AutoCAD 2000 丛书

AutoCAD 2000

Visual LISP 自学教程

刘志刚 叶以农 孙爱充 曹敬波／编著

中国电力出版社
www.infopower.com.cn

AutoCAD 2000 从书

AutoCAD 2000

Visual LISP 自学教程

刘志刚 叶以农 孙爱充 董敬波 / 编著

中国电力出版社

内 容 提 要

本书通过开发 AutoCAD 的新命令绘制花园小径并以圆形花砖填满这个花园小径，介绍了 Visual LISP 的强大功能。全书共分七章，第一章提出了程序设计的目标，第二章至第五章则介绍了如何实现这个目标，第六章至第七章则分别介绍了反应器和程序的集成。全书语言流畅，通俗易懂。

本书适合 AutoCAD 设计人员及 LISP 使用者阅读。

图书在版编目（CIP）数据

AutoCAD 2000 Visual LISP 自学教程/叶以农 编著.-北京：中国电力出版社，2001.9

ISBN 7-5083-0794-1

Ⅰ.A… Ⅱ.叶… Ⅲ.①计算机辅助设计-应用软件，AutoCAD 2000 ②LISP 表处理语言-程序设计 Ⅳ.TP391.72

中国版本图书馆 CIP 数据核字（2001）第 066533 号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

实验小学印刷厂印刷

各地新华书店经售

*

2001 年 10 月第一版 2001 年 10 月北京第一次印刷

787 毫米×1092 毫米 16 开本 8 印张 173 千字

定价 15.00 元

序　　言

本书将介绍 Visual LISP™对 AutoCAD®的程序设计环境所具有的强大功能，并介绍 AutoLISP®语言新的特色。

本书的目的是使用自动绘图工具来绘制一个花园小径，减少绘图时间。你将学习到如何建立图形程序，此图形程序会自动生成复杂的图形而不用每次都重新绘制。

读者对象

本书适合有经验的 AutoCAD 设计人员及已熟悉 LISP 或 AutoLISP 的使用者阅读。本书也假设你了解基本的 Windows®文件管理操作，如建立目录、复制文件及浏览硬盘或网络上的文件系统等。

本书简介

本书的目的是开发 AutoCAD 的新命令，来绘制花园小径并以圆形花砖填满这个花园小径。本书分为七章。从前一章进入到下一章时，对于如何完成每个操作，你所收到的指导细节将越来越少。如果有任何问题，可查阅 VLISP 文件中的帮助。

第 4 章和第 5 章属于中级程度，其知识范围会超过 AutoLISP 的基本概念。第 6 章和第 7 章属于高级程度并包含相当复杂的程序设计操作，是为有经验的 AutoLISP 开发人员准备的。

每一个开发阶段所有绘制花园小径的源代码都在 AutoCAD 安装 CD 上，只有选择“完全”安装，或选择“自定义”安装并选取“例子”项目时，这些教程文件才会包含在安装结果中。如果之前安装 AutoCAD 时没有安装例子，请回到安装步骤，选择“自定义”安装，并且只选取“例子”项目。

源代码文件的目录结构是依照本书的计划而定的，如：

```
<AutoCAD directory>\Tutorial\VisualLISP\Lesson1
<AutoCAD directory>\Tutorial\VisualLISP\Lesson2
```

等等。

建议不要修改 AutoCAD 提供的例程源代码文件。如果程序运作得不太正确，你可能想复制提供的源代码到工作目录中。本书的工作目录均以下列方式提及：

```
<AutoCAD directory>\Tutorial\VisualLISP\MyPath
```

如果选择不同的路径作为工作目录，请在适当的时候替换目录名称。

最后，请阅读我社出版的《Visual LISP 开发人员指南》一书。那里有需要我们了解的许多概念。

目　　录

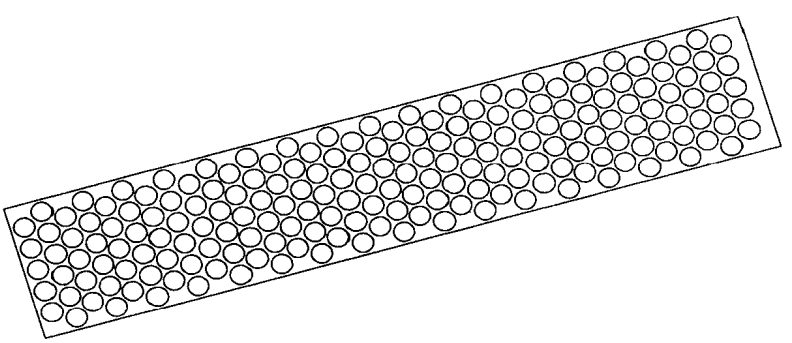
第 1 章

设计及编写程序

从本章起，我们将开始学习应用程序的各种功能。使用 Visual LISP（VLISP）DC，可以建立一个 LISP 程序，然后编写 AutoLISP 程序代码。在此过程中，我们将了解如何利用 VLISP 更加便捷地开发应用程序。

