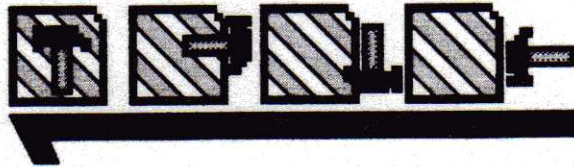


DrawTools 3.1



by Ken O. Burtch

Copyright 1992-3 by Pegasoft of Canada
Copyright 1992-3 par Pegasoft of Canada

For questions or comments, please write to the following address:

Pegasoft
Honsberger Avenue, R.R.#1
Jordan Station, Ontario, Canada
L0R 1S0

Some examples use libraries from ORCA/Pascal, Copyright 1991, The Byte Works, Inc.

Unless otherwise noted, trademarks belong to their respective companies.

Table of Contents

Part I. User Guide	
1. Introduction	pg. 1
2. Graphics on the IIGS	pg. 4
3. Animation	pg. 8
4. Other Functions	pg. 22
Part II. Reference	
Introduction	pg. 28
Housekeeping Tools	pg. 30
Low-level Drawing Tools	pg. 32
Drawing Tools	pg. 34
Library Management Tools	pg. 37
Animation Tools	pg. 38
Screen Tools	pg. 42
Scrolling Tools	pg. 45
Palette and Colour Tools	pg. 46
SCB Interrupt Tools	pg. 49
Printing Tools	pg. 51
Driver Tools	pg. 53
Miscellaneous Tools	pg. 56
Part III. Appendices	
Appendix A - DrawTools' error summary	pg. 60
Appendix B - Direct Page Usage	pg. 61
Appendix C - DrawTools and Other Toolsets	pg. 62
Appendix D - Game and Net Drivers	pg. 63
Appendix E - Pic Ed	pg. 66
Appendix F - Library Converter	pg. 67
Appendix G - Changes Since DrawTools 3.0	pg. 68
Tool Index	pg. 69

I. User Guide

1. Introduction

1.1. Introduction & Legal Stuff

The part of this manual is a general introduction to DrawTools. It isn't a tutorial on computer graphics, although some basic topics are discussed. For more in depth information on specific tools, consult the reference section.

We'd love to hear from you. If you have any questions, comments, or complaints, please feel free to write to Pegasoft at:

Pegasoft
Honsberger Avenue, R. R. #1
Jordan Station, Ontario, Canada
L0R 1S0

This manual and the related software contained on the diskettes are copyrighted materials. All rights reserved. Duplication of any of the above described materials, for other than personal use of the purchaser, without express written permission of Pegasoft of Canada is a violation of the copyright law of the United States and Canada, and is subject to both civil and criminal prosecution.

Pegasoft and DrawTools are trademarks of Pegasoft of Canada.

1.2. What is DrawTools?

Welcome to DrawTools, a collection of over 100 useful graphics and animation tools for the IIGS. The first version was released as shareware around the fall of 1990. Since then, it has significantly grown, with new features and more versatility.

Feel free to distribute the TOOL098 file with any programs you make, but if you wish to distribute any other files on the DrawTools disks, please get prior permission from Pegasoft.

1.3. System Requirements

DrawTools 3.1 requires the following:

An Apple IIGS with system software 5.0.2 and at least 9K free RAM in bank 0.

To use DrawTools, the following toolsets must be active: Tool Locator, Misc. Tools, Memory Manager, QuickDraw II.

1.4. Installation

1. Copy the TOOL098 file to the Tools folder of your startup disk. (This is DrawTools.)
2. Copy the DT.Drivers folder (the folder and its contents) to the System:Tools folder of your startup disk.
3. Copy the icon file to your Icons folder.

The DrawTools disks also contain the following:

- a. PicEd 3.0, a simple editor for picture libraries
- b. Lib.Converter 1.2, a utility which translates a screen template into a picture library. The folder includes some sample templates.
- c. Demo.Game, a small assembly language game that demonstrate some of the animation tools

- d. **Demo.Sys16**, a demo program written in Micol Advanced Basic 4.2
- e. sample programs for a wide variety of computer languages

1.5. Using DrawTools with ...

Complete/TML Pascal II - an interface file written in TML Pascal II is included on the disk in the TML.Pascal folder. Copy the object file to the folder containing the interface files for the other toolsets. Include DrawTools in your USES list at the beginning of your program.

Micol Advanced BASIC - You need to use the TOOLBOX command. A set of aliases are supplied for users with the latest version of BASIC: you can copy these into your program or you can use the INCLUDE command. Each alias requires a space after the tool name.



DrawShadow will not work unless you are running a stand-alone application. There are also some tools that require a Pascal string (not a BASIC string): a length (byte) followed by the text of the string. You cannot use these tools directly: you will either have to construct a string with POKEs, or use Micol Macro. All the toolsets that DrawTools requires are started for you when you use HGR or HGR2.

