Reimagining Personal Computing with E ink

Alexander Soto, Founder
modos.tech • alex@modos.tech
Re-imagining personal computing with a focus on creating calm, inclusive, and humane computing.
Meet the team

Wenting
Lead EPDC Developer

Brodie
CAD / Manufacturing

Michael
Software Architecture

Alex
Founder
Thank you to our Community and NLnet Foundation

300+ contributors
5k+ subscribers to mailing list
3k+ survey testimonials emphasize the need for healthier options
Community Survey Findings

- Reading: 65%
- Writing: 60%
- Coding: 51%
- Focused: 41%
“I lose hours and days and weeks of my life getting sucked down rabbit holes of entertainment content, and often miss deadlines or whole work sessions because my work and my play are a button away and the play buttons are so bright and colorful. It feels like my brain can’t focus or rest at the end of the day while I’m staring at a bright screen for hours…”

“Eye strain is a real problem. Using an e-ink display for creative writing would be much more pleasant than a standard display. I’d love to see e-ink technology continue to advance and become more accessible and applicable…”
"I have to use technology extensively for my career and I get issues on commercial screens (headaches, eye strain, overstimulation). I think e-ink is a much healthier alternative for everyone as it is a more natural and healthy way to use technology..."

"I’ve been programming since 11, now 33 years old. My eyes hurt, even though wearing blue light filtering glasses, go outside regularly, etc. I have been wanting this for years..."
Desire for a Balanced Digital Life

- Reducing screen time on social media/entertainment.
- Digitally unplug and spend more time outdoors, away from screens.
- Seeking less visually stimulating digital environments.
- Reducing digital clutter and minimizing distractions.
Health and Comfort Concerns

- Eye Fatigue and strain from prolonged screen exposure.
- Specific Health Issues:
  - Myopia
  - Epilepsy
  - Light sensitivity
  - Headaches
  - Migraines
  - Traumatic brain injury
  - Post-concussion syndrome
“All of us are temporarily able-bodied and at some point in our lives, will face new kinds of exclusion as we age, when we design for inclusion, we are designing for our future selves.”

- Kat Holmes

Mismatch: How Inclusion Shapes Design
There’s a need for...

- Creating technology that satisfies our essential needs while protecting our well-being.
- Redefining the role of our digital devices to foster a healthier, balanced life.
- Creating a new class of devices, built from scratch, to embody the principles of ‘humane technology’ through hardware and software design.
What about Software?

- How can we be productive in a "calm" world?
- Can work by synced in the cloud without interrupting focus?
- Can we collaborate without notifications?
- Can we scale minimalist UI to more than just reading and writing?
Write

a simple text editor for prose & code

Reference

an exploration into “browsing for information” using Gemini

First: The Basics

Lorem Ipsum Dolor


Fast Thoughts on a Fast Language

For open the bullish weekend diving into Ada, a language I tried out many years ago but have since put on the backburner.

I digressed this, since that time, I’ve become much more well-versed in C and Java, and also have some knowledge from college with a C++ engines. maybe I can make some notes of Ada’s idiosyncrasies and such.

In this post I’ll outline some initial thoughts from this weekend project.

Thought 1: Ada is a beautiful language

Many languages have a syntax I think (few used since C++) that Ada has designed. Sometimes it feels like a grand hat that symmetrically defines orthogonality to each other, but once you start writing software of ambitions it is crystal clear that all those abstracts fit into place besides each other. The result is that you can basically look at any source file and know exactly what is doing. the line by line level since you have a minimum amount of experience.

This is the exact opposite type of “learning” to what we’ve put into something like Common Lisp, which totally eschews this control and puts you having your own preferred way of doing things. In terms of actually writing a program (as opposed to being someone who is thinking about what to do), Common Lisp basically lets you do everything, while Ada provides you to stop the writing a program pants and get on with the doing something useful part.

The system is suitable, yes, but in the program is being structured in a way to remove the unnecessary of what everyone was already doing in C, and that is really really awesome. The whole “everything is an object” nature of parameter declaration is really a breath of fresh air when coming from C, for example: I’ve gone a big leg of languages where do more programs without abstract substructures for common operations. Ada’s powerful function libraries are something that truly express means in that way, and is easy to read through.

By far what I like the most about Ada is the language, though, in that it was designed from the ground up as a mediocre language for messy programming. It has Structs which allow you to show level stuff such as define the bit layout of types, and then specific meta functions in C. Such a change Hellman around and adapts to them with rails, or deal with high-level abstractions using lexical container types.

[URL] ada.com/clamp/fast-thoughts-on-a-fast-language.txt
How Do We Make This Scale?
A core framework in development designed to tailor applications and documents for use across various devices and end users’ needs.

Meet Modalix

Source

Fast Thoughts on a Fast Language

I’ve spent the holidays writing a language called Modalix. It’s not nearly as fast as Fast, and in this post I’ll reflect on some initial thoughts from this weekend project.

**Thought 1:** Ada is a very difficult language.

Memory in Ada is a challenge. I spent much of this weekend trying to compile some Ada or translating it into some other language. Ada is a very low-level language, and many super-features are built into the language itself.

**Thought 2:** Ada is an extremely difficult to get started with.

I’ve spent nearly as much time trying to compile some Ada or translating it into some other language. Ada is a very low-level language, and many super-features are built into the language itself.

**Thought 3:** Ada is an extremely difficult to get started with.

I’ve spent nearly as much time trying to compile some Ada or translating it into some other language. Ada is a very low-level language, and many super-features are built into the language itself.

**Thought 4:** Ada is an extremely difficult to get started with.

I’ve spent nearly as much time trying to compile some Ada or translating it into some other language. Ada is a very low-level language, and many super-features are built into the language itself.

I have a few ideas for future projects to try out maybe next weekend. I do really like Ada

- Reading it or writing it in
- Part of some of my C programs into Ada to find real edge cases in the Language.
- Do something with Ada’s built-in signal handler system to replace getting terminators.
Modalix

- Adapts the interface to user preferences, including dynamic adjustments to meet specific requirements.
- Utilizes a semantic model to guide adaptation across different modalities, such as screen types, operating systems, devices, and input methods.
- Developers and users can extend and enhance the experience for each modality.
Approaches

1. Re-structure the inherent complexity of user interfaces
2. Moving complexities associated with specific modalities, higher in the stack to prevent them from affecting each other
Challenges

1. The complexity in the semantic representation of user interfaces.
2. Backward compatibility with existing applications
Next Steps

1. Crowdfunding Campaign
2. Get Involved with Modos