TR-91-010

中文字的放大與縮小系統

中文字的放大與縮小系統

中央研究院
資訊科學研究所

王福吉

指導教授：黃俊雄

中華民國七十九年七月十六日

目 錄

摘要

本系統主要利用影像處理的技巧，將中文字掃描進來。而後再將該中文字的檔案，放到Sun工作站上來處理中文字放大與縮小的問題。所利用的主要觀念，便是利用該中文字選出控制點(Control point)，再利用直線(line)和Bezier曲線(curve)的結合來表示中文字的外形。本文將說明本系統實施的技巧，並說明本系統的使用方法。

第一章 緒論

最近幾年來，國內各界對於電腦中文化的普及一直不遺餘力，其中，中文字體的放大與縮小，亦是許多業界(或個人)所欲待解決的問題。在許多的場合(例如排版印刷、廣告設計、辦公室自動化等)中，常常必須對中文字體做放大或縮小的處理，但是經過處理後的中文字體，經常地會產生鉅齒狀及銳角變鈍角的失眞，使得處理後的中文字體極不美觀。因此，如何使得中文字體在經過放大或縮小之後，仍然能夠保持美觀，便是主要問題之一。

在做中文字體放大與縮小的處理時，亦將有助於將中文字體有效地儲存由於中文字的數量龐大，又有楷書、隸書、行書、草書等字體之分，所以若欲儲存所有不同的中文字體，其所用的儲存空間是非常驚人的，這便是電腦中文化所要解決的主要問題。而在中文字放大與縮小處理的同時，我們可以將一個中文字的表示法，僅用其控制點(control point)的集合來表示，如此一來，中文字的儲存便不是將該字的邊界點儲存，而是僅儲存控制點，因而將節省許多儲存空間。

傳統用來處理中文字放大與縮小的方法，有一種是直接將原先中文字的一個像素(pixel)轉換成爲四個像素(pixel)，然後再將它做平滑(smooth)的工作，但此種方法所產生的結果仍然會有嚴重的鋸齒狀失眞。另外有一種做法是，利用 B_spline數學公式來描述中文字的邊界(boundary)，而控制點的選擇，則是利用 CAD的方法來選擇，若要達到美觀的標準則必須花相當多時間來調整及選擇控制點。

Donald E.knuth曾提出用圖形法來產生美觀的英文字母，此種圖形法也就是他所提出的METAFONT，其中主要的觀念便是利用三次軟樣曲線(cubic spline curve)來表示英文字母的外形。另外，對於英文字母的處理，尚有Adobe Systems公司所提出的PostScript語言。這二種方法，對於英文字母

的放大與縮小，都能夠產生快速且美觀的英文字母，但中文字形比英文字形要來得複雜許多，利用此二種方法來處理中文字形，則是待加以實驗的問題。

本系統所採用方法的主要觀念，乃是利用直線(line)和Bezier曲線結合，來表示中文字的形狀。目前在本系統中，控制點和填黑點(filling point)的選擇均是透過人工的選擇，然而這僅是本系統初步實施的雛型，本系統仍待完成的目標是，將所有控制點和填點的產生均能自動化或半自動化。

第二章 實施方法

第一節 使用設備

本系統的設計主要是在Sun工作站進行，所使用的設備如下：

1.個人電腦(256K RAM及10MB以上硬式磁碟機)
2.全友型影像處理機(MS-300A)
3.全友影像處理系統(EyeStar)
4.Sun 3/60

除了上面所列的主要設備之外，其餘當然尚包含裝在個人電腦中的影像處理機(MS-300A)的介面卡、圖形卡(Graphic card)或彩色/圖形卡(Color/Graphic card)以及印表機等。

第二節 圖形檔案的轉換

由於本系統處理中文字的放大與縮小，主要是在Sun工作站上進行，而利用全友智慧型影像處理機(MS-300A，以下將簡稱爲影像處理機)掃描進來中文字是在個人電腦上進行，這兩種系統的圖形檔案格式並不相同，因此必需做適當的轉換，才能在Sun工作站上顯示出所掃描進來的中文字。

全友影像處理系統(以下簡稱爲EyeStar)的圖形檔案格式爲：啓始位置有256個位元組的檔案標頭，用以說明整個檔案；圖形資料是以一行一行的掃描線來儲存的，在每一行掃描線前面有3個位元組的行標頭，用以儲存有關那一行的資料。檔案標頭和行標頭的格式請參考附錄一(有關EyeStar系統的操作，可參考全友"EyeStar影像處理系統")。

Sun的圖形檔案格式主要分爲三部份：(1)包含8個32位元整數的標頭，用以說明整個檔案，(2)可能爲0個位元組或768個位元組的彩色圖(colormap)值，(3)圖形資料，是以一行一行的方式來儲存。其詳細的格式請參考附錄二。

知道這兩種不同系統的圖形檔案格式之後，便可在二者間作一轉換，轉換的程式(mi2sun.c)如附錄三所示，實例的說明將在第三章敘述。

底下的章節將說明本系統各個主要的程式片斷，而本系統的程式(load256.mouse.c)可參見附錄四。

第三節 圖形檔案的載入與顯示

如何將已經轉換爲Sun工作站的圖形檔案，在Sun工作站上顯示出來呢？底下將從程式 load256.mouse.c中取出某些片斷來說明。在主程式(main program)中可看到：

```
in=fopen (argv[1],"rb");
input_image=pr_load (in,colormap);
block256=pr_region (input_image, X_IN,Y_IN, 256, 256);
for (i=0;i<256;i++)
    for (j=0;j<256;j++)
       img_ary[i][j]=pr_get(block256, i, j,)
                    .
                    .
                    .
                    .
show_boundary(img_ary);
```

圖形檔案的載入主要便是利用pr_load()和pr_region()函數，其詳細的用法說明可參考Pixrect Reference Manual。另外，爲了將該圖形檔案顯示出來，我們利用函數 pr_get()將該圖形檔案放到一個二維陣列(img_ary[][])，然後副程式show_boundary()便是做 "顯示該圖形檔案"的工作。參考附錄四中的副程式show boundary()，可看出它用以顯示該圖形檔案的邊界(boundary)，且主要是利用Pixwin中的函數pw_put()來做到此點。有關Pixwin的詳細原理與用法，可參考SunViewl Programmer's Guide。

第四節 產生按鈕(Button)

本系統共使用了18個按鈕，按鈕的產生主要由副程式create-panel_button_subwindow()來達成。而此副程式主要的片斷則是：

```
panel=window_create (frame, PANEL,
   WIN_BELOW,      canvas_5x5,
   WIN_RIGHT_OF,   canvas,
   0);

panel_create_item(panel, PANEL_BUTTON,
   PANEL_ITEM,           ATTR_ROW(0),
   PANEL_ITEM_Y,         ATTR_COL(0),
   PANEL_LABEL_IMAGE,    panel_button_image(panel,"Select_pt", 0, 0),
   PANEL_NOTIFY_PROC,    Select_a_point,
   0);
```

