

List Manipulation Functions

In this chapter, you will learn how to use the following AutoLISP functions to World Class standards:

1. **Manipulating Entities without the Command Function**
2. **Understanding the AutoCAD Database**
3. **Starting the Changetext Code**
4. **Learning to Use Ssget and the Print Function**
5. **Using the Getstring Function**
6. **Using the Sslength Function and Setting the Counter to Zero**
7. **Using a While Loop in a LISP Routine**
8. **Using the Ssname and Entget Functions in a LISP Routine**
9. **Using the Assoc Function in a LISP Routine**
10. **Using the Cons Function in a LISP Routine**
11. **Using the Subst Function in a LISP Routine**
12. **Using the Entmod Function in a LISP Routine**
13. **Finishing the Loop and Ending the Program**
14. **Loading and Running the Changetext Program**

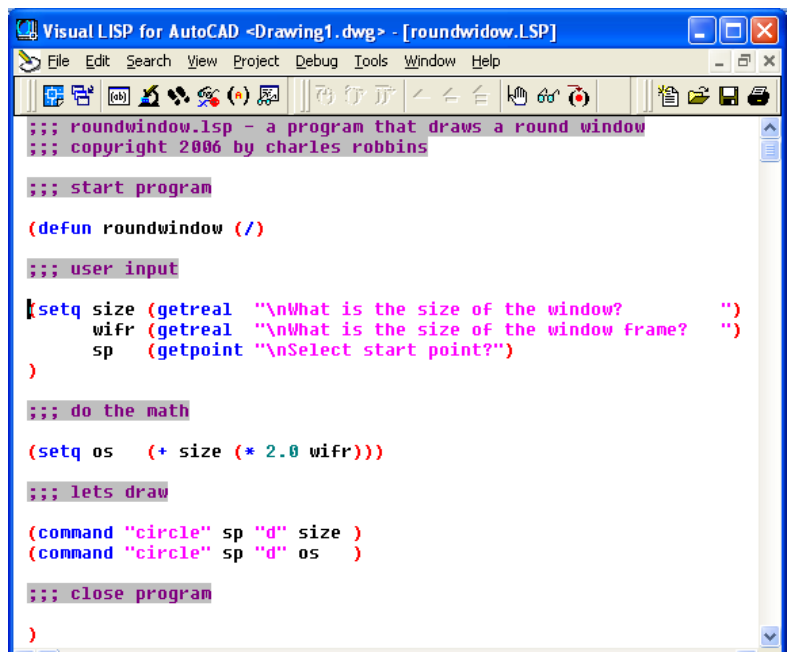
Manipulating Entities without the Command Function

In this chapter, we will explore two new concepts in the art of programming. The first being using subroutines which are programs that are common to more than one routine. The second being the manipulation of entities without using the command functions such as line, circle or text. And both of these ideas are important when creating drawings that require more freedom of design than what we receive when using the Construction code.

Let me explain. Did you notice that when you were using the Construction code, that although this powerful technique will allow you to draw any orthographic view of an existing design, the placement of the vertices are limited to the layout sketch that enabled you to quickly program the routine. The Construction code will always be important since the technique enables a new user to enter the field of Engineering Programming quickly and efficiently. What other way would one will learn the 100 important AutoLISP functions and be able to write their 50 initial programs, practicing to type the LISP syntax and troubleshooting their work. But there will be a time in your professional experience where the customer demands more design flexibility and the Construction code will be too constricting of a method to rely upon.

But let me be perfectly clear, flexibility does not mean disorganization. The subroutines that we will build to be part of a larger code have commonality and therefore we can use programming templates to both standardize and organize the routines so that any one in our engineering offices can both read and understand our work. When do we use the Construction code? Whenever we require a detail that is common to any drawing, we want to make a program that will construct those orthographic views or 3D components using the seven steps which are Start the Program, Drawing Setup, User Input, Do the Math, Point Assignments, Let's Draw and End the Program.

One of the techniques that we will use in future programs will appear at the beginning of a program. After the **defun** statement, and before the program name, we would place a **c:** which made the program name a new command function after loading the program into AutoCAD. When we write subroutines which can be run in any master routine, we will leave the **c:** out of the code. In Figure 8.1, the **c:** is left out of the program. The program is intentionally written as a subroutine.



```
Visual LISP for AutoCAD <Drawing1.dwg> - [roundwindow.LSP]
File Edit Search View Project Debug Tools Window Help
::: roundwindow.lsp - a program that draws a round window
::: copyright 2006 by charles robbins
::: start program
(defun roundwindow (/)
::: user input
[setq size (getreal "\nWhat is the size of the window? ")
wifr (getreal "\nWhat is the size of the window frame? ")
sp (getpoint "\nSelect start point?")
)
::: do the math
(setq os (+ size (* 2.0 wifr)))
::: lets draw
(command "circle" sp "d" size )
(command "circle" sp "d" os )
::: close program
)
```

Figure 8.1 – An AutoLISP Subroutine

This code can be pasted at the end of the any larger program and when we want the subroutine to execute, we will just write the following in our coded master syntax.

(roundwindow)

At that point in the master program, the loaded **roundwindow** subroutine will execute.

This type of programming gives the designer flexibility in the master program. If the **roundwindow** subroutine was part of many different window options in a door program, then the program user can select many types of door enhancements with in the same master door program. Subroutines enable us to maintain smaller master programs and we have the capability of adding or removing subroutines with simplicity.

