

Making AutoCAD Work Faster

(for People Who Don't Want to Learn to Code)

T.J. Meehan AIA, LEED AP

Vice President of Technology Solutions





Who is this person talking to you?

T.J. Meehan AIA, LEED AP

- AutoCAD nerd since 1990 and self-professed “propeller head in training”
- Started my career as an architect (currently president of my local AIA chapter)
- Transitioned to a technology consultant working with architects, engineers, contractors, building owners
- V.P. of Technology Solutions at CADD Microsystems, Inc.
- tj.meehan@caddmicrosystems.com

QUICK POLL: How did you end up in this room?

A)

Every other class
was full

B)

I just followed
someone into the
room

C)

I have no idea...it's
Vegas and I'm still
on A bender since
9pm last night

D)

LEARNING OBJECTIVES

- Understand the simplest way to build your own functions and take them with you
- Learn how to create dozens of time-saving functions in AutoCAD
- Walk away with a library of hundreds of these functions that you can use
- Understand the differences between script files, LISP files, and PGP files



What are we **NOT** going to cover today?

NO COMPLICATED CODING

- Creating anything more complicated than replicating commands on the command line
- No “IF/THEN” statements, no variables, no anything that looks like this guy wrote it:



From the 1993 film, Jurassic Park, from Universal Pictures

NO FANCY EDITORS

- Editing files in anything other than good 'ole Notepad
- No VLISP editor, no Visual Studio

```
<Untitled-1>
(defun C:SLOT ( )
  (setvar "CMDECHO" 0)
  (setvar "BLIPMODE" 0)
  (setq oldsnap (getvar "OSMODE"))

  (setq diam (getdist "\nSlot Diameter : ")
        lngth (getdist "\nSlot Length : "))

  (while
    (setq pt1 (getpoint "\nInsertion point: "))
    (setvar "OSMODE" 0)
    (setq pt2 (polar pt1 0.0 (/ (- lngth diam) 2.0))
          pt3 (polar pt2 (/ pi 2.0) (/ diam 4.0))
          pt4 (polar pt3 pi (- lngth diam))
          pt5 (polar pt4 (* pi 1.5) (/ diam 2.0))
          pt6 (polar pt5 0.0 (- lngth diam)))

    (command "PLINE" pt3 "W" (/ diam 2.0) "" pt4
              "ARC" pt5 "LINE" pt6 "ARC" "CLOSE")
    (setvar "OSMODE" oldsnap)
  );while
  (princ)
);defun
(princ)
```

Image from AfraLISP, www.afralisp.net

File Types



What are the **FILE TYPES** we're going to discuss?



PGP

AutoCAD
Program
Parameters



SCR

AutoCAD Script



LSP

AutoLISP
Application

- You can edit these files with any text editing program, such as Notepad
- You don't need specialized software
- Each has a special purpose (we'll get into this in detail)
- You can take these files with you (we'll also get into this)

Program Parameters (PGP) File





Program Parameters (PGP) File

PURPOSE

- Only need 1 of these files
- Part of every AutoCAD Installation
- Sometimes called the “Pig Pen” file
- Main purpose is to remember the “Key-ins” or “**Aliases**” for the standard commands

Typing in “T” at the command line starts MTEXT command

Typing in “C” at the command line starts CIRCLE command
(even though we all want it to be COPY...more on this later)

```
acad.pgp - Notepad
File Edit Format View Help

; -- Sample aliases for AutoCAD commands --
; These examples include most frequently used commands. NOTE: It is recommended
; that you not make any changes to this section of the PGP file to ensure the
; proper migration of your customizations when you upgrade to the next version of
; AutoCAD. The aliases listed in this section are repeated in the User Custom
; Settings section at the end of this file, which can safely be edited while
; ensuring your changes will successfully migrate.

3A,      *3DARRAY
3DMIRROR, *MIRROR3D
3DNavigate, *3DWALK
3DO,     *3DORBIT
3DP,     *3DPRINT
3DPLOT,  *3DPRINT
3DW,     *3DWALK
3F,      *3DFACE
3M,      *3DMOVE
3P,      *3DPOLY
3R,      *3DROTATE
3S,      *3DSCALE
A,       *ARC
AC,      *BACTION
ADC,     *ADAPTER
AECTOACAD, *-ExportToAutoCAD
AA,      *AREA
AL,      *ALIGN
3AL,     *3DALIGN
AP,      *APPLOAD
APLAY,   *ALLPLAY
AR,      *ARRAY
-AR,     *-ARRAY
ARR,     *ACTRECORD
ARM,     *ACTUSERMESSAGE
-ARM,    *-ACTUSERMESSAGE
```




Program Parameters (**PGP**) File

LOCATION

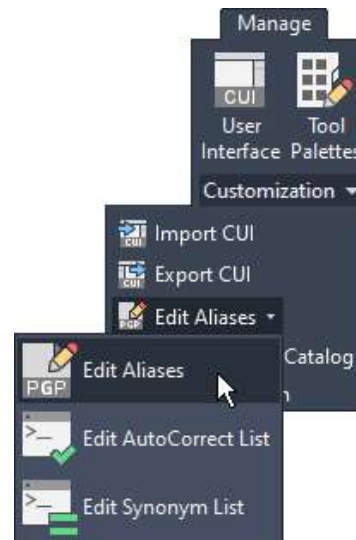
- By default, it is located here:

```
C:\ Users \ USERNAME \ AppData \ Roaming \ Autodesk \ ACA 2020 \ enu \ Support
```

Replace with
your username

Replace with
your product

- You can also access it from within AutoCAD from the **Manage** tab on the ribbon

