Crafting Joy in the Digital Age

Sandy Urazayev, Microsoft

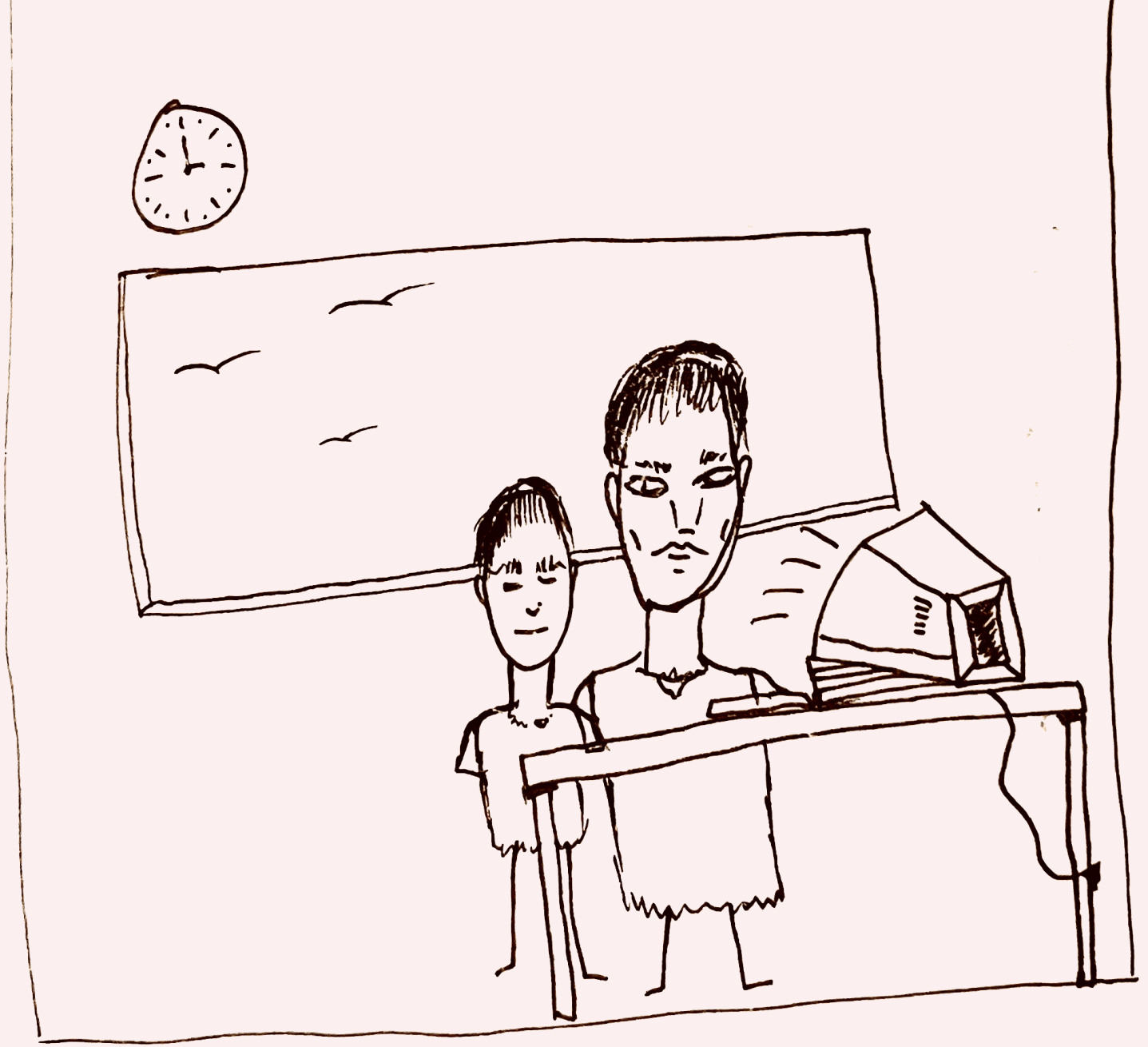
Tuesday, June 10th, 12025 H.E.

Encountering a computer

When I was a fifth of my current age, the computer hadn’t meant anything but a cool toy in my father’s office that I could play games on.

The games themselves and memories of them have given up in the face of passing time, but the feelings of interacting with this strange machine remain strong and wholly intact.

Computers had already taken over the world by then; seldom anyone had remotely known how insatiable its appetite would grow to become.

