Snappy ANSI Startup Screens

By Ted Roche

I've never had a client show up for a product demonstration with a stopwatch, but all of my clients are well aware that time is money. While we developers may have used our stopwatches to benchmark custom routines in order to tune software performance or to select a DBMS, the raw performance speeds of code are usually not the concern of the client. What does influence the client is the perceived speed of the application — how fast it appears to solve the problem for which it was purchased. We may be called upon to present FoxPro as the solution.

FoxPro 2.0 is a great system, with good performance characteristics and maintenance tools. So try to sell it to a client. Sit them down before a machine and start your application. FoxPro starts to load... and load... and load. When the startup screen appears, try to remember to breathe again, and start your sales pitch.

MakeANSI eliminates this awkward moment by giving you the tools to present a snappy startup screen immediately upon starting your application. Entertaining the operator while FoxPro loads may seem like a frivolous use of code, but pause and consider it for a minute. When you click the mouse or hit a key to start an application, do you get a response from your system? With the growing popularity of GUI-based systems, clients have come to expect immediate feedback from their actions: press a button and an icon on-screen is depressed. I have found that using a MakeANSI-generated startup screen improves a client's opinion of the application, since they feel they get a reaction from starting the system and therefore feel the entire system is more responsive.

MakeANSI allows you to design a screen interactively, using familiar FoxPro commands, rather than requiring you to learn a new set of controls for a screen painter. MakeANSI creates a file that can be displayed from the DOS prompt. This file recreates the original screen saved from FoxPro, reproducing the varied colors, intensities, and blinking attributes possible in the FoxPro screen. Since it contains ANSI escape codes, the ANSI.SYS device driver must be loaded in CONFIG.SYS for it to display properly. (See the ANSI sidebar.)

Continued

How to use MakeANSI

There are four steps to creating and integrating a MakeANSI-generated screen into your application. These steps are described below in detail, but here is an overview:

1. Create the title screen you wish to present to the operator.
2. Save the screen as a memory variable and the memory variable to a .MEM file.
3. Run MakeANSI to create the text file.
4. Integrate the screen into your application by creating a batch file to display the startup screen and invoke your application.

Since this is a developer utility, it is assumed the reader is familiar with the FoxPro and DOS commands discussed. For further information on those commands, please consult the appropriate reference manuals.

Creating the title screen

If you have an existing startup screen, you may be able to jump right to the next step, "Saving the Results." However, you might want to skim this section for some ideas on "jazzing up" the screen you have.

FoxPro 2.0 contains a number of commands for manipulating screen characters. Since the Command Window is separate from the display screen, you can interactively paint the screen by typing commands from the Command Window. The commands @...SAY, @...BOX, @...CLEAR TO..., and @...FILL allow you to produce a variety of characters, patterns, and effects. The COLOR and COLOR SCHEME clauses that can be used with these commands have simplified screen painting since the FoxBASE days! You may want to refer to your FoxPro reference books to review all of the capabilities of these commands.

To aid with placement of objects at a particular spot on the screen, I have found the command:

```
ON KEY LABEL RIGHTMOUSE ;
   WAIT WINDOW NOWAIT ;
   "ROW "+LTRIM(STR(MROW()))+ ;
   " COL "+LTRIM(STR(MCOL()))
```

to be of enormous help. After typing this command, you may press the right mouse where you want an object to be placed, and the coordinates will appear in a WAIT window. **[Editor's Note:** *See Matt Peirse's utility, RULER.PRG, in the August issue for a more extensive positioning aid.*]

The method I use to design the screen usually involves a combination of programmed and interactive techniques. Typically, I start from the Command Window and then paste the resulting commands into a program as I find effects I like. When you've developed routines that create your company logo or other patterns, you can save them as programs and use them to start the design process. Simple FOR...NEXT loops with FoxPro's "Special Characters" can produce regular "plaid" or graduated backgrounds. An example program, used to develop the background for the MakeANSI program, is included with this article.