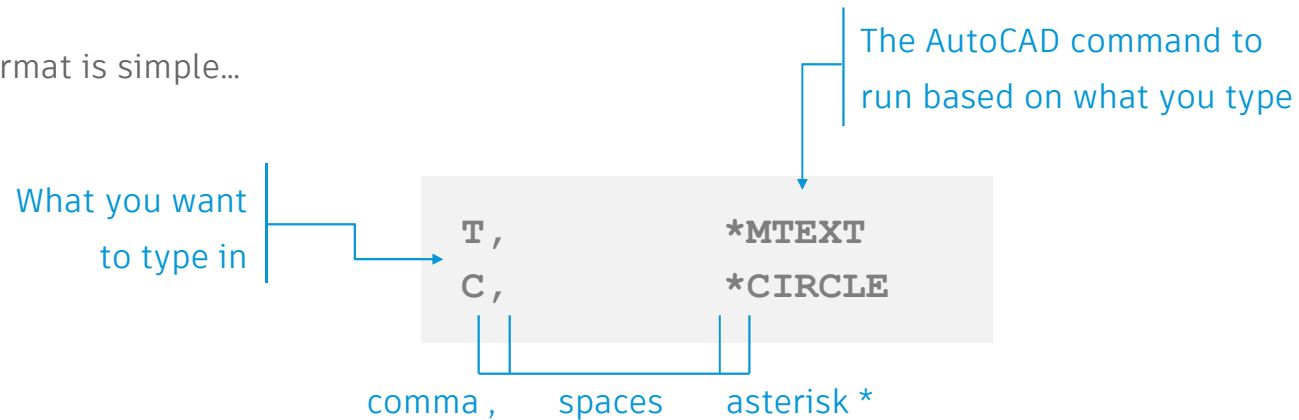




Program Parameters (**PGP**) File

FORMAT

- The format is simple...



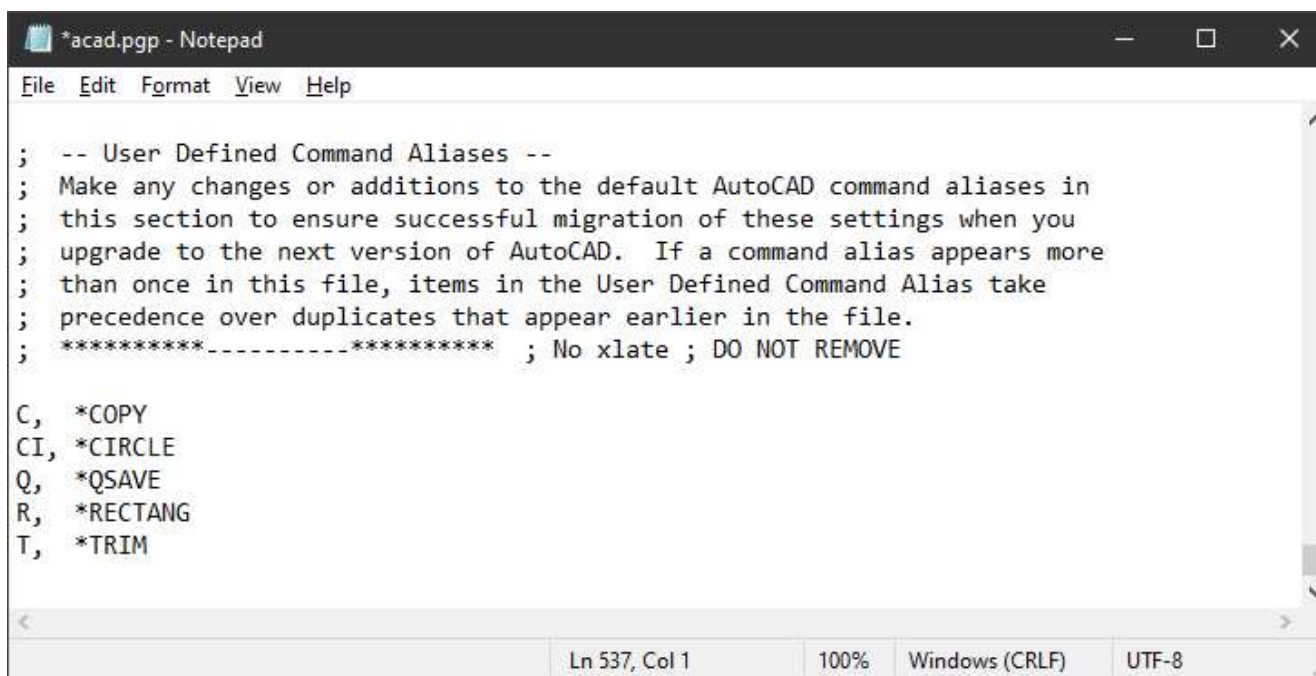
- A semicolon ; denotes a “remark” where you can put in notes
- The number of spaces between the comma , and the asterisk * does not matter (you can use TABs to help)



Program Parameters (**PGP**) File

HOW TO USE

- There are instructions at the bottom of the PGP file...

A screenshot of a Notepad window titled '*acad.pgp - Notepad'. The window has a menu bar with 'File', 'Edit', 'Format', 'View', and 'Help'. The text inside the window is as follows:

```
; -- User Defined Command Aliases --  
; Make any changes or additions to the default AutoCAD command aliases in  
; this section to ensure successful migration of these settings when you  
; upgrade to the next version of AutoCAD. If a command alias appears more  
; than once in this file, items in the User Defined Command Alias take  
; precedence over duplicates that appear earlier in the file.  
; *****-----***** ; No xlate ; DO NOT REMOVE  
  
C, *COPY  
CI, *CIRCLE  
Q, *QSAVE  
R, *RECTANG  
T, *TRIM
```

The status bar at the bottom shows 'Ln 537, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.



