








Point Line Poly

A Technical Friend for PC ARC/INFO® Users
and Other ESRI Desktop Software

Volume 7 Number 3

Inside:

	PC ARC/INFO 3.5.2	1
	Idrisi/Grid Conversion	5
	The Programmer and the Troll	6
	Modifying Table Structures	11
	ARCSHAPE Workarounds	14
	Inserting Text Files	15
	Angled Annotation	16



PC ARC/INFO 3.5.2

Version 3.5.2 of PC A/I, a free upgrade to 3.5.1 owners, has been shipping since this past Fall. Besides being a bug fix, addressing every significant bug that was reported in 3.5.1, it also has a number of enhanced features.

In the 3.5.2 %ARC% directory is a file, WHATSNEW.TXT that describes the updates in a fair amount of detail. By default, a shortcut to it is also added to the program group when first installed. Some of the more significant changes include the following.

Syntax Changes

Two syntax changes important to SML writers have been made. The first is to the BUILD command:

```
BUILD [cover] [POLY / LINE / POINT]
```













The POLY option is no longer a default option; this is to prevent the accidental building of a point coverage as a polygon coverage and nuking its attributes. Because of this, SML developers should check their old applications for BUILD commands and update the syntax if necessary.

An SML bug in existence since 3.5.0 has been fixed. Formerly, &IF would not always properly resolve when numeric strings were involved. Now, using double quotation marks at all will force a string comparison:

```
&goto pass1 &if &ne "1" "1.0"
&type "The first comparison failed in 3.5.1."
&goto test2
&label pass1
&type "The first comparison passed in 3.5.2."
&label test2
&goto pass2 &if &ne "x1" "x1.0"
&type "You should not see this message"
&goto end
&label pass2
&type "The second comparison passed in 3.5.1 and
3.5.2."
&label end
```

(Continued on p. 2)

Icon Key

 Macro	 Good Ideas
 Buried Jewel	 Positions
 Lesson / Tutorial	 Calendar
 Editorial	 Doctor's Column
 Review	 Letters
 Off the Wire	 Announcements

Old code using “x” to force string comparisons need not be rewritten, save perhaps to make it look nicer, but because the &IF bug no longer needs to be kept in mind, new code should be less prone to errors in execution.

New or Updated Functionality

Starter Kit: The CLEAN command now has a LINE option (hooray!)—now, link coverages (line/point topology) may have intersections resolved in one step without nuking the PAT:

```
CLEAN SURVEY # # 0.1 LINE
```

Also, some new digitizer format files have been added (it should be noted that Version 4.0 will support Wintab drivers, thus solving compatibility issues for many users).

Data Conversion: Updates have been made to SHAPEARC, AGFSHAPE, SHAPEAGF, MIFSHAPE, ARCDXF, and DXFARC. TIGERARC itself has not changed, but the online help has been updated to discuss changes made to certain field contents in the 1997 files.

Unfortunately, the ARCSHAPE command in 3.5.2 has a bug, in the form of incompletely written polygon island rings. It is not known at this time if or when a patch will be issued, but two workarounds are discussed on p. 14 of this issue.

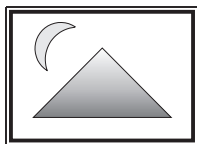
Arcedit: DRAWENVIRONMENT now has a NODE ALL option. (NODECOLOR is listed as a new command, but it was present in 3.5.1.)

Overlay: A weed factor has been added to the BUFFER command to help eliminate spiking due to coordinate resolution problems.

Arcplot: MAPENV.SML has been added to the UTOOL directory, allowing users to save the page environment of map compositions for later restoration.

Windows Extensions

Dialog backgrounds in Win95/98/NT are no longer forced to white, but match the display appearance settings for 3D objects. Although the widgets themselves are not actually drawn any differently, in nonwhite backgrounds the white highlights show up to enhance their “3D” appearance:



POINT LINE POLY is published by Pierssen Publishing, 3125 West Wilson Drive, Flagstaff, AZ USA 86001. Domestic subscriptions: 6 issues \$35. Foreign: 6 issues \$50. Single Issues \$7 domestic, \$10 foreign. Send subscriptions and fulfillment questions to *PLP* Circulation Dept., 3125 West Wilson Drive, Flagstaff, AZ USA 86001. Or call (520) 774-7905. Submit material and technical questions to The Editor, at the address above.

© Copyright 1999, Pierssen Publishing. All rights reserved. *Point Line Poly* is an independently produced publication of Pierssen Publishing.

ISSN: 1099-2324

ESRI, ARC/INFO, PC ARC/INFO, ArcView, and ArcCAD are registered trademarks; Atlas GIS, Data Automation Kit (DAK), Simple Macro Language (SML), and Avenue are trademarks of Environmental Systems Research Institute, Inc. All other company and product names mentioned are property of their respective owners.

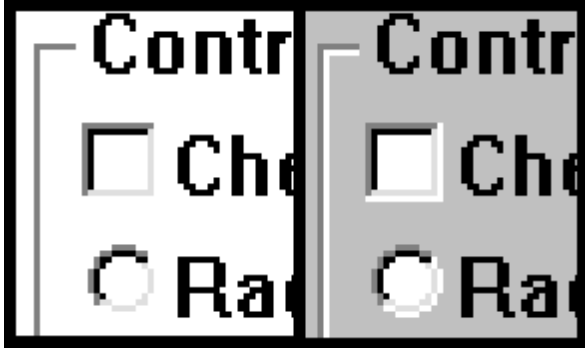
POSTMASTER: Send address changes to: Point Line Poly
Pierssen Publishing e-mail: piersen@primenet.com
3125 West Wilson Drive
Flagstaff, AZ USA 86001

NOTICE: Due to differences in hardware and software configurations, and differing user requirements, information published in *PLP* may or may not be applicable to specific installations and user requirements. To ensure the accuracy of the information published in *PLP*, Pierssen Publishing specifically disclaims responsibility for errors and omissions or the ability of users to implement recommendations published in *PLP*.

