

Rust/C++ Interop

Boulder Rust Meetup

February 7, 2024

Tyler Weaver
Staff Software Engineer
maybe@tylerjw.dev

- Regular C++ Programmer
- Rust Cult Member
- Open-source Robotcist
- Wrote a Rust Library with C++ Bindings

- No AI generation tools were used
- Slides and more at tylerjw.dev

What Are We Going To Cover



- Social Objections to Rust
- Details of Interop
- Examples of Useful Patterns
- Code Generation Tools

Quality of your project has more to do with the people that build it than the tools selected. It is fine and good for the people to like their tools.

- C++ code that exists has value
- A little Rust is better than no Rust

Golden Gate Bridge



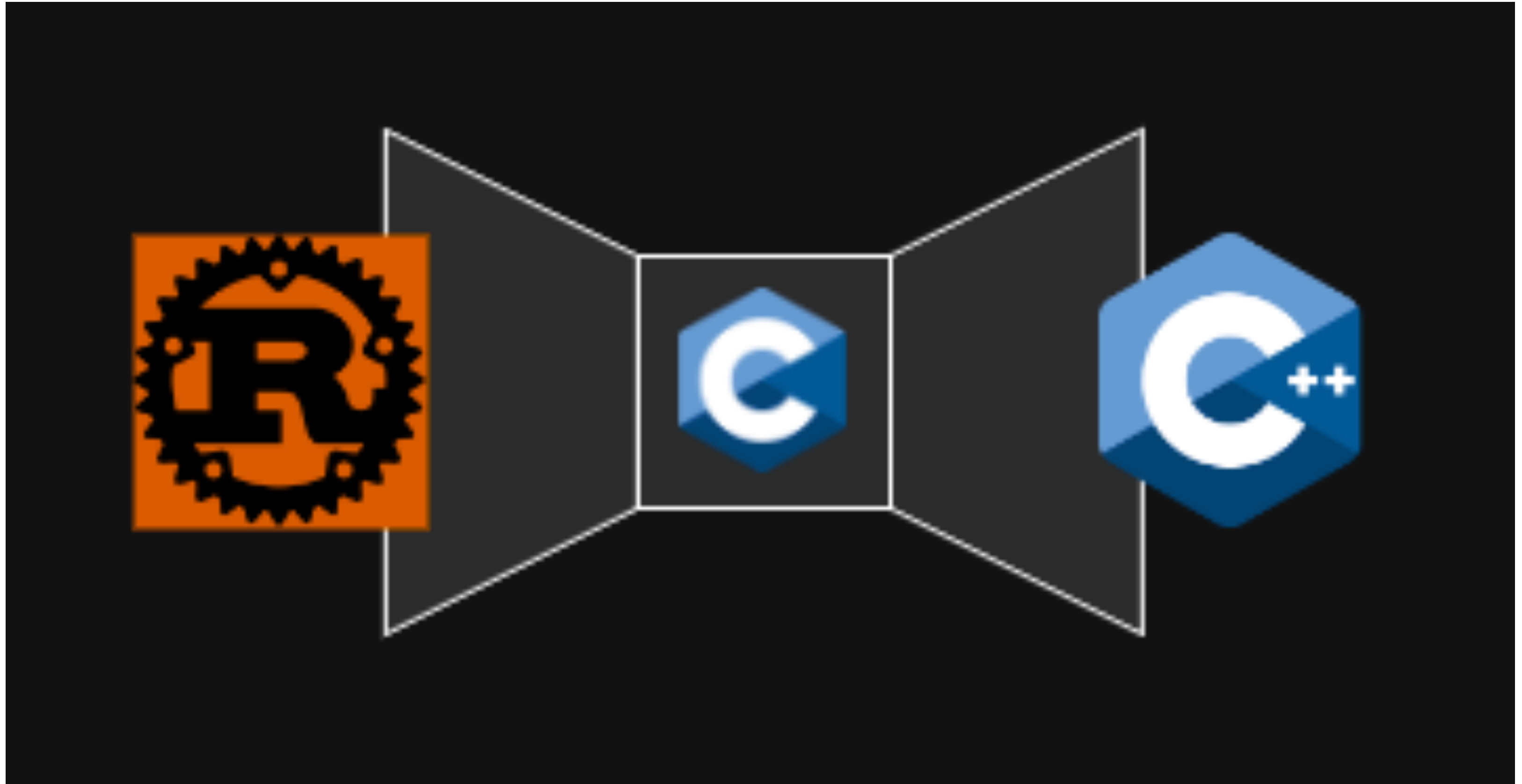
Code Generators

- cxx – Safe interop between Rust and C++
- bindgen – generate Rust FFI to C/C++ headers
- cbindgen – generate C headers for Rust FFI