Merlin 16+ - a macro file (Draw.Macs.S) is included on the disk. Copy it into your MACRO.LIBRARY subdirectory, and USE it in your source files.

ORCA/Pascal - an interface file (Drawtools.int) is included on the disk. Copy this file into OrcaPascalDefs. In your program, include DrawTools in your USES list.

ORCA/M - A macro file (m16.DrawTools) is included on the disk. Copy this file into your ainclude folder. Use it like any other macro file.

ORCA/C - There are no interface files available: you can use DrawTools if you use the necessary tool definitions.



DrawTools will not work with **Prizm**.

Pegasus Pascal - Follow the ORCA/Pascal directions.

Example: Starting DrawTools 3.1.

Pegasus Pascal: Start it like any other toolset.

```
Uses Common, ..., DrawTools
| start required tools, or use StartGraphics
LoadOneTool 98, 0 | Load DrawTools
DPHandle = NewHandle(256, MyID, $C005, 0) | allocate direct page space
DP = ord( DPHandle* ) | convert to an integer
DrawStartUp DP, MyID | start DrawTools
ExtendBuffers | if using a lot of pixies
```

ORCA/Pascal: Start it like any other toolset.

```
Uses Common, ..., DrawTools;
{ start required tools, or use StartGraphics }
LoadOneTool(98, 0); { Load DrawTools }
DPHandle := NewHandle(256, MyID, $C005, 0); { allocate direct page space }
DP := ord( DPHandle* ); { convert to an integer }
DrawStartUp( DP, MyID ); { start DrawTools }
ExtendBuffers; { if using a lot of pixies }
```

BASIC: Use the following commands:

```

REM Start required tools, or use HGR/HGR2.
TOOLBOX(1, 15: 98, 0) : REM Tool Locator's LoadOneTool
DrawTools_Handle = 256 : REM Allocate direct page space
DrawTools_Address = 0
POKE 202, 1
Get_Mem( DrawTools_Handle, DrawTools_Address)
Address% = INT(DrawTools_Address): REM Convert to an integer
MyID% = Peek(238) + Peek(239) * 256
TOOLBOX( ~DrawStartUp : Address%, MyID%)
REM Without aliases: TOOLBOX( 98, 2 : Address%, MyID% )
TOOLBOX( ~ExtendBuffers ) : REM If using a lot of pixies

```

Merlin 16+: Start it like any other toolset.

```

USE 4:Draw.Macs
~LoadOneTool #98;#0
~NewHandle #100;MyID;#C005;#0
PLA
PushWord MyID
_DrawStartUp
_ExtendBuffers ; if using a lot of pixies

```

ORCA/M: Start it like any other toolset.

```

MCOPIE m16.DrawTools
ph2 #98 ; load DrawTools
ph2 #0
_LoadOneTool
ph4 #0
ph4 #100
ph2 MyID
ph2 #C005
ph4 #0
_NewHandle
pha
ph2 MyID
_DrawStartUp
_ExtendBuffers ; if using a lot of pixies

```

Examples: Stopping DrawTools 3.1.

Pegasus Pascal:

```
DrawShutDown
```

ORCA/Pascal:

```
DrawShutDown;
```

BASIC:

```

TOOLBOX(~DrawShutDown )
REM Without aliases: TOOLBOX(98, 3)

```

Merlin 16+:

```
_DrawShutDown
```

ORCA/M:

```
_DrawShutDown
```

2. Graphics on the IIGS

2.1. A Brief Introduction

Graphics is the art of drawing with a computer. In the IIGS, there is a special toolset dedicated to drawing called "QuickDraw II", or QuickDraw for short. QuickDraw provides all the basic drawing functions for the average application: it draws lines, rectangles, ovals, text, cursors and many other things you see on the screen. It's impossible to make a complete list of the QuickDraw tools here since there are well over 200; consult the Apple IIGS Toolbox Reference or any book introducing IIGS programming for more information.



BASIC: Whenever you use HCOLOR, HPLOT, or the other BASIC commands, BASIC uses QuickDraw.

Before discussing the details of DrawTools, you should know a little bit of how pictures are displayed on the IIGS screen. We will be discussing 320 mode to keep things simple. The super high-resolution graphics screen is located in bank \$E1 of memory. Each dot on the screen, or "pixel", consists of half a byte of memory, or 4 bits. This means up to sixteen colours can normally be displayed on the screen. The screen consists of 320 pixels horizontally and 200 pixels vertically. These pixels are located in the area \$E12000 to \$E19CFF of memory.