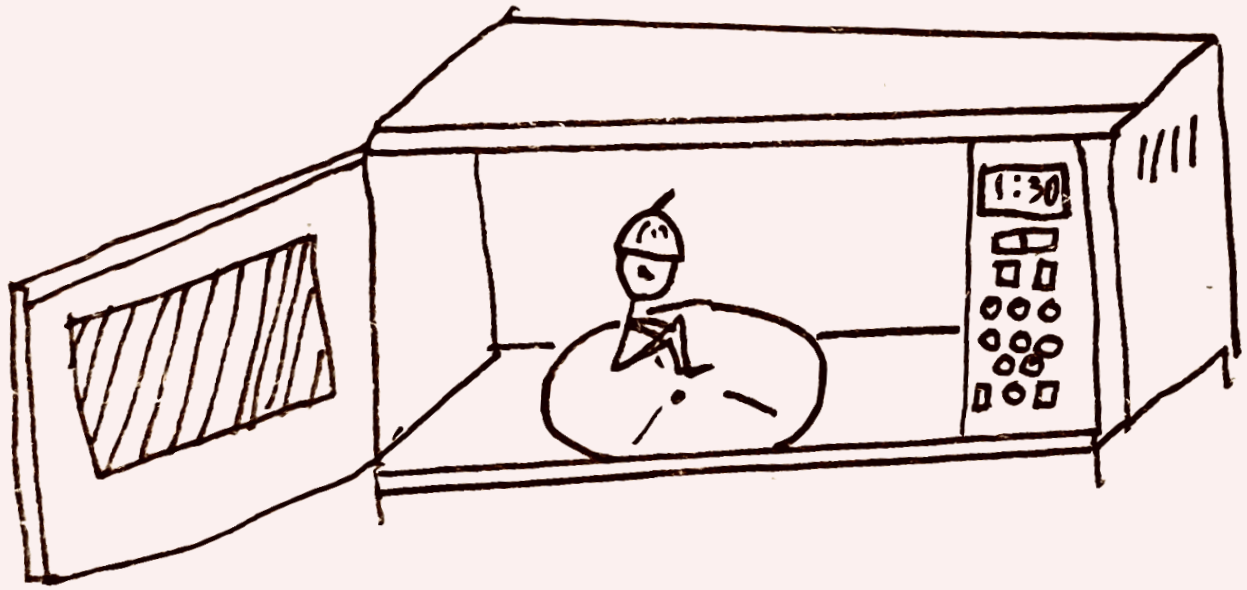


Technology as the driving force

In twenty years since I first discovered the computer, this object had found its way into every facet of my life—it would be fair to say, there is no way to lead a modern lifestyle without surrendering yourself to it.

In literature, we challenge the status quo; we challenge the established; we challenge what we know and believe in. With technology, there is no other. Why is it as it is? Why did we give ourselves away to it?

If a civilized person must keep himself informed on politics and his social duties, shouldn’t the same be applied to the technology of his time?


1:30

The flattened world

“It feels as if the whole world has been transformed into images of the world and has thus been drown into the human realm, which now encompasses everything.

There is no place, no thing, no person or phenomenon that I cannot obtain as image or information.

One might think this adds substance to the world, since one knows more about it, not less, but the opposite is true: it empties the world; it becomes thinner.”

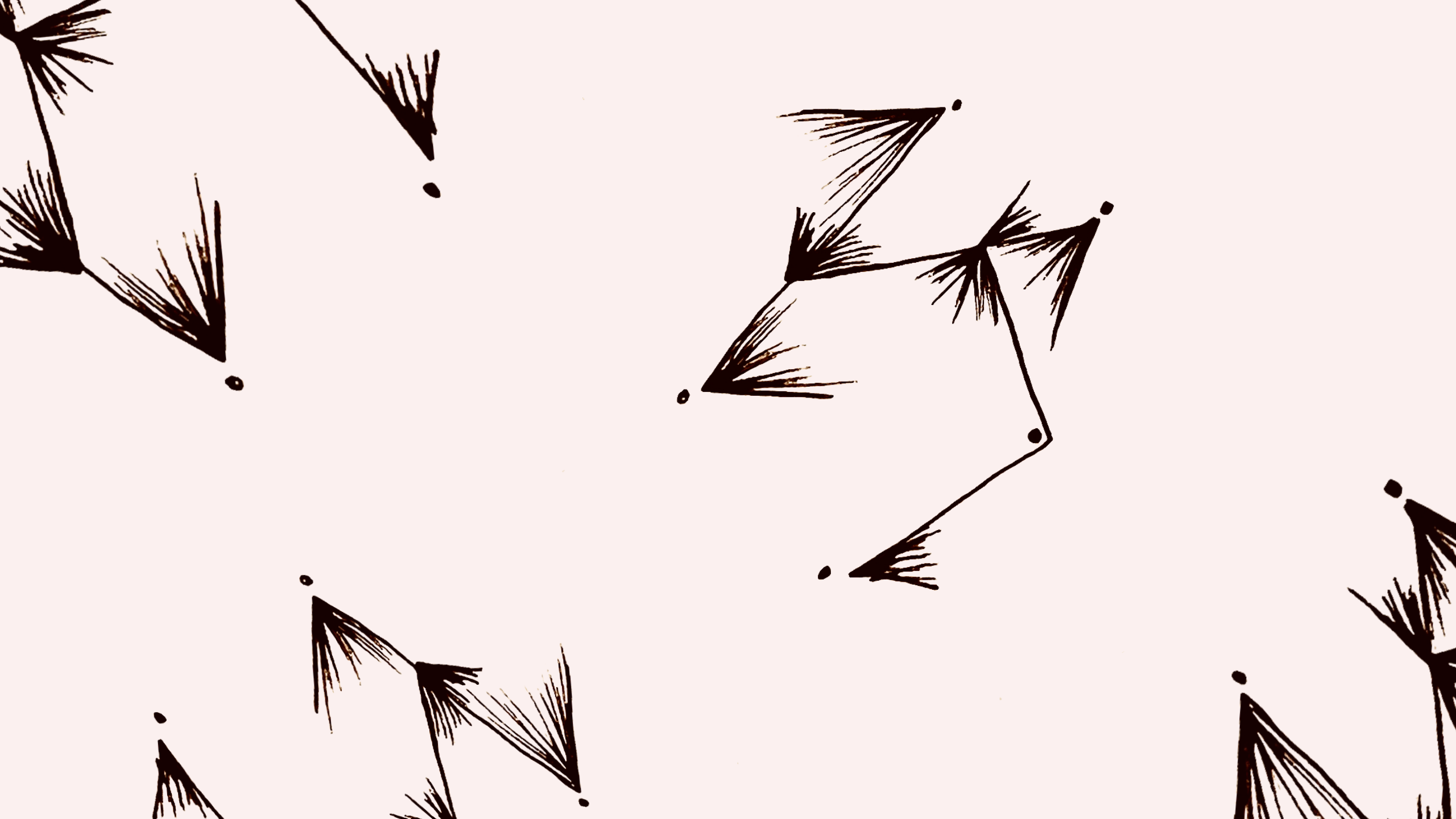
– *Karl Ove Knausgaard*

The loss of the world through abstraction

Seeing images of far away lands, even the ones I have been to, they have a habit of stubbornly remaining just that—images that I cannot touch; images that I cannot influence; images that seek no participation.

