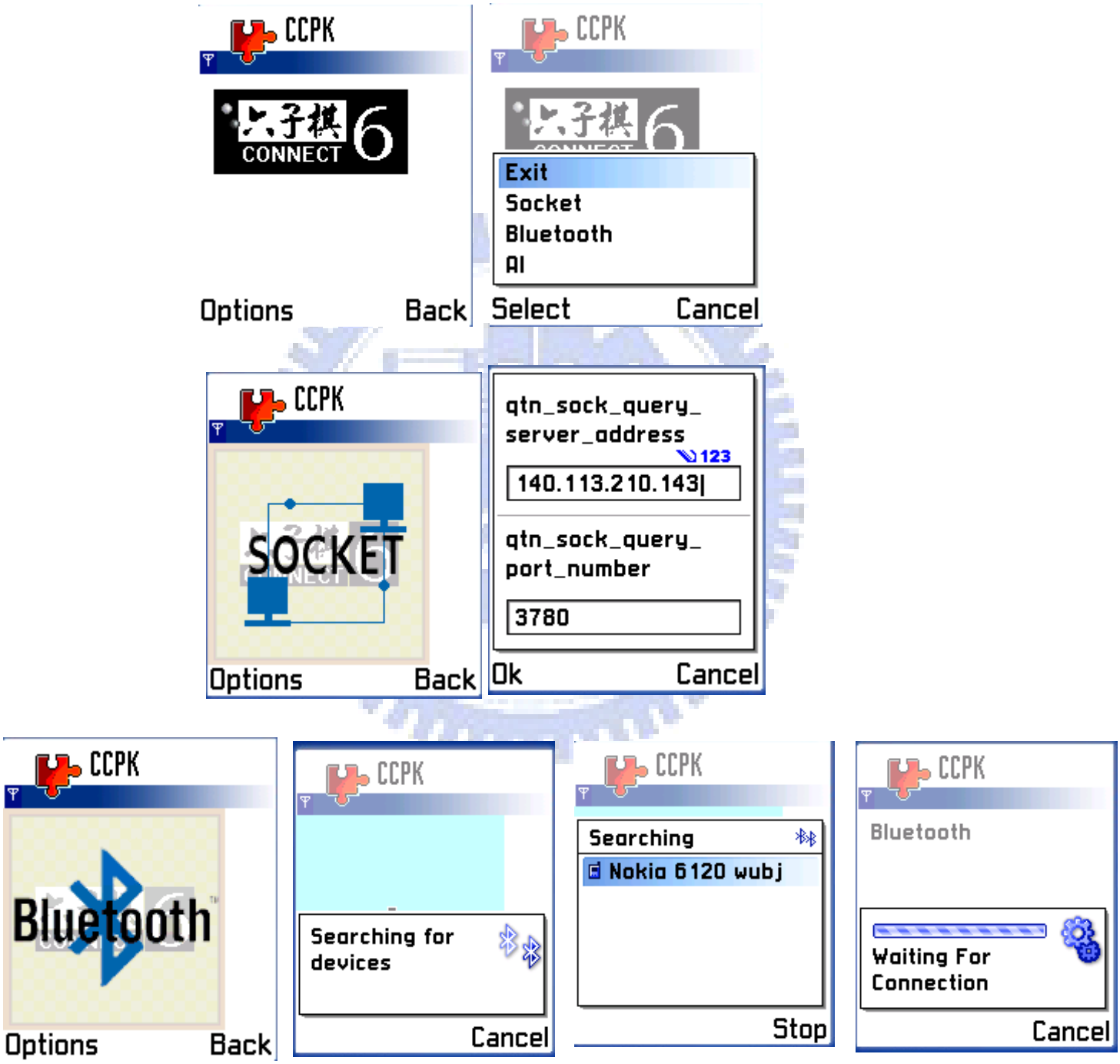



圖 3-12、遊戲流程-Stage 1

- 階段 2 遊玩：

  在這裡又可細分爲主動與被動兩種階段。主動階段表示輪到使用者的回合，此時使用者可以輸入選擇，像是下棋、選卡等。被動階段則代表輪到網路端對手的回合，此時大部分的 UI 會被封鎖，等待對方完成輸入。


圖 3-13、遊戲流程-Stage 2

- 階段 3 結束：

  在不斷往返的 Stage2 後，遊戲有了勝負結果，來到了結束階段。此時會先顯示結束畫面，再來要提供結束時的選項，看是要再來一盤(Stage2)、回到歡迎畫面(Stage1)、還是結束程式。

