Visual LISP

P. K. Jain

Learning Objectives

After completing this chapter, you will be able to:
- *Start Visual LISP in AutoCAD.*
- *Use the Visual LISP Text Editor.*
- *Load and run Visual LISP programs.*
- *Load existing AutoLISP files.*
- *Use the Visual LISP Console.*
- *Use the Visual LISP formatter and change formatting options.*
- *Debug a Visual LISP program and trace the variables.*

Visual LISP

Visual LISP is another step forward to extend the customizing power of AutoCAD. Since AutoCAD Release 2.0 in the mid 1980s, AutoCAD users have used AutoLISP to write programs for their applications; architecture, mechanical, electrical, air conditioning, civil, sheet metal, and hundreds of other applications. AutoLISP has some limitations. For example, when you use a text editor to write a program, it is difficult to locate and check parentheses, AutoLISP functions, and variables. Debugging is equally problematic, because it is hard to find out what the program is doing and what is causing an error in the program. Generally, programmers will add some statements in the program to check the values of variables at different stages of the program. Once the program is successful, the statements are removed or changed into comments. Balancing parentheses and formatting the code are some other problems with traditional AutoLISP programming.

Visual LISP has been designed to make programming easier and more efficient. It has a powerful text editor and a formatter. The text editor allows color coding of parentheses, function names, variables and other components of the program. The formatter formats the code in a easily readable format. It also has a watch facility that allows you to watch the values of variables and expressions. Visual LISP has an interactive and intelligent console that has made programming easier. It also provides context-sensitive help for AutoLISP functions and apropos features for symbol name search. The debugging tools have made it easier to debug the program and check the source code. These and other features have made Visual LISP the preferred tool for writing LISP programs. Visual LISP works with AutoCAD and it contains its own set of windows. AutoCAD must be running when you are working with Visual LISP.

Overview of Visual LISP

In this overview you will learn how to start Visual LISP, use the Visual LISP Text Editor to write a LISP Program, and then how to load and run the programs. As you go through the following steps, you will notice that Visual LISP has some new capabilities and features that are not available in AutoLISP.

Pull-down:	Tools > AutoLISP > Visual LISP Editor
Command:	VLIDE

Starting Visual LISP

1. Start AutoCAD.

2. In the **Tools** menu select **AutoLISP** and then select **Visual LISP Editor**. You can also start Visual LISP by entering the **VLIDE** command at the Command prompt. AutoCAD displays the Visual LISP window, Figure 35-1. If it is the first time in the current drawing session that you are loading Visual LISP, the Trace window might appear, displaying information about the current release of Visual LISP and errors that might be encountered when loading Visual LISP. The Visual LISP window has four areas: Menu, Toolbars, Console window, and Status bar, Figure 35-1. The following is a short description of these areas:

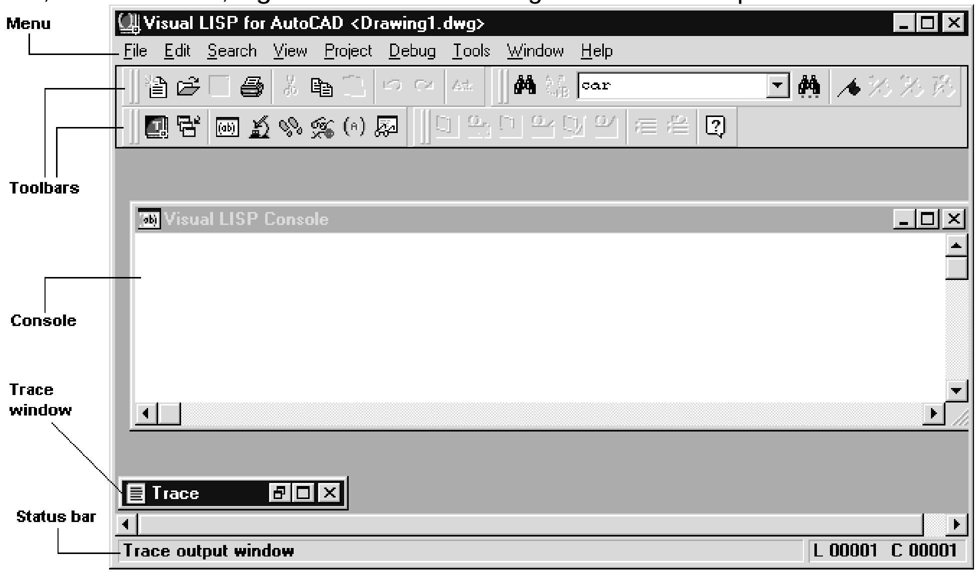


Figure 35-1 *Visual LISP window*

PKJain

Menu. The menu bar is at the top of the Visual LISP window and lists different menu items. If you select a menu, AutoCAD displays the items in that menu. Also, the function of the menu item is displayed in the status bar that is located at the bottom of the Visual LISP window.

Toolbars. The toolbars provide a quick and easy way to invoke Visual LISP commands. Visual LISP has 5 toolbars; Debug, Edit, Find, Inspect, and Run. Depending on the window that is active, the display of the toolbars changes. If you move the mouse pointer over a tool and position it there for a few seconds, a tool tip is displayed that indicates the function of that tool. A more detailed description of the tool appears in the status bar.

Console window. The Visual LISP Console window is contained within the Visual LISP window. It can be scrolled and positioned anywhere in the Visual LISP window. The Console window can be used to enter AutoLISP or Visual LISP commands. For example, if you want to add two numbers, enter (+ 2 9.5) next to the $ sign and then press the **Enter** key. Visual LISP will return the result and display it in the same window.

Status bar. When you select a menu item or a tool in the toolbar, Visual LISP displays the related information in the status bar that is located near the bottom of the Visual LISP window.

Using the Visual LISP Text Editor

1. Select **New File** from the **File** menu. The Visual LISP Text Editor window is displayed, Figure 35-2. The default file name is Untitled-0 as displayed at the top of the editor window.

2. To activate the editor, click anywhere in the Visual LISP Text Editor.

3. Type the following program and notice the difference between the Visual LISP Text Editor and the text editors you have been using to write AutoLISP programs (like Notepad).

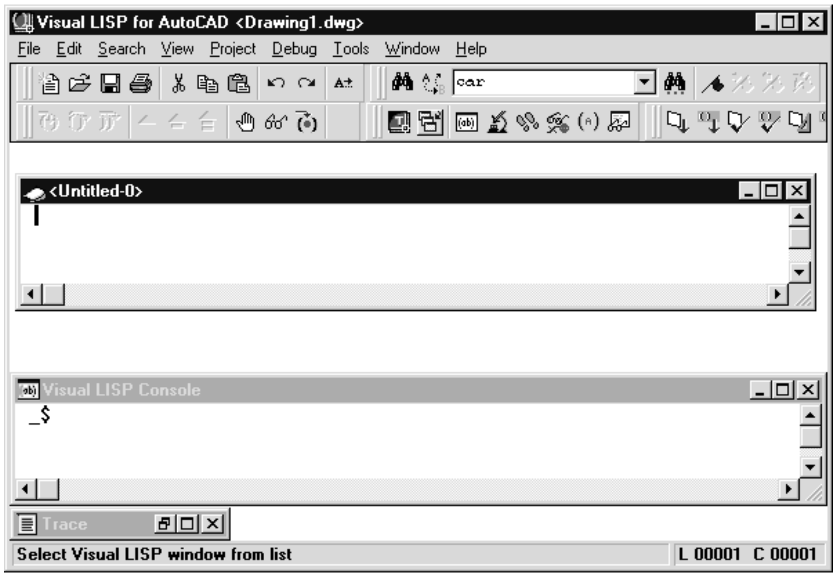
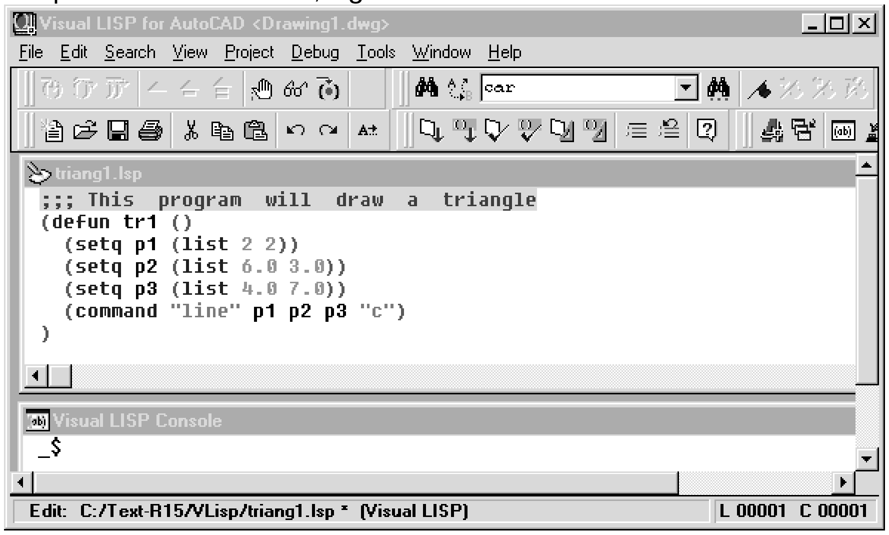

Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
car
<Untitled-0>
Visual LISP Console
_$
Trace
Select Visual LISP window from list
L 00001 C 00001

Figure 35-2 *Visual LISP Text Editor*

PKJain

```
;;;This program will draw a triangle
(defun tr1 ()
(setq p1(list 2 2))
(setq p2(list 6.0 3.0))
(setq p3(list 4.0 7.0))
(command "line" p1 p2 p3 "c")
)
```

5. Choose the **File** menu and select the **Save** or **Save As** option. In the **Save As** dialog box enter the name of the file, **Triang1.lsp**. After you save the file, the file name is displayed at the top of the editor window, Figure 35-3.