每一個函數panel_create_item()均產生一個按鈕，詳細的用法請參考SunViewl Programmer's Guide。

第五節 顯示滑鼠(mouse)位置

顯示滑鼠位置，主要由副程式mouse_position_proc()來達成，茲說明於下：

```
void mouse_position_proc(canvas,event)
Canvas  canvas;
Event   *event;
{
        int  mouse_x, mouse_y;

        mouse_x=event_x(event);
        mouse_y=event_y(event);
        if (event_is_button(event) && event_is_down(event)){
           x_pos = mouse_x;
           y_pos = mouse_y;
           show_img_5x5();
           printf("Position=(%d,%d)\n",x_pos,y_pos);
        }
}
```

其中，利用函數event_x()和evnet_y()來得到滑鼠所指的位置，而副程式show_img_5x5()主要的作用，則是顯示滑鼠位置的鄰近的5x5的圖素(pixel)值。

第六節 直線與Bezier曲線

本系統中直線的產生，乃是直接呼叫SunView中Pixwin的函數pw_vector()而得，而Bezier曲線的產生主要由副程式generate_bezier()中的程式片斷而得。Bezier曲線的方程式如下：

$$bezier\_x = x4 * t^3 + 3 * x3 * t^2 * (1-t) + 3 * x2 * t * (1-t)^2 + x1 * (1-t)^3$$

$$bezier\_y = y4 * t^3 + 3 * y3 * t^2 * (1-t) + 3 * y2 * t * (1-t)^2 + y1 * (1-t)^3$$

其中 (x_4, y_4) (x_3, y_3) (x_2, y_2) (x_1, y_1) 分別代表4點的座標，在副程式

generate_bezier()中，bezier_x和bezier_y的值爲上述值再加上0.5，主要便是做四捨五入的效果，以產生更眞確的Bezier曲線。

第七節 放大縮小與旋轉

中文字圖形檔案的放大縮小與旋轉，分別在副程式create_size_menu_proc()和rotate()中進行，而此二個副程式中主要的部份則在於它們所呼叫的二個副程式：transformation_of_array_of_point()和generate_curve_from_ary_pt()。副程式 transformation_of_array_of_point()乃是將原先中文字的座標做一轉換(也就是放大縮小或旋轉之後的中文字)。

此處放大縮小與旋轉的基本做法爲:令未經處理的中文字中心座標爲(SRC_X，SRC_Y)，組成該字的所有控制點的座標陣列爲buf[]，而該字經過處理(放大、縮小或旋轉)之後所放的中心座標爲(DST_X, DST_Y)。考慮座標陣列buf[]中每一點與(SRC_X, SRC_Y)的位移(offset)設爲(offset_x, offset_y)，而放大倍數爲factor，則座標陣列buf[]中的每一點座標均要轉換爲(DST_X +factor*offset_x +0.5,DST_Y+factor *offset_Y +0.5)。同理地，若是經過旋轉θ度，則每一點座標均要轉換爲：

$$(DST\_X + offset\_x * \cos\theta + offset\_y * \sin\theta + 0.5,$$

$$DST\_Y + offset\_x * \sin(-\theta) + offset\_y * \cos\theta + 0.5)$$

在上面的座標轉換中，尾數均加上0.5，主要在做四捨五入的效果。

第八節 中文字的填黑

本系統中文字的填黑乃是利用Boundary_Fill演算法，其方法乃是只要在一封閉區域內選一點開始填黑，直到該封閉區域全部填黑爲止，該副程式boundary_ fill()如下：

```
void boundary_fill ( x, y, value )
int x,y;
int value;
{
   if (pw_get(pw,x,y)==value) return;
   else {
        pw_put (pw,x,y,value);
        boundary_fill (x+1,y,value);
        boundary_fill (x-1,y,value);
        boundary_fill (x,y+1,value);
        boundary_fill (x,y-1,value);
   }
 }
```

第三章 使用說明

本章在說明本系統之使用，底下將配合實例加以說明。 圖形檔案的轉換已在第二章提及，在此將以實例說明。令EyeStar所產生的中文字檔案名稱爲PCfile，而所欲轉換爲Sun工作站上的圖形檔案名稱爲SUNfile，則欲達到此轉換的目的，只要鍵入：

```
mi2sun    PCfile    SUNfile
```

得到轉換完成的圖形檔案SUNfile之後，便可進入本系統，進入的方式只要鍵入：

```
load256.mouse   SUNfile
```

此時會出現圖1的畫面。

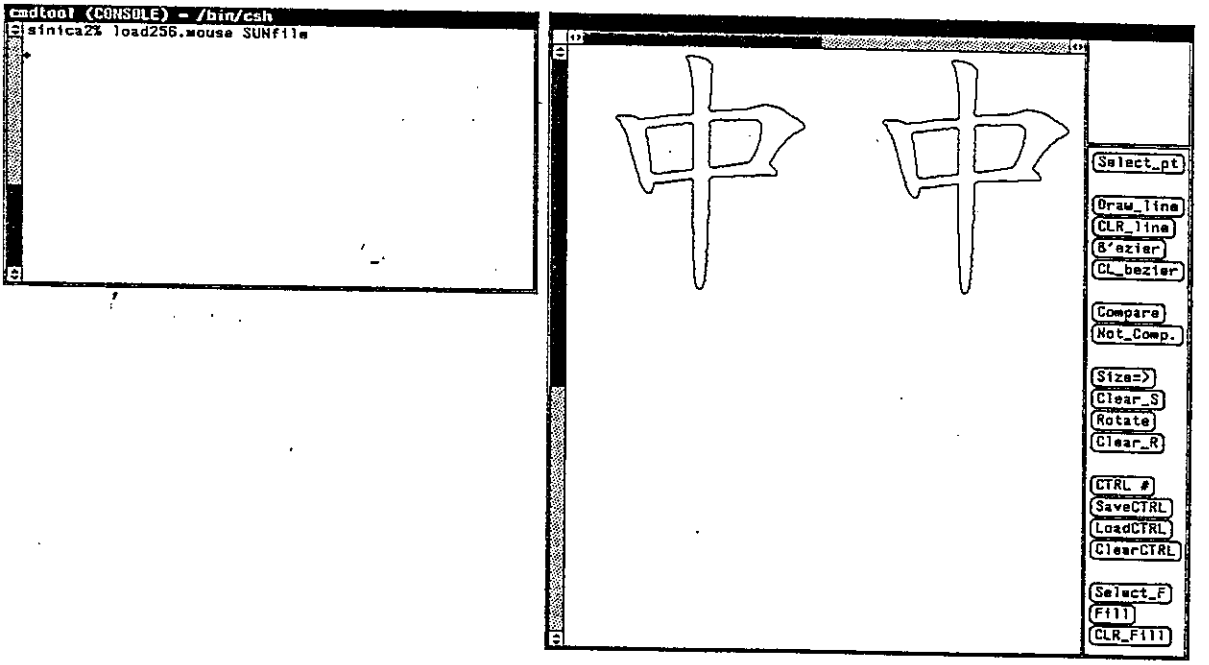


圖1

圖1所示，在CONSOLE鍵入load256.mouse SUNfile，則會開啓另一顯示出中文字的窗(window)。爲了說明方便起見，底下所用的中文字原始的大小爲256x256圖素(pixel)，且以SUNfile檔案裏面的"中"字爲例來做說明(若要以Sunfile檔案裏面的其它字來做處理，只需修改程式中定義的X_IN和Y_IN的值即可)。

