

# 第十一章 物件

物件就是事物，它在程式設計中之所以有用，是因為程式設計者可以無須知道事物的構成以及內部工作情況而在程式中使用它們。這將給程式設計師帶來了極大的方便。

Visual LISP 與 ActiveX、ObjectARX 和 VBA 一樣都是物件導向的語言的。透過 Autodesk 公司開發的 Visual LISP 的 ActiveX 物件，AutoCAD 物件模型在交叉應用整合方面將具有更好的適應性，這就使用戶所開發的應用程式不僅相容於 AutoCAD，而且同其他使用 ActiveX 的應用程式一樣相容（透過聯合資料庫），如此將解決了應用程式彼此間智慧化、整合化的問題。

您將在本章中學到：

●AutoCAD 物件模型

●在 Visual LISP 中使用 ActiveX 物件



## 11-1 AutoCAD 物件模型

在物件程式撰寫中，首先您要瞭解並掌握物件的屬性、方法和事件，才能在程式的編寫中靈活地運用。

### 11-1-1 物件的屬性



AutoCAD 物件模型中的物件均具有一個或多個屬性。以圓為例，圓可以用半徑、圓心來描述。所有的特徵就稱為「屬性」。對於一個圓來說，它就有：Center，Application，Area，Circumference，Color，Diameter，Document，Handle，HasExtensionDictionary，Hyperlinks，Layer，Linetype，LinetypeScale，Lineweight，Normal，ObjectID，OwnerID，PlotStyleName，Radius，Thickness，Visible 這些屬性。在 ActiveX 函數中要獲得 AutoCAD 圖形物件的有關特徵就要用到對應的屬性名稱。由於不同的物件就有不同的屬性，所以您可以在 AutoCAD 的線上說明文件中查到所有詳細的資料。

### 11-1-2 物件的方法

物件還包含方法，方法是用來指定物件的執行方式。ActiveX 物件也有對應的方法，即可知道要對 ActiveX 的物件做如何的操作。如圓的方法就有：

ArrayPolar，ArrayRectangular，Copy，Delete，GetBoundingBox，GetExtensionDictionary，GetXData，HighlightIntersectWith，Mirror，Mirror3D，Move，Offset，Rotate，Rotate3D，ScaleEntity，SetXData，TransformBy 等。

在 Visual LISP 中，ActiveX 方法的履行是透過一系列新增的 Visual LISP 函數（即 vl 開頭的函數），您也可在線上說明文件中悉數查詢到該類函數的詳細語法資訊。通常，在方法設定任務之前都會需要資訊，而這些資訊就是參數。例如，如果要將圓半徑改為 5，那麼 5 就是參數。

### 11-1-3 物件的事件

事件就是程式執行所需要的條件。一旦某個事件被啟動，系統就會執行對應的程式碼，這在以後的程式中我們可以經常看到。

AutoCAD ActiveX 中新增了事件驅動。當 AutoCAD 的目前狀態發生變化時或某些事件發生後，AutoCAD 就會激發對應的事件與消息，用戶可以在事件處理中加入自定義的事件處理副程式。在 AutoCAD 中共有三類 40 個事件：

#### 1. 應用程式層次事件

此類事件如：回應指令行指令、回應系統變數的改變、單文字檔下回應新建、打開、儲存、關閉、列印文件，回應載入和卸載 LISP 與 ARX 程式。如，Vlr-CommandEnded 事件。



## 2. 文字檔層次事件

如：回應新建、刪除、修改物體、回應快顯功能表、物件選擇、重新產生視圖，多文字檔下回應載入與卸載 LISP 與 ARX 程式。如，Vlr-rxAppLoaded 事件。

## 3. 物件層次事件

當實體物體被修改時就會激發此事件，如：vlr-modified，vlr-erased 事件。

### 11-1-4 物件的集合

值得一提的是，在 AutoCAD 的物件模型中，所有物件都是以集合來分組的，例如選擇組 selectionsets 就是圖形中所有選取圖形的集合，圖塊組 blocks 就是 AutoCAD 圖形中所有圖塊的集合。因此，在存取選擇組或圖塊時均要從物件的集合中呼叫出來。有關選擇組的操作請參考下一章。

### 11-1-5 物件的使用

物件操作的語法如下：（以 VBA 為例）

#### ● 設定物件 A 屬性

物件 A.屬性=某個輸入的數值

circle.radius=5（設定圓的半徑為 5）

#### ● 擷取物件 A 屬性

變數=物件 A.屬性

radius=circle.radius（將圓的半徑存入 radius 變數中）

#### ● 如果這個物件 A 具有這種方法 I，就執行某個操作

物件 A.方法 I

circle.delete（刪除圓）



### 11-1-6 AutoCAD 的物件結構

如圖 11-1 所示，AutoCAD 中的物件是以階層式結構來組織的，其根部是應用程式物件。物件的繼承階層結構就稱為「物件模型」，物件模型將顯示物件的階層關係，各級屬性的使用必須依照該物件模型中的繼承階層關係來運作。

