
Apple Manuals

Apple][Pascal

Macintosh

SwyftCard

Canon Cat

Jef Raskin Information

DOCUMENT TITLE

ETYMOLOGY OF QUICKDRAW

SOURCE

DOUG McKENNA MARCH 1992

DOCUMENT NO. 2

EX LIBRIS
DAVID T. CRAIG
736 EDGEWATER, WICHITA KS 67230 USA
E-MAIL: 71533.606@COMPUSERVE.COM


Apple Manuals

Apple][Pascal

Macintosh

SwyftCard

Canon Cat

"RaskinDoc002_0.PICT" 141 KB 2000-02-21 dpi: 300h x 300v pix: 2206h x 2875v

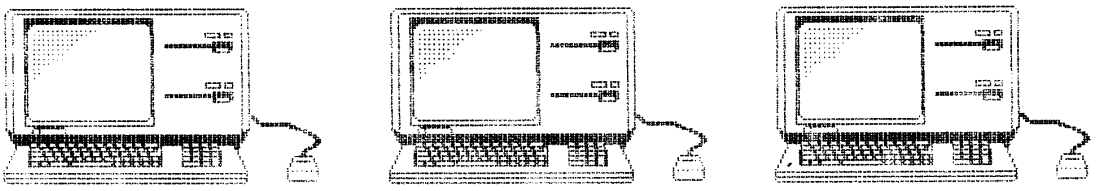
Doc #2

The Etymology of QuickDraw

Doug McKenna

09 March 1992

Jef Raskin • Bill Atkinson



"RaskinDoc002_01.PICT" 151 KB 2000-02-21 dpi: 300h x 300v pix: 2062h x 3113v

3 pages

So it's really early history you want, eh?

Here's some good stuff that really should get into the record. I realize it doesn't particularly address the technical aspects of boiling compiled Pascal down into assembler, but it's history nonetheless. Sorry about intertwingling the History of QuickDraw with the History of the Macintosh, but that's life...

FAX # 814-863-2653
814-865-3489 inter-lib loan dept. (Nov.93)

When he was a graduate student at Penn State University, Jef Raskin designed a computer graphics package for making drawings on both raster and vector displays. He was particularly interested in drawing music notation by computer, and to this day he has framed plotter output, drawn with his software, of some Beethoven on one of his walls. Raskin's display-list system was inspired by Evans & Sutherland's first stuff, and he submitted it as the work for his Master's degree. The system was designed to work speedily, and thus his thesis and software were entitled,

"The QuickDraw Graphics System". (Subtitle) call #: THESIS 1967 M RASKI, JE

This was 25 years ago, in 1967. Some of the routines in his software are still in use at Penn State. His thesis used "QuickDraw" as a single word (MiddleCap) although occasionally it was hyphenated as "Quick-Draw".

After graduating, he ended up in San Diego, and did a variety of things, the most important of which was becoming a professor of computer science at U. C. San Diego. In the early 70's, when so much wonderful stuff was starting to happen at Xerox PARC, Raskin was a visiting researcher at the Stanford A.I. Lab, and often visited PARC, which was the center of as much social activity as scholarly. For any of us who were in computer science school in the mid 70's, PARC was Mecca, with all their sweet revolutionary ideas (BitMaps, BitBlt's, Graphic Interfaces, Event Loops, Personal Computers, Dynabooks, etc.).

Raskin stopped teaching and moved north to the Silicon Valley area, where he formed his own documentation company, whimsically called Bannister & Crun. He contracted with Apple to write some of their first manuals, such as the Basic Manuals for the Apple II, and eventually, in early 1978, Raskin joined Apple to became employee #31.

One of the first things Raskin tried to get Apple to do right was to support some form of high-level language, because he felt that BASIC and assembly language were not going to be good enough in future products, and he knew that the programmers coming out of schools were learning Pascal. Because of the interpretive p-Code design of the UCSD version of Pascal, it was the only possible high-level language Raskin could find that would work in the Apple II's limited configuration. Raskin had trouble convincing Apple that this was important, and consequently, Raskin paid out of his own pocket the license fee for Apple to use the UCSD Pascal interpreter (Apple never reimbursed him for it). He had no particular preference for Pascal over anything else, other than that nothing else was available that didn't cost a lot.

