A Road Map to the Yerk Manual

Before you dive into Yerk, it will be most helpful if you have an overview of the organization of this rather hefty manual.

Yerk was designed as both a serious Macintosh development language and a language that beginning, casual programmers can learn to access the powers of the Mac. This manual addresses both audiences, and some in between.

The manual is divided into the following major parts:

INTRODUCTION
- PART I. TUTORIAL
- PART II. USING YERK
- PART III. PREDEFINED CLASSES
- PART IV. THE YERK GLOSSARY
- APPENDIXES

INTRODUCTION.
This section is for everyone, regardless of expertise. It should be the first section of the manual you study, since it instructs you on how to make backup and working copies of your Yerk disk. Since the disk included in this package does not contain the Macintosh System Folder, the instructions in the Introduction are very important for all Yerk users.

PART I. TUTORIAL.
We have devoted a large section of the Yerk Manual to a 19-lesson tutorial, which will lead the beginner -- even someone who has never programmed a computer before -- and more experienced programmers through the essential elements of Yerk. We recommend that every Yerk owner, regardless of expertise, work through the Tutorial. Even if you think parts of Yerk resemble a language you already know, there will be significant differences that will be reinforced in the Tutorial. Plan to spend ten to twelve separate sessions over several days with the lessons in the tutorial. Yerk is based on Forth, and if you have some familiarity with that language, it will certainly help. Yerk adds the power of the Object Oriented Paradigm to traditional Forth, as well as other very useful tools.

PART II. USING YERK.

Six chapters in this Part provide numerous details about Yerk, including considerable expansion on topics touched on in the Tutorial. Read this part only after going through the Tutorial. Chapters in this part include:

1. The Yerk Menu Bar
2. Using an Editor

3. Writing New Classes
4. Advanced Yerk Constructs
5. Putting Together a Yerk Application
6. Utility Modules

Chapters 1 and 2 are good reading for Mac beginners, since they provide operational details about the Yerk menus. Chapters 3 and 5 should be studied with great care. They contain many ideas and suggestions to help you plan and develop applications for your own use or for commercial distribution. It is also a good idea to at least read through Chapter 4 to gain a familiarity with some of the more advanced features of Yerk. You will be able to write sophisticated programs without the facts in Chapter 4, but some of the ideas presented there may spark further thoughts about your own programs. Chapter 6 is a quick summary of three utility modules that you might find useful for your application development.

PART III. PREDEFINED CLASSES.

This part of the manual will become one of the most used reference sections once you begin writing Yerk programs. It contains details about the parts of the Yerk language that have already been written for you to help you communicate your program ideas to the Macintosh's unique way of doing things. Each chapter is devoted to a category of predefined classes, and begins with a general discussion about the class. You should be familiar with the content of Part III before writing programs.

PART IV. THE YERK GLOSSARY.

As you'll learn in the Tutorial, Yerk is largely a language based on a list of defined words in memory. Yerk, itself, has almost 1000 predefined words. The meanings of many of those words are detailed in this Glossary. The Tutorial will show you how to use the Glossary effectively. The Glossary, too, will become a much-used reference section once you begin programming in Yerk. All entries are listed according to the way the words sort on an ASCII-based sorting program (i.e., generally alphabetical, with numbers and symbols listed according to their associated ASCII value -- described fully in the Tutorial). Once you finish with the Tutorial, you may want to scan through the words in the Glossary to discover further predefined building blocks you have at your disposal.

APPENDICES.

In two appendices, we present further reference material of value to Yerk programmers. The first is a listing of all possible error messages that can occur during programming, debugging, and the running of a program. Along with the errors are also one or more possible solutions to each error condition. Next, we provide a summary section of useful information from Inside Macintosh.

Additionally, the Yerk disk in this package contains many files with the source code listings for many parts of Yerk. These files are actually further documentation for you. You will get instructions in the Tutorial about how to print these files and organize them for ready reference.

CONVENTIONS USED IN THIS MANUAL

We use a couple of conventions throughout this manual that you should be aware of.

In the Tutorial, we present many examples of things you should type into the computer. To differentiate the characters you type from the characters that the computer generates on the screen, we **boldface** those characters you type. The computer's prompt and other responses are printed in regular type.

In both the Tutorial and the chapters in Part II, whenever we introduce a new Yerk term, or intend for you to pay special attention to the terminology, we underline the key words. When you see underlined words and phrases, it means that we are trying to acquaint you with a new term or re-emphasize a term or concept already mentioned.

You will also find many cases in the manual and in the Yerk source code of Yerk words being capitalized in what may seem odd places, such as in the middle of a word like bArray. While this style of capitalization is common among experienced programmers for the sake of ease of reading, rest assured that you won't have to master any scheme of capitalization in learning Yerk. Yerk, itself, is case insensitive, which means that you can type a word in all lower case, all upper case, or any combination thereof, and Yerk will recognize it as the same word. As you become more experienced in Yerk, the capitalization standards we have set will make more sense to you.

Special note to experienced Forth and Smalltalk programmers: Yerk shares some characteristics of both Forth and Smalltalk. Much of Yerk's stack-based, threaded architecture is inherited from Forth, and we would like to acknowledge Yerk's debt to the Forth researchers that have brought that language to its present state. In like manner, Yerk's object-oriented features owe a lot to the work done in the Smalltalk group at Xerox PARC. While Yerk departs significantly from both languages, we are grateful to the ideas that provided its foundation. Experienced programmers in either Smalltalk or Forth should take care not to jump to any conclusions regarding Yerk's behavior on the basis of previous experience, and to read carefully through the tutorial.

• • •