Qt World Summit 2017

October 10-12 | Berlin, Germany

Using Virtual Keyboards on Q

Jan Arne Petersen, Senior Software Engineer at KDAB

What is needed

- **What is needed**
- What is provided by Qt
- Use Qt input method API in Qt applications

What kind of text needs to be inputted?

- Just some PIN or a WLAN password
- Machine name or simple setup
- Full text input for search or editing
- Support for browser/3rd party applications

What kind of embedded device?

- What kind of screen?
- What hardware button?
- What other kinds of inputs?

What is provided by Qt

- What is needed
- **What is provided by Qt**
  - Input Method API
  - QPA platforms
  - Qt Virtual Keyboard
- Use Qt input method API in Qt applications

What is provided by Qt

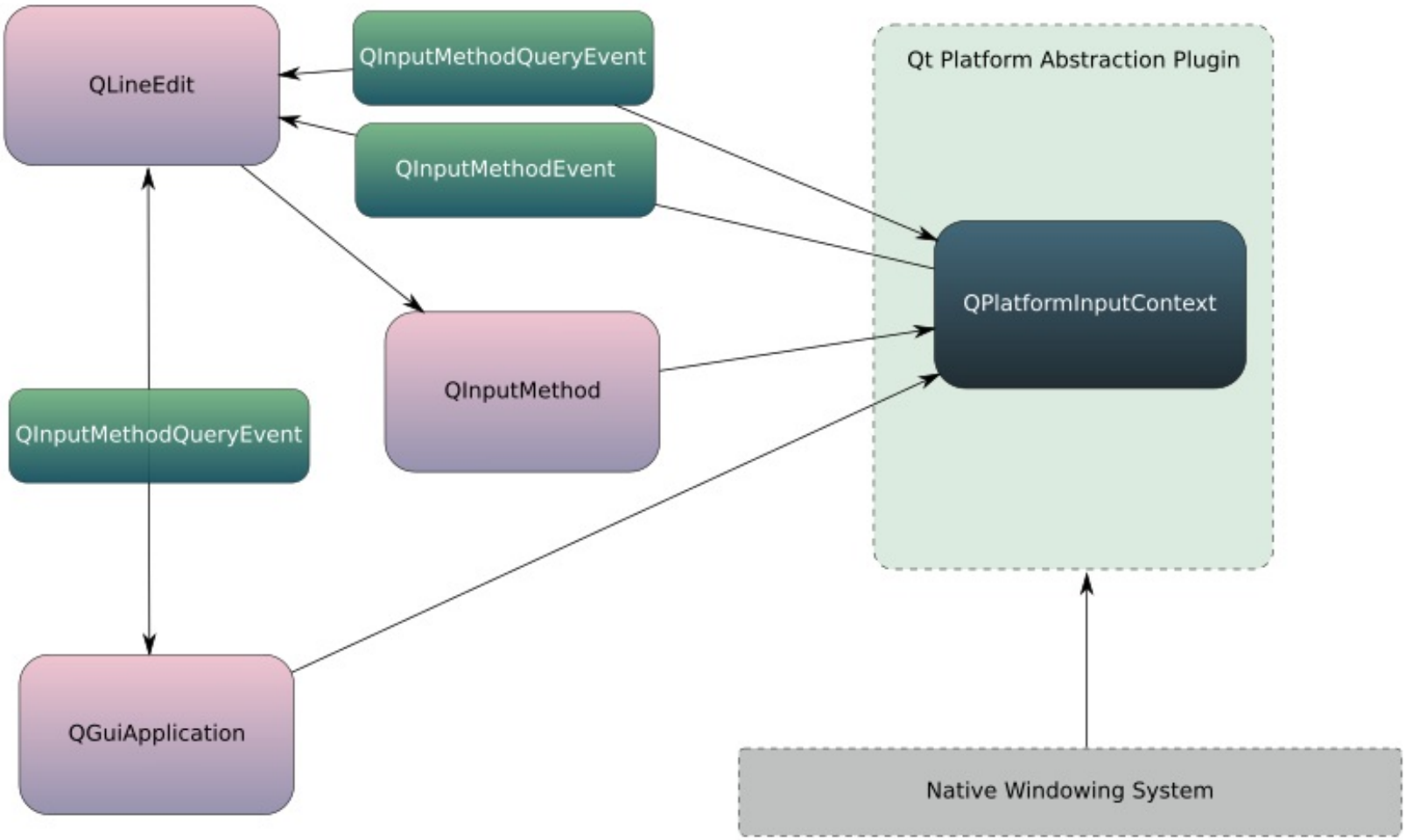
- **Input Method API**
- QPA platforms
- Qt Virtual Keyboard