Saving the results

Working from the Command Window, you may save the current screen image to a memory variable using the "SAVE SCREEN TO m.MemVar" command, and likewise restore the screen with the "RESTORE SCREEN FROM m.MemVar" command. In addition, if you "SAVE SCREEN to m.Screen1 | m.Screen2 | m.Screen3" before each major change, you can "RESTORE SCREEN from m.Screen3" to give yourself an "UNDO" feature to the screen painting.

After saving the screen image to a memory variable, save the data to disk with the command SAVE ALL LIKE Screen* to ScrnFile.MEM. You could save a series of these .MEM files to use as templates for future work as well. Another advantage to being able to save the screen memory variables to disk is that the screen does not have to be designed in one session. Any time you find more pressing things to do, just follow the instructions above to save your work in progress. Recall the screen with RESTORE FROM ScrnFile.MEM ADDITIVE and RESTORE SCREEN FROM m.Screen.

Running MakeANSI

Run MakeANSI by invoking the MakeANSI.BAT batch file. Of course, a MakeANSI-generated startup screen will immediately appear, and the MakeANSI FoxPro program will load. You will be prompted for the name of the .MEM file saved in the step above. After specifying that file, you will be asked for the name of an output file. Any valid file name can be specified — this is the file name you specify within your startup batch file, as described in the following section. MakeANSI will begin to generate the ANSI escape codes to reproduce your screen from DOS. To entertain you while this happens, the screen will clear and display (without color) the screen being generated. When file generation is complete, you are returned to the DOS prompt.

Adding the screen to your application

A sample batch file is included here to show how the generated screen will be used. When your users invoke this batch file, the generated screen image is TYPEd to the screen before the application is started. The FoxPro startup option -T, suppressing the FoxPro title screen, ensures that the screen will not be disturbed by FoxPro's startup until your program is in control.

When your program starts, you may want to further manipulate the startup screen by adding a version number, date and time, or other information on top of the displayed screen. Changing only a small portion of the screen, or creating a simple animated effect, can add pizazz to your startup. Don't, however, expect your users to patiently wait through several minutes of animation each and every time your system starts. That said, prudent use of some of the newer commands, such as the SCROLL command, can create striking and dramatic effects to let the operator know the system is now ready for input.

There are two different behaviors that will be seen with a MakeANSI-developed startup. If your users are running the interactive FoxPro (i.e., starting up with FOX, FOXPROL, or FOXPROLX), a few seconds after the ANSI screen is displayed, FoxPro will take over, clearing the screen first to black, then blue, before your application takes control. At that point, you may want to restore the screen from a copy of the .MEM file.

But if you are distributing an application, where a polished presentation is often more important (by running FOXR YourApp.APP or a compact or standalone .EXE), the screen will not be cleared prior to your application gaining control. This can allow you to create a much smoother transition during startup. If you discover that your screen is still clearing during startup, it may be that certain command settings in your configuration file (CONFIG.FP) are responsible. For example, setting SCOREBOARD or STATUS either ON or OFF will cause the screen to be blanked. Since these commands are set OFF as the default and rarely used (note the Commands & Functions manual remarks that these commands are "Included for backward compatibility"), they probably can be deleted from your CONFIG.FP for a smoother startup appearance.

How MakeANSI works

A screen memory variable contains information about the characters displayed on the screen: their values, colors, and attributes. When saved to disk as a .MEM file, two bytes are written for each screen position (for example, an 80 x 25 screen would have 4,000 bytes of information: 80 x 25 x 2 = 4,000). The first byte of each pair specifies the actual character displayed. The second byte

Continued

ANSI Escape Codes

ANSI.SYS is an enhanced device driver for the standard input and output devices, and is provided with MS-DOS. It takes up little memory (approximately 4K) and can be "loaded high" if your memory manager (or later version of DOS) allows this feature. It must be loaded into the computer at startup from the CONFIG.SYS file, with a line like:

```
DEVICE = ANSI.SYS
```

