AutoLISP® Strategies for CAD Management

Robert Green

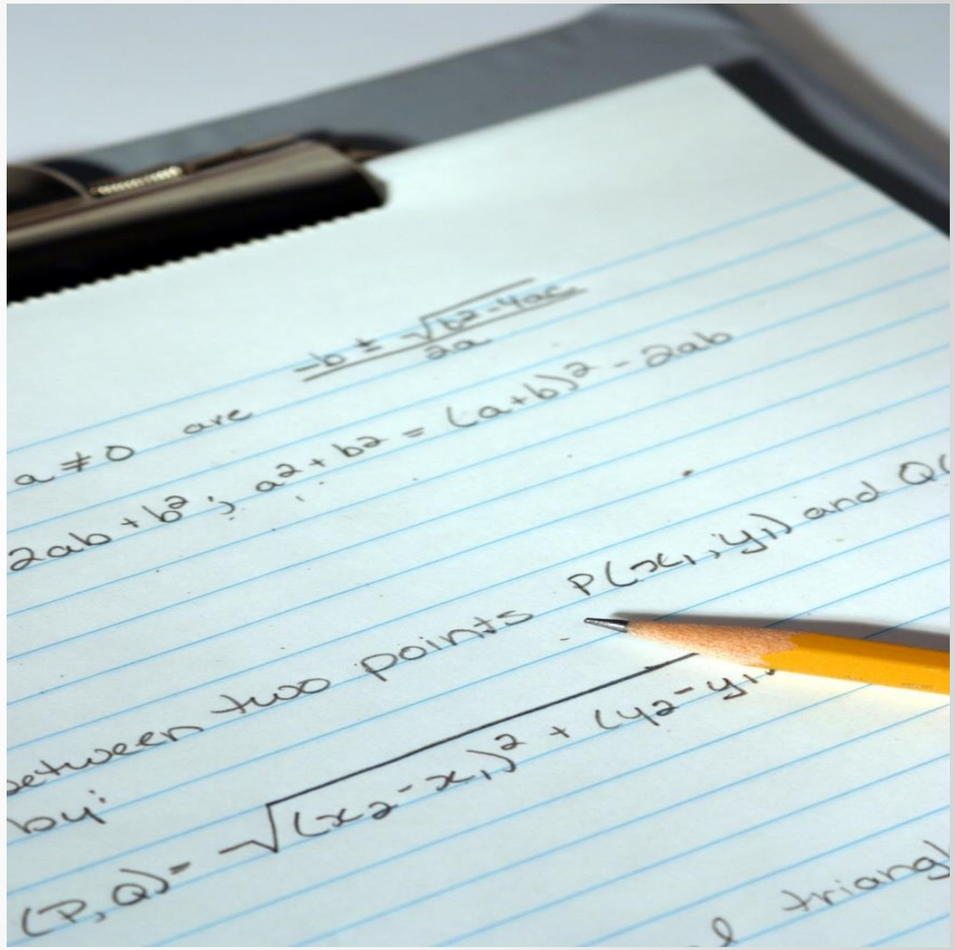
Robert Green Consulting

My AutoLISP Philosophy

You can manage AutoCAD based tools with this language so it behooves you to know it.

Some fundamentals

- **LISP – LISt Processor**
- **A widely used language**
- **Everything is in lists**



Should we still use AutoLISP?

- Yes!
- AutoLISP is still very valid!
- A HUGE base of LISP code exists
- LISP shareware is abundant
- LISP does many things very compared to VB
- It is great for controlling AutoCAD configurations
- You don’t have to know much to get great results …

Key Files and Variables

Defining some standard “rules” for the purposes of our class.

Key Files and Variables

Defining some standard “rules” for the purposes of our class.

Key files and why they matter …

- LSP
- Load in order
- What to mess with
- What not to!

All those wacky file names …

- ACAD20XX.LSP (system file – XX is version)
- ACAD.LSP (This is your file)
- ACAD20XXDOC.LSP (system file - XX is version)
- ACADDOC.LSP (This is your file)
- CUINAME.MNL (loads with CUI)
- So what does it all mean?

What do they do, where they live …

- They load on startup of AutoCAD
- They load in a certain order (listed on previous slide)
- Some have code in them and some don’t (ACADDOC.LSP and ACAD.LSP don’t as an example)
- They reside in the SUPPORT folder …

