# C++ E QT, BINOMIO PERFETTO

Luca Ottaviano
Software developer, Develer



Luca Ottaviano lottaviano@develer.com

# CHI SONO

- Team leader e sviluppatore embedded presso Develer
- Sviluppo con Qt da 8 anni, mi occupo principalmente di realizzare interfacce grafiche
- Contribuisco al progetto BeRTOS (www.bertos.org)
- Conferenze e formazione (interna ed esterna)

# WE'RE HIRING!

- https://www.develer.com/jobs/

- Oppure sono a disposizione al banco in fondo

# FRAMEWORK QT

- Framework per lo sviluppo di applicazioni multipiattaforma
- Basato principalmente attorno ad un modello di programmazione asincrona
- Fornisce un'API uniforme su tutte le piattaforme supportate

# UN PO' DI STORIA

- Primo rilascio: 1995
- Inizialmente disponibile solo per Linux/X11 e Windows. La versione Windows era solo commerciale
- Qt 3.0 (fine 2001): aggiunto supporto per Mac OS X
- Qt 4.0 (metà 2005): enorme riorganizzazione della struttura interna della libreria, ad oggi solo leggermente cambiata

# MODULI QT

- Qt Core: contenitori e strutture dati, loop degli eventi, altre classi di utilità non collegate alla grafica
- Qt Gui: modulo base per i componenti grafici
- Qt Widgets: contiene le classi widget per creare intefacce grafiche a finestre
- Qt Network: funzionalità di rete

# MODULI QT

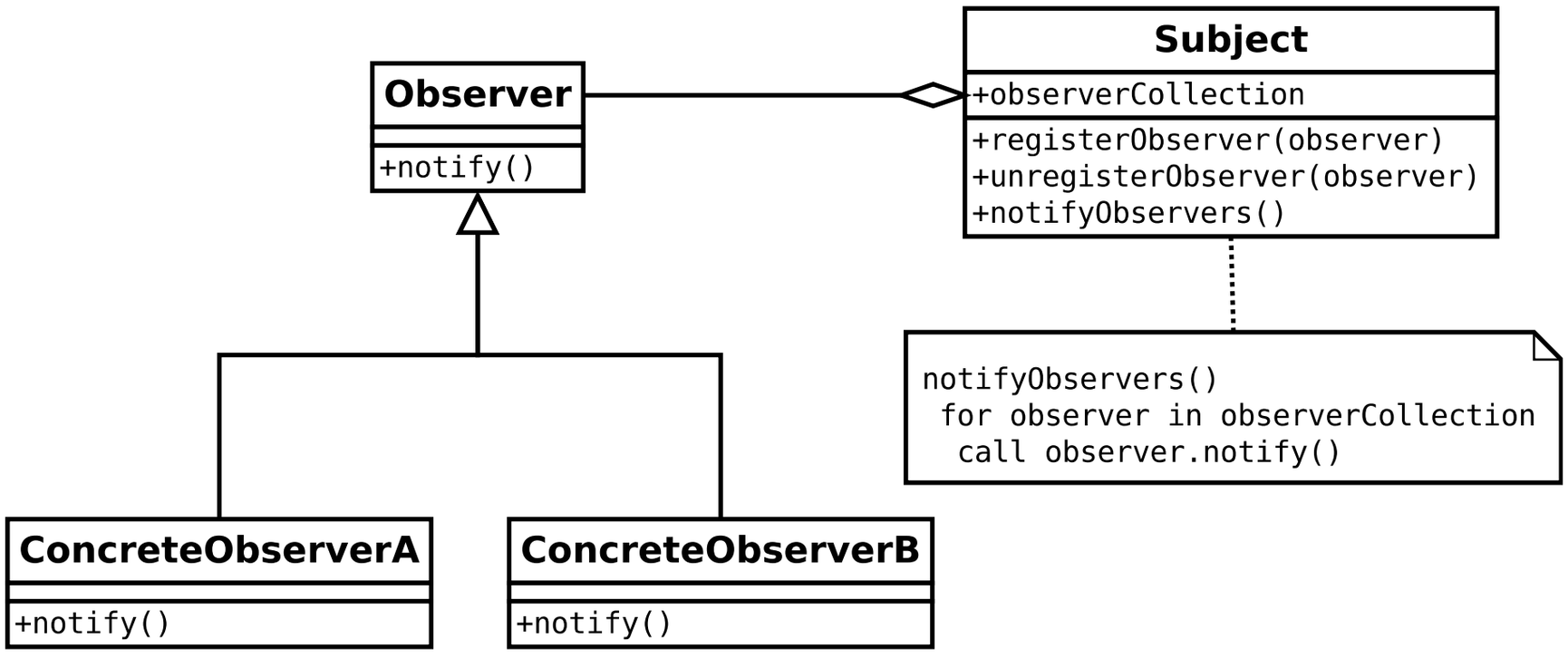
- Qt WebKit: integrazione con il web engine WebKit
- Qt Xml: varie API (DOM, streaming) per l'accesso a file XML
- Qt Test: modulo per la scrittura di unit tests
- Qt Multimedia: funzionalità audio e video