After the device is loaded, escape code sequences sent to the screen when DOS is in control of the display will have the effect of changing the resolution, colors, or attributes of the display. These escape codes may be sent to the screen as part of the PROMPT or ECHO commands, or TYPEd or COPY'd to the screen from a file. Here is a list of some of the effects and their codes:

```
ERASING THE SCREEN
     Esc[2J          Clears entire screen.
     Esc[k           Erases from current cursor
                     position to right side.

CHANGING DISPLAY
     Esc[#;...;#m    Changes display attributes or colors:

     Attributes:
     0 = Reset - white/black, 1 = High intensity on,
     4 = Underscore (Monochrome only), 5 = Blinking,
     7 = Inverse-black/white, 8 = Invisible-black on black

     Foreground Colors:
     30 = Black, 31 = Red, 32 = Green, 33 = Yellow,
     34 = Blue, 35 = Magenta, 36 = Cyan, 37 = White

     Background Colors:
     40 = Black, 41 = Red, 42 = Green, 43 = Yellow,
     44 = Blue, 45 = Magenta, 46 = Cyan, 47 = White

CHANGING SCREEN MODE
     Esc[=#h or Esc[=#l changes screen to black & white (BW)
     or color (C) with rows by columns resolution of:
     0 = BW 40x25, 1 = C 40x25, 2 = BW 80x25, 3 = C 80x25,
     4 = C 320x200, 5 = BW 320x200, 6 = BW 640x200
     Esc[=7h enables end-of-line-wrap, Esc[=7l disables it

CURSOR POSITIONING
     Esc[r;cH     or Esc[r;
                  cf  repositions cursor at Row r, Column c
     Esc[rA       Move r rows up.
     Esc[rB       Move r rows down.
     Esc[cC       Move c columns right.
     Esc[cD       Move c columns left.
     Esc[s        Save current cursor position
                  for restoring later.
     Esc[u        Restore to last saved cursor position.
```

specifies the characteristics of that screen position: the character color, background color, whether the character should blink, and whether the character should be displayed in high intensity. See the remarks at the beginning of the MakeANSI program for the details on how this information is encoded into one byte.

MakeANSI asks you to specify the input file name, using the built-in GETFILE() FoxPro function, and then asks for the output file name using the PUTFILE() function. MakeANSI then reads the .MEM variable into a string and loops through the processing of that string into the equivalent ANSI codes. See the sidebar on ANSI codes for an explanation of the syntax of those commands. When MakeANSI has completed the processing of the string, it adds the ANSI commands to "home" the cursor and reset the DOS colors to their default so that the user will be presented with a "normal" DOS screen upon return from your FoxPro application. You may need to adjust these settings if your default environment differs from the default DOS dull-white on black screen.

Adding MakeANSI to your toolbox

Like many of the programs contained on these pages, MakeANSI is skeletal. The basic ideas of working with low-level file functions and providing at least minimal error checking are essential, but MakeANSI is more of a concept than a finished work. It is not intended as an end-user tool, but rather as a utility to be added to a developer's toolbox. Developers are encouraged to make the modifications necessary to fit it to their needs. An explanation of some of the limitations and areas in which it could be expanded follows:

- Alert readers will note that MakeANSI reads and processes all but the last character (the last two bytes) of the saved screen image. This is to compensate for a "feature" of DOS that automatically scrolls the screen when a character is typed in the lower right corner of the screen.
- You are not restricted to running FoxPro in 25-line by 80-column mode. Fox appears to be quite happy to run in whatever screen mode it was started in. If you choose to use an alternate screen mode, such as 43 x 80 or 50 x 80, make sure that the machines for which the system is being developed are capable of displaying that mode.

  The screen display can be switched to an alternate mode before starting FoxPro in a number of ways. Commercial programs, such as Ultravision, will alter the screen. Publicly-distributable or shareware software, such as EMODE.COM (available in CompuServe's FoxForum), will also do the trick. Finally, if MS-DOS 5.0 is installed, an enhanced option of the MODE command will switch screen modes, using the syntax:

```
MODE CON: lines = yy cols = xx
```

  where yy can be 25, 43, or 50, and xx can be 40 or 80 column mode. Note that it is the developer's responsibility to ensure that the user's display is placed into the desired mode before the MakeANSI-generated screen is displayed.