To bring the Pascal system to the Apple II, Raskin wisely hired his former talented student from San Diego, Bill Atkinson, who had been thinking originally of a career in neurobiology but had discovered computers to be more to his liking.

DAVID T. CRAIG

During 1979, the two main projects at Apple were the Lisa and the Apple III, the latter of which was Steve Jobs' project. Raskin, having moved up to a management position, was convinced that both these projects were weak: the Apple III was not going to be a powerful enough advance over the Apple II, and the Lisa was going to be too expensive and cumbersome, to be successful. He proposed a new project to build a bitmap-display personal computer inspired by some of the things he had seen at Xerox, but he intended it to be much simpler, smaller, and cheaper than either the Lisa or PARC's Alto's. He code named the project "Macintosh" after his favorite Apple, and hired the original team of talented hardware, software, and user-interface designers. In hindsight, given what happened to the Apple III and the Lisa, it seems his analysis of the situation was entirely on the mark. Atkinson very much wanted to work on the Mac project, but Jobs was opposed, and it was about a year before Atkinson could start using his talents on the Mac.

Raskin's first design documents included the Mac's small footprint; its purely graphic engine requiring, for instance, square pixels; a one-button (not multi-button) mouse; and a built-in screen. Originally, the Mac's system was going to be a much more text-based interface (all done as BitBlt'ed (a.k.a CopyBits'ed) graphics, though, instead of using character generators); this was because Raskin recognized that nearly all computer usage was for text processing, and he had envisioned text as being the primary thing users would interact with.

During the early part of the project life, Raskin was on one occasion forced to appeal directly to Apple's Chairman of the Board (Mike Markula) to keep Steve Jobs from killing the Macintosh project. Fortunately, after Raskin defended the Mac, Markula sided with him, and because Apple was growing so fast, Raskin was able to sequester the Mac project in a different building ("skunkworks"), to help shield it from Jobs' interference.

Meanwhile, Atkinson worked on the graphics system for the Lisa, and he relied on many references for graphic algorithms, such as Bresenham's DDA line drawing algorithm, that he found himself or that his former professor had told him about, including Raskin's own thesis. Somehow, he or they simply started/continued calling the library "QuickDraw", and the name stuck. Nothing technical in Raskin's original system survived except the name, and Atkinson, in his usual manner, went to town and did an amazing job creating the basic graphics package we all know and love. When Atkinson eventually got to work on the Mac, he boiled the whole thing down to assembly, and since then, many folks have improved and extended QuickDraw.

Jobs eventually saw the Macintosh light, and when he took over the project in about 1982 Raskin resigned rather than put up with him. Under Jobs' direction, a lot of changes were made by many talented people to the prototype over the next 2 years, mostly good (graphical Finder, resources, sound, no fan, etc.) and some bad (no expansion slot, unnecessary disk formatting); Atkinson went on to write MacPaint (not the first raster graphics paint program, by any means - the first well-designed one was written by Alvy Ray Smith at NYIT - but certainly the most popular), and the rest is pretty well known.

Except that in the years just after the introduction of the Mac, Apple's "history-making" PR machine elevated (or allowed the media to elevate) Jobs to god-like status while Raskin's name basically got buried and forgotten. Apple

essentially went around letting the world know that Jobs had created the Macintosh. In fact, many people had added many great ideas to the Macintosh, and some of the important ones were due to Raskin's foresight 5 years earlier, which included hiring the right people and sheparding the thing through the birth pangs of what appears to have been a somewhat hostile environment. Raskin eventually had to inform Apple's legal department that all the hype was improper and beginning to hamper his ability to make a living; and so Apple stopped. Apple eventually presented Raskin with one of the six Mac Plus's designated "The Millionth Macintosh" to come off the assembly line.

I did not work at Apple during any of this, and the above has been distilled primarily from the many interesting stories Jef has told me over the years as I have worked with him on other things (the story about paying for the Pascal p-Code license out of his own money he told me a only day ago). These days, as a user-interface consultant, Raskin is just an ordinary Mac user with no special connections to Apple (nor an AppleLink account), and I personally think more people should know about his contributions to Apple's direction in the days of yore.

Anyway, that's the etymology of "QuickDraw" as I understand it.