圖1中有二個相同的"中"字，這兩個"中"字乃是原先在Sunfile中粗黑字經過處理後而得到的邊界(boundary)。此二"中"字所在的副窗(subwindow)，可顯示的大小爲1024x1024個圖素，此時所看到的僅該大小的左半部，若欲看右半部，可利用窗上的滑桿(scrollbar)而看到。

在窗的右上角有另一副窗，用以顯示滑鼠(mouse)所指位置的5x5個鄰近點，該5x5鄰近點的中心點，即爲滑鼠所指到的位置(註"0"表示白，"1"表示黑)。

其次還有18個按鈕(button)，以下將針對各個按鈕加以說明。

1. Select_pt

選點的方法，便是以最左邊的"中"字爲藍本，沿著此字的邊界點選點，每選擇好一點之後，便按一下此鈕，儘量使所產生的曲線能和原來的"中"字邊界相符。(註:每按一下滑鼠時，便會在CONSOLE出現Position=(x,y)的訊息，該座標(x,y)即爲滑鼠所指的位置，可供選點參考。)

2.Draw_line

若按此鈕，則會將剛剛所選的兩點畫一直線。若只選一點便按此鈕時，則會以前面所選過的最後一點爲起始點，而和該點連線。

3.CLR_line

清除剛剛所畫的直線，並會從儲存選擇點的陣列中去除最後的一個選擇點。

4.Bezier

依所選的四點畫一bezier曲線。若所選的點只有三點，則會以前面所選過的最後一點爲起始點，而和此三點畫一bezier曲線。

5.CL_ bezier

清除剛剛所畫的bezier曲線，並會從儲存選擇點的陣列中去除最後的三個選擇點。

6.Compare

按下此鈕後，則每次在左上角"中"字所畫出的曲線或直線，均會在右下角產生，如下圖2所示。此鈕的功能，主要在用以顯示經過選擇點所產生的"中"字(右下角)。而從左上角的"中"字便可看出右上角與右下角的"中"字的差異處。

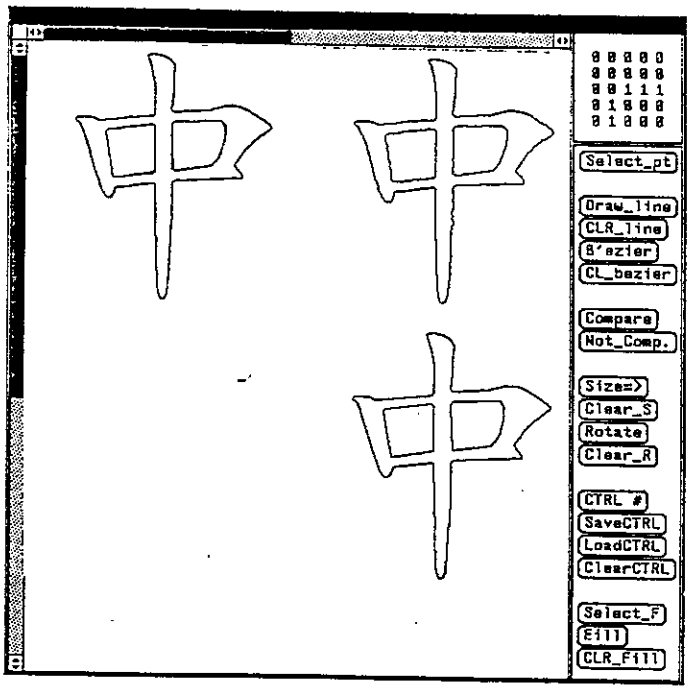


圖2

7.Not_Comp.

按下此鈕後，則不會在右下角產生曲線或直線。

8.Size=>

在本系統中，只有此按鈕與其它按鈕不同。其它按鈕，都是由滑鼠移到該按鈕後，再按滑鼠上左邊的鍵來啓動。而此鈕的啓動，必須按滑鼠的右邊的鍵，此時會出現三個選擇項: Half、Origin和 Double，分別代表欲將該字放大 0.5、1和 2倍。圖3所示爲放大2倍後的"中"字，圖4所示則爲縮小 0.5倍的"中"字。

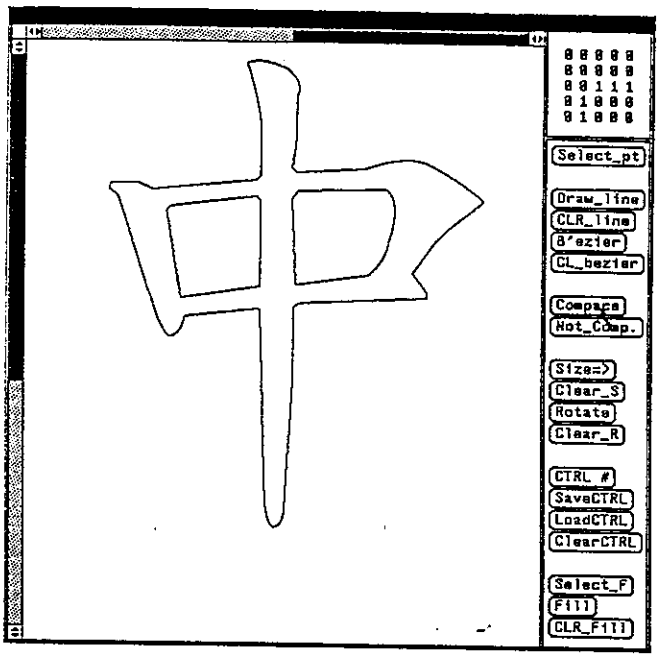


圖3

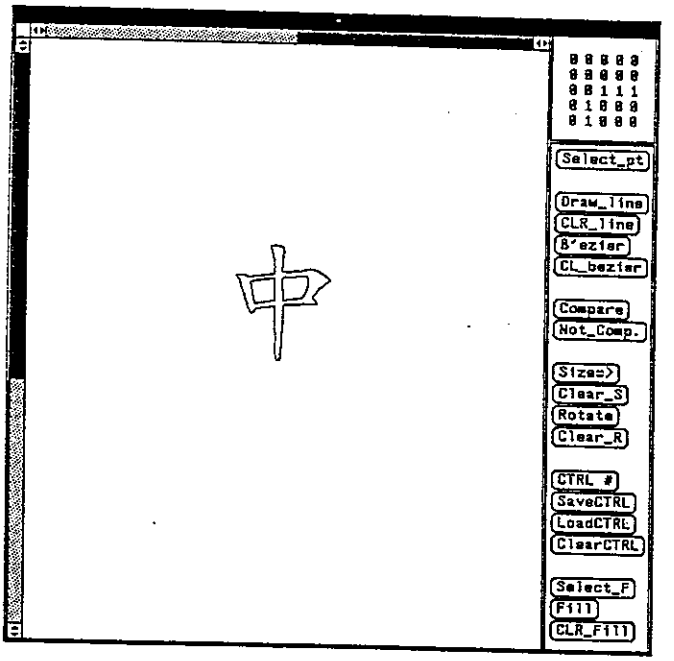


圖4

有一點值得注意的是，當你已完成選點而產生一個中文字時，接著你要將它放大、縮小或旋轉，其結果顯示在圖1中二個"中"字的右半部，也就是說，你必須利用滑桿(scroolbar)移到右半部來看處理後的結果。

