

第十三章 程式設計的介面

在您開始正式設計程式以前，有一些重大的觀念是要在本章中先認知的。這些觀念將影響到您程式產品的方向與品質。它們包括：

- 何謂人機介面？
- 最佳的人機介面要如何設計？
- 解決人機介面間的技巧
- 參數設計法的正確觀念



13-1 軟體設計的介面類型

在一般情況下，一個 CAD 系統將包括：硬體系統與軟體系統。在硬體系統方面，由於不需要再去設計硬體，所以只要根據 CAD 系統的技術要求，就可以確定 CAD 硬體系統應提供的操作能力以及所需的輸入、輸出設備。而軟體系統的設計就要根據不同的使用要求來進行 CAD 系統的整體規劃與功能設定。依 CAD 系統是否具有人機介面的功能可以分為：「互動式」和「自動式」兩大類。

1. 互動式系統就是指具有人機介面功能的系統。它的作業過程要在人的直接參與下，以人機對話的交互作業方式來進行工作。這種作業方式將以人為中心，所以很適用於那些需要先經由人回答過問題後，才能

處理的工作。因此，所謂的「**互動式**」事實上就是「操作親和力」。換句話說，既然需要有人去輸入，那麼設計一個讓操作者可以輕鬆輸入的介面，就是程式設計師的挑戰。

2. 自動型系統就是指不具有人機對話或很少有人機介面功能的系統。它在作業過程中不需人的參與或只要很少的人工參與。電腦將根據設計師所編寫的程式來自動地完成各個設計步驟。這種作業方式將以電腦為中心，適用於目標明確的流程或設計條件固定的標準工件。例如，自動畫出一個標準的螺絲。

13-2 簡單的傳統型人機介面範例

傳統型的人機介面就表現在詢問使用者的問題上，這也是初學者最常用的方式。其所用的 `getxxx` 類函數我們在 5-1 節中已經學到很多。下面，我們就使用一個簡單的 AutoLISP 來示範這樣的介面。

- 本範例程式名稱：XYSQUARE.LSP（本電子書範例光碟）
- 程式本文如下：

```
(1)(defun C:XYSQUARE (/ x y cp ptm pt1 pt2 pt3 pt4)
(2)(setq pt1 (getpoint "\n 矩形的左下角點: "))
```

說明：`defun` 函數是 LISP 程式的開頭固定語法；在此語法中，函數名稱的格式必須為 `[C:XXX]`，其中所有的字均須大寫；同時，名稱中的 `C:` 一定要永遠固定。至於 `XXX` 的指令名稱部份可以自行規定，但此名稱不得與 AutoCAD 現有指令、內建或外部指令名稱重覆。而 `setq` 則是專門用來定義變數的函數。這兩條程式的意思就是說：將此程式指令的名稱定為：XYSQUARE；同時，希望使用者輸入一左下角點，此點的座標資料將由 `getpoint` 函數擷取到之後，再將之記錄到 `pt1` 變數中。

```
(3)(setq x (getdist "\n 矩形的長: "))
(4)(setq y (getdist "\n 矩形的寬: "))
```

說明：接下來，我們希望使用者輸入矩形的長寬數值；所以，在此我們必須使用 `getdist` 函數；而得到的數值，就分別記錄到 `x` 與 `y` 變數中。這也就是傳統式的人機介面。

```
(5)(setq pt2 (polar pt1 (/ pi 2) y))
(6)(setq pt3 (polar pt2 0 x))
(7)(setq pt4 (polar pt1 0 x))
```

說明：現在，我們就需要來計算各點座標了。我們要使用一個很方便的座標擷取方式，即使用 polar 函數。

由於 pt2 (第二點) 的座標值是相對於 pt1 點座標值的 90 度方向，所以為 $\pi/2$ (須以弧度量表示)。其中，pi 即為 π 。同理，pt3 與 pt4 均以相同方式算出角度方向，再配以適當的距離值即可得到正確的點座標。如圖 13-1 所示：

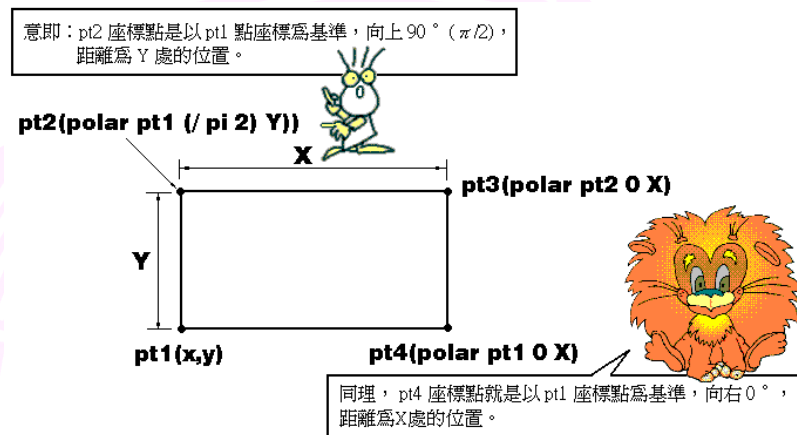


圖 13-1 本範例相關的設計條件圖

在此程式中，我們看見了 $\pi/2$ 的表示方式是：(/ pi 2)。

```
(8)(command "PLINE" pt1 pt2 pt3 pt4 "C")
```

說明：矩形的四個點座標既然已算出，那麼就要使用 command 函數來執行囉....command 函數後所跟的，就是您下的那個 AutoCAD 指令時的實際操作流程。所以，您可以在本例中看到，當下了 PLINE 指令後，該指令就要求操作這回答點座標。此時，此程式就會自動將 pt1、pt2、pt3 與 pt4 等變數所記錄的點座標值回答出來；最後，再輸入了一個「C」(表示 Close 選擇項)，來讓線再畫回起點，即完成此矩形的繪製。

```
(9)(setq cp (list (+ (car pt1) (/ x 2)) (+ (cadr pt1) (/ y 2))))
```

說明：畫出矩形後，還沒完哩！由於欲以矩形中心作為移動的起始點。所以，我們必須先算出中心座標位置值，並將記錄於 cp 變數中。

如此，我們將 X、Y 距離值除以 2 之後，再分別加上由 pt1 中所擷取的 X、Y 座標值，即可組成中心點的座標串列值。最後，再使用那個 list 函數來將分離的數值組合成座標值（即串列）。

(10)(setq ptm (getpoint "\n 新的矩形中心點位置: "))

說明：同前述，請使用者點取欲移動四邊形至那點的所在位置。

(11)(command "MOVE" "L" "" cp ptm)

說明：最後仍使用 command 函數來執行 MOVE 指令，並給予適當的資料來自動作答覆。

(12))

說明：開頭 defun 函數的對稱括號。

● 載入並執行此程式後的操作：

矩形的左下角點: (請點取矩形左下角點)

矩形的長: (請鍵入矩形長數值或以點取兩點來定數值)

矩形的寬: (請鍵入矩形寬數值或以點取兩點來定數值)

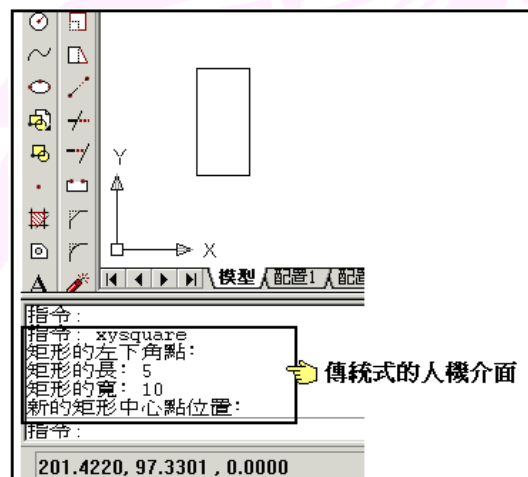


圖 13-2 本範例的人機介面

分析：

● 本範例的優點

當程式執行後，操作非常方便，用戶不需要手動重複的去呼叫繪圖指令 LINE，即可自動畫出符合指定條件的圖形。

● 本範例的缺點

1. 輸入提示文句一條一條出來，無法回頭去修正想再變更其值的設計條件。
2. 無法直接全覽並檢查所輸入的條件。
3. 當遇到設計條件很多的狀況，太多的輸入提示文句會讓操作者感到很煩。
4. 在設計條件很多的情況下，無法一次顯示設計條件的預設值（因為如果可以，就可以省略那些可採用預設輸入的部分而節省輸入時間）。
5. 對不會程式設計的人來說，程式完成後，圖形的畫出結構就固定了，當要改變圖形的外形以適合自己時，除非找人寫程式，否則就無法可想。

其中，第 1、2、3、4 條缺點是可以在程式設計本身的功力範疇內處理的，請參考 13-3-1 或 13-3-2 節。而第 5 條缺點，則需要用「參數設計法」的功能來處理，請參考 13-4 節。

13-3 交談式的人機介面範例

坦白說，提示文句式的人機介面是傳統過時的。現在只用於設計條件簡單，要使用者輸入少的狀況。在正式的程式設計裡，以交談式視窗來設計人機介面互動的方式才是主流。對 VLISP 來說，需要配合 DCL 語言來設計交談式輸入視窗，而對 VBA 來說，可速成的交談式輸入視窗設計正是其優勢。

13-3-1 配合 VLISP 所使用的 DCL 語言

設計交談式視窗的「**交談式控制語言**」(Dialogue Control Language, DCL)，乃是一文字檔 (ASCII 檔)。一個 DCL 檔案的副檔名是 .dcl。單一的 .dcl 檔可以包含一個或是多個交談式視窗的描述，或是它僅包含由其他 .dcl 檔所使用的典型標題以及副組合。換句話說，DCL 檔案將用來設計視窗與視窗內容，可是要讓這個 DCL 檔案「活」起來，卻仍需要相匹配的 LSP 檔案來處理。所以，

在 Visual LISP 裡要用到交談式視窗，就要有一 .dcl 檔案與一對應的 .lsp 檔案。

13-3-1-1 我們可以使用 DCL 來做什麼？

使用 DCL 語言，您將可以：

1. 設計交談式視窗

交談式視窗是以一包含 DCL 語法的文字檔來定義的。該檔案中的 DCL 語法描述將會定義交談式視窗該以什麼樣的方式出現以及它將會包含些什麼；如：按鈕、列表、文字等等。

2. 在您的應用程式中支援交談式視窗

在某些範圍內，交談式視窗的一部份可定義它的行為是如何表現的。例如：被提示可去按哪些按鈕、顯示出一列表，以方便操作者做一選擇等等。

當您在設計一個交談式視窗時，您同時也需要考量到使用者在輸入資料時，將會變化的順序。如此，則會加強這個設計結構，而比起一般地設計要來得較不具「線性」，但可反映出操作者的工作方式；所以，它在經過一些練習後，也可變成較具直覺性的操作方式。

假如在您開始規劃和修正之前，同時也將交談式視窗和應用程式做好細節的計劃，則您將可節省許多時間和省去不少麻煩。

13-3-1-2 您應該準備什麼來撰寫 DCL？

要能順利學習 DCL，您必須有以下的準備：

1. 您必須要懂的一些 AutoLISP 程式設計的觀念；這也是為什麼本章先要您學習 AutoLISP 的原因。為什麼呢？因為交談式視窗的設計與 AutoLISP 是息息相關的；您可以說：AutoCAD 使用 AutoLISP 程式來操縱 DCL。換句話說：DCL 專門是安排視窗配置的，而搭配的 AutoLISP 程式將讓它活動起來。
2. 您應該會使用類似「記事本」、「WordPad」或「Word」這類的文字處理軟體來撰寫 DCL 程式。

準備好之後，您就可以開始以下的實作章節了。

13-3-1-3 踏出 DCL 的第一步

您就要踏出第一步了，因為事實上很簡單；所以，必然是順利的，請安心的踏出去。首先，您要開始做的就是使用文字處理軟體來撰寫其檔案副檔名為 .DCL 的檔案。

在撰寫之前，您心中對要設計的交談式視窗配置一定要先有一梗概。最好的方式就是先用筆在一張白紙上先畫一畫；雖然畫出來的配置將來可能因為種種原因導致與這張手繪圖不太一樣，但也至少不會太離譜。因此，當您要設計自己用的交談式視窗時，請不要忽略了此一重要步驟。在此，當然我有畫啦，但手繪稿畫的太爛了，所以圖 13-3 是完成後的畫面：

