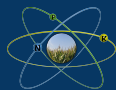




Développement mobile avec Qt / Qml



1. Le marché des applications mobiles



2. L'environnement technologique

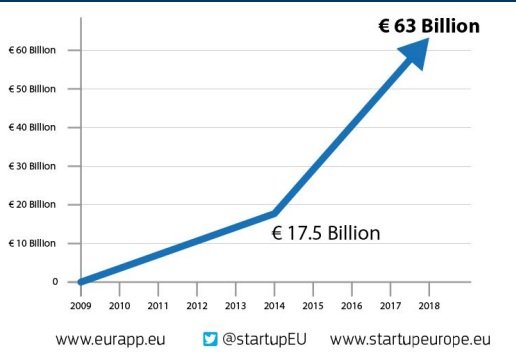


3. Présentation Qt, QtQuick, QML

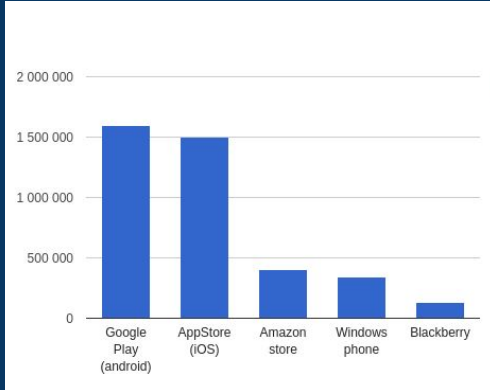
4. Démonstration



Le marché du DÉVELOPPEMENT D'APPLICATION



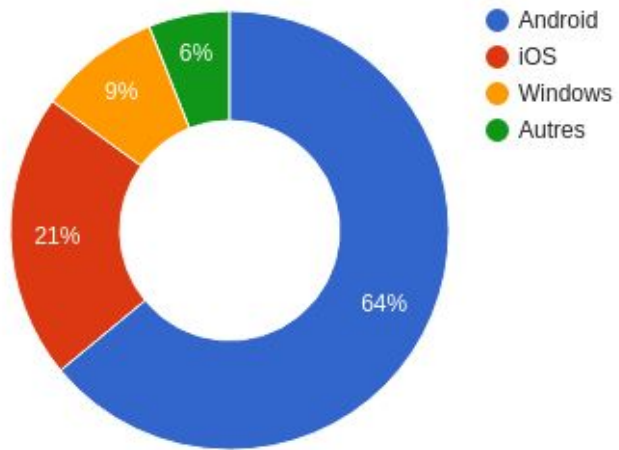
Bienvenue au far-west !



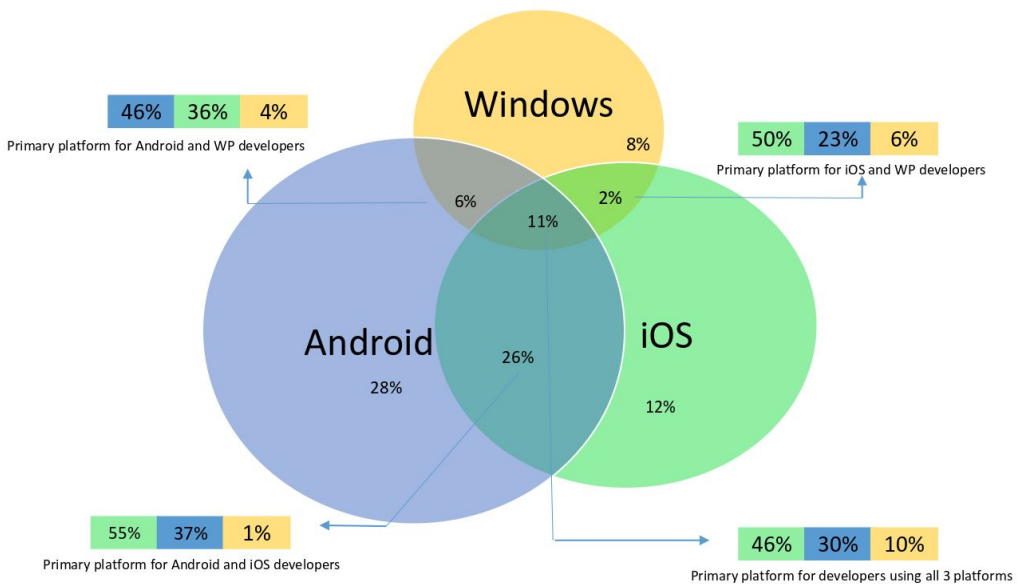
NASDAQ	+0.5	NYSE	-0.7
GOOG	+3.3	FB	-2.3
AAPL	+1.5	NWSA	-3.1
ASND	+2.7	SNE	+2.2
RDSA	-6.2	MSFT	+4.2