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
car
triang1.lsp
Visual LISP Console
Edit: C:/Text-R15/VLisp/triang1.lsp * (Visual LISP)
L 00001 C 00001

```
;;; This program will draw a triangle
(defun tr1 ()
  (setq p1 (list 2 2))
  (setq p2 (list 6.0 3.0))
  (setq p3 (list 4.0 7.0))
  (command "line" p1 p2 p3 "c")
)
```

```
_$
```

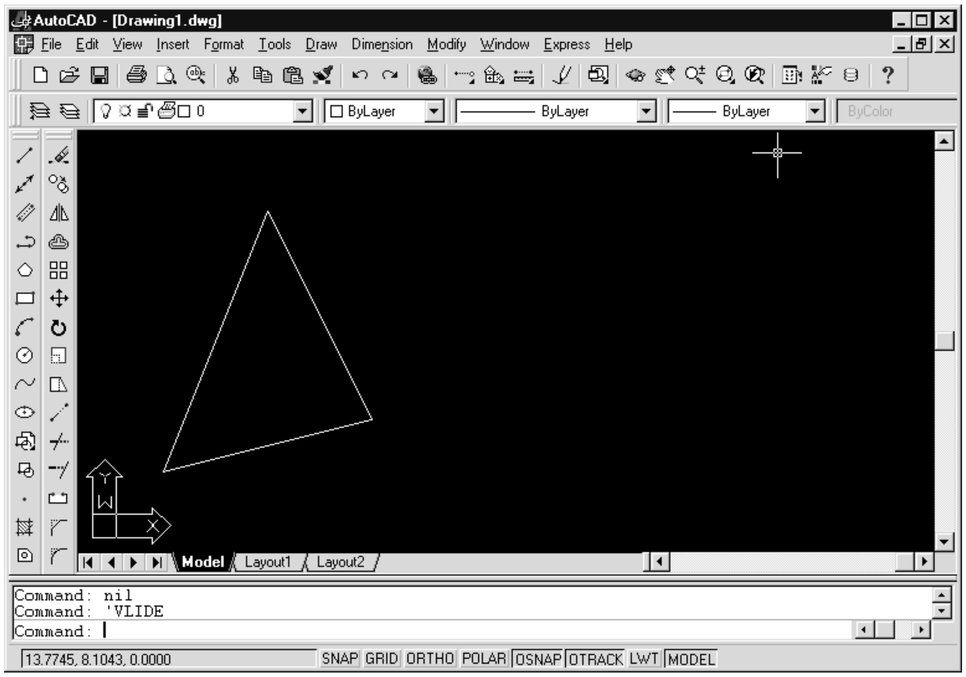
Figure 35-3 *Writing program code in Visual LISP Text Editor*

Loading and Running Programs

1. Make sure the Visual Lisp Text Editor window is active. If not, click a point anywhere in the window to make it active.

2. In the **Tools** menu select **Load Text in Editor**. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. Visual LISP displays a message in the Console window indicating that the program has been loaded. If there was a problem in loading the program, an error message will appear.

3. To run the program, enter the function name **(tr1)** at the console prompt (-$ sign). The function name must be enclosed in parentheses. The program will draw a triangle in AutoCAD. To view the program output, switch to AutoCAD by choosing **Activate AutoCAD** in the **View** toolbar. You can also run the program from AutoCAD. Switch to AutoCAD and enter the name of the function **TR1** at the Command prompt. AutoCAD will run the program and draw a triangle on the screen, Figure 35-4.


AutoCAD - [Drawing1.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Window Express Help
ByLayer
ByColor
Model
Layout1
Layout2
SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL

```
Command: nil
Command: 'VLIDE
Command:
```

```
13.7745, 8.1043, 0.0000
```

Figure 35-4 *Program output in AutoCAD*

Exercise 1 — *General*

Write a LISP program that will draw an equilateral triangle. The program should prompt the user to select the starting point (bottom left corner) and the length of one of the sides of the triangle, Figure 35-5.

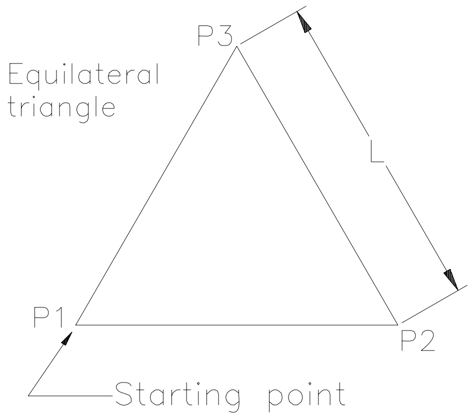


Figure 35-5 *Drawing for Exercise 1*

Loading Existing AutoLISP Files

You can also load the AutoLISP files in Visual LISP and then edit, debug, load, and run the programs from Visual LISP.

1. Start AutoCAD. In the **Tools** menu select **AutoLISP** and then select **Visual LISP Editor**. You can also start Visual LISP by entering the **VLIDE** command at the Command prompt.

2. In the Visual LISP Text Editor, choose **Open File** in the **File** menu. The **Open file to edit/ view** dialog box is displayed on the screen.

3. Select the AutoLISP program that you want to load and then choose **Open**. The program is loaded in the Visual LISP Text Editor.

4. To format the program in the edit window, select the **Format edit window** tool from the **Tools** toolbar. The program will automatically get formatted.

5 To load the program, choose **Load active edit window** in the **Tools** toolbar or choose **Load Text in Editor** from the **Tools** menu.

6. To run the program, enter the function name at the console prompt ($) in the Visual LISP Console window and then press enter. If it gives nil, this means it has executed the drawing on the graphical screen.

Visual LISP Console

Most of the programming in Visual LISP is done in the Visual LISP Text Editor. However, you can also enter the code in the Visual LISP console and see the results immediately. For example, if you enter **(sqrt 37.2)** at the console prompt (after the -$ sign) and press the **Enter** key, Visual LISP will evaluate the statement and return the value 6.09918. Similarly, enter the code **(setq x 99.3)**, **(+ 38 23.44)**, **(- 23.786 35)**, **(abs -37.5)**, **(sin 0.333)** and see the results. Each statement must be entered after the $ sign as shown in Figure 35-6.

Features of the Visual LISP Console

1. In the Visual LISP console you can enter more than one statement on a line and then press the Enter key. Visual LISP will evaluate all statements and then return the value. Enter the following statement after the $ sign:
   -$(setq x 37.5) (setq y (/ x 2))
   Visual LISP will display the value of X and Y in the console, Figure 35-7

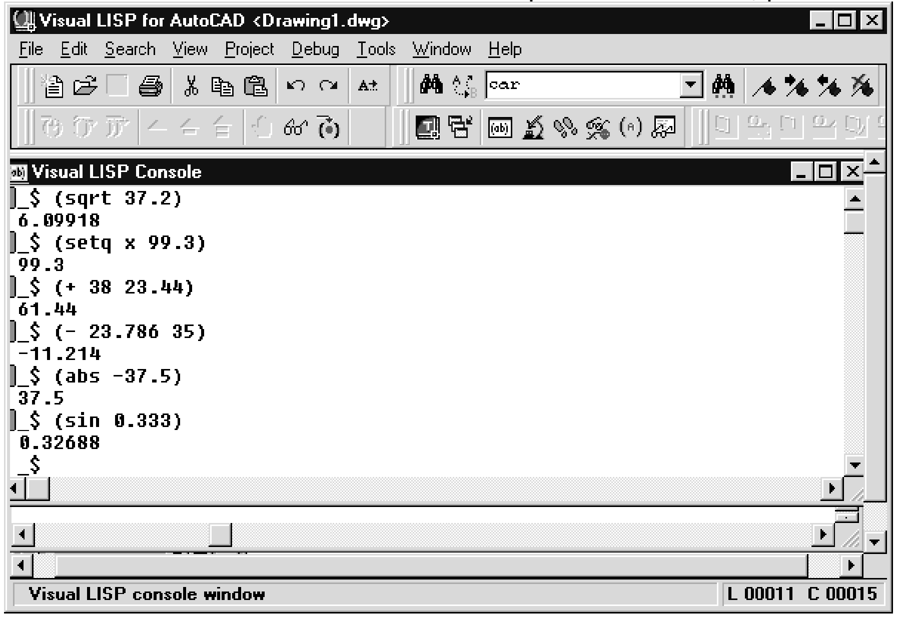
2. If you want to find the value of a variable in the code being developed, enter the variable name after the $ sign and Visual LISP will return the value and display it in the console. For example, if you want to find the value of X, enter X after the $ sign.

3. Visual LISP allows you to write an AutoLISP expression on more than one line by pressing **Ctrl+Enter** at the end of the line. For example, if you want to write the following two expressions on two lines, enter the first line and then press the **Ctrl+Enter** keys. You will notice that Visual LISP does not display the $ sign in the next line, indicating continuation

of the statement on the previous line. Now enter the next line and press the **Enter** key, Figure 35-7.

$ (setq n 38) Press the **Ctrl+Enter** keys
(setq counter (- n 1))

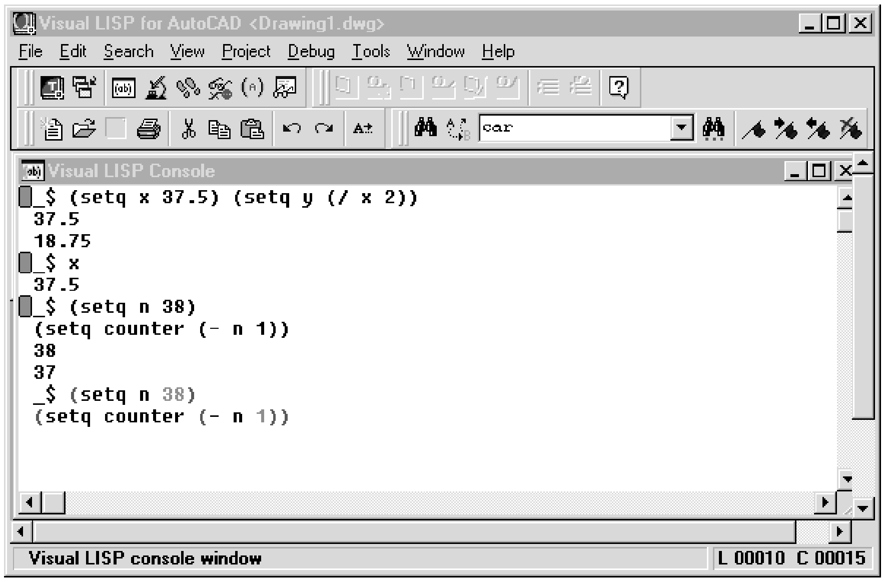
4. To retrieve the text that has been entered in the previous statement, press the **Tab** key


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
car
Visual LISP Console

```
_$ (sqrt 37.2)
6.09918
_$ (setq x 99.3)
99.3
_$ (+ 38 23.44)
61.44
_$ (- 23.786 35)
-11.214
_$ (abs -37.5)
37.5
_$ (sin 0.333)
0.32688
_$
```

Visual LISP console window L 00011 C 00015

Figure 35-6 *Entering LISP code in Visual LISP Console*


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
car
Visual LISP Console

```
_$ (setq x 37.5) (setq y (/ x 2))
37.5
18.75
_$ x
37.5
_$ (setq n 38)
(setq counter (- n 1))
38
37
_$ (setq n 38)
(setq counter (- n 1))
```

Visual LISP console window L 00010 C 00015

Figure 35-7 *Entering LISP code in Visual LISP Console*

while at the Console prompt (-$). Each time you press the **Tab** key, the previously entered text is displayed. You can continue pressing the **Tab** key to cycle through the text that has been entered in the Visual LISP Console. If the same statement has been entered several times, pressing the Tab key will display the text only once and then jump to the statement before that. Pressing the **Shift+Tab** key displays the text in the opposite direction.

