Bohemia Automation

Cross-platform desktop apps in Rust and Qt

Serhij Symonenko / Bohemia Automation

Bohemia Automation

Why desktop apps in 2023?

- No HTTP: industrial devices, databases, clients for 3rd party web services etc.
- Sometimes desktop apps are just faster than web ones
- Sometimes apps require more access to the local machine than WASM can provide
- No limitations - use any crates, use multiple threads, use async runtimes (tokio)

Bohemia Automation

Why Qt?

- Two big grown-up players: **Qt** and **GTK**
- Qt usually looks neater and more Enterprise-ready (but take into account the license)
- Qt has got commercial support, including lots of 3rd party consulting companies
- Qt has got Qt Designer and QML
- Qt 5 or 6? Check KDE: https://iskdeusingqt6.org

Bohemia Automation

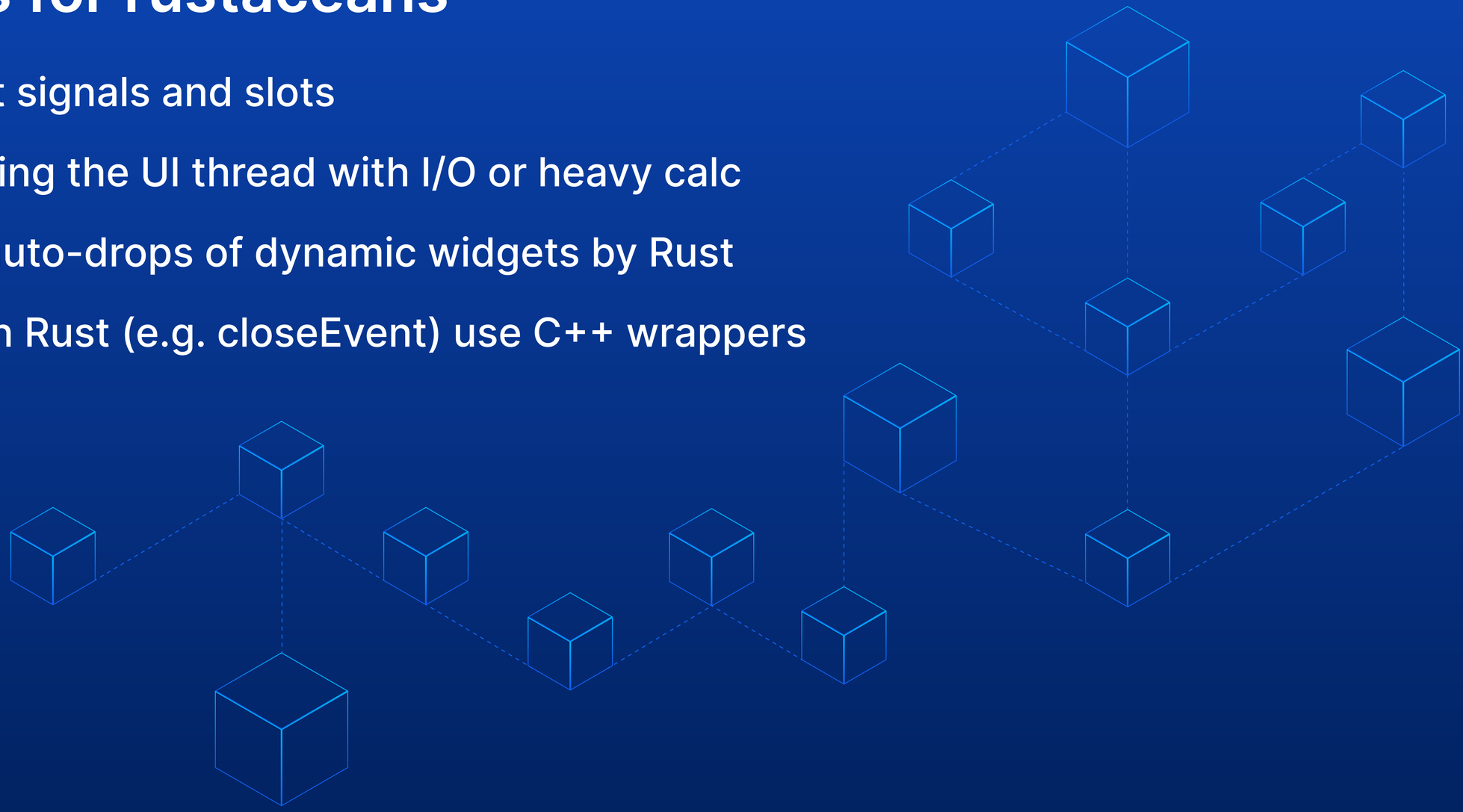
Qt in Rust

- **Ritual**
  https://rust-qt.github.io
- **qmetaobject**
  https://github.com/woboq/qmetaobject-rs
- **cxx-qt**
  https://www.kdab.com/cxx-qt/

