

Fourth Dimension Development System

Introduction

The following paper describes a new application development tool available for Apple Macintosh computers call Fourth Dimension (4D). This program has been available in Europe for several years, and there are currently over 4,000 4D developers within France, its country of origin. Terms which appear in *italics* are defined in a glossary at the end of this paper.

Fourth Dimension has been called a database manager, a development system, a database language interpreter, an information query interface, and a file system/communications bridge. In my opinion 4D is all of these and many others its users have only begun to explore. Its importance as a new step in the evolution of human/computer interaction arises from the fact that it provides a synthesis of tools such as Macintosh user interface, relational database, communications, query/reporting, and extendable programming language. It is an environment for orchestrating all of the various elements of such activities and whose foundation is a high end, multi-user database management capability.

Fourth Dimension, hereafter referred to as 4D, is published by;
ACIUS, 20300 Stevens Creek, Suite 495, Cupertino, CA 95014

4D is a hybrid tool, providing several different "*environments*" for the creation and use of information. An inexpensive *runtime* version is available so that programs with a *turnkey*, off the shelf appearance and behavior can be created.

4D's suggested list prices are \$695 for the full *developers* system and \$250 for four (4) *runtime* systems. Discounted prices are available for volume purchasers, dealers, and educational institutions. At the time this paper is being written 4D is scheduled to be announced to a limited audience of Macintosh programmers at the MacHack 87 Macintosh developers conference on June 11th, and formally on or about July 1st, 1987.

Why Macintosh developers and users should care

At the time of its introduction in 1984 the Macintosh provided two choices for users, MacPaint and MacWrite, and one choice for developers, buy a Lisa and its Pascal programmers development system. Things changed very little for the first year, a trickle of programs in 1985, and a flood of applications and compilers in 1986. The primary reason that software for the Macintosh was delayed is that Macintoshes are hard to program. Concepts that most programmers heard in computer science lectures but seldom use are ever present in Macintosh code. No matter what language a programmer might choose there are hundreds of additional ROM routines and data structures with which they must be familiar. File systems, screen size, available memory, ROM, and other aspects of the Macintosh have continued to change and evolve over the years, adding another point of confusion to the issue.

Better, powerful, more easily learned programming tools are desperately needed. Products which address this need have emerged during 1987 include MacApp object oriented development system, MacExpress and Programmers Extender third generation language libraries, and enhanced development features for database management systems. These are all important contributions, and each contains one or more of the pieces of the puzzle. What sets Fourth Dimension apart from these other products is the completeness and attention to detail of its coverage of all these areas. 4D has been used to create conference registration, airline reservation, point of sale, electronic mail, computer conferencing, bulletin board, personnel, inventory, and medical information/billing applications. Its power and versatility meet or exceed that of any other database product with which I am familiar on micro, mini, or mainframe computer. Fourth Dimension is a world class product which I predict will have a major influence on the way programs are designed and implemented for years to come.

In the remainder of this paper I will outline the problems faced by Macintosh developers (and bitmapped, iconic interface programmers in general), and the way in which Fourth Dimension addresses these problems. These comments are based on strong opinions about the way computers and humans should interact, and a profound dissatisfaction with the majority of existing programming languages and development systems. I have programmed computers for 15 years, not because I love writing code but because I need programs which no one else has or will write. With better tools I can get back to the real work and creative expression which interest me and forget words like operating system, syntax error, and type coercion.

Statement of the problem

There are three basic functions of a computer information handling or database program; STORAGE, RETRIEVAL, and MANIPULATION of information. Features which support and expand these functions are desirable, while those which interfere with the speed and flexibility of these three functions are limiting and unwanted. There are several measures of information management software performance which are commonly used, most of which are associated with recurring weak points and failures in existing and historical products. In my formulation, which is admittedly subjective but which I find useful these measure of performance are;

- Efficiency of data storage (size data files and indices occupy on the storage device)
- Speed of execution (how fast data is recovered, sorted, tabulated, and displayed)
- Modification and reorganization (adding or deleting of fields, changing field type)
- Data import and export (reading and writing information to/from other programs)
- Backup/data integrity (protection from loss or corruption of data)
- Multiple *views* of data (create and use many different entry and report formats)
- Multi-file relationships (ability to combine data from several files into a single *view*)
- Ease of use (time to learn, convenience of data entry, error handling and help)
- Power (works well with large numbers of records, high level programming options)

Most commercial database products are weak or fail to satisfactorily address these important functions, and at this writing no database product adequately provides tools for the creation of *turn-key applications*. As a product 4D has gone further than any other on microcomputer, minicomputer, or mainframe in providing solutions to these requirements. It is not a perfect product, but like the first Macintoshes it points the way towards a dramatically new relationship between programmer, user, and information management. Unlike the earliest 128K RAM Macintosh 4D is capable of providing a significant percentage of its potential immediately, thanks to the pioneering efforts of its author and the experience gained from tens of thousands of European developers and users.

In the following sections of this paper I will provide a short introduction to many but not all of the features of 4D which relate to the requirements listed above, as well as commentary on the advantages or limitations of portions of the product. Please remember that these are the opinions of the author, based upon roughly six months of beta testing and application development experience but subject to error due to the complexity and evolving nature of the product. I will present what I have learned and imagined in as honest a manner as possible, trusting the reader to use judgement in their own explorations of Fourth Dimension.

The Three Basic Environments

4D provides one or more *environments* as indicated on the table below. Also shown are the differences between *runtime* and *development* versions with regard to the availability of these *environments*. A further elaboration of this chart indicating the effect of passwords and startup procedures on the accessibility of *environments* will be provided later.

	Developer System	Runtime System
Design Environment	yes	no
User Environment	yes	no
Custom Environment	yes	yes

DESIGN ENVIRONMENT

allows the creation and editing of files, fields, subfields, links, layouts, procedures, menus, and passwords. It is in this environment that the structure of the datasets are defined and the different views used to enter and report data constructed.

USER ENVIRONMENT

A default set of tools for the actual entry, modification, deletion, display, search, sort, graph, and reporting of data using the structure created in the Design Environment.

CUSTOM ENVIRONMENT

A programmer controlled environment containing dialogs, windows, menus, controls and other portions of the Macintosh user interface allowing users to interact with the various datasets, layouts, and other features of the database in a unique manner.