Разработка приложений на платформе MeeGo

Васильев Валентин

Содержание лекции

1. Синхронные/асинхронные интерфейсы
2. Коммуникация приложений с помощью QtDBus
3. Графический интерфейс мобильных приложений (Handset UI)
4. Дополнительные библиотеки (Platform API)
   - Библиотека meego-touch
5. Qt Quick
   - QML
   - Примеры

- В некоторых случаях MeeGo SDK предоставляет синхронные (блокирующие) и асинхронные интерфейсы.
- Пример, работа с контактами:
  - QContactManager::contacts — QContactFetchRequest
  - QContactManager::relationships — ContactRelationshipFetchRequest
- Для организации асинхронного взаимодействия в Qt существуют сигналы/слоты
  - QObject::connect(obj1, SIGNAL(methodOUT()), obj2, SLOT(methodIN()));

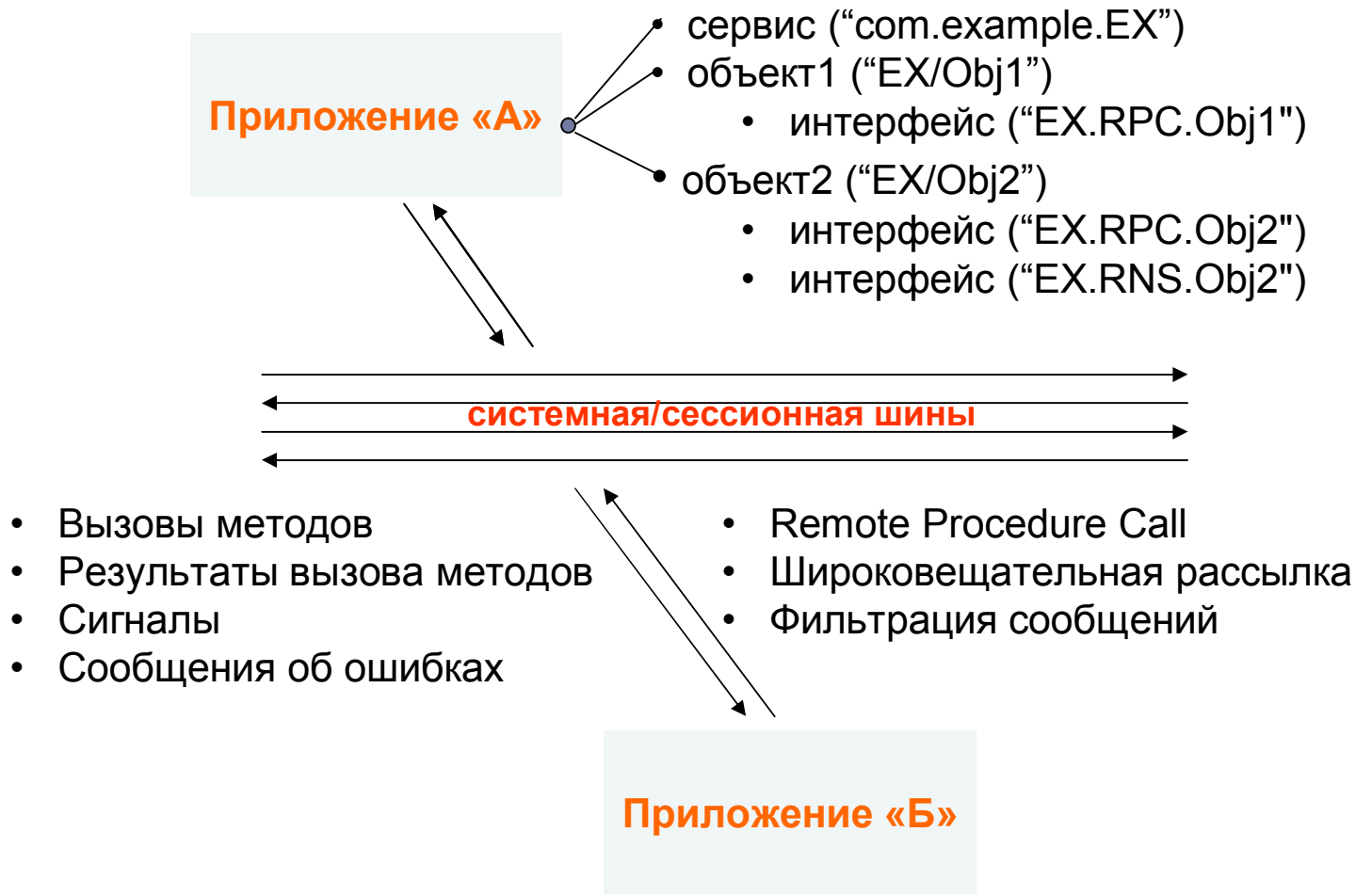
Запись аудио

```
QAudioCaptureSource audiosource;
QMediaRecorder capture ( *audiosource );

capture.setOutputLocation( QUrl( "test.raw“ ) );

capture.record(); //неблокирующее действие
```

```
capture.stop();
```