Program Parameters (**PGP**) File

TIPS / BEST PRACTICES

1. Only one PGP file can load in AutoCAD
2. Place the **path to your customization folder** in the Support File Search Path higher in the list than the default location to ensure your PGP file loads first (more on this later)
3. I suggest you **don't edit the Acad.pgp file that gets installed by default**. Instead, make a copy and place it in your customization folder. There are two reasons for this:
 - 1) the default installation folder is not usually backed up so you could lose your edits
 - 2) having your own folder makes it easier to manage
4. If you are not sure which PGP file is loading, you can type following at the command line to know the path...

```
(findfile "acad.pgp")
```

The result will add some extra slashes, but you'll know which file is loading...



Script (SCR) Files





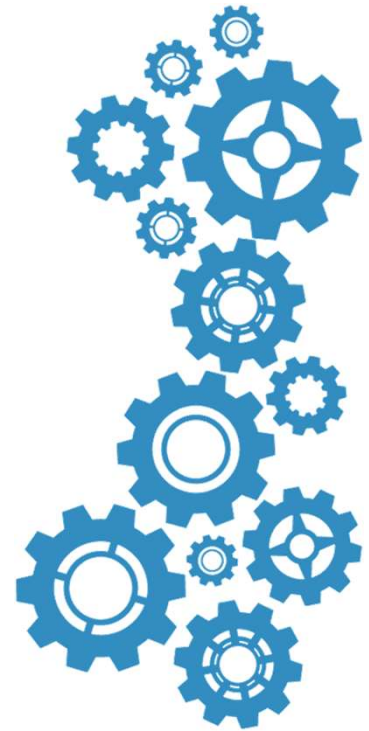
Script (SCR) Files

PURPOSE

- Ideal for **saving a series of AutoCAD commands** to a text file to run them all at once
- Examples:
 - Creating all your standard layers in a drawing
 - File cleanup steps (set Model Space current, Purge, Audit, Zoom Extents, etc.)
- You can **build as many as you want** and share them
- Script files **cannot access dialog boxes, toolbar buttons, palettes, etc.**
They are command line syntax only!

LOCATION

- Script files can be saved anywhere
- Best to centrally locate them for sharing





Script (SCR) Files

FORMAT

- Simply text files you can edit in Notepad
- Format is to mimic what you would type at the command line...

COMMAND LINE

```
Command: -LAYER
Current layer: "0"
Enter an option [?/Make/Set/New/Rename/ON/OFF/Color/
Ltype/LWeight/Transparency/MATerial/Plot/PStyle/Freeze/
Thaw/LOck/Unlock/stAte/Description/rEconcile/Xref]:
make
Enter name for new layer (becomes the current layer)
<0>: A-Wall
Enter an option [?/Make/Set/New/Rename/ON/OFF/Color/
Ltype/LWeight/Transparency/MATerial/Plot/PStyle/Freeze/
Thaw/LOck/Unlock/stAte/Description/rEconcile/Xref]:
color
New color [Truecolor/Colorbook] : 4
Enter name list of layer(s) for color 4 (cyan) <A-
Wall>:
Enter an option [?/Make/Set/New/Rename/ON/OFF/Color/
Ltype/LWeight/Transparency/MATerial/Plot/PStyle/Freeze/
Thaw/LOck/Unlock/stAte/Description/rEconcile/Xref]:
ltype
Enter loaded linetype name or [?] <Continuous>:
Continuous
Enter name list of layer(s) for linetype "Continuous"
<A-Wall>:
Enter an option [?/Make/Set/New/Rename/ON/OFF/Color/
Ltype/LWeight/Transparency/MATerial/Plot/PStyle/Freeze/
Thaw/LOck/Unlock/stAte/Description/rEconcile/Xref]:
```

SCRIPT FILE

```
Layer Creation Script Example - I...
File Edit Format View Help
-layer
make
A-Wall
color
4

ltype
Continuous

Ln 10, Co 100% Windows (CRLF) UTF-8
```




Script (SCR) Files

FORMAT

- Just like the command line, an **ENTER** or a **SPACE** count the same
- Notice the **three line returns at the end** to account for the three ENTERs you need to finish the command
- You'll also notice that the **-layer** command starts with a "-" (dash/hyphen) at the beginning. That's the universal designation to force a command to not use its dialog box but, instead, run completely from the command line. There are many popular commands that support this, including:
 -LAYER -PLOT -BLOCK -PURGE And many others

SCRIPT FILE

```
File Edit Format View Help
-layer
make
A-Wall
color
4

ltype
Continuous

Ln 10, Co 100% Windows (CRLF) UTF-8
```



Script (**SCR**) Files

HOW TO USE

1. Type **SCRIPT** (or even SCR, which is the alias for it)
2. Click the “**Run Script**” button on the “Manage” ribbon



Both methods will open a browse window for you to navigate to your script file



Script (SCR) Files

TIPS / BEST PRACTICES

1. Script files are **not version specific**, but occasionally command options or even the entire commands change...

Sometimes it is just the order of the options in the command

Make sure you test on new versions of software before deploying

2. Careful of **extra lines/rows in the script file** as each is considered an ENTER (and will repeat your last command)

3. If you open your file in Microsoft Word, you can use the **show/hide non-printing marks** button to find them.



4. Since SPACE is considered an ENTER, if you have to type something in that has a space in it (like a layer name), **enclose it in quotes**. For example...

```
-layer make "Arch Walls" color 4 ltype continuous
```



Script (SCR) Files

TIPS / BEST PRACTICES

5. You don't need to use a separate line for every piece of text. You can keep the entire command and all its options in one line, utilizing SPACE's instead of ENTER's. For example, this script file...