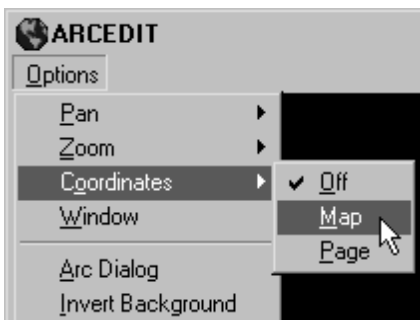
Say, what?

“We only think we occupy a solid, Brick-and-Timber City,— in Reality, we live upon a Map.”

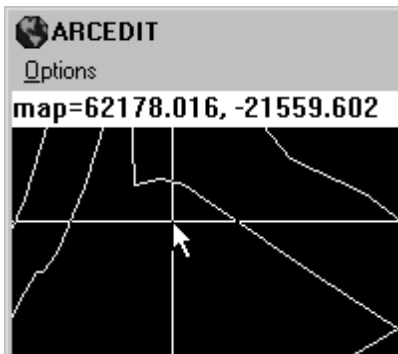
Thomas Pynchon, *Mason & Dixon*



In graphic displays, a “Coordinates” entry has been added to the “Options” menu:



This activates or deactivates coordinate tracking of the cursor during coordinate entry:



A separate display list is now kept for zoom windows with “Auto Update” disabled. Thus, for example, you could draw additional features in the main display without their appearing in the zoom window when it's panned or zoomed.

Additional WIN EXEC commands have been added:

- 18 - Zoom Window with Auto Update Off.
- 19 - Show / Hide Text Window. Returns 0 if the window is hidden and 1 if it is visible.
- 20 - Coordinates Off. Turns Coordinate Tracking Off.

- 21 - Coordinates Map. Turns Coordinate Tracking On using Map Coordinates.
- 22 - Coordinates Page. Turns Coordinate Tracking On using Page Coordinates.

Finally, in ARCPLOTW, the command “TEXTFONT *” is equivalent to “WIN EXEC 9” followed by “TEXTFONT %1”.

WinNT Issues

The NT discussions in the “Frequently Asked Questions” section of the online help are unchanged, but ARCW.BAT has been updated to match the example given.

Although some instability issues may remain in 3.5.2, for Version 4.0 the development team is taking steps to nail it down altogether. For example, all code will be Win32-based, and all file manipulation (renaming, deletion, etc.) will be handled through Win32 calls. Because of this, the “Faster Response” option in Win95/98 will also no longer be needed.

Also, dual installations of PC ARC/INFO and ARC/INFO NT are now possible without conflict.

OEM/ANSI Text Handling

[Note: The examples in this section assume United States settings. Results may differ for international users.]

Perhaps the most significant improvement in 3.5.2 is control over OEM vs. ANSI text handling. For example, the FORMS command dialog has added a radio button:

Field	Type	Value
AREA	13 N	615942710.000000
PERIMETER	13 N	651564.270000
VEG_	11 I	033
VEG_ID	11 I	032
MUNTNUMB	3 I	0290
TYPE	1 I	01
TYPE_TXT	20 C	0Conifer

The windows font numbering system has been modified from that of 3.5.1 to accommodate the

option. The ten thousand digit ("G" in -GSTTT), which formerly specified spacing (0 = hardware, 1 = IGL), has been expanded to the following:

- 0 = No IGL spacing / OEM to ANSI conversion (like 3.5.1)
- 1 = IGL spacing / OEM to ANSI conversion (like 3.5.1)
- 2 = No IGL spacing / No OEM to ANSI conversion (like 3.4.2)
- 3 = IGL spacing / No OEM to ANSI conversion (like 3.4.2)

"OEM to ANSI conversion" means that character values (0-255) are assumed to be OEM, and are converted to the equivalent ANSI value before being printed using the font. If an equivalent ANSI character doesn't exist (e.g. for Greek or line drawing characters), an underscore is substituted. For example, if font family 004 were Arial, character 233 (Greek small theta in OEM, small e with acute accent in ANSI) would appear as follows:

Font -00004: _

Font -20004: é

For application developers, WIN menu and dialog commands have been updated to allow OEM/ANSI character support. By default, menu files are not displayed using OEM to ANSI conversion. Thus, the following menu file composed using OEM characters:

```
POPOP Composers
MENU1 Bartók,      Bartók
MENU1 Duruf lé,    Duruf lé
MENU1 Dvoràk,      Dvoràk
MENU1 Schönberg,   Schönberg
POPEND
```

appears as follows:



and the return value:

```
UAR.  VALUE
%0001 Dvoràk
```

By default, lowercase characters are converted to uppercase in the return value. When the command

line "OEM" is added to beginning of the menu file (it won't work if it's added to the end), the menu appears properly:



and the return value is exactly as specified:

```
UAR.  VALUE
%0001 Dvoràk
```

Note that in neither case is the return value converted from OEM to ANSI. In dialog definition statements, the [pitch] parameter allows specification of ANSI or OEM:

- V = variable (ANSI)
- F = fixed (ANSI)
- VO = variable (OEM)
- FO = fixed (OEM)