The other equally important technique which we will explore in this chapter is the use of functions to manipulate the entities that already exist in the drawing and to read information from existing lines, circles and text. The way that we do this is through obtaining a selection set, then manipulating the list information contained in the AutoCAD database within the drawing. This means that we will have to learn new functions and discover the techniques in which to use them.

Understanding the AutoCAD Database

Before we continue to learn write another expression, we need to take the time to understand how AutoCAD controls the drawing database. We can do this by creating a single line of text on the graphical display. So go ahead and place the word "Text" on the graphical display as shown in Figure 8.2.



Figure 8.2 – A Single Line of Text in AutoCAD

At the command line, type the expression listed below. The **ssget** function by itself will allow the user to pick objects with a mouse.

(setq ss1 (ssget))

The AutoCAD program will return with a comment on the command line such as **<Selection set: 4>**. Anytime we capture a data in a selection set we will notice the number changing. Now type the following expression on the command line.

(setq entityname (ssname ss1 0))

We are using **entityname** as the variable name. The AutoLISP function **ssname** gives the name of an entity of the first item in the selection set **ss1**. The command line will come back with **<Entity name: 4006ae98>**. An AutoCAD name has eight digits that can have 0 through 9 or A to Z in each digit. To retrieve the first entity in the selection set using **ssname**, we follow the function with an integer. AutoCAD starts counting with the number **0**, so we have to remember to always start our counts in a program at zero and not **1** when using the **ssname** function. Now type the following expression on the command line.

(setq entitylist (entget entityname))

We are using **entitylist** as the variable name. The AutoLISP function **entget** will return the entire database list that describes the line entity on the graphical display which is the first item in the selection set **ss1**. The command line will come back with following data. Your data will be different since your text is in a different location.

```
((-1 . <Entity name: 4006ae98>) (0 . "MTEXT") (330 . <Entity name: 4006acf8>)
(5 . "5B") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 .
"AcDbMText") (10 12.6265 30.8991 0.0) (40 . 0.125) (41 . 38.6837) (71 . 1) (72
. 5) (1 . "Text") (7 . "mechanical") (210 0.0 0.0 1.0) (11 1.0 0.0 0.0) (42 .
0.334243) (43 . 0.127046) (50 . 0.0) (73 . 1) (44 . 1.0))
```

For many AutoCAD expert operators this is a time of great discovery. The AutoCAD drawing file has header information which contains information regarding how to set up the environment for the actual drawing and then comes a complete collection of records which contain the data for every drawing entity which you see in the drawing. The AutoCAD file is actually a three dimensional database, where each coordinate in the list of data contains an X, Y and Z measurement.

Every data field inside of each set of parenthesis begins with a group code. There is a table of group codes in Appendix Z in this textbook. Common group codes that we use to extract data all the time are in the table below.

0	Entity Type
1	Text
2	Block Name
8	Layer Name
10	Start or Insertion Point
11	End Point

In this chapter, we will learn how to change a text entity using the list manipulation functions.

Starting the Changetext Code

The program we are going to write is called Changetext. We begin the coded routine with multiple comments stating the program name, a brief description stating the program's functionality and the copyright statement.

After the initial comments, the **alert** function is written so that an AutoCAD Message Box is launched containing the program name, copyright and execution keyword for the use to see. Next, the Start Program comment is added and is followed by the **defun** statement

(defun c:ct (/)

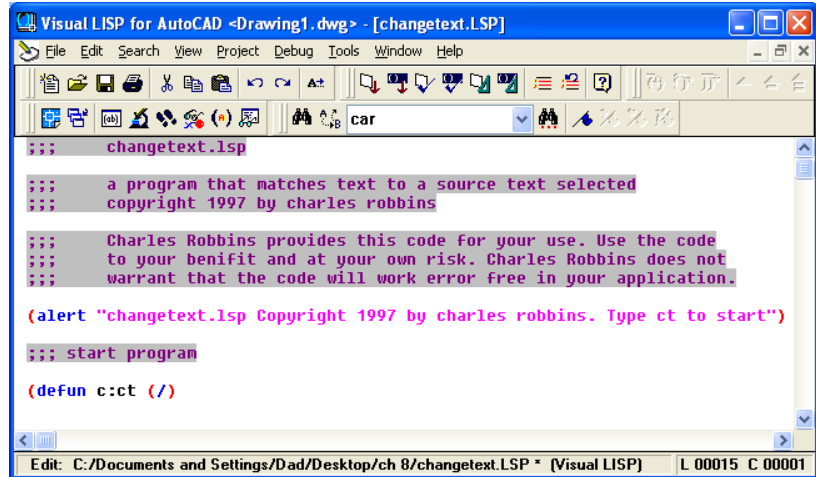


Figure 8.3 – Starting the Changetext Program

The letters **ct** represents the word **changetext** so the program user does not have to type the entire text string when beginning the program.

Learning to Use Ssget and the Print Function

In prior exercises, we have used the **ssget** function using the last drawn and the filter options. In both these cases, the user did not have to select any entities with their mouse, since **ssget "L"** selects the last drawn object and the **ssget "X" '((8 . "ball"))** will use a filter to select all the entities on layer called **"ball"**. Sometimes, we will want the user to make decision in a program. There are many advanced programs, where we allowed the code to select the entities automatically and then each entity was presented to the viewer, one by one to decide whether they want to change the object or let the item remain as is. We always called this technique Presentation code, and we used the technique as a checking method where we wish to have 100% confirmation of all changes. In this program however, we will just allow the user to select the text they wish to modify by using the straightforward **ssget** function.

