C++ Club UK Meeting 147

Gleb Dolgich

2022-04-07

Contents

C++ and Rust interoperability 2
Minimum viable declarative GUI in C++ 3
Twitter 3

C++ and Rust interoperability

An article was published on the Tetrane blog describing the current state of Rust and C++ interoperability. The article explains all the available options in detail, including code snippets, but for a short summary let's read a comment on the Reddit thread by the original poster:

> The post proposes 3 approaches based on 3 available libraries in the Rust ecosystem:
>
> - **bindgen**: Start from the C or C++ headers of a C/C++ library and generate Rust code that exposes functions able to call the C/C++ library. Then you can just link with this library (statically or dynamically) and call its functions! It is automatic, but it doesn't attempt to reconcile the differences of concepts between C++ and Rust, and more importantly, it doesn't attempt to translate what C++ and Rust have in common (iterators, vectors, `string`, `unique_ptr`, `shared_ptr`, ...), so it is best suited for very "C-like" libraries.
> - **cpp** uses Rust's macro system to let you write C++ inline inside of your Rust. The C++ snippets are then compiled by a C++ compiler, and the Rust code to call them using the C ABI is generated. Since the C++ snippets are C++, you can directly call other C++ libs from the C++ snippets. However the boundary between C++ and Rust remains somewhat low-level with this solution (it has native understanding of `unique_ptr`s but that's pretty much it).
> - **cxx**: uses Rust's macro system to let you declare a special Rust module containing items (types, functions) to be either shared (understood by both C++ and Rust, and passed by value between the languages) or opaquely exposed from one language to the other (you'll need to manipulate the type behind a pointer when on the other language). This approach is nice because it pre-binds for you some C++/Rust standard types (vectors, strings) and concept (exceptions and Rust's `Result` type).
>
> At the basic levels, all three libraries are built upon the C ABI/API, since it is the common language that both Rust and C++ understand. In cxx however you don't really see the use of the basic C API since some higher-level concepts are translated between C++ and Rust.

I read that Microsoft is exploring Rust for some of their code bases, wonder what they'll use if they need C++ interop.

Minimum viable declarative GUI in C++

Jean-Michaël Celerier wrote an article that introduces a minimal declarative C++ GUI library. Like, really minimal, where declaring a struct is enough to define a user interface. Later this declaration is included in another 'magical' file which produces the declared UI. The resulting interface can be rendered by Qt via QML or another backend, like Nuklear (a C-based immediate mode UI engine).

An example UI declaration is on GitHub.

In the Reddit thread, people are generally impressed, but not when they discover all the macros the author had to add to improve the syntax.

Also, the code is under GPLv3, so be careful not to remember any of it or you'll have to open-source your brain.

Twitter

Viktor Zverovich (@vzverovich):

Patricia Aas (@pati_gallardo):

Patricia Aas again (@pati_gallardo):