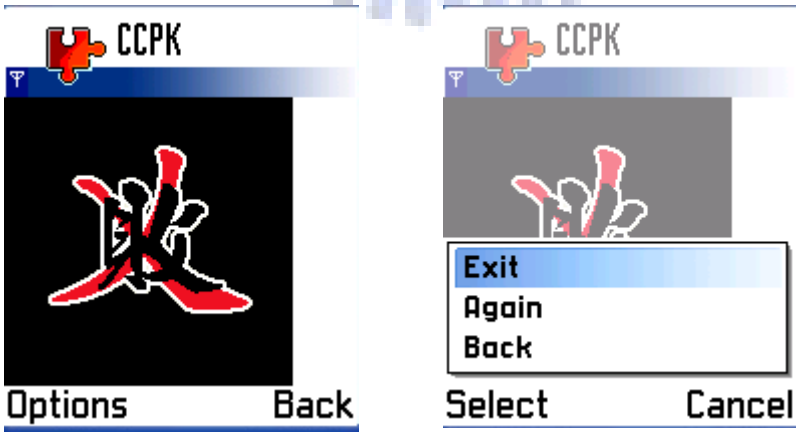



圖 3-14、遊戲流程-Stage 3

### 3.2.2 遊戲元件

在 3.1 我們曾介紹過三大元件：Application、GUI、Network。當然以遊戲開發來看，這樣的架構是不足的。所以我們在三大元件上新增了不少功能，並隱藏過於瑣碎複雜的部份，僅保留和遊戲設計有關的地方。另外也在 Application 模組中增加了 State 和 Rule 兩個類別，在 Network 模組中增加了 AI 類別，藉以輔助遊戲的運行。

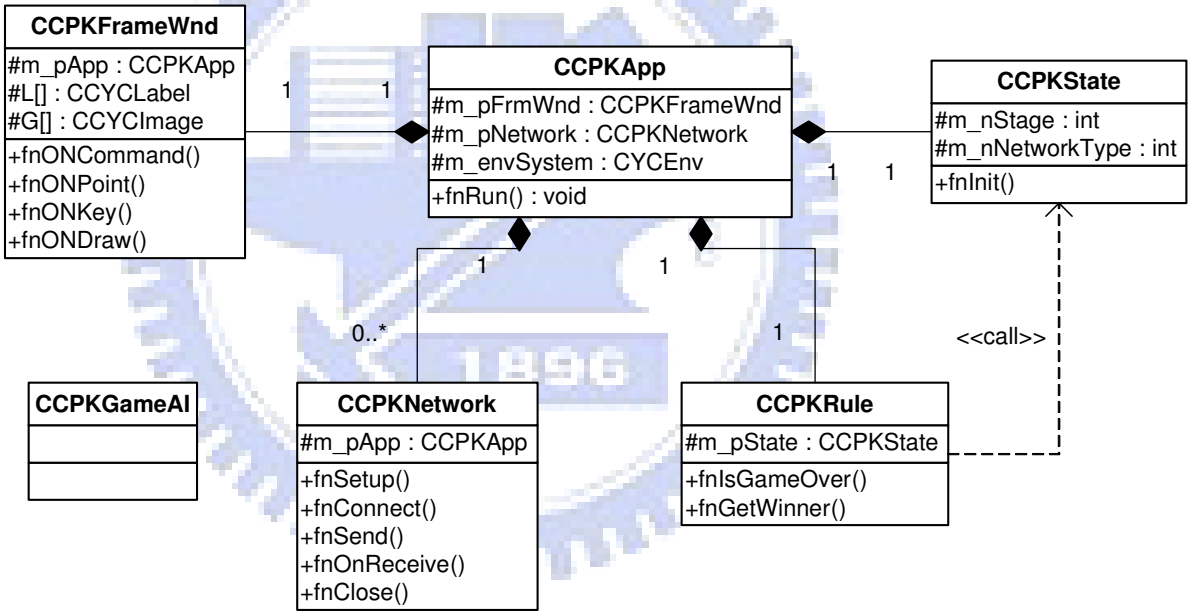



圖 3-15、Class Diagram

以下就來介紹每個類別：

- Application 類別：

  在 Impendent 層中 GameApp 類別繼承了原本的 CCPKApp，填補了原本單純的功能，並且隱藏了原本與其他模組間複雜的交流模式。這裡規劃了兩個重要函式供使用者覆寫。第一個函

式 Core Method，在棋類樣板(template)中稱之爲 MakeMove()，牌類樣板中稱之爲 SelectCard()。它就像整個遊戲的心臟，玩家所有和遊戲進行有關的動作像是下子、選牌，最終都會呼叫到這個函式。在這個函式裡玩家需要藉助 State 和 Rule 的協助，撰寫遊戲中最重要的邏輯部份。第二個函式 RecvMessage()，這個函式負責接收網路端的訊息，做出相對應的反應。這件事原本是由 Network 模組中的 fnOnRecv()負責，但我們認爲遊戲資料的處理應該要集中到一處，因此把它移到 Application。

- State 類別：

  State 類別是把遊戲中的資料、玩家的設定和從模組 CYCEnv 得到的系統資訊集中起來，可以分成幾類：私人數據〔用戶名、密碼、積分〕、系統數據〔使用平台、螢幕大小、記憶體容量、網路類型〕、遊戲數據〔回合、得分、棋盤情況〕。這些資料大都由 Application 和 Rule 更新、檢索。

- Rule 類別：

  這個類別定義出和遊戲規則有關的函式，像是 CheckMove〔確認此步合不合法〕、IsGameOver〔確認遊戲結束沒〕、GetWinner〔回傳勝利者〕。從另一角度看，遊戲主要邏輯就是在這些函式中完成。這些函式皆由 Application 呼叫。