Bohemia Automation

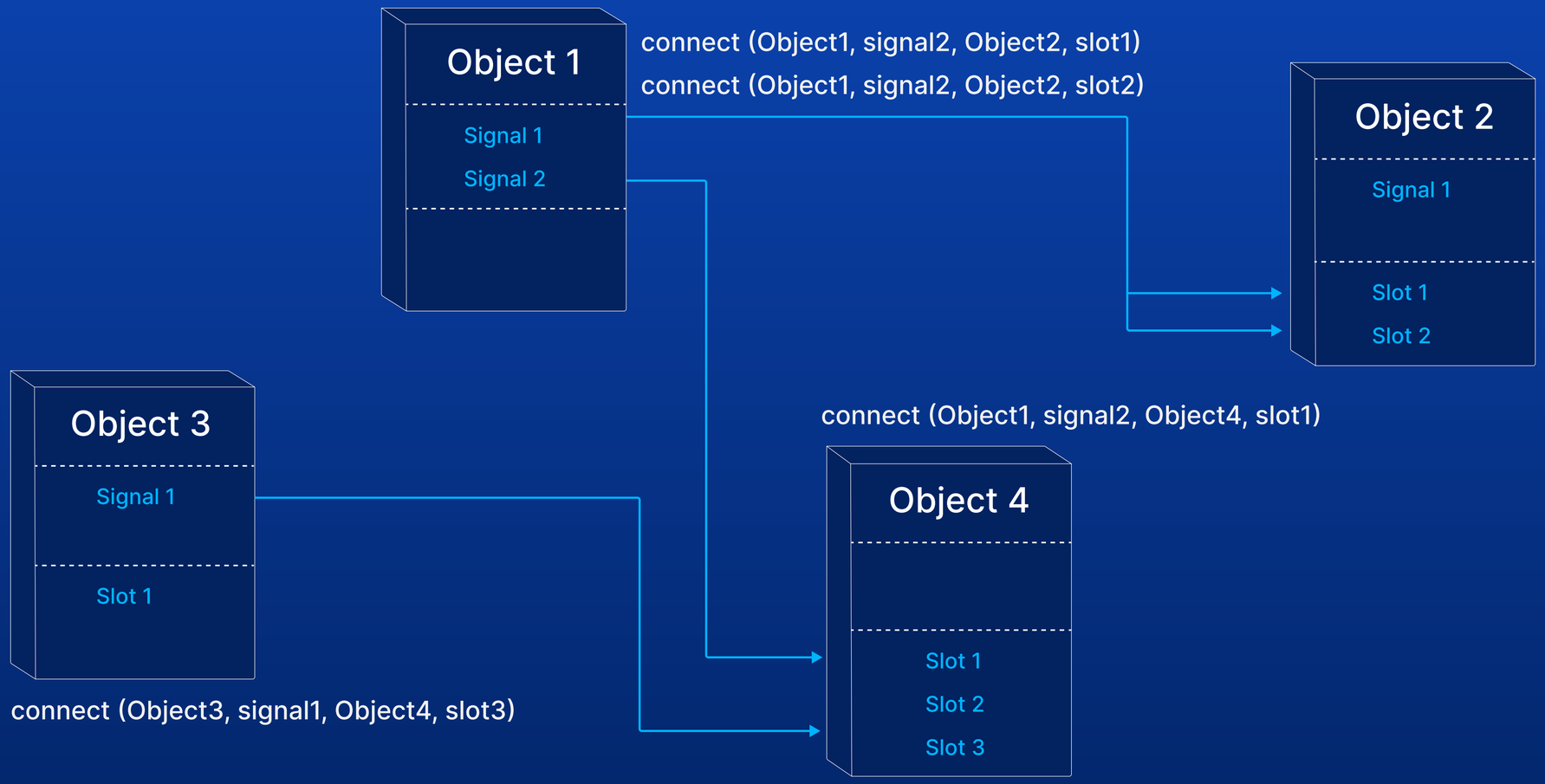
Qt basics for rustaceans

- Learn about signals and slots
- Avoid blocking the UI thread with I/O or heavy calc
- Beware of auto-drops of dynamic widgets by Rust
- No events in Rust (e.g. closeEvent) use C++ wrappers