5. Pressing the **Ese** key clears the text following the Console prompt (-$). However, if you press the **Shift+Esc** keys, the text following the Console prompt is not evaluated and stays as is. The cursor jumps to the next Console prompt. For example, if you enter the expression **(setq x 15)** and then press the **Esc** key, the text is cleared. However, if you press the **Shift+Esc** keys, the expression (setq x 15) is not evaluated and the cursor jumps to the next Console prompt without clearing the text.

6. The **Tab** key also lets you do an associative search for the expression that starts with the specified text string. For example, if you want to search the expression that started with (sin, enter it at the Console prompt (-$) and then press the **Enter** key. Visual LISP will search the text in the Console window and return the statement that starts with (sin. If it cannot find the text, nothing happens, except maybe a beep.

7. If you click the **right mouse** button (right-click) anywhere in the Visual LISP Console or press the **Shift+F10** keys, Visual LISP displays the context menu, Figure 35-8. You can use the context menu to perform several tasks like Cut, Copy Paste, Clear Console Window, Find, Inspect, Add Watch, Apropros Window, Symbol Service, Undo, Redo, AutoCAD Mode, and Toggle Console Log.

8. One of the important features of Visual LISP is the facility of context-sensitive help available for the Visual LISP functions. If you need help with any Visual LISP function, enter or select the function and then choose the Help button in the **Tools** toolbar. Visual LISP will display the help for the selected function, Figure 35-9.

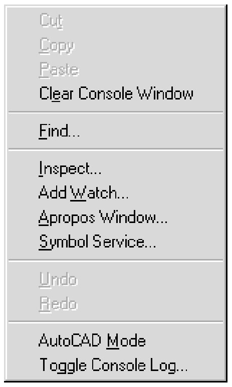


Figure 35-8 *Context menu displayed when you click the right mouse button*

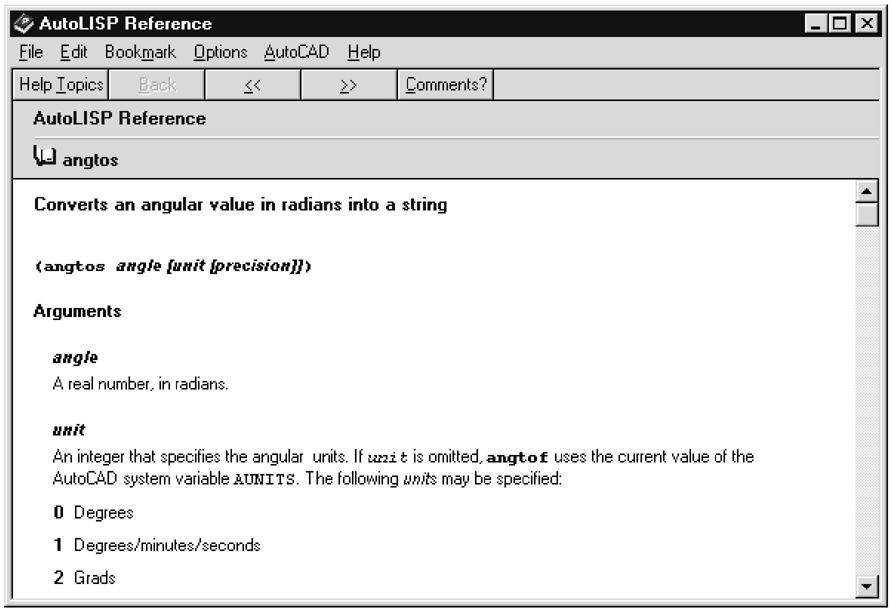
9. Visual LISP also allows you to keep a record of the Visual LISP Console activities by saving them on the disk as a log files (.log). To create a log file, select **Toggle Console Log** in the **File** menu or select **Toggle Console Log** from the context menu. The Visual LISP Console window must be active to create a log file.

10. As you enter the text, Visual LISP automatically assigns a color to it. The colors are assigned based on the nature of the text string. For example, if the text is a built-in function or a protected symbol, the color assigned is blue. Similarly, text strings are magenta, integers are green, parentheses are red, real numbers are teal, and comments are magenta.

11. Although native AutoLISP and Visual LISP are separate programming environments, you

PKJain

can transfer any text entered in the Visual LISP Console to AutoCAD Command line. To accomplish this, enter the code at the Console prompt and then select **AutoCAD Mode** in the context menu or from the **Tools** menu. The Console prompt is replaced with the Command prompt. Press the **Tab** key to display the text that was entered at the Console prompt. Press the ENTER key to switch to the AutoCAD screen. The text is displayed at AutoCAD's Command prompt.



AutoLISP Reference

angtos

Converts an angular value in radians into a string

(angtos *angle [unit [precision]]*)

Arguments

angle
A real number, in radians.

unit
An integer that specifies the angular units. If *unit* is omitted, **angtof** uses the current value of the AutoCAD system variable AUNITS. The following *units* may be specified:

0 Degrees
1 Degrees/minutes/seconds
2 Grads

Figure 35-9 *Context-sensitive help screen*

Exercise 2

General

Enter the following statements in the Visual LISP Console at the console prompt (-$):

```
(+ 2 30 4 38.50)
(- 20 39 32)
(* 2.0 -7.6 31.25)
(/ -230 -7.63 2.15)
(sin 1.0472)
(atan -1.0)
< 3 10)
(<= -2.0 0)
(setq variable_a 27.5)
(setq variable_b (getreal "Enter a value:"))
```

Using the Visual LISP Text Editor

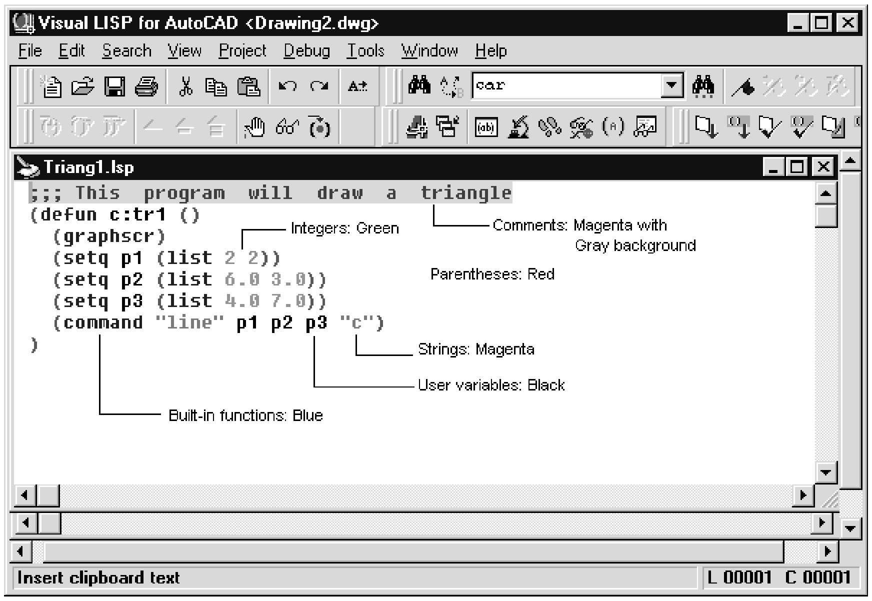
You can use the **Visual LISP Text Editor** to enter the programming code. It has several features that are not available in other text editors. For example, when you enter text, it is automatically assigned a color based on the nature of the text string. If the text is in parenthesis, it is assigned red color. If the text is a Visual LISP function, it is assigned blue color. Visual

LISP Text Editor also allows you to execute an AutoLISP function without leaving the text editor, and also checks for balanced parentheses. These features make it an ideal tool for writing Visual LISP programs.

Color Coding

Visual LISP Text Editor provides color coding for files that are recognized by Visual LISP. Some of the files that it recognizes are LISP, DCL, SQL, and C language source files. When you enter text in the text editor, Visual LISP automatically determines the color and assigns it to the text, Figure 35-10. The following table shows the default color scheme:

Visual LISP Text Element	**Color**
Parentheses	Red
Built-in functions	Blue
Protected symbols	Blue
Comments	Magenta with gray background
Strings	Magenta
Real numbers	Teal
Integers	Green
Unrecognized items	Black (like variables)



```
;;; This program will draw a triangle
(defun c:tr1 ()
  (graphscr)
  (setq p1 (list 2 2))
  (setq p2 (list 6.0 3.0))
  (setq p3 (list 4.0 7.0))
  (command "line" p1 p2 p3 "c")
)
```

Figure 35-10 *Color coding for LISP code*

Words in the Visual LISP Text Editor

In most of the text editors the words are separated by space. In the Visual LISP Text Editor, the word has a different meaning. A word is a set of characters that are separated by one or more of the following special characters:

Character Description	**Visual LISP Special Characters**
Space	
Tab	
Single quote	‘
Left parenthesis	(
Right parenthesis	)
Double quotes	“
Semicolon	;
Newline	\n

For example, if you enter the text (Command”Line”P1 P2 P3”c), Visual LISP treats Command and Line as separate words because they are separated by double quotes. In a regular text editor (Command”Line” would have been treated as a single word. Similarly, in **(setq p1,** setq and p1 are separate words because they are separated by a space. If there was no space, Visual LISP will treat **(setqp1** as one word.