Qt Input Method API

Virtual Keyboards are integrated in Qt via the Input Method API:

- QInputMethod - access virtual keyboard from application
- QPlatformInputContext - virtual keyboard side
- QInputMethodQueryEvent - send information from application to virtual keyboard
- QInputMethodEvent and QKeyEvent - send input events from virtual keyboard to application

Input Methods in Qt - Overview



What is provided by Qt

- Input Method API
- **QPA platforms**
- Qt Virtual Keyboard

QPlatformInputContext

- Virtual keyboard side of API
- Part of Qt Platform Abstraction
- Two kinds of QPlatformInputContext for virtual keyboards
  - Native provided by platform
  - Custom via QPlatformInputContextFactory

Native Virtual Keyboards

- Uses the virtual keyboard provided by the system
- Supported QPA platforms:
  - android
  - ios
  - qnx
  - wayland
  - windows

Custom Virtual Keyboards

- QPlatformInputContextFactory
- Supported QPA platforms (in Qt 5.11):
  - bsdfb
  - cocoa
  - directfb/linuxfb
  - eglfs
  - integrity
  - mirclient
  - vnc
  - wayland
  - windows
  - xcb

QPlatformInputContextFactory

- Plugin is defined via QT_IM_MODULE
- Input context creation harmonized in Qt 5.6
  - null: default platform context
  - empty: no context
  - set: set one, if it exists and is valid (otherwise no context)

```
1 QString icStr = QPlatformInputContextFactory::requested();
2 if (!icStr.isNull()) {
3     mInputContext.reset(QPlatformInputContextFactory::create(icStr));
4 } else {
5     QPlatformInputContext *ctx = new QWaylandInputContext(mDisplay.data());
6     mInputContext.reset(ctx);
7 }
```

Wayland

- The default platform context uses the keyboard provided by the compositor via the "text-input" protocol
- Next official version in wayland-protocol will be "text-input-unstable-v3"
- When using a QtWayland compositor the default context forwards the Qt Input Method API from the application to the compositor so that one can just use any QPlatformInputContext on compositor side
- There is also the "input-method" protocol for out-of-process virtual keyboards (not supported in QtWayland yet)