The veil of abstraction delivered by technology tells me these things that I know are real; I know they are happening—it is not that I refuse to believe it—it is not here; not with me; somewhere there.

The prospect of being actively alienated from the world by getting exposed to endless amounts of it terrifies me. The numbness of the mind.



The loss of autonomy through choice

The surplus in number of tools technology offers us shapes the methods we use to interact with the world in the ways that we have no control over.

I never asked for WhatsApp, Gmail, YouTube, Snapchat, Instagram, and others (too many to count) to mold my view of the world and how I interact with the ones dear to me.

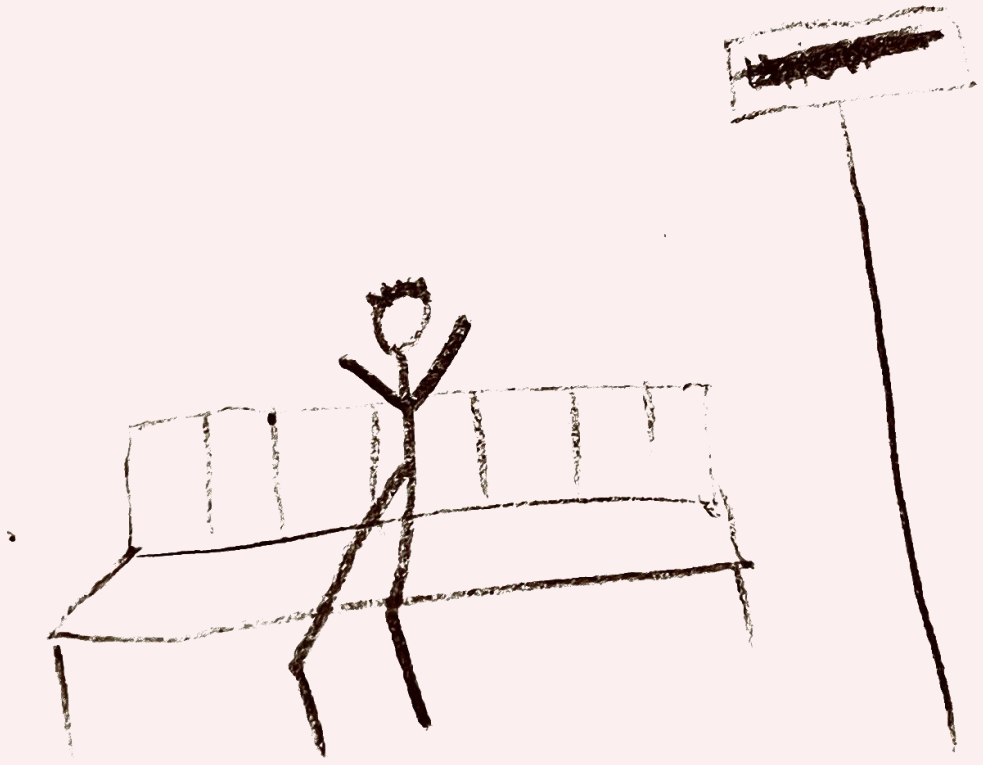
The unknown forces that present themselves as ready packages, where we become both the test subject and the product in how they succeed.

Taking the passenger’s seat through life

Being averse to adapting to changing times is a sign of stubbornness that would leave one sitting at a station when the last train had already departed.

One should also be aware of taking the first train to nowhere from the destination just reached—nowhere. Supporting Plato’s axiom, one should lead a conscious life.

What is the way to do is in the world that wants to take us for a ride?

