Multiplatform szoftverfejlesztés

Qt alapok

Mi a QT?

- Teljes alkalmazás fejlesztői keretrendszer
  - C++ osztálykönyvtár
  - GUI
    - De nem csak az – hálózat, SQL, unit test, multimédia, stb.
  - QT Creator
    - Kód szerkesztő
    - Fordító választható (VS, GCC, Clang, stb.)
    - Debugger (QML debugger is)
    - Source Control integrált
  - Multiplatform támogatás
  - Dokumentáció, tutorials, példakódok

Felhasználása

- Mindenen fut, ami programozható
  - Desktop (Windows, Linux, Mac OS) és szerverek
  - Beágyazott rendszerek
  - Mobil eszközök (iOS, Android, …)
- Elterjedt
  - Skype, Spotify, VLC, Maya, Google Earth
- Licence
  - Ingyenes, LGPL
  - Fizetős/open source, ha a teljes csomag kell (pl. Qt Charts)
  - Qt módosítása csak akkor lehetséges, ha open source a módosítás

Qt C++ Hello World

```
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QLabel label("Hello World!");
    label.show();
    return app.exec();
}
```

Qt Quick

- QML – deklaratív felület leírás, ami kódot is tartalmazhat
- Vezérlők gyűjteménye
- Model-View támogatás
- Animációs keretrendszer
- JS futtató motor
- Minden platformon natív kinézet
  - De meg is adható a stílus

QML betöltő

- QQmlApplicationEngine
  - Akárhogyan létre lehet hozni (vermen is)
  - Destruktor töröl mindent
- Load, vagy konstruktor létrehozza az objektumokat
- QUrl a resource fájlba tud hivatkozni
  - Sima fájlt is be lehet tölteni

```
#include <QApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QQmlApplicationEngine
engine(QUrl("qrc:/main.qml"));
    return app.exec();
}
```

QML Hello World

```
import QtQuick 2.4
import QtQuick.Controls 1.3
import QtQuick.Window 2.2
import QtQuick.Dialogs 1.2

ApplicationWindow {
    title: "Hello World"
    width: 640
    height: 480
    visible: true
}
```

Qt Quick

Vezérlők

QML nyelv

- Felület deklarálására
  - Miért jó deklaratívan megadni felületet?
- JavaScript Object szerű (nem JSON)
  - Egyszerű
  - Picit kényelmesebb, mint XML
- Struktúra megegyezik a hierarchiával
  - Szinte mindig
- Tartalmazhat kódot, JavaScript

QML vezérlők

- Típus (Button)
- Tulajdonságok (text)
- Értékek (“hello”)
- Bármibe tehetünk bármit
  - Buttonban Rectangle
  - A Rectangle itt rárajzol a Button szélére
- Sok hasonlóság XAML-lel
  - Egyszerűbb (jó és rossz is)
  - Gyorsabb/kevesebb memóriát eszik

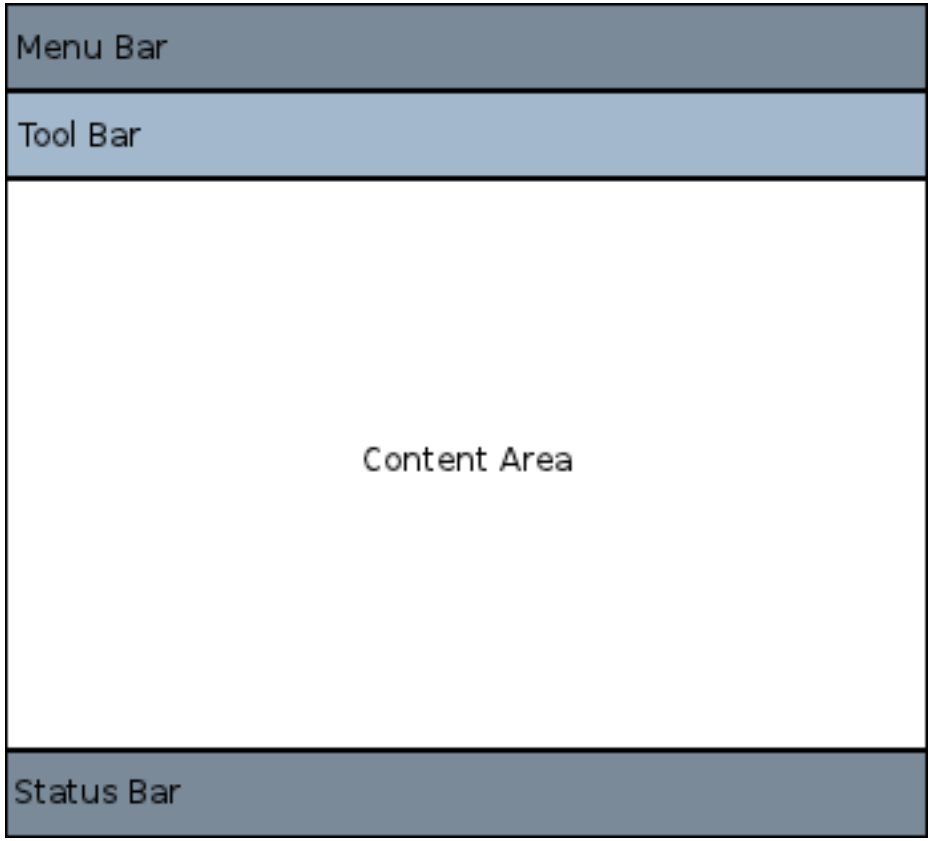
```
Button{
    text: "hello"
    Rectangle{
        width: 10
        height : 10
        color : "red"
    }
}
```

