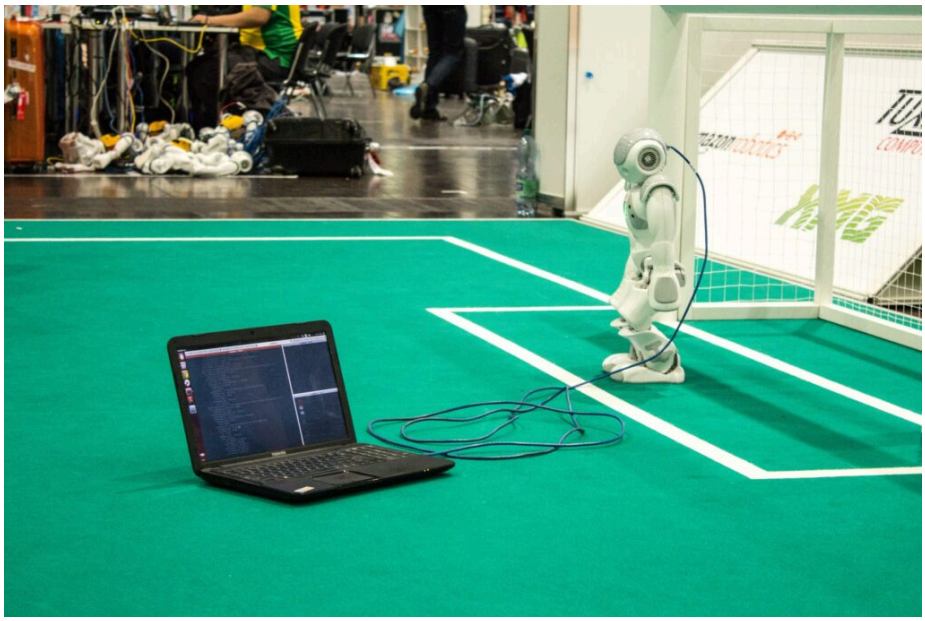
Issue 028: Programming As A Hobby

Graham Lee

January 4th, 2021



Welcome to the twenty-eigth issue of *De Programmatica Ipsum*, dedicated to the subject of *Programming as a Hobby*. In this edition:

- Adrian explains how hobbies prevent specialization[1] from taking over our minds.
- Graham argues that the current approach to bring new generations to programming is flawed[2].
- In the Library section[3], Adrian reviews "Dealers of Lightning[4]" by Michael Hiltzik.

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

Cover photo by C M[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/specialization-is-for-insects/
[2] https://deprogrammaticaipsum.com/zx2020/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/michael-hiltzik/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@ubahnverleih?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/programming-hobby-diy?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Specialization Is For Insects

Adrian Kosmaczewski

January 4th, 2021

There is a whole field called "Recreational Mathematics[1];" a moniker that might as well be replaced with "Mathematics as a Hobby." I personally enjoy dabbling in it a lot; I am mostly interested in number theory feats: Ramanujan[2]'s hypergeometric series and continuous fractions converging to Pi in the most weird ways are among my favorites.

I regularly code as a hobby, as I have for the past 20 years. My hard drives are filled with unfinished experiments, code from books, unfinished ideas, in lots of different programming languages. Many of those experiments have to do with numerical approximations of those series or fractions. For example, the last such experiment was playing with Binet's formula[3] to calculate Fibonacci numbers.

Coding for fun is relaxing, if anything, precisely because money is not involved. I can write and rewrite and refactor the code as much as I want. I can release new versions, even if they break previous ones. I can change the architecture without having to call for design meetings. Heck, I do not even need to run daily standup meetings with myself, or to be interrupted by Slack notifications every minute.

I also write as a hobby. My own blog[4] and this very magazine were born out of it.

