

# Tutorial

## Cosa c'è di Qt4 in KDE4

### **Premessa**

Questo tutorial è rilasciato sotto Licenza

Creative Commons: Attribution-NonCommercial-NoDerivativeWorks

(<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.it>).

Questo documento può quindi essere riprodotto senza violare nessuna legge, sia in versione elettronica,

sia in versione cartacea, purché sia riprodotto integralmente in tutte le sue parti, compresa la pagina che

contiene queste informazioni:

Versione originale scaricabile dal sito

<http://www.sereno-online.com/site/>

Tutti i marchi riportati in questa pubblicazione appartengono ai rispettivi proprietari.

### **Cosa c'è di Qt4 in KDE4**

La flessibilità e perchè no, la bellezza del nuovo KDE4, sono dovuti anche alle novità introdotte nella nuova major release (4) delle librerie Qt.

Il lavoro fatto dal team che in Trolltech si occupa dello sviluppo di questo toolkit è davvero

notevole; già a partire dalla versione 4.0.0, che era ancora acerba, si sono potute ammirare alcune

"chicche" interessanti. Qui di seguito vedremo le novità principali di Qt4 che hanno impatti

"pesanti" sul K Desktop Environment.

A partire dalla quarta major release, sono state introdotte nuove tecnologie nel "core" della

Libreria, particolarmente utili per KDE.

Fondamentalmente esse sono:

-Arthur, il nuovo sistema di disegno dei widgets.

-Scribe, il renderer per testo unicode

-Mainwindow, ovvero la nuova modalità di gestione della main window di una qualsiasi applicazione Qt.

Unitamente a ciò, è stato introdotto il nuovo motore di scripting QtScript che va a sostituire

il "glorioso", ma meno performante QSA.

## Arthur

A partire da Qt4, è stata introdotta la classe astratta QPaintEngine; tale classe fornisce funzionalità necessarie per il rendering di alcuni tipi specifici, come ad esempio immagini (QImage, QPixmap). QPaintEngine viene usata solamente da QPainter (la classe responsabile del disegno di ogni widget, croce e delizia di chi si realizza in proprio un custom widget) e QPaintDevice, cioè la classe di astrazione di uno spazio dimensionale a due dimensioni che rappresenta lo spazio entro cui si muove QPainter. Questo nuovo engine di disegno è nascosto al programmatore applicativo (a meno che non si decida di reimplementare i propri device di disegno, ma non è un'esigenza così comune) e fornisce i suoi "servizi" per queste piattaforme:

- Un engine di disegno "a pixel"
- OpenGL
- Postscript
- QuickDraw w CoreGraphics
- X11 e X render extension

Il principale beneficio di questa organizzazione del supporto al disegno permette di aumentare l'efficienza delle operazioni necessarie ad uniformare il painting per qualsiasi piattaforma (ricordiamo infatti che il Qt toolkit è multiplatforma):

con la stessa API si può scrivere codice per UNIX/Linux/Solaris MAC e Windows e con la variante derivata Qtopia, la stessa API può essere usata per i dispositivi embedded quali PDA o cellulari).

Un ulteriore beneficio ottenuto da questa organizzazione del sistema di disegno risiede nel fatto che l'aggiunta di nuove funzionalità (ad esempio disegno di nuovi tipi) è resa estremamente più semplice rispetto ad esempio a ciò che avveniva con la precedente major release (Qt3).

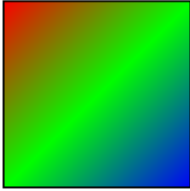
In termini pratici, ovvero per chi non vede le Qt da un punto di vista della programmazione, bensì da un punto di vista dell'utente, Arthur consente la realizzazione di un gran numero di effetti grafici accattivanti.

Uno dei più belli esteticamente rappresenta sicuramente la possibilità di realizzare gradienti di colore che permettono, se usati sapientemente, di disegnare controlli grafici 3D con giochi di luce ed ombre veramente realistici.

I gradienti di colore offerti da Qt 4 sono davvero molti interessanti e fanno la felicità di chi realizza controlli grafici per le applicazioni più disparate e vuole ottenere quell'effetto accattivante che cattura l'attenzione dell'utente.

