## Learning Rust

My first game

Starting out in the industry there's been one thing I've heard continually. Rust is cool, and you should learn it. So, after endless nagging by a few colleagues and the swathes of social media that pushes it, I’ve made the plunge.

A few colleagues and I chose to partake in GitHub’s GameOff 2023. So, we had to choose a set of technologies to work with. In this case, we chose Rust as the language for our game. But we were left scratching our heads with what game engine to use (or indeed whether to use one at all). We boiled it down to two options: Gadot, a relatively famous C# engine which had added some support for the Rust developers, and Bevy, a lightweight, Rust first game engine that was still in its infancy. We ended up choosing Bevy in this case for a ‘full Rust experience’ as one of the members on our team who was much more into Rust said that Gadot’s support for Rust was good but not great.

### So, how’s Rust?

Well, I found Rust to be a rather refreshing affair. Rust’s borrowing and explicit pointer management and were not odd for me and I found that only two things that left my head spinning were lifetimes and closures. Those Darn pipes just look so strange, and it took me just way too long to make the connection that they’re just anonymous functions like in any other language. The compiler was the best part of Rust. Its error messages are clear, concise, helpful and holds your hand throughout the whole process. I never had a moment where I felt like I was fighting it.

Overall, I’d say Rust is a great language to learn if you’re looking to build some memory efficient code and if you’re looking to get a start with learning some lower-level concepts. Unfortunately, I can’t comment on how the language holds up in an enterprise environment as the game’s scope was just too small. I can see that it’d be awesome but, of course, any language is only as good as its support. I’m sure that there are plenty of crates to help you and/or your company build whatever you need. But I’ll leave that to your own investigations.

### How’s Bevy?

Bevy is an entity component and systems engine (ECS) which means that when you’re writing the game code, you’ll be:

1. Creating components
   1. These are simple structures like `Weight`, or `Direction` which can be combined to form Entities.

<Code snippet here>

1. Spawning and Querying for Entities
   1. These are the individual ‘things’ in the Game world. The thing that took a while to understand is that nearly everything is an entity. If I made a fish, it was an entity. If I was looking for the death timer for a fish then, that itself was also an entity.

<Code snippet here>

1. Writing systems
   1. These are the functions that run the game. You might create a system like `fish\_movement()` which would find all the fish in the world and then move them in the direction that they’re meant to go.

<Code snippet>

All this is to say that when you first start, the game engine follows a paradigm in ECS and that paradigm is absolute. So, if you’ve worked with other engines, this might be a bit of a head spin to start with. As it was for a few of the members who worked on gone-fish with me.

Overall, as you end up building isolated systems, you end up with many opportunities to work with Rust in small doses. So, you can write simple code, and then immediately see the results of that code. I found it to be great for learning!

Here’s an example of one of these small systems:

<Code snippets and Images of the games results>

### Conclusions

I found that both Rust and Bevy were, when combined, an excellent first experience with both Rust and Game development. I’d recommend this setup for anyone who’s looking to take their first foray into game development and the Rust programming language.

I’d recommend that if you’re going to give this expedition a try here is a list of things that I’d do for the best experience:

1. Start off with the Rust Book. It’s a near perfect document for learning the language and doing it will give you a huge head start to the process of working on the game.
   1. I’d complete up to section 3 (3 included) if you want to speed run the learning experience. If you want to be fully prepped, then I’d go all the way to 6 (6 included).
2. Then move on to the Bevy Book where I’d just follow the guided tour and I’d finish the first part. Ending in the Audio section.
   1. There’s also a phenomenal series by Jaques on YouTube which will take you through building a full game in Bevy! I found it much better than the book. However, not because the book was bad. I just prefer video where possible. <https://www.youtube.com/watch?v=TQt-v_bFdao>
   2. There’s a caveat to both suggestions though. As Bevy is a new, and constantly in flux game engine. Neither of these is entirely up to date with the version of Bevy that I used to build the game gone-fish. The book, at time of writing, was sitting somewhere between v0.9 and v0.11 and the tutorial linked is at v0.10.

If I were to do this all over again, I’d build my first game on the same versions as these tutorials, either v0.9 or v0.10. That way you don’t need to do any migratory steps while you’re still learning both.