When using the **ssget** function, the computer aided design user is able to select single or multiple entities with their mouse. But we need to place a print statement into the program before placing the **ssget** function. This will allow the statement **"Select the text to change"** to appear on the AutoCAD command line.

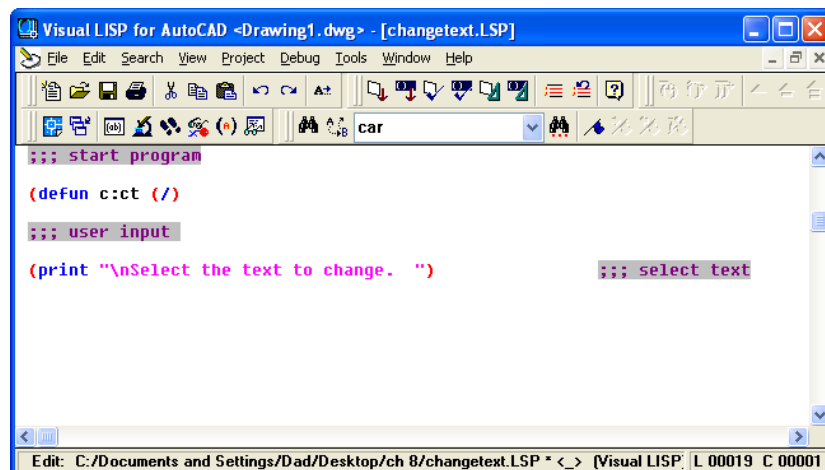


Figure 8.4 – Inserting a Print Statement

After the print statement, we will make a selection set using **ssget** and store the information collected in the variable **ss1**. When we use the basic **ssget** tool, the user can select as many objects as they wish. If they accidentally select an entity they do not desire, they can still hold down the shift key, reselect the object to remove the entity from this selection set.

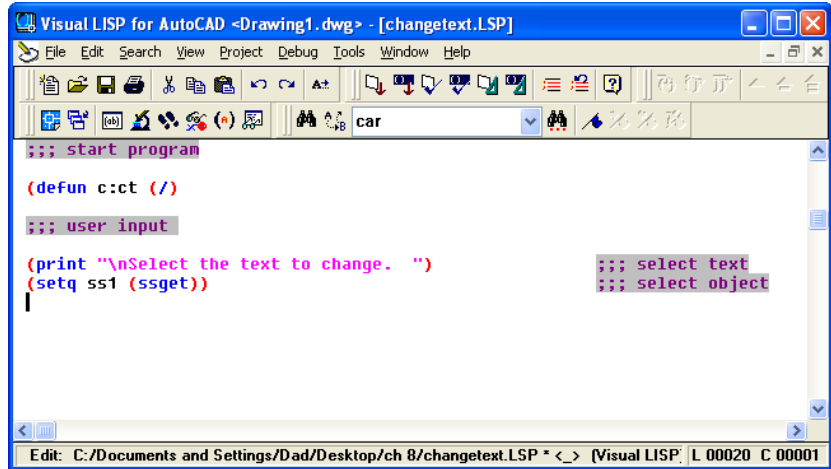


Figure 8.5 – Using the Ssget Function

Practice typing the following examples of the **print** and **ssget** function at the command line of AutoCAD.

Function	Name	Description
print	Print Function	Will print a text string to the command line
Example		
Print "Select text" at the command line	(print "Select text")	Answer: "Select text"
Print "Program Done" at the command line. If this is the last line of the routine, use the (princ) function	(print "Program Done")(princ)	Answer: "Program Done"
Function	Name	Description
ssget	Obtain a Selection Set Using a Filter	Will allow the user to select entities using the mouse
Examples		
Pick an object	(setq ss1 (ssget))	Selection Set: 4

Using the Getstring Function

The next part of the user input section of the routine is to ask the user to “type the new text” which will be stored in the variable named **textstring**. We need to use the **getstring** function to obtain a text string and be sure to place the capital **T** behind the function name so that spaces can be added by the user.

```

Visual LISP for AutoCAD <Drawing1.dwg> - [changetext.LSP]
File Edit Search View Project Debug Tools Window Help
car
;;; start program
(defun c:ct (/)
  ;;: user input
  (print "\nSelect the text to change. ")
  (setq ss1 (ssget))
  (setq textstring (getstring T "\nType the new text. "))
  ;;: select text
  ;;: select object
  ;;: ask for new text
  
```

Figure 8.6 – Asking Questions Using the Getstring Function

Although we are using the **getstring** function, the user can enter any type of text at the keyboard, such as capital letters, lowercase letters, numbers and special characters. All text in the AutoCAD drawing or external files must be input as strings, so in this coded routine the **getstring** function with a capital **T** is most desirable.

Practice typing the following examples of the **getstring** function at the command line of AutoCAD.

Function	Name	Description
getstring	Get a Text String	Allows the user to obtain a text string by allowing the user to type at the keyboard
Examples		
Looking for a single word response	(setq matl (getstring "\nWhat is the material?"))	Answer: What is the material? Then type: Aluminum “Aluminum”
What happens when two or more words are typed	(setq matl (getstring "\nWhat is the material?"))	Answer: What is the material? Then type: Stainless Steel “Stainless”
Fix the space bar problem with a T after getstring	(setq matl (getstring T "\nWhat is the material?"))	Answer: What is the material? Then type: Stainless Steel “Stainless Steel”

Using the Sslength Function and Setting the Counter to Zero

After selecting the entities with the **ssget** function, we do not want to keep track of how many objects that were picked with a mouse, so the **sslength** tool is the apparatus which we choose to count the selections in the set.