I gradienti forniti da Qt4 sono:

Lineare



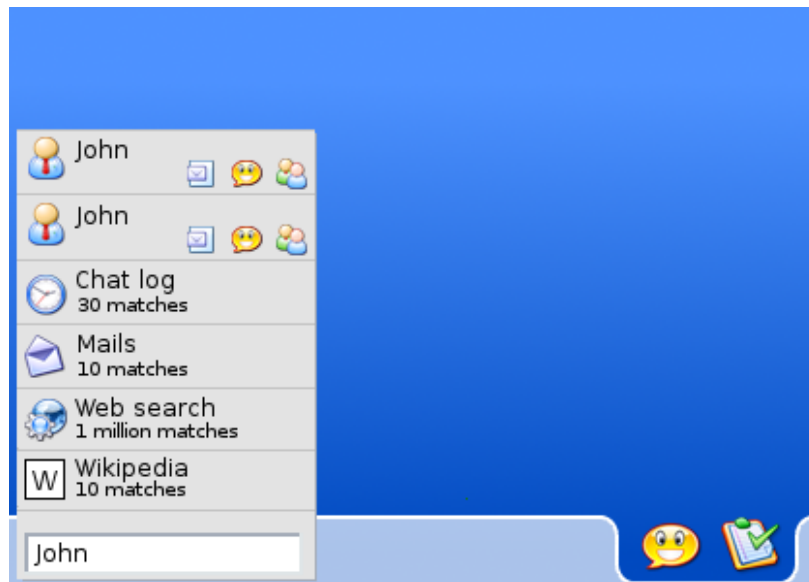
Radiale



Conico



Grazie a questi effetti, è quindi possibile realizzare gli effetti dello sfondo, delle icone e dei push button di KDE4:



Unitamente ai gradienti, le nuove Qt4 offrono molti altri metodi utili per rendere accattivante un desktop, quali il supporto antialiasing, il double buffering automatico

(in tal modo si risolve il problema di flickering causato da operazioni frequenti di painting), il supporto per il disegno di curve di Bezier (per dare forme “morbide” ai controlli), possibilità di inserire textures e rendering del testo evoluto (colorazione a gradiente, antialiasing, operazioni di trasformazione geometrica).

Un’ulteriore importante novità di Qt4 è il supporto nativo del formato SVG (Scalable Vector Graphics).

Lo Scalar Vector Graphics è un linguaggio usato per descrivere sia oggetti grafici statici, sia in movimento. La versione 4 del toolkit Qt include il supporto per la versione SVG 1.2 in modo estremamente efficiente, grazie infatti all’adozione dell’infrastruttura sopra descritta di QPaintDevice, è possibile delegare al toolkit ogni responsabilità di gestione degli oggetti SVG all’interno di uno specifico widget QSvgWidget.

Un esempio tipico, riportato nella documentazione ufficiale Trolltech riguarda lo scenario SVG di figura seguente:



esso rappresenta un insieme di figure geometriche (4 sfere) che ruotano sopra un’immagine (logo Trolltech). Come si può intuire, le possibilità aperte da SVG sono innumerevoli per il KDE; si va dalla gestione delle animazioni per cursori, alla realizzazione di controlli grafici animati, a sfondi e screen saver “evoluti”. L’immagine riportata nella figura precedente, viene realizzata programmaticamente tramite il semplicissimo codice sorgente:

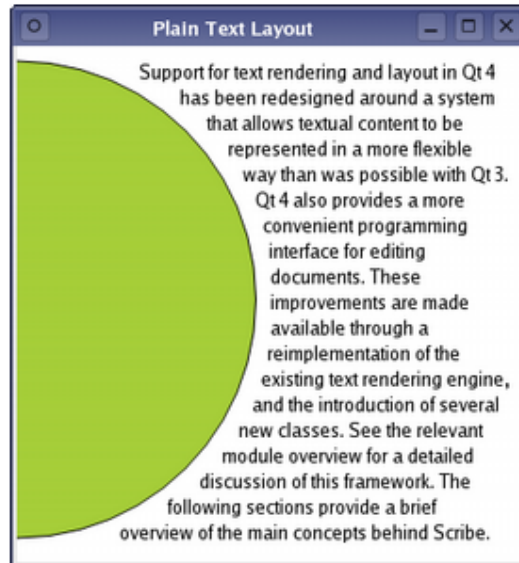
```
QSvgWidget window(":/files/spheres.svg");  
window.show();
```

### **Scribe**

Mediante Scribe, Trolltech introduce un insieme di classi per il rendering testuale più potente ed efficiente di quanto non fosse possibile ottenere con il “vecchio” rich text engine di Qt3.

Scribe aggiunge quindi nuove funzionalità per il disegno ed il posizionamento del testo all’interno di un contesto grafico.

Quanto detto, viene esplicitato nella documentazione ufficiale Trolltech con il semplice esempio seguente:



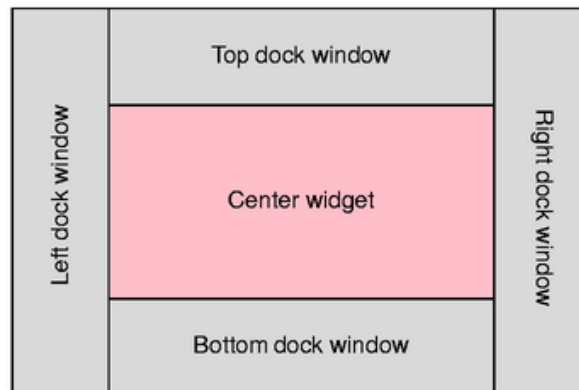
Come si può osservare, il disegno e posizionamento del testo in figura precedente è soggetto alle operazioni di layering che, fino a Qt3, erano ad uso esclusivo dei componenti grafici. Ora invece è possibile combinare la grafica con aree testuali in modo molto semplice (da un punto di vista di programmazione) ed efficiente (da un punto di vista dell'utente).

