GUEST ARTICLE

Self-Teaching a Love of STEM: A Personal Tale

by Christopher Mitchell

Christopher Mitchell is currently pursuing a Ph.D. in Computer Science at NYU. Within the community of calculator programmers, he is a renowned afficionado. He recently published a book for beginners about programming the TI-83/84 graphing calculators, which you can order at manning.com/mitchell**.**

At the age of five, I was obsessed with trains. One day, my mother took me to the New York Transit Museum, which was holding a workshop about electricity, complete with batteries, light bulbs, magnet wire, and compasses. From that day, I knew I wanted to study electrical engineering, and began teaching myself about circuits, gadgets, and later, programming. I learned several programming languages, designed and built gadgets and hardware modifications, and eventually earned three degrees in electrical engineering and computer science. Relatively early in my programming and engineering career, I started to do teaching of my own, first online, and later for continuing education and undergraduate classes. I developed a conviction that all students and fledgling coders and engineers deserve the same opportunities to explore and teach on their own that I was given.

After my introduction to electronics at the Transit Museum, I sought out electronic design books, asked for Radio Shack's then-admirable Forrest Mims 130-in-One and 300-in-One kits, and taught myself about components, circuits, and even the rudiments of the underlying math. In early elementary school, I learned LOGO and toyed with QBASIC, but I continued to mostly focus on designing and building circuits. Once I received my first graphing calculator, however, my focus began to shift.

When I was in seventh grade, I got a trusty TI-83 for Christmas. At first I thought it was little more than a fancy math tool, but as I began to use it more, I discovered it was something else entirely. I found that it had a program editor, and that by putting together a few commands, I could make the calculator do my bidding. The concepts of programming were not entirely foreign to me, thanks to my earlier exploration of LOGO and QBASIC, but I began to build a much greater breadth and depth of knowledge as I worked with my calculator. I started with simple animations, creating programs that drew and erased characters to create primitive ASCII art. After seeing a few games on friends' calculators, I took faltering steps into what I later learned was reverse-engineering. I examined other programs, figured out how they worked, and used my new knowledge to improve and expand my own projects. I got involved in the international TI calculator programming and hobbyist community and published some of my projects. I fielded feedback, compliments, and criticism, and learned to grow as a person, a programmer, and even a marketer of my own work.

The TI-BASIC language that I had learned was easy but powerful, and I learned to do a great deal with it. However, I was frustrated to notice that some of the programs I encountered seemed far more advanced and powerful than anything I could make. When I tried to view their source code with the calculator's built-in editor, I was confronted by a sea of random symbols. I eventually learned that these programs were written in z80 assembly language, created on a computer and assembled into a form the calculator could understand. Over a summer, I began to work with the language, at first painstakingly typing out the hexadecimal for each opcode on my calculator, and later gaining access to a computer to use an assembler.

By gradually honing my skills, I learned about the internals of processors, memory, and I/O, skills that matured into a love of low-level programming and hardware design as an undergraduate and graduate electrical engineer. I wrote a graphical shell for the TI-83+/84+ calculator, a mouse-based GUI library, games, a music and video player, and even a decentralized networking protocol. Looking back at all of my experiences with graphing calculator programming, I realize that it reinforced my enjoyment of working with circuits and taught me to enjoy hacking in the positive sense. I enjoy the challenge of making an extremely low-resource device do as much as possible, and pushing myself to complete projects that others might dismiss as impossible. Having taken myself from the simplest commands in BASIC to complex hand-coded z80 and x86 assembly, I decided I wanted to share my love of coding (and particularly calculator coding) with the masses.

When I was still taking my early steps with TI-BASIC, I founded an online forum and community website called Cemetech ("KE-me-tek"). I used it to publish my own programs and projects, but also began to use it to amass skilled and beginner programmers alike, who learned from each other and began to post their own projects. To date, Cemetech has amassed about three thousand users, and incubated software and hardware projects for calculators, computers, embedded systems, and the web. I was invited to teach beginner and advanced Java programming courses for my alma mater's continuing education program. As a graduate student, I have twice taught my advisor's undergraduate students about operating systems, C and x86 assembly programming, and reverse engineering. Their Computer Systems Organization class challenges them to launch buffer-overflow exploits, implement and optimize their own malloc design, and reverse-engineer raw x86, among other labs. I enjoy the challenge of teaching them these low-level concepts, and feel that the sense of accomplishment I feel when a student finally has a moment of understanding makes it worthwhile. I was therefore thrilled to be asked by Manning Publications to write a book about programming graphing calculators. "Programming the TI-83+/84+" is due in print this September, and is written to instill in readers young and old the same love of programming that I developed.

I believe that self-education and an early exposure to engineering and programming is vital to prompting a life-long love for these fields. Although I believe I had a predisposition towards technical fields and hobbies, the opportunities I was given fueled early interest that matured as I aged. In particular, without the ability to program my calculator constantly, whether at lunch, at home, or (perhaps unfortunately) during class and while walking to school, I doubt I'd have the love for and

intuition into programming that I now have. In chatting with many current and ex-calculator programmers, I have heard countless versions of my own story: the self-driven exploration of the calculator's features, the thirst to learn what made programs and games tick, the love of surmounting a good challenge. I think the burden lies on museums, libraries, and even technology companies to be good citizens and make such opportunities available to children and teenagers.

A year and a half ago, I wrote an editorial criticizing Texas Instruments, who had taken a nearly Apple-esque position in locking down their new TI-Nspire graphing calculator. Native programming in assembly, C, or even TI-BASIC was impossible with their new calculator, a restriction aimed at placating teachers upset about students playing games in class. Only via third-party hacks could the device be unlocked, and with each new operating system (OS) release, TI squashed the existing unlock exploits. In my piece, I decried TI's attitude with the Nspire as astonishingly short-sighted; it yielded a calculator that would not allow students to explore programming, a stark contrast to the TI-83+/84+ series. Texas Instruments vociferously promotes STEM (Science, Technology, Engineering, and Mathematics) education, but requiring "jail-breaking" to even write usable programs on the calculators showed exactly the opposite attitude. Thankfully, my editorial and other negative press forced them to partially reverse their decision, and they have now made the Lua language writeable on the calculator. Nevertheless, the Nspire continues to largely cater to a narrow view of the needs of teachers, rather than encompassing the equally-important needs of the students who buy and use Texas Instruments' calculators.

Perhaps you have a similar story of how you got into STEM fields, where you pushed yourself to learn from existing programs, from taking things apart, and from books. Even if your knowledge of technology and engineering comes entirely from formal instruction in classes, I believe you can appreciate the value of earlier exposure to the fields in every form, from hackable, programmable gadgets to easy-to-access fora with free expert programming help. I encourage educational and commercial institutions large and small to press forward in educating younger generations and to give them the opportunity to make the same self-driven discoveries that many of us once made ■