Qt Quick Designer for Desktop Applications

Shantanu Tushar

shantanu.tushar@kdab.com

KDAB

About Me

- Software Engineer at KDAB
- Contributor to KDE since 2009
  - Mentored during Season of KDE and Google Summer of Code
- <3 working on desktop software

KDAB

Before we talk about Qt Quick Designer...

A Refresher on QML and Qt Quick

Qt Quick?

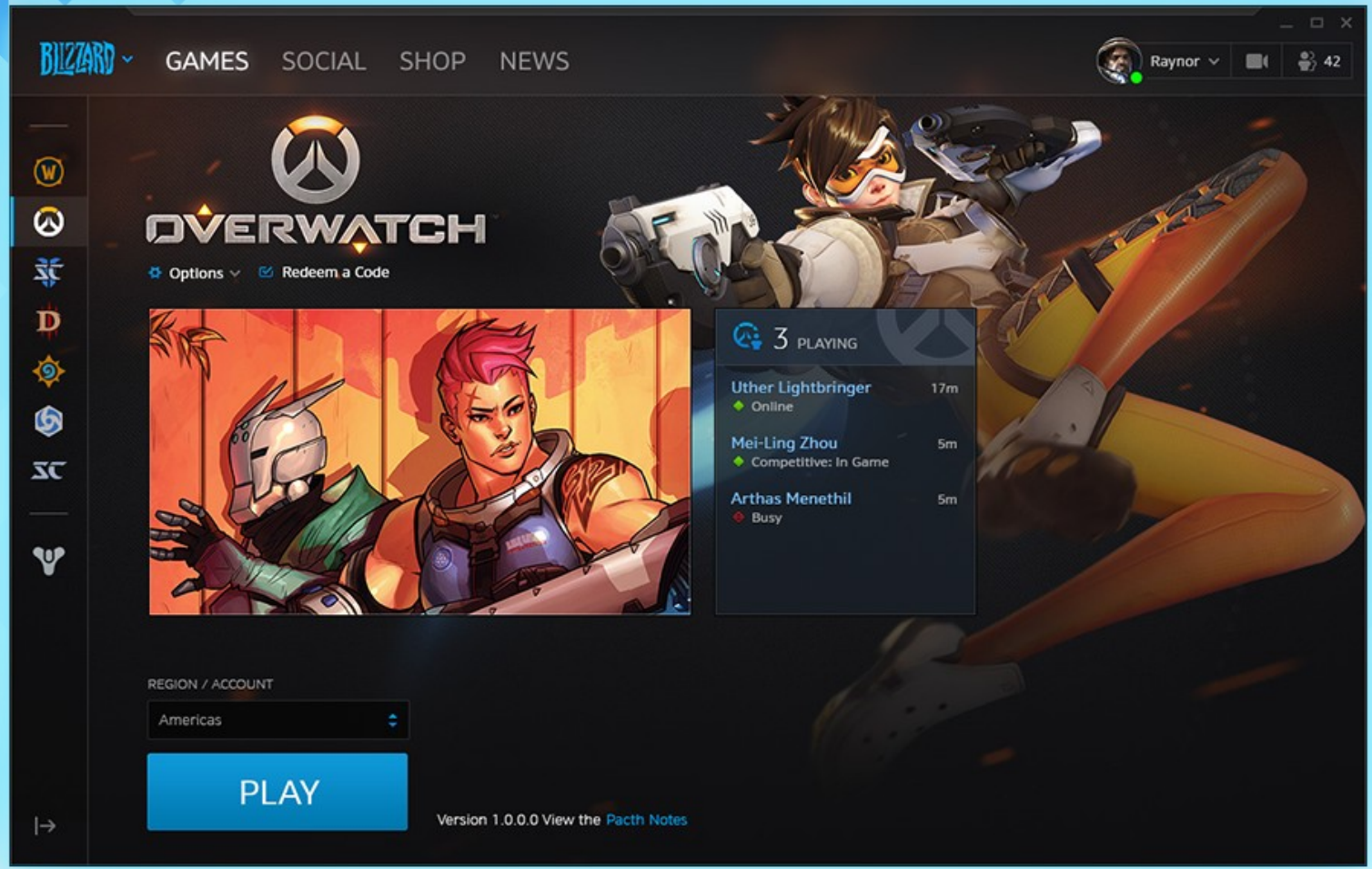
- Qt Quick is a collection of tools to create User Interfaces using QML:
  - Visual elements (Rectangle, MouseArea, Button etc)
  - States, Transitions and Animations
  - Model/View classes
  - Particle and Graphical Effects

Key considerations

- UI “Complexity”
  - Density
  - Component availability (widgets etc)
- Styling
  - Branding, Native look & feel
  - Nature of the design team
- Availability of GPU acceleration

