Extracted from:

Hands-on Rust

Effective Learning through 2D Game Development and Play

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Hands-on Rust

Effective Learning through 2D Game Development and Play

Herbert Wolverson

edited by Tammy Coron

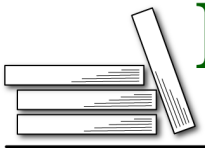
Hands-on Rust

Effective Learning through 2D Game Development and Play

Herbert Wolverson

The Pragmatic Bookshelf

Raleigh, North Carolina



Pragmatic Bookshelf

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Tammy Coron
Copy Editor: Vanya Wong
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-816-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2021

Preface

Rust is an exciting programming language that combines the power of C with memory safety, fearless concurrency, and productivity boosters. It offers close-to-the-metal power and performance, while also providing a safety net to avoid many of the more common bugs found in low-level languages. Because of its features, Rust is a very competitive systems and game development language and is currently enjoying rapid growth amongst industry giants, including Amazon, Google, Microsoft, and many game development houses.

A great way to learn and study Rust is through game development. Don't be discouraged by the scale and polish of AAA titles. Small indie games are fun, and hobby game development can kickstart careers in professional game dev or unrelated development fields. Every successful game developer started small, gradually gaining skills until they could work on the game of their dreams.

In this book, you'll learn Rust by walking through game development examples. You'll gain knowledge and confidence in both Rust and game development as you work through a series of practical examples, building increasingly complicated games. The text emphasizes a pragmatic "learn by doing" approach. Theory sections are short and are followed by concrete examples for you to try. By the end of the book, you'll have mastered the basics of the Rust language and be well-equipped for tackling more complicated game development problems.

Who Should Read This Book

This book assumes that you have some prior programming experience. It gently introduces you to Rust and game development concepts. If you've written anything more complicated than a "Hello, World" program in another programming language, you should feel comfortable working through this book's examples.

This book is ideal for anyone who wants to give Rust a try—it does not assume that you know the Rust language. It's also well suited for Rust developers who want to try their hand at game development. Alongside an introductory programming tutorial, it could also be helpful to new developers.

What's in This Book

This book walks you through a typical game developer's journey while also teaching Rust's key concepts. Each chapter adds to your knowledge and skillset as you build playable games:

Chapter 1, Rust and Your Development Environment, on page ?, begins your Rust journey. In this chapter, you'll install the language toolchain and work with Rust source code in a text editor. You'll step through creating a "Hello, World" program and learn to use Rust tools like *Cargo* and *Clippy* to improve your productivity.

Chapter 2, First Steps with Rust, on page ?, walks you through the basics of Rust development. You'll hone your skills by building a treehouse guest manager. The chapter covers text input and output, grouping data in structures, and core Rust concepts such as iterators, pattern matching, `if` statements, functions, and loops.

The first two chapters teach you everything you need to know to make simple games. Chapter 3, Build Your First Game with Rust, on page ?, puts this knowledge to use as you create your first game—*Flappy Dragon*.

Chapter 4, Design a Dungeon Crawler, on page ?, helps you plan your game design. You'll learn to make a game design document, and you'll use it to transform an idea into a playable game. You'll design a roguelike dungeon crawler game and whittle your ideas down to a *Minimum Viable Product (MVP)*.

In Chapter 5, Build a Dungeon Crawler, on page ?, you'll begin building the dungeon crawler you designed. You'll learn about random numbers, map structure, and handling an interactive player. You'll also add the beginnings of monsters to your map, and you'll discover how to make tile-based graphics.

Games become increasingly complex as they grow. In Chapter 6, Compose Dungeon Denizens, on page ?, you'll use *Entity Component Systems (ECS)* to tame complexity, reuse code, and manage interactions between game entities. You'll implement the player and monsters with the ECS, reusing systems to cut down on the code that you have to write. You'll finish the chapter with a multi-threaded, concurrent game.

In Chapter 7, Take Turns with the Monsters, on page ?, you'll add a turn-based structure to your game where the player moves, and then the monsters move. You'll learn how to structure games to implement game rules and schedule different ECS systems based upon the game phase. You'll also learn to make your monsters wander randomly.

Chapter 8, Health and Melee Combat, on page ?, gives game entities hit points. You'll also make a Heads-Up Display to show the player's current status. You'll learn to make monsters search for the player, and you'll implement a combat system to slay—and be slain by—your player's enemies.

In Chapter 9, Victory and Defeat, on page ?, you'll add a game over screen, indicating that the player has lost the game. You'll also add a winning condition to the game and a second screen to congratulate the player on their victory.

Before Chapter 10, Fields of View, on page ?, your character is omniscient—they can see the entire map. This chapter introduces vision and a way to increase the player's knowledge of the map as they explore more of it. Using ECS systems, you'll give the same restriction to monsters—if they don't know where you are, they can't chase you.

Chapter 11, More Interesting Dungeons, on page ?, introduces new map generation techniques. This chapter also teaches you the more advanced Rust topic of *traits*, and how they can provide interchangeable functionality with a common code interface—particularly useful when working in teams.

