What was built

I set out to create a small text adventure game in Rust.
In its current state, what I built is a small text adventure game with less features than I wanted, but with a couple features that I honestly didn't expect I'd be able to implement.

Current features include:

- Automatic saving when quitting the game.
- Ability to load saved game from main menu.
- A map with 11 total rooms.
- The ability to move the player from room to room with the 'go' command.
- The ability to interact with objects in the rooms with the 'look' command.
- Non-player characters (2 total) that the player can interact with using the 'talk' command.
- Locked areas that must be interacted with in order to access, using the 'look' command.
- Locked areas that can only be accessed after collecting a certain item or talking to a certain NPC.
- Relatively fleshed-out room descriptions, item descriptions, and NPC dialogue.
- NPCs give quests, accept quest items, and then end quests after they have received them.
- The ability for the player to reach an area that causes the game to end. Essentially, a way to win the game that makes some sense.

The map implementation is extremely rudimentary, in that it just consists of a number that corresponds to a set of information returned from functions in the room module. There is no Room struct or Map struct. There is no Player struct either.
The game state, however, is abstracted out to a struct that carries basic player info and various flags that denote the actions or accomplishments of the player.
NPCs also have their own struct and their own status flags, etc.
main is given a Vec of NPCs to work with, which may or may not make things more usable as the number of NPCs is scaled up.
Some helper functions that are used in multiple modules are defined in their own module: helpers.
Most of the text descriptions, dialogue, etc. was extracted out to and is read from external files located in the data folder.

How it worked

I first tried to make a Room struct and a Player struct to manage the basic functionality of movement and player status, but (predictably) this quickly turned into a giant fight with the borrow checker. I decided to scrap the OOP idea after toiling with it for a few hours and go with a more minimalistic (and less usable) approach. This worked out okay, but as I learned more about Rust and got more practice, I decided to use structs for most everything else with minimal hassle, so I think that I would use a more "structured" approach if I were to start a similar project now.
One thing that I thought was a bit ambitious but decided to try anyways, was the addition of NPCs. I wasn't sure I'd be able to pull it off in the time frame, but I did (with relative success). That being said, I spent the majority of my development time working on NPC implementation and neglected some other parts that were a little more important, like testing.
I have a knack for being oblivious to the priorities in these kinds of projects, and it showed a bit in this

project.
I only was able to implement a handful of unit tests at the last minute, but given a bit more time, I feel confident I could get a decent amount of the code covered with unit testing.
The tendency is to spend the majority of one's time on playtesting, since it feels more intuitive and useful in some respects, but unit tests definitely have their place in a project like this, regardless.

What doesn't work

There isn't really anything that doesn't work, although there is a lot more functionality that is yet to be added.
The only real issue is that the user must cargo run from inside the src/ directory to play the game, since the file-paths used in the code are all relative. This is an issue that I was not able to spend enough time on to solve, but is surely solvable.

Lessons learned

I learned a lot about Rust, predictably. The borrow checker and I got very well acquainted. I feel like I understand a lot more about the how a rust program is set up (module structures, etc.) than I did before. Doc comments were a fun thing to figure out, and are really quite nice. And of course, I got another lesson in managing time and priorities.