So what should I do …

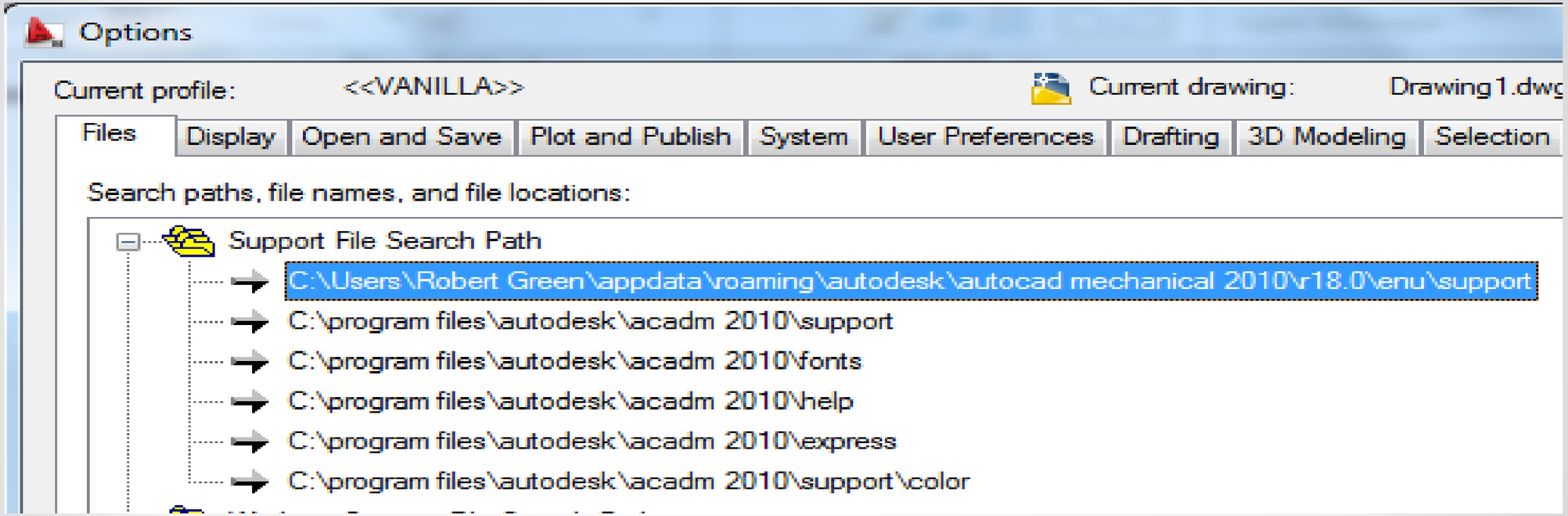
- Use ACADDOC.LSP to load your code
- Put in in the SUPPORT folder and start hacking
- If you mess up too bad, just delete!
- Later, after debugging, deploy to the network.

Make sure it works …

- Create ACADDOC.LSP
- Load from network?
- Verify operation

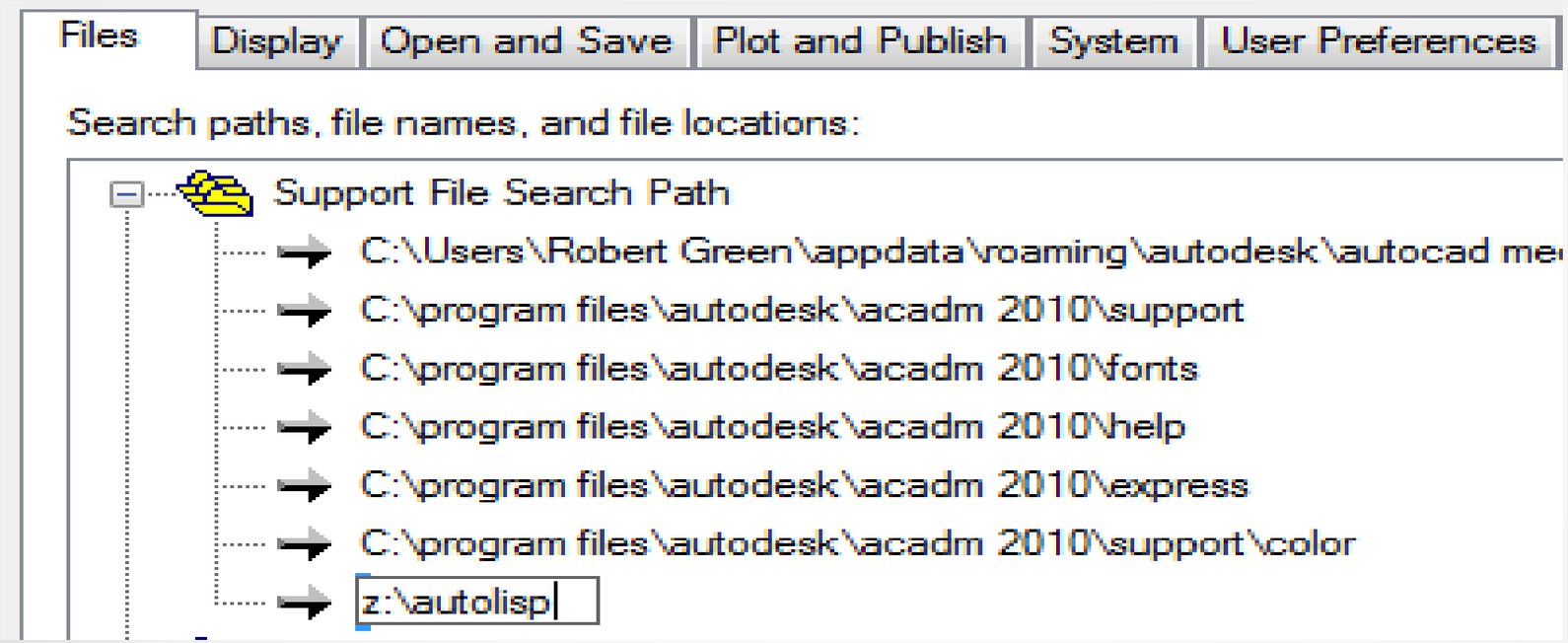
Find the support folder …

- **Use OPTIONS to find the folders …**



Add a network folder?