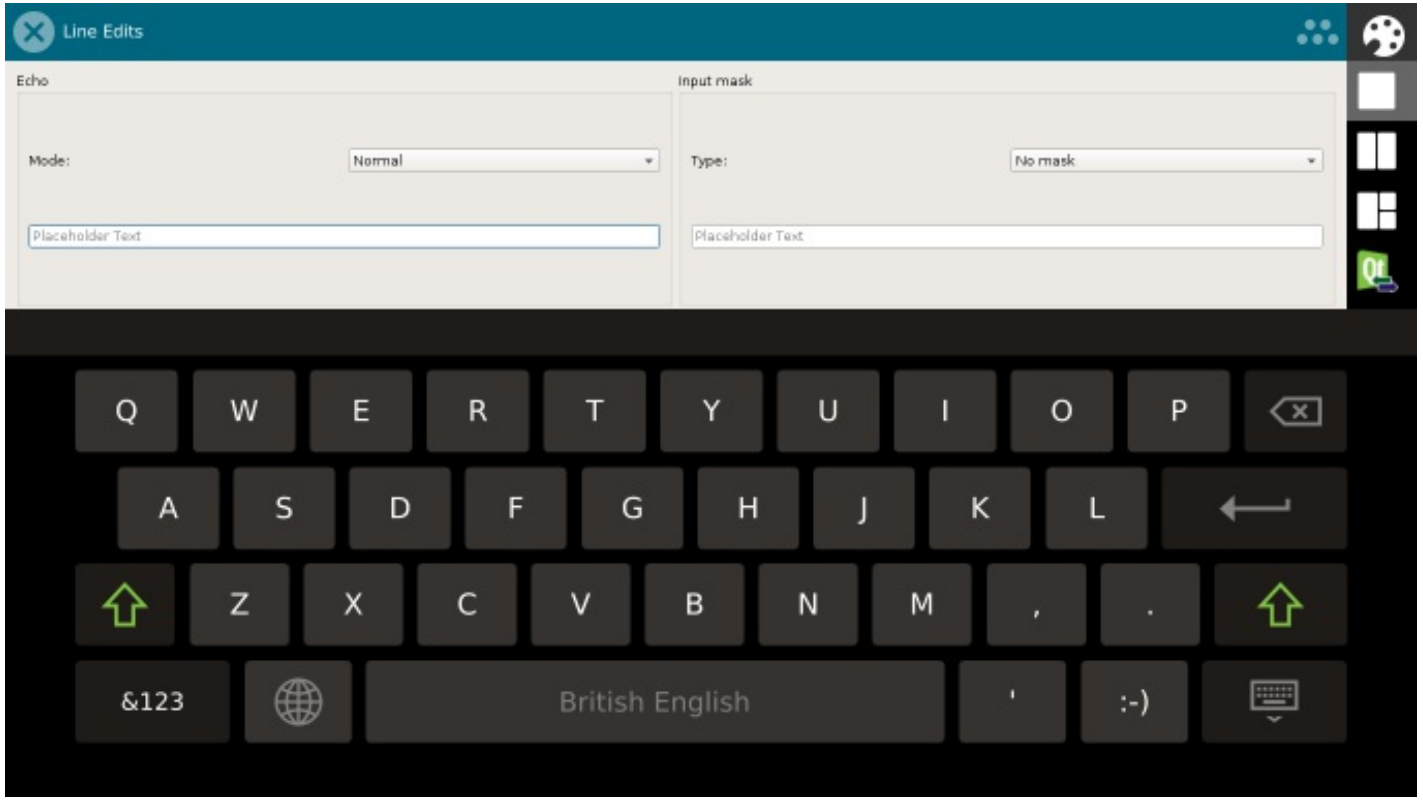
```
1 import QtQuick 2.0
2 import QtWayland.Compositor 1.1
3
4 WaylandCompositor {
5 ...
6     TextInputManager {
7     }
8
9 }
```

Embedding keyboard in application

- Some virtual keyboards (like Qt Virtual Keyboard) allow embedding in an application
- Especially useful for platforms without multiple window management like eglfs
- With previously mentioned patch it can be used in a QtWayland compositor to embed the Qt Virtual Keyboard in the compositor

```
1  import QtQuick 2.5
2  import QtQuick.VirtualKeyboard 2.1
3
4  InputPanel {
5      id: inputPanel
6      visible: active
7      y: active ? parent.height - inputPanel.height : parent.height
8      anchors.left: parent.left
9      anchors.right: parent.right
10 }
```