- Кроссплатформенный свободный (GPL, AFL) механизм для организации коммуникации между приложениями
- Разрабатывался для среды GNOME, но впоследствии стал использоваться в KDE-4
- Для идентификации объектов используются пути, именуемые в стиле Unix
- Пример: сам D-BUS доступен по адресу “/org/freedesktop/DBus”

Разработка приложений на платформе MeeGo
Взаимодействие приложений/Схема



- Существуют официальные рекомендации и требования по разработке приложений для handset-устройств с которыми желательно ознакомиться перед началом разработки (http://meego.com/developers/ui-design-guidelines)

- Основные принципы:
  - многозадачность целевого устройства
  - использование экранной и подключаемой клавиатуры
  - простота использования
  - настраиваемость
  - использование фреймворка Qt, Qt Quick

Разработка приложений на платформе MeeGo
Platform API

- Дополнительные библиотеки не входящие в состав ядра интерфейсов (MeeGo Core API), но являющиеся частью MeeGo API

- Возможно, но не рекомендуется использовать для создания приложений

- Примеры:
  - GStreamer 0.10
  - PulseAudio 0.9.21
  - Web Runtime 1.2 (Preview)

- Для каждой библиотеки существует аналог из MeeGo Core API

Qt, Qt Quick

- разработка сразу для всех типов устройств (телефоны, нетбуки, и т.д.)

- максимальная мобильность и стабильность API
- портирование и разработка на не-MeeGo системах (Windows, Mac OS X, ...)
- использование Qt и Qt Quick — официально признанный путь развития MeeGo

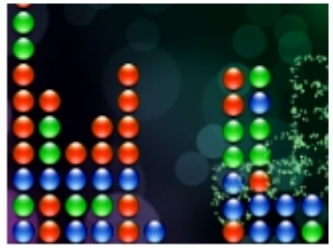
MeeGo Touch Framework

- Средство разработки GUI для handset-версии MeeGo
- Не входит в состав MeeGo Core API, но всё ещё является частью MeeGo Platform API
- Официально не поддерживается начиная с релиза MeeGo API 1.1
- Построен на базе стандартной библиотеки Qt Graphics View
- Предоставляет все необходимые компоненты для создания приложений под устройства с тачскрином

- пользовательский интерфейс специально для устройств с тачскрином
- заинтересованность в использовании возможностей MeeGo на более низком уровне
- фреймворк продолжает развиваться и есть высокая вероятность, что часть его интерфейсов войдёт в будущие релизы MeeGo Core API

Qt Quick

- средство быстрого создания, прототипирования приложений, интерфейсов и отдельных элементов управления
- пользовательский интерфейс и его поведение описывается с помощью языка QML
- создание компонентов в диапазоне от простых кнопок и ползунков
- технология может использоваться как для использования вместе с C++, так и для создания приложений с нуля
- всё доступное взаимодействие производится через модуль QtDeclarative



QML

- гибридный язык, описывающий интерфейс и поведение приложения
- слаботипизированный, скриптовый (интерпретатором выступает модуль QtDeclarative)

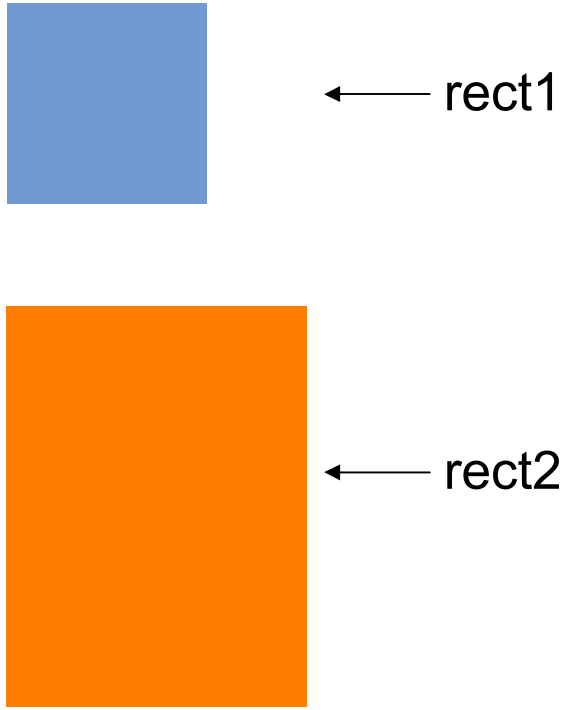
*.qml

- описывает дерево объектов со свойствами
- предоставляет графические элементы (области, прямоугольники и т.д.) и блоки, задающих поведение — переходы, анимацию, состояния
- расширяем по средствам C++

```
import QtQuick 1.0

Rectangle {
    width: 200
    height: 200
    color: "blue"

    Image {
        source: "pics/logo.png"
        anchors.centerIn: parent
    }
}
```