Given the following dialog file:

```
BEGIN 0 0.00 0.00 3.00 14.00 v Test
PBUT -51 1.00 1.00 1.50 12.00 v
```

and the following command in OEM characters:

```
$ 51 37° 00' 00"
```

the result appears as follows:



When "v" is changed to "vo":

```
PBUT -51 1.00 1.00 1.50 12.00 vo
```

the result appears as follows:



EDITDLG has not yet been updated to include VO and FO. To update it yourself, change the following line in EDITDLG.R:

```
&write "DROP 988 14.00 27 4.0 5 v"
```



Idrisi/Grid Conversion

Free command line utilities are now available to convert ARC/INFO grids to Idrisi images and vice versa. Because they utilize ArcView's Gridio API, ArcView version 3.0 or greater must be installed and that the BIN32 is in the runtime path. For example, the following statement may be typed at an MS-DOS command line or added to AUTOEXEC.BAT:

```
path =
%path%;c:\esri\av_gis30\arcview\bin32
```

In NT, you also can modify the path variable in the system|environment settings in the control panel. The commands are executed at the MS-DOS command prompt and may be used in batch files:

```
IDR2GRID westboro westboro
```

Any binary Idrisi image may be converted to a grid (however, the row and column resolutions must be

equal), but because integer grids have 32-bit values, not all values may successfully be converted to 16-bit Idrisi values. The utilities may be downloaded at:

<http://www.primenet.com/~piersen/idrisi/grid.htm>

PLP

PLP Online

<http://www.primenet.com/~piersen/PLP>

User Name: plpv7n3

Password: i9gc46u8

to:

```
&write "DROP 988 14.00 27 6.0 5 v"
```

and the following lines (there are two places where they occur):

```
&write "v"
&write "f"
```

to:

```
&write "v"
&write "f"
&write "vo"
&write "fo"
```

then recompile:

```
COMPSML editdlg r editdlg
LINKSML
  CML file:editdlg
  Main routine:editdlg
  Report file:
  LML file:editdlg
  LML file:
```

A new dialog command, T, allows you to switch a widget to OEM or ANSI:

```
T [widget_id] [OEM / ANSI]
or
T [widget_id] [O / A]
```

More information is available in the online help under "Using Extended Characters". There's also a good example of populating a widget with a list of ANSI text and then switching it to return OEM.

Conclusion

PC ARC/INFO keeps growing and improving to meet the needs of its users. Although some functionality may never be implemented (to prevent marketing conflicts with Host ARC/INFO), nonetheless ESRI has shown its continued commitment to this product. The arrival of Version 4.0, I'm sure, will be a much welcomed event.

PLP



The Programmer and the Troll

Peter Aronson, ESRI

~ A Dialogue Explaining the Myth of Fuzzy Creep ~

Open Curtain

(The programmer is about to cross a bridge in the woods, when suddenly, a great, big hairy troll leaps out from under the bridge!)

Troll: Pay the toll or be eaten!

Programmer: What? Huh? Why would you want to do that?

Troll: Because I'm just a big, fuzzy, creep, that's why!

Programmer: But there is no such thing as fuzzy creep, it's a myth!

Troll: And what are Trolls, chopped liver? But you can't fool me, even a poor stupid troll like myself knows what fuzzy creep is.

Programmer: OK, and what's that?

Troll: Why, it's when you run CLEAN on a coverage more than, ah, err (*he stops to quickly counts his fingers and toes*), ah, many, many times; all of the arcs will run like melted wax, because of the fuzzy tolerance. So there!

Programmer: Look buddy, if you CLEAN a *million* times on a coverage, none of the arcs will move after the first time. Do you even know what fuzzy tolerance is?

Troll: Don't get fresh! I could still eat you!

Programmer: No you can't, I'm a programmer!

Troll: So?

Programmer: Where do programmers eat?

Troll: I don't know. Or care.

Programmer: You'd better care, buddy. I'll tell you where programmers eat. *El Burito*, that's where they eat. Day in and day out. Year after year.

Troll: You mean—

Programmer: Exactly! By now, my body is *entirely saturated* in El Burito hot sause. One bite could lead to death!

Troll: Eee-yuch! You're right, I can't eat you. But I could still tear you to little itsy, bitsy pieces if I feel like it. And I will unless you explain to me just what fuzzy tolerance is.

Programmer: Don't push your luck, bridge-breath, I've killed dragons with my bare hands. But since I *love* talking about computer stuff, I'll take the time to enlighten you. Assuming that that's possible.

OK, the most important concept behind fuzzy tolerance is that of *equality*.

Troll: You mean like "All Trolls were created equal"?

Programmer: You can lead a Troll to water, but you can't make him think. No, more like this:

(*He bends over and writes something on the surface the bridge with a piece of chalk*)

There! What does that say?

Troll: It says " $1 = 1$ ".

Programmer: Would you say that is a true statement?

Troll: Of course! What do you take me for?

Programmer: Let's not get into that. OK, what about this:

(*Writes something else on the bridge*)

Troll: It says: “ $1 \diamond 2$ ”. That’s obvious!

Programmer: You’ve never tried to program a PRIME 2250, have you? But yes, one does not equal two. OK, next: *(Writes something longer on the bridge.)*

Troll: “ $P(1,1)=P(1,1)$ ”? Oh, I see! You mean a point with coordinate X equals one, Y equals one, is equal to another point with the same coordinates!

Programmer: Hey, that’s, pretty smart of you. I don’t know why they say you’re dumb. Anyway, if you’ve figured that out, this should be pretty easy to figure out, too.

(Draws some more on the bridge.)

Troll: “ $P(1,1) \diamond P(2,2)$ ”. That’s true, too. So what?