The **sslength** function as well as the rest of the list manipulation tools is very easy to learn since this syntax of the written code is very simple in application. When using The **sslength** function, an integer or whole number is returned and we will store that figure in the variable named **qty**.

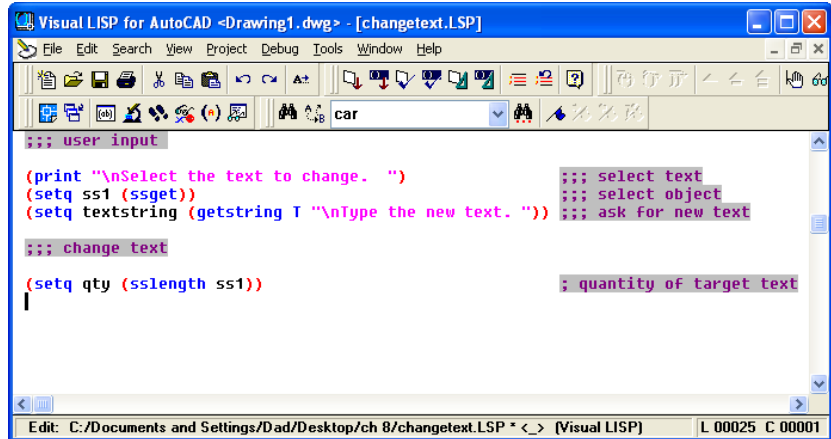


Figure 8.7 – Obtaining the Size of the Selection Set

Type the following code is the Changetext program.

```
(setq qty (sslength ss1))
```

The **sslength** function is important when we use a while loop to sort through a selection set automatically, since this represents the number of loops the program will run.

Practice typing the following examples of the **sslength** function at the command line of AutoCAD.

Function	Name	Description
sslength	Selection Set Length	Returns the number of entities in a selection set
Examples		
Select one object with the code (setq ss1 (ssget))	(setq qty1 (sslength ss1))	Answers: 1
Select three objects with the code (setq ss2 (ssget))	(setq qty2 (sslength ss2))	Answers: 3
Select ten objects with the code (setq ss2 (ssget))	(setq qty3 (sslength ss3))	Answers: 10

Now we will set the variable named counter to zero by typing

(setq counter 0)

Normally each time we begin a while loop we will set a counter to zero. Another technique we will use in other programs is to ask the user if they wish to continue.

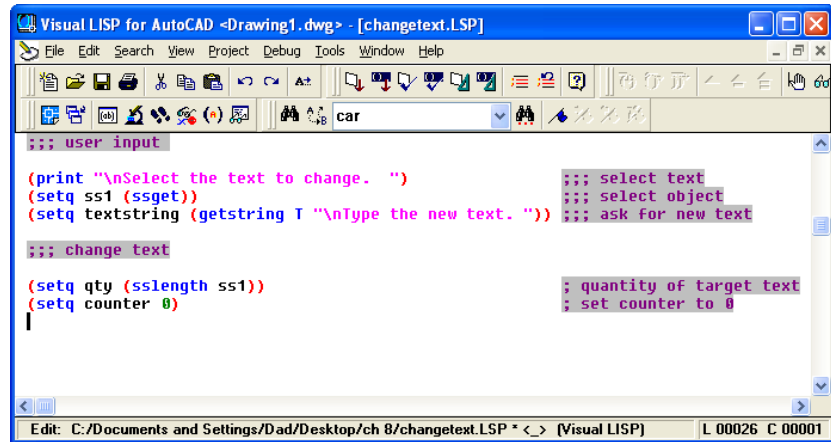


Figure 8.8 – Setting the Counter to Zero

Using a While Loop in a LISP Routine

The next expression is the code is the entrance to the while loop.

(while (< counter qty)

A condition statement trails the **while** function where we ask whether the number contained in variable named **counter** is less than the number contained in variable named **qty**.

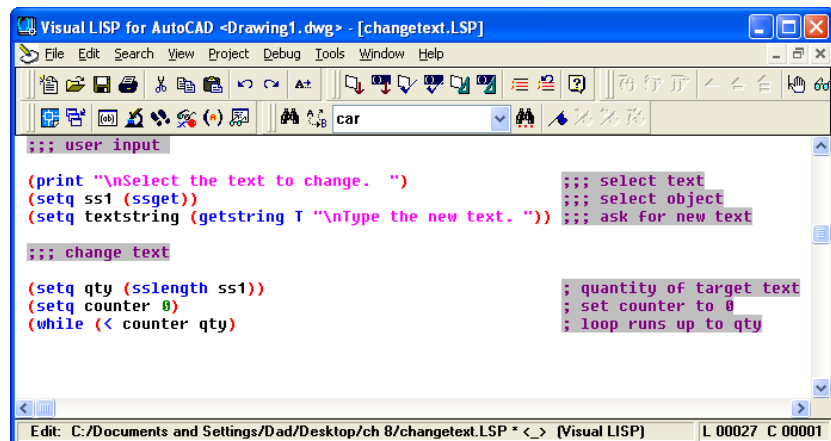


Figure 8.9 – Starting the While Loop

A while loop will run anytime the condition set returns a true response in the code. If the user selects one text object when the program is running then the integer value for **qty** will be 1. The first time into the loop the condition is (< 0 1) which is true so all of the expression inside the while loop will be read in the program. The second time into the loop the condition is (< 1 1) which is false and so the while loop will not execute and the next expression in the code will be read.

