國立交通大學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

手寫輸入裝置之取樣模組的設計與實作

Design and Implementation of a Sampling Module for Handwriting Input Devices

研 究 生：吳培舜

指導教授：陳登吉　教授

中 華 民 國 九 十 八 年 六 月

手寫輸入裝置之取樣模組的設計與實作

Design and Implementation of a Sampling Module for Handwriting Input Devices

研 究 生：吳培舜　　Student：Pei-Shuen Wu

指導教授：陳登吉　　Advisor：Dr. Deng-Jyi Chen

國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士班

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

in

Industrial Technology R & D Master Program on

Computer Science and Engineering

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

手寫輸入裝置之取樣模組的設計與實作

學生：吳培舜　　指導教授：陳登吉

國立交通大學資訊學院產業研發碩士班

摘　　要

圖形化介面(Graphic user interface)是目前流行的操作介面之一，而指向裝置(Pointing device)與點擊式操作是圖形化介面的特色。在指向裝置的種類裡，數位板和互動式電子白板等設備都是以手寫輸入。因為手寫輸入的操作方式類似於紙筆繪圖的習慣，常被用於電腦繪圖等領域。

在 Windows 平台上，大部分的手寫式輸入裝置由廠商提供驅動程式將輸入事件轉換為原有的滑鼠訊息，使其可相容於一般應用程式。繪圖程式以原有的以原有的滑鼠訊息機制取得輸入事件時，因為滑鼠訊息機制是以佇列方式處理滑鼠訊息，佇列處理的時間會影響程式由接收輸入事件與回饋視覺效果的間隔時間。因此干擾程式回應使用者的視覺回饋效果，而影響需要快速視覺回饋的操作行為，如書寫或手繪等。

本研究實作了以 Windows Hook 為基礎的滑鼠取樣模組，並以非佇列式訊息轉換原本系統發送的滑鼠訊息，以加強繪圖程式使用手寫式輸入裝置的視覺回饋效果。最後進行實驗比較原有滑鼠訊息和本論文設計的效能差異，以驗證本論文的設計確實改善了其效果。

Design and Implementation of a Sampling Module for Handwriting Input Devices

Student：*Pei-Shuen Wu* Advisor：*Dr. Deng-Jyi Chen*

Industrial Technology R & D Master Program of
Computer Science College
National Chiao Tung University

ABSTRACT

One of the popular user interface, graphic user interface (GUI), is a point device characterized by its "point-and-click" interaction. Among all of the pointing devices, tablets and interactive whiteboards are handwriting equipment. Because using handwriting equipments are similar to drawing with pens and papers, these devices are becoming more prevalent. Furthermore, the visual feedbacks of these devices highly depend on the sampling techniques. Therefore, in this study, we design and implement a sampling module to improve the quality of the visual feedbacks for window-based handwriting devices.

In the Windows operating system, manufacturers of the handwriting devices provide drivers to convert input events into mouse messages, so that the device is compatible with the drawing application.

The mouse message system employs a queue for messaging and the queuing delay affects the duration between the input event and the visual feedback, and thus, the user's visual feedback. Therefore, the queuing delay will interfere with real-time activities, such as handwriting and drawing, which require rapid visual feedbacks.

In this study, we implemented the mouse sampling module using Windows Hook, and converted mouse messages to input events by a non-queue messaging mechanism, to enhance visual feedbacks of the drawing program with the handwriting devices. We compared the efficiencies of the mouse queue mechanism and the design of this study.

We found that our design improved the performance of the handwriting devices significantly.

誌謝

首先感謝指導教授陳登吉博士，老師悉心的教導使我得以體會軟體工程領域的深奧，討論並指點我正確的方向，使我在這些年中獲益良多。本論文的完成另外亦得感謝曾建超的大力協助。因為有你及幫忙，使本論文能夠更加完整。

兩年裡的日子，感謝逸珅、乃宣、陳志軒等同學的共同砥礪，僅此致謝，希望往後你妳們的人生能更加的精彩。女朋友在背後的默默支持更是我前進的動力，沒有妳，相信這兩年的生活將是很不一樣的光景。

最後感謝父母和家人在背後的支持，你妳們的鼓勵成為我求學時強大的後援，使我得以順利完成學業。僅此感謝我親愛的家人和朋友們。

目錄

摘要 i
ABSTRACT ii
誌謝 iv
目錄 v
表目錄 vii
圖目錄 viii
第一章. 緒論 1
1.1 研究動機 1
1.2 研究目的 1
1.3 章節概要 1
第二章. 相關研究探討 3
2.1 相關技術介紹 3
2.1.1 Windows Message 3
2.1.2 Windows Hook 6
2.2 一般滑鼠取樣的方式 7
2.2.1 視窗滑鼠訊息 8
2.3 造成程式視覺回饋效果不好的原因 9
2.3.1 取樣數量不足 9
2.3.2 取樣時間過長 11
第三章. 取樣模組的分析與設計 13
3.1 加強視覺回饋問題的策略 13
3.1.1 可能加以改良的方向 13
3.1.2 本論文選擇的策略 14
3.2 本論文的做法 14
第四章. 模組設計與實作 16
4.1 模組概觀 16
4.1.1 模組架構圖 16
4.2 模組設計細節 16
4.2.1 滑鼠取樣方式 16
4.2.2 滑鼠訊息轉換成的非佇列訊息 17
4.2.3 滑鼠訊息和本論文使用訊息的比較 19
4.2.4 應用端程式開發 20
4.3 實驗 20
4.3.1 實驗環境 20
4.3.2 實驗內容: 訊息傳遞執行的時間分析 20
4.3.3 實驗:取樣數量分析 24
4.3.4 實驗結論 28

第五章. **應用範例** 29
5.1 應用取樣模組程式展示 29
5.1.1 滑鼠訊息與自訂訊息中斷方式 29
5.1.2 WM_COPYDATA 訊息與自訂訊息中斷方式 29
第六章. 結論 30
6.1 總結 30
6.2 未來發展方向 30
參考文獻或資料 31

表目錄

表 1 本論文提供的應用程式介面 16
表 2 滑鼠訊息和本論文轉換後的非佇列式訊息在用途上的比較 19
表 3 實驗環境 20
表 4 以滑鼠為輸入裝置得到的結果，單位為毫秒(millisecond) 21
表 5 以手寫數位板為輸入裝置得到的結果，單位為毫秒(millisecond) 21
表 6 以電子白板為輸入裝置得到的結果，單位為毫秒(millisecond) 22
表 7 以滑鼠裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數) 25
表 8 以手寫數位板的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數) 25
表 9 以電子白板的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數) 26

圖目錄

圖 1 訊息迴圈範例.....3
圖 2 佇列式訊息處理流程.....4
圖 3 非佇列式訊息處理流程.....5
圖 4 一個視窗訊息處理程式的例子.....6
圖 5 Windows Hook 回呼程式範例.....6
圖 6 佇列式訊息 Hook 處理流程.....7
圖 7 視窗硬體輸入模型處理流程〔6〕.....8
圖 8 滑鼠訊息在應用程式的處理流程.....9
圖 9 以視窗滑鼠訊息方式取樣的效果.....10
圖 10 明顯發生折線處.....10
圖 11 順線前的線條.....11
圖 12 順線後的線條.....11
圖 13 簡化的滑鼠事件處理流程.....13
圖 14 滑鼠取樣模組架構圖.....16
圖 15 滑鼠取樣模組運作方式.....17
圖 16 WM_COPYDATA 方式運作流程.....18
圖 17 自訂訊息中斷方式運作流程.....19
圖 18 三種傳遞方式在各裝置訊息傳遞時間的比較.....22
圖 19 使用滑鼠為輸入裝置，各次訊息傳遞時間的折線圖.....23
圖 20 使用手寫數位板為輸入裝置，各次訊息傳遞時間的折線圖.....23
圖 21 使用電子白板為輸入裝置，各次訊息傳遞時間的折線圖.....24
圖 22 以滑鼠裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖.....26
圖 23 以手寫數位板裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖.....27
圖 24 以電子白板裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖.....27
圖 25 加入整合取樣模組前後的畫筆軟體效果對照圖.....29
圖 26 本論文提出兩種方式的畫筆軟體效果對照圖.....29

第一章. 緒論

1.1 研究動機

近年來觸控面板〔1〕和互動式電子白板(Interactive whiteboard)〔2〕…等手寫式裝置日益流行，其操作方式適合書寫以及手繪，在美術設計與數位學習的領域中常被使用。

在 Windows 平台上，由於手寫輸入裝置的廠商會為了使其硬體與已存在的應用程式相容而透過驅動程式將手寫式裝置的輸入事件轉換為滑鼠訊息(Mouse Message)〔3〕。但是因為滑鼠訊息為佇列式訊息(Queued Message)〔3〕，而訊息佇列的處理時間在模擬一般的滑鼠操作上或許不至於有太大的影響，但是在需要較精細處理的動作，如筆跡或繪圖時，會因而被影響，而使得其顯示的繪圖或寫字效果較不理想。

1.2 研究目的

本研究的目標為設計製作 Windows 平台的滑鼠取樣模組，此模組取樣的速度需比較原有視窗滑鼠訊息機制更快，以及具有不用安裝，即可以相容於原有視窗訊息機制的特性。

1.3 章節概要

本論文大致分為六個章節，以下簡述各章節概要：

第一章. 提出本論文研究之動機與目標，先概述目前手寫式輸入裝置的發展，並提出本論文的動機與目標。

第二章. 介紹 Windows 硬體輸入模型，滑鼠訊息及 Windows hook 等和本研究相關的背景知識，並提出原先的滑鼠訊息機制對於程式回應的問題。

第三章. 分析第二章提出的問題可改進的部分，並提出本論文加強原有訊息處理機制的方式。

第四章. 根據前一章提出的需求，對模組進行規畫設計。除了設計實作外，再進行實驗

驗證，先對第二章所述問題進行驗證，再對提出兩種不同的訊息傳遞方式進行實驗比較以顯示本論文的設計確實對於原先的問題有幫助。

第五章. 模組使用實例展示。

第六章. 總結本研究的研究結果，並提出未來發展及展望。

第二章. 相關研究探討

本章主要介紹在 Windows 平台上本論文相關的背景知識，並由此導出一般的滑鼠事件取樣方式及其較不適於手寫等操作的原因。

2.1 相關技術介紹

2.1.1 Windows Message

在 Windows 作業系統中，視窗程式為訊息驅動(Event-Driven)的架構。Windows 系統會將系統中發生的事件轉換為視窗訊息(Windows Message)〔3〕，並將訊息以參數的方式，傳遞給視窗程式的視窗訊息處理程式(Windows procedure)〔3〕。

程式設計師必需在該程式加入如下圖所示的訊息接收迴圈(Message Loop)〔3〕，並依照程式的目的實作視窗訊息處理程式，負責接收處理和程式功能相關的訊息。視窗訊息處理程式是負責處理訊息的程式，為一種回呼函式(Callback Function)〔4〕，回呼函式的意思是該函式被系統呼叫而非由程式自行呼叫，也就是當訊息發生時，是由系統呼叫視窗訊息處理程式處理訊息。而對所有和視窗事件相關的工作皆由此程式處理。除了程式設計師實作的視窗訊息處理程式外，系統也提供預設的視窗訊息處理程式，當程式設計師實作的視窗訊息處理程式收到不需處理的訊息時，就將該訊息轉發給預設的視窗訊息處理程式處理。

```
MSG msg;
while(GetMessage(&msg,NULL,NULL,NULL))     //由佇列取得訊息
{
        TranslateMessage(&msg);    //轉換訊息
        DispatchMessage(&msg);     //將訊息轉送至對應的視窗處理程式
}
```

