Autodesk University 2007

Soak. Wash. Rinse. Spin. Repeat 1000X: Passionate Scripting For Everyone

dave espinosa-aguilar – City of Richland, WA

GD311-3P: Some processes in AutoCAD are about as exciting as washing an old pair of dirty socks -- over and over again. But it needn't be this way with the easy and powerful Scripting language in AutoCAD. In a nutshell, if you can type anything in AutoCAD, then you can script it. This class explains how scripts can automate the redundant and mind-numbing aspects of your daily design. We'll demonstrate advanced scripting techniques that will destroy myths regarding scripting limitations, and also introduce users to the latest version of ScriptPro, a utility that easily applies scripts to batches of drawings at a time.

About the Speaker: A consultant in the CAD and multimedia industry since 1986, dave espinosa-aguilar has trained architectural and engineering firms on the general use, customization, and advanced programming of design and visualization applications from Autodesk including AutoCAD, AutoCAD Map, Architectural Desktop, Land Desktop and Civil 3D, 3DS Max, and Autodesk VIZ. His passion is streamlining and automating design production environments through onsite customization and programming, and he has authored the facilities management applications of several Fortune 500 companies using AutoCAD ObjectARX, VB/VBA, AutoLISP/DCL and MAXScript technologies. dave has also produced graphics applications and animations for Toxic Frog Multimedia and has co-authored several books including NRP's "Inside 3D Studio MAX" series. He has been a speaker at Autodesk University since its inception, and served on the AUGI Board of Directors for 6 years including the office of President in 1996. dave currently works for the City of Richland, Washington's, GIS department as a software engineer.

Autodesk®


Autodesk
University
2007

Autodesk University 2007

Boredom to the Rescue!

There may have been a time in your life when using AutoCAD to do just about anything (including your wedding invitations) was incredibly exciting. But after years (maybe it was after months? after weeks perhaps?) of being emersed in drafting production work, certain tasks in AutoCAD became downright painful for you to slog through... especially those that required mind-numbing, repetitive processes. There are times when being bored out of your mind can be a powerful cue for you to consider and investigate new methods for dealing with a situation.

Or old ones. Really old ones that work so well that most people who discover them can't believe the solution has been right under their nose practically since the product was first launched back in the 1980's. The solution is called **scripting**. It's free, it's always been there, it's easier than learning how to edit vertices using the PEDIT command, and it is forehead-slapping simple to use. In fact, if you know how to **type** any commands in AutoCAD, then you already know how to script in AutoCAD. Toolbars and pulldown menus can look very attractive when you're doing a thing once or twice in a minute. However, they are absolute hell to use when you have to do a thing 1000 a day. Welcome to the ultimate out-of-the-box automating technology in AutoCAD: the Command prompt.

Taking Command of ... the Command Prompt!

The command prompt has been there in AutoCAD since the beginning, and there are a lot of solid reasons why it is still in there. Over the years, some folks have suggested removing the command prompt, thinking it a waste of screen realty or an outdated interface to be bypassed by sexier dialogs, tool palettes, toolbars, dashboards and the like. But anyone who's been around a while, who's spent a decade or more in the AutoCAD production trenches, knows that it is not only the fastest interface to any deep functionality in AutoCAD (ever notice the Top Dogs are always typists?), it if not only the most consistent interface to AutoCAD in the last two decades (I'll bet you can draw a LINE instantly in a next new release by typing it, no matter where it gets buried in the next pulldown menu)… it is the only automatable interface to AutoCAD which does not require programming skills. In fact, it is quite literally AutoCAD's stream of consciousness interface—a direct line to your mind. What a user types, a script types: keystroke for literal keystroke. Therefore, the more you know about the way a command functions by typing it, the more your scripts can immitate your steps verbatum to do it. Over and over a thousand times. It is a lightening fast keyboard feeding machine, and in those cases where a thing has to be 1000 times, it can be a lifesaver.

Streams of Consciousness

For some time now, the F2 key has launched the Text Window which reports AutoCAD's Command Prompt history. Content in the window behaves like simple ASCII text, and it can be copied and pasted easily to Windows Notepad or any other Windows program. The history includes all the AutoCAD prompts, commands and system variables, default values and settings, error messages, reports, and of course everything the user has literally typed or "typed by invoking another interface." By typing any process you would like to be repeated easily in one fast step, all you need to do is manually perform the process once (by typing), capture it, copy/paste it to Notepad, edit it down to only your keystrokes, and save it to an .SCR file. Yes. It is literally that simple. And what kind of a mind-numbing process might compell you to verify this for yourself?

Imagine working for an idiot who actually thinks the best way to organize his architectural layers is to number them ("1", "2", etc.). Imagine that whenever he sends you his drawings, you never know what layers he has frozen or thawed or how zoomed into the drawing he was when he last saved it. Imagine that his POINT are shown points as invisible dots which can't be seen at all. Imagine he always puts vital information on layer "0" which really needs to be moved to a dedicated layer of his own, and imagine he always includes useless information on layer "1." So every time you open a drawing from this guy, you wind up having to:

1. thaw all the drawing layers
2. zoom to extents
3. set PDMODE to something other than 0 (maybe 1)
4. create a layer called "idiot" and move everything on layer 0 to that layer
5. rename layer "2" to "wall"

And the guy sends you on average 10-20 drawings per project that use this insane "standard." Guess who gets to use all those new and thrilling interfaces in the latest release of AutoCAD to change all of these things over to a coherent standard of your own? You do. And there you sit, wondering whatever possessed you in the first place to take on a job like this. Oh that's right: money! So you slog your way thru these 5 steps every time you open a drawing from this guy.

