

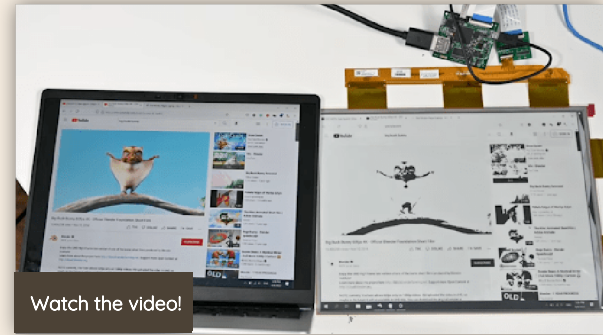


Reimagining Personal Computing with E ink

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



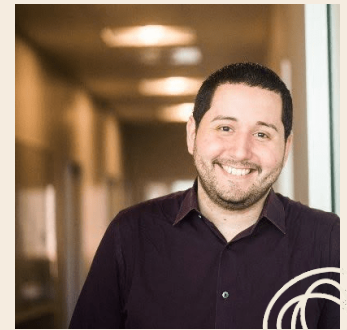
Wenting
Lead EPDC
Developer



Brodie
CAD /
Manufacturing



Michael
Software
Architecture



Alex
Founder

Thank you to our Community and NLnet Foundation

300+

contributors

5k+

subscribers to
mailing list

3k+

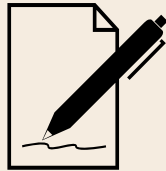
survey testimonials
emphasize the need for
healthier options



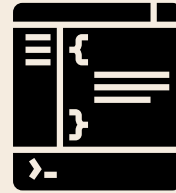
Community Survey Findings



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 be synced in the cloud without interrupting focus?
- Can we collaborate without notifications?
- Can we scale minimalist UI to more than just reading and writing?

First: The Basics

10:30 AM

Lorem Ipsum Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed in pharetra metus, in consequat ipsum. Donec pretium nisi sed posuere mollis. Nam volutpat nulla quis urna euismod aliquet. In purus lorem, tristique vel dolor aliquet, lobortis convallis nunc. Vivamus iaculis est dolor, nec mattis elit pharetra vel. In lacus nunc, congue eu lacus vel, porttitor pulvinar nisl. Nam id tincidunt tortor. Cras cursus elit ac odio pretium tempor.

Sed tincidunt bibendum nibh molestie suscipit. Curabitur a ipsum malesuada, vulpate purus eget, rutrum urna. Sed elementum feugiat eleifend. Aliquam tincidunt arcu non feugiat iaculis. Fusce non placerat arcu. Etiam facilisis est mi, sit amet ornare nibh interdum vitae. Duis faucibus molestie dolor id egestas. Proin id gravida dolor. Duis accumsan imperdiet neque, vitae consectetur orci viverra non. Aliquam volutpat nisi sed hendrerit egestas. Nam sed feugiat nisl. Nulla vehicula consectetur lectus at lacinia.

Phasellus cursus vulpate ipsum a malesuada. Morbi consequat libero at leo scelerisque, sit amet posuere enim gravida. Nullam pellentesque metus nec sodales pellentesque. Mauris eget nibh a arcu ornare ullamcorper sed eu mauris. Nulla iaculis blandit tortor, non auctor sem molestie quis. Sed in sem tristique, iaculis velit non, ultrices ante. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Nunc pharetra quis augue a consectetur. Sed auctor fermentum risus, malesuada mattis quam. Donec eget placerat erat. Fusce ac imperdiet ipsum. Curabitur vel magna molestie, hendrerit augue eget, scelerisque ante. Maecenas erat dolor, mollis eu augue vel, finibus posuere nunc

Write

a simple text editor for
prose & code

Reference

an exploration into "browsing for
information" using Gemini

Fast Thoughts on a Fast Language

I've spent the holiday weekend diving into Ada, a language I tried out many years ago but have since put on the backburner.

I figured that, since that time, I've become much more well-versed in C and Rust, and also you know graduated from college with a CS degree so maybe I can make more sense of Ada's idioms and whatnot.

In this post I'll collect some initial thoughts from this weekend project.

Thought 1: Ada is a beautiful language

More so than almost any language I think I've used since OCaml, Ada feels *designed*. Sometimes it feels like a giant list of syntactic structures defined orthogonal to each other but once you start writing software of sufficient size you realize that all those structures fit into place besides each other. The result is that you can basically look at any stanza of Ada code and know exactly what it's doing at the line-by-line level once you have a minimum amount of experience.

