C++ Day 2016

29 Ottobre / Firenze

Lo stack grafico di Qt

Luca Ottaviano – lottaviano@develer.com

www.italiancpp.org

Chi sono

Team leader & project manager @Develer

Lavoro con C++ da 11+ anni

Sviluppo con Qt da 8+ anni

Relatore a conferenze da 6 anni

Develer

Azienda di sviluppo software e hardware

Sviluppiamo con tecnologie open source in molti campi diversi:

- Embedded real-time
- Linux e Android
- Desktop multipiattaforma
- Cloud e mobile

Siamo sempre alla ricerca di talenti

We’re hiring!

www.develer.com/jobs/

Agenda

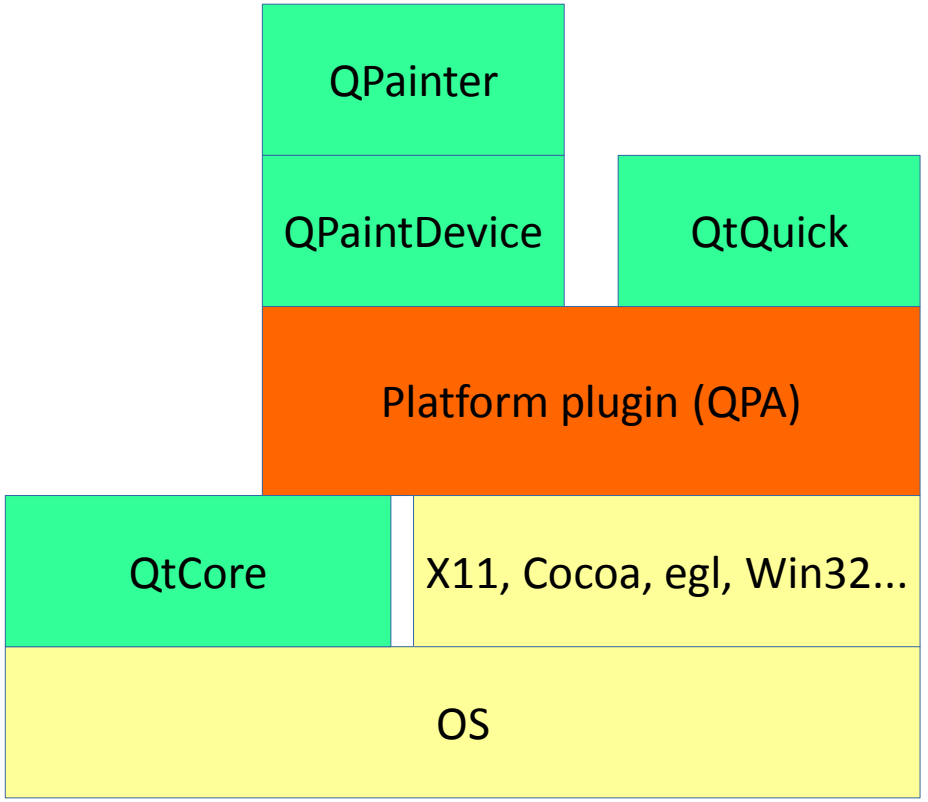
Come si disegna un widget?

Come si crea un pdf?

Come si disegna una scena QtQuick?

Come ci si integra con il sistema operativo?