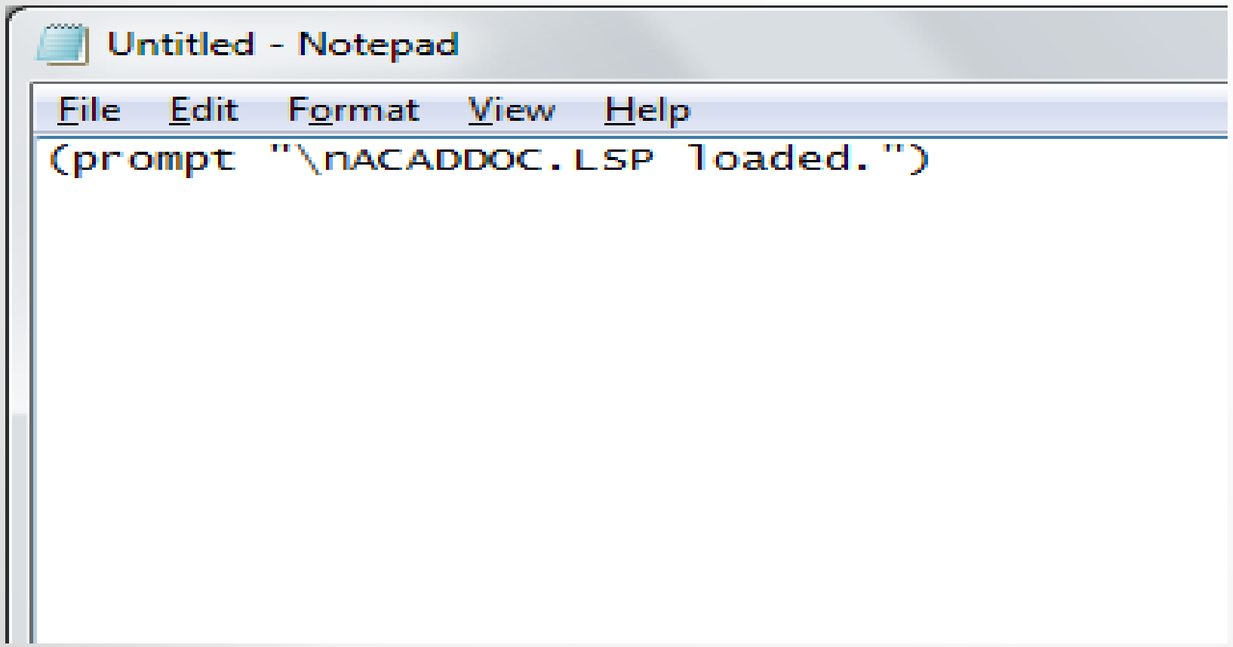
- **Use OPTIONS to add it …**



- **Push the network path to the top of the list**

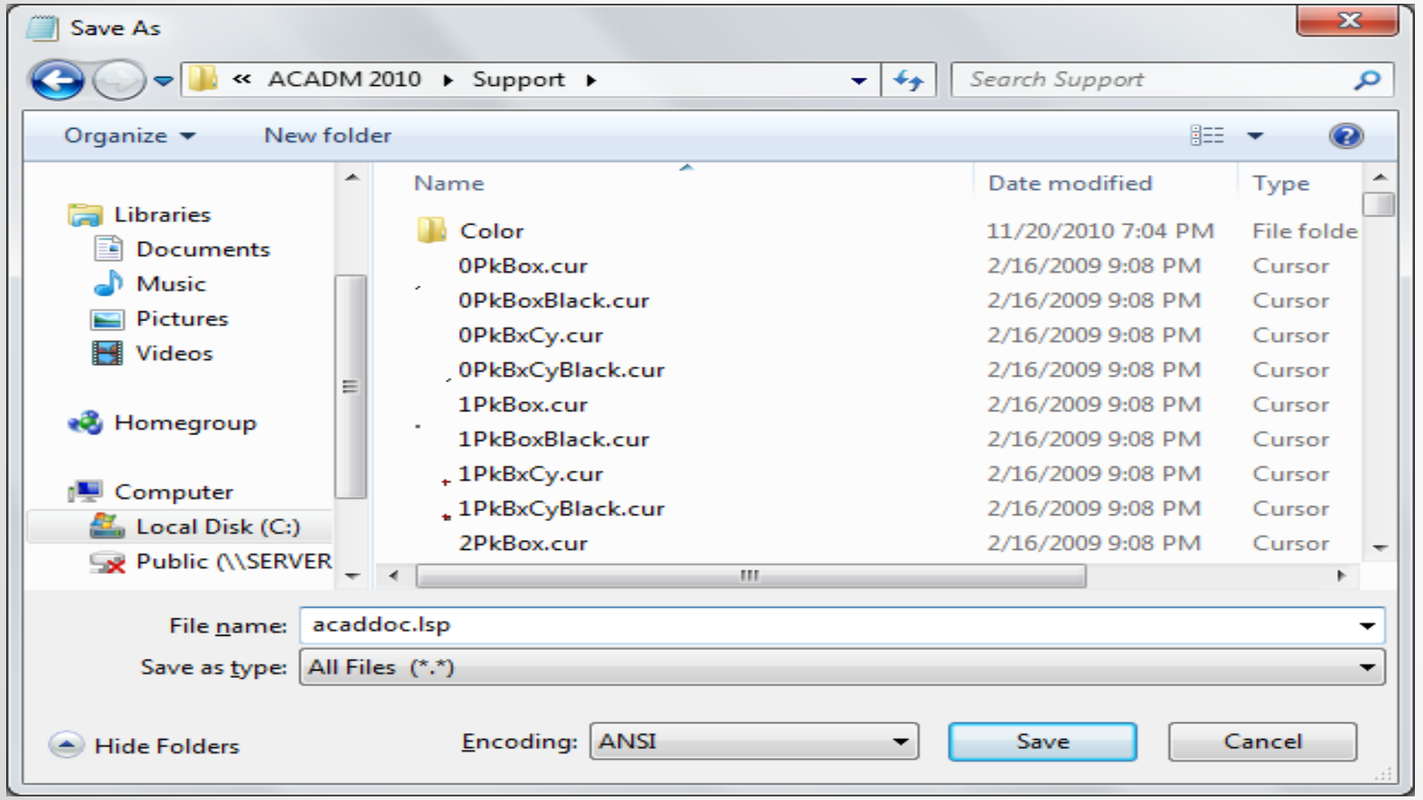
Create the file …

- **Use Notepad – not Word!**
- **You may also use Visual LISP VLIDE if you prefer**
- **Use (prompt "\nACADDOC.LSP loaded.") as text**