Le marché du développement d'application

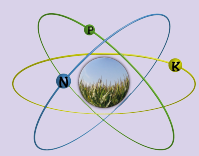
Répartition des smartphones par système d'exploitation



© Mobile Marketing Association France – aout 2015



Survey source: Vision Mobile – “Cross-platform tools 2015”



L'environnement TECHNOLOGIQUE

WEB

accessible depuis
un explorateur

Web mobile

Site internet optimisé pour le mobile
Adaptative design / Responsive design

Web App

site web avec look&feel d'application

Hybrid

Contenu dans une application

Native CPT

APP

Accessible depuis
un magasin

Native



BlackBerry



JQuery, Vaadin,
Angular JS, Ember,
Backbone.js,
React/Flux

PhoneGap (Apache Cordova), Rho
Mobile, Ionic, Sencha Touch, Intel SDK,
Onsen UI, JQuery mobile, Mobile
Angular UI...

Qt (C+, QML, JS)
Xamarin (C#)
Appcelerator Titanium (JS)
Ruby motion (Ruby)
Unity (C#,BOO)
Marmalade (*)

Java

Objective C / swift

C#

Déploiement immédiat

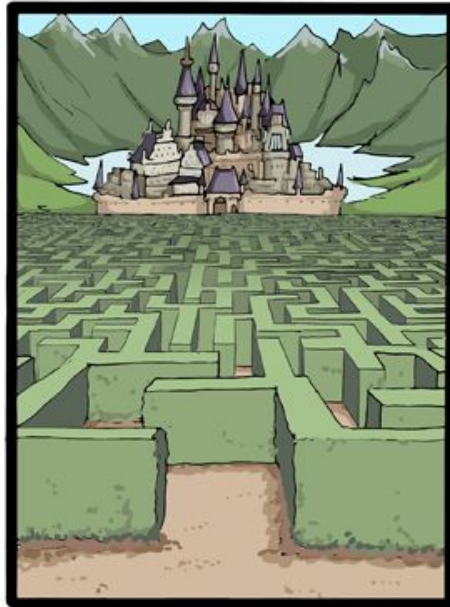
Fonctionnalités limitées

Fonctionnalités limitées

Coût au développement et
à la maintenance pour
plusieurs plate-formes

Le DILEmme...

Développer une app native
par plate-forme et maintenir
3 codes-sources



Utiliser une techno unique
(Phonegap, Adobe Air,
Appcelerator) et maintenir
1 seul code-source



CommitStrip.com

Benchmark CPT

research2guidance 17: Compatible programming skills by CP Tool and service

Tools	No programming skills	Java Script	HTML	CSS	Java	Lua	C++	C	Other	Comments
Appcelerator Titanium										
Embarcadero Appmethod									Pascal	
Marmalade									Objective C	
NeoMAD									XML	
Qt									QML	
Smartface App Studio										
Unity									C# Boo	Users choose between C#, Javascript & Boo
V-Play									QML	QML & Java Script for scripting with Qt Quick; Mixing with C++ possible
Xamarin									C#	

BENCHMARK CPT

Value for money: 81% are satisfied with their Cross-Platform Tool cost-performance

Top 10 Cost-performance ratio

Rank	Tool	Poor value or costly	Average	Okay or good value	# Ratings
1	Qt	-2%	0%	98%	104
2	Titanium	-6%	2%	92%	51
3	Unity	-4%	5%	91%	103
4	Corona SDK	-7%	2%	91%	97
5	Windows Visual Studio	0%	16%	84%	64
6	Cocos 2D	0%	17%	83%	54
7	Adobe Air	-4%	13%	83%	82
8	Xamarin	-7%	13%	80%	99
9	PhoneGap	-3%	17%	80%	88
10	KonyOne	-11%	11%	78%	55
Benchmark (Average all tools)		-5%	14%	81%	

1 Indie developer

- 1 app developer
- 1 app project
- Duration: 1 year
- Publishing on 2 platforms
- Deployment to commercial app stores
- No additional support requested

2

App studio / agency

- Company with 10 developers
- 10 app projects
- Duration: 1 year
- Publishing on 4 platforms
- Deployment to commercial app stores
- Advanced support requested: e.g. tutorial/ training, email and phone support

