A Brief History of Apple Computer's Work with the Pascal Language

Written by David T. Craig
736 Edgewater, Wichita, Kansas 67230
07 October 1992

David T. Craig

TABLE OF CONTENTS

INTRODUCTION

This document provides a concise history of the work that Apple Computer has done with the Pascal computer language. This history was written so that this rather interesting bit of computer technology would be recorded for those with an interest in technical histories like this.

The author has never worked for Apple Computer, but has been involved with software development for Apple's many computers since 1978 and has extensively used all of Apple's Pascal implementations. As such, the facts contained herein are based upon personal memories and Apple technical documents and may be inaccurate. The author hopes that those in the know about Apple's Pascal work will correct and extend upon this paper so that the history of Apple's Pascals will be made complete.

APPLE][COMPUTER PASCAL

Apple first became involved with the Pascal language for its Apple][computer series around 1978. At this time Apple's software development efforts were based upon the BASIC and 6502 assembly languages.

The key players in bringing Pascal to the Apple][were Bill Atkinson and Jef Raskin. Atkinson convinced Apple's management that the Pascal language would be much better for Apple's own software development activities than either BASIC or 6502 assembly. Atkinson and Raskin licensed the Pascal language and development system from the University of California at San Diego (UCSD) and began an internal Pascal development group.

Apple's][Pascal system included a Pascal compiler, modal editor, 6502 assembler, a powerful filer, and several utility programs (e.g. program segment mapper). The system was controlled via a command line user interface.

The Apple][Pascal compiled to P-Code which was based upon byte codes and a stack architecture. This P-Code was interpreted by an interpreter written in 6502 assembly. 6502 assembly language routines could also be linked with P-Code programs. Identifiers in][Pascal were recognized only up to 8 characters, a painful limitation which Apple did not remove until its Macintosh MPW Pascal. Pascal's memory use was restricted to 64k bytes due to the Apple][hardware architecture.][Pascal supported program modules called UNITs which could be either regular or shared (a.k.a. intrinsic). Units could be segmented so that they would reside in memory only when needed.][Pascal also supported a fairly sophisticated graphics unit called Turtle Graphics created by Atkinson.

Technical documentation for][Pascal was reasonably good and consisted of a language reference and an operating system manual.

Several major Apple][programs were developed in Apple][Pascal, including Apple's *Apple Presents the Apple][* by JDEisenberg and *PFS File* by PFS Inc.

Apple produced several versions of Apple][Pascal: 1.0, 1.1, 1.2., 1.3. As of the middle 1980's Apple seems to have abandoned Pascal for the Apple][. Current Apple][Pascal work is done via an Apple-developed Macintosh cross-development system called Macintosh Programmer's Workshop.

APPLE /// COMPUTER PASCAL

When Apple developed its Apple /// computer (1980 - 1985) it developed most of its software for this machine with Pascal.

Apple used the Apple][Pascal system to create the Apple /// Pascal system. As such, /// Pascal generated P-Codes for a stack-based architecture. The ///'s Pascal language syntax was extended (e.g. *otherwise* clause in case statements) and like Apple][Pascal also supported separate program units. Access to the ///'s native 6502-based operating system, SOS (Sophisticated Operating System, or Sara's OS), also existed via a special Pascal unit called SOSIO. This OS unit also allowed programmers access to the ///'s larger memory, 256k bytes vs. 64k bytes for the][computer.

Apple produced several versions of Apple /// Pascal: 1.0, 1.1, 1.2, 2.0. Version 2.0 of /// Pascal was unique in that it came with extensive technical documentation called the /// Pascal Workbench and consisted of around 1,000 pages. Version 2.0's Pascal compiler also produced a compilation listing consisting of the generated P-Code interleaved with the Pascal source code lines. /// Pascal also supported via a conditional compilation directive the compilation of Apple][Pascal programs and produced Apple][format code files.

Documentation for the /// Pascal system was excellent. This included a language manual, an operating system manual, and a very detailed runtime architecture manual which also listed all the P-Codes.

For /// Pascal Apple developed a very sophisticated floating-point implementation called SANE (Standard Apple Numeric Environment). SANE was based upon the IEEE floating-point standard. /// Pascal versions 1.0 and 1.1 supported only 4 byte REAL numbers, version 1.2 supported the higher quality 8 byte EXTENDED numbers. SANE went on to become the floating-point engine for Apple's Lisa and Macintosh computers.

Most of Apple's software for the /// was developed with the Pascal system. This included the *System Utility Program* and *Backup ///*.

The key players for Apple /// Pascal were Ira Rubin and Al Hoffmann.

