

Products

Multiple reference readers and one reference writer in Rust

Ask Question

Asked 1 year, 8 months ago Active 1 year, 8 months ago Viewed 222 times



1



I'm currently working on a small Rust game to start with the language and have basically the following code (I only wrote a minimal example here):

```
struct Player<a> {
  pub ship: &'a Ship<'a>,
impl<'a> Player<'a> {
  pub fin run(&mut self) {
     // Does some computing with self.ship.x/self.ship.y
struct Ship <a> {
  pub players: Vec<Player<a>>>,
   pub x: f64,
  pub y: f64,
impl<'a> Ship<'a> {
  pub fin add_player(&mut self, player: Player<'a>) {
    self.players.push(player);
fn main() {
   let mut ship = Ship {
     players: vec![],
     x: 0.0,
  y: 0.0,
};
  // At some point create a player for the ship let player = Player { ship: &ship };
  ship.add_player(player); // <- Forbidden
```

The most important thing here is that all Player's have access to the ship they belong to with an immutable reference, so that they easily have access to the position (x/y) of their ship (which changes over time, as the game runs). However, this code doesn't compile:

I understand that player is borrowing ship as immutable and that I'm still trying to modify ship after the borrow occurs, but I can't find what is the right smart pointer or wrapper I should use for this kind of case? Would you use a RwLock, or a RefCell, or something else?

rust reference borrow-checker ownership

```
Improve this question
Follow
edited Apr 5 '20 at 18:36

asked Apr 5 '20 at 14:07

Add acon mark

1 Answer
```

Your privacy

By clicking "Acceptable cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.





You have the right idea in that you would probably need to use RefCell, or RwLock or maybe even Re. However, these concepts are more advanced and I do not recommend trying to use them when you're just starting to learn the language. Instead, I would remove the Ship reference from the Player struct and just have the Ship contain references to Players.

If you haven't already, I highly recommend the official rust book, its a great introduction to the language with great examples!

Share

Improve this answer

Follow

answered Apr 5 '20 at 22:36



Thanks, will take a look! Still curious about this issue... Are you able to confirm that something like Vec<RefCell<Player<a>>>> would be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and there's no way to make it will be the solution to my problem and the my problework just by using simple references?

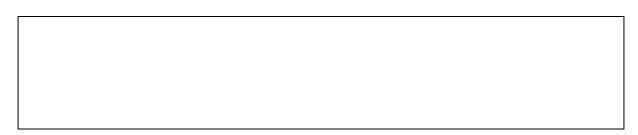
- Yannick Utard

Apr 6 '20 at 8:15

Yes, I believe Vec-RefCell-Player<a>>>> that would work. Since player is keeping a borrowed reference of the ship, there's no way to mutate the ship while the player has it, thus, I do not believe there is any way to make it work using simple references

Apr 7 '20 at 20:20

Your Answer



Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged nust reference borrow-checker ownership or ask your own question.

The Overflow Blog

Sequencing your DNA with a USB dongle and open source code

Don't push that button: Exploring the software that flies SpaceX rockets and..

Featured on Meta

Q

Providing a JavaScript API for userscripts

Congratulations to the 59 sites that just left Beta

Related

What are the differences between a pointer variable and a reference variable in C++?

cannot move out of borrowed content when unwrapping a member variable in a &mut self method

Why can't I store a value and a reference to that value in the same struct?

Borrow errors for multiple borrows

Conflicting lifetime requirements when mapping mutable references with self

Prevent cannot borrow '*self' as immutable because it is also borrowed as mutable when accessing disjoint fields in struct?

Mutating fields of Rc Refcell depending on its other internal fields

Hot Network Questions

¥ Repeating slices of an array incrementally

- iii Given many questions as to whether Jesus was born on 25 December or not, I ask if the ambiguity in scripture is meant to teach us something? $\underline{\hspace{0.1cm}}$ Question on connections in general relativity and particle physics Mine sweeping game for the terminal How do I get the http endpoint to work for cardano-wallet? more hot questions

Question feed

STACK OVERFLOW Questions Jobs Developer Jobs Directory Salary Calculator Help Mobile

PRODUCTS

Teams Talent Advertising Enterprise

COMPANY

About
Press
Work Here
Legal
Privacy Policy
Terms of Service
Contact Us
Cookie Settings
Cookie Policy

STACK EXCHANGE NETWORK

Technology Culture & recreation Life & arts Science Professional Business API Data

Blog Facebook Twitter LinkedIn Instagram

site design / logo © 2021 Stack Exchange Inc; user contributions licensed under cc by-sa. rev 2021.12.22.41046