使用 C++在 Linux 上實作
事件驅動式之文字介面視窗類別庫
Construct a GUI class library with Event-driven method by C++ in Linux console mode

馮振祥 王壘 柏小松
逢甲大學電機工程系 逢甲大學電機工程系 逢甲大學電機工程系
feng0020@pchome.com.tw lwang@pine.iecs.fcu.edu.tw ssbor@fcu.edu.tw

摘要

在軟體系統越趨複雜的今日，GUI (Graphical User Interface, 圖形使用者介面)對使用者而言雖較為親切，但對程式設計師而言，則與欲解決之問題不甚相關，反而成為程式設計上的另一負擔。因而在本論文中，作者即利用事件驅動的方法，M-V-C (Model-View-Controller)三層式架構，實際建構了一組類別函式庫，以支援具有分時多工的菜單(Menu Bar)狀態列(Status Line)等介面設計，以提供一般應用程式設計之使用。

由於本論文中所提出之函式庫乃以 C++實作，因而其 OO 特性在擴充性可預留較大的空間，可用繼承的方式再建立更切合需求的功能。而在 Linux 和 Unix 的平台上，只要有支援 "ncurses" 函式庫和相容 "POSIX 1003.1c"標準，皆可直接運用本函式庫。

關鍵詞: 圖形使用者介面, 類別庫, 事件驅動, M-V-C, OOP

GUI (Graphical User Interface) is a friendly interface for users to face the complicate application program. However, it is also an overhead to application programmer. This overhead is not the critical part of application but an added penalty for programmers.

In this paper, we use the event-driven method, M-V-C (Model-View-Controller) structure of classes to construct a class library with time-division multiplexed property. The functions include Menu Bar, Status Line, text Viewer, etc..

This class library is implemented by using OOP (Object-Oriented Programming) method with C++. The OOP method creates a flexible space for extension by the feature of inheritance. The library will work under the environment of Linux or Unix OS with "ncurses" or "POSIX 1003.1c" compatible standards.

Keyword: GUI, Class Library, Event-driven, M-V-C, OOP

一、緒論

現在的作業系統已經是"圖形化"的時代，用視覺化的方式讓使用者更容易上手是極為重要的課題;但在 Linux 上想要使用 X Windows，對程式開發者而言仍十分困難。首先，Xwindows 程式設計甚為難學。再者，有時程式也不會用到那麼強大的功能，只是簡單的幾個視窗要用，甚至根本就只是打算在文字模式下處理而已。

若在文字模式下想要有視覺的效果，一般的情況只能"因陋就簡"的單字符號去拼湊。否則即需借助借助市面上已有的函式庫來代為解決問題。然而，在 Linux 環境上由於商業軟體較少，較普遍的 ncurses 函式庫又因是 C 的函式庫，因而不能用繼承等物件技術以擴充功能，而且實際上並沒有提供視窗管理的功能，導致在 Linux 環境下開發視窗應用的困難。

有鑑於上述的問題，因而本論文即提出一套類別庫，以提供 Linux 程式開發者輕易的達到以下的設計:

1.在文字模式下執行，而且支援終端機。

2.使用視窗當做介面。

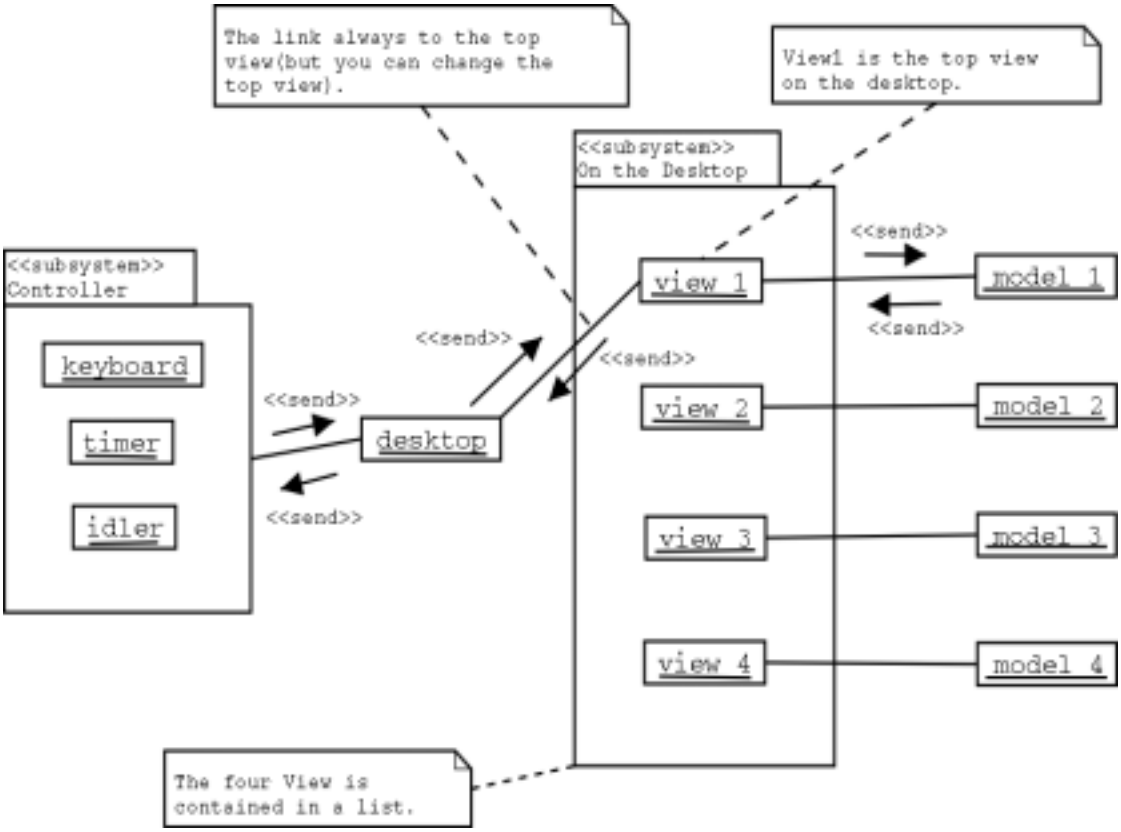
3.快速就能建構出一個小型的系統。

4.能夠用 OOP 方便的擴充功能。

由於本論文所提出之類別庫是採事件驅動方法，以 M-V-C(Model-View-Controller)三層式架構設計，故首先介紹原理及架構如下:

1.1. 事件驅動式程式設計方法

事件驅動式設計方法，是一種程式間互相溝通的程式設計方法。把程式間各種相關聯的操作(如一個視窗)，或實際存在的硬體



圖一: M-V-C 事件傳遞架構解說圖

(如鍵盤)，分成一個個邏輯上獨立的個體。而每個個體又包裝著一組事先定義好的溝通介面(事件傳遞的介面)。如此一來，只要各個個體都確實的實現介面，個體間就可以依照定義的方法互相溝通(傳遞事件)。接收訊息的一端可以經由傳遞過來資訊(事件)得知傳送端的狀態，或執行所要求的功能。只要依照此種溝通方法，就可以很容易的寫出一個公用的管理程式(main program model)，統籌規劃系統的運作。

由於是〝驅動式〞，代表個體是處於〝主動〞的狀態，相對於其他程式的被動。舉例而言:當管理鍵盤的個體發現有字元輸入時，會〝啟動〞一個鍵盤事件，傳遞至公用管理程式，再由其他的預設函式處理之。

1.2. M-V-C 程式架構

MVC 的程式架構是源自於 Smalltalk 語言的架構，原本的用途就是用來建立使用者介面，不過也可以用來組成其他用途的程式。MVC 是由三種物件〔Object〕組成。

1. Model 應用程式模組:

應用程式模組(Model)與表現層(View)鏈結，負責所有應用程式的行為。

2. View 表現層:

表現層(View)管理 Model 在螢幕上的輸出，就是螢幕上的小視窗。它接受代理物件(Controller)的事件訊息。如果自己能處理(如游標等)，就把先行處理掉。如不屬於自己的管轄範圍，傳給與之鏈結的模組(Model)。

3. Controller 代理物件層:

代理物件層(Controller)管理所有的事件產生器。例如:鍵盤(keyboard)，時間中斷(timer)，自定的空閒事件產生器(idler)。 它負責產生事件，並將之傳給視界。系統中所有的事件都由它開始發出。

