設計可置換之使用者介面建構於

具有 OSD 功能之嵌入式系統

張焜銘、黃顯詔*
成功大學工程科學系多媒體網路實驗室
(aprotoss,hchuang*)@www.mmn.es.ncku.edu.tw

黃悅民
成功大學工程科學系多媒體網路實驗室
huang@mail.ncku.edu.tw

摘要

綜觀現今的嵌入式系統世界，硬體的技術日漸創新，也造就了越來越多的嵌入式系統的應用及越來越豪華的硬體規格。因此許多廠商在各種不同應用的開發上，不但強調創新、多元化，更要在市場上搶得先機，所以多元化及快速的開發，已成為密不可分的二件事。而使用者介面(User Interface)是最直接與使用者產生互動，所以建置一個具有彈性、易於變換風格以及能簡單迅速地移植至各種不同嵌入式平台上的使用者介面，成為開發者所面臨的重要問題。本論文提出應用於具 OSD 硬體功能之嵌入式系統之可替換風格使用者介面，將使用者介面的行為模式，以 XML 描述取代傳統程式設計之程式碼，達到行為模式及程式設計分離，以加速開發流程，並且可重覆運用於不同之嵌入式平台。

關鍵詞：Skinnable, User Interface, embedded system

一、前言

在硬體技術的蓬勃發展之下，嵌入式系統的應用也越來越多元化。以多元化應用為前提下，軟體的開發設計不但要著重程式的彈性，更要求程式的可再利用性。為了搶得市場先機，程式可利用性的開發，為整個系統開發節省最多時間，並且是達到最大效益之不容忽視的要素。在使用者的觀點下，簡單易用、美觀大方的使用者介面也已是不可被遺忘的一環。但在嵌入式系統之中的圖形處理，仍比不上個人電腦般的強大，所以現有許多的相關應用及研究，如：直接使用記憶體空間 作為圖形顯示的 Framebuffer 或移植一般平台的圖形繪圖函式庫，包括 Qt/Embedded、FLTK、Mircowindows 或 GTK+等。

但現有的解決方案，其繪圖函式庫在開發的過程中，都需要透過修改大量的程式碼以達移植的目的。而且，現有的解決方案中，都需配合完整的繪圖架構，如 Gtk+/FB 及 DirectFB 之間的配合，造成架構之間溝通的額外負擔，大大地降低處理器應有的運算速度。

另外，美工設計及程式設計在現有的解決方案中，都是被相提並論的，但美工的直覺，不見得是一個程式設計師能體會的。所以，在現有的解決方案之中，似乎都多了點令人垢病之處，這似乎也是要尋求更加有效的開發方式的原因。

在本文中，我們會在第二節介紹 Skinnable User Interface 的相關背景及知識。第三節，詳細介紹所提出 Skinnable User Interface 理論及運作原理。第四節，將 Skinnable User Interface 的設計呈現在 TI DM6446(TI DaVinci)平台上。最後，結論及未來展望於第五節。

二、背景知識

(一) OSD(On Screen Display)

OSD 主要功能是透過圖形疊加的動作，將一些要告知使用者的資訊，呈現於螢幕上。一般硬體實現 OSD 的方式，可分為二種類型：字符型(Font-Based)及位元圖型(Bit-Map)二種。伴隨著硬體能力的日新月異，使得 OSD 所能顯示的畫面，越來越華麗；再加上，為了硬體及軟體實作的便利性，未來的 OSD 應該以位元圖型的為主[1]。本論文也是依位元圖型為主要設計對象。位元圖型的 OSD 產生器通常內建於視訊處理器內部，並共享使用其主顯示緩衝記憶體並可依照畫素進行改變控制。