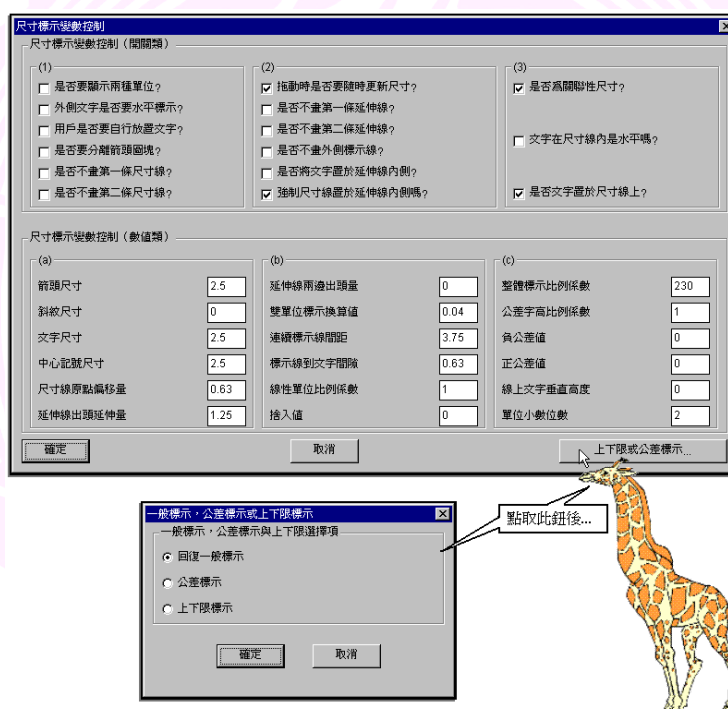


圖 13-3 本範例的完成圖例

如果您問我與手繪稿差多少？老實說，差蠻多的！不過，設計經驗多了，就會愈準。要配置這樣的交談式視窗畫面，它的控制程式都放在一檔案名稱為 SAMPLE.DCL 的檔案中。這個程式版權是屬於我們工作室的，但身為本電子書讀者的您可以修改來使用。

下一節就是這個程式的原文。此程式的檔案依附錄D安裝後應該被放在：

\\03M\\Samples\\Volume4\\

目錄中。您可叫出來搭配以下的說明來研究。以下，我要對您述說的說明將摻雜在原文適當之處；像看戲一樣，一齣一齣的演。

13-3-1-4 SAMPLE.DCL 的原文

1. dcl_settings : default_dcl_settings { audit_level = 0; }

說明：這條程式是在 DCL 檔中第一條首先必須定義的。它將定義審查階層。

2. sample : dialog {

說明：視窗的起始定義是由這裡開始的，視窗的名稱是 sample 以方便以後的呼叫開啓或關閉。這個語法是固定的，沒什麼道理。由 { 符號開始到下面這個 sample 視窗結束的一個對稱的 } 符號，就是 sample 視窗配置的全部內容。

3. label = /*MSG1*/"尺寸標示變數控制";

說明：這是宣告 sample 視窗的標題。它的語法也是固定，遵從它就是了。在雙引號內的就是視窗標題的名稱。

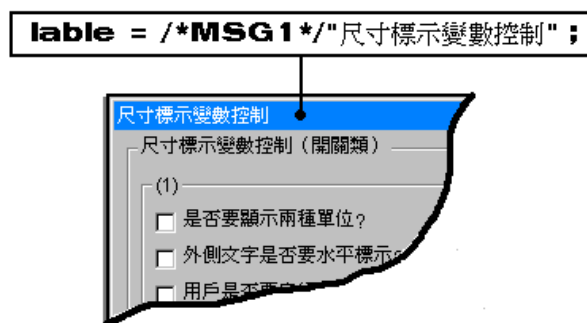


圖 13-4 視窗標題的語法

4. :boxed_column {

說明：因為在視窗內的項目我們希望以線將它們框起來，所以，我們在此使用盒行的語法。由 { 符號開始到下面這個盒行結束的一個對稱的 } 符號，就是此大盒行的全部內容。

5. label = /*MSG2*/"尺寸標示變數控制（開關類）";

說明：這是宣告此大盒行的標題。與上面視窗標題的表示方法一樣；不過，那個 `/*MSG2*/` 是註解，我們將按順序編下來，由 1 開始編。

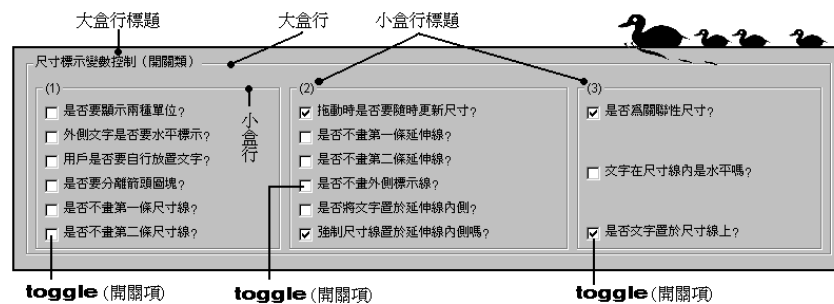


圖 13-5 盒行標題的效果圖例

```
6. :row {
  :boxed_column {
    label = "(1)";

    :toggle {
      label = /*MSG3*/"是否要顯示兩種單位?";
      key = "dimalt";
    }
    :toggle {
      label = /*MSG4*/"外側文字是否要水平標示?";
      key = "dimtoh";
    }
    :toggle {
      label = /*MSG5*/"用戶是否要自行放置文字?";
      key = "dimupt";
    }
    :toggle {
      label = /*MSG6*/"是否要分離箭頭圖塊?";
      key = "dimsah";
    }
    :toggle {
      label = /*MSG7*/"是否不畫第一條尺寸線?";
      key = "dimse1";
    }
    :toggle {
      label = /*MSG8*/"是否不畫第二條尺寸線?";
```

```

        key = "dimse2";
    }
}

```

說明：這一段是此大盒行內再加一小盒行，然後，是小盒行內的項目標題與配置。由於我們希望這 6 個項目能橫列排在一起；因此，我們使用列的語法來設計。然後，我們又考慮到這些項目的特性是開關。因此，我們就挑選方格開關 `toggle` 來定義這些項目的特性。隨同方格開關定義的是標籤 `lable` 與關鍵值 `key`。標籤是用來設定盒行內這一系列項目的名稱，它的用法您應該已不陌生。`key` 則用來定義這些尺寸標示變數的名稱，很重要的；因為將來要靠這些 `key` 來查覺使用者做了哪些改變。此外，每一 `{` 符號都有一對稱的 `}` 符號匹配，請不要拆散這些對佳偶；若您少了一個或多了一個這樣的符號，電腦將替您扮演喬太守，亂配一氣。所以，程式寫完以後，請務必檢查對稱符號。

```

7.      : boxed_column {
        label = "(2)";

        : toggle {
            label = /*MSG9*/"拖動時是否要隨時更新尺寸?";
            key = "dimsho";
        }

        : toggle {
            label = /*MSG10*/"是否不畫第一條延伸線?";
            key = "dimsd1";
        }

        : toggle {
            label = /*MSG11*/"是否不畫第二條延伸線?";
            key = "dimsd2";
        }

        : toggle {
            label = /*MSG12*/"是否不畫外側標示線?";
            key = "dimsoxd";
        }

        : toggle {
            label = /*MSG13*/"是否將文字置於延伸線內側?";
            key = "dimtix";
        }
    }

```

```

: toggle {
    label = /*MSG14*/"強制尺寸線置於延伸線內側嗎?";
    key = "dimtofl";
}
}

```

```

: boxed_column {
    label = "(3)";

```

```

: toggle {
    label = /*MSG15*/"是否為關聯性尺寸?";
    key = "dimaso";
}
: toggle {
    label = /*MSG16*/"文字在尺寸線內是水平嗎?";
    key = "dimtih";
}
: toggle {
    label = /*MSG17*/"是否文字置於尺寸線上?";
    key = "dimtad";
}
}
}

```

說明：上述道理與上一說明相同。我們總共在此盒行內配置了 3 列共 15 個項目。

8. }

說明：這是與上面大盒行起始時 { 符號的對稱符號。表示這部份的盒行內容到此為止。

9. spacer_1;

說明：爲了讓視窗看起來不會太擁擠，在此我們加上一單位的空間器。

spacer_1; 的效果

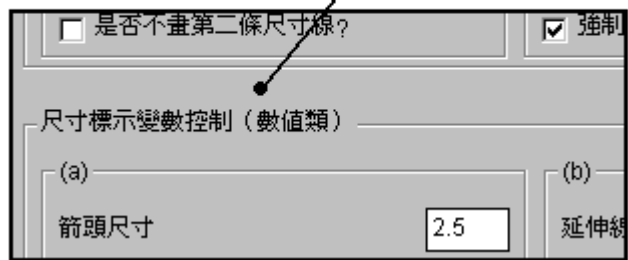


圖 13-6 空間器的效果圖例

10. :boxed_column {

說明：這是第二個盒行起始。

11. label = /*MSG18*/"尺寸標示變數控制 (數值類)";

說明：同理，給這個盒行定義標題。

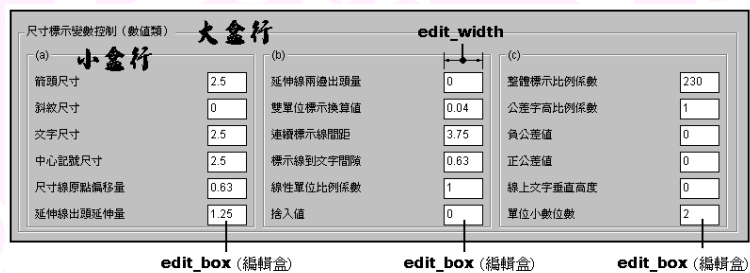


圖 13-7 編輯盒行的效果圖例

12. :row {

:boxed_column {
label = "(a)";

:edit_box {
label = /*MSG19*/"箭頭尺寸";
key = "dimasz";
edit_width = 4;
}

:edit_box {
label = /*MSG20*/"斜紋尺寸";
key = "dimtsz";
edit_width = 4;

```

    }
    : edit_box {
        label = /*MSG21*/"文字尺寸";
        key = "dimtxt";
        edit_width = 4;
    }
    : edit_box {
        label = /*MSG22*/"中心記號尺寸";
        key = "dimcen";
        edit_width = 4;
    }
    : edit_box {
        label = /*MSG23*/"尺寸線原點偏移量";
        key = "dimexo";
        edit_width = 4;
    }
    : edit_box {
        label = /*MSG24*/"延伸線出頭延伸量";
        key = "dimexe";
        edit_width = 4;
    }
}

```

說明：這一段也是此大盒行內小盒行的項目標題與配置。由於，我們也希望這些小盒行項目能橫列排在一起；因此，我們仍使用列(row) 的語法來設計。然後，我們再考慮到這些項目的特性是輸入數值。所以，我們就挑選編輯盒 `edit_box` 來定義這些項目的特性。隨同編輯盒定義的是標籤 `lable`、關鍵值 `key` 與編輯寬度 `edit_width`，標籤仍是用來設定盒行內這一系列項目的名稱。`key` 則是用來定義這些尺寸標示變數的名稱；而 `edit_width` 則用來設定編輯盒的寬度，設定為 4 是經嘗試後修正的結果。仍提醒您不要忘了檢查對稱符號。

```

13.      : boxed_column {
          label = "(b)";

          : edit_box {
              label = /*MSG25*/"延伸線兩邊出頭量";
              key = "dimdle";
          }
      }

```

```
edit_width = 4;
}
: edit_box {
label = /*MSG26*/"雙單位標示換算值";
key = "dimaltf";
edit_width = 4;
}
: edit_box {
label = /*MSG27*/"連續標示線間距";
key = "dimdli";
edit_width = 4;
}
: edit_box {
label = /*MSG28*/"標示線到文字間隙";
key = "dimgap";
edit_width = 4;
}
: edit_box {
label = /*MSG29*/"線性單位比例係數";
key = "dimlfac";
edit_width = 4;
}
: edit_box {
label = /*MSG30*/"捨入值";
key = "dimrnd";
edit_width = 4;
}
}

: boxed_column {
label = "(c)";

: edit_box {
label = /*MSG31*/"整體標示比例係數";
key = "dimscale";
edit_width = 4;
}
: edit_box {
```



```

label = /*MSG32*/"公差字高比例係數";
key = "dimtfac";
edit_width = 4;
}
: edit_box {
label = /*MSG33*/"負公差值";
key = "dimtm";
edit_width = 4;
}
: edit_box {
label = /*MSG34*/"正公差值";
key = "dimtp";
edit_width = 4;
}
: edit_box {
label = /*MSG35*/"線上文字垂直高度";
key = "dimtvp";
edit_width = 4;
}
: edit_box {
label = /*MSG36*/"單位小數位數";
key = "dimdec";
edit_width = 4;
}
}
}

```

說明：上述道理與上一說明相同。我們總共在此大盒行內配置了 3 列共 18 個項目。

14. }

說明：這是與上面大盒行起始時 { 符號的對稱符號。表示這部份的大盒行到此為止。

15. : row {
ok_button;
cancel_button;

```

: button {
    label = /*MSG37*/"上下限或公差標示...";
    key = "lort";
    mnemonic = /*MSG38*/"L";
    fixed_width = true;
}
}