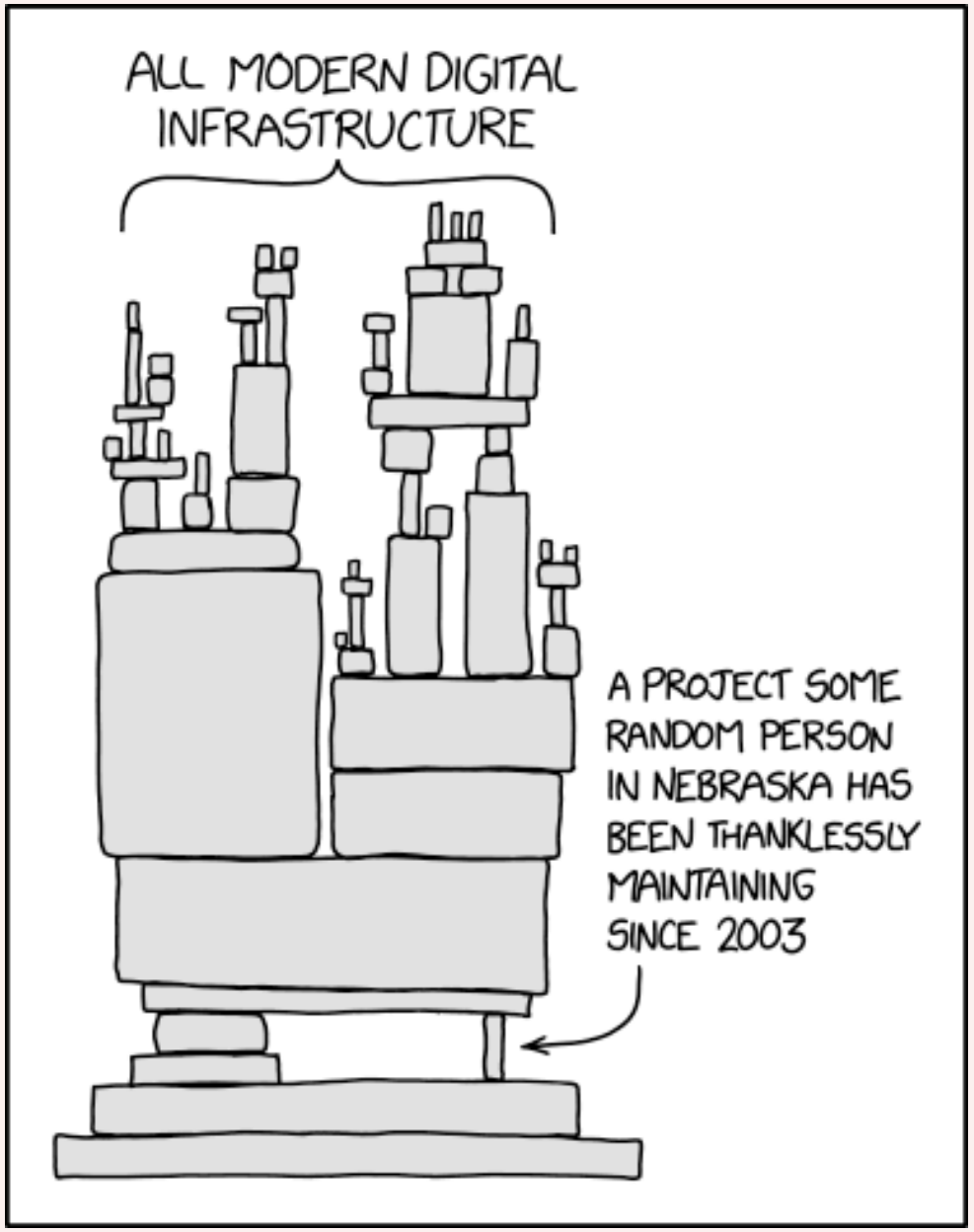


Building software is a double-edge sword

Living in the world of software engineering, we get exposed to the underbelly of the industry, seeing how the sausage is made, building the technology that will affect billions, ourselves included.

Even then, the sense of alienation is ever-present. We build systems, eradicating chance and unpredictability by mechanizing and formalizing the world into abstract terms.

Where the abstract terms then get translated into computer languages, once again, taking its effect on the world.



Dependency, https://xkcd.com/2347/

On programming

Treating programming primarily as a way to appease the computer to do our bidding might seem the correct and practical application of it; we were taught to do so and think so.

However, in this way, we end up speaking the language of the computer; the language that is fundamentally foreign to us and how we communicate with each other.

This leaves us with little leeway for programmers to share a human connection. As the machine with its cold and predictable logic becomes our conduit.

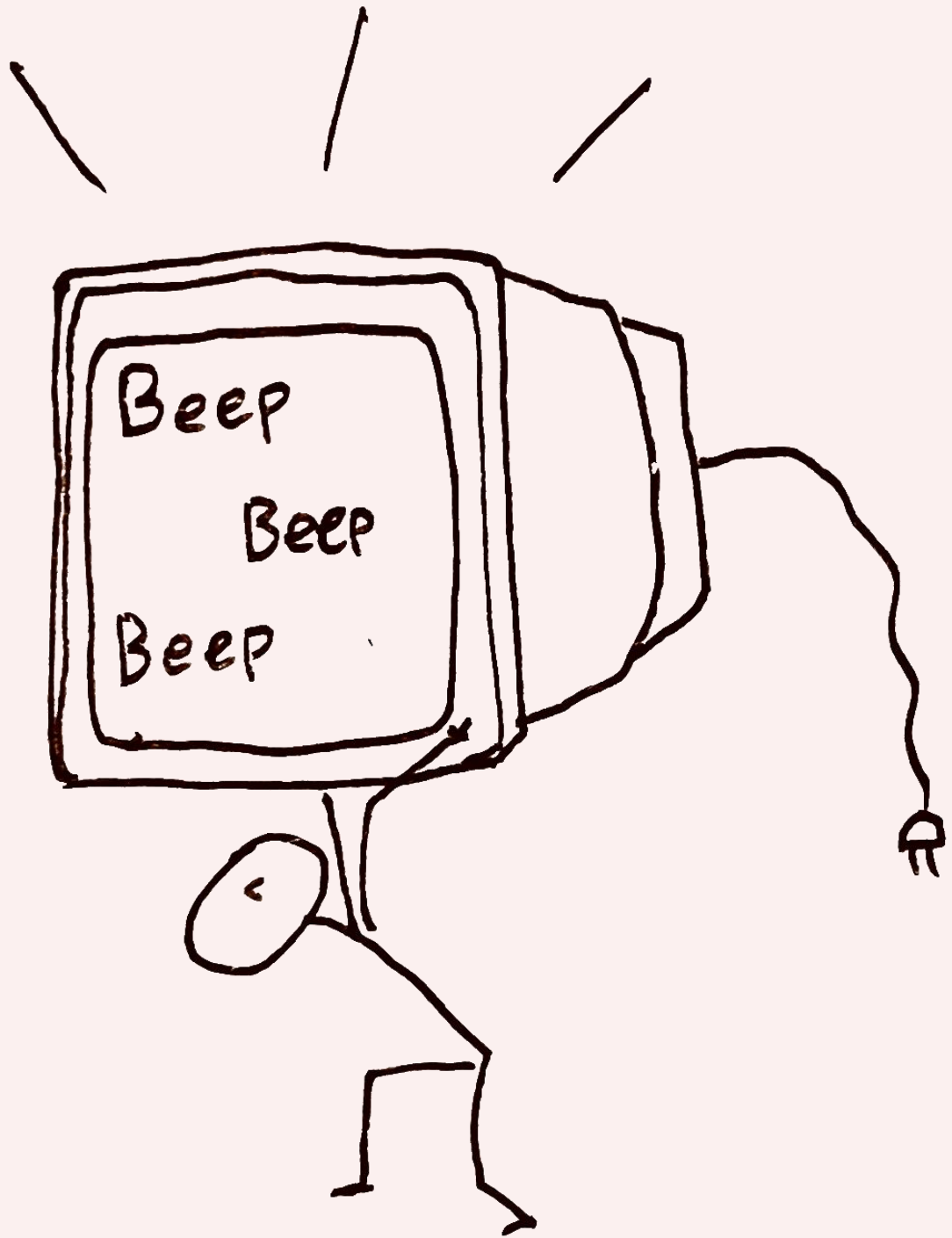

Beep
Beep

Anecdotes of large systems

Discovering any large system that has been written, re-written, refactored, updated, re-arranged, and bloated throughout many years always has a similar feeling to that of fearing the unknown.

