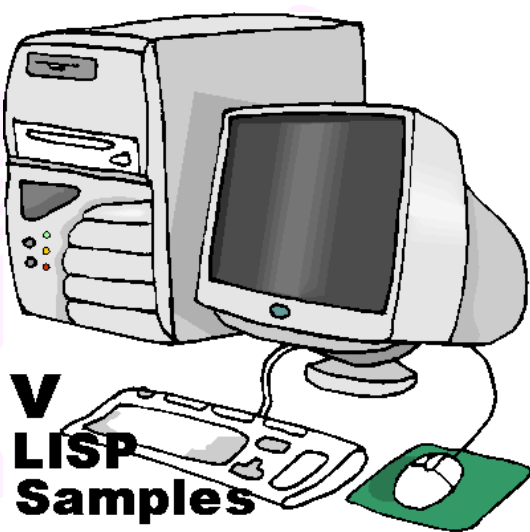


## 第十四章 四個 Visual LISP 的程式設計範例

在本章中，我們將透過 4 個 Visual LISP 的範例來有效的學習 Visual LISP。同時，在本章範例中將出現一些前面章節未介紹到的函數和語法，您也將能在範例的練習過程中體會。在本章中要練習的範例有：

- 繪製油杯
- 繪製軸承
- 繪製螺帽
- 會“動”的圓中心線



### 14.1 油杯的繪製

- 本範例完成圖形

本範例將畫出一個如圖 14-1 所示的油杯：

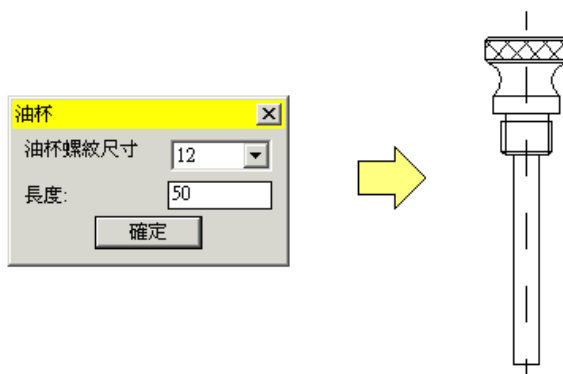


圖 14-1 本程式的油杯完成圖

本例將主要用到幾個程式，其中：oilcup 為主程式，其餘的 getdata、txt2list、3parc、vla-ObjectList->Variant、List->Variant、load-line-types、find-line-type 與 draw 等為副程式。以下部分我們將詳細來講述它們的功能以及程式設計的技巧。

- 本範例檔案：oilcup.lsp（本電子書範例光碟(03M)Samples\Volume4 目錄）  
oilcup.dcl（本電子書範例光碟(03M)Samples\Volume4 目錄）  
oilcup.txt（本電子書範例光碟(03M)Samples\Volume4 目錄）

- 主程式 oilcup 部分

**作用：**呼叫副程式以繪出油杯圖形

(1)(defun c:oilcup ()

**說明：**這條程式用來定義主程式：oilcup。

(2)(setq osmode (getvar "osmode")

(3) cmdecho (getvar "cmdecho")

(4) blipmode (getvar "blipmode")

(5))

(6)(setvar "osmode" 0)

(7)(setvar "cmdecho" 0)

(8)(setvar "blipmode" 0)

**說明：**這群程式將用來將需要的環境變數儲存起來，以備恢復。設定環境變數時，特別要注意的是 AutoCAD 的圖形鎖點模式 ("osmode")，它在許多情況下將影響繪圖的準確性，但它在人工的互動式繪圖中非常有用。為此，在程式執行的過程中將其關閉（即設為 0），程式執行完了再將其恢復。

(9)(vl-load-com)

**說明：**這條程式的作用是将 Visual LISP 延伸功能載入到 AutoLISP，該函數將載入 Visual LISP 所提供的延伸函數。Visual LISP 延伸函數將透過 AutoLISP 來履行對 ActiveX 的支援，同時還提供了 ActiveX 實用程式、資料轉換函數、曲線測量..等函數。換句話說，要使用 ActiveX 物件，就必須執行這行程式，您甚至可以將它加入自動載入指令中，使其在啟動 AutoCAD 時執行。

(10)(setq acadObject (vlax-get-acad-object))

(11)(setq acadDocument (vla-get-ActiveDocument acadObject))

(12)(setq mSpace (vla-get-ModelSpace acadDocument))

```
(13)(setq lts (vla-get-Linetypes acadDocument))
```

**說明：**從第十一章中我們可以看出：AutoCAD 的物件是以階層結構組織的，因此在使用各級屬性時，必須依照該物件模型中的繼承關係。以 mSpace 為例，爲了在模型空間中加入圖素，就要先取得模型空間變數 mSpace。根據階層關係，模型空間變數要從 Document 物件中取得，而 Document 又要從整體 Application（即 acadObject）中擷取。所以透過上述四條程式，我們就可以得到模型空間變數 mSpace 和線型變數 lts。

```
(14)(setq it_dcl (load_dialog "oilcup.dcl"))
```

```
(15)(if (< it_dcl 0)
```

```
(16) (exit)
```

```
(17))
```

**說明：**這幾條程式是用來載入一個 DCL 檔案，load\_dialog 將傳回一整數值 it\_dcl，來供 new\_dialog 使用。如果 DCL 檔案載入成功，則傳回大於 0 的整數值，反之，則傳回一小於 0 的整數值，並跳出程式。

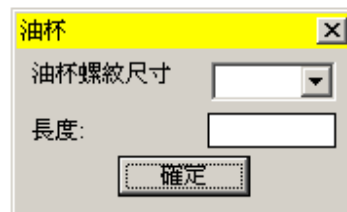
```
(18)(if (not (new_dialog "oilcup" it_dcl))
```

```
(19) (exit)
```

```
(20))
```

**說明：**顯示一條條件輸入交談式視窗。如圖 14-2 所示：

```
oilcup :dialog {  
  label="油杯";  
  :popup_list {  
    label="油杯螺紋尺寸";  
    key="oilcup_dim";  
    edit_width=8;  
  }  
  :edit_box {  
    label="長度:";  
    key="long";  
    edit_width=9;  
  }  
  ok_button;  
}
```



### **oilcup.dcl 檔案的內容**

圖 14-2 呼叫 oilcup.dcl 所造成的交談視窗

```
(21)(start_list "oilcup_dim")
```

```
(22)(setq dimlist '("12" "16" "20"))
```

```
(23)(mapcar 'add_list dimlist)
```

```
(24)(end_list)
```

**說明：**oilcup.dcl 的介面如圖 14-2 所示，其中有一個列表框(list)，用以取得油杯螺紋尺寸，初始時，交談式視窗的列表框(list)內為空。以上四條程式的作用就是向 list 中加入內容 ("12" "16" "20")，如圖 14-3 所示。

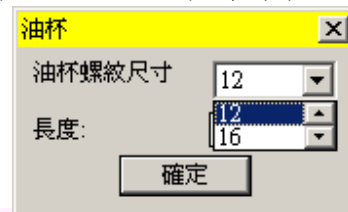


圖 14-3 油杯列表框的內容

(25)(action\_tile "accept" "(getdata)(DONE\_DIALOG)")

**說明：**如圖 14-4，當您輸入油杯的設計條件後，點取「確定」鈕後，即啟動交談式視窗的 "accept" 語法，並順序執行(getdata)(DONE\_DIALOG)。其中，(getdata) 的副程式呼叫將用於從交談式視窗中取得對應的油杯資料，即油杯螺紋尺寸與長度，然後分別將它們儲存於變數 oilcup\_dim 和 length 中。(DONE\_DIALOG) 的語法是用來關閉交談式視窗的。

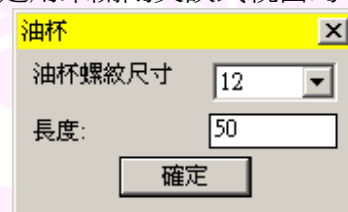


圖 14-4 於交談視窗中輸入油杯條件

(26)(start\_dialog)

**說明：**這條程式是用來啟動上述 (new\_dialog) 語法所顯示交談式視窗，並將控制權移交給交談式視窗。注意：在 (start\_dialog) 之前一定要檢測(new\_dialog) 的傳回值是否為真（即 T），否則會發生不可預期的後果。

(27)(setq f (open (strcat \*fn\* "oilcup.txt") "r"))

**說明：**在取得油杯的關鍵尺寸之後，我們要從油杯的資料庫中得到其他的幾何形狀參數。資料庫為文字檔案 "oilcup.txt"，如圖 14-5 所示。其中資料之間將以空格分開。

```
d2 d3 h a b c d dd d1 c1 md dj f
12 6 28 10 6 4 20 16 4 2 10.2 0.8 1.5
16 6 35 12 8 5 26 22 4 2 13.6 1 2
20 8 42 15 10 6 32 26 6 2.5 17 1.2 2.5
```

圖 14-5 油杯資料檔案

當然，在第三章中我們就已提及，資料的儲存方式有許多種，而本例我們以這種方式為代表，給讀友一個新鮮的體認。換句話說：本句程式的用意是以讀取的方式打開檔案 "oilcup.txt"，並將檔案變數存於檔案描述子 f 中，以便後續的呼叫。

(28)(read-line f)

**說明：**這條程式是用來讀取 f 所指定的已開啓檔案中行結束符號前的字串，這行程式傳回值應為：

"d2 d3 h a b c d dd d1 c1 md dj f"

