



# T OOLS TO DRAW ON

Take control of the powerful drawing tools built into your GS.  
Several techniques and shortcuts help you design advanced  
graphics and build animation into your programs.

By **JOE ABERNATHY**

LEARNING TO USE THE GS' BUILT-IN PROGRAMMER'S toolbox is much like learning to use the computer itself. You should begin exploring it region by region. As a BASIC programmer, you also have access to some of the finest tools available in any language.

The Toolbox is a set of some 900 procedures that let you implement such features as pull-down menus, sound, graphics, dialog boxes, and so on. These various tools are organized into topical kits such as Menu Manager, Sound Manager, QuickDraw II, and Dialog Manager.

## START SMALL

Learning how to program all 900 tool calls would seem to be a career in itself. This indeed can be the case with languages that take a traditional approach to implementing the Toolbox (following the sophisticated syntax of Apple's Toolbox documentation faithfully). Within the various dialects of BASIC, however, are numerous techniques and shortcuts that can bring tool programming more within reach.

One good starting point is graphics. It's much of what makes the IIGS special. A subset of the Toolbox entitled QuickDraw II lets you display pictures, build in animation, and design sophisticated dialogs.

Most QuickDraw commands create primitive graphics such as points, lines, rectangles, circles, and polygons. So animation using QuickDraw might follow this simple recipe: Draw an object, erase it, then redraw it at a new location.

## ANIMATION BASICS

Working in structured, modular BASIC, the most elegant approach is to design a set of tools to handle each phase of this simple animation. And they'll give you full access to QuickDraw graphics via easy one-line calls in the programs you write.

The first step in using QuickDraw or any toolset is to turn it on. Traditionally, this involves making sure other toolsets that QuickDraw

uses internally are active, allocating memory for QuickDraw's functions, and finally issuing a call to activate the toolset. Fortunately, our IIGS BASICS simplify that process. In Micol Advanced BASIC, for example, single commands enable the various graphics modes:

```
GR      { 40x40 low-res graphics mode }
HGR     { super-high-res 320x200 mode }
HGR2    { super-high-res 640x200 mode }
TEXT    { turn off any graphics mode }
```

Here we must diverge from a general discussion of the various dialects, since there are significant differences among their QuickDraw commands. Although you might not own and use a particular language, you should read each section.

## MICOL ADVANCED BASIC

Micol offers a gentle introduction to graphics, much of it being familiar if you've done Applesoft programming with Micol. There are a number of built-in shortcuts, and you can easily build tools based on the Toolbox. By combining the two, you can create uniquely powerful tools for advanced graphics and fast animation.

A number of built-in commands afford drawing capabilities that otherwise are more difficult to program. For example:

## PROGRAM SHRDemo:

### ROUTINE Main

```
HGR          { start up QuickDraw, 320 mode }
BKCOLOR = 14 { light gray background }
HCOLOR = 4   { blue drawing pen }
HPLOT 4, 4   { locate pen at position 4, 4 }
DRAWSTR ("Hit any key to continue...")
HPLOT 4, 5   { horizontal = 4, vertical = 5 }
HPLOT TO 314, 186 { draw a line }
GET a$ { await keypress }
TEXT { shutdown QDII, restore text screen }
```

END

The graphics shortcuts demonstrated above can provide a lot of flexibility, but you need to call QuickDraw directly to achieve more sophisticated IIGS graphics. The following sample shows two ways to accomplish animation using a combination of simple commands and direct QuickDraw calls: ▶