Previous developers might have been nice enough to leave bits of documentation here and comments there; we have no choice but to hope that it is still correct, possibly outdated, and misleading at its worst.

The feeling of dread that permeates these large systems is caused by the desire to satisfy the machine and not the fellow programmer.


Beep
Beep
Beep

Literate Programming

If composers can learn the rhythm and craft of the maestros by listening to their works; if writers can learn the grammar and composition of the poets by reading their literature; if artists can learn the brush strokes and colors of the greatest painters by studying their canvases—

Then why can’t programmers learn the beauty of programs and their structure of great software by simply using it?

Literate Programming is an underexplored paradigm that could finally save us from what we have done to ourselves and what we call computing.

When I find a mention of Literate Programming

It is about creating works of literature

“Literature of the program genre is performable by machines, but that is not its main purpose. The computer programs that are truly beautiful, useful, and profitable must be readable by people.

So we ought to address them to people, not to machines. All of the major problems associated with computer programming—issues of reliability, portability, learnability, maintainability, and efficiency—

—are ameliorated when programs and their dialogs with users become more literate.”

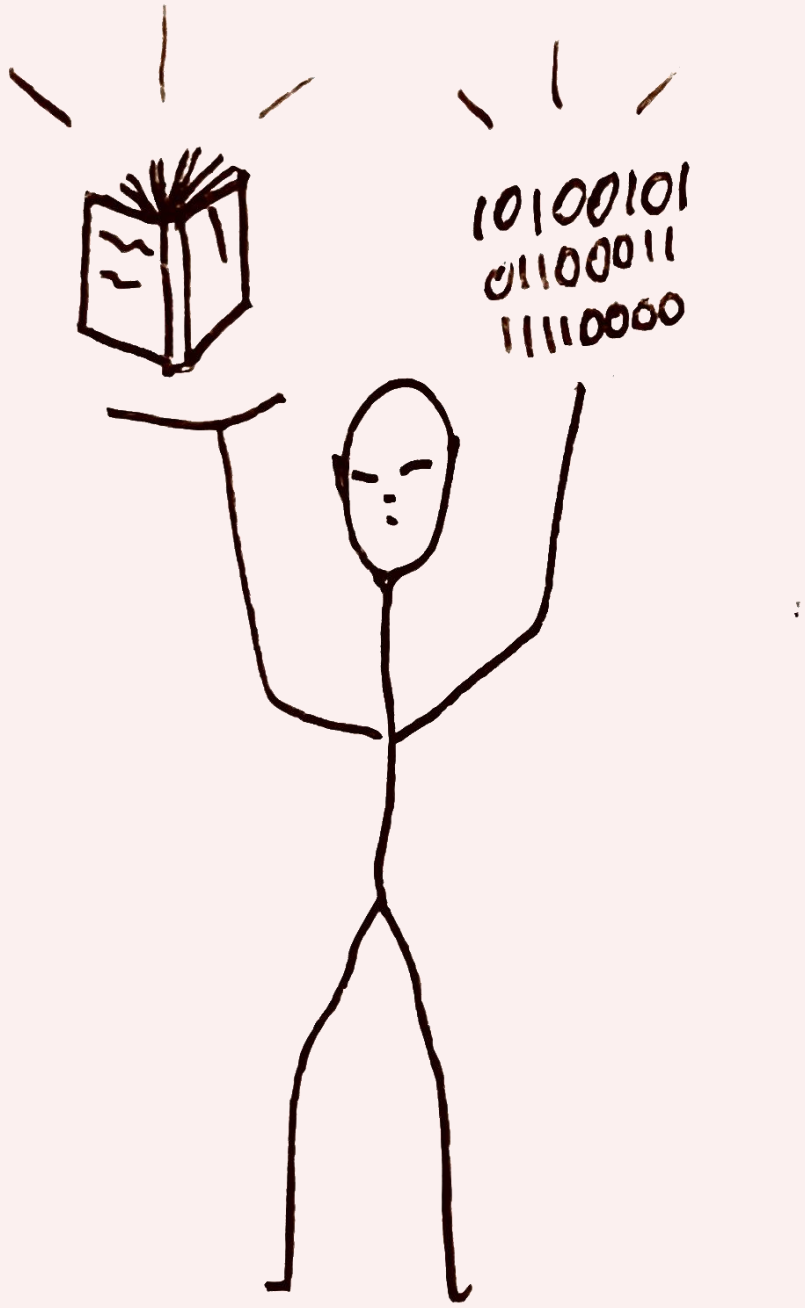
— Donald Knuth, *Literate Programming*

Infinitely more to explore

I could explore all the implications that Donald Knuth explores in what is science, what is art, is Computer Science a form of art or a discipline of science or a branch of mathematics or an application of engineering?

