Using the Qt Scene Graph from C++ with QSkinny

Who am I?

» working on Qt since 2008

» former QtNetwork maintainer

» @peha23 on Twitter

What is this talk about?

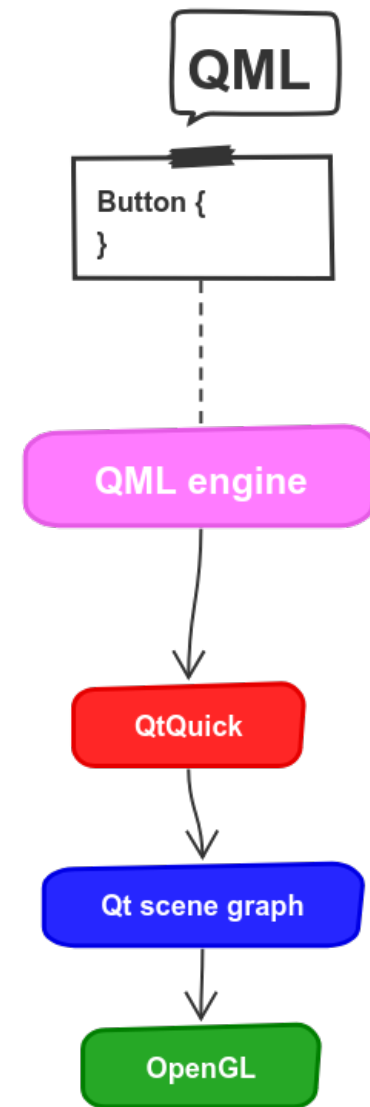Using the Qt graphic stack from C++

Agenda

1. QML under the hood

2. The QML / C++ boundary

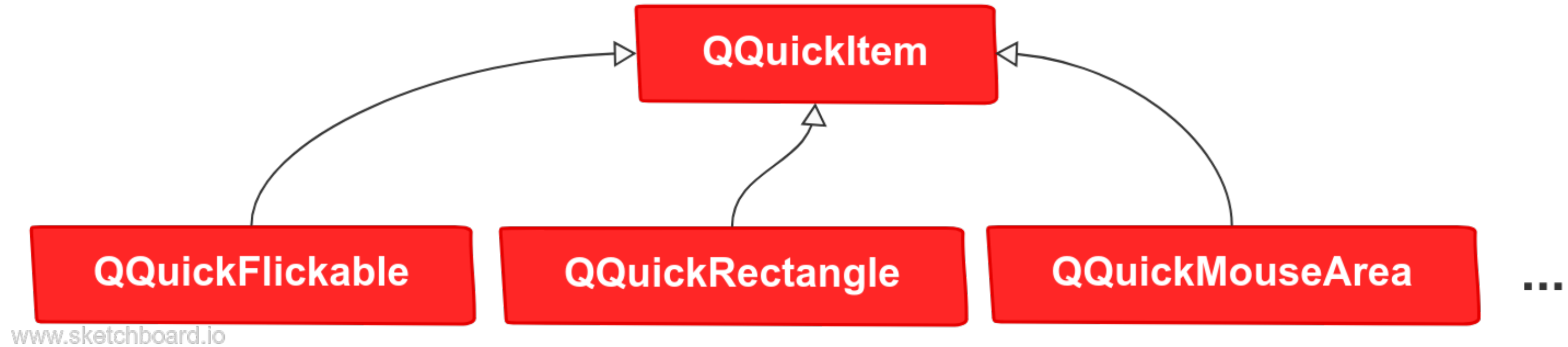3. QSkinny

4. Outlook

Agenda

1. <u>QML under the hood</u>

2. The QML / C++ boundary

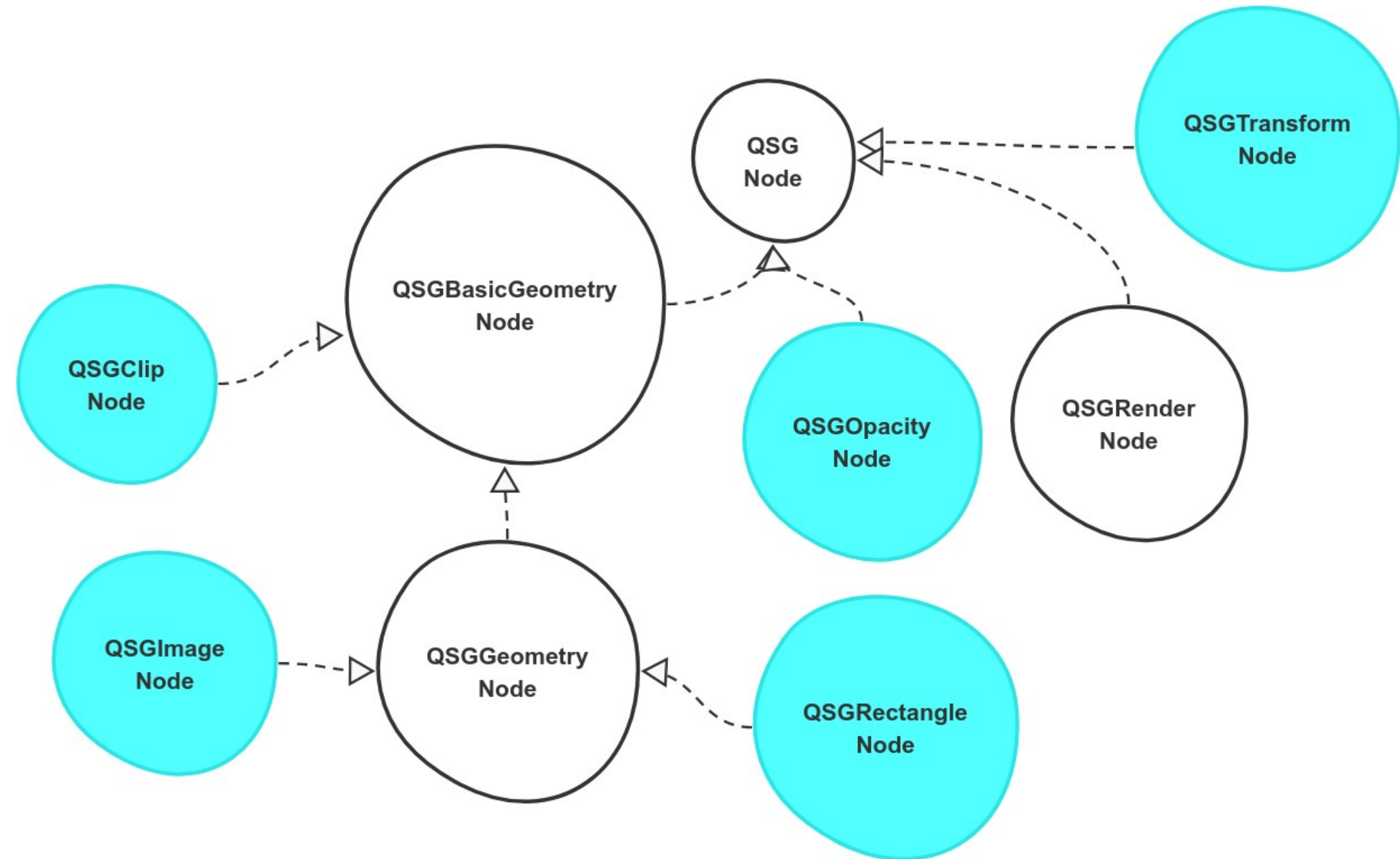3. QSkinny

4. Outlook

QML under the hood

Types of QtQuick items

www.sketchboard.io

https://github.com/uwerat/qskinny

Types of scene graph nodes

www.sketchboard.io

https://github.com/uwerat/qskinny

QML example

```
Rectangle {
    id: outterRectangle
    width: 200
    height: 200
    color: "red"
    opacity: 0.5

    Rectangle {
        id: innerRectangle
        width: 50
        height: 50
        clip: true
        anchors.bottom: parent.bottom
        anchors.right: parent.right
        color: "green"
    }
}
```