The FAQ[5] of the Cello[6] C library explicitly puts hobbies at the antipodes of production systems:

[1] https://en.wikipedia.org/wiki/Recreational_mathematics
[2] https://en.wikipedia.org/wiki/Srinivasa_Ramanujan
[3] https://en.wikipedia.org/wiki/Jacques_Philippe_Marie_Binet
[4] https://akos.ma/
[5] https://github.com/orangeduck/Cello#faq
[6] http://libcello.org/

> Can it be used in Production? It might be better to try Cello out on a hobby project first.

Hobbies are based on pleasure, and dare I say, a certain cult of hedonism. Hitting the keys of my keyboard during weekends and holidays has exactly to do with that. I can escape the daily ailments of reality and learn, learn, and learn again.

Sometimes I re-learn things I had forgotten years ago. Just like great books are not meant to be read just once[7], some things are not meant to be learnt just once.

I do not personally find more or less beauty with functional programming languages than with, say, object-oriented or procedural ones. Sometimes I enjoy writing in a language, sometimes in another. There is a certain Tao[8]-like feeling when moving my brain from language to language.

Specialization, so valued and established in the world of business, can be left aside. Or as Robert Heinlein said, it can be left for insects[9]. In a hobby setting, you can pursue your own interests, regardless of business needs, requirements, specifications, or screaming stakeholders. It is, arguably, the *only* situation where art can happen[10] through programming.

During the pandemic of 2020 I worked mostly on just one of those experiments. My own recreation, in 20+ programming languages, of Conway's Game of Life[11]. Faithful readers of this magazine might remember that I have already talked about that project in these pages; that was in my article[12] tracing my first steps writing a Smalltalk application.

As fun and satisfying as my little project is to me, I do not think it might make me a better programmer. At best, it gives me the option to try things I read elsewhere; I am not in the business of discovering anything new. I am not advancing science, by any stretch of the mind. I just learn, try things, fail once and again, and sometimes write about it.

Hobbies are not meant to be maintained[13]. They just exist per se, they are valid through and by themselves, without the need for external validations or justifications. They are a whim. They exist with the sole purpose of pleasure and enlightenment.

Sadly, the question, "Do you code in your free time" comes often in programming interviews. The answer is used as a gauge with which to measure the degree of "passion" a certain victim candidate might possess.

By all means, it is the humble wish of this author for this practice to end, on both sides of the equation. Recruiters must stop asking this question, just as we have to teach future candidates to stop answering it. If the practice continues on the side of the recruiters, which it will, we must then teach young members of our industry to ask for an automatic raise in case of an affirmative answer.

The fact that a person codes or not in their free time is absolutely irrelevant to their potential in a job. This is a fact. There is no correlation between those two things. Brilliant consultants from nine to five might just as well enjoy paragliding on weekends; otherwise unengaged and bored code monkeys might spend evenings creating brilliant games on their home PC.

[7] https://www.kqed.org/arts/10145618/thank-you-for-re-reading-reading-books-about-reading-books

[8] https://www.mit.edu/~xela/tao.html

[9] https://en.wikipedia.org/wiki/Competent_man

[10] https://deprogrammaticaipsum.com/a-brief-history-of-programming-artists/

[11] https://gitlab.com/akosma/Conway

[12] https://deprogrammaticaipsum.com/the-absolute-no-frills-quite-ignorant-very-incomplete-and-certainly-flawed-beginners-guide-to-smalltalk/

[13] https://unmaintained.tech/

Coding for fun, or coding for money, are two absolutely, completely, and irrevocably orthogonal activities. This offset applies not only to coding, by the way; performing any human activity for the mere objective of getting paid, substantially modifies the dynamics of the activity in question, and more importantly, of the person doing it, in ways that neither psychology nor economics fully understand.

I am not calling for, nor citing studies in either discipline at this moment. After all, at this point in time this magazine is a hobby, and it is solely in your hands[14] to help us make a business out of it.

Cover photo by Pedro Santos[15] on Unsplash[16].

