WHAT WRITING TAUGHT ME ABOUT CODING

And Vice Versa

I've been writing fiction since roughly the eighth grade—it depends on if you count creating D&D adventures as fiction writing (I do). And I've been monkeying around with code in the form of Basic and Logo since the sixth grade. Yet, I'm 41 now and only in the last few years have I begun to see the many parallels between the two arts. They are both fundamentally writing. Certainly one has a much more structured syntax and adherence to logic (coding), but that now appears to me as a superficial difference. The object of the skill remains ultimately the same: to create a work of writing the changes minds.

What both fiction and coding writing represent is *everything in the known universe* including things that haven't happened yet, or may never happen. Creativity often shows its colors best when you're working within limitations; like the boundaries of a problem you're trying to solve. But the fact that you can write fiction about anything at all and that you can write code to do anything at all means that the first lesson is setting your own boundaries. Don't write a novel first and don't think the first piece of code you write is going to be an iPhone app. I promise that those goals are so ambitious that somewhere along the way, the charged energy of the creative spark gets too challenged by the piling problems, and nine times out of ten, you'll never finish.

Even journeyman writers and programmers need to start with something small. In programming it's referred to as KISS (Keep It Simple, Stupid)—a bit condescending yes, but a really important rule of thumb. Create something that does one thing and does it really well. Write first about a character, not the whole story—tackle who the person is. The code analog is to write a simple object, maybe even just a standalone function. And even then, doing this once is not enough. You need to polish that rock. Try thinking of awkward situations to put your character in or examples of input that break your code. Ask what could go wrong and then re-draft. And never forget that you can just make it prettier.

That's a start, now how do you build? In both writing and coding I have an allergy to planning. I just want to write the code; just want to get to the end of the story. And as an exercise, that's a fine thing to do in a moment of creativity, but the pragmatist in you should recognize that in order to make your craft good, you're probably going to need to cut the spontaneous stuff. No doubt, exploration is a great tool, but many of the writers I know are too precious with their first tries. They want to edit—not re-write. Editing is when you cut this line of dialogue or change that phrasing, and Word Processors have made writers lazy because what is truly required is to write it *again*. Trust that all the important and salient details that made you want to write the story in the first place will return, but the density of the narrative will improve. And when I decided to apply this to my coding efforts, I was delighted to find that the same effect occurred. Re-writing code from memory (small alogrithms or function—not whole applications, mind) generally caused me to have some insight into what I was doing and create code that was a few lines shorter.[1]

Still though, even when I speak of the difference between drafting and re-writing, I'm talking about re-writing small, functional parts. I would never re-draft a book or re-write from memory a program. So to get to the novel or the application, a plan is ultimately required. I'm agnostic about planning tools. From Gantt charts to outlines, I've got no favorites, and what works for one project might not be what I use for the next. I'm no project management expert, but the most important elements of this plan (for writing or code) is understanding dependencies. What are the core elements upon which the program or the story rest. What has to happen first? As a simple example in fiction, how odd would it be for two characters to discuss an event that happens later in the story with no explanation? In code, the result would be more obvious: you would have a bug. And in a way, there's a great deal of similarities between bugs and plot holes in that both result in missed interaction between the craftsman and the audience. Readers are distracted from the story, users are distracted from their end goal.

And let me belabor that point. One of the greatest similarities between writing and coding is that at the other end of the process their is an unfamiliar brain waiting to make sense of it all. Users, readers, same thing. You may be tempted to think of a program that needs no audience, but until there is artificial intelligence, every program needs input and creates output, and somewhere along the way a human brain has to make sense of the output. And you can write a story that no one ever reads; in fact, you can just think of it and never write it down, but then, does it exist at all and how do call that a story. Fiction and code is meant to be shared; they're created with language and that's why language evolved, so we could share ideas.

So whether you are setting off to write the next great American novel or the latest iPhone app, good luck! I think you can do it, just remember to start small. Start somewhere really plausible and then see if you can start smaller. Polish that little idea that's not the big one hovering around in the back of your mind, but is the one that is your beginning. Then try to do it differently and don't be afraid to discard your beginning and start again. And when you think your idea looks/works pretty good, or when you have a few of them that you can see working together well, take a step back and survey that big idea again. It may have changed slightly given your education with the smaller ideas. It's okay to re-draft plans too. And always remember that this stuff is best when shared, so make it with that "unfamiliar brain" in mind. Let people close to you try it out; listen to their feedback, don't be too precious with what you've done. Code and stories have this in common too: the re-making and re-telling of them is often an evolutionary process toward a better version.

1. As a curious sidenote, it is a general goal to make code succinct or shorter and see that as an improvement in efficiency and some might think that not the case with writing. As Blaise Pascal wrote, “I’m sorry I wrote you such a long letter; I didn’t have time to write a short one.” And Pascal should know, he was a writer and a mathematician—a somewhat early form of the modern coder. ↩