Uno sguardo d’insieme



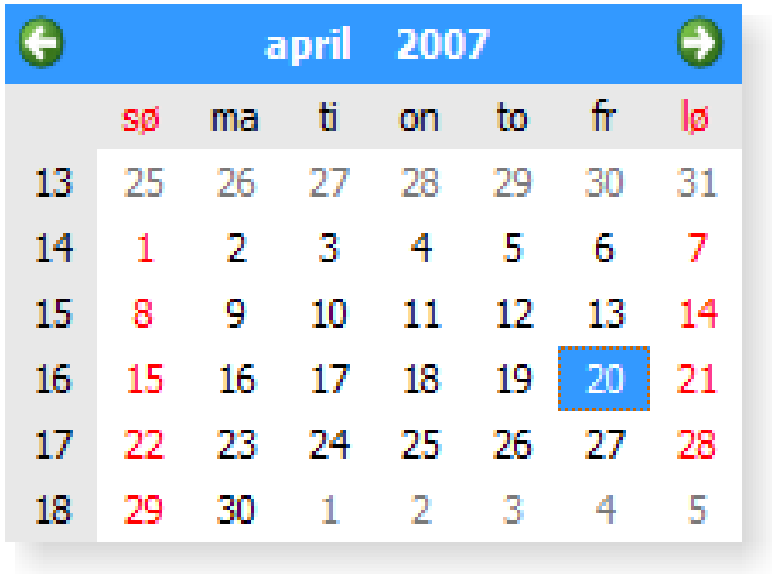
Widget

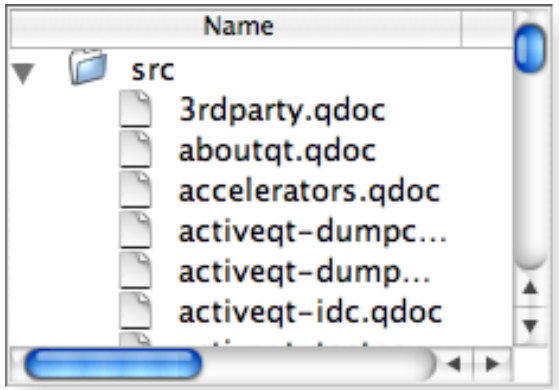
Widget = Form

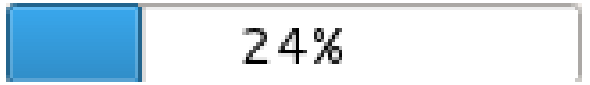
Sono elementi grafici per interagire con l'utente

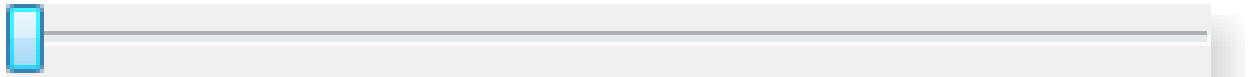
- Button, Label, Checkbox
- Slider, Combobox, Spinbox
- Tab, Docks
- Treeview, Tableview

Widget gallery









Disegnare su un widget

È facile creare widget personalizzati

Possiamo disegnare durante il paintEvent()

Usiamo un QPainter!

QPainter

È la classe che disegna su un canvas

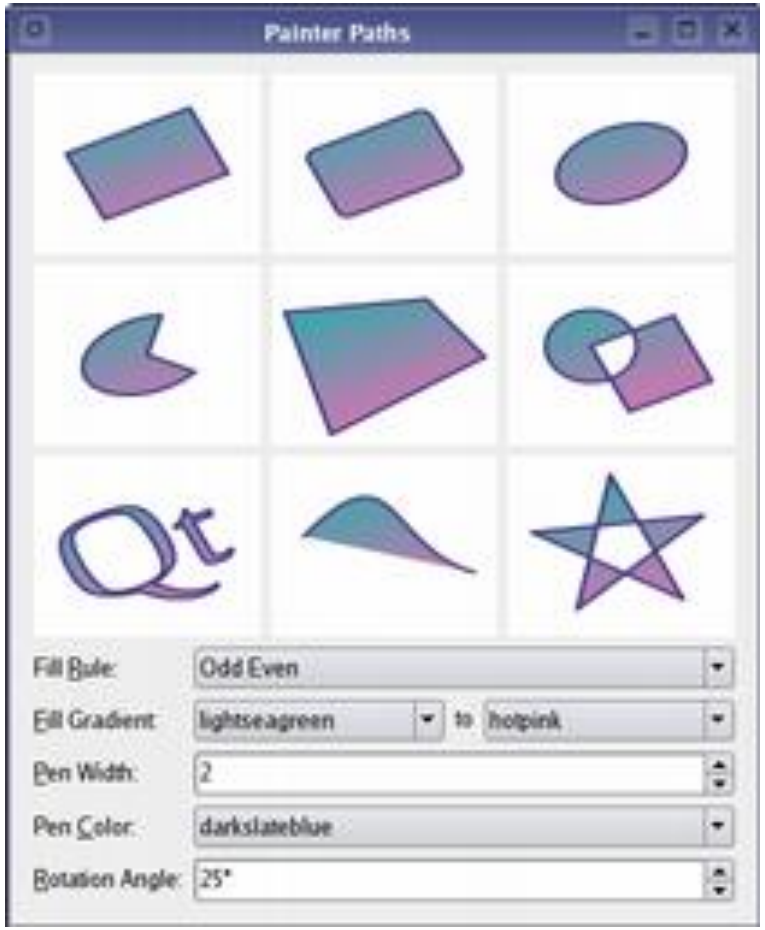
Può disegnare molte primitive

- Linee
- Testo
- Rettangoli ed ellissi
- Poligoni
- Settori circolari e archi di cerchio
- Immagini