[14] https://deprogrammaticaipsum.com/contribute/

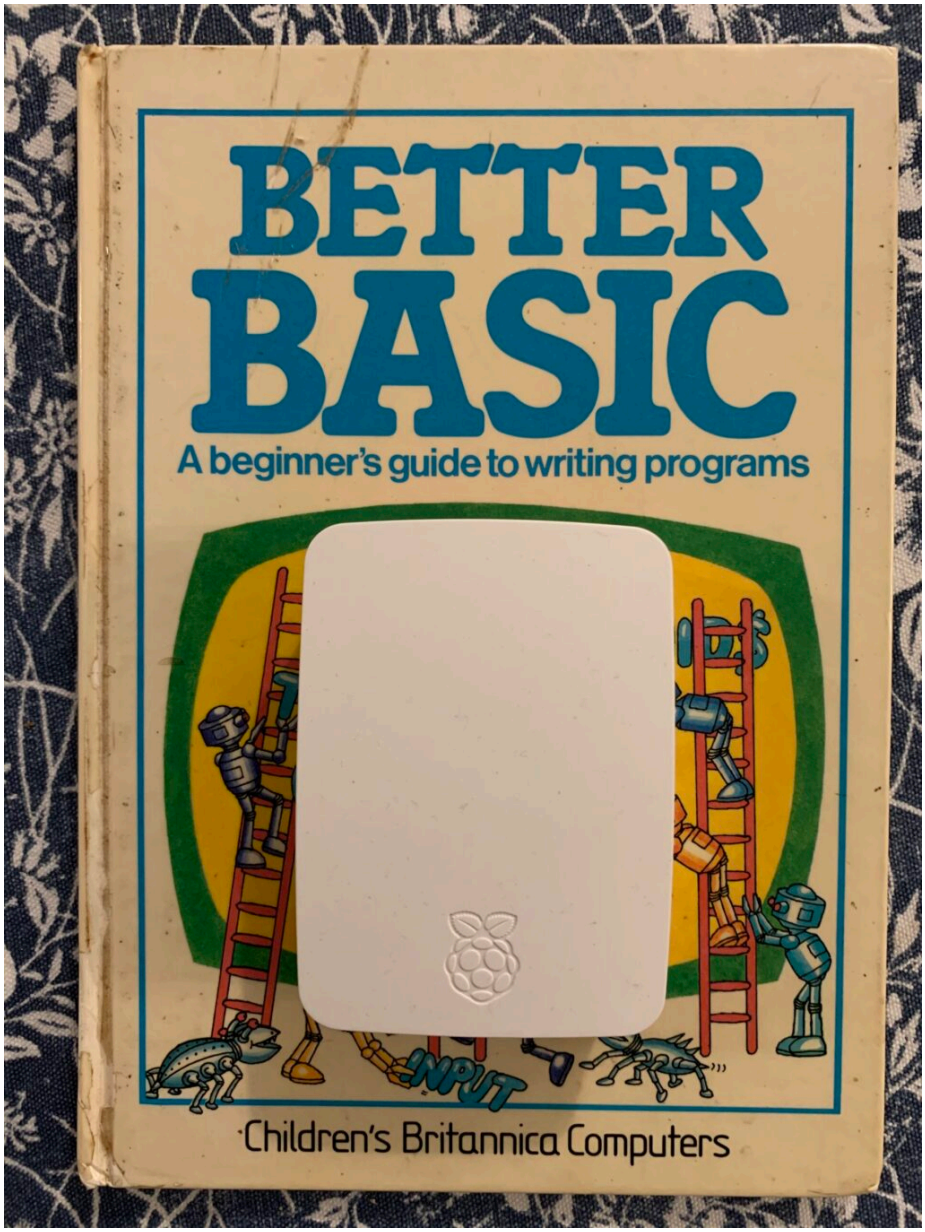
[15] https://unsplash.com/@pedro_ag_santos?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[16] https://unsplash.com/s/photos/monopoly-game?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

ZX2020

Graham Lee

January 4th, 2021



There is a common trope that says we would get more children interested in programming as a hobby if programming as a hobby was like the programming *our generation* did as a hobby. By our generation, I mean a broad swathe of people in WEIRD (Westernised, Educated, Industrialised, Rich, and Democratic) societies who are mid-late Gen X and early-mid Gen Y, who can claim to have "grown up" with microprocessors either as kit boards or in the form

of slab-form-factor microcomputers, ending approximately with the dominance of the PC and Windows 95.