圖 1 訊息迴圈範例

而訊息又分為佇列式訊息(Queued Message)和非佇列式訊息(Non-Queued Message)〔3〕。佇列式訊息基本上是使者輸入的結果〔4〕，系統將該訊息放入該程式對應的訊息佇列裡，等待上述的訊息接收迴圈將其取出，例如 WM_KEYDOWN、WM_KEYUP 等鍵盤訊息都屬於佇列式訊息。而非佇列式訊息是由系統直接發送訊息至該程式的訊息處理程式而不經過訊息佇列。如 WM_ACTIVE, WM_SETFOCUS 等訊息都是非佇列式訊息。

佇列式訊息處理的流程如下圖所示：

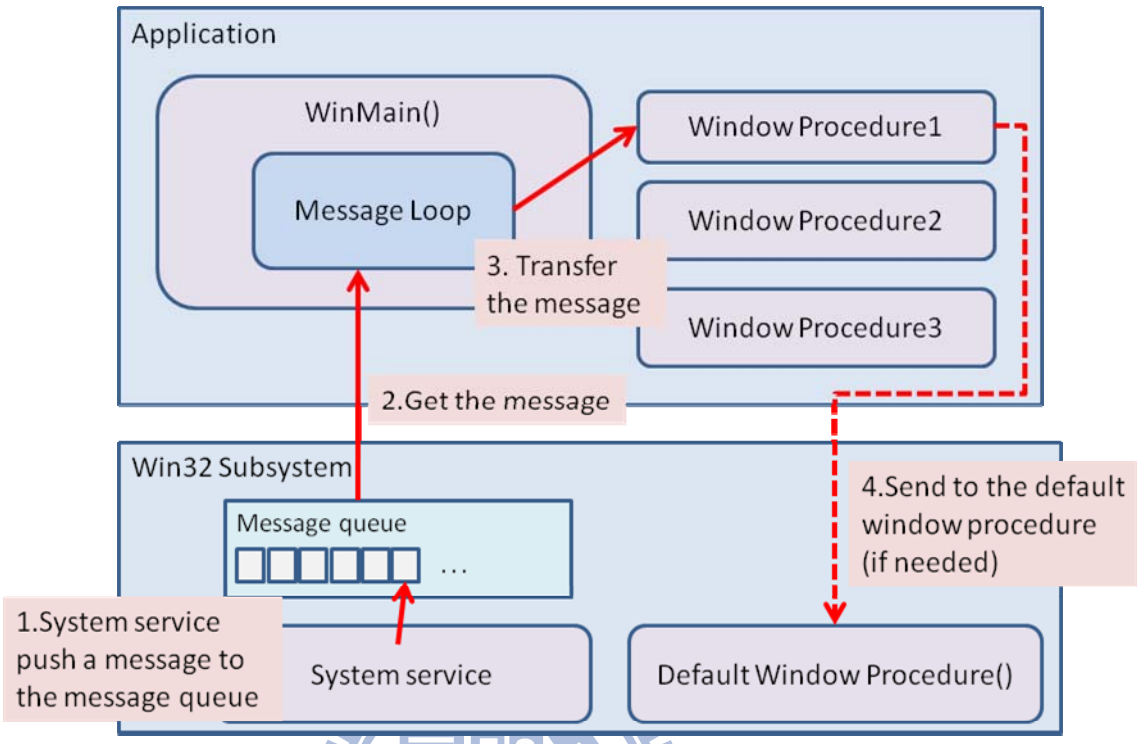

圖 2 佇列式訊息處理流程

佇列式訊息的處理流程如下：訊息首先被發送至訊息佇列中，當視窗程式的訊息迴圈從訊息佇列取出訊息並解讀其內容時，將其以參數方式傳送至需要處理該訊息的視窗訊息處理程式。最後視窗訊息處理程式處理完訊息後，需要系統處理該訊息，再將其傳遞給預設的視窗訊息處理程式，或直接傳回 0 值，即完成佇列式訊息的處理流程。

而非佇列式訊息的處理流程如下圖所示：

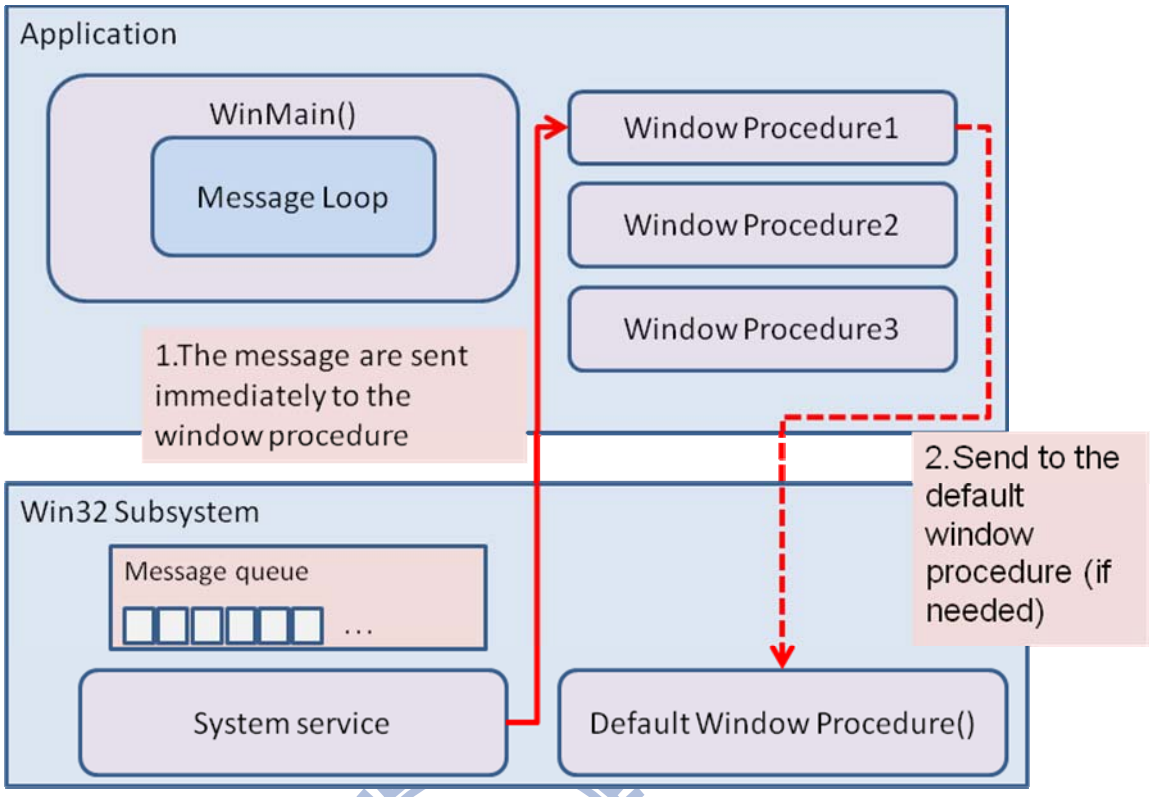

圖 3 非佇列式訊息處理流程

當一個非佇列式訊息被視窗系統處理時，系統會直接尋找並將訊息以參數方式送至負責處理該訊息的視窗訊息處理程式，最後如果視窗訊息處理程式需要系統處理該訊息，再將該訊息傳遞給預設的視窗訊息處理程式，或傳回 0 值，即完成一非佇列式訊息處理程式。

以設計一個視窗程式為例，該視窗程式在使用者執行該程式並按下鍵盤的 F1 按鍵時，顯示一個訊息方塊(Message Box)提示使用者，這時程式設計師需要撰寫如下所示的視窗訊息處理程式：

```
LRESULT CALLBACK WndProc
(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {
    case WM_KEYDOWN :
        {
            switch(wParam)
            {
            case VK_F1:
                MessageBox(hwmd, L"Press F1！", L"", 0);
                break;
```

```
            }
            return 0;
        }
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    }
    return DefWindowProc(hwnd, message, wParam, lParam);
}
```

圖 4 一個視窗訊息處理程式的例子

WndProc 為該視窗訊息處理程式的名稱，當按下 F1 鍵時，系統會發送 WM_KEYDOWN 訊息到該程式的訊息佇列。當程式的訊息迴圈將訊息取出後，會以 WM_KEYDOWN 及 VK_F1 為參數，呼叫 WndProc，使程式呼叫 MessageBox API 顯示訊息方塊提示使用者。而其餘的訊息，都交由 Windows 平台預設的視窗訊息處理程式負責處理，也就是上圖的 DefWindowProc。

2.1.2 Windows Hook

Windows Hook〔5〕是 Windows 系統提供給應用程式對系統裡訊息交流進行監視與干涉的一種機制。應用程式要使用 Windows Hook，需先呼叫 SetWindowsHookEx API 設定過濾訊息用的回呼函式(Hook Callback Function)〔5〕，當有指定要過濾的訊息發生時，系統便呼叫該回呼函式處理此訊息，當訊息處理完後，再由該回呼程式將此訊息傳遞給下一個負責的回呼函式，當所有的過濾訊息用的回呼函式都完成處理訊息之後，再交由目的視窗的訊息處理程式進行處理。

```
LRESULT CALLBACK CallWndProc
( int nCode,WPARAM wParam,LPARAM lParam)
{
   //進行對應的處理
   …

   //將攔截的訊息交由下一個訊息處理函式處理
   return CallNextHookEx(g_hkHandle, nCode, wParam, lParam);
}
```

圖 5 Windows Hook 回呼程式範例

以 Windows Hook 機制處理佇列式訊息的流程為例：

首先系統會將被攔截的訊息發送至以 SetWindowsHookEx API 登錄的回呼函式，回呼函式處理完訊息，再將其訊息送至訊息佇列中，最後以上述佇列式訊息流程依序進行。