## Listing 1. Program ShowAPic.

```
{ ----- }
{ ShowAPic (C)1989, inCider }
{ By Joe Abernathy. All Rights Reserved }
{ Compiler: Micol Advanced BASIC (GS) }
{ ----- }
{ Requires an uncompressed SHR picture. }

{ ----- }
{ PROC DoShowPic -- display SHR picture }

PROC DoShowPic[picfile$]

  { DoShowPic is based on an example
    written by Ron Lewin of Micol
    Systems Canada. Our thanks. }

  IF FILE(picfile$) THEN BEGIN
    HGR { start QuickDraw II }
    BLOAD picfile$,14753792,32766

    { the BLOAD command copies 32766
      bytes -- 65 disk blocks, the
      length of an uncompressed pic
      file -- from the picture file
      name passed in picfile$, to the
      screen at location 14753792 }

    GET a$ { await key press }
    TEXT { Shut down QuickDraw II }
    ENDIF
  ENDPROC { DoShowPic }

{ ----- }
{ ShowPic -- DoShowPic control routine }
{ for use with inCider prog }
{ SHOWFILE.CDA (June 1989). }

PROC ShowPic
  HOME
  HTAB (1)
  VTAB (6)
  PRINT "  Display Picture  "
  VTAB (8)
  INPUT "Enter full pathname of picture -> ";x$
  IF x$ = "" THEN Terror! = TRUE
  IF FILE (x$) THEN BEGIN
    GOSUB DoShowPic[x$]
  ENDIF
  InputErr! = FALSE
ENDPROC { ShowPic }

{ ----- }

{ Main loop -- DoShowPic demo }

ROUTINE Main
  Terror! = FALSE
  REPEAT
    GOSUB ShowPic
  UNTIL Terror!
  Terror! = FALSE
END { ShowPic }

{ ----- }
```

## PROGRAM Graphics

INT (A-Z)

DIM Buffer (10)

PROC Draw\_Rect [ Func\_Num, Min\_X, Min\_Y, Max\_X, Max\_Y ]

```
{ Draw_Rect by Ron Lewin of Micol Systems }
LSB = ADDR (Buffer) { address of buffer; syntax cq }
MSB = PEEK (202) { bank num of buffer }
TOOLBOX (4, 74: MSB, LSB, Min_X, Min_Y, Max_X, Max_Y)
TOOLBOX (4, Func_Num: MSB, LSB)
ENDPROC { Draw_Rect }
```

## ROUTINE Main

```
HGR { Start QuickDraw in 320 mode }
HCOLOR = 4 { drawing color blue }
BKCOLOR = 0 { black background, used also to erase }
```

```
{ draw a line of circles .. }
j% = 5
FOR i% = 100 TO 320 STEP 5
  j% = j% + 5
  GOSUB Draw_Rect [89, j%, 5, i%, 70]
NEXT i%
```

```
{ erase sequentially ... }
j% = 5
FOR i% = 100 TO 320 STEP 5
  j% = j% + 5
  GOSUB Draw_Rect [90, j%, 5, i%, 70]
NEXT i%
```

```
{ by changing the draw/erase order, }
{ you get another kind of animation .. }
j% = 5
FOR i% = 100 TO 320 STEP 5
  j% = j% + 5
  GOSUB Draw_Rect [89, j%, 5, i%, 70]
  GOSUB Draw_Rect [90, j%, 5, i%, 70]
NEXT i%
```

GET a\$

TEXT

END { PROGRAM Graphics }

## ROUND IT OFF

This example achieves its effects by calling the flexible Draw\_Rect procedure. This tool can draw rectangles or ovals in any size, and in one of three capacities: framed, color-fill, or background color-fill, which will erase a previously drawn color-fill object.

A similar procedure, Draw\_Arc, provides the same capabilities for round-cornered rectangles and arcs:

PROC Draw\_Arc [ Func\_Num, Min\_X, Min\_Y, Max\_X, Max\_Y,  
Start\_Angle, Angle\_Length]

{ Draw\_Arc by Ron Lewin of Micol Systems }

```
LSB = ADDR (Buffer()      {Syntax is correct}
MSB = PEEK (202)
TOOL BOX (4, 74: MSB, LSB, Min_X, Min_Y, Max_X,
           Max_Y)
TOOLBOX (4, Func_Num: MSB, LSB)
ENDPROC
```

Together, Draw\_Rect and Draw\_Arc generate a complete set of QuickDraw graphics primitives, and you should enter them into your permanent programmer's library. The Func\_Num values that determine what you draw are listed in the Micol manual and in volume 2 of the *Toolbox Reference Manual*.

**Listing 1** is another example of super-high-resolution (SHR) graphics in Micol Advanced BASIC. It illustrates how to load a "full-sized" (32K) SHR picture from disk and display it on screen. This procedure is designed so that you can add it to the *inCider* Show File utility presented in June's GS BASICs ("Think It Through," p. 86) or add it as a procedure to your permanent programmer's library.

