國立交通大學

資訊學院　資訊學程

碩士論文

跨平台手機遊戲開發框架在 Windows CE 手機之研究

The Study and Development of Windows CE for Cross-Platform Mobile Game Framework

研 究 生：許珉嘉

指導教授：吳毅成　教授

中 華 民 國　九 十 七　年 六 月

跨平台手機遊戲開發框架在 Windows CE 手機之研究

The Study and Development of Windows CE for Cross-Platform Mobile Game Framework

研 究 生：許珉嘉　　Student：Min-Chia Hsu

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

資 訊 學 院　資 訊 學 程

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

跨平台手機遊戲開發框架在 Windows CE 手機之研究

研究生：許珉嘉　　指導教授：吳毅成

國立交通大學　資訊學院　資訊學程碩士班

摘要

隨著手機越來越流行，硬體功能的加強以及3G無線網路的普及，越來越多人不再只是為了電話而攜帶手機，手機所帶給人們的娛樂功能也越來越被受到重視。而遊戲正是娛樂功能中相當重要的一環，不論是電信業者、手機業者或是遊戲業者都在手機遊戲產業中積極推廣並開拓其中之商機。然而面對多平台的手機環境，遊戲開發人員面對的是必須了解各平台的特性，人力因此而分散於各個作業系統的研究上，而無法專注在遊戲內容的開發，至為可惜。

在本論文中，我們依據作業系統的特性與遊戲開發的特性分為兩層的框架。一層為與平台相關的框架，透過這一層的封裝，將各個平台中的差異性予以統一，讓上層使用者可以透過統一的介面操縱程式。另一層則為與遊戲開發相關的框架，透過這一層所建立的框架，遊戲開發者可以依照遊戲流程快速的開發遊戲。希望此篇研究能夠帶來跨平台遊戲開發的解決方案。

The Study and Development of Windows CE for Cross-Platform Mobile Game Framework

Student: Min-Chia Hsu Advisor: I-Chen Wu

Degree Program of Computer Sceinece
National Chiao Tung University

ABSTRACT

Accompany with mobile phone more and more popular, mobile hardware capability enhancement and 3G wireless network prevalent, more and more people take mobile phone not only for phone function but for entertainment function. The entertainment capabilities of mobile phone become more and more important. Game is one of important role of entertainment. No matter telecom, mobile device producer or game developers are expanding business in it. However, since mobile phones are multi-platform, so game developer needs to spend much resource on the character of each platform. It is pity that resources focus on platform characters not rich game functionality.

The focus of this thesis, we design two layers framework base on platform characters and game development characters. By capsule platforms differential with common interface, developer can implement program by the common interface. By follow game development framework, game developer can create their game quick and easily. We hope this research can provide a solution for cross-platform mobile game development.

誌謝

首先，要感謝的人是我的指導教授，吳毅成博士，由於他的循循善誘及耐心的教導，並提供許多寶貴的意見與指正，正確的引導研究方向，這篇論文才能順利完成。

特別要感謝的是學弟高偉傑，因為與他不斷相互討論，共同研究，才能激發出此篇論文，同時也感謝學弟李柏甫幫忙工具的開發以及學長汪益賢及實驗室的學長學弟們所給予的協助。此外，還要感謝公司的主管廖正國讓我能夠在工作上有十足的彈性，在工作與求學中取得平衡。

最後要感謝我的父母，在我工作與求學的雙重壓力下，給予我最大的支持與鼓勵。謹以此論文，獻給我最摯愛的家人以及關心我的朋友們。

目錄

圖目錄

第一章、 緒論

本章會說明目前行動裝置遊戲在市場的推廣狀態，同時指出面對多樣平台的行動裝置時，程式開發者所面臨到的問題。最後，會說明本論文的論文大綱。

1.1 前言

隨著手機這 20 幾年來的蓬勃發展，手機已經成為十分普及的隨身電子產品，它帶來的便利性，也使它成為多數人生活中不可或缺的工具，深深影響著我們日常生活型態。

手機被視為下一波電子產業的重要產品，也因此越來越高階的處理器被整合到手機，賦予手機越來越高的運算能力。這樣的能力僅拿來單純的通話使用當然是太過於浪費，手機相關業者無不希望在單純的通話功能外，在娛樂功能上面有更多加值的服務。再加上無線網路與 3G 時代的推波助瀾，手機網路功能也越來越受到重視。依據 Juniper Research 的研究報告，2007 年全球手機遊戲市場為 30 億美金，2011 年可望成長到 176 億美金。手機娛樂功能從單純的影音功能延伸到遊戲是目前的趨勢，不論是手機下載遊戲或是手機連線遊戲都越來越受到使用者的注目。

目前的智慧型手機作業系統主要有 Symbian[15], Windows CE[11], Palm[12] and Linux[9]。其中又以 Symbian 與 Windows CE 兩種作業系統分佔市佔率的第一，第二名。此外，此兩種平台也有完整的開發環境以及完整的技術支援，使得程式開發者得以快速的開發程式。

然而，面對不同的作業系統，遊戲開發業者勢必培養程式開發人員對於不同作業系統進行實作，此舉勢必相對減少對於遊戲內容的開發資源。但是豐富的遊戲內容才能吸引使用者，因此如何減少支援多平台時所需的人力以及縮短遊戲發展時程就相形的重要了。因此建立一個跨平台的遊戲開發框架（Framework），將成為本論文的目標。

1.2 研究動機

要能支援多平台的程式，代表著需要培養更多人去學習不同平台的開發環境，以及各個平台的特別架構。而隨著各個平台隨時間演進，又需要更多人力去了解不同版本的差異。因此，如何設計出一套跨平台的函式庫，使得只需少數人專精於跨平台的函式庫開發，而讓更多人力可以去開發更豐富、有趣的遊戲內容是本論文想要達成的目標。

此外，本論文以較為靜態的遊戲（如棋類、撲克牌類）為切入點[17]，探討設計高階遊戲的函式庫，針對遊戲流程開發建立一套通用的框架，使得此類遊戲能夠縮短開發時程。

藉由本論文提供的函式庫，它隱藏了不同平台的差異，程式開發人員只需了解本函式庫的用法，透過統一的應用程式介面，便可使用同一套程式碼，在不同的開發平台上使用本函式庫開發遊戲。

1.3 論文章節說明

本論文第一章為緒論，簡單介紹目前行動裝置遊戲在市場上推廣的狀況，以及要對多平台的行動裝置開發遊戲時所會面臨到的問題，

並說明本論文提出的解決方法。第二章則介紹目前常見的行動裝置作業系統，以及哪些解決方案已被提出，並且說明本論文選擇 C/C++做為開發語言的原因。第三章則說明本論文針對跨平台程式開發所設計的 Framework，其中分為與系統相依層以及與遊戲流程開發相依層的設計。第四章則以實作雙人大老二遊戲為例，說明開發時所遇到的問題，以及所採取的解決方式。最後第五章提出結論及未來的展望。

第二章、 背景說明

本章以介紹目前常見的行動裝置作業系統，並且介紹目前已知提出跨平台解決方案的例子及其缺點，最後並說明選擇 C/C++做為語言開發的理由。

2.1 常見平台概述

根據市場調查研究機構 Canalys 所公佈的資料顯示，2007 年智慧型手機出貨中，Symbian 作業系統佔有 67%拔得頭籌，Windows CE 以 13%市佔率居次。本節中將介紹上述兩種手機作業系統以及 2007 年剛進入手機業即備受矚目 iPhone 作業系統[1]。

2.1.1 Windows CE 作業系統

Windows CE 是微軟視窗作業系統中最小的作業系統，是專門為記憶體及處理器資源較少的硬體所設計的作業系統。它支援 Win32 子集合的 API，具有輕量（與 Windows XP 作業系統比較而言）、支援多執行緒（Multithreaded）的特性，並且可以選擇是否在系統中包含圖形使用者介面元件，也因為其彈性的客製化能力，作業系統的大小從幾百 KB 到 64MB 都可以，目前已經應用在 PDA、智慧型手機、車用電腦、網路消費性電子產品等上[11]。

在系統特性上，它支援 32-bit 虛擬記憶體定址方式，並且在各個應用程式間也有記憶體保護機制，可以避免不當的應用程式干擾其

他應用程式的執行。在虛擬記憶體配置上，它支援 heaps、stacks 以及 memory-mapped files。與 Windows XP 相似，它也是先佔式多工架構（Preemptive multitasking）以及多執行緒的作業系統。

在硬體支援部分，Windows CE 支援多種處理器，如 X86, ARM, SHx, MIPS。並且在作業系統中即設計了 OAL（OEM Adaptation Layer），使得手機製造業者能夠依照本身的硬體元件修改原始碼之後，可以快速的將自己的硬體元件與作業系統整合，加速了硬體開發時程。

在軟體開發部分，雖然其核心與 Windows ME、Windows XP 不同，但是提供與 Win32 API 高相容性的 API，使得 Windows 桌上型程式開發人員能夠快速的移轉到 Windows CE 平台上進行開發，大幅度縮減技術人力導入時間。此外，它也支援 Windows 作業系統中常用的函式庫，如 COM（如 ActiveX、DirectShow 等）、MFC、ATL、STL 以及.NET（在 Windows CE 中有一個專門的.NET 版本，稱為.NET Compact Framework)[3][4]。