Notice that the expression below does not contain an ending parenthesis for the **while** function.

(while (< counter qty)

The ending parenthesis comes at the end of the list of expression inside of the while loop.

Practice typing the following examples of the **while** function at the command line of AutoCAD.

Function	Name	Description
While	While Loop	Will automatically select entities using a filter such as layer name, entity type like circle.
Examples		
Using a counter	<code>(while (< counter 5))</code>	Will continue 5 times
Using a question	<code>(while (= ball "y"))</code>	Will continue as long as ball equals "yes"

Using the Ssname and Entget Functions in a LISP Routine

The AutoLISP function **ssname** gives the name of an entity in a selection set. In our program example, **ssname ss1 counter** the entities are stored in **ss1**. The way we grab each object in the set is that with each pass in the while loop the counter will progress by one. AutoCAD starts counting with the number **0**, so we set the counter in the program to zero and not one when using the **ssname** function.

Then to open the list, we place the **entget** function in front of the **(ssname ss1 counter)** expression. Now that the data before the entity is written to a list, we will store the information into the variable named **entitylist**. This type of compound expression keeps our code simple and neat, and we only use one variable name.

```

Visual LISP for AutoCAD <Drawing1.dwg> - [changetext.LSP]
File Edit Search View Project Debug Tools Window Help
car
::: change text
(setq qty (sslength ss1))           ; quantity of target text
(setq counter 0)                   ; set counter to 0
(while (< counter qty)            ; loop runs up to qty
  (setq entitylist (entget (ssname ss1 counter))) ; acquires text database
)

```

Figure 8.10 – Using the Entget and Ssname Functions

Now type the following expression on the command line.

`(setq entitylist (entget (ssname ss1 counter)))`

Practice typing the following examples of the **ssname** and **entget** functions at the command line of AutoCAD.

Function	Name	Description
ssname	Selection Set Name	Returns the AutoCAD entity number of 8 characters
Examples		
After using the ssname function	<code>(setq entityname (ssname ss1 0))</code>	Answers: <Entity name: 4006ad98>

Function	Name	Description
entget	Get the Entity List	Returns the entity list of a single AutoCAD entity
Examples		
Pick a text string	<code>(setq entitylist (entget entityname))</code>	
Answer:		
<pre>((-1 . <Entity name: 4006ae98>) (0 . "MTEXT") (330 . <Entity name: 4006acf8>) (5 . "5B") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbMText") (10 12.6265 30.8991 0.0) (40 . 0.125) (41 . 38.6837) (71 . 1) (72 . 5) (1 . "Text") (7 . "mechanical") (210 0.0 0.0 1.0) (11 1.0 0.0 0.0) (42 . 0.334243) (43 . 0.127046) (50 . 0.0) (73 . 1) (44 . 1.0))</pre>		

Using the Assoc Function in a LISP Routine

With the entire data list containing all the information pertaining to a single entity in the variable named entitylist, then to pick out a single attribute such as the text string, we ignore other data such as layer, insertion point and text height. We do this with the **assoc** function, adding the group code of the attribute we desire.

```

::: change text
(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  ; quantity of target text
  ; set counter to 0
  ; loop runs up to qty
  ; acquires text database
  ; acquires old text
)

```

Figure 8.11 – Pulling Out the Old Text String with Assoc

Now type the following expression on the command line.

`(setq oldtext (assoc 1 entitylist))`

The named variable **oldtext** will now contain a list surrounded by parentheses beginning with the group code number one, separated by a period and finishing with the text string inside of quotes.

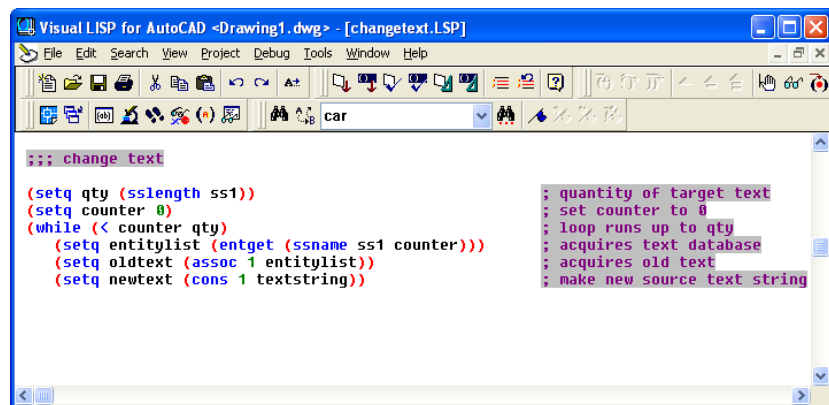
(1 . "Text")

Practice typing the following examples of the **assoc** function at the command line of AutoCAD.