...could instead be written this way...

```
-layer make A-Wall color 4 ltype continuous
```

Often, people will put an entire command and its options on one line in the script file this way to save space and better see all the commands that are running.

(Imagine if you were creating 100 layers how long the script file could be if you didn't place each command on a single line)

```
-layer  
make  
A-Wall  
color  
4  
  
Ltype  
continuous
```

AutoLISP (LSP) Files





AutoLISP (**LSP**) Files

INTRODUCTION

- Even though AutoLISP is a coding language, **DON'T PANIC**
- We are only going to cover it related to replicating a series of commands that you would type at the command line

A VERY BRIEF HISTORY

- AutoLISP is a programming language used in some of the very first releases of AutoCAD (circa 1986)
- It's a modified version of earlier LISP and XLISP programming languages, both of which designed for “list processing”
- AutoLISP continues to be supported today in all AutoCAD-based products.



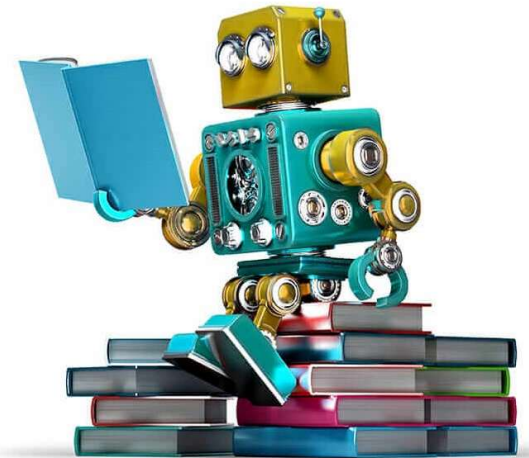
From the 2011 film, The Muppets, from Walt Disney Pictures



AutoLISP (**LSP**) Files

PURPOSE

- Extremely powerful and can dig deep into entities and the DWG file overall
- Includes its own User Interface (UI) called Dialog Control Language (DCL)
- You can write almost any function in AutoCAD via AutoLISP
 - Gathering user input
 - Querying the entities in the drawing
 - Performing if/then statements and condition loops
- The original LISP language is even the bases for some Artificial Intelligence (AI) applications today





AutoLISP (**LSP**) Files

FORMAT

- Just like PGP and SCR files, an LSP file is **just a text file** that can be edited in Notepad
- Instead of a series of commands that all run at the same time (like a SCR file), LSP files **define a bunch of functions** that you run by typing in a command string that you choose
- For an LSP file, the command string looks like this...

```
(command "-layer" "make" "A-Wall" "color" "4" "" "ltype" "Continuous" "" "")
```

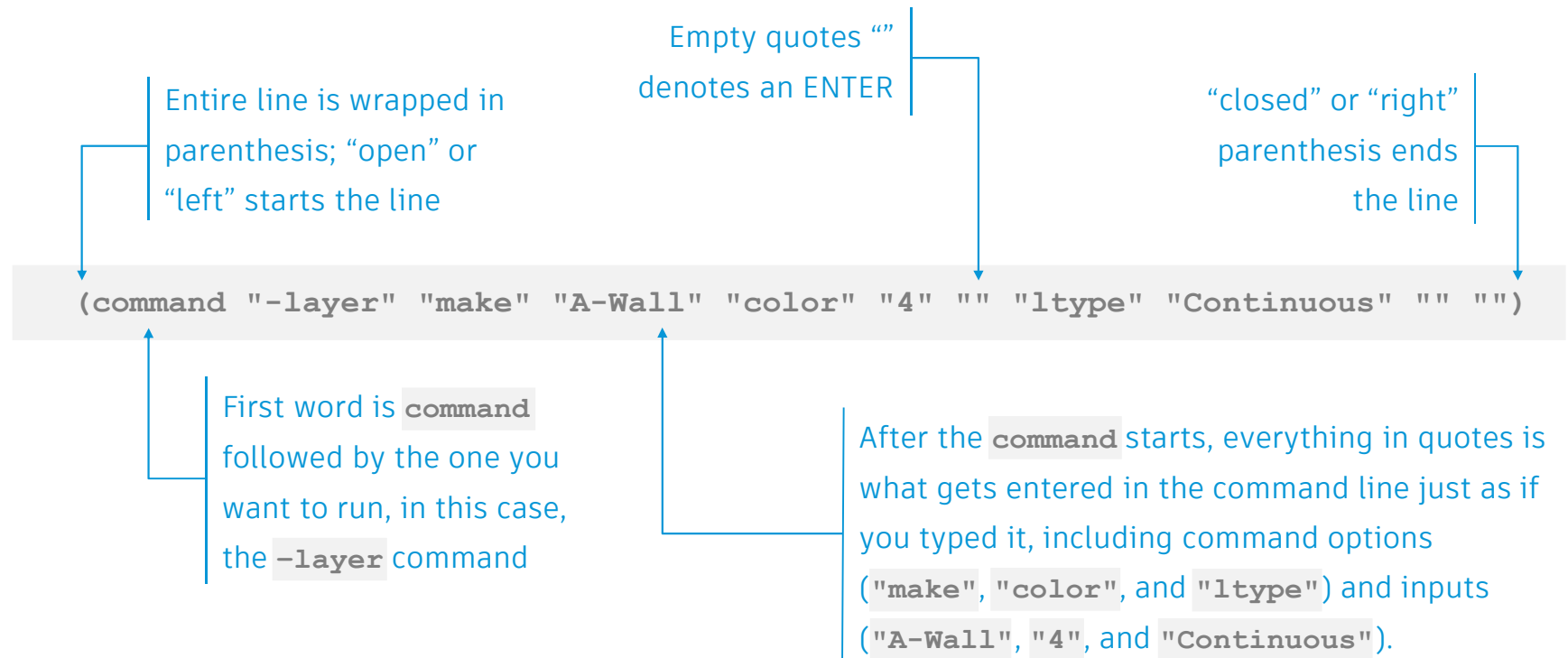
For a SCR file, the same command string looks like this (don't forget the two extra spaces at the end)...

```
-layer make A-Wall color 4 ltype continuous
```