圖 2-1 為 Windows CE 訊息流程圖，其表示在 Windows CE 中開發程式，程式與系統的關係[6]。系統收到該程式對應的訊息後，便將訊息放在該程式對應的訊息佇列（Message Queue）。程式便不斷的去讀取訊息佇列中的訊息，並且透過自己的處理程序（Procedure）分配對應的函式來處理相關的訊息，若是沒有配對的函式，則呼叫 Windows CE 預設的函式“DefWindowProc()”來處理。

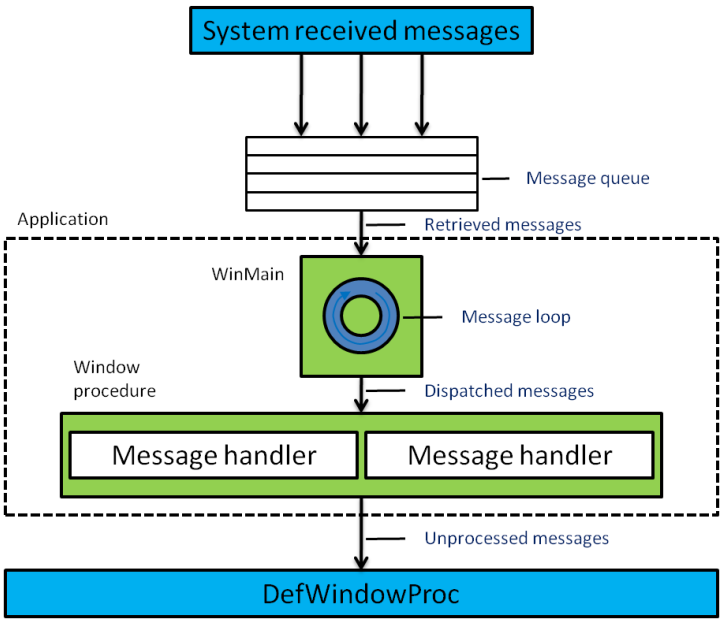


圖 2-1、Windows CE 訊息流程圖

2.1.2 Symbian 作業系統

Symbian 作業系統是由 Symbian 公司所發展的作業系統，而 Symbian 公司則為 Nokia、Ericsson、Sony Ericsson、Panasonic 以及 Samsung 公司所共有。Symbian 一開始就是為了行動裝置所設計，因此對於記憶體資源有極高的效率，採用節省記憶體、清除堆疊等技術。在處理器上則是採用 ARM 系列。

Symbian 是以 EPOC（從 Epoch 命名而來）作業系統為基礎修改而來，與許多桌上型作業系統相似，它支援先佔式多工以及多執行緒，並且在不同的應用程式間提供記憶體保護的機制[8]。

有關 Symbian 作業系統更詳細的分析，請參照同實驗室學弟高偉傑的論文[18]。

2.1.3 iPhone 作業系統

iPhone 作業系統是由蘋果公司（Apple Inc.）所發展，並且應用在 iPhone 與 iPod touch 裝置中，最近一版預計與 iPhone 3G 於 2008 年 7 月 11 日釋出。除了支援多工架構，系統核心分成三層抽象層[1]:

- Core Services 層: 系統底層核心及硬體溝通功能。
- Media 層: 2D/3D 繪圖，影音相關功能。
- Cocoa Touch: 圖形介面以及三度立體感應器功能。

iPhone 最讓人驚艷莫過於它的人機介面的設計，它採用直覺式操作設計，並且使用多點觸碰功能(Multi-Touch)，而所使用的處理器則為 ARM 系列。

其開發環境採用 Xcode 開發環境，並且提供 iPhone SDK 專門為 iPhone 作業系統開發，支援的程式語言有 C/C++以及 Objective-C/C++。

2.2 相關解決方案

面對手機多種的作業系統，想要提供跨平台的程式開發的解決方案在之前就已經有人提出了，本節將介紹兩種之前提出的解決方案，並說明其優缺點。

2.2.1 ECDS

ECDS (Embedded Cross-Platform Development Solution) 是由中國大陸的軟體公司-武漢卓睿軟件有限公司所開發[16]，是為手機開發應用程式實現跨平台所設計的一套開發軟體。支援開發平台有 Symbian S60 1st/2nd/3rd、Pocket PC/SmartPhone 2003 以及 Windows Mobile 5.0，並且可以在 Windows 2000/XP 平台上進行開發以及除錯。提供了圖像、聲音、使用者介面、網路等函式庫，使得程式開發人員可以不需了解 Symbian、Windows CE 等不同作業系統的特性及可進行程式開發。

然而由於並非開放源碼 (Open Source)，在加上本身程式上有一些問題 ，如網路函式庫的臭蟲 (bugs)，因此限制了程式開發的可靠性。另外，該開發軟體在 2008 年初就不再提供新的版本更新了。

2.2.2 MobileCore

MobileCore[5]是一位名為 Jacco Bikker 的荷蘭籍程式開發人員於 2003 年在 SourceForce[13]上開啟的一個 open source project，目的即在對於為開發跨平台的手機程式，建立一套函式庫。目前它支援的平台有 Symbian，Windows CE and Windows PC。

它的核心相當簡單，擁有三個主要的類別(class)：

- Core-處理平台的程式入口以及主要的視窗程式。
- Sample-處理聲音相關的功能。
- Surface-處理影像相關的功能。

然而，由於該函式庫所提供的功能過少，例如沒有提供常用的使用者物件，如選單（Menu）、文字區塊（Label）等，也沒有提供網路的功能，因此對於遊戲開發而言是明顯的不足。此外，由於後繼的人力不足，因此在 2003 年底之後就沒有更新版本釋出了。

2.3 選擇開發之程式語言

一般提到跨平台程式開發，常被討論到的就是 Java。的確，Java 具有跨平台的能力，只要該平台有對應 JVM (Java Virtual Machine) 環境即可。但是，不同平台的 JVM 支援的能力不盡相同，因此也造成實際上還是需要在各平台上除錯，才能真正肯定執行沒問題。而本論文所提出的函式庫選擇的是 C/C++語言，它的好處如下:

- C/C++在多平台的環境下皆支援。
- 比起 Java 而言，少了 JVM 則可以減少記憶體的消耗，對於行動裝置而言是很珍貴的資源。
- C/C++可針對平台的特性做最佳化的處理，也因此其執行速度能夠優於 Java。

第三章、 CCPK 框架設計

本章說明本論文設計一個遊戲開發的框架（Framework），命名為 CYC Cross-Platform Kit（CCPK），其主要的設計想法，並且詳細解釋 CCPK 內，兩層的設計方式，一層為平台相依層，將不同的平台底層隱藏，提供一個統一的程式介面；另一層則為獨立遊戲層，設計一個標準框架利於遊戲開發流程。

3.1 簡介

將不同平台內的功能予以分析，並分成模組類別，以求得跨平台的標準流程，是設計 API 的主要方向。設計優良的跨平台 API，可以使得程式開發人員不必花費多餘的心思在平台的特性上，而只要面對單一的跨平台 API，從而有更多的資源放在發展更多更有趣的遊戲。

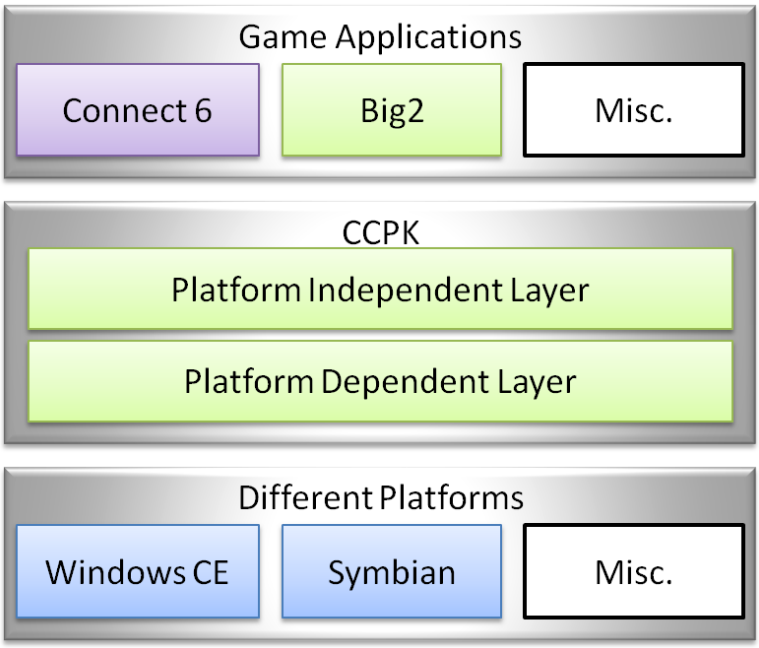


圖 3-1、使用 CCPK 開發遊戲架構圖

圖 3-1 中，遊戲應用程式使用 CCPK 的框架（Framework），其中包含平台相依層以及獨立遊戲層，只需要同一份程式碼，就可在不同平台上執行遊戲。

3.2 平台相依層設計（Design of Platform Dependent Layer）

本節說明如何將事件驅動程式設計導入 CCPK 中，並且介紹平台相依層中與系統相關的三個模組: 應用程式模組（Application Module）、圖形使用者介面模組（GUI Module）、網路模組（Network Module）。

3.2.1 事件驅動程式設計（Event-Driven Programming）

事件驅動程式設計是一種程式設計模式，這種模式的程式，其流程掌控是透過使用者的動作(如按下按鈕，在觸控螢幕上點擊等)或者是系統通知程式特定的事件發生，甚或是程式觸發給程式本身來決定的。相對的另一種程式設計模式則為批次程式設計（Batch programming），其程式流程掌控則是由程式開發人員所決定。