Function	Name	Description
assoc	Association	Looks in the data list for an entity and returns the individual data string matching the group code
Examples		
Searches the entire data list for “text” and returns the list with group code 1, text string	(setq oldtext (assoc 1 entitylist))	Answer: (1 . "Text")
Searches the entire data list for “text” and returns the list with group code 10, insertion point	(setq ip (assoc 10 entitylist))	Answer: (10 12.6265 30.8991 0.0)
Searches the entire data list for “text” and returns the list with group code 12. Returns nil if there is none found.	(setq ip (assoc 12 entitylist))	Answer: nil

Using the Cons Function in a LISP Routine

The next function we will learn is for list construction using the **cons** tool. We have used the **list** function to make a new point, but the dot is not placed between the group code and the data. The **cons** function is the means to build a new data field that is correctly formatted.



```
Visual LISP for AutoCAD -Drawing1.dwg> - [changetext.LSP]
File Edit Search View Project Debug Tools Window Help
car
;;; change text
(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  (setq newtext (cons 1 textstring)))
; quantity of target text
; set counter to 0
; loop runs up to qty
; acquires text database
; acquires old text
; make new source text string
```

Figure 8.12 – Make a New Text List With Cons

When writing a construction expression begin with an open parenthesis, then the function name, **cons**, the group code number, which in this case is **1** for text. Next, the text string will follow which is stored in the variable named **textstring**. Lastly, end the cons expression with a closed parenthesis. To assign the new list to a variable use the **setq** function as shown below.

(setq newtext (cons 1 textstring))

Practice typing the following examples of the **cons** function at the command line of AutoCAD.

Function	Name	Description
cons	Construct	Creates a new list with the group code and data
Examples		
Create a new list with a group code and text	<code>(setq newtext (cons 1 "hello"))</code>	Answer: (1 . "hello")
Create a new list with a group code and text inside variable textstring	<code>(setq textstring "new")</code> <code>(setq newtext (cons 1 textstring))</code>	Answer: (1 . "new")
Create a new list with a group code and coordinate inside variable pt	<code>(setq pt (list 2.0 3.5 0.0))</code> <code>(setq newtext (cons 10 pt))</code>	Answer: (10 2.0 3.5 0.0)

Using the Subst Function in a LISP Routine

To make a revised data record for the new text in the AutoCAD, we will learn how to use the substitution tool, **subst**. This is a very straightforward function where we state the new text list is changing out the old text list in the data record. Then we save the data record to the variable named **entitylist**.

```

Visual LISP for AutoCAD -Drawing1.dwg> -[changetext.LSP]
File Edit Search View Project Debug Tools Window Help
car
::: change text
(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  (setq newtext (cons 1 textstring))
  (setq entitylist (subst newtext oldtext entitylist)))
; quantity of target text
; set counter to 0
; loop runs up to qty
; acquires text database
; acquires old text
; make new source text string
; substitute old with new text
Edit: C:/Documents and Settings/Dad/Desktop/ch 8/changetext.LSP * <> [Visual LISP] L 00031 C 00001

```

Figure 8.13 – Use the Subst Function

When writing a substitution expression begin with an open parenthesis, then the function name, **subst**, the variable containing the new list, which in this case is **newtext**. Next, the variable named **oldtext** will follow, and then the variable representing the entire data record, **entitylist**. Lastly, end the cons expression with a closed parenthesis. To assign the new list to a variable use the **setq** function as shown below.

`(setq entitylist (subst newtext oldtext entitylist))`

Practice typing the following examples of the **subst** function at the command line of AutoCAD.

Function	Name	Description
Subst	Substitute	Replaces an old list with a new list in the AutoCAD entity data list

Examples

Replaces the old list with the new list in the entity data list

(setq entitylist (subst newtext oldtext entitylist))

Answer:

```
((-1 . <Entity name: 7ef60e98>) (0 . "MTEXT") (330 . <Entity name: 7ef60cf8>)
(5 . "8B") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 .
"AcDbMText") (10 8.64868 13.9756 0.0) (40 . 0.2) (41 . 6.75825) (71 . 1) (72 .
5) (10 2.0 3.5 0.0) (7 . "Standard") (210 0.0 0.0 1.0) (11 1.0 0.0 0.0) (42 .
0.733333) (43 . 0.2) (50 . 0.0) (73 . 1) (44 . 1.0))
```

Using the Entmod Function in a LISP Routine

When we want to update the current drawing database, the entity modification function, **entmod** is used. Type the following expression and the drawing database is updated.

(entmod entitylist)

```
Visual LISP for AutoCAD -Drawing1.dwg - [changetext.LSP]
File Edit Search View Project Debug Tools Window Help
car
::: change text
(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  (setq newtext (cons 1 textstring))
  (setq entitylist (subst newtext oldtext entitylist))
  (entmod entitylist)
  ; quantity of target text
  ; set counter to 0
  ; loop runs up to qty
  ; acquires text database
  ; acquires old text
  ; make new source text string
  ; substitute old with new text
  ; drawing database updated
)
Open an existing file L 00032 C 00001
```

Figure 8.14 – Use the Entmod Function

Practice typing the following examples of the **entmod** function at the command line of AutoCAD.