```

說明：接下來，則是定義按鈕的部份。依照習慣，在此部份至少應該至少有 OK 按鈕與 Cancel 按鈕可供操作者選擇儲存或是放棄所做的設定。所以，ok_button 與 cancel_button 應當是沒有問題的固定語法。由於，這些按鈕都是橫列排列；因此，一開始就使用 row 語法。

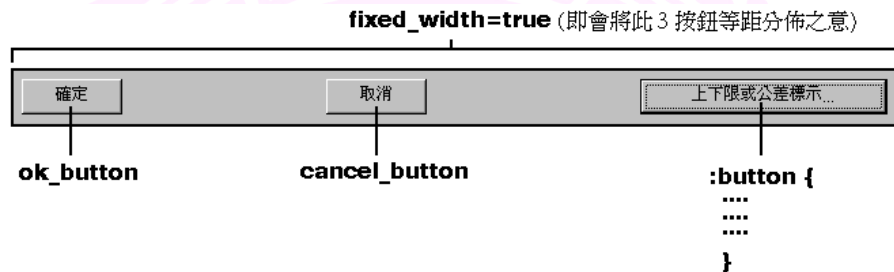


圖 13-8 row 語法的效果圖例

現在，問題可能是再下面的這個自設按鈕部份。這部份在一開始並沒有想到要設計，後來根據使用 AutoCAD 的多年經驗知道：尺寸變數中的公差開關(DIMTOL)與上下公差開關(DIMLIM)兩者是互為開關的。也就是說，如果 DIMTOL 為 ON，則 DIMLIM 就會自動 OFF，反之亦然。所以，就爲了這兩個變數特別製作這個按鈕，讓操作者點取後，還會出現一個視窗來選擇設定之。在 button { 下面的 label 與 key 屬性應該不用再解釋了。記憶鍵值 mnemonic 是表示這個按鈕的鍵盤按鍵值。依此例，使用者可以經由按下 "L" 或 "l" 鍵後出現黑色圓點，再按下空間棒就可選取並執行此按鈕的功能，而不一定要經由滑鼠或數位板的按鍵來點取。fixed_width = true 是表示在配置區變大時，此按鈕的寬度還是固定的。

16. }

說明：這是與上面視窗起始時 { 符號的對稱符號。表示這部份的視窗到此爲止。

17. lort : dialog {

```
label = /*MSG39*/"一般標示，公差標示或上下限標示";
```

```
:boxed_column {
```

```
label = /*MSG40*/"一般標示，公差標示與上下限選擇項";
```

說明：由此起是 `lort` 視窗。不過，這個視窗應該是一個副視窗。它是經由使用者點取「上下限或公差標示...」按鈕時，應該出現的一個視窗。它的樣子配置如下：

radio_button (同心按鈕)

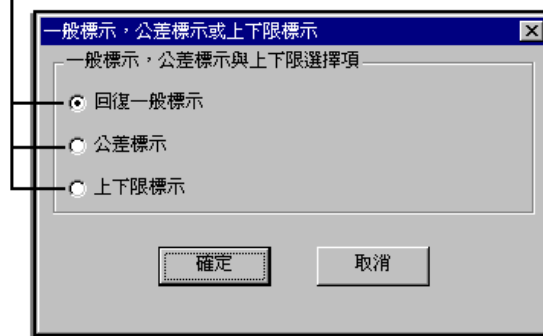


圖 13-9 同心按鈕的效果圖例

```
18. :radio_column {  
    :radio_button {  
        label = /*MSG41*/"回復一般標示";  
        key = "none";  
        value = "1";  
    }  
    :radio_button {  
        label = /*MSG42*/"公差標示";  
        key = "dimtol";  
    }  
    :radio_button {  
        label = /*MSG43*/"上下限標示";  
        key = "dimlim";  
    }  
}
```

說明：這一段是此盒行內項目的標題與配置。由於，我們也希望這三個項

目能直行排在一起；因此，我們仍使用了同心行的語法來設計。為什麼我們要使用同心行的方式來設計呢？因為我們考慮到這些項目是三選一的開關。所以，我們認為使用同心按鈕來定義這些項目的特性是最適當不過的了。跟隨同心按鈕定義的是：您已熟悉的標籤 `lable` 與關鍵值 `key`。此外，您還看到一 `value` 屬性。這個屬性設定為 1 是表示這個按鈕預設是為開啓的狀態；所以，您看到的這個按鈕其中心部份將已經是黑色的了；除非使用者點取其它任兩個按鈕。

19. }

說明：這是上面 `boxed_column` 的對稱符號。

20. `spacer_1;`
 `ok_cancel;`
 `errtile;`

說明：這是設定 OK 與 Cancel 按鈕的另一種固定語法。

21. }

說明：這是與上面視窗起始時 { 符號的對稱符號。表示這部份的副視窗到此為止。

注意：以上的程式是我們已經撰寫並測試好的，如果您是初次撰寫這樣的程式，那麼您一定會希望知道偵錯的過程。根據經驗，要偵測出 `.dcl` 檔的錯誤之處，您可以先在 AutoCAD 指令提示號下依下述方式來鍵入：

指令: `(load_dialog "sample.dcl")`

指令: `(start_dialog)`

不過，這樣的方式可能並不足以讓我們知道問題所在；所以，您最好是擷取下一節的部份 LISP 程式來呼叫此 `.dcl` 檔。在執行的過程中，有些錯誤系統會將錯誤訊息寫至一名稱為 `acad.dce` 檔中，您只要使用文字處理程式來叫出檔案就可以知道錯誤原因，但是若遇到嚴重錯誤就會跳掉而不知問題所在了。如果正常，視窗畫面將會出現但很快的消失(因為您的這個 LISP 檔尚未完整)，您可以在畫面出現完之際趕快按下鍵盤上的 Pause 鍵來觀看位置佈置的缺失。無論如何，這可能不是一種好方法，但在缺乏偵錯系統的情況下，這也是不得已而為之的。或許您有更好的方法也說不定，以上

純就個人經驗有感而發。

這個程式簡單易懂吧！現在，如果您回頭再去看前面幾章保證您有恍然大悟的感覺。就讓我們再繼續「向前行」吧！

13-3-1-5 操縱傀儡的手 — DIMDIA.LSP

與現在我們要談的 AutoLISP 程式比起來，一個 DCL 檔真像布袋戲裡的傀儡。因為真正讓 DCL 檔案「活」起來的是 AutoLISP 程式。

現在，就讓我們來看看這個操控 DCL 程式的原文內容。此檔案依附錄D安裝後應該被放在：

\\03M)Samples\Volume4\

目錄中。其檔名爲 DIMDIA.LSP。

以下就是這個程式的原文。我要對您述說的說明將摻雜在原文適當之處：

1. ;===== 以下是載入時的錯誤檢查 =====

```
(defun ai_abort (app msg)
  (defun *error* (s)
    (if old_error (setq *error* old_error))
    (princ)
  )
  (if msg
    (alert (strcat " Application error: "
                  app
                  "\n\n"
                  msg
                  "\n"
    )
  )
  )
  (exit)
)
```

;;; 這是要查看 AI_UTILS 是否已載入。如果沒有，那麼就嘗試找到它並
;;; 將之載入。如果找不到或找到但無法載入，那麼就立刻放棄載入這個
;;; 檔並且保留 (autoload) 的殘存功能。

```
(cond
  ( (and ai_dcl (listp ai_dcl)))          ; 它已經被載入了

  ( (not (findfile "ai_utils.lsp"))      ; 搜尋之
    (ai_abort "sample"
      (strcat "Can't locate file AI_UTILS.LSP."
        "\n Check support directory.))))

  ( (eq "failed" (load "ai_utils" "failed")) ; 載入之
    (ai_abort "sample" "Can't load file AI_UTILS.LSP"))

)

(if (not (ai_acadapp))                ; 定義一個在 AI_UTILS.LSP 檔中的
  (ai_abort "sample" nil)             ; Nil <msg> 來抑制 ai_abort 的警
)                                     ; 告視窗。
```

說明：AI_UTILS.LSP 檔是 AutoCAD 本身提供的一智慧型公用程式檔。在此檔案中將有許多好用的公用程式指令。例如，自動載入您自行設計的 LISP 程式或錯誤訊息程式集..等，以提供程式設計撰寫者一些工具程式。由於，當我們寫好這個程式以後，將來還要將這個 DIMDIA.LSP 的 LISP 程式在一進入 AutoCAD 2000 時就自動載入。而且，我們還要它自動去判斷是不是這個 DIMDIA.LSP 檔已經載入過了，如果已載入，就直接執行，以節省每一 LISP 程式第二次呼叫過程的執行時間，同時又節省記憶體中的空間。這樣的功能在以前的版本中，我是自己寫一個 LISP 程式來解決的。而現在，AI_UTILS.LSP 中的程式已提供了。因此，像上述的程式語法，您可以將其視為固定的。每一次當您撰寫一新的 LISP 程式時，都可以「依樣畫葫蘆」。

2. ;;;===== 正式程式將由此開始 =====

```
(setq lin_pos '(-1 -1))                ;設定視窗的遇設初始位置(中央)

(defun c:dimdia (/ done onoroff vallist tol lim next_d next_d1)
```


說明：啓動本程式的指令就是 dimdia。在括號中的變數名稱是表示這些變數屬於局部變數，也就是當此程式執行完後，這些變數將被釋放。

```
3. (setq tol 0)
   (setq lim 0)
```

;以下開始載入 sample.dcl 檔案

```
(setq lin_dcl (load_dialog "sample.dcl"))
(if (< lin_dcl 0) (exit))
```

說明：load_dialog 是載入 .dcl 檔的函數語法。如果所欲載入的 .dcl 檔有問題或找不到，則上述 if 語法將判別出並中斷此程式的執行並跳出。

4. ;;===== 控制主視窗的副程式將由此開始 =====

```
(defun main_dia ()
  (setq next_d 5)
  (setq next_d1 nil)
  ;;
  ; 以下將顯示主視窗
  ;;
  (while (< 1 next_d)
    (if (not (new_dialog "sample" lin_dcl "" lin_pos)) (exit)))
```

說明：因為我們會有兩個視窗在運作，所以我們使用 while 函數語法來控制這兩個視窗在一條件下運作。if 函數語法用來判別 sample 視窗(在 sample.dcl 檔中的)是否可以載入，如果可以，就載入；反之，則中斷此程式的執行並跳出。

5. ;
;以下則將 sample 視窗中第一部份的 AutoCAD 變數（開關變數）名稱
;放到以串列型態存在的 onoroff 變數中，以供稍待取用。
;

```
(setq onoroff '("dimalt" "dimtoh" "dimupt" "dimsah" "dimse1"
  "dimse2" "dimsho" "dimsd1" "dimsd2" "dimsoxd" "dimtix"
  "dimtofl" "dimaso" "dimtih" "dimtad"))
```

)

說明：請注意：這些變數名稱的順序必須與在 .dcl 檔中的順序相同。

6.;

;以下則將 sample 視窗中第二部份的 AutoCAD 變數（數值變數）名稱
;放到以串列型態存在的 vallist 變數中，以供稍待取用。
;

```
(setq vallist '("dimasz" "dimtsz" "dimtxt" "dimcen" "dimexo"  
               "dimexe" "dimdle" "dimaltf" "dimdli" "dimgap" "dimlfac"  
               "dimrnd" "dimscale" "dimtfac" "dimtm" "dimtp" "dimtvp"  
               "dimdec"))  
)
```

說明：請注意：這些變數名稱的順序必須與在 .dcl 檔中的順序相同。

7.;

;以下是要將目前 AutoCAD 的預設變數值放到 sample 視窗中
;

```
(mapcar 'set_on_off onoroff)  
(mapcar 'set_value vallist)
```

說明：我們希望 sample 視窗在一開始的運作中，開關變數與數值變數的預設目前值就表現在視窗內。所以，就設計引用了這個 mapcar 函數。由於並沒有適當的函數語法可以作這種轉換，我們就必需自行設計 set_on_off 與 set_value 這兩個副程式來作這個擷取並顯示預設設定的動作。這兩個副程式將在下面說明。

8.;

;接下來，我們要考量如果使用者點取了視窗中的按鈕時要如何處理？
;

```
(action_tile "accept" "(ok)(setq done 1)(done_dialog)")  
(action_tile "cancel" "(done_dialog)(setq done 1)")  
(action_tile "lort" "(show_lort)")
```

說明：action_tile 是專門處理使用者的點取動作的。"accept" 那一列是表示當使用者按下<Enter>鍵時就相當於點取了 OK 鈕（反之亦然），並於執行 OK 副程式的同時，又結束視窗的操作。OK 副程式將在下面說明，它用來執行儲存操作者所作的設定改變。"cancel" 那一列程式是表示放棄操作者所作的設定改變並結束視窗的執行。"lort" 那一列則是表示當使用者點取「上下限或公差標示...」按鈕時，就顯示 show_lort 的副視窗。show_lort 副程式將在下面提及。

9. (cond

```
((= next_d1 3)
 (show_lort)
 (if (/= 3 next_d1)(setq next_d (start_dialog)))
 )
(T (setq next_d (start_dialog)))
)
```

說明：上面這一段程式是設定一執行條件來執行 show_lort 副程式。next_d 與 next_d1 變數都是為條件控制而設計的，以方便判別。

10.)