It's not as bad as it sounds. Not for 10-20 drawings that are like this… or a 1000. Crack your knuckles, open Notepad, and get ready to do this process one more time, completely typed at the Command prompt… and for the last time in your life. The command prompt sequence would look something like this (the boldfaced typing indicates your strokes):

Command: **-layer**
Current layer: "5"
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **t**
Enter name list of layer(s) to thaw: *
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]:

Command: **zoom**
Specify corner of window, enter a scale factor (nX or nXP), or
[All/Center/Dynamic/Extents/Previous/Scale/Window/Object] <real time>: **e**
Regenerating model.

Command: **pdmode**
Enter new value for PDMODE <0>: **1**
Regenerating model.

Command: **-layer**
Current layer: "0"
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **m**
Enter name for new layer (becomes the current layer) <0>: **idiot**
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **s**
Enter layer name to make current or <select object>: **0**
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **f**
Enter name list of layer(s) to freeze or <select objects>: *****
Cannot freeze layer "0". It is the CURRENT layer.
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]:

Command: **change**
Select objects: **all**
4 found
Select objects:
Specify change point or [Properties]: **p**
Enter property to change
[Color/Elev/LAyer/LType/ltScale/LWeight/Thickness/Material]: **la**
Enter new layer name <0>: **idiot**
Enter property to change
[Color/Elev/LAyer/LType/ltScale/LWeight/Thickness/Material]:

Command: **-layer**
Current layer: "0"
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **t**
Enter name list of layer(s) to thaw: **1**
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **s**
Enter layer name to make current or <select object>: **1**
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]: **f**
Enter name list of layer(s) to freeze or <select objects>: *****
Cannot freeze layer "1". It is the CURRENT layer.
Enter an option
[?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/MATerial/Plot/Freeze/Thaw/LOck/Unlock/stAte]:

Command: **erase**
Select objects: **all**
2 found
Select objects:

Command: **-rename**
Enter object type to rename
[Block/Dimstyle/LAyer/LType/Material/Style/Tablestyle/Ucs/VIew/VPort]: **la**
Enter old layer name: **2**
Enter new layer name: **wall**

Command:

-layer
t
*

zoom
e
pdmode
1
-layer
m
idiot
s
0
f
*****

change
all

p
la
idiot

-layer
t
1
s
1
f
*****

erase
all

-rename
la
2
wall
-layer
t
*****

Press F2 to bring up the Text Window and all your keystrokes are recorded there. Select them with your mouse (prompts and all), copy them to the clipboard and paste them intoto Notepad. Now you whittle them down to only your keystrokes.

Places where you pressed ENTER in the sequence are left as a blank line. It would look something like what you see in the column to the right of the command prompt sequence! Note again that blank lines signify <ENTER> keystrokes. Save this information to a file called **idiot.scr**. Make sure it is saved as an ASCII text file and **with a .SCR file extension** (not .TXT) so that AutoCAD can recognize the file specifically as a script file. Now when you open a drawing from this client, you can drop and drag idiot.scr right into the graphics area of the drawing… and watch the automated steps scream by as if you had typed them all at 10,000 words per minute! Need to do it again on his next drawing? Same one-step drop-and-drag. And the script can be supplemented or edited as needed if your client's "standard" ever changes. This is the essential magic of scripting, and it really is this simple to automate 10, 100 or 1000 steps in any drawing.

Pre-Scriptions

Now that you've seen the basic concept of scripting in action, it's time to discuss the typical types of hiccups, weirdness, and problems that arise in writing scripts. Copying/Pasting from the Text Window is an easy way to not have to remember the exact sequence of commands and values you need to create a script, but sometimes the following things happen:

1. you accidentally type the wrong thing and need to backtrack: no problem. undo your step and then redo it. when you copy the Text Window, make a note to yourself where the bad sequence of steps are and chop them out.
2. you forget where the <ENTER> key strokes are: no problem. anything that shows blank values after a colon, simply verify the sequence manually and then make sure a blank line is in your script to represent it. Too many blank lines (or extra spaces!!) can send you unexpectedly back into the last command typed (just as if you had pressed the ENTER key too many times). Don't beat yourself up over this. EVERYONE goes thru this tweaking process until they get their scripts working perfectly. Welcome to the world of "debugging."
3. you use aliases but the script is run on someone else's system that use different aliases: no problem, type the full command when you do the write the script. use a period in front of the command if you want to guarentee the command behavior (ex: **.LINE**), and avoid using aliases in any script.
4. a command sequence changes in a new release of AutoCAD: no problem (and welcome to the wonderful world of migration). save the original script in a release-dedicated directory, make changes to the original and save it in a new release directory. keep directory versions of your scripts safely apart from each other.
5. the script doesn't seem to be behaving: never run your new scripts on production work. always test them out on junk copies in safe places. follow the script till it breaks, tweak that portion of the script till you get it right, and then after plenty of testing you can move it to production drawing use. many things can caus a script to bomb… make sure hat any command you use passes its instructions to the command prompt only. disable dialogs (CMDDIA, FILEDIA, etc) before running the rest of your script (you can include SETVAR commands to do this for you in the script itself!) and be sure to restore those settings when the script is finished (that too can be done with the SETVAR command in the script itself).