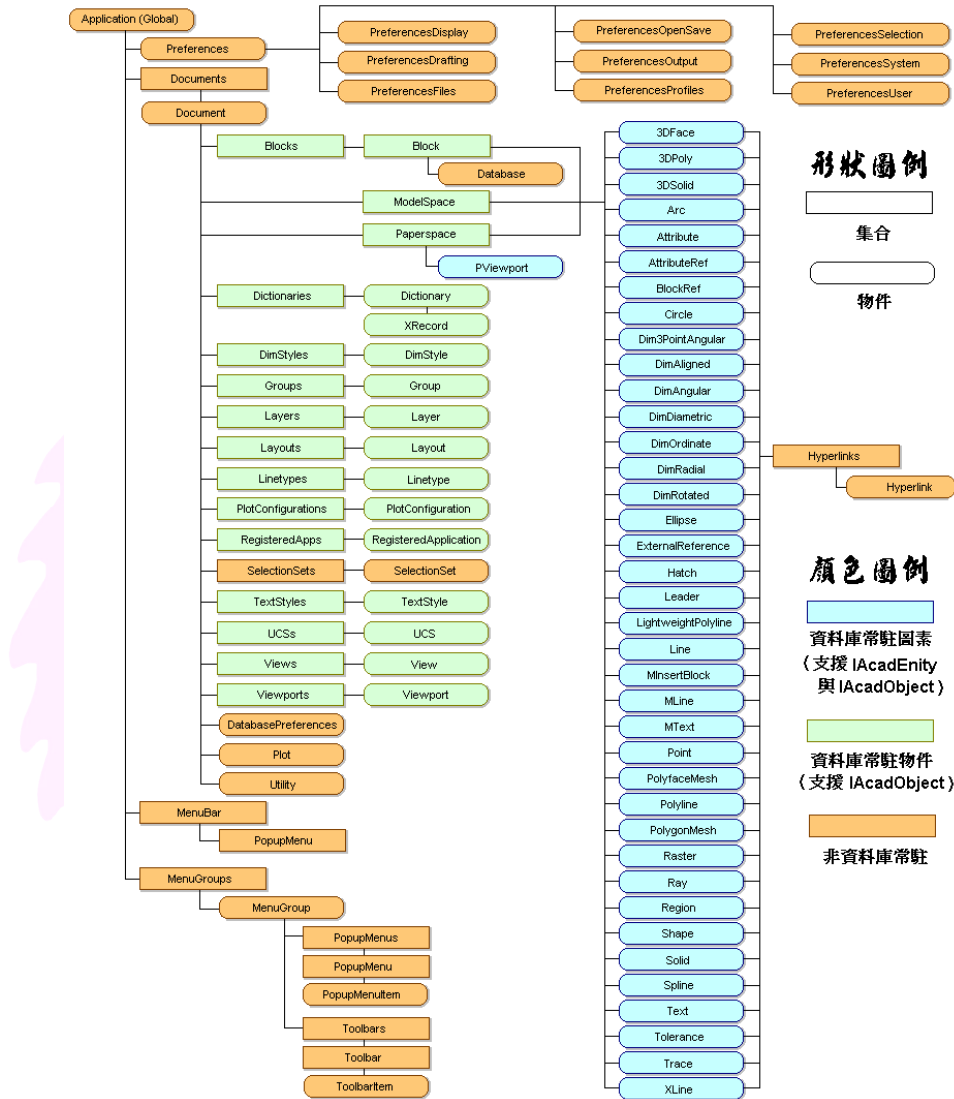


圖 11-1 ActiveX 的物件模型結構

最高層物件是 AutoCAD Application，它被用來管理 AutoCAD 的應用程式，它將包含以下幾個物件：

- Preferences 集合物件（即 Preferences 設定視窗）
- Documents 集合物件（用來處理目前活動的圖形，它包含多個子物件，其中有圖形檔案所有的物件集合，輔助工具群等）



- MenuBar、MenuGroups 集合物件（用來針對操作功能表與工具欄）

## 11-2 在 Visual LISP 中使用 ActiveX 物件

Visual LISP 將提供一系列函數來處理 ActiveX 物件，這些函數名稱都是以“vla-”為字首的。例如，vla-addLine、vla-get-Color、vla-put-Color 等。換句話說，字首為“vla-”的函數都是 ActiveX 函數，它們將對 AutoCAD 資料物件呼叫對應的方法來對其屬性進行操作。

### 11-2-1 ActiveX 物件的分類

依上所述，這些在 VBA 裡的函數可依其使用物件的方法來進一步分類：

- vla-函數與每個 ActiveX 方法相對應。我們可用這些函數來呼叫 ActiveX 方法（如，以 vla-addCircle 呼叫 AddCircle 方法）。
- 設定物件的屬性。vla-put- 函數和每個特性相對應，可更新特性的值（如，以 vla-put-Color 來更新物件的顏色特性）。
- 取得物件的屬性。vla-get- 函數和每個特性相對應，可擷取 ActiveX 物件特性的值（如，以 vla-get-Color 來擷取物件的顏色特性）。

Visual LISP 還提供了一些和 ActiveX 有關的函數，其字首為 vlax-。這是一些更綜合的 ActiveX 函數，它們可被應用到許多方法、物件或特性。譬如說：在 ActiveX 的支援下，Visual LISP 就可以像 VBA 一樣存取 Microsoft Word、Excel 等應用程式，以進行不同程式之間的資料共用。利用 vlax-get-property 函數，可擷取任意 ActiveX 物件的任意特性。如果圖形包括自定義的 ActiveX 物件，或需要從其他應用程式中存取物件，我們也可使用如 vlax-invoke-method、vlax-get-property 和 vlax-put-property 這類的函數來存取物件的方法和特性。

在 AutoCAD 提供的 ActiveX Automation 線上說明文件中將詳細介紹了所有“vla-”類函數所涉及的物件屬性和方法，但是均以 VBA 的形式來說明的，在下節中，我們將說明如何在 Visual LISP 中使用 ActiveX 函數。

### 11-2-2 如何呼叫 Visual LISP 所提供的 ActiveX 函數

Visual LISP 中的 ActiveX 函數將提供我們運用 ActiveX 物件方法的途徑，



下面我們以在圖形中加入一個圓為例，來說明如何利用 ActiveX 物件的方法：

```
(setq myCircle(vla-addCircle *mSpace* (vlax-3d-point '(5.0 5.0 0)) 3.0))  
#<VLA-OBJECT IAcadCircle 00f0c504>
```

這兩條語法將表示利用 `vla-addCircle` 函數繪製圓。其中 `*mSpace*` 是指向模型空間的 VLA 變數。

如果您不知道如何在 AutoCAD 圖形中利用 ActiveX 類函數來畫圓，那麼您可以參考 ActiveX Automation 線上說明文件，查詢利用 ActiveX 如何定義圓圖形物件。圖 11-2 就是查詢圓物件定義的操作：

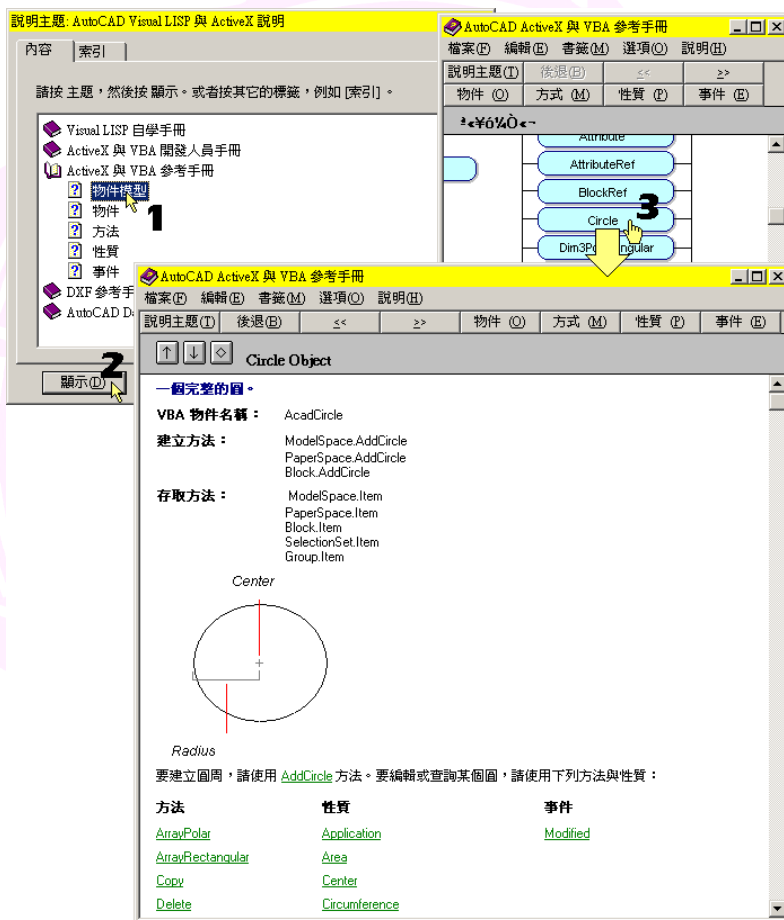


圖 11-2 查詢圓物件定義的操作

在圖 11-2 的圖形物件的說明中都有很多文字說明，列出了物件所擁有的方法、屬性和事件。如果找到了需要的方法，那麼在方法名稱前加上“vla-”字首即為執行該方法的 Visual LISP 函數。

