eWEEK® Enterprise IT Technology News, Opinion and Reviews

30 Years Ago: Turbo Pascal, BASIC Turn PCs Into Programming Engines

By **Darryl K. Taft** | Posted 2013-09-05

30 Years Ago: Turbo Pascal, BASIC Turn PCs Into Programming Engines

Even as *eWEEK* celebrates its 30th anniversary, Borland International's Turbo Pascal, a seminal work in the evolution of software development tools, will celebrate its 30th anniversary in November 2013.

Speaking fondly of "the good old days," upon request for a comment on programming in the '80s, Turbo Pascal creator Anders Hejlsberg (http://en.wikipedia.org/wiki/Anders_Hejlsberg) told *eWEEK* it was ironic that he was just thinking of how November will mark the 30th anniversary of Turbo Pascal as the first truly integrated development environment (IDE) (http://www.webopedia.com/TERM/I/integrated_development_environment.html) . "It's a bit scary how long ago that was," he said.

Turbo Pascal and other early PC development tools helped to democratize software development. The magic of Turbo Pascal was that it integrated an editor, debugger and compiler into one tool, which made things easier for developers.

"The '80s were a fun time—the beginning of the democratization of computing that we've all lived for the past 30 years," Hejlsberg said. "Performance, capacity, reliability or any other metric really of machines back then was pretty horrible. And the development tools—if you could even call them that— were even more so.

"But of course it was all better than the nothingness that preceded it. The BASIC interpreters of the time were pretty easy to use, but programs ran very slow. Some C and Pascal compilers were available, but line-oriented editors and super-slow compilation speeds that required you to switch floppies between compiler passes were just insanely painful," he said.

"The edit-compile-debug cycle could best be characterized as glacial. There was a very real need for a development experience that coupled productivity with efficient code," said Hejlsberg.

The first PCs allowed hobbyists and budding developers to try their hand at writing BASIC applications. But as enterprises started deploying PCs in large numbers, professional developer started looking for PC tools to write applications in almost every imaginable language, including Pascal, FORTRAN, COBOL, C, C++ and beyond.

Microsoft got its start selling BASIC interpreters for the personal computers, and development tools continued to grow into an important part of its business. Other companies, such as Borland, arrived on the scene in the 1980s and also made fortunes in development tools.

Turbo Pascal included a compiler and an IDE for the Pascal programming language running on CP/M, CP/M-86 and DOS, developed by Borland under co-founder and CEO Philippe Kahn's leadership. Kahn saw an opportunity in the development tools world where programmers saw their workflow in terms of the edit/compile/link cycle, with separate tools dedicated to each task.

Programmers wrote source code (http://en.wikipedia.org/wiki/Source_code) and entered it using a text editor (http://en.wikipedia.org/wiki/Text_editor) , a compiler then created object code (http://en.wikipedia.org/wiki/Object_code) from source (often requiring multiple passes) and then shifted to a linker (http://en.w ikipedia.org/w iki/Linker_(computing)) that combined object code with runtime libraries to produce an executable program.

Kahn's idea was to package all these functions in an integrated programming toolkit, have it run with much better performance and charge one low price for it all. Instead of selling the kit through established sales channels such as retailers or resellers, his new tool would be sold inexpensively via mail order. The product sold for $49.95.

"Being a developer myself I needed a tool that would make me more productive," Kahn told *eWEEK*. "I had worked on integrated development environments as a student of Niklaus Wirth at the ETH [*Eidgenössische Technische Hochschule]* in Zurich. I give a lot of credit to Niklaus for inspiration.

"Then it was a matter of putting it all together. My partner wrote the compiler, and it was fast. Yet most of the development time was used in going back and forth between source code and execution. Then debugging the executed program. So we decided that a fast one-pass compiler was the way to go because it allowed us to pinpoint runtime errors right into the source code," Kahn said.

30 Years Ago: Turbo Pascal, BASIC Turn PCs Into Programming Engines

"Once we integrated the whole system, we had an almost instantaneous and continuous 'Edit -> Compile -> Run -> Debug -> Edit' productive cycle. The efficiency gains were fantastic. These are the kind of tools that we are building today at Fullpower/MotionX(http://w w w .fullpow er.com/) for wearable technology," Kahn said.

Indeed, Kahn told *eWEEK* he believes "Turbo Pascal was to the original IBM PC what Xcode is to the iPhone: a fast, powerful, efficient, interactive integrated development system." Moreover, "Before Turbo Pascal, all the professional tools that generated executable code on the PC were multi-pass, command-line style compilers," he said. "Turbo Pascal set a new standard and changed all of that."

"Borland made developer tools accessible by selling a complete editor/compiler for an incredibly low price," Mike Sax, founder of Asigo (http://asigo.com/) , an accounting software company, said. "I believe it was 10 times cheaper than leading competitors. At least as important, and often overlooked, Borland made tools more accessible by requiring far less memory and disk space than other vendors, combining the editor and the compiler in a single, incredibly efficient executable. These were the days when PCs without hard drives were still common."