Scripting is an artform. Some folks are so proficient at it that they can immediately start writing scripts without having to manually do the commands at the command prompt ebcause they have the sequence of commands down so well. They also know the typical problems that a script runs into and how those problems get solved quicky. As with anything in AutoCAD, the best way to get better with a feature set or technology is to use it more and more.

Script Stript: More Automated Boredom

Over the years, users have written some clever routines to reduce Text Window content to only their own keystrokes. The AutoLISP routine below is an example of one such "stripper" which examines every line in a script file and deletes anything it finds left of (and including) a colon. Even a crude utility can save tons of manual editing time on big scripts.

```
(defun C:STRIPPER ()
  (setvar "cmdecho" 0)  (setq fnam (getstring "\nEnter text window file to script-strip: "))
  (setq f (open fnam "r"))
  (if f (progn
    (setq g (open (strcat (substr fnam 1 (- (strlen fnam) 4)) "-build.scr") "w"))
    (setq txt (read-line f))  (setq tflag 0)
    (while txt
      (setq ca 1 ta (strlen txt) cb 1)
      (while (<= ca ta)
        (setq test (substr txt ca 1))
        (if (= test ":") (setq cb (+ ca 1) tflag 1))
        (setq ca (+ ca 1))
      )
      (if (= tflag 1) (write-line (substr txt cb) g))
      (if (= tflag 0) (write-line txt g))
      (setq tflag 0)
      (setq txt (read-line f))
    )
    (close f)
  ))
```

```
  (princ)
)
```

Script Flipt: PowerUser Inspirations

Time for some radical thinking. Give these amazing ideas a whirl. PowerUsers came up with them for a reason:

1. A drawing can be entirely scripted instead of saved as a DWG. That's right. Bizarre but true. The AutoLISP routine below collects all LINEs in a drawing and spits them out as a SCR file. There are all sorts of reasons why portions of a drawing might be more efficient to script than to draw… especially if the drawing has replaceable elements to it or processes involved with it. Give this idea a thorough thinking… it is more powerful than most realize on first hearing.

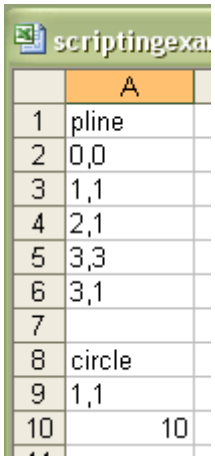
```
(defun C:LINESCRIPTER ()
  (setq fnam (getstring "\nEnter script name: ")) (setq sa (ssget "x" (list (cons 0 "LINE"))))
  (if sa (progn
    (setq f (open fnam "w") ca 0 ta (sslength sa))
    (while (< ca ta) (setq enta (ssname sa ca) ea (entget enta) typ (cdr (assoc 0 ea)))
      (if (= typ "LINE") (progn
        (setq txt (strcat "LINE " (rtos (car (cdr (assoc 10 ea))) 2 8)
                          ","     (rtos (cadr (cdr (assoc 10 ea))) 2 8)
                          ","     (rtos (caddr (cdr (assoc 10 ea))) 2 8)
                          " "     (rtos (car (cdr (assoc 11 ea))) 2 8)
                          ","     (rtos (cadr (cdr (assoc 11 ea))) 2 8)
                          ","     (rtos (caddr (cdr (assoc 11 ea))) 2 8)
                          " "
        )) (write-line txt f) )) (setq ca (+ ca 1)) ) (close f) )) (princ)
)
```

<u>If LINEs exist in the drawing, the following type of SCR file might be produced:</u>

```
LINE 20.20230924,29.40713991,0.00000000 18.01560702,30.69538384,0.00000000
LINE 17.31273844,26.04989818,0.00000000 20.20230924,29.40713991,0.00000000
LINE 22.85759048,26.16701123,0.00000000 17.31273844,26.04989818,0.00000000
LINE 21.99852893,20.23328165,0.00000000 22.85759048,26.16701123,0.00000000
LINE 28.83197335,25.23010658,0.00000000 21.99852893,20.23328165,0.00000000
LINE 33.08823301,20.93596014,0.00000000 28.83197335,25.23010658,0.00000000
```

Spaces act as <ENTER>. A space following the end of the line also acts as <ENTER>. Therefore, scripts can look more like sentences than line-by-line sequences. The original script first shared in this paper could look like this (note: the $ character represents blank spaces actually --- you'll need to replace $'s with blank spaces to make the SCR work):

```
-layer$t$*$$zoom$e$pdmode$1$
-layer$m$idiot$s$0$f$*$$change$all$$p$la$idiot$$
-layer$t$1$s$1$f$*$$erase$all$$
-rename$la$2$wall$
-layer$t$*$$
```