Bohemia Automation

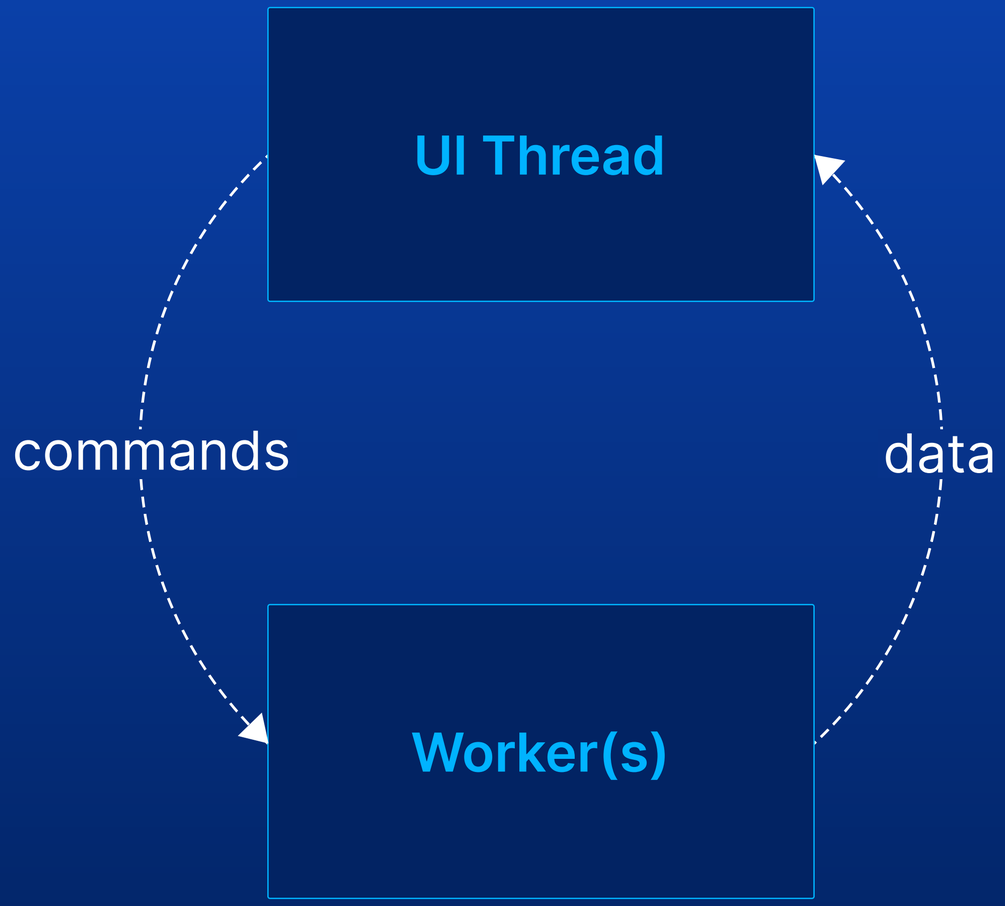
Signals and Slots



Note: Signal objects are not completely thread-safe but signal.emit() function is an exception