  If you are developing in one screen mode but intend the users to work in another, be sure to switch modes before capturing the screen for MakeANSI to convert. The SAVE SCREEN TO command captures the entire screen, and MakeANSI translates the entire memory variable into an ANSI file. Thus, it will read and output the correct number of characters for the screen mode selected when the screen was saved. An enhancement you may consider to the MakeANSI program is to sense the size of the captured screen and offer the operator the option to change the screen resolution. Within the .MEM file, the memory variable name, type, and size information are stored in the 32 bytes preceding the memory variable's data. Using low-level file functions to read and parse this information could also allow you to enhance MakeANSI to process .MEM files containing more than one memory variable (this version of MakeANSI assumes that the screen variable is the only variable stored in the file).
- Yet another enhancement that may be attractive to some developers, especially if many images are involved, is to save the memory variables to memo fields using the SAVE...TO MEMO syntax and then manipulating the memo field using the string variable functions in a fashion similar to that in MakeANSI. This would eliminate cluttering a disk with .MEM files and provide a .DBF file, which can more easily be manipulated and distributed.

MakeANSI solves a simple problem: how to entertain the operator while the software loads. Far from being a trivial "special effect," the immediate feedback to users assures them that your system is designed to respond quickly to their actions. The MakeANSI utility allows you to develop this startup screen with a minimum of

resources and effort, using a language and techniques with which you are already familiar, and to deliver this to your end users with a minimum of overhead on their systems. The techniques demonstrated in MakeANSI show how low-level file functions and parsing of .MEM files may be integrated into an application. I hope it stimulates your imagination on how the expansion of these techniques may benefit your applications.

```
************************************************************
* MakeANSI.PRG - create file of ANSI escape codes from a
*                saved screen .MEM file
*
* A .MEM file stores a screen image as a 4,000-char string.
*
* The odd-numbered bytes (1,3,5,7...) are the screen char,
* from the top left corner, left to right, down the screen.
*
* The even byte (2,4,6,8...) following it is the color
* code for that screen character using the following scheme:
*
* mColorAttr has Bits  8--7--6--5--4--3--2--1
* Bit 8       - Blink Attribute 1 = Yes, 0 = No
* Bits 7,6,5 - Color of background, 0 to 7 (see table)
* Bit 4       - Intensity Attribute 1 = Yes, 0 = No
* Bits 3,2,1 - Color of foreground, 0 to 7 (see table)
*
***************** Popup Startup Screen ******************
* If the file exists and not running runtime
if file("MAKEANSI.MEM") and ! "Support" $ vers(1)
   restore from makeansi.mem additive
   restore screen from mScreen
   release mScreen
else
   clear
endif
************************ SETUP ************************
set talk off
private mOldColor, mCharacter, mESCape, mPrompt, Read_File
private Read_Handle, Read_Size, Read_Point, Read_String