事件驅動程式設計的崛起是因為互動的操作需求所導致，目前大部分的圖形使用者介面的程式大多採用事件驅動程式設計，當然遊戲也不例外。此外，對程式開發人員而言，事件驅動程式設計的開發也

相對的比多執行緒的程式單純，因為事件驅動程式設計大多是利用同一個執行緒去處理不同的事件處理器（Event handler），因此可避免多執行緒需要額外掌控程式同時讀取資源（如記憶體，資料庫等）所發生衝突。而且可以減少除錯時的困難度[7]。

3.2.1.1 事件驅動程式設計模式（Event-Driven Programming Model）

一個事件驅動程式模式如圖 3-2 所示，當系統收到該程式對應的事件時，系統會將這些事件依序放在事件佇列中。程式則會有一個函式迴圈不斷的檢查事件佇列中是否有尚未執行的事件，並且透過分發器（Dispatcher）配對適合的事件處理器，以處理相對的工作。

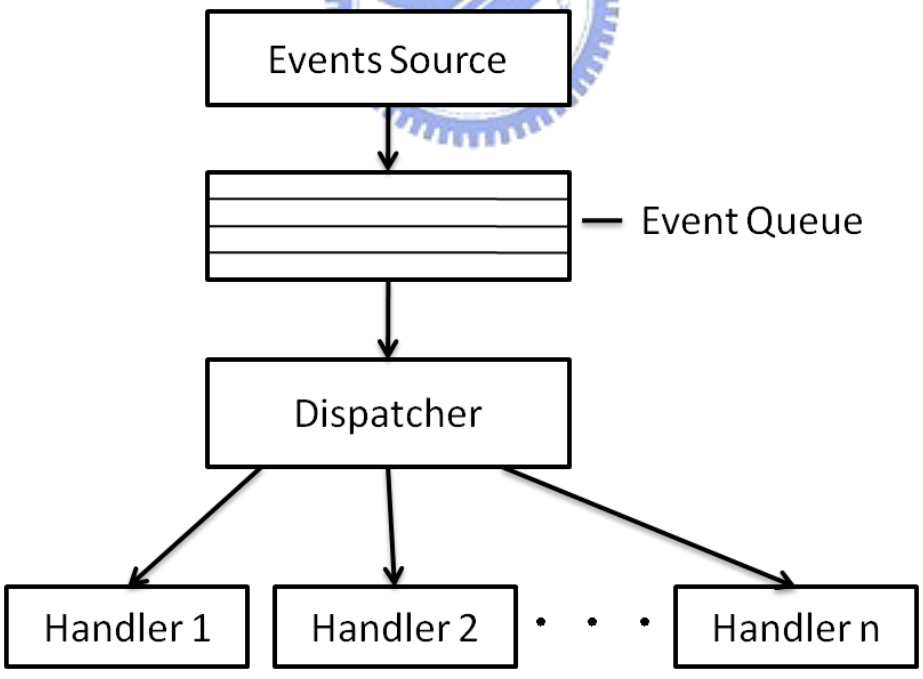


圖 3-2、事件驅動程式設計模式

3.2.1.2 Windows CE 事件驅動程式設計模式

在 Windows CE 中事件驅動又被稱為訊息驅動（Message - Driven）。如圖 3-3 所示，Windows 程式的入口為“WinMain()”，透過“WndProc()”將對應訊息分配到適當處理函式。當系統將屬於該程式的訊息放置於訊息佇列中（Message Queue），程式內會有一個讀取佇列的迴圈不斷去取出佇列並交由分配器處理對應的處理函式[2]。

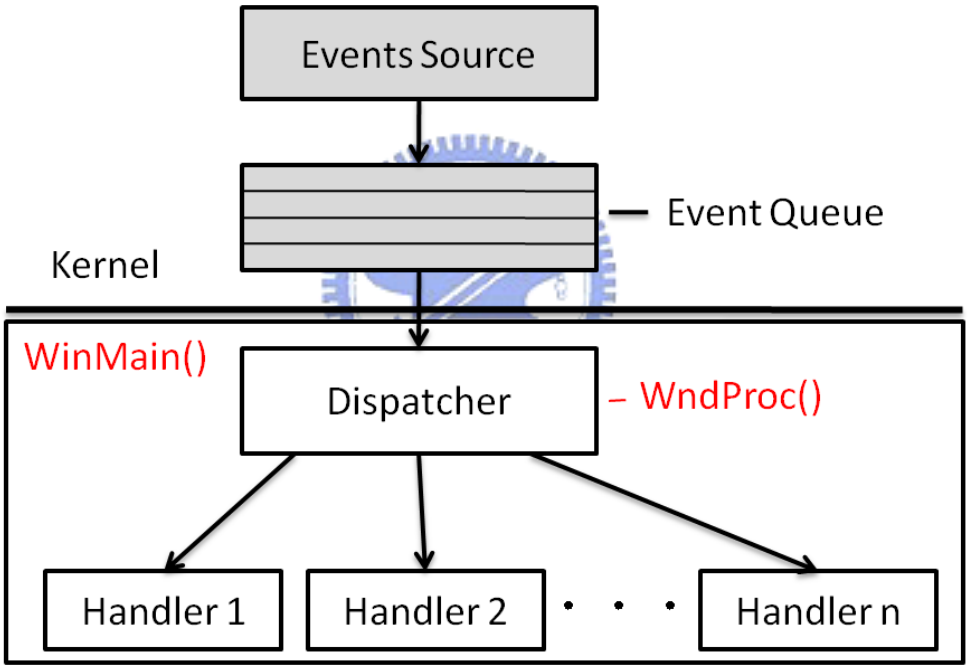


圖 3-3、Windows CE 事件驅動程式設計模式

WinMain()主要的功能在於產生一個主要視窗以便接收訊息佇列，並且執行一迴圈去讀取訊息佇列，其虛擬程式碼如下。

```
int WINAPI WinMain(HINSTANCE hInstance,…)
{
    MSG msg;
    // Create main frame window
    RegisterClass(hInstance, szWindowClass))
```

```
    HWND hWnd = CreateWindow(szWindowClass, …);
    ShowWindow(hWnd, …);
    // Main message loop:
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return (int) msg.wParam;
}
```

WndProc()則為分配器的角色，它在註冊主要視窗類別(RegisterClass()) 的時候，與視窗建立關係。之後一旦程式讀取訊息佇列後，系統就會呼叫此函式，程式便可以執行對應的功能。若是程式不處理該訊息，則呼叫系統的 API-DefWidowsProc()。其虛擬程式碼如下。

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
    case WM_COMMAND:
        break;    // do something you want when menu was selected
    case WM_CREATE:
        break;    // do something you want when the window just created
    case WM_PAINT:
        break;    // do something you want when windows need repaint
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

3.2.1.3 Symbian 事件驅動程式設計模式

由於在 Symbian[13]系統中，其提供的框架（Framework）已經將底層的細節包裝成物件提供程式開發人員使用，如圖 3-4 所示

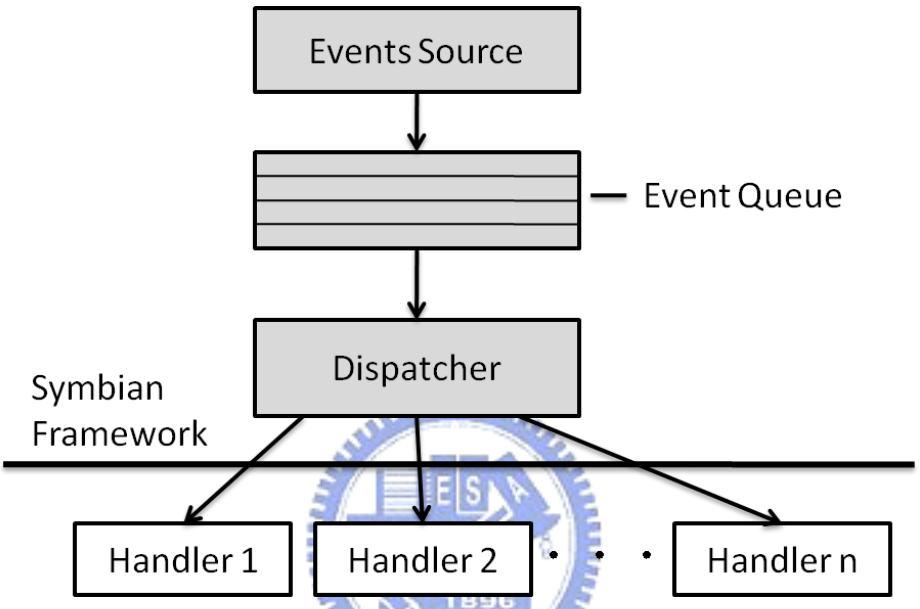


圖 3-4、Symbian 事件驅動程式設計模式

事件分配器已經由 Symbian Framework 所負責，因此程式開發人員可在 Symbian Framwork 所提供的函式中，實作對應的程式碼。關於更詳細的 Symbian 事件驅動模式，請參照同實驗室學弟高偉傑的論文[18]。

3.2.1.4 CCPK 事件驅動程式設計模式

依據事件驅動程式設計模式，Windows CE 函式分類的功能以及 Symbian Framework 的方式，本論文提出 CCPK 事件驅動程式設計模

式，如圖 3-5 所示，我們利用應用程式模組來處理訊息迴圈，並負責聯繫其他模組。而利用圖形使用者介面模組做為分配器的角色，使得特定的事件，例如使用者按下按鍵、在觸控螢幕上點選或者是要求程式重畫畫面等。另外新增網路模組來處理網路功能，並且設計此網路模組使用事件驅動的模式（fnOnRecv()）來通知程式資料已從網路另一端接收完成。圖 3-5 中白色的部分即代表程式開發人員僅需在 CCPK Framework 中所提供的虛擬函式（Virtual Functions）[14]，如 fnOnKey()、fnOnDraw()等中撰寫自己的程式碼即可收到這些系統的事件，而不需了解各個平台底層不同的事件驅動模式。