由於這一行程式只用來表示資料庫各列資料的含義，因此我們不需要使用 (setq (read-line f)) 來取得其值。

```
(29)(cond ((= oilcup_dim 12)
(30)      (setq rowdata (read-line f))
(31)      )
(32)      ((= oilcup_dim 16)
(33)      (setq rowdata (read-line f))
(34)      (setq rowdata (read-line f))
(35)      )
(36)      ((= oilcup_dim 20)
(37)      (setq rowdata (read-line f))
(38)      (setq rowdata (read-line f))
(39)      (setq rowdata (read-line f))
(40)      )
(41))
```

**說明：**cond 是一個類似 VB 中 "case" 的選擇語法，它將於判斷判別式的值後，來執行對應的程式。例如，如果油杯的尺寸選 12，即 oilcup\_dim=12，則它將讀取 f 所指定的檔案中行結束符號前的字串：

"12 6 28 10 6 4 20 16 4 2 10.2 0.8 1.5"

並賦予一值給 rowdata。因為前面的程式碼已執行過一個 (read-line f)，所以這次的(read-line f)將讀取 "oilcup.txt" 的第二行。例如，如果油杯的尺寸選 20，即 oilcup\_dim=20，則 rowdata 值為：

"20 8 42 15 10 6 32 26 6 2.5 17 1.2 2.5"。

(42)(txt2list rowdata)

**說明：**因為 rowdata 是一個字串，為了方便得到各參數的值，所以有必要呼叫 txt2list 副程式來將其轉換為串列 (list)。

(43)(close f)

**說明：**這條程式將關閉 f 所指定的檔案。注意：在使用該檔案後應及時將其關閉，否則由於程式執行過程中的失誤操作或意外當機等因素，均會引起不可預期的後果。

(44)(draw la)

**說明：**這條程式是用來呼叫 (draw) 副程式，來繪出油杯圖形。

(45)(setvar "osmode" osmode)

(46)(setvar "cmdecho" cmdecho)

(47)(setvar "blipmode" blipmode)

**說明：**這些程式是在當程式執行到尾聲時，將環境變數恢復原來的設定。這並不影響原有的操作。

(48)(princ)

(49))

## ● 副程式 getdata 部分

**作用：**擷取交談式視窗中變數的值、油杯螺紋尺寸與長度。

```
(defun getdata ()  
  (setq li      (GET_TILE "oilcup_dim")  
        list_index (atoi li)  
  )
```

**說明：**根據 oilcup.dcl 中的 key 值("oilcup\_dim")，oilcup\_dim 將對應列表框來擷取油杯螺紋長度值。這四行程式的作用是要將列表框中的索引值賦予 li，即選 "12" 對應 "0"，"16" 對應 "1"，"20" 對應 "2"。此時得到的 li 值為字串，所以還要再用 atoi 將其轉換為整數值來賦予 list\_index。



```
(setq od (nth list_index dimlist)
  oilcup_dim (atoi od)
)
```

**說明：**根據索引值，從列表框內容 list\_index（值為 ("12" "16" "20")）中擷取所需的螺紋長度。

```
(setq lg (GET_TILE "long")
  length (atof lg)
)
```

**說明：**根據 oilcup.dcl 中的 key 值 ("long")，long 將由對應的輸入框擷取油杯長度值。這三行程式是將輸入框中取值賦於 lg。同理，需將其轉換為浮點型並賦於 length。

## ● 副程式 txt2list 部分

**作用：**將包含資料的字串轉換成數值表。在前面章節的範例中，我們已見過類似的例子。呼叫時要求輸入字串變數。

```
(defun txt2list (rc)
  (setq p 1
    sp 1
    len 1
    sl (strlen rc)
    la (list nil)
  )
)
```

**說明：**這些程式行用來設定一些變數。其中，p 代表指標，用於確定字串中的位置，初始值為 1；sp 表示開始位置，記錄開始讀取字串時的位置初始值為 1；len 代表字元長度，用於確定讀取字串的長度初始值為 1；sl 是字串的總長度；la 初始值為一個只含有 nil 的串列，用來儲存最後得到的數值表。

```
(while (<= p (1+ sl))
(progn
  (setq temp (substr rc p 1))
  (if (or (= temp "") (= temp " "))
    (progn
      (setq str (substr rc sp len)
        sp (1+ p)

```

```

len 0
data (atof str)
la (append la (list data))
)
)
(setq len (1+ len))
)
(setq p (1+ p))
)
)
(setq la (vl-remove nil la))
)

```

**說明：**while 迴圈中，程式的功能是用來依次讀取字串中的數值的。其中，利用數值與數值之間的空格為數值分界。讀到的字元若不是空格，則指標  $p$  加 1，字元長度  $len$  也加 1，再讀取下一個；如果遇到空格，則將讀取從字元開始位置  $sp$  起的  $len$  個字元，並將其轉換為浮點數後加入串列  $la$  中；然後再將字串長度  $len$  的值重置為 0，字串開始位置設為  $p+1$ 。最後，重複這個過程直至指標  $p$  超過字串總長  $sl$  為止。程式中有一個新函數 `vl-remove`，它的作用是從串列中刪除元素。

### ● 副程式 3parc 部分

**作用：**以三個點作出一個弧。我們都知道：三個點就可以畫出一弧。我們只要簡單地使用 AutoLISP 的 (command "arc" a b c) 就可以辦到。其中， $a, b, c$  分別代表三個點。不過，本範例要為您說明的是：要如何使用 ActiveX 來以三個點建立一段弧。根據 ActiveX 中建立弧物件的語法，如圖 14-6 所示。我們先需確定弧中心點  $O$ ，半徑  $Oa$  及起始角  $\alpha$  及終止角  $\beta$ 。所以，呼叫時要求輸入三個點及容器物件（一般為模型空間或圖塊）。

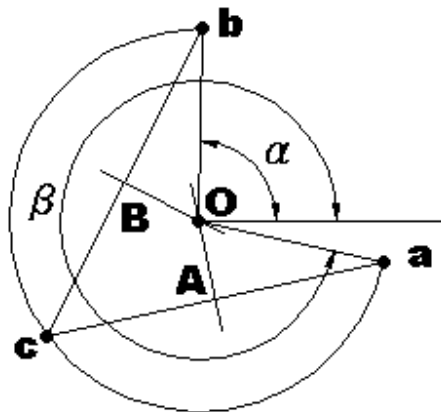




圖 14-6 三點畫弧的設計原理

```
(defun 3parc (pt1 pt2 pt3 mspace / midpt1
             midpt2 lang1 lang2 midpt1e midpt2e interPt
             vinterp arc2r angs2 ange2
)
  (setq midpt1 (list (/ (+ (car pt1) (car pt2)) 2)
                    (/ (+ (cadr pt1) (cadr pt2)) 2)
)
  (setq midpt2 (list (/ (+ (car pt2) (car pt3)) 2)
                    (/ (+ (cadr pt2) (cadr pt3)) 2)
)
  )
```

**說明：**求 ac 及 bc 的中點：midpt1 和 midpt2

```
(setq lang1 (angle pt1 pt2))
(setq lang2 (angle pt2 pt3))
(setq lang1 (+ (/ pi 2) lang1)
      lang2 (+ (/ pi 2) lang2)
)
(setq midpt1e (polar midpt1 lang1 100)
      midpt2e (polar midpt2 lang2 100)
)
```

**說明：**求出中垂線上另兩點：midpt1e 和 midpt2e

```
(setq interPt (inters midpt1 midpt1e midpt2 midpt2e nil))
```

**說明：**求出兩條中垂線交點 interPt

```
(setq vinterp (vlax-3d-point interPt))
```

**說明：**將此交點轉換為與 ActiveX 相容的（變式）三維點結構 vinterp

```
(setq arc2r (distance interPt pt1))
```

**說明：**求半徑

```
(setq angs2 (angle interPt pt1))
```

**說明：**求起始角

```
(setq ange2 (angle interpt pt3))
```

**說明：**求終止角

```
(vla-addarc mspace vinterpt arc2r angs2 ange2)  
)
```

**說明：**在模型空間中畫出弧

- 副程式 vla-ObjectList->Variant 部分

**作用：**將 ActiveX 物件串列轉換為變數。當 ActiveX 方法（method）需要物件的時候，就需要使用包含 ActiveX 物件的安全陣列（safearray）變數。如果後面還涉及添加的圖案填滿，則呼叫時就會要求輸入物件類型與物件串列。舉例說明，如果已建立一圓物件 circ，如果物件串列中只包含圓，那麼此呼叫將命名為：  
(vla-ObjectList->Variant (list circ))。

```
(defun vla-ObjectList->Variant (ObjectList)  
  (List->Variant  
    vlax-VbObject  
    ObjectList  
  )  
)
```

**說明：**呼叫另一個副程式 List->Variant 來履行這個程式的功能。

- 副程式 List->Variant 部分

**作用：**用 vla-ObjectList->Variant 傳來的 vtype 物件類型建立一個 inlist 長度的安全陣列(safearray)，並將 inlist 填入這個安全陣列中，最後再將其轉換為變式 (variant)資料類型。呼叫時需要輸入物件類型與物件串列。

```
(defun List->Variant (vtype inlist)  
  (vlax-make-variant  
    (vlax-safearray-fill  
      (vlax-make-safearray
```

```

        vtype
        (cons 0 (1- (length inlist)))
    )
    inlist
)
)
)
)

```

### ● 副程式 load-line-types 部分

**作用：**載入所需線型以供使用。呼叫時需要輸入線型及線型檔案，線型檔案一般為 acad.lin，當然也可以是自己建立的線型檔案。

```

(defun load-line-types (line-type file-name / tmp res)
  (if (and (setq tmp (vlax-get-acad-object))
          (setq tmp (vla-get-activedocument tmp))
          (setq tmp (vla-get-linetypes tmp)))
    )
  )

