Other Programming Issues of AutoCAD

Part of the information included in this section was based upon the reference book by Christian Immler, *AutoCAD 12 Programming*, Abacus, 1993. The book is available at the UVic bookstore.

1. AutoCAD Script Files and Slide Show

AutoCAD is an open system, which means that users can apply different functions of AutoCAD to their own needs to build a custom CAD system. In addition to the AutoLISP, ADS and ARX interactive graphics programming environments, a user can also avoid the time-consuming and repetitive tasks by writing a simple *script* file.

A script file is particularly convenient in generating a slide show, which can be used to quickly replay pre-stored images for either an automated slide show, or an off-line computer animation of moving objects.

An AutoCAD script file looks like a *macro* programming language that allows a user to save a series of commands, and to recall them later. There are two ground rules to creating effective script files:

- Use the keyboard for all inputs.
- Avoid all shortcuts. (do not use the tablet and AutoCAD menus)
- To start a script file, type `SCRIPT` at the "Command:" prompt. The file selection window appears with all `*.SCR` files in the current directory. Select the *script* file that you want to use.

An Example Script File

An example script file, POLYGONS.SCR, that illustrates the difference between POLYGON circumscribed and POLYGON inscribed:

ZOOM W 0,0 50,30	Zoom the window of (0,0) and (50,30)
CIRCLE 10,10 5	Draw a circle at (10,10) with a radius of 5
CHPROP L	Change properties, select the last object.
	Blank line (no other objects should be selected).
C RED	Change the circle's color to red
	Blank line (no other properties should be changed).
COPY L	Copy, Select the last object
	Blank line (no other objects should be selected).
10,10 30,10	Starting point (10,10), target point (30,10)
POLYGON 6 10,10 I 5	Draw six sided polygon
POLYGON 6 30,10 C 5	Draw six sided polygon
TEXT 5,20 1.5 0 Inscribed	Write the text: font: 1.5; angle: 0; at: (5,20) above the circle
TEXT 25,20 1.5 0 Circumscribed	Write the text: font: 1.5; angle: 0; at: (5,20) above the circle
REDRAW	

The drawing generated by POLYGONS.SCR is illustrated in Figure 1.

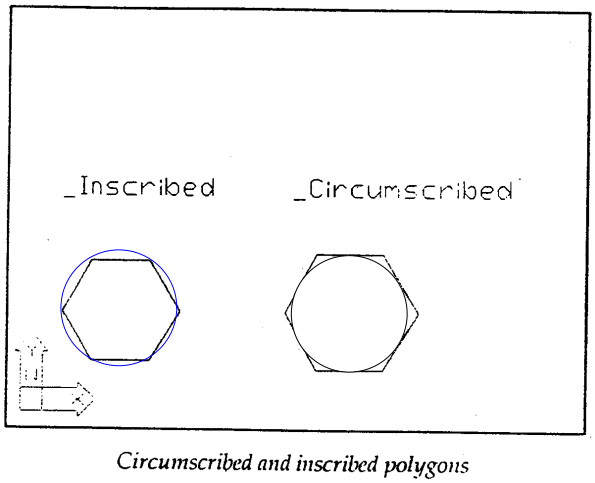

Figure 1. Graphical Output of the Script File, POLYGONS.SCR

Script Files for Time-Consuming Tasks

One of the important applications of a script file is to carry out time-consuming and repetitive tasks in AutoCAD package. Instead of sitting next to the computer and make an entry every few minutes, the script file will carry out the task and keep the computer busy for a few hours. The preparation of slides is a good example of script file application. The slides are created by generating different view projections of a computer model and the model can be shaded or with hidden lines removed. A script file can automatically execute the time-consuming and repetitive task.

First load a processor-intensive 3D drawing. For example, you can use the AutoCAD sample drawing SITE-3D.DWG. Open SITE-3D.DWG, then use the following to set TILEMODE to a value of 1:

```
SETVAR
TILEMODE
1
```

Next, run the ANGLECHG.SCR script file:

```
VPOINT -1,-1,1
HIDE
MSLIDE vo_lf
VPOINT -1,1,1
HIDE
MSLIDE hi_lf
VPOINT 1,1,1
HIDE
MSLIDE hi_rt
VPOINT 1,-1,1
HIDE
```

```
MSLIDE vo_rt
```

This script file generates a slide with hidden lines from four different viewpoints. The whole thing can be comfortably designed using sections. Let's say you have defined the following sections in a drawing:

```
entry
annex
complete
```

These sections indicate zoomed isometric views of the component. Then the following script generates slides of the different sections using the same name:

```
VIEW RESTORE entry
HIDDEN
MSLIDE entry
VIEW RESTORE annex
HIDDEN
MSLIDE annex
VIEW RESTORE comp1ete
HIDDEN
MSLIDE complete
```

This way you can generate many slides from a drawing overnight, and they can then be used in a type of slide show.