根據上述之 MVC 架構，本論文所建立之程式架構即如圖一所示，其中 Desktop 也是一種 View，負責管理其它所有 View(子 View)，主程式模組和它連結。所以，事件傳遞的規則是: 所有代理物件(Controller)的事件訊息都先傳到 Desktop，然後再傳到子 View，子 View 再傳到所鏈結的模組。意思也就是:代理物件層(Controller)是整個系統架構的**唯一**主動部份，而表現層(View)和應用程式模組(Model)都只是消極的等待接收事件並做出回應而已。

由以上可知，MVC 的架構可以類別清楚的分割，事件傳遞的介面也可用繼承達到統一的標準。C++有支援物件導向，且兼顧執行速

圖二: DEMO 程式執行結果

度，因此本論文以 C++設計該類別庫。

二、系統架構

本論文所欲建構的介面視窗類別庫，將可產生如前所述之 MVC 系統架構，如圖一所示，系統中以 Controller 為公用管理程式，管理所有的事件產生器。其中 run( )函式是整個系統的引擎，之中呼叫所有事件產生器的 pool( )函式，將新產生的事件放入事件串列 msgQueue，等待傳送。

系統中將建有三個事件產生器：Timer 可產生固定時間事件(evTimer)，Idler 在系統空閒時產生事件(evIdler)，Keyboard 在按下鍵盤時產生事件(evKeyDown)。遵循父類別 ContlItem 所宣告的介面，由 pool( )和 Controller 溝通。

Model 應用程式模組。以父類別 Model 來提供一組介面，主要供程式設計者繼承，來實現自己所需的應用程式。以函式 fromView( )和所屬的表現層溝通。

View 表現層是程式對外的窗戶，需和程式設計者所提供的 Model 鏈結，事件需經由 View 而傳入 Model。以 fromController( )和桌面溝通，事件經由 dispatch( )傳遞至鏈結的模組。

桌面是 View 的子類別，所有的 View 用 viewList 管理。包含一組虛擬螢幕 VScreen。程式中大部分的輸出是寫入記憶體中的虛擬螢幕，只有少部分的輸出因為速度的關係，直接轉往終端機。故桌面就是整個系統的 View。

由於桌面要管理所有的 View，所以擁有許多對 View 的操作。程式設計者需在與桌面鏈結的主程式(Main program model)中自訂各種發生的狀況。

至此，MVC 的架構已經完全。以類別繼承提供各自的介面，減少互相間相依的程度，增進模組化，系統流程的複雜度降低。而且使用者只要依照介面來增加新的模組，可以在幾乎不改變之前程式碼的情況下，讓系統馬上開始運作。符合軟體 IC 的精神。

程式中所有輸出的部分，由類別 PObject 定出了一組介面。內定有兩個子類別，Screen 預設實體螢幕，VScreen 預設虛擬螢幕。如果程式中只使用介面中所宣告的操作，Screen 和 VScreen 就可互相交換使用(多型 polymorphism)。所以如有需要，可以由 PObject 衍生出新的類別 Printer，將某一個 View 的輸出或整個程式的輸出，直接轉往列表機印出。或者是將兩個不同的 View 分別送往不同的終端機顯示，每一個 View 的輸出可以在程式執行中動態決定。

整個系統將建立於 ncurses 函式庫的最低階操作，藉由 ncurses 來當作和 Linux 基本輸出入的介面。由類別 Terminal 重新包裝出一組較高階的介面，捨去了瑣碎的細部設定。

對字元的處理上，將螢幕上的每個像素(字元)用類別 CharacterUnit 抽象化，每個字元都紀錄自己的顏色，屬性。

三、文字介面視窗類別庫

類別庫已全部設計完成，有關類別庫的

使用方法已於檔案“ test_m_v_c.cc ”中以事先設計的 demo 程式表達(如上頁圖二所示)。有關的初值參數，常數，都放在標頭檔”type.h”裡。至於 Makefile，預設編譯器是用 gcc，但可自行修改 Makefile 檔