說明：while 迴圈的終點對稱括號。

11.)

說明：main_dia 副程式的終點對稱括號。

12. ;;===== ok 副程式將由此開始 =====

```
;
;如果使用者點取了 OK 按鈕或 <Enter> 鍵，將執行此程式以儲存使用者
;所作的設定改變
;
```

```
(defun ok ()
```

```
;
;以下將由視窗中將值擷取出以更新對應的 AutoCAD 尺寸標示變數
;
```

```
(mapcar 'get_on_off onoroff)
(mapcar 'get_value vallist)
```

說明：get_on_off 與 get_value 這兩個副程式是用來執行更新動作的。它們將在下面說明。

13. (setq lin_pos (done_dialog 1))

說明：若系統傳回 1 將啟動此 ok 副程式。「lin_pos」將包含視窗的位置。下一個呼叫將會使用此位置。

14.)

說明：ok 副程式的終點對稱括號。

15. ;;===== set_on_off 副程式將由此開始 =====

```
(defun set_on_off (varname)
  (setq dint (getvar varname))
  (set_tile varname (itoa dint))
)
```

說明：這個副程式會將 AutoCAD 的尺寸標示開關變數預設值傳送至視窗中的相對應位置上。其中，set_tile 是關鍵函數。

16. ;;===== set_value 副程式將由此開始 =====

```
(defun set_value (varname)
  (setq dreal (getvar varname))
  (set_tile varname (rtos dreal))
)
```

說明：這個副程式會將 AutoCAD 的尺寸標示數值變數預設值傳送至視窗中的相對應位置上。同樣的，set_tile 是關鍵函數。

17. ;;===== get_on_off 副程式將由此開始 =====

```
(defun get_on_off (varname)
  (setvar varname (atoi (get_tile varname))))
)
```

說明：這個副程式會依使用者在視窗中所改變過的 AutoCAD 尺寸標示開關狀態設定之。其中，get_tile 是關鍵函數。

18. ;;===== get_value 副程式將由此開始 =====

```
(defun get_value (varname)
  (setvar varname (distof (get_tile varname))))
)
```

說明：這個副程式會依使用者在視窗中所改變過的 AutoCAD 尺寸標示數值狀態設定之。同樣的，get_tile 是關鍵函數。

19. ;;===== show_lort 副程式將由此開始 =====

```
(defun show_lort()
  (if (not (new_dialog "lort" lin_dcl)) (exit))
)
```

說明：這也是要測試在 sample.dcl 檔中的 lort 視窗定義是否可載入。如果可以，就載入啟動；反之，則中斷此程式的執行並跳出。

```
20. (action_tile "none" "(setq tol 0)(setq lim 0)")
    (action_tile "dimtol" "(setq tol 1)")
    (action_tile "dimlim" "(setq lim 1)")
    (action_tile "accept" "(ok1)(done_dialog)")
    (action_tile "cancel" "(done_dialog)")
```

說明：與上述相同，我們仍將使用 action_tile 來處理在此副視窗中，操作者的點取動作。由於，我們在此副視窗中使用了同心按鈕架構，所以當使用者點取了三個同心按鈕中的一個時，就會啟動上述的其中之一動作。
"none"、"dimtol" 或 "dimlim" 都是在 sample.dcl 檔中，lort 視窗定義內的各 radio_button 所定義的 key 值。所以，兩方面都必須一致，不能打錯字。在它們之後的是：設定 tol 或 lim 變數值為 1（開）或 0（關），以方便稍後

ok1 的執行設定。"accept" 那一列是表示當操作者按下了鍵盤上的<Enter>鍵時，就相當於點取了 OK 鈕（反之亦然），並執行 OK1 副程式，同時也結束此副視窗的操作。OK1 副程式將在下面說明，它是用來執行儲存操作者所作的設定改變。"cancel" 那一列是表示欲放棄操作者所作的設定改變並結束此副視窗的執行。

```
21. (setq next_d1 (start_dialog))  
    (if (= 3 next_d1)  
        (done_dialog 2)  
    )
```

說明：上面這一段程式是製造一條件來執行返回主視窗的程式。

```
22. )
```

說明：這是 show_lort 副程式的終點對稱括號。

```
23. ;;===== ok1 副程式將由此開始 =====
```

```
(defun ok1 ()  
  (setvar "dimtol" tol)  
  (setvar "dimlim" lim)  
)
```

說明：這段副程式將用來設定操作者在 lort 副視窗中所作的改變。

```
24. ;;===== dimmod_main 副程式將由此開始 =====
```

```
(defun dimmod_main()  
  (setq done 0)  
  (while (/= done 1)  
    (main_dia)  
  )  
)
```

```
;;===== 副程式區至此結束 =====
```

說明：這段副程式是用來在一條件下呼叫上面的 main_dia 副程式的。可能您會認為只有一個副視窗似乎不需使用此副程式。不過，這個程式是用來

為將來有多個複合視窗時，所考慮設計的架構以供未雨綢繆。

25. ;; 設定錯誤函數

```
(setq old_cmd (getvar "cmdecho"); 儲存目前的 cmdecho 系統變數設定
      old_error *error*          ; 儲存目前的錯誤函數
      *error* ai_error           ; 新的錯誤函數
)

(setvar "cmdecho" 0)

(cond
  ( (not (setq dcl_id (ai_dcl "sample")))) ;.DCL 檔是否已載入？
    (T
      (dimmod_main)
    )
  )
)

(setq *error* old_error)
(setvar "cmdecho" old_cmd)
(princ)

(unload_dialog lin_dcl) ;由記憶體中釋放 DCL

;;=====
```

26.)

說明：這個 sample 程式的終點對稱括號。

13-3-1-6 更進階的範例

承 13-2 節範例，若要將之設計成更進階的交談式的人機介面，那就最好用「圖」的方式來導引使用者操作囉！其技巧在於使用了 .sld 的幻燈片檔案。請依下步驟來設計：

- DCL 範例程式名稱：SQUARE.DCL（本電子書範例光碟）

程式本文如下：

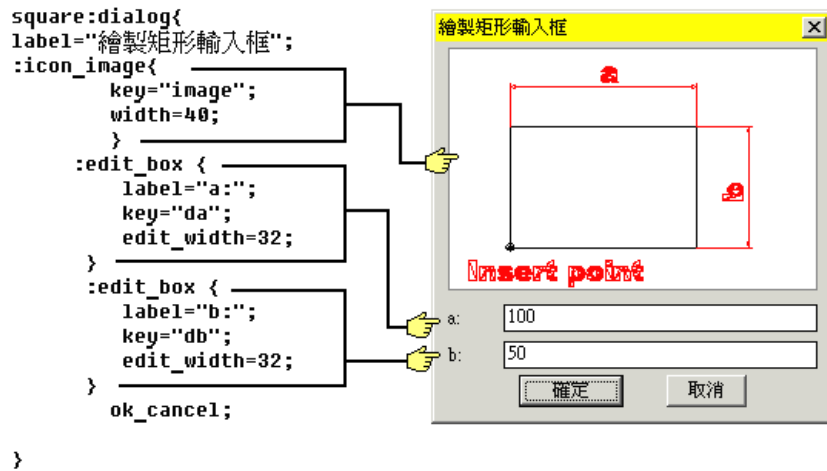


圖 13-10 SQUARE.DCL 的程式本文與效果

- LSP 範例程式名稱：SQUARE.LSP（本電子書範例光碟）
- 配合檔案名稱：SQUARE.SLD（本電子書範例光碟）

程式本文如下：

- (1)(defun c:square (/ Image_ID sldname width height)
- (2)(setvar "cmdecho" 0);設定 "cmdecho" 系統變數以避免執行指令的顯示
- (3)(setq Image_ID (load_dialog "square.dcl"));擷取 DCL 的 ID 碼
- (4)(if (< Image_ID 0);如果 DCL 的影像 ID 碼不存在就跳出
- (5) (exit)
- (6)
- (7)(new_dialog "square" Image_ID); 啓動 DCL 檔
- (8);;開始插入幻燈片檔案的程式段
- (9)(setq sldname (strcat "square.SLD"));擷取幻燈片檔案的名稱
- (10)
- (11)(setq width (dimx_tile "image"))
- (12) height (dimy_tile "image"));擷取幻燈片尺寸
- (13)(start_image "image");初始幻燈片
- (14)(fill_image 0 0 WIDTH HEIGHT 0);填滿幻燈片
- (15)(slide_image 0 0 WIDTH HEIGHT SLDNAME);填滿方框
- (16)(end_image)
- (17);;結束插入幻燈片檔案的程式段

(18)(set_tile "da" "100");設定顯示於輸入視窗裡的 a 預設值為 100

(19)(set_tile "db" "50") ;設定顯示於輸入視窗裡的 b 預設值為 50

(20)(action_tile "accept" "(get_xy)(done_dialog)")

(21)(action_tile "cancel" "(done_dialog)");define button-press action

(22)(start_dialog)

(23)(unload_dialog Image_ID)

(24);;;end dialog

(25)(draw);call draw function

(26)(princ)

(27))

(28)(defun draw (/ pt1 pt2 pt3 pt4 Points Pt1stlen pointDataA pointdata plineobj cp
ptm)

(29)(vl-load-com) ;載入 activeX 支援

(30)(setq *acadObject* (vlax-get-acad-object))

(31)(setq *acadDocument* (vla-get-ActiveDocument *acadObject*))

(32)(setq *mSpace* (vla-get-ModelSpace *acadDocument*))

(33);;;以上三條程式是用來設定一些整體變數

(34)(setq pt1 (getpoint "\n 矩形的左下角點:"))

(35)(setq pt2 (polar pt1 (/ pi 2) db))

(36)(setq pt3 (polar pt2 0 da))

(37)(setq pt4 (polar pt1 0 da))

(38);;;以上三條程式是用來計算出這三個點的方位

(39)(setq Points (mapcar 'float (append pt1 pt2 pt3 pt4)))

(40)(setq Pt1stlen (length Points))

(41)(setq PointDataA

(42) (vlax-make-safearray

(43) vlax-vbDouble

(44) (cons 0 (1- pt1stlen))

(45))

(46))

(47)(vlax-safearray-fill PointDataA Points)

```

(48)(setq PointData (vlax-make-variant
(49)      PointDataA
(50)      (logior vlax-vbarray vlax-vbDouble)
(51)      )
(52))
(53);;;以上程式用來為即將畫出的聚合線改變 4 點為變式

(54)(setq plineobj (vla-Addpolyline *mspace* PointData))
(55)(vla-put-closed plineobj T)
(56);;;以上程式用來畫出聚合線

(57)(setq cp (list (+ (car pt1) (/ da 2)) (+ (cadr pt1) (/ db 2))))
(58)(setq ptm (getpoint "\n 新的矩形中心點位置:"))
(59)(vla-move plineobj (vlax-3d-point cp) (vlax-3d-point ptm));move square to new
position
(60)(princ)
(61))
(62)(defun get_xy ()
(63)  (setq da (atoi (get_tile "da")))
(64)      db (atoi (get_tile "db")))
(65)  )
(66))
(67);;;以上程式用來將聚合線移到指定的新位置上

```

分析：從圖 13-10 您即可看出，這樣的介面不但清楚的以圖示方式來告訴操作者輸入值的意義，同時只要在點取「確定」鈕前，都可以更改 a 與 b 的值，同時常用的 a 與 b 的值也會預設的顯示在輸入框中。這在輸入項目多的時候，對使用者來說就會很方便而實用。

當然，在此範例最後的插入點與新位置的輸入部分，我們仍然使用傳統的問答方式。這部份當然也可以將之併入圖 13-10 裡，不過，這是要留到習題裡做的。

13-3-2 VBA 的作法

對交談式介面來，VBA 就不用那麼麻煩了。在 AutoLISP/VLISP 裡的原始寫法，是利用一個 LISP 程式再搭配一個 DCL 語言所寫的程式來運作的。但是，如果要換成 VBA 來寫，那就簡單了，只要一個 .DVB 程式就搞定了。