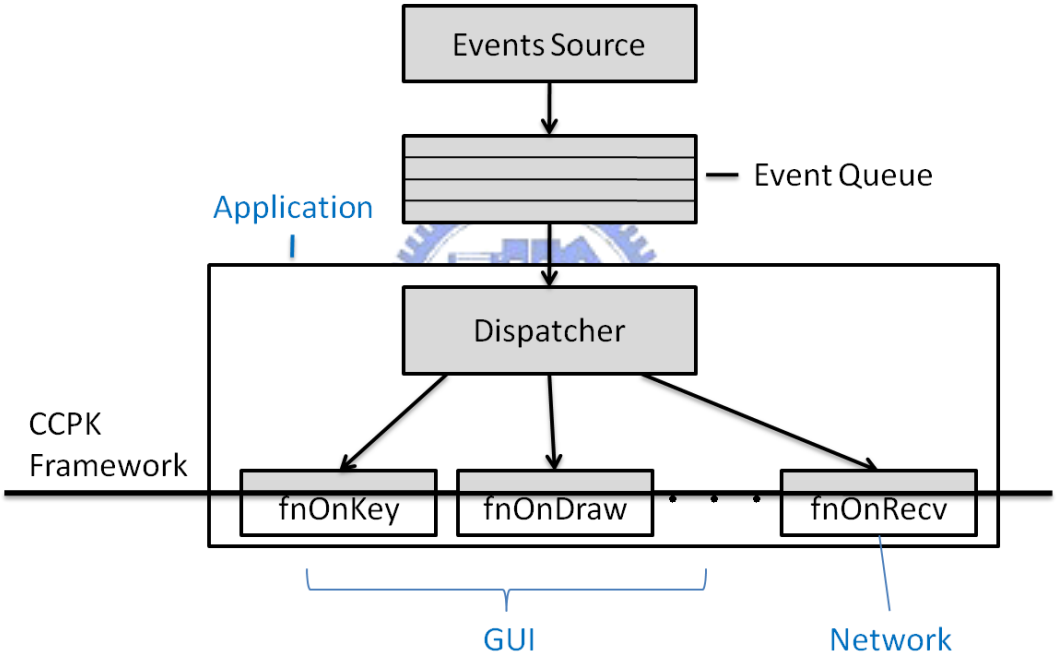


圖 3-5、CCPK 事件驅動程式設計模式

3.2.2 CCPK 系統模組設計（Design of CCPK）

在 CCPK 中，我們設計三個模組: 應用程式模組，圖形使用者介面模組以及網路模組，將底層的平台差異性隱藏起來。透過這些模組

統一的開發介面，使用者不需要知道底層對應的作業系統是哪一種，即可使用同一套程式碼，而運行在不同的作業系統上。

圖 3-6 說明上述三個模組的關聯性。

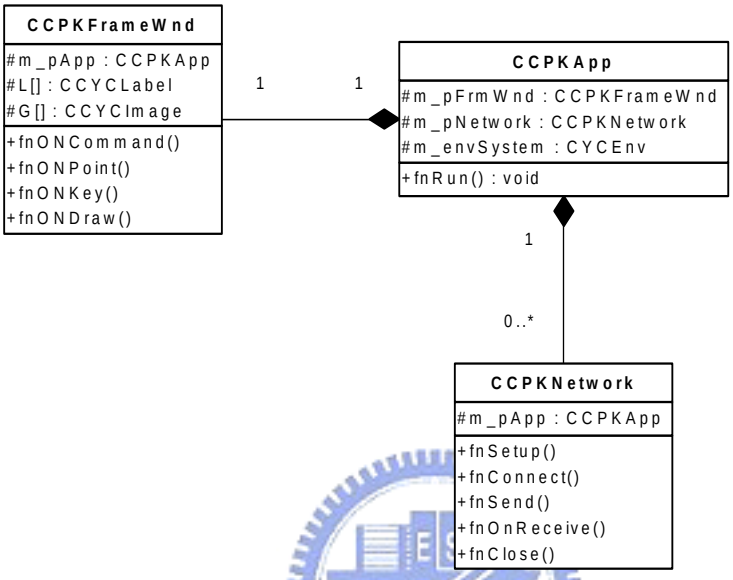


圖 3-6、CCPK 系統模組關聯圖

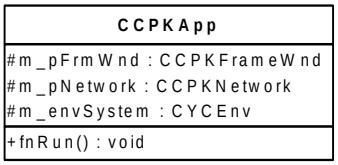
應用程式模組內包含的 classes 有： CCPKApp 以及 CYCEnv，詳見 3.2.2.1 節。圖形使用者介面模組內包含的 classes 有: CCPKFrameWnd、CYCLabel、CYCButton、CYCGraphicContext 以及 CYCImage，詳見 3.2.2.2 節。網路模組則擁有 CCPKNetwork 做為基礎類別 (base class)，並衍生出兩個 classes：CCPKSocket 以及 CCPKBluetooth，詳見 3.2.2.3 節。

3.2.2.1 應用程式模組 (Application Module)

此模組為程式的主控中心，亦即整個 CCPK 的核心模組，並做為與其他模組的溝通橋樑。主要的功能如下:

- 設定主框架視窗，使得程式開發人員得以看到程式的主畫面。
- 設定網路模組，使得其他模組可以藉由它來使用網路功能。
- 建立模組間的關聯性，使得各模組可透過應用程式模組來達成彼此溝通的目的。
- 提供系統相關的資訊，如記憶體使用狀況，螢幕大小等。
- 設定各平台特有的初始環境，如網路系統環境初始化。

此模組主要的 class 為 CCPKApp，主要的定義如下:



其中一個重要的虛擬函式 fnRun()，程式開發人員可覆載(Override) 此函式，此函式除了本身就是 CCPK Framework 的程式入口外，程式開發人員也可在此函式中加入想要額外的初始程式碼。另外尚有一個物件包含在此模組-CYCEnv，它即為負責系統相關資訊以及設定各平台特有的初始環境功能的 class，以 Windows CE 為例，要啟用 COM 或是網路功能前，都須透過 CYCEnv 事先初始 COM 或是網路功能的系統環境，如此才能開始使用相關功能的系統 API。

3.2.2.2 圖形使用者介面模組 (GUI Module)

本節會介紹 GUI 元件以及 GUI 事件在 CCPK 中的定義，並介紹 GUI 模組中主要的兩類物件: 視窗元件以及繪圖元件。最後說明資源設定的問題。

- 圖形使用者介面元件與圖形使用者介面事件 (GUI Units and GUI Events):

圖 3-7 以 Windows CE 的環境展示 CCPK 中 GUI 元件的定義。其中 GUI 的元件有主框架視窗 (Frame Window)，文字區塊 (Label)，按鈕 (Button)，選單 (Menu)，影像 (Image)。

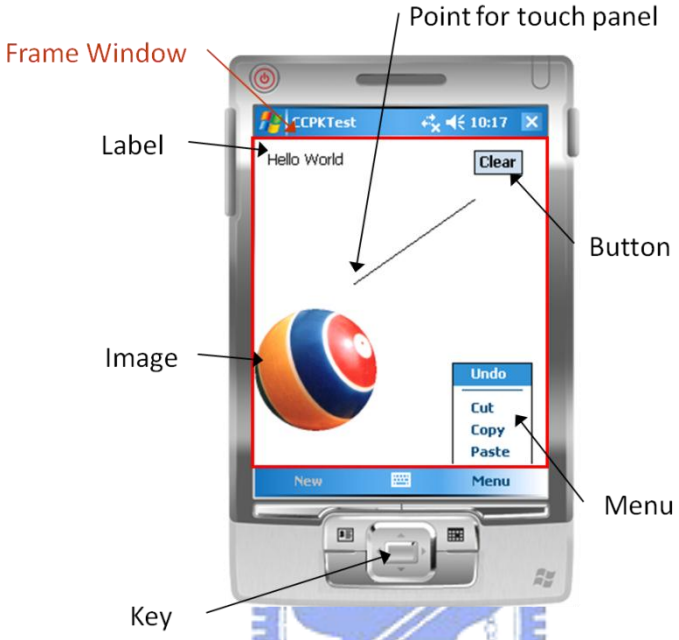


圖 3-7、GUI 元件展示 (Windows CE 平台)

而 CCPK 亦定義如下的 GUI 事件使得程式開發人員只需覆載(Override)CCPK 定義的虛擬函式(Virtual function)即可收到該 GUI 事件並且執行自己對應的功能。

- 按鍵事件 (Key Event): 使用者按下行動裝置上的硬體按鍵。
- 點擊事件 (Point Event): 使用者在點擊在觸控螢幕上。
- 按鈕/選單事件 (Button/Menu Event): 使用者選取了按鈕或是選單。在 CCPK 中，將此兩種事件統稱為指令事件 (Command Events)。
- 繪圖事件 (Draw Event): 當系統得知此程式有部分或全部區域需要重畫，便會通知程式此事件。它可能由系統主動發出，如有其他程式遮住此程式部分區域然後又恢復，也可能

程式本身主動發出需要重畫的需求。

◆ 視窗元件 (Widgets):