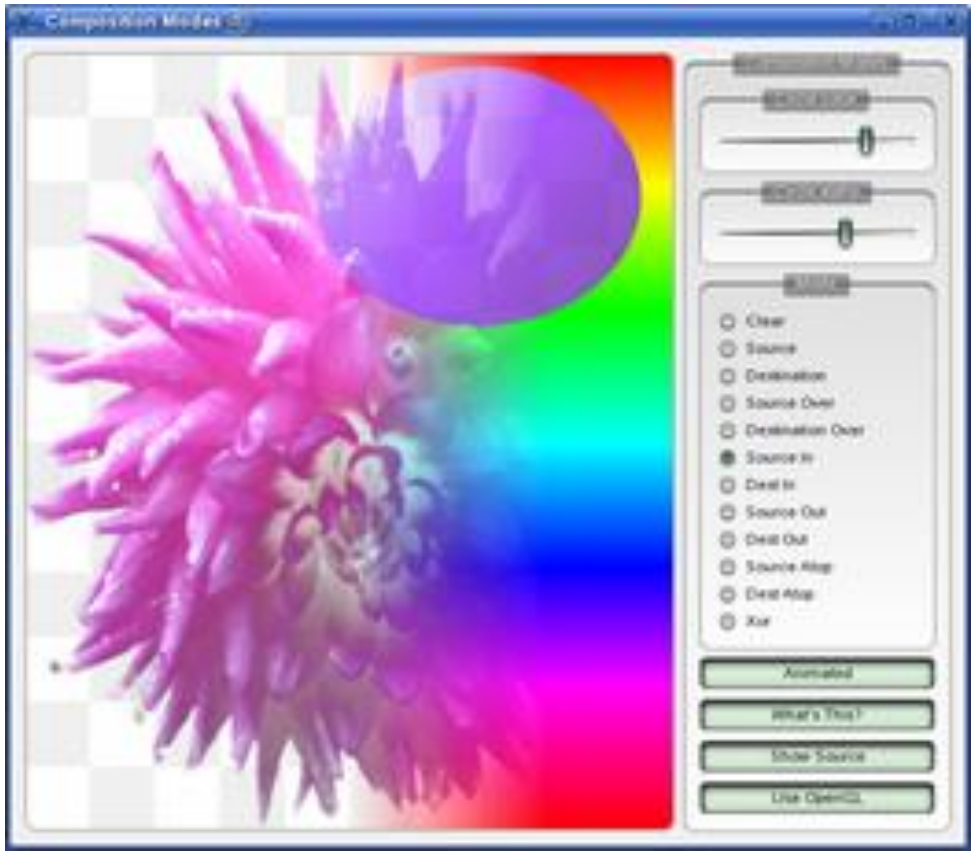
Può applicare trasformazioni 2.5D

Ha diversi algoritmi per comporre le immagini

QPainter (esempi)



QPainter (esempi)