What did that hobby look like? For some, it is the shape of the computer: a slab with integrated keyboard, and sockets for a video signal, external media, and additional controllers (joysticks; light pens; later mice) and outputs (printers primarily). These are the people who welcome the Raspberry Pi 400[1] as the product that will rejuvenate the "youth fascinated by computers" craze that brought us Codemasters[2], Microsoft, Apple, and so many other famous names in the industry.

The Raspberry Pi 400 is without doubt an amazing device. It, along with similar product from the likes of Pine64[3], costs around £100, which is the same list price as 1980's ZX80[4] in its pre-assembled form factor but is 1/4 the real-terms price. Like the ZX80, it does not need specialist equipment to get working: if you already have an HDTV, you are good to go (that is 96% of us in the UK but you will go further if you have already got home broadband internet too; 39% of us in the UK).

For some, it is what happens when you turn on the computer. Anyone who used one of the 8-bit microcomputers probably got faced with with a BASIC prompt saying something like READY or OK and had to write a program to get anything done. Any angry Jupiter Ace[5] users who want to correct me that actually they used the superior Forth[6] language are welcome to tweet me at @akosma[7] ;-).

Regardless of the language, whatever you wanted to do, whether it was load a game from tape, or do your finances, or write the next killer app, you started in immediate mode by writing a computer programme.

Gradually, that power was marginalised; the first Amigas had BASIC on a disk that came with the OS, later ones had ARexx[8]. Acorn computers running RISC OS[9] could still access BBC BASIC[10]. Even MS-DOS still had QBASIC[11]. They all still had a programming environment available to all users. Nowadays, all desktops have one that is two clicks away (open your browser, then open the console), and no tablets or smartphones do.

Back to the Raspberry Pi, then: some want to boot directly into a Python prompt, or a Scratch[12] or Pencil Code[13] workspace, to recapture that part of the magic. Maybe the problem with kids learning to code is that we are not *exposing* them to the code, so let us set them the same challenge we were set: in order to not program your computer, you need to write a computer program.

Such a computer would be objectively worse at any non-programming task than any other computer. While human-computer interaction is far from a solved problem (and as I argued

[1] https://www.raspberrypi.org/products/raspberry-pi-400/?resellerType=home
[2] https://en.wikipedia.org/wiki/Codemasters
[3] https://pine64.com/
[4] https://en.wikipedia.org/wiki/ZX80
[5] https://en.wikipedia.org/wiki/Jupiter_Ace
[6] https://en.wikipedia.org/wiki/Forth_(programming_language)
[7] https://twitter.com/akosma
[8] https://en.wikipedia.org/wiki/ARexx
[9] https://en.wikipedia.org/wiki/RISC_OS
[10] https://en.wikipedia.org/wiki/BBC_BASIC
[11] https://en.wikipedia.org/wiki/QBasic
[12] https://en.wikipedia.org/wiki/Scratch_(programming_language)
[13] https://en.wikipedia.org/wiki/Pencil_Code

in an earlier issue, the state of the art on mobile is getting worse[14]), no other computer demands you satisfy the computer in its own terms before it will work for you. Want to play Donkey Kong[15] on your phone? Tap the Donkey Kong button. Want to do it on this mythical BASIC Pi? Crack open the manual and learn the commands for loading Donkey Kong.

Oh, it did not come with a manual? Too bad. Some people point to a dearth of the sorts of programming books and magazines that were available in the micro revolution. With modern versions of Input Magazine[16] or Peter Usborne's books[17], children could explore their new computers with friendly and informative guides. Those books do in fact exist, and Usborne is still one of the publishers behind them. The economics are different these days, and one might expect a 52-issue partwork magazine series on programming to come with a free computer in the first issue.