```

**說明：**這是取得線型組變數的另一種寫法。並將線型變數存入 tmp 中。

```

(if (setq res (find-line-type line-type tmp))

```

**說明：**呼叫副程式 find-line-type 檢查線型是否已載入。

```

res

```

**說明：**如果已載入，res 為已載入的線型變數；如果沒有載入，則為 nil。

```

(vla-load tmp line-type file-name)

```

**說明：**如果所需線型沒有載入，則載入。

```

)
nil
)
)

```

### ● 副程式 find-line-types 部分

**作用：**尋找所需線型是否已經載入。呼叫時需要輸入線型及線型組。事實上，它是用來判斷目前線型組中是否有所需要的線型名稱。

```
(defun find-line-type (line-type line-type-collection / res)
  (setq line-type (strcase line-type))
```

**說明：**將線型名稱轉成大寫

```
(vlax-for l-obj line-type-collection
  (if (= (strcase (vla-get-name l-obj)) line-type)
    (setq res l-obj)
  )
)
```

**說明：**遍歷線型組，查找是否有與所需線型名稱相同的線型；若有，則賦予值給 res。

```
res
)
```

**說明：**傳回值為 res

## ● 副程式 draw 部分

**作用：**畫出油杯圖形。呼叫時需要輸入包含油杯幾何參數的數值表。本函數的設計思路就是：先詢問操作輸入條件，然後繪製。

```
(defun draw (la)
  (setq pt (getpoint "\n 圖形插入點:"))
  vpt (vlax-3d-point pt)
)
```

**說明：**提示輸入圖形插入點 pt，並將其轉換與 ActiveX 相容的（變式）三維點結構。

```
vpt
  (setq d2 (nth 0 la)
        d3 (nth 1 la)
        h   (nth 2 la)
```

```

a (nth 3 la)
b (nth 4 la)
c (nth 5 la)
d (nth 6 la)
dd (nth 7 la)
d1 (nth 8 la)
c1 (nth 9 la)
md (nth 10 la)
dj (nth 11 la)
f (nth 12 la)
)

```

**說明：**將數值表 la 中的數值分別賦予值。

```
(setq ptd (vlax-variant-value vpt))
```

**說明：**將 vpt 變數值（安全陣列 safearray）存入 ptd

```
(setq ptx (vlax-safearray-get-element ptd 0))
(setq pty (vlax-safearray-get-element ptd 1))
```

**說明：**取得三維點結構 vpt 中的 x 值和 y 值，並將之分別存為 ptx，pty。

```

(setq x0 ptx
      y0 pty
      r (/ b 2)
      x1 (+ x0 (/ dd 2))
      y1 y0
      x2 (- x0 (/ dd 2))
      y2 y0
      x3 x2
      y3 (+ y0 c)
      x4 x1
      y4 y3
      x5 (+ x0 (/ d2 2))
      y5 (- y0 c1)
      x6 (- x0 (/ d2 2))
      y6 y5
      x7 x6
      y7 (- y0 (- a (/ (- d2 md) 2)))
      x8 (- x0 (/ md 2))

```

y8 (- y0 a)  
x9 (+ x8 md)  
y9 y8  
x10 (+ x7 d2)  
y10 y7  
x11 (- x0 (/ md 2))  
y11 y0  
x12 x11  
y12 y5  
x13 (+ x11 md)  
y13 y0  
x14 x13  
y14 y5  
htmp (- h b a c)  
x15 (- x0 (/ d3 2))  
y15 y8  
x16 (+ x15 d3)  
y16 y15  
x17 x16  
y17 (- y16 length)  
x18 x15  
y18 y17  
x19 (+ dj (- x0 (/ d 2)))  
y19 (+ y0 c htmp)  
x20 (- (+ x0 (/ d 2)) dj)  
y20 y19  
x21 (+ x0 (/ d 2))  
y21 (+ y20 dj)  
x22 x21  
y22 (- (+ y21 b) dj dj)  
x23 (- x22 dj)  
y23 (+ y22 dj)  
x23 (- x22 dj)  
y23 (+ y22 dj)  
x24 x19  
y24 y23  
x25 (- x22 d)  
y25 y22  
x26 x25  
y26 y21  
x27 x0



```

y27 (+ y0 c (/ htmp 2))
x28 x0
y28 (- (+ y0 h) a)
x29 (+ x3 f)
y29 y19
x30 x29
y30 y3
x32 (- x4 f)
y32 y4
x31 x32
y31 y20
x33 (+ x0 (/ d2 2))
y33 y27
x34 (- x33 d2)
y34 y27
x35 x0
y35 (+ y28 3 (/ d3 2))
x36 x0
y36 (- y18 3)
)

```

**說明：**以  $x_0$ ， $y_0$  為基點，繪出確定油杯結構所需點的橫座標和縱座標，本例爲了簡單清晰地繪出圖形，用了較多的點。

```

(setq p1 (list x1 y1)
      p2 (list x2 y2)
      p3 (list x3 y3)
      p4 (list x4 y4)
      p5 (list x5 y5)
      p6 (list x6 y6)
      p7 (list x7 y7)
      p8 (list x8 y8)
      p9 (list x9 y9)
      p10 (list x10 y10)
      p11 (list x11 y11)
      p12 (list x12 y12)
      p13 (list x13 y13)
      p14 (list x14 y14)
      p15 (list x15 y15)
      p16 (list x16 y16)
      p17 (list x17 y17)

```

p18 (list x18 y18)  
p19 (list x19 y19)  
p20 (list x20 y20)  
p21 (list x21 y21)  
p22 (list x22 y22)  
p23 (list x23 y23)  
p24 (list x24 y24)  
p25 (list x25 y25)  
p26 (list x26 y26)  
p27 (list x27 y27)  
p28 (list x28 y28)  
p29 (list x29 y29)  
p30 (list x30 y30)  
p31 (list x31 y31)  
p32 (list x32 y32)  
p33 (list x33 y33)  
p34 (list x34 y34)  
p35 (list x35 y35)  
p36 (list x36 y36)  
)

**說明：**將橫座標和縱座標組成串列，以確定各點。

```
(setq lwpline1 (vlax-make-safearray vlax-vbdouble '(0 . 9)))
```

**說明：**建立一個可儲存 10 個資料類型為 double 的安全陣列，因為我們要繪出的是 LightWeightPloyline，所以每個點只需兩個座標，5 個點共有 10 個值。

```
(vlax-safearray-put-element lwpline1 0 x1)  
(vlax-safearray-put-element lwpline1 1 y1)  
(vlax-safearray-put-element lwpline1 2 x2)  
(vlax-safearray-put-element lwpline1 3 y2)  
(vlax-safearray-put-element lwpline1 4 x3)  
(vlax-safearray-put-element lwpline1 5 y3)  
(vlax-safearray-put-element lwpline1 6 x4)  
(vlax-safearray-put-element lwpline1 7 y4)  
(vlax-safearray-put-element lwpline1 8 x1)  
(vlax-safearray-put-element lwpline1 9 y1)
```

**說明：**將 5 個點的座標值填入 lwpline1 中

```
(setq pline1 (vla-AddLightWeightPolyline mspace lwpline1))
```

**說明：**在模型空間中，依 lwpline1 中所儲存的點來繪製聚合線 pline1。

```
(setq lwpline2 (vlax-make-safearray vlax-vbdouble '(0 . 13)))  
(vlax-safearray-fill lwpline2 (list x5 y5 x6 y6 x7 y7 x8 y8 x9 y9 x10 y10 x5 y5))
```

**說明：**這是填入座標的另一種寫法。

```
(setq pline2 (vla-AddLightWeightPolyline mspace lwpline2))
```

```
(setq lwpline3 (vlax-make-safearray vlax-vbdouble '(0 . 7)))  
(vlax-safearray-fill lwpline3 (list x15 y15 x18 y18 x17 y17 x16 y16))  
(setq pline3 (vla-AddLightWeightPolyline mspace lwpline3))  
(setq line1 (vla-addline mspace (vlax-3d-point p11) (vlax-3d-point p12)))
```

**說明：**在模型空間中加入端點為 p11 和 p12 的直線 line1。

```
(setq line2 (vla-addline mspace (vlax-3d-point p13) (vlax-3d-point p14)))  
(setq line3 (vla-addline mspace (vlax-3d-point p7) (vlax-3d-point p10)))  
(setq arc1 (3parc p3 p34 p19 mspace))
```

**說明：**呼叫副程式 3parc，以三點在模型空間中畫弧，並使用變數 arc1 來代表。

```
(setq arc2 (3parc p20 p33 p4 mspace))
```

```
(setq lwpline4 (vlax-make-safearray vlax-vbdouble '(0 . 9)))  
(vlax-safearray-fill lwpline4 (list x21 y21 x22 y22 x25 y25 x26 y26 x21 y21))  
(setq pline4 (vla-AddLightWeightPolyline mspace lwpline4))
```

```
(setq lwpline5 (vlax-make-safearray vlax-vbdouble '(0 . 7)))  
(vlax-safearray-fill lwpline5 (list x21 y21 x20 y20 x19 y19 x26 y26))  
(setq pline5 (vla-AddLightWeightPolyline mspace lwpline5))
```

```
(setq lwpline6 (vlax-make-safearray vlax-vbdouble '(0 . 7)))  
(vlax-safearray-fill lwpline6 (list x22 y22 x23 y23 x24 y24 x25 y25))  
(setq pline6 (vla-AddLightWeightPolyline mspace lwpline6))
```

```
(setq pName "ANSI37"  
  pType 0  
  bA T
```