Esempio: custom widget

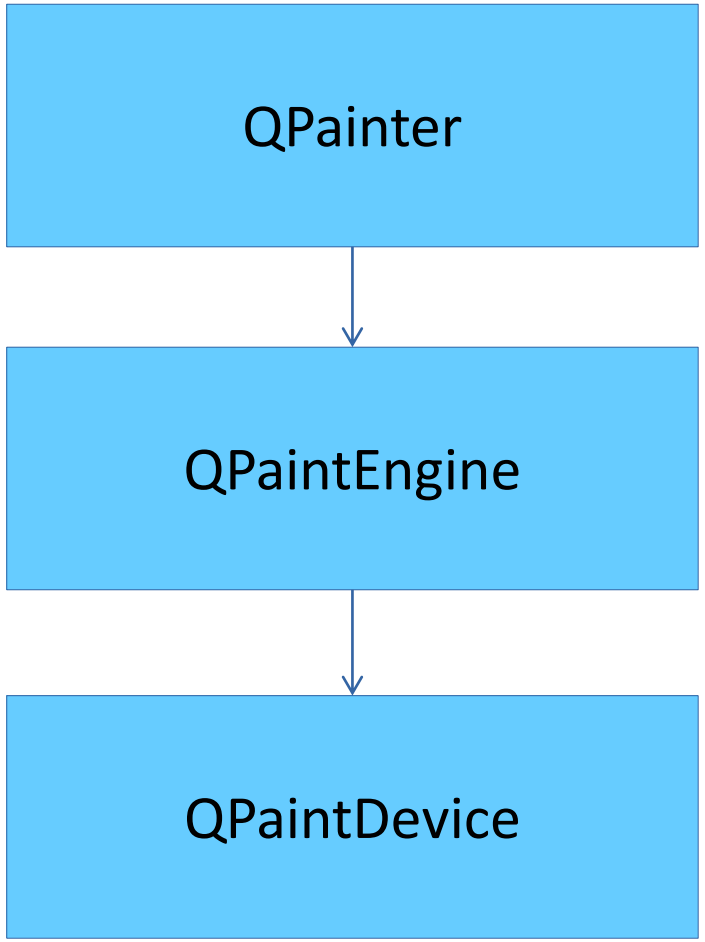
```
void Painted::paintEvent(QPaintEvent * /*event*/)
{
        QPainter p(this);

        p.setPen(Qt::darkBlue);
        p.setBrush(Qt::white);
        p.drawRoundRect(100, 100, 100, 100);

        p.setPen(Qt::red);
        p.setBrush(Qt::yellow);
        p.drawEllipse(30, 30, 240, 30);

        p.setPen(Qt::black);
        p.drawText(QPoint(50, 50), "Painting with QPainter is easy!");
}
```

Stack grafico dei widget



QPaintEngine: interfaccia usata da QPainter per disegnare su un device

QPaintDevice: astrazione per una superficie 2D

Paint devices

I widget non sono gli unici paint devices

- QImage
- QPrinter
- QPdfWriter
- …

Si può usare il QPainter su tutte queste superfici

Viene scelto il QPaintEngine corretto

Si possono creare altri paint devices tramite plugins

Qt Quick

QtQuick è un insieme di tecnologie per la creazione di GUI fluide

- Noto per lo più con il nome di QML

È un linguaggio che parla di oggetti da trasformare e animare

Qt Quick

```
Rectangle {
    PathView {
        delegate: Column {
            id: wrapper
            Image { //...
            }
            Text { //...
            }
        }
        path: Path {
            startX: 120; startY: 100
            PathQuad { x: 120; y: 25; controlX: 260; controlY: 75 }
            PathQuad { x: 120; y: 100; controlX: -20; controlY: 75 }
        }
    }
}
```

Item QtQuick

Gli Item possono essere trasformati in molti modi:

- Opacity
- Rotate
- Scale
- Transform origin
- Trasformazioni generiche tramite matrici 4x4

Tutti i cambiamenti possono essere animati

Gli Item sono aree rettangolari su schermo

Rendering di Item

Vi viene in mente una API con queste primitive?