注意：在 Visual LISP 中，函數名稱不區分大小寫，“vla-addcircle”與“vla-AddCirle”是同一個函數。



找到與所需方法相對應的 Visual LISP 函數後，要如何呼叫這些函數呢？目前在 Visual LISP 語法中並沒有詳細闡述這類函數的呼叫方法，開發人員必須在 ActiveX Automation 說明文件中查詢與應方法對應的函數呼叫方法。例如，在模型空間集合的物件說明文件中可單擊有底線的「AddCircle」項目來查看該方法的定義，如圖 11-3 所示：



圖 11-3 「AddCircle」方法的資訊查詢

您也可以在 ActiveX Automation 和說明視窗中單擊頂部的「方式(M)」鈕，在方法列表中選擇「AddCircle」方法來擷得該方法的使用說明，如圖 11-4 所示。



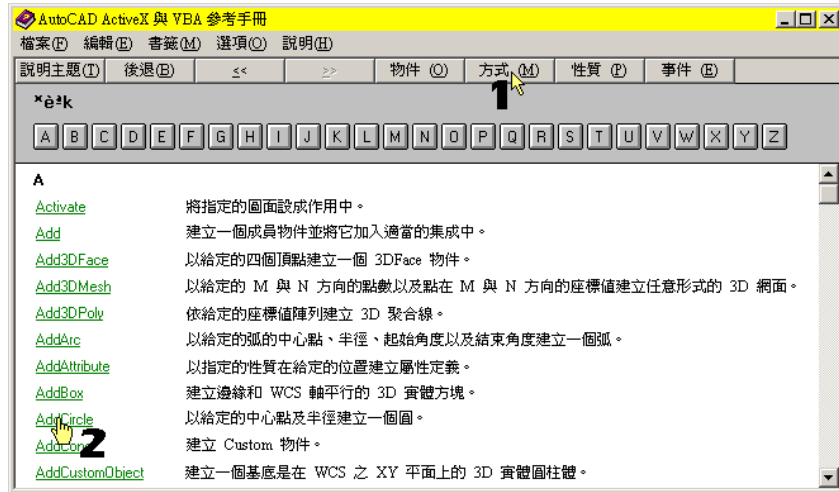


圖 11-4 「AddCircle」方法的另一種查詢方式

從圖 11-4 可看出：在 Automation 說明文件中，方法的執行是用 VBA 來定義的。例如，對於 AddCircle 方法可定義如下：

```
RetVal=object.AddCircle(Center,Radius)
```

在一般的具體應用中，如果要在模型空間中以 5,5 為圓心，3 為半徑畫圓，請輸入以下語法：

```
Dim Center(0 to 2) As Double, Radius as Double
Center(0)=5.0 : Center(1)=5.0 : Center(2)=0
Radius=3.0
Set myCircle=Thisdrawing.ModelSpace.AddCircle(Center,Radius)
```

在 VBA 中，傳回值是前置的，即 myCircle，傳回值是在 Automation 說明文件中所定義的圓物件。請注意：賦予值給物件變數時要用 Set。相對地，在 Visual LISP 中是用下面的方式來呼叫方法的：

```
(vl-load-com)
(setq *acadObject* (vlax-get-acad-object))
(setq *acadDocument* (vla-get-ActiveDocument *acadObject*))
(setq *mSpace* (vla-get-ModelSpace *acadDocument*))
(setq center (vlax-make-safearray vlax-vbDouble ' (0 . 2)))
(vlax-safearray-fill center ' (5.0 5.0 0))
(setq radius 3.0)
(setq myCircle(vla-addCircle *mSpace* center radius))
```



在 Visual LISP 中，ActiveX 函數傳回值 myCircle 是 AutoCAD 物件，亦即 VLA 物件資料類型。

方法名稱 (object.AddCirCle) 前所引用的物件 object，在 vla 函數呼叫時，通常是第一個參數，該物件是目前要修改或處理的物件。在此範例中就是模型空間 ModelSpace。因此，在 Visual LISP 中應該要這樣寫：

```
(vla-addCircle *mSpace*...)
```

\*mSpace\* 指的是目前的模型空間物件。物件具有階層關係，模型空間物件的階層性如圖 11-5 所示：

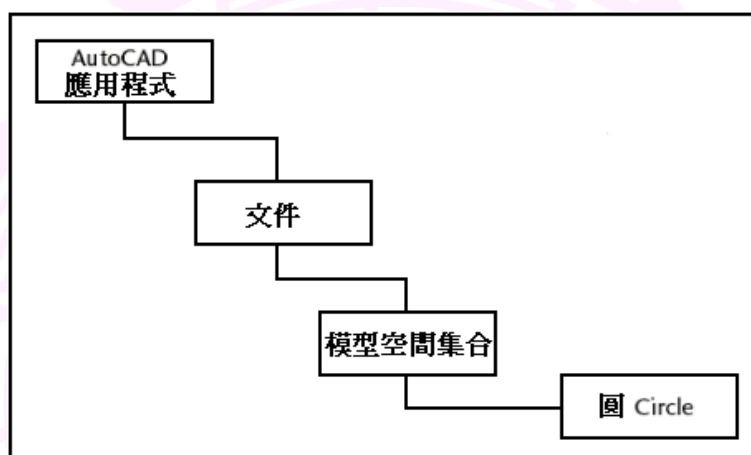


圖 11-5 模型空間物件的階層性

根據物件的繼承性，我們可以逐個處理各階層的屬性。在上例中，我們已提出了處理目前圖形中模型空間的途徑。在 VBA 中，Thisdrawing 將代表目前文檔（即 Document），Thisdrawing.Document 即取得模型空間集合物件。

在 Visual LISP 中，先以 (setq \*acadObject\* (vlax-get-acad-object)) 來取得 AutoCAD 應用程式物件；以 (setq \*acadDocument\* (vla-get-ActiveDocument \*acadObject\*)) 來取得目前的文檔；以 (setq \*mSpace\* (vla-get-ModelSpace \*acadDocument\*)) 來取得模型空間集合物件。

### 11-3 Visual LISP 中的 ActiveX 變數

本節將為您介紹有關 Visual LISP 中所使用的各種 ActiveX 變數。



### 11-3-1 安全陣列

讓我們再仔細分析一下上例，以瞭解如何將 VBA 變數轉換成 Visual LISP 變數：

```
Dim Center(0 to 2) As Double, Radius as Double  
Center(0)=5.0 : Center(1)=5.0 : Center(2)=0  
Radius=3.0
```

在 VBA 範例中，利用 ActiveX 方法向 AutoCAD 的圖形模型空間添加一個圓物件時引用了兩個變數 Center 與 Radius。其中，Center 是一維陣列，由三個雙精度陣列組成，這將在 WCS 座標系統中定義了圓心座標。Radius 是非負值雙精度數，用來定義圓的半徑。

```
(setq center (vlax-make-safearray vlax-vbDouble ' (0 . 2)))  
(vlax-safearray-fill center ' (5.0 5.0 0))  
(setq radius 3.0)
```

在 Visual LISP 中，我們使用了以 vla- 為字首的函數 vla-AddCircle，這類函數運算元的資料類型是 ActiveX 的資料類型。在此範例中，center 的資料類型是安全陣列（safearray）。如果要在主控台視窗中查看 center，將得到：

```
_$ center  
#<safearray...>
```