Apps from The Wild

KDAB

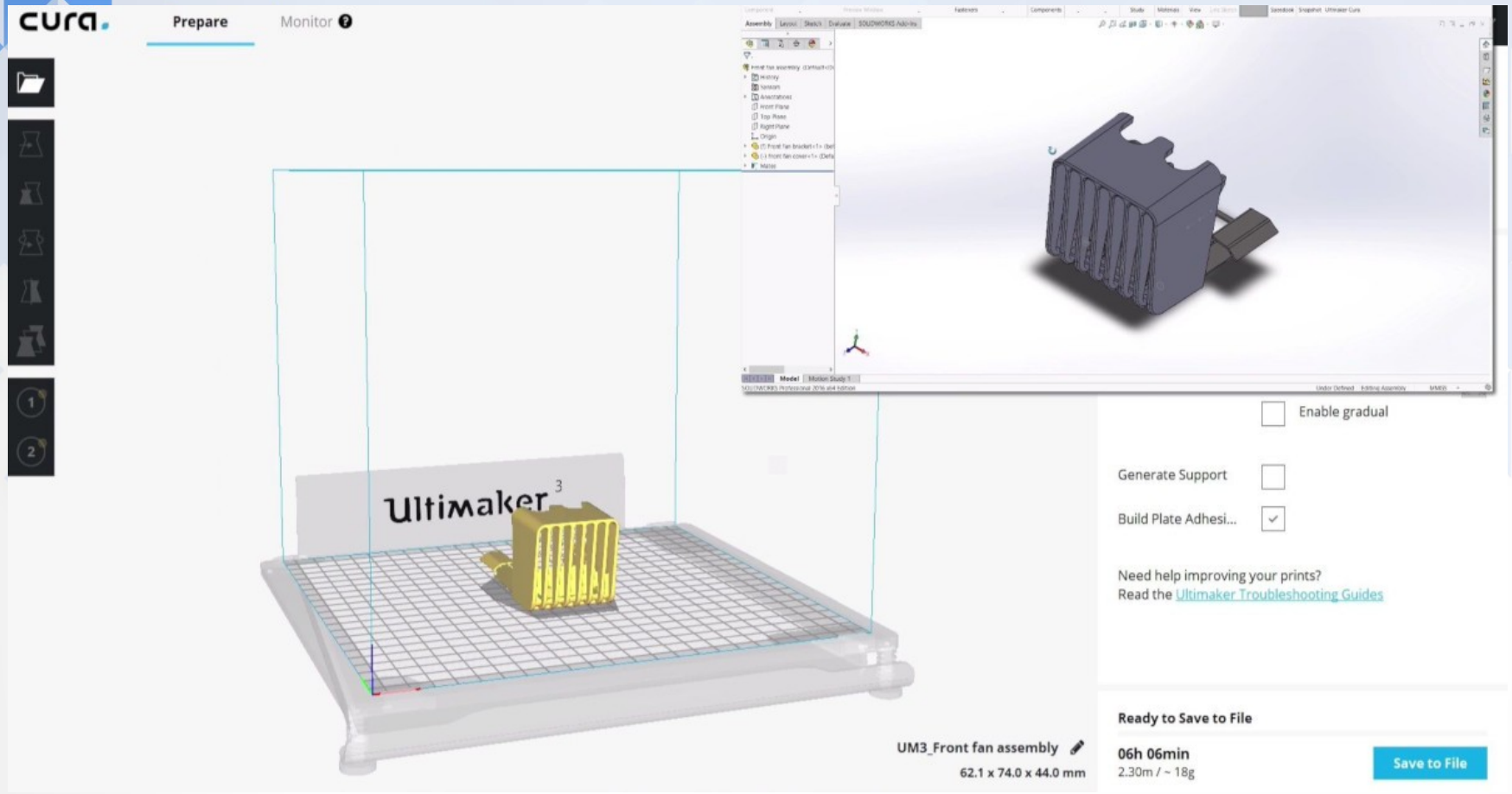


Blizzard Battle.net

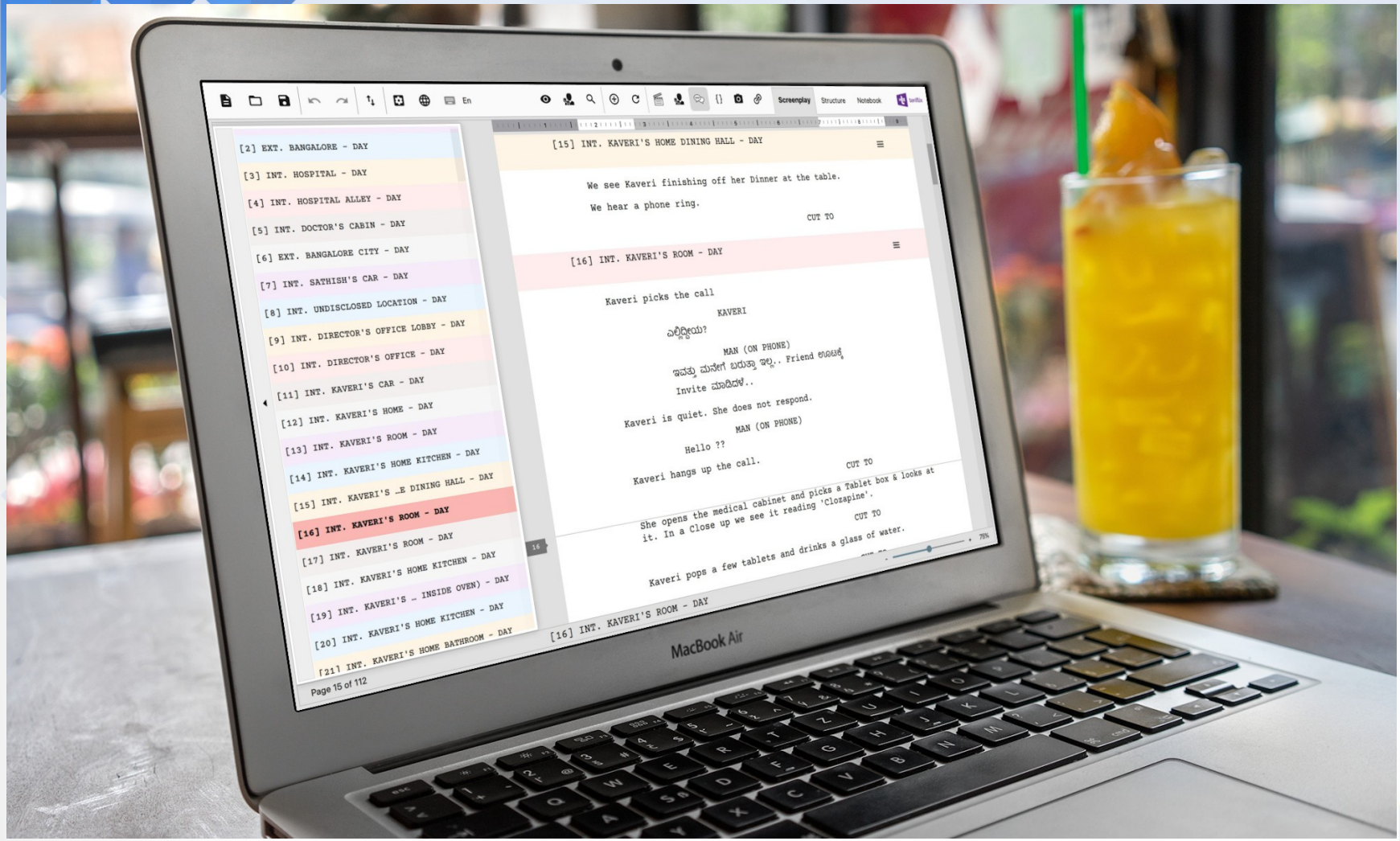
KDAB

SoStronk esports

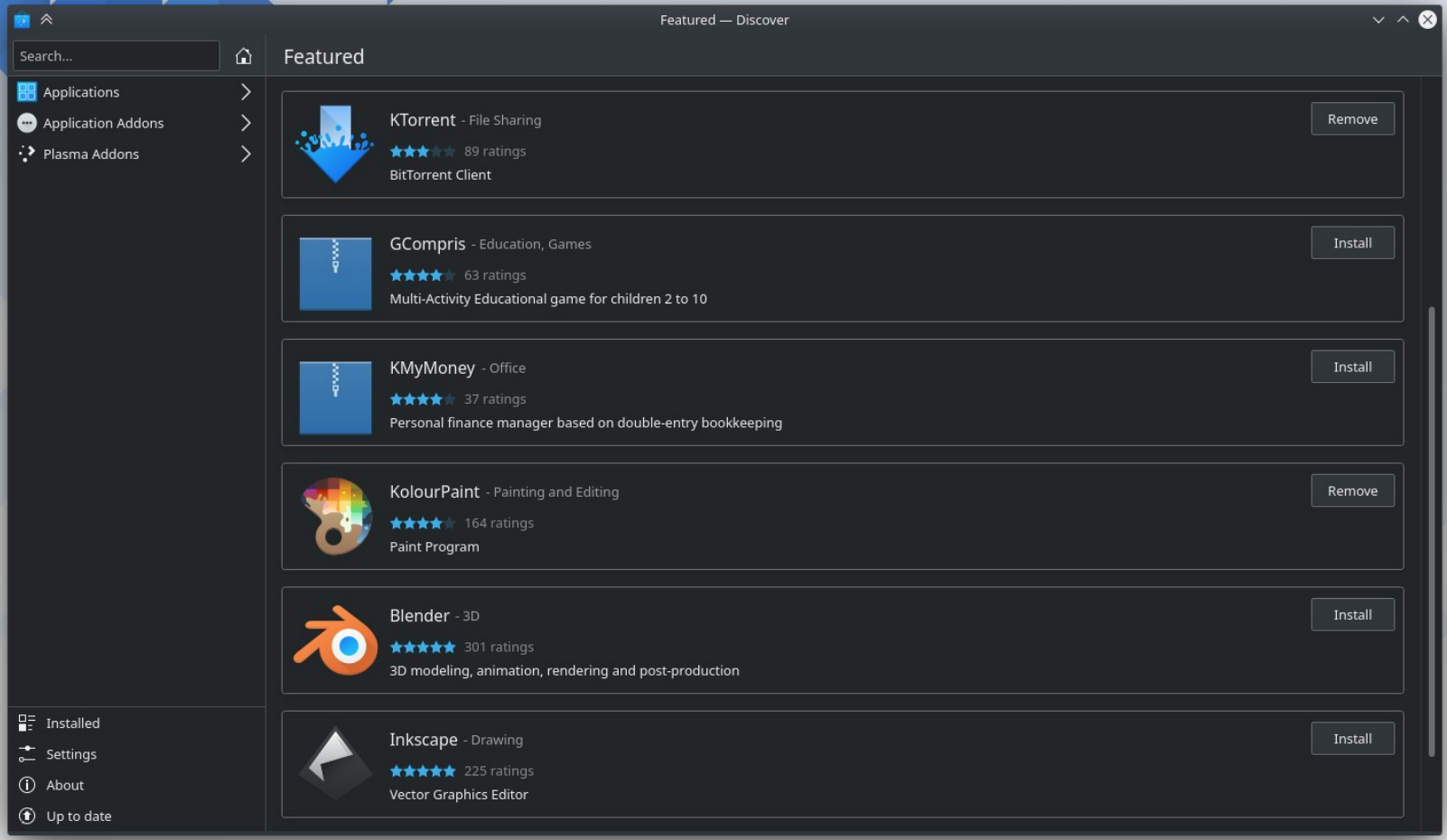
KDAB



Ultimaker Cura – 3D Printing



Scrite - Multilingual Screenplay App



Discover – Software Center

What is involved?

Codebase structure

- QML, JS and design assets are easier with Qt's Resource Sytem
- Data models and business logic in C++
- Share business logic across multiple UI options (form factor etc)
  - common/
  - desktop/
  - mobile/

Managing QML Imports

- One “main” .qrc with main.qml etc
- Keep an “imports” dir with custom and 3rd party imports
  - imports/