- Oggetti rettangolari
- Matrici 4x4

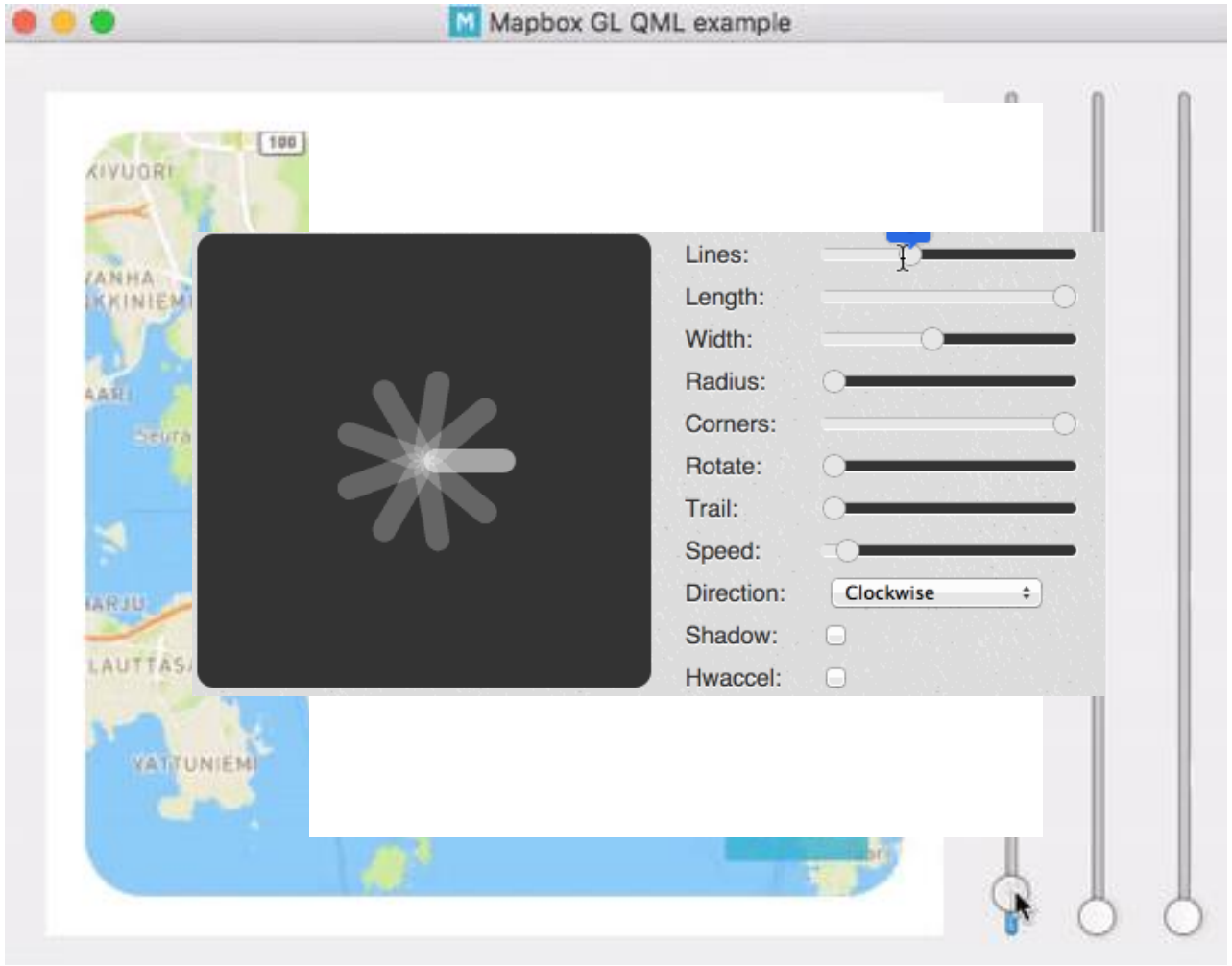
OpenGL!

Ogni operazione su un Item si mappa su OpenGL

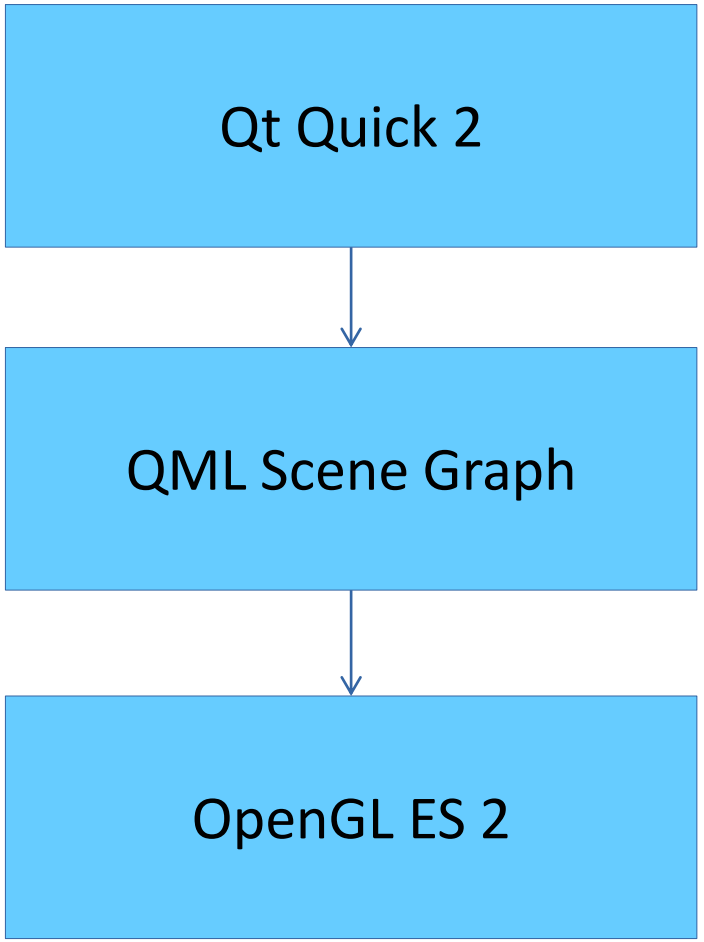
Non ci serve il Qpainter

- Anzi, è dannoso

QtQuick (esempi)