Programmer: Just be patient. This next one’s critical. We’ll go back single numbers for the moment: they’re easier to draw.

(He draws something longer on the bridge)

What do you think of this?

Troll: “ $1.2345678 = 1.2345679$ ”? That’s not right! Those numbers are different!

Programmer: To you or me yes. But to a computer, maybe not. You see, numbers in computers are often stored in what is called *single precision floating point format*. Floating point because the decimal point can “float” around in the number, single precision because it contains half as much information as double precision.

Troll: How much information does double precision contain?

Programmer: Twice as much as single precision. Don’t interrupt. Anyway, single precision floating point numbers, or, single for short, usually contain 32 bits of binary data. In the normal day-to-day terms of decimal numbers, that’s about 6 to 7 (depending on format) decimal places of information. And if you look at our little example, you can see that coordinates are identical for their first 7 decimal places. It’s in that tricky 8th place that they’re different. So, as far as the computer can tell, the numbers are the same, and the points are identical.

Troll: Could the computer tell the points apart if those numbers were stored in double precision floating point numbers?

Programmer: Another good question! The answer is yes in this case, but not in all cases. You see, double precision floating point numbers (let’s call them doubles for short), store twice as much information as singles, which means 64 bits on binary data, or 14 decimal places. Now, what if we had two points like these:

(He draws something even longer on the bridge.)

Troll: *(Looking slightly overwhelmed.)* Ah, “ $1.23456789012345 = 1.23456789012346$ ”? That’s a pretty long number! But they’re still different. Oh, the computer can’t tell that, can it?

Programmer: Very good!

Troll: And by the way, who says I’m dumb!?

Programmer: Oh, just people. Don’t worry about it. Now as even you can see, the concept of equality is different to a computer than it is to us. The computer can only see so many decimal places: the rest are invisible to it.

Troll: But so what? Those differences you just showed me are very small. Who cares about them?

Programmer: Well, that’s true in the examples I’ve shown you so far, but not in all cases. What do you know about UTM coordinates?

Troll: Universal Troll Mercator? Well, they’re in meters.

Programmer: *(Muttering)* Universal Troll Mercator? *(Speaking)* And what else?

Troll: Ah, they’re usually large?

Programmer: Right! Now assume we’re looking at the USGS’s Grants, New Mexico, Quad sheet. Its corners are:

Stoops to draw on the bridge again)

SW	261530.93	3917597.26
NW	261897.75	3931460.84
NE	273237.61	3931167.03
SE	272886.02	3917303.06

Now, let's look at those Y values. We have a minimum Y value of 3917303.06 and a maximum Y value of 39314060.84. Now those numbers have 7 decimal places in front of the decimal place. Now, assuming we're using single precision numbers, any difference between two Y coordinates that is smaller than 1 meter is invisible to the computer. A whole meter!

Troll: But, if the numbers are so large, what does on piddly little meter matter?

Programmer: You'd be surprised. If you subtract the minimum Y from the maximum Y above, like so:

(He looks around the bridge for a clean spot, then erases an area and draws on it.)

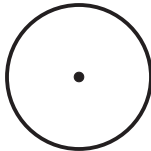
$$\Delta y = Y_{\max} - Y_{\min}$$

We find that $39314060.84 - 3917303.06 = 14157.78$. Now 14,000 meters is still pretty large, but still, a meter is noticeable there. And it could be more noticeable. If we were to CLIP out a smaller part of the data set, say one 1400 by 1400 meters, why then a meter would be pretty important, wouldn't you say?

Troll: I guess so. But what does all this have to do with fuzzy tolerance?

Programmer: I'm getting to that. Now that we've established that there's a limit to how well the computer can tell numbers apart, what does it mean? Well, to begin with, it means that *any* program that manipulates numbers on the computer has a limited resolution. If those numbers happen to be X's and Y's (or Lat's and Long's), then it means that there is a limited area that a computer can resolve a point's location down to. You can imagine each point surrounded by a zone, or halo. Like this:

(He draws a circle with a dot in the centre on the bridge.)

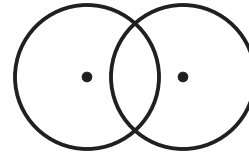


As far as the computer can tell, that point could be *anywhere* in that circle.

Troll: So what?

Programmer: Well, what about this situation, when you have two points very close to each other, like this?

(He draws two overlapping circles with dots at their centres.)



Look at these two points. Since their possible values overlap, the computer can't consistently tell if they are the same point or two different points. Sometimes it will decide one, and sometimes it will decide the other. This sort of problem occurs in *every computer program that manipulates numbers!*

Troll: Then how do those problems ever work?

Programmer: Sometimes they don't. Sometimes it doesn't make a difference. And sometimes the programmer works around it by introducing a tiny fudge factor that he or she uses to insure that they get constant results.

Troll: How do they do that?

Programmer: Basically, by redefining the concept of equality. You remember equality, don't you? We just discussed it a few minutes ago.

Troll: Oh, yeh. Sure. Of course.

Programmer: Right. Well, what happens is that the concept of equality is made looser, of fuzzy (a use of the term borrowed from fuzzy logic and fuzzy sets). Instead of tests like:

```
if A = B
```

The programmer uses tests like:

```
if A >= (B - fudge factor) and A <= (B + fudge factor)
```

In the past, the programmer generally either selected a fudge factor that he or she knew made sense based on the data, or calculated one based on the data ranges, or just took a wild guess and hoped. The first GIS system that allowed the user to actually choose this number themselves was ODYSSEY, from the Harvard Lab for Computer Graphics and Spatial Analysis. The second was ARC/INFO.