9.Clear_S

清除剛剛經過放大或縮小所產生的中文字。

10.Rotate

此鈕的作用，在於旋轉剛剛經過放大或縮小的中文字。當按下此鈕後，會在CONSOLE出現下面訊息:

Enter the degree(integer)you want to rotate:

此時，你要輸入所欲旋轉的度數(需爲任一整數)，接著本系統便會先自動清除原先的中文字之後，再顯示出經過旋轉之後的中文字。圖5 所示爲將放大的"中"字旋轉5度的結果。

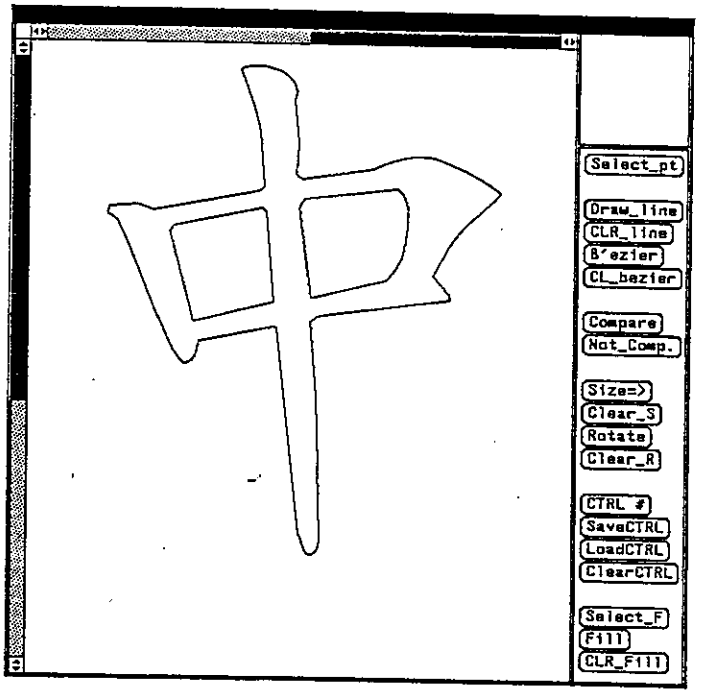


圖5

11.Clear_R

清除剛剛經過旋轉的中文字。

12.CTRL#

在CONSOLE顯示所選的控制點的總數。

13.SaveCTRL

用以儲存前面所選過的控制點。按下此鈕之後，會在CONSOLE出現下面訊息:

Enter the file name you want to save:

這時，要輸入所欲儲存的檔案名稱。若原先已存在有該檔案，則舊的檔案會被覆蓋。經過存後的檔案可參考附錄五，其中每一行的前二個值分別爲控制點的X座標和Y座標，最後一個值做爲標示(tag)，若爲1，表示該點爲直線的起點，若爲2，表示該點爲Bezier曲線的起點。

14.LoadCTRL

用以將原先已經儲存的控制點檔案呼叫出來。當你按下此鈕之後，會在CONSOLE出現下面訊息:

Enter the file name you want to load:

此時，你要輸入所欲呼叫的檔案名稱。此外，它並會自動地清除目前所有的控制點。

15.Select_F

用以選擇所欲填黑區域內部的任何一點。

16.Fill

按下此鈕之後，本系統會依據剛剛所選擇欲填黑的區域加以填黑。因此，若一個中文字有許多分離的封閉區域，則必需在每一個封閉區域內均加以選點，才能將每一封閉區域填黑。有一點值得注意的是，一個欲填黑的區域必需是爲封閉的，否則便會有填黑的錯誤發生。圖6 所示爲"中"字經過填黑處理之後的結果。

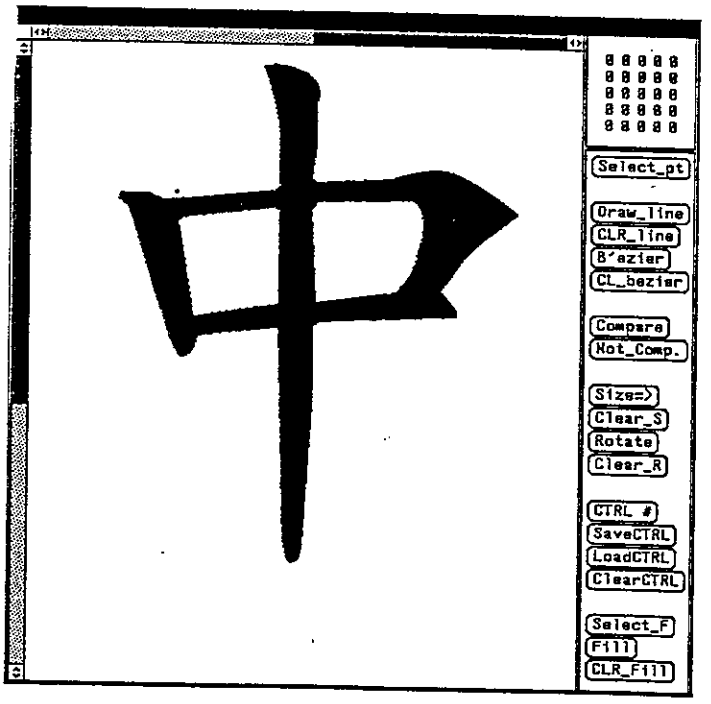


圖6

17.CLR_Fill

用以清除剛剛經過填黑的中文字。

參考資料：

1.Kuang_Yao Chang，Bor_Shenn Jeng，Gan_How Chang and Tsann_Shyong Liu，"A Fast Automatic Chinese Multi_style Characters Zooming and Generating System: A Novel Vector Approach"，CPCOL，1988.

2.鄭國揚、陳克健，"中文字體自動產生之系統模型"，中央研究院資訊科學研究所技術報告TR_82_018，民國七十一年.

3.Donald E. Knuth，TEX and METAFONT:New Directions in Typesetting，American Mathematical Society and Digital Press，1979.

4.Adobe Systems Incorporated，PostScript Language Reference Manual，Addison_wesley Publishing Company，Eighth Printing，December 1987.

5.全友，EyeStar影像處理系統，1987.

6.Sun Microsystems，Pixrect Reference Manual，May 1988.

7.Sun Microsystems，SunView 1 Programmer's Guide，May 1988.

8.Donald Hearn and M. Pauline Baker，Computer Graphics，Prentice_Hall，1986.