#### AC/BASIC

AC/BASIC is also designed to give you the power of the Toolbox without the trouble of tool calls. Hence, you can draw most graphics primitives with a one-word command. AC's slick mouse-interface calls also offer convenient on-screen graphics control of objects.

Different AC graphics commands affect different objects, which you can combine to create sophisticated images. You can add graphics and disk-based pictures to your programs, enhance on-screen text, and even record to disk the composite picture your program's commands create.

AC/BASIC graphics objects include lines, boxes, arcs, ovals, polygons, rectangles, and round-cornered rectangles. You can choose to paint, invert, frame, fill, or erase the last five of those objects. To get a feel for this, type in **Listing 2**, and compile it with the default menus and window selections on.

This AC/BASIC example shows how to accomplish animation similar to that of the Micol program above, but apparently without making any tool calls. The compiler is making them for you in the background. **Listing 2** also shows how to manipulate fonts and colors quickly. Using polygon commands, you can create figures or objects as complex as your imagination.

And as you can see by comparing source code, much of your knowledge will be portable if you later add another BASIC compiler, or even another language, to your repertoire. One consideration that counts against AC/BASIC is its lack of speed, which manifests itself as "flashy" animation. So try to keep the size of animated figures small.

Another technique to use is pseudo animation—manipulating the color palette as demonstrated in the DoMenu routine in **Listing 3**. **Listing 3** also shows how to load a picture from disk and display it in either 320 or 640 mode. (**Listing 3** requires the original *inCider.Shell* for AC/BASIC. See "Studio BASIC," April 1989, p. 86.)

#### TML BASIC

TML BASIC is unique in letting you write unfettered desktop-style programs. AC/BASIC limits you to those desktop capabilities built into the language (unless you know machine language), and Micol Advanced ►

**Listing 2.** Graphics plotting example.

```
-----
' Graphics plotting example
' By Joe Abernathy. (C)1989, inCider
' All Rights Reserved.
' Compiler: AC/BASIC for Apple IIGS
'-----

DIM rect%(3) ' for QDII plotting
DIM pat%(3) ' QDII pen pattern
SCREEN 1 ' 320x200 screen
TEXTBCOLOR 14 ' light gray background
TEXTCOLOR 0 ' black
TEXTFONT 5 ' Venice
TEXTSIZE 14 ' 14-pt.
PRINT
PRINT "This is Venice 14, black on gray."
PRINT
PRINT "Click mouse to continue."
WHILE MOUSE (0) <> 0
    WEND ' Clear mouse buffer
WHILE MOUSE (0) = 0
    WEND ' Await real mouse click
BACKCOLOR 2 ' brown background
CLS ' clear screen to brown
FORECOLOR 6 ' orange graphics pen color
rect%(0) = 5 ' upper left y
rect%(1) = 5 ' upper left x
rect%(2) = 70 ' lower right y
rect%(3) = 150 ' lower right x
FRAMERECT VARPTR(rect%(0))
MOVETO 10, 100 ' x, y
PRINT "Click mouse to continue."
WHILE MOUSE (0) <> 0
    WEND ' Clear mouse buffer
WHILE MOUSE (0) <> 1
    WEND ' Await real mouse click
ERASERECT VARPTR(rect%(0))
CLS ' clear to brown

' animate with a line of circles ..
j = 5
rect%(0) = 5 ' upper left y
rect%(2) = 70 ' lower right y
FOR i = 100 TO 300 STEP 5
    j = j + 5
    rect%(1) = j ' upper left x
    rect%(3) = i ' lower right x
    FILLOVAL VARPTR(rect%(0)), VARPTR(pat%(0))
    ERASEOVAL VARPTR(rect%(0))
NEXT i

MOVETO 10, 100 ' x, y
PRINT "Click mouse to continue."

WHILE MOUSE (0) <> 0
    WEND ' Clear mouse buffer
WHILE MOUSE (0) <> 1
    WEND ' Await real mouse click
FOR i = 1 TO 1000
    NEXT i
END
```