2. Entire environments can be set up with a single script. Do you have to work on other people's stations regularly? Tired of their profiles not being like yours? Every system variable can be typed to your desired setting. By running a "MYWAY.SCR" on any system, you can effectively configure it with one drop-and-drag from your flashdrive. Nifty!
3. Script files can be loaded and run many ways:
   a. You can use the **SCRIPT** command which will show .SCR files and run them.
   b. You can drop-and-drag .SCR files into the graphics area of AutoCAD to run them.
   c. You can copy/paste scripting lines right to the command prompt!
   d. You can load scripts from toolbars, pulldown menus and tool palettes. Simply use the format **^c^cscript;c:/myscript.scr;** to run the SCRIPT command. The / character is used for directory trees, the \ character is a pause and a semicolon acts like the <ENTER> key in this macro language. The **^c^c** characters act as 2 escape keystrokes. CUI/menu files are full of statements like these.
4. Scripts can be saved as Microsoft EXCEL files (.XLS). That's right, you can save each line as a row and copy paste an entire Excel column to the command prompt as a script stream. By using the =CONCATENATE function, you can even build new columns of "glued together" data that can be pasted right to AutoCAD's command prompt. This makes an ideal method for importing COGO data points and other civil engineering data collector type stuff.
5. By placing a hyphen in front of certain commands that bring up dialogs, you can not only bypass the dialog (and therefore get to the command prompt versions of the command), but more often than not, additional functionality of the command **not available in the dialog interface** is now accessible. Case in point: there is no REGAPP ability in the **PURGE** command (dialog format), but REGAPP is available in the **-PURGE** command. Tons of commands in AutoCAD provide additional functionality when their hyphenated versions are used instead.
6. Can scripting be used to **PLOT/-PLOT** or **PUBLISH/-PUBLISH** drawings? You bet your sweet PLT file it can.

7. You can place comments in a script by beginning the line with a semicolon (;) which makes tracking activity easy!

Opening Lines In The Script

You can launch AutoCAD to immediately run a script after opening a drawing! You can do this by setting a Desktop icon's target, or use the RUN dialog box using a special command line syntax which governs what script AutoCAD should use. If you want a script to run every time the user opens a file within an AutoCAD session, add the script to the Startup function in the Acaddoc.lsp file. Use this type of format to start AutoCAD from a Desktop icon or the Run dialog:

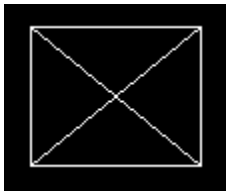
```
"C:\Program Files\Autodesk Map 3D 2007\acad.exe" /b "D:\Docs\Users\MyStartupScript.scr"
```

Why would this helpful?

1. You could manipulate layer states, zooms and views, insert titleblocks and populate their attributes from system and network variables and dates, or even scrub out certain types of data from any drawing launched with a specific desktop icon. Our office uses a desktop icon called AutoCAD Scrubber which opens any drawing and immediately cycles thru hundreds of processes to insure the drawing meets our office standards. Unlike the STANDARDS command which reports on limited aspects of the drawing, our script simply acts upon any open drawings and ensures it conforms to standards. Thru clever layer manipulation, selection and editing, every drawing opened obeys the rules… by simply being opened.
2. Session presets could be globalized on any system on a network and set to new values with a single script edit. Standards, templates, styles and numerous other system environment settings can be controlled on any system hat launches because it refers to a single source script for instructions. There are virtually no limits to how AutoCAD can be pre-configured on any network system according to the instructions contained in a single script file. More radical thinking here… especially for CAD Managers wanting consistency on all their CAD systems.

Script Tipt: Getting User Input

A great myth that has circulated for years about scripts is that they cannot be stopped in midstream to accept user feedback and then continue on using that feedback. There is some truth to this myth in that scripts, by design, do not accept user feedback … but never tell an AutoCAD user something can't be done. They will always surprise you, and as you might expect, many very clever ways to have scripts accept user input in midstream have been developed over the years. Some techniques are more elegant than others, but all of them work reliably and consistently.



Method #1: The AutoLISP/RESUME method to draw a box with diagonals:

Script files can use a number of specialized script-based commands including :

- **DELAY:** which delatys the script for a specified number of milliseconds (ex: **DELAY 3000**)
- **GRAPHSCR:** which switches from Text Window mode back to Graphics Screen mode
- **TEXTSCR:** which switches from Graphics Screen mode to Text Window mode
- **RSCRIPT:** which repeats a script
- **RESUME:** which resumes a script not completed. Hmmm.. so.. check out the following "script":

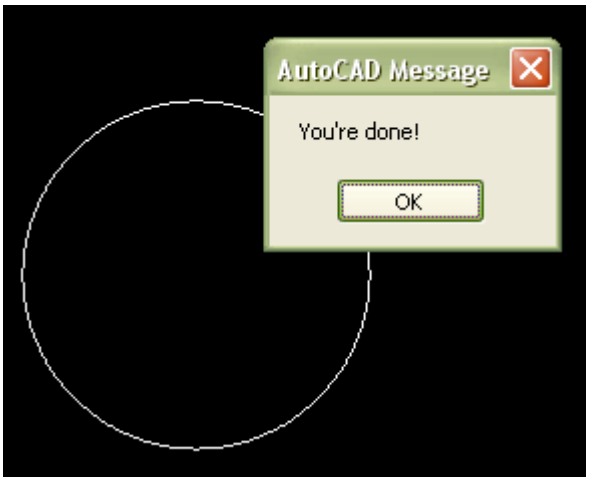
```
(setq p1 (getpoint "\nEnter point: "))
use_RESUME_to_continue
(setq p2 (getpoint p1 "\nEnter next point: ") p3 (list (car p1) (cadr p2) 0.0) p4 (list (car p2) (cadr p1) 0.0))
use_RESUME_to_continue
line !p1 !p3 !p2 !p4 !p1 !p2  line !p3 !p4
```