    - imports/Views
    - imports/Controls
    - imports/3rdParty/Baz
- Each import with a .qrc and qmldir
- Use Qt’s file selectors when appropriate

KDAB

Managing assets

- Use Qt's resource system
- Automatically pick files-

```
RESOURCES += \
    $$files(images/*) \
    $$files(sounds/*)
```

KDAB

Managing QML Imports (..contd)

- Use namespaces when importing
  - import MyApp.media 1.0 as MyMedia
  - MyMedia.Playlist {}
- Avoids collisions with other imports
- Singletons from QML
  - singleton Theme 1.0 Theme.qml

KDAB

UI Building Blocks in Qt Quick

- Basic elements – Item, Rectangle, MouseArea etc
- Views – ListView, GridView, PathView
- Loader, Repeater
- Qt Quick Controls 2 – Button, CheckBox, TextInput etc
- Qt Quick Layouts – GridLayout, RowLayout, ColumnLayout

Arranging UI elements

- Raw geometry
  - x, y, width, height
  - Not very responsive
  - Easy to animate
- Anchors
  - Relative positioning
  - Mostly responsive
  - Harder to animate
- QtQuick Layouts
  - Positioning based on layout hints
  - Responsive
  - Very difficult to animate

Styling Qt Quick Controls 2

- Use one of the inbuilt styles (Default, Fusion, Material etc)
- Customize via code (create your own Button.qml etc)
- Provide design assets (aka Imagine style)
  - Controls as images (PNG)
  - Each state (pressed, checked etc) is one file
  - Animations via WEBP