Doug McKenna
Mathemaesthetics, Inc.
March 9, 1992

DTC Notes:
- Atkinson called QuickDraw "Lisa Graf" for awhile. When ported to Macintosh, some Mac people called QD "MacGraf" (see Chris Espinosa's 2/82 QD chapter draft for Inside Macintosh).

=FINIS=

FOUND THE THING I
WROTE LAST YEAR
IN RESPONSE TO AN
APPLELINK REQUEST FOR
INFORMATION ON THE
HISTORY OF QUICKDRAW

-DOUG MCKENNA

develop
Magazine
(issue #6)
[1991]

A BRIEF HISTORY OF QUICKDRAW

There have been a number of QuickDraw versions since the introduction of the Macintosh in 1984. Table 1 summarizes the major QuickDraw versions. Many minor revisions and bug fixes have also occurred along the way, of course.

Table 1
A Summary of Major QuickDraw Versions

Date	Version	Where Documented
Jan 1983	1.0 *	
January 1984	Original B&W QuickDraw (Macintosh 128K)	*Inside Macintosh* Volume I
January 1986	B&W QuickDraw (Macintosh Plus)	*Inside Macintosh* Volume IV
March 1987	Color QuickDraw B&W QuickDraw (Macintosh II)	*Inside Macintosh* Volume V
May 1989	32-Bit QuickDraw v. 1.0	*Inside Macintosh* Volume VI
September 1989	32-Bit QuickDraw v. 1.1 (System 6.0.4, Macintosh IIci, IIfx, IIsi, and LC)	*Inside Macintosh* Volume VI
March 1990	32-Bit QuickDraw v. 1.2 (System 6.0.5)	*Inside Macintosh* Volume VI
April 1991	Color QuickDraw B&W QuickDraw (System 7.0)	*Inside Macintosh* Volume VI

Note: QuickDraw is revised for system releases and, in the past, major revisions have coincided with hardware releases. In the future, it's likely that major system releases will be independent of hardware releases.

The version of black-and-white QuickDraw that accompanied the Macintosh Plus system added the SeedFill, CalcMask, and CopyMask calls. The Macintosh II revision introduced Color QuickDraw (which supported indexed devices only) and revised the existing black-and-white QuickDraw (which is still used on 68000-based machines) to display pictures (data of type 'PICT') created in the color version.

* LISA QUICKDRAW (CALLED LISAGRAF ORIGINALLY). B&W, DOCUMENTED IN LISA PASCAL LANGUAGE MANUAL.

1/3

Version 1.0 of 32-Bit QuickDraw, released as an INIT at the Developers Conference in 1989, added direct-color capability to QuickDraw. No black-and-white QuickDraw update was provided. Version 1.1 of 32-Bit QuickDraw is in ROM on the Macintosh IIci, IIfx, IIsi, and LC. Version 1.2 of 32-Bit QuickDraw, released as an INIT with System 6.0.5 and patched by the system on machines that have version 1.1 in ROM, added the OpenCPicture call and the capability of recording font names into pictures.

The System 7.0 version of Color QuickDraw integrates the functionality of 32-Bit QuickDraw into all Color QuickDraw machines and adds a variety of new features and bug fixes. In addition, System 7.0 has a new version of black-and-white QuickDraw that includes some of Color QuickDraw's functionality. (See "QuickDraw Features New in System 7.0" on the next page for more information.)

ABOUT COPYBITS

The CopyBits procedure, along with the CopyMask and CopyDeepMask calls, is the core of QuickDraw's image-processing capability. CopyBits transfers a bit image from one bitmap to another and clips the result to a specified area. With CopyBits you can perform such image-processing operations as resizing (by stretching, shrinking, or clipping the image), colorizing, and changing the pixel depth. You can use it to display on-screen the contents of an off-screen buffer.

In the System 7.0 version of QuickDraw, as in previous versions, the CopyBits procedure is defined as

```
PROCEDURE  CopyBits (srcBits,dstBits: BitMap;srcRect,dstRect: Rect;
                     mode: INTEGER; maskRgn: RgnHandle);
```

In the original black-and-white QuickDraw, CopyBits used six explicit parameters (srcBits, dstBits, srcRect, dstRect, mode, and maskRgn) and one global variable (thePort). The introduction of Color QuickDraw required an additional global variable, theGDevice, which is used to determine color information for the destination.