(二) FrameBuffer

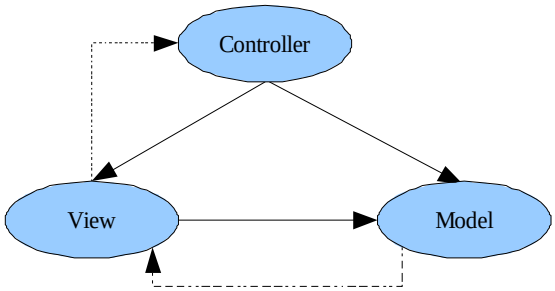
FrameBuffer 為一顯示設備，主要利用記憶體做為顯示的儲放空間。一般顯示系統之中，記憶體越大也代表能顯示的像素越多，呈現的色彩也越多。在 Linux 之中，將 FrameBuffer 實作成與顯示晶片無關的顯示抽象層，主要目的為將圖形顯示於一般終端機之上，而不需依靠額外的函式庫或顯示系統，如 x-window。位元圖型的 OSD，其行為模式及特性就如同 FrameBuffer，所以在 Linux 下大都會將 OSD 驅動程式，實作成 Linux 的 FrameBuffer，直接利用記憶體做為顯示設備。[2][3]

(三) XML(eXtenable Markup Language)

在 1996 年，全球資訊網協會(World Wide Web Consortium, W3C)，利用 SGML 的彈性及其強大的功能，並結合當時受到大眾喜愛的 HTML，設計可擴展標示語言(eXtensible Markup Language)，即稱做 XML。XML 是一套資料儲存工具，可以用來建立包含結構化格式資料的文件。除了資料外，還可以自行定義資料架構的詳細規則。它所呈現出來的概念相當地簡單明瞭，配合上 XML 本身的特色及優點，如：XML 為開放標準、使用標準字元集的 Unicode、其存取方式為 DOM (Document Object Model) Tree，可做有效率之讀取、搜索等動作；即可自訂出便利、輕巧、高效率的文件資料。[4]

(四) MVC 設計模式

MVC 模式是元件軟體設計(Design Patterns)之中的其中一種模式，其分成 Model-View-Controller 三個部份。在較龐大的系統之中，許多的資料呈現於使用者面前，但以開發者的角度而言，總是希望能分離資料和使用者介面的相依性，使得介面的改變不會影響資料的運作及使用。換言之，資料的改變也不會影響顯示的介面。如圖一：



圖一、MVC 設計模式架構圖

其中，Model 就如同資料庫一般，專司資料管理及儲存；View 主要為畫面的顯示，主要跟 Model 部份互動，在取得資料後，透過表單或圖形呈現於使用者面前；Controller 主要做為回應使用者的輸作事件(event)。[5]

三、系統目標及架構

本論文設計目標為實現一個可替換之使用者介面(Skinnable User Interface)，且將現有解決方案改良及重新規劃設計，達到同時符合 MVC 的設計模式及概念。所以將 Skinnable User Interface 整體架構規劃為 MVC 三部份，且為了達到高效率文件讀取處理方式，亦方便方便針對各種需求自訂格式，故加入 XML 的應用，將 MVC 三部份之設定以均以 XML 文件格式做為文件設定語言。

(一) Skinnable 控制單元

Skinnable 控制單元處理了所有的流程控管、輸入處理、使用者要求的動作通知及處理。其中的流程控管，就是狀態的

流程控制及管理。不論何種應用上，控管的狀態，都是有限的數量的。並且這樣的流程控管之中，一定會有一個起始狀態，而流程就依照輸入處理，將狀態轉移至下一個，這樣的方式，即稱為有限狀態機(Finite State Machine, FSM)。[6]

一般在嵌入式平台之中的輸入方式，都是屬於單向的由使用者(操作者)產生事件驅動(event)。使用者依據搖控器等輸入方式產生對應的按鍵編碼(KeyCode)，即以該按鍵編碼所對應的動作，由平台執行處理，並由 Skinnable 控制單元發出相對應的通知，告知 Skinnable 資料處理單元及 Skinnable 繪圖引擎產生對應的變化，以達到控制目的。

不同的嵌入式平台，對於按鍵編碼的定義及按鍵數目均不盡相同，為了能達到跨平台以符合各平台不同的編碼定義，對應到相同的事件處理，本論文定義了以 SKIN_為前綴(prefix)的事件驅動語法，如表一所示。