Sax knows a bit about development tools and the early software market. He launched and ran a successful software business for years. Sax Software specialized in creating software components to help developers build better software faster.

Borland greatly sped up the development cycle because developers could launch the compiler and eventually their application directly from the editor, Sax said. "In all other systems, you had to edit your code, run the compiler, restart the editor if you had made a typo, and run your app," he said.

"When developing for Windows 1.0, the development cycle was even worse: Edit your code, quit the editor, run the compiler, reboot the computer (Ctrl+Alt+Delete) into Windows, start your app, exit Windows and reboot again, and start the editor," Sax said.

"So Borland's tools weren't just 10 times cheaper; they actually let you develop software 10 times faster. Furthermore, the programs that were written in Turbo Pascal took advantage of the same highly optimized runtime that the compiler/editor itself was using, resulting in very small and fast applications (which was extremely important at the time)."

Like many of the early revolutionary PC products, Turbo Pascal was viewed by many as a toy. But many of the ideas ended up being incorporated in other products. For instance, Microsoft released QuickBasic shortly thereafter, as well as its CodeView visual debugger, Sax said.

The Turbo Pascal compiler was based on the Blue Label Pascal compiler originally produced for the NasSys cassette-based operating system of the Nascom (http://en.w ikipedia.org/w iki/Nascom) microcomputer in 1981 by Hejlsberg. Borland licensed Hejlsberg's "PolyPascal" compiler core and added the user interface and editor. Poly Data was the name of Hejlsberg's company in Denmark. He then joined Borland as an employee and was the architect for all versions of the Turbo Pascal compiler and the first three versions of Borland Delphi (http://en.w ikipedia.org/w iki/Embarcadero_Delphi) .

"Before embarking on my first Pascal compiler, I had written an interactive (WYSIWYG) editor and a symbolic assembler in Z-80 assembly code for a British kit computer called the NASCOM 2," Hejlsberg explained.

"I actually enjoyed assembly coding and found that I could be very productive and could write code that was really small. The NASCOM came with a 12K Microsoft ROM BASIC, so I decided to write a plug-replacement 12K Pascal environment. It had an interactive editor, a subset Pascal compiler and a runtime library, all written in assembly code, Hejlsberg said.

"The very simple one-pass compiler compiled directly to in-memory machine code or, for large programs, to cassette tape, which you could then rewind and reload later. Having the editor and compiler in a single integrated IDE was just the natural thing to do," he noted.

30 Years Ago: Turbo Pascal, BASIC Turn PCs Into Programming Engines

"They were both part of the same ROM, and the system wouldn't have made sense without both," Hejlsberg said. "This 12K Pascal subset implementation grew into a more complete implementation for CP/M-80, and then further evolved into the product called Turbo Pascal in a collaboration with the original Borland founders."

The other product that was at least as revolutionary as Turbo Pascal was Visual Basic, Sax said. Cooper Software, headed by Alan Cooper (http://en.w ikipedia.org/w iki/Alan_Cooper) —known as the "Father of Visual Basic (http://w w w .cooper.com/alan/father_of_vb.html) "—had developed a replacement for the first Windows shell that he called "Tripod" and later renamed "Ruby." You could draw controls like text-boxes and buttons onto a window or form, Sax said. Fellow developer Mike Geary also was part of the team.

In his post on his involvement with Visual Basic, Cooper said he began to show his prototype around the industry. "In March of 1988, I showed this prototype to Bill Gates (http://w w w .cooper.com/alan/bill_gates.html) , and he immediately saw its potential," Cooper wrote. "He declared that it was 'cool' and that it would have significant impact across their entire product line. Bill said he wanted to buy it, and over the next few months we hammered out a deal. Because the name Tripod had had so much exposure, we promptly changed it to 'Ruby.' Meanwhile, I put together a team of skilled programmers to build a release-quality program."

Sax says Geary developed some more controls like the directory and file folder boxes that are now part of every File Open dialog box. "I believe it was Bill Gates who decided that Ruby should be extensible with controls that weren't originally part of the product," he said.

"Microsoft bought Ruby from Cooper Software, and then the Windows Shell didn't get replaced after all. However, the QuickBasic people decided to combine their p-code compiler with Ruby and created Visual Basic," Sax said.

"When Visual Basic came out, it revolutionized Windows programming by making it cost-effective (in terms of development time) and doable (in terms of the learning curve) for people to write business applications for Windows," said Sax. "This also spawned a whole new development tools industry, championed by Tom Button at Microsoft."