Chapter 12, Map Themes, on page ?, adds new ways to render your maps, building upon the trait knowledge from the previous chapter. You can turn your dungeon into a forest or any other setting by changing the map's tile-set.

Chapter 13, Inventory and Power-Ups, on page ?, adds items, backpack management, and power-ups to your game.

Chapter 14, Deeper Dungeons, on page ?, replaces your single-level dungeon with a sprawling, deep dungeon. It walks you through using tables to provide increasing difficulty as the player progresses.

Chapter 15, Combat Systems and Loot, on page ?, adds "loot tables" and better items as you progress through the dungeon. You'll find more interesting swords and adjust combat to reflect their powers. You'll also balance increasing difficulty with better loot, and learn about the risk/reward curve.

Finally, in Chapter 16, Final Steps and Finishing Touches, on page ?, you'll learn to package your game for distribution. You'll also find suggestions for making the game your own, and taking your next steps in game development.

What's Not in This Book

This book emphasizes learning theory by following practical examples and explaining the theory behind the technique you learned. This book isn't an in-depth guide to every facet of the Rust language; instead, this book guides you to sources for this information when it introduces a concept.

Likely, this book doesn't describe the game idea you've always wanted to make. That's OK. This book teaches *concepts* that are useful whether you want to make the next great online board game or a shooter. These concepts are readily transferable to other engines, including *Unity*, *Godot*, *Unreal*, and *Amethyst*. By the end of the book, you'll be much better equipped to start work on the game you've always wanted to make.

How to Read This Book

If you're new to Rust development, you'll want to work through the book and associated examples in order. If you're experienced with Rust, you may want to skim over the introductory sections and dive straight into game development. If you're already a skilled game developer, you can still learn a lot about Rust and data-oriented design from this book.

Tutorials are as much about the journey as they are the destination. Working through tutorials should give you ideas for what *you* want to make next. Start keeping notes about what you'd like to create and how the game development concepts in this book can help you. When you finish the book, you'll be ready to start making your own games.

Conventions Used in This Book

The code accompanying this book is provided as a Rust *Workspace*. This combines several projects into one. The code is divided into directories as follows:

```
root
   /chapter_name
       /example_name
          /src --- the source code for this example
          /resources --- files that accompany this example
          Cargo.toml --- a file telling Rust's Cargo system how to build/run
            the example.
```

```
 /src --- a simple program source reminding you that you probably meant to
          navigate to an example and run that---not the workspace.
Cargo.toml --- a file telling Rust's Cargo system that the other
          projects are part of the workspace.
```

You can run code examples by navigating to the chapter_name/example_name directory and typing cargo run.

As you progress through chapters, the example code is referenced with the location in the book's source code. The linked code sometimes refers to a different project within the chapter's code directory. It's designed to provide you with working examples for each stage of incremental development and to help you follow along. For example, you might see file snippets referencing code/FirstStepsWithRust/hello_yourname. Later in the same chapter, you might see code in code/FirstStepsWithRust/treehouse_guestlist_trim/.

Online Resources

Here are some online resources that can help you:

- *Rust by Example* provides a good, example-driven introduction to the Rust Language.[1]
- *The Rust Programming Language [KN19]* supplies in-depth concepts and tutorials to learn the finer details of Rust. It is also available online.[2]
- *The Rust Standard Library* documentation provides detailed descriptions of everything found in Rust's std library. It is a great reference when you can't remember how something works.[3]
- Reddit hosts several useful communities. /r/rust and /r/rust_gamedev are excellent resources. /r/roguelikedev is very helpful for games similar to the dungeon crawler in this book. These subreddits also include links to Discord forums full of people eager to help you.

Wrap-Up

Whether you're focused on learning Rust or want to dabble in game development, this book can help. It's exciting to finish a new game—or section of a game, run it, and enjoy the rush of seeing your creation in action. Let's start by setting up your Rust development environment and jumping straight into your first Rust code.

1. https://doc.rust-lang.org/rust-by-example/
2. https://doc.rust-lang.org/book/
3. https://doc.rust-lang.org/std/index.html

Herbert says:

My Game Development Journey

I was very lucky growing up. My father was teaching computing skills and introduced me to many of the computers of the day. One fateful day he brought home a BBC Micro, Model B. It featured 32k of RAM, color graphics, and programs that loaded from audio cassettes—it felt like an incredible machine. My parents provided an ever-expanding library of games to play, from puzzle games like *Repton* to clones of arcade machine games. It didn't take long for me to want to make my own games, and my father patiently walked me through learning *BASIC*. My early games were objectively terrible—and that didn't matter at all. I'd made something and discovered the thrill of showing it to my friends.

My early BASIC games set me on a fun path. I learned *Pascal* and later *C* and *C++*. I learned to make games for Windows and later Linux. I collaborated on a few group projects, and eventually found work writing business and networking software—much of which benefits from my experience creating games. When I first tried Rust, it felt like the perfect fit—and I haven't looked back.

This book teaches Rust and game development. More than anything, I hope that it inspires you to go forth and make something fun.