private Write_File, Write_Handle
mOldColor = 0
mESCape = chr(27)
* These are the ANSI Escape color codes *
dimension mANSIf(8)   && ANSI (f)oreground color
          mANSIf(1) = '30'
          mANSIf(2) = '34'
          mANSIf(3) = '32'
          mANSIf(4) = '36'
          mANSIf(5) = '31'
          mANSIf(6) = '35'
          mANSIf(7) = '33'
          mANSIf(8) = '37'
dimension mANSIb(8)   && ANSI (b)ackground color
          mANSIb(1) = '40'
          mANSIb(2) = '44'
          mANSIb(3) = '42'
          mANSIb(4) = '46'
          mANSIb(5) = '41'
          mANSIb(6) = '45'
          mANSIb(7) = '43'
          mANSIb(8) = '47'
dimension mBits(8)  && Array holds bit values of color byte
          mBits = 0
********************* SPECIFY I/O **********************
mPrompt = 'Where is the saved screen .MEM file?'
Read_File = GetFile("MEM",mPrompt)
if empty(Read_File)
   return
endif

mPrompt = 'Select output ANSI text file'
Write_File = PutFile(mPrompt,'Output.TXT','TXT')
if empty(Write_File)
   return
endif

************************ READ *************************
Read_Handle = FOPEN(Read_file,0)
if Read_Handle<=0
   WAIT WINDOW "Error opening file. Aborting" NoWait
   return
endif

* Move to EOF to determine file size
Read_Size = FSEEK(Read_Handle, 0, 2)
IF Read_Size <= 4032 && Is File too small?
   mPrompt = 'This file does not contain a screen image.'
   WAIT WINDOW mPrompt NOWAIT
    = FCLOSE(Read_Handle) && Close the file
    return
endif

* Move to BOF + 32 and store contents to string
Read_Point = FSEEK(Read_Handle,32, 0)
Read_String = FREAD(Read_Handle, Read_Size-33)
if not FCLOSE(Read_Handle) && Close the file
   do ErrorChk
   return
endif

************************ WRITE ************************
Write_Handle = FCREATE(Write_File)
if ErrorChk()
   return
endif

* Home the cursor and clear the screen
if FWRITE(Write_Handle,mESCape+"[2J") = 0
   do ErrorChk
   return
endif

clear
for x=1 to len(Read_string)-2 step 2
   mCharacter = substr(Read_String,x,1)
   mColorAttr = asc(substr(Read_String,x+1,1))
   if mColorAttr # mOldColor  && recalculate only if
      mOldColor = mColorAttr  && the color has changed
      =BitSplit(mColorAttr,@mBits)
      mBlink = iif(mBits(8)=1,"5;","")
      mBackColor = mAnsib(4*mBits(7)+2*mBits(6)+mBits(5)+1)
      mIntense = iif(mBits(4)=1,"1;","")
      mForeColor = mAnsif(4*mBits(3)+2*mBits(2)+mBits(1)+1)
      ANSIString = mESCape+"["+"0;"+mIntense+mBlink+ ;
                   mForeColor+";"+mBackColor+"m"
      * send the ANSI color string
      if FWRITE(Write_Handle,ANSIString) = 0
         do ErrorChk
         return
      endif
   endif
   ?? mCharacter  && display on screen to entertain
   if FWRITE(Write_Handle,mCharacter) = 0
      do ErrorChk
      return
   endif
next

* Home the cursor and reset the color
if FWRITE(Write_Handle,mESCape+"[0;0H"+mESCape+"[0m") = 0
   do ErrorChk
   return
endif
if not FCLOSE(Write_Handle)
   do ErrorChk
   return
endif
return

Procedure BitSplit
parameters mParam, mBitArray  && the character to take apart
external array mBitArray      && Avoids a BUILD APP error
private mNum                  && Temporary scratch number
private mExp                  && Exponent, Array Pointer

mNum = mParam
for mExp = 7 to 0 step -1
   if mNum >=2^mExp
      mBitArray(mExp+1) = 1
      mNum = mNum - 2^mExp
```