Embedding keyboard in QtWayland compositor



What is provided by Qt

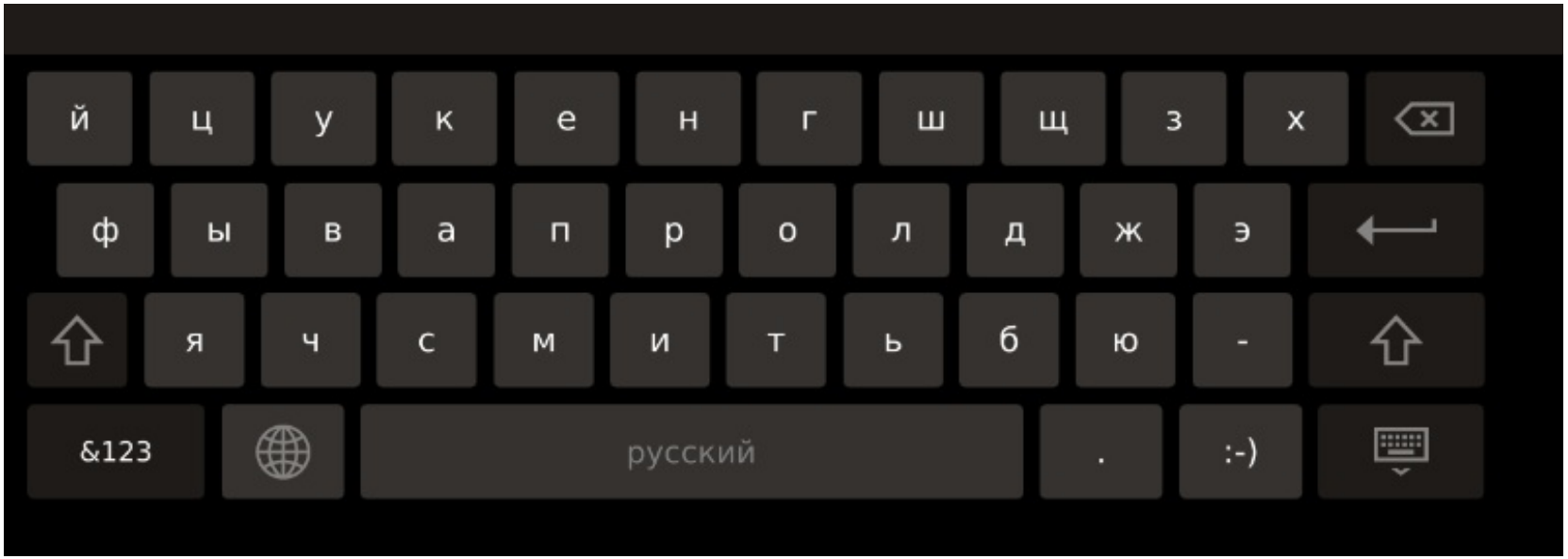
- Input Method API
- QPA platforms
- **Qt Virtual Keyboard**

Qt Virtual Keyboard

- Commercial and GPL
- For xcb platform it displays automatically in a separate window
- For other QPA platforms it allows embedding in application window
- Uses QML
- Supports multiple languages like: English, French, German, Russian, Arabic, ...
- Supports Chinese, Japanese and Korean
- Supports text correction (hunspell)
- Supports handwriting (Lipi toolkit)

Qt Virtual Keyboard

QT_IM_MODULE=qtvirtualkeyboard



Qt Virtual Keyboard

Supports additional commercial engines

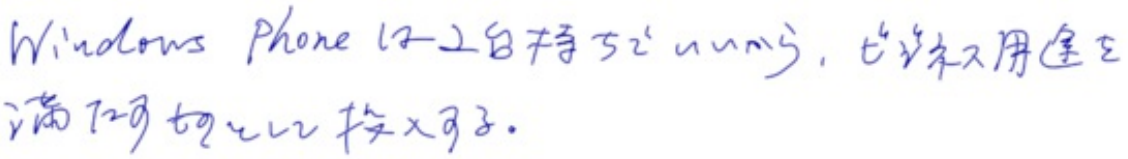
- KDAB worked together with MyScript and The Qt Company to support MyScript's handwriting input technology in the Qt Virtual Keyboard for the Qt Automotive Suite.
- Should be included in Qt 5.11