AutoLISP (LSP) Files

FORMAT





AutoLISP (LSP) Files

FORMAT

- The **amount of spaces between the quoted strings does not matter** and you can use that to keep your files clean and readable. For example, if you were going to create several layers, you may format it this way:

```
(command "-layer" "make" "A-Wall"      "color" "4" "" "")  
(command "-layer" "make" "A-Wall-Patt" "color" "8" "" "")  
(command "-layer" "make" "A-Wall-D"    "color" "1" "" "")
```

- One of the great things about the AutoLISP format is that it is **very easy to see exactly what is being typed** in at the command line, including the ENTER's.
- You can **run this format in your script files**, which is what I do to help maintain a clean look.



AutoLISP (**LSP**) Files

HOW TO USE

LSP file need to be **loaded in AutoCAD**. There are several ways to do this:

- 1 Using the command line by typing `(load "MyLISPfunction.lsp")`
- 2 Using the **APpload** command, either every time or once and adding to the [Startup Suite](#)
- 3 Using one AutoLISP file to load another
(you can have a LSP file that simply loads all your other ones, that way you can manage 1 file)
- 4 Use some built-in functionality with AutoCAD...as we will see in the next few slides

Deeper Dive





LSP Deeper Dive

BASICS OF DEFINING A FUNCTION

- We've already discussed how you can write commands in LSP
- Now, let's bundle more than one command together into a function
- Our example is:

Zoom to the extents of the drawing

Zoom out 10%

Do so with only typing "ZZ" at the command line

- You already know the first 2 parts...

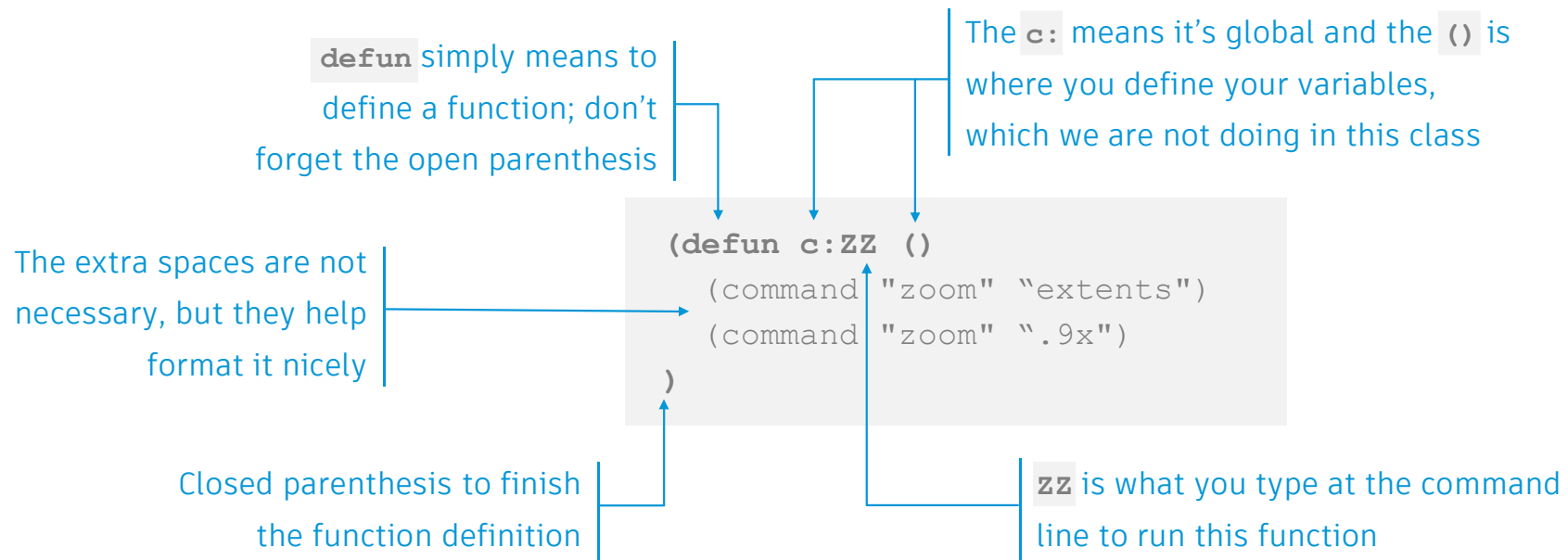
```
(command "zoom" "extents")  
(command "zoom" ".9x")
```



LSP Deeper Dive

BASICS OF DEFINING A FUNCTION

- To bundle these together, we add a start and an end...