Why Not

- Eigen C++ types \Leftrightarrow Nalgebra Rust types

Hourglass Language Bridge



Project Layout



```
├── Cargo.toml
├── crates
│   ├── robot_joint
│   │   ├── Cargo.toml
│   │   └── src
│   │       └── lib.rs
│   └── robot_joint-cpp
│       ├── Cargo.toml
│       ├── CMakeLists.txt
│       ├── cmake
│       │   └── robot_jointConfig.cmake.in
│       ├── include
│       │   └── robot_joint.hpp
│       └── src
│           ├── lib.cpp
│           └── lib.rs
└── README.md
```

Zakim Bridge



```
pub struct Joint {  
    name: String,  
    parent_link_to_joint_origin: Isometry3<f64>,  
}  
  
impl Joint {  
    pub fn new() -> Self;  
}
```

```
use robot_joint::Joint;

#[no_mangle]
extern "C" fn robot_joint_new() -> *mut Joint {
    Box::into_raw(Box::new(Joint::new()))
}

#[no_mangle]
extern "C" fn robot_joint_free(joint: *mut Joint) {
    unsafe {
        drop(Box::from_raw(joint));
    }
}
```



```
struct RustJoint;

class Joint {
public:
    Joint();
    ~Joint();

    // Disable copy as we cannot safely copy opaque pointers to rust objects.
    Joint(Joint& other) = delete;
    Joint& operator=(Joint& other) = delete;

    // Explicit move.
    Joint(Joint&& other);
    Joint& operator=(Joint&& other);

private:
    RustJoint* joint_ = nullptr;
};
```

robot_joint-cpp/src/lib.cpp



```
#include "robot_joint.hpp"

extern "C" {
extern RustJoint* robot_joint_new();
extern void robot_joint_free(RustJoint*);
}
```

```
Joint::Joint() : joint_(robot_joint_new()) {}

Joint::~~Joint() {
    if (joint_ != nullptr) {
        robot_joint_free(joint_);
    }
}

Joint::Joint(Joint&& other) : joint_(other.joint_) {
    other.joint_ = nullptr;
}

Joint& Joint::operator=(Joint&& other) {
    joint_ = other.joint_;
    other.joint_ = nullptr;
    return *this;
}
```


See My Blog for a link to CMake example
tylerjw.dev

Fremont Bridge



robot_joint/src/lib.rs

```
impl Joint {  
    pub fn calculate_transform(&self, variables: &[f64]) -> Isometry3<f64>;  
}
```

robot_joint-cpp/include/robot_joint.hpp

```
class Joint {  
    public:  
        Eigen::Isometry3d calculate_transform(const Eigen::VectorXd& variables);  
};
```

```
#[repr(C)]
struct Mat4d {
    data: [c_double; 16],
}

#[no_mangle]
extern "C" fn robot_joint_calculate_transform(
    joint: *const Joint,
    variables: *const c_double,
    size: c_uint,
) -> Mat4d {
    unsafe {
        let joint = joint.as_ref().expect("Invalid pointer to Joint");
        let variables = std::slice::from_raw_parts(variables, size as usize);
        let transform = joint.calculate_transform(variables);
        Mat4d {
            data: transform.to_matrix().as_slice().try_into().unwrap(),
        }
    }
}
```

```
struct Mat4d {
    double data[16];
};

extern "C" {
extern struct Mat4d robot_joint_calculate_transform(
    const RustJoint*, const double*, unsigned int);
}

Eigen::Isometry3d Joint::calculate_transform(const Eigen::VectorXd& variables)
{
    const auto rust_isometry = robot_joint_calculate_transform(
        joint_, variables.data(), variables.size());
    Eigen::Isometry3d transform;
    transform.matrix() = Eigen::Map<Eigen::Matrix4d>(rust_isometry.data);
    return transform;
}
```


Red Cliff Bridge



**Rust / C++ Interop is Straightforward
Don't Listen to the Naysayers**

Kyle's OptIk
github.com/kylc/optik