- GUI 類別：

  在 Independent 層中 GameFrameWnd 類別繼承了原本的

CCPKFrameWnd，新的類別中預先幫使用者寫好各種 UI 的反應，像是按鍵、螢幕碰觸等，大部分情況下使用者只要直接用就好。另外還針對每一個階段定義出不同的 Show 函式，用來表示不同階段的畫面表現。比較需要提的是功能表事件的處理，在使用我們提供的資源生成工具後可以得到像是 CmdSocket、CmdAI 等資源代號。這時只要判斷事件處理的函式傳進來的參數符不符合，就可確定是哪個功能表事件，如下列程式碼所示。

```
virtual bool GUIC6::fnOnCommand(int cmd)
{
  switch( cmd )
  {
  case CmdSocket:…;break;
  case CmdBluetooth:…;break;
  case CmdAI:…;break;
  case CmdGO:break;
  }
}
```

- Network 類別：

  目前我們預設支援的有 TCP/IP 和 Bluetooth 在高階開發環境中，網路模組已完全包裝，使用上只要在 Stage 1 選擇使用的連線類型。

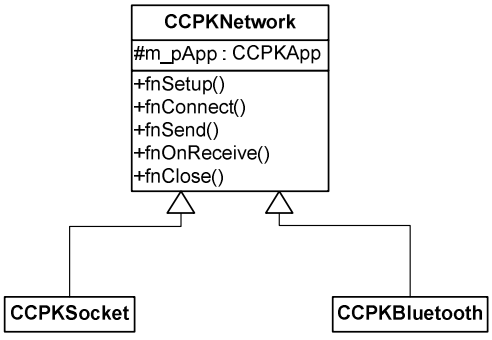



圖 3-16、Network 模組

- AI 類別：

  AI 或許在一般應用程式上不是必須的，但一款手機遊戲 AI 是不可或缺的。雖然手機都會有連線功能，但難保正好沒人上線，AI 的存在可以避免這種情形。那麼 AI 究竟是什麼呢？簡單來講就是根據輸入的遊戲狀況做出選擇，像是下子、出牌。不過一個好的、聰明的 AI 其所需的演算法勢必更複雜，在資源有限的手機裝置下，其運算時間勢必加長。所以我們想出一個方法，將 AI 視做爲網路介面中的一種連線，就像 Socket 和 Bluetooth 一樣。同樣地 AI 也要對應網路模組所規定的格式。如下列：

  - 設定 AI 程式(對應 fnSetup)：

    一個強大的 AI 在真正運算前一定會有許多東西要載入，像是統計表或常數設定等，在這裡使用 fnSetup 來對應。

  - 開始 AI 程式(對應 fnConnect)：

    通知 AI 程式要開始對弈了。

  - 接收玩家輸入(對應 fnSend)：

    接收輸入。

  - 回傳 AI 決定(對應 fnOnRecv)：

    經過計算後回傳 AI 結果。

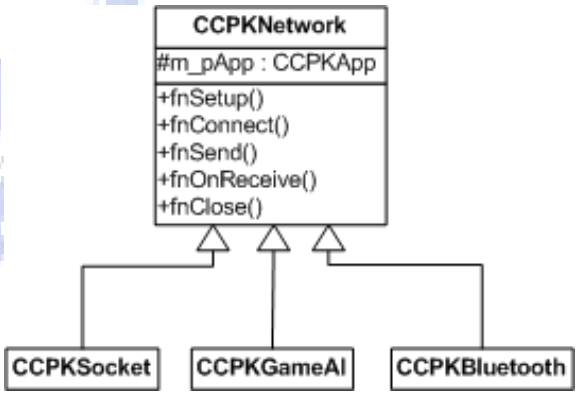



圖 3-17、AI 模組 in Network

■ 停止 AI 程式(對應 fnClose)：

此時必須釋放系統資源，以

免影響之後使用。

可是將 AI 轉換成網路格式還有一個問題就是：Event-driven 模式，下圖爲一般的網路流程，會有一個觀察者(observer)以事件驅動方式處理 fnOnRecv 事件。

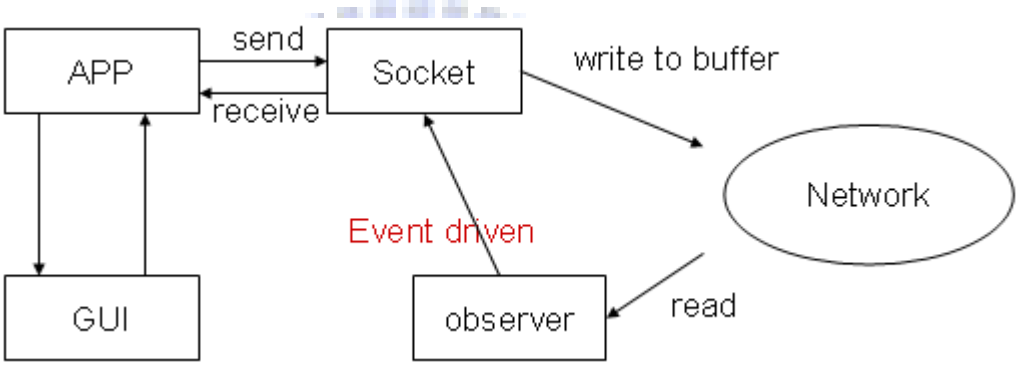



圖 3-18、網路事件驅動流程

而在 AI 之下並沒有觀察者的存在，在系統呼叫 send 之後，直接交由 AI 程式運算，等到運算完成再直接呼叫 fnOnRecv。這樣的做法會使 AI 運算造成 block，如下圖。幸好目前所開發的 AI 在實機測試時並沒有顯著的 delay。