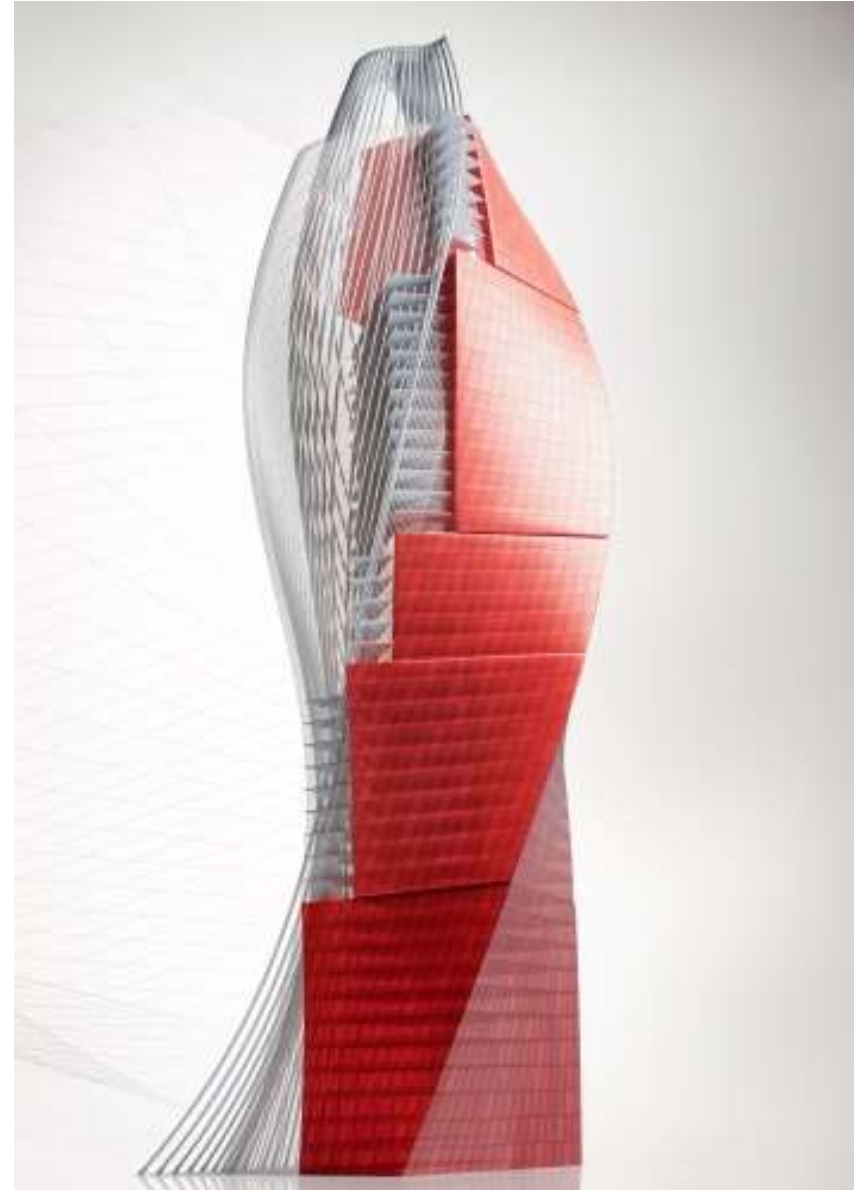




LSP Deeper Dive

ACAD.LSP

- Every installation of AutoCAD looks for a file called **Acad.lsp**
- But this file is **not installed** with AutoCAD
- We will take advantage of this by creating our own and putting all our LSP functions in it
- More on this later



Tips / Best Practices





LSP Tips / Best Practices

SEMICOLONS

- Any text on a line written after a semicolon ; is ignored
- Used to create notes / remarks
- **You cannot have too many of these in your LSP files!**
- For example:

```
(command "zoom" "extents") ;Zooms to Extents
```

TESTING VIA THE COMMAND LINE

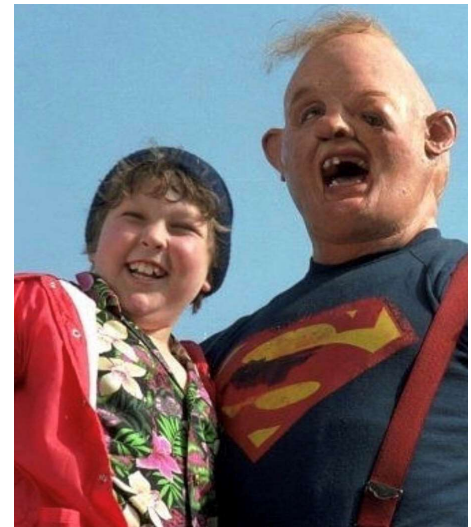
- Always test your functions on the command line
- You can type in commands directly into the command line
- Some commands change their options depending on what you choose (e.g. **-PLOT**)



LSP Tips / Best Practices

TESTING IN CHUNKS

- If you have an AutoLISP function that you built that you just can't seem to get to work, test it in chunks. You can do this by either:
 1. Copying and pasting parts of your code into the command line to make sure they work as expected
 2. Add semicolons to remark out some lines of your code, slowly removing them to narrow down which line has the problem



From the 1985 film, The Goonies, from Warner Bros.