Troll: You mean, the *fuzzy tolerance* is nothing more than a fudge factor!? I thought it had to do with snapping gaps, removing slivers, and stuff like that!

Programmer: I wouldn't say "nothing" more than a fudge factor; equality is a subtle and important concept. And there's a strong connection between that concept and the use of fuzzy tolerance to close gaps and collapse slivers. We've talked about the computer's limits of resolution, but we haven't talked about the data's resolution yet.

Troll: The data's resolution?

Programmer: Is there an echo in here? The data's resolution depends on a whole series of things, but we'll discuss only two of them. First, there's the resolution of the data source. Let's assume it was a paper map. That resolution is affected by a number of factors; how small were the smallest features included during compilation, what was the narrowest pen available to the cartographer, how much does ink smear during the printing process? Second, there's digitising cross-hairs? How good was the operator's eyesight; how small could they see? It all adds up, and it always gets worse, not better. If the data set goes through five different processes that affect the resolution, you end up with the worst of the lot.

Troll: So fuzzy tolerance should be set to the resolution.

Programmer: It can be. Then, CLEAN will remove all features smaller than the resolution, since none of those features can really be said to exist: they're all "noise" introduced into the data by the various operations it's undergone.

Troll: But the gaps and slivers you see in *real* data are much larger than resolution. Even in *my* shop. I mean, even with me standing behind them, ready to eat any of them that screw up, my goblin digitising slaves still manage to slip in all sorts of errors.

Programmer: (*Appalled*) You *eat* them?

Troll: Hey, don't look at me like that. It's not like I want to eat them (Have you ever tasted goblin? Bleh?) but it's my job.

Programmer: You must work for a tough shop. A *really* tough shop.

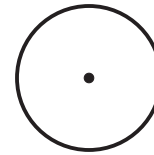
Troll: Well, my boss is a real Ogre. A real Ogre, get it?

Programmer: Got it.

Troll: Good.

Programmer: Back to fuzzy tolerance. You brought up a good point: in addition to resolution, data has *error*. Error can be informally defined as uncertainty in the true value of a measurement. Like resolution, you can represent error for a point as a zone around the point:

(*He draws a circle with a dot in the centre of the bridge.*)



(Actually, surveyors like to represent error as an ellipse rather than a circle, because traverses produce error that is uneven in direction.) Map data picks up error in every step processing. When it's first collected, there's error. When it's made into a map, there's error. When it's digitised, there's error. And worse, unlike resolution, error doesn't simply take the worst value: it is additive! That is, you add all of your error up. It *never* gets any better.

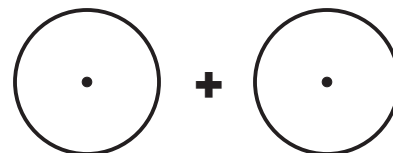
Troll: What, never?

Programmer: No, Never!

Troll: What, never?

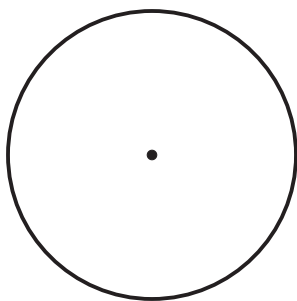
Programmer: Sorry, this is not Gilbert and Sullivan. The answer is still "no, never". So, if you subject data to two different proses, both of which add four meters of error:

(*He draws two circles with dots in the centre on the bridge.*)



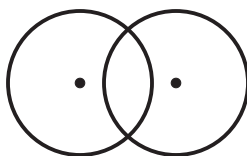
The data will end up with 8 meters of error, like so:

(*He draws again*)



The real location of the point could be anywhere inside the circle. So, if we have a case where two points are within their accumulated error of each other, like this:

(He draws two overlapping circles with dots at their centres.)

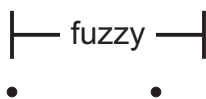


The two points can be considered to be equal. And remember fuzzy tolerance is used to adjust equality!

Troll: How does that apply to gaps and slivers?

Programmer: Very nicely, thank you. You see, gaps and slivers are, as you noted earlier, errors. They should be smaller than your estimated total error for the data set. Since two points within error distance of each other are equal, they are really the same point. What CLEAN does is take all of the points that are effectively the same as another point (within error distance) and make 'em *really* the same points. So, if we have a gap that's smaller than fuzzy tolerance, like this:

(He draws a diagram on the bridge.)



Then, all that really happens is that the two points are made really equal as well as effectively equal, like so:

(He redraws the diagram slightly)



Slivers work much the same way: all of the points on one side of the sliver are made identical with all of the points on the other side, and, "poof!", the sliver is gone.

Troll: *(Starts dancing around the bridge)* I see. I see! I see!! *That's* what fuzzy tolerance is all about! *(Stops suddenly)* But you still haven't explained to me why fuzzy creep is a myth.

Programmer: I was going to explain that, when you started your John Travolta imitation. You needed to know how fuzzy tolerance worked first. OK, what fuzzy processing does is to make points that are effectively the same actually the same. That's causes the arcs that contain those points to move. What people are afraid of is that those arcs will *keep on* moving, each time the coverage is processed. However, if points are already actually equal, then there is no need to move them again. Once you've run CLEAN on a coverage, all of the points in that coverage are now greater than fuzzy tolerance from all other points in that coverage. Like this:

(He draws a diagram on the bridge.)



So the next time you CLEAN it, only newly added or moved points will move, since all of the old points are still further than fuzzy tolerance from all other points. So you see, there is no such thing as fuzzy creep! You CLEAN a coverage as often as you want with out danger, as long as you don't increase the fuzzy tolerance.

