GIZMODO (/)
(/)

How Steve Wozniak Wrote BASIC for the Original Apple From Scratch (http://gizmodo.com/how-steve-wozniak-wrote-basic-for-the-original-apple-fr-1570573636)

Steve Wozniak

Filed to: HISTORY (/TAG/HISTORY) Yesterday 6:52pm (http://gizmodo.com/how-steve-wozniak-wrote-b

To celebrate the 50th anniversary of BASIC, (http://bit.ly/1koOQMm) *Steve Wozniak has written some memories about his first experiences with this popular language—and how he created his own BASIC from scratch for the Apple I and Apple II computers. An incredible feat. Enjoy!—JD*

In 1967 or 1968, as a senior in high school, our electronics teacher (the best teacher of my life in many teaching regards) arranged for me to go to a company in Sunnyvale (Sylvania) to program a computer because I already knew all the electronics in class at school. Mr. McCollum did this for students with electronics abilities every year, finding local companies with engineers and projects that would let high school students come and and get some experience. I learned and programmed in FORTRAN on this IBM computer.

I first experienced BASIC in high school that same year. We didn't have a computer in the school but GE, I think, brought in a terminal with modem to promote their time-sharing business. A very few of we bright math students were given some pages of instruction and we wrote some very simple programs in BASIC. I saw that this was a very simple and easy-to learn language to start with, but that terminal was only in our school for a few days. I was asked by our head math teacher to write a page on why I thought this would be a good thing for our school to have. I did indeed write a page about logical thinking and problem solving but our school didn't go ahead and get into the time-share program.

In my college years my higher level programming languages were of the scientific mold, FORTRAN, PL-1, Algol.

Of course I programmed a lot in many assembly languages too, in college and on my own.

In the Homebrew computer club we had a couple of books going around that I like to call 'bibles'. One was Computer Lib/Dream Machine by Ted Nelson, describing a future world of hyperlinks to further the meaning of things in writing. His ideas were like science fiction but we all knew that they were achievable, technically, and we were all apostles for this way of looking at the future of computing. The other 'bible' was a book "101 Games in BASIC." I was a fan of computer games and knew that as soon as I had a computer of my own I would want to type in all these games to play. Judging by my own feelings, I assumed that this would be a key to starting a home computer revolution. The non-businessman in me prevents me from talking about markets or finance.

I did not know for sure what a real computer needed to do the big financial jobs that computers did for companies, the computers that sold for huge amounts of money. Those were worthwhile computers. All I knew was that which was close at hand. I ran simulations of chip designs and logic designs at HP, working on calculators. My computer would have to do that. My computer would also have to play games. At least then I was sure that my computer could possible do the important things that high priced computers do, but I wasn't sure.

The key to games was BASIC. Bill Gates was unknown except in the electronics hobby world. Everyone in our club knew that he'd written BASIC for the Intel microprocessor. I sniffed the wind and knew that the key to making my computer good (popular) was to include a high-level language and that it had to be BASIC. Engineers programming in FORTRAN were not going to be what would start a home computer revolution.

Learning the language and writing a BASIC interpreter

The problem was that I had no knowledge of BASIC, just a bare memory that it had line numbers from that 3-day high-school experience. So I picked up a BASIC manual late one night at HP and started reading it and making notes about the commands of this language. Mind that I had never taken a course in compiler (or interpreter) writing in my life. But my friend Allen Baum had sent me xerox copies of pages of his texts at MIT about the subject so I could claim that I had an MIT education in it, ha ha. In my second year of college I had sat in a math analysis class trying to teach myself how to start writing a FORTRAN compiler, knowing nothing about the science of compiler writing. I went back to this memory and started writing code for my 6502 microprocessor to read in lines that a user typed for analysis and error checking.

I knew about syntax charts and created one for this BASIC. I did not know that HP BASIC was extremely different than DEC BASIC, the one that the book "101 Games in BASIC" used and that Bill Gates had written. I figured all BASIC's were the same but HP's differed greatly when it came to strings of letters. I finished my syntax diagrams and they were complete. But I had in the back of my head that I could be a star, that I could get a little fame in the hobby world, like Bill Gates, if I created the first BASIC for the 6502. To save a bit of time, I stripped the floating point operations out of my syntax diagram. I needed to write integer based simulations for my HP work and games are based on logic, which is integers. I dropped floating point numbers (ones with decimal points) only to save a few weeks and have a better chance of being the first to develop this BASIC on the 6502. You might look back and see that I included floating point arithmetic routines in the ROM of the Apple][but I never went back to work them into the BASIC. When you code by hand (couldn't afford a time-share

account) it's hard to make changes in the middle structure of things that have to be at fixed addresses.