MyScript background

- Problems to solve
  - Overcome the HMI Complexity
  - Decrease the Driver Distraction
- MyScript at a glance
  - 19 years of expertise
  - 120 employees o/w 25 PhDs and 75 engineers
  - Over 400 M users in the world
  - Over 200 value added partners: OEM, ISV and System Integrators
  - Millions of cars on the road use MyScript
  - 2007: 1st Concept Car (Audi, Tokyo car show)
  - 2010: 1st Car on the road with the Audi
  - 2013: 1st Mercedes
  - 2014: 1st Tesla
  - 2015: 1st Porsche and 1st VW
  - 2016: 1st OEM Automotive App with VW

MyScript technology

- The Most flexible engine with carefree writing styles
  - Support cursive Latin writing in all languages



  - Automatic space insertion between words
  - Flexible letter alignment
  - Write words or part of words on top of each other
  - Same engine and API support handwriting recognition and keyboard
  - Transliteration
  - Prediction
  - Spelling correction
- Support of up to 65 languages for word recognition
  - 99 languages for character-by-character recognition

Today supports all available languages of the Qt virtual keyboards

Use Qt input method API in Qt applications

- What is needed
- What is provided by Qt
- **Use Qt input method API in Qt applications**

Improve the user experience

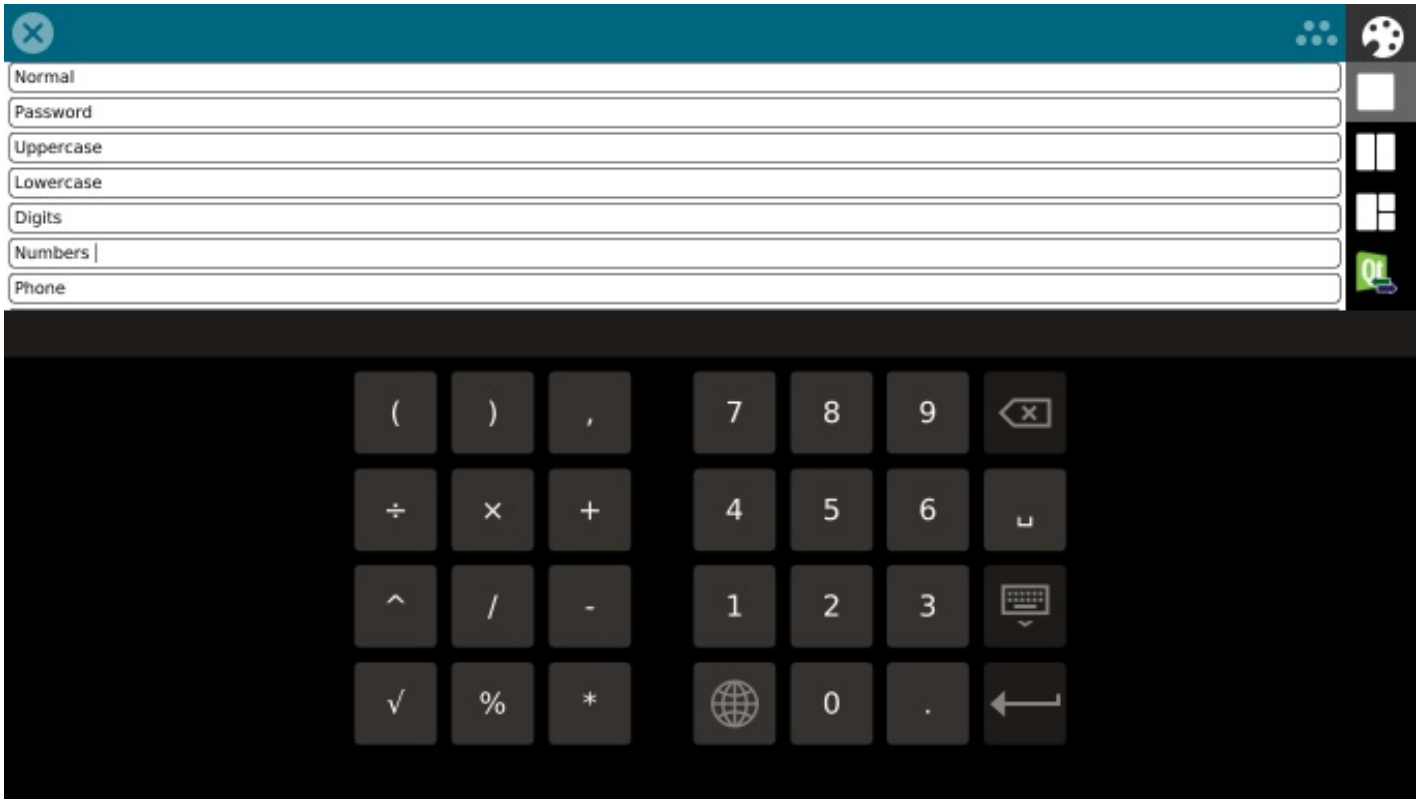
- Qt input fields have builtin support for the input method API but there are still ways for application developers to improve the user experience with virtual keyboards:
  - Define purpose of text input fields
  - Alter apperance of Return key
  - Change UI depending on keyboard