QQuickRootItem

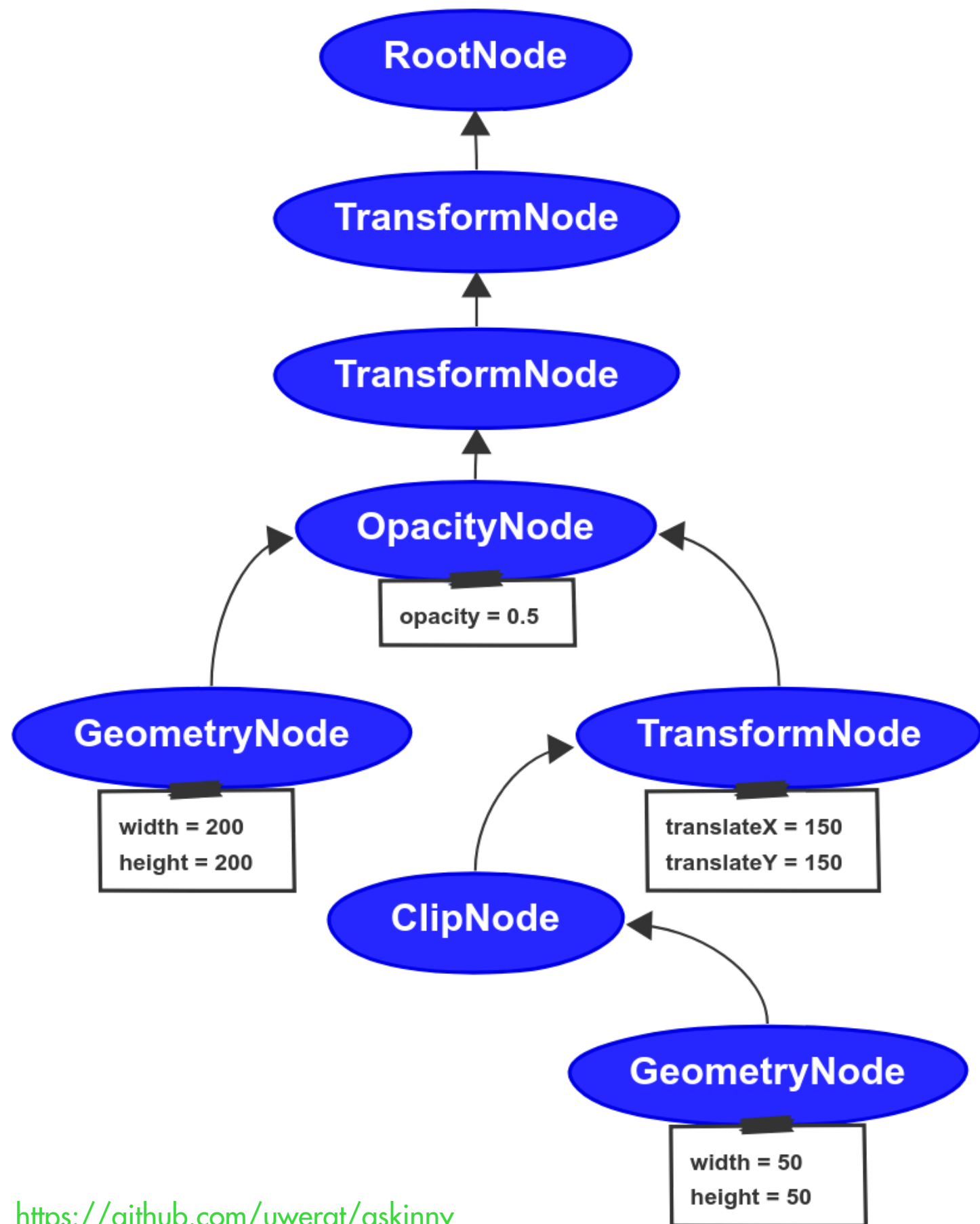is a QQuickItem

QQuickRectangle

is a QQuickItem

qreal x = 0
qreal y = 0
qreal width = 200
qreal height = 200
qreal opacity = 0.5

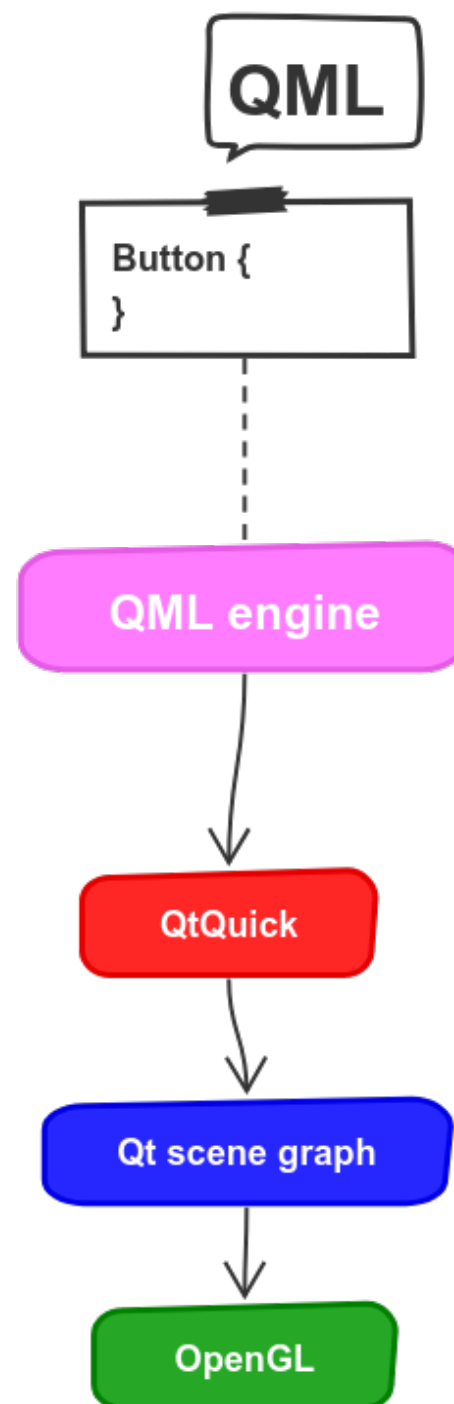**QQuickRectangle**

is a QQuickItem

qreal x = 150
qreal y = 150
qreal width = 50
qreal height = 50

RootNode

TransformNode

TransformNode

OpacityNode

opacity = 0.5

GeometryNode

width = 200
height = 200

TransformNode

translateX = 150
translateY = 150

ClipNode

GeometryNode

width = 50
height = 50

https://github.com/uwerat/qskinny

Agenda

1. QML under the hood

2. <u>The QML / C++ boundary</u>

3. QSkinny

4. Outlook

https://github.com/uwerat/qskinny

QML

Button {
}

QML engine

QtQuick

Qt scene graph

OpenGL

https://github.com/uwerat/qskinny

QtQuickControls 1

Write everything in QML

```
Control {
    id: slider
    (...)
    style: Settings.styleComponent(Settings.style, "SliderStyle.qml", slider)
    property Component tickmarks: Repeater {
    Rectangle {
        color: "#777"
        width: 1
        height: 3
        y: (...)
        x: (...)
        }
    }
}
```