Bohemia Automation

Organizing data flow



Bohemia Automation

Why do people still use Ritual

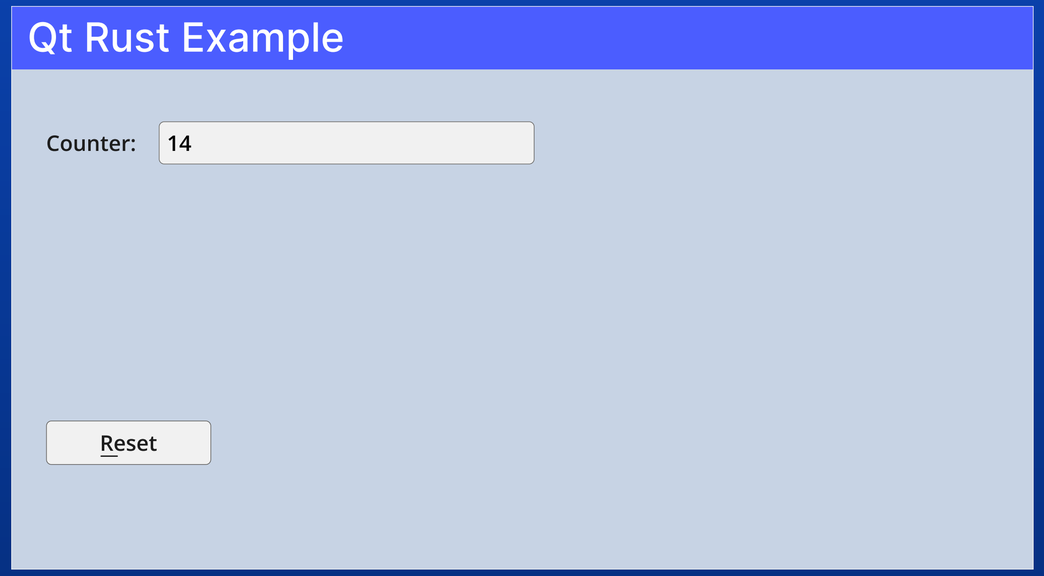
- The oldest and the most popular Rust-Qt project with no serious issues
- The most complete bindings for Qt5
- Supports Qt Designer-generated forms
- Supports dynamic widget creation
- Minimal Rust overhead
  - Qt has got a good documentation and issue discussions. Majority of that can be used with Ritual almost as-is
  - In case of project death and zero alternatives,
    UI logic can be ported to pure C++ with low costs

When Ritual should be avoided

- You prefer to keep away from unsafe blocks in your code
- You use QML only
- You do not care about Qt knowledgebase and prefer a Rust way
- You need Qt6 support

Bohemia Automation

Program example (Ritual)



- A background worker increases the counter value every second
- When the value is increased, a data event is being sent to the UI thread
- UI reset button can send reset commands to the background worker to reset the counter
- UI and the worker thread talk via standard mpsc channels
- The worker thread uses Qt Signal with no arguments to notify UI thread about new events in the data channel

The main function

```
fn main() {
    QApplication::init(|_| {
        // command channel
        let (command_tx, command_rx) = mpsc::sync_channel::<Command>(64);
        // data channel
        let (data_tx, data_rx) = mpsc::sync_channel::<Data>(64);
        // construct UI
        let ui = Ui::new(command_tx.clone(), data_rx);
        unsafe {
            // data signal
            let data_signal = unsafe_send_sync::UnsafeSend::new(SignalNoArgs::new());
            // connect data signal with UI handle_data slot method
            data_signal.connect(&ui.slot_handle_data());
            // run the background worker
            thread::spawn(move || {
                worker(command_rx, data_tx, data_signal);
            });
            // display the UI
            ui.show();
            // exec the Qt application
            let result: i32 = QApplication::exec();
            // optionally terminate the background worker
            command_tx.send(Command::Quit).unwrap();
            result
        }
    })
}
```

UI and Main window

```rust
// main window
#[ui_form("../ui/main.ui")]
struct Main {
    widget: QBox<QWidget>,
    counter: QPtr<QLineEdit>,
    btn_reset: QPtr<QPushButton>,
}

// UI
struct Ui {
    window: Main,
    command_tx: mpsc::SyncSender<Command>,
    data_rx:  ::Receiver<Data>,
}

// reqired to transform Rust functions into slots
impl cpp_core::StaticUpcast<QObject> for Ui {
    unsafe fn static_upcast(ptr: Ptr<Self>) -> Ptr<QObject> {
        ptr.window.widget.as_ptr().static_upcast()
    }
}
```

Bohemia Automation

UI Implementation

```
impl Ui {
    // Rc is required to transform Rust functions into slotsimpl Ui {
    fn new(command_tx: mpsc::SyncSender<Command>,
                                    data_rx: mpsc::Receiver<Data>) -> Rc<Self> {
        unsafe {
            let window = Main::load();
            let ui = Rc::new(Ui { window, command_tx, data_rx, });
            ui.window.btn_reset.clicked().connect(&ui.slot_handle_btn_reset());
            //let ctx = ui.command_tx.clone(); // an alternative slot implementation
             //ui.window.btn_reset.clicked().connect(&SlotNoArgs::new(&ui.window.widget, move || {
             //    let _ = ctx.send(Command::Reset);
             //}));
            ui
        }
    }
    #[slot(SlotNoArgs)]
    unsafe fn handle_btn_reset(self: &Rc<Self>) {
            let _ = self.command_tx.send(Command::Reset);
    }
    unsafe fn show(self: &Rc<Self>) {
            self.window.widget.show();
    }
}
```