)

**說明：**上面的這群程式將賦予值給一些變數，並定義圖案填滿。pName 是圖案填滿的圖案名稱，本例為："ANSI37"，pType 是圖案填滿類型，本例為：0，代表「預先定義」；bA 代表圖案填滿的關聯性，T 即 True，代表關聯，反之為不關聯。

```
(setq hatch (vla-addhatch mspace pType pName bA))
```

**說明：**在模型空間中建立 hatch 物件。

```
(vla-AppendOuterLoop  
  hatch  
  (vla-objectlist->variant (list pline4))  
)
```

**說明：**在 pline4 所圍成的空間中（即以 pline4 為外邊界，對應 vla-AppendOuterLoop；如果是內邊界，則對應 vla-AppendInnerLoop），畫出圖案填滿。

**注意：**

1. 此處使用副程式 (vla-objectlist->variant) 將一個變式值傳入本副程式。
2. 欲填滿圖案的物件邊界的一定要是個封閉線段。

```
(vla-Evaluate hatch)
```

**說明：**確定目前的圖案填滿視圖。

```
(setq  
  line4 (vla-addline mspace (vlax-3d-point p8) (vlax-3d-point p12)))  
(setq  
  line5 (vla-addline mspace (vlax-3d-point p9) (vlax-3d-point p14)))  
(setq  
  line6 (vla-addline mspace (vlax-3d-point p35) (vlax-3d-point p36)))  
(load-line-types "DASHDOT" "acad.lin")
```

**說明：**載入線型 "DASHDOT"，用於畫出圖案的中心線。

```
(vla-put-Linetype line6 "DASHDOT")
```

**說明：**將 line6 線物件的線型設為 "DASHDOT"。

```
(setq llen (vla-get-Length line6)
  lts (/ llen 5)
)
```

**說明：**計算線物件 line6 的長度，用其 1/5 來確定一個適當的線型比例，並將之存在 lts 中。

```
(vla-put-LinetypeScale line6 lts)
```

**說明：**設定線物件 line6 的線型比例為 lts。

```
(setq ang (getangle pt "\n 請輸入旋轉角:"))
```

**說明：**提示輸入角度，並存入變數 ang 中。

```
(mapcar 'vla-rotate
  (list pline1  pline2  pline3  pline4  pline5  pline6
    line1      line2    line3    line4    line5    line6
    arc1       arc2     hatch
  )
  (list vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt
    vpt)
  (list ang ang ang ang ang ang ang ang ang ang ang ang ang ang
    ang)
)
)
```

**說明：**依基點 vpt 將構成油杯的所有圖素旋轉一個角度 ang。

## ● 本範例的執行步驟

假設 oilcup.lsp，oilcup.dcl 與 oilcup.txt 均在：

**\\03M)Samples\Volume4\**

目錄下，如果您將這些檔案放在另外的目錄裡，那麼本範例中有關存取這些檔案的路徑也要更改（請參考 2-5 節）。然後，再依下步驟執行：

1. 在 AutoCAD 指令提示號後輸入（參考圖 2-22 也可以）：

指令: (load "oilcup.lsp") <Enter>  
指令: oilcup <Enter>

2. 此時，會出現一個交談式視窗，請於填入適當的數值後，按「確定」鈕。
3. 然後，程式將於指令提示區中提示您輸入插入點。請於圖面上點取一插入點，當油杯圖形畫出後，程式將詢問您輸入旋轉角。

## 14-2 軸承的繪製

- 本範例完成圖形

本範例將畫出一個如圖 14-7 所示的軸承：

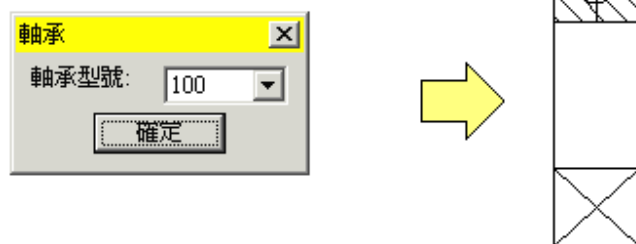


圖 14-7 本程式的軸承完成圖

本範例共由 7 個函數所組成。其中，bearing 為主程式，其餘的 getdata1、txt2list1、draw1、selarc1、vla-ObjectList->Variant1 與 List->Variant1 為副程式。以下，我們將詳細講述它們的功能以及程式設計的技巧。但是請注意：對前面範例已經講述過的語法和函數將不再重複。

- 本範例檔案：
  - bearing.lsp (本電子書範例光碟(03M)Samples\Volume4 目錄)
  - bearing.dcl (本電子書範例光碟(03M)Samples\Volume4 目錄)
  - gb276.txt (本電子書範例光碟(03M)Samples\Volume4 目錄)

- 主程式 bearing 部分

作用：呼叫副程式來繪出軸承圖形

(1)(defun c:bearing ()



**說明：**定義主程式 bearing。

```
(2)(setq osmode (getvar "osmode")  
(3)      cmdecho (getvar "cmdecho")  
(4)      blipmode (getvar "blipmode")  
(5))
```

**說明：**將一些系統變數儲存起來，以備圖形繪出後恢復。

```
(6)(setvar "osmode" 0)  
(7)(setvar "blipmode" 0)  
(8)(setvar "CMDECHO" 0)
```

**說明：**設定一些需要的系統變數。

```
(9)(vl-load-com)
```

**說明：**載入 Viual LISP 延伸功能。

```
(10)(setq acadObject (vlax-get-acad-object))  
(11)(setq acadDocument (vla-get-ActiveDocument acadObject))  
(12)(setq mSpace (vla-get-ModelSpace acadDocument))
```

**說明：**擷取模型空間變數。

```
(13)(setq it_dcl (load_dialog "bearing.dcl"))  
(14)(if (< it_dcl 0)  
(15)  (exit)  
(16))
```

**說明：**載入軸承 DCL 檔案。

```
(17)(if (not (new_dialog "bearing" it_dcl))  
(18)  (exit)  
(19))
```

**說明：**顯示如圖 14-8 所示的軸承 DCL 交談式視窗：

```

        (start_list "btype")
        (setq typelist '("100" "101" "102" "103" "104"
                          "105" "106" "107" "108" "109"
                          "110" "111" "112" "113" "114"
                          "115" "116" "117" "118" "119"
                          "120" "121" "122" "200" "201"
                          "202" "203" "204" "205" "206"
                          "207" "208" "209" "210" "211"
                          "212" "213" "214" "215" "216"
                          "217" "218" "219" "220" "300"
                          "301" "302" "303" "304" "305"
                          "306" "307" "308" "309" "310"
                          "311" "312" "313" "314" "315"
                          "316" "317" "318" "319" "405"
                          "406" "407" "408" "409" "410"
                          "411" "412" "413" "414" "415"
                          "416" "417" "418"))
    )
    (mapcar 'add_list typelist)
    (end_list)
  )
  bearing :dialog {
    label="軸承";
    :popup_list {
      label="軸承型號:";
      key="btype";
      edit_width=8;
    }
    ok_button;
  }

```

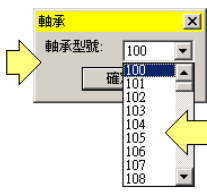


圖 14-8 軸承 DCL 交談式視窗

```

(20)(start_list "btype")
(21)(setq typelist '("100" "101" "102" "103" "104"
(22) "105" "106" "107" "108" "109"
(23) "110" "111" "112" "113" "114"
(24) "115" "116" "117" "118" "119"
(25) "120" "121" "122" "200" "201"
(26) "202" "203" "204" "205" "206"
(27) "207" "208" "209" "210" "211"
(28) "212" "213" "214" "215" "216"
(29) "217" "218" "219" "220" "300"
(30) "301" "302" "303" "304" "305"
(31) "306" "307" "308" "309" "310"
(32) "311" "312" "313" "314" "315"
(33) "316" "317" "318" "319" "405"
(34) "406" "407" "408" "409" "410"
(35) "411" "412" "413" "414" "415"
(36) "416" "417" "418"
(37) )
(38))
(39)(mapcar 'add_list typelist)
(40)(end_list)

```

**說明：**上述程式用來將所有軸承型號載入列表框中。

```
(41)(action_tile "accept" "(getdata1)(DONE_DIALOG)")
```

**說明：**如按下「確定」鈕，則執行(getdata1)副程式，以擷取對應的軸承資料。

```
(42)(start_dialog)
```

**說明：**啟動交談式視窗。

```
(43)(setq f (open (strcat *fn* "gb276.txt") "r"))
```

**說明：**以讀取的方式打開 gb276.txt 中的軸承資料。

```
(44)(setq rowdata (read-line f))
```

**說明：**讀取資料檔案中的第一行，並將其作為字串存入變數 rowdata 中。

```
(45)(setq blist (txt2list1 rowdata))  
(46)      bt      (nth 0 blist)  
(47))
```

**說明：**呼叫 txt2list1 副程式，將字串 rowdata 變為串列並存入 blist 變數中，然後擷取串列中的第一個資料（即軸承型號）存入變數 bt 中。

```
(48)(while (/= bt btype)  
(49)  (setq rowdata (read-line f))  
(50)  (setq blist (txt2list1 rowdata))  
(51)      bt (nth 0 blist)  
(52)  )  
(53))
```