Context-Sensitive Help

One of the important features of Visual LISP is the facility of context-sensitive help available for the Visual LISP functions. If you need help with any Visual LISP function, enter or select the function in the text editor and then choose the **Help** button in the **Tools** toolbar. Visual LISP will display the help for the selected function.

Context Menu

If you click the **right mouse** button (right-click) anywhere in the Visual LISP Text Editor or press the **Shift+F10** keys, Visual LISP displays the context menu, Figure 35-11. You can use the context menu to perform several tasks like Cut, Copy, Paste, Clear Console Window, Find, Inspect, Add Watch, Apropos Window, Symbol Service, Undo, Redo, AutoCAD Mode, and Toggle Console Log. The following table contains a short description of these functions:

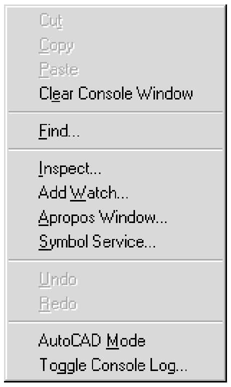


Figure 35-11 *Context menu displayed when you click the right mouse button*

<u>**Functions in the context menu**</u>	<u>**Description of the function**</u>
Cut	Moves the text to the clipboard
Copy	Copies the text to the clipboard
Paste	Pastes the contents of the clipboard at the cursor position
Find	Finds the specified text
Go to Last Edited	Moves the cursor to the last edited position
Toggle Breakpoint	Sets or removes a breakpoint at the cursor position
Inspect	Opens the **Inspect** dialog box
Add Watch	Opens the **Add Watch** dialog box
Apropos Window	Opens **Apropos options** dialog box

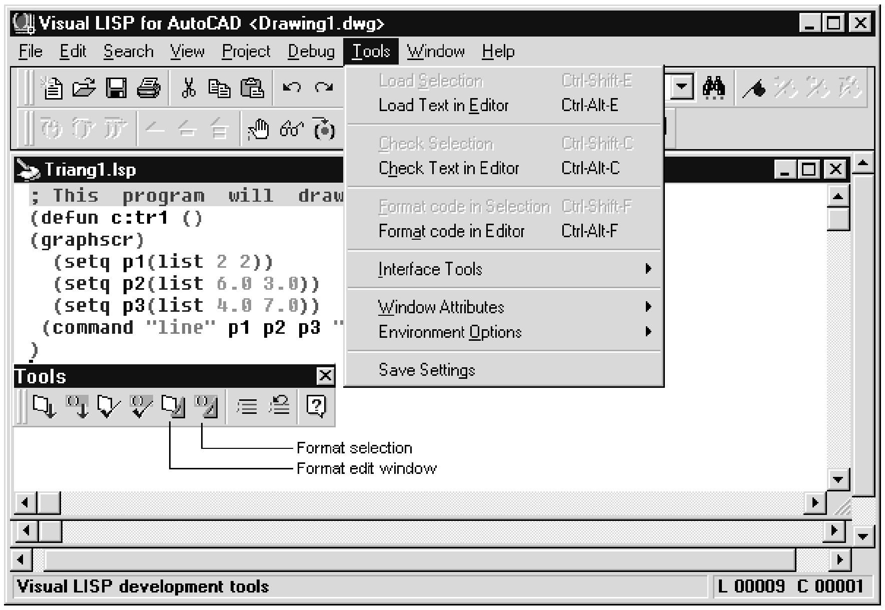
Symbol Service	Opens S**ymbol Service** dialog box
Undo	Reverses the last operation
Redo	Reverses the previous Undo

Visual LISP Formatter

You can use the Visual LISP Formatter to format the text entered in the text editor. When you format the text, the formatter automatically arranges the AutoLISP expressions in a way that makes it easy to read and understand the code. It also makes it easier to debug the program when the program does not perform the intended function.

Running the Formatter

With the formatter you can format all the text in the Visual LISP text editor window or the selected text. To format the text, the text editor window must be active. If it is not, click a point anywhere in the window to make it active. To format all the text, select the **Format edit window** tool in the **Tools** toolbar or select **Format code in Editor** from the **Tools** menu. To format part of the text in the text editor, select the text and then select the **Format selection** tool in the **Tools** toolbar or select **Format code in Selection** from the **Tools** menu. Figure 35-12 shows the **Tools** menu and **Tools** toolbar.


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
Load Selection Ctrl-Shift-E
Load Text in Editor Ctrl-Alt-E
Check Selection Ctrl-Shift-C
Check Text in Editor Ctrl-Alt-C
Format code in Selection Ctrl-Shift-F
Format code in Editor Ctrl-Alt-F
Interface Tools
Window Attributes
Environment Options
Save Settings
Triang1.lsp
Tools
Format selection
Format edit window
Visual LISP development tools
L 00009 C 00001

```
; This  program  will  draw
(defun c:tr1 ()
(graphscr)
  (setq p1(list 2 2))
  (setq p2(list 6.0 3.0))
  (setq p3(list 4.0 7.0))
 (command "line" p1 p2 p3 "
)
```

Figure 35-12 *Using Format selection or Format window tools*

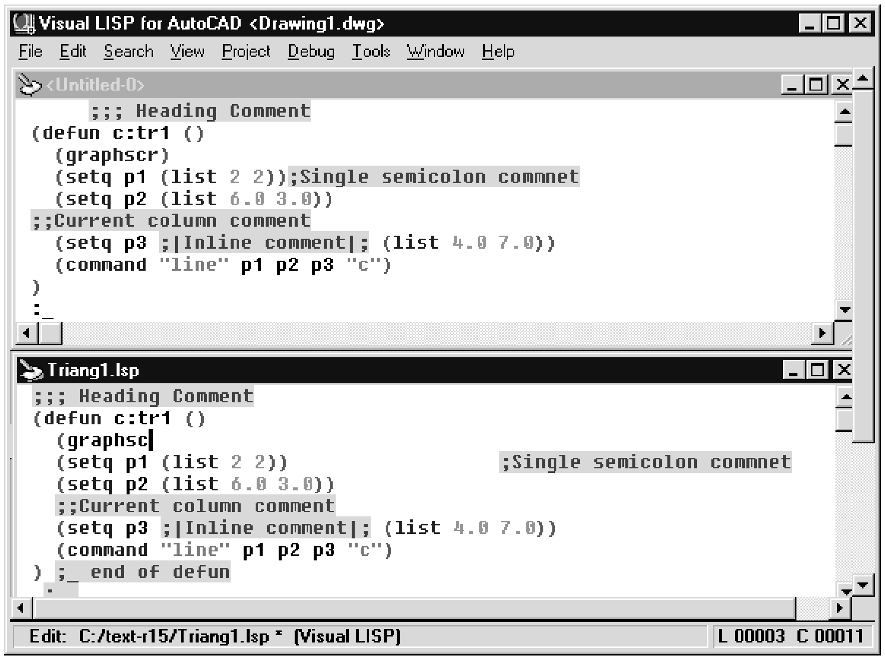
Formatting Comments

In Visual LISP you can have five types of comments and depending on the comment in the program code, Visual LISP will format and position the comment text according to its type. Following is the list of various comment types available in Visual LISP:

Comment formatting	**Description**
; Single semicolon	The single semicolon is indented as specified in the

	Single-semicolon comment indentation of AutoLISP Format option.
;; Current column comment	The comment is indented at the same position as the previous line of the program code.
;;; Heading comment	The comment appears without any indentation.
;| Inline comment|;	The Inline comments appear as any other expression.
;_ Function closing comment	The function closing comment appears just after the previous function, provided Insert form-closing comment is on in the AutoLISP Format options.

Two text editor windows are shown in Figure 35-13. The window at the top has the code without formatting. The text window at the bottom has been formatted. If you compare the two, you will see the difference formatting makes in the text display.


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
<Untitled-0>
Triang1.lsp

```
      ;;; Heading Comment
 (defun c:tr1 ()
   (graphscr)
   (setq p1 (list 2 2));Single semicolon commnet
   (setq p2 (list 6.0 3.0))
 ;;Current column comment
   (setq p3 ;|Inline comment|; (list 4.0 7.0))
   (command "line" p1 p2 p3 "c")
 )
 ;_
```

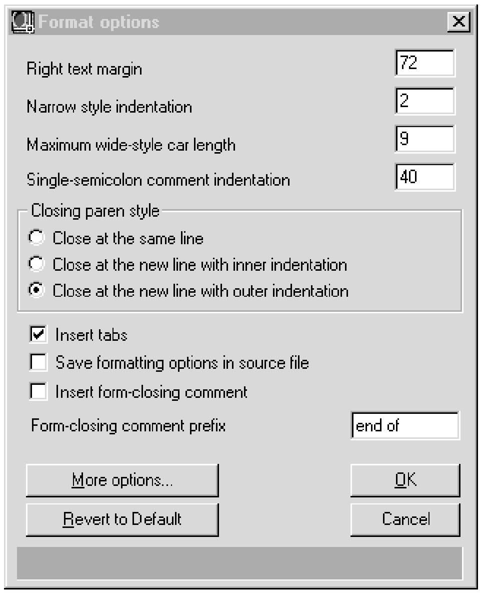
```
;;; Heading Comment
(defun c:tr1 ()
  (graphsc
  (setq p1 (list 2 2))              ;Single semicolon commnet
  (setq p2 (list 6.0 3.0))
  ;;Current column comment
  (setq p3 ;|Inline comment|; (list 4.0 7.0))
  (command "line" p1 p2 p3 "c")
) ;_ end of defun
```

```
Edit: C:/text-r15/Triang1.lsp * (Visual LISP)    L 00003  C 00011
```

Figure 35-13 *Top window shows code without formatting and the bottom window shows the same code after formatting the programming code*

Changing Formatting Options

You can change the formatting options through the **Format options** dialog box (Figure 35-14) that can be invoked by choosing the **Tools** menu, **Environment Options**, and then **Visual LISP Format Options**. In this dialog box if you select **More** options button, Visual LISP will display another dialog box (Figure 35-15) listing more formatting options.