Breakout *on Integer Basic running on the original Apple II.* (http://www.applefritter.com/content/integer-basic-breakout-ala-1977-rev-0-apple-ii)

I didn't know about compiler writing other than perchance. But I did know about stacks and things like converting expressions into RPN using stacks. Our HP calculators used RPN, in fact. Thinking about how to write this language I came up with my own techniques, not like anything out of a book. I wound up with what I called NOUN and VERB stacks (operands and operators). I put tags in my syntax diagram but every operator in it had a number according to it's linear position in this 256-byte or 512-byte (I forget) table. The 41st operand had the code of operator #41.

I also had a list for every operator of 2 priorities. One was its tendency to go ahead of other operators. For example, the + operator would cause a * operator to occur first. But I needed a second table of the resistance to being pushed into action to handle things like parentheses. I had no idea if I was on a correct track but it worked correctly and did what I needed. It didn't have to come from a book.

I enjoyed demonstrating this BASIC at the Homebrew Computer Club. I never saw my name in print so I didn't get that 'Bill Gates' fame, but I was known in my club. This was even before Steve Jobs saw that my computer existed. As time went by I only had to write one routine after another for each of those numbered operators. Every club meeting I'd have a few more commands that worked fully.

With the Apple][I had the video and computer memory one and the same so that the microprocessor, changing maybe a million (exaggerated) numbers a second, would change a million screen bytes a second. Atari arcade games were hardware then but now the games could be implemented as software, using 6502 machine language programming. BASIC is an interpreted language. BASIC goes over the individual letters of each statement as it executes, determining what to do. It is maybe 100 or 1000 times slower than machine language as a result. But one day I was curious as to whether you could program moving objects just in BASIC and see them move like realistic animation.

I had designed Breakout for Atari in hardware. I wondered if I could program this simple animated arcade game in BASIC? I knew I could program it in machine language. Since it was my own BASIC I went to the syntax chart and added commands to plot color and to draw horizontal and vertical lines. I then searched chip manuals and chose a chip with 4 timers (555 style timers) on one chip. I used that with some software to read paddle positions off potentiometers, dials that changed resistance according to where you turned the dial. Once I had these mechanisms installed (burning new EPROMS for the BASIC additions) I sat down and wrote some simple FOR loops to plot bricks in different colors. I must have tried 30 color combinations in just minutes. Then I added paddles and score and a ball. I could adjust program parameters to change the ball speeds and angles. By the way, I think this is when I added a speaker with 1-bit audio just because you needed sounds when a ball hit a brick, etc.

I called Steve Jobs over to my apartment to see what I'd done. I demonstrated to him how easily and instantly your could change things like the color of the bricks. Most importantly, in one-half hour I had tried more variations of this game than I could have done in hardware over 10 years. Steve and I both realized how

important it was going to be now that animated (arcade style) games could be software. More than that, being in BASIC meant that anyone of any age could program it.

I kept about 50 chronological folders of papers throughout all my BASIC design work. Each one was labelled GAME BASIC. So you can see where my head was coming from.

Update

Another note: In high school or my first year of college I told my dad that someday I'd own a 4K Data General NOVA. He said it cost as much as a down payment on an expensive house. I was stunned and told him I'd live in an apartment.

Why 4KB?

Because that was the minimum needed to run a higher level language. To me a computer had to have more than switches an lights. It had to be able to run programs.

I'd built a switches and lights computer of my own design 5 years before the Apple I. Back then there was no way to afford 4KB of RAM so it had 256 bytes only.

In the Homebrew days, the summer of 1975, 3 companies introduced 4K DRAM's. This was the first time that 4KB was truly affordable. I had to have that much in order to have BASIC be a part of this computer. No choice about it. Hence the minimum RAM on an Apple I or Apple][was 4KB. Had I not cared about BASIC, I probably would have just built another switches and lights computer with minimal static memory and been done with it.

Editor's note: *If you want to know more about Steve Wozniak's life, you can check out his biography* iWoz. (http://www.amazon.com/iWoz-Computer-Invented-Personal-Co-Founded/dp/0393330435/ref=la_B001HPHPIC_1_1?s=books&ie=UTF8&qid=1398988369&sr=1-1&tag=gizmodoamzn-20&ascsubtag=[type|link[postId|1570573636[asin|0393330435[authorId|5722109538904747664)*—JD*