**說明：**重複讀取軸承資料檔案的每一行，直到此行的第一個資料（軸承型號）與用戶從交談式視窗中所選擇的一致為止。然後，再將此行資料變為串列存入變數 blist 中。

```
(54)(setq c (nth 1 blist))  
(55)      d (nth 2 blist)  
(56)      b (nth 3 blist)  
(57))
```

**說明：**將 blist 中的軸承資料賦予參數 c、d、b 中。

```
(58)(draw1)
```

**說明：**呼叫 draw1 副程式來繪製軸承。

```
(58)(setvar "osmode" osmode)
```

```
(59)(setvar "cmdecho" cmdecho)
```

```
(60)(setvar "blipmode" blipmode)
```

**說明：**恢復系統變數，不影響原有操作。

```
(61)(PRINC)
```

```
(62))
```

## ● 副程式 draw1 部分

**作用：**根據主程式的參數來繪製軸承。

```
(defun draw1 ()
```

**說明：**定義副程式 draw1。

```
(setq pt (getpoint "\n 請輸入軸承插入點:")  
vpt (vlax-3d-point pt)  
)
```

**說明：**提示輸入軸承插入點。

```
(setq ptd (vlax-variant-value vpt))  
(setq ptx (vlax-safearray-get-element ptd 0))  
(setq pty (vlax-safearray-get-element ptd 1))
```

**說明：**擷取插入點的 X 座標和 Y 座標。

```
(setq a (/ (- c d) 2)  
r1 (* a 0.25)  
x0 (+ (/ b 2) ptx)  
y0 (+ pty (- (/ c 2) (/ a 2)))  
x1 (* r1 -1 (sin (/ pi 3)))  
x1 (+ x1 x0)  
y1 (* r1 -1 (cos (/ pi 3)))  
y1 (+ y1 y0)  
x2 (* (- x1 x0) -1)  
x2 (+ x2 x0)  
y2 y1  
x3 x2
```

```

y3  (* (- y1 y0) -1)
y3  (+ y3 y0)
x4  x1
y4  y3
x5  (* b 0.5 -1)
x5  (+ x5 x0)
y5  y1
x6  x5
y6  (* a 0.5 -1)
y6  (+ y6 y0)
x7  (/ b 2)
x7  (+ x7 x0)
y7  y6
x8  x7
y8  y2
x9  x7
y9  y3
x10 x7
y10 (/ a 2)
y10 (+ y10 y0)
x11 x5
y11 y10
x12 x5
y12 y4
x13 x5
y13 (- y6 d)
x14 x5
y14 (- y11 c)
x15 x7
y15 y14
x16 x7
y16 y13
)

```

**說明：**以 X0，Y0 為基點，畫出輔助點的橫座標與縱座標。

```

(setq p1 (list x1 y1)
      p2 (list x2 y2)
      p3 (list x3 y3)
      p4 (list x4 y4)
      p5 (list x5 y5)

```

```
p6 (list x6 y6)
p7 (list x7 y7)
p8 (list x8 y8)
p9 (list x9 y9)
p10 (list x10 y10)
p11 (list x11 y11)
p12 (list x12 y12)
op (list x0 y0)
p13 (list x13 y13)
p14 (list x14 y14)
p15 (list x15 y15)
p16 (list x16 y16)
p17 (list (- x0 r1 -2) y0)
p18 (list (+ x0 r1 -2) y0)
p19 (list x0 (+ y0 r1 -2))
p20 (list x0 (- y0 r1 -2))
)
```

**說明：**計算一些需要的點。

```
(setq radius (distance op p1))
```

```
(setq arc1 (selarc1 p1 p2 radius mspace))
```

```
(setq arc2 (selarc1 p3 p4 radius mspace))
```

**說明：**呼叫副程式 selarc1，以根據兩點及半徑畫弧。

```
(setq lwpline1 (vlax-make-safearray vlax-vbdouble '(0 . 11)))
```

```
(vlax-safearray-fill
```

```
lwpline1
```

```
(list x2 y2 x5 y5 x6 y6 x7 y7 x8 y8 x1 y1)
```

```
)
```

```
(setq pline1 (vla-AddLightWeightPolyline mspace lwpline1))
```

```
(setq lwpline2 (vlax-make-safearray vlax-vbdouble '(0 . 11)))
```

```
(vlax-safearray-fill
```

```
lwpline2
```

```
(list x3 y3 x12 y12 x11 y11 x10 y10 x9 y9 x4 y4)
```

```
)
```

```
(setq pline2 (vla-AddLightWeightPolyline mspace lwpline2))
```



**說明：**根據前面已計算出來的點來畫出聚合線。

```
(setq pName "ANSI31"  
      pType 0  
      bA      T  
      )  
(setq hatch (vla-addhatch mspace pType pName bA))  
(vla-AppendinnerLoop  
  hatch  
  (vla-ObjectList->Variant1 (list pline1 arc1))  
  )  
(vla-put-PatternScale hatch (* b 0.06))
```

**說明：**在這裡我們要講述一個技巧：如果對型號較大和型號較小的圖形均使用相同的填滿比例，則有可能在較大的圖中只能看見距離過近的實體填滿。所以，找出填滿比例與圖形型號的關聯係數，就可在圖案填滿上得到適當的填滿比例。

```
(vla-Evaluate hatch)  
(setq hatch2 (vla-addhatch mspace pType pName bA))  
(vla-AppendOuterLoop  
  hatch2  
  (vla-ObjectList->Variant1 (list pline2 arc2))  
  )  
(vla-put-PatternAngle hatch2 (/ 3.14 2))  
(vla-Evaluate hatch2)  
(vla-put-PatternScale hatch2 (* b 0.06))
```

**說明：**填滿兩塊區域。

```
(vla-regen acaddocument t)
```

**說明：**重生圖形。

```
(setq line1 (vla-AddLine mspace (vlax-3d-point p13) (vlax-3d-point p15))  
      line2 (vla-AddLine mspace (vlax-3d-point p14) (vlax-3d-point p16))  
      )
```

**說明：**繪製二條線。

```
(setq arc3 (selarc1 p4 p1 radius mspace))
```



**說明：**依所輸入的旋轉角度來旋轉圖形。

- 副程式 `getdata1` 部分

**作用：**與 14-1 節的 `getdata` 副程式類似。

```
(defun getdata1 ()  
  (setq li (GET_TILE "btype")  
    list_index (atoi li)  
  )  
  (setq od (nth list_index typelist)  
    btype (atoi od)  
  )  
)
```

- 副程式 `txt2list1` 部分

**作用：**與 14-2 節的 `txt2list` 副程式類似。但包含將資料的字串轉換成數值串列。

```
(defun txt2list1 (rc)  
  (setq p 1  
    sp 1  
    len 1  
    sl (strlen rc)  
    la (list nil)  
  )  
  (while (<= p (1+ sl))  
    (progn  
      (setq temp (substr rc p 1))  
      (if (or (= temp "") (= temp " "))  
        (progn  
          (setq str (substr rc sp len)  
            sp (1+ p)  
            len 0  
            data (atof str)  
            la (append la (list data)))  
        )  
      )  
    )  
  )
```

```

    )
  )
  (setq len (1+ len))
)
(setq p (1+ p))

)
)
(setq la (vl-remove nil la))
)

```

### ● 副程式 selarc1 部分

**作用：**利用兩點（起始點和終點）和半徑畫弧。如下圖例所示，本例關鍵之處即在於確定中點。首先，在三角形 AEC 中利用反正切函數來定出角度  $\alpha_1$ （即程式中的 ang1c），再得出 SE 線的夾角  $\alpha_2$ （即程式中的 ang2c）。顯而易見， $\alpha_3 = \alpha_1 + \alpha_2 - 2\pi$ 。SC 與水平線的夾角即為  $\alpha_3$ ，SC 長度即為半徑。如此，用 polar 函數根據基點 S 定出圓心 C，然後再計算出弧的起始角和終止角，就可畫出弧。

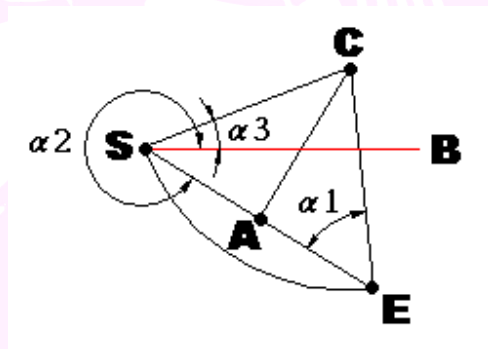


圖 14-9 以二點為半徑畫弧

```

(defun selarc1 (sp ep radius mspace)
  (setq distc (distance sp ep))
  (setq ang1c
    (atan
      (/ (* 2 (sqrt (- (expt radius 2) (expt (/ distc 2) 2))))
      distc
    )
  )
  (setq ang2c (angle sp ep))
  (setq ang3c (- (+ ang1c ang2c) (* 2 pi)))

```

```

(setq cenpt3 (polar sp ang3c radius))
(setq vcenpt3 (vlax-3d-point cenpt3))
(setq ang3 (angle cenpt3 sp)
      ange3 (angle cenpt3 ep)
)

(vla-addarc mspace vcenpt3 radius ang3 ange3)
)

```

- 副程式 vla-ObjectList->Variant1 部分

**作用：**與 14-1 節的範例類似，將 ActiveX 物件串列轉換為變數。

```

(defun vla-ObjectList->Variant1 (ObjectList)
  (List->Variant1
   vlax-VbObject
   ObjectList
  )
)