附錄一 EyeStar圖形檔案格式

檔案標頭格式

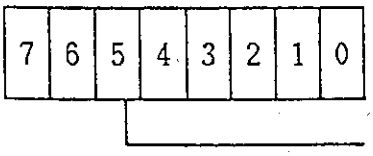
第幾個位元組 (十六進位)	位元組數 (十進位)	內容舉例	說　　明
00 ~ 0F	16	MICROTEK SCANNER	公司名稱和機器 (EyeStar 檔案均如此)
10 ~ 12	3	300	圖形解像度，單位：點數/英吋
13 ~ 17	5	(都是十六進位的00)	保留
18	1	十六進位01	檔案型態 ("1"是圖形檔)
19 ~ 1A	2	(十六進位的E40C)	掃瞄線的總數，High Byte 是 0C，Low Byte 是 E4，(即十六進位的 0CE4，等於十進位的 3300)
1B ~ 1C	2	(十六進位的F609)	High Byte 是 09，Low Byte 是 F6，(即十六進位的 09F6，等於十進位的 2550)
1D ~ FF	227	(都是十六進位的00)	保留

256 bytes 容納檔案標頭，檔案其餘的部分是掃瞄線，
每一行前面都有一個行標頭。

行標頭格式

行標頭中的第一個位元組叫做 "型式(Pattern) 位元組"，包含了掃瞄線的掃瞄模式及資料存到磁碟時是否需要壓縮。在這個位元組當中，實際上只有用到 1 個位元。

型式位元組位元的分配



壓縮位元。
"0" 此列資料儲存時未經壓縮。
"1" 此列資料儲存時已經壓縮。
其餘的七個位元保留未用。

其他的兩個位元組是表示這一列在磁碟上所占的位元組總數。

附錄二 Sun工作站圖形檔案格式

```
/*      @(#)rasterfile.h 1.9 88/02/07 SMI       */

/*
 * Description of header for files containing raster images
 */
struct rasterfile {
        int     ras_magic;              /* magic number */
        int     ras_width;              /* width (pixels) of image */
        int     ras_height;             /* height (pixels) of image */
        int     ras_depth;              /* depth (1, 8, or 24 bits) of pixel */
        int     ras_length;             /* length (bytes) of image */
        int     ras_type;               /* type of file; see RT_* below */
        int     ras_maptype;            /* type of colormap; see RMT_* below */
        int     ras_maplength;          /* length (bytes) of following map */
        /* color map follows for ras_maplength bytes, followed by image */
};
#define RAS_MAGIC       0x59a66a95

        /* Sun supported ras_type's */
#define RT_OLD          0       /* Raw pixrect image in 68000 byte order */
#define RT_STANDARD     1       /* Raw pixrect image in 68000 byte order */
#define RT_BYTE_ENCODED 2       /* Run-length compression of bytes */
#define RT_EXPERIMENTAL 0xffff  /* Reserved for testing */

        /* Sun registered ras_maptype's */
#define RMT_RAW         2
        /* Sun supported ras_maptype's */
#define RMT_NONE        0       /* ras_maplength is expected to be 0 */
#define RMT_EQUAL_RGB   1       /* red[ras_maplength/3],green[],blue[] */

/*
 * NOTES:
 *      Each line of the image is rounded out to a multiple of 16 bits.
 *   This corresponds to the rounding convention used by the memory pixrect
 *   package (/usr/include/pixrect/memvar.h) of the SunWindows system.
 *      The ras_encoding field (always set to 0 by Sun's supported software)
 *   was renamed to ras_length in release 2.0.  As a result, rasterfiles
 *   of type 0 generated by the old software claim to have 0 length; for
 *   compatibility, code reading rasterfiles must be prepared to compute the
 *   true length from the width, height, and depth fields.
 */
```

```
/* mi2sun.c     79.2.23 ~ 79.3.14 */

/* transform image of MICROTEK to that of SUN */
/* Usage: mi2sun MIfile1 SUNfile2
            wherein MIfile1: input file of MICROTEK image,
                    SUNfile2: output file of SUN image        */
/* Note: send binary file from PC to SUN, It must use:
           copy PCfile1 SUNfile2                              */

#include <stdio.h>

#include <rasterfile.h>
#include <sys /types.h>
#include <pixrect /pixrect.h>
#include <pixrect /pr_io.h>

#define RAS_WIDTH       2544          /* 318 * 8 */
#define RAS_HEIGHT      0x0c65
#define RAS_DEPTH       1
#define RAS_LENGTH      1009014
#define RAS_TYPE        1
#define RAS_MAPTYPE     0
#define RAS_MAPLENGTH   0

main(argc, argv)                                              main
int argc;
char *argv[];
{
     FILE *in, *out;
     struct rasterfile *rh;
     colormap_t *colormap = 0;
     int i, cut;
     unsigned char ch;

     in=fopen(argv[1],"rb");
     out=fopen(argv[2],"w+b");

     rh->ras_magic = RAS_MAGIC;
     rh->ras_width = RAS_WIDTH;
     rh->ras_height= RAS_HEIGHT;
     rh->ras_depth = RAS_DEPTH;
     rh->ras_length= RAS_LENGTH;
     rh->ras_type  = RAS_TYPE;
     rh->ras_maptype=RAS_MAPTYPE;
     rh->ras_maplength=RAS_MAPLENGTH;

     pr_dump_header(out, rh, colormap);

     for (i=1; i<=1063178; i++) {
          ch = fgetc(in);
          if (i>=1 && i<=256) continue;
          if ((cut = (i-257) % 322) == 0) continue;
          if ((i-258) % 322 == 0) continue;
          if ((i-259) % 322 == 0) continue;
          if (cut >= 321 && cut <= 321) continue;
          fputc(ch, out);
     }
}
```

```
/* load256.mouse.c          79.7.2 * /
/* Loading a block(256 * 256) from coordinate (X_IN, Y_IN) of input_image * /
/* Print the position (x, y) that mouse point to * /
/* Cretae a 5*5 subwindow to show the neighberhood of mouse position * /
/* Draw a B´ezier curve * /
/* Enlarge & Reduce    79.6.15 O.K.   79.6.19 debug O.K. * /
/* Create BUTTON(of Size) associated with MENU   79.6.20 O.K.* /
/* Rotate 79.6.21 O.K. * /
/* Saving, Loading, Clearing file of control points && Rotate Modified
   79.6.28 O.K.* /
/**Fill   79.7.3 O.K.* /

/* note: "0" for White, "1" for Black * /

#include <stdio.h>
#include <math.h>
#include <sys /types.h>
#include <pixrect /pixrect.h>
#include <pixrect /pr_io.h>

#include <suntool /sunview.h>      /*pixwin* /
#include <suntool /canvas.h>
#include <suntool /panel.h>

#define X_IN 900        /* (X_IN,Y_IN)=(307,415) for First * /
                           /* (X_IN,Y_IN)=(605,417) for Second * /
                           /* (X_IN,Y_IN)=(900,415) for Third * /
#define Y_IN 415


Frame frame;
Canvas canvas;
Canvas canvas_5x5;
Panel panel;
Pixwin *pw;
int compare_flag = 0;

void show_boundary();

main(argc, argv)                                                    main
int argc;
char *argv[];
{
      Pixrect *input_image, *block256;
      FILE *in;
      colormap_t *colormap;
      unsigned char img_ary[256][256];
      int i, j;

      colormap = (colormap_t *) malloc (sizeof(*colormap));
      colormap->type = 0;

      in = fopen(argv[1],"rb");

      input_image = pr_load(in, colormap);

      block256 = pr_region(input_image, X_IN, Y_IN, 256, 256);
      for (i=0; i<256; i++)
            for (j=0; j<256; j++)
                  img_ary[i][j] = pr_get(block256, i, j);

      frame = window_create(NULL, FRAME, 0);

      create_canvas_subwindow();
```