下圖為當 Windows Hook 機制運作時，系統處理佇列式訊息的流程：

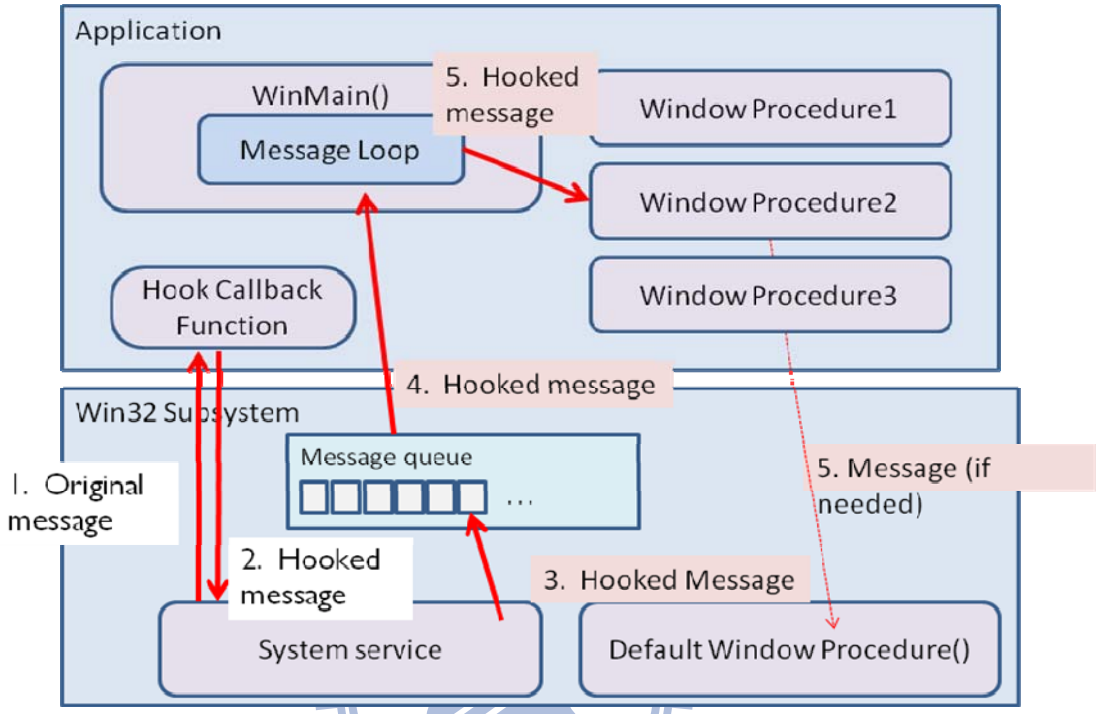

圖 6 佇列式訊息 Hook 處理流程

2.2 一般滑鼠取樣的方式

視窗硬體輸入模型

視窗硬體輸入模型〔6〕是 Windows 系統處理輸入裝置輸入，將其轉換為應用程式訊息的過程。下圖是處理的流程。

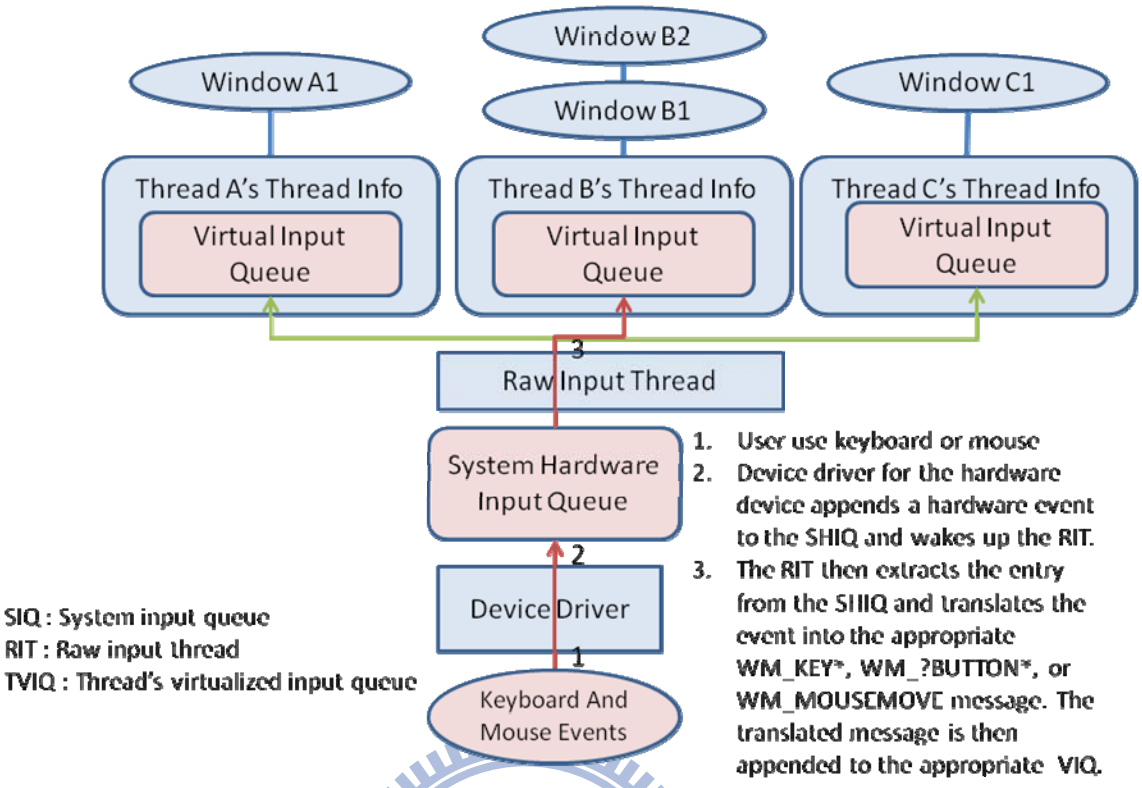

圖 7 視窗硬體輸入模型處理流程〔6〕

處理流程共分為三大步驟：

1. 當輸入裝置發出硬體事件，該硬體事件由裝置驅動程式負責處理。
2. 裝置驅動程式處理完會將該事件放入系統輸入佇列，並喚醒原始輸入執行緒。
3. 被喚醒的原始輸入執行緒先由系統輸入佇列取出硬體事件，並將其轉換為相對的訊息，例如滑鼠左鍵按下的訊息，會被轉換為 WM_LBUTTONUP 的訊息，並送至適當的執行緒的虛擬輸入佇列。

上述步驟 3 中適當的執行緒是指在事件發生時，應當接收該訊息的執行緒，例如當滑鼠在上圖的 Window A1 上移動時，原始輸入執行緒會將 WM_MOUSEMOVE 的訊息及滑鼠的座標，發送至對應於 Window A1 的 Thread A 的虛擬輸入佇列中。

2.2.1 視窗滑鼠訊息

滑鼠視窗訊息〔7〕是由上節所述的硬體輸入模型裡，由原始輸入執行緒轉換滑鼠事件後所產生的視窗滑鼠訊息。其中應用程式常使用的訊息如下所示：

滑鼠訊息	該訊息代表的滑鼠操作事件
WM_LBUTTONDOWN	代表按下滑鼠左鍵按下

WM_LBUTTONUP	代表放開滑鼠左鍵
WM_MOUSEMOVE	代表滑鼠移動
WM_RBUTTONDOWN	代表按下滑鼠右鍵
WM_RBUTTONUP	代表放開滑鼠右鍵

當視窗訊息處理程式中收到滑鼠訊息時，其 lParam 參數代表了該訊息發生的座標，而 rParam 代表其他按鍵的狀態。滑鼠訊息屬於 2.1.1 節中的佇列式訊息。

滑鼠訊息的處理流程如下圖所示：

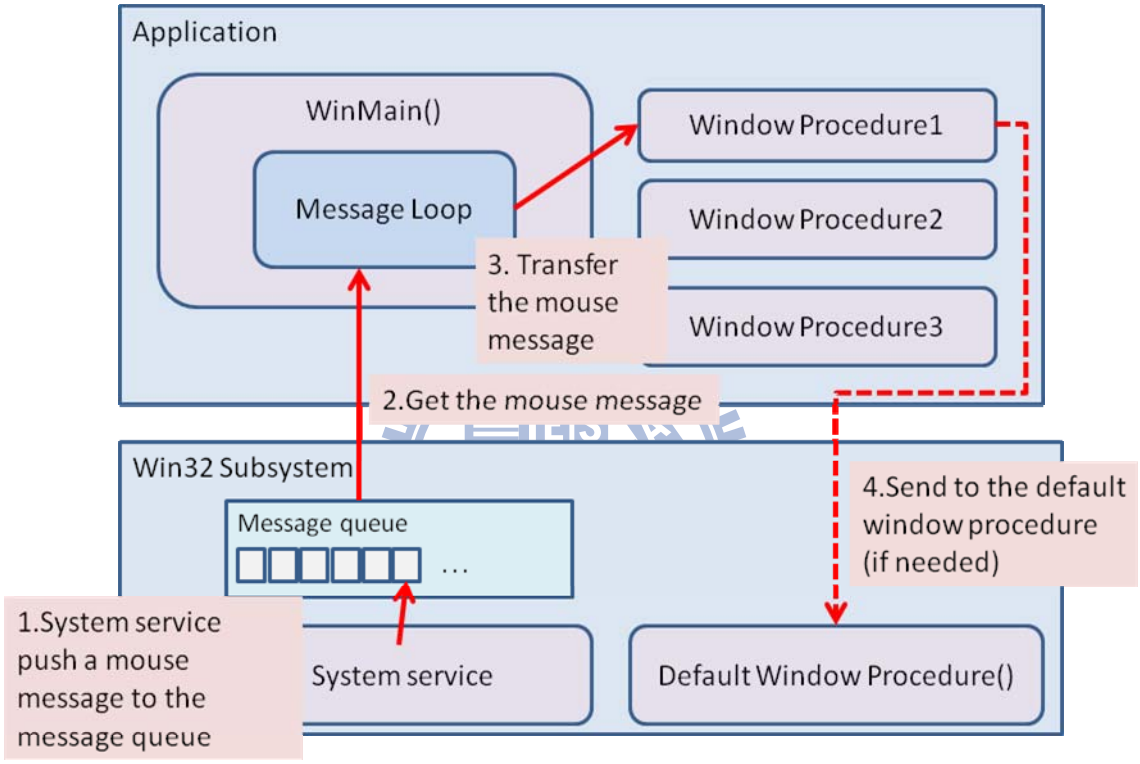

圖 8 滑鼠訊息在應用程式的處理流程

2.3 造成程式視覺回饋效果不好的原因

2.3.1 取樣數量不足

Microsoft Developer Network (MSDN)上提到，當訊息處理程式來不及處理滑鼠訊息時，Windows 系統會盡可能使其取得更多的滑鼠訊息〔7〕，也就是滑鼠訊息在程式過度忙碌時會有遺失的現象。

滑鼠繪圖類型的程式往往會對手繪的顯示方式處理，例如柔化線條的邊緣…等，往往會消耗系統運算資源，使得處理時間變長，因此影響到視窗程式取得滑鼠訊息的數量。

滑鼠取樣不足的問題在以滑鼠或手寫式裝置寫字或是畫曲線時效果更加明顯不

佳。因為畫圖程式往往會直接將取樣到的點以直線相接的方式反應在畫面上，當取樣點數不足時，將使其線段相接的轉角處更加的明顯。

Paint.Net 軟體是一套在 Windows 平台上的畫圖軟體。下圖是當系統內負載較高時，由 Paint.Net 軟體畫出的線條軌跡，原本要畫出一條曲線，但是由於系統的工作量較多，造成取樣不足，使得其中的取樣點的間隔相當明顯。