```

- 副程式 List->Variant1 部分

**作用：**與 14-1 節類似，使用 vla-ObjectList->Variant 傳來的 vtype 物件類型建立一個 inlist 長度的安全陣列(safearray)，並將 inlist 填入這個安全陣列(safearray)，再將其轉換成變式(variant)資料類型。

```

(defun List->Variant1 (vtype inlist)
  (vlax-make-variant
   (vlax-safearray-fill
    (vlax-make-safearray
     vtype
     (cons 0 (1- (length inlist))))
    inlist
  )
)
)

```

- 本範例的執行步驟

假設 bearing.lsp、bearing.dcl 與 gb276.txt 均在：

\\03M\Samples\Volume4\

目錄下，如果您將這些檔案放在另外的目錄裡，那麼本範例中有關存取這些檔案的路徑也要更改（請參考 2-5 節）。然後，再依下步驟執行：

1. 在 AutoCAD 指令提示號後輸入（參考圖 2-22 也可以）：

指令: (load "bearing.lsp") <Enter>

指令: bearing <Enter>

2. 此時，會出現一個交談式視窗，請於填入適當的數值後，按「確定」鈕。
3. 此時，請選擇軸承型號。於出現軸承圖形後，程式還將詢問旋轉角，輸入後程式即可完成。

## 14-3 螺帽的繪製

### ● 本範例完成圖形

本範例將畫出一個如圖 14-10 所示的螺帽：

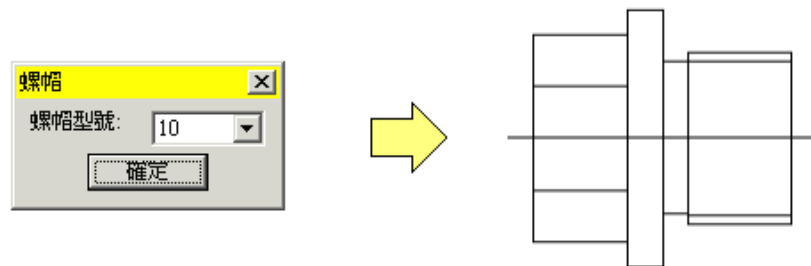


圖 14-10 本程式的螺帽完成圖

本範例共由 4 個函數所組成的。其中，screw\_tuck 為主程式，其餘的 getdata2、txt2list2 與 draw2 為副程式。以下，我們將詳細講述它們的功能以及程式設計的技巧。但是請注意：對前面範例已經講述過的語法和函數將不再重複。

- 本範例檔案：  
screw\_tuck.lsp （本電子書範例光碟(03M)\Samples\Volume4 目錄）  
screw\_tuck.dcl （本電子書範例光碟(03M)\Samples\Volume4 目錄）  
screw\_tuck.txt （本電子書範例光碟(03M)\Samples\Volume4 目錄）

● 主程式 screwtuck 部分

**作用：**呼叫副程式並繪出圖形。

```
(1)(defun c:screwuck ()  
(2)  (setq osmode (getvar "osmode")  
(3)      cmdecho (getvar "cmdecho")  
(4)      blipmode (getvar "blipmode")  
(5)  )
```

**說明：**先儲存一些系統變數。

```
(6)(setvar "osmode" 0)  
(7)(setvar "blipmode" 0)  
(8)(setvar "CMDECHO" 0)
```

**說明：**設定所需的一些系統變數。

```
(9)(vl-load-com)  
(10)(setq acadObject (vlax-get-acad-object))  
(11)(setq acadDocument (vla-get-ActiveDocument acadObject))  
(12)(setq mSpace (vla-get-ModelSpace acadDocument))
```

**說明：**擷取模型空間變數。

```
(13)(setq it_dcl (load_dialog "screwuck.dcl"))  
(14)(if (< it_dcl 0)  
(15)  (exit)  
(16))
```

**說明：**載入螺帽的 DCL 檔案。

```
(17)(if (not (new_dialog "screwuck" it_dcl))  
(18)  (exit)  
(19))
```

**說明：**顯示如圖 14-11 所示的螺帽 DCL 交談式視窗：



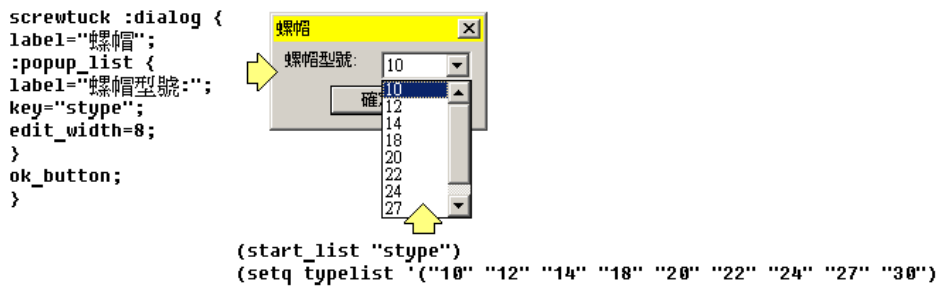


圖 14-11 螺帽的 DCL 交談式視窗

(20)(start\_list "stype")

(21)(setq typelist ("10" "12" "14" "18" "20" "22" "24" "27" "30"))

(22))

(23)(mapcar 'add\_list typelist)

(24)(end\_list)

**說明：**將所有螺帽的型號載入列表框中。

(25)(action\_tile "accept" "(getdata2)(DONE\_DIALOG)")

**說明：**如按下「確定」鈕，就執行 (getdata2)，並擷取對應的螺帽資料。請注意：我們將於此處呼叫 getdata2 副程式，其實從前面範例一路下來，getdata、getdata1 與 getdata2 的意義都是相同的，為了保證三個主程式可單獨執行，且在同時載入又不會相互影響，因此將前二個範例的「getdata」副程式改名。如果 getdata 很常用，且可單獨儲存為一 LISP 檔案，那麼我們就建議您將此段副程式加到名為 acad.lsp 的檔案中，讓其在 AutoCAD 啟動時自動載入，如此，在這三個程式中就均可呼叫 getdata 副程式，同時於它們各自的 lsp 檔案中刪除這些 getdata 副程式。

(26)(start\_dialog)

**說明：**啟動交談式視窗。

(27)(setq f (open (strcat \*fn\* "screw.tuck.txt") "r"))

**說明：**以讀取的方式打開 screw.tuck.txt 中螺帽資料。

(28)(setq rowdata (read-line f))

**說明：**讀取資料檔案中的第一行，將其當作字串存入變數 rowdata 中。

(29)(setq slist (txt2list2 rowdata))

```
(30)      st      (nth 0 slist)
(31))
```

**說明：**呼叫 txt2list2 副程式，將字串 rowdata 轉換為串列並存入 blist 變數中，並擷取串列中的第一個資料（即螺帽型號），以將之存入變數 st 中。其實這裡所呼叫 txt2list2 副程式與前二範例中的 txt2list 以及 txt2list1 副程式相同。原因與 getdata 副程式的狀況相同，請參考上面第(25)條程式的說明。

```
(32)(while (/= st stype)
(33)  (setq rowdata (read-line f))
(34)  (setq slist (txt2list2 rowdata)
(35)      st      (nth 0 slist)
(36)  )
(37))
```

**說明：**以上程式將重複讀取螺帽資料檔案的每一行，直到此行的第一筆資料（螺帽型號）與用戶從交談式視窗中所選擇的一致為止。然後再將此行資料轉換為串列存入變數 slist 中。

```
(38)(setq m(nth 0 slist)
(39)  d1 (nth 1 slist)
(40)  d  (nth 2 slist)
(41)  e  (nth 3 slist)
(42)  l  (nth 4 slist)
(43)  h  (nth 5 slist)
(44)  b  (nth 6 slist)
(45)  b1 (nth 7 slist)
(46))
```

**說明：**將 blist 中的資料賦予螺帽參數。

```
(47)(draw2)
```

**說明：**呼叫 draw2 副程式來繪製螺帽。

```
(48)(setvar "osmode" osmode)
(49)(setvar "cmdecho" cmdecho)
(50)(setvar "blipmode" blipmode)
```

**說明：**還原系統變數。

(51)(PRINC)

(52))

● 副程式 draw2 部分

**作用：**根據主程式的參數來繪製螺帽。

```
(defun draw2 ()  
  (setq pt (getpoint "\n 請輸入螺帽插入點:"))  
  vpt (vlax-3d-point pt)  
)
```

**說明：**提示操作者輸入螺帽插入點。

```
(setq ptd (vlax-variant-value vpt))  
(setq ptx (vlax-safearray-get-element ptd 0))  
(setq pty (vlax-safearray-get-element ptd 1))
```

**說明：**擷取插入點的 X 座標和 Y 座標。

```
(setq x0 ptx  
  y0 pty  
  r (/ b 2)  
  x1 x0  
  y1 (+ y0 (/ D 2))  
  x2 x0  
  y2 (- y1 D)  
  x3 (- x0 b)  
  y3 y2  
  x4 x3  
  y4 y1  
  x5 x4  
  y5 (+ y0 (/ e 2))  
  x6 (- x0 h)  
  y6 y5  
  x7 x6  
  y7 (- y6 e)  
  x8 x5  
  y8 y7  
  x9 x8
```

```

y9 (+ y8 (/ e 4))
x10 x7
y10 y9
x11 x7
y11 (+ y10 (/ e 2))
x12 x9
y12 y11
x13 x0
y13 (- y0 (/ d1 2))
x14 (+ x13 b1)
y14 y13
x15 x14
y15 (- y0 (/ m 2))
x16 (+ x0 (- 1 h))
y16 y15
x17 x16
y17 (+ y16 m)
x18 x15
y18 y17
x19 x15
y19 (+ y0 (/ d1 2))
x20 x0
y20 y19
x21 (+ x0 b1)
y21 (+ y0 (* 0.9 (/ m 2)))
x22 (+ x0 (- 1 h))
y22 y21
x23 x21
y23 (- y0 (* 0.9 (/ m 2)))
x24 x22
y24 y23
x25 (- x0 h 3)
y25 y0
x26 (+ x25 1 6)
y26 y0
)

