# QT实现 语音视频交互系统

## 1 使用 Qt Quick

Qt 目前有两种创建用户界面的方式：

* Qt Widgets
* Qt Quick

其中 Qt Widgets 是传统的桌面界面库，而 Qt Quick 是新一代的高级用户界面技术，可以轻松的用于移动端、嵌入式设备等界面开发。

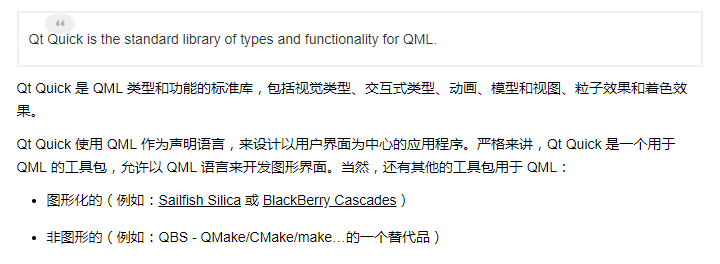
目前 Qt Widgets 已经基本处于维护阶段，已经非常稳定且成熟。而 Qt Quick 是未来发展的主要方向，其开发更加简捷方便，用户体验更加好

Qt Quick基本介绍

众所周知，Qt 为跨平台而生，而 QML/Qt Quick 作为 Qt 新生力量，完完全全继承了 Qt 包罗万象的特点，它的诞生为 Qt 进军移动领域迈出了历史性的一步。

随着 Qt 的不断迭代，QML/Qt Quick 也可用于开发传统的桌面程序，而且效率越来越高，这样以来，便可以用 QML/Qt Quick 做任何你想做的事情。

QML/QtQuick 作为 Qt 的绝对核心，特别是对于界面要求较高的开发者来说，其作用更为重要。



**2 设计交互界面**

有 2 个主要 UI 界面：

登录频道界面；

视频通话界面；

以及 3 个辅助 UI 界面：

欢迎界面；

视频参数设置界面；

设备设置界面；

UI 之间的交互逻辑，已经用对应红色线框标记出来。

**3 导入资源**

**3.1 导入 images 资源**

我们先将准备好的图标等资源，导入到项目中。

1. 将 images 文件夹拷贝到工程目录中；
2. 在 Qt Creator 的项目视图中，右键点击 Resources/qml.qrc 文件；
3. 选择添加现有路径；
4. 选择 images 文件夹；
5. images 文件夹下的所有资源，会自动添加到 qml.qrc 文件中；

**3.2 导入 controls 资源**

在 Qt Quick 中使用按钮等控件时，有两种方式：

1. 使用 Qt Quick 定义的控件；优点是不用自己开发，可以快速集成使用。
2. 使用用户自定义控件；优点是样式可以自己定义，且可以定义更多官方不提供的控件。

我们这里使用事先准备的一些控件，所以先按照步骤导入到项目中。

1. 将 controls 文件夹拷贝到工程目录中；
2. 在 Qt Creator 的项目视图中，右键点击 Resources/qml.qrc 文件；
3. 选择添加现有路径；
4. 选择 controls 文件夹；

controls 文件夹下的所有控件，会自动添加到 qml.qrc 文件中；

### 4.3 导入 [Agora.io](http://agora.io/) 音视频通话 SDK

使用音视频通话功能，需要导入 [Agora.io](http://agora.io/) 对应的 SDK，可以注册 [Agora.io](http://agora.io/) 的开发者账号，并从 [SDK 下载地址](https://docs.agora.io/cn/Agora%20Platform/downloads)中获取对应平台的 SDK。

下载后将对应的头文件拷贝到项目的 include 文件夹中，静态库拷贝到项目中的 lib 文件夹中，动态库则拷贝到项目中的 dll 文件夹中。

需要借助第三方工具包实现



设计完成 UI 后，对应的按钮所触发的业务逻辑需要对应添加。在创建 QtQuick UI File 的时候，例如说创建 Splash UI 时，默认会创建两个 qml 文件：

* SplashForm.ui.qml：UI 的声明描述；
* Splash.qml：UI 对应事件的响应和部分 UI 业务逻辑；

所以，例如说 Button 的点击事件、鼠标事件等，都通过对应控件的 id 进行关联处理。

之后通过基本 UI 业务逻辑完成后，一般需要 QML 与 C++ 之间的逻辑交互。例如按下进入频道的 Join 按钮后，我们需要在 C++ 中调用 Agora 的音视频相关逻辑，来进入频道进行通话。

在 QML 中使用 C++ 的类和对象，一般有两种方式：

在 C++ 中定义一个 QObject 的子类，注册到 QML 中，在 QML 中创建该类的对象；

在 C++ 中创建对象，并将该对象设置为 QML 的上下文属性，在 QML 中使用该属性；

这里使用第二种方式，定义 MainWindow 类，用来作为核心窗体加载 main.qml，并在其构造函数中将本身设置为 QML 的上下文属性：

setWindowFlags(Qt::Window | Qt::FramelessWindowHint);

resize(600, 600);

​

m\_contentView = new QQuickWidget(this);

m\_contentView->rootContext()->setContextProperty("containerWindow", this);

m\_contentView->setResizeMode(QQuickWidget::SizeRootObjectToView);

m\_contentView->setSource(QUrl("qrc:///main.qml"));

​

QVBoxLayout \*layout = new QVBoxLayout;

layout->setContentsMargins(0, 0, 0, 0);

layout->setSpacing(0);

layout->addWidget(m\_contentView);

​

setLayout(layout);

6 视频渲染

Agora SDK 提供接口，使得用户可以自己定义渲染方式。

具体描绘部分，在 updateTextures 中实现如下：

这样就可以将 Agora SDK 回调中的 Frame，绘制在具体的 Widget 上了。

7 核心业务逻辑

我们需要简单封装 Agora SDK 的相关逻辑，以提供音视频通话的功能。

7.1 回调事件

Agora SDK 会提供很多事件的回调信息，例如远端用户加入频道、远端用户退出频道等，我们需要继承 agora::rtc::IRtcEngineEventHandler 事件回调类，并重写部分需要的函数，来进行事件的响应。

这里我们将事件从 AgoraRtcEngine 的信号函数发出，并在 UI 中进行响应，不做复杂的处理逻辑。