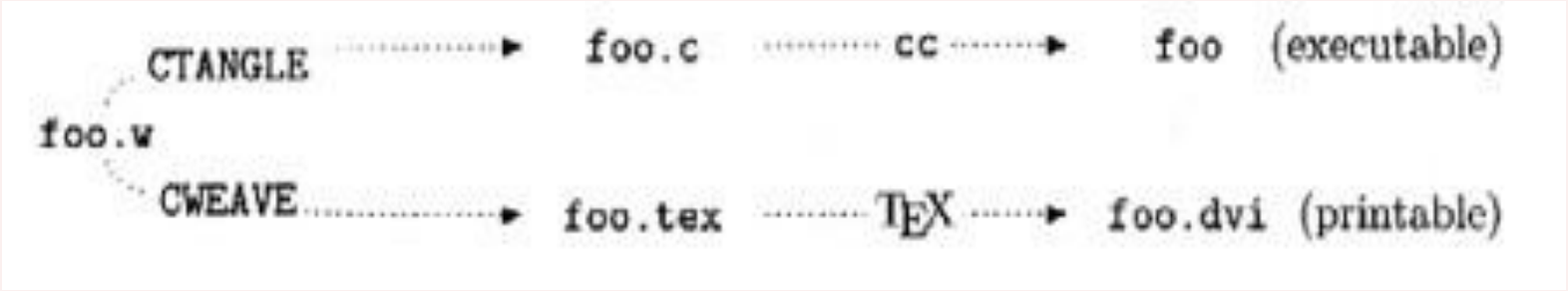
All of those questions are fascinating to explore, which I have done in my previous talks throughout my college career and invited speaker venues.

What I wish to focus on is the art of communicating oneself expressly and succinctly in both the language we are taught as kids and in code as a pleasant consequence.


10100101
01100011
11110000

WEB, the language of mechanized literature

Expressing yourself not merely just in code with optional text, but text-first and interweaving code as means of codifying the algorithm described by the text is the core idea behind WEB and one of its dialects, CWEB.

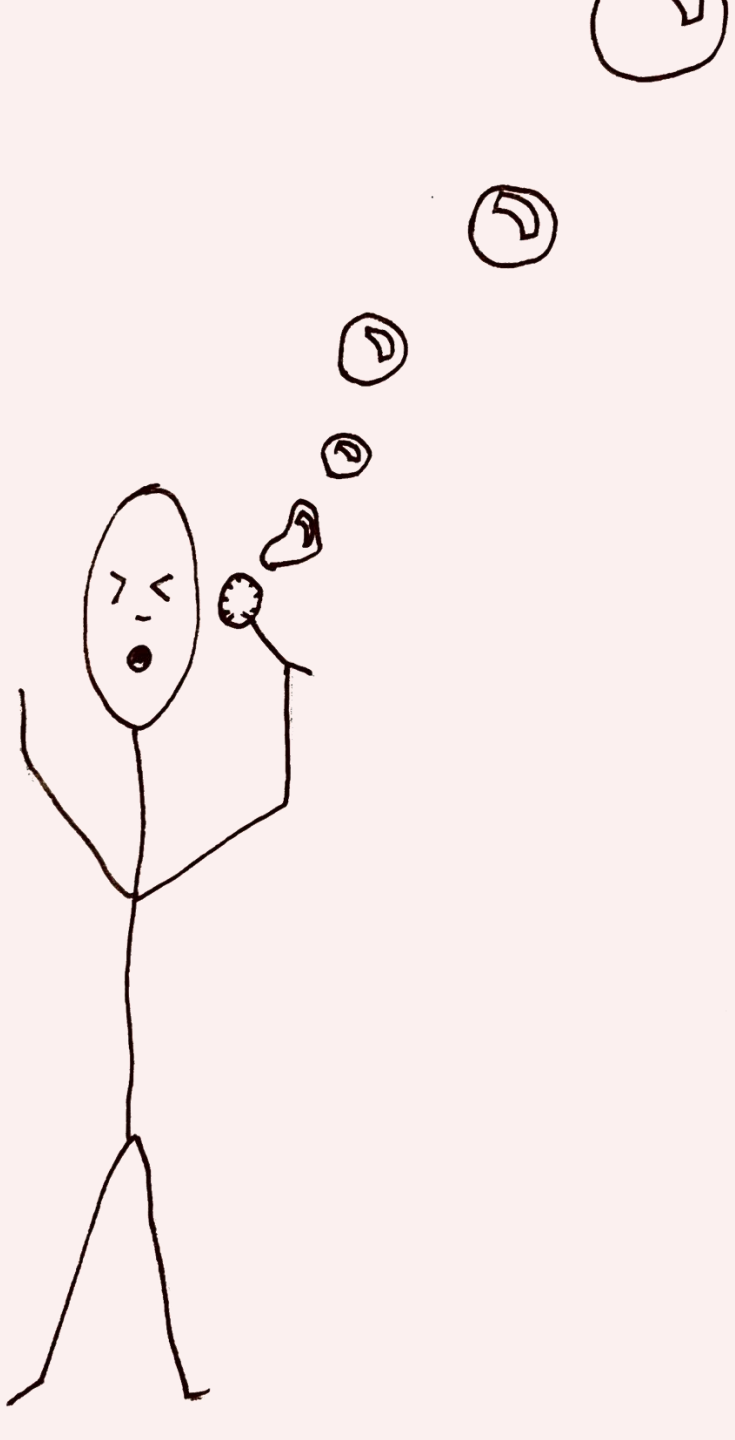


The result of pushing this work of literature through specialized tools is both a completed book for others to read, and marvel at; and a finished piece of executable code to transform the abstract into machine's signals.

Literate sorting

My interest in Literate Programming roots deep from the days I first got serious in Computing. The mentor who I owe my passion and love of programming to had taught me that it is imperative to write beautiful programs, as life is too short to spend writing bad code.