(setq center (vlax-make-safearray vlax-vbDouble ' (0 . 2))) 將建立一個一維的，由三個雙精度陣列成的空安全陣列，vlax-vbDouble 表示雙精度，其中點對' (0 . 2) 表示陣列下標從 0 到 2。此句相當於 VBA 中的：

```
Dim Center(0 to 2) As Double
```

(vlax-safearray-fill center ' (5.0 5.0 0)) 會向剛建立的空陣列中填入資料，相當於 VBA 中的：

```
Center(0)=5.0 : Center(1)=5.0 : Center(2)=0
```

安全陣列是 ActiveX 的資料類型，可在 Visual LISP 中向 vla- 類函數傳遞值。但是請注意：radius 卻是 Visual LISP 中的實數（Real）類型的數值，但它



也可以向 vla- 類函數傳遞值。在傳遞的過程中，radius 將自動轉換為 ActiveX 所需的雙精度數（vlax-vbDouble）。

### 11-3-2 變數的轉換

要在 Visual LISP 中順利運用 ActiveX 函數，我們必須依 ActiveX 方法定義裡的變數類型來提供對應的 LISP 變數類型。如果發生錯誤（例如，需要整數型變數時卻傳遞了實數型變數），將導致應用程式當機。

例如，當您利用 command 與 entmake 函數來產生 AutoCAD 圖素時，Visual LISP 就可提供靈活的點座標表示方法（也就是說：點可以是二維點也可以是三維點）。在利用 ActiveX 函數時，我們也必須保證參數傳遞的正確性，使用 vlax-3d-point 函數，就可以將表示二維點或三維點的串列、座標值轉換成有效的三維點。該函數的語法如下：

```
(vlax-3d-point <list>)  
(vlax-3d-point x y [z])
```

例如：

```
_$ (setq aPoint (vlax-3D-point (list 5 5)))  
(5.0 5.0 0.0)  
_$ (setq aPoint (vlax-3d-point 5 5))  
(5.0 5.0 0.0)
```

vlax-tmatrix 函數可以將模型有效地轉換到通用轉換矩陣中，從而可以被 vla-TransformBy 函數所呼叫。該函數還可以將由四個串列所定義的轉換矩陣中的每一元素轉換成實數，如下所示：

```
_$ (vlax-tmatrix (list (list 1 1 1 0)(list 1 2 3 0)(list 2 3 4 5)(list 2 9 8 3)))  
#<variant 8197 ...>
```

表 11-1 將列出 ActiveX 函數針對要求的 ActiveX 資料類型所能接受的 Visual LISP 資料類型。串列的每一行都代表 ActiveX 函數所用的一種資料類型，每一列則代表一種 Visual LISP 的資料類型。其中一些可以自動轉換，如前述的實數型 Real 轉換為雙精度型 vlax-vbDouble。而有一些則需要函數轉換，例如，一次轉換：圖素名稱到 VLA 物件的轉換要使用 vlax-ename->vla-object 函數；二次轉換：圖素名稱到變式要先將其轉換為 VLA 物件，再用 vlax-make-variant 函數轉



換為變式變數。

表 11-1 ActiveX 資料類型所能接受的 Visual LISP 資料類型

nil(0)/	整數	實數	圖素名稱	字串	串列	非 nil(非 0)
整數		●				
長型整數		●				
單精確度浮點數			●			
雙精度浮點數			●			
字串					●	
VLA 物件				●		
變式	●	●	●	●	●	●
安全陣列						●
vlax-true/ vlax-false	●					

### 11-3-3 變式 (Variant)

「變式」是一種資料型態，它可以包含數值、字串或日期資料。Variant 資料型態有 16 位元組的空間，可以包含的資料可達 Decimal 的範圍，或 22 位元組（加上字串長度）的字元儲存大小，同時可存放任何字元。VarType 函數定義了變體中的各種資料型態。如果一變數沒有明確宣告其他資料型態，那麼均會變成「變式」資料型態。

有關前面的那個畫圓的 VBA 範例還可以這樣寫：

```
Dim Center As Variant, Radius as Double
ThisDrawing.Utility.CreateTypedArray Center, vbDouble, 5.0,5.0,0
Radius=3.0
Set myCircle=ThisDrawing.ModelSpace.AddCircle(Center,Radius)
```

於此，我們將圓的中心點定義為一個變式 (Variant) 變數。變式變數可以代表任何類型的變數，因此它的使用也很靈活。在 AutoCAD 的開發中經常用來表示一個點或多個點。CreateTypedArray 將賦予 Center 值，使之成為一個一維並具有三個雙精度數的陣列。對應且使用變式的 Visual LISP 程式應為：

```
(1)(vl-load-com)
(2)(setq *acadObject* (vlax-get-acad-object))
(3)(setq *acadDocument* (vla-get-ActiveDocument *acadObject*))
```



```
(4)(setq *mSpace* (vla-get-ModelSpace *acadDocument*))
(5)(setq sacenter (vlax-make-safearray vlax-vbDouble '(0 . 2)))
(6)(vlax-safearray-fill sacenter '(5.0 5.0 0))
(7)(setq center (vlax-make-variant sacenter))
(8)(setq radius 3.0)
(9)(setq myCircle(vla-addCircle *mSpace* center radius))
```

或者

```
(1)(vl-load-com)
(2)(setq *acadObject* (vlax-get-acad-object))
(3)(setq *acadDocument* (vla-get-ActiveDocument *acadObject*))
(4)(setq *mSpace* (vla-get-ModelSpace *acadDocument*))
(5)(setq center (vlax-3d-point '(5.0 5.0 0)))
(6)(setq radius 3.0)
(7)(setq myCircle(vla-addCircle *mSpace* center radius))
```

請注意：上面前段的第 (5)~(7) 行與後段的第 (5) 行程式碼的作用是一樣的，都是為了得到儲存一個安全陣列的變式變數 center。

通常我們會使用以下的 4 個 Visual LISP 函數來建立和使用變式：

- 1.vlax-make-variant      建立變式
- 2.vlax-variant-type    傳回變式的資料類型
- 3.vlax-variant-value    傳回變式變數的值
- 4.vlax-variant-change-type    改變變式變數的資料類型

### 11-3-3-1    vlax-make-variant

vlax-make-variant 函數將接受兩個參數：值與類型。值參數是要賦予該變式的值。類型參數則是指定存儲到該變式中的資料類型。可供指定的類型參數以及其所代表整數值將如表 11-2 所示：

表 11-2 vlax-make-variant 函數所指定的類型參數

類型參數	意義	值
vlax-vbEmpty	未初始化（預設值）	0
vlax-vbNull	不包括有效資料	1



vlax-vbInteger	整數	2
vlax-vbLong	長型整數	3
vlax-vbSingle	單精確度浮點數	4
vlax-vbDouble	雙精度浮點數	5
vlax-vbString	字串	8
vlax-vbObject	物件	9
vlax-vbBoolean	布林類型	11
vlax-vbArray	陣列	8192+n

註：8192+n 中的 n 表示某種資料類型的安全陣列 (vlax-vbArray)。如雙精度陣列 (vlax-vbDouble) 為 8197 (8192 + 5)。