LSP Tips / Best Practices

LETTER CASE

- AutoLISP is not case sensitive
- Best to use upper and lower case consistently
- I tend to use upper case for function names and input values, lower case for every else
- For example, LSP to create standard revision layers...

```
(defun c:REVLAYERS ()  
  (command "-layer" "make" "A-ANNO-REV1" "ltype" "CONTINUOUS" "" "")  
  (command "-layer" "make" "A-ANNO-REV1" "ltype" "CONTINUOUS" "" "")  
  (command "-layer" "make" "A-ANNO-REV1" "ltype" "CONTINUOUS" "" "")  
)
```



LSP Tips / Best Practices

PERIODS

- Used to force the function to use **native** command, even if undefined

UNDERSCORES

- Used to force the function to use **English** command, even if working in a different language

PAUSE

- If you are using a command that requires someone to pick a point, add the pause text in your function
- For example...

```
(command "._break" pause "first" pause "@")
```



LSP Tips / Best Practices

S::STARTUP

- Special function in the [Acad.lsp](#) that runs any functions as soon as the file is loaded (on drawing open)
- For example...

```
(defun s::startup ()  
  (princ "\n\n...My Custom AutoLISP functions loaded.\n\n")  
  (princ)  
)
```

- This is a great way to know if your [Acad.lsp](#) has loaded
- Notice the `princ` function which sends text to the command line
Or a blank line, if nothing after the function
The `\n` is just a line break

Setting It Up

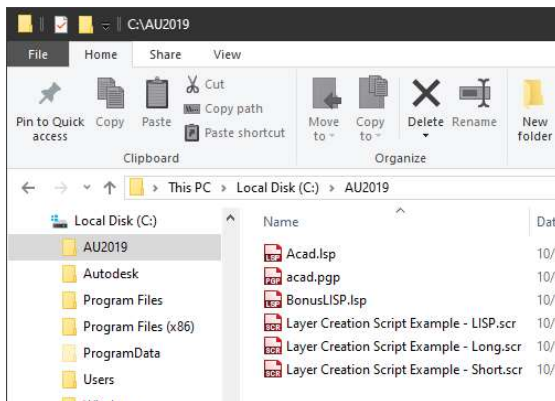


Setting It Up

FOLDERS AND FILES

Create a folder to hold your new custom files:

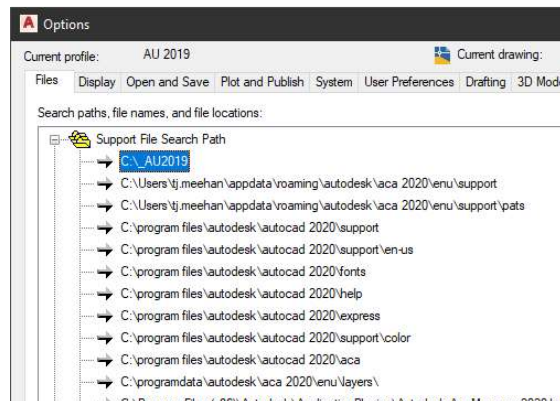
- One [Acad.pgp](#) file
- AutoCAD [Script](#) files
- One [Acad.lsp](#) file



SUPPORT FILE SEARCH PATH

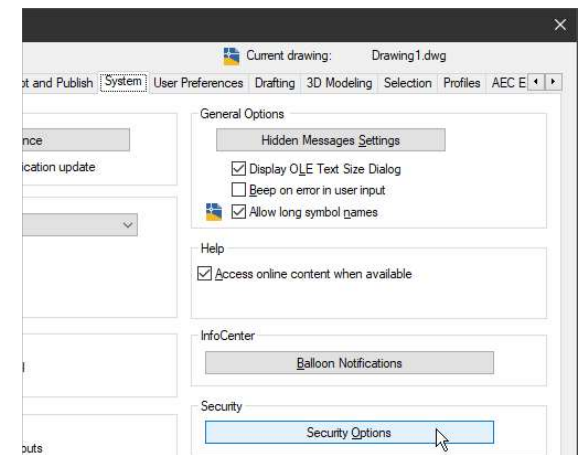
For your custom files to be found, they must be in your AutoCAD [Support File Search Path](#).

Add this in your [OPTIONS](#) and move that folder to the top.



SECURITY OPTIONS

Also in your [OPTIONS](#), you can **“Load acad.lsp when opening each drawing”** from the Security Options button.



Examples



Examples ► Offsets

These functions allow you to offset a preset distance by simply typing in that number. No need to even enter the **OFFSET** command...

- All values **incrementing by 1**, from 1 to 30
- All values **incrementing by 2**, from 32 to 70
- All values **incrementing by 12**, from 72 to 144
- Standard **stud sizes**:
 - 158 for 1 5/8"
 - 35 for 3 1/2"
 - 358 for 3 5/8"
 - 55 for 5 1/2"
 - 558 for 5 5/8"

```
(defun c:1  () (command ". _offset" "1"))
(defun c:2  () (command ". _offset" "2"))
(defun c:3  () (command ". _offset" "3"))
(defun c:4  () (command ". _offset" "4"))
(defun c:5  () (command ". _offset" "5"))
(defun c:6  () (command ". _offset" "6"))
(defun c:7  () (command ". _offset" "7"))
(defun c:8  () (command ". _offset" "8"))
(defun c:9  () (command ". _offset" "9"))
(defun c:10 () (command ". _offset" "10"))
(defun c:11 () (command ". _offset" "11"))
(defun c:12 () (command ". _offset" "12"))
(defun c:13 () (command ". _offset" "13"))
(defun c:14 () (command ". _offset" "14"))
(defun c:15 () (command ". _offset" "15"))
(defun c:16 () (command ". _offset" "16"))
(defun c:17 () (command ". _offset" "17"))
(defun c:18 () (command ". _offset" "18"))
```