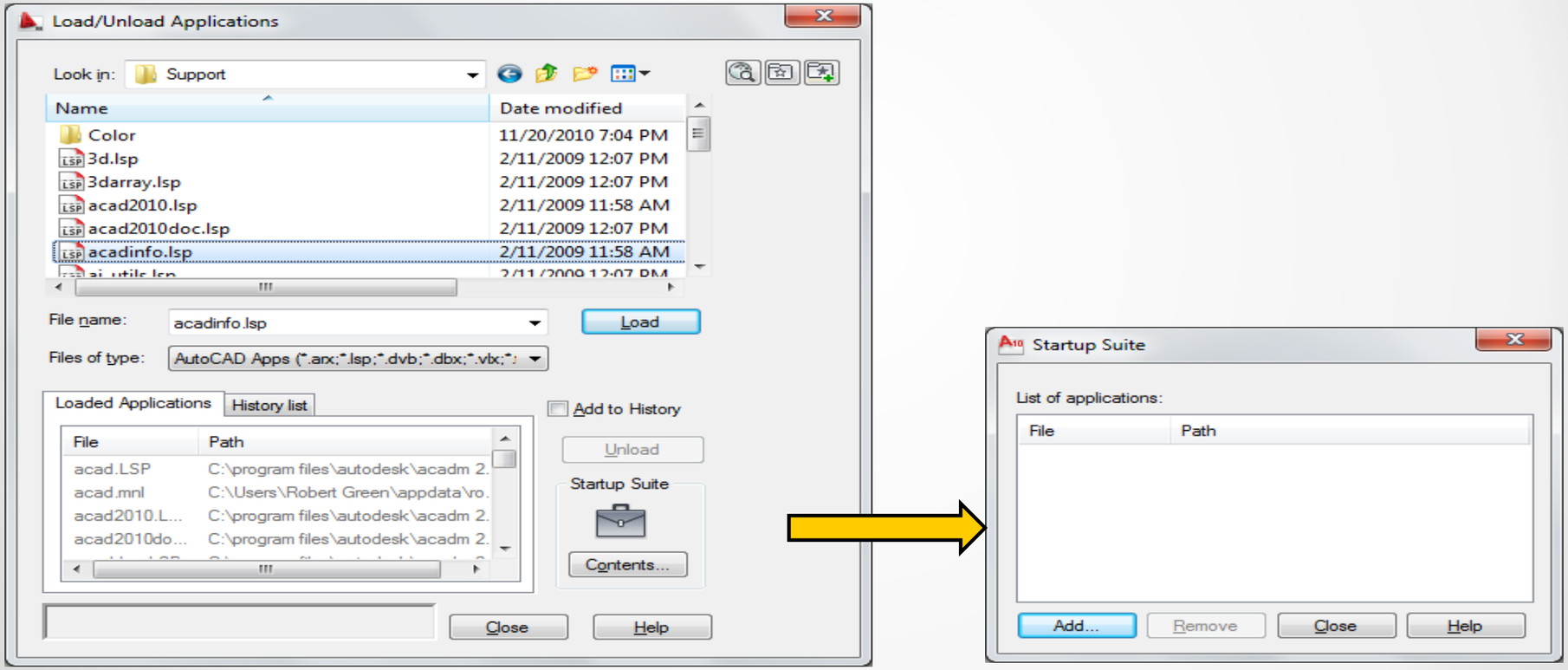
Save the file …

- To the SUPPORT folder
- Use ACADDOC.LSP as the name



Alternately …

- You can use APPLOAD to load files
- You can use STARTUP SUITE to load at each start



Syntax Examples

Controlling Variables

Using SETVAR functions

SETVAR (SET VARiable)

- To set the system variable for the fillet radius to 0.25 do this:
  (setvar “filletrad” 0.25)

- To set the system variable for expert status do this:
  (setvar “expert” 5)

- To set the system variable for dimension style do this:
  (setvar “dimstyle” “am_ansi”)

Command line access

- (command “viewres” “y” “5000”)
- (command “-color” “BYLAYER”)
- (command “-linetype” “set” “BYLAYER” “”)
- (command “menu” “menuname.cuix”)

- That’s not so bad …intuitive actually …

SETQ (SET eQual)

- To create the variable MYNAME and set it with your proper name as a value do this:

  (setq username “Robert Green”)

- To store a system variable for a custom path do this:

  (setq lisp_path “z:\\lisp”))