3

Enterprise

- 20 app developers
- 50 app projects
- Duration: 1 year
- Publishing on 4 platforms
- Deployment of 25 apps to commercial app stores and 25 internal
- Advanced support services requested: e.g. on-site project support, initial training/ tutorial and other


















Tools	Scenario 1:	Scenario 2:	Scenario 3:
	Costs	Costs	Costs
CP IDEs			
Embarcadero Appmethod	\$ 598	\$ 30.590	\$ 66.175
Marmalade	\$ -	\$ 15.000	\$ 70.000
NeoMAD	\$ 675	\$ 13.500	\$ 27.000
Qt	\$ 1.308	on request	on request
Smartface App Studio	\$ -	\$ 97.000	\$ 210.000
Unity	\$ 2.700	\$ 24.000	on request
V-Play	\$ 168	\$ 11.990	\$ 47.980

Benchmark CPT

research2guidance 12: Output by CP Tool and service

Tools	Native app	Web app	Hybrid app	Mobile web page
Web App Toolkits				
Vaadin				
CP IDEs				
Appcelerator Titanium				
Embarcadero Appmethod				
Marmalade				
NeoMAD				
Qt				
Smartface App Studio				
Unity				
V-Play				

research2guidance 13: Platform support by CP Tool and service

	mobile															desk-top	
																	
Tools	iOS	Android	WP 8	HTML 5	BlackBerry 10	Kindle Fire	Firefox OS	Tizen	Java ME	Ubuntu	WP 7	BlackBerry 7	Sailfish	Symbian	Bada	Windows	Apple OS X
Web App Toolkits																	
jQuery																	
Vaadin																	
CP IDEs																	
Adobe Air																	
Appcelerator Titanium																	
Corona SDK																	
Embarcadero Appmetho																	
Marmalade																	
NeoMAD																	
PhoneGap																	
Qt																	
Smartface App Studio																	
Unity																	
V-Play																	
Xamarin																	



C'EST QUOI ?

Un framework C++

- Réseau et communication
- Multithreading
- Multimédia et capteurs
- Base de données
- 3D (OpenGL)
- Web
- XML, JSON
- génération PDF...

Une boîte à outils

- IDE sobre et performant
- Debugger et Profiler
- Outils de localisations
- Documentation de qualité
- GUI designer
- Build suite (QBS et QMake)

Des concepts innovants

- QtQuick
- Signals / Slots
- Widget
- Model-View-Delegate

Disponible sous plusieurs modèles de licences

- Propriétaire
- GPL
- LGPL

Une promesse : “Code once, deploy everywhere !”





La PHILOSOPHIE DE QTQUICK

Qt Quick = C++ +



Back-End

Front-End

Intégrer le meilleur des 2 mondes :
la performance du C++ et les
interfaces utilisateurs qui déchirent
avec la simplicité du QML !



C'EST QUOI ?

Evolutable simplement

Un langage déclaratif

Animations

Rendu graphique accéléré
matériellement

Une arborescence de
composant

Bibliothèque de 80
composants

```
import QtQuick 2.0

Rectangle {
    id: page
    width: 320; height: 480
    color: "lightgray"

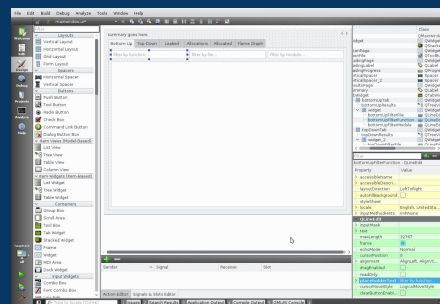
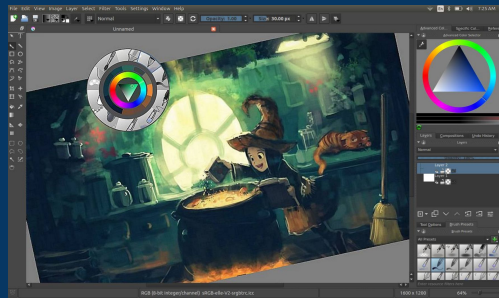
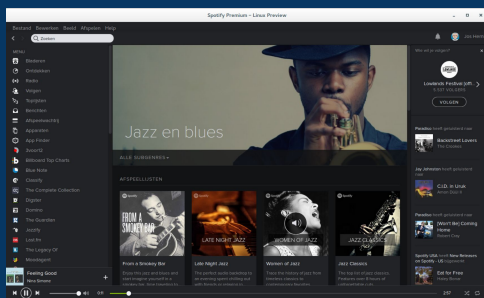
    Text {
        id: helloText
        text: "Hello world!"
        y: 2 * 15
        anchors.horizontalCenter: page.horizontalCenter
        font.pointSize: 24; font.bold: true

        MouseArea {
            anchors.fill : parent
            onClicked: {
                if (true)
                    Console.log(parent.text)
            }
        }
    }
}
```

Peut contenir du Javascript

UI : 3 APPROCHES POSSIBLES...