## Listing 3. File: inCider.Shell, v1.1 revisions.

```

-----
' File: inCider.Shell, v1.1 revisions
' By Joe Abernathy
' (C)1989, Joe Abernathy. All Rights Reserved.
' Compiler: AC/BASIC for the Apple IIGS.
-----
' Portions of this program include material copyrighted (C) by
' Absoft Corp. 1988. Used with permission. All other copyrights
' acknowledged.
-----
' Make these changes to the shell to implement picture-handling.

' Add this at beginning of program:
DIM Pict$(16384+4)          ' Picture array

-----
' Replace previous "menuproc" with this:

menuproc:
  menunum = MENU(0)          ' Interpret menu events
  itemnum = MENU(1)          ' Read which menu
  IF menunum = 1 THEN        ' Read which item
    IF itemnum = 1 THEN      ' .. FILE menu
      GOSUB 10              ' New
    ELSEIF itemnum = 2 THEN  ' Edit
      GOSUB 20              ' Delete
    ELSEIF itemnum = 3 THEN  ' Print
      GOSUB 30              ' Type File
    ELSEIF itemnum = 5 THEN  ' Quit
      GOSUB 50              ' GOODIES menu
    ELSEIF itemnum = 6 THEN  ' Show picture
      GOSUB 60
    END IF
  ELSEIF menunum = 2 THEN
    IF itemnum = 1 THEN
      GOSUB 70
    END IF
  END IF
RETURN

-----
' Add capability to view SHR picture:

70:
  f$ = "null"                ' Show SHR picture
  WHILE f$ <> ""
    f$ = FILES$(1)           ' Open file dialog
    IF f$ <> "" THEN
      WINDOW 2
      OPEN f$ AS #1
      BLOAD #1,Pict$(4),32768 ' Load 32K of pic data
      CLOSE #1
      Pict$(0) = 640         ' Width can be 320 or 640
      Pict$(1) = 200         ' HEIGHT
      Pict$(2) = 128         ' MODE: use 0 for 320 mode
      Pict$(3) = 160         ' line bytes rounded to mult of 8
      PUT (0,0),Pict$        ' Show picture
      WHILE MOUSE(0) <> 0     ' Clear event queue of
        WEND                 ' ghost mouse clicks
      WHILE MOUSE(0) = 0     ' Await real mouse click
        WEND
      WINDOW CLOSE 2
      END IF
    WEND
  MENU
  RETURN

-----
' Replace old DoMenu with this:

SUB DoMenu
  FOR p = 1 TO 6              ' Create menu bar
    FOR e = 0 TO 12 STEP 4    ' No screen flicker
      PALETTE p,e+0,1,1,1    ' white out menu bar
    NEXT
  NEXT
  MENU 1,0,1,"File"          ' Build FILE menu
  MENU 1,1,1,"New"            ' and its entries.
  MENU 1,2,1,"Edit"
  MENU 1,3,1,"Delete"
  MENU 1,4,1,"Print"
  MENU 1,5,1,"Type"
  MENU 1,6,1,"Quit"
  MENU 2,0,1,"Goodies"
  MENU 2,1,1,"View Picture"
  FOR p = 1 TO 6
    FOR e = 0 TO 12 STEP 4
      PALETTE p,e+0,0,0,0
    NEXT
  NEXT
END SUB

-----
' End of inCider.Shell v1.1.

```

## Listing 4. TML BASIC animation demo.

```

-----
' TML BASIC Animation Demo
' By Joe Abernathy. (C)1989, inCider
' All Rights Reserved.
' Compiler: TML BASIC v1.10 for Apple IIGS
-----
LIBRARY "QuickDraw"          ' Load QDII toolset

DIM arect$(3)                ' rectangle
DIM colors$(15)              ' color palette

GRAF INIT 320                ' 320x200 graphics screen
GRAF ON                      ' Turn it on
  ClearScreen(-1)            ' Clear screen to white
PROC Ovals                   ' Draw line of ovals
PROC EraseOvals              ' and sequentially erase
PROC AnimOvals               ' Animated ovals
PROC Finis                   ' Press a key
GET$ Key$                    ' Animation example
END

DEF PROC Ovals
  ' Based on example written by TML Systems
  LOCAL j%
  j% = 20
  SetSolidPenPat(13) ' Light blue
  FOR i% = 100 TO 300 STEP 5
    j% = j% + 5
    _SetRect(VARPTR(aRect$(0)),j%,20,i%,100)
    _PaintOval(VARPTR(aRect$(0)))
  NEXT i%
END PROC

DEF PROC EraseOvals
  LOCAL j%
  j% = 20
  FOR i% = 100 TO 300 STEP 5
    j% = j% + 5
    _SetRect(VARPTR(aRect$(0)),j%,20,i%,100)
    _EraseOval(VARPTR(aRect$(0)))
  NEXT i%
END PROC

DEF PROC AnimOvals
  LOCAL j%
  j% = 20
  SetSolidPenPat(13) ' Light blue
  FOR i% = 100 TO 300 STEP 5
    j% = j% + 5
    _SetRect(VARPTR(aRect$(0)),j%,20,i%,100)
    _PaintOval(VARPTR(aRect$(0)))
    _EraseOval(VARPTR(aRect$(0)))
  NEXT i%
END PROC

DEF PROC Finis
  _MoveTo(10,150)
  _DrawString("Press any key to continue.")
END PROC