确定程序目标

开发 AutoLISP 程序是想使 AutoCAD 在某方面提高自动化。这就需要加速重复绘制草图的功能，或是简化复杂的操作。在本书中，我们希望所绘制的花园小径，是一个基于使用者的原始输入，到具有不同数目图元的复合造型。这就是它的样子：



利用程序绘制花园小径时必须做到以下几点：

- 给定一个起点、一个终点及宽度，绘制一个直线的边界。这个边界可以是任意方向的，而且不限制它的大小。

- 向使用者提示花砖大小及花砖间距的值。花砖是简单的圆形且会布满边界内的区域，但必须不会互相重叠或越过边界。
- 以交替列的方式来放置花砖。

可以执行 AutoCAD 提供的完整版本的应用程序，看程序是如何运行的。

执行提供的例子

步骤 1 在“AutoCAD 工具”菜单中选择“加载应用程序”。
步骤 2 选取 gardenpath.vlx，此文件位于 Tutorial\VisualLISP 目录下，并选取“加载”。
步骤 3 单击“关闭”。
步骤 4 在 Command 提示中输入 gpath。
步骤 5 在 AutoCAD 图形窗口中，选一个起点和终点来响应前两个提示。
步骤 6 在“Half Width of Path”提示下输入 2。
步骤 7 当“Garden Path Tile Specifications”对话框显示时，选择“OK”。

看到了应用程序是如何运行的，我们就可以开始使用 VLISP 来开发了。首先，我们以实例说明当 VLISP 等待从 AutoCAD 返回控制权时，会发生什么事情。你可能已经遇到过这种情况。

Visual LISP 等待从 AutoCAD 返回控制权

步骤 1 在 AutoCAD“命令”提示下输入 vlisp 来激活 Visual LISP。
步骤 2 切换回 AutoCAD 窗口（从工具栏选取 AutoCAD 或按 ALT+TAB 键后选择 AutoCAD 皆可），然后在 AutoCAD“命令”提示下输入 gpath。
步骤 3 在响应 gpath 的提示之前，先切换回 VLISP 窗口。

在 VLISP 窗口，鼠标指针变成 VLISP 符号，我们不能在 VLISP 窗口中选择任何命令或在任何地方输入文字。指针符号是暗示以 VLISP 继续执行工作之前，在 AutoCAD 中有一个活动需要用户完成。每当看到 VLISP 指针时，请不要忘记。

步骤 4 返回 AutoCAD 窗口，响应所有 gpath 的提示。

用户现在已经开始构建花园小径应用程序了。

用 Visual LISP 开始开发应用程序

步骤 1 在 VLISP 的“文件”菜单中选择“新建文件”。
步骤 2 在文字编辑器窗口（即以<无标题-0>为标题的窗口）输入下列程序代码；也可以省略注释：

```
;;; Function C:GPath is the main program function and defines the
;;; AutoCAD GPATH command.
(defun C:GPath ()
  ;; Ask the user for input: first for path location and
  ;; direction, then for path parameters. Continue only if you have
  ;; valid input.
  (if (gp:getPointInput) ;
    (if (gp:getDialogInput)
      (progn
        ;; At this point, you have valid input from the user.
        ;; Draw the outline, storing the resulting polyline
        ;; "pointer" in the variable called PolylineName.
        (setq PolylineName (gp:drawOutline))
        (princ "\nThe gp:drawOutline function returned <")
        (princ PolylineName)
        (princ ">")
        (Alert "Congratulations - your program is complete!")
      )
      (princ "\nFunction cancelled.")
    )
    (princ "\nIncomplete information to draw a boundary.")
  )
  (princ)  ; exit quietly
)
;;; Display a message to let the user know the command name.
(princ "\nType gpath to draw a garden path.")
(princ)
```

步骤 3　在菜单中选择“文件”下的“另存为”，将新文件中的程序代码保存为<AutoCAD directory>\Tutorial\VisualLISP\ MyPath\gpmain.lsp。