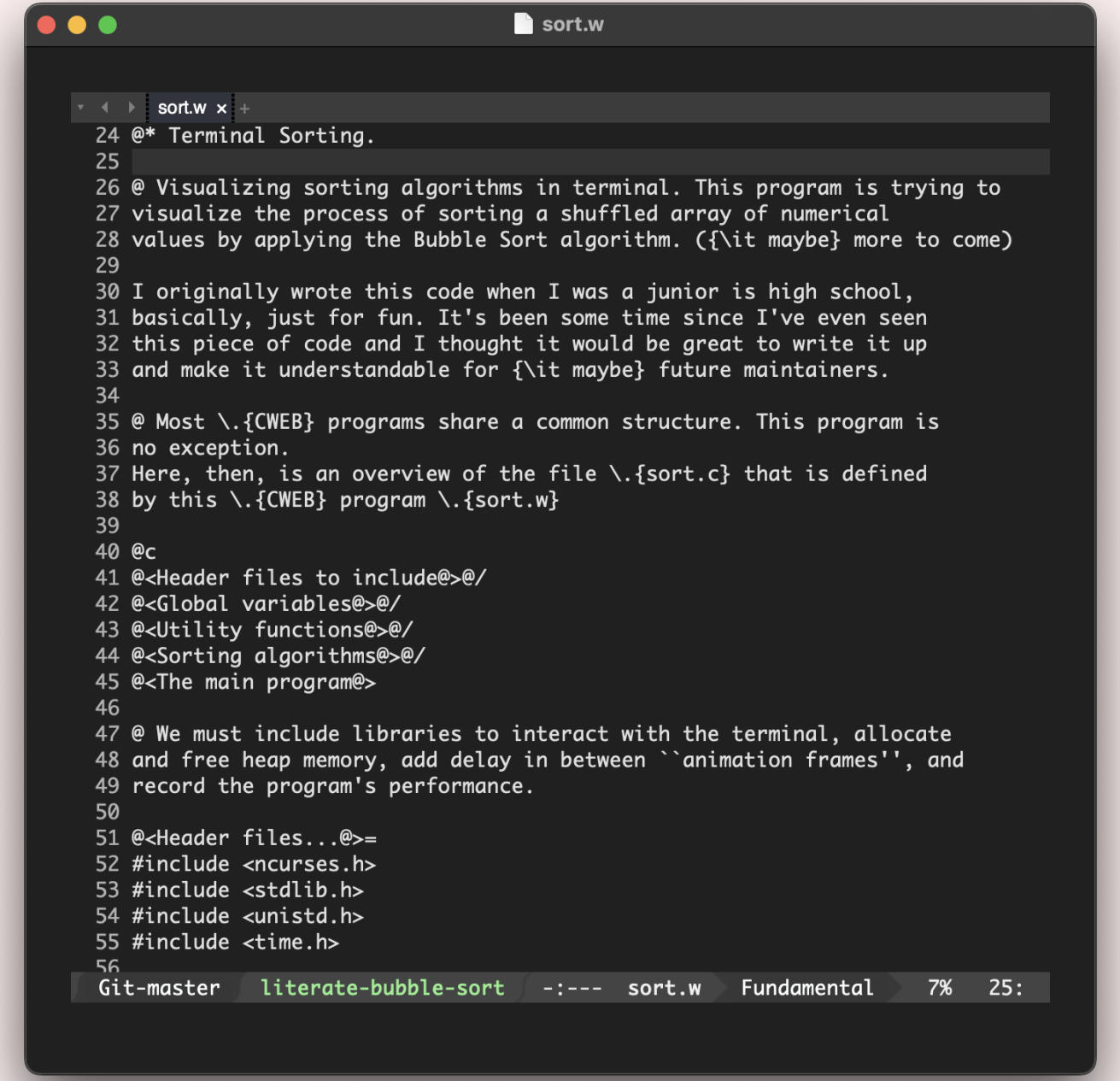
Armed with nothing but CWEB in my hand, I wrote a bubble sort visualizer in the terminal and to this day recall it as being one of the most fulfilling programming experiences I have had since the day I first learned PASCAL TURBO to cheat on my math homework.



CWEB for bubble sort

Writing in the mindset of Literate Programming is akin to teaching someone how to code—how to think.

In doing so, I have found myself thinking with so much clarity that I do not get when conversing with a machine.



November 12, 2022 at 21:49

1. Terminal Sorting.

2. Visualizing sorting algorithms in terminal. This program is trying to visualize the process of sorting a shuffled array of numerical values by applying the Bubble Sort algorithm. (*maybe* more to come)

I originally wrote this code when I was a junior is high school, basically, just for fun. It's been some time since I've even seen this piece of code and I thought it would be great to write it up and make it understandable for *maybe* future maintainers.

3. Most `CWEB` programs share a common structure. This program is no exception. Here, then, is an overview of the file `sort.c` that is defined by this `CWEB` program `sort.w`

⟨ Header files to include 4 ⟩
⟨ Global variables 6 ⟩
⟨ Utility functions 13 ⟩
⟨ Sorting algorithms 22 ⟩
⟨ The main program 7 ⟩

4. We must include libraries to interact with the terminal, allocate and free heap memory, add delay in between "animation frames", and record the program's performance.

⟨ Header files to include 4 ⟩ ≡

```
#include <ncurses.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
```

This code is used in section 3.

5. We also should have some quick definitions, such as delay between our frames in milliseconds. We can set the default value to 0, because the terminal refresh rate itself is a visual bottleneck, which we will talk

Demystifying computing

Now imagine if all the software were written with such care and attention. Writing beautiful software in the form of literature is not a skill locked behind some expensive education or innate talent—

—it is born out of passion for the craft of it, and in the absence of that, the ambition of being a professional and earning a deserved reward.

Caring primarily and only on the correctness of a computer program will lead to all the negative cases and examples we are all too aware of. Donald Knuth himself wrote TeX with Literate Programming.

The passenger becomes the driver

If technology and the modern surroundings constantly drive us to a wall at a pace that is truly inhuman—we can take back control of how we stay connected to the world by the virtue of building things and becoming more conscious of what we do and how we do it.

When everything is abstract, we get a sense that much can be learned abstractly, but that is a lie. Only experiencing events, people, and times can keep us connected to the ground we are standing on.

Only by questioning, challenging, and improving the archaic foundations of today would we actually be able to make *leaps* in our thinking.