Although the number of variables used by CopyBits hasn't changed from earlier QuickDraw versions, several things *have* changed:

- The way transfer operations specified by the mode parameter are performed has changed to make their results predictable regardless of whether the destination device uses indexed or direct color.
- The way the notCopy transfer operation is performed has changed to improve the quality of color inversions.

2/3

QUICKDRAW FEATURES NEW IN SYSTEM 7.0

NEW IN COLOR QUICKDRAW

Custom drawing for specific screen depths. The DeviceLoop call lets applications do custom drawing for specific screen depths rather than having QuickDraw do the color translation. (If QuickDraw does the translation, a picture of a color wheel may turn out solid black when drawn on a black-and-white screen.) With DeviceLoop, you pass a drawing region, flags, and a pointer to a callback procedure that DeviceLoop will call for each different device that intersects the drawing region.

Picture Utilities. The Picture Utilities package ('PACK' 15) provides an easy way to profile the contents of a picture. It can tell you which fonts are used inside a picture so you can warn the user if one of the fonts is not available. It can also calculate the optimal color table or palette (using a predefined color pick method, or you can write your own) for displaying the picture.

Any bit depth for a mask. Before System 7.0, CopyMask's mask parameter could be only 1 bit deep. This caused the mask to be used very much like a region, selecting whether or not to copy a specific source pixel. In 7.0, the mask can be any bit depth. It specifies a blending value for merging the source and destination: black selects the source, white selects the destination, and gray provides a blend between the source and destination. Color masks can be used to blend only specific color components.

New version of the CopyMask call. The new CopyDeepMask call is an extension of CopyMask that includes a mode parameter and a region parameter. CopyDeepMask enables a blend of the source and destination to be applied to the destination using any transfer mode (not just srcCopy). Like previous versions of CopyMask, CopyMask and CopyDeepMask calls are not saved in pictures and do not print in System 7.0. (The resulting image can be printed, of course!) This may change in a future version.

NEW IN BLACK-AND-WHITE QUICKDRAW

New calls. The calls RGBForeColor, RGBBackColor, GetForeColor, GetBackColor, and QDError are now available in B&W QuickDraw.

Font names in pictures. Font names, rather than just font IDs (which may be different on different machines), are recorded into pictures, as in 32-Bit QuickDraw v. 1.2.

Custom drawing for specific screen depths. The DeviceLoop call, as described for Color QuickDraw, exists on all 7.0 machines, but because B&W QuickDraw supports only one screen device, the call is trivial.

Native resolution. OpenCPicture enables you to specify a picture's native resolution. This makes it easy to create pictures with resolutions other than 72 dpi. This feature was first available in 32-Bit QuickDraw v. 1.2.

Picture Utilities. See description for Color QuickDraw. In B&W QuickDraw, the Picture Utilities will not return a palette when you request color information.

Version 2 pictures. B&W QuickDraw previously could display version 2 pictures created on color machines, but could create only version 1 pictures. In 7.0, pictures created with OpenCPicture are version 2.

Display of 16- and 32-bit PICTs. Before System 7.0, B&W QuickDraw could display only PICTs containing indexed pixMap data; in 7.0, it can display pictures containing direct-color data.

1-bit GWorlds. In 7.0, 1-bit GWorlds are available in B&W QuickDraw. You must access the data with GetGWorldPixMap. You cannot dereference the GWorldPtr directly. On black-and-white machines, GetGWorldPixMap returns a handle to an extended bitmap (only 1 bit is supported), rather than a pixMap. You can then call GetPixBaseAddr to access the pixels.

3/3

DAVID T. CRAIG

Pascal
Reference Manual
for the Lisa™

1983

029-0391-A

QuickDraw

E.1 About This Appendix

This appendix describes QuickDraw, a set of graphics procedures, functions, and data types that allows a Pascal or assembly-language programmer of Lisa to perform highly complex graphic operations very easily and very quickly. It covers the graphic concepts behind QuickDraw, as well as the technical details of the data types, procedures, and functions you will use in your programs.