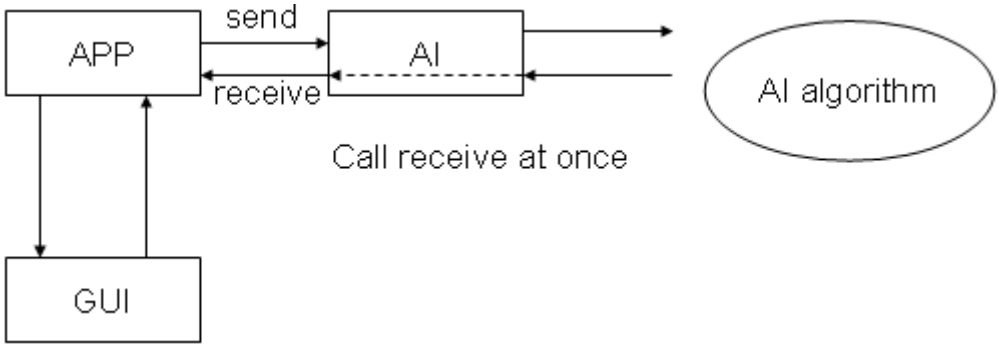



圖 3-19、AI 模組模擬網路示意圖

另外要注意的是，本框架在攜帶裝置上發展，以目前而言手持裝置的能力必定遜於一般電腦。因此發展 AI 時需十分注意系統資源的使用，像是記憶體與磁碟空間等。建議是先設計比較單純的 AI 模組，之後根據裝置效能來加強使用資源的力度。

### 3.2.3 類別列表

這裡列出目前所有已設計出的類別與相對應的函式，並會介紹每一元件的函式應當如何運用。

| Application | Stage | Description |
|---|---|---|
| fnRun() | 0 | Application entry |
| NetworkEnable() | 1 | Chose enable network |
| Core() | 2 | MakeMove, SelectCard |
| RecvMessage() | 2 | Receive network's side message |
| End() | 3 | Procedure when game is over |

表 3-1、類別列表 Application

GameApp 和 CCPKApp 一樣是整個應用程式的中樞，此外它還負責讓遊戲正常運行。同樣地，GameApp 使用者也需要覆寫 fnRun()這個初始化函式，但我們在 template 中已經寫好一個範例，幾乎不要做更動。因爲我們在第一階段把網路模組包裝得比較完整，舉凡選擇連線方式及後續連結到遊戲內溝通，所以這裡提供了 NetworkEnable()函式讓使用者選擇要使用哪些通訊格式。Core()函式在各個樣板中有不同名稱：MakeMove、SelectCard，這個函式代表整個程式的心臟，所有遊戲邏輯、GUI 呼叫、State 更新、Rule 使用大都在此完成。這

裡定義出來讓使用者覆寫，當然他們可以自己定義類似功能的函式。原本在網路模組中有 fnOnRecv()這個接收網路訊息的處理函式，可是我們的構想中應該把處理遊戲資料的動作集中到同一類別，因此我們把這項功能拿到 Application 中，以 RecvMessage()處理。End()函式則是在遊戲結束時呼叫。

| State | Stage | Description |
|---|---|---|
| fnInit() | 0 | Class Initialization |
| update() | 2 | Update game data |

表 3-2、類別列表 State

State 類別其實大都儲存一些靜態資料，比較少函式需要覆寫。但因爲遊戲的資料一定不會相同，所以使用者在繼承時勢必要多宣告些參數。fnInit()是初始化函式，通常是在 Application 的 fnRun()中呼叫。update()則是當遊戲狀況有變動，譬如有人下子時，呼叫 update()去更新資料。

| Rule | Stage | Description |
|---|---|---|
| CheckMove() | 2 | Check if the move is legal |
| IsGameOver() | 2 | Check if the game is over |
| GetWinner() | 2 | Return game's winner |

表 3-3、類別列表 Rule

Rule 類別裡定義的函式主要是根據 State 內的資料計算結果，通常由 Application 呼叫。這裡定義的函式只是最基本的，使用者可以自行擴充。CheckMove()判斷傳進來的遊戲選擇(如：下子)合不合法、

IsGameOver()判斷遊戲有沒有勝負、GetWinner()回傳勝利者是誰。

| GUI | Stage | Description |
|---|---|---|
| fnInit() | 0 | Class Initialization |
| ShowX() | X | Screen show at Stage X |
| fnOnCommand() | 1/2/3 | Menu command event handler |
| fnOnKey() | 2 | Key event handler |
| fnOnPoint() | 2 | Touch screen event handler |
| fnOnButton() | 2 | Button event handler |