步骤 4　检查完成的工作。

Visual LISP 程序代码格式化

VLISP 可分辨出 AutoLISP 程序文件并以不同的颜色强调字符的各种类型及单字。这就

允许我们快速地查看到错误。例如，如果在文字字符串之后漏了右引号，所有继续键入的文字都会以紫红色显示，这是表示字符串的颜色。当输入右引号时，VLISP 会根据它代表的语言元素，以正确的颜色来表示字符串之后的文字。

输入文字时，VLISP 也会加入间距及缩排来编排格式。要想用 VLISP 编排从其他文字编辑器复制来的程序代码，请在 VLISP 菜单中选择“工具”下的“设置编辑窗口格式”。

defun 函数定义新的函数。请注意主函数命名为 C:GPath。前置 C: 建立的这个函数是可从 AutoCAD“命令”行调用的。GPath 是使用者在 AutoCAD“命令”提示下输入的要调用的应用程序的名称。绘制花园小径轮廓的函数是 gp:drawOutline。这些名称用前置 gp: 来表示它们是花园小径应用程序特有的名称。这种前置不是必要的，但这是一个好的命名习惯，可以用来区分应用程序特定的函数和一般常用的公用函数。

在主函数中，如果程序执行成功，princ 表达式会显示程序的结果，如果程序遇到无法预期的事件，则会出现警告信息。例如，像在第 2 章中会看到的，如果使用者按下 ENTER 键而不是点击屏幕上的点，gp:getPointInput 调用会提前结束，将 nil 传给主函数。这会造成程序发出下列的 princ 信息“绘制边界的信息不完整。”

接近程序末端的 princ 调用是作提示用的。根据应用程序的加载，提示会通知使用者需要键入的内容来开始花园小径的绘制。最终没有字符串参数的 princ 会强迫程序结束，表示未传回主函数最终表达式的值。

填满程序中的间隙

要让这个新文件中的程序代码正确地运行，我们必须编写三个以上的函数。主要的花园小径程序代码包含调用三个定制的函数：

- gp:getPointInput。
- gp:getUserInput。
- gp:drawOutline。

现在，我们要编写 stubbed-out 函数。stubbed-out 函数是为了要完成后续工作，而做位置保留用的。可以在加入所有完成应用程序的细节之前，用它来试验我们的程序代码片段。

定义应用程序的 stubbed-out 函数

步骤 1 将光标放置在文字编辑器窗口程序代码的顶端，按几下 ENTER 键加入空白行。

步骤 2 在插入空白行的地方，输入下列程序代码：

```
;;; Function gp:getPointInput will get path location and size
```

```
(defun gp:getPointInput ()
  (alert
    "Function gp:getPointInput will get user drawing input"
  )
  ;; For now, return T, as if the function worked correctly.
  T
)
;;; Function gp:getDialogInput will get path parameters
(defun gp:getDialogInput ()
  (alert
    "Function gp:getDialogInput will get user choices via a dialog"
  )
  ;;For now, return T, as if the function worked correctly.
  T
)
;;; Function gp:drawOutline will draw the path boundary
(defun gp:drawOutline ()
  (alert
    (strcat "This function will draw the outline of the polyline"
    "\nand return a polyline entity name/pointer."
    )
  )
  ;; For now, simply return a quoted symbol. Eventually, this
  ;; function will return an entity name or pointer.
  'SomeEname
)
```

在每一个输入函数结束之前都有一行程序代码，这一行代码只有一个 T。它用来给调用它的函数传回值。所有 AutoLISP 函数都会传回一个值给调用它们的函数。T 这个字母在 AutoLISP 中代表“true”，加入后会使函数传回一个真值。构建 gpmain.lsp 的方法是它调用的每一个输入函数必须传回一个不是 nil 的值（nil 代表“空值”），让程序进入下一个步骤。

AutoLISP 函数会依照缺省值，传回最后一个表达式计算的值。在 stubbed-out 函数之中，唯一的表达式是调用 alert 的函数。但是 alert 一定会传回 nil。如果这是 gp:getPointInput 的最后一个表达式，它一定会传回 nil，且永远不会通过 if 到达 gp:getDialogInput 函数。