Define purpose of text input fields

- The keyboard can change the layout depending on the purpose of the text field
- For example entering digits, emails, phone numbers
- Qt::InputMethodHints enum and Qt::ImHints Qt::InputMethodQuery
- For example:
  - Qt::ImhNone - No hints
  - Qt::ImhHiddenText - The input method should not show the characters while typing
  - Qt::ImhDigitsOnly - Only digits are allowed
  - Qt::ImhFormattedNumbersOnly - Only number input is allowed
  - Qt::ImhDialableCharactersOnly - Only characters suitable for phone dialing are allowed
  - Qt::ImhEmailCharactersOnly - Only characters suitable for email addresses are allowed
- Multiple hints can be combined. For example for password fields:
  - Qt.ImhNoAutoUppercase | Qt.ImhNoPredictiveText | Qt.ImhSensitiveData | Qt.ImhHiddenText

Define purpose of text input fields - example

```
1 TextInput {
2     id: input
3
4     inputMethodHints: Qt.ImhFormattedNumbersOnly
5 }
```



Alter apperance of Return key

- Can be used to display alternative key instead of Return
- Qt::EnterKeyType enum and Qt::ImEnterKeyType Qt::InputMethodQuery
- For example:
  - Qt::EnterKeyDone - Show a "Done" button
  - Qt::EnterKeySend - Show a "Send" button
  - Qt::EnterKeySearch - Show a "Search" button
  - Qt::EnterKeyReturn - Show a Return button that inserts a new line
  - Qt::EnterKeyNext - Show a "Next" button which should be used to navigate to next input field
- Not all of these values are supported on all platforms. For unsupported values the default key will be used instead.

Future (Qt 5.11?): Qt::ImEnterKeyLabel and Qt::ImEnterKeyEnabled

Alter apperance of Return key - Example

```
1 TextInput {
2     id: input
3
4     EnterKey.type: Qt.EnterKeySearch // Show a "Search" button
5 }
```