Slide Shows

Slide shows can be set up very easily using script files. The individual pictures must be present as AutoCAD slides in *.SLD* format. By repeating the commands *VSLIDE* and *DELAY*, simple script files can be generated as simply as this file, *SHOWSENW.SCR*:

```
VSLIDE south
DELAY 2000
VSLIDE east
DELAY 2000
VSLIDE north
DELAY 2000
VSLIDE west
DELAY 2000
RSCRIPT
```

The loading time of the individual pictures can be reduced if you file the slides in a slide library using *SLIDELIB.EXE*. We start with a text file containing the names of the slide files:

```
south
east
north
west
```

Then we access *SLIDELIB.EXE* using the following syntax:

```
SLIDELIB SNEWSET<SNEW.TXT
```

Now we want to display the slides. The script file *SHOWSLBI.SCR* looks like this:

```
VSLIDE SNEWSET(south)
DELAY 2000
VSLIDE SNEWSET(east)
DELAY 2000
VSLIDE SNEWSET(north)
DELAY 2000
VSLIDE SNEWSET(west)
DELAY 2000
RSCRIPT
```

Although this shortens the loading time, there will be periods of darkness on the monitor before the next picture is displayed, especially for elaborate slides. There is one more way to use the time during which a slide can be viewed on the screen to load the next one into memory and to then be able to quickly display it.

To load a slide into memory, type an asterisk *(*)* preceding the name of the slide in the *VSLIDE* command. *VSLIDE* without a slide name displays the picture. The example *SHOWSLB2.SCR* script file looks like this:

```
VSLIDE SNEWSET(south)
VSLIDE *SNEWSET(east)
DELAY 2000
VSLIDE
VSLIDE *SN:-WSET(north)
DELAY 2000
VSLIDE
VSLIDE *SNEWSET(west)
DELAY 2000
VSLIDE
DELAY 2000
RSCRIPT
```

Now that the slide is in memory, the picture display depends only on the speed of the graphics card. If you have slides that only consist of lines, and no filled surfaces, you can use this to create a flow almost resembling animation. However, the best DELAY times can only be obtained through experiments, since delay depends upon the specific computer and the complexity of the computer model.

2. The Menus of AutoCAD and Their Modifications

AutoCAD has a default menu system that can be replaced or modified by users for their special applications.

AutoCAD Menus

- **Button Menu** – the assignment of buttons on the digitizing device.
- **Screen Menus** – the menus displayed in the right column of the screen.
- **Pull-down Menus** – (or popup menus) menus that are hidden underneath the top line of the screen.
- **Icon Menu** – a menu that uses simple drawings to display various choices that a user can select (e.g. AutoCAD text font selection menu).
- **Tablet Menu** – a menu that allows the tablet to be used as a command/cursor position input device.
- **Cursor Menu** – a menu that allows you to quickly find commands that are used occasionally. It is activated by pressing a special key on the pointing device.

AutoCAD Default Menu System

The default AutoCAD menus are defined by the file *ACAD.MNU*. AutoCAD can only work with one menu file at a time. A user can load in a new menu and use this new menu until it becomes necessary to reload the AutoCAD default menu. A user can also modify the AutoCAD default menu by adding new command lines to the menu file *ACAD.MNU*.

To load a menu, simply type in menu and select the menu file from the window. The menu files have the following types:

.MNU: Menu text file

This file is the source code version of a menu file. *.MNU* files can be edited with any editor or a word processor that saves and loads ASCII text.

.MND: Menu definition for menu compilers

When setting up menus it is much nicer if you don't have to repeatedly retype sections of text which are used several times. Instead, AutoCAD provides a menu compiler, *MC.EXE*, which generates normal *.MNU* files from these definition files.

.MNX: Compiled menu file

This file is a compiled version of an *.MNU* file. AutoCAD compiles the *.MNU* files automatically in an internally readable, compiled format. This should not be confused with the menu compiler previously mentioned.

.MNL: AutoLISP functions

This file defines AutoLISP functions that are used in the menus.

Functions of the MENU File

The MENU command does the following:

AutoCAD searches for an *.MNU* file in the current directory. If none is found, all directories indicated in the environment variable ACAD are searched. If an *.MNU* file is found, the system looks for a corresponding *.MNX* file with the same name. If the *.MNX* file found has a more recent date or the same date, it is loaded. If no file is found, or if only an older one is found, AutoCAD automatically compiles the *.MNU* file. AutoCAD saves the compiled *.MNX* file in the same directory as the *.MNU* file, then loads the *.MNX* file into memory.