從表 11-2 中可以看出：這些類型參數都對應著一個整數值。它們的值在 AutoCAD 未來的版本可能會作修改，所以應該儘量使用類型參數名稱，而不要直接使用它所對應的整數值。例如，下述函數將用來呼叫並建立一個整數變式，並將其值設為 6：

```
_$ (setq varint (vlax-make-variant 6 vlax-vbInteger))
# <variant 2 6>
```

傳回值將表明變式的資料類型（2 表示 vbInteger）和變式值（5）。如果呼叫 vlax-make-variant 時不指定資料類型，那麼函數將使用其預設類型。例如，下述函數呼叫將用來建立一個變式，並將其值設為 6，但沒有指定資料類型：

```
_$ (setq varint (vlax-make-variant 6))
# <variant 3 6>
```

在預設的情況下，vlax-make-variant 會將指定的整數值賦予長型整數類型，而不是整數類型。當您將一個數值賦予變式時，就應該明確地說明想要的資料類型。如果不指定值和資料類型，vlax-make-variant 將建立一個未初始化的變式 (vlax-vbEmpty)。

### 11-3-3-2 vlax-variant-type

vlax-variant-type 將用來取得變式變數的類型，其用法如下：

```
(vlax-variant-type var)
```



其中，參數 var 變數為變式，傳回值即為表 11-2 中類型參數所代表的整數值。例如：以下程式會將變式設為整數，並將變式明確定義為整數型：

```
_$ (setq varint (vlax-make-variant 6 vlax-vbInteger))  
# <variant 2 6>  
_$ (vlax-variant-type varint)  
2
```

如果要將變式設為長型整數，然後再顯示變式的資料類型，則程式如下：

```
_$ (setq varint (vlax-make-variant 6))  
# <variant 3 6>  
_$ (vlax-variant-type varint)  
3
```

以下程式可將建立雙精度的安全陣列，並將安全陣列指定為變式，然後再顯示變式的資料類型：

```
_$ (setq 3dubs (vlax-make-safearray vlax-vbDouble ' (0 . 2)))  
# <safearray...>  
_$ (setq center(vlax-make-variant 3dubs))  
# <variant 8197 ...>  
_$ (vlax-variant-type center)  
8197
```

### 11-3-3-3 vlax-variant-value

vlax-variant-value 將用來取得變式變數的值，其用法如下：

```
(vlax-variant-value var)
```

其中，參數 var 變數為變式，傳回值即表 11-2 中類型參數所代表的整數值。例如：下示程式可將變式設為整數，並將變式明確定義為整數型，再取得變式的值：

```
_$ (setq varint (vlax-make-variant 6 vlax-vbInteger))
```



```
# <variant 2 6>
_$(vlax-variant-value varint)
6
```

如果欲將變式設為長型整數，然後再顯示變式的資料值，則程式為：

```
_$(setq varint (vlax-make-variant 6))
# <variant 3 6>
_$(vlax-variant-value varint)
6
```

當要建立雙精度的安全陣列，並將安全陣列指定為變式，然後再顯示變式的資料類型時，程式如下：

```
_$(setq 3dubs (vlax-make-safearray vlax-vbDouble ' (0 . 2)))
# <safearray...>
_$(setq center(vlax-make-variant 3dubs))
# <variant 8197 ...>
_$(vlax-variant-value center)
# <safearray...>
```

## 11-4 擷取及設定物件的屬性

Visual LISP 將提供 vla-get- 函數來取得物件的屬性，以及 vla-put- 函數來設定物件的屬性。為了說明這些函數的應用，請您先在 Visual LISP 的主控台提示號下執行下示語法：

```
(vl-load-com)
(setq *acadObject* (vlax-get-acad-object))
(setq *acadDocument* (vla-get-ActiveDocument *acadObject*))
(setq *mSpace* (vla-get-ModelSpace *acadDocument*))
(setq myCircle(vla-addCircle *mSpace* (vlax-3d-point (getpoint"\nPick the Center
point for a Circle:"))3.0))
```

然後在 AutoCAD 的圖形視窗中，在畫面上點取一點，就可以看到以此點為圓心，半徑為 3.0 的圓。



### 11-4-1 擷取物件的屬性

用來查看物件屬性的函數名字首為「vla-get-」，語法如下：

```
(vla-get-property object)
```

例如，vla-get-center 將傳回圓的圓心。您可以利用該函數來繪製一個與上例同心的圓：

```
(vla-addCircle *mSpace*(vla-get-center myCircle)2.0)  
#<VLA-OBJECTIAcadCircle 03ad0alc>
```

圖 11-6 就是在 AutoCAD 圖形視窗中出現的兩個圓：

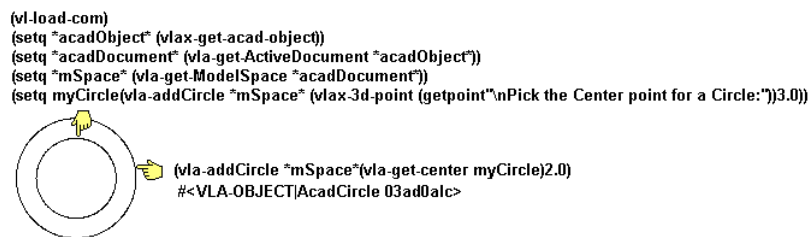


圖 11-6 利用 ActiveX 函數所產生的兩個圓

### 11-4-2 設定物件屬性

用於更改屬性的函數其函數名稱字首均為「vla-put-」，其語法為：

```
(vla-put-property object new-value)
```

例如，vla-put-center 可改變圓的圓心，以下指令可將上例中圓(myCircle)的圓心 X 座標向左偏移 1 個繪圖單位，然後再利用 vla-put-center 函數來更新該圓：

```
_$ (setq myCenter (vla-get-center myCircle))
```

將傳回 myCircle 的圓心座標。

```
_$ (setq x (- (car myCenter) 1))
```



```
_$ (setq newcenter (list x (cadr myCenter) (caddr myCenter)))
```

將傳回新的圓心座標。

```
_$ (vla-put-center myCircle newcenter) 傳回 nil
```

於 AutoCAD 的圖形視窗中將顯示圖 11-7 所示的樣子：

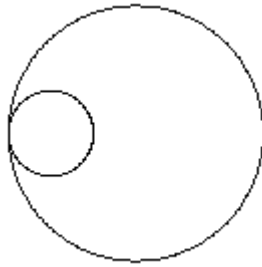


圖 11-7 利用 ActiveX 函數更改圖形物件的屬性

注意：圖形物件屬性被更改後，在 AutoCAD 的圖形視窗中，該圖形顯示並不會立即刷新，這是因為 AutoCAD 允許同時更改圖形物件的多個屬性。

若要更新顯示，則需執行下示的函數：

```
(vla-update object)
```

通常，您可以使用已預先定義的類型參數來更改物件的屬性。例如，要將圓變為紅色，您可以使用「acRed」來替代我們常用的顏色代碼：

```
$(vla-put-color myCircle acRed)
```

您可以在 VBA 的編輯視窗中，選擇「檢視(V)」下拉式功能表下的「瀏覽物件(O)」選項，以在視窗中查詢以 ac 為字首的類型參數。

### 11-4-3 判斷物件是否可以存取

在應用程式中處理目前的 AutoCAD 物件時，如果該物件也同時被其他應用程式所呼使用，那麼該物件可能就不能在該應用程式中被存取。例如，圖層被鎖定將導致改變物件屬性的動作失敗，而使程式中斷。