This is the exact opposite type of "beauty" as I ascribe to something like Common Lisp, which hands you a world without rules and lets you build up your own preferred way of doing things. In terms of actually writing a program (as opposed to doing something useful through developing software, which is unrelated), Common Lisp basically lets you do anything, while Ada prods you to skip the writing a program part and get on with the doing something useful part.

The syntax is verbose, yes, but in the process it builds up a structured way to reason about the semantics of what everyone was already doing ad hoc in C, and that is really really awesome. The whole "in/out/in out" system of parameter declaration is really a breath of fresh air when coming from C; for example, I'm also a big fan of languages which divide program sections into ordered subsections for common operations; Ada's procedure/function header is reminiscent of Ruby's implicit-rescue syntax in that way, and it's enjoyable to read through.

By far what I like the most about Ada the language, though, is that it was designed from the ground up as a cohesive language for systems programming. It has features which allow you to do low-level stuff such as define the bit layout of types, use paradigms more familiar to C such as throw buffers around and assign to them willy-nilly, or deal with higher-level abstractions using logical container types,

gemini:// tilde.pink/~slondr/fast-thoughts-on-a-fast-language.gmi



How Do We Make This Scale?

Meet Modalix

A core framework in development designed to tailor applications and documents for use across various devices and end users' needs.

Source

```
# Fast Thoughts on a Fast Language
I've spent the holiday weekend diving into Ada, a language I tried out many years ago
I figured that, since that time, I've become much more well-versed in C and Rust, and
in this post I'll collect some initial thoughts from this weekend project.
# Thought 1: Ada is a beautiful language
Moreso than almost any language I think I've used since OCaml, Ada feels designed/.
This is the exact opposite type of "beauty" as I ascribe to something like Common Li
The syntax is verbose, yes, but in the process it builds up a structured way to reas
By far what I like the most about Ada the language, though, is that it was designed
# Thought 2: Ada is incredibly difficult to get started with
I've spent nearly as much time trying to compile some tool or toolchain or solve som
I use Arch Linux, btw. Arch packages the GCC front-end for Ada, GNAT, as the gcc-ada
Unfortunately, every other Ada program ever written uses something called gprbuild,
# Thought 3: Ada is fast
I mean, it's as fast as C (it compiles to C object code and goes through all the opt
But like, I mean, I'm writing naive algorithms in Ada which are beating optimized (y
# Thought 4: Where I'm going next
I have a few ideas for projects to try out maybe next weekend. I do really like Ada
• Erlang NIF or port in Ada
• Port some more of my C programs into Ada to find new edge cases in the language, I
• Do something with Ada's built in signal handler system, maybe try getting it running on a
```

Minimalist Reader

Fast Thoughts on a Fast Language

I've spent the holiday weekend diving into Ada, a language I tried out many years ago but have since put on the backburner.

I figured that, since that time, I've become much more well-versed in C and Rust, and also you know graduated from college with a CS degree so maybe I can make more sense of Ada's idioms and whatnot.

In this post I'll collect some initial thoughts from this weekend project.

Thought 1: Ada is a beautiful language

Moreso than almost any language I think I've used since OCaml, Ada feels *designed*. Sometimes it feels like a giant list of syntactic structures defined orthogonally to each other, but once you start writing software of sufficient size you realize that all those structures fit into place besides each other. The result is that you can basically look at any stanza of Ada code and know exactly what it's doing at the line-by-line level once you have a minimum amount of experience.

This is the exact opposite type of "beauty" as I ascribe to something like Common Lisp, which hands you a world without rules and lets you build up your own preferred way of doing things. In terms of actually writing a program (as opposed to doing something useful through developing software, which is unrelated), Common Lisp basically lets you do anything, while Ada prods you to skip the writing a program part and get on with the doing something useful part.

The syntax is verbose, yes, but in the process it builds up a structured way to reason about the semantics of what everyone was already doing ad hoc in C, and that is really really awesome. The whole "in/out/in out" system of parameter declaration is really a breath of fresh air when coming from C, for example. I'm also a big fan of languages which divide program sections into ordered subsections for common operations. Ada's procedure/function header is reminiscent of Ruby's implicit-rescue syntax in that way, and it's enjoyable to read through.