Normally when AutoLISP statements are included in a script, one gets the message *Can't reenter LISP*. However, the statement does complete even though the script does not. If the user then uses the **RESUME** command, lo and behold, the script picks right back up where it left off and continues to run. The above script asks for a point to be picked. It then "crashes," but it still allows the point to be picked and stored in a "holding pen" (variable) named p1. **RESUME** command is used, and the next point is requested, picked and stored in a different "holding pen" named p2. RESUME command is used again, and the **LINE** command is then invoked and the values stored in "holding pens" p1 and p2 are passed to it. The **use_RESUME_to_continue** line is a command that doesn't exist -- which causes a desired bomb out. Without getting into AutoLISP programming explanations, simple tricks like this can allow a script to gather:

- *points* with (getpoint "\nPut your request for a point here: ")
- *real numbers* with (getreal "\nPut your request for a real number here: ")
- *integers* with (getint "\nPut your request for an integer here: ")

This technique works rather well if you simply explain to the user that once the script deliberately "crashes", he/she simply has to use the **RESUME** command to continue it. Note: there are double spaces after the second **!p2** variable and 2 blank spaces after the **!p4** variable to finish out the **LINE** command. This takes very little AutoLISP understanding too.

Here's another example of how some users have set up simple AutoLISP/RESUME scripts using the (alert) and (command) functions to keep things running smoothly:

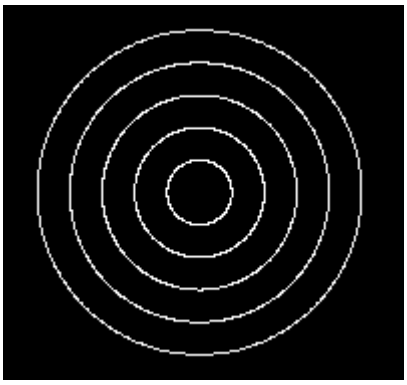



```
(alert "Keep using RESUME until you're told you're done.")
(setq p1 (getpoint "\nPick point: "))
(command)
(setq r1 (getreal "\nEnter radius: "))
(command)
circle
!p1
!r1
(alert "You're done!")
```

By learning a tiny bit of AutoLISP, scripting can really a scripter far. The key point here is simple: there are ways to stop a script in its tracks, get some user input, and then continue the script running.

Method #2: System Variable holding pens

Some AutoLISP functions and statements are completely compatible with scripts from the get-go, for example (grread), (setq), (strcat) and (command). The (getvar) function is another example, and it pulls information from system variables. If the user invokes the **LASTPOINT** function to pick a point, and then uses the **SETVAR** command to fill system variable **USERR1** (a real number holding pen) with a real number value before running the following script, the script will run with no problems:



```
circle (getvar "lastpoint") (getvar "userr1")
circle (getvar "lastpoint") (* (getvar "userr1") 2)
circle (getvar "lastpoint") (* (getvar "userr1") 3)
circle (getvar "lastpoint") (* (getvar "userr1") 4)
circle (getvar "lastpoint") (* (getvar "userr1") 5)
```

The script below is fairly intuitive to understand, but notice it pulls system environment information and glues it to a filename before exporting it. Experimentation and Autodesk discussion forums can really help with scripting like this:

```
explode all
zoom e
-LAYER M 0
-LAYER C white *
change all properties color white
REGEN ZOOM E
(setq outfile (strcat (getvar "dwgprefix") "presentation.eps"))
(command "_export" outfile)
```

Method #3: "Script-Hopping"

It is possible to have the last statement in a script be an AutoLISP (load) command. And it is possible to run a script from an AutoLISP program. Which again in turn can load another AutoLISP routine ad infinitum. The AutoLISP routines and scripts hop each other, AutoLISP collecting the data and the scripts processing it as needed. At any point in this sequence of scripts and AutoLISP routines, user feedback can be invoked, and everything flows smoothly with this technique. Take a serious look at the following sequence of scripts and AutoLISP routines. A script starts the process out, loads an AutoLISP routine to get a string value, and then launch another script. The file naming technique between the two file types also helps immensely:

zoom-a.scr

```
erase
all

rectang
0,0
1,1
zoom
e
```

```
(load "zoom-b.lsp")
```

zoom-b.lsp

```
(setq mag1 (getstring "\nEnter a zoom magnification: "))
(command "script" "zoom-c.scr")
```

zoom-c.scr

```
zoom
!mag1
```

Another benefit to this technique is that scripts or AutoLISP routines can be swapped out in mid-process if a single component needs to be varied per an application. Some tasks are best handled in one or the other… this uses the best of both worlds.

Hey, That's Not In The Script!