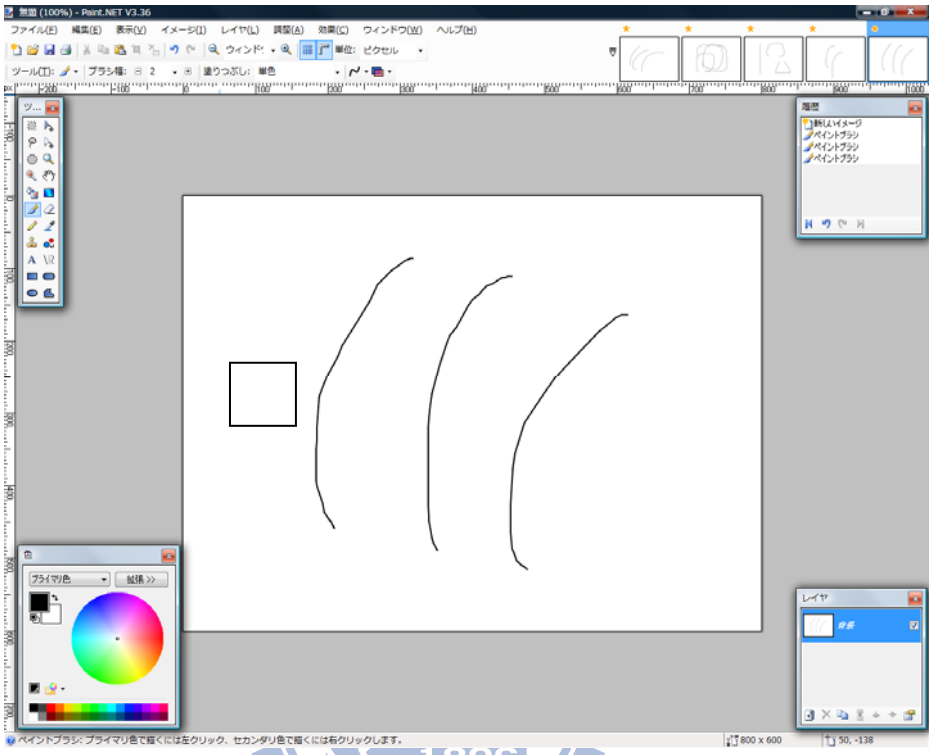

圖 9 以視窗滑鼠訊息方式取樣的效果

下圖為將上圖方框取出後放大兩倍之後的結果，可見到圓圈圈出的地方有明顯發生折線的現象。

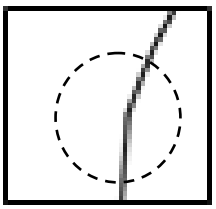

圖 10 明顯發生折線處

繪圖軟體為了應付這個問題，往往會加入順化線條的功能，在畫線時先以直線相接的方式顯示視覺回饋，使用者在畫完線條之後再調整收到的點，使其線條的效果更好。

下圖為 Windows Vista 內建的螢幕剪裁工具(snap tool)。該工具提供了在其剪裁的畫面上繪畫以及註記的功能，該工具即提供了順線功能使線條更加美化。

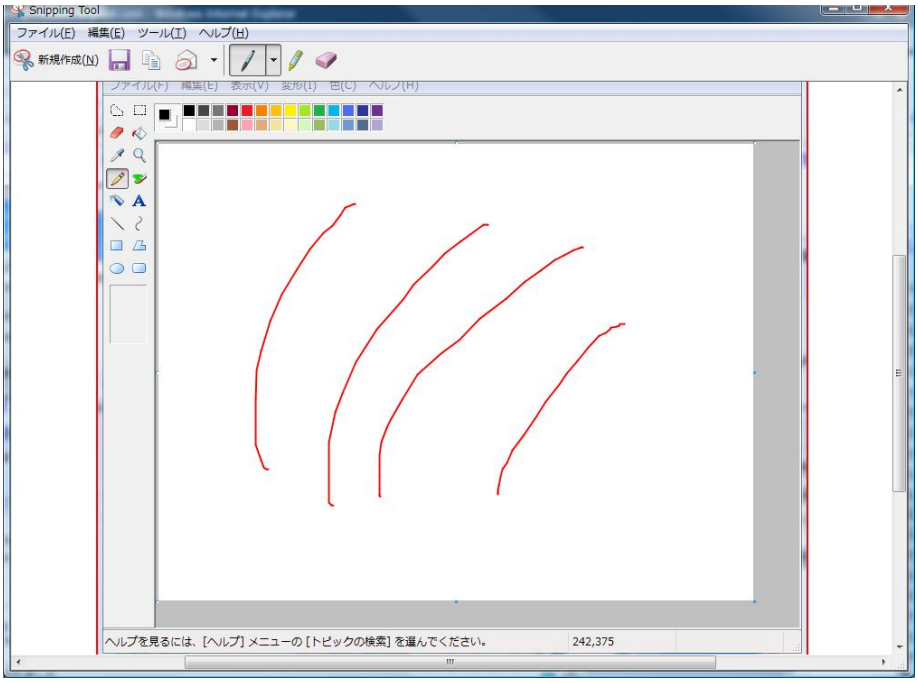

圖 11 順線前的線條

由於該工具的順化線條功能會在使用其提供的橡皮擦工具被觸發，下圖是將最右邊的線條使用橡皮擦工具清除後，畫面被順化線條功能處理完的結果。

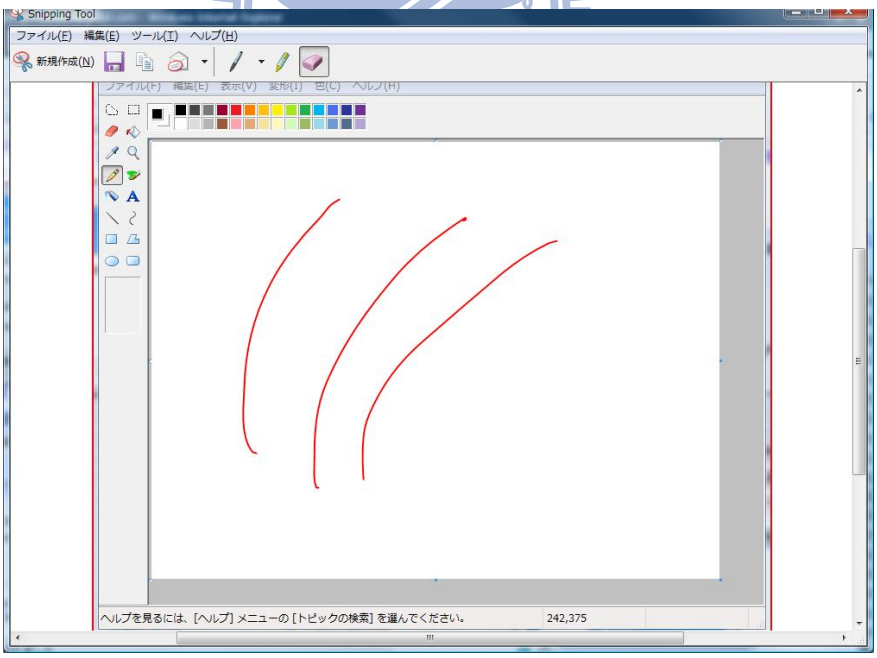

圖 12 順線後的線條

彎曲較多的線條被修正成曲線，但是修正的同時也造成線條的失真。

2.3.2 取樣時間過長

當滑鼠事件發生後，必須先由驅動程式處理，再由原始輸入執行緒發派到適當執行緒的虛擬輸入佇列裡。由於滑鼠訊息本身為佇列式訊息，應用程式必須在訊息迴圈使用 Windows API 取出並解讀訊息，並發送至適當的視窗訊息處理程式。當視窗訊息處理程式接收到滑鼠訊息後，尚需執行相關的繪圖指令，將視覺回饋效果反應至使用者。

若滑鼠事件處理流程使用的時間和使用者觸發滑鼠事件的總和時間間隔太長，使得使用者從操作事件發生後到視覺回饋的時間太久，會導致使用者有程式反應太慢的觀感。另外因為取樣的時間變長，使得在同樣時間內的取樣次數變少，進而造成取樣數不足的問題。

第三章.　取樣模組的分析與設計

3.1　加強視覺回饋問題的策略

3.1.1　可能加以改良的方向

從滑鼠事件發生到應用程式處理完畢的處理流程如下圖：

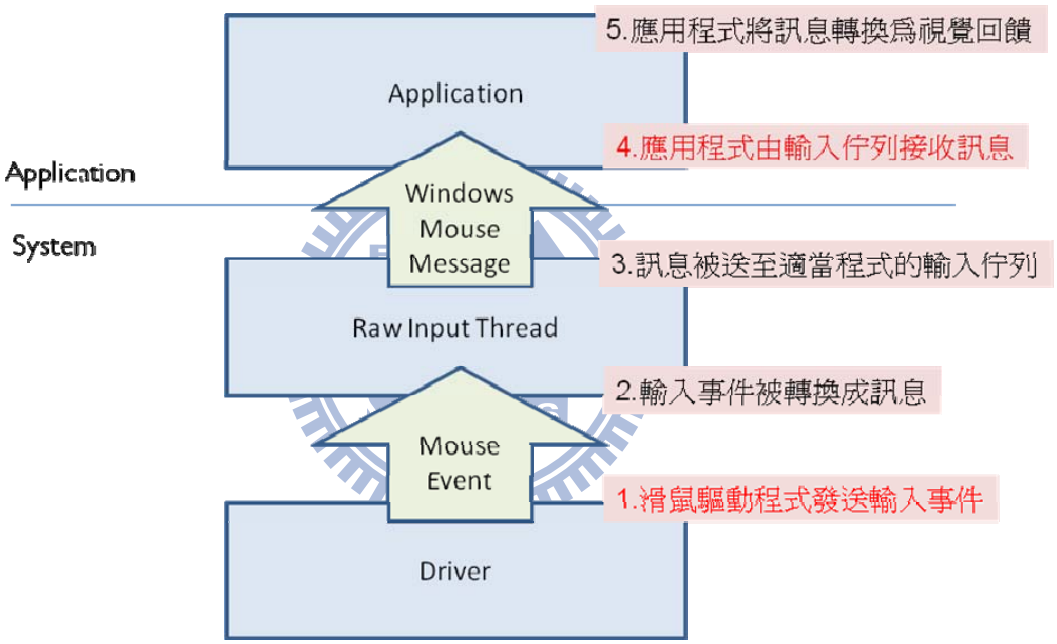

圖 13 簡化的滑鼠事件處理流程

從輸入事件被轉換為訊息乃至於訊息被放至適當的程式輸入佇列為止，是由原始輸入執行緒處理負責，因為是由系統提供的機能而無法修改，另外將訊息轉換為視覺回饋是由繪圖的程式完成，由搭配滑鼠取樣模組負責。因此滑鼠取樣模組可加以強化的只有這兩個部分:滑鼠驅動程式發送輸入事件以及應用程式由輸入佇列接收訊息的行為。以下針對相對應的修改驅動程式和加速滑鼠訊息發送至訊息佇列這兩個不同方向的優缺點作分析說明：

- 修改驅動程式

以驅動程式強化滑鼠取樣的方法即可以在滑鼠事件被轉換為滑鼠訊息前就先攔截並加以處理。但是因為驅動程式是在核心模式下執行，必須設計驅動

程式與應用程式溝通的介面，另外驅動程式需要預先在系統內被安裝才能使用。

- 加速訊息發送至應用程式的時間

這個部分的優點是不需要預先安裝或是導致該程式需要以系統管理員的權限，才可以使用該模組的應用程式。若在其他應用的程式不需要在系統裡留下設定的情況下，可搭配成為綠色軟體(Green Software)〔9〕，但是因為取得的資料已被原始輸入執行緒處理過，取得的滑鼠訊息會受原始輸入執行緒的限制。