Support for /// Pascal ended when Apple discontinued the /// computer in 1985. From this date onwards Apple seems to have ended its involvement with Pascal which generated P-Code, opting instead to deal fully with native code generators.

APPLE LISA COMPUTER PASCAL

The 68000-based Lisa computer (1979 - 1985) was unique in that the majority of its system and application software was written in a powerful extended version of Pascal called Lisa Pascal. This software included the operating system (90,000 lines), the Workshop development environment (100,000 lines), 7 application programs (each around 50,000 lines [e.g. LisaWrite, LisaCalc]), and an avalanche of development support programs (e.g. mouse-based editor, telecommunications program, and Pascal language utilities).

Apple's Lisa Pascal developments began from scratch for Apple when it licensed in 1981 a Motorola 68000 native code Pascal compiler from Silicon Valley Software in California. This compiler was based upon the older P4 compiler from Niklaus Wirth of ETH in Switzerland and consisted of two general passes. Pass 1 produced I-Code, a low-level representation of the high-level Pascal constructs. Pass 2, the code generator, converted the I-Codes to optimized 68000 object code. Apple even considered early in the Lisa's development using a custom Apple processor which would execute P-Code directly, but the expense of developing such a chip was too much for Apple's accountants and this project was dropped.

All Lisa development was done with the Lisa Workshop, a command line based development environment similar to Apple's earlier][and /// Pascal systems. The Workshop provided an excellent development environment with its Pascal compiler, mouse-based editor, 68000 assembler, and an extensive collection of utility programs (e.g. Format, XRef, ProcNames). The Workshop supported other languages which included BASIC, COBOL, and C.

Apple produced several versions of Lisa Pascal: 1.0, 2.0, 3.0. Version 3.0 was the last version and supported the creation of intrinsic units which allowed a single copy of a unit to exist in the system and be used by several programs simultaneously. All versions supported language extensions such as *otherwise* clause in case statements, *longint* 32 bit integers, @ operator for data addresses, data type coercion, and relaxed ordering of const/type/var sections. A set of bit-based functions were also supported (e.g. *BITAND*) to make bit bashing easier. Inline 68000 assembly code could also be used.

Like the Apple /// Pascal, Lisa Pascal also supported SANE, Apple's IEEE floating-point implementation. In addition to SANE Apple developed a fairly comprehensive math package called MathLib. Apple also developed an IEEE floating-point test program which attempted to ring out the bugs in SANE (Jerome Coonen of Apple wrote this test program and its associated test vectors).

Lisa Pascal was a superset of Apple's older][and /// Pascals and, as such, was able to compile][and /// Pascal programs which did

not make any][or /// system calls. Lisa Pascal could also produce a compilation listing complete with the Pascal source code interleaved with the generated 68000 assembly statements.

Apple also developed the Lisa QuickDraw (a.k.a. LisaGraf) graphics library which formed the heart of what was called Lisa Technology. Tho written in 68000 assembly, Apple provided an extensive Pascal interface to this rather remarkable graphics environment (QuickDraw was later patented by Apple and Bill Atkinson, QuickDraw's creator).

When Apple introduced the Macintosh computer in 1984 all Macintosh programming by Apple and others was done with Lisa Pascal and the Lisa Workshop. For this activity Apple created an extensive set of libraries containing the Macintosh toolbox interfaces. Many important Macintosh programs were written in Lisa Pascal, including MacPaint, MacWrite, MacDraw, and MacTerminal.

The key players behind Lisa Pascal were Ira Rubin and Al Hoffmann.

Apple provided extensive documentation for the Lisa Workshop and Lisa Pascal which included a language manual, an operating system manual, and a Workshop manual. Many internal memos were also released by Apple which described special features of Lisa Pascal.

Lisa Pascal was also used by Apple to develop the Lisa Clascal object-oriented language and the Clascal-based Toolkit class library.

Apple discontinued the Lisa in 1985 in favor of the Macintosh and when Apple's native Macintosh development environment (MPW) became operational in 1986 Apple dropped support for Lisa Pascal and the Lisa Workshop. Therefore, Lisa Pascal lasted from 1981 to 1986, an eternity in the field of microcomputer languages.

APPLE LISA COMPUTER CLASCAL

Apple's first foray into the world of object-oriented programming was with the Lisa Clascal (Classes + Pascal) language. Developed from 1983 to 1985 Clascal was an extension of Lisa Pascal which included several new keywords such as *methods* and *subclass*. The main concepts behind Clascal were derived from Simula and Smalltalk, Xerox PARC's experimental object-oriented language.