13-3-2-1 將 DIMDIA.LSP 與 SAMPLE.DCL 改用 VBA 的寫法

範例程式檔案名稱：DIMVAR.DVB（本電子書範例光碟）

這個「尺寸變數設定大全」程式是要搭配一個自製的視窗來執行的。所以，原先在 13-3-1-4 所談的這個用 DCL 語言所寫的 DCL 檔就是在處理視窗的排列部份，而再使用一個 LISP 程式來控制它。在 VBA 中，它將 VBA 裡的表單功能來取代 DCL 的部份。請依下述步驟進行：

1. 請進入 AutoCAD 裡。
2. 點取「工具(T)」下拉式功能表下，「巨集(M)」選擇項後的「Visual Basic 編輯器(B)」項目。
3. 請依圖 13-11 來建立視窗標題文字：

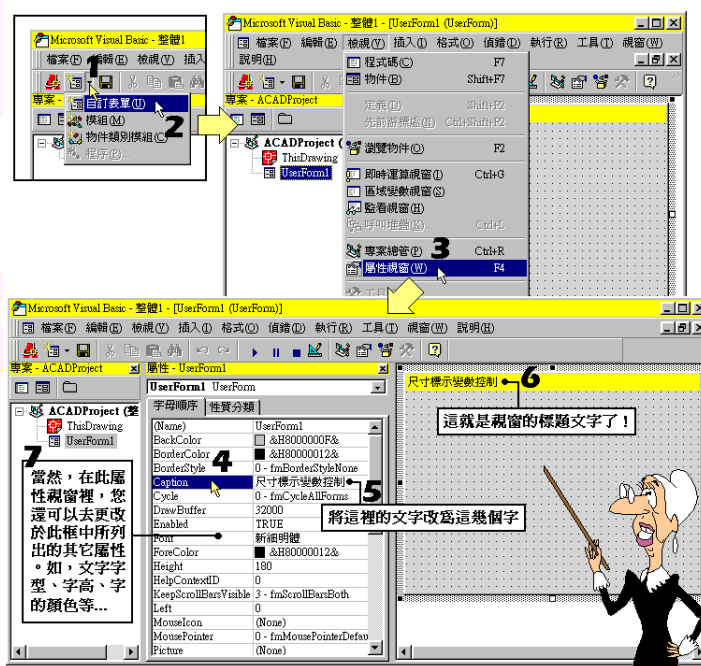


圖 13-11 建立視窗標題文字的操作

4. 繼續，請再依下圖例來建立副視窗標題文字：

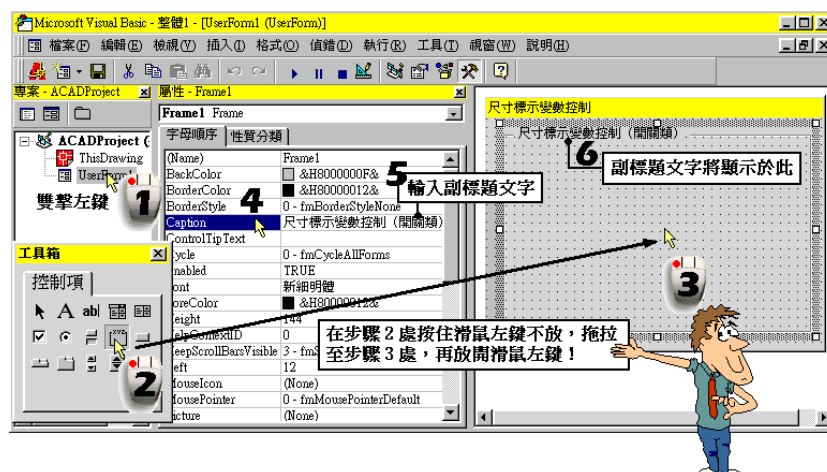


圖 13-12 建立副視窗標題文字的操作

5. 同上操作，再建一個標題文字為 [1] 的框。完成圖如圖 13-13 所示：

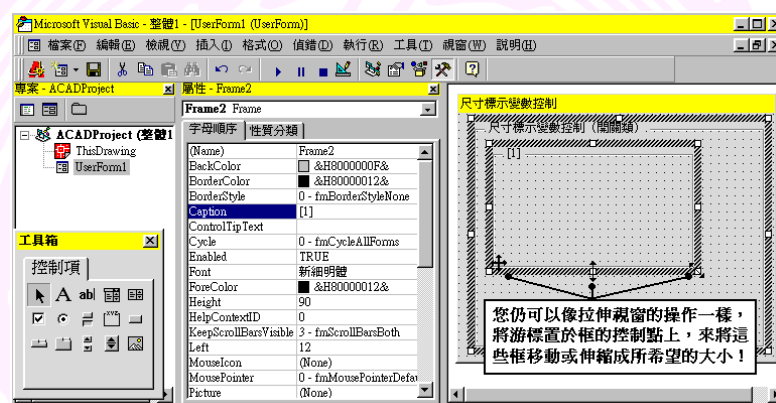


圖 13-13 建立標題文字為 [1] 的框

6. 接下來，就是框內項目的建置了。請依圖 13-14 操作：

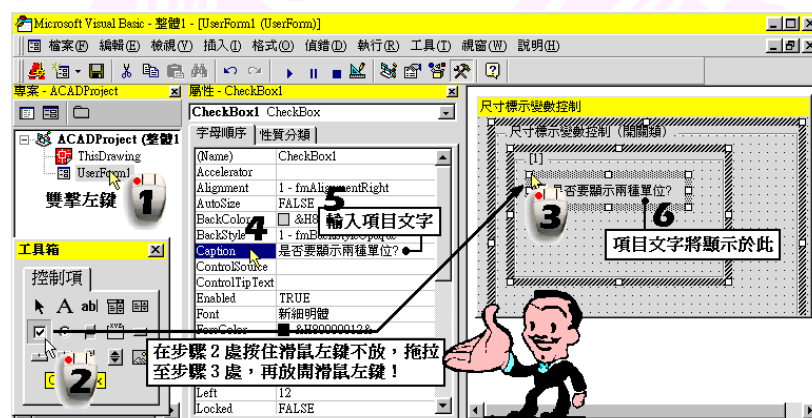


圖 13-14 框內項目的建置操作

7. 同上操作，再將其餘的 5 個開關項建立好。但是完成後，可能各項目的間距並不對齊，請依圖 13-15 修正：

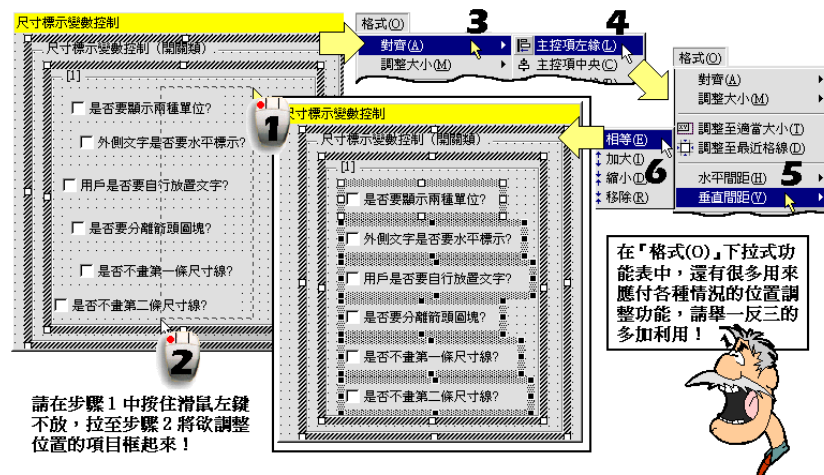


圖 13-15 修正項目間距的操作

8. 同理，再將其餘的 2 個框裡的内容建好。完成圖如圖 13-16 所示：

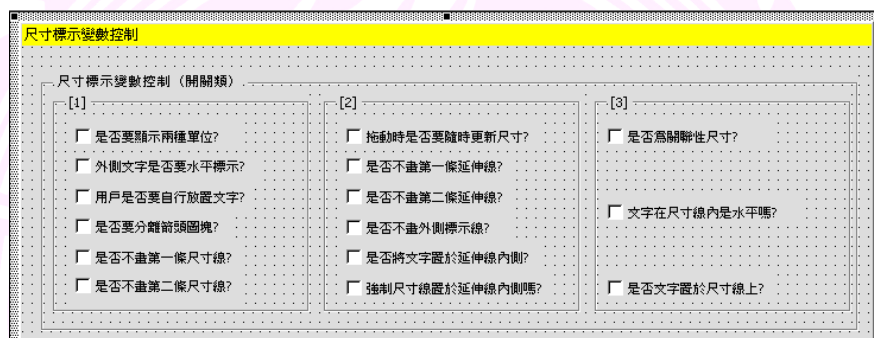


圖 13-16 「開關類」的完成圖示

9. 再下來的「數值類」尺寸變數項目的佈置作法與上述相同，只是在「工具箱」視窗中所選用的元件不同而已。如圖 13-17 所示：

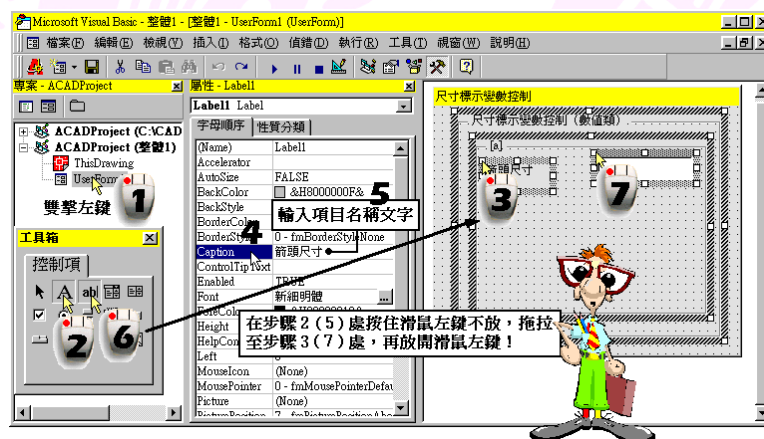


圖 13-17 「數值類」的佈置操作

10. 然後，您可以一樣開視窗來調整位置，也一樣可以開視窗將整個框連同裡面的項目都框起來，再使用「編輯(E)」下拉式功能表下的「複製(C)」

與「貼上(P)」等選項，來複製另二個框([B] 與 [C]) 以及其內的項目（再修改其內的項目名稱就好）。完成後，如圖 13-18 所示：

圖 13-18 「數值類」的完成圖示

11. 接下來，就是「按鈕」功能的製作了。請依圖 13-19 所示操作：

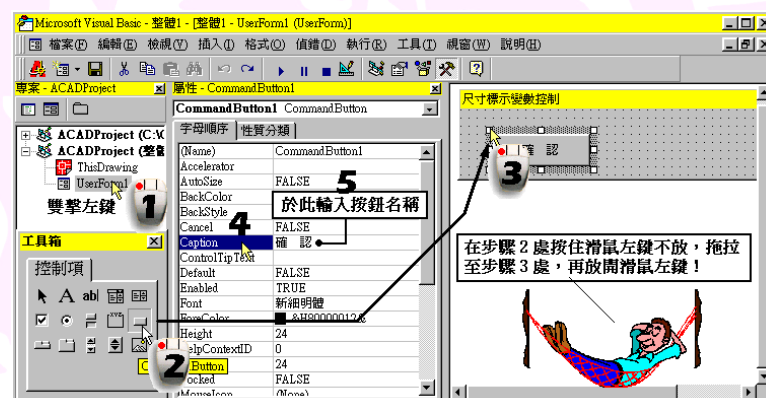


圖 13-19 「數值類」的佈置操作

12. 再重覆上示操作二次將另二個按鈕佈置好。完成後，如圖 13-20 所示：

尺寸標示變數控制 (開關類)

[1] ☐ 是否要顯示兩種單位?
☐ 外側文字是否要水平標示?
☐ 用戶是否要自行放置文字?
☐ 是否要分離箭頭圓圈?
☐ 是否不畫第一條尺寸線?
☐ 是否不畫第二條尺寸線?

[2] ☐ 拖動時是否要隨時更新尺寸?
☐ 是否不畫第一條延伸線?
☐ 是否不畫第二條延伸線?
☐ 是否不畫外側標示線?
☐ 是否將文字置於延伸線內側?
☐ 強制尺寸線置於延伸線內側嗎?

[3] ☐ 是否為關聯性尺寸?
☐ 文字在尺寸線內是水平嗎?
☐ 是否文字置於尺寸線上?