3.1.2 本論文選擇的策略

在先前提到影響視覺回饋效果的兩個因素中，由於驅動程式的開發較為困難以及針對應用程式的設備有高度相關性，因此本論文選擇的策略是透過減少取樣時間進而加速訊息傳遞至應用程式的時間。減少取樣時間的好處是可以減少使用者操作輸入裝置時，視覺上對操作到達回應時的延遲。

3.2 本論文的做法

由於滑鼠訊息為佇列式訊息，本論文的做法是在滑鼠訊息進入訊息迴圈前，使用 Windows Hook 的機制將滑鼠訊息以非佇列式訊息先發送至接收視窗的訊息處理程式，繞過訊息迴圈進行處理，節省滑鼠訊息停留在訊息迴圈的時間。

這種作法有以下兩個優點：

1. 不需要安裝

近年來流行綠色軟體(Green Software)，其最大的特點是不需要將軟體安裝於作業系統設定中即可使用，移除後也不會將任何記錄留在執行該軟體的電腦上。其好處除了不會留下使用記錄外，更重要是安裝軟體的動作往往需要以較高權限的管理者帳號進行。如果是在企業的應用環境下，較高權限的帳號往往只只受限於該企業的 Management Information System (MIS)〔10〕管控，在使用時就需要先由 MIS 進行安裝才能使用，但是在臨時需要使用的情況下就會遇到執行上的困難。由於本論文的方法是使用系統提供的 Windows Hook 機制，因此只需要在執行時呼叫使用該功能的 API 即可使用，而不需要任何安裝動作。

2. 與訊息驅動架構相容

由第二章可知視窗程式是由訊息處理程式對滑鼠訊息做處理，如果新增的滑鼠取樣模組和原先的訊息處理架構不同，勢必會需要對原先程式的運作進行調整。本模組採用將滑鼠訊息轉換成非佇列式訊息的方式，好處是只要在應用程式的訊息

處理程式將原先的滑鼠訊息轉換成對應的非佇列式訊息即完成整合，而不需對原先訊息驅動的架構做調整，減輕修改程式的負擔。

第四章. 模組設計與實作

4.1 模組概觀

滑鼠取樣模組的主體為一 Dynamic Link Library (DLL)檔案，使用模組的程式需透過 LoadLibrary API 將模組載入使用，並呼叫模組的 OpenPen 和 ClosePen API 作為開始和結束取樣的操作。再經由 GetPoint API 解讀並轉換完成非佇列式訊息，在將原本處理滑鼠訊息的部分代換為處理本模組的訊息。

4.1.1 模組架構圖

本論文的模組內提供的應用程式介面

表 1 本論文提供的應用程式介面

名稱	說明
OpenPen	開始取樣
Close	結束取樣
GetPoint	取得該點的滑鼠資料

下圖為本模組的架構，主要由 Hook Callback Function 和提供呼叫的應用程式介面組成，Hook Callback Function 負責攔截滑鼠訊息後的處理工作，而使用本取樣模組的繪圖程式呼叫應用程式介面以使用本模組的功能:

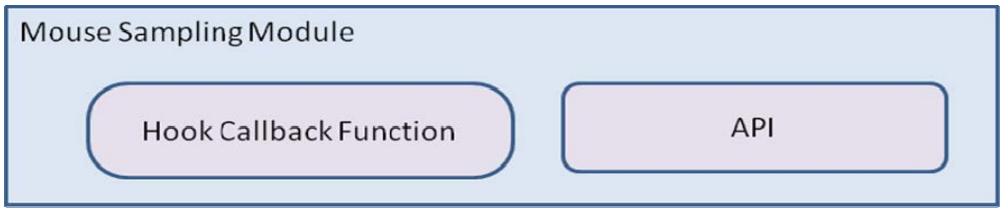

圖 14 滑鼠取樣模組架構圖

4.2 模組設計細節

4.2.1 滑鼠取樣方式

本模組使用 Windows 系統提供的 Windows Hook 機制作為滑鼠取樣的方式。使用 Windows Hook 的方式有兩個優點，第一個是使用 Windows Hook 機制可在該訊息進入視

窗程式的處理佇列前先被接收。第二、是使用 Windows Hook 機制可不需在系統裡進行安裝，允許在較低權限的帳戶下直接使用，如果使用驅動程式的方式進行取樣，則需要以較高權限的帳號進行安裝設定才能使用，導致用此模組的軟體就不可能成為綠色軟體。

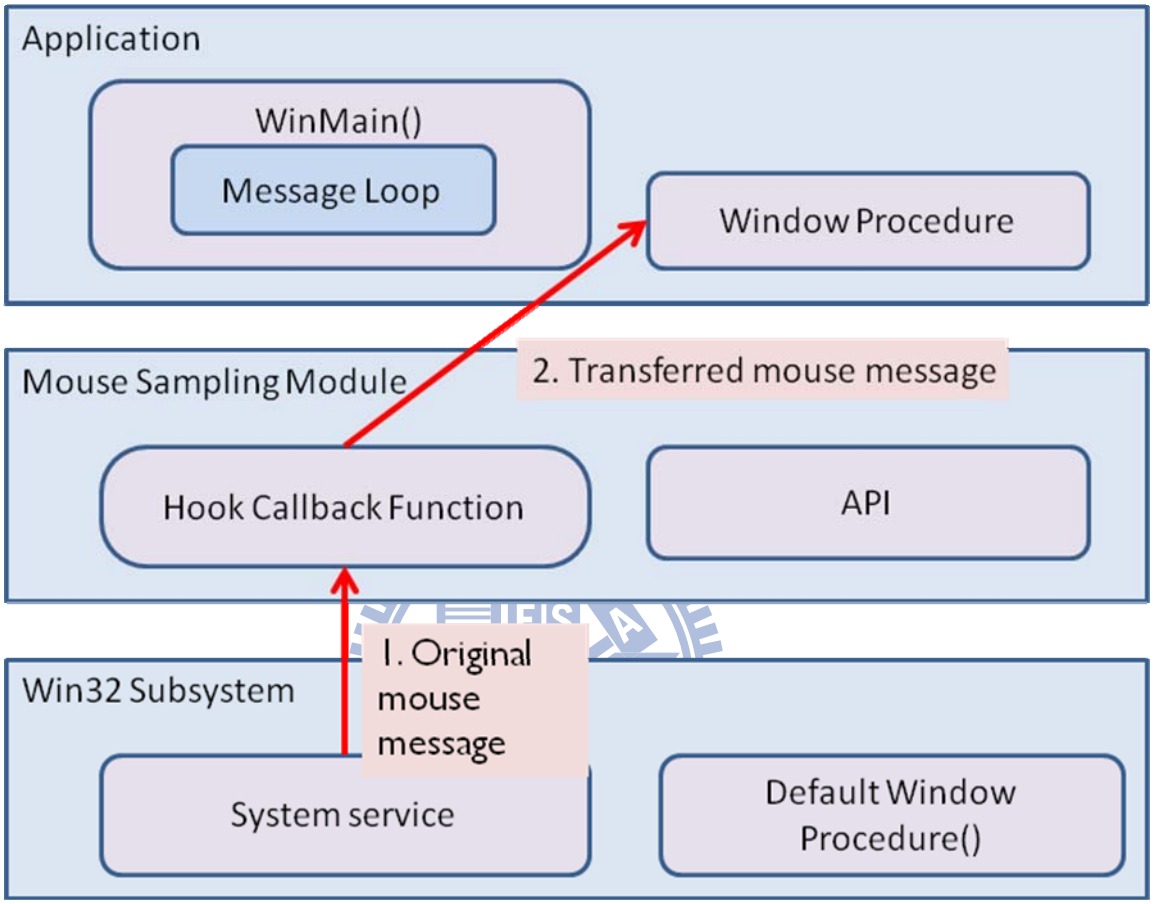

圖 15 滑鼠取樣模組運作方式

4.2.2 滑鼠訊息轉換成的非佇列訊息

滑鼠事件包含了事件的發生和該事件的相關資料，例如：指標的位置…等。本論文以兩種不同的非佇列訊息分別進行事件傳遞，並且在實驗內針對效率加以比較。整合時應用程式只需修改訊息處理函式處理滑鼠訊息的部分為接收對應產生的訊息，並且呼叫解讀訊息的 API，再將訊息處理改為原本應用程式對應的訊息處理方式即可，而不需對應用程式的處理流程或架構另做修改。

可供轉換的非佇列式訊息有以下兩種：

1. WM_COPYDATA 訊息

WM_COPYDATA〔11〕是 Windows 系統提供給應用程式作為少量資料行程間通訊的訊息，使用時對接收資料的客戶端視窗程式以 SendMessage API 傳送 WM_COPYDAYA 訊

息，並且將要傳遞的資料其位址和大小加入參數中即可。接收端的訊息處理程式只需要對 WM_COPYDATA 訊息做適當的處理。

2. 應用程式自訂訊息(Application Defined Message)

Windows 系統提供了自訂訊息〔3〕讓程式設計師建立自己使用的訊息，該訊息的數值需位於 WM_APP 的實際數值和 0xBFFF 的區間內。

本論文以此兩種訊息設計了兩種不同轉換滑鼠訊息的方法：

1. WM_COPYDATA 方式

利用 WM_COPYDATA 訊息負責觸發滑鼠事件的處理和相關資料的傳遞。由於 WM_COPYDATA 本身具有傳遞資料的能力，滑鼠事件的相關資料由該訊息的參數傳遞至視窗處理程式。

