MacSciTech Review

The Macintosh Scientific and Technical Users Association (MacSciTech) promotes Mac scientific, engineering, and industrial applications. The March 1993 issue of its SciTech Journal included a review of the programming tools currently available to Mac developers. Excerpts from this review are presented below. The author is Mark Worthington who has "been programming the Mac since 1988 and consults on various engineering and programming projects".

Programming Your Macintosh: It's Not Just For Propellerheads Anymore by Mark Worthington

High Level Languages (HLLs)

Writing a complete stand-alone Macintosh application in a high-level language requires a lot of code to manage windows, menus, and events. The simpler your underlying objective (calculate the Bessel function), the more creating a Mac interface is a needless distraction which dominates your real work. As an example, the Pascal code for MiniEdit - a simple non-styled text editor which doesn't print - takes up 51 pages at the end of volume two of Stephen Chernicoff's wonderful Macintosh Revealed. And only a small fraction of that does the text editing.

To write a real Macintosh program that doesn't do anything but display an uneditable window, you've got to know the following areas of the Macintosh Toolbox: the Memory Manager, GrafPorts & QuickDraw, the Resource Manager, the Segment Manager, the Event Manager, Windows, and Menus. If you want Chernicoff's simple MiniEdit, you can add TextEdit, Controls (buttons & scrollbars), the scrap (clipboard), and Dialogs. ...

Ultra-High Level Programming

...The most general purpose and widely available program and one of the pioneers in this category is HyperCard. Bill Atkinson, its inventor, billed it as "a software erector set", an apt description. A new HyperCard stack is the blankest of blank slates. You place onto any number of blank cards in the stack the basic visual tools of the software trade: buttons, text fields, text, and paint. You switch to writing text-based scripts in HyperTalk to describe the action you want the various objects to take.

HyperCard has generated a lot of criticism from frustrated stack authors who have run up against its limits, but the fact is, you can get an awful lot of sophisticated work done in HyperCard, fast, and it throws in all the other stuff - handling windows, update events, controls, scrolling, etc. - for free. ...

Visual Programming Tools

...With Serius, you essentially "wire together" pre-compiled mini-applications provided by Serius (or written yourself in C or Pascal, but that's what you're trying to avoid). Serius may find itself competing with Interapplication Communications in the future.

Hooking together mini-apps doesn't give you much flexibility and I doubt most scientists and engineers will find this approach useful. ...

Prograph on the other hand provides a visual interface builder and an industrial-strength visual language that is truly object oriented. Although the Prograph class library handles most of the standard Mac behavior for you, the language provides complete access to the Macintosh toolbox. The debugger has gotten raves from users and application source code can be edited without recompiling. The source code, fully editable, is completely visual. Prograph doesn't spit out C code for you to compile; what you see is the source code.

The primary user complaint about Prograph seems to be that while Prograph ranks with HyperCard in ease of learning generally, and the documentation explains each piece of Prograph well, it doesn't do a good job orienting new users, so a larger investment of time is required for your first project than necessary. The other problem is that it can be difficult to follow large amounts of visual source code, though that may also be a function of its object orientation. ...

Prototypers

A prototyper is a tool for creating quickly and graphically the visual user interface elements of a program (windows, buttons, menus, dialogs, icons, etc.), as well as a few basic actions (show an Alert box) with user actions (pressing a button labeled "Self-destruct"). Within the prototyper environment, your menus pull down, buttons hilite, and graphical elements in general behave just as you would expect them to, so you can get an idea of how the front end of your final creation is going to operate. When you have completed the prototyping process, you can then save your work in the form of a resource file filled with your menus, dialog items, etc., and a file of ready-to-compile source code. You then add your application-specific code and compile.

The leading prototyper tool today is AppMaker from Bowers Development. AppMaker generates code for THINK C & Pascal, MPW C & Pascal, MacApp and THINK Class Library object frameworks, Language Systems FORTRAN, and the cross-platform environment called XVT.

I bought an earlier prototyper a few years back called Prototyper 1.0. I didn't find it useful. It was hardly any more trouble to create my resources with ResEdit, the links between buttons, menus, and dialogs was minimal, and the source code was basically a shell that never really changed. And I didn't like the shell. ...

Canned Libraries

Canned (pre-compiled) libraries callable from high-level languages are available commercially, as shareware and public domain. They range in complexity from precision mathematical routines to word processors, sophisticated database engines, and DXF compliant CAD/CAM routines. Some also have source code available. The number of these libraries prohibits further discussion in this space, except for one which really defies categorization.

ViewIt from FaceWare is a better product than its marketing. It is a canned library, but like no other. Like a prototyper, it lets you create and edit windows populated with buttons, text, and fields. But it doesn't create the source code for the interface like a prototyper. Instead, ViewIt creates these windows, etc. from code resources as intelligent objects which know how to manage themselves. It also handles application level concerns such as the menu bar and event handling. All this is accomplished by calls in your source code to a single procedure, "FaceIt", with varying parameters. The entire source code for the application in Figure 3 came to 2 pages.

ViewIt comes with no hardcopy documentation; it is all online but printable. While I understand keeping costs down (ViewIt is only $95), it makes it difficult for newcomers to the product to understand just what the heck ViewIt is. Perhaps FaceWare could offer a small printed "Overview and Getting Started" manual, possibly as an option for a few dollars.

ViewIt recently became available as shareware in a version which includes nearly everything the commercial package does.

Object Frameworks

MacApp and the THINK Class Library are the two major object frameworks for the Mac. ... Object oriented programming (OOP) stresses reusability and modularity of HLL code. Frameworks are source code to handle all the windows, events, and other general Mac behavior. Object classes are hierarchical and inherit their parent's behavior. The result is a more efficient programmer.

But there is a costly learning curve associated with OOP and object frameworks. I suspect that cost is prohibitive for all but professional programmers.

MacSciTech
49 Midgley Lane
Worcester, MA 01604
508-755-5242