...main

```
    create_canvas_5x5_subwindow();

    create_panel_button_subwindow();


    pw = canvas_pixwin(canvas);

    show_boundary(img_ary);

     /* To show the Center of word_256x256 * /
/*     pw_vector(pw, 0, 0, 255, 255, PIX_SRC, 1);
    pw_vector(pw, 0, 255, 255, 0, PIX_SRC, 1);
    pw_vector(pw, 0, 255, 255, 255, PIX_SRC, 1);
    pw_vector(pw, 255, 255, 255, 0, PIX_SRC, 1);   * /

    window_fit(frame);

    window_main_loop(frame);

    fclose(in);

}
```

create_canvas_subwindow

```
create_canvas_subwindow()
{
    void mouse_position_proc();

    canvas= window_create(frame, CANVAS,
                CANVAS_AUTO_SHRINK,          FALSE,
                WIN_CONSUME_PICK_EVENT,      LOC_DRAG,
                WIN_WIDTH,                   512,
                WIN_HEIGHT,                  600,
                CANVAS_WIDTH,                1024,
                CANVAS_HEIGHT,               1024,
                WIN_VERTICAL_SCROLLBAR,      scrollbar_create(0),
                WIN_HORIZONTAL_SCROLLBAR,    scrollbar_create(0),
                WIN_EVENT_PROC,              mouse_position_proc,
                0);
}
```

create_canvas_5x5_subwindow

```
create_canvas_5x5_subwindow()
{
    canvas_5x5, = window_create(frame, CANVAS,
                WIN_WIDTH,                   100,
                WIN_HEIGHT,                  100,
                0);

}
```

create_panel_button_subwindow

```
create_panel_button_subwindow()
{
    void select_a_point(),
         draw_proc(),
         clear_line_proc(),
         bezier_proc(),
         clear_bezier_proc(),
         compare_proc(),
         not_compare_proc(),
         create_size_menu_proc(),
         clear_previous_curve_proc(),
         rotate_proc(),
         clear_previous_rotate_proc(),
         ctrl_pt_num_proc(),
```

load256.mouse.c

...create_panel_button_subwindow

```
        save_ctrl_pt_proc(),
        load_ctrl_pt_proc(),
        clear_ctrl_pt_proc(),
        select_fill_pt_proc(),
        fill_proc(),
        clear_fill_proc();

panel = window_create(frame, PANEL,
                WIN_BELOW,              canvas_5x5,
                WIN_RIGHT_OF,           canvas,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(0),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Select_pt",0,0),
                PANEL_NOTIFY_PROC,      select_a_point,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(2),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Draw_line",0,0),
                PANEL_NOTIFY_PROC,      draw_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(3),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"CLR_line",0,0),
                PANEL_NOTIFY_PROC,      clear_line_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(4),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"B´ezier",0,0),
                PANEL_NOTIFY_PROC,      bezier_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(5),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"CL_bezier",0,0),
                PANEL_NOTIFY_PROC,      clear_bezier_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(7),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Compare",0,0),
                PANEL_NOTIFY_PROC,      compare_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(8),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Not_Comp.",0,0),
                PANEL_NOTIFY_PROC,      not_compare_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(10),
```

load256.mouse.c

...create_panel_button_subwindow

```
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Size=>",0,0),
                PANEL_EVENT_PROC,       create_size_menu_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(11),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Clear_S",0,0),
                PANEL_NOTIFY_PROC,      clear_previous_curve_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(12),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Rotate",0,0),
                PANEL_NOTIFY_PROC,       rotate_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(13),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Clear_R",0,0),
                PANEL_NOTIFY_PROC,       clear_previous_rotate_proc,
                0);


panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(15),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"CTRL  #",0,0),
                PANEL_NOTIFY_PROC,      ctrl_pt_num_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(16),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"SaveCTRL",0,0),
                PANEL_NOTIFY_PROC,      save_ctrl_pt_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(17),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"LoadCTRL",0,0),
                PANEL_NOTIFY_PROC,      load_ctrl_pt_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(18),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"ClearCTRL",0,0),
                PANEL_NOTIFY_PROC,      clear_ctrl_pt_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(20),
                PANEL_LABEL_IMAGE,      panel_button_image(panel,"Select_F",0,0),
                PANEL_NOTIFY_PROC,      select_fill_pt_proc,
                0);

panel_create_item(panel, PANEL_BUTTON,
                PANEL_ITEM_X,           ATTR_ROW(0),
                PANEL_ITEM_Y,           ATTR_COL(21),
```

```
                    PANEL_LABEL_IMAGE,      panel_button_image(panel,"Fill",0,0),
                    PANEL_NOTIFY_PROC,      fill_proc,
                    0);

    panel_create_item(panel, PANEL_BUTTON,
                    PANEL_ITEM_X,           ATTR_ROW(0),
                    PANEL_ITEM_Y,           ATTR_COL(22),
                    PANEL_LABEL_IMAGE,      panel_button_image(panel,"CLR_Fill",0,0),
                    PANEL_NOTIFY_PROC,      clear_fill_proc,
                    0);


}

void show_boundary(img_in)
unsigned char img_in[][256];
{
    int i, j;
    int row_change, col_change;

    for (i=0; i<256; i++) {
        col_change = 0;
        for (j=0; j<256; j++) {
            if ((col_change == 0) && (img_in[i][j] == 1)) {
                pw_put(pw, i, j, 1);
                pw_put(pw, i+256, j, 1);  /* put secondary boundary */
                col_change = 1;
            }
            if ((col_change == 1) && (img_in[i][j] == 0)) {
                pw_put(pw, i, j-1, 1);
                pw_put(pw, i+256, j-1, 1);  /* put secondary boundary */
                col_change = 0;
            }
        }
    }

    for (j=0; j<256; j++) {
        row_change = 0;
        for (i=0; i<256; i++) {
            if ((row_change == 0) && (img_in[i][j] == 1)) {
                pw_put(pw, i, j, 1);
                pw_put(pw, i+256, j, -1);  /* put secondary boundary */
                row_change = 1;
            }
            if ((row_change == 1) && (img_in[i][j] == 0)) {
                pw_put(pw, i-1, j, 1);
                pw_put(pw, (i-1)+256, j, 1);  /* put secondary boundary */
                row_change = 0;
            }
        }
    }
}


void show_img_5x5();

int x_pos, y_pos;

void mouse_position_proc(canvas, event)
Canvas canvas;
Event *event;
{
    int mouse_x, mouse_y;
```