QML alaptípusok

- Fontosabbak
  - bool, int, double, string, list, url
  - color, date, font
  - point, rect, size,
  - vector2d, vector3d, vector4d, matrix4x4
- Vannak ezeknek is tulajdonságai, de nem generálnak eseményt, ha változnak
  - vector4d.x nem generál, de vector4d igen

QML vezérlők - ablak

- ApplicationWindow – felső szintű ablak
  - Fontosabb tulajdonságai: title, color, width, height, visible
- MenuBar - menü
- ToolBar
  - ToolButton, stb.
- StatusBar – státusz sor



QML vezérlők - Label

- Szöveg kiírása
- Text-ből származik, ami mindent tud
  - Label annyiban más, hogy a operációs rendszer alapértékeit használja
- Text
  - Fontosabb tulajdonságok: text, font, color, elide, horizontalAlignment, verticalAlignment, wrapMode
  - textFormat
    - PlainText – azt írja ki, ami benne van
    - StyledText – HTML 3.2
    - RichText – HTML 4, lassú

QML vezérlők - Button

- Fontosabb tulajdonságok: text, isDefault, pressed
  - tooltip: ha a platform támogatja
  - style: stílus megadása
- Benyomva maradó gomb: checkable, checked
- Dropdown: menu
- Benyomásra kikapcsol más gombot: exclusiveGroup
  - Ez a RadioButtonnál lényeges
- Signal: clicked()

QML vezérlők – RadioButton

- Buttonnal is meg lehet csinálni
  - A platform style elvész

```
ColumnLayout {
    ExclusiveGroup { id: tabPositionGroup }
    RadioButton {
        text: "Top"
        checked: true
        exclusiveGroup: tabPositionGroup
    }
    RadioButton {
        text: "Bottom"
        exclusiveGroup: tabPositionGroup
    }
}
```

QML vezérlők

- CheckBox
  - text, pressed, checked, style
  - checkedState, partiallyCheckedEnabled, exclusiveGroup
- ProgressBar
  - value, maximumValue, minimumValue
  - indeterminate, orientation, style
- Slider
  - value, maximumValue, minimumValue, stepSize
  - tickmarksEnabled, style, orientation, pressed

QML vezérlők

- SpinBox (numeric up down)
  - value, maximumValue, minimumValue, stepSize
  - decimals, font, horizontalAlignment, style
  - prefix, suffix
  - editingFinished()
- Switch (toggle button)
  - checked, exclusiveGroup, pressed, style, clicked()
- Calendar
- BusyIndicator

QML vezérlők – TextField (input)

- text, placeholderText, inputMask, readOnly
- Jelszó támogatás: echoMode, displayText
- Szerkesztés
  - canPaste, canUndo, canRedo
  - length, maxLength
  - selectedText, selectionStart, selectionEnd
- Kinézet
  - font, textColor, style
  - horizontalAlignment, verticalAlignment
  - cursorPosition, cursorRectangle

QML vezérlők – TextField (input)

- Szótár támogatás – mobil eszközökön
- Több billentyűzet lehetséges
- Van automata korrekció is
- inputMethodHints
  - Qt.ImhSensitiveData – ne tárolja a szótárban
  - Qt.ImhNoAutoUppercase – első karakter ne legyen nagy
  - Qt.ImhPreferNumbers – más billentyűzet jöhet elő
  - Qt.ImhPreferUppercase, Qt.ImhPreferLowercase
  - Qt.ImhDate, Qt.ImhTime
  - Qt.ImhMultiLine – enterre ne záródjon be a billentyűzet
  - Qt.ImhEmailCharactersOnly, Qt.ImhUrlCharactersOnly
  - Qt.ImhDigitsOnly – csak számok
  - Qt.ImhFormattedNumbersOnly – számok, -, .