本設計如下圖：

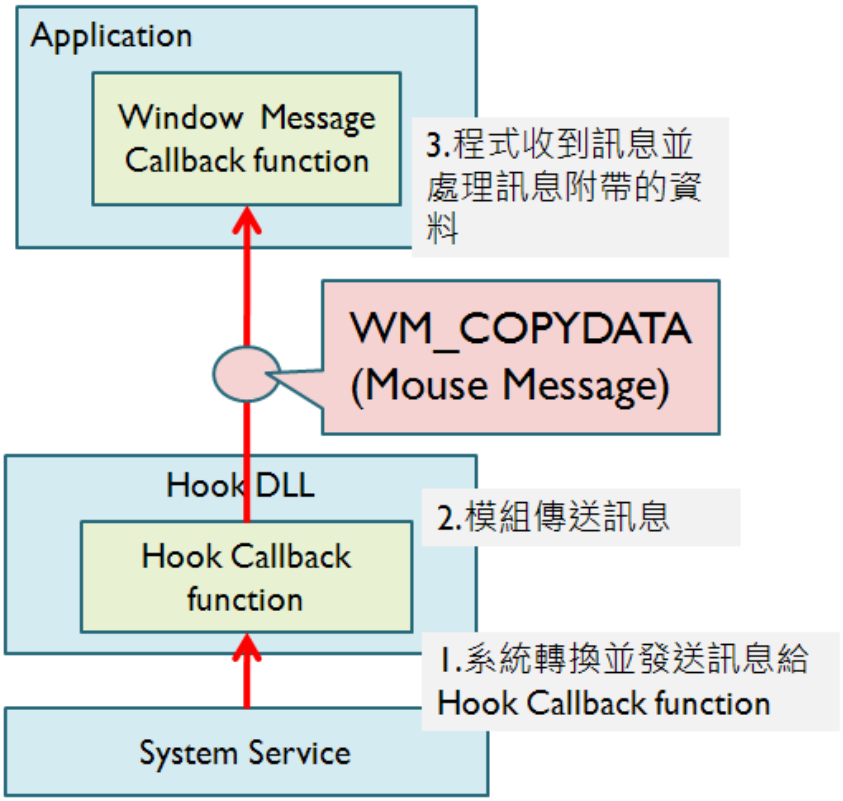

圖 16 WM_COPYDATA 方式運作流程

2. 自訂訊息中斷方式

以自訂訊息觸發處理事件在模組裡提供了取回資料的 API。本模組自訂訊息定

義為 WM_BWDLL，當應用程式收到該訊息時，呼叫該 API 以取得事件的相關資料。

本設計如下圖：

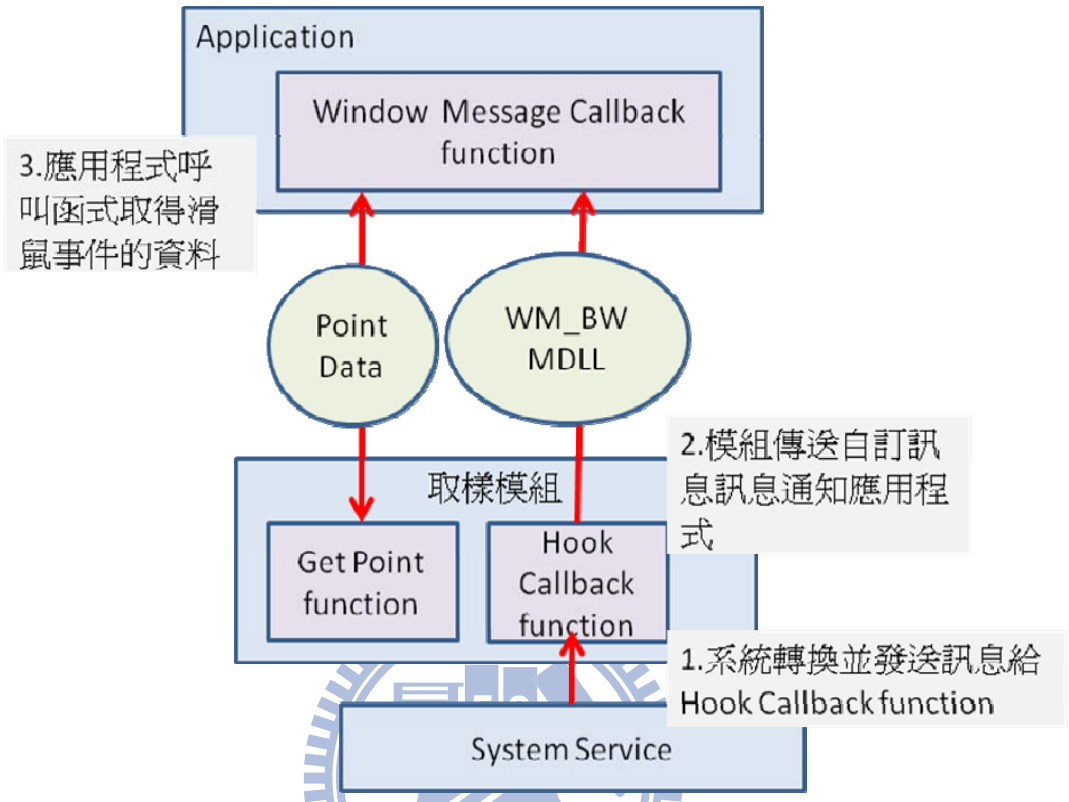

圖 17 自訂訊息中斷方式運作流程

4.2.3 滑鼠訊息和本論文使用訊息的比較

下表為三種訊息於用途上的比較：

表 2 滑鼠訊息和本論文轉換後的非佇列式訊息在用途上的比較

	用途	傳遞滑鼠事件方式
滑鼠訊息	負責提示應用程式滑鼠事件發生	由視窗處理程式的參數傳遞。
WM_COPYDATA 訊息	少量資料行程間通訊。	由視窗處理程式的參數傳遞，供應給程式自訂通知訊息使用。
應用程式自訂訊息	由訊息處理程式的參數傳遞。	模組需提供 API，於訊息到達時由呼叫 API 取得事件內容。

4.2.4 應用端程式開發

本論文的實驗應用環境為智勝國際〔12〕所開發的 SoezBoard 繪圖核心測試程式，以該程式取得取樣時間的功能進行測量細部工作時間。

4.3 實驗

4.3.1 實驗環境

本論文的實驗環境如下表所示 ：

表 3 實驗環境

電腦規格	Intel(R) Pentium 4 2.8GHZ DDR400 512MB Windows XP Professional
手寫數位板規格	WACOM BAMBOO
電子白板規格	Jector jp835xsp

4.3.2 實驗內容: 訊息傳遞執行的時間分析

本實驗將比較透過原先滑鼠訊息方式、使用本模組使用 WM_COPYDATA 訊息和自訂訊息傳遞滑鼠事件等三種不同方法針對訊息傳遞執行的時間。

實驗是為以滑鼠、手寫數位板和電子白板等三種不同的輸入裝置各得到 1000 個取樣點所花費的平均時間，分別共進行五次。

以下為實驗取得的數據，及由數據產生的比較圖：

1. 以滑鼠為輸入裝置得到的結果

表 4 以滑鼠為輸入裝置得到的結果，單位為毫秒(millisecond)

	滑鼠訊息	WM_COPYDATA 訊息	應用程式自訂訊息
第一次	6.36	0.61	0.016
第二次	7.057	0.017	0.015
第三次	7.795	0.718	0.005
第四次	5.486	1.644	0.003
第五次	4.827	1.72	0.001
平均	6.305	0.942	0.008

2. 以手寫數位板為輸入裝置得到的結果

表 5 以手寫數位板為輸入裝置得到的結果，單位為毫秒(millisecond)

	滑鼠訊息	WM_COPYDATA 訊息	應用程式自訂訊息
第一次	6.19	0.14	0.001
第二次	7.406	0.045	0.015
第三次	8.454	0.042	0.004
第四次	3.248	0.54	0.008
第五次	6.578	0.024	0.005
平均	6.3752	0.1582	0.0066

3. 以電子白板為輸入裝置得到的結果

表 6 以電子白板為輸入裝置得到的結果，單位為毫秒(millisecond)

	滑鼠訊息	WM_COPYDATA 訊息	應用程式自訂訊息
第一次	12.145	1.426	0.573
第二次	12.261	2.113	0.019
第三次	11.245	3.539	0.673
第四次	9.549	0.853	0.086
第五次	13.725	0.85	0.052
平均	11.4018	1.7562	0.2806