表 3-4、類別列表 GUI

GUI 類別要做的最重要的兩件事爲：畫面呈現和事件處理。ShowX()函式的 X 代表 Stage 數，如 Show1()代表 Stage1 時的畫面要怎麼畫。一些基本的圖形展示各個樣板和遊戲設計層 API 都已經提供好，使用者要做的就只有微調和載入自己的圖片。要再提的是，在 Stage 0,2,3 的畫面展示比較單純，通常只有一種表示方法。但在 Stage1 因爲 Set up 的步驟比較多，Show1()要根據目前進行到哪個步驟而有不同的畫法。事件處理方面各樣版其實已經大都實做好了，使用者同樣地可以根據自己的需要作改動。

| Network | Stage | Description |
|---|---|---|
| fnSetup() | 1 | Set up the network environment |
| fnConnect() | 1 | Connect to network |
| fnSend() | 2 | Send |
| fnOnRecv() | 2 | Receive message event handler |
| fnClose() | 3 | Close connection |

表 3-5、類別列表 Network

前面有提到在遊戲設計這一層，網路模組已經包裝得很完整。表內列出的函式都不用在覆寫，像是 fnSetup()、fnConnect()的使用都已在樣板內隱藏起來，使用者只要會用 fnSend()傳送訊息，並在 Application 用 RecvMessage()接收，就可輕鬆使用網路功能。

# 第四章、實做及相關問題探討

本章會述說系統的一般性開發原則，並針對實做時發生的問題與特例來探討，最後會展示以此框架開發出來的範例遊戲〔六子棋〕。

## 4.1 一般性開發原則

目前本論文以 Symbian 2.0 與 WinCE 爲開發平台，除了要熟知各平台的基本資源、開發方式外，還要分析兩者間差異，設計出可以通用的結構。在實做過程中，我們規納了幾個設計要點，藉此解決平台間的異質性。

- 同質化：

  盡量運用基本常見的資料型態，或是定義出共通的元件，再交由各平台開發者實做。

- 簡單化：

  使用基礎的函式或是 C/C++規格的型態。不使用平台本身才有的物件。或許各平台有提供較好或較方便的設計，但若另一平台難以仿製，不如採用較簡單的設計。

- 多型：

  定義出更多的高階函式供使用者覆寫，達成簡單客製化的目的。

- 避免差異：

  變數以*PVOID 傳遞，在各平台內實做時再轉換爲平台內部的型別，將平台獨有的結構完全隱藏。

- 繼承原則：

  有些系統功能和平台關聯很緊密，在實做時我們就直接繼承系統本身的預設類別，如：GUI。而有些功能比較像是外掛性質，有時要在基礎元件上增加新 功能。這時我們就定義一個共通界面，由這個界面去操控原始元件。

## 4.2 問題與特例

在一些行動裝置上其實並沒有支援螢幕碰觸，如果只是單純的把 Point 這個事件切換掉，不是什麼問題。但是在使用 GUI 模組中的 Button 元件時，一定得使用螢幕碰觸來觸發。一般的遊戲中 Button 時常被使用也十分方便，因此隨意關掉 Button 元件會造成開發上的窒礙。我們的解決辦法是在使用未支援螢幕碰觸的裝置時，將 Button 元件直接轉換爲 Menu 元件(如圖)，這樣就可避免不支援的問題。

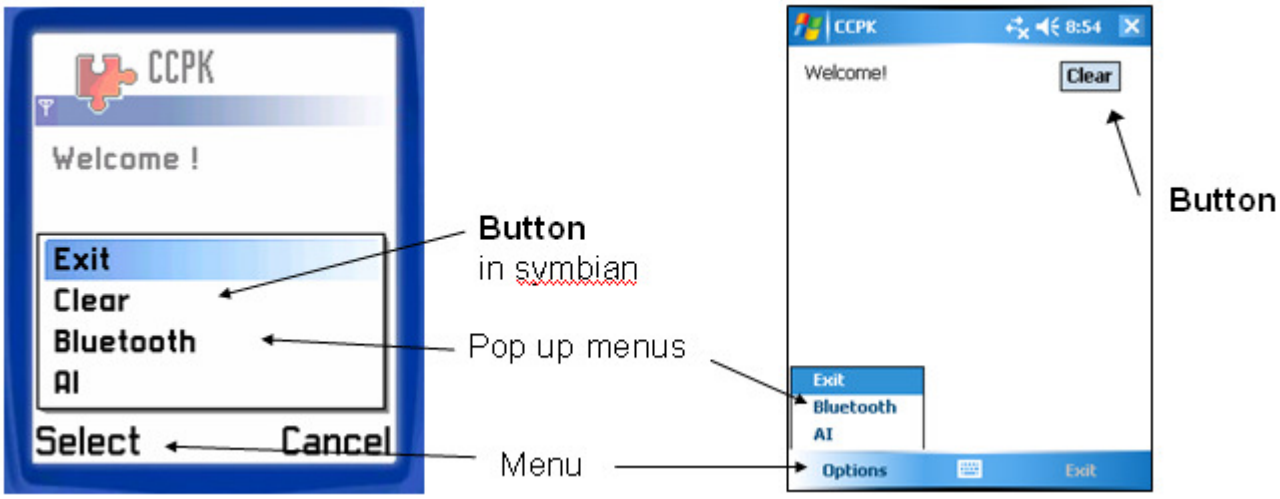





圖 4-1、GUI Button 特例

## 4.3 實驗結果

我們選擇的 Symbian 版本爲 8.0a: S60_2nd_FP2_CW 套件，開發出來的應用程式執行於 PC 模擬器(Windows XP, AMD 1.6G, 2G RAM)與 Nokia 手機 N70，螢幕大小皆爲 144x192。