Troll: Well, how did this myth get started, then?

Programmer: Well, in the bad old days many of the ESRI trainers didn't really understand fuzzy tolerance. As I said, it's a subtle concept. And there actually is something real called fuzzy creep, but it's a technical term involving growth of the band in a band-sweep intersection detector—it has nothing to do with how much or how often arcs move.

Troll: Well, now I know. Thank you very much for the lecture, Mr Programmer. And now I have something to tell you.

Programmer: And that is?



Modifying Table Structures

[Because Version 4 will introduce some significant changes in table handling, this topic will be revisited after its release.]

Currently, there are only four command prompt options for modifying existing table structures: ADDITEM, DROPITEM, MODITEM, and PULLITEM (ADDXY isn't included in this discussion).

MODITEM is similar to the TABLES command ALTER in Host ARC/INFO, save that it's run from the ARC/W prompt. You can use the command to rename an item and/or alter its definition. Given the following table:

```
(C:\OHOME\TEST\PLP)[ARC]items veg.species
```

COLUMN	ITEM NAME	WIDTH	TYPE	N.DEC
1	MUNTNUMB	4	C	0
5	MUNTCOMP	12	N	2
17	SPP	5	C	0
22	PCT	12	N	2
34	LIFEFORM	5	C	0

The following commands will convert MUNTNUMB to an integer field, MUNTCOMP to a character field, rename SPP to SPECIES, rename PCT to PERCENT, and increase the number of decimal places of PERCENT to 4:

Troll: *I love El Burito hot sauce! I eat a garbage burrito a week to keep my meanness up. I was only stringing you out until you told me what I wanted to know. Now I'm going to eat you, and there's nothing you can do about it! Only a knight or a wizard can destroy a Troll, and you're only a measly little programmer.....*

Programmer: Ah, but I'm a UNIX wizard! Take:

```
kill -9 `ps -aux | grep ^troll | cut -c9-14`
```

you dastardly Troll! (The Troll fades into thin air in mid leap. The Programmer strides across the bridge, whistling a merry tune.)

~ The End ~

PLP

```
(C:\OHOME\TEST\PLP)[ARC]moditem veg.species
[PC ARC/INFO 3.5.2 MODITEM - 07/15/98]
:m muntnumb muntnumb 4 n 0
:m muntcomp muntcomp 12 c
:m spp species
:m pct percent 12 n 4
:end
(C:\OHOME\TEST\PLP)[ARC]items veg.species
```

COLUMN	ITEM NAME	WIDTH	TYPE	N.DEC
1	MUNTNUMB	4	N	0
5	MUNTCOMP	12	C	0
17	SPECIES	5	C	0
22	PERCENT	12	N	4
34	LIFEFORM	5	C	0

Specifying a smaller width allows parsing of one item into several—the online help for MODITEM has an example of parsing an address field. Note that when you alter the width you must specify a new item name; the remaining portion retains the old item name.

MODITEM cannot be used simply to alter the width of an item. However, an item width may be modified by adding a temporary item and copying over the values. The following routine uses SCRATCHN and SCRATCHI from **V6N6**¹:

```
&routine setwidth

&define table -11 &var
&define item -12 &var
&define width -13 &var
&define dec -14 &var
&define type -15 &var
&define tmpfile -19 &var

&if &eq "%-1" &do
    &delim < >
    &type "Usage:  &r SETWIDTH [table]
[item] [width] {decimal_places}"
    &delim [ ]
    &return
&end
&if &ninfo %-1 &do
    &type "%-1 is not a table."
    &return
&end
&extract [table] -1 1
&extract [item] -2 1
&value [width] -3
&if &eq "%-4" &do
```

```

    &sv [dec] #
&else
    &value [dec] -4
&end
&sv [type] "NOT FOUND"

&rem **** get item definition

&r scratchn -wksp
&rv [tmpfile]
&openw [tmpfile]
ITEMS [table]
&closew
&open [tmpfile] error
&while &do
    &read -1 [break]
    &extract -2 -1 1
    &if &nm "%-2" &do
        &extract -3 -1 2
        &extract -4 -1 4
        &if &eq "%-3" [item] &do
            &value [type] -4
            &if &eq "[dec]" "#" &do
                &extract [dec] -1 5
            &end
            &break
        &end
    &end
&end
&end
&close
& DEL [tmpfile]
&if &eq "[type]" "NOT FOUND" &do
    &type "Item [item] not found."
    &return
&end

&rem **** add and calculate temp item

&r scratchn -suffix SML -wksp
&rv [tmpfile]
&openw [tmpfile]
&write "SEL [table]"
&write "&r scratchi"
&write "&rv 51"
&write "ADDITEM"
&write "%%51"
&write "[width]"
&write "[type]"
&if &eq "[type]" "N" &do
    &write "[dec]"
&end
&write "[item]"
&write " "
&if &eq "[type]" "C" &do
    &write "MOVE [item] TO %%51"
&else

```

```

    &write "CALC %%51 = [item]"
&end
&write "DROPITEM [item]"
&write "Y"
&write "Q"
&closew
TABLES [tmpfile]

&rem **** rename temp item

&openw [tmpfile]
&write "M %%51 [item]"
&write "END"
&closew
MODITEM [table] [tmpfile]
& DEL [tmpfile]

&return

&label error
&type "I/O Error."