We assume that you are familiar with the Lisa Workshop Manager, Lisa Pascal, and the Lisa Operating System's memory management. This graphics package is for programmers, not end users. Although QuickDraw may be used from either Pascal or assembly language, all examples are given in their Pascal form, to be clear, concise, and more intuitive; Section E.11 describes the details of the assembly-language interface to QuickDraw.

The appendix begins with an introduction to QuickDraw and what you can do with it (Section E.2). It then steps back a little and looks at the mathematical concepts that form the foundation for QuickDraw: coordinate planes, points, and rectangles (Section E.3). Once you understand these concepts, read on to Section E.4, which describes the graphic entities based on them--how the mathematical world of planes and rectangles is translated into the physical phenomena of light and shadow.

Then comes some discussion of how to use several graphics ports (Section E.6), a summary of the basic drawing process (Section E.7), and a discussion of two more parts of QuickDraw, pictures and polygons (Section E.8).

Next, in Section E.9, there's a detailed description of all QuickDraw procedures and functions, their parameters, calling protocol, effects, side effects, and so on--all the technical information you'll need each time you write a program for the Lisa.

Following these descriptions are sections that will not be of interest to all readers. Special information is given in Section E.10 for programmers who want to customize QuickDraw operations by overriding the standard drawing procedures, in Section E.11 for those who will be using QuickDraw from assembly language, and in Section E.12 for those interested in creating three-dimensional graphics using the Graf3D unit.

Finally, there are listings of the QuickDraw interface (Section E.13), two sample programs (Section E.14), and the **QDSupport** unit (E.15); and a glossary that explains terms that may be unfamiliar to you (Section E.16).

E.2 About QuickDraw

QuickDraw allows you to organize the Lisa screen into a number of individual areas. Within each area you can draw many things, as illustrated in Figure E-1.

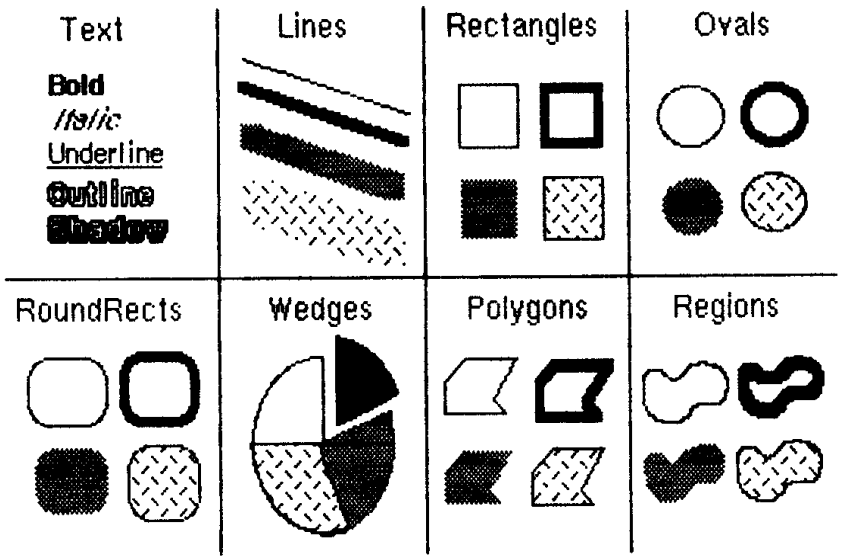


Figure E-1
Samples of QuickDraw's Abilities

You can draw:

- Text characters in a number of proportionally-spaced fonts, with variations that include boldfacing, italicizing, underlining, and outlining.
- Straight lines of any length and width.
- A variety of shapes, either solid or hollow, including: rectangles, with or without rounded corners; full circles and ovals or wedge-shaped sections; and polygons.
- Any other arbitrary shape or collection of shapes, again either solid or hollow.
- A picture consisting of any combination of the above items, with just a single procedure call.

In addition, QuickDraw has some other abilities that you won't find in many other graphics packages. These abilities take care of most of the "house-

keeping"--the trivial but time-consuming and bothersome overhead that's necessary to keep things in order.