第一個範例遊戲爲六子棋，六子棋的規則爲黑先下一子，之後兩方輪流下兩子，直到有人連成六子。下圖中第一個畫面爲一開始先選擇連線方式，第二張展示遊玩的過程棋盤的狀態，第三個畫面展示勝負揭曉後的情形。

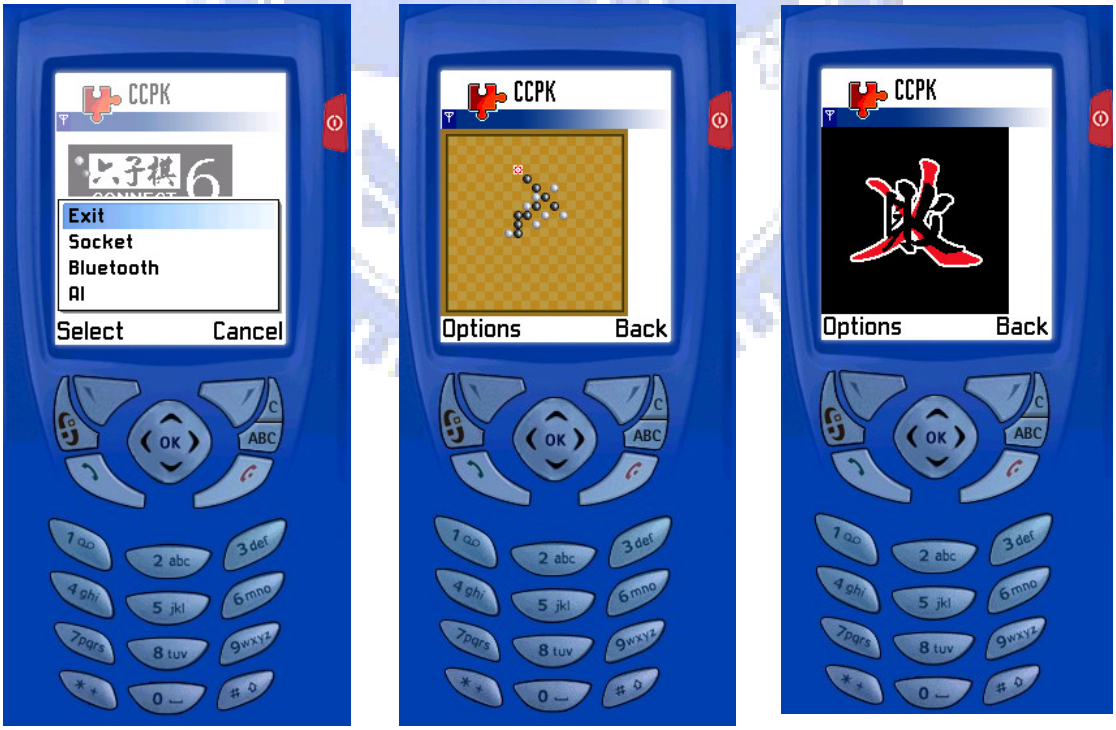



圖 4-2、Connect 6 at Symbian

第二個範例遊戲爲二人大老二，遊戲規則爲兩人根據牌型出牌，誰先出完牌即爲勝者。第一張圖同樣代表一開始先選擇連線方式，第二張爲遊戲中出牌的情況，第三章展示結束的情形。