```

Use COMPSML SETWIDTH N to create an SML file from the routine. Note that it uses global variable 51 to pass the temporary item name to MODITEM.

```
&r setwidth veg.species percent 6 2
```

In addition to being a quick way to drop several items at one step, PULLITEM can be used to rearrange items in a table:

```

Enter the 1st item: muntnumb
Enter the 2nd item: muntcomp
Enter the 3rd item: percent
Enter the 4th item: lifeform
Enter the 5th item: species
Enter the 6th item: end
Do you wish to use the above items (Y/N)?
Y
(C:\0HOME\TEST\PLP)[ARC]items veg.species

```

COLUMN	ITEM NAME	WIDTH	TYPE	N.DEC
1	MUNTNUMB	4	N	0
5	MUNTCOMP	12	C	0
17	PERCENT	6	N	2
23	LIFEFORM	5	C	0
28	SPECIES	5	C	0

TABLESW

The "Revise Items" dialog of TABLESW is accessed via the File Menu. The dialog has a number of powerful functions for modifying table items. For example, the name, size, and definition of an item may be altered all at once:

Item Modification

Name	Col	Wid	T	Dec
MUNTNUMB	1	4	I	0
MUNTCOMP	5	12	C	0
PERCENT	17	6	N	2
LIFEFORM	23	5	C	0
SPECIES	28	5	C	0

New Item Information

Name:

Width:

Type:

Dec:

Buttons: Close, Delete, Merge, SizeUp, SizeDn, Add, Modify, Split

The “Split” function, as in MODITEM, splits an item in two, while “Merge” concatenates two items into one. “SizeUp” increases the size of numeric items to a default width (11 N 0 or 13 N 6), and “SizeDn” reduces the size of an item to the largest value contained in the records of that item.

TABLESW and SML

Although functionality uniquely available to TABLESW is not available to commands executable via SML—hopefully some richer tools will be available for application developers in Version 4—TABLESW itself may be launched from an SML application. If you launch TABLESW without any arguments, you'll want to make sure that WIN PATH matches the current directory²:

```
@SYSDIR 11
WIN PATH %11
TABLESW
```

That's not an issue if the name of a table is being supplied as an argument, but both the WIN PATH and the working directory will be changed to one level above current one (unless you're already at the root directory):

```
&sv 10 veg.species
@SYSDIR 11
TABLESW %10
CD %11
WIN PATH %11
```

Similarly, when setting the join environment for TABLESW, specify the full paths of the tables involved or you'll get a “not found” error:

```
@SYSDIR 11
&goto skip &if &eq %<lpos \ %11> %<len %11>
    &sv 11 %11\
    &label skip
    &r scratchn -wksp
    &rv -9
    &openw % -9
    &write "%11veg.pat"
    &write "%11veg.species"
    &write "muntnumb"
    &write "ordered"
    &write "%11veg.pat"
    &write ""
    &closew
TABLESW * % -9
& DEL % -9
WIN PATH %11
CD %11
```

¹SCRATCHN and SCRATCHI can also be downloaded at:
<http://www.primenet.com/~piersen/PC/arc.htm>

²SYSDIR is available in the %ARC%\TOOLS\SYSTEM directory. See also the online help under @SYSDIR.

PLP

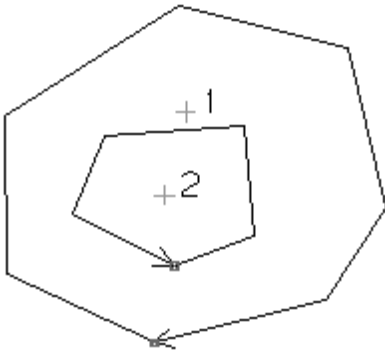


ARCSHAPE Workarounds

In 3.5.2, when ARCSHAPE creates a polygon shapefile, polygon records containing islands are written improperly: polygon parts representing island rings are missing the starting point which is necessary to match the closing point. This will lead to strange results in ArcView when zooming closely into such polygons or editing/cleaning them. Also, SHAPEARC will not close those rings.

The bug may be detected as follows.

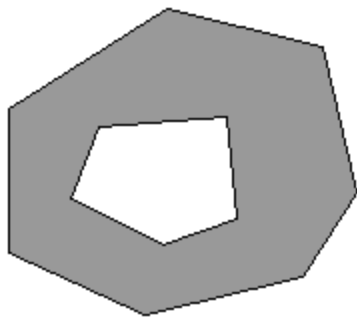
- 1) Create a coverage TEST1 containing a polygon with an island:



- 2) Use ARCSHAPE to create shapefile TEST1:

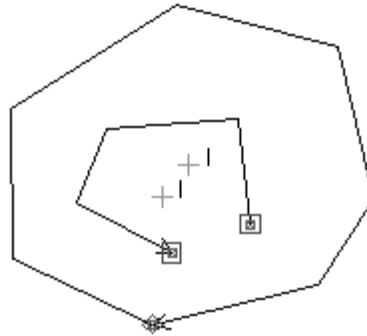
```
ARCSHAPE TEST1 TEST1 POLY
```

- 3) In ArcView, select the outer polygon and convert to shapefile TEST2:



- 4) Use SHAPEARC to create coverage TEST2:

```
SHAPEARC TEST2 TEST2
```



There are at least two workarounds. If your 3.5.1 CD-ROM is available, you can restore the 3.5.1's version of ARCSHAPE.EXE. At an MS-DOS window, CD to the %ARC%\PROGRAMS directory and execute the following:

```
>ren arcshape.exe arcshape.352  
>pkunzip d:\arcexe\programs\datacon.zip  
arcshape.exe
```

where "d:" is the drive letter of the CD-ROM drive.