Some fun facts about scripts:

- a single U command will undo everything a script has done since a script is treated as a group of commands!
- there are some commands that have no typed interface (and only have a dialog interface)
- using the backspace key halts a script, (which as we now know can be continued with the **RESUME** command)
- **AutoCAD LT** cannot run AutoLISP routines … but it can run scripts!!!
- culprit system variables that typically crash scripts: **SDI**, **CMMDIA**, **FILEDIA**, **INITDIA**, **OSMODE**, **OSNAPCOORD**, **EXPERT**, **CTAB**, **TILEMODE**, **CVPORT**, **FONTALT**, **FONTMAP**, **ISAVEBAK**, **LIMCHECK**, **LIMMIN**, **LIMMAX**… you have no idea how handy this list is.
- check out the **NOMUTT** system variable… some interestin functionality there
- scripts can launch scripts… but the launching script concludes when it does so
- using the DOS prompt can help to write scripts with many file names involved in a directory. for example:

```
dir *.dwg /p/b/s >blockreplacer.scr
```

- there is no command line option for **QSELECT** command. You can use the **SSX** command or the AutoLISP command (ssget).
- use **-TEXT** command for single line/word text instead of **DTEXT** command in script files
- remember that some system variables are undocumented (ex: *_ToolPalettePath … if you want to use more than one location for tool palettes, then each path should be separated by a semicolon, like so:

```
(setvar "*_ToolPalettePath" "x:\\your\\path;y:\\your\\other\\path")
```

- make sure your scripts folder is in the AutoCAD search path when commands go looking for them
- never run new scripts on production drawings while testing. never.
- remember you have to tell AutoCAD to save if you want changes made by scripts to be preserved

If AutoLISP Can “Script-Hop”, Can VBA?

Absolutely. A notification strategy using system variables can be used as one method to keep timing between scripts and VBA clean. For example, if you create a script file "c:\proveit.scr" containing the following commands:

```
zoom
extents
(alert "Executing AutoLISP command now.")
(setvar "USERS1" "completed")
```

The following VBA project will be able to run, launch the script, wait for a “flag” indicating the script is finished, and then resume its own stuff:

```
Public Sub Test()
Dim sLoadScript As String
ThisDrawing.SetVariable "USERS1", ""
ThisDrawing.SetVariable "FILEDIA", 0
sLoadScript = "script c:\proveit.scr" & Chr(13)
ThisDrawing.SendCommand sLoadScript
ThisDrawing.SetVariable "FILEDIA", 1
Do While ThisDrawing.GetVariable("USERS1") <> "completed"
Loop
MsgBox "AutoLISP is completed. Continuing on with VBA…"
End Sub
```

There is one major element to scripting that has not been addressed yet. Suppose we re-examine the idiot client who inspired us to examine our first script we examine which required you to do the following with every drawing he sent you:

1. thaw all the drawing layers
2. zoom to extents
3. set PDMODE to something other than 0 (maybe 1)
4. create a layer called “idiot” and move everything on layer 0 to that layer
5. rename layer “2” to “wall”

Now imagine that he sends you 10-20 drawings a project… or 50. or 100. And maybe he sends you 10 projects in a year. Great client money-wise, but hell to deal with. Wouldn't it be great to be able to take the script you wrote which automates these 5 steps… and apply them to an entire directory of his drawings all at once? Well, you can.

A Batch Made In Heaven: ScriptPro

For a number of years, Autodesk has been providing a utility that amazingly enough very few people even know about. It is usually included in the Migration tools (probably part of the reason why so few people know about it), it is free, and according to Autodesk's own documentation, it does the following:

"ScriptPro is a batch processing utility that allows you to apply a set of commands to multiple drawings. Simply specify a script file that contains the commands you want to run on a single drawing, and then use ScriptPro to apply that script to as many drawings as you like. ScriptPro will handle opening and closing each drawing for you. ScriptPro takes AutoCAD scripting to a new level with an easy-to-use interface, logging, re-usable project files, enhanced scripting with new keywords and utilities, robust error recovery so your processing continues even when AutoCAD can't, sample scripts, and multi-machine processing capabilities."

Wow!

- ScripPro includes its own special scripting language which makes it possible to save scripted edits to the current or past release AutoCAD formats.
- Most first-time users of ScriptPro immediately understand how its interface works. A drop-and-drag interface makes building the drawings list easy and quick.
- ScriptPro assumes that AutoCAD is installed on the system already, and it runs independently of AutoCAD.
- ScriptPro launches a new session of AutoCAD for every drawing but only has one session open at a time. The reason why it closes down is so a fresh AutoCAD session opens every time a script is run. If an error has occurred in a drawing, it won't effect the remaining drawings in the list. This behavior ensures that the script runs on all drawings possible, and it logs any file errors for later handling.
- ScriptPro project files can be created, edited and saved as .SCP files. The project files store the script to be applied, the drawings to be treated, status information about each drawing (whether it was treated completely by the script or not, etc), and settings which govern how the script should behave. You can stop a project in progress and begin it again later since it knows where it left off in its status information. You can also edit the status information of any drawing in the project.
- Remember that, as with scripts themselves, any long file names involved must use double quotes. For example, if you wanted the script to insert a drawing with spaces in its directory or file name, you would use the format:

```
-insert "c:\My Project Files\mydrawing.dwg"
```