程式預設安裝於目錄 /usr/local/bin 下，編譯完後會自動產生函式庫 libwin.a 。

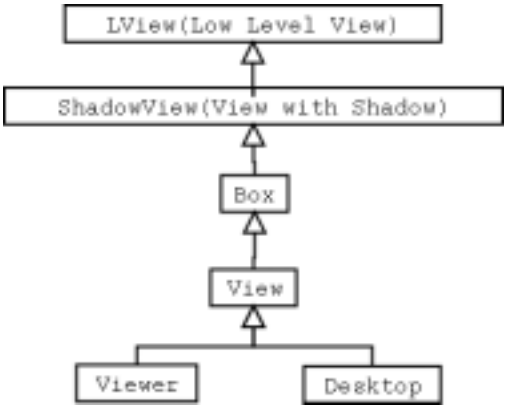
3.1. 類別說明

3.1.1 與 Model 層有關的類別:

<< Class Model >>

Model 類別，應用程式和系統之間溝通的介面在此宣告。主要是讓使用者繼承使用。

3.1.2 與 View 層有關的類別，以圖三說明:



圖三: View 家族的階層關係

<< Class LView >>

View 家族的最原始類別，只代表一塊有顏色的矩形。

<< Class ShadowView >>

在 Lview 旁邊加上陰影。由於 Linux 的預設終端機不支援'dim'(亮度減半)屬性，所以使用並無實際效果。

<< Class Box >>

Box 是 Lview(低階 View) + shadow(陰影) + frame(邊框)。在 M-V-C 的架構下通常不會用到這個，因為它不能和 Model 鍊結(View 以後才行)。但在一般的程式中可以有其他用途，可以用它隨時呼叫出一個訊息視窗出來。

<< Class View >>

View 特別的地方，就是開始可以和 Model 鏈結，其他的父類別都還不可。

<< Class Viewer >>

一個簡易的文字流覽器。

<< Class DeskTop >>

一種特殊化的 View，負責管理其它所有的 View，在訊息傳遞中佔有重要的地位。Desktop 的主要不同處，就是它包含了兩個一樣大小的虛擬螢幕(Virtual Screen)，以減少重繪時所需的計算量。因為使用虛擬螢幕，所以結果要用 draw()來對終端機直接輸出。

<< Class Frame_Style >>

Box 的 Member class，在 View 家族中，只要有邊框的，邊框樣式都由在此決定，可以自由更改。

<< Class View_Style >>

View 的 member， 一些有關於 View 特殊顏色的參數，都存在裡面。每個 View 都可有自己的 View_Style，可以自由更改。

<< Class Port >>

Port 是用來儲存 View 裡面顯示文字的類別。一個 LineBuffer 是一行，而 Port 裡面有 DoublList 來管理 LineBuffer，所以就可以組成一篇文章。

3.1.3 有關 Controller 層的類別:

<< Class ContlItem >>

所有事件產生器(Controller 代理物件所管理)的父類別。由 Controller 的 pool（）函式輪循呼叫在此的 pool（）函式，為其對外溝通的介面。

<< Class Controller >>

代理物件層的管理員。裡面的 run()就是整個系統運作的核心，系統中唯一的主動者。

<< Class Idler >>

系統空閒時間事件產生器。供模組申請使用，系統空閒時就會產生事件。

<< Class IdlePoint >>

Idler 內部用的類別。

<< Class Keyboard >>

系統鍵盤事件產生器。當使用者按下鍵盤時，會產生鍵盤事件。由於複雜度太高，所以程式並沒有支援特殊鍵。

<< Class Timer >>

系統時間事件產生器。每到設定的時間，就會對所申請的模組發出時間事件。

<< Class TimerCounter >>

Timer 內部用的類別。

3.1.4 有關菜單(Menu)的類別:

<< Class Control >>

跟菜單(Menu)控制鈕(control buttom)有關類別的父類別，宣告一些共有的虛擬函式，基本共通的介面。

<< Class Menu >>

菜單類別。

<< Class MenuBar >>

畫面最上方的選單列。事件會經由 MenuBar 傳往菜單 Menu。

<< Class MenuItem >>

Menu 裡面的選項，只負責代表資料不負責螢幕輸出。

<< Class MenuList >>

父類別是環狀串列，可說是專門為 Menu 作出的 RingList。包含者類別應當使用模板(Templete)來實做。

<< Class MenuPoint >>

有關菜單的類別 Menu， MenuBar，MenuItem，共同的父類別。由於須要用包含者類別統一管理，所以需要有共同的父類別，然後用多型來指定。

3.1.5 有關終端機的類別:

<< Class Terminal >>

有關終端機底層細節的類別。終端機在程式啟動時，需要設定的參數，都放在建構者函式中，自動設定完成，一般情況下不須再自行設定。

<< Class PObject >>

這是螢幕輸出類別的介面，Screen 和 Vscreen 都是符合此一介面的類別。View 家族都有一個更換螢幕的函式 changeScr()，只要符合介面的類別，都可以抽換。

<< Class Screen >>

螢幕輸出類別，符合介面 PObject。 有別於虛擬螢幕(VScreen)，直接輸出到終端機。由於終端機如果輸出碰到右下角的字元，螢幕就會捲動，這種情形不是一般情況所需要的。所以定義了常數'OFFSET'，指定螢幕底端留空白的行數。

<< Class VScreen >>

虛擬螢幕類別(virtual screen)。

<< Class Cursor >>

游標的定位，需要透過終端機低階的操作。因而用此類別包裝出高階的介面，讓使用者能更方便的使用游標。

<< Class VCursor >>

虛擬螢幕用的游標。可以設定來限制游標移動的範圍。雖然是針對虛擬螢幕，但它的輸出是直接轉往終端機（為了節省計算量）。也就是在虛擬螢幕中，您是看不到游標的。

<< Class Beep >>

當程式遇狀況需要警告(提醒)使用者(比如說操作錯誤等等)，往往以鳴叫(beep)或閃爍(flash)的方式輸出，故設計同時兼具此二功能的 class 以作終端機控制。

<< Class CharacterUnit >>

在 Dos 中，要在螢幕上寫上一個有顏色的字，並不是一件難事。只要再加上一個屬性字元及可。然而在 Linux 的環境下，因涉及終端機的控制，屬性是一列字串，須要透過"ncurses"函式庫來操控，故設計此類別，以一種新的單位，來代表螢幕上的一個點，就是 CharacterUnit。而其最底層的輸出函數就是 putChar()。

3.1.6 其他類別:

<< Class CPoint >>

此類別代表一個點。

<< Class CRect >>

此類別代表一個矩型範圍。

<< Class CEvent >>

系統傳遞事件時，裝載事件資訊的類別。

<< Class DoubleList >>

這是系統所用的包含者類別(Container class)，以及環狀(RingList)的變形。

<< Class RingList >>

環狀串列，DoubleList 的子類別，供繼承用。

<< Class Node >>

DoubleList 裡面所用的內部類別，也就是實際儲存物件指標的節點。我們可以透過 Node 指向所有 CObject 的子類別，這也是 DoubleList 為包含者類別的原因。

<< Class CObject >>

這就是 M-V-C 架構下，所有類別的父類別。其中包含 DEBUG 用的變數。

<< Class LineBuffer >>

存著由 CharacterUnit 所組成的字串，Screen 中就有包含這成員類別。在填背景時很常用。Port 中也由它來儲存字串。

<< Class String >>

自定的處理字元字串函數，可以考慮參酌使用。

<< Class StatusLine >>

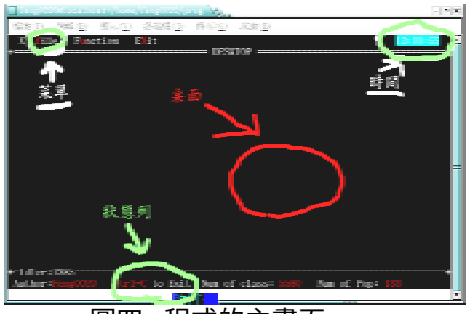
就是畫面底端的狀態列。狀態列預設是直接對終端機(Screen)輸出，但可以改變。

<< Class Xerror >>

測試例外處理時所用的類別。

3.2.Demo 範例程式說明

“test_m_v_c.cc”是參考的範例程式，編譯完後會產生可執行檔”demo”，可藉執行此程式以瞭解本類別庫的執行效果:



圖四: 程式的主畫面

主畫面如圖四所示，要叫出菜單請按下'Ctrl'+'\'(這是內定值，但是可以改的，在”type.h”中.)。按下'Ctrl'+'\'後，菜單拉下。用'h'，'j'，'k'，'l'，來在菜單中切換。(這就是 vi 中所用的左上右下。因為程式現階段還不支援特殊鍵，所以還不能夠用方向鍵。)，

Function 中的 move 請特別注意。當桌面上有視窗後，選擇”move”，然後就可以用'h'，'j'，'k'，'l'，移動視窗。用'H'，'J'，'K'，'L'， 控制視窗大小。結束”move”模式請按'M'.