If no *.MNU* file is found, the system looks for an *.MNX* file. The order in which the directories are searched is the same as previously described for the *.MNU* files. AutoCAD then loads the *.MNX* file. The result is that the *.MNU* files are actually not really necessary for the application to run. This way, application programmers don't even need to supply the user with source code for an application's menus. The *.MNX* files cannot be read, and cannot be easily reprogrammed. However, this conflicts somewhat with the concept of AutoCAD as an open system.

After the menu has been loaded, AutoCAD searches for an *.MNL* file of the same name, and if found, loads this file as well.

If neither an *.MNU* file nor an *.MNX* file is found, AutoCAD returns an error and requests that a new menu name be selected or entered.

The system does not look for *.MND* files; these must first be converted into *.MNU* files with the menu compiler.

If you want to work without menus temporarily, press the Type it button and, at `the "Menu file name or. for none <acad>:"` prompt, type a period (`.`) and press Return. This turns off all menus. Once you have done this, AutoCAD can only be operated from the keyboard.

Creating an .MNU File

Each menu file can have different sections. The sections start with the following keywords:

```
***Comment
      Comment lines until the next *** header
*** BUTTONS
      Button menu for digitizer or mouse buttons
***AUX1
      Auxiliary function box menu
***POP0
      Pull-down menu; opens up if a button is pressed while the
      crosshair is in position
***POP1
      Pull-down menus, numbered from left to right
```

```
***POP2
***POP3
***POP4
***POP5
***POP6
***POP7
***POP8
***POP9
***POP10

***ICON
      Icon menu
***SCREEN
      Screen menu on the right edge of the screen
***TABLET.1
       Tablet menu areas whose position has to be specified when
       configuring the tablet
***TABLET2
***TABLET3
***TABLET4
```

Below this menu division indicated by the `***keywords`, there is another division with sections which begin with `**headers`. These headers can be selected without limitation.

Within the `***menu` section, the individual menu items are each in their own line. A menu item consists of:

***BUTTONS	A button on the pointing device
***POP0	A line in the cursor menu
POP1…POPn	A line in the pull-down menu
***ICON	An icon in the icon menu
***SCREEN	A line in the screen menu
***TABLE1...4	A field an the tablet

For example, the file *SIMPLE.MNU* looks like this:

```
***SCREEN
LINE
PLINE
CIRCLE
ARC
POLYGON
```

If one of the menu items in the screen menu is selected, this has the same effect as pressing a key and then the Spacebar. So clicking on the word `CIRCLE` serves the same purpose as typing CIRCLE Return on the keyboard.

- Menu item names

In our example the command phrases are the same as the messages displayed in the menu. Normally, though, the series of commands are considerably longer than could be displayed on the

screen. Most graphics cards only allow eight characters. To remain universally compatible, you should not use names longer than this in the ***SCREEN menu area. Besides, with complex menu functions it would be very confusing if the entire command call was always displayed.

For this reason, it is possible in any of the menu sections to place a name in squire bracket preceding the actual command call. A section from the original *ACAD.MNU* file is listed as follows:

```
[PLINE:] ^C^C_PLINE
[POINT:] ^C^C_POINT
[POLYGON:] ^C^C_POLYGON
[SHAPE:] ^C^C_SHAPE
[SKETCH:] ^C^C_SKETCH
[SOLID:] ^C^C_SOLID
[TEXT:] ^C^C_TEXT
[TRACE:] ^C^C_TRACE
[3DFACE:] ^C^C_3DFACE
```

- Commands

To ease working with menus, you should break earlier commands that have not ended before you execute a command in a menu. This way you avoid getting an error message after making a selection from the menu, for example with commands that continuously repeat. Also, when drawing and you change your mind about something, you don't have to use [CANCEL] to end the old command; instead you can immediately execute the new command from the menu.

To do so, add ^C^C immediately after the name in a menu line. This corresponds to pressing Crtl+C on the keyboard, and it breaks old commands. Two times are necessary because with some commands, Crtl+C only jumps back to the last entry and a complete break is only made after the second Crtl+C is executed. When creating the menu file, you do not type Crtl+C; instead, you enter a caret "^" and the letter "C" twice.

- Return Code

To end command input and to execute the command, AutoCAD requires Spacebar or Return to be pressed on the keyboard one time. A space is automatically generated at the end of a menu item. However, sometimes it is necessary to end a command with Return, such as *TEXT*. Other commands require a second Return to completely end, and some text editors do not leave a blank space at the end of a line. In these kinds of situations, you use the semicolon (;). This character is automatically replaced by Return in menu lines.