Examples ► Zoom

ZA	ZOOM all	<code>(defun c:ZA () (command "._zoom" "all"))</code>
ZE	ZOOM extents	<code>(defun c:ZE () (command "._zoom" "extents"))</code>
ZI	ZOOM in 10%	<code>(defun c:ZI () (command "._zoom" "1.1x"))</code>
ZO	ZOOM out 10%	<code>(defun c:ZO () (command "._zoom" ".9x"))</code>
ZP	ZOOM previous	<code>(defun c:ZP () (command "._zoom" "previous"))</code>
ZW	ZOOM window	<code>(defun c:ZW () (command "._zoom" "window"))</code>
Z2	ZOOM to 1/2" scale	<code>(defun c:Z2 () (command "._zoom" "1/24XP"))</code>
Z4	ZOOM to 1/4" scale	<code>(defun c:Z4 () (command "._zoom" "1/48XP"))</code>
Z8	ZOOM to 1/8" scale	<code>(defun c:Z8 () (command "._zoom" "1/96XP"))</code>
Z16	ZOOM to 1/16" scale	<code>(defun c:Z16 () (command "._zoom" "1/192XP"))</code>
Z25	ZOOM to 1:25 scale	<code>(defun c:Z25 () (command "._zoom" "1/25XP"))</code>
Z50	ZOOM to 1:50 scale	<code>(defun c:Z50 () (command "._zoom" "1/50XP"))</code>
Z100	ZOOM to 1:100 scale	<code>(defun c:Z100 () (command "._zoom" "1/100XP"))</code>
Z200	ZOOM to 1:200 scale	<code>(defun c:Z200 () (command "._zoom" "1/200XP"))</code>

My [all-time favorite](#) is zoom to extents and then back out 10%...

```
(defun c:ZZ ()  
  (command "._zoom" "extents")  
  (command "._zoom" ".9x")  
)
```

Examples ► Views

Functions to let you change your **View Direction** by typing in the option...

```
(defun C:TOP      () (command "._-view" "top"))
(defun C:FRONT    () (command "._-view" "front"))
(defun C:LEFT     () (command "._-view" "left"))
(defun C:RIGHT    () (command "._-view" "right"))
(defun C:BACK     () (command "._-view" "back"))
(defun C:SW       () (command "._-view" "swiso"))
(defun C:SE       () (command "._-view" "seiso"))
(defun C:NW       () (command "._-view" "nwisso"))
(defun C:NE       () (command "._-view" "neiso"))
```

Functions to let you change your **View Rotation** by typing in the option...

```
(defun c:RV0      () (command "ucs" "world") (command "plan" "world"))
(defun c:RV45     () (command "ucs" "new" "z" "45" ) (command "plan" "c"))
(defun c:RV90     () (command "ucs" "new" "z" "90" ) (command "plan" "c"))
(defun c:RV135    () (command "ucs" "new" "z" "135") (command "plan" "c"))
(defun c:RV180    () (command "ucs" "new" "z" "180") (command "plan" "c"))
(defun c:RV225    () (command "ucs" "new" "z" "225") (command "plan" "c"))
(defun c:RV270    () (command "ucs" "new" "z" "270") (command "plan" "c"))
(defun c:RV315    () (command "ucs" "new" "z" "315") (command "plan" "c"))
```

Examples ► Layers

LAYER ON ALL

```
(defun C:LOA ()  
  (command "._layer" "on" "*" "")  
)
```

LAYER THAW ALL

```
(defun C:LTA ()  
  (command "._layer" "thaw" "*" "")  
)
```

Examples ► Productivity

BREAK an entity at a point

```
(defun C:BA () (command "._break" pause "first" pause "@"))
```

COPY previous

```
(defun c:CP () (command "._copy" "p" ""))
```

MOVE previous

```
(defun c:MP () (command "._move" "p" ""))
```

Set the **FILLET** radius

```
(defun C:FR () (command "._fillet" "radius"))
```

Set the **FILLET** radius to 0

```
(defun C:F0 () (command "._fillet" "radius" "0"))
```

Examples ► File Maintenance

AUDIT

```
(defun c:AU ()  
  (command "._audit" "yes")  
)
```

PURGE ALL 4 TIMES

```
(defun c:PG ()  
  (repeat 4  
    (command "._-purge" "all" "*" "no")  
    (command "._-purge" "regapps" "*" "no")  
    (command "._-purge" "zero")  
    (command "._-purge" "empty")  
  )  
)
```

SET ALL ENTITIES TO “BYBLOCK”

```
(defun c:BB ()  
  (command "change" "all" "" "properties" "color" "ByBlock" "ltype" "ByBlock" "lweight" "ByBlock" "")  
  (command "._regenall")  
)
```

Wrap Up



What Did We Learn Today?

- 1 The 3 different **types of files** you can use: PGP, SCR, and LSP
- 2 Different ways you can **take advantage** of each of those file types
- 3 How you can **manage your custom copies** of these files and take them with you
- 4 Many **examples** of great (and simple) LISP functions

What's Next?

DOWNLOAD CLASS FILES

You can download the class files from the AU web site, including

- Sample [Acad.pgp](#) file
- Some AutoCAD [Script](#) file examples
- One [Acad.lsp](#) file with all the LSP examples you saw today plus many more!



SEE OTHER CLASSES

I recommend you go see:

- TR322385
[AutoCAD Scripting Extreme with VBA, Excel, and AutoCAD Core Console](#)
- Presented by Michael Best
- San Polo 3503
- 4:30pm-5:30pm today

There are many other sessions at AU for this topic as well



CONNECT WITH ME

Feel free to reach out to me any time with questions you have on the information presented today.

tj.meehan@caddmicrosystems.com

You can also usually find me at our [booth in the Exhibit Hall](#) for CADD Microsystems.



Questions





AUTODESK®

Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.