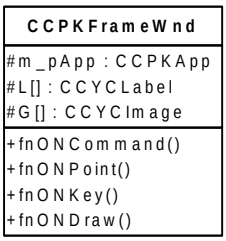
下面主要介紹在 GUI 模組中的視窗元件種類,提供的三個視窗元件: 主框架視窗、按鈕以及文字區塊。

■ 主框架視窗 (Frame Window): 負責程式主畫面的表現，並且提供事件處理器 (Event Handler)的虛擬函式，提供程式開發人員覆載 (Override)，主要的虛擬函式如下:

- fnONCommand(): 當使用者選取選單或者是點擊程式中的按鈕物件時，CCPK 會將此事件轉成呼叫此函式，並且傳入代表該選單或是按鈕的識別代號。
- fnONPoint(): 當使用者在觸控螢幕上點擊主框架視窗時，CCPK 會將此事件轉成呼叫此函式，並且傳入點擊的座標點以及該事件是由短壓螢幕或是長壓螢幕所造成。在 Windows CE 的裝置上，短壓螢幕等同在 PC 上按下滑鼠左鍵，長壓螢幕等同在 PC 上按下滑鼠右鍵。而上述的座標點是相對於主框架視窗左上角座標點為(0, 0)的相對座標。
- fnONKey(): 當使用者按下行動裝置上的硬體按鍵時,如方向鍵或是 Enter 鍵等，CCPK 會將此事件轉成呼叫此函式，並且傳入代表該按鍵的識別代號以及使用者目前是按下或者是放開該按鍵。
- fnONDraw(): 觸發此函式主要有兩類，第一類是屬於系統主動發出,其發生的時機可能為 (1) 當程式第一次要展現的時候,(2) 有其他視窗元件遮住主框架再移開後；第二類是程式開發人員希望程式重新將畫面重畫時，呼叫CCPK所提供的函式fnDrawNow()。在上述事件發生時，CCPK 會將這些事件轉成呼叫此函式，並且傳入畫布元件

(在 CCPK 實作中為 CYCGraphicContext 物件),提供程式開發人員在此畫布上繪製圖案。

CCPK 實作 class 為 CCPKFrameWnd，主要定義如下:



- 按鈕 (Button): 提供程式開發人員在指定的視窗位置上放置按鈕元件，CCPK 實作 class 為 CYCButton。欲產生此元件時，程式開發人員可呼叫其函式 fnCreate()，指定其欲放置的位置以及該按鈕的識別代號，則當使用者按下此按鈕時，CCPK 就會呼叫上述的 fnONCommand()虛擬函式。

- 文字區塊 (Label): 提供程式開發人員在指定的視窗位置放置顯示文字，CCPK 實作 class 為 CYCLabel。欲產生此元件時，程式開發人員可呼叫其函式 fnCreate()，指定其欲放置的位置，並且透過其函式 fnSetCaption()改變顯示的文字內容。

透過 CCPK 提供的上述基本視窗元件，程式開發人員便可方便的產生主程式視窗以及使用按鈕及文字區塊與使用者互動。

◆ 繪圖元件 (Graphics):

下面主要介紹在 GUI 模組中的繪圖元件種類,提供的兩個繪圖元件: 畫布以及影像。

- 畫布 (Graphic Context): 此元件代表著與硬體螢幕溝通的

主要管道，CCPK 實作此元件，使得各平台底層負責與螢幕溝通的物件成為統一的介面，在 CCPK 中實作的 class 為 CYCGraphicContext。主要功能如下:

- 連結螢幕顯示: 程式開發人員在此元件上進行的繪圖動作會展現在硬體的螢幕上。
- 具有基本的繪圖能力: 提供畫點(fnSetPixel())、畫線(fnDrawLineTo())以及區塊色彩填充(fnDrawRect)的函式。使用者也可設定使用畫筆(Pen)以及筆刷(Brush)的函式在上述的繪圖函式中使用。
- 展現影像內容: 透過其函式 fnBitBlt()，可以將指定的影像元件繪於指定的座標點。
- 支援雙緩衝區(Double Buffering): 當螢幕繪製時，一連串的繪圖動作依序完成時，並直接反應在硬體螢幕上，容易產生畫面閃爍的問題。CCPK 實作了雙緩衝區的功能，利用另一個繪圖 buffer (Offscreen buffer)，所有的繪圖動作都作用在此 offscreen buffer，當一連串的繪圖動作完成後，再一舉反映在硬體螢幕上。

■ 影像 (Image): 主要功能為讀取影像檔，CCPK 實作的 class 為 CYCImage，利用其函式 fnCreate()指定載入的影像檔案並進行解碼 (decode)後，藉由畫布將其影像內容表現出來。目前 CCPK 在 Windows CE 平台上支援 BMP，PNG 以及 JPG 三種影像檔格式，在 Symbian 平台上支援 BMP 影像檔案格式。

在使用 CYCGraphicContext 時必須注意的是，CCPK 透過 fnONDraw()將此元件的指標 (Pointer)傳入時，使用者應當只使用該指標，而不該記住該指標，留待其他事件時使用。因為此指標為每次 Draw 事件發生時，CCPK 會產生該次 Draw 事件的 CYCGraphicContext 指標，因此當 fnONDraw()函式呼叫結束後，該指標即失效。

◆ 資源 (Resources):

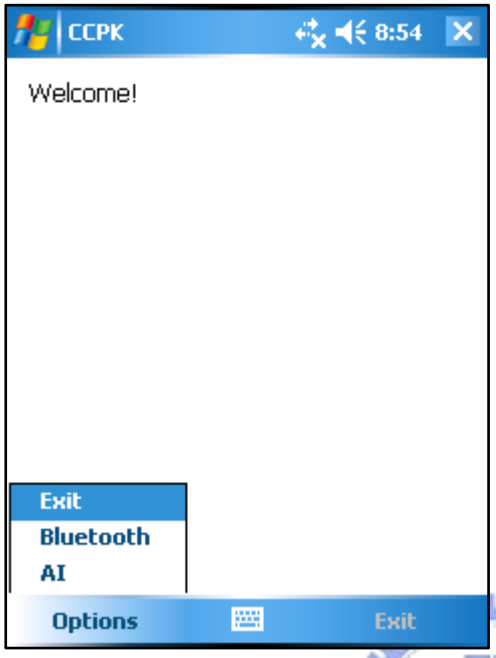
程式資源依不同平台及不同方法,通常在編譯時可合併於主程式執行檔中或者獨立為單獨的資源檔 (在 Windows CE 中，此單獨的資源檔副檔名通常為 dll 或者 mui，在 Symbian 中則分成副檔名為 rsc、aif 以及 mbm)。

由於程式資源是由各平台編譯器 (Compiler)依據本身特定的資源格式的資源檔定義編譯而成，因此 CCPK 在程式資源提出的解決方案就是提供一資源轉換工具,在此工具中編輯如選單或是按鈕的資源，然後令其轉出對應平台所需的資源檔。程式開發人員即可使用此資源檔定義交由該平台編譯器進行編譯。在 Windows CE 中，Resource.h 定義了每個資源的代碼，而 Data.rc 則包含了該資源的使用定義及展示字串[2]。在 Symbian 中，Resource.hrh 定義了每個資源的代碼，而在 Data.rss 中定義了該資源的使用定義,在 Data.loc 中定義其對應的展示字串[8]。圖 3-8 即為此轉換工具的操作畫面。

圖 3-8、資源轉換工具

如圖 3-9 所示，在 Windows CE 平台中，資源檔定義於

resource.h 以及.rc 檔案中。




```
(In Resouce.h file)
#define CmdExit    40019
#define CmdBT      40020
#define CmdAI      40021
```

```
(In CCPK.rc file)
IDR_MENU MENU
BEGIN
    POPUP "Options"
    BEGIN
        MENUITEM "Exit", CmdExit
        MENUITEM "Bluetooth", CmdBT
        MENUITEM "AI", CmdAI
    END
END
```

圖 3-9、選單資源定義（Windows CE 平台）

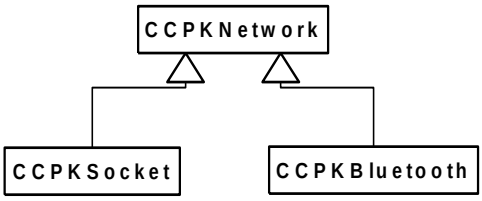
3.2.2.3 網路模組（Network Module）

本節說明網路模組的特性，以及提供簡單的函式讓程式開發人員簡單就能使用網路功能，並且說明此網路模組採用事件驅動模式與程式本身溝通，最後說明在 Windows CE 平台中如何達成事件驅動模式。

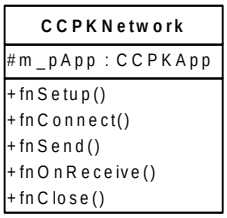
CCPK 網路模組設計有下列主要特點：

- 封裝底層網路複雜的實作，提供程式開發人員簡便的函式就能操作網路功能。
- 利用事件驅動模式，通知程式已經收到對方所送來的資料，避免程式阻塞（block）於等待對方的資料函式。

- CCPK 支援 TCP/IP 以及 Bluetooth，CCPK 實作的 class 為 CCPKNetwork，並且衍生出處理 TCP/IP 與 Bluetooth 的兩個 classes: CCPKSocket 與 CCPKBluetooth。



CCPKNetwork 主要的定義如下:



CCPKNetwork 提供的函式，使得程式開發人員能夠簡便的使用網路的功能而不需處理複雜的網路函式，也不需了解各平台底層的差異性。下列詳述各個函式的功能:

- fnSetup(): 負責網路連接的設定，此函式為使用網路模組時，第一個呼叫的函式。對應的 class 為 CCPKSocket 時，此函式會跳出對話框讓使用者可以填入欲連往的 IP address 以及 port number。對應的 class 若為 CCPKBluetooth 時，此函示會跳出對話框讓使用者選擇做為 serve 端或是 client 端，若選擇 client 端，則搜尋鄰近的 Bluetooth 裝置並讓使用者選擇欲連結的對象。
- fnConnect(): 執行網路連接，此函式為使用網路模組時，第二個呼叫的函式。依據上述的 fnSetup()的設定結果，開始執行網路功能，若為 server 端則等待 client 端的連入，若為 client 端則直接與 server 端連結。

- fnSend(char *pBuf, int nLen): 進行資料傳遞-送出。當上述 fnConnect()完成後，程式便可透過此函式發送資料給對方。
- fnOnReceive(char* pBuf, int nLen): 通知資料傳遞-接收的事件。此為 CCPK 將網路資料接收設計為事件驅動模式的函式，程式開發人員可覆載（Override）此函式以便接收已從網路接收下來的資料。詳細的事件驅動設計在下面的段落會再說明。
- fnClose(): 結束網路連線。當成是要中斷此網路連線時，呼叫此函式，CCPK 會將此連線結束並釋放相關的資源。

圖 3-10 說明 CCPK 所設計的網路模組事件驅動模式。當網路模組建立連線後，它會呼叫 recv()等待另一端傳送資料過來。一旦網路模組收到對方的資料時，它會觸發內部的事件，轉成事件驅動模式，再透過虛擬函式 fnOnReceive()通知程式取得對方的資料。

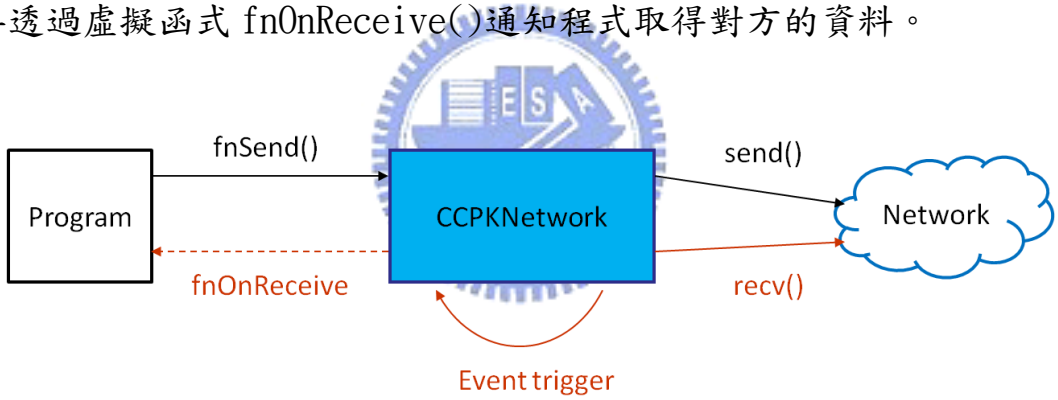


圖 3-10、網路模組事件驅動模式

在 Windows 桌上型作業系統，它提供一組非同步（Asynchronous）的 socket API，可以讓程式開發人員透過事件處理器（Event Handler）方便的處理 socket 資料。然而微軟為了節省系統程式的大小，因此在 Windows CE 平台上，取消了非同步的 socket API，也就是說在 Windows CE 上，不支援 WSAAsyncSelect()[10]之類的 API，使得沒有可用的事件處理器用來處理 recv()。

由於 Windows CE 上不支援非同步的 socket API[4]，只有單純

的 send()/recv()/select()之類的 API 可用，所以本論文提出一解決方案，如圖 3-11 所示，在網路模組內新增一個事件接收者(Receiver)，並且產生一個監控執行緒 (Monitor thread)。當網路模組與對方網路連接上時，監控執行緒便會呼叫 select()，等待有等待讀取資料的事件產生。當收到有等待讀取資料的事件發生時，監控執行緒便會透過私有的事件通知私有的事件接收者。事件接收者收到此私有事件時，便呼叫 recv()讀取資料，讀取完成後，透過虛擬函式 fnOnReceive()通知程式取得的資料內容。

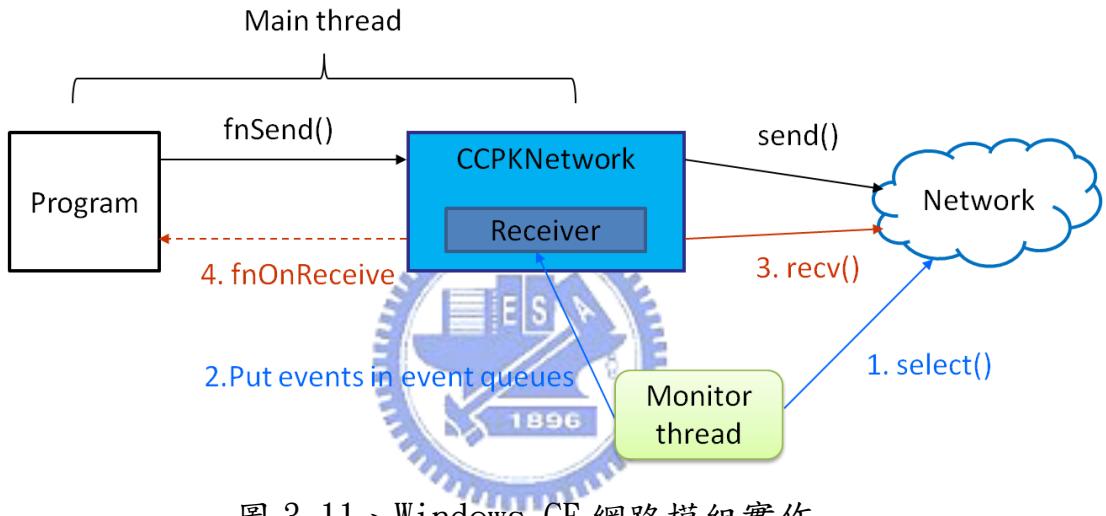


圖 3-11、Windows CE 網路模組實作

在上述的 Windows CE 模組實作中，透過事件驅動的方式，程式與 CCPKNetwork 皆處於同一個執行緒，避免程式開發人員需要自行撰寫多執行緒的程式，減低其開發網路程式的複雜度。

3.3 獨立遊戲層設計 (Design of Platform Independent Layer)

本論文希望透過對靜態遊戲（如棋類、撲克牌等）的分析，建立

起一套標準流程，從而縮減程式開發人員開發此類遊戲的開發時程。本節會剖析遊戲進行的基本流程，接著再討論 CCPK 對於此流程所加入的遊戲規則模組 (Rule Module)與遊戲狀態模組(State Module)。關於此節的詳細論述，請參照同實驗室學弟高偉傑的論文[18]。

3.3.1 遊戲進行流程

如圖 3-12 所示，我們分析遊戲進行的流程可分為四個主要階段:

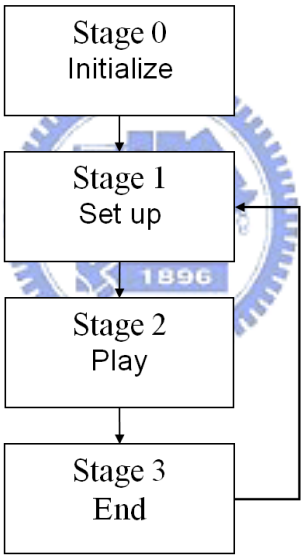


圖 3-12、遊戲進行流程圖

- 階段0: 程式初始化，此時產生的元件有應用程式、主框架視窗、遊戲規則以及遊戲狀態模組，並將上述模組進行初始化的動作。接著取得系統資訊 (如螢幕大小，記憶體大小)以及展現主框架視窗。

圖 3-13、遊戲流程-階段 0 畫面

◆ 階段 1: 設定遊戲玩法，並呈現程式初始畫面（如歡迎畫面），並讓使用者選擇對戰方式（如網路對戰模式或是單機模式與人工智慧對戰）。

圖 3-14、遊戲流程-階段 1 畫面

◆ 階段 2: 對戰階段，依照遊戲規則，雙方輪流出牌或下子，當尚未輪到自己出牌或是下子時，將己方的出牌或下子功能封鎖。

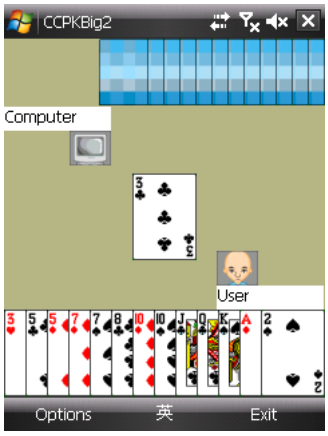


圖 3-15、遊戲流程-階段 2 畫面

◆ 階段 3: 依據遊戲規則，判斷勝負。確認目前遊戲局結束後，依據使用者的選擇結束程式或者重新遊戲。

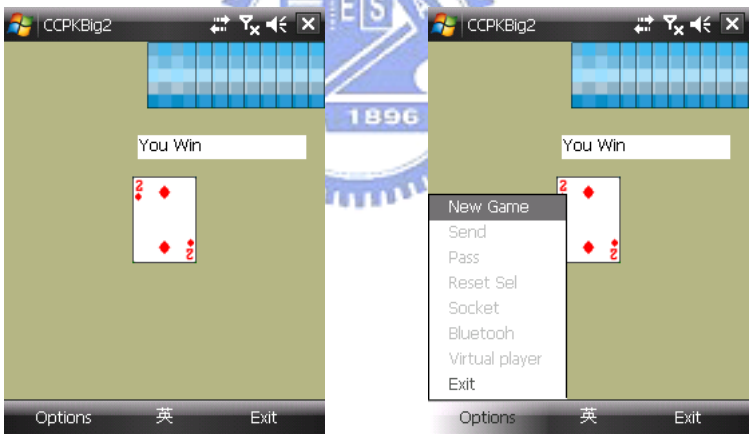


圖 3-16、遊戲流程-階段 3 畫面

3.3.2 遊戲模組

依據上述的遊戲階段分析，本論文新增兩個模組來負責遊戲規則

以及遊戲資料。

- 遊戲規則模組 (Rule Module): 負責判斷出牌或下子是否合法，確認遊戲局是否開結束以及決定何者勝利。
- 遊戲資料模組 (State Module): 提供目前遊戲處於哪一個階段，目前的對戰方式，遊戲進行的暫存資料以及玩家的資料。

