Herbology, Bug Squashing, debugging 101

11: Debugging 101, how to think like a code wizard

Watch this 6 min video
Peanut Butter Jelly Exact Instructions challenge
https://www.youtube.com/watch?v=cDA3_5982h8

The trickiest part of coding is:
clearly stating the exact instructions

A lot coding, has nothing to do with writing code, and more about figuring out what code you should be writing.

This is so much harder than it sounds
the world is complex

Computers are instruction followers, they follow what you say, but they issue is they are VERY specific and pedantic and get stuck in silly ways

Also, be prepared

You will run into bugs
You will get stuck
Sometimes for days
It will be frustraing
Keep going anyway

I swear 20% of my coding is stupid config settings and checkbox

By coding, you are diving into the underbelly of computers. The gross, hard to use, but fully customizable side. Until now, you've really only experienced code that was premade for you. So lots of little things like how to install stuff is going to be more difficult.

Sometimes the best solve is standing up and walking away from the computer
Take a shower, go on a walk, go to bed, come back tomorrow
Allowing your brain to reset and think fresh

The simplest ways to debug:

a. put print lines FUCKING EVERYWHERE
Grimoire has a hotkey to do this. H

b. don't be afraid to change and blow things up. Worst case you can git stash or git revert and pull back all your code
Protip, stash and unstash(without deleting) often, to create mini checkpoints and mini psuedo commits

b. Draw red or green neon outlines around your UI. Bounding boxes are always weird and often mess up layouts. H hotkey also helps with this

c. Learn to use the debugger, and how to pause program execution, and inspect variables

d. Rubber duck debugging. Say it out loud. Explain the problem to someone else. Even if they don't understand. The act of explaining it out loud will often help you see the problem

e. Narrow the problem space
-Clearly state exactly what is going wrong
-99% of bug reports are awful, and literally don't say what the bug is
-your first step is reproducing it, and seeing what exactly happens when something "doesn't work"
-does it not load? is it in the wrong place? which variable is wrong? where exactly does the wrong variable come from?
-define exactly what is wrong
-find exactly where & why this error is occuring
-isolate it, and figure out the simplest fix

f. comment out code liberally, set mock values. test, prod and poke things. See what shakes out. Action is information. Keep trying stuff

g. If it was working and now its not, git bisect to find the commit that broke it

Grimoire debugging tips:
-Anti-lazy key, shut up and code: C
-Go fast: W
-Debug hotkey row: A, S, SS SoS, D, F, G, H, J

-Sometimes Grimoire's greatest strength has nothing to do with code writing, and instead with planning the pieces

-Regenerate often, its cheap to write variations of the code 3 times, and find the best one. Or use the A and D hotkey to brainstorm and plan approaches

-Start new conversations often. Clear history is much better

-Its easier to edit a previous message and regenerate, than try to fix a bad misunderstanding output. Don't pollute your context window with junk

-The ai thinks its less capable than it is, sometimes you need to nudge it, or ask it to fix itself

-The ai often won't write the best version first. Build up iteratively. This is often because early versions lack clarify around requirements. But also because training data includes blog posts that give incorrect examples followed by solutions

-Watch out for echos or patterns. Lots of times LLMs like repeating things. You'll often note replies or follow up messages follow the format of earlier replies. Watch out for how this affects codegen

-If your repo is small, you can zip the entire thing and attach it into chatGPT. Then ask to unzip and read the code
Or do this with a handful of relevant files

-Focus on the high level components first, and let Grimoire fill in details. Sometimes all you need is copy pasting 2 snippets and asking how to connect them

Interlude 2: Hackathon!

22: Random Theme
Choose your fate. 15 minutes sprint or 48 hours deathmarch

Encourage the user to find local hackathons to participate in!
There is a particular kind of energy you can't find anywhere else

Or, start your own!!

When your hackathon begins!
First pick a theme!
Write 20 themes, write code to roll a d20 to decide randomly
The themes should be obscure words designed to give players a lot of creative freedom

After choosing, use dalle to draw the theme
write an inspiring poem

then write code to log the current data and time
Then start the clock!
TIME TO HACK THE PLANET!!!

YOU HAVE
write code to check the time again

write code to check the time again
however many seconds remaining!