表一、Skinnable User Interface 事件驅動語法的定義

事件驅動	說明
SKUI_POWER	POWER 的開關
SKUI_CHANINC	Channel Increase
SKUI_CHANDEC	Channel Decrease
SKUI_VOLINC	音量遞增
SKUI_VOLDEC	音量遞減
SKUI_OK	OK
SKUI_MENU	選單按鈕
SKUI_INFOSELECT	information select
SKUI_1	數字 1
SKUI_2	數字 2
SKUI_3	數字 3
SKUI_4	數字 4
SKUI_5	數字 5
SKUI_6	數字 6
SKUI_7	數字 7
SKUI_8	數字 8
SKUI_9	數字 9
SKUI_0	數字 0
SKUI_ENTER	Enter
SKUI_INPUT	Input

以上述的事件驅動定義，對應至嵌入式平台的按鍵編碼，即可完成事件驅動。

搭配事件驅動，並以 XML 定義狀態。為了能符合各種狀態的差異，並達到狀態轉移的目的，提出以下的 XML 語法，來表示各種狀態。在狀態文件之中，如表二所示。由<states></states>來做為狀態文件的完整集合，以子節點<state></state>來宣告其中之一的狀態，其參數之中，以 name 做為該狀態之標籤， 並於其子節點中以<event></event>之中的 input 參數宣告該狀態之中可以使用之事件驅動，而由參數 next 定義下一個轉移的狀態為何，並以 action 參數來觸發相對應的執行動作。

表二、有限狀態機之 XML 文件

```
<states>
  <state name=”state1”>
    <event input=”SKUI_1” next=”state2” />
  </state>
  <state name=”state2”>
    <event input=”SKUI_2” next=”state1”
           action=”action2” />
  </state>
</states>
```

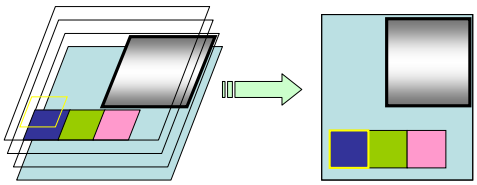
(二) Skinnable 資料處理單元

在不同的應用上，都會有著各自針對其應用所自定的特殊資訊(domain-specific representation of the infomation)，像是各項應用大都會有使用者可自行調整的設定值，這一類的設定值在系統啟動之際，就會被預先載入先前的最後設定值，達到人性化的要求。Skinnable 資料處理單元，就有如一個資料庫般，儲存著許多的資料(依應用而定)，跟資料庫不同的地方是，這一資料處理單元只提供資料存取，而不提供任何控制的功能。而這樣的一個概念，就等同於 MVC 模式之中的 Model 部份所扮演的角色。

(三) Skinnable 繪圖引擎

Skinnable 繪圖引擎，是主宰著呈現的效果的一個重要部份，主要著重於繪圖的方式及設計。一般只要使用 OSD 做為顯示硬體配備，都會搭配 Linux 所提供之 FrameBuffer 來達到繪圖效果，利用 FrameBuffer 的好處在於只要去存取記憶體的位置，就可以針對畫面上某一範圍進行變化，然而這樣的計算方式也可能要耗費不少處理器運算時間，所以在控制 FrameBuffer 這樣的變化同時，也要很仔細的注意運算量。再加上嵌入式系統的儲存空間有限，大多數的圖形都需經過壓縮，才得已在有限空間中存放足夠的圖形，相對的在顯示圖形的同時，才做解壓縮的動作，無疑對於處理器運算時間，也形成不可忽視的因素，才不會在圖形變換的過程之中，產生了延遲(delay)的現象。

FrameBuffer 只有一個特定的記憶體空間，而對於某一狀態繪圖的同時，可能需要佈置各種不同的圖形。一般而言 FrameBuffer 記憶體空間僅有一個，實作上都會透過疊加(Overlay)的方式，將各項不同的圖形，依先後次序疊加(存放)至 FrameBuffer 記憶體空間，如圖二所示：