```

**說明：**以 X0，Y0 為基點，畫出輔助點的橫座標與縱座標。

```

(setq p1 (list x1 y1)
      p2 (list x2 y2)

```

```
p3 (list x3 y3)
p4 (list x4 y4)
p5 (list x5 y5)
p6 (list x6 y6)
p7 (list x7 y7)
p8 (list x8 y8)
p9 (list x9 y9)
p10 (list x10 y10)
p11 (list x11 y11)
p12 (list x12 y12)
p13 (list x13 y13)
p14 (list x14 y14)
p15 (list x15 y15)
p16 (list x16 y16)
p17 (list x17 y17)
p18 (list x18 y18)
p19 (list x19 y19)
p20 (list x20 y20)
p21 (list x21 y21)
p22 (list x22 y22)
p23 (list x23 y23)
p24 (list x24 y24)
p25 (list x25 y25)
p26 (list x26 y26)
)
```

**說明：**計算出需要的點。

```
(setq lwpline1 (vlax-make-safearray vlax-vbdouble '(0 . 9)))
(vlax-safearray-fill
 lwpline1
 (list x1 y1 x2 y2 x3 y3 x4 y4 x1 y1)
)
(setq pline1 (vla-AddLightWeightPolyline mspace lwpline1))
(setq lwpline2 (vlax-make-safearray vlax-vbdouble '(0 . 7)))
(vlax-safearray-fill
 lwpline2
 (list x5 y5 x6 y6 x7 y7 x8 y8)
)
(setq pline2 (vla-AddLightWeightPolyline mspace lwpline2))
(setq lwpline3 (vlax-make-safearray vlax-vbdouble '(0 . 15)))
```

```

(vlax-safearray-fill
  lwpline3
  (list x13 y13 x14 y14 x15 y15 x16 y16 x17 y17 x18 y18 x19 y19 x20
        y20)
)
(setq pline3 (vla-AddLightWeightPolyline mspace lwpline3))

(setq line1 (vla-addline mspace (vlax-3d-point p11) (vlax-3d-point p12))
  line2 (vla-addline mspace (vlax-3d-point p9) (vlax-3d-point p10))
  line3 (vla-addline mspace (vlax-3d-point p14) (vlax-3d-point p19))
  line4 (vla-addline mspace (vlax-3d-point p21) (vlax-3d-point p22))
  line5 (vla-addline mspace (vlax-3d-point p23) (vlax-3d-point p24))
  line6 (vla-addline mspace (vlax-3d-point p25) (vlax-3d-point p26))
)

```

**說明：**由於螺帽是由線段所組成的，所以上示程式將畫出需要的直線與聚合線。

```
(setq ang (getangle pt "\n 請輸入螺帽旋轉角:"))
```

**說明：**提示操作者輸入旋轉角。

```

(mapcar 'vla-rotate
  (list pline1 pline2 pline3 line1 line2 line3 line4 line5 line6)
  (list vpt vpt vpt vpt vpt vpt vpt vpt vpt vpt)
  (list ang ang ang ang ang ang ang ang ang ang)
)
)

```

**說明：**依所輸入的旋轉角來旋轉圖形。

## ● 副程式 getdata2 部分

**作用：**與 14-1 節中的 getdata 副程式相同。

```

(defun getdata2 ()
  (setq li (GET_TILE "stype")
    list_index (atoi li)
  )
  (setq od (nth list_index typelist)
    stype (atoi od)
  )
)

```

```
)  
)
```

- 副程式 txt2list2 部分

**作用：**與 14-1 節中的 txt2list 副程式相同，並將包含資料的字串轉換成數值串列。

```
(defun txt2list2 (rc)  
  
  (setq p 1  
        sp 1  
        len 1  
        sl (strlen rc)  
        la (list nil))  
  )  
  (while (<= p (1+ sl))  
    (progn  
      (setq temp (substr rc p 1))  
      (if (or (= temp "") (= temp " "))  
          (progn  
            (setq str (substr rc sp len)  
                  sp (1+ p)  
                  len 0  
                  data (atof str)  
                  la (append la (list data)))  
            )  
          )  
      (setq len (1+ len))  
    )  
    (setq p (1+ p))  
  )  
  )  
  (setq la (vl-remove nil la))  
  )  
)
```

- 本範例的執行步驟

假設 screwtuck.lsp、screwuck.dcl 和 screwtuck.txt 均在：



\\03M\Samples\Volume4\

目錄下，如果您將這些檔案放在另外的目錄裡，那麼本範例中有關存取這些檔案的路徑也要更改（請參考 2-5 節）。然後，再依下步驟執行：

1. 在 AutoCAD 指令提示號後輸入（參考圖 2-22 也可以）：

指令: (load "screw.tuck.lsp") <Enter>

指令: screw.tuck <Enter>

2. 此時，會出現一個交談式視窗，請於填入適當的數值後，按「確定」鈕。
3. 此時，請選擇螺帽型號。於出現螺帽圖形後，程式還將詢問旋轉角，輸入後程式即可完成。

## 14-4 跟著圓一起“動”的圓中心線

看過以前龍震工作室有關 LISP 書的讀友一定還記得：CCEN.LSP 那個畫圓中心線的程式，但是這個範例將給大家一個新感覺，因為這個圓中心線將跟著圓“動”起來。和以往 AutoLISP 不同的是：由於現在的 Visual LISP 已可支援 ActiveX，而 ActiveX 其中的一個很重要的功能就是「事件反應器」。「事件反應器」就是附加在 AutoCAD 圖素上的特殊物件，該物件的主要功能是：當用戶對附加有事件反應器的圖形物件進行操作時，如修改、移動、編輯、選擇..等，AutoCAD 就可自動識別這些事件並執行與該事件相關的既定程式（這類程式又稱為「回授函數」）。簡言之，就是如果用戶更改了某一圖素的屬性，如果在該圖形上附有事件反應器，那麼 AutoCAD 就能識別該圖素被改變，並執行對應的回授函數。在本範例中，更改圓的大小或移動圓都會導致圓的中心線跟著圓來縮放或移動。其原因就是因為圓上附有事件反應器。事實上，這個範例是屬典型且簡單的，相信讀者瞭解後，就可以舉一反三撰寫出更好的程式。例如，在設計某個的零件時，改動零件的某一個參數就可讓其他的相關尺寸也自動地發生變化。下面就讓我們來學習這個範例。

### ● 本範例完成圖形

本範例將對任意大小的圓畫出一個適當比例的中心線圖形（這沒什麼），同時當您變更圓的半徑或移動圓時，此中心線也會跟著一起做改變或移動（這就利害了！）：

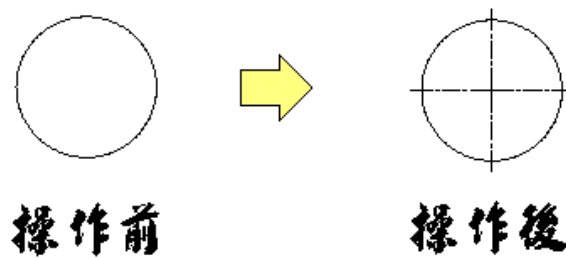


圖 14-12 本程式的完成圖

本範例共由 4 個函數所組成，其中 ccen 為主程式，其餘的 load-line-types、find-line-types 與 mark 為副程式。以下，我們將詳細講述它們的功能以及程式設計的技巧。但是請注意：對前面範例已經講述過的語法和函數將不再重複。

- 本範例檔案：ccen.lsp （本電子書範例光碟(03M)SamplesVolume4 目錄）

- 主程式 ccen 部分

作用：畫出圓的中心線並在其上附加上事件反應器。

(1)(defun C:CCEN ()

說明：定義主程式。

(2)(vl-load-com)

說明：載入 Viual LISP 延伸功能。

(3)(setq acadObject (vlax-get-acad-object))

(4)(setq acadDocument (vla-get-ActiveDocument acadObject))

(5)(setq mSpace (vla-get-ModelSpace acadDocument))

(6)(setq util (vla-get-Utility acadDocument))

(7)(setq lts (vla-get-Linetypes acadDocument))

說明：擷取模型空間變數、Utility 變數與線型變數。

(8)(setq selsets (vla-get-SelectionSets acadDocument))

說明：擷取目前圖形中的選擇組。

以下程式目的是為所選取的圖素做一些準備工作：

```
(9)(setq i (vla-get-count selsets))
```

**說明：**將選擇組中的圖素數存入變數 `i` 中。

```
(10)(while (> i 0)
(11)  (setq sset(vla-item selsets 0))
(12)  (vla-delete sset)
(13)  (setq i (- i 1))
(14) )
```

**說明：**這個迴圈程式是說：如果選擇組中有圖素存在，則將其從選擇組中刪除。  
注意！這並不會從圖面上真正刪除圖素。

```
(15)(setq sset (vla-add selsets "sset"))
```

**說明：**新建選擇組 `sset`。如果沒有以上的準備工作，則在下一次執行程式時，由於 `sset` 選擇組已經存在，執行到此就無法再新建一個同名選擇組，程式會提示一錯誤訊息。

```
(16)(vla-SelectOnScreen sset)
(17)(setq notallcircle nil)
```