尺寸標示變數控制 (數值類)

[a] 箭頭尺寸:
斜紋尺寸:
文字尺寸:
中心記號尺寸:
尺寸線原點偏移量:
延伸線出頭延伸量:

[b] 延伸線兩邊出頭量:
雙單位標示計算值:
連破標示線間距:
標示線到文字間隙:
線性單位比例係數:
插入值:

[c] 整體標示比例係數:
公差字高比例係數:
負公差值:
正公差值:
線上文字垂直高度:
單位小數位數:

確認 取消 上下限或公差標註

圖 13-20 視窗底下按鈕的完成圖示

13. 由於點取「上下限或公差標註」鈕還會出現一個視窗。因此，請依圖 13-21 來再建一個新的表單，並依前述所學到的方法（但是選取圓形中間有黑點的「選擇項」元件），將之佈置起來：

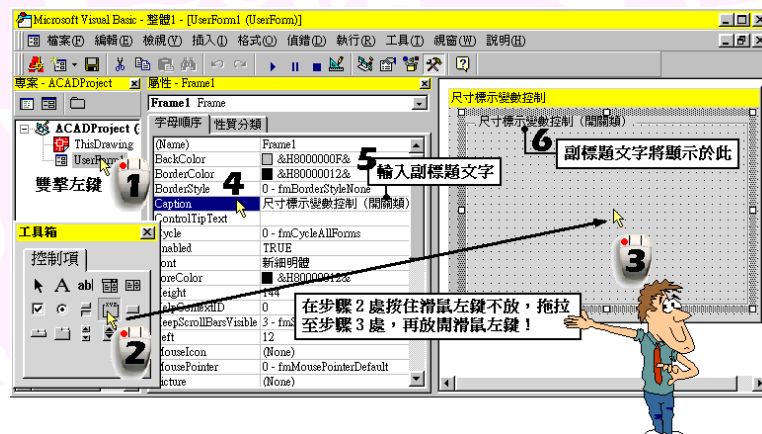


圖 13-21 「上下限或公差標註」的設計

14. 繼續要進行的，則是要將以上的視窗佈置項目來連結 VBA 程式。這部份就是 VBA 程式設計，請依圖 13-22 所示操作：

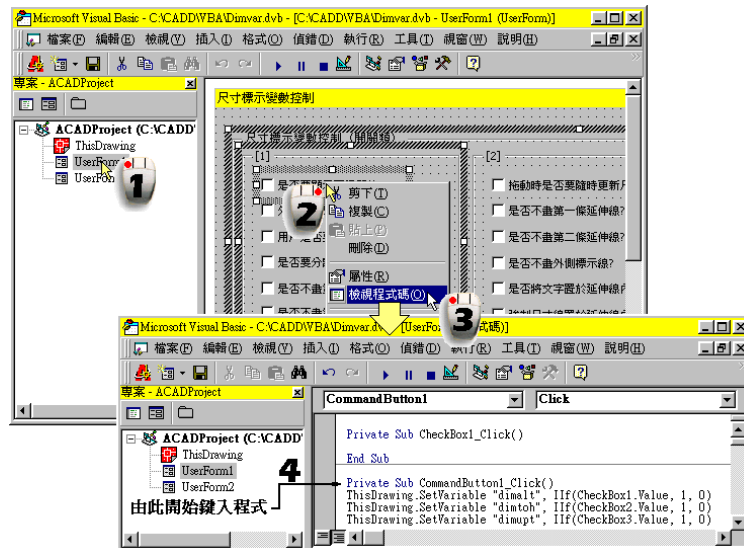


圖 13-22 開關項的 VBA 程式原文視窗

控制於此的所有開關項部份的程式原文是：

```
Private Sub CommandButton1_Click()
ThisDrawing.SetVariable "dimalt", IIf(CheckBox1.Value, 1, 0)
ThisDrawing.SetVariable "dimtoh", IIf(CheckBox2.Value, 1, 0)
ThisDrawing.SetVariable "dimupt", IIf(CheckBox3.Value, 1, 0)
ThisDrawing.SetVariable "dimsah", IIf(CheckBox4.Value, 1, 0)
ThisDrawing.SetVariable "dimse1", IIf(CheckBox5.Value, 1, 0)
ThisDrawing.SetVariable "dimse2", IIf(CheckBox6.Value, 1, 0)
ThisDrawing.SetVariable "dimsho", IIf(CheckBox7.Value, 1, 0)
ThisDrawing.SetVariable "dimsd1", IIf(CheckBox8.Value, 1, 0)
ThisDrawing.SetVariable "dimsd2", IIf(CheckBox9.Value, 1, 0)
ThisDrawing.SetVariable "dimsoxd", IIf(CheckBox10.Value, 1, 0)
ThisDrawing.SetVariable "dimtix", IIf(CheckBox11.Value, 1, 0)
ThisDrawing.SetVariable "dimtofl", IIf(CheckBox12.Value, 1, 0)
ThisDrawing.SetVariable "dimaso", IIf(CheckBox13.Value, 1, 0)
ThisDrawing.SetVariable "dimtih", IIf(CheckBox14.Value, 1, 0)
ThisDrawing.SetVariable "dimtad", IIf(CheckBox15.Value, 1, 0)
```

```
ThisDrawing.SetVariable "dimasz", CDbl(TextBox1.Text)
ThisDrawing.SetVariable "dimtsz", CDbl(TextBox2.Text)
ThisDrawing.SetVariable "dimtxt", CDbl(TextBox3.Text)
ThisDrawing.SetVariable "dimcen", CDbl(TextBox4.Text)
ThisDrawing.SetVariable "dimexo", CDbl(TextBox5.Text)
```

```
ThisDrawing.SetVariable "dimexe", CDb1(TextBox6.Text)
ThisDrawing.SetVariable "dimdle", CDb1(TextBox7.Text)
ThisDrawing.SetVariable "dimaltf", CDb1(TextBox8.Text)
ThisDrawing.SetVariable "dimdli", CDb1(TextBox9.Text)
ThisDrawing.SetVariable "dimgap", CDb1(TextBox10.Text)
ThisDrawing.SetVariable "dimlfac", CDb1(TextBox11.Text)
ThisDrawing.SetVariable "dimrnd", CDb1(TextBox12.Text)
ThisDrawing.SetVariable "dimscale", CDb1(TextBox13.Text)
ThisDrawing.SetVariable "dimtfac", CDb1(TextBox14.Text)
ThisDrawing.SetVariable "dimtm", CDb1(TextBox15.Text)
ThisDrawing.SetVariable "dimtp", CDb1(TextBox16.Text)
ThisDrawing.SetVariable "dimtvp", CDb1(TextBox17.Text)
ThisDrawing.SetVariable "dimdec", CDb1(TextBox18.Text)
Unload Me
End Sub
```

```
Private Sub CommandButton2_Click()
Unload Me
End Sub
```

```
Private Sub CommandButton3_Click()
UserForm2.Show
End Sub
```

```
Private Sub TextBox1_Change()

End Sub
```

```
Private Sub UserForm_Initialize()
CheckBox1.Value = If(ThisDrawing.GetVariable("dimalt"), True, False)
CheckBox2.Value = If(ThisDrawing.GetVariable("dimtoh"), True, False)
CheckBox3.Value = If(ThisDrawing.GetVariable("dimupt"), True, False)
CheckBox4.Value = If(ThisDrawing.GetVariable("dimsah"), True, False)
CheckBox5.Value = If(ThisDrawing.GetVariable("dimse1"), True, False)
CheckBox6.Value = If(ThisDrawing.GetVariable("dimse2"), True, False)
CheckBox7.Value = If(ThisDrawing.GetVariable("dimsho"), True, False)
CheckBox8.Value = If(ThisDrawing.GetVariable("dimsd1"), True, False)
CheckBox9.Value = If(ThisDrawing.GetVariable("dimsd2"), True, False)
```



```
CheckBox10.Value = If(ThisDrawing.GetVariable("dimsoxd"), True, False)
CheckBox11.Value = If(ThisDrawing.GetVariable("dimtix"), True, False)
CheckBox12.Value = If(ThisDrawing.GetVariable("dimtofl"), True, False)
CheckBox13.Value = If(ThisDrawing.GetVariable("dimaso"), True, False)
CheckBox14.Value = If(ThisDrawing.GetVariable("dimtih"), True, False)
CheckBox15.Value = If(ThisDrawing.GetVariable("dimtad"), True, False)
```

```
TextBox1.Text = ThisDrawing.GetVariable("dimasz")
TextBox2.Text = ThisDrawing.GetVariable("dimtsz")
TextBox3.Text = ThisDrawing.GetVariable("dimtxt")
TextBox4.Text = ThisDrawing.GetVariable("dimcen")
TextBox5.Text = ThisDrawing.GetVariable("dimexo")
TextBox6.Text = ThisDrawing.GetVariable("dimexe")
TextBox7.Text = ThisDrawing.GetVariable("dimdle")
TextBox8.Text = ThisDrawing.GetVariable("dimaltf")
TextBox9.Text = ThisDrawing.GetVariable("dimdli")
TextBox10.Text = ThisDrawing.GetVariable("dimgap")
TextBox11.Text = ThisDrawing.GetVariable("dimlfac")
TextBox12.Text = ThisDrawing.GetVariable("dimrnd")
TextBox13.Text = ThisDrawing.GetVariable("dimscale")
TextBox14.Text = ThisDrawing.GetVariable("dimtfac")
TextBox15.Text = ThisDrawing.GetVariable("dimtm")
TextBox16.Text = ThisDrawing.GetVariable("dimtp")
TextBox17.Text = ThisDrawing.GetVariable("dimtvp")
TextBox18.Text = ThisDrawing.GetVariable("dimdec")
```

End Sub

15. 接著是數值輸入框的部份。請依圖 13-23 所示操作：

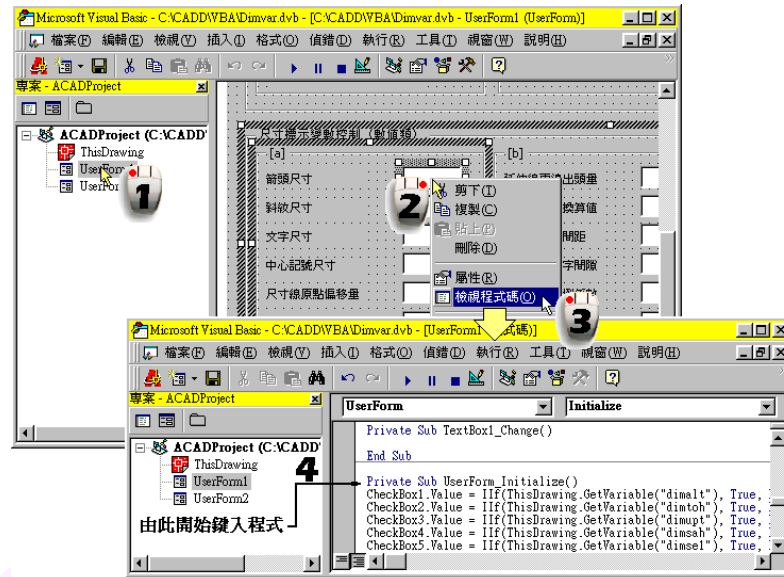


圖 13-23 數值類的 VBA 程式原文視窗

控制於此的所有數值輸入框部份的程式原文是：

```
Private Sub UserForm_Initialize()
    CheckBox1.Value = If(ThisDrawing.GetVariable("dimalt"), True, False)
    CheckBox2.Value = If(ThisDrawing.GetVariable("dimtoh"), True, False)
    CheckBox3.Value = If(ThisDrawing.GetVariable("dimupt"), True, False)
    CheckBox4.Value = If(ThisDrawing.GetVariable("dimsah"), True, False)
    CheckBox5.Value = If(ThisDrawing.GetVariable("dimse1"), True, False)
    CheckBox6.Value = If(ThisDrawing.GetVariable("dimse2"), True, False)
    CheckBox7.Value = If(ThisDrawing.GetVariable("dimsho"), True, False)
    CheckBox8.Value = If(ThisDrawing.GetVariable("dimsd1"), True, False)
    CheckBox9.Value = If(ThisDrawing.GetVariable("dimsd2"), True, False)
    CheckBox10.Value = If(ThisDrawing.GetVariable("dimsoxd"), True, False)
    CheckBox11.Value = If(ThisDrawing.GetVariable("dimtix"), True, False)
    CheckBox12.Value = If(ThisDrawing.GetVariable("dimtofl"), True, False)
    CheckBox13.Value = If(ThisDrawing.GetVariable("dimaso"), True, False)
    CheckBox14.Value = If(ThisDrawing.GetVariable("dimtih"), True, False)
    CheckBox15.Value = If(ThisDrawing.GetVariable("dimtad"), True, False)

    TextBox1.Text = ThisDrawing.GetVariable("dimasz")
    TextBox2.Text = ThisDrawing.GetVariable("dimtsz")
    TextBox3.Text = ThisDrawing.GetVariable("dimtxt")
    TextBox4.Text = ThisDrawing.GetVariable("dimcen")
End Sub
```

```
TextBox5.Text = ThisDrawing.GetVariable("dimexo")
TextBox6.Text = ThisDrawing.GetVariable("dimexe")
TextBox7.Text = ThisDrawing.GetVariable("dimdle")
TextBox8.Text = ThisDrawing.GetVariable("dimaltf")
TextBox9.Text = ThisDrawing.GetVariable("dimdli")
TextBox10.Text = ThisDrawing.GetVariable("dimgap")
TextBox11.Text = ThisDrawing.GetVariable("dimlfac")
TextBox12.Text = ThisDrawing.GetVariable("dimrnd")
TextBox13.Text = ThisDrawing.GetVariable("dimscale")
TextBox14.Text = ThisDrawing.GetVariable("dimtfac")
TextBox15.Text = ThisDrawing.GetVariable("dimtm")
TextBox16.Text = ThisDrawing.GetVariable("dimtp")
TextBox17.Text = ThisDrawing.GetVariable("dimtvp")
TextBox18.Text = ThisDrawing.GetVariable("dimdec")
```