It also helps make longer menu lines easier to read. Here's a fictitious example of a pair of menu items. Each one sets layers named *EC* and *OG* respectively, turning on both *GRID* and *SNAP*:

```
[EG] ^C^CLAYER S EG; GRID ON; SNAP ON
[OG] ^C^CLAYER S OG; GRID ON; SNAP ON
```

- Separation lines

Screen menus and pull-down menus are easier to read if associated functions are displayed in separate areas. Separation lines for doing this can be inserted with [--]. The separation lines are always displayed across the entire available width in a pull-down menu. This is especially important since the length can vary here. In the screen menu, [---------] should be used instead of [--]; otherwise only two hyphens would be displayed.

```
Menu file:            Screen display:

                    (Screen)         (Pull-down)
[CIRCLE: ]          CIRCLE           |CIRCLE    |
[ARC     ]          ARC              |ARC       |
[LINE:   ]          LINE             |LINE      |
[--]                --               |----------|
[COPY    ]          COPY             |COPY      |
[MOVE    ]          MOVE             |MOVE      |
```

Input During a Menu Command

Sometimes when you are accessing a menu item, you want to add parameters or coordinates from the keyboard or from digitizer devices. To do so, simply add a backslash (\) to the menu file line where the input is needed. This is easy to see in the following section out of the original *ACAD.MNU* (screen menu ARC):

1. [ARC:] ^C^CARC
2.
3. [3~point:] ^C^C_ARC \ \DRAG
4. [S,C,E:] ^C^C_ARC _C \DRAG
5. [S,C,A:] ^C^C_ARC _C _A DRAG
6. [S,C,L:] ^C^C_ARC _C _L DRAG
7. [S,E,A:] ^C^C_ARC _E _A
8. [S,E,R:] ^C^C_ARC \ E _R
9. [S,E,D:] ^C^C_ARC ~E _D DRAG
10. [C,S,E:] ^C^C_ARC _C \ \ DRAG
11. [C,S,A:] ^C^C_ARC _C \ _A DRAG
12. [C,S,L:] ^C^C_ARC _C \ _L DRAD
13. [CONTIN:] ^C^C_ARC ; DRAG

The line numbers are added for the following explanations:

1. Menu title line and the default method for executing the command without options.
2. Blank line.
3. The ARC command usually defaults to 3-point mode, so that the first and second point can be selected with two consecutive entries, then the DRAG option is used to mark the third point and graphically depict the form the arc will take. The system variable DRAGMODE determines whether the word DRAG is even needed here. No "\" is needed to enter the third point, since the menu line ends after this.

4. [S,C,E] refers to Start point, Center, End point. The first "\" for start point input follows the word ARC, followed by the _C (_Center) option. The third option, which specifies the end point, can be input directly by DRAG.

5-9. These options are similar to the ones previously described.

10. This is the first option that does not use the start point default; it uses the center point instead. The start point and end point are dragged values.

11-12. As above, except that an option (angle or length) must be selected for the third point.

13. To connect an arc tangentially to the last line, you must respond to the first prompt with Return. This *is* taken care of here by the ";". The blank space before simply ends the word _ARC, like it does in the other cases. After the ";' only one more point can be input, from which the radius and the arc length are derived.

- Input with the SELECT command

With SELECT, the function does not continue to run until all the desired objects have been selected. This way it *is* possible to select several objects even within a menu item:

```
[Se1-Move]^C^CSELECT \MOVE P;;0,0 @-1,0
```

This menu item pushes all selected objects one unit to the left.

Screen Menus

The screen menu area is the default menu in AutoCAD. If a menu file does not have any menu areas with *** headers in it, AutoCAD automatically assumes that the default is a screen menu.

```
1.  ***SCREEN
2.
3.  [AutoCAD]^C^CP(ai_rootmenus) ^P
4.  [* * * *]$S=OSNAPB
5.  [ASE] ^C^C^P(ai_aseinit_chk) ^P
6.  [BLOCKS] $S=X $S=BL
7.  [DIM:] ^C^C_DIM
8.  [DISPLAY] $S=X $S=DS
9.  [DRAW]$S=X $S=DR
10. [EDIT]$S=X $S=ED
11. [INQUIRY] $S=X $S=INQ
12. [LAYER...] $S=LAYER '_DDLMODES
13. [MODEL] $S-x .$S=SOLIDS
14. [MVIEW) $S=MVIEW
15. [PLOT...] ^C^C_PLOT
16. [RENDER] $S=X $S=RENDER
17. [SETTINGS] $S=X $S=SET
18. [SURFACES] $S=X $S=3D
19. [UCS:] ^C^CUCS
20. [UTILITY] $S=X $S=UT
21.
22. [SAVE:] ^C^C_QSAVE
```

Explanation:

1. Title for screen menu section.

2. Title of the menu page. You should use short, easily remembered names for frequently used menu pages (e.g., "**S for the main screen menu).