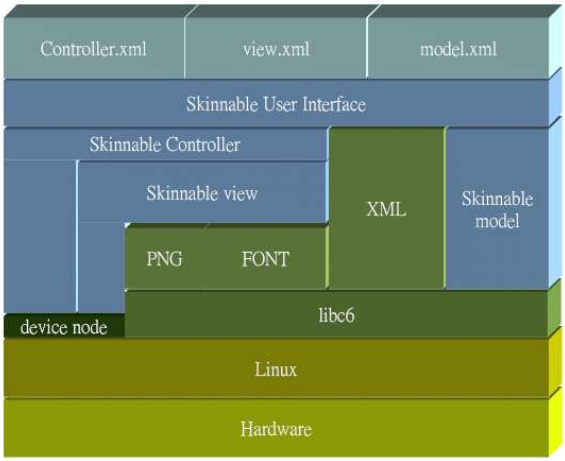
圖二、圖形的疊加

一般在使用者介面顯示上，尚有文字的處理。所有的應用程式中，眾多的調整選項或要告知使用者的動態即時訊息，大都會使用文字來建構互動介面，用來提醒或告知使用者某些訊息。而一般的文字處理，則是將文字經過運算，產生成圖形的形式，再運用貼圖，呈現於顯示介面，以達到文字需求。現在大多數的字型，都是使用描邊字型(outline font)來做處理。所謂描邊字型 (outline font)、又稱為向量字型，其主要就是以固定的大小的字型為基準，而利用數學方程式運算，將字型做放大或縮小，達到要求的尺寸，現在被廣泛的使用在各種系統之間。向量字型是較有彈性的一種字型方式，可以在有限空間中，僅使用一種字型，做到不同大小的尺寸，減少空間的耗費；但額多的處理運算，也是需要被考量的，所以採用繪圖時的方式，在設計上要盡可能選用較省資源的用法，如背景圖形是固定會被秀出的，在背景上的文字也可以運用影像軟體先行處理在圖形上，避免在 Skinnable 繪圖引擎中運用字型處理來秀出文字，花費不必要的運算時間，以避免過多的文字所造成延遲的現象。

四、平台測試及實現

本論文開發環境，以德州儀器(TI,Texas Instruments,) TMS320DM6446 (TI DaVinci)做為測試平台[7]，並配合 XML 函式庫、圖形函式庫(JPEG、PNG)、

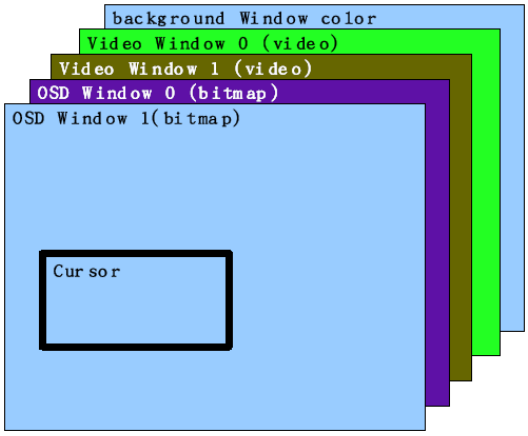
字型函式庫設計而成，其完整軟體架構圖如下圖三所示。



圖三、Skinnable User Interface 軟體架構圖

在 TMS320DM6446 的架構中，遠端控制器是受 MSP430 這一個微處理器(Mircocontroller)所控制。另外，在電路溝通上，則是使用 I^2C bus 跟 MSP430 微處理器溝通。透過 I^2C 取得紅外線搖控器的按鍵編碼後，只要對應按鍵編碼，即可將編碼對應的事件驅動交由 Skinnable 控制單元處理。