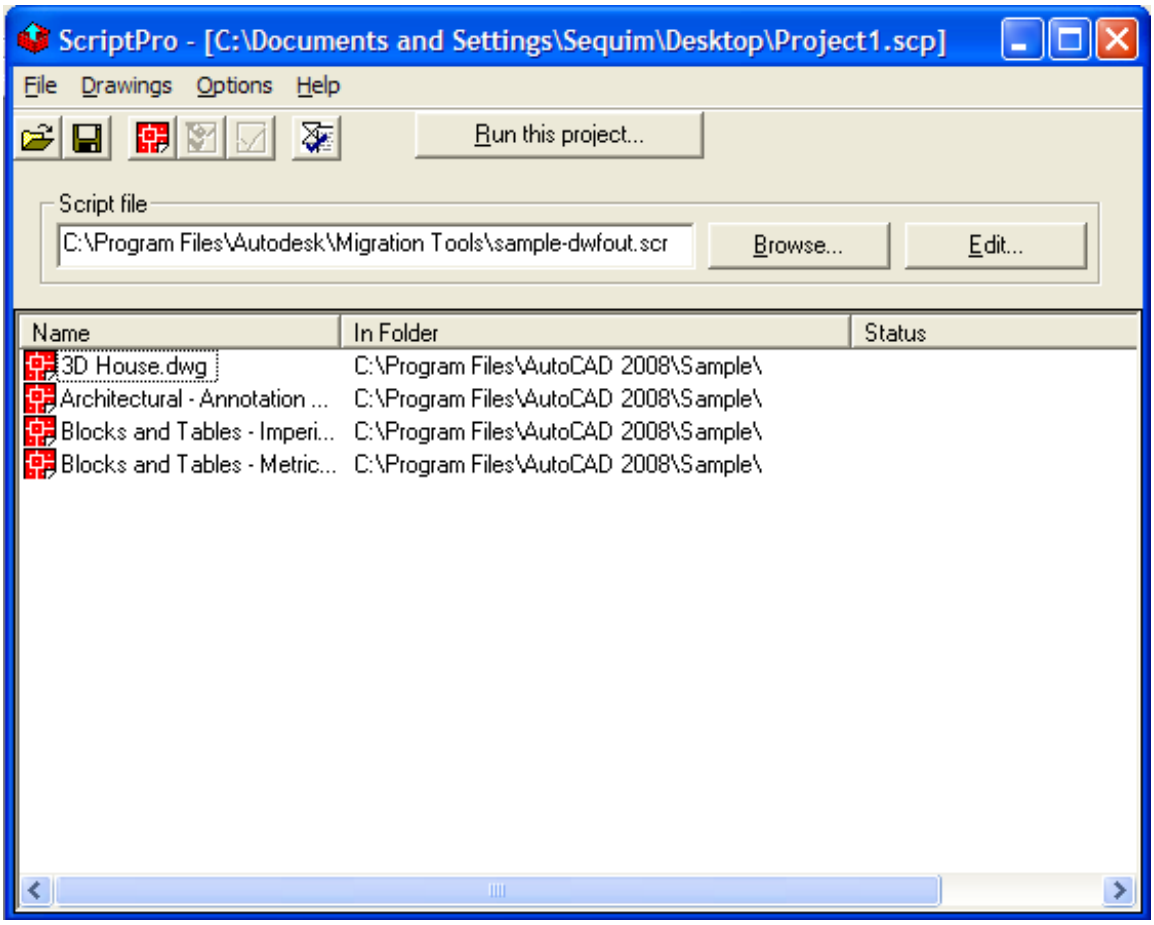
- You can instruct ScriptPro to save any scripted edits by including one of its special save-as functions in the script. This even includes prior releases in DXF and DWg format, but use with caution always on test drawings to make sure the project is behaving the way you want it to. These commands can overwrite existing files if they encounter them while the script is trying to save to them. Examples of ScriptPro saving options:

SCR-DXFOUT-2004	Saves the drawing in AutoCAD 2004 DXF format
SCR-DXFOUT-2000	Saves the drawing in AutoCAD 2000 DXF format
SCR-DXFOUT-R12	Saves the drawing in AutoCAD Release 12 DXF format
SCR-SAVEAS-2004	Saves the drawing in AutoCAD 2004 DWG format
SCR-SAVEAS-2000	Saves the drawing in AutoCAD 2000 DWG format

- ScriptPro comes with sample files to show you how to tweak saving options.
- ScriptPro processes drawings in the order you describe in the project file. The project editor allows you to prioritize which drawings you want script-processed first.
- Every drawing in the project has a status which can have one of the following values:

done	The drawing was successfully processed with no errors.
done*	The drawing was successfully processed but there was a minor error. Usually this is an indication that a dialog box was displayed and successfully dismissed by ScriptPro. Details about the error can be found in the project log file.
error	The drawing was not processed successfully. A short description of the error will be displayed in the Status field and details about the error can be found in the log file.
skip	The drawing was skipped. You can set the skip status for those drawings that you do not want to have processed. Select the drawing in the drawing list, and then, on the Drawings menu, click Set Status to Skip.
<empty>	An empty Status field indicates that the drawing is ready to be processed.