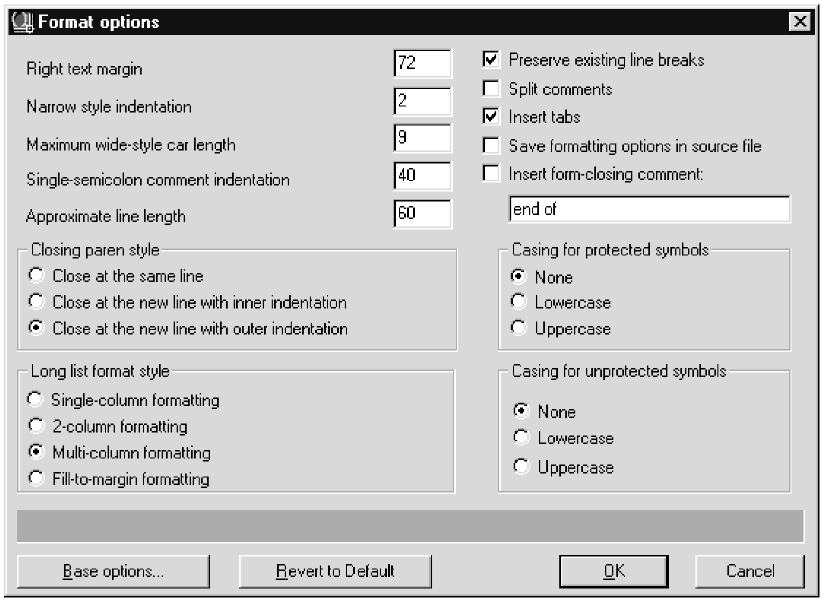
Right text margin: 72
Narrow style indentation: 2
Maximum wide-style car length: 9
Single-semicolon comment indentation: 40

Closing paren style
- Close at the same line
- Close at the new line with inner indentation
- [✓ Close at the new line with outer indentation]

- [✓] Insert tabs
- [ ] Save formatting options in source file
- [ ] Insert form-closing comment

Form-closing comment prefix: end of

Figure 35-14 *Format options dialog box*



Right text margin: 72
Narrow style indentation: 2
Maximum wide-style car length: 9
Single-semicolon comment indentation: 40
Approximate line length: 60

- [✓] Preserve existing line breaks
- [ ] Split comments
- [✓] Insert tabs
- [ ] Save formatting options in source file
- [ ] Insert form-closing comment: end of

Closing paren style
- Close at the same line
- Close at the new line with inner indentation
- [✓ Close at the new line with outer indentation]

Casing for protected symbols
- [✓ None]
- Lowercase
- Uppercase

Long list format style
- Single-column formatting
- 2-column formatting
- [✓ Multi-column formatting]
- Fill-to-margin formatting

Casing for unprotected symbols
- [✓ None]
- Lowercase
- Uppercase

Figure 35-15 *Format options dialog box with more options*

DEBUGGING THE PROGRAM

When you write a program, it is rare that the program will run the first time. Even if runs, it may not perform the desired function. The error could be a syntax error, missing parentheses, a misspelled function name, improper use of a function, or simply a typographical error. It is time consuming and sometimes difficult to locate the source of the problem. Visual LISP comes with several debugging tools that make it easier to locate the problem and debug the code. Make sure the Debug toolbar is displayed on the screen. If it is not, in the Visual LISP window click on the **View** menu and then select **Toolbars** to display the **Toolbars** dialog box. Select the **Debug** check box, choose the **Apply** button, and then choose the **Close** button to exit the dialog box. The **Debug** toolbar will appear on the screen. Figure 35-16 shows the toolbar with tool tips.

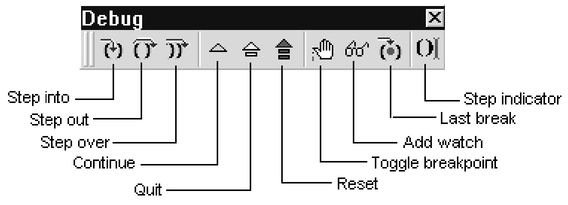


Figure 35-16 *Tools in Debug toolbar*

1. To try some of the debugging tools, enter the following program in the Visual LISP Text Editor and then save the file as **triang2.lsp**:

```
;;; This program will draw a triangle and an arc
(defun tr2 ()
  (setq p1 (getpoint "\nEnter first point p1:"))
  (setq p2 (getpoint "\nEnter second point p2:"))
  (setq p3 (getpoint "\nEnter third point p3:"))
  (command “arc” p1 p2 p3)
  (command "line" p1 p2 p3 "c")
)
```

2. To format the code, choose the **Format edit window** tool in the **Tools** toolbar. You can also format the code by selecting **Format code in Editor** in the Tools menu.

3. Load the program by choosing the **Load active edit window** tool in the **Tools** toolbar. You can also load the program from the **Tools** menu by selecting **Load Text in Editor**.

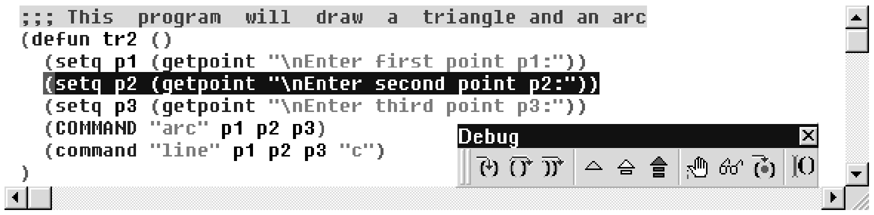
4. In the Visual LISP Text Editor, position the cursor in front of the following line and then choose the **Toggle breakpoint** tool in the **Debug** toolbar. You can also select it from the **Debug** menu. Visual LISP inserts a breakpoint at the cursor location.

```
|(setq p2 (getpoint "\Enter second point p2:"))
```

5. In the Visual LISP Console, at the Console prompt, enter the name of the function and then press the **Enter** key to run the program.

   -$ **(tr2)**

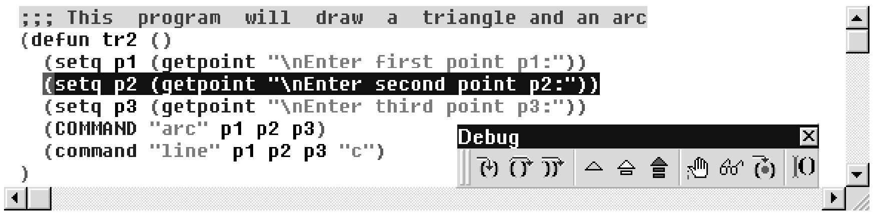
   The AutoCAD window will appear and the first prompt in the program (Enter first point p1:) will be displayed in the Command prompt area. Pick a point and the Visual LISP window will appear. Notice, the line that follows the breakpoint is highlighted as shown in Figure 35-17.



```
;;; This  program  will  draw  a  triangle and an arc
(defun tr2 ()
  (setq p1 (getpoint "\nEnter first point p1:"))
  (setq p2 (getpoint "\nEnter second point p2:"))
  (setq p3 (getpoint "\nEnter third point p3:"))
  (COMMAND "arc" p1 p2 p3)
  (command "line" p1 p2 p3 "c")
)
```

Figure 35-17 *Using Step Into command to start execution until it encounters the next expression*

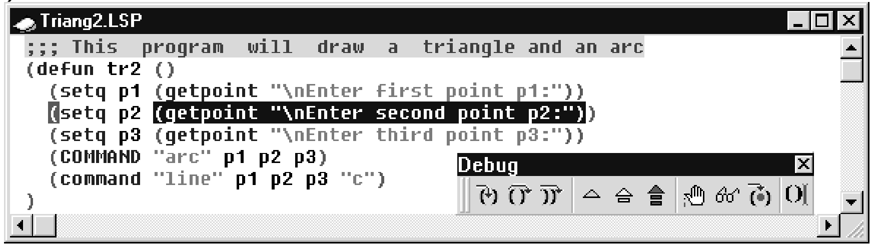
6. Choose the **Step Into** button in the **Debug** toolbar or select **Step Into** from the **Debug** menu. You can also press the function key **F8** to invoke the **Step Into** command. The program starts execution until it encounters the next expression. The next expression that is contained within the inside parentheses is highlighted, Figure 35-18. Notice the position of the vertical bar in the Step indicator tool in the Debug toolbar. It is on the left of the Step indicator tool. Also, the cursor is just before the expression ("getpoint "\nEnter second point:).



```
;;; This  program  will  draw  a  triangle and an arc
(defun tr2 ()
  (setq p1 (getpoint "\nEnter first point p1:"))
  (setq p2 (getpoint "\nEnter second point p2:"))
  (setq p3 (getpoint "\nEnter third point p3:"))
  (COMMAND "arc" p1 p2 p3)
  (command "line" p1 p2 p3 "c")
)
```

Figure 35-18 *Using Step Into command again, the cursor jumps to the end of the expression*

7. Choose the **Step Into** button again. The AutoCAD window appears and the second prompt in the program (Enter second point p2:) is displayed in the Command prompt area. Pick a point, and the Visual LISP window returns, Figure 35-19. Notice the position of the vertical bar in the **Step indicator** tool in the **Debug** toolbar. It is just to the right of the Step Indicator. Also, the cursor is just after the expression ("getpoint "\nEnter second point:).



```
;;; This  program  will  draw  a  triangle and an arc
(defun tr2 ()
  (setq p1 (getpoint "\nEnter first point p1:"))
  (setq p2 (getpoint "\nEnter second point p2:"))
  (setq p3 (getpoint "\nEnter third point p3:"))
  (COMMAND "arc" p1 p2 p3)
  (command "line" p1 p2 p3 "c")
)
```

Figure 35-19 *Using Step Into command again, the vertical bar is displayed to the right of Step Indicator symbol*

8. Choose the **Step Into** button again to step through the program. The entire line is

PKJain

highlighted and the cursor moves to the end of the expression (getpoint “Enter second point:). Notice the position of the vertical bar in the **Step indicator** tool in the **Debug** toolbar.

9. Choose the **Step Into** button again. The next line of the program is highlighted and the cursor is at the start of the statement.

10 To step over the highlighted statement, choose the **Step over** tool in the **Debug** toolbar, or select **Step Over** in the Debug menu. You can also press the **Shift+F8** keys to invoke this command. The AutoCAD window is displayed and the prompt (Enter third point:) is displayed in the Command prompt area. Select a point, and the Visual LISP window appears.