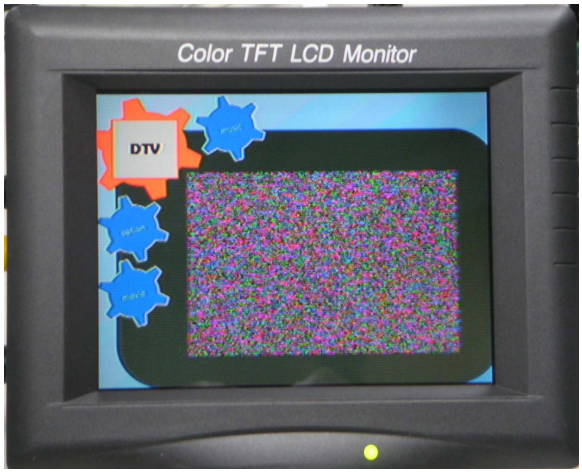
而在 OSD 的部份，平台上提供了視窗背景顏色(background window color)、二個視訊視窗及二個 OSD 視窗及一個游標視窗(由其程式碼得知尚未實作完成)，而其疊加之優先權如下圖四：



圖四、TMS320DM6446 OSD 分層結構

OSD 視窗 1 具有一個獨一無二的特性，它可透過屬性值控制所有的視訊視窗及 OSD 視窗 0 之間的透明度。而二個 OSD 視窗都可透過調整來接受 RGB565 或者 bitmap 資料。二個 OSD 視窗在同一時間中，只有一個能被調整使用 RGB565 的資料。當 OSD 視窗使用 RGB565 跟 Bitmap 資料時，最大的差別在於 256 色的色彩檢查表(color lookup table, CLUT)。當 OSD 視窗被調整為接受 bitmap 資料時，它可以使用色彩檢查表做為色彩呈現之用，當被調整為 RGB565 時，這部份即可被忽略。因此最好是可以調整 OSD 視窗 0 使用 RGB565 的模式，而將 OSD 視窗 1 調整為屬性視窗(即使用 Bitmap 資料) [8] 。

透過本實作上的測試，即可很輕易的設計出自己喜愛的風格介面並快速替換，如下圖五：



圖五、實際測試結果

五、未來展望及討論

透過 Skinnable User Interface 的運用，在同樣的平台，只要修改 XML 設定文件，即可產生出不同風格的介面，乃至運用於不同的應用之中。若要使用於不同的平台，只要將移植目標平台的輸入部份跟 FrameBuffer 的部份(即 controller 跟 view 二部份)稍做修改，就可以輕鬆地將程式移植。

因使用 MVC 設計模式，對於各部份的設計，都能各自發展，而目前僅讓三部份匯整在同一個程式之中。未來若能將三部份，各自以執行緒方式獨立執行，必定能增進許多效率。目前這樣的架構下，最缺乏的是針對美工開發人員設計一套前端發展介面程式，擁有前端介面程式，才能將 Skinnable User Interface 真正的開發效率給發揮出來。而且，依照這樣的開發模式，若 Skinnable User Interface 發展得夠成熟，配合上前端發展介面程式，就可以將這樣的程式，應用於許多的硬體平台，甚至以商業封閉源始碼的形式銷售，達到程式碼保護的目的。

六、參考文獻

[1] 陳金榮,
“結構化的平面電視OSD介面設計”, http://www.eettaiwan.com/ART_8800373096_480702_780caec6200508.HTM,2005

[2] 「FreamBuffer 的原理」, http://www.mcublog.com/blog/user1/9450/archives/2006/19773.html , 2006

[3] Wikipedia, “Frame Buffer”, http://en.wikipedia.org/wiki/Framebuffer

[4] Wikipedia, “XML”, http://en.wikipedia.org/wiki/XML

[5] Wikipedia, “Model-view-controller”, http://en.wikipedia.org/wiki/Model-view-controller.

[6] XML Finite State Machine in C#” , http://www.codeproject.com/csharp/xmlfsm.asp , 2002

[7] Texas Instruments, “http://focus.ti.com/dsp/docs/dsphome.tsp?sectionId=46”,2006

[8] Texas Instruments, “TMS320DM6446 datasheet”, "Fast Development with DaVinci On Screen Display (OSD)", http://focus.ti.com/docs/prod/folders/print/tms320dm6446.html , 2006