The next 200 bytes, starting at \$E19D00, are for the Scanline Control Bytes, or SCB's, one for each line on the screen. The SCB's determine the attributes for that line:

bit 0...3 - the palette of the line (0 to 15)

bit 4 - zero

bit 5 - 1 if fill mode is active. With fill mode active, colour 0 (usually black) behaves differently.

If you draw an area of the screen in colour 0, it will appear in the same colour as the area of the screen to the immediate left. The colour is "pulled" across the black areas of the screen, filling them in.

bit 6 - 1 will cause an SCB interrupt on this line

bit 7 - 1 for 640 resolution; 0 for 320 resolution

The memory located from \$E19E00 to \$E19FFF contains the 16 colour palettes (or "color tables"). Each palette contains 16 integer RGB values that describe the 16 colours you can see on the screen. QuickDraw only uses palette 0 (see Figure 1). Palettes and RGB colour words are discussed more below.

Figure 1: The QuickDraw II colours (in 320 mode)

#	Name	RGB	#	Name	RGB
0	black	\$000	8	flesh pink	\$FA9
1	dark grey	\$777	9	yellow	\$FF0
2	brown	\$841	10(\$A)	green	\$0E0
3	purple	\$72C	11(\$B)	light blue	\$4DF
4	blue	\$00F	12(\$C)	lilac purple	\$DAF
5	dark green	\$080	13(\$D)	periwinkle blue (desktop)	\$78F
6	orange	\$7F0	14(\$E)	light grey	\$CCC
7	red	\$D00	15(\$F)	white	\$FFF

This whole section of memory, from \$E12000 to \$E19FFF, can be "shadowed" from \$012000 to \$019FFF in bank 1. This area is called the shadow screen. You can use the shadow screen if you set bit 15 in the Master SCB when you start QuickDraw up. When the shadow screen is in use, drawing takes place much faster than usual. In addition, the shadow screen can be made invisible (with DrawTools ShadowOff) so that QuickDraw & DrawTools

draw many times faster than without shadowing, but the pictures will remain hidden until use DrawTools' QuickWipe.

2.2 Working with Colour

On the IIGS, the super hires screen can display 16 colours at a time with a single palette. You can change the current drawing colour using QuickDraw's SetSolidPenPat(c) or BASIC's HCOLOR=c. The hue and brightness of each colour is described by an RGB colour word, a combination of red, green and blue components. Each component can be in a range from 0 to 15. For example, black is 0,0,0; white is 15,15,15; bright red is 15,0,0; orange is 15,7,0.

DrawTools has a tool called SetColour that will take the red, green and blue components and give you the corresponding RGB value.

Example: Creating the colour "orange" with SetColour.

Pascal: RGBColour := SetColour(15, 7, 0);

BASIC: TOOLBOX(~SetColour : 0, 15, 7, 0; RGBColour%)

REM Include 0 at start for RGBColour%! Add one 0 for each result value.

Example: You can use QuickDraw's SetColorEntry to change a default colour:

Pascal: SetColorEntry(0, 5, SetColour(15, 7, 0));

BASIC: TOOLBOX(~SetColour : 0, 15, 7, 0; RGBColour%)

TOOLBOX(4, 16: 0, 0, RGBColour%)

Besides SetColour, there is a SetColPercent will do the same thing, accept you use percentages (0...100) of red, green and blue components instead of values from 0...15. FadeColour will make an RGB value darker or brighter. BlendColour will blend to colours together to make a new colour. FindColour will find the closest colour in a palette to the colour word you specify.

Although QuickDraw uses one palette, the IIGS can actually display colours from 16 different palettes at one time. Each line must have only one palette. DrawTools has a tool called SetPalette that will change the palette for a set of lines.

Example: Changing the top half of the screen (lines 0...99) to palette 1.

Pascal: SetPalette(0, 99, 1);

BASIC: TOOLBOX(~SetPalette : 0, 99, 1)

Now anything you draw on the top half of the screen will appear in the colours of palette 1 instead of palette 0. You can set the colours of any of the palettes using SetColorEntry(palette, colour, RGBvalue); or in BASIC, TOOLBOX(4, 16: palette%, colour%, RGBvalue%).

FadePal will make all the colours in a palette darker. UnfadePal will make all the colours in a palette brighter. A more powerful version of FindColour is FindPalette. Give FindPalette a palette of colours, and it will try to match them up to colours in the current palette. This tool is useful for NDAs: you can never be sure which colours are on the screen if an NDA is running under a paint program. FindPalette can tell you if the colours have changed, and to what.

Example: See the reference for more details. If you want to find the closest colours on the screen to the standard 320 palette:

Define the colours array:0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Define the Palette:\$000, \$777, \$841, \$72C, ..., \$78F, \$CCC, \$FFF

After the call is made, the values in colour list will change to reflect the actual numbers for these colours on the current screen (or the closest colours them).