Stack grafico QtQuick



Qt Quick 2: descrive la scena

QML Scene Graph: gestisce gli oggetti sulla scena

OpenGL ES 2: rendering finale su schermo

QWindow

QPainter e lo scene graph devono mandare immagini su schermo

Serve un’astrazione comune per richiedere una finestra al SO

QWindow è l’astrazione Qt per una surface del SO

Funziona anche in sistemi embedded

- Si chiede una superficie OpenGL grande quanto tutto lo schermo

Qt Platform Abstraction (QPA)

QWindow è un wrapper sulle finestre di sistema

L'integrazione con la piattaforma è fatta da plugins

Ogni piattaforma ha il suo plugin specifico

È facile scrivere plugin per nuove piattaforme :)

L'API non è stabile :(

La classe factory principale è QPlatformIntegration

Qt Platform Abstraction

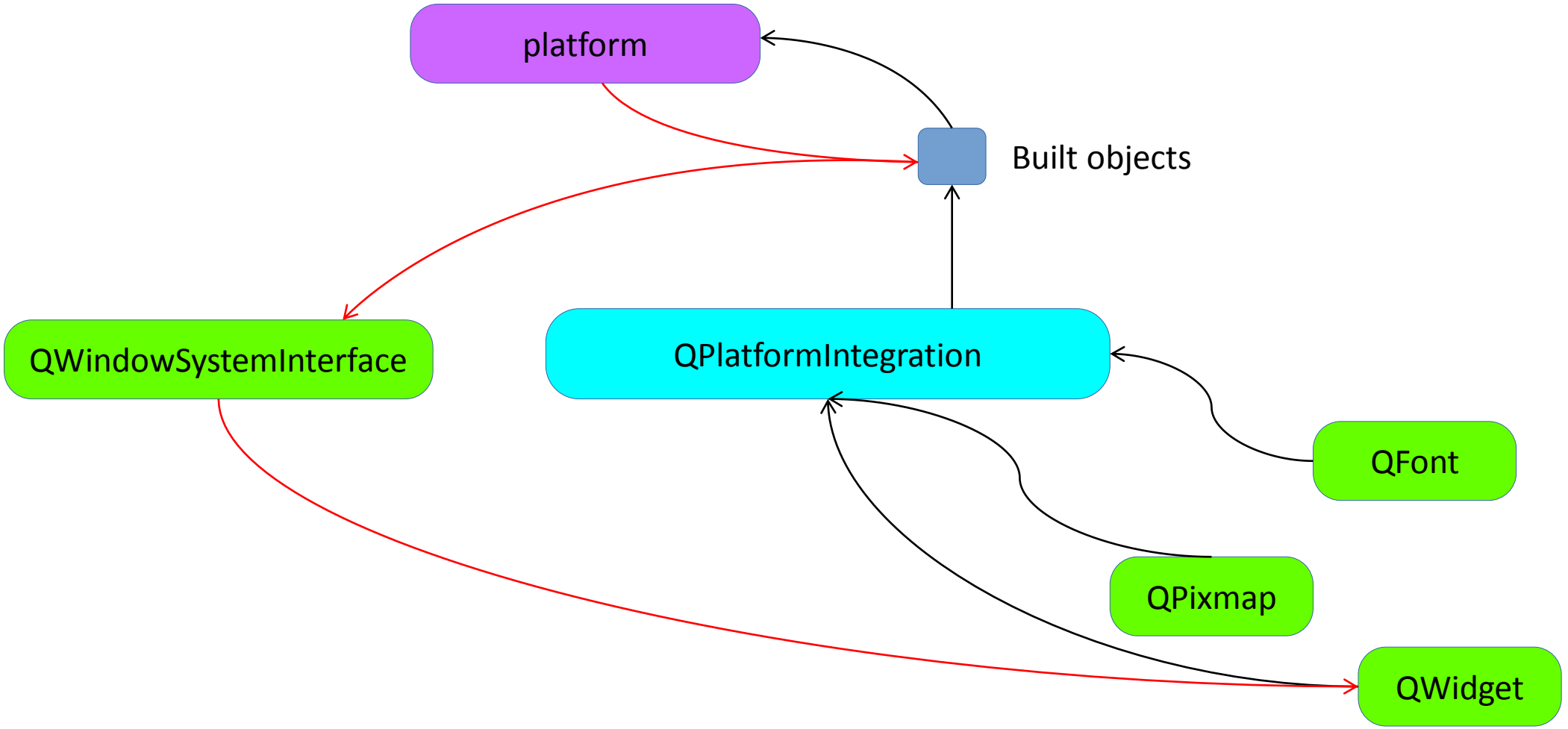
QPA fornisce sia le finestre ma non solo:

- Funzioni di query degli schermi
- Costruzione del database dei font
- Pixel storage per QPixmap

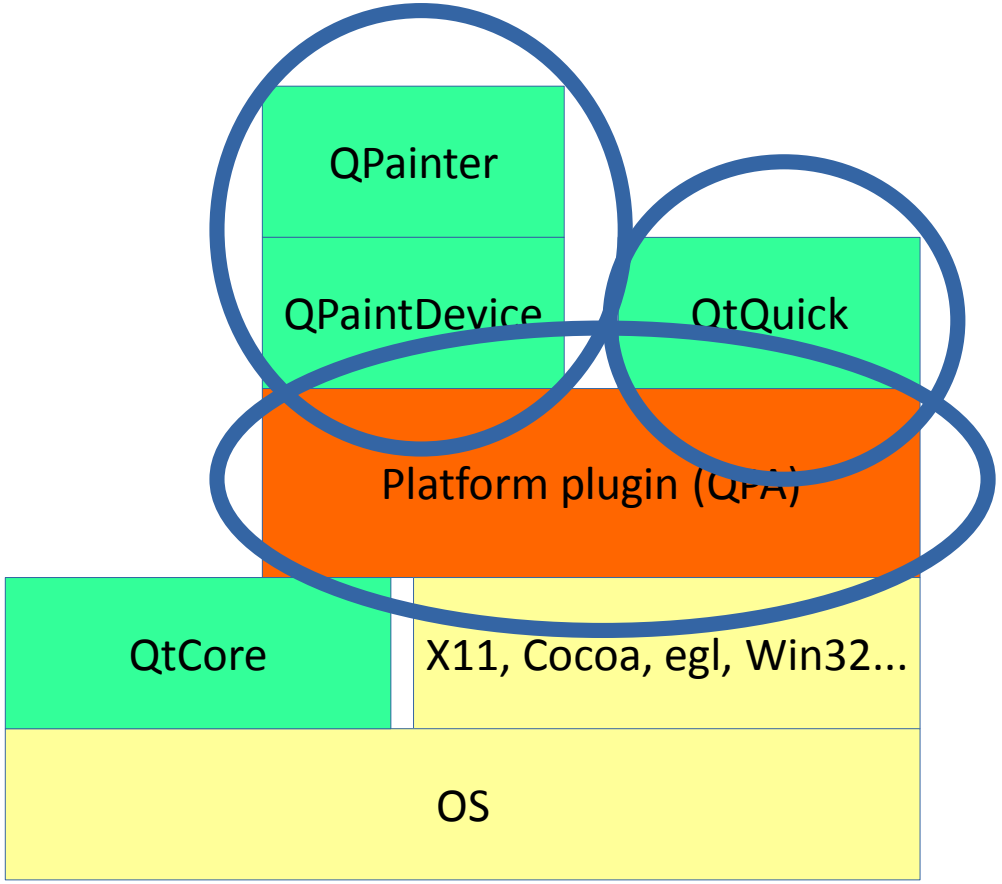
Gli eventi del SO sono gestiti da QWindowSystemInterface, con molte funzioni per:

- Gestire mouse, tastiera, touch
- Eventi delle finestre: enter, exit, focus, close
- Cambio degli schermi di sistema (geometria, numero)

QPA diagramma



Sommario


QPainter
QPaintDevice
QtQuick
Platform plugin (QPA)
QtCore
X11, Cocoa, egl, Win32...
OS

Domande?

Email: lottaviano@develer.com

Web: lucaotta.tumblr.com

Twitter: @lucaotta