Continued

```
   else
      mBitArray(mExp+1) = 0
   endif
next
return

Function ErrorChk
if FERROR() # 0
   close all
   DEFINE WIND alert FROM 7,17 TO 12,60 DOUBLE COLO SCHEME 7
   DO CASE
      CASE FERROR() = 2
         reason = 'File not found'
      CASE FERROR() = 4
         reason = 'Too many files open (out of handles)'
      CASE FERROR() = 5
         reason = 'Access denied'
      CASE FERROR() = 6
         reason = 'Invalid file handle'
      CASE FERROR() = 8
         reason = 'Out of memory'
      CASE FERROR() = 25
         reason = 'Seek error - BOF encountered'
      CASE FERROR() = 29
         reason = 'Disk full'
      CASE FERROR() = 31
         reason = 'General failure'
      OTHERWISE
         reason = 'Unrecognized error code '+ ;
                  Ltrim(Str(FERROR()))
   ENDCASE

   ACTIVATE WINDOW alert
   @ 1,7 SAY 'Unable to open file'
   @ 2,7 SAY 'Reason: ' + reason
   @ 3,7 SAY 'Press a key to exit'
   set cursor off
   wait ""
   DEACTIVATE WINDOW alert
   CANCEL
endif
return FERROR() # 0  && can use as function() and procedure

************************************************************

* MKANSISC - Make MakeANSI.PRG startup screen
activate screen
clear
set talk off
set sysmenu off
@     0,0 say replicate(CHR(176),80) color RB+/w
@ row(),0 say replicate(CHR(176),80) color RB/w
@ row(),0 say replicate(CHR(176),80) color B+/w
@ row(),0 say replicate(CHR(176),80) color B/w
@ row(),0 say replicate(CHR(176),80) color BG+/w
@ row(),0 say replicate(CHR(176),80) color BG/w
@ row(),0 say replicate(CHR(176),80) color G+/w
@ row(),0 say replicate(CHR(176),80) color G/w
@ row(),0 say replicate(CHR(176),80) color GR+/w
@ row(),0 say replicate(CHR(176),80) color GR/w
@ row(),0 say replicate(CHR(176),80) color R+/w
@ row(),0 say replicate(CHR(176),80) color R/w

@ row(),0 say replicate(CHR(177),80) color RB+/w
@ row(),0 say replicate(CHR(177),80) color RB/w
@ row(),0 say replicate(CHR(177),80) color B+/w
@ row(),0 say replicate(CHR(177),80) color B/w
@ row(),0 say replicate(CHR(177),80) color BG+/w
@ row(),0 say replicate(CHR(177),80) color BG/w
@ row(),0 say replicate(CHR(177),80) color G+/w
@ row(),0 say replicate(CHR(177),80) color G/w
@ row(),0 say replicate(CHR(177),80) color GR+/w
@ row(),0 say replicate(CHR(177),80) color GR/w
@ row(),0 say replicate(CHR(177),80) color R+/w
@ row(),0 say replicate(CHR(177),80) color R/w

@ 1,5 to 5,40 double
@ 6,7 FILL to 6,41 color N/N   && simulate drop-shadows
@ 2,41 FILL to 6,41 color N/N
@ 2,6 CLEAR TO 4,39
@ 2,7 SAY "
@ 3,7 SAY "MAKEANSI
@ 4,7 SAY "
@ 18,40 to 22,75 double
@ 23,42 FILL to 23,76 color N/N
@ 19,76 FILL to 23,76 color N/N
@ 19,41 CLEAR TO 21,74
@ 20,41 SAY "  Screen .MEM => ANSI Text File  "
@ 21,41 SAY "          By Ted Roche           "
* Rem this out for .MEM file; use for .SCR
@ 19,41 SAY "      Loading... Please Wait" color GR+*/B
*
set sysmenu on
return
* Return control to Command Window; save or
* modify further from there
************************************************************

* SAMPLE BATCH FILES for use:

* REM MAKEANSI.BAT
* @echo off
* type makeansi.scr
* FOX makeansi +R -T
* CLS

* REM SAMPLE.BAT
* @ECHO OFF
* TYPE STARTUP.SCR
* FOX AppName.APP -T +R
* CLS
************************************************************
```

About the author:

Ted Roche is a Senior Programmer for The New Hampshire Insurance Group in Manchester, New Hampshire. In six years of analysis, design, and programming, he has completed a variety of projects in FoxPro, dBASE, and Clipper for clients in the manufacturing, state regulatory, and financial services industries. Ted is actively exploring opportunities for full-time employment or long-term contracting in the central and southern New Hampshire areas. He can be reached at 603/746-4017, or on CompuServe at 76400,2503.

Editorial

Continued from page 2

load soft fonts to your HP printer — without resorting to the RUN command.

Ted Roche gives us an astonishingly simple and useful technique to provide another form of output — application startup screens. When I say "simple," I don't mean it was simple to think of, or to design. Ted is sharing some very special expertise with us, and he does so in a vastly entertaining manner.

Chuck Werner extends Scott Grabo's Structure Builder to allow it to load existing tables into Scott's Data Dictionary table. I think you'll agree that this capability expands the tool in an important way. We've received other interesting recommendations for enhancements as *FoxTalk* readers continue to build on one another's achievements.

This process of refinement continues, as always, in The Workshop. We have some discussion of Whil Hentzen's USPS Bar Codes by Whil

Continued on page 19