- Méthode historique, basée sur les QWidgets pour un résultat...contemporain :



Méthode moderne (QtQuick) avec deux variantes :

- composants graphique basique : liberté totale
- contrôles pour un look natif : très rapide





MODULES QML ET ÉLÉMENTS UTILES

QtQML

Binding Component Connections Date Instantiator Locale Number Qt QObject String Timer

Basic

- **Item** (élément non visible, base de tous les éléments graphique)
- **Rectangle**
- **Image**
- **Text**
- **TextEdit**
- **TextInput**

- **Row** (positionneur en ligne)
- **Column** (positionneur en colonne)
- **Grid** (positionne les éléments enfants en grille)
- **Flow** (positionne les éléments enfants à la suite)

- **Flipable**
- **Flickable**

- **MouseArea, MouseEvents, WheelEvents**
- **Shortcut, Key, KeyEvents**
- **PinchArea, PinchEvents**
- **Drag, DragEvent, DropArea**

- **Loader**

- **ListView**
- **GridView**
- **Repeater**

- **ListModel**

- **Animations**
- **States**

Windows, Dialogs, Tests, Layouts, XML List models, Local Storage, Controls

QtQuick

Qt3D

QtAudioEngine

QtBluetooth

QtLocation

QtPositioning

QtMultimedia

QtNfc

QtSensors

QtWebsockets

QtWebview

QtTest

Plusieurs moyens sont disponibles pour faire communiquer C++ et QML :

1. Interfacer des attributs de classe C++ (dérivée de QObject) avec du QML
 - a. Propriétés : <http://doc.qt.io/qt-5/qtqml-cppintegration-exposecppattributes.html>
 - b. signaux : il suffit que le signal soit public
 - c. méthode : il suffit que la méthode soit publique ou d'utiliser Q_INVOKABLE
2. Rendre une classe C++ (dérivée de QObject) disponible en tant que Composant QML :
3. Passer un objet C++ (dérivée de QObject) dans le contexte QML
4. Créer un nouveau composant QML graphique en C++ : il faut dériver le composant d'un QQuickViewItem
5. Récupérer un Item QML à partir de C++



concrètement !

Un visualiseur d'image (contenu RSS Flickr)

Etapes :

1. Créer un projet avec Qt creator et montrer les fichiers générés
2. jouer avec le QML, montrer des éléments simples (Rectangle, Item, layout, image...)
3. Afficher une image et un textedit contenant son url
4. Afficher plusieurs images avec un model QML (statique) et une vue en grille
5. Utiliser un model XML avec le flux RSS de flicker (exemple inspiré par [Jens Bache-Wiig](#))
6. Raffinement : afficher en haute qualité, gérer le chargement, les modes d'affichage, faire un filtre pour gérer le contenu de flicker...



HISTORIQUE DE QT

1990

élaboration de Qt

1995

Troll Tech - et Qt 0.90 (X11/Linux)

1998
1999

KDE free Qt foundation & Qt 2.0

2001

Qt 3.0 - support MacOS X, Qt designer

2005

Qt 4.0 - support Windows

2008

Acquisition par Nokia

2009

Qt Creator (IDE)

2010

Qt Quick et support Symbian

2011

Acquisition par Digia

2012

Qt 5.0 et QPA, Qt Quick 2.0, support WinRT

2014

Création de "The Qt Company"

2015

Qt a 20 ans et plus 800k de développeurs



HISTORIQUE releases QT 5 POUR LES MOBILES

5.0 19/12/2012 : QML, QtQuick, Location (géolocalisation)

5.1 03/07/2013 : Quick controls, Sensors (Android, IOS, Blackberry), Qt for Android (Technology preview), Qt for IOS (Technology preview)

5.2 12/12/13 : Positionning, NFC, Bluetooth, Windows Extra, Mac extra, Android extra, Qt for Android, Qt for IOS

5.3 20/05/2014 : Bluetooth LE (Android only and Technology Preview), Qt for WinRT, Android native style

5.4 10/12/2014 : Bluetooth LE (Android only and Technology Preview), Qt for WinRT, Android native style

5.5 01/07/2015 : QT3D, Location (Route), Canvas 3D, Webview (IOS)

Release TOUS Les 6 mois environ

Merci... !