### MainWindow

Sicuramente le novità introdotte in Qt4 per la gestione della main window rappresentano un grosso passo avanti per chi realizza applicazioni di tipo multiple document interface (MDI).

Le modifiche alla classe QMainWindow (cioè la classe che implementa la finestra principale di un'applicazione, con tanto di menu bar, tool bar laterali e status bar) sono state fortemente volute dalla comunità di sviluppatori (KDE in primis...) per consentire una più rapida prototipazione e sviluppo del software applicativo.

L'impostazione della finestra principale è rimasta quella definita a suo tempo per le precedenti versioni di Qt, ovvero:

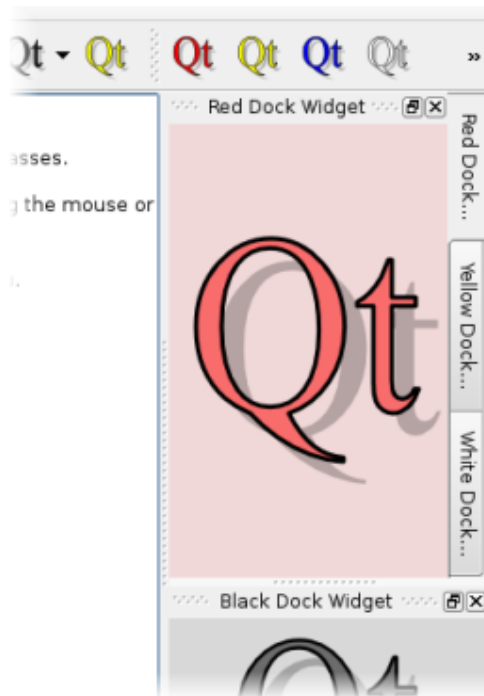


Come si può osservare, la finestra è divisa in cinque principali aree: 4 sono riservate alle dock window (tool bar laterali come ad esempio avviene per gli applicativi open office) e al centro troviamo il Center Widget che è l'area di lavoro (che quindi può contenere un'immagine, un file di testo etc...)

Grazie ad una ridefinizione della API di QMainWindow e ad una sua reimplementazione più efficiente, Qt4 permette di creare in modo molto più rapido l'intera infrastruttura di una qualsiasi applicazione. Con poche righe di codice è ora possibile scrivere un applicazione tipo "notepad" e in un futuro molto prossimo, grazie all'integrazione con WebKit, si potrà realizzare in pochi minuti un web browser; il segreto è infatti quello di impiegare i nuovi widget di Qt4 per rappresentare immagini, documenti HTML e posizzionarli nell'area del Center widget.

L'adozione infine dei fogli di stile, ha dato a Qt4 quel qualcosa in più; tramite un foglio di stile è possibile ridefinire l'aspetto esteriore di un qualsiasi widget Qt, senza per questo modificarne il codice sorgente: come immediata conseguenza di tutto ciò possiamo aspettarci un fiorire di personalizzazioni e skin per tutte le applicazioni KDE.

Un esempio di stile



### QtScript

Sin dalla release numero 3, il toolkit Qt è stato dotato di un proprio linguaggio di scripting (il linguaggio QSA) con sintassi ECMA, più precisamente basato sullo standard ECMA-262. Tale linguaggio, simile a javascript ha permesso ai programmatori di interagire con le applicazioni scritte con Qt mediante uno scripting esterno all'applicazione stessa e in grado di manipolare qualsiasi oggetto contenuto al suo interno (a patto che tale oggetto fosse reso visibile a QSA). Tale possibilità è stata usata in modo estensivo da applicazioni CAD, poiché permetteva di usare un linguaggio esterno ed interpretato per interagire con gli oggetti grafici gestiti dal CAD stesso.

Lo svantaggio di QSA però risiedeva nella sua bassa efficienza; questo linguaggio esterno doveva interagire con i meccanismi interni di Qt mediante complessi (e lenti)

binding che ne rendeva di fatto poco praticabile l'utilizzo per applicazioni particolarmente estese o con necessità di tempi di reazione non troppo lunghi. A partire da Qt4, il meccanismo di scripting è stato portato direttamente dentro al toolkit, per cui è ora possibile rendere un qualsiasi oggetto "scriptable" ed esportarne metodi e proprietà che possono essere manipolate direttamente in un file ECMA script.

Sicuramente questa possibilità verrà sempre più sfruttata dai progettisti KDE per consentire la personalizzazione sia del desktop environment, sia delle numerose applicazioni che popoleranno il mondo KDE dei prossimi anni.