End Sub

16. 同樣的操作，但是針對「確認」按鈕部份，其程式原文是：

```
Private Sub CommandButton1_Click()
ThisDrawing.SetVariable "dimalt", IIf(CheckBox1.Value, 1, 0)
ThisDrawing.SetVariable "dimtoH", IIf(CheckBox2.Value, 1, 0)
ThisDrawing.SetVariable "dimupt", IIf(CheckBox3.Value, 1, 0)
ThisDrawing.SetVariable "dimsah", IIf(CheckBox4.Value, 1, 0)
ThisDrawing.SetVariable "dimse1", IIf(CheckBox5.Value, 1, 0)
ThisDrawing.SetVariable "dimse2", IIf(CheckBox6.Value, 1, 0)
ThisDrawing.SetVariable "dimsho", IIf(CheckBox7.Value, 1, 0)
ThisDrawing.SetVariable "dimSD1", IIf(CheckBox8.Value, 1, 0)
ThisDrawing.SetVariable "dimSD2", IIf(CheckBox9.Value, 1, 0)
ThisDrawing.SetVariable "dimsoxd", IIf(CheckBox10.Value, 1, 0)
ThisDrawing.SetVariable "dimtix", IIf(CheckBox11.Value, 1, 0)
ThisDrawing.SetVariable "dimtofl", IIf(CheckBox12.Value, 1, 0)
ThisDrawing.SetVariable "dimaso", IIf(CheckBox13.Value, 1, 0)
ThisDrawing.SetVariable "dimtih", IIf(CheckBox14.Value, 1, 0)
ThisDrawing.SetVariable "dimtad", IIf(CheckBox15.Value, 1, 0)
```

```
ThisDrawing.SetVariable "dimasz", CDBl(TextBox1.Text)
```

```
ThisDrawing.SetVariable "dimtsz", CDb1(TextBox2.Text)
ThisDrawing.SetVariable "dimtxt", CDb1(TextBox3.Text)
ThisDrawing.SetVariable "dimcen", CDb1(TextBox4.Text)
ThisDrawing.SetVariable "dimexo", CDb1(TextBox5.Text)
ThisDrawing.SetVariable "dimexe", CDb1(TextBox6.Text)
ThisDrawing.SetVariable "dimdle", CDb1(TextBox7.Text)
ThisDrawing.SetVariable "dimaltf", CDb1(TextBox8.Text)
ThisDrawing.SetVariable "dimdli", CDb1(TextBox9.Text)
ThisDrawing.SetVariable "dimgap", CDb1(TextBox10.Text)
ThisDrawing.SetVariable "dimlfac", CDb1(TextBox11.Text)
ThisDrawing.SetVariable "dimrnd", CDb1(TextBox12.Text)
ThisDrawing.SetVariable "dimscale", CDb1(TextBox13.Text)
ThisDrawing.SetVariable "dimtfac", CDb1(TextBox14.Text)
ThisDrawing.SetVariable "dimtm", CDb1(TextBox15.Text)
ThisDrawing.SetVariable "dimtp", CDb1(TextBox16.Text)
ThisDrawing.SetVariable "dimtvp", CDb1(TextBox17.Text)
ThisDrawing.SetVariable "dimdec", CDb1(TextBox18.Text)
Unload Me
End Sub
```

```
Private Sub CommandButton2_Click()
Unload Me
End Sub
```

```
Private Sub CommandButton3_Click()
UserForm2.Show
End Sub
```

```
Private Sub Label1_Click()

End Sub
```

```
Private Sub TextBox1_Change()

End Sub
```

```
Private Sub UserForm_Initialize()
CheckBox1.Value = IIf(ThisDrawing.GetVariable("dimalt"), True, False)
```

```
CheckBox2.Value = IIf(ThisDrawing.GetVariable("dimtoh"), True, False)
CheckBox3.Value = IIf(ThisDrawing.GetVariable("dimupt"), True, False)
CheckBox4.Value = IIf(ThisDrawing.GetVariable("dimsah"), True, False)
CheckBox5.Value = IIf(ThisDrawing.GetVariable("dimse1"), True, False)
CheckBox6.Value = IIf(ThisDrawing.GetVariable("dimse2"), True, False)
CheckBox7.Value = IIf(ThisDrawing.GetVariable("dimsho"), True, False)
CheckBox8.Value = IIf(ThisDrawing.GetVariable("dimsd1"), True, False)
CheckBox9.Value = IIf(ThisDrawing.GetVariable("dimsd2"), True, False)
CheckBox10.Value = IIf(ThisDrawing.GetVariable("dimsoxd"), True, False)
CheckBox11.Value = IIf(ThisDrawing.GetVariable("dimtix"), True, False)
CheckBox12.Value = IIf(ThisDrawing.GetVariable("dimtofl"), True, False)
CheckBox13.Value = IIf(ThisDrawing.GetVariable("dimaso"), True, False)
CheckBox14.Value = IIf(ThisDrawing.GetVariable("dimtih"), True, False)
CheckBox15.Value = IIf(ThisDrawing.GetVariable("dimtad"), True, False)
```

```
TextBox1.Text = ThisDrawing.GetVariable("dimasz")
TextBox2.Text = ThisDrawing.GetVariable("dimtsz")
TextBox3.Text = ThisDrawing.GetVariable("dimtxt")
TextBox4.Text = ThisDrawing.GetVariable("dimcen")
TextBox5.Text = ThisDrawing.GetVariable("dimexo")
TextBox6.Text = ThisDrawing.GetVariable("dimexe")
TextBox7.Text = ThisDrawing.GetVariable("dimdle")
TextBox8.Text = ThisDrawing.GetVariable("dimaltf")
TextBox9.Text = ThisDrawing.GetVariable("dimdli")
TextBox10.Text = ThisDrawing.GetVariable("dimgap")
TextBox11.Text = ThisDrawing.GetVariable("dimlfac")
TextBox12.Text = ThisDrawing.GetVariable("dimrnd")
TextBox13.Text = ThisDrawing.GetVariable("dimscale")
TextBox14.Text = ThisDrawing.GetVariable("dimtfac")
TextBox15.Text = ThisDrawing.GetVariable("dimtm")
TextBox16.Text = ThisDrawing.GetVariable("dimtp")
TextBox17.Text = ThisDrawing.GetVariable("dimtvp")
TextBox18.Text = ThisDrawing.GetVariable("dimdec")
```

End Sub

17. 同樣的操作，但是針對「取消」按鈕部份，其程式原文是：

```
Private Sub UserForm_Initialize()  
CheckBox1.Value = IIf(ThisDrawing.GetVariable("dimalt"), True, False)  
CheckBox2.Value = IIf(ThisDrawing.GetVariable("dimtoh"), True, False)  
CheckBox3.Value = IIf(ThisDrawing.GetVariable("dimupt"), True, False)  
CheckBox4.Value = IIf(ThisDrawing.GetVariable("dimsah"), True, False)  
CheckBox5.Value = IIf(ThisDrawing.GetVariable("dimse1"), True, False)  
CheckBox6.Value = IIf(ThisDrawing.GetVariable("dimse2"), True, False)  
CheckBox7.Value = IIf(ThisDrawing.GetVariable("dimsho"), True, False)  
CheckBox8.Value = IIf(ThisDrawing.GetVariable("dimsd1"), True, False)  
CheckBox9.Value = IIf(ThisDrawing.GetVariable("dimsd2"), True, False)  
CheckBox10.Value = IIf(ThisDrawing.GetVariable("dimsoxd"), True, False)  
CheckBox11.Value = IIf(ThisDrawing.GetVariable("dimtix"), True, False)  
CheckBox12.Value = IIf(ThisDrawing.GetVariable("dimtofl"), True, False)  
CheckBox13.Value = IIf(ThisDrawing.GetVariable("dimaso"), True, False)  
CheckBox14.Value = IIf(ThisDrawing.GetVariable("dimtih"), True, False)  
CheckBox15.Value = IIf(ThisDrawing.GetVariable("dimtad"), True, False)
```

```
TextBox1.Text = ThisDrawing.GetVariable("dimasz")  
TextBox2.Text = ThisDrawing.GetVariable("dimtsz")  
TextBox3.Text = ThisDrawing.GetVariable("dimtxt")  
TextBox4.Text = ThisDrawing.GetVariable("dimcen")  
TextBox5.Text = ThisDrawing.GetVariable("dimexo")  
TextBox6.Text = ThisDrawing.GetVariable("dimexe")  
TextBox7.Text = ThisDrawing.GetVariable("dimdle")  
TextBox8.Text = ThisDrawing.GetVariable("dimaltf")  
TextBox9.Text = ThisDrawing.GetVariable("dimdli")  
TextBox10.Text = ThisDrawing.GetVariable("dimgap")  
TextBox11.Text = ThisDrawing.GetVariable("dimlfac")  
TextBox12.Text = ThisDrawing.GetVariable("dimrnd")  
TextBox13.Text = ThisDrawing.GetVariable("dimscale")  
TextBox14.Text = ThisDrawing.GetVariable("dimtfac")  
TextBox15.Text = ThisDrawing.GetVariable("dimtm")  
TextBox16.Text = ThisDrawing.GetVariable("dimtp")  
TextBox17.Text = ThisDrawing.GetVariable("dimtvp")  
TextBox18.Text = ThisDrawing.GetVariable("dimdec")
```

```
End Sub
```


18. 同樣的操作，但是針對「上下限或公差標註」按鈕部份。請依圖 13-24 所示操作：

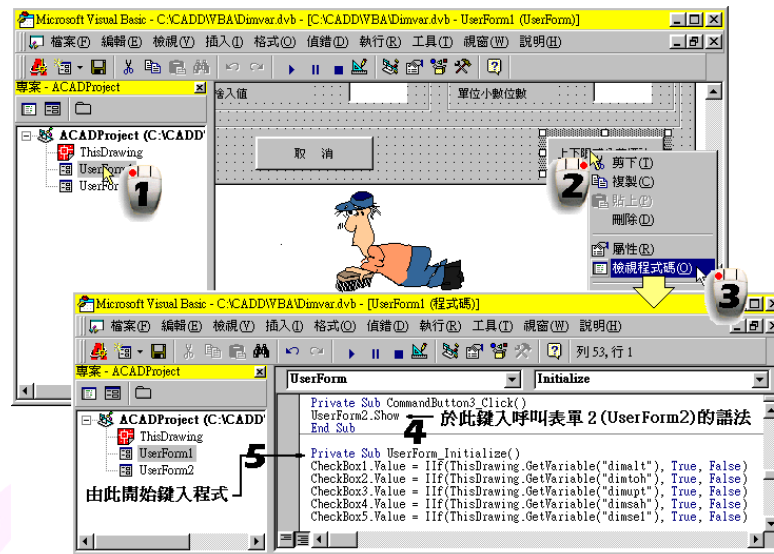


圖 13-24 控制「上下限或公差標註」按鈕的 VBA 程式原文視窗

控制「上下限或公差標註」按鈕部份的程式原文是：

```
Private Sub UserForm_Initialize()
    CheckBox1.Value = If(ThisDrawing.GetVariable("dimalt"), True, False)
    CheckBox2.Value = If(ThisDrawing.GetVariable("dimtoh"), True, False)
    CheckBox3.Value = If(ThisDrawing.GetVariable("dimupt"), True, False)
    CheckBox4.Value = If(ThisDrawing.GetVariable("dimsah"), True, False)
    CheckBox5.Value = If(ThisDrawing.GetVariable("dimse1"), True, False)
    CheckBox6.Value = If(ThisDrawing.GetVariable("dimse2"), True, False)
    CheckBox7.Value = If(ThisDrawing.GetVariable("dimsho"), True, False)
    CheckBox8.Value = If(ThisDrawing.GetVariable("dimsd1"), True, False)
    CheckBox9.Value = If(ThisDrawing.GetVariable("dimsd2"), True, False)
    CheckBox10.Value = If(ThisDrawing.GetVariable("dimsoxd"), True, False)
    CheckBox11.Value = If(ThisDrawing.GetVariable("dimtix"), True, False)
    CheckBox12.Value = If(ThisDrawing.GetVariable("dimtofl"), True, False)
    CheckBox13.Value = If(ThisDrawing.GetVariable("dimaso"), True, False)
    CheckBox14.Value = If(ThisDrawing.GetVariable("dimtih"), True, False)
    CheckBox15.Value = If(ThisDrawing.GetVariable("dimtad"), True, False)

    TextBox1.Text = ThisDrawing.GetVariable("dimasz")
    TextBox2.Text = ThisDrawing.GetVariable("dimtsz")
    TextBox3.Text = ThisDrawing.GetVariable("dimtxt")
End Sub
```



```

TextBox4.Text = ThisDrawing.GetVariable("dimcen")
TextBox5.Text = ThisDrawing.GetVariable("dimexo")
TextBox6.Text = ThisDrawing.GetVariable("dimexe")
TextBox7.Text = ThisDrawing.GetVariable("dimdle")
TextBox8.Text = ThisDrawing.GetVariable("dimaltf")
TextBox9.Text = ThisDrawing.GetVariable("dimdli")
TextBox10.Text = ThisDrawing.GetVariable("dimgap")
TextBox11.Text = ThisDrawing.GetVariable("dimlfac")
TextBox12.Text = ThisDrawing.GetVariable("dimrnd")
TextBox13.Text = ThisDrawing.GetVariable("dimscale")
TextBox14.Text = ThisDrawing.GetVariable("dimtfac")
TextBox15.Text = ThisDrawing.GetVariable("dimtm")
TextBox16.Text = ThisDrawing.GetVariable("dimtp")
TextBox17.Text = ThisDrawing.GetVariable("dimtvp")
TextBox18.Text = ThisDrawing.GetVariable("dimdec")