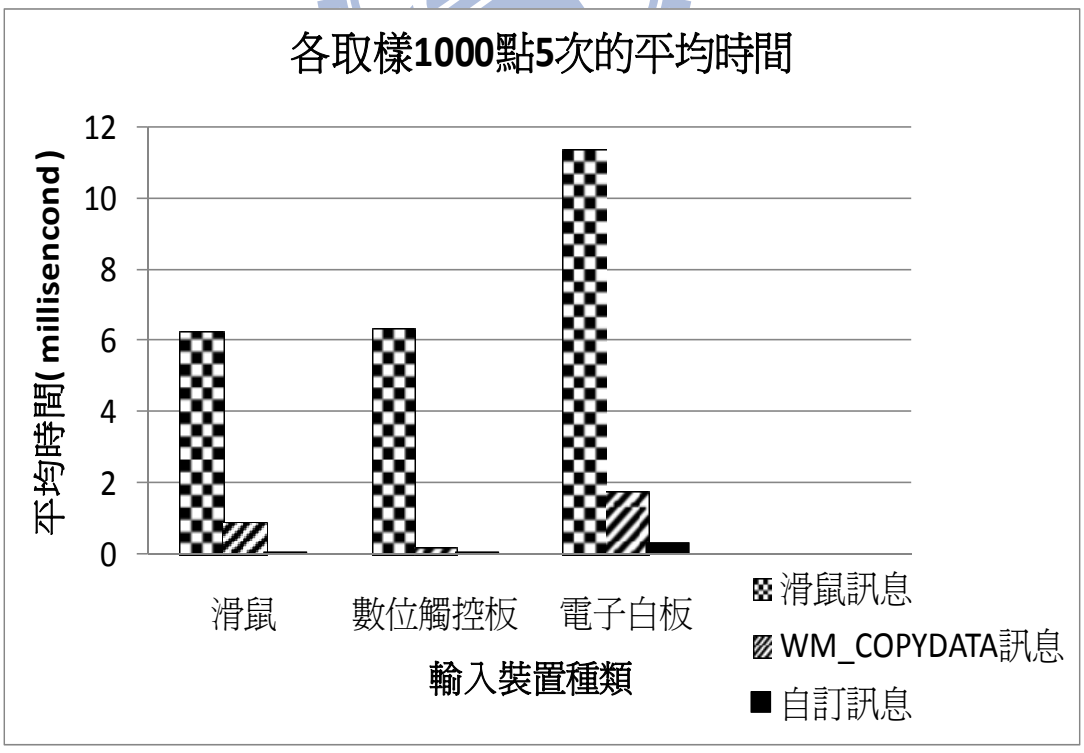

圖 18 三種傳遞方式在各裝置訊息傳遞時間的比較

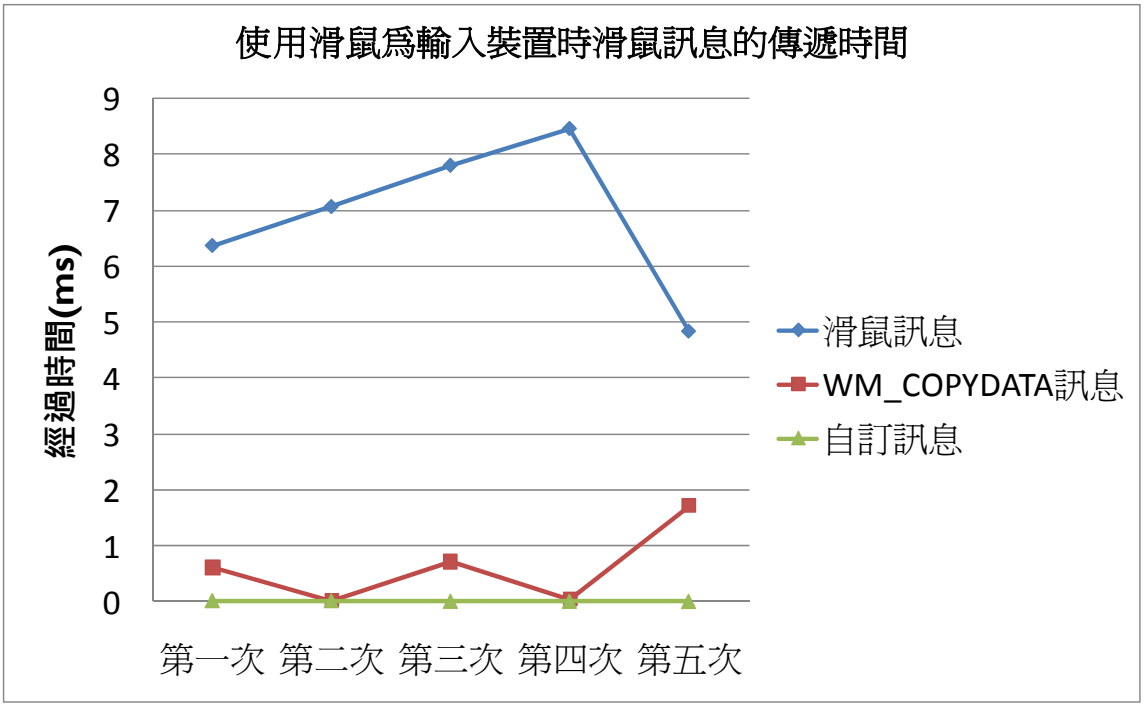

圖 19 使用滑鼠為輸入裝置，各次訊息傳遞時間的折線圖

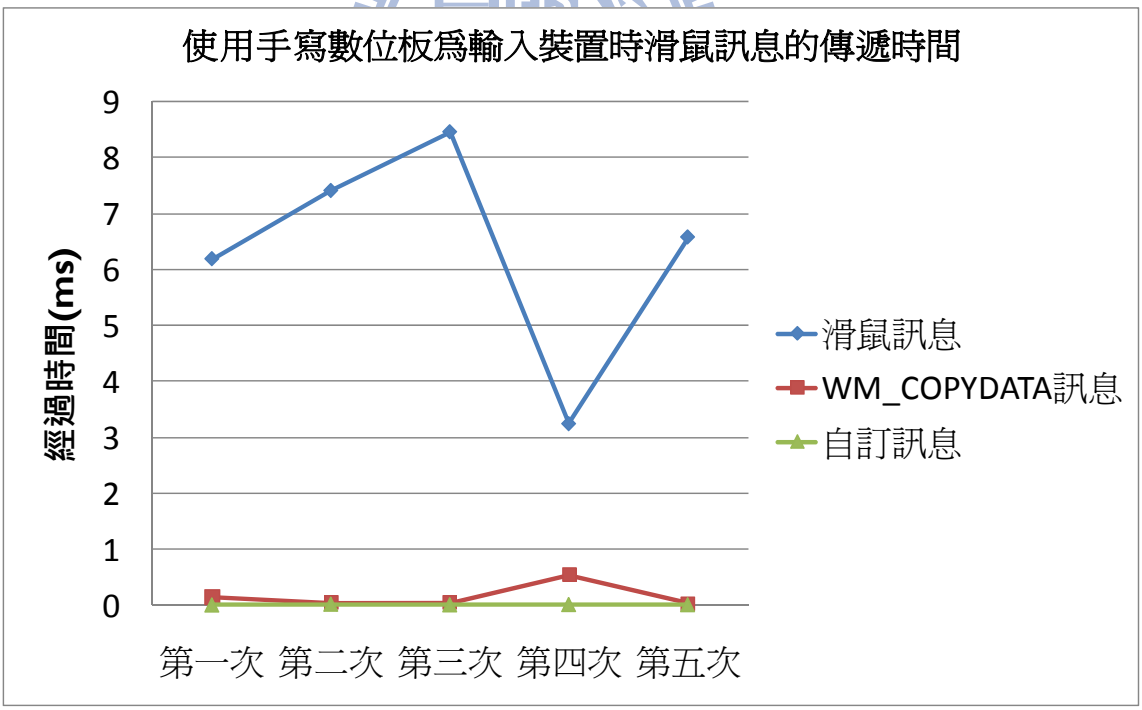

圖 20 使用手寫數位板為輸入裝置，各次訊息傳遞時間的折線圖

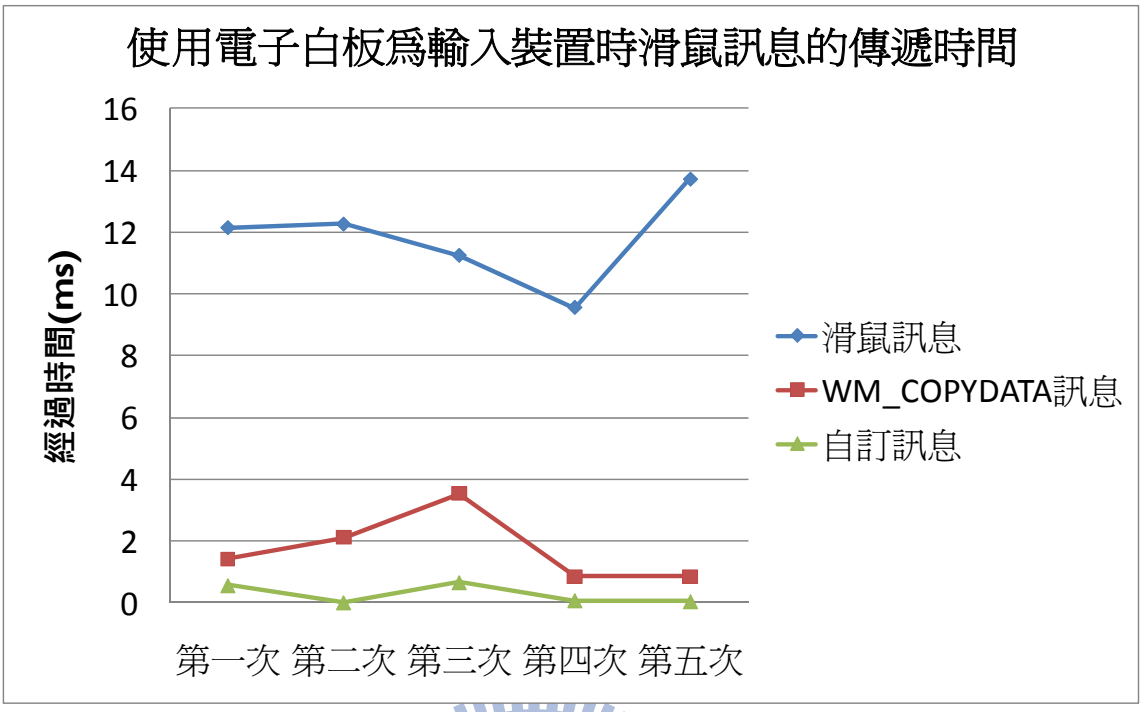

圖 21 使用電子白板為輸入裝置，各次訊息傳遞時間的折線圖

由實驗得知本論文所提出的方法其花費的時間較原有的滑鼠訊息方式更少。由各折線圖中可發現每次的花費時間皆不一樣，係因為 Windows 平台為多工設計，雖然實驗時關閉系統提供的各項服務，但是無法被關閉的服務仍占用了一部分的運算資源，而造成訊息傳遞時間的變化。

人類視覺暫留的時間約為 40 毫秒〔8〕，因此一旦繪圖程式的反應時間超過了 40 毫秒，便會被眼睛查覺。本論文與原有滑鼠訊息取樣使用的時間皆小於 40 毫秒，視覺效果上皆無法被查覺。由上表可知本論文的取樣時間又比原有滑鼠訊息少了約 10 毫秒，可使取樣模組的繪圖程式有較多的時間處理繪圖工作。

4.3.3 實驗:取樣數量分析

以下分析執行時間設定為一秒時，比較繪圖程式可取樣的數量。由於無法控制與滑鼠取樣模組搭配的繪圖程式處理執行繪圖後置處理所花費的時間，因此假設繪圖程式所需的繪圖時間為 A，本論文研發的滑鼠取樣模組取樣時間花費為 B，則可知當使用者觸發滑鼠事件，與程式產生視覺回饋反應完成的時間間隔為 (A + B)，並可知在程式執行一秒內共可取得 1/(A + B) 個樣本點。利用此公式將可計算本論文開發的取樣模組效果，下表為以三種不同進行的取樣數量預測：

表 7 以滑鼠裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數)

繪圖工作時間 (MS) / 滑鼠取樣方式	10	20	30	40	50	60	70	80
滑鼠訊息方式	61	38	28	25	18	15	13	12
WM_COPYDATA 訊息方式	91	48	32	22	20	16	14	12
自訂訊息中斷方式	100	50	33	24	20	17	14	13

表 8 以手寫數位板的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數)

繪圖工作時間 (MS) / 滑鼠取樣方式	10	20	30	40	50	60	70	80
滑鼠訊息方式	61	38	27	22	18	15	13	12
WM_COPYDATA 訊息方式	98	50	33	25	20	17	14	12
自訂訊息中斷方式	100	50	33	25	20	17	14	13

表 9 以電子白板的滑鼠取樣時間與繪圖工作時間的預測取樣點數(單位:點數)

繪圖工作時間(MS) / 滑鼠取樣方式	10	20	30	40	50	60	70	80
滑鼠訊息方式	47	32	24	19	16	14	12	11
WM_COPYDATA 訊息方式	85	46	31	24	19	16	14	12
自訂訊息中斷方式	97	49	33	25	20	17	14	12

以下為各裝置的預測取樣數比較圖：

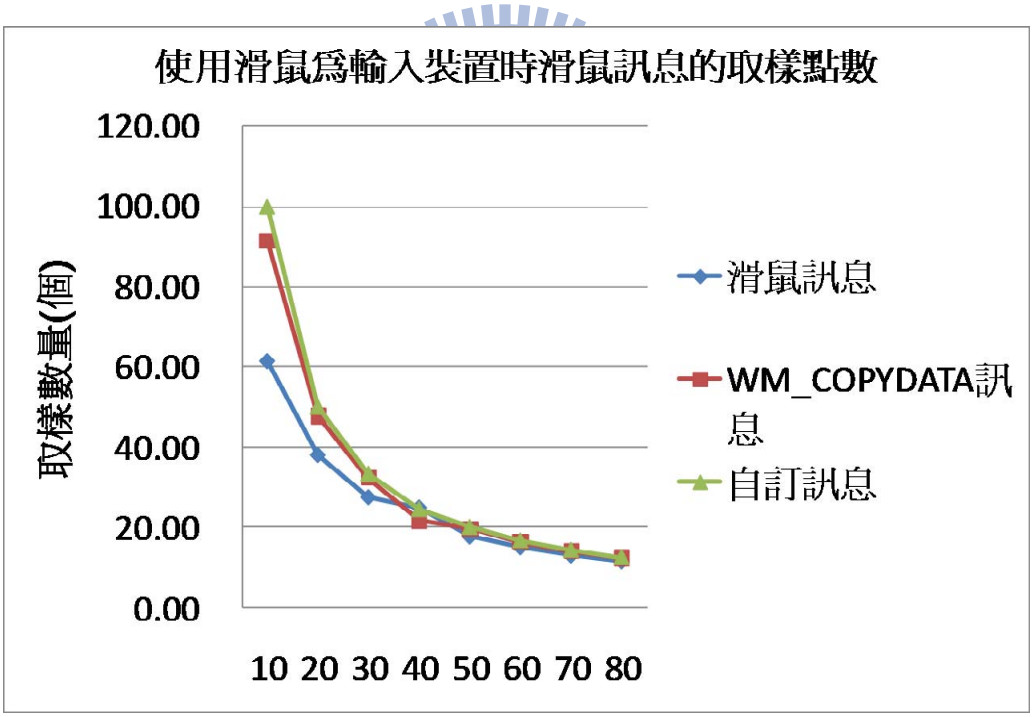

圖 22 以滑鼠裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖

(單位:點數)