Function	Name	Description
entmod	Entity Modification	Updates the drawing database with the new entity list
Examples		
Updates the drawing database with the new entity list	(entmod entitylist)	
Answer:		
<pre>((-1 . <Entity name: 7ef60e98>) (0 . "MTEXT") (330 . <Entity name: 7ef60cf8>) (5 . "8B") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbMText") (10 8.64868 13.9756 0.0) (40 . 0.2) (41 . 6.75825) (71 . 1) (72 . 5) (10 2.0 3.5 0.0) (7 . "Standard") (210 0.0 0.0 1.0) (11 1.0 0.0 0.0) (42 . 0.733333) (43 . 0.2) (50 . 0.0) (73 . 1) (44 . 1.0))</pre>		

Finishing the Loop and Ending the Program

The next expression we will place in the code will add one to the variable named **counter**. We will use the **1+** function to add 1 to the 0 in the first loop of the running program, so the variable named **counter** will now be 1. Then we save the number to the same variable named **counter**.

```

;;; change text
(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  (setq newtext (cons 1 textstring))
  (setq entitylist (subst newtext oldtext entitylist))
  (entmod entitylist)
  (setq counter (1+ counter))
)

```

Figure 8.15 – Add one to the Counter with 1+

Type the following expression in the while loop.

```
(setq counter (1+ counter))
```

Now we can add the final parenthesis to the end of the while loop under the last expression.

To end the program, we will need to place a parenthesis at the end of the code to close the **defun c:ct** function. Type the following code.

```

;;; end the program

(gc)
(princ)
)

```

```

(setq qty (sslength ss1))
(setq counter 0)
(while (< counter qty)
  (setq entitylist (entget (ssname ss1 counter)))
  (setq oldtext (assoc 1 entitylist))
  (setq newtext (cons 1 textstring))
  (setq entitylist (subst newtext oldtext entitylist))
  (entmod entitylist)
  (setq counter (1+ counter))
)

;;; end of program

(gc)

```

Figure 8.16 – Erase Selection Sets with Garbage Collection

But before the very last parenthesis, the garbage collection function **gc** is entered, so the selection sets stored in the memory are erased. If we continue to accumulate selection sets indefinitely then we will run out of storage space.

```

(setq entitylist (entget (ssname ss1 counter)))
(setq oldtext (assoc 1 entitylist))
(setq newtext (cons 1 textstring))
(setq entitylist (subst newtext oldtext entitylist))
(entmod entitylist)
(setq counter (1+ counter))
)

;;; end of program

(gc)
(princ)
)

```

Figure 8.17 – End the Program

Then we write the **princ** function to cut out any erroneous data from printing on the command line when the program is complete.

Loading and Running the Changetext Program

Now that the program is finished, we need to double check our typing with the text in this manual and then save our program to our folder named “Visual AutoLISP Programs”.

Make sure the Look in list box is displaying the Visual LISP Programs folder and then select the program “Changetext” and press the Load button. At the bottom – left corner of the Load / Unload Applications window you will see a small text display that was blank initially but now displays the text as shown in Figure 8.18,

“Changetext.LSP successfully loaded”

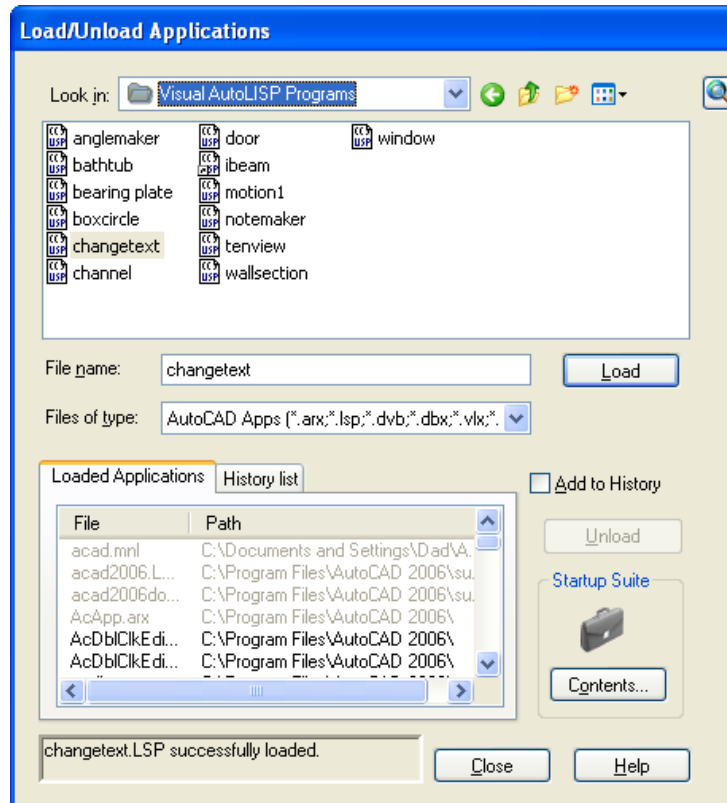


Figure 8.18 – Load the Changetext Program

After noting that the program is loaded, press the Close button and now when you are in the AutoCAD program, an AutoCAD message appears in the middle of the graphics display stating: “Changetext.lsp – copyright 1997 by charles robbins. Type ct to start” Press the OK button if you agree with the message.



Figure 8.19 – The AutoCAD Message

In the AutoCAD file, place some text on the graphical display as shown in Figure 8.18 using the Mtext, Dtext or Text command. The Changetext LISP program will work with text strings made with any AutoCAD command.