QML vezérlők – TextField (input)

- Validálás
  - validator: Int, Double, RegExp
  - editingFinished() signal enter után mindig jön
  - accepted() signal csak akkor jön, ha valid

```
TextField {
    validator: IntValidator {bottom: 1; top: 10;}
}
```

QML vezérlők – TextField (input)

- Függvények
  - copy(), paste(), undo(), redo()
  - selectAll(), selectWord(), select(start, end), deselect()
  - insert(pos, text), getText(start, end)

QML vezérlők - TextArea

- Többsoros szövegszerkesztő
- Sima szöveg, vagy Rich Text
- Tab és Enter
- Tördelés: NoWrap, WordWrap, WrapAnywhere, Wrap
- Syntax highlighting
  - textDocument tulajdonság
  - C++-ban lehet csak megírni
  - Van kiindulási alap: QSyntaxHighlighter

QML vezérlők – GroupBox

- Keret egy szöveggel
- Lehet CheckBox is a szöveg, letiltja a belső vezérlőket, ha nincs pipálva
- Flat stílusú is lehet
  - Minden eszközön más
  - Mobilon bal-jobb-alsó keret eltűnik (stílus függő)

Qt Quick

Alapok

QML Item

- Felület alapegysége
- Szinte minden ebből származik
- Hierarchia itt van implementálva
  - children: list<Item> - látszó gyerekek
  - resources: list<Object> - nem látszó gyerekek
  - data: list<Object> - összes gyerek, default property
- Ide kerül, amit minden vezérlő tud
  - Pl. enabled, parent

QML Item

- Megjelenítés
  - Opacity, antialiasing, clip, smooth
- Input fókusz
  - Focus, activeFocus, activeFocusOnTab
- Méret és pozíció
  - visible, x, y, z, width, height, rotation, scale, transform, transformOrigin
  - Nem 3D-s (nincs depth tulajdonsága)
  - De 3D térben van (x, y, z, és forgatni is lehet térben)

QML Item - anchors

- Automata pozícionálás
- Lehet horgonyozni
  - Nem csak szülőhöz, hanem testvérhez is
  - top, bottom, left, right, fill
  - topMargin, bottomMargin, leftMargin, rightMargin
  - A Margin hozzáadódik a kiszámolt értékhez

```
Item {
    Image {
        id: pic
    }
    Text {
        anchors.top: pic.bottom
        anchors.topMargin: 5
    }
}
```

QML Item - anchors

- A fill csak egy egyszerűsítés, hogy ne kelljen 4-et beállítani
- alignWhenCentered: fél pixelre nem teszi
  - Amikor kell: statikus elrendezés, éles kép
  - Amikor nem kell: Animált, folytonos mozgás
- BaseLine igazítás
  - baslineOffset: Hol van a baseline ebben a vezérlőben
  - achors.baseLine: A másik vezérlő baseline-ja
  - achors.baseLineOffset: Igazítás finomhangolása

QML Item - anchors

- Középre igazítás
  - anchors.centerIn: kényelmi beállítás a kettő helyett
  - anchors.horizontalCenter
  - anchors.verticalCenter
  - anchors.horizontalCenterOffset
  - anchors.verticalCenterOffset

QML Item

- Szintén Item-ben implementálták
  - Állapotok és átmenetek
  - Animációs alrendszer
  - Post process effectek

Saját vezérlő

- Létrehozunk egy .qml fájlt
- Nincs megkötés a gyökérelemre
  - Célszerű Itemet használni minimum (layout logikát tudja)
- A vezérlő neve a fájl/erőforrás neve lesz
  - Vagy ha külső fájl, akkor importálni kell

```
import QtQuick 2.0
Item {
    id: root
    Rectangle{
        anchors.fill: parent
        color: "white"
    }
}
```

Qt Quick

Layout

QML layout

- Column és Row
  - Egymás alá/mellé teszi a gyerekeit
  - spacing: megadható mennyi helyet hagyjanak ki
  - Nincs hátterük
- Grid
  - column tulajdonsága van csak (nincs row)
  - Előbb egymás mellé teszi őket, majd következő sorba
  - Ha üres cellára van szükség, akkor oda kell tenni valamit, ami nem rajzol

QML layout

- Flow
  - Hasonló Gridhez
    - De nincs oszlopszám megadva
    - Nincsenek igazítva egymás alatt az elemek
  - Addig teszi a vezérlőket egy sorba, amíg van hely, utána új sort nyit – hasonló a HTML layout logikához

