Работа с основными API SailFishOS.

Марк Заславский, mark.zaslavskiy@fruct.org

План

Ограничения эмулятора

Специфика текущей версии SDK

Поддерживаемые QtQuick API

Platform-specific API

libsailfishapp

SAILFISH OS

Вопрос в зал (Работал ли кто-нибудь с QtQuick?)

Ограничения эмулятора

- Отсутствует большая часть встроенных приложений
- Минимальные настройки
- Нет доступа к сенсорам/position
  - Можно сделать свой MockObject с помощью C++

Специфика SDK

- Многие модули требуют зависимостей:
  - Установка зависимостей Sailfish/Target/<>/manage - обязательно.
  - Указание зависимостей в *.yaml - рекомендуется для автоматической настройки зависимостей на других устройствах.

Специфика SDK - пример

Runtime dependencies which are not automatically detected

Requires:

- sailfishsilica-qt5 >= 0.10.9
- **qt5-qtdeclarative-import-positioning**

Qt API (5.2.2)

- **Network**: nfc, bluetooth.
- **Sensors**: датчики - акселерометр, гироскоп.
- **Location**: карты, геокодирование, построение маршрутов.
- **Position**: определение местоположения различными способами.
- **Webview**
- **Multimedia support**: аудио, видео, камера.

Positioning API

- API для определения географических координат (а также скорости и направления), работы с геообластями, адресами, источниками координат.
- Позволяет отслеживать перемещение объектов в рамках сложных границ.
- Позволяет подключать свои источники данных о местоположении через C++.

Positioning API - основные типы

- **PositionSource** - подписка на уведомления источника информации о местоположении.
- **Position** - время, координаты, скорость, информация о точности полученных данных.
- **Location** - человеко-читаемое описание местоположения.
- **Coordinate** - широта, долгота и высота.

Positioning API - demo

Редактируем **qml/pages/FirstPage.qml**

1. Внедряем в код страницы **PositionSource.**
2. Добавляем новые **Label** для широты и долготы.
3. Указываем id для новых элементов.
4. PositionSource.active=true и updateInterval.
5. Настраиваем обработчик для onPositionChanged, либо сразу привязываем Label к полям PositionSource.

Positioning API - код

```
PositionSource {
        id: src
        updateInterval: 1000
        active: true
// Можно еще проще ---->
/*        onPositionChanged: {
            var coord = src.position.coordinate;
            latitude.text = "Lat: " + coord.latitude;
            longitude.text = "Lon: " + coord.
longitude;
        }*/
    }
```

```
Label {
    id: latitude
    text: src.position.latitude }
Label {
    id: longitude
    text: src.position.longitude}
```

Sensors API

- API для доступа к сенсорам:
  - Поддержка большинства известных сенсоров
  - Интерфейсы для добавления нестандартных сенсоров
  - Встроенные классы по распознаванию и управлению сенсорными жестами
  - Фильтры для асинхронного обращения к данным сенсоров

Sensors API - основные типы

- **Accelerometer** - генератор уведомлений
  - reading
  - active
  - onReadingChanged
- **AccelerometerReading** - структура, в которой лежат проекции ускорения на декартовы оси.

Sensors API - demo (акселерометр)

Редактируем **qml/pages/FirstPage.qml**

1. Внедряем в код страницы **Accelerometer.**
2. Добавляем новые **Label** для x, y, z.
3. Указываем id=accel для Accelerometer.
4. Указываем для **x.text = *accel*.reading.x** и по аналогии для **y** и **z.**

Sensors API (акселерометр) - код

```
Label{ text: "x:"+accel.reading.x  }
Label{ text: "y: "+accel.reading.y }
Label{ text: "z:"+accel.reading.z  }
 Accelerometer{
            id: accel
            active: true
 }
```

Sensors API - demo (магнитометр)

```
Label {text: "x:"+mangetometer.reading.x }
Label {text: "y:"+mangetometer.reading.y }
Label {text: "z:"+mangetometer.reading.z}
Magnetometer{
    active: true
    id: mangetometer}
```

Canvas - описание

- Тип данных для двумерного рисования в заданной области (http://doc.qt.io/qt-5/qml-qtquick-canvas.html#paint-signal)
- Аналог HTML5 Canvas
- Взаимодействие с типом Context2D (http://doc.qt.io/qt-5/qml-qtquick-context2d.html)
- Поддерживает стандарт https://www.w3.org/TR/2dcontext/
- Очень подробный пример - StockQt (встроен в SDK)

Canvas - demo

- Добавляем на FirstPage объект Canvas и два Button (Draw и Clear)
- Canvas{ id: *mycanvas* width: 250 height: 250}
- Добавляем обработчики :

```
        console.log("Draw button was pressed!");
        var ctx = mycanvas.getContext("2d");
        ctx.fillStyle = Qt.rgba(1, 0, 0, 1);
        ctx.fillRect(0, 0, mycanvas.width, mycanvas.height);
//////////////////////……...
        console.log("Clear button was pressed!");
        var ctx = mycanvas.getContext("2d");
        ctx.reset();
```

- Запускаем и видим, что ….