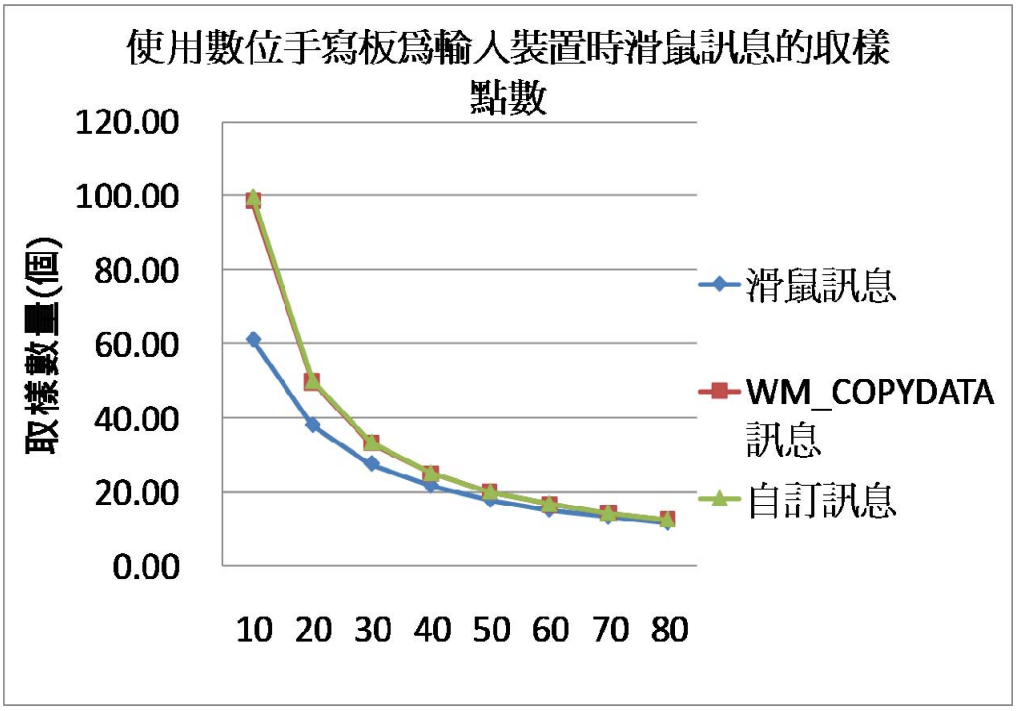

圖 23 以手寫數位板裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖

（單位:點數）

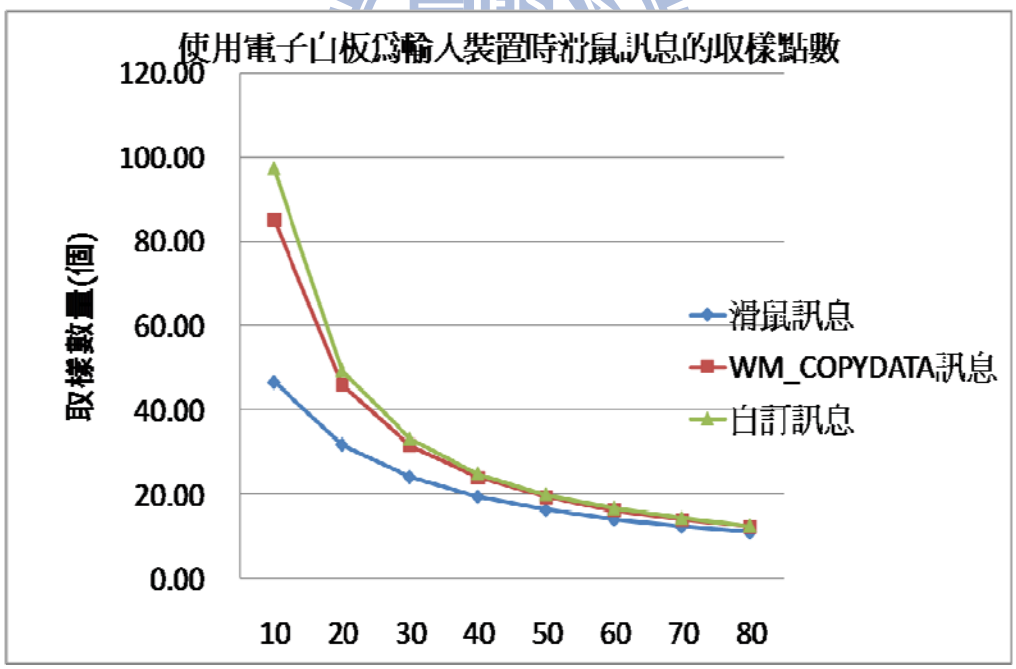

圖 24 以電子白板裝置的滑鼠取樣時間與繪圖工作時間的預測取樣點數比較圖

（單位:點數）

由前述圖表可發現當繪圖模組的繪圖工作時間假定為四十毫秒以下時，使用本取樣模組可取得比原本滑鼠訊息取樣的方式取得點數超過 5 點以上。但是若繪圖工作時間過長時，則效果不明顯。

4.3.4 實驗結論

本論文設計的滑鼠取樣模組確實能減少滑鼠訊息在傳遞的時間和取樣點數。在處理時間上，本論文的方法可比原本滑鼠訊息方式的處理時間少約 10 毫秒。在取樣點數上，若繪圖程式的繪圖工作時間小於 40 毫秒時，滑鼠取樣模組能取得較原本滑鼠訊息方式 5 點以上的效果。

第五章. 應用範例

5.1 應用取樣模組程式展示

5.1.1 滑鼠訊息與自訂訊息中斷方式

以下為加入整合取樣模組前後的畫筆軟體效果對照圖：

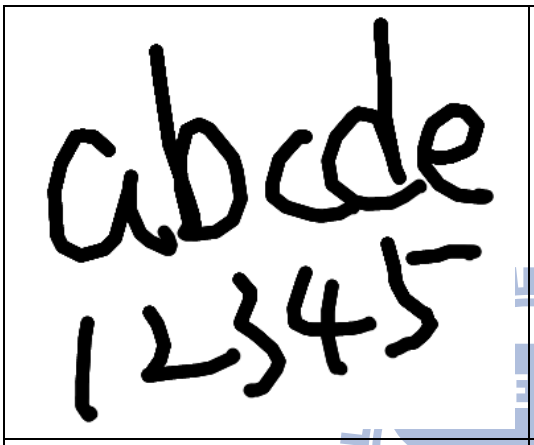	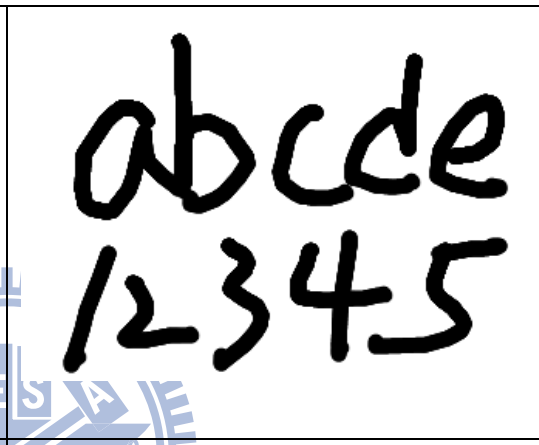
整合取樣模組前	整合取樣模組後

圖 25 加入整合取樣模組前後的畫筆軟體效果對照圖

在整合前的筆跡裡，其中”a”的第一筆因缺少取樣點而有較大的缺口，另外”3”字其因為取樣不足而變成線段相接。

5.1.2 WM_COPYDATA 訊息與自訂訊息中斷方式

以下為本論文兩種方式的畫筆軟體效果對照圖

	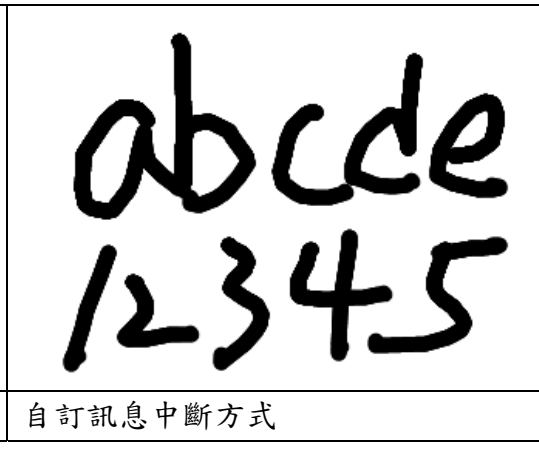
WM_COPYDATA 訊息方式	自訂訊息中斷方式

圖 26 本論文提出兩種方式的畫筆軟體效果對照圖

兩種方式的繪圖效果差異並不大。

第六章. 結論

6.1 總結

本論文所設計的滑鼠取樣模組，可以改善手寫式輸入裝置對於繪圖應用程式的顯示效果，同時擁有不需安裝、即時可使用的特性，更可應用於綠色軟體。另外，可與原有的視窗訊息機制相容，減輕了使應用時對原本程式的修改程度。

本論文的模組在實作上利用原本 Windows 系統所提供的 WM_COPYDATA 訊息，也利用應用程式自訂訊息將滑鼠訊息由佇列視訊息轉換為非佇列式訊息，藉以取得更好的反應速度，並對兩者消耗的時間進行比較。最後在實驗中，對原本的滑鼠訊息和本論文兩種機制取樣效果進行比較，驗證取樣模組設計的方法確實能有效改善滑鼠取樣的效果。

6.2 未來發展方向

以下提出對本論文所設計的滑鼠取樣模組可改善的方向:

可採用 Windows 系統提供的其他行程間通訊方式進行資料傳遞，以期望傳送滑鼠事件相關資料的時間能更加的縮短。

為了使取樣的效果更好，在能安裝的環境下，可以使用過濾式驅動程式的方式進行取樣。

參考文獻或資料

〔1〕 Holzinger, Andreas, “Finger Instead of Mouse: Touch Screens as a means of enhancing Universal Access”, Theoretical Perspectives, Practice, and Experience Lecture Notes in Computer Science, 2003, pp. 387-397.

〔2〕 Glover, D and Miller, D, Averis, D and Door, V., “The interactive whiteboard: a literature survey. Technology, Pedagogy and Education”, Vol. 14, 2005, pp. 155–170.

〔3〕 Microsoft Corporation, “Windows Message”, http://msdn.microsoft.com/en-us/library/ms632590(VS.85).aspx

〔4〕 Charles Petzold, “Programming Windows Fifth Edition”, Microsoft Corporation，1999

〔5〕 Microsoft Corporation, “Windows Hook”, Available HTTP: http://msdn.microsoft.com/en-us/library/dd458650(VS.85).aspx

〔6〕 Jeffrey Richter, “Programming Applications for Microsoft Windows Windows”, 1999

〔7〕 Microsoft, “Windows mouse message”, Available HTTP: http://msdn.microsoft.com/en-us/library/ms645601(VS.85).aspx

〔8〕 Visual staying phenomenon, Available HTTP: http://en.wikipedia.org/wiki/Persistence_of_vision

〔9〕 Green software, Available HTTP: http://en.wikipedia.org/wiki/Portable_application

〔10〕 O’Brien, J (1999). “Management Information Systems – Managing Information Technology in the Internetworked Enterprise”, Boston：Irwin McGraw-Hill, 1999

〔11〕 Microsoft Corporation, “WM_COPYDATA Message”, Available HTTP: http://msdn.microsoft.com/en-us/library/ms649011(VS.85).aspx

〔12〕 Bestwise, Available HTTP: http://www.caidiy.com/