Visual LISP 將提供如下於使用某一物件前先判斷該物件能否被存取的函數：



- vlax-read-enable-p(判斷物件能否讀取)
- vlax-write-enable-p(判斷物件能否更改)
- vlax-erased-p(判斷物件是否已被刪除，被刪除的物件仍將儲存在圖形資料庫中)

若檢測結果為 True，這些檢測函數的傳回值將為 T，否則為 nil。下面的範例將用來說明如何檢測圓直線物件（假設 myCircle 圓物件已存在）：

```
$ (vlax-read-enable-p myCircle)    傳回  T
$ (vlax-write-enable-p myCircle)   傳回  T
$ (vlax-erased-p myCircle)         傳回  nil
```

如果刪除了 myCircle 圓物件，則：

```
$ (vlax-read-enable-p myCircle)    傳回  nil
$ (vlax-erased-p myCircle)         傳回  T
```

#### 11-4-4 使用變數來儲存 ActiveX 函數的傳回值

在 Visual LISP 中呼叫 ActiveX 方法時，有些 ActiveX 方法將要求提供變數，以使它們能對賦予變數一個值。GetBoundingBox 方法就是一例，以下的說明就是它的定義：

傳回值要求將儲存在變數中。例如，GetBoundingBox 方法就是此類函數。圖 11-8 將讓您充份的明瞭 ActiveX Automation 說明中對該函數的定義。



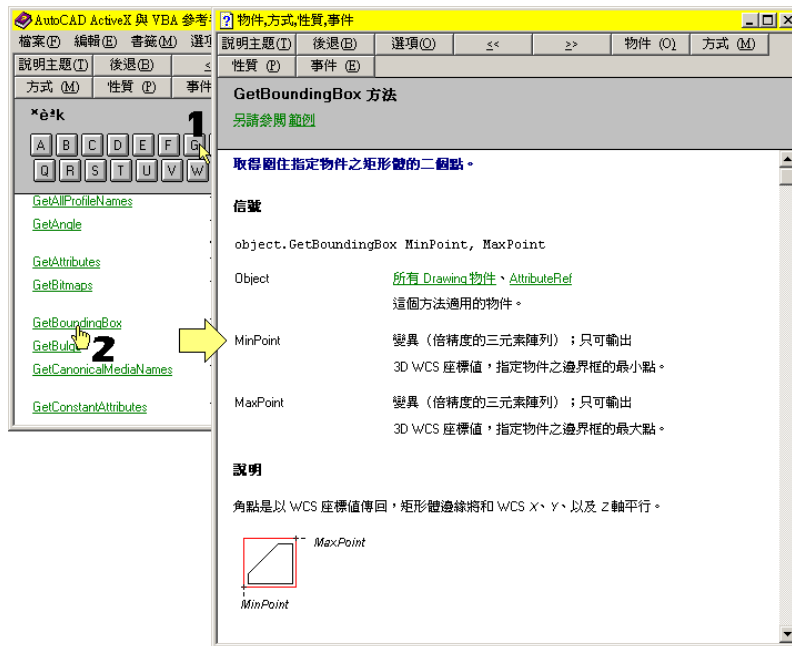


圖 11-8 查詢 GetBoundingBox 方法的操作

在該方法定義中，MinPoint 及 MaxPoint 兩個參數將用來儲存傳回值，參數的資料類型是由三個雙精實數組成的一維陣列。

從下面的範例就可以看出 Visual LISP 函數要如何來獲得圓的邊界點，並將傳回值儲存在對應的變數中：

```

_$ (vla-getboundingbox myCircle 'minpoint 'maxpoint)
nil
_$ minpoint
#<safearray...>
_$ maxpoint
#<safearray...>

```

於此，我們可以使用 vlax-safearray->list 查看其值：

```

_$ (vlax-safearray->list minpoint)
(231.127 130.823 -1.0e-008)
_$ (vlax-safearray->list maxpoint)
(237.127 136.823 1.0e-008)

```

注意：在上示函數中所套用的兩個符號參數將成為 Visual LISP 的變數，就如同用setq 函數來賦予一值給一個變數一樣。正因如此，在函數定義時，您就應該將它們宣告成函數的局部變數，以免變為整體變數。



### 11-4-5 列出物件的屬性及方法

欲列出物件的屬性，如下圖例，您可以在 Visual LISP 的主操作視窗中，選取「檢視(V)」下拉式功能表下的「檢查(I)...」選項：

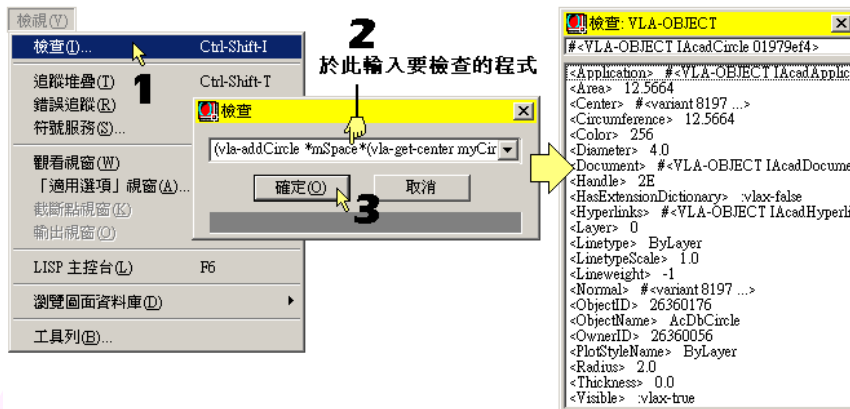


圖 11-9 檢查物件的屬性

同樣的，您也可以利用 `vla-dump-object` 函數來查看物件的屬性。在 Visual LISP 的主控制台與應用程式中執行該函數，即可列出指定物件的屬性。以下的程式碼將從模型空間中獲得物件，然後在利用 `vla-dump-object` 函數來列出物件的屬性。其標準語法如下：

`(vla-dump-object obj [T])`

其中，obj 代表物件，T 選項可有可無，不含 T 時，就只列出物件的屬性；如果包含 T 則列出物件的屬性和方法。例舉如下（假設 myCircle 已存在）：

```
_$ (vla-dump-object myCircle)
; IAcadCircle: AutoCAD Circle 介面
;特性值:
;   Application (RO) = #<VLA-OBJECT IAcadApplication 00a99b84>
;   Area = 28.2743
;   Center = (234.127 133.823 0.0)
;   Circumference = 18.8496
;   Color = 256
;   Diameter = 6.0
;   Document (RO) = #<VLA-OBJECT IAcadDocument 00eafbce>
;   Handle (RO) = "3F"
```



```
; HasExtensionDictionary (RO) = 0
; Hyperlinks (RO) = #<VLA-OBJECT IAcadHyperlinks 00f0a164>
; Layer = "0"
; Linetype = "ByLayer"
; LinetypeScale = 1.0
; Lineweight = -1
; Normal = (0.0 0.0 1.0)
; ObjectID (RO) = 1074171384
; ObjectName (RO) = "AcDbCircle"
; OwnerID (RO) = 1074171128
; PlotStyleName = "ByLayer"
; Radius = 3.0
; Thickness = 0.0
; Visible = -1
T
```