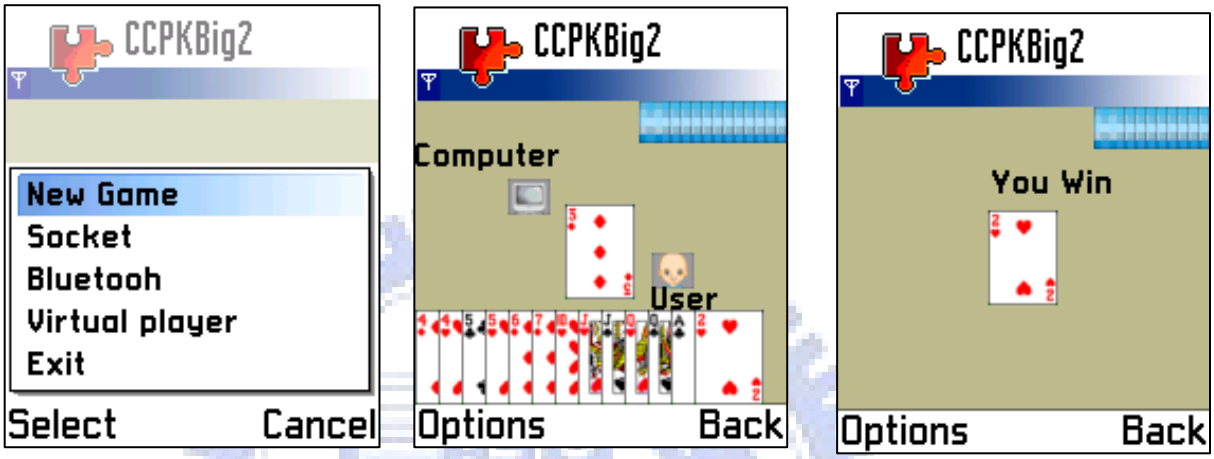



圖 4-3、Big2 at Symbian

# 第五章、結論與未來發展

在本論文當中，我們設計了一個跨平台的手機遊戲發展框架，這個框架主要是用來發展棋類、牌類的手機遊戲。此框架有以下特點：

- 一個跨平台行動裝置遊戲框架：

  爲因應不同平台，我們設計了一種新的架構。最底層會分成不同平台而有不同內容。第二層則是使用者可以碰觸的地方。藉由統一的程式呼叫，以達到跨平台的功能。第三層針對遊戲設計，提供一個包含網路、圖形、UI 的 Framework，使使用者能輕鬆的完成一款桌面遊戲。

- 定義高階開發環境：

  在完成一款桌面遊戲後(六子棋、撲克)必會對手機桌面遊戲所需要的項目有所了解，我們以此經驗抽離出共通、常用的功能，最後規劃出一個模組流程。使用者只要像填空一樣就可造出一款遊戲。

- 共通網路介面：

  我們設計了一套網路介面格式，能讓使用者不需理解底層通訊協定，就能輕鬆使用網路功能。對開發者，也提供一套簡便合理的網路元件開發規範，只要按照規範設計，便能直接將新的連線方式接合到我們的框架。

開發一整套跨平台軟體框架，其實要考慮的地方有很多。本論文至此提出了一個基礎的解決方案，未來仍有些課題可以繼續深入探討。以下列出一些思考方向，希望能夠針對這些項目加以改進，進而完善我們的框架。

- 對應更複雜的網路模組：

  目前本框架實做了最基礎的 Socket、Bluetooth、AI 等網路通訊協定。但是一款上線的網路遊戲還包含了用戶系統、認證機制等，且部分類型遊戲還會與網頁做連結互動。希望未來能對應此類型的網路溝通模式，以達更廣泛的運用。

- 支援更多平台：

  目前我們實做了 WinCE、Symbian 平台，但還有其他平台，例如：Palm OS、Android 等。希望未來能支援更多平台。

- 雙緩衝(目前最新版已支援)：

  雙緩衝(Double Buffering)是在繪圖技術中一個相當實用的作法，但目前在我們的設計中並沒有包含這個技術。經過初步研究，在手機上使用雙緩衝是可行的。

- 動態遊戲：

  目前我們針對的遊戲類型主要以桌面遊戲爲主，此類遊戲對畫面即時表現要求不高。如果在即時性的動態遊戲中，手機裝置的效能會成爲一個很大的課題。

目前我們已使用本論文發展的框架設計了幾個範例遊戲，雖然就遊戲內容方面都很完善，但僅侷限在於靜態的桌面遊戲。希望未來繼承本篇研究的學弟妹能更加擴展框架的功能和應用性，或是增加支援的平台。也希望有興趣開發跨平台手機遊戲的朋友，能從本篇論文得到幫助。

# 附錄

這裡會列出本論文由開始研究到系統成型所使用到的參考資料和知識，包含：平台程式開發方式、系統安裝、偵錯步驟等。及本框架細部的使用流程，包含：預備設定、API 控制、開發技巧等。

以下介紹 Symbian 開發方式。

- 基本檔案格式解說：
  - .bld 專案檔，Import .bld file，載入專案。
  - .mmp 內爲系統資訊檔，包含系統基本選項、專案內的檔案、使用 Library、圖片定義等。
  - .pkg 定義實體安裝使用的 sis 檔格式，在此務必將圖片路徑設定正確。
- 框架實做部分：
  - AppC6::Run()內註解起來的 code 是有試出來的功能，包含 Time、File I/O。
  - 可用#define 達到類似 Refractor 的功能(CCPKTestApp.cpp)
  - TCallBack 實做 timer 和 Refractor 所使用的重要類別。
  - Douber Buffering，注意 GUI 的 iOffScrnBmp。
  - Socket 使用了兩個 Active Object，用在 Send/Receive。
  - Bluetooth 類似 Socket 的作法。