Bohemia Automation

Worker and Events

```rust
// commands to the background worker
enum Command {
  Reset,
  Quit,
    }
// data from the background worker (alternative: slots, but objects must be Qt-ized)
 enum Data {
  Counter(u64),
    }
// background worker
 fn worker(
  command_rx: mpsc::Receiver<Command>,
  data_tx: mpsc::SyncSender<Data>,
  data_signal: unsafe_send_sync::UnsafeSend<QBox<SignalNoArgs>>,
  ) {
  let mut counter = 0;
  loop {
    while let Ok(command) = command_rx.try_recv() {
     match command {
     Command::Reset => counter = 0,
     Command::Quit => break,
     }
    }
    if data_tx.send(Data::Counter(counter)).is_ok() {
     unsafe { data_signal.emit(); }
    }
    thread::sleep(Duration::from_secs(1));
    counter += 1;
  }
}
```

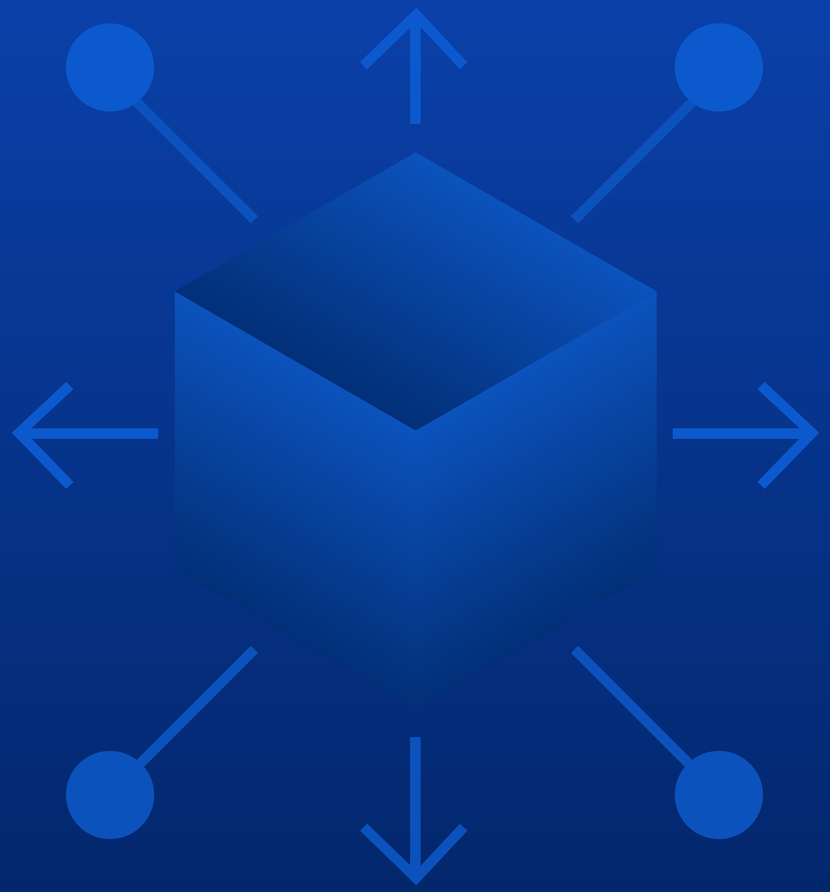
Handling data events in UI

```
impl Ui {
   // .................
   // .................
   // add the following method to UI
   #[slot(SlotNoArgs)]
   unsafe fn handle_data(self: &Rc<Self>) {
      while let Ok(data) = self.data_rx.try_recv() {
   match data {
      Data::Counter(v)= >{
        self.window.counter.set_text(&qs(v.to_string()));
      }
     }
    }
  }
}
```

Bohemia Automation

How to distribute apps

- cargo bundle (Linux/OSX)
  https://github.com/burtonageo/cargo-bundle
- cargo wix (Windows)
  https://github.com/volks73/cargo-wix



Thank you for watching!

The presentation and the source code:

https://github.com/divi255/qtx

Production app example:

https://info.bma.ai/en/actual/eva4/ecmui/

Bohemia Automation

Cross-platform desktop apps in Rust and Qt

Serhij Symonenko / Bohemia Automation