- The ability to define many distinct *ports* on the screen, each with its own complete drawing environment--its own coordinate system, drawing location, character set, location on the screen, and so on. You can easily switch from one such port to another.
- Full and complete *clipping* to arbitrary areas, so that drawing will occur only where you want. It's like a super-duper coloring book that won't let you color outside the lines. You don't have to worry about accidentally drawing over something else on the screen, or drawing off the screen and destroying memory.
- Off-screen drawing. Anything you can draw on the screen, you can draw into an off-screen buffer, so you can prepare an image for an output device without disturbing the screen, or you can prepare a picture and move it onto the screen very quickly.

And QuickDraw lives up to its name! It's very fast. The speed and responsiveness of the Lisa user interface are due primarily to the speed of the QuickDraw package. You can do good-quality animation, fast interactive graphics, and complex yet speedy text displays using the full features of QuickDraw. This means you don't have to bypass the general-purpose QuickDraw routines by writing a lot of special routines to improve speed.

E.2.1 How To Use QuickDraw

QuickDraw can be used from either Pascal or MC68000 machine language. It has no user interface of its own.

If you're using Pascal, you must write a Pascal program that includes the proper QuickDraw calls, compile it against the files **QD/QuickDraw.OBJ** and **QD/QDSupport.OBJ**, link it with the files listed in **QD/QDStuff.TEXT**, and execute the linked object file.

If you're using machine language, your program should include the proper QuickDraw calls, and **.INCLUDE** the file **QD/GRAFTYPES.TEXT**. Assemble the program, link it with the files listed in **QD/QDStuff.TEXT**, and execute the linked object file.

A programming model, **QDSample**, is included with the Workshop software in the file **QD/QDSample.TEXT** (listed in Section E.14.1); it shows the structure of a properly organized QuickDraw program. What's best for beginners is to read through the text, and, using the superstructure of the program as a "shell", modify it to suit your own purposes. Once you get the hang of writing programs inside the presupplied shell, you can work on changing the shell itself.

Note that all files related to QuickDraw are prefixed by "**QD/**".

QuickDraw includes only the graphics and utility procedures and functions you'll need to create graphics on the screen. Procedures for dealing with the

mouse, cursors, keyboard, and screen settings, as well as those allowing you to generate sounds and read and set clocks and dates, are described in Appendix F, Hardware Interface.

E.2.2 QuickDraw Data Types

QuickDraw defines three general data types, **QDByte**, **QDPtr**, and **QDHandle**:

```
type QDByte   = -128..127
     QDPtr    = ^QDByte
     QDHandle = ^QDPtr
```

Other data types are described throughout this appendix in the sections in which they're relevant. For a summary of all QuickDraw data types, see Section E.13.2.

E.3 The Mathematical Foundation of QuickDraw

To create graphics that are both precise and pretty requires not super-charged features but a firm mathematical foundation for the features you have. If the mathematics that underlie a graphics package are imprecise or fuzzy, the graphics will be, too. QuickDraw defines some clear mathematical constructs that are widely used in its procedures, functions, and data types: the *coordinate plane*, the *point*, the *rectangle*, and the *region*.

E.3.1 The Coordinate Plane

All information about location, placement, or movement that you give to QuickDraw is in terms of coordinates on a plane. The coordinate plane is a two-dimensional grid, as illustrated in Figure E-2.

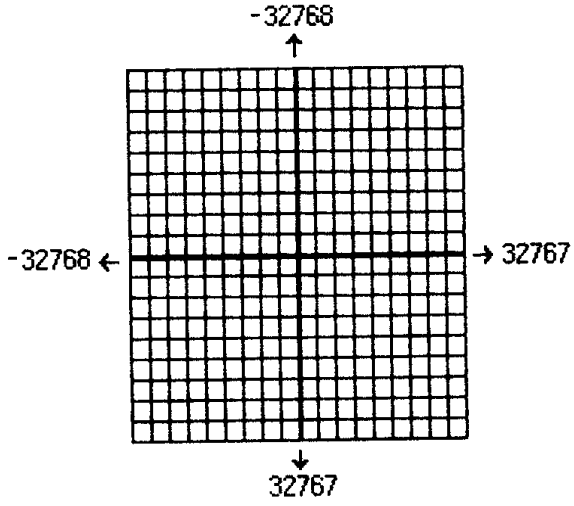


Figure E-2
The Coordinate Plane