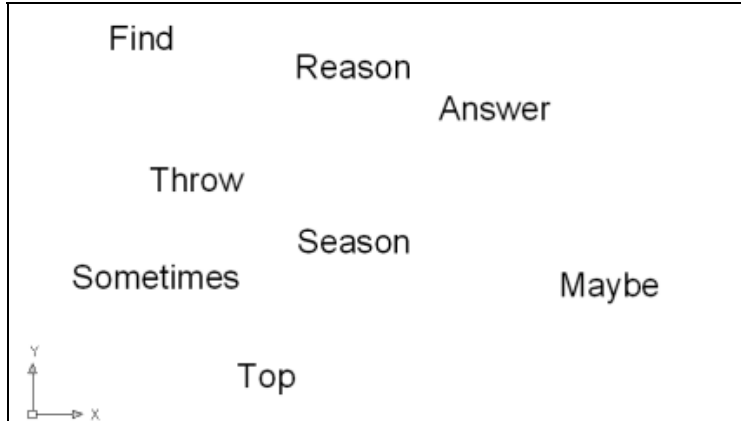


Figure 8.20 – Some Text on the AutoCAD Display

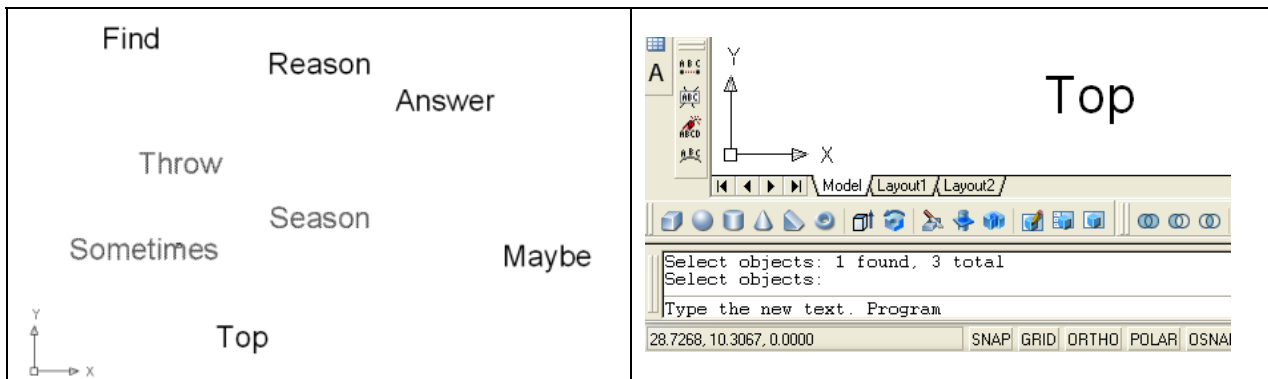


Figure 8.21 – Select Any of the Text Strings Figure 8.22 – Type the New Text

Type **ct** to start the program and select three separate text strings. Enter to exit the selection set being made by the **ssget** function.

The three text strings will change to the new text. Enter or type **ct** to repeat the program to change more text. In the Appendix following this chapter, write modifications of this code to match text, or count items. There are endless types of small subroutines we can write using the knowledge we have gained in this chapter.

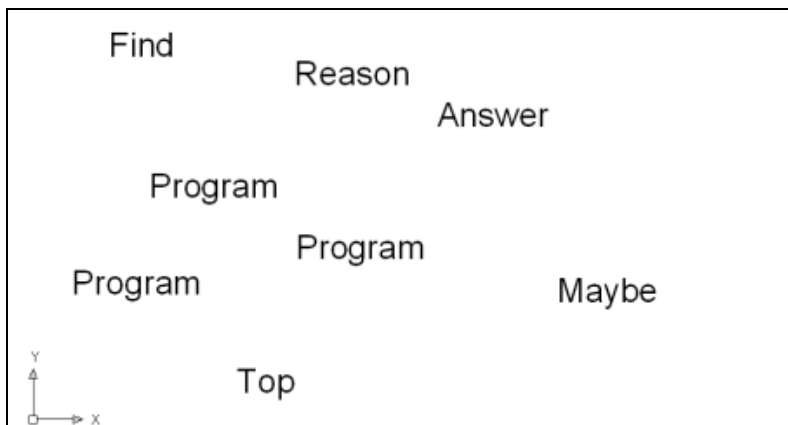


Figure 8.23 – Type in the Replacement Text

Written below is the entire Changetext.LSP code for your benefit.

```
;;; changetext.lsp

;;; a program that matches text to a source text selected
;;; copyright 1997 by charles robbins

;;; Charles Robbins provides this code for your use. Use the code
;;; to your benefit and at your own risk. Charles Robbins does not
;;; warrant that the code will work error free in your application.

(alert "changetext.lsp -copyright 1997 by charles robbins. Type ct to start")

;;; start program

(defun c:ct (/)

  ;;; select text

  (print "Select the text to change. ")
  (setq ss1 (ssget))

  ;;; ask questions

  (setq textstring (getstring T "\nType the new text. "))

  ;;; change text
  (setq qty (sslength ss1)) ; quantity of target text
  (setq counter 0) ; set counter to 0
  (while (< counter qty) ; loop runs up to qty
    (setq entitylist (entget (ssname ss1 counter))) ; acquires text database
    (setq oldtext (assoc 1 entitylist)) ; acquires old text
    (setq newtext (cons 1 textstring)) ; make new source text string
    (setq entitylist (subst newtext oldtext entitylist)) ; substitutes old with new text
    (entmod entitylist) ; drawing database updated
    (setq counter (1+ counter)) ; add one to counter
  )

  ;;; end of program

  (gc) ; removes selection sets from memory
  (princ)
)
```