QT SIGNALS AND THE COROUTINES TS

JEFF TRULL

27 SEPTEMBER 2018

QT SIGNALS

A KIND OF GENERALIZED EVENT

- like "timer expired", "button pressed", "packet received", etc.
- can have parameters

YOU CONNECT THEM TO "SLOTS"

- These can be member functions, lambdas, etc. - a kind of handler
- You can take further action, update variables, etc.

SEQUENCING LOGIC IS A PAIN

- Often things have to happen in a specific order
- Respond to several signals in a row
- Change action based on the sequence
- state must be introduced

EXAMPLE: DRAWING A LINE

Sequence of states:

- No points entered
- First point entered

EXAMPLE: DRAWING A LINE

```
bool got_first_point{false};
QPointF first_point;

QObject::connect(&cr, &ColorRect::click,
                 [&](QPointF p) {
                     if (got_first_point) {
                         // draw
                         cr.setLine(first_point, p);
                         got_first_point = false;
                     } else {
                         first_point = p;
                         got_first_point = true;
                     }
                 });
```

WOULD BE CLEARER TO SAY

- get first point
- get second point
- draw line

CO_AWAIT ON SIGNALS

This is a classic use case for coroutines.

DESIRED CODE

```
QPointF first_point = co_await make_awaitable_signal(&cr, &ColorRect::
QPointF second_point = co_await make_awaitable_signal(&cr, &ColorRect:
cr.setLine(first_point, second_point);
```

USING THE COROUTINES TS

Requirements

We need:

- a "promise type" embodying a handle for the creating code
- an "awaitable" type that configures suspending and resuming, in our case:
  - Connecting and disconnecting from the Signal
  - Marshalling the `co_await` result, if any

THE CODE

METAPROGRAMMING STUFF

Goals

- co_await should produce void, T, or std::tuple<> depending on signature
  - signal(A, B, C) -> std::tuple<A, B, C>
  - signal(A) -> A
  - signal() -> void
- Thanks to `#metaprogramming` on Slack, and particularly Arthur O'Dwyer

The Awaitable

(non-void case)

```
template<typename Object, typename... Args>
awaitable_signal(Object* src, void (Object::*method)(Args...),
                 std::experimental::coroutine_handle<>& coro_handle) {
    // create slot and connect
    signal_conn_ =
        QObject::connect(src, method,
                         make_slot<Result, Args...>()(signal_conn_,
                                                      derived()->signa
                                                      coro_handle));
}
```

The Awaitable

(single parameter type T case)

```
template<typename Arg>
struct make_slot<Arg, Arg> {
    auto operator()(QMetaObject::Connection& signal_conn,
                    Arg& result,
                    std::experimental::coroutine_handle<>& coro_handle
        // disconnect signal, marshal result, resume awaiting code
        return [&signal_conn, &coro_handle, &result]
            (Arg a) {
            QObject::disconnect(signal_conn);
            result = a;
            coro_handle.resume();
        };
    }
};
```

DEMO

CODE

- Two coroutines running out of the Qt event loop
  - line drawer running off click events (single QPointF param)
  - background color changer running off timer (no params)
- Blog: https://git.io/fAhaQ

LET'S TRY IT