File 裡面有四個子選項:

:+ Open (Menu Item)

:+ New (sub-menu)

:+ Save (Menu Item)

:+ Close (Menu Item)

Open 和 Save 暫時沒有功用。Close 可以關上最上一層的視窗， New 是一個子菜單，可以選擇要開一個視窗或文字流覽器。移動游標讓”New” Focus， 按下 'Enter'，打開子菜單，我們選擇開一個文字留覽器(Viewer)，如圖五:

圖五: 開一個文字流覽器

文字瀏覽器內有游標可以移動，請試試看。('h'='left'，'j'='down'，'k'='up'，'l'='right')這次就多開幾個 View，請注意因為這是 Model 'MyApp' 的視界(View)，而 MyApp 有申請時間事件產生器。所以每隔一段時間，View 就會自己更新一次(時間間隔可自由設定)。結束程式，請按'Ctrl'+'C'或者選擇菜單'Exit'。

四、結論

本論文針對 Linux 環境缺乏一具 OO 擴充性的視窗化公用程式庫，提出一組利用 M-V-C 三層架構，以事件驅動式方法建立的類別函式庫，能支援分時多工的菜單、狀態列等介面設計。整個類別函式庫的功能函式已全部完成，可隨時提供 programmer 在具”ncurses”函式庫和”POSIX 1003.1c”相容標準的環境下使用。

經過初步的適用及檢討，發現本函式庫雖可正常如預期運作無礙，但仍有以下數點值得後續研究改進。

4.1.效能的改進

首先就是效能的問題。程式在 Xwindow 執行的時候效能不佳。但在純文字模式下執行則無此現象。經推測其問題應當在於 X Windows 的終端機模擬器更新比較慢的關係。再來就是螢幕處理方面。程式裡有內建的虛擬螢幕(雙層)交由 Desktop 管理，其實也只是簡單的一次更新整個螢幕而已，以後可以在此方面再加強。

4.2 .輸入功能方便化的改進

終端機的部份，相關的設定都集合起來變成公共物件 termianl。輸出部份，變成公共物件 screen。輸入部份由於較棘手，目前皆是以代理物件(Controller)keyboard 自行解決，而且沒有辦法支援特殊鍵(F1，F2，Left，Right…)，此點直得在下階段重新改良。

4.3 .移植性的改良

本類別庫為達到分時多工的執行，利用在類別 Timer 的建構者函式，丟出一個執行緒，專門來設定 Timer 的旗標。然後在 Timer 的解構者函式中把執行緒銷毀。因此技巧用到執行緒，所以必須在符合 POSIX 1003.1c 以上的平台上執行。這樣降低了可移植性。因而再重新改良 Timer 的控制(如直接以組語控制硬體配合)，即可排除此一限制。

五、參考資料

[1]. 古清德,*事件驅動式程式設計*, 旗標, 1995.

[2]. 洪錦魁, *Turbo C 入門與應用徹底剖析*, 松崗, 1993.

[3]. Bjarne Stroustrup, *The C++ programming Language, The Third Edition*, Addison-Wesley. 1997

[4]. Erich Gamma , *Design Pattern*, Addison-Wesley. 1995

[5]. Grady Booch, James Rumbaugh, and Ivar Jacobson, *Unified Modeling Language UserGuide, TheFirst Edition*, Addison-Wesley.1999.

[6]. Herb Sutter.Exceptional, *C++:47 Engineering Puzzles, Programming Problem*, Addison-Wesley.

[7]. James Rumbaugh, Ivar Jacobson, and Grady Booch.The, *Unifed Modeling Language Reference Manual*, Addison-Wesley, 1999.

[8]. jesse Liberty and David B. Horvath. Sams, *Teach Yourself C++ for Linux in 21 Days*, Sams Publishing. 2000.

[9]. Richard Stones & Neil Mathew.*Beginning Linux Programming,* Wrox Press Ltd. 1999.

[10]. Stanley B.Lippman and Josēe Lajoie, *C++ Primer, Third Edition*, Addison-Wesley.1998