Rounding up

I don’t have all the answers to the open-ended statements introduced at the beginning of this presentation. I can express on what I see and what I feel, but those are rather uninteresting in the face of exploring on why we see what we see and why we think what think.

What an interesting world it would be if more people would *make* stuff they use. If more people *built* their houses. If more programmers *loved* their code. If more of us refused to see the world through the reductive abstractions of the world and experience it first-hand.

Thank you.

Bibliography

See the next slides.

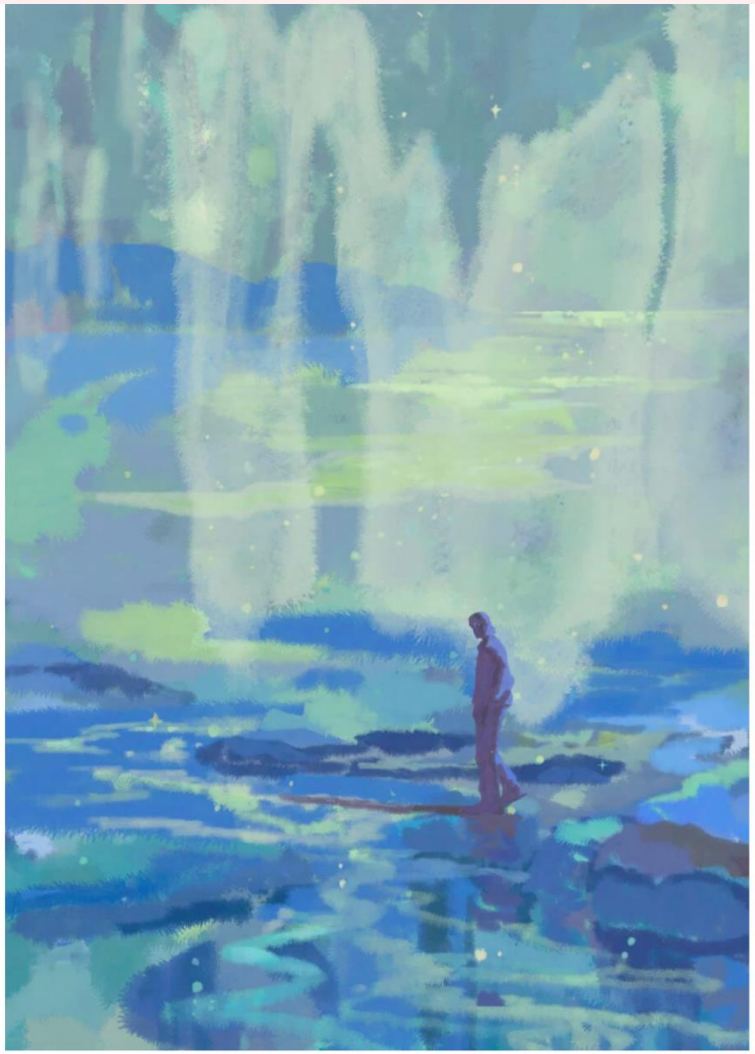


Illustration by Xiao Hua Yang

[Essay]

The Reenchanted World

On finding mystery in the digital age

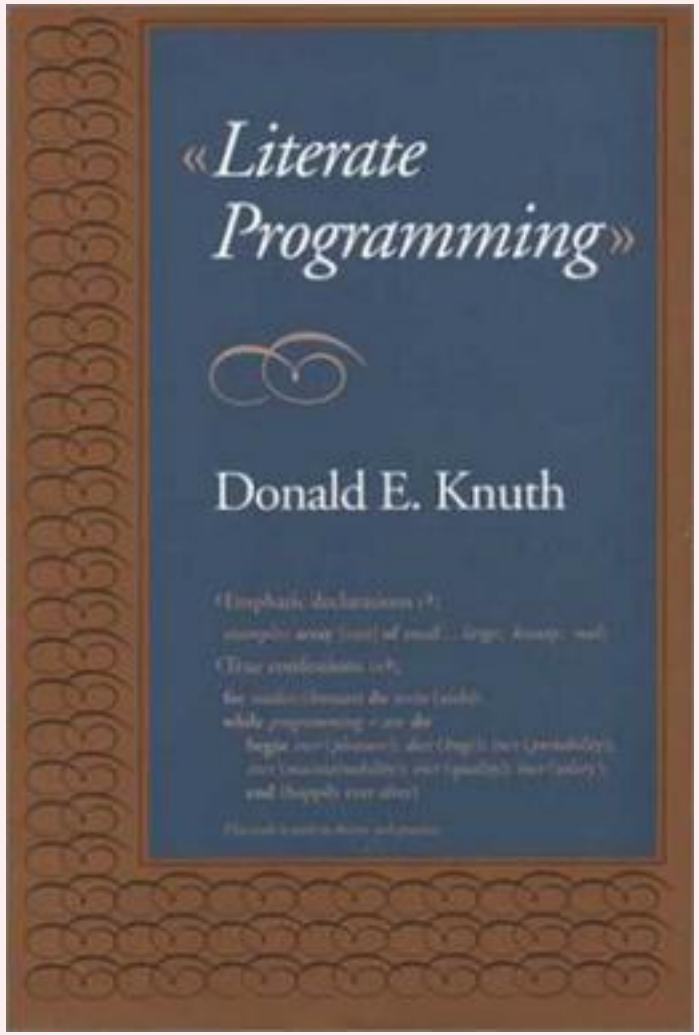
by **Karl Ove Knausgaard**, Translated by **Olivia Lasky, Damion Searls**

Adjust **Share**

Magnificent essay.

Inspired this presentation.

https://harpers.org/archive/2025/06/the-reenchanted-world-karl-ove-knausgaard-digital-age/



Literate Programming

Donald Knuth goes in-depth and explores Literate Programming paradigm in his magnum-opus book, “Literate Programming.”

If you are interested to learn more, this is the single source of truth.