ScriptPro processes only those drawings that have an empty Status field. You can easily reset the status for a drawing by selecting it on the drawing list and then clicking Reset Status on the Drawings menu. You can also reset the status for all drawings by going to the Drawings menu, clicking Select All, and then clicking Reset Status.



Briefly stated, the safe process for using ScriptPro is this:

1. write a script to be applied to as many drawings as you wish
2. manually test the script out on a typical junk drawing to be batch-processed and tweak it to perfection
3. create a sample directory of copied junk drawings to test the project out on
4. startup ScriptPro from the Windows menus as you would any other Windows application (you can also start ScriptPro with the **SCRIPTPRO** command if you're already in AutoCAD… you can even launch a series of ScriptPro project files as a DOS batch file (ahhh… good ole DOS to the rescue always) like so:

   ```
   <install directory>\scriptpro /run <project name>
   ```

   where <install directory> is where the scriptpro.exe is located and <project name> is the .SCP file.
5. in ScriptPro's Project Editor, specify the junk drawings to apply the script to
6. in ScriptPro's Project Editor, specify the script to be applied
7. run the project on the junk test drawing
8. if all works well, make sure the original drawings are archived somewhere safe, adjust the project file to point to the real drawings to be treated, and then run the project.

The importance of securing the original drawings in a safe and separate place before running a project, and testing the script out on junk copies of drawings cannot be overstated. This can save your digital life.

Script Tipt: Successfully Running ScriprPro

When ScriptPro runs a script in a drawing, it uses a special version of the script command called **SPSCRIPT** which provides special keywords for manipulating files and the **CALL** command for calling a script from within a script. ScriptPro also provides some common save commands that deal with unexpected dialog boxes. Consider the following things as you create your script files:

AutoCAD support path: place extra directories on the AutoCAD support path if needed to ensure that all needed files can be found. This is true for AutoLISP files, scripts, fonts, images, xrefs, and so on. Also, set the AutoCAD Start In Folder option in ScriptPro to the desired setting.
Layers: for any layer, do the following:

- Make sure it exists.
- Unlock it if you plan to modify objects on it.
- Make sure it's visible if you plan to select objects on it or make it the current layer (on/thawed/vplayer thawed).

System variables: Remember to manage AutoCAD system variables as needed. It helps to be aware of, set and restore the following settings to fit the needs of your particular task:

- If your script does not rely on object snaps, it's generally a good idea to turn object snaps (OSMODE) off during script execution.
- OSMODE is closely tied to the OSNAPCOORD system variable. OSNAPCOORD values include the following:
  0 - Running object snap settings override keyboard coordinate entry
  1 - Keyboard entry overrides object snap settings
  2 - Keyboard entry overrides object snap settings except in scripts
- Turn group selection on or off as needed (pickstyle 1/0). You could end up selecting more than is necessary if group selection is turned on. On the other hand, if your script relies on this feature, it is a good idea to make sure it is on.
- The EXPERT system variable can change the prompting behavior of many commands.
- Most commands will automatically revert to prompting for a file name at the command line when a script is active, but some do not. Set FILEDIA to 0 if the commands you are using do not automatically prompt at the command line when a script is running.
- Be aware of what space (paper or model) you are in. System variables related to this issue include the following:
  CTAB - current layout tab name
  TILEMODE - 1=model space, 0=paper space
  CVPORT - current viewport number. 1=paper space. A value other than 1=model space viewport is active.
- The FONTALT and FONTMAP system variables allow you to specify alternate fonts to use if AutoCAD cannot locate a particular font that is referenced in a drawing. If FONTALT and FONTMAP are not set, this event will cause AutoCAD to stop script execution when such a drawing is opened. Although ScriptPro will not stop execution completely in this event, it will cancel any dialog boxes that are displayed as a result and it will automatically move on to the next available drawing if execution does not resume. It's a good idea to set up a FONTMAP file and set FONTALT. See AutoCAD Help for details.
- ISAVEBAK, if set to 1, will cause AutoCAD to automatically create a BAK (backup file) file each time you save the drawing. It is a very good idea to set this to 1 while doing batch processing of drawings.
- If LIMCHECK is turned on, AutoCAD will not allow you to draw outside the limits specified in LIMMIN and LIMMAX. The default is 0 (off), but it is recommended that you turn it off in your script.

File names: Specify file names in double quotes. Spaces within file names can get interpreted as an ENTER unless the file is enclosed in double quotes (for example: "c:\my project\my cool drawing.dwg").

postScript

Scripts and ScriptPro are extremely easy to learn, use and benefit by. The more you use them, the more powerful they get. Anytime you find yourself bored out of your mind with a process that seems repetitive, redundant, or procedurally similar to other processes, think about scripts as a solution to them. Scripting has been around a long time, a lot of people know how to use them, how to tweak them, how to bypass their "limitations" (the Autodesk forums are invaluable with this aspect of them), and scripting is bar-none the simplest in troduction to customizing AutoCAD. Scripts can be used from within toolbars, tool palettes, pulldown menus and numerous other interfaces. They're incredibly frugal since they're ASCII based… even folks who do not own or use AutoCAD can gt used to creating or modifying them. The technology is fast, portable, and here for the long run. Pay attention to your boredom--- it's your cue for scripting and your key to higher productivity.