Syntax Examples

Custom functions that add to the command set

User functions

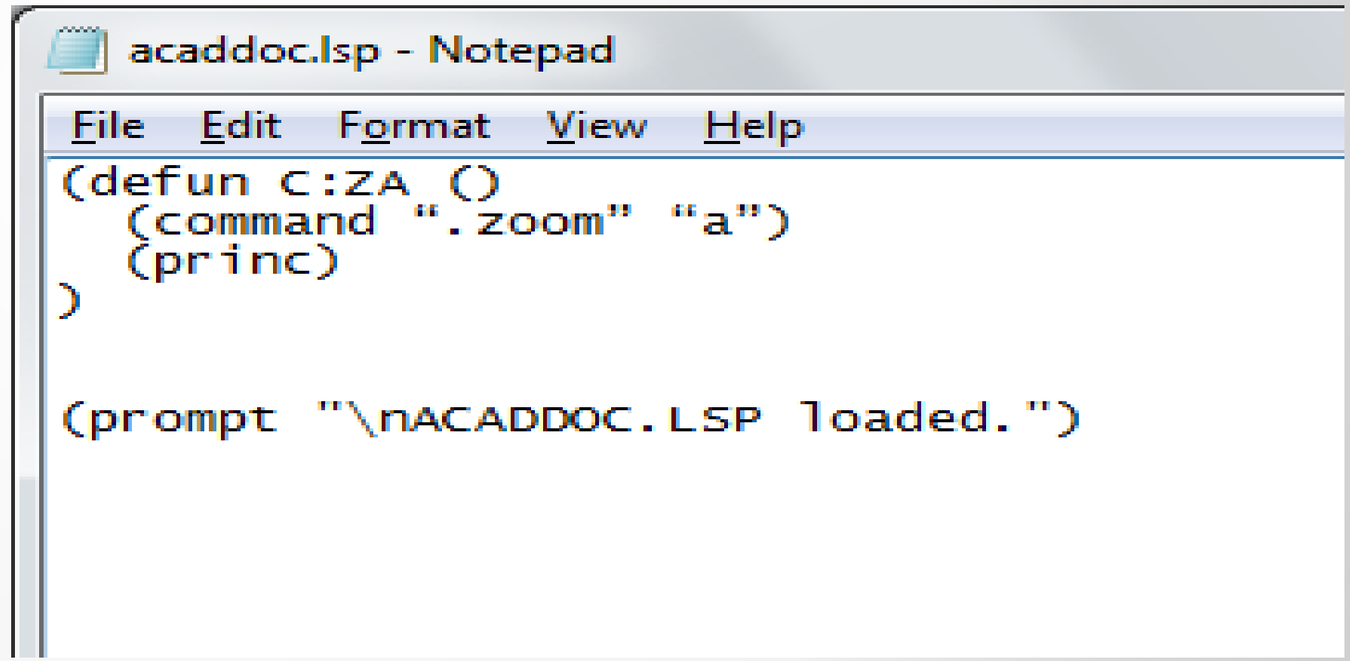
- Speed for the user
- Lower support for you
- A win-win scenario
- Let’s put everything we’ve learned into action to build some functions.

User Function Examples

```
(defun C:ZA ()
  (command ".zoom" "a")
  (princ)
)
```

User Function Examples

```
(defun C:ZA ()
  (command ".zoom" "a")
  (princ)
)
```



Auto Purge Function

Auto Purge

```
(defun c:atp ()
  (command “-purge” “a”  “*” “n” “.qsave”)
  (princ)
)

(defun c:atp ()
  (command “-purge” “b”  “*” “n” “.qsave”)
  (princ)
)
```

User Function Examples

```
(defun C:VR ()
  (command “viewres” “y” “5000”)
  (princ)
)

(defun C:BL ()
  (command “-color” “BYLAYER”)
  (command “-linetype” “set” “BYLAYER” “”)
  (princ)
)
```

Fillet Zero Function

Fillet Zero

```
(defun c:fz ()
  (setvar “filletrad” 0.0)
  (command “.fillet” pause pause)
  (princ)
)
```

What have I not done in this function?

What does PAUSE do?

Improved Fillet Zero

```
(defun c:fz ()
  (setq old_filletrad (getvar “filletrad”))
  (setvar “filletrad” 0.0)
  (command “.fillet” pause pause)
  (setvar “filletrad” old_filletrad)
  (princ)
)
```

* Note how we store and recall the FILLETRAD so the function puts things back the way they were!

Syntax Examples

Using Error Handlers

(Dealing with crashes and user ESC scenarios)

Fillet Zero Glitch!

We can see from the FZ command is that if the user hit ESC just prior to the FILLET line that the FILLETRAD system variable would never get reset.

```
(defun c:fz ()
  (setq old_filletrad (getvar “filletrad”))
  (setvar “filletrad” 0.0)
  (command “fillet” pause pause)
  (setvar “filletrad” old_filletrad)
  (princ)
)
```

The Error Handler Function

To get around this problem lets create an error handling routine that'll set FILLETRAD back to the prior value:

```
(defun *fz_error* (msg)
  ;; If an error (such as CTRL-C or ESC) occurs
  (if (/= msg "Function cancelled")
    (princ (strcat "\nError: " msg "\n"))
    (princ)
  )
  (if old_filletrad (setvar " filletrad " old_ filletrad))
  (if *old_error* (setq *error* *old_error*))
  (princ)
)
```

The Error Handler Function

Add error function references to C:FZ

```
(defun c:fz ()
  (setq old_filletrad (getvar “filletrad”))
  (setq *old_error* *error*) ; Store the old error handler
  (setq *error* *fz_error*)     ; Set new error handler
  .
  .
  (setq *error* *old_error*) ; Put the error handler back
  (princ)
)
```

Error Handler Recap

- Every C: type function gets an error handler
- The error handler resets any variables defined in the C: function
- The default error handler is always *error*
- You just swap your error handlers in/out

- Print out the handout and really look at the C:FZ and FZ_ERROR functions and it’ll start to make sense …

Syntax Examples

Calling external programs

Call outside programs …

- To invoke a browser do this:

  (command “browser” “file goes here”)

- To invoke an app:

  (startapp “C:\\folder\\progname.exe”)

From a function it looks like this …

```
(defun c:np ()
  (startapp “notepad.exe”)
  (princ)
)

(defun c:myprog ()
  (startapp “c:\\progpath\\myprogram.exe”)
  (princ)
)
```

Syntax Examples

Undefining commands to subtract from the command set

Undefining …

- **(command “.undefine” “LINE”)**
- **(command “.undefine” “TORUS”)**

- **Don’t want users messing with a command?**
- **Just undefine it …**

▶ Now you can SUBTRACT from the AutoCAD Command set in your ACADDOC.LSP file.