By far what I like the most about Ada the language, though, is that it was designed from the ground up as a cohesive language for systems programming. It has features which allow you to do low-level stuff such as define the bit layout of types, use paradigms more familiar to C such as throw buffers around and assign to them willy-nilly, or deal with higher-level abstractions using logical container types.

gemini://tilde.pink/~slondr/fast-thoughts-on-a-fast-language.gmi ← →

Low-Vision Reader

Fast Thoughts on a Fast Language

I've spent the holiday weekend diving into Ada, a language I tried out many years ago but have since put on the backburner.

I figured that, since that time, I've become much more well-versed in C and Rust, and also you know graduated from college with a CS degree so maybe I can make more sense of Ada's idioms and whatnot.

gemini://tilde.pink/~slondr/fast-thoughts-on-a-fast-language.gmi

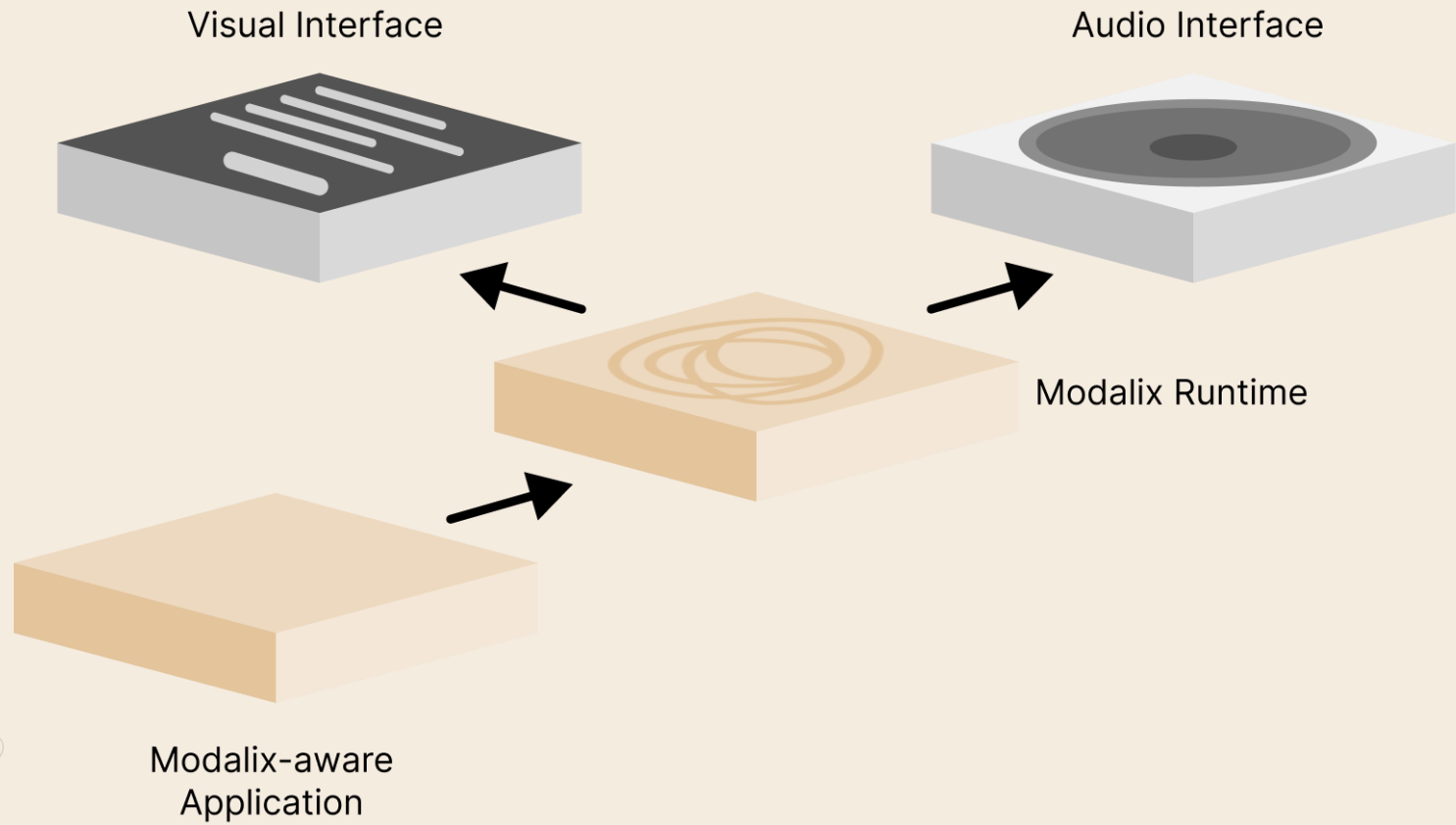


Modalix

- 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.