11. Choose the **Step over** button again; Visual LISP highlights the next line. If you choose the Step over button again, the entire expression is executed.

12. Choose the **Step over** button until you go through all the statements and the entire program is highlighted.

General Recommendations for writing the LISP file

1. Built-in functions and arguments should be spaced.
2. Arguments should be spaced.
3. Every function must start with an open parenthesis.
4. Every function must end with a closed parenthesis.

Example 1

Write a LISP program to draw the I-section as shown in Figure14-20. The starting point is p1. The point P1, length, width, and the thicknesses of flange and web are user-defined values that are to be entered at the command prompt.

Step 1: Understanding the program algorithm

Before writing the program it is very important to understand what is given, what is the desired output, and how we can get that output from the given information. Analyze the program as follows:

<u>Input</u>

P1	Starting point of the I-section
L	Length of the I-section
W	Width of the I-section
T1	Thickness of the flange
T2	Thickness of the web

<u>Output</u>

I section as shown in Figure 35-20

Process

Define all points like p2, p3, p4, p5, p6, p7, p8, p9, p10, p11 and p12

Draw lines between points

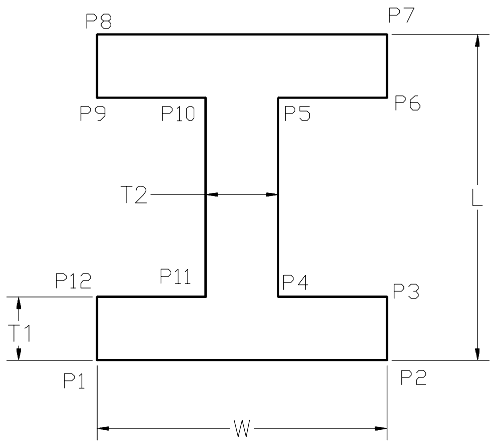


Figure 35-20 *I-section for Example 1*

Step 2: Writing the Visual LISP program

Select **Tools> AutoLISP> Visual LISP Editor** menu or enter **VLIDE** at the Command prompt to display the **Visual LISP** window. Select **New File** from the **File** menu to display the **Visual LISP Text Editor** window. Write the following program in the **Visual LISP Text Editor**.

```
 (defun isec ()
 (setq   p1  (getpoint "\n Enter the starting point of the I-section: ")
         l   (getdist "\n Enter the length of the I-section: ")
         w   (getdist "\n Enter the width of the I-section: ")
         t1  (getdist "\n Enter the thickness of the flange: ")
         t2  (getdist "\n Enter the thickness of the web: ") p2
         (list (+ (car p1) w) (cadr p1))
         p3  (list (car p2) (+ (cadr p2) t1))
         p4  (list (- (car p3) (/ (- w t2) 2)) (cadr p3))
         p5  (list (car p4) (+ (cadr p4) (- l (* 2 t1))))
         p6  (list (car p3) (cadr p5))
         p7  (list (car p6) (+ (cadr p6) t1))
         p8  (list (car p1) (+ (cadr p1) l))
         p9  (list (car p8) (- (cadr p8) t1))
         p10 (list (- (car p5) t2) (cadr p5))
         p11 (list (- (car p4) t2) (cadr p4))
         p12 (list (car p1) (cadr p11))
 )
 (command "PLINE" p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p1 "")
)
```

PKJain

Step 3: Loading and running the Visual LISP program

Save the file as **isec.lsp** using the **Save** or **Save As** option in the **File** menu. To load the program, select **Load Text in Editor** in the **Tools** menu. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. **Visual LISP** displays a message in the **Console** window indicating that program has been loaded. If the Console window doesn't appear on the screen, select the **Tile Horizontally** option from the **Window** toolbar to display it on the screen. It also gives the error message, if there is any error in the program.

The step-by-step debugging can be done as described under the heading **"DEBUGGING THE PROGRAM."** The mismatch in the parentheses can be corrected by choosing the **Format Edit Window** in the **Tools** toolbar. The nature of the error and its meaning are given at the end of this chapter under the heading "**ERROR CODES AND MESSAGES.**" To run the program, enter the function name **(isec)** at the console prompt adjacent to the $ sign. The function name must be enclosed in parentheses. When you enter the function name and press enter, the AutoCAD screen is displayed prompting you to enter the information about the **I**-section. After you enter the information, the **I**-section will be drawn on the screen.

Tip

1. This program can also be entered using any text editor like Notepad. The program can be loaded by entering the ***APPLOAD*** *command at the Command prompt as described in the AutoLISP chapter.*

2. The above example can also be solved by defining only half of the I-section and then mirroring the I-section with the ***MIRROR*** *command to get the complete I-section. You can also draw only one-fourth of the I-section and then use the* ***MIRROR*** *command to get the remaining section.*

Step 4: Writing the Visual LISP program using POLAR function

The points can also be calculated by using the POLAR function and then draw the lines to create the **I**-section. In this program d1=distance between P2 and P3, d2=distance between P3 and P4, and d3 = distance between P4 and P5. Notice the dtr function used to convert the degrees into radians. The following file is the listing of the program using POLAR function.

```
((defun dtr (a)
  (* a (/ pi 180.0))
)
(defun isec ()
 (setq   p1 (getpoint "\n Enter the starting point of the I-section: ")
         l  (getdist "\n Enter the length of the I-section: ")
         w  (getdist "\n Enter the width of the I-section: ")
         t1 (getdist "\n Enter the thickness of the flange: ")
         t2 (getdist "\n Enter the thickness of the web: ")
 )
 (setq   d1 t1
         d2 (- (/ w 2.0) (/ t2 2.0))
         d3 (- l (* 2.0 t1))
 )
 (setq   p2  (polar p1 (dtr 0) w)
         p3  (polar p2 (dtr 90) d1)
         p4  (polar p3 (dtr 180) d2)
         p5  (polar p4 (dtr 90) d3)
         p6  (polar p5 (dtr 0) d2)
         p7  (polar p6 (dtr 90) d1)
         p8  (polar p7 (dtr 180) W)
         p9  (polar p8 (dtr 270) d1)
         p10 (polar p9 (dtr 0) d2)
         p11 (polar p10 (dtr 270) d3)
         p12 (polar p11 (dtr 180) d2)
 )
 (command "PLINE" p1 p2 p3 p4 p5 p6 p7       p8 p9 p10 p11 p12 p1 "")
)
```

Example 2

Write a Visual LISP program that will extrude the **I**-section of Example 1 and create a 3D shape of the section, Figure 35-21. The additional input to the program is vpoint (v), extrusion height (h), and extrusion angle (a).

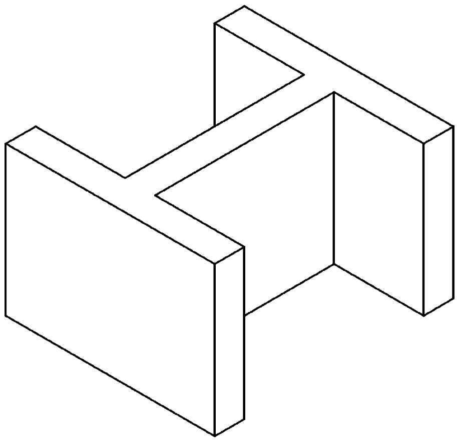


Figure 35-21 *3D I-section for Example 2*

Step 1: Understanding the program algorithm

In addition to the inputs of Example 1, add the Vpoint, Extrusion, and Angle of Extrusion as additional inputs in the program. Also add the commands corresponding to vpoint, extrusion, zoom, and hide. Vpoint, extrusion height, and angle of extrusion have been represented as **v**, **h**, and **a** respectively. Angle has been entered using **getreal** function to accept the angle in decimal degrees. Do not use **getangle** function because this function returns the angle in radians. The extrusion angle in the **EXTRUDE** command must be in degrees.

PKJain

Step 2: Writing the Visual LISP program

Select **Tools> AutoLISP> Visual LISP Editor** menu or enter **VLIDE** at the Command prompt to display the **Visual LISP** window. Select **New File** from the **File** menu to display the **Visual LISP Text Editor** window. Now write the program as shown below in the **Visual LISP Text Editor** or open the LISP file of Example 1 and add the new lines to this program.

```
(defun dtr (a)
 (* a (/ pi 180.0)))
(defun isec ()
 (setq p1 (getpoint "\n Enter the starting point of the I-section: ")
         l  (getdist "\n Enter the length of the I-section: ")
         w  (getdist "\n Enter the width of the I-section: ")
         t1 (getdist "\n Enter the thickness of the flange: ")
         t2 (getdist "\n Enter the thickness of the web: ")
         h  (getdist "\n Enter Extrusion Height: ")
         v  (getpoint "\n Enter the Viewpoint: ")
         a  (getreal "\n Enter Angle of Extrusion: ")
 )
 (setq   d1 t1
         d2 (- (/ w 2.0) (/ t2 2.0))
         d3 (- l (* 2.0 t1))
 )
 (setq   p2  (polar p1 (dtr 0) w)
         p3  (polar p2 (dtr 90) d1)
         p4  (polar p3 (dtr 180) d2)
         p5  (polar p4 (dtr 90) d3)
         p6  (polar p5 (dtr 0) d2)
         p7  (polar p6 (dtr 90) d1)
         p8  (polar p7 (dtr 180) W)
         p9  (polar p8 (dtr 270) d1)
         p10 (polar p9 (dtr 0) d2)
         p11 (polar p10 (dtr 270) d3)
         p12 (polar p11 (dtr 180) d2)
 )
 (command "PLINE" p1 p2 p3 p4 p5 p6 p7       p8 p9 p10 p11 p12 p1 "")
 (command "VPOINT" v)
 (command "EXTRUDE" "All" "" h a)
 (command "ZOOM" "All")
 (command "HIDE")
)
```

Step 3: Loading and running the Visual LISP program

Save the file as **isec3d.lsp** using the **Save** or **Save As** option in the **File** menu. To load the program, select **Load Text in Editor** in the **Tools** menu. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. **Visual LISP** displays a message in the **Console** window indicating that program has been loaded. To run the program enter the function name **(isec3d)** at the console prompt adjacent to $ sign. The function name must be

PKJain

enclosed in parentheses. The program will prompt the user to enter the starting point followed by length, width, thicknesses, vpoint, extrusion height and angle at the Command prompt. After entering all the inputs, a 3D I-section will be displayed on the AutoCAD screen.