QML layout

- GridLayout
  - Méretezi az elemeket (nem úgy, mint Grid)
  - XAML Grid, CSS grid szerű működése van
  - Automata cellakiosztás
    - Flow tulajdonság adja meg, hogy milyen sorrendben használja fel az elemeket, ha nincs megadva sor/oszlop
    - GridLayout.LeftToRight az alap, ekkor columns tulajdonság adja meg az oszlopok számát
    - GridLayout.TopToBottom esetén rows-t kell beállítani

QML layout

- GridLayout
  - Cella megadható kézzel is attached property-kel
    - Layout.row, Layout.column
    - Layout.rowSpan, Layout.columnSpan
  - Méret megkötéseket adhatunk
    - Min, max, prefered – width, height
  - Kitöltés: Layout.fillWidth és Layout.fillHeight
  - Igazítás cellán belül: Layout.alignment

QML layout

- RowLayout és ColumnLayout
  - Az algoritmus ugyanaz, mint a GridLayout
  - Picit kényelmesebb megadni, ha csak egy sor/oszlop van

QML layout

- Ablakhoz kötés
  - Ha az ablak méretezhető (pl. PC)
  - Tipikusan egy layout konténer az ApplicationWindow alatti első elem
    - Állítsuk be, hogy töltse ki az ablakot: achors.fill: parent
  - Az ablakon
    - width: layout.implicitWidth
    - height: layout.implicitHeight
    - minimumWidth: layout.Layout.minimumWidth
    - minimumHeight: layout.Layout.minimumHeight
    - Ha van max a layouton, akkor azokat is be kell állítani

Qt Quick

Események

Események – Signal

- Komponens definiálja
  - Sajátot is létrehozhatunk
  - Lehet C++ és QML is
- Fel lehet rá iratkozni
  - Bárhonnan, ha az esemény láthatósága megengedi
  - A feliratkozás helyén nem látszik, hogy mik a paraméterei, de a deklarálásnál igen (és a doksi is megadja a beépítettekhez)

Események – Signal Handler

- Az objektumban deklarálni kell egy Signal Handlert
  - on<Signal> a neve, ahol <Signal> az esemény neve (nagybetűvel)
  - Értéke a JS kód
  - Kívülről és C++-ból is fel lehet iratkozni rá, de ez a szintaktika a legkényelmesebb

```
Button{
    text: "hello"
    onClicked : { rect.color = "blue"}
}
```

Események – Property Change

- Minden tulajdonsághoz automatikusan tartozik egy Property Change Signal
  - Az alaptípusok tulajdonságaihoz nem
  - A doksiban ezek külön nincsenek benne, de mindenhez van
  - Változás után hívódik meg, az értékét simán a tulajdonság lekérdezésével tudhatjuk meg
- Feliratkozni az on<Property>Changed deklarációval lehet
  - <Property> a tulajdonság neve (nagybetűvel)

Események – Attached Signal Handler

- Más forrásból is kaphatunk eseményt, ha az definiál Attached Signalt
- Példa a Component objektum, aminek a completed eseménye mindenhonnan elérhető

```
Rectangle{
    width: 200; height: 200
    Component.onCompleted : {
        console.log("The rectangle's color is", color)
    }
}
```

Események – Timer

- Időzítő
- Triggered signal
- Animációt nem ezzel csinálunk

```
Timer{
    interval: 500; running: true; repeat: true
    onTriggered : time.text = Date().toString()
}
Text{ id: time }
```

Események – saját Signal

- Saját vezérlőben signal kulcsszó
- Eseményt generálni a meghívásával lehet, mintha függvény lenne (nem az)

```
Item {
    id: root
    signal clicked ( )
    Rectangle{
        anchors.fill: parent
        color: "white"
    }
    MouseArea{ anchors.fill: parent; onClicked: root.clicked()}
}
```

Események – saját Signal

- Feliratkozni a szokásos módon lehet
- Tegyük fel, hogy a saját vezérlő típusneve ClickRect

```
ClickRect {
    onClicked: console.log("Hello")
}
```

Események – connect()

```
Item {
    id: root
    signal clicked ( )
    Rectangle {
        anchors.fill: parent
        color: "white"
        Component.onCompleted: {
            root.clicked.connect(clickThisToo)
        }
    }
    function clickThisToo(){ }
    MouseArea{ anchors.fill: parent; onClicked: root.clicked() }
}
```

- Több függvényt is rá lehet kötni egy signalra
  - A Handler mellett
  - Másik signalt is

Kérdések?