# PECULIARITA' DI QT

- Alcune funzionalità offerte da Qt
  - Comunicazione tramite signal/slot
  - Gestione semplificata della memoria
  - Introspezione a runtime

# OBSERVER PATTERN

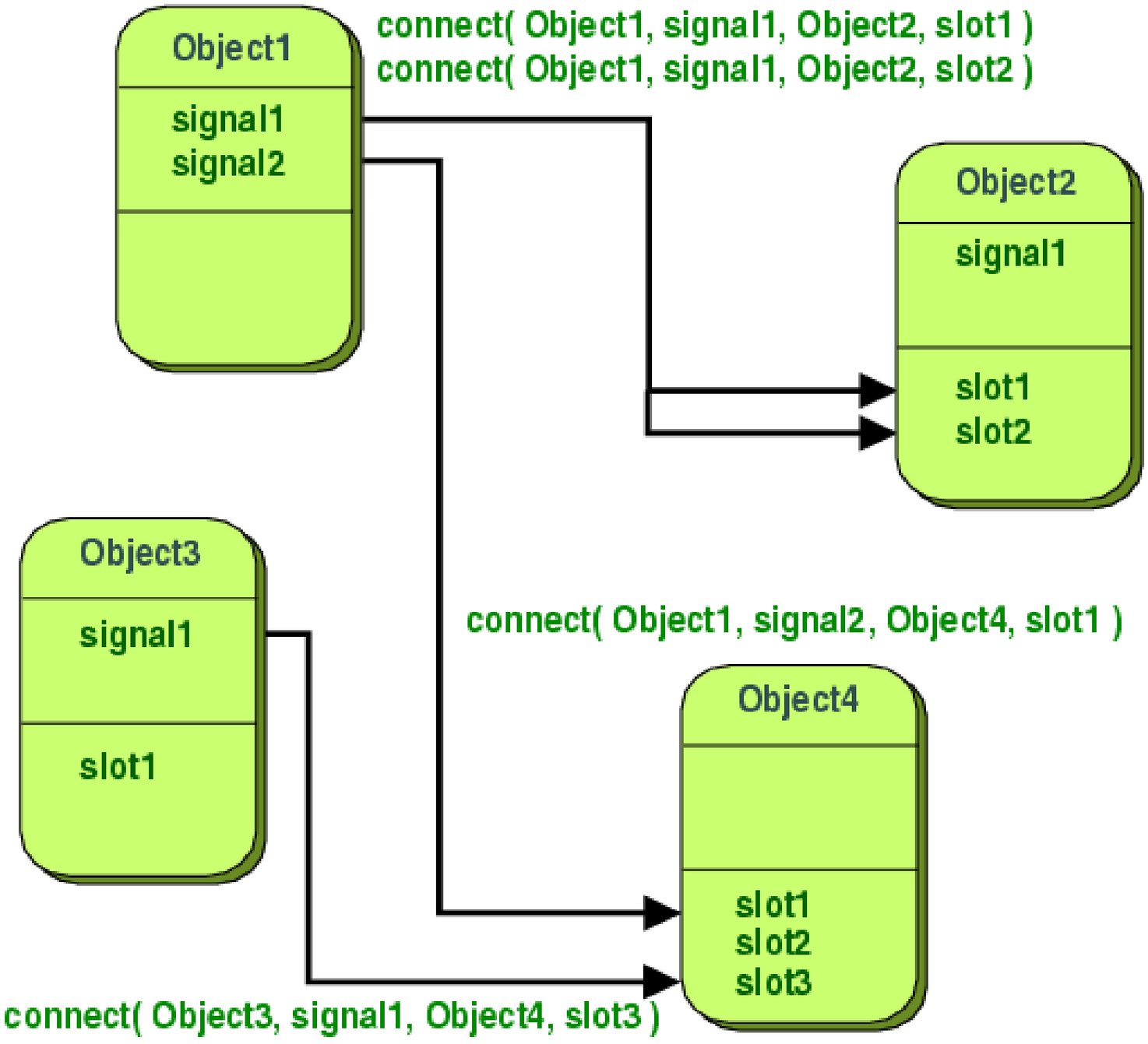
- Basato su uno o più oggetti (listener) che stanno in ascolto per modifiche ad un oggetto osservato (subject)