```

## PRODUCT INFORMATION

### AC/BASIC

Absoft Corporation  
2781 Bond Street  
Rochester Hills, MI 48307  
(313) 853-0050  
\$125

### Micol Advanced BASIC

Micol Systems  
9 Lynch Road  
Willowdale, Ontario M2J 2V6  
Canada  
(416) 495-6864  
\$145

### TML BASIC

TML Systems  
8837-B Goodbys  
Executive Drive  
Jacksonville, FL 32217  
(904) 636-8592  
\$125



BASIC lacks the intent and documentation to do desktop applications, even though the ability ostensibly is there.

However, TML offers only one high-level shortcut to QuickDraw graphics, the GRAF INIT command, which starts up QuickDraw and its interdependent toolsets. But sophistication in data handling takes a lot of the sting out of TML's tool interface.

**Listing 4** shows how to implement our simple animation techniques using TML BASIC. It also shows readily portable examples of other tool calls that may be useful to Micol programmers. TML has no ready-made shortcuts for displaying an SHR picture, as do the other BASICs. If you need this capability, you must program it in traditional Toolbox fashion as prescribed in the *Toolbox References* and in the TML manual.

In addition to strong support for desktop programming, TML includes a broad-ranging discussion of QuickDraw graphics in its manual—the best of any of the BASICs. On-disk source-code examples show how to implement every QuickDraw graphics primitive and a number of useful screen-display techniques.

## PROJECTS

There are many programs you might write to further explore QuickDraw graphics, while creating something worthy in the process. For instance, using AC/BASIC, you have access to powerful mouse-tracking commands that will let you program an art-capture function easily.

In conjunction with automated scaling demonstrated in **Listing 3**, this can become the basis for applications such as an SHR label-making program or a database that mixes text and graphics. Refer to chapter 16, "Advanced Memory," in the AC/BASIC manual.

## TRICKS OF THE TRADE

Micol Advanced BASIC has the speed, tools, and Applesoft compatibility to bring a lot of good but outdated software back to life. You can add backgrounds to Applesoft adventure games, or use object-oriented graphics to make a simulation or educational program more exciting. I like these kinds of projects because you can do them in a weekend and get a good upload to the networks with your name on it.

In addition, Micol promises to release an upgrade for the GS Advanced BASIC (version 3.0) by early this fall that should make graphics a snap, especially IIGS desktop windows, dialogs, menus, and mouse control.

With TML BASIC, the disk examples and manual treatment of QuickDraw enhance TML's status as the best compiler for desktop programming. You may even prefer it for game design, because of the wide availability of examples (in TML Pascal as well as BASIC). The only feature missing in TML is a GS/OS-compatible compiler update.

## MOVING ON

You don't need advanced knowledge to do significant graphics programming. After a couple of hours of experimentation, you'll have a great deal of confidence. And after a couple of projects, you'll no doubt feel—and program—like a graphics pro. □

CONTRIBUTING EDITOR JOE ABERNATHY IS A JOURNALIST WITH *THE HOUSTON CHRONICLE*. HE'S A CERTIFIED APPLE DEVELOPER AND THE AUTHOR OR COAUTHOR OF EIGHT APPLE II PROGRAMS. WRITE TO HIM AT P.O. BOX 66046, HOUSTON, TX 77266-6046. ENCLOSE A STAMPED, SELF-ADDRESSED ENVELOPE IF YOU'D LIKE A PERSONAL REPLY.