如圖 3-12 所示,CCPK 新增兩個 classes: CCPKRule 及 CCPKState 來負責遊戲規則及遊戲資料的功能。

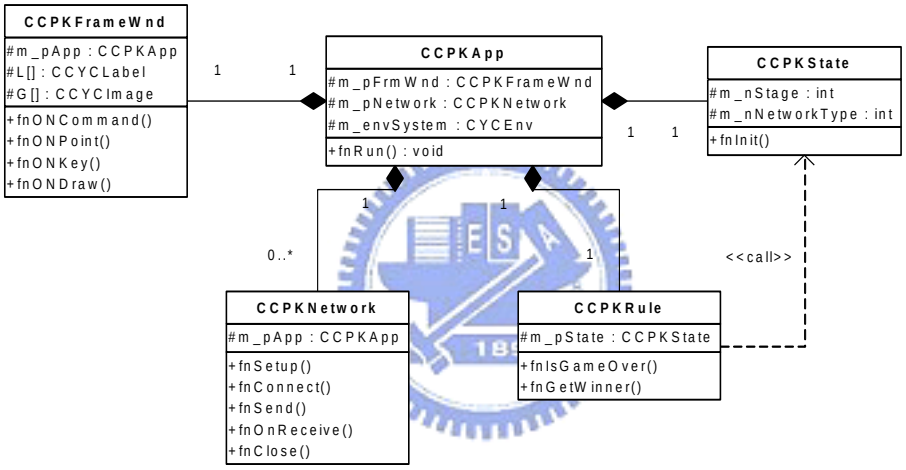


圖 3-12、CCPK 遊戲模組關聯圖

獨立遊戲層除了新增上述兩的新模組外，也針對應用程式模組及主框架視窗對遊戲的整合功能與以強化，增強的功能說明如下:

- 應用程式模組: 實作 GameApp class，其繼承自 CCPKApp，並設計了 Core Method，使得選牌或是下子動作會呼叫此函式，藉此與 Rule 以及 State 模組確認動作的合法性以及遊戲的階段流程控管。此外，由於此模組也是程式核心，因此本論文的設計讓網路端的資料直接送到此模組，以便後續處理。為了達成此目的，設計了一個函式 RecvMessage()，提供網路模組在 fnOnReceive() 函式中直接呼叫。

- 圖形使用者介面模組: 實作 GameFrameWnd class，其繼承自 CCPKFrameWnd。提供 Show 函式用來表現不同階段的畫面表現。此外，在選單中已經有固定的項目，如 New Game、Socket 以及 Bluetooth 等，此模組也提供如 OnNewGame()、OnSocketConnect() 以及 OnBTConnect()的函式，讓使用者直接使用。

此外，針對單機人工智慧模式，也設計人工智慧模組 (AI Module) 並且利用網路模組與程式原先的互動方式，使得人工智慧模組可以直接繼承網路模組的介面函式，快速的整合在遊戲當中。圖 3-13 即說明人工智慧模組與網路模組的關係。

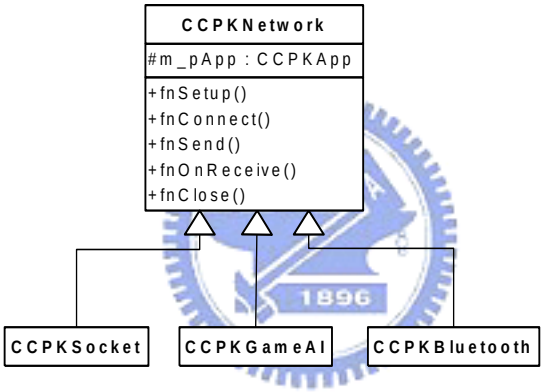


圖 3-13、人工智慧模組與網路模組關聯圖

第四章、 設計經驗與實作範例

本章將說明在跨平台程式設計時,所應用的程式介面設計技巧以及實作遊戲成果。

4.1 程式介面設計技巧

在跨平台設計程式介面時,首先遇到的就是各平台都有各自的資料結構及類別,如何統整這些差異性就是實作上的重點。本節將我們在實作時所採用的解決方式整理如下:

- C/C++基本型別: 由於我們採用的開發程式為 C/C++,因此盡量採用 C/C++的基本型別,如 char、wchar_t、int 以及 bool 等。如此可使程式開發人員操作已經熟悉的型別。

- 定義基本結構: 在程式發展中,一些常用的資料結構,如大小、矩形定義以及顏色等,我們自行定義其資料結構並命名為 CYCSIZE、CYCRECT 以及 CYCCOLOR 等 CCPK 所屬的資料結構,並在各平台的實作中利用 C++運算子 (Operator) 的功能,簡便的轉成各平台所對應的資料結構。

- 善用多型 (Polymorphism): 除了在 class 設計時採用多型機制,使得程式開發人員得以繼承 CCPK 內實作的 class 並加以覆載 (Override) 其虛擬函式。最重要的應用莫過於在主框架視窗中使用的虛擬函式,如 fnONKey()以及 fnONDraw()…等。它將各平台的事件驅動轉化成對應的虛擬函式,使得程式開發人員可以簡易的覆載 (Override) 這些事件處理器 (Event Handler) 的虛

擬函式，而不必了解各平台底層的事件驅動模式。

- 隱藏差異性: 在實作中常會遇到各平台的特定資料結構,但是在其它平台則沒有的狀況，或者是各平台都具有的底層物件，但是卻不需要暴露在 CCPK 的程式介面中。我們採用 C 語言的一個基本型態特性-void*，透過轉型為 void*，編譯器便可以接受各式不同的資料結構或 classes 的轉型，而 CCPK 在收到此 void 指標(Pointer) 時,在實作中即可自行轉型成各平台的特定資料結構或 classes 來使用。

- 視窗元件定義: 在所有 class 中,與各平台特性連結最緊密的就屬視窗元件，因為它常牽涉到很多各平台其它的標頭檔，不易與平台關聯切斷。因此對於視窗元件的實作，我們採取的是“is-a”的概念，也就是說 CCPK 的視窗元件是從各平台的特有基礎視窗元件繼承下來再予以實作成為 CCPK 的視窗元件。也因此 CCPK 的視窗元件標頭檔會區分為各平台的專屬的標頭檔,當然對於 CCPK 所設計的程式介面仍然保持一致。相對於其它的 CCPK 模組，其標頭檔則為共用的狀況。

- 封裝平台底層物件: 為了統一 CCPK 的物件介面，對於部分模組並沒有與各平台的其他標頭檔連結很緊密，我們採取“has-a”的概念，也就是說，實作上我們利用 CCPK 的 class 擁有該平台的物件當作 class 的 member data，而程式開發人員透過 CCPK 的 class 操作該物件。在 CYCGraphicContext、CCPKSocket 以及 CCPKBluetooth 中，我們便採用此技巧，如此可大幅縮短平台開發人員的開發時程。

4.2　成果

在開發 Windows CE 平台時，我們採用的是 Windows Mobile 5.0 Pocket PC SDK 所開發出來的應用程式。並執行於 HTC P3470 (CPU: OMAP850 201MHz，ROM 為 256MB，RAM 為 128MB，螢幕為 240x320 的觸控式螢幕，作業系統為 Windows Mobile 6 Professional，對應的 Windows CE 核心作業系統為 5.2 版)。此外亦可執行於 Mobile 5.0 SDK 所附屬的 PC 模擬器 (Windows XP，Intel 1.5GHz，1G RAM)，螢幕亦為 240x320。

第一個實作範例遊戲為兩人版大老二，遊戲規則由發完牌後，持有最小牌的人有出牌優先權，先把牌出完的人獲勝。第一張圖表示開始遊戲後，玩家可以選擇透過網路對戰或是單機版 AI 對玩，第二張圖則為兩人對戰的過程，第三張圖則是玩家勝利的畫面。

圖 4-1、兩人版大老二實作範例

第二個實作範例遊戲為六子棋，遊戲規則為黑子先下一子，之後便輪流每次下兩子，先六子連一線的人獲勝。第一張圖表示開始遊戲後，玩家可以選擇對戰的方式，第二張圖則為遊戲進行的畫面，第三張則是玩家失敗的畫面。此遊戲內容為實驗室學弟高偉傑所做，並將