3,5. AutoLISP functions from ACAD.MNL are called here.

7., 19., 22.
These lines contain immediate command calls, so colons (:) should be used throughout in the names here. Unfortunately, Autodesk does not use this notation consistently any more in Release 12.

12., 15.
Dialog boxes are called up here. This is indicated in the menu name by the use of the three periods "...".

All the other lines call up further menu pages. Here you can see that several menu pages can be called up one after another. The following example for the CIRCLE command shows how they interact with one another

The circle *is* called up using the following series of menus:

```
**S
.
.
 [DRAW] $S=X $S=DR
.
.
Next menu page:
**DR 3
.
.

[CIRCLE] $S=CIRCLE
.
.
```

What you see on the graphics screen is illustrated below. The "=" sign represents blank lines in the different sections. The "3" in the menu page names means that the new menu doesn't begin until the third line; this way, the first two menu lines are always available. In the first line of the menu, you can display the main menu again at any time and you can restore other settings back to their original status. The second line in the menu brings the Object Snap menu onto the screen. By entering a period here you can use Object Snap. Also, this line displays any tablet menu areas that might have been exchanged. If possible, these two options should always remain available.

Cursor Menus

These cursor menus are handled just like pull-down menus; even the syntax is the same. Just like pull-down menus, cursor menus can contain menu pages that are accessed in the same manner. The cursor menu occupies section ***POP0 of the menu file. You can tell just from the name that there is a relationship to the pull-down menus. If the pull-down menus are disabled in the configuration, the cursor menu won't be available, either.

Cursor menu from the standard *ACAD.MNU:*

```
***POP0

Menu **S     Menu **X 3   Menu **DR 3   Screen display

[AutoCAD ] ----------------------------> [AutoCAD ]
[* * * * ] ----------------------------> [* * * * ]
[ASE     ]   =             [ARC     ] ----> [ARC     ]
[BLOCKS  ]   =             [ATTDEF: ] ----> [ATTDEF: ]
[DIM:    ]   =             [BHATCH: ] ----> [BHATCH: ]
[DISPLAY ]   =             [CIRCLE: ] ----> [CIRCLE: ]
[DRAW]       =             [DONUT:  ] ----> [DONUT:  ]
[EDIT    ]   =             [DTEXT:  ] ----> [DTEXT:  ]
[INQUIRY ]   =             [ELLIPSE:] ----> [ELLIPSE:]
[LAYER...]   =             [HATCH:  ] ----> [HATCH:  ]
[MODEL   ]   =             [INSERT: ] ----> [INSERT: ]
[MVIEW   ]   =             [LINE:   ] ----> [LINE:   ]
[PLOT... ]   =             [MINSERT:] ----> [MINSERT:]
[RENDER  ]   =             [OFFSET: ] ----> [OFFSET: ]
[SETTINGS]   =             =          ----> =
[SURFACES]   =             [next    ] ----> [next    ]
[UCS:    ]   =          ---------------> =
[UTILITY ]   [__LAST__] ---------------> [__LAST__]
=            [  DRAW  ] ---------------> [  DRAW  ]
[SAVE:   ]   [  EDIT  ] ---------------> [  EDIT  ]
=          ------------------------------> =
=          ------------------------------> =
=          ------------------------------> =
=          ------------------------------> =
=          ------------------------------> =
=          ------------------------------> =
=       ---------------------------> =
```

```
[Osnap]
[Center] _center
[Endpoint) _endp
[Insert] _ins
[Intersection] _int
```

[Midpoint] _mid
[Nearest] _nea
[Node] _nod
[Perpendicular] _per
[Quadrant] _qua
[Tangent] _tan

Pull-down Menus

Here is the section ***POP4 from *ACAD.MNU* as an example:

***POP4
[Construct]
[Array] ^C^C_array
[Array 3D] ^C^C3darray
[Copy] ^C^C_copy
[Mirror] C~C_mirror
[Mirror 3D] C~C_mirror3d
[- -]
[Chamfer] ^C^C_chamfer
[Fillet] ^C^C_fillet
[- -]
[Divide] ^C^C_divide
[Measure] ^C^C_measure
[Offset] ^C^C_offset
[- -]
[Block] ^C^C_block