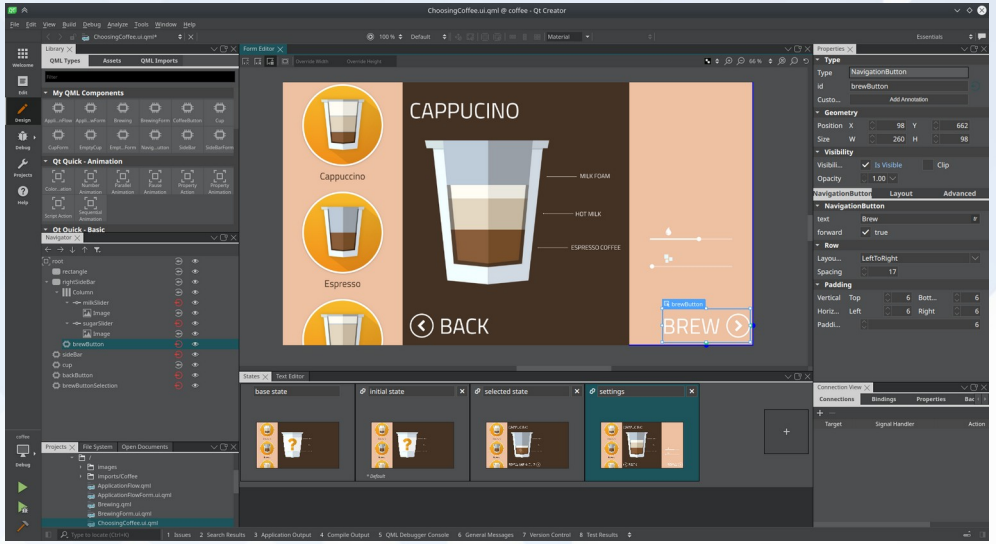
KDAB

Qt Quick Designer

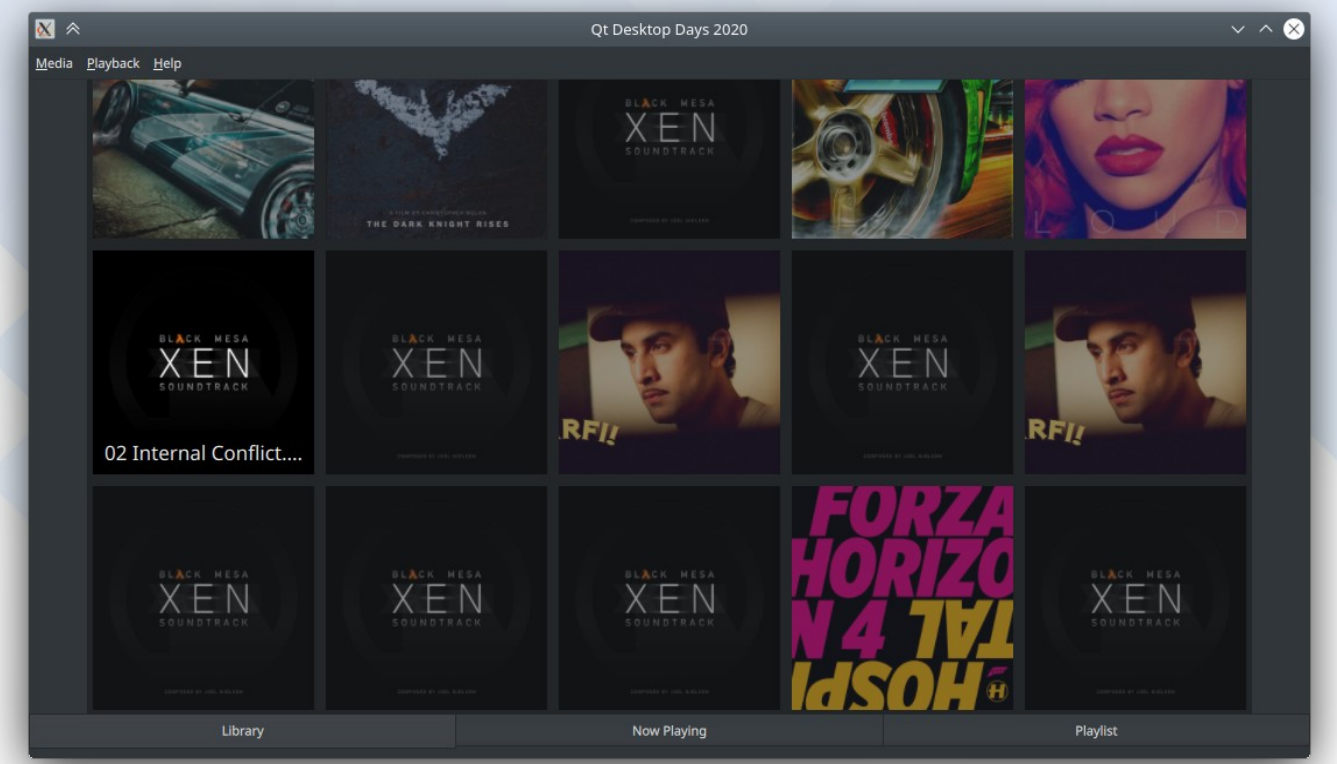
KDAB

Using Qt Quick Designer

- UI design tool with live preview
- Workflow varies
  - Initial layout
  - Complete development with a design team
  - Previewing states
- Design file restrictions



Demo App – a simple music player



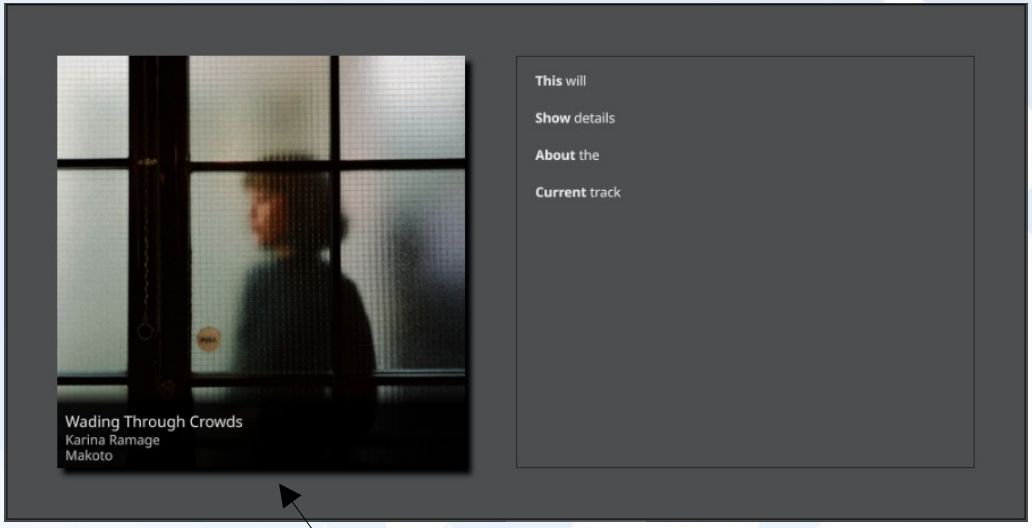
https://github.com/shaan7/qtdd2020player