In addition to Clascal Apple also developed an extensive set of class libraries which were collectively called the Lisa Toolkit. This class library allowed Lisa developers to create Lisa desktop applications (a.k.a. tools) in a fairly short time frame.

Tho really used only internally by Apple for its own R & D efforts, Clascal was made available to external developers but without any support by Apple (David Redhed of Seattle Washington created an independent Clascal/Toolkit support group called the Toolkit User's Group [TUG], but this unfortunately did not last too long). Apple considered enhancing Clascal, but this Clascal-85 effort was extinguished by Apple's work with Object Pascal for the Macintosh.

Several significant programs were developed with Clascal and the Toolkit including a calendar program, Apple's Lisa-to-Macintosh data conversion program, a desktop publishing program, and a fuzzy logic program.

The key player behind Clascal was Larry Tesler, one of the Lisa computer's designers, and several former Xerox Smalltalk experts.

When Apple dropped all support for Clascal and the Toolkit it provided the Toolkit source code to the public, but the source code to the Clascal compiler was, alas, not made public.

APPLE MACINTOSH COMPUTER PASCAL

Apple's Pascal for the Macintosh computer (1984 - present) was a port of Apple's earlier Lisa Pascal compiler. When Apple began development of the Macintosh (1982) Apple used Lisa Pascal and the Lisa Workshop for system software development.

With the introduction by Apple of the Macintosh Programmer's Workshop (MPW) in 1986 Apple had a very professional development environment which from an extendability perspective surpassed the Lisa Workshop. MPW tools could be written that would be executed by selecting their names in a mouse-based window environment and performing the *do-it* command by pressing the Enter key on the keyboard (shades of Smalltalk). MPW provided a Pascal compiler, 680x0 series assembler, an improved 680x0 linker, and a cornucopia of source code utility programs (e.g. PasMat, PasRef). Many languages were also supported, both from Apple and outsiders (e.g. C, FORTRAN, COBOL, Modula-2).

MPW Pascal extended Lisa Pascal by supporting the newer Motorola 68000 processors, the 68020, 68030, and 68040. Also supported were the Motorola floating-point and memory management chips. Conditional compilation directives allowed MPW Pascal to support the direct generation of floating-point chip instructions so that floating-point speed could be as fast as possible. Identifiers in Pascal programs were recognized up to 63 characters, a limitation which existed in all prior Apple Pascals beginning with Apple][Pascal.

Several versions of MPW Pascal existed: 1.0, 2.0, 3.0, 3.2. Version 3.2, the current 1992 version, was created to support development of System 7 programs and had several extensions to make this compiler support very large programs and support the calling of C functions with variable number of parameters.

MPW Pascal has improved upon the optimization of object code and includes such features as peep hole optimization and better register allocation of local and global constants.

Extensive documentation exists for MPW and MPW Pascal.

MPW Pascal was enhanced to support object-oriented programming and was renamed to Object Pascal. This extended version was used by Apple to develop the Macintosh class library called MacApp.

Apple also contracted with THINK Technologies who created an interactive Pascal learning environment that was called Macintosh Pascal (this interpreter was later turned into a real compiler).

Around 1988 Apple started to use the C and C++ languages for its own internal software development. The ascendancy of C/C++ at Apple has initiated a marked descendancy of Pascal at Apple and by outside developers.

APPLE MACINTOSH COMPUTER OBJECT PASCAL

Object Pascal for the Macintosh was developed by Apple starting in 1985 to support more rapid and more standardized development of Macintosh programs. Available for only MPW, Object Pascal is a descendant of the Lisa Clascal compiler.

The key Apple player behind Object Pascal was Larry Tesler who recruited the help of Niklaus Wirth, the creator of Pascal, to clean up the syntax of Clascal.

Object Pascal was used to develop the extensive MacApp class library. This library was fully documented by Apple via several books and the source code for MacApp was provided to developers.

Object Pascal is being eclipsed by C/C++ at Apple and most outside developers are following Apple's lead. The Object Pascal MacApp class library has been rewritten in C++.

SUMMARY

Apple's involvement with the Pascal language was extensive during the 1980's. From Apple's introduction to Pascal on the Apple][computer in 1978, it has advanced to the powerful Macintosh 680x0 series Pascals of 1992.

Since 1988 Apple's reliance on Pascal has been eclipsed by the emergence of C/C++ as Apple's preferred system and application development language. Tho Pascal will have the support of a small but vocal minority at Apple, C/C++ will be the dominant development language for Apple and outsiders for the next decade.

That's all Folks !