The DOT form …

- Invoke commands like this: .LINE
- Note the dot “.” character?
- This allows you to invoke a command whether it has been undefined or not!
- This is our little secret right ...

Syntax Examples

Redefining commands that have been undefined

Redefining …

- **(command “.redefine” “LINE”)**
- **(command “.redefine” “TORUS”)**

- **Want to be sure that a command is active?**
- **Just redefine it …**

▶ Now you can UNSUBTRACT from the AutoCAD Command set with ease.

Undefining revisited …

- What if your users find out about REDEFINE and start REDEFINING your UNDEFINES?
- Just undefine the redefine like this:
- (command “.undefine” “REDEFINE”)
- Now they know who’s boss …

Syntax Examples

Alerting and Undefining Functions
(to change the way commands operate)

Alerting the user …

- **You can send a message to the user like this:**

  (alert “Message goes here”)

Redefining …

- **You can undefine a command and redefine it like this:**

```
(command ".undefine" "TORUS")

(defun C:TORUS ()
  (alert "Don't use that command!")
  (princ)
)
```

- Now you do whatever you want!

What Does This Do?

```
(command ".undefine" "QSAVE")

(defun c:qsave ()
  (command "-purge" "b"  "*" "n")
  (command ".qsave")
  (princ)
)
```

Command echo (CMDECHO)

Run in STEALTH mode like this:

```
(defun C:QSAVE ()
  (setvar “cmdecho” 0)
  (command “-purge” “b”  “*” “n”)
  (command “.qsave”)
  (setvar “cmdecho” 1)
  (princ)
)
```

*** CMDECHO is on or off so no need to store it’s value**

Syntax Examples

CUI Manipulation/Loading

Load a CUI like this …

(command “menu” “c:\\path\\cuiname.cuix”)

- **Note: Not MENULOAD but MENU …**

Load a workspace from the CUI like this …

(command “-workspace” “workspacename”)

Supporting Power Users

LSP/CUI Manipulation/Loading

When power users need freedom

Give them “backdoor” way to load their own LISP or CUI files.

However, make sure that the backdoor is standard!

Let’s see how …

USER.LSP

Agree that all LSP files for the power user will reside in a file called USER.LSP

Detect the file, if there, and load it.

The power is totally responsible for the file!

USER.LSP

```
(if (findfile “user.lsp”)
  (load “user.lsp”)
)
```

USER.CUIx

We’ll use the same trick for CUI files:

```
(if (findfile “user.cuix”)
  (command “menu” “user.cuix”)
)
```

USER.CUIx + Workspace

Now we can load the CUI file and set a current workspace.
Note the use of PROGN:

```
(if (findfile “user.cuix”)
  (progn
     (command “menu” “user.cuix”)
     (command “-workspace” “user”)
   )
)
```

Network Loading

ACADDOC.LSP local loads a network INIT.LSP

Wouldn’t it be great if …

We could maintain all our code on a server.

Go to the user’s machine just once and set it so it loads from the server?

Let’s see how …

ACADDOC.LSP on user's machine

```
(setq lisp_path "X:\\AUTOLISP\\")   ; sets the path

(if (findfile (strcat lisp_path "init.lsp"))
   (load (strcat lisp_path "init.lsp"))
)
```

Have a different LISP_PATH for each AutoCAD version!

INIT.LSP

This is the file on the network drive

It contains all the SETVAR, functions, commands that we've talked about to this point.

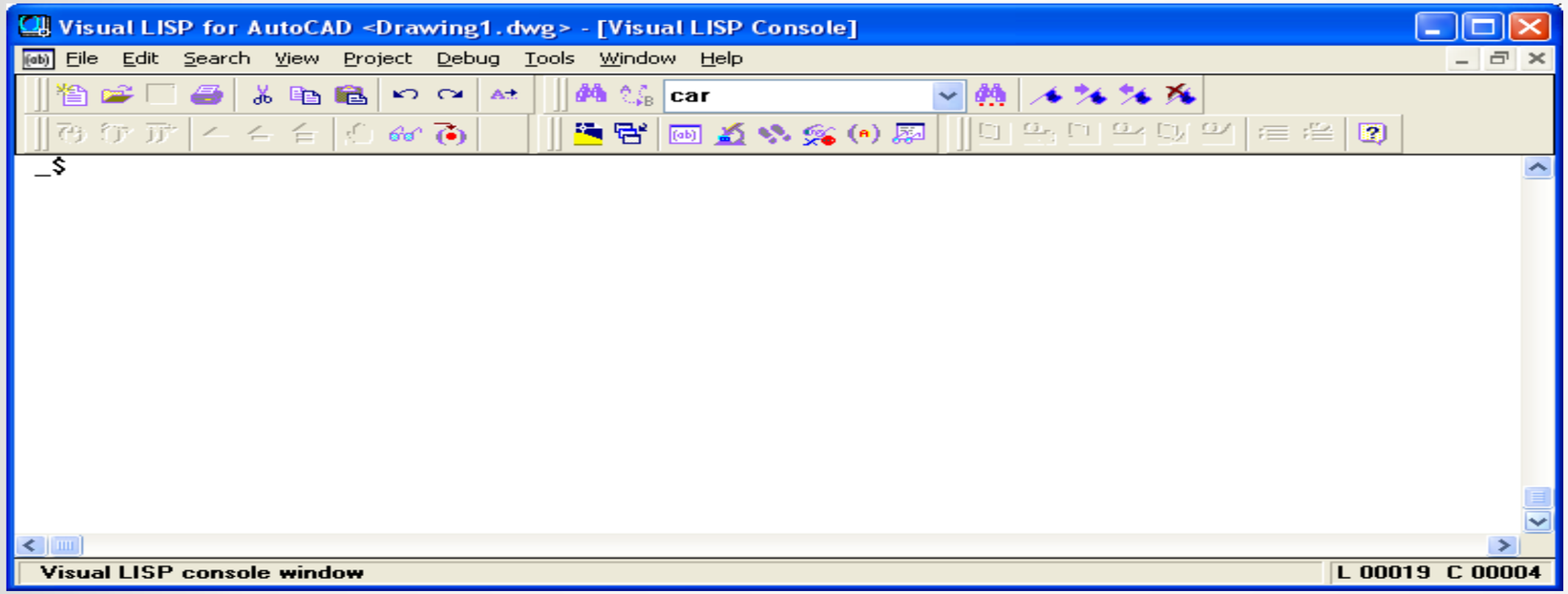
Now you can update everybody very easily