Разработка приложений на платформе MeeGo
QML / Пример 2

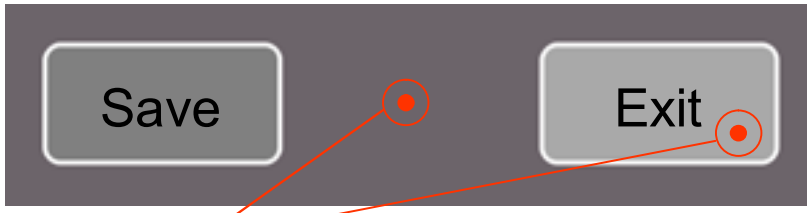
```
Item {
    Rectangle {
        id: rect1
        width: 100
        height: 100
        color: "blue"
    }
    Rectangle {
        id: rect2
        width: rect1.width+100
        height: 200
        color: “red"
    }
}
```



SPRINT SPbSU-Intel Laboratory

Разработка приложений на платформе MeeGo
Handset UI / Пример 3

```
Row{
        Button{
            id: saveButton
            buttonColor: “darkgrey"
            label: “Save"
        }

        Button{
            id: exitButton
            label: "Exit"
            buttonColor: “lightgrey"

            onButtonClick: Qt.quit()
        }

        MouseArea {
            onPressed:
                if (mouse.button == Qt.RightButton)
                    console.log("Right button")
        }
}
```



SPRINT
SPbSU-Intel Laboratory

Использование функций javascript напрямую

```
Item {
    function factorial(a) {
        a = parseInt(a);
        if (a <= 0)
            return 1;
        else
            return a * factorial(a - 1);
    }

    MouseArea {
        anchors.fill: parent //установить геометрию как у родителя
        onClicked: console.log(factorial(10))
    }
}
```

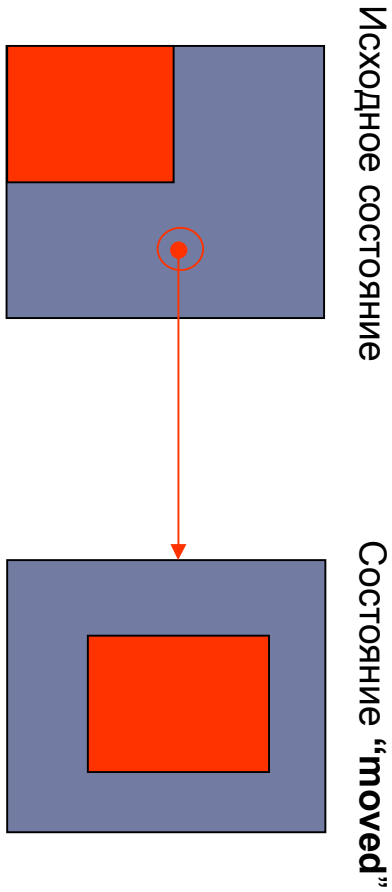
Вызов функций из файла

```
import "factorial.js" as MathFunctions
Item {
    MouseArea {
        anchors.fill: parent
        onClicked: console.log(MathFunctions.factorial(10))
    }
}
```

- Приложение написанное на чистом Qt Quick представляет собой набор *.qml, *.js файлов.

Передвижение прямоугольника

```
Item {
    id: myItem

    Rectangle {
        id: myRect
        x: 0; y: 0
    }
    states: [   State {
                        name: “moved”
                        PropertyChanges {
                            target: myRect
                            x: 50; y: 50
                } } ]
    MouseArea {
        anchors.fill: parent
        onClicked: myItem.state = “moved”
} }
```



Вращение относительно оси

```
Image {
    id: image1
    source: "images/qt-logo.svg“
    anchors.fill : parent

    transform: Rotation {
        origin.x : 30; origin.y : 30
        axis { x:1; y:0; z:0 }
        angle : 0
        NumberAnimation on angle  {
            from: 0; to: 72;
            duration: 3000;
            loops: Animation.Infinite
        }
    }
}
```

Вопросы?

Валентин Васильев (gnome@bk.ru)