As you can tell, I am downbeat about many of the attempts to rekindle the excitement about computers everybody with a computer in the 1980s felt. The difference between those times and these is not that humanity has become jaded and lost the capacity for excitement, though 2020 certainly tried hard in that respect. There are more computer programmers now than four decades ago, more materials (including freely-available materials) on programming, more options for programming environments and tools, and of course cheaper computers.

What has happened is not that we have become disengaged with computers. What has happened is that most of us with a computer *do not care about it as a computer*. Owning a Commodore 64 is no longer a badge of membership in the hobbyist computer club. That does not mean that the hobbyists are not out there, just that you have got to look harder to find them.

And that means that we do not need to go back and rediscover the excitement of the 1980s. We need to go out and nurture the excitement that exists now, in the 2020s. We need to do it on 2021's terms, with 2021's technology and with the expectations and access denizens of 2021 have at their grasp.

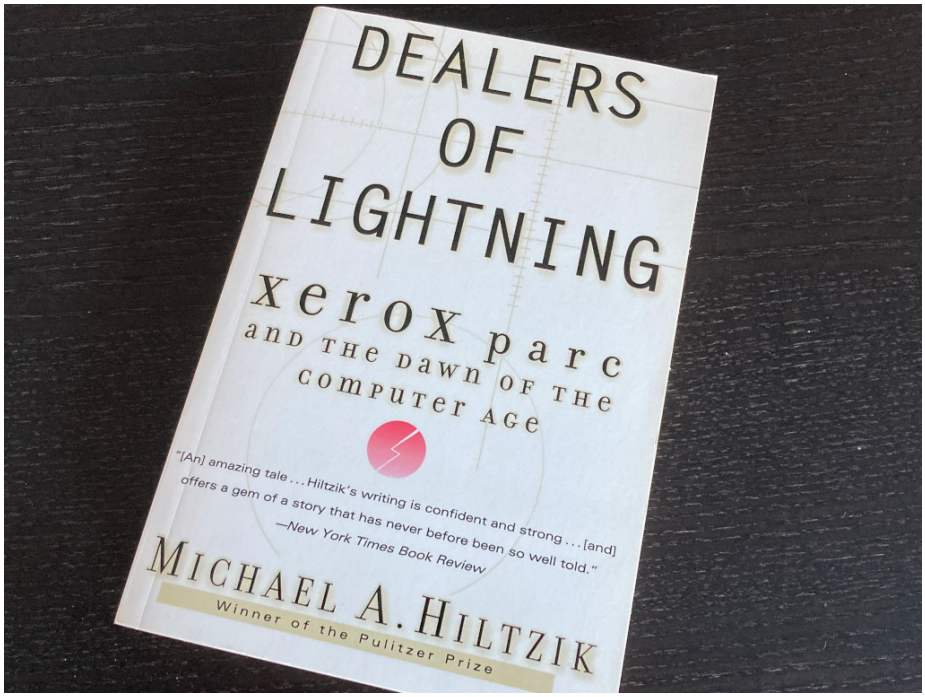
Cover photo by the author.

[14] https://deprogrammaticaipsum.com/the-rise-and-fall-of-mobile/
[15] https://en.wikipedia.org/wiki/Donkey_Kong
[16] https://archive.org/details/inputmagazine
[17] https://usborne.com/books/computer-and-coding-books

Michael Hiltzik

Adrian Kosmaczewski

January 4th, 2021



Imagine that you are a fourth grader in California, in 1973. You were 6 or 7 when Neil Armstrong and Buzz Aldrin walked on the moon, and there were still astronauts up there just last year. On the radio you can hear Pink Floyd, Elton John or Led Zeppelin. One day your teacher receives an invitation for an experiment involving school kids in a laboratory somewhere in Palo Alto, a location 40 minutes south of San Francisco. Even stranger, the invitation comes from a well-known firm in the photocopier business.