Change UI depending on keyboard

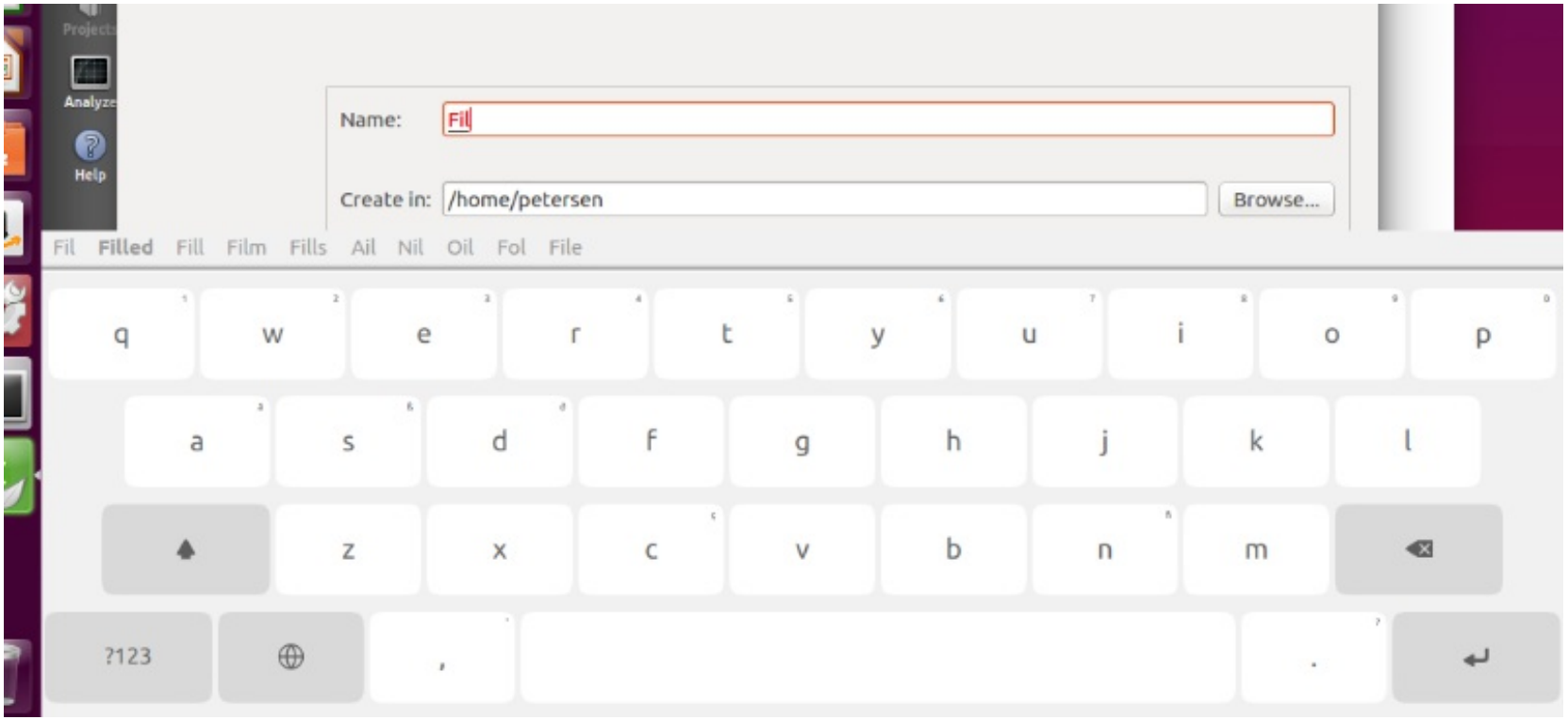
- When a virtual keyboard is shown it might overlap some parts of the application
- In particular not so nice to overlap the focused input field
- QInputMethod::visible property can be used to figure out if a virtual keyboard is displayed
- QInputMethod::keyboardRectangle property holds the virtual keyboard's geometry in window coordinates

Questions

Maliit Framework/Keyboard (based on Ubuntu Keyboard)

- Open Source: LGPL-3
- Keyboard runs in a separate process
- Uses QML
- Supports multiple languages like: English, French, German, Russian, Arabic, ...
- Supports Chinese, Japanese, Korean
- Supports text correction and prediction

Maliit Keyboard



Thank you!

www.kdab.com

jan.petersen@kdab.com