# SIGNAL/SLOT

- Implementazione del pattern observer
- Aggiunge un po' di zucchero sintattico per rendere leggibile il codice
- connect(): connette un segnale ad uno slot (ossia un listener con un subject)
- emit signalName(): un oggetto può emettere un segnale (notifica)

# SIGNAL/SLOT


Object1
signal1
signal2
connect( Object1, signal1, Object2, slot1 )
connect( Object1, signal1, Object2, slot2 )
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect( Object1, signal2, Object4, slot1 )
Object4
slot1
slot2
slot3
connect( Object3, signal1, Object4, slot3 )

# GESTIONE DELLA MEMORIA

- Gestire la memoria in programmi complessi e altamente dinamici non è facile
- Qt fornisce un sistema agevole per gestire il lifetime degli oggetti
- I QObject sono organizzati in gerarchie ad albero

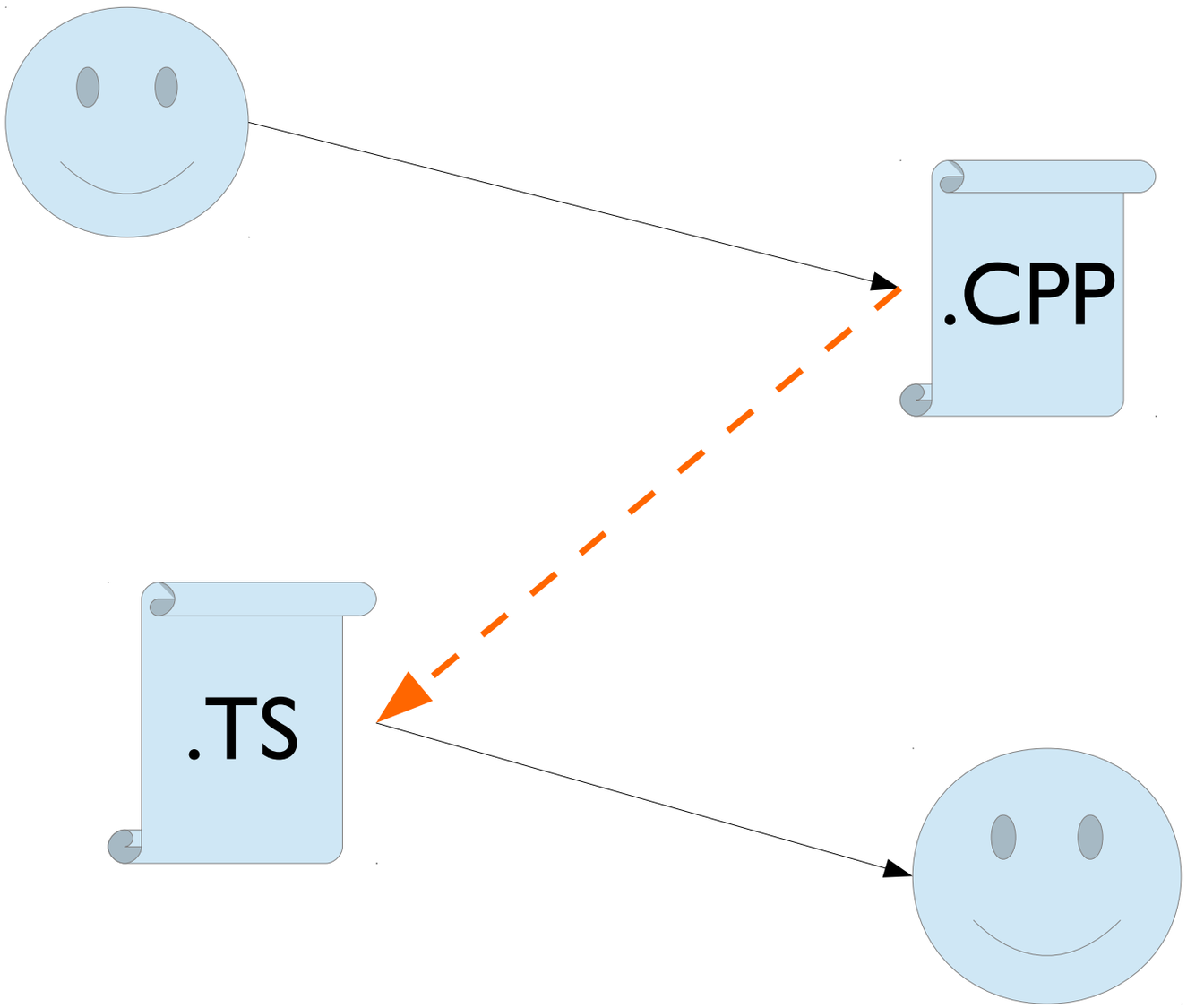
# GERARCHIE DI QOBJECT

- Ogni QObject ha un parent e N children
- Quando il parent viene distrutto, si distruggono anche tutti gli oggetti nel sotto-albero associato
- Quando un figlio viene distrutto, il parent viene notificato
- Tutti i QObject con un parent vanno allocati sullo heap

# INTERNAZIONALIZZAZIONE (I18N)

- Tante sfide da risolvere
  - Encoding dei caratteri
  - Testo BiDi e meccanismi di input
  - Convenzioni per numeri, date...
- Qt supporta la maggior parte delle lingue esistenti
- Supporto per il workflow di traduzione