Imagine that you follow your teacher, and a young man greets you at the entrance of the building. A few minutes later you are standing in front of a rather strange contraption sitting on top of a desk. It consists of something looking like a television set standing on its side, a typewriter without paper, and a small corded box with colored buttons on top. Next to the desk, there is something that looks like a huge photocopier. Cables all over the floor connect these pieces of equipment together. There are a few more similar machines in the room, and more cables in between.

The young man turns the machine on, and a series of rectangles, pictures and strange words appear on the screen. He explains that this is a computer, and that what you see on the screen is a language that tells it what to do.

The man was Alan Kay[1]. The computer was the Alto[2]. The photocopier was actually one

[1] https://en.wikipedia.org/wiki/Alan_Kay
[2] https://en.wikipedia.org/wiki/Xerox_Alto

of the firsts laser printers[3]. The cables were Ethernet[4] ones. The language on the screen was Smalltalk[5]. The laboratory was the Palo Alto Research Center[6]. The company was Xerox[7]. The year was 1973.

Without knowing it, these kids were inside of a time machine, witnessing what their own future was going to look like, merely 20 years later.

This is the world described by Michael Hiltzik in his book, "Dealers of Lightning." The first modern office, exactly as we have known it for the past 20 years, with networked workstations enabling people to write and print documents in a WYSIWYG editor, create applications using Smalltalk, send and receive e-mail and much more.

As incredible as it may sound, Xerox had the future in its hands, and Hiltzik explains how it let it slip through its fingers. The author conducted interviews with the original characters, but this book is actually a novel. There is suspense and plot twists, and it is written to keep you on the edge of your seat.

> "Millions of people will write non-trivial programs, and hundreds of thousands will try to sell them.(...) Almost everyone who uses a pencil will use a computer, and although most people will not do any serious programming, almost everyone will be a potential customer for serious programs of some kind... Such a mass market will require mass distribution."
>
> Butler Lampson, in 1972, quoted in chapter 15, "On The Lunatic Fringe."

Michael Hiltzik approached this story as a Pulitzer prize article, using his experience as a journalist for newspapers and magazines in the United States. The book is divided in chapters for each of the "inventors" of the key technologies created in PARC from 1971 to 1984, and how each of those inventors ended up creating their own enterprises (like Adobe) or working for other companies willing to build the next wave of computers (like Apple or Microsoft.)

The most important part of the plot, in my opinion, remains the inner tension in Xerox, between the East and the West coast; between Rochester and Palo Alto. Two very different views of the future of technology ultimately decided the fate of PARC. The management of Xerox, three timezones away, simply could not resign itself to dismiss the photocopier, and this observation led to the dismissal of the Alto as a potential product until too late.

Xerox decided to not enter the personal computer market in 1977, 4 years before the introduction of the IBM PC, 6 years before the introduction of the Apple Lisa, 7 years before the Mac. Xerox decided its fate and changed the course of technology forever. In a way, Xerox was the first company to be disrupted by software, or from within. We are happy to say that every company is a software company[8] these days, but few understand the implications of this assertion. In an era where software companies challenge classic industries like automobile or retail, there are more than a few bits of wisdom to learn from the fate of Xerox.

This book is intended to all audiences, technical or not. The descriptions of the breakthroughs and technical challenges faced by the scientists in the laboratories are not explained

[3] https://history-computer.com/ModernComputer/Basis/laser_printer.html
[4] https://en.wikipedia.org/wiki/Ethernet
[5] https://deprogrammaticaipsum.com/issue/issue-025-smalltalk/
[6] https://en.wikipedia.org/wiki/PARC_(company)
[7] https://en.wikipedia.org/wiki/Xerox
[8] https://www.forbes.com/sites/techonomy/2011/11/30/now-every-company-is-a-software-company/?sh=32f8c793f3b1

in drastic detail; there are, however, interesting analogies with the world of the year 2000, which provide the necessary context for the reader.

Cover photo by the author.