And it all started with BASIC. Microsoft BASIC was Microsoft's foundation product. It first appeared in 1975 as Altair BASIC, which was the first BASIC by Microsoft and the first high-level programming language available for the Altair 8800 microcomputer. The Altair BASIC interpreter was developed by Microsoft founders Paul Allen (http://en.w ikipedia.org/w iki/Paul_Allen) and Bill Gates with help from Monte Davidoff, using a self-made Intel 8080 software simulator running on a Digital Equipment Corp. PDP-10 minicomputer.

The early software development tools industry was incredibly incestuous. Cooper early on worked with Gordon Eubanks (http://en.w ikipedia.org/w iki/Gordon_Eubanks) to develop, debug, document and publish his business programming language, CBASIC (http://en.w ikipedia.org/w iki/CBASIC) , an early competitor to Gates and Allen's Microsoft BASIC.

Eubanks later became CEO of Symantec. At one time, Symantec was also known for its development tools, particularly THINK Pascal, THINK C (http://en.w ikipedia.org/w iki/THINK_C) , Symantec C++ and Visual Cafe (http://en.w ikipedia.org/w iki/Visual_Cafe) packages, which were popular on the Macintosh and IBM PC-compatible platforms. These product lines resulted from acquisitions made by the company in the late 1980s and early 1990s. Symantec exited this business in the late-1990s as competitors such as Metrowerks (http://en.w ikipedia.org/w iki/Metrow erks) , Microsoft and Borland gained significant market share.

In 1992, Borland sued former Borland-turned-Symantec executive Gene Wang, Symantec CEO Eubanks and the Symantec corporation for misappropriation of Borland trade secrets (http://w w w .w ired.com/w ired/archive/1.02/undelete_pr.html) and unfair competition.

30 Years Ago: Turbo Pascal, BASIC Turn PCs Into Progamming Engines

Wang, who had been a Borland vice president, abruptly left to join archrival Symantec. Borland claimed he left incriminating evidence of plans to steal Borland secrets. Borland also pursued criminal charges in the case, and both Wang and Eubanks were indicted. However, the criminal case was quietly dropped with no one going to jail.

In his book "In search of Stupidity: Over 20 Years of High-Tech Marketing Disasters," Merrill R. (Rick) Chapman said, "From its inception, Borland International was the *Animal House* of high tech, a group of self-proclaimed barbarians who broke all the rules and had all the fun."

Meanwhile, despite the onset of the IDE, many code warriors preferred and still today prefer to simply use a text editor for creating software. "For much of my early career, my mantra was simply give me EMACS and a place to stand and I will move the world," said Grady Booch, co-creator of the Unified Modeling Language (UML) and chief scientist for software engineering at IBM Research.

"EMACS was for the longest time the primary development platform for many because it was so highly customizable and so very useful. But, one must remember that the nature of software development in the '80s was quite different than that of today," Booch observed.

"Back then, we were just making the transition from structured methods and languages to object-oriented ones; we mostly built larger programs, not smaller apps; and the vast majority of the things we built were stand-alone fortresses with little interaction with other such fortresses," he said.

At this time, noted Booch, "the rich primordial soup that became the Web was just beginning to move from its humble origins at CERN."

Booch, who is also a computer historian (http://w w w .ew eek.com/c/a/IT-Infrastructure/IBM-Chief-Scientist-to-Launch-TV-Series-on-Computing-340515/) , said, "There's no doubt that Turbo Pascal was a game changer; the collective functionality it brought, the (for the time) advanced user interface, and certainly its astonishing price point brought modern IDEs to the masses. But remember also at the same time we had the first public release from Xerox PARC of Smalltalk, which was both a language and development environment as well as a new way of thinking about programming," he said.

"Maestro from Softlab Munich was also out in the wild; though little known in the U.S., save for the defense community, it played an important role in the history of development environments," he recalled.

"This, by the way, was also the time in which Rational (http://en.w ikipedia.org/w iki/Rational_Softw are) was developing its Ada (http://w w w .w ebopedia.com/TERM/A/Ada.html) Development Environment, which for us was both software as well as hardware—the R1000 ... because there were no machines yet powerful enough to run our IDE."

Every era of computing has its nuances when it comes to development. The '80s and early '90s were no different.

"Writing software back then was really a craft," Hejlsberg said. "To get maximum efficiency, you had to write in assembly code. But not just that. In order to squeeze your program into 64K and still have room for the user's data, you had to hand-optimize the assembly code, reusing common instruction sequences, rearranging code to take advantage of short jump instructions, shortening error messages to gain a few bytes here and there.

"It was a fun puzzle—if you're into that sort of thing. The entire Turbo Pascal 1.0 executable—editor, compiler, runtime library—was only 33K, about a third the size of today's jQuery in minified form," Hejlsberg observed.

Moreover, "Developer productivity is always a key driver," Kahn said. "It will always be. In some ways, the state of the PC in 1983 was similar to the state of wearable computing today in 2013: the next paradigm shift in need of great tools and solutions."