ACCU 2022

PROPERTIES - BINDINGS FOR MODERN C++

JAMES TURNER

Motivation

- At KDAB we use Qt
  - But not all the time ☺
- Qt has some nice pieces, though
- Declarative language for describing related properties

Signals, Properties

Signal-slot idiom

- Qt, libsigc++, boost::signals
- Independent objects can communicate
  - Type-safe signal parameters
  - Standard type conversion rules
- `emit` signal
  - All connected slots are notified
- Slot is any callable with `operator()`
  - Including lambdas

Arguments

- Signal has arbitrary arguments
  - Pass values when emit()-ing
- `connect` arguments can be specified
  - Enables trivial connections to member functions

Properties

- Nothing new under the sun ☺
- Template on arbitrary type
- `::get, ::set`
  - automatic conversion to T
  - Assignment from T
- Change notification via signals
  - Pre-change: old and new value
  - Post-change: new value
- Destruction signal

Code

```
class Widget {
public:
    Signal<> clicked;
};

class Controller {
public:
  void doPrint();
};

Widget w;
Controller c;
w.clicked.connect(&Controller::doPrint, c);
```

```
class Widget {
public:
    Property<int> height;
    Property<bool> visible;
};

w.height.valueChanged.connect([](int h) {
    std::cout << “Height is now:” << h;
});
```

Bindings

Primary vs secondary state

- Compute values based on others
- Ubiquitous in user interfaces
  - Common in many other cases
- Rate of change is moderate
- Cascade of changes can be considerable
- Correct propagation is a chore
- Let’s automate it ☺

Bindings

- Read-only property
- Computed from operations on other properties
  - C = A * B + 123
- `Area = width * height`
- `capsString = to_upper(stringProp)`
- `Visible = myVis && parent->isVisible()`
- Cascade change notifications

(More) Code

The Qt, OpenGL and C++ experts

What’s supported

- Standard arithmetic operators
- Various `std::` functions (abs, clamp)
- Pass a function to `makeBoundProperty`
- Wrap a free function in
  - `KDBINDINGS_DECLARE_FUNCTION`
  - Result depends on passed `Property<>` arguments

Evaluation

- Default evaluation mode is immediate
- Optionally, create explicit evaluator
  - Specify when creating bindings
  - Explicit `::evaluateAll()`

Practicalities

- Header-only
- MIT licensed
- On GitHub
  - https://github.com/KDAB/KDBindings
  - Patches welcome ☺
- Requires C++17

Future extensions

- Support for STL containers of values
- Multi-threaded binding evaluation
- Deferred connections
    - Especially for multi-threaded setups