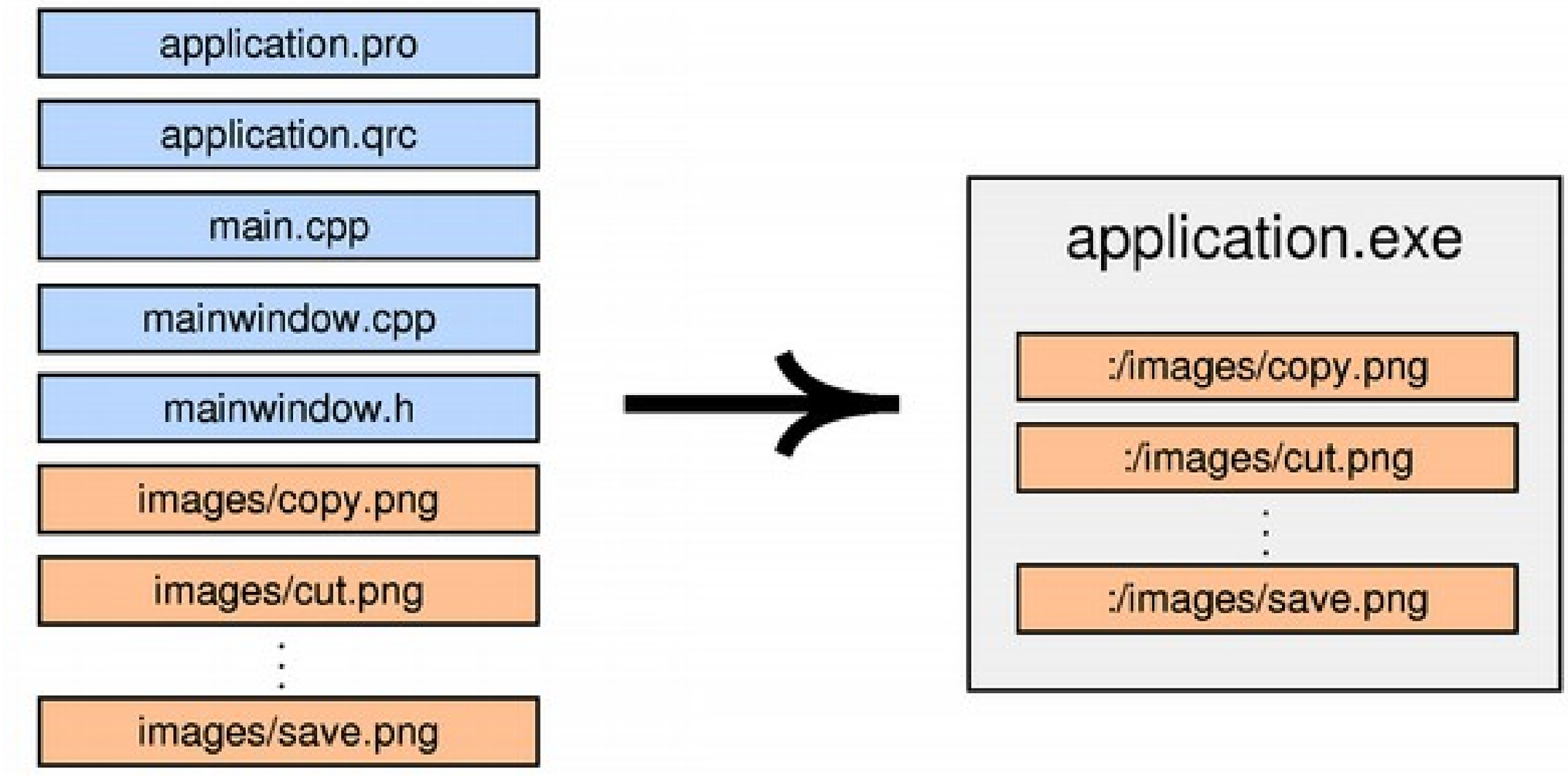
# FLUSSO PER LA TRADUZIONE



# SISTEMA DI RISORSE

- Risorsa: file binario richiesto dall'applicazione
  - Immagini
  - Dati di vario tipo
  - File di configurazione
- Il deploy su molteplici piattaforme diverse è un problema

# SISTEMA DI RISORSE



# SISTEMA DI RISORSE

- Il sistema di risorse risolve il problema del deploy
  - File inseriti all'interno dell'eseguibile finale
  - Procedura integrata nel sistema di build
  - Il binario di porta dietro i file di cui ha bisogno

# CONTAINER CLASSES

- Sono classi contenitori simili ai contenitori STL, ma ottimizzati per espandere in meno codice possibile
- Hanno una API un po' più comoda dei rispettivi contenitori STL
- Sono contenitori Copy-On-Write
  - I dati sono contenuti in memoria esterna
  - I dati vengono copiati quando c'è un accesso non const

# CONTAINER CLASSES

- Il COW rende Qt/C++ quasi un linguaggio di scripting

```
Qstring(“Hello world”).split().isEmpty();
```