KDAB

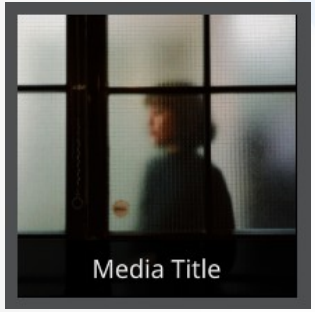
Components

Media Library

Current Media



Media Library Delegate



Media Info

KDAB

Implementation ↔ Design

CurrentMedia.qml

```
CurrentMediaDesign {

    MediaPlayer {

        id: player

    }

    mediaInfo.title.text:
player.metaData.title

    mediaInfo.album.text:
player.metaData.albumTitle

}
```

CurrentMediaDesign.ui.qml

```
Item {
    property alias mediaInfo: mediaInfo

// … other UI elements …

    MediaInfo {
        id: mediaInfo
    }

// … more UI elements …
}
```

KDAB

Demo for adding alias via Designer

User Interaction

main.qml

```
ApplicationWindow {
// ... UI elements ...
    Library {
        onMediaSelected: {
            currentMedia.mediaSource = url;
            swipeView.currentIndex = 1;
        }
    }
// ... UI elements ...
}
```

LibraryDesign.ui.qml

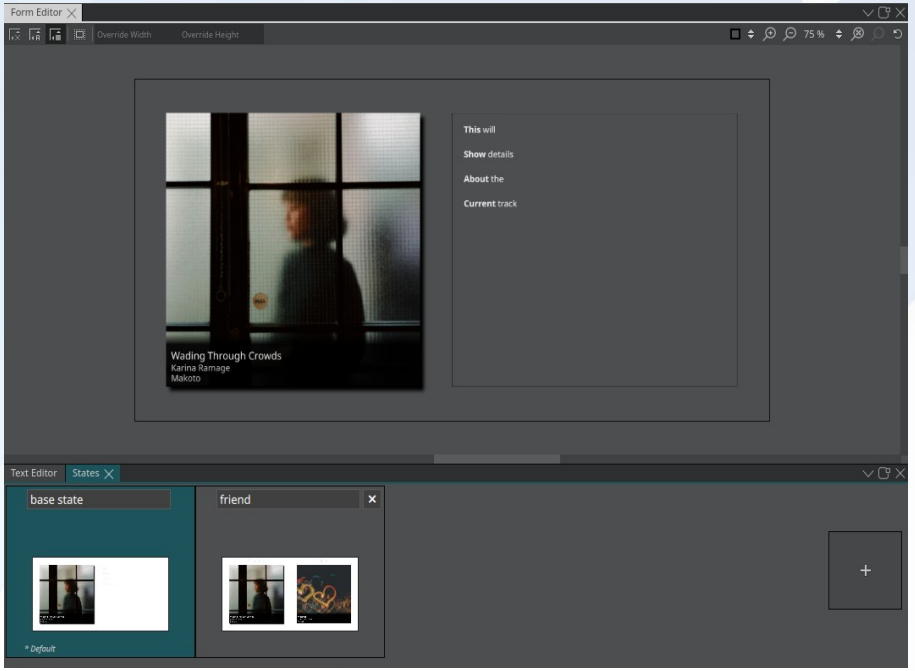
```
Item {
    id: root
    signal mediaSelected(url url)
// .. UI elements ..
    GridView {
        delegate: ItemDelegate {
            id: delegateRoot

            Connections {
                target: delegateRoot
                onClicked: root.mediaSelected(fileUrl)
            }
        }
// ... UI elements ...
}
```