VLIDE

Compiling/writing code

VLIDE environment …

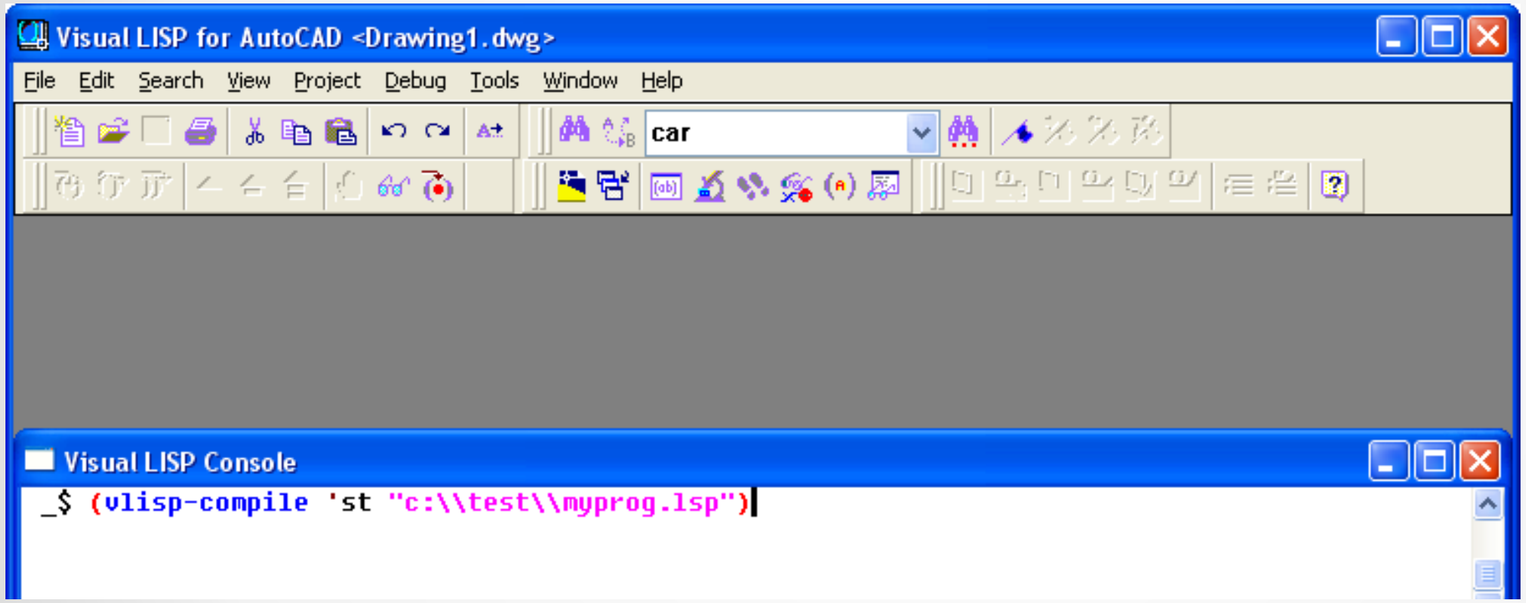
- **You can write code in the VLIDE window like this:**



Compile your code …

- **Use the VLIDE environment like this:**

(vlisp-compile ‘st “c:\\test\\myprog.lsp”)



- **You’ll get MYPROG.FAS as a result …**

Load compiled code …

- **Use a LOAD statement like this:**

  (load “c:\\test\\myprog.fas”)

- **Now your LSP code is secure!**

- **Be sure not to lose your source LSP file though!**

Bypassing Profiles

LSP/CUI Manipulation/Loading

Wouldn’t it be great if …

You could set printer directories, palette directories, etc, without using profiles?

AutoLISP gives us a powerful, yet easy, set of tools to do so.

Lets see how …

GETENV and SETENV

These functions read (GET) or write (SET) profile values in the user's current active profile.

What does this mean for CAD managers?

It means we no longer care which profile is current!

The tricky part …

You've got to know the exact profile variable key to read/write from the CURRENT USER registry area.