Also, bad shapefiles produced by 3.5.2 may be repaired in ArcView using the following script (shapefiles which do not have polygons with islands need not be processed using this script):

```
'**** get input and output shapefiles  
  
t1 = "Repair Shapefile"  
fnInput = FileDialog.Show("*.shp",  
"Shapefiles", t1)  
if (fnInput = nil) then return nil end  
inFTab =  
FTab.Make(fnInput.AsString.AsSrcName)  
if (inFTab.GetShapeClass.GetClassName  
<> "Polygon") then  
    MsgBox.Error("Shapefile must be  
polygon.", t1)  
    return nil  
end  
fnDefault =  
FileName.GetCWD.MakeTmp("shape", "shp")  
fnOutput = FileDialog.Put(fnDefault,  
"*.shp", "Output Shape File")  
if (fnOutput = nil) then exit end
```




Inserting Text Files

Joe Zastrow, ESRI

Sometimes when working with dialog boxes you have to fill several list boxes with some items. Or when you are making a dialog box you need to insert one or more files into the file you are writing. It is very common to do something like this:

```
&openw [wksp]t$calc.dlg
&write "BEGIN      0  5.00 20.00 13.50
54.00 v Calculate"
..
..
&closew
& type mylist >>t$calc.dlg
&openw t$calc.dlg a
&write ..
..
&closew & type mylist2 >>t$calc.dlg
&openw t$calc.dlg a
&write ..
..
&closew
```

```
&openw [wksp]t$calc.dlg
&write "BEGIN      0  5.00 20.00 13.50
54.00 v Calculate"
..
..
&display mylist
&write ..
..
&display mylist2
&write ..
..
&closew
```

Since the t\$calc.dlg was opened with **&openw** on file handle one, the output from the **&display** command goes to the t\$calc.dlg file, instead of the screen. I just recently rediscovered this handy feature. You save some file opens and closes, you also save a shell to DOS.

Instead of closing the file, using and then using “& type” to shell to the type command and then reopen the file, it is much faster to use **&display**:

PLP

```
'**** set up output shapefile

fnOutput.SetExtension("shp")
outFTab = FTab.MakeNew(fnOutput,
POLYGON)
inFields = inFTab.GetFields
newFields = List.Make
for each f in inFields
  if (f.GetName <> "shape") then
    newFields.Add(f.Clone)
  end
end
outFTab.AddFields(newfields)

'**** do it!

nCount = 0
nRecs = inFTab.GetNumRecords
inSF = inFTab.FindField("shape")
outSF = outFTab.FindField("shape")
for each r in inFTab
  nRecNew = outFTab.AddRecord
  for each inF in inFTab.GetFields
    fName = inF.GetName
    if (fName = "shape") then
```

```
    p1 =
inFTab.ReturnValue(inSF,r)
    p2 =
Polygon.Make(p1.AsPolyLine.AsList)
outFTab.SetValue(outSF,nRecNew,p2)
  else
    outF =
outFTab.FindField(fName)
    val =
inFTab.ReturnValue(inF,r)

outFTab.SetValue(outF,nRecNew,val)
  end
end
nCount = nCount + 1
av.SetStatus((nCount / nRecs) *
100)
end
av.ClearStatus
av.ClearMsg
av.PurgeObjects
MsgBox.Info("Done.",t1)
return nil
```

PLP



Angled Annotation

Here's a quickie SML that sets the angle of a selected POINT1 annotation by converting it to POINT2:

ta.sml:

```
&goto usage &if &eq "%-1"
SHOW ANNOTYPE -2
SHOW ANNOPOS -3
SHOW ANNOFIT -4
SHOW COORDINATE -5
ANNOTYPE POINT2
ANNOPOS LL
ANNOFIT OFF
COORDINATE KEY
SHOW SELECT 1 -6
SHOW ANNO %-6 POINT 1 -7 -8
SHOW ANNO %-6 SIZE -9
SHOW ANNO %-6 TEXT -10
&cv -11 %-9 * len "%-10"
&cv -12 %-1 cos * %-11 + %-7
&cv -13 %-1 sin * %-11 + %-8
REPOS
1 %-7 %-8
1 %-12 %-13
ANNOTYPE %-2
ANNOPOS %-3
ANNOFIT %-4
COORDINATE %-5
&type " "
&return
&label usage
&type "Usage:  &r ta [angle]"
&return
```

Before:

```
: sel
: &r ta 30
```

After:

This routine allows the user to specify the annotation angle as it's being added:

&routine addang

```
SHOW ANNOTYPE -2
SHOW ANNOPOS -3
SHOW ANNOFIT -4
SHOW COORDINATE -5
SHOW ANNOSIZE -6
&if &eq %-6 0 &do
    SHOW MAPE -11 -12 -13 -14
    &cv -9 %-14 - %-12 / 100
&else
    &value -9 -6
&end
ANNOTYPE POINT2
ANNOPOS LL
ANNOFIT OFF
&sv -20
&while &do
    &res -10 "Text: "
    &if &eq "%-10" &do
        &break
    &end
    &res -1 "Angle: " 0
    &type "Coordinate (9 to cancel):"
    &getxym -7 -8 -19
    &if &eq %-19 9 &do
        &break
    &end
    &cv -11 %-9 * len "%-10"
    &cv -12 %-1 cos * %-11 + %-7
    &cv -13 %-1 sin * %-11 + %-8
    COORDINATE KEY
    ADD
    %-10
    1 %-7 %-8
    1 %-12 %-13
    %-20
    &type " "
    COORDINATE %-5
&end
ANNOTYPE %-2
ANNOPOS %-3
ANNOFIT %-4
ANNOSIZE %-6
```

Use COMPSML ADDANG N to create ADDANG.SML.

PLP