KDAB

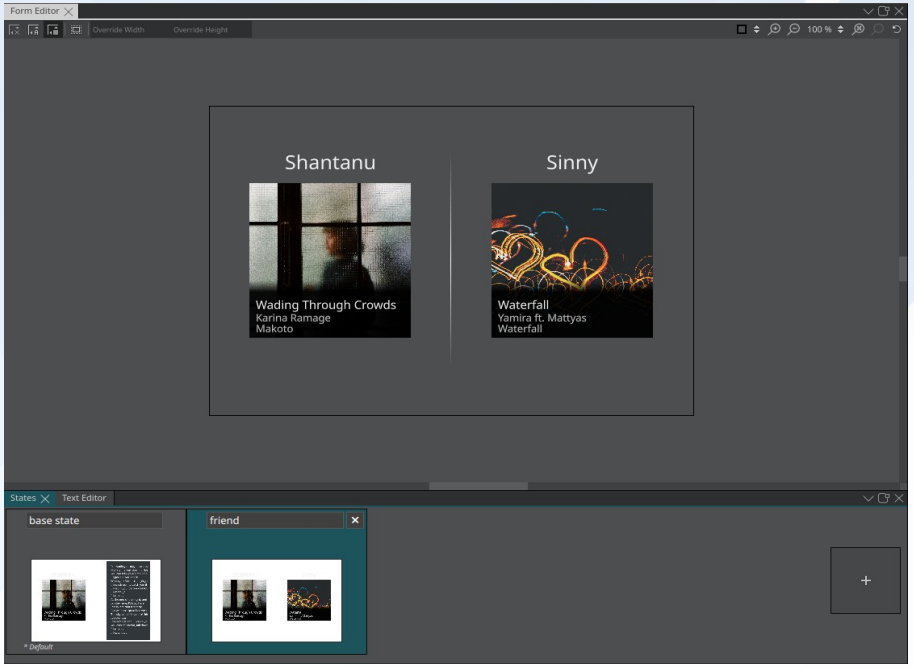
Using Designer To Preview States

- Useful to preview states that will take more effort to test in a running app
- States are listed at the bottom
- One set of States per QML file
- Example –>



Using Designer To Preview States (..contd)

- Example
  - Second state with two people sharing music status
  - No need to run two apps and then invite user on app2 from app1



KDAB

More tips

- Avoid using hardcoded dimensions when developing for desktop
- Use qtquickcontrols2.conf to configure styling
- Using test data to help previews
  - Use prepopulated models to provide data to lists
  - ListModel or a JavaScript Array

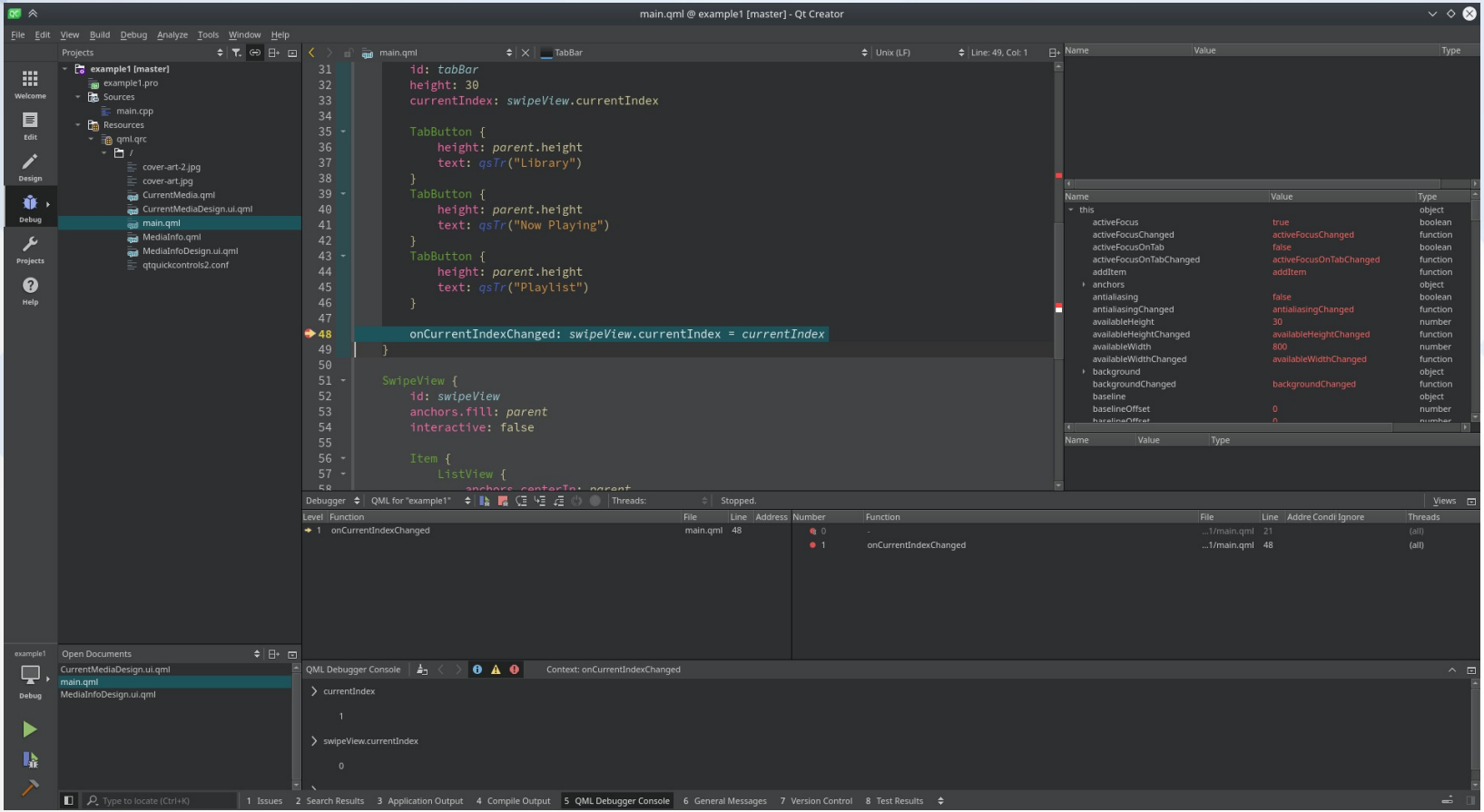
Qt Quick Designer Limitations

- Only single-line JS expressions allowed
- No function calls allowed (except signals)
- Unsupported – Transitions, Timer
- Quirks when using Repeater