https://github.com/uwerat/qskinny

QtQuickControls 2

some parts QML, some C++

```
--- qquickslider_p.h:

class Q_QUICKTEMPLATES2_PRIVATE_EXPORT QQuickSlider : public QQuickControl
{
    Q_OBJECT
    Q_PROPERTY(qreal from READ from WRITE setFrom NOTIFY fromChanged FINAL)
    Q_PROPERTY(qreal to READ to WRITE setTo NOTIFY toChanged FINAL)
    (...)
};

--- Slider.qml:

T.Slider {
    id: control
}
```
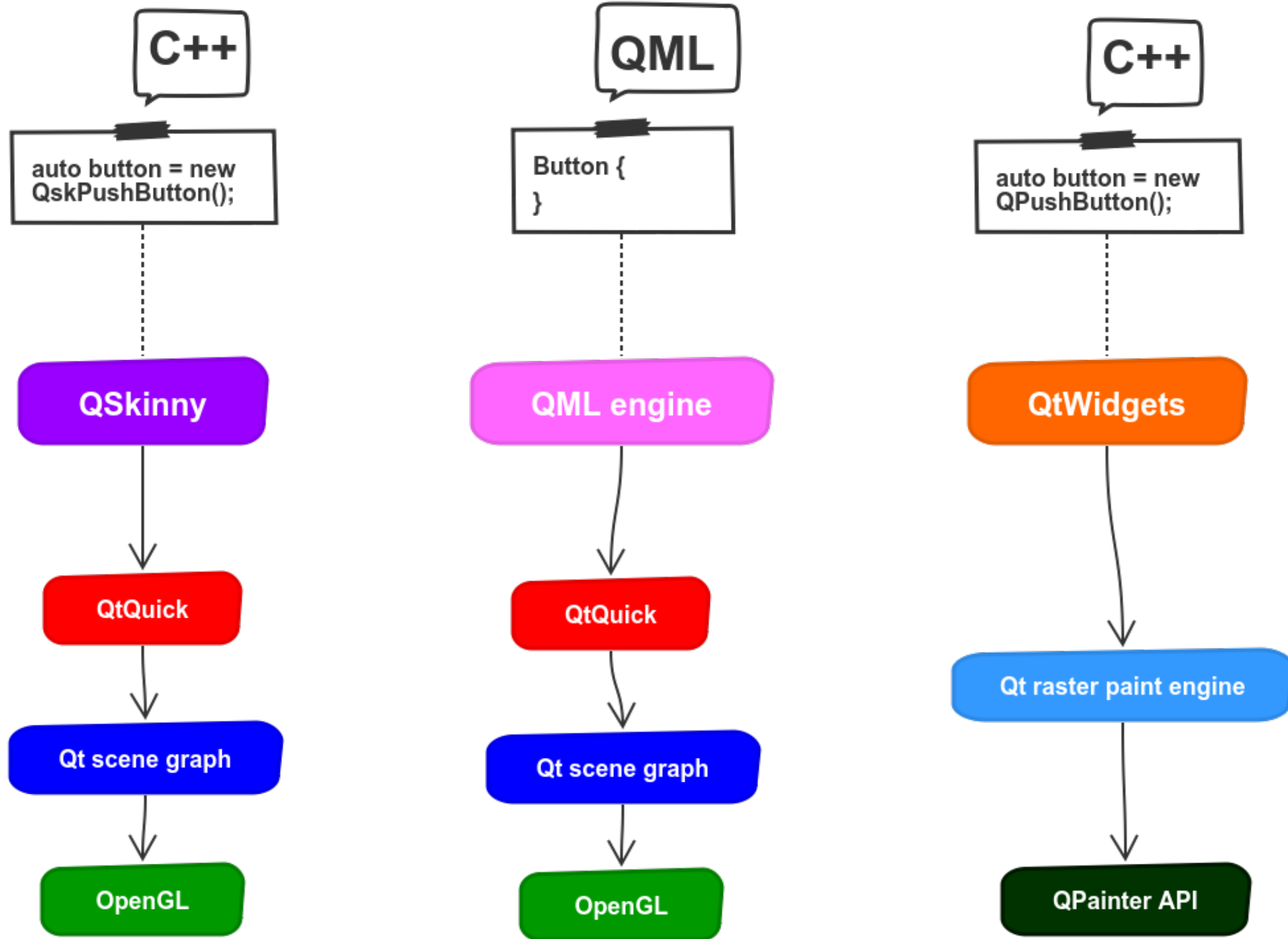
Agenda

