Proposal (Draft) // Platform Studies Series

My Computer Likes Me*
*When I Speak in BASIC.

Garrison LeMasters

"If you want to talk to computers, you got [sic] to learn a language. There are lots of languages for talking to computers. Most of them are O.K. [sic] for computer freaks but lousy for people. We will use the computer language called BASIC — great for people, not so great for computer freaks."

("BASIC; or, You Too Can Control A Computer!", The People's Computer Newsletter, October 1972).

Introduction

In the summer of 2009, a member of Google's software engineering team added an enthusiastic post to the search-engine giant's popular Open-Source Blog. Entitled "Programming made Simple!", the brief entry introduced the world to Simple, "a BASIC dialect for developing Android applications." Citing as inspiration the ease with which users had mastered Microsoft's BASIC in the 90's, Simple aimed to bring "an easy to learn and use language to the mobile world and the Android platform." As such, the language is "particularly well suited for non-professional programmers," the post's author explained, before adding, parenthetically, "(but not limited to)."

The response was unambiguous. "Whoever approved this at Google should be fired," begins the first comment. "Please tell me this is an April Fools' joke." Another: "This is the dumbest idea I have ever heard." Some offered alternatives: "Why BASIC? Why not something that is a programming language like Python?" And a handful grew nostalgic about their early experiences in Visual Basic. But it was clear from the tenor of those first comments not only that Simple was unwelcome among Google's open-source development community, but that the worldview it embodied was suspect, too. "I wonder who still believe [sic] in 'programming made simple,'" asked one incredulous commenter, mocking the title of the original post. "This is an oximoron" [sic].

In retrospect, the skepticism is hardly surprising. Among professional developers, there is a long tradition of animosity towards the BASIC programming language. Its seminal articulation is in the hyperbolic aphorisms of Dutch academic Edsger W. Dijkstra, who offered that "it is practically impossible to teach good programming to students that have had a prior exposure to BASIC: As potential programmers they are mentally mutilated beyond hope of regeneration" (1975). It is an institutionalized attitude that persists, unabashed, in computer science textbooks, on blogs, in forums, and at conferences. Wikipedia identifies over 280 variants of BASIC, many of which are still actively supported. But it is rare to find any variant of the language featured on university syllabi.

This total absence is something of a surprise, however: In the 1960's, BASIC was at the vanguard of computational infrastructure and curricula at North American universities and prep schools. In the 1970's, it was the language of choice among hackers and hobbyists. A decade later, BASIC was integral to nearly every microcomputer sold to the public. By the 1990's, it was one of the most successful products of one of the most successful corporations in history. It is available today in varieties that support most of the fashionable trends in computer science: Dynamic typing, reflexivity, scriptability, polymorphism, and OOP; it is available in open-source and commercial variants; with some effort, applications written in BASIC can even be deployed to Apple's tightly-controlled iOS platform.

And yet, the language that was to reclaim computation for the people — BASIC, the Beginner's All-Purpose Symbolic Instruction Code — remains an object of scorn and derision.

Through the analysis of this language as "platform," that is, as a more or less stable site of computation, abstraction, and expression, this book asks, "What happened to BASIC?"

The Plan of the Book

Chapter One establishes the main theme of the book: Essentially, that the story of BASIC is the story of the (North American) public's relationship with computation. Events surrounding the 1975 implementation of BASIC on the first microcomputer (MITS' Altair 8800) have had long-term technical, social, economic, and philosophical implications for computing. I look at two competing versions of BASIC, MicroSoft's Altair BASIC and TinyBASIC, touted by The People's Computer Company, in order to argue that these represented essentially competing visions of the future of computing. I discuss BASIC's involvement in the first recorded instance of data piracy, and Bill Gate's "Open Letter to Hobbyists."

Chapter Two closely examines the genesis of CardBASIC ("Dartmouth BASIC"), with special attention paid to the way that university culture and physical infrastructure shaped the archetypal iteration of the language. Drawing distinctions between norms in FORTRAN, ALGOL, and BASIC, I demonstrate ways in which CardBASIC represented an ideal environment for youthful experimentation and play. Finally, echoing the events surrounding the pirating of AltairBASIC, and foreshadowing the coming problems with macros and VBS, I look closely at how networks were instrumental in the success of BASIC, both on-campus and off.

Chapter Three is essentially an exercise in computational comparativism. 1977 was an annus micro mirabilis, as the nearly-simultaneous introduction to market of three microcomputers promised to alter radically the average North American's relation to computation. All three machines featured a BASIC prompt as the de facto interface. This chapter examines how subtle differences in implementations of that language (Tandy's use of TinyBASIC, and the BASIC dialects designed by Bill Gates for Commodore and Apple) led to substantively different outcomes.

Chapter Four looks to the 1980's to consider BASIC in print: The popular phenomenon of the paper-based diffusion of BASIC code. While professional and academic journals occasionally featured short excerpts of programs in COBOL or ALGOL, their readers had ready access to mass storage and digital networks for the exchange of code. But in the early 1980's, there was a dearth of readily-available software for home users, and the idea of spending income on computer applications was still novel. Popular magazines like Compute! and books like the Creative Computing series featured lengthy program listings for readers to enter, by hand, on their new home computers. I argue that this fostered an intense intimacy with the BASIC language, and that readers' ad-hoc translation between variants of BASIC encouraged a rare kind of fluency.

Chapter Five examines the more sinister implications of BASIC's ubiquity and accessibility. As the language became integral to Microsoft's Office Suite, macro viruses and harmful scripts became a reality. BASIC had depended on the Dartmouth campus network for its early success, and later on a loose network among New England high schools and prep schools for its diffusion. Now, with the advent of the Internet and users' increasing dependence upon electronic mail for communication, the reach of BASIC was cause for alarm.

About BASIC as a Platform

The study of BASIC as a platform is distinct from previous scholarship on the subject. Among historians and journalists, BASIC is generally regarded as a piece of contested intellectual property, a prop for colorful personalities in conflict. Educators tend to see it in functional terms, as a first-step towards computational literacy. And programmers (even its adherents) often characterize BASIC as a "quick and dirty" or "inelegant" way of achieving an end through computation. In surveying the history of the language as a computing platform — that is, as a technological basis for social and cultural expression — BASIC is all of these things. But it is also something more.

In its conception of BASIC as a platform, as "a perspective on parts of a computing system" (Bogost and Montfort), this book identifies five interrelated and increasingly particular domains of expression: Discourse, Language, Code, Program, and Execution.

* The human discourse that surrounds, supports, and documents BASIC;

* The ideal constraints and affordances of the BASIC language in the abstract;

* The material constraints and affordances of any single instantiation of BASIC;

* The instantiation of that code as a program, typically written for use in a specific setting;

* The work that is accomplished through that program's execution.