Tip

Step by step debugging can be done by choosing the Toggle breakpoint tool in the Debug toolbar as described under the heading ***Debugging the Program (Page 35-14).***

Example 3

Write a Visual LISP program to draw a c-section as shown in the Figure 35-22. The program should prompt the user to enter the starting point, length, width, and thickness.

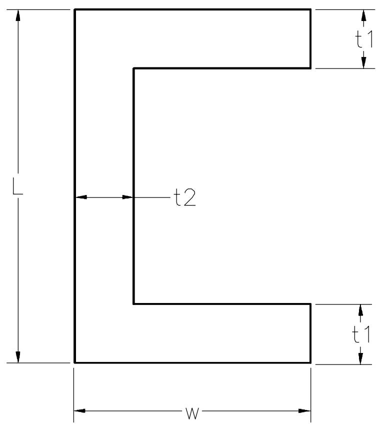


Figure 35-22 *C-section*

Step 1: Writing the Visual LISP program

Select **Tools> AutoLISP> Visual LISP Editor** menu or enter **VLIDE** at the Command prompt to display the **Visual LISP** window. Select **New File** from the **File** menu to display the **Visual LISP Text Editor** window. Now write the program in the **Visual LISP Text Editor**. You can also use any text editor like Notepad to write the LISP program. The following file is the listing of the program that will draw a c-section.

```
;;This program draws a C-Section
(defun csec ()
  (setq p1 (getpoint "\n enter the point:"))
  (setq L (getdist "\n enter the length:"))
  (setq W (getdist "\n enter the width:"))
  (setq t1 (getdist "\n enter the thickness of the flange:"))
  (setq t2 (getdist "\n enter the thickness of the web:"))
  (setq p2 (list (+ (car p1) w) (cadr p1)))
  (setq p3 (list (car p2) (+ (cadr p2) t1)))
```

PKJain

```
 (setq p4 (list (- (car p3) (- w t2)) (cadr p3)))
 (setq p5 (list (car p4) (+ (cadr p4) (- L (* 2 t1)))))
 (setq p6 (list (+ (car p5) (- w t2)) (cadr p5)))
 (setq p7 (list (car p6) (+ (cadr p6) t1)))
 (setq p8 (list (car p1) (cadr p7)))
 (command "pline" p1 p2 p3 p4 p5 p6 p7 p8 p1 "")
)
```

Step 2: Loading and Running the Visual LISP program

Save the file as **csec.lsp** using the **Save** or **Save As** option in the **File** menu. To load the program, select **Load Text in Editor** in the **Tools** menu. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. **Visual LISP** displays a message in the **Console** window indicating that program has been loaded. To run the program enter the function name **(isec3d)** at the console prompt adjacent to the $ sign. The function name must be enclosed in parentheses.

Exercise 3 *General*

Write a Visual LISP program to draw a T-section as shown in Figure 35-23. The program should prompt the user to specify the starting point, length (L), width (w), and thicknesses (t1 and t2) of the T-section.

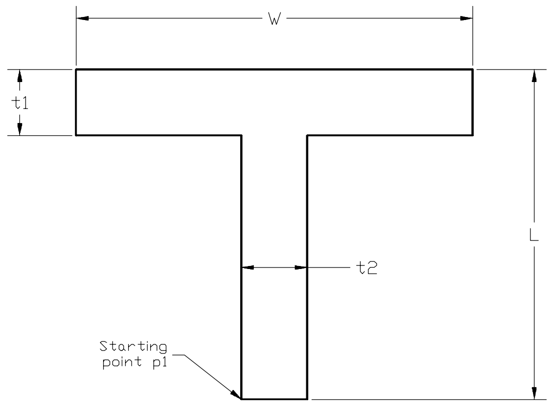


Figure 35-23 *T-section for Exercise 3*

Exercise 4 *General*

Write a Visual LISP program to draw a 3D T-section with the same specifications as Exercise 3. In addition to the inputs in Exercise 3, the program should also prompt the user to specify vpoint, extrusion height, and angle of extrusion.

Exercise 5

General

Write a Visual LISP program to draw an L-section as shown in Figure 35-24. The program should prompt the user to specify starting point, length, width, and thicknesses of the L-section.

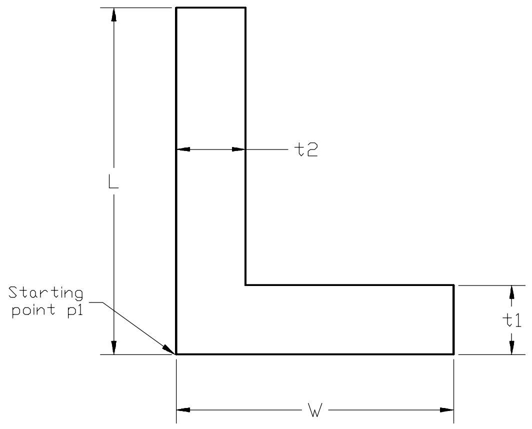


Figure 35-24 *L-section for Exercise 5*

Example 4

Write an Visual LISP program that will generate a flat layout drawing of a transition and then dimension the layout. The transition and the layout without dimensions are shown in Figure 35-25.

Step 1: Writing the Visual LISP program

Select **Tools> AutoLISP> Visual LISP Editor** menu or enter **VLIDE** at the Command prompt to display the **Visual LISP** window. Select **New File** from the **File** menu to display the **Visual LISP Text Editor** window. Now write the program in the **Visual LISP Text Editor** window. You can also use any text editor like Notepad to write the LISP program. The following file is the listing of the **LISP** program for Example 4. The LISP programs do not need to be in lowercase letters. They can be uppercase or a combination of uppercase and lowercase.

```
;This program generates flat layout of
;a rectangle to rectangle transition
;
(defun TRANA ()
(graphscr)
(setvar “cmdecho” 0)
(setq L (getdist “\n Enter length of bottom rectangle: “))
(setq W (getdist “\n Enter width of bottom rectangle: “))
(setq H (getdist “\n Enter height of transition: “))
(setq L1 (getdist “\n Enter length of top rectangle: “))
(setq W1 (getdist “\n Enter width of top rectangle: “))
```

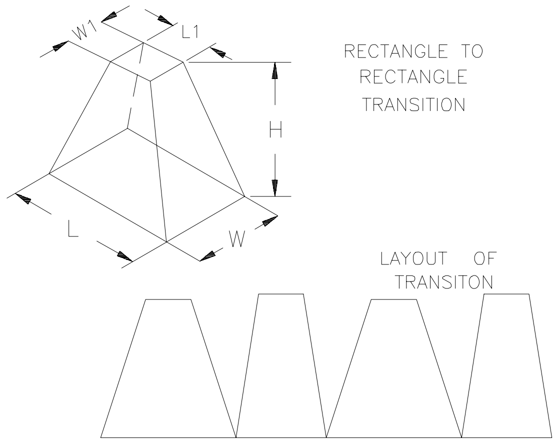


Figure 35-25 *Flat layout of a transition*

```
(setq x1 (/ (- w w1) 2))
(setq y1 (/ (- l l1) 2))
(setq d1 (sqrt (+ (* h h) (* x1 x1))))
(setq d2 (sqrt (+ (* d1 d1) (* y1 y1))))
(setq s1 (/ (- l l1) 2))
(setq p1 (sqrt (- (* d2 d2) (* s1 s1))))
(setq s2 (/ (- w w1) 2))
(setq p2 (sqrt (- (* d2 d2) (* s2 s2))))

(setq t1 (+ l1 s1))
(setq t2 (+ l w))
(setq t3 (+ l s2 w1))
(setq t4 (+ l s2))
(setq pt1 (list 0 0))
(setq pt2 (list s1 p1))
(setq pt3 (list t1 p1))
(setq pt4 (list l 0))
(setq pt5 (list t4 p2))
(setq pt6 (list t3 p2))
(setq pt7 (list t2 0))
(command “layer” “make” “ccto” “c” “1” “ccto” “”)
(command “line” pt1 pt2 pt3 pt4 pt5 pt6 pt7 “c”)

(setq sf (/ (+ l w) 12))
(setvar “dimscale” sf)
(setq c1 (list 0 (- 0 (* 0.75 sf))))
(setq c7 (list (- 0 (* 0.75 sf)) 0))
(setq c8 (list (- l (* 0.75 sf)) 0))
```

```
(command "layer" "make" "cctd" "c" "2" "cctd" "")
(command "dim" "hor" pt1 pt2 c1 "" "base" pt3 "" "base" pt4 "" "exit")
(command "dim" "hor" pt4 pt5 c1 "" "base" pt6 "" "base" pt7 "" "exit")
(command "dim" "vert" pt1 pt2 pt2 "" "exit")
(command "dim" "vert" pt4 pt5 pt5 "" "exit")
(command "dim" "aligned" pt1 pt2 c7 "" "exit")
(command "dim" "aligned" pt4 pt5 c8 "" "exit")
(setvar "cmdecho" 1)
(princ))
```

Step 2: Loading and Running the Visual LISP program

Save the file as **trana.lsp** using the **Save** or **Save As** option in the **File** menu. To load the program, select **Load Text in Editor** in the **Tools** menu. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. **Visual LISP** displays a message in the **Console** window indicating that the program has been loaded. To run the program enter the function name **(isec3d)** at the console prompt adjacent to the $ sign. The function name must be enclosed in parentheses.

Example 5

Write an Visual LISP program that can generate a flat layout of a cone as shown in Figure 35-26. The program should also dimension the layout.