程式移到 Windows CE 上編譯而成。

圖 4-2、六子棋實作範例

第五章、 結論與未來發展

CCPK 是本論文中所提出的 Framework，用來解決跨平台行動裝置遊戲開發時所面臨到的問題-需要投入較多的人力在熟悉各種平台的特性，以及如何縮短遊戲開發時程。

首先，我們設計了平台相依層，並在此層分成應用程式模組、圖形使用者介面模組以及網路模組。透過這三個基本模組，程式開發人員只須了解這些模組所提供的統一程式介面，並且使用它，即可快速的實作出跨平台的程式原型。而其過程中不需要理解不同平台間的系統特性，因為 CCPK 已經將不同平台間的差異性，藉由所提供的這些模組隱藏起來。

再者，對於遊戲流程的開發，我們提供一個獨立遊戲層，將遊戲流程予以統一規畫，程式開發人員只需依照設計流程以及對應的函式，便可快速的發展出相同類型的遊戲程式。

而在實際經驗上，我們也實作了六子棋與二人版大老二兩套遊戲來驗證本論文所提出的 Framework 可行，並透過實作時所得到的經驗來修正原始的設計概念，讓 CCPK 更具有發展的彈性以及良好的程式介面。

然而，在本論文的研究過程中，我們知道我們僅提出一個基本而可行的解決方案，但是還有許多項目可以更深入的研究，以改善以及加強此 Framework 的設計，並且能夠廣泛的被採納。

CCPK 所提供的函式庫，雖然並不如大型函式庫(如微軟的 MFC[6])那麼完整，但是已經將主要的模組建立出來，只要依據模組的概念，日後持續增加模組內的 class，如更多的 Widgets 元件，便可應付多樣化的遊戲開發。此外，日後平台開發人員可透過目前所定義的程式

介面，針對支援的平台予以擴增，如 iPhone 作業系統、Palm 作業系統等更能驗證本論文設計的相容性。

目前 CCPK 只提供了基本的 Socket 與 Bluetooth 的功能，除了可以橫向發展，提供更多的無線通訊功能，如紅外線等之外，也可以縱向發展，提供如 HTTP 或是手機上常用的 WAP 通訊協定，以利程式開發人員使用更多元化的通訊功能。

由於不同平台的會有不同的螢幕尺寸，因此針對不同尺寸的螢幕而動態調整遊戲的畫面可以增加遊戲的適應性。在未來的發展中可以針對這一點加以提高程式開發的便利性。

本論文中的實作範例皆屬於較為靜態的遊戲，對於即時動態遊戲而言，繪圖能力的提升更是必須的能力，除了使用 double buffering 可以改善螢幕快速繪圖時閃爍的問題，也可發展支援 OpenGL 或 DirectX 等更高速繪圖能力。

參考文獻

[1] Apple, "iPhone OS", available from http://developer.apple.com/iphone/, 2008.

[2] Charles Petzold, Programming Windows, 5th ed., Microsoft Press, Washington, 1999.

[3] Christopher Tavares, et al., ATL Internals: Work with ATL 8, 2nd ed., Addison Wesley, 2006.

[4] Douglas Boling, Programming Microsoft Windows CE .NET, 3rd ed., Microsoft Press, Washington, 2003.

[5] Jacco Bikker, "MobileCore: A Cross-Platform Framework For ARM-Base Mobile Games", 2003.

[6] Jeff Prosise, Programming Windows with MFC, 2nd ed., Microsoft Press, Washington, 1999.

[7] John K. Ousterhout, "Why Threads Are A Bad Idea (for most purposes)", Invited Talk at the 1996 USENIX Technical Conference , January 25, 1996.

[8] Leigh Edwards, et al., Developing Series 60 Applications, Addison Wesley, 2004.

[9] Linux, available from http://www.linux.com/, 2008.

[10] Microsoft, "MSDN: Microsoft Developer Network", available from http://msdn.microsoft.com/, 2008.

[11] Microsoft, "Windows Embedded", available from http://www.microsoft.com/taiwan/windows/embedded/default.mspx, 2008.

[12] Palm, "Palm OS", available from http://www.palm.com, 2008.

[13] SourceForge.net, available from http://sourceforge.net/, 2008.

[14] Stanley B. Lippman, Josée Lajoie, C++ Primer, 3rd ed., Addison Wesley, 1998.

[15] Symbian, "Symbian OS", available from http://www.symbian.com/, 2008.

[16] 武漢卓睿軟件有限公司，「嵌入式跨平台開發解決方案(ECDS-MUI)」，2007。

[17] 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，民國 88 年。

[18] 高偉傑，「跨平台手機遊戲開發框架在 Symbian 手機之研究」，國立交通大學資訊工程學系，碩士論文，民國 97 年。

附錄、平台相依層在 Windows CE 之說明

本章將介紹在 Windows CE 上的開發環境設定以及平台相依層在 Windows CE 上的函式庫架構,最後並用一範例來說明如何使用 CCPK。

1.1 開發環境

本論文在開發 Windows CE 的整合介面是使用 Microsoft Visual Studio 2005，而 Windows CE 的版本則為 5.0，其對應所需要的 SDK 則為 Windows Mobile 5.0 SDK for Pocket PC，並且上述的 SDK 也包含在 PC 上的模擬器。此外，由於是在開發 Windows Mobile 裝置程式，因此尚需安裝 Microsoft ActiveSync v4.5。不論是在模擬器或是在實體 Mobile 5.0 Pocket PC 裝置上進行程式開發，都在同一個整合介面上即可，實體 Mobile 5.0 Pocket PC 裝置只需使用 USB 線連結 PC 與裝置後，便可開始程式開發。

1.2 函式庫架構說明

我們將 CCPK 的實作包裝成 static library，檔名為 CCPKCore.lib，使用者透過 CCPKCore.lib 以及所提供的 CCPK class 的標頭檔就可開始撰寫程式,而 CCPK 的表頭檔列表也整合在 CYC.h。設計成 static library 的用意如下：

- 容易提供他人使用: static library 可以封裝內部的程式碼，使得在不需要提供原始碼的狀況下，讓程式開發人員得以使用

CCPK。而程式開發人員也只需標頭檔與 CCPKCore.lib 即可方便的使用 CCPK 的 Framework。

- 減少 DLL 的使用: 因為 Windows CE 5.0 的作業系統架構設計的關係[4],所以在虛擬記憶體上的配置需要減少程式所使用的DLL數量以避免過多的 DLL 推疊影響程式本身可使用的記憶體大小。
- 版本控制: 程式使用 static library 後，在發佈時便可單獨提供一個執行檔而不需要附帶相關的 DLL。反之，若 CCPK 提供的是 Dynamic-Link library，則發佈時必須要附帶相對應版本的 CCPK DLL，增加版本控制的負擔。

由於 CCPK 內包含 resource，例如網路模組使用對話窗讓使用者輸入 server 的 IP 位址或是找尋鄰近的 Bluetooth 裝置,再加上 CCPK 使用 static library 的方式，因此程式開發人員除了撰寫程式部分需要 include CYC.h 以及 import CCPKCore.lib 外，resource 部分尚需 include CCPK 的 resource 檔: CCPKCoreResource.h 以及 CCPKCore.rc。

1.3 使用範例

本節列出利用 CCPK 架構出一簡單的程式的步驟，依據這些步驟便可產生一個具有 GUI 介面的程式。

- 步驟一: 使用 Visual Studio 2005 產生一個 C++ Win32 Smart device project，使其具有 WinMain()的程式入口。並 include CCPK 的標頭檔 CYC.h 以及 import CCPKCore.lib。另外，在 resource file-xxx.rc 中 include CCPK 的 CCPKCoreResource.h 以及 CCPKCore.rc。

◆ 步驟二: 繼承 CCPK 的 Application 及 Frame Windows 的 classes 成為自己的 classes，並且可以 override 想要的處理的 event handler，如 fnONCommand()。範例程式碼如下所示。

```
class CMyApp : public CCPKApp
{
public:
      CMyApp();
      virtual ~CMyApp();
};

Class CMyFrmWnd : public CCPKFrameWnd
{
public:
      CMyFrmWnd();
      virtual ~CMyFrmWnd();

      //--- override CCPKFrameWnd event handler function
      virtual bool fnONCommand(int nID);
};
```

◆ 步驟三: 在 WinMain()函式中將自己的 Application 及 Frame Window class 產生出來，並且建立關聯，最後呼叫 Application class 的 fnRun()即可讓程式展現 GUI 介面，並且成為事件驅動模式，等待使用者互動。範例程式碼如下所示。

```
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst,
LPTSTR pCmd, int iCmdShow)
{
	// prepare CCPK major component-Application
	CMyApp *pApp = new CMyApp ();
	if (!pApp) return -1;

	pApp->m_envSystem.fnSetInstance(&hInst);

	// prepare CCPK major component,-Frame Window
	CMyFrmWnd *pMainFrm = new CMyFrmWnd ();
	if (!pMainFrm) return -1;
	if (!pMainFrm->fnCreate(L"CCPKBig2", IDR_MENU)) {
		return -1;
	}

	// set relation of CCPK component
	pApp->fnSetFrmWnd(pMainFrm);
	pMainFrm->fnSetApp(pApp);

	// start program in event-driven mode
	pApp->fnRun();

	// after program terminated, release resources
	delete pApp;
	delete pMainFrm;
	return 0;
}
```

IDR_MENU 爲 resource 中定義的 menu 代碼

依據上述的三個步驟，程式開發人員便可快速的產生基本的程式框架，再依據本身想要處理的 event handler，將對應的虛擬函式 override，便可打造屬於自己的程式。