load256.mouse.c

```
        mouse_x = event_x(event);
        mouse_y = event_y(event);

        if (event_is_button(event) && event_is_down(event)) {
            x_pos = mouse_x;
            y_pos = mouse_y;
            show_img_5x5();
            printf("Position= (%d,%d)\n",x_pos, y_pos);
        }
}

void show_img_5x5()
{
        Pixwin *pw_5x5;
        int i, j;
        int img_5x5[5][5];
        char dot;

        pw_5x5 = canvas_pixwin(canvas_5x5);

        for (i=0; i<5; i++)
            for (j=0; j<5; j++)
                img_5x5[i][j] = 0;

        for (i=0; i<5; i++)
            for (j=0; j<5; j++)
                img_5x5[i][j] = pw_get(pw, (x_pos - 2) + i, (y_pos - 2) + j);

        for (i=0; i<5; i++)
            for (j=0; j<5; j++) {
                if (img_5x5[i][j] == 0) dot = '0';
                else dot ='1';
                pw_char(pw_5x5, 15 + 15*i, 25 + 15*j, PIX_SRC, NULL, dot);
            }
}
#define BSIZE    300      /* size of buf[] */

struct point {
        int x;
        int y;
        int tag;      /* 1: line, 2: b'ezier, otherwise:nothing */
} buf[BSIZE];

static int ptr=0;

void add_point_into_buf();

/* Select a point for the line */
void select_a_point()
{
        struct point *pt;

        pt = (struct point *) malloc(sizeof(*pt));
        pt->x = x_pos;
        pt->y = y_pos;
        pt->tag = 0;

        add_point_into_buf(pt, buf);
}
```

load256.mouse.c

```
/* Add a point into buffer */
void add_point_into_buf(pt, buf)
struct point *pt;
struct point *buf;
{
    if (ptr >= BSIZE) {
        printf("Error: Buf[300] is not enough !!\n");
        exit(1);
    }
    buf[ptr] = *pt;
    ptr++;
}

/* Draw a line */
void draw_proc()
{
    struct point *pt1, *pt2;

    pt1 = (struct point *) malloc(sizeof(*pt1));
    pt2 = (struct point *) malloc(sizeof(*pt2));

    if (ptr-2 < 0) {
        printf("Error: The points you select isn't enough !!\n");
        return;
    }

    *pt1 = buf[ptr - 2];
    *pt2 = buf[ptr - 1];
    pw_vector(pw, pt1->x, pt1->y, pt2->x, pt2->y, PIX_SRC, 1);

    if (compare_flag == 1)
        pw_vector(pw, (pt1->x)+256, (pt1->y)+256, (pt2->x)+256, (pt2->y)+256, PIX_SRC, 1);

    buf[ptr-2].tag = 1;
}

/* Clear the previous drawed line */
void clear_line_proc()
{
    struct point *pt1, *pt2;

    pt1 = (struct point *) malloc(sizeof(*pt1));
    pt2 = (struct point *) malloc(sizeof(*pt2));

    if ((ptr - 2) < 0) {
        printf("Error: The points you select isn't enough !!\n");
        return;
    }

    *pt1 = buf[ptr - 2];
    *pt2 = buf[ptr - 1];

    pw_vector(pw, pt1->x, pt1->y, pt2->x, pt2->y, PIX_SRC, 0);
    if (compare_flag == 1)
        pw_vector(pw, (pt1->x)+256, (pt1->y)+256, (pt2->x)+256, (pt2->y)+256, PIX_SRC, 0);

    ptr--;
}


void generate_bezier();

int xa, ya, xb, yb, xc, yc, xd, yd;
```

load256.mouse.c

```
/* Draw a Bezier curve */
void bezier_proc()
{
    struct point *pt1, *pt2, *pt3, *pt4;
    float t;

    pt1 = (struct point *) malloc(sizeof(*pt1));
    pt2 = (struct point *) malloc(sizeof(*pt2));
    pt3 = (struct point *) malloc(sizeof(*pt3));
    pt4 = (struct point *) malloc(sizeof(*pt4));

    if ((ptr - 4) < 0) {
        printf("The points you select isn't enough !!\n");
        return;
    }

    *pt1 = buf[ptr - 4];
    *pt2 = buf[ptr - 3];
    *pt3 = buf[ptr - 2];
    *pt4 = buf[ptr - 1];

    xa = pt1->x; ya = pt1->y;
    xb = pt2->x; yb = pt2->y;
    xc = pt3->x; yc = pt3->y;
    xd = pt4->x; yd = pt4->y;

    generate_bezier(xa, ya, xb, yb, xc, yc, xd, yd, 1);

    buf[ptr-4].tag = 2;
}


void clear_bezier_proc()
{
    float t;
    int bezier_x, bezier_y;

    generate_bezier(xa, ya, xb, yb, xc, yc, xd, yd, 0);

    ptr = ptr - 3;
}

void compare_proc()
{
    compare_flag = 1;
}

void not_compare_proc()
{
    compare_flag = 0;
}

void transformation_of_array_of_point();
void generate_curve_from_ary_pt();

struct point changed_buf[300];
struct point rotated_buf[300];

void create_size_menu_proc(item, event)
Panel_item item;
Event *event;
{
    Menu menu;
    int i;
```

load256.mouse.c

```
        menu = menu_create(MENU_NCOLS, 1, MENU_STRINGS,
                        "Half", "Origin", "Double", 0, 0);


        if (event_action(event) == MS_RIGHT && event_is_down(event)) {
            compare_flag = 0;
            switch (menu_show(menu, panel, event, 0)) {
                case 1: {
                            transformation_of_array_of_point(buf, changed_buf, 0.5);
                            generate_curve_from_ary_pt(changed_buf, 1);
                            break;
                        }
                case 2: {
                            transformation_of_array_of_point(buf, changed_buf, 1.0);
                            generate_curve_from_ary_pt(changed_buf, 1);
                            break;
                        }

                case 3: {
                            transformation_of_array_of_point(buf, changed_buf, 2.0);
                            generate_curve_from_ary_pt(changed_buf, 1);
                            break;
                        }

            }
        } else
            panel_default_handle_event(item, event);
}

void rotate();

void rotate_proc()
{
        int rotate_degree;

        printf("Enter the degree(integer) you want to rotate: ");
        scanf("%d",&rotate_degree);
        printf("\n");

        rotate(changed_buf, rotated_buf, rotate_degree);
        generate_curve_from_ary_pt(rotated_buf, 1);
}

void clear_previous_curve_proc()
{
        generate_curve_from_ary_pt(changed_buf, 0);
}

#define SRC_X       127
#define SRC_Y       127
#define DST_X       768       /* (512+1024) /2 * /
#define DST_Y       255

void rotate(ary_pt_in, ary_pt_out, degree)
struct point *ary_pt_in, *ary_pt_out;
int degree;
{
        float pie = 3.141592653;
        int num;
        int x, y, offset_x, offset_y;
        float deg;

        clear_previous_curve_proc();
```