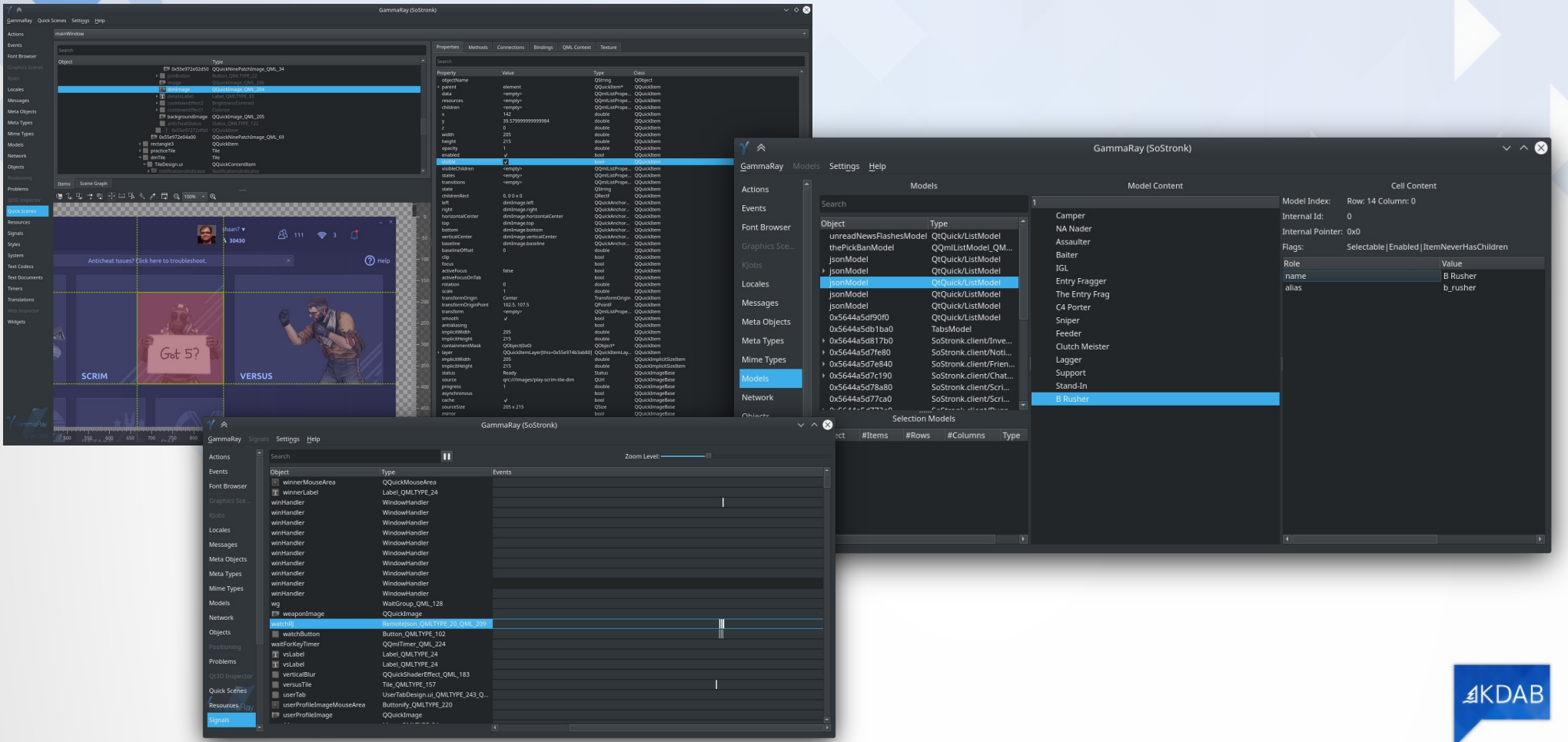
More Tooling

KDAB

QML Debugger



GammaRay


GammaRay (SoStronk)
Models
Model Content
Cell Content
Selection Models
Actions
Events
Font Browser
Locales
Messages
Meta Objects
Meta Types
Mime Types
Network
Objects
Signals
Camper
NA Nader
Assaulter
Baiter
IGL
Entry Fragger
The Entry Frag
C4 Porter
Sniper
Feeder
Clutch Meister
Lagger
Support
Stand-In
B Rusher

Questions?

shantanu.tushar@kdab.com

KDAB