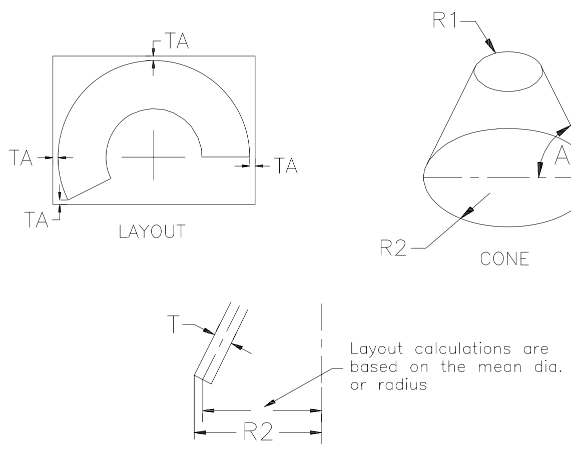


Figure 35-26 *Flat layout of a cone*

Step 1: Writing the Visual LISP program

```
;The following file is a listing of the Visual LISP program for Example 5:
;This program generates layout of a cone
;
;DTR function changes degrees to radians
(defun DTR (a)
 (* PI (/ A 180.0))
 )
;RTD  Function  changes  radians  to degrees
(defun rtd (a)
(* a (/ 180.0 pi))
)

(defun tan (a)
(/ (sin a) (cos a))
)
(defun cone-1p ()
(graphscr)
(setvar “cmdecho” 0)
(setq r2 (getdist “\n enter outer radius at larger end: “))
(setq r1 (getdist “\n enter inner radius at smaller end: “))
(setq t (getdist “\n enter sheet thickness:-”))
(setq a (getangle “\n enter cone angle:-”))
;this part of the program calculates various parameters
;needed in calculating the strip layout
(setq x0 0)
(setq y0 0)
(setq sf (/ r2 3))
(setvar “dimscale” sf)
(setq ar a)
(setq tx (/ (* t (sin ar)) 2))
(setq rx2 (- r2 tx))
(setq rx1 (+ r1 tx))
(setq w (* (* 2 pi) (cos ar)))
(setq rl1 (/ rx1 (cos ar)))
(setq rl2 (/ rx2 (cos ar)))

;this part of the program calculates the x-coordinate
;of the points
  (setq x1 (+ x0 rl1)
       x3 (+ x0 rl2)
      x2 (- x0 (* rl1 (cos (- pi w))))
      x4 (- x0 (* rl2 (cos (- pi w))))
   )

;this part of the program calculates the y-coordinate
;of the points
  (setq y1 y0
      y3 y0
      y2 (+ y0 (* rl1 (sin (- pi w))))
      y4 (+ y0 (* rl2 (sin (- pi w))))
      )

  (setq p0 (list x0 y0)
         p1 (list x1 y1)
         p2 (list x2 y2)
          p3 (list x3 y3)
         p4 (list x4 y4)
         )
      (command “layer” “make” “ccto” “c” “1” “ccto” “”)
      (command  “arc” p1 “c” p0 p2)
      (command  “arc” p3 “c” p0 p4)
      (command  “line” p1 p3 “”)
      (command  “line” p2 p4 “”)

      (setq f1 (/ r2 24))
      (setq f2 (/ r2 2))
      (setq d1 (list (+ x3 f2) y3))
      (setq d2 (list x0 (- y0 f2)))

    (command “layer” “make” “cctd” “c” “2” “cctd” “”)
    (setvar “dimtih” 0)
    (command “dim” “hor” p0 p1 d2 “” “baseline” p3 “” “baseline” p2 “” “baseline” p4 “”
    “exit”)
    (command “dim” “vert” p0 p2 d1 “” “baseline” p4 “” “exit”)
    (setvar “dimscale” 1)
    (setvar “cmdecho” 1)
    (princ)
    )
```

Step 2: Loading and running the Visual LISP program

Save the file as **cone-1p.lsp** using the **Save** or **Save As** option in the **File** menu. To load the program, select **Load Text in Editor** in the **Tools** menu. You can also load the program by choosing **Load active edit window** in the **Tools** toolbar. **Visual LISP** displays a message in the **Console** window indicating that program has been loaded. To run the program enter the function name **(isec3d)** at the console prompt adjacent to the $ sign. The function name must be enclosed in parentheses.

TRACING VARIABLES

Sometimes it becomes necessary to trace the values of the variables used in the program. For example, let us assume the program crashes or does not perform as desired. Now you want to locate where the problem occurred. One of the ways you can do this is by tracing the value of the variables used in the program. The following steps illustrate this procedure:

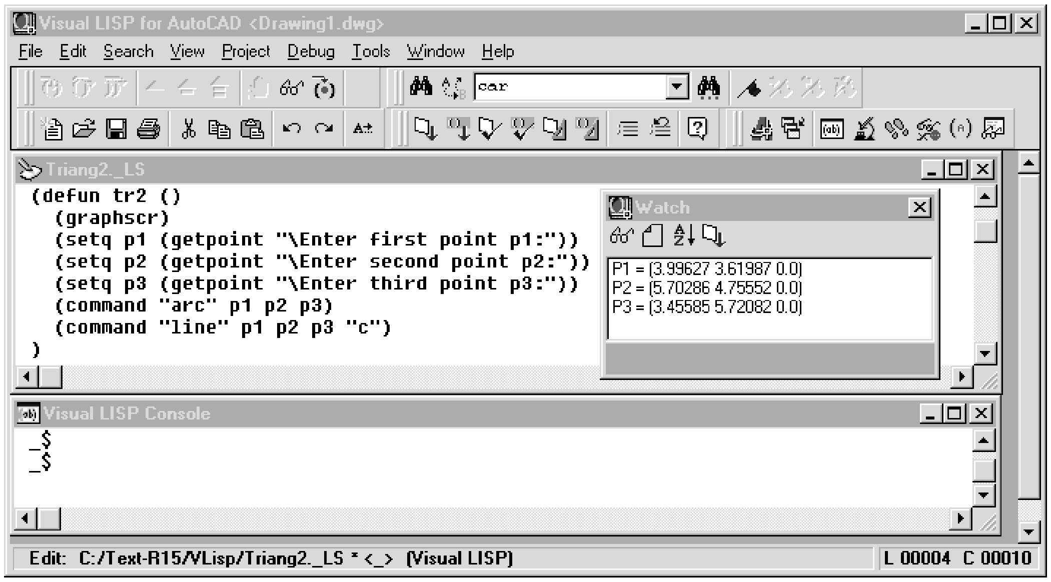
1. Open the file containing the program code in the Visual LISP Text Editor. You can open the file by selecting Open File from the File menu.

2. Load the program by selecting Load Text in Editor from the Tools menu or by selecting Load active edit window tool in the Tools toolbar.

3. Run the program by entering the name of the function (tr2) at the Console prompt (-$) in the Visual LISP Console. The program will perform the operations as defined in the program. In this program, it will draw a triangle and an arc between the user-specified points. Now, we want to find out the coordinates of the points p1, p2, and p3. To accomplish this we can use the Watch window tool in the View toolbar.

4. Highlight the variable **p1** and then choose the **Watch window** tool in the **View** toolbar. You can also select **Watch Window** in the **View** menu. Visual LISP displays the **Watch** window that lists the X, Y, and Z coordinates of point p1.

5. Position the cursor near (in front, middle, or end) the variable p2 and then choose the **Add Watch** tool in the **Watch** window. Select the **OK** button in the **Add Watch** window. The value of the selected variable is listed in the **Watch** window, Figure 35-27. Similarly you can trace the value of other variables in the program.


Visual LISP for AutoCAD <Drawing1.dwg>
File Edit Search View Project Debug Tools Window Help
car
Triang2._LS
Watch
Visual LISP Console

```
(defun tr2 ()
  (graphscr)
  (setq p1 (getpoint "\Enter first point p1:"))
  (setq p2 (getpoint "\Enter second point p2:"))
  (setq p3 (getpoint "\Enter third point p3:"))
  (command "arc" p1 p2 p3)
  (command "line" p1 p2 p3 "c")
)
```

```
P1 = (3.99627 3.61987 0.0)
P2 = (5.70286 4.75552 0.0)
P3 = (3.45585 5.72082 0.0)
```

```
_$
_$
```

Edit: C:/Text-R15/VLisp/Triang2._LS * <_> (Visual LISP) L 00004 C 00010

Figure 35-27 *Using Add Watch to trace variables in the program*

VISUAL LISP ERROR CODES AND MESSAGES

This section describes error messages that you may encounter at the console prompt while running the Visual LISP program and their corresponding meaning. The errors are arranged depending upon the frequency with which they occur during running the Visual LISP program. The common errors encountered have been listed first.

Malformed list
Premature ending of the list. Common cause is the mismatch between the number of opening and closing parentheses.

Malformed string
Premature ending of the string being read from a file.

Null function
A function was evaluated which has nil definition.

Too few arguments
The built-in function was given less arguments.

String too long
The system variable SETVAR has been given too long string.

Too many arguments
The built-in function has been given too many arguments.

Invalid argument
The type of argument is improper.

Invalid argument list
The function has been given an invalid argument list.

Incorrect number of arguments
It is expected to provide one argument within the quote function, but some other arguments have also been given.

Incorrect number of arguments to a function
Mismatch between the number of formal arguments specified and the number of arguments to the user-defined function.

Function cancelled
The user entered Ctrl+C or Esc in response to an input prompt.

Extra right paren
One or more extra right parentheses have been encountered.

Exceeded maximum string length
A string given to a function is greater than 132 characters.

Divide by zero
Division by zero is invalid.

Divide overflow
Division by a very small value has resulted in a invalid quotient.

Console break
The user entered Ctrl+C while a function is in progress.

Bad argument type
A function was passed an incorrect type of argument.

Bad association list
The list of the assoc function does not consist of key value lists.

Visual LISP stack overflow
The program has exceeded the Visual LISP stack storage space. This can be due to very large function arguments list.

Bad ENTMOD list
The argument passed to ENTMOD is not a proper entity data list.

Bad conversion code
An invalid space identifier was passed to the trans function.

Bad entmod list value
One of the sublists in the association list passed to entmod contains an improper value.

Bad function
The first element in the list is not a valid function name.

Bad list
An improper list was given to a function.

Bad ssget list
The argument passed to (ssget"x") is not a proper entity data.

Bad ssget list value
One of the sublists in the association list passed to (ssget"x") contains an improper value.

Bad ssget mode string
Caused when ssget was passed an invalid string in the mode argument.

Base point is required
The getcorner function was called without the required base point argument.

Can't evaluate expression
A decimal point was improperly placed, or some other expression was poorly formed.

Can't open file for input- LOAD failed
The file name in the load function could not be found, or the user does not have read access to the file.

Input aborted
An error or premature end of file condition has been detected, causing termination of the file input.

Invalid character
An expression contains an improper character.

Invalid dotted pair
You might get this error message if you begin a real number with a decimal point; you must use a leading zero in such a case.

Misplaced dot
A real number begins with a decimal point. You must use a leading zero in such a case.