當包含 T 選項時：

```
_$ (vlax-dump-object myCircle t)
; IAcadCircle: AutoCAD Circle 介面
;特性值:
; Application (RO) = #<VLA-OBJECT IAcadApplication 00a99b84>
; Area = 28.2743
; Center = (234.127 133.823 0.0)
; Circumference = 18.8496
; Color = 256
; Diameter = 6.0
; Document (RO) = #<VLA-OBJECT IAcadDocument 00eafbcc>
; Handle (RO) = "3F"
; HasExtensionDictionary (RO) = 0
; Hyperlinks (RO) = #<VLA-OBJECT IAcadHyperlinks 00f0b8e4>
; Layer = "0"
; Linetype = "ByLayer"
; LinetypeScale = 1.0
; Lineweight = -1
; Normal = (0.0 0.0 1.0)
; ObjectID (RO) = 1074171384
; ObjectName (RO) = "AcDbCircle"
```



```
; OwnerID (RO) = 1074171128
; PlotStyleName = "ByLayer"
; Radius = 3.0
; Thickness = 0.0
; Visible = -1
;支援的方法:
; ArrayPolar (3)
; ArrayRectangular (6)
; Copy ()
; Delete ()
; GetBoundingBox (2)
; GetExtensionDictionary ()
; GetXData (3)
; Highlight (1)
; IntersectWith (2)
; Mirror (2)
; Mirror3D (3)
; Move (2)
; Offset (1)
; Rotate (2)
; Rotate3D (3)
; ScaleEntity (2)
; SetXData (2)
; TransformBy (1)
; Update ()
T
```

#### 11-4-6 判斷物件的方法或屬性是否可用

對於指定的物件，引用不屬於該物件的方法與屬性將導致錯誤。為此，當不確定某個方法或屬性能否使用時，就可以使用 `vla-method-applicable-p` 與 `vla-property-available-p` 這兩個函數來加以判斷。如果引用的方法或屬性適用於該物件，則這兩個函數的傳回值為 `T`，否則為 `nil`。判斷方法的函數語法如下：

```
(vla-method-available-p object method)
```

其中，`object` 為物件，`method` 為要判斷的方法，例如（假設 `myCircle` 已存



在)：

```
_$(vlax-method-applicable-p myCircle 'copy)    傳回  T  
_$(vlax-method-applicable-p myCircle 'addbox)    傳回  nil
```

判斷屬性的函數語法如下：

```
(vlax-property-available-p object property [T])
```

其中，object 為物件，property 為要判斷的屬性，如果在呼叫 vlax-property-available-p 函數時使用了“T”參數，那麼將改變判斷方式。如果物件擁有該屬性並且該屬性可以被更改，則函數傳返回值為“T”；如果物件沒有該屬性或該屬性不可更改，函數傳返回值為“nil”。例如（假設 myCircle 已存在）：

以下表示圓心和圖層均為 myCircle 的屬性：

```
_$(vlax-property-available-p myCircle 'center)  傳回  T  
_$(vlax-property-available-p myCircle 'layer)    傳回  T
```

以下表示 application 屬性屬於 myCircle，但不可修改：

```
_$(vlax-property-available-p myCircle 'application)  傳回  T  
_$(vlax-property-available-p myCircle 'application T) 傳回  nil
```

#### 11-4-7 使用物件集合

在 AutoCAD 物件模型中的所有物件都是用集合來分組的。例如，圖層集合就是由 AutoCAD 文字檔中的所有圖層所組成的。Visual LISP 提供了 vlax-map-collection 和 vlaxfor 函數來處理集合中的 AutoCAD 物件。

vlax-map-collection 函數可以將另一個函數應用到集合中的每一個物件。其語法為：

```
(vlax-map-collection collection-object function)
```

其中，collection-object 代表一個集合物件，function 為要應用的函數或 lambda 運算式，函數傳返回值即為集合物件。例如，下示程式碼會將模型空間所有圖形顏色改為綠色（\*mSpace\* 表示模型空間物件）：



```
_$(vlax-map-Collection *mSpace* '(lambda (x) (vla-put-color x acGreen)))  
#<VLA-OBJECT IAcadModelSpace 00f0eeb4>
```

如果要對集合中的每一個物件使用一系列函數來求值，您可使用 `vlax-for`：

```
(vlax-for symbol collection [expressions]...)
```

與 `foreach` 函數用法類似，`vlax-for` 將傳回 `for` 迴圈中最後一個運算式求值的結果。下示範例將定義了一個函數，它用 `vlax-for` 將模型空間中所有物件的顏色改為綠色（`*mSpace*` 表示模型空間物件）：

```
(vlax-for obj *mSpace* (vla-put-color obj 3))
```

注意：如果在遍歷操作某集合時修改該集合（添加或刪除成員），可能會引起錯誤。

#### 11-4-8 取出物件集中的物件

物件集合由物件組成，有時需要對其中某個或一些物件進行一些操作時，就要自物件集合中的取出所需的物件。`Item` 方法可用來在物件中尋找相關的成員，而 `Count` 屬性可用於統計物件集中物件的數目。利用 `Item` 方法及 `Count` 屬性，就可以對物件集中的物件進行單獨的操作。例如，我們可以尋找模型空間中所有的物件，並判斷這些物件的類型是什麼，然後對需要處理的物件進行單獨的處理。

以下的程式段將讓模型空間裡所有圓物件顏色變為綠色：

```
(setq index 0)  
(repeat (vla-get-count *mSpace*)  
  (if (= "AcDbCircle"  
    (vla-get-objectname  
      (vla-item *mSpace*  
        index  
      )  
    )  
    )  
    (vla-put-color (vla-item *mSpace* index) acGreen)
```



```
)  
(setq index (+ index 1))  
)
```

注意：Item 方法和 Count 屬性對群組與選擇組也同樣適用。

#### 11-4-9 將物件從記憶體中釋放

AutoCAD 中可以將多個變數指向同一個 AutoCAD 圖素，也可以將多個 VLA 物件指向同一個圖形物件。一般只要用 equal 函數就可以比較兩個 VLA 物件，如果兩個物件要指向同一個圖形物件，equal 函數將傳回 T。

只要 VLA 物件還指向圖形物件，AutoCAD 就會保留該物件所需的記憶體。如果不再需要引用該物件，就可使用函數 vlax-release-object 來通知 AutoCAD：

```
(vlax-release-object object)
```

其中，object 為某個要取消引用的 VLA 物件。例如（\*mSpace\* 為模型空間 VLA 物件）：

```
_$ (vlax-release-object *mSpace*) 傳回 3
```

與關閉文件類似，刪除某物件後，就不能再使用該 VLA 物件指標了，但是呼叫 vlax-release-object 函數時並不會釋放記憶體。不過，如果釋放了物件的所有引用，AutoCAD 在必要時就會釋放相關記憶體。如果要測試是否釋放了某物件，就可以使用函數 vlax-object-released-p：

```
(vlax-object-released-p object)
```