```

End Sub

19. 同樣的操作，但是針對「一般標示，公差標示或上下限標示」視窗部份。
請依圖 13-25 所示操作：

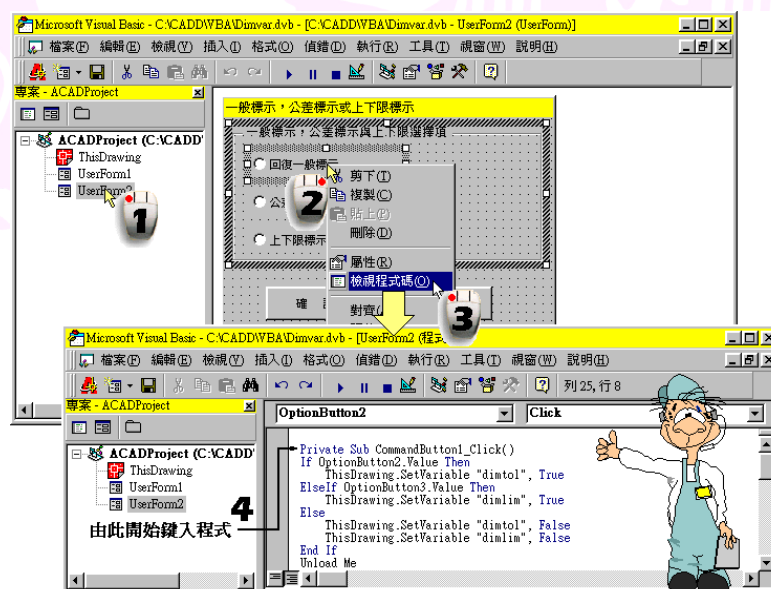


圖 13-25 控制「一般標示，公差標示或上下限標示」視窗的 VBA 程式原文視窗

控制「一般標示，公差標示或上下限標示」視窗部份的程式原文是：

```
Private Sub CommandButton1_Click()  
If OptionButton2.Value Then  
ThisDrawing.SetVariable "dimtol", True  
ElseIf OptionButton3.Value Then  
ThisDrawing.SetVariable "dimlim", True  
Else  
ThisDrawing.SetVariable "dimtol", False  
ThisDrawing.SetVariable "dimlim", False  
End If  
Unload Me  
End Sub
```

```
Private Sub CommandButton2_Click()  
Unload Me  
End Sub
```

```
Private Sub OptionButton1_Click()  
  
End Sub
```

```
Private Sub UserForm_Initialize()  
If ThisDrawing.GetVariable("dimtol") Then  
OptionButton2.Value = True  
ElseIf ThisDrawing.GetVariable("dimlim") Then  
OptionButton3.Value = True  
Else  
OptionButton1.Value = True  
End If
```

```
End Sub
```

比起 LISP 與 DCL 的語法來，VBA 的寫法可能還是算容易一點。所以，請多花一些耐心，來瞭解本範例整個控制語法與視窗項目內的關係。

13-3-2-2 更進階的範例

承 13-2 節範例，若要將之設計成交談式的人機介面，那麼在 VBA 裡的製作與設計過程如下：

- VBA 範例程式名稱：SQUARE.DVB（本電子書範例光碟）
- 配合檔案名稱：SQUARE.WMF（本電子書範例光碟）

程式本文如下：

● 模組 1(module1)

- (1)'定義二個整體變數
- (2)Public dlength As Double
- (3)Public dwidth As Double

● 程序（ThisDrawing）

- (1)Public Sub square()
- (2)Dim Pt1 As Variant
- (3)Dim Pt2 As Variant
- (4)Dim Pt3 As Variant
- (5)Dim Pt4 As Variant
- (6)Dim UtilObj As Object
- (7)Set UtilObj = ThisDrawing.Utility
- (8)Dim plineObj As AcadPolyline
- (9)Dim location As Variant
- (10)Dim pi As Double
- (11)pi = 3.141592654
- (12>UserForm1.Show '讀取並顯示 UserForm1 表單
- (13)'以下程式將用來計算矩形的四個點
- (14)Pt1 = ThisDrawing.Utility.GetPoint(, "矩形的左下角點:")
- (15)Pt2 = ThisDrawing.Utility.PolarPoint(Pt1, 0, dlength)
- (16)Pt3 = ThisDrawing.Utility.PolarPoint(Pt2, pi / 2, dwidth)
- (17)Pt4 = ThisDrawing.Utility.PolarPoint(Pt3, pi, dlength)

(18)'以下程式將用來建立變式

(19)UtilObj.CreateTypedArray location, vbDouble, _

(20)Pt1(0), Pt1(1), 0, _

(21)Pt2(0), Pt2(1), 0, _

(22)Pt3(0), Pt3(1), 0, _

(23)Pt4(0), Pt4(1), 0, _

(24)Pt1(0), Pt1(1), 0

(25)Set plineObj = ThisDrawing.ModelSpace.AddPolyline(location) '畫出矩形

(26)Dim newcenterPt As Variant

(27)Dim centerPt(0 To 2) As Double

(28)centerPt(0) = Pt1(0) + dlength / 2

(29)centerPt(1) = Pt1(1) + dwidth / 2

(30)centerPt(2) = 0

(31)'以下程式用來計算矩形中心

(32)newcenterPt = ThisDrawing.Utility.GetPoint("新的矩形中心點位置:")

(33)plineObj.Move centerPt, newcenterPt '移動矩形

(34)End Sub

● 表單 (UserForm1)

表單的建立操作原理請參考 13-3-2-1 節。其內容與程式碼段如圖 13-26 所示：

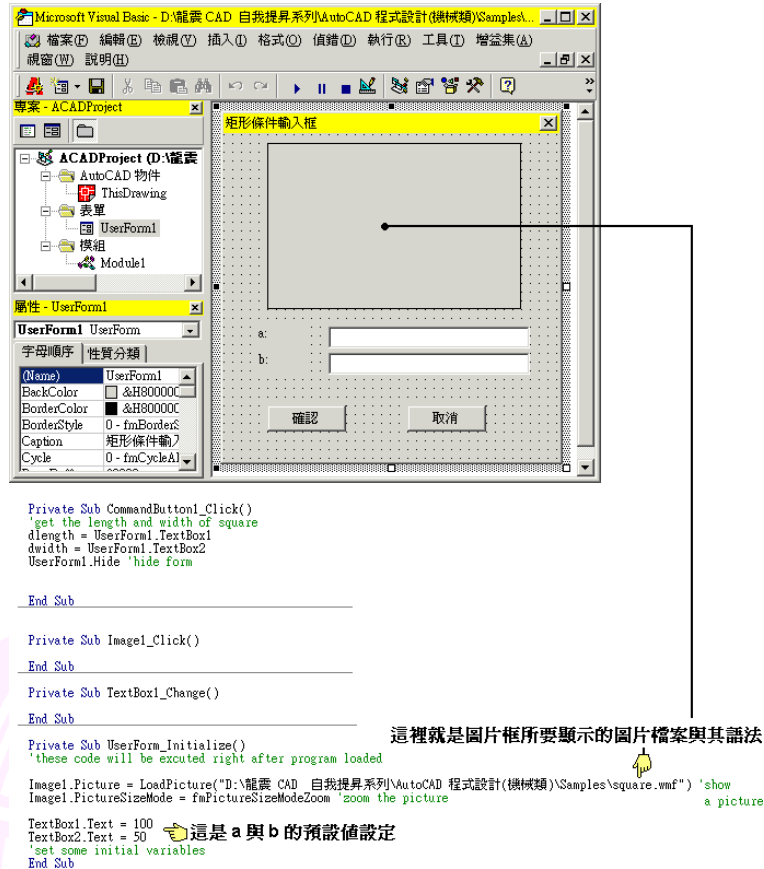


圖 13-26 UserForm1 表單的內容

分析：從表面上看來，VBA 的效果與 VLISP 的效果是一樣的。不過，細心的您一定會發現在圖片的顯示方面，VBA 要比較清楚。這是因為在 VLISP 裡所用的圖片是 AutoCAD .SLD 的幻燈片格式，而在 VBA 裡用的則是.WMF 的影像檔格式所致。我們使用.WMF 的原因是該圖是在 AutoCAD 裡畫的，而簡單的使用 AutoCAD 裡的 WMFOUT 指令就可以將圖面輸出成 WMF 影像格式（Windows 中繼檔）。如果您要使用 BMP、JPG、TIFF 之類的影像檔格式應該也可以。

除此之外，其他的分析內容則與 13-3-1-6 節的分析相同。

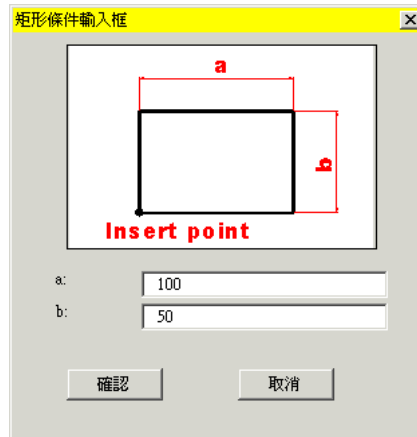


圖 13-27 VBA 的介面完成圖例

13-4 參數設計法的正確觀念

「參數設計法」這個名辭已經很普遍，可是卻有很多人將前面所談的「交談視窗」式的人機介面誤認是「參數設計法」。

CAD 裡可以使用高階語言來自動畫圖當然令人心動，很多人想學，所以我們才撰寫本電子書。可是程式設計這種東西是很專業且有深度的，並不是人人都可輕易上手的，因此聰明的 CAD 軟體設計師才又想出「參數設計法」這種功能來協助繪圖者。

所謂「參數設計法」就是：當您在設計畫圖時，可以將物件尺寸的值以一公式或關係式輸入，而不論是公式或關係式，其內都可包含此變數值。如此，當該變數值所代表的圖形尺寸發生變化時，所有相關的圖形尺寸也都會變化。這有點像是圖形式的 Excel。「參數設計法」實際上是 CAD/CAM 大系統下早已具有的觀念，只是現在再延續到 PC 級的軟體上而已。換句話說，「參數設計法」應該是 CAD 裡的一種讓使用者可以去方便操作功能，而不是泛指程式設計的一種流行寫法。您或許可以寫一條程式來模擬「參數設計法」功能，但是它卻不是程式撰寫時的基本理論。

有關「參數設計法」的實際操作，請參考本工作室出版《Pro/ENGINEER Wildfire3.0 進階設計》(上奇書號：EB446)一書。

啓發性習題

實作題

1. 請修改本章範例的 SQUARE.DVB，讓插入點與新位置的提示文句等傳統提問介面也融入交談式視窗的介面中。（解答檔案：NEWSQUARE.DVB）
2. 請設計一如下的交談式視窗，以方便的讓我們來設計、撰寫、修改、載入與列印一 LISP 程式（以 AutoLISP/DCL 撰寫）：



注意：點取「編輯檔案」鈕，就是執行 DOS EDIT 文字處理程式指令所出現的畫面。

解答檔案名稱：EDITLISP.LSP，EDITLISP.DCL

3. 請設計一如下的交談式視窗，讓我們很快的選取並執行尺寸標示指令（以 AutoLISP/DCL 撰寫）：



解答檔案名稱：PDIM.LSP，PDIM.DCL