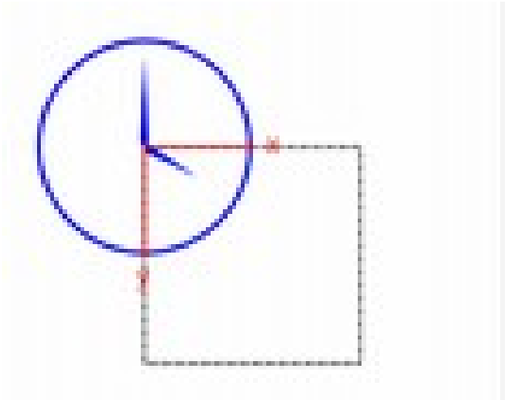
```
file.readAll().split(“\n”).mid(3);
```

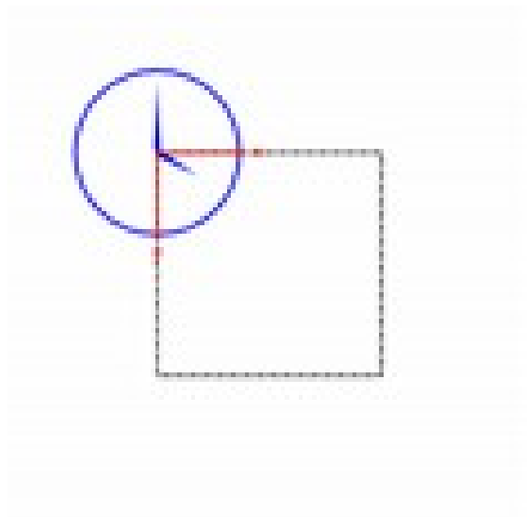
# IMMAGINI

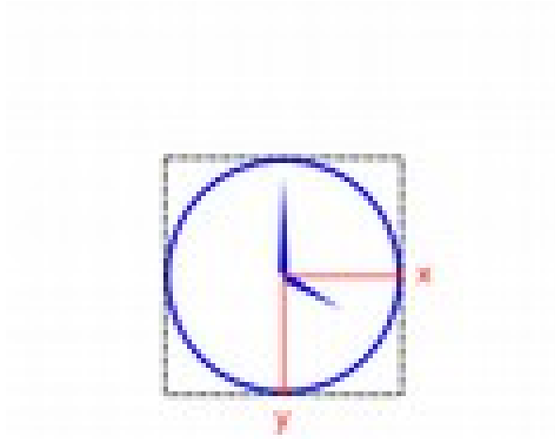
- Qt dispone di molte funzionalità per la manipolazione di immagini
  - Utilizzabili anche per programmi che non fanno uso di finestre
- Molti formati già supportati out-of-the-box
- Espandibile tramite sistema a plugin

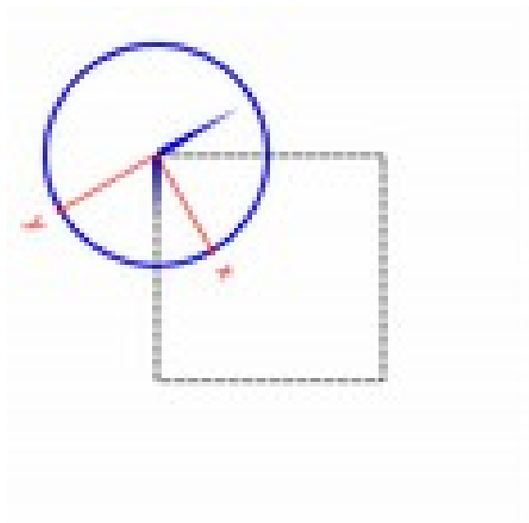
# MANIPOLAZIONE DI IMMAGINI

- Ogni immagine può essere scalata, ruotata etc.