load256.mouse.c

```
    num = 0;
    deg = (degree / 180.0) * pie;
    while (num < ptr) {
        x = ary_pt_in[num].x;
        y = ary_pt_in[num].y;

        offset_x = x - DST_X;
        offset_y = y - DST_Y;

        ary_pt_out[num].x
          = DST_X +(offset_x*cos(deg) + offset_y*sin(deg)) + 0.5;
        ary_pt_out[num].y
          = DST_Y +(offset_x*(-sin(deg)) + offset_y*cos(deg)) + 0.5;
        ary_pt_out[num].tag = ary_pt_in[num].tag;
        num++;
    }
}

void clear_previous_rotate_proc()
{
    generate_curve_from_ary_pt(rotated_buf, 0);
}

void transformation_of_array_of_point(ary_pt_in, ary_pt_out, factor)
struct point *ary_pt_in, *ary_pt_out;
float factor;
{
    int num;
    int x, y, offset_x, offset_y;

    num = 0;
    while (num < ptr) {
        x = ary_pt_in[num].x;
        y = ary_pt_in[num].y;

        offset_x = x - SRC_X;
        offset_y = y - SRC_Y;

        ary_pt_out[num].x = DST_X + factor * offset_x + 0.5;
        ary_pt_out[num].y = DST_Y + factor * offset_y + 0.5;
        ary_pt_out[num].tag = ary_pt_in[num].tag;

        num++;
    }
}

void generate_curve_from_ary_pt(ary_pt, value)
struct point *ary_pt;
int value;              /* 0: White, 1: Black */
{
    int num;
    struct point pt1, pt2, pt3, pt4;
    int skip;

    num = 0;
    while ( num < ptr) {
        skip = 1;
        if (ary_pt[num].tag == 1) {
            pt1 = ary_pt[num];
            pt2 = ary_pt[num+1];
            pw_vector(pw, pt1.x, pt1.y, pt2.x, pt2.y, PIX_SRC, value);
        }
        else if (ary_pt[num].tag == 2) {
```

load256.mouse.c

```
                        pt1 = ary_pt[num];
                        pt2 = ary_pt[num+1];
                        pt3 = ary_pt[num+2];
                        pt4 = ary_pt[num+3];
                        generate_bezier(pt1.x, pt1.y, pt2.x, pt2.y, pt3.x, pt3.y, pt4.x, pt4.y, value);
                        skip = 3;
                    }
            num= num + skip;
        }
}

float power();

void generate_bezier(x1, y1, x2, y2, x3, y3, x4, y4, value)
int x1, y1, x2, y2, x3, y3, x4, y4;
int value;
{
        float t;
        int bezier_x, bezier_y;

        for (t=0.0; t<=1.0; t=t+0.005) {
            bezier_x =
               x4*power(t,3) + 3*x3*power(t,2)*(1-t) + 3*x2*t*power(1-t,2) + x1*power(1-t,3) + 0.5;   /* 0.5 for round_off */
            bezier_y =
               y4*power(t,3) + 3*y3*power(t,2)*(1-t) + 3*y2*t*power(1-t,2) + y1*power(1-t,3) + 0.5;

            pw_put(pw, bezier_x, bezier_y, value);
            if (compare_flag == 1)
                pw_put(pw, bezier_x + 256, bezier_y + 256, value);
        }
}

float power(x, y)
float x;
int y;
{
        int i;
        float tmp;
        float z;

        z = x;
        tmp = 1;

        if (y == 0) return(1);

        for (i=1; i<=y; i++) {
            x = z * tmp;
            tmp = x;
        }

        return(x);
}

/* Print the number of control points */
void ctrl_pt_num_proc()
{
        printf("Number of control points = %d\n", ptr);
}

void save_ctrl_pt_proc()
{
        FILE *fp;
        char savefile[20];
        int num;
```

```
        int x, y, tag;


        printf("Enter the file name you want to save: ");
        scanf("%s",savefile);
        printf("\n");
        fp = fopen(savefile, "w");

        num = 0;
        while (num < ptr) {
                x = buf[num].x;
                y = buf[num].y;
                tag = buf[num].tag;
                fprintf(fp, "%d %d %d\n", x, y, tag);
                num++;
        }
        fclose(fp);
}

void load_ctrl_pt_proc()
{
        int i;
        FILE *fp;
        char loadfile[20];
        int x, y, tag;
        int num;

        for (i=0; i<BSIZE; i++) {
                buf[i].x = 0;
                buf[i].y = 0;
                buf[i].tag = 0;
        }

        printf("Enter the file name you want to load: ");
        scanf("%s", loadfile);
        printf("\n");

        if ((fp = fopen(loadfile, "r")) == NULL) {
            printf("Cannot open the file ! Try again.. \n");
            return;
        }

        num = 0;
        while (fscanf(fp, "%d%d%d", &x, &y, &tag) != EOF) {
                buf[num].x = x;
                buf[num].y = y;
                buf[num].tag = tag;
                num++;
        }
        ptr = num;
        fclose(fp);
}

/* Clear the current array of control points */
void clear_ctrl_pt_proc()
{
        int i;

        for (i=0; i<BSIZE; i++) {
                buf[i].x = 0;
                buf[i].y = 0;
                buf[i].tag = 0;
        }
}
```

load256.mouse.c

```
#define FILL_PT_SIZE    10

struct point fill_pt[FILL_PT_SIZE];
static int fill_ptr=0;

void select_fill_pt_proc()
{
     struct point *pt;

     pt = (struct point *) malloc(sizeof(*pt));
     pt->x = x_pos;
     pt->y = y_pos;
     pt->tag = 0;

     if (fill_ptr >= FILL_PT_SIZE) {
          printf("Error: fill_pt[10] is not enough !!\n");
          exit(1);
     }
     fill_pt[fill_ptr] = *pt;
     fill_ptr++;
}

void boundary_fill();

void fill_proc()
{
     int i;
     int x, y;

     for (i=0; i<fill_ptr; i++) {
          x = fill_pt[i].x;
          y = fill_pt[i].y;
          boundary_fill(x, y, 1);
     }
}

void clear_fill_proc()
{
     int i;
     int x, y;

     for (i=0; i<fill_ptr; i++) {
          x = fill_pt[i].x;
          y = fill_pt[i].y;
          boundary_fill(x, y, 0);
     }
     fill_ptr = 0;
}

void boundary_fill(x, y, value)
int x, y;
int value;
{
     if (pw_get(pw, x, y) == value) return;
     else {
          pw_put(pw, x, y, value);
          boundary_fill(x+1, y, value);
          boundary_fill(x-1, y, value);
          boundary_fill(x, y+1, value);
          boundary_fill(x, y-1, value);
     }
}
```

```
47 68 1
60 68 1
67 71 1
117 66 1
119 64 1
119 39 2
118 29 0
114 15 0
112 9 2
118 7 0
125 9 0
131 12 2
137 16 0
138 22 0
136 25 1
136 45 1
134 60 1
136 62 1
169 60 2
185 54 0
194 55 0
200 58 2
214 65 0
220 70 0
226 75 2
221 81 0
209 87 0
201 98 1
192 111 1
199 120 1
199 123 1
138 126 1
135 128 1
135 163 1
131 228 2
130 234 0
129 235 0
126 236 2
123 235 0
123 228 0
122 226 1
119 130 1
118 129 1
83 133 2
82 138 0
80 143 0
75 143 2
68 134 0
68 126 0
65 120 2
58 99 0
53 80 0
50 74 1
46 71 1
47 68 0
76 79 1
117 74 1
119 76 1
119 118 1
81 124 1
74 81 1
76 79 0
```