1. QML under the hood
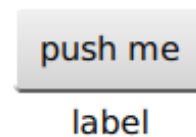
2. The QML / C++ boundary

3. <u>QSkinny</u>

4. Outlook
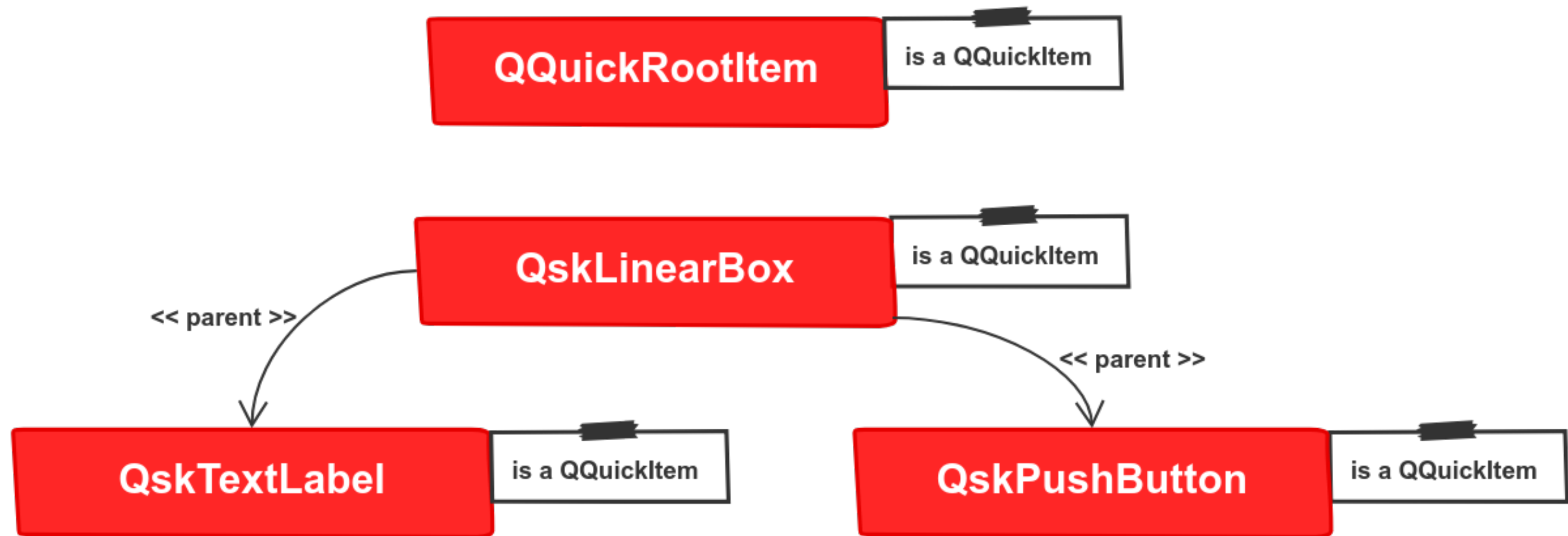
https://github.com/uwerat/qskinny

www.sketchboard.io

QSkinny design goals

» lightweight

» flexible theming

» dynamic sizing

QSkinny API

```cpp
QskWindow window;