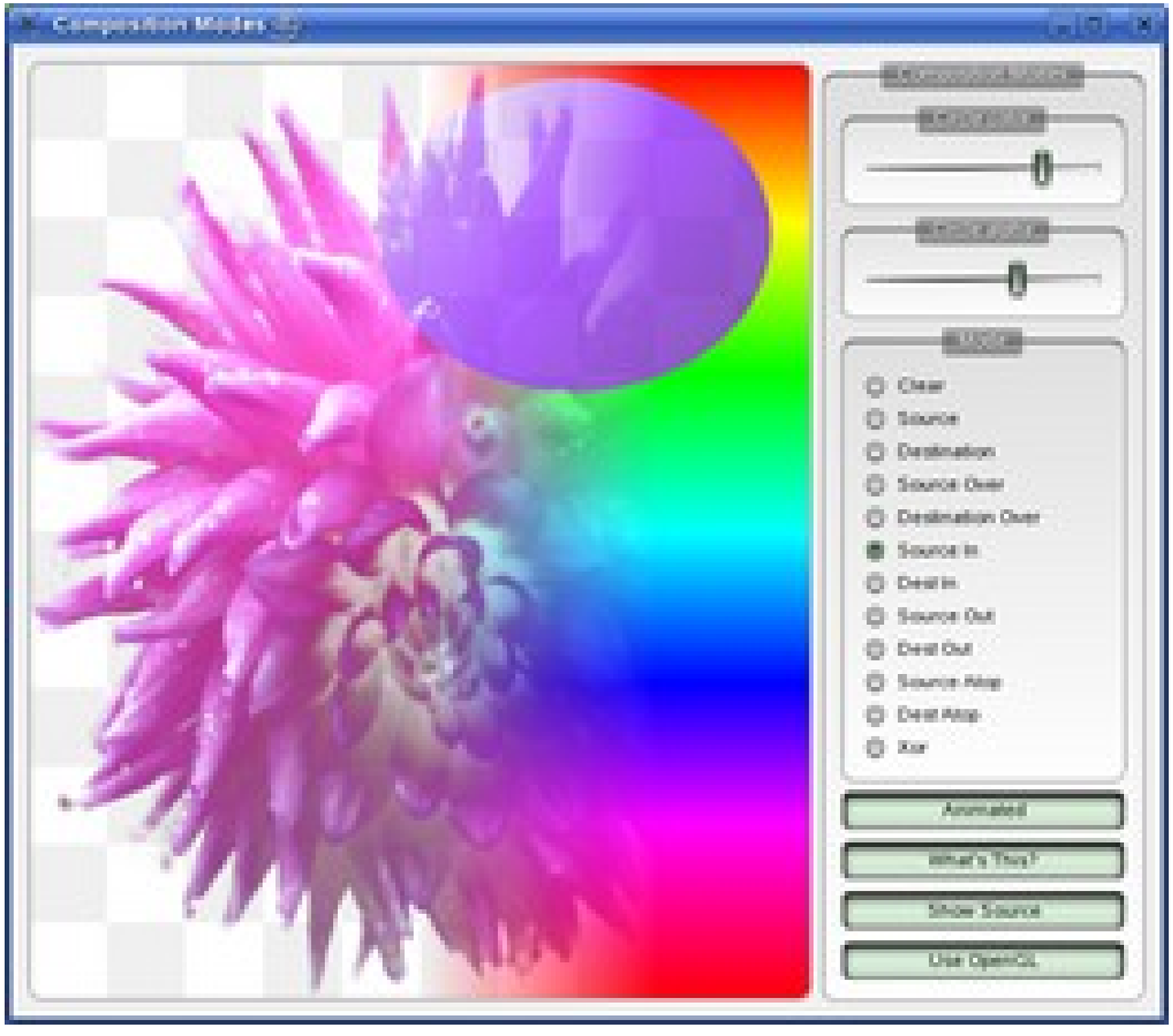




# MANIPOLAZIONE DI IMMAGINI

- Si possono comporre immagini usando l'API 2D fornita dal QPainter
  - API molto ricca e completa
  - Permette di aggiungere testo, comporre più immagini, disegnare tramite primitive grafiche etc.

# COMPOSIZIONE DI IMMAGINI



# QT CONCURRENT

- API di alto livello per eseguire lavori multi-thread
- Nessun problema di sincronizzazione
- Scala automaticamente su tutti i core disponibili
- Operazioni supportate:
  - Map
  - Reduce
  - Filter

# CONTATTI


telefono

+39 055 3984627

e-mail

lottaviano@develer.com

web

develer.com/luca-ottaviano