I get asked "what is that weird arse editor you're using?" by people .I a lot bespoke*use*a

Developers tend to argue about stylistic choices quite often; things like tabs vs spaces, snake case vs camel case (python vs perl??), requiring semi-colons/trailing commas vs no semicolons/trailing commas... things that don't impact function, just form. This is because most developers suffer from a condition that requires them to a) always be rightand b) do the least ammount of work possible, which as we all know the is to offer unsolicited advice and be shitty when the advised doesn't immidiately fall in love with the advice they didn't ask for.

How Acme got it right but got it wrong

The advent of windowed terminals has given each user what amounts to an array of teletypes, a limited and unimaginative use of the powers of bitmap displays and mice. Systems like the Macintosh that do involve the mouse as an integral part of the interaction are geared towards general users, not experts, and certainly not programmers. Software developers, at least on time-sharing systems, have been left behind.

source: acme

At the time it was introduced acme went all in on the mouse, not any old mouse a very specific 3 button old mouse, which was the ubiquitous pointing device.

source: acme mouse

The reason I put such a clickbaity heading at the beginning of this section is because, I can take the statement above, update a couple of technologies with more current ones and the general statement still holds true.

The advent of "cloud computing" has given each user what amounts to an array of mobile devices, a limited and unimaginative use of the powers of infinite compute and mutli-touch displays. Systems like the iPad that do involve the multitouch as an integral part of the interaction are geared towards general users, not experts, and certainly not programmers. Software developers, at least on traditional desktop systems, have been left behind.

So I would suggest what acme didn't get quite get right was the specific input device type, but

exploring a alternative input device was I think the right call.

At the time, there was really one pointing device and had Rob Pike predicted multitouch would be this prevelant, I suspect he would have retired to a private island in the carribiean with all the money he made from predicting the future.

What I have tried to do with this experiment is have acme be a more general purpose editor.

I was able to embed tree-sitterinto acme and have a functional ASTs.

While the ASTs generated by tree-sitter aren't the most accurate representation of what the respective languages would generate, we only need document level node parsing to be able to accurately manipulate the said ASTs.

Effectively, it doesn't matter if it can't correctly parse all the symbols in a buffer, we are only interested in the document, not what nodes resolves to.

First improvement I made was an indicator to show the active window

Second thing I've done was to make the indicator also use a color to indicate what mode the window was in.

And third thing I added was the ability to use Escfor chanding ala vi mode, Nfor the next node and Pfor the parent node, and Cfor the child node.

I'm convinced this is the right direction for acme.

Next steps for me will be removing all code that handles user input and moving them to their own input servers.

basically...

```
mouse-coordinates -> mouse server -> acme-commands   keyboard-opcodes -> keyboard server -> acm
gestures -> gesture server -> acme-commands          /
```