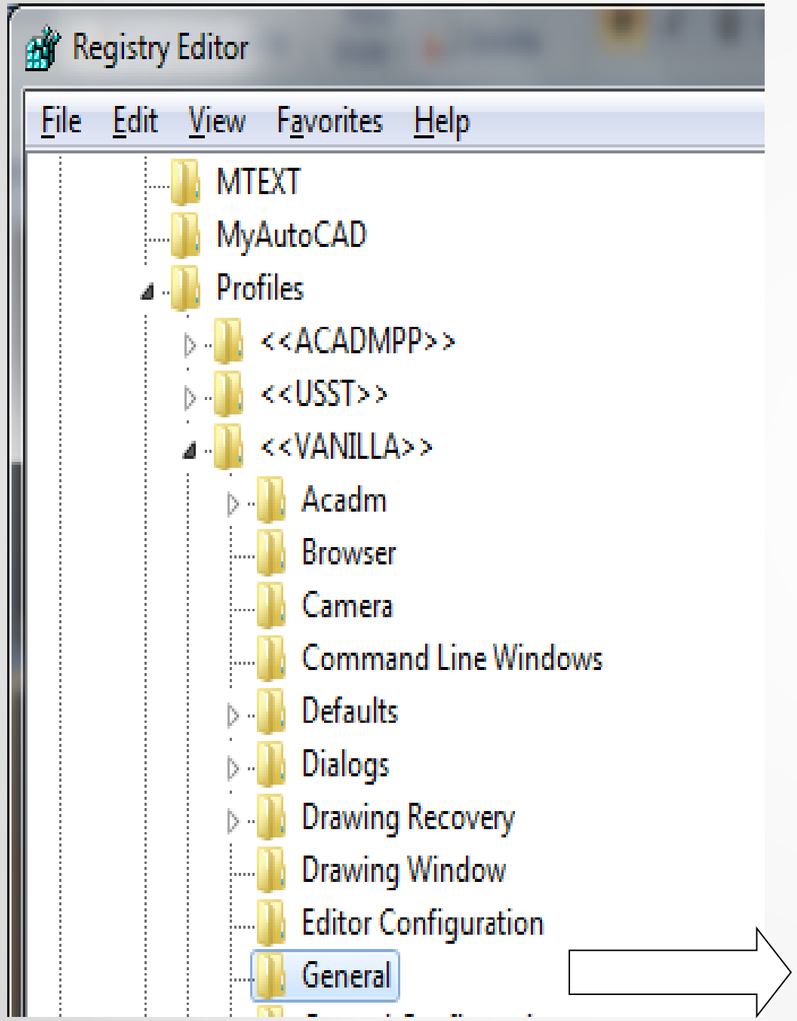
This can require some detective work on your part.

Let's see how …

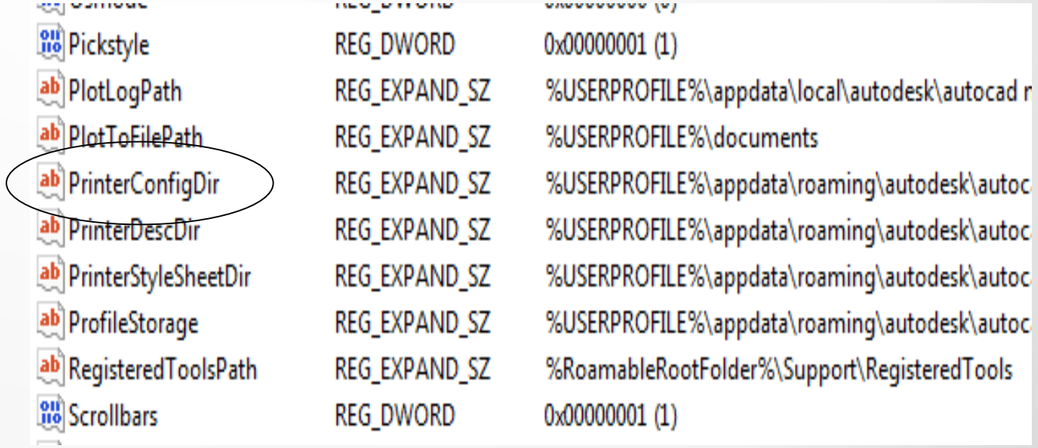
Useful cheat codes …

Support path: **ACAD**
DRV path: **ACADDRV**
Printer definitions: **PrinterConfigDir**
Palettes directory: **ToolPalettePath**
Templates location: **TemplatePath**
Plot Styles: **PrinterStyleSheetDir**
Current CUI: **MenuFile**
Enterprise CUI: **EnterpriseMenuFile**
Current Workspace: **WSCURRENT**
2014 Trusted Paths: **TRUSTEDPATHS**

Using REGEDIT …



Find the current profile
Look under GENERAL
Find the key
Note exact spelling



Now the code …

Set the printer path like this:

```
(setenv "PrinterConfigDir" "Z:\\acad\\Plotters")
```

This is better:

```
(setenv "PrinterConfigDir"
  (strcat "Z:\\acad\\Plotters;" (getenv “PrinterConfigDir”))
)
```

Note …

In this example:

```
(setenv "PrinterConfigDir"
  (strcat "Z:\\acad\\Plotters;" (getenv “PrinterConfigDir”))
)
```

I had to use “;” in the path to support multiple paths!

Better still …

```
(if (not (wcmatch (getenv “PrinterConfigDir”)
                  *Z:\\acad\\Plotters*) )
  (setenv "PrinterConfigDir"
    (strcat "Z:\\acad\\Plotters;" (getenv “PrinterConfigDir”))
  )
)
```

I had to use “;” in the path to support multiple paths!

Wrapping Up

Drawing some conclusions

AutoLISP is for CAD Managers!

I hope you've started to see what a great management tool AutoLISP is!

I will have final downloadable handouts, code samples, and slides in PDF form tonight.

Thanks for Attending!

AutoLISP® Strategies for CAD Management

Robert Green

Robert Green Consulting