例如，下列程式碼將用來測試了執行 vlax-release-object 前後的 \*mSpace\* 物件（\*mSpace\* 為模型空間 VLA 物件）：

```
_$ (vlax-object-released-p *mSpace*) 傳回 nil  
_$ (vlax-release-object *mSpace*) 傳回 3  
_$ (vlax-object-released-p *mSpace*) 傳回 T
```

釋放了該物件，函數將傳回 T，否則傳回 nil。



### 11-4-10 物件資料轉換

Visual LISP 還提供了許多引用 AutoCAD 圖形物件的方法，包括：

- 由 ActiveX 函數傳回的 VLA 物件。
- 由 entget 和 entsel 傳回的圖素名稱 (ename)。此圖素名稱將指向某開啟圖形中的物件。
- 由 handent 傳回的控制碼。控制碼的圖素在不同 AutoCAD 任務期間將保持不變。
- 物件 ID 碼。ARX 將用它來標識物件。

爲了靈活地使用這些物件，Visual LISP 提供了將一種物件標誌符號轉換爲另一種標誌符號的函數。我們就以下述小節來分述之。

#### 11-4-10-1 在圖素名稱和 VLA 物件之間做轉換

要將 entget 等函數擷取到的圖素名稱轉換成物件的函數，您可以使用 ActiveX 函數，以及將 VLA 物件轉換成圖素名稱的函數。欲將圖素名稱轉換成 VLA 物件，就要用到 vlax-ename->vla-object 函數，例如：

```
_$ (setq ename-circle (car (entsel "\nPick a Circle:")))
<圖素名: 40068e00>
_$ (setq vlaobj-circle(vlax-ename->vla-object ename-circle))
#<VLA-OBJECT IAcadCircle 020027b4>
```

同樣的，將 VLA 物件轉換成圖素名稱則要利用 vlax-vla-object->ename 函數：

```
(setq new-ename-circle(vlax-vla-object->ename vlaobj-circle))
<Entity name: 1bed9c8>
```

#### 11-4-10-2 物件間的轉換

對相同圖形物件的引用可以有多種資料類型：控制碼字串、圖素名稱、VLA



物件或 ARX 物件 ID 碼，這些類型之間的相互轉換函數如下：

- 利用 DXF 碼 5 就可以在圖素資料表中經由圖素得到圖素控制碼而檢  
索出來

```
_$ (setq handle-circle(cdr (assoc 5 (entget ename-circle))))  
"40"
```

- 由圖素控制碼得到圖素名稱，可以利用 handent 函數

```
_$ (handent handle-circle)  
<圖素名: 40068e00>
```

- 由圖素控制碼得到 VLA 物件，可使用 vla-handleToObject 函數  
（\*acadDocument\* 表示目前 AutoCAD 文檔物件）：

```
_$ (setq vla-circle(vla-HandleToObject *acadDocument* handle-circle))  
#<VLA-OBJECT IAcadCircle 00f0bdb4>
```

- 由 VLA 物件得到圖素控制碼，可使用 vla-get-handle 函數

```
_$ (vla-get-Handle vla-circle)  
"40"
```

- 可以利用 ObjectID 屬性由 VLA 物件來得到 ARX 的 ObjectID

```
_$ (setq objid-circle(vla-get-ObjectID vla-circle))  
1074171392
```

- 爲了將 ARX 的 ObjectID 轉換成 VLA 物件，需要利用 ObjectIDtoObject  
方法來處理 AutoCAD 文檔物件

```
_$ (vla-ObjectIDtoObject *acadDocument* objid-circle)  
#<VLA-OBJECT IAcadCircle 00f0bdb4>
```

注意：雖然使用 Visual LISP 的 ActiveX 程式執行速度更快，而且程式更直接，但有時開發週期會較長。例如，使用 ActiveX 物件來執行 AutoCAD 的「TTR」畫圓指令，就要寫出十幾行的程式碼來計算圓心位置，而一般利



用 AutoCAD 的指令行語法就可以很快畫出。因此，在 AutoCAD 圖素名稱與 VLA 物件（Visual LISP ActiveX 物件）之間進行轉換，使得 AutoCAD 中的一些高效率函數得以充分的利用，就可以縮短開發時間，提高開發效率。

## 啓發性習題

### 一.選擇題(單複選混合)

- 1.(    ) 以下何者是 VBA 中設定物件屬性的語法？
  - (a) radius=circle.radius
  - (b) circle.radius=10
  - (c) circle.delete
  - (d) 以上皆非
- 2.(    ) 以下何者是 VBA 中擷取物件屬性的語法？
  - (a) radius=circle.radius
  - (b) circle.radius=10
  - (c) circle.delete
  - (d) 以上皆非
- 3.(    ) 以下何者是 VBA 中指定具有某方法就執行某操作的語法？
  - (a) radius=circle.radius
  - (b) circle.radius=10
  - (c) circle.delete
  - (d) 以上皆非
- 4.(    ) 在 Visual LISP 裡，提供一系列函數來處理 ActiveX 物件的函數名稱都是以何字首開頭的？
  - (a) vla-
  - (b) vlax-
  - (c) ac-
  - (d) 以上皆非



5.( ) vla- 與 vlax- 函數的差別在於：

- (a) 差不多，只是 vlax- 屬延伸型的 ActiveX 函數
- (b) 完全不同，vla- 被用於物件的建立上
- (c) vlax- 函數是一些更綜合的 ActiveX 函數，它們可被應用到許多方法、物件或特性
- (d) 以上皆非

6.( ) 所謂的「安全陣列」就是：

- (a) 可在 Visual LISP 中向 vla- 類函數傳遞值
- (b) ActiveX 的一種資料類型
- (c) 可以不為病毒所破壞的保護函數
- (d) 以上皆非

## 二.實作問答題

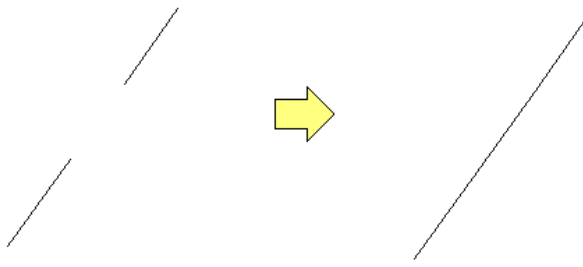
1. 有很多從事水電空調業或配線工程的學生對 AutoCAD 的繪圖精密度與效益均已抱持肯定的態度。但是，在他們的這個專業中，計算線材估料已是不可或缺的工作之一。儘管 AutoCAD 可繪出正確的圖形，但卻沒有指令可讓他們用來計算線的長度（尤其是不規則的線長度）。您可以設計一個可計算圖面中所有使用 PLINE 指令所繪出的曲線段總長度的程式嗎？（以 VLISP/VBA 撰寫）。

設計詢問句：無（先畫出一些聚合線再直接執行即可）

解答檔案名稱：V-P\_LEN.LSP ， P\_LEN.DVB

2. 我們經常在畫圖時，會因為需要而將線切段，可是又會因為編輯變更的原因，又希望將在同一線上的線段連起來，以減少存檔容量與方便後續的編輯。請寫出一個可以將二條斷線變成一條線的程式（以 VLISP 撰寫）。





設計詢問句：

1. 請選擇第一條線:
2. 請選擇第二條線:

解答檔案名稱：2IN1.LSP