以下介紹 Symbian 系統安裝。

- Symbian 安裝法：
  - 安裝 ActivePerl。

- 安裝 S60_2nd_fp2_sdk。
- 安裝 Carbide.c++_v1.2。
- Java 版本不可太新 1.4~1.5。

這裡講解 Symbian 開發技巧，包含模擬器、藍芽功能、偵錯等。

- Socket 模擬器使用法：

  模擬器 Perference 內 Ethernet Settings： 填 IP 等。

- 藍芽模擬器安裝使用方法：
  - 安裝藍芽 USB 接收器(使用光碟驅動程式)。
  - 至裝置更新接收器的驅動程式(s60bt) 。
  - 在使用的 symbian OS 目錄下跑 sbt.exe(S60_2nd_FP2_CW)
  - 在模擬器(release)內的 perference->PAN ， Enable Bluetooth， COM 選 sbt 找到的，HCI 選 H4。
  - 參考文件: S60 Bluetooth Driver Installation Guide.pdf。
  - 實機：不能使用 Global 變數，不然無法安裝到實體機器。

框架細部結構部份，Independent Layer 的五大類別加 AI 已經在論文中列出了。框架 API 列表，請參照 header 檔。棋類樣板請看六子棋範例，牌類請看大老二範例。

這裡是一些框架的開發技巧：

- 繼承的物件(C6GUI 之類)最好是內部在宣告一個關連物件，例如 AppC6 內處理 MyGUI 這個 GUIC6 物件而不是直接用 CCPKApp 下定義的 CCPKFrameWnd。
- 靜態資料放到 State 內，動態函式放到 Rule 內。

下面展示在不同平台 Initialization 的實際情形。

這部份是 CCPK 所定義的核心，使用者需要繼承覆寫類別像是 GUI、App 等類別，並且實做類別定義出的虛擬函式。等到程式真正執行時，CCPK 會自己在相應的時間呼叫它們。

```
Class GUIC6 : public CCPKGUI {
    void fnOnKeyEvent(…);
    void fnOnDraw(…);
…
}
Class C6App : public CCCPKApp {
    void fnRun(…);
…
}
```

另外使用者還要覆寫 CCPK_Init 這個函式，指定建構的類別型態。

```
void CCPK_Init()
{
    cycMainFrm = new GUIC6;
    cycApp = new C6App;
}
```

這是 WinCE 的情形，框架裡會做的動作。WinCE 把框架所定義的類別直接宣告在 Global 處，在 WinMain 呼叫初始化函式。

```
CCPKGUI * cycMainFrm;
CCCPKApp * cycApp;
int WINAPI WinMain
{
        CCPK_Init();
}
```

這是 Symbian 的情形，框架裡會做的動作。因爲 Symbian 包裝的比較完整，框架定義出的類別必須放到 AppUI 類別下，再由此類別的初始函式負責初始化的工作。

```
Class CtestAppUi : public CAknAppUi
{
CCPKGUI * cycMainFrm;
CCCPKApp * cycApp;
};
CCPKAppUi::ConstructL()
{
 CCPK_Init();
}
```

另外各位還可以參考許珉嘉學長[6]這篇論文內的附錄。

# 參考文獻


[1]武漢卓睿軟體有限公司(Corbile Software Co., LTD)，ECDS-MUI 跨平臺手機應用程式開發庫，2007。

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

[3]陳淩彬，「網際網路遊戲發展平臺之研究」,國立交通大學資訊工程學系，碩士論文，June 2001。

[4]陳智文，「手機遊戲發展平臺」，國立交通大學資訊工程學系，碩士論文，June 2004。

[5]經濟部工業局，2007 台灣數位內容產業年鑒，2008。

[6]許珉嘉，跨平台手機遊戲開發框架在 WinCE 手機之研究，國立交通大學資訊工程學系，碩士論文，July 2008。

[7]Andreas Janecek, Helmut Hlavacs; Programming Interactive Real-Time Games over WLAN for Pocket PCs with J2ME and .NET CF; ACM Press New York, NY, USA, 2005.

[8]J. Ousterhout, Why threads are a bad idea (for most purposes), invited talk at the 1996 USENIX Technical Conference, San Diego, CA, USA, 1996, see also http://home.pacbell.net/ouster/threads.pdf (last access: May 2005).

[9]Jacco Bikker, Mobilecore: A Cross-Platform Framework For ARM-Based Mobile Games, Gamasutra, 2003.

[10]Jo Stichbury, Symbian OS Explained, Symbian Press, Oct 2004.

[11]Michael Morrison，從做中學：J2ME 手機遊戲程式設計，2005。

[12]Michael J Jipping, Symbian OS Communications Programming, Symbian Press, June 2002.

[13]Mobile Information Device Profile, v2.0 (JSR-118), JCP Public Draft Specification Java 2 Platform, Micro Edition TM.

[14]Nokia, #course 4300 Symbian_OS_Basics_Workbook_v3.1, January 2008.

[15]Nokia, #course 5300 Developer Training Program Series 60 Platform C++ Development, August 2004.

[16]Nokia, Developer Training #course 6300 7100, 2005.

[17]palm, palmOS, available from http://www.palm.com/tw/zh/, 2008.

[18]Richard Harrison, Symbian OS C++ for Mobile Phones, Volume 1, Symbian Press, April 2003.

[19]Richard Harrison, Symbian OS C++ for Mobile Phones, Volume 2, Symbian Press, Aug 2004.

[20]Wikipedia, palmOS, available from http://zh.wikipedia.org/wiki/Palm_OS, 2008.