同理，gp:DrawOutline 函数的结尾会传回一个有引号的符号（'SomeEname）做位置保

留用。有引号的符号是 LISP 不计算的结构。

让 Visual LISP 检查程序代码

VLISP 有一个很强的功能可以检查程序代码的语法错误。在执行程序之前使用这个工具，可以帮助我们发现一般的打字错误，如遗漏了括号或引号及其他的语法问题等。

检查程序代码的语法

步骤 1 请确定包含 gpmain.lsp 的文字编辑器窗口是当前窗口（按一下窗口的标题栏，使之成为当前窗口）。

步骤 2 在 VLISP 菜单中，选择“工具”下的“检查编辑器中的文字”。

步骤 3 “编译输出”窗口将出现语法检查的结果。如果 VLISP 没有检查到任何错误，窗口会有和下列相似的文字：

```
[检查文字 PMAIN.LSP 正在加载...]
......
; 检查完成
```

如果有问题需要帮助，请参考《Visual LISP 开发人员指南》，看是否可以确定问题发生在何处。如果寻找出了问题之所在，请使用 gpmain.lsp 例程文件来继续学习，此文件位于 lesson1 目录下。

使用提供的 gpmain.lsp 程序

步骤 1 关闭含有输入 gpmain.lsp 程序代码的文字编辑器窗口。

步骤 2 选择 VLISP 菜单“文件”下的“打开文件”，打开 gpmain.lsp 文件，此文件位于\Tutorial\VisualLISP\lesson1 目录下。

步骤 3 选择“文件”下的“另存为”，将文件保存在\Tutorial\VisualLISP\MyPath 目录下，文件名称为 gpmain.lsp，以替换建立的备份。

用 Visual LISP 执行程序

在 VLISP 中执行 AutoLISP 程序，允许我们使用 VLISP 的很多调试功能，检查可能发

生的问题。

加载并执行程序

步骤 1　将文字编辑器窗口置为当前窗口，在 VLISP 菜单中选择“工具”下的“加载编辑器中的文字”。

步骤 2　在 VLISP“控制台”窗口的_$提示下输入（C:GPath）。

步骤 3　“控制台”窗口需要输入 AutoLISP 语法的命令，所以全部的函数名称必须包含在括号内。

步骤 4　按一下 ENTER 键或单击“确定”来响应信息窗口。最后系统显示信息“Congratulations – your program is complete!”

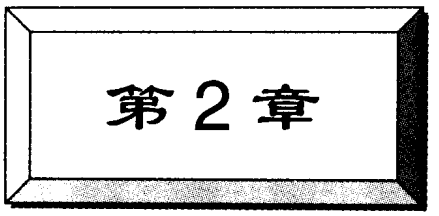
请注意如果执行 gpath 时将 AutoCAD 最小化，我们将无法看到提示，除非将 AutoCAD 窗口还原（使用工具栏或按 ALT+TAB 键）。

重点回顾

在这一章中，我们已经：

- 定义了程序的目标。
- 学习到 stub 函数的作用。
- 学习到有关函数的命名，以区别它是我们的应用程序特有的函数，还是重复使用的一般函数。
- 学习到如何使用 VLISP 来检查我们的程序代码。
- 学习到如何在 VLISP 中加载并执行程序。

现在已经学完了这一章。再一次保存程序文件，确定它是最后修订的结果。



使用 Visual LISP 调试工具

本章将介绍如何使用一些有用的 VLISP 调试工具，这些工具可以加速 AutoLISP 程序的开发。我们还会学习到局部变量和全局变量的差异及何时使用何种变量。我们的程序会变得更活泼——系统将提示我们输入某些信息。这些信息会存放在表中，我们会了解到在 AutoLISP 程序中使用表的作用。毕竟，LISP 是一种“表处理”（LIST Processing）语言。