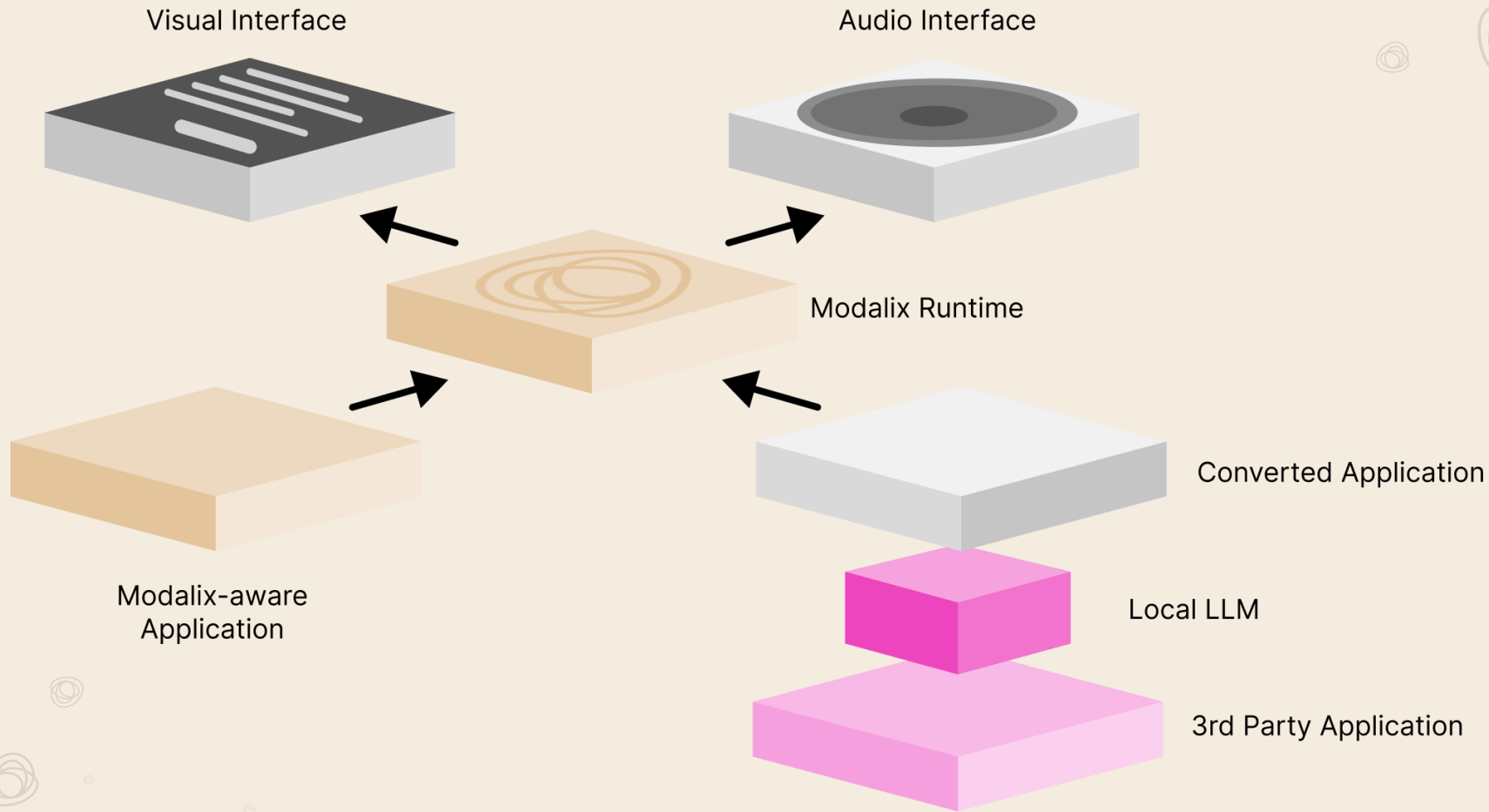
Approaches

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



<https://www.modos.tech> • alex@modos.tech