**說明：**設定變數 `notallcircle` 來判斷是否所選物件均為圓。

```
(18)(setq ssetcount (vla-get-count sset))
```

**說明：**擷取選擇組中圖素的數量。

```
(19)(while (> ssetcount 0)
(20)  (setq obj (vla-item sset (- ssetcount 1)))
(21)  (setq objname (vla-get-objectname obj))
(22)  (if (/= objname "AcDbCircle")
(23)    (setq notallcircle t)
(24)  )
(25)  (setq ssetcount (- ssetcount 1))
(26) )
```

**說明：**上面的迴圈程式將用來判斷是否所選的圖素均為圓（用圖素名判斷），若非，設定變數 `notallcircle` 為 `t`。

```
(27)(while (and (vla-get-count sset) notallcircle)
(28) (prompt "所選圖素中至少有一非圓的圖素，請再試一次，或按 ESC 結束！
")
(29) (vla-clear sset)
(30) (vla-SelectOnScreen sset)
(31) (setq notallcircle nil)
(32)
(33) (setq ssetcount (vla-get-count sset))
(34)
(35) (while (> ssetcount 0)
(36)   (setq obj (vla-item sset (- ssetcount 1)))
(37)   (setq objname (vla-get-objectname obj))
(38)   (if (/= objname "AcDbCircle")
(39)     (setq notallcircle t)
(40)   )
(41)   (setq ssetcount (- ssetcount 1))
(42) )
(43))
```

**說明：**上示程式的作用為：若選擇組為空的或所選圖素非均為圓，則出現一錯誤訊息，並要求重新輸入。

```
(44)(setq circ_d (vla-get-Radius obj))
```

**說明：**擷取圓半徑。

```
(45)(setq circ_cen (vla-get-center obj))
```

**說明：**擷取圓心。

```
(46)(setq pt (vla-PolarPoint util circ_cen 0 (+ 5 circ_d)))
```

```
(47)(setq line (vla-addline mspace circ_cen pt))
```

**說明：**畫 1/4 中心線。

```
(48)(load-line-types "CENTER" "acad.lin")
```

```
(49)(vla-put-Linetype line "CENTER")
```

**說明：**將中心線線型設定為 "CENTER"。

```
(50)(setq lts (/ circ_d 5))  
(51)(vla-put-LinetypeScale line lts)
```

**說明：**設定線型比例。

```
(52)(setq  
(53) linearray (vla-ArrayPolar line 4 (/ (* pi 2 (1- 4)) 4) circ_cen)  
(54))
```

**說明：**對這條中心線做陣列，使之完整。

```
(55)(vla-delete sset)
```

**說明：**刪除選擇組以釋放記憶體空間。

```
(56)(setq circleReactor  
(57) (VLR-Object-Reactor  
(58) (list obj)  
(59) "Circle Reactor"  
(60) '(:VLR-modified . mark))  
(61) )  
(62))  
(63))
```

**說明：**設定反應器，並將反應器物件添加到圖形資料庫中。其標準語法為：

```
(vlr-object-reactor owners data callbacks)
```

其中，參數：

- **owners**：表示 VLA 物件的 AutoLISP 串列，標識要觀察的圖形物件。請注意這裡一定要使用 VLA 物件，若不是，請使用 VLAX-ENAME->VLA-Object 函數轉換。本範例中為圓物件 obj。
- **Data**：任意要與反應器物件關聯的 AutoLISP 資料。如果不包含 dat 參數，則為 nil。請注意：這裡就不一定要使用 VLA 物件，EntName 也可以。本範例中將省略此參數。
- **callbacks**：串列，其中將包含格式：（事件名稱·回授函數）的點對。本例中的回授函數為 mark。

## ● 副程式 load-line-types 部分

作用：與 14-1 節的 load-line-types 副程式相似，用於呼叫線型。

```
(defun load-line-types (line-type file-name / tmp res)
  (if (and (setq tmp (vlax-get-acad-object))
          (setq tmp (vla-get-activedocument tmp))
          (setq tmp (vla-get-linetypes tmp))
      )
      (if (setq res (find-line-type line-type tmp))
          res
          (progn
            (vla-load tmp line-type file-name)
            (if (vla-item tmp line-type)
                (vla-item tmp line-type)
                nil
            )
          )
      )
      nil
  )
)
```

#### ● 副程式 find-line-types 部分

作用：與 14-1 節的 find-line-types 副程式相似，用於尋找線型。

```
(defun find-line-type (line-type line-type-collection / res)
  (setq line-type (strcase line-type))
  (vlax-for l-obj line-type-collection
    (if (= (strcase (vla-get-name l-obj)) line-type)
        (setq res l-obj)
    )
  )
  res
)
```

#### ● 副程式 mark 部分

作用：即回授函數，在圖發生大小變化或位置移動時就會激發此副程式。其目



的是使圓的中心線也發生移動。它的動作是：先刪除舊的中心線，再添加新的中心線。當然，也可以將之寫成四條中心線的起點和終點發生對應的變化。

```
(defun mark (notifier-object reactor-object parameter-list)
  (vl-load-com)

  (setq circ_d (vla-get-Radius obj))
  (setq circ_cen (vla-get-center obj))
  (setq pt (vla-PolarPoint util circ_cen 0 (+ 5 circ_d)))
  (vla-delete line)
  (setq linesafearray (vlax-variant-value linearray))
  (vla-delete (vlax-safearray-get-element linesafearray 0))
  (vla-delete (vlax-safearray-get-element linesafearray 1))
  (vla-delete (vlax-safearray-get-element linesafearray 2))
  (setq line (vla-addline mspace circ_cen pt))
  (load-line-types "CENTER" "acad.lin")
  (vla-put-Linetype line "CENTER")
  (setq lts (/ circ_d 5))
  (vla-put-LinetypeScale line lts)
  (setq
linearray (vla-ArrayPolar line 4 (/ (* pi 2 (1- 4)) 4) circ_cen)
)
)
```

#### ● 本範例的執行步驟

假設 ccen.lsp 被置於：

**\\03M)Samples\Volume4\**

目錄下，如果您將這些檔案放在另外的目錄裡，那麼本範例中有關存取這些檔案的路徑也要更改（請參考 2-5 節）。然後，再依下步驟執行：

1. 在 AutoCAD 指令提示號後輸入（參考圖 2-22 也可以）：

指令: (load "ccen.lsp") <Enter>

2. 先在 AutoCAD 中先畫一個圓，再於 AutoCAD 指令提示號後輸入：

指令: ccen <Enter>

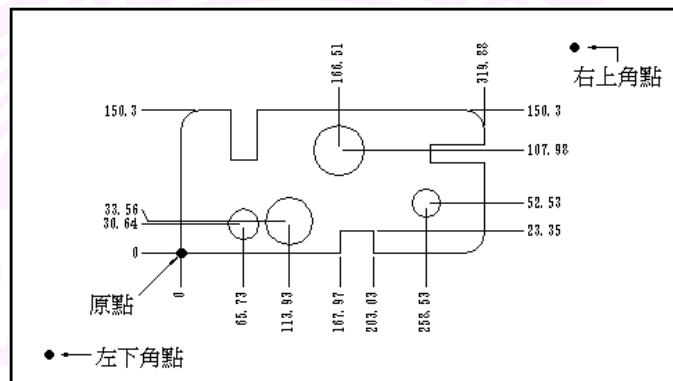


3. 選擇圓並按滑鼠右鍵。
4. 當您使用 PROPERTIES 指令來改變圓的半徑或使用 MOVE 來移動圓時，此中心線也會跟著一起改變或移動。

## 啟發性習題

### 實作題

1. 請以座標標示法設計一模具尺寸自動標示功能（以 AutoLISP 撰寫）。如下圖例所示：



設計條件：

- (1) 請鍵入原點:
- (2) 請點取可包含欲標示圖形窗框的左下角點:
- (3) 請點取可包含欲標示圖形窗框的右下角點:
- (4) 請輸入繪圖比例 <1>: <Enter>
- (5) 請指定標示字的字高 <mm>: 10 <Enter>
- (6) 請輸入尺寸標示的小數點精確位數: 2 <Enter>

解答檔案名稱：AUTODIM.LSP

搭配範例繪圖檔案：ADDEMO.DWG（即先叫入此範例圖檔，再載入並執行 AUTODIM）

2. 請設計一個立體彈簧。（以 AutoLISP 撰寫）。

設計詢問句：

- (1) Spring center point: （彈簧中心點）
- (2) <Spring radius>/Diameter: （彈簧半徑）

- (3) <Tube radius>/Diameter: (圓管半徑)
- (4) Height per rotation: 75 (每圈高度)
- (5) Number of rotations: 3 (旋轉圈數)
- (6) Number of radial segments (8-24) <16>: (圓管幅射精度)
- (7) Number of tube segments (8-24) <16>: (圓管段精度)

解答檔案名稱：3DSPRING.LSP

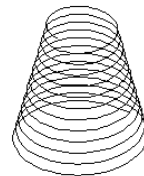
完成圖例：



3. 上一個習題是以立體分段的方式來撰寫。而這題，請以螺旋線方程式來設計一個立體螺旋線。(以 AutoLISP 撰寫)。

設計詢問句：

彈簧起始直徑: 24 <Enter>  
 彈簧終點直徑: 12 <Enter>  
 彈簧總高(每圈高度 X 圈數): 25 <Enter> {2 X 12.5}  
 總角度數(360 X 圈數): 4500 <Enter> {360 X 12.5}  
 節點數([總角度數/間隔角度]\*1): 751 <Enter> {[4500/6]\*1}



解答檔案名稱：SPL.LSP