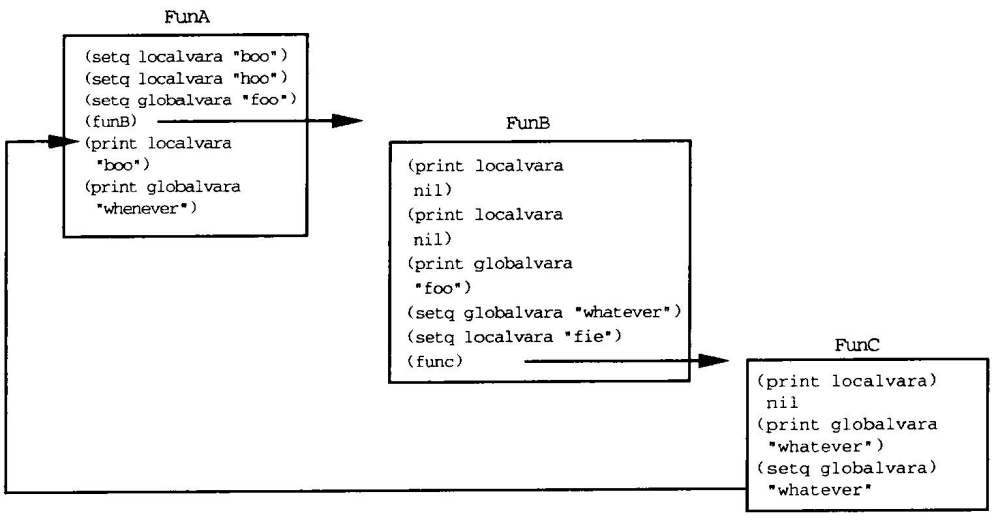
局部变量和全局变量之间的差异

局部变量是在某个程序中使用的变量。全局变量是可以让所有加载文件（AutoCAD 图形）的函数存取的变量，在程序将它们定义完成后这些变量会保留它们的值。我们稍后会看到这样的例子。

只有当定义局部变量的函数正在执行时，局部变量才会保留它的值。在函数执行完成之后，局部变量的值会自动放弃，系统会收回变量使用的内存空间。这和自动无用存储单元收集（garbage collection）相同，而且这也是大部分 LISP 开发环境（如 VLISP）的特征。局部变量比全局变量能更有效地使用内存。

局部变量的另一个优点是使调试和维护应用程序变得更加容易。有了全局变量之后，我们永远不需要确定何时或在使用哪个函数时要修改变量的值；有了局部变量，我们不用再这样追踪。我们通常会以较少的副作用（也就是程序的一部分影响程序另一部分的变量）结束。

因为局部变量有这些优点，所以本书几乎仅使用局部变量。



注意

如果已经使用过 AutoLISP 一段时间，你可能已建立了在开发时使用全局变量的练习，并在构建程序时检查它。现在不再需要这个练习，因为 VLISP 已给出了强而有力的调试工具。

在程序中使用局部变量

参考在第 1 章中建立的 **gp:getPointInput** 函数：

```
(defun gp:getPointInput ()
  (alert
    "Function gp:getPointInput will get user drawing input"
  )
  ;; For now, return T, as if the function worked correctly.
  T
)
```

到目前为止，这个函数没有太多的功能。我们现在可以加入函数，在它的基础上继续进行完善，以获得输入，我们由这些定义起点、终点及路径的宽度。

用建立 AutoLISP 程序来模仿 AutoCAD 的操作过程是一个很好的做法。所以，不要以

中心线端点为参照选取一个点来指出宽度，我们的程序应要求选取一半宽度。

一旦 **gp:getPointInput** 函数完成后，这个变量和指定给它的值一样，就不再存在了。因此，程序会将我们提供的值存放在局部变量中。以下就是这个函数的代码：

```
(defun gp:getPointInput (/ StartPt EndPt HalfWidth)
  (if (setq StartPt (getpoint "\nStart point of path: "))
    (if (setq EndPt (getpoint StartPt "\nEndpoint of path: "))
      (if (setq HalfWidth (getdist EndPt "\nHalf width of path: "))
        T
      )
    )
  )
)
```

这个局部变量是在斜线字符（**defun** 表达式中）之后声明的。第一个调用 **getpoint** 提示指出起点，在选择起点之后才会得到终点。当选取终点时，我们会发现从起点处延伸出了一条拖引线。同样地，当设定宽度一半的值时，我们会看见另一条拖引线，这次是代表距离，从终点延伸出来。

gp:getPointInput 的执行

步骤 1 输入 **gp:getPointInput** 程序代码到 VLISP“控制台”窗口中。

步骤 2 在“控制台”窗口中，当光标停在最后一个程序代码块（或在它之下的下一行）括号之后按下 ENTER 键，将替换任何以前加载的 **gp:getPointInput** 函数版本。

步骤 3 在“控制台”窗口的“控制台”提示处输入（**gp:getPointInput**）来执行这个函数。

步骤 4 提示选择一个点，并输入宽度一半的值。

检查 gp:getPointInput 函数

执行 **gp:getPointInput** 函数时，控制权会自动从 VLISP 传到 AutoCAD。用户响应三个提示后，哪一个控制权从 AutoCAD 传回 VLISP，相应的 T 符号会显示在“控制台”窗口中。