Canvas - demo

- … ничего не работает.
- Почему?

```
Button {
    text: "Draw"
    onPressed: {
        var ctx = mycanvas.getContext("2d");
        ctx.fillStyle = Qt.rgba(1, 0, 0, 1);
        ctx.fillRect(0, 0, mycanvas.width, mycanvas.height);
        mycanvas.requestPaint();
    }
}
```

Bluetooth - описание

- **BluetoothDiscoveryModel** - модель bluetooth-устройства или сервиса.
  - **discoveryMode** - режим поиска устройств/сервисов (FullServiceDiscovery, MinimalServiceDiscovery, DeviceDiscovery).
  - **onDeviceDiscovered(device)**
  - **onErrorChanged(BluetoothService service)**
- **BluetoothService** - описание сервиса + описание устройства.
- Необходимая зависимость - **qt5-qtconnectivity-qtbluetooth**
- **http://doc.qt.io/qt-5/qtbluetooth-qmlmodule.html**

Bluetooth - демо

```
BluetoothDiscoveryModel{ // Аналог PositionSource
    id: myDiscoveryModel
    running: true
    discoveryMode: BluetoothDiscoveryModel.DeviceDiscovery
    onServiceDiscovered: {
        list.text += service.deviceName +" "+service.serviceName;
    }
    onDeviceDiscovered: {
        list.text += device;
    }
}
```

Воспроизведение аудиозаписей - описание

- **SoundEffect** - класс для воспроизведение wav-файлов с низкой задержкой.
- **Audio, MediaPlayer** - воспроизведение различных форматов (wav, mp3)
- Пример wav-файла для демо http://download.wavetlan.com/SVV/Media/HTTP/sample26.wav

Воспроизведение аудиозаписей - demo 1

- Создаем компонент с SoundEffect внутри

```
Component {
    id: effectComponent
    SoundEffect {
        id: effect
        source: "/usr/share/multimedia/qml/pages/explosion.wav"
        muted: false
```

- Добавляем обработчики для удаления объекта после окончания воспроизведения

```
        Component.onCompleted: effect.play();
        onPlayingChanged: if(!playing)  effect.destroy();  } }
```

Воспроизведение аудиозаписей - demo 2

- Добавляем функцию для воспроизведения, которая создает SoundEffect

```
function playSound(){  effectComponent.createObject(page);  }
```

- Добавляем кнопку воспроизведения

```
Button {
   text:"Play"
   onClicked: page.playSound();
 }
```

- Добавляем

PkgConfigBR: - gstreamer-0.10

Location API - описание

- API для геокодирования (прямого и обратного) и навигации.
- Возможности (provider-agnostic):
  - Геокодирование.
  - Отрисовка карт с пользовательскими слоями и подложками.
  - Построение маршрутов с ограничениями.
  - Points of Interest:
    - Отображение изображений, рейтингов, отзывов.
    - Категоризация.
    - Поиск.
    - Рекомендации.

Location API - классы

- **Map** - область отображения карты.
- **Plugin** - источник картографических данных.
- Пакеты:
  - qt5-qtdeclarative-import-positioning
  - qt5-qtdeclarative-import-location
  - qt5-plugin-geoservices-osm
- Устаревший, но полезный пример https://github.com/b0bben/SailfishOS_MapTutorial

Location demo 1

- Для корректного отображения элемент Map размещаем внутри Rect

```
Rectangle {
    id: rect
    anchors.fill: parent
    Map {
        id: map
        anchors.fill: parent
        gesture.enabled: true
    }
}
```

Location demo 2

- Объявляем плагин

```
plugin : Plugin {
    id: plugin
    allowExperimental: true
    preferred: ["osm"]
    required.mapping:   Plugin.AnyMappingFeatures
    required.geocoding:    Plugin.AnyGeocodingFeatures
}
```

Location demo 3

- Настраиваем обновление центра карты при смене позиции

```
PositionSource {
    id: positionSource
    updateInterval: 100
    active: true
    onPositionChanged: {
        console.log(positionSource.position.coordinate);
        map.zoomLevel = 12;
        map.center = positionSource.position.coordinate;
    }
}
```

libsailfishapp

- libsailfishapp
  - Рекомендуется к использованию разработчиками приложений
  - Содержит:
    - свои фабрики для стандартных классов QT (application, view)
    - интерфейсы для получения путей к файлам
    - интерфейсы интернационализации

Ссылки

1. https://sailfishos.org/wiki/Qt
2. https://sailfishos.org/develop/docs/libsailfishapp/
3. https://sailfishos.org/develop/docs/
4. **https://github.com/Acce0ss/advcompogallery-sailfish**

Спасибо за внимание!

Вопросы?