auto box = new QskLinearBox(Qt::Vertical);
auto button = new QskPushButton("push me", box);
auto label = new QskTextLabel("label", box);
window.addItem(box);
window.show();
```
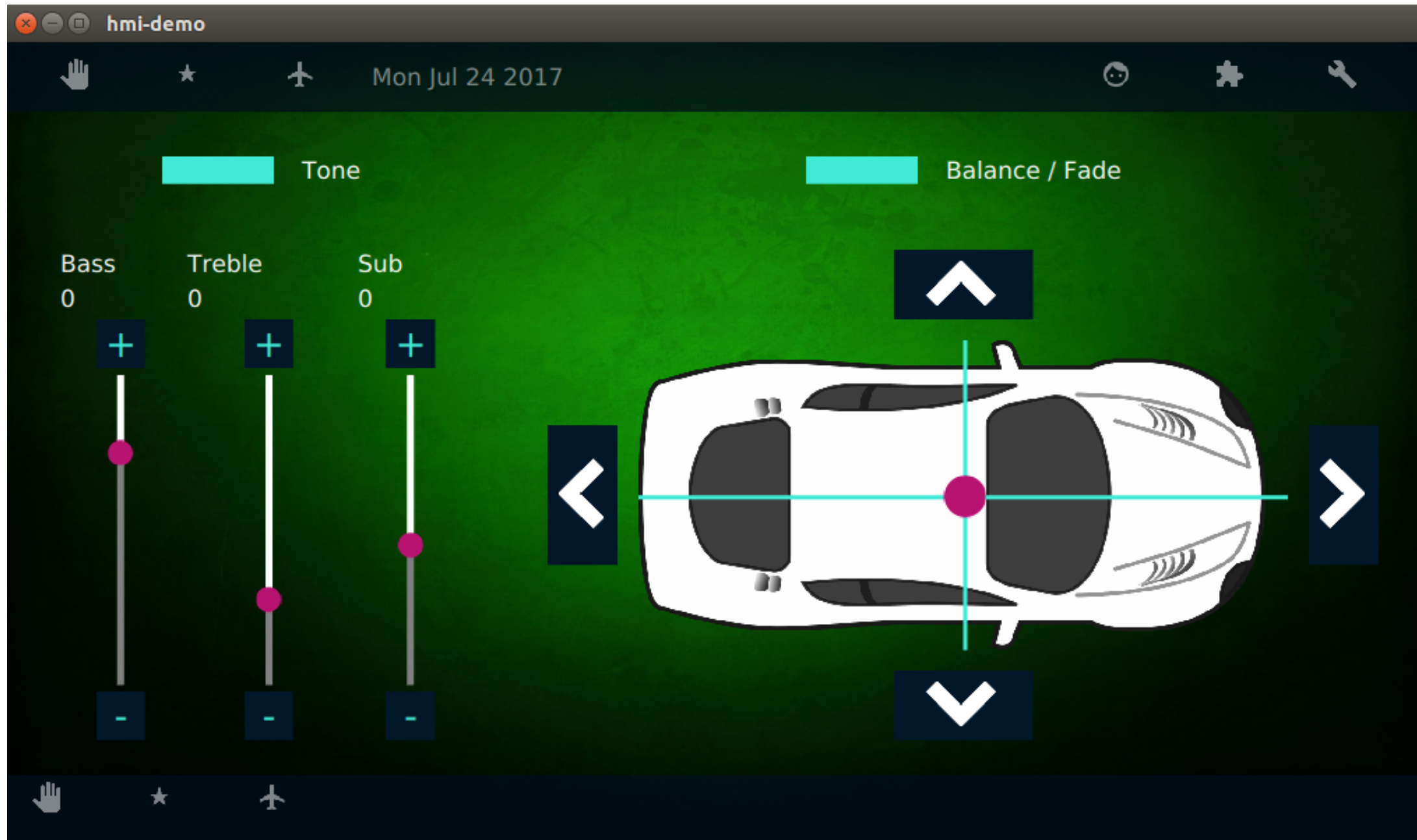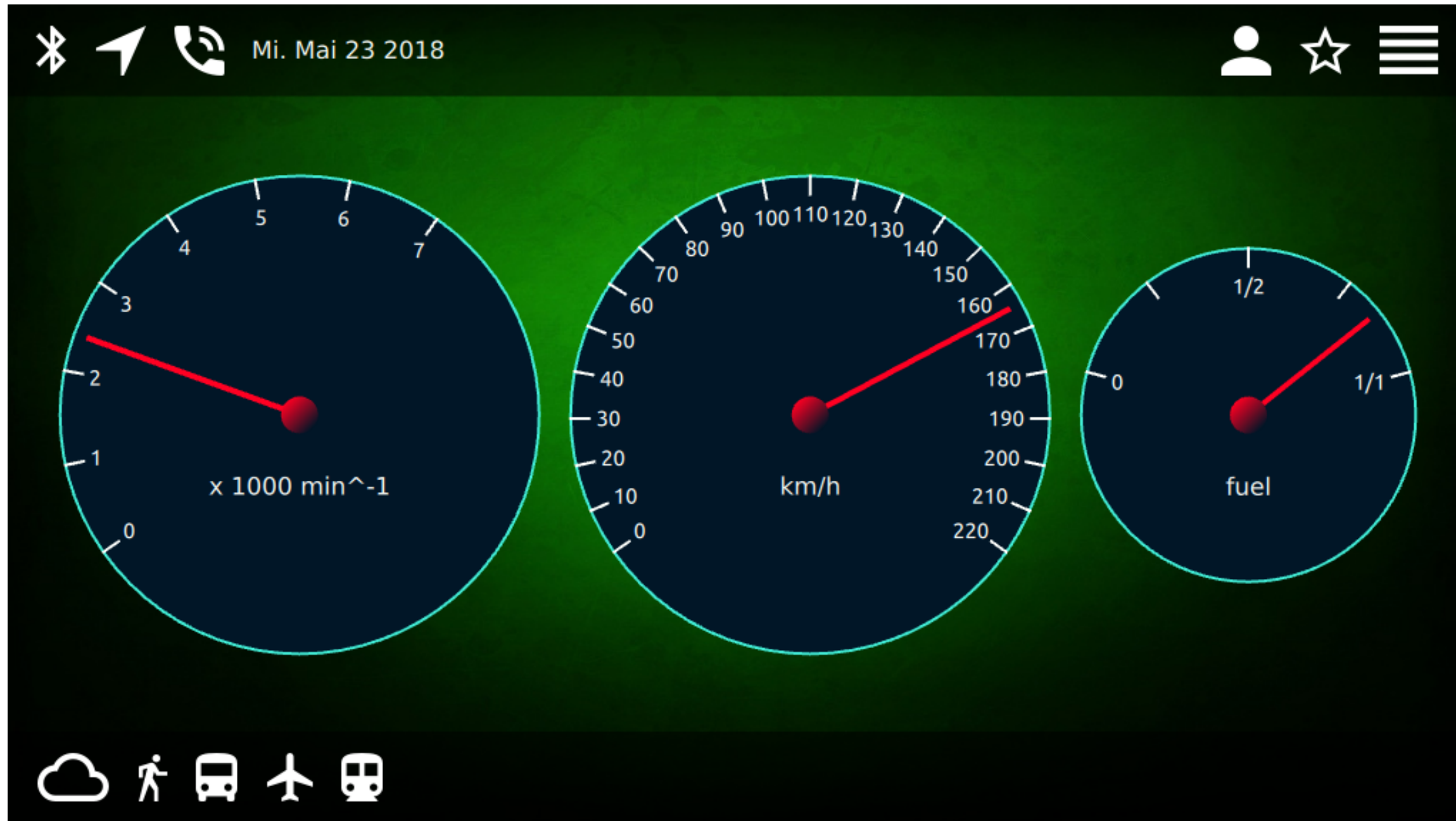
QQuickRootItem — is a QQuickItem

QskLinearBox — is a QQuickItem

<< parent >>

<< parent >>

QskTextLabel — is a QQuickItem

QskPushButton — is a QQuickItem

www.sketchboard.io

https://github.com/uwerat/qskinny

example

example

Agenda

1. QML under the hood

2. The QML / C++ boundary

3. QSkinny

4. <u>Outlook</u>

QSkinny

» polishing

» documentation

Qt 6

» new styling?

» opening up QtQuickControls 2?

Discussion

feedback to @peha23 on Twitter