## 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. Coming from a university background where we learned with languages like C++, Haskall, Python and JavaScript. So, Rust’s borrowing and explicit pointer management and were not odd for me. I found that only two things that left my head spinning over the month of development was 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.

I found the compiler to be the best part of Rust. It’s very clear and 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.

### How’s Bevy?

Bevy is an entity component and systems engine 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.
2. 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.
3. 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.

All this is to say that when you first start the game engine follows a paradigm 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.

### Conclusions

I found that both Rust and Bevy were, when combined and 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, 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. Then, I’d star the Bevy Book