di Matteo Lucarelli > mlucarelli@infomedia.it

FLTK: verso la nuova release

L'FLTK, leggero ed essenziale GUI toolkit, si avvia ad una nuova major release (2.0). Scopriamone le novità e le migliorie introdotte.

In tutti gli ambiti di programmazione non orientata al desktop (interfacce per sistemi embedded, tool diagnostici e quant'altro), è preferibile disporre di strumenti linkabili staticamente (e quindi leggeri), privilegiando maneggevolezza e reattività e rinunciando invece alla disponibilità di attraenti, quanto inutili, abbellimenti, piuttosto che all'integrazione con il desktop manager all'ultimo grido.
La scelta del toolkit, cioè della libreria grafica che fornisce le primitive per la realizzazione di interfacce, è quindi una delle prime difficoltà che si incontrano non appena abbandonato il confortante, ma obbligato, sentiero offerto dagli ambienti proprietari "all in one" in favore della maggiore libertà offerta dal mondo open source.
In particolare molte delle scelte possibili presentano delle forti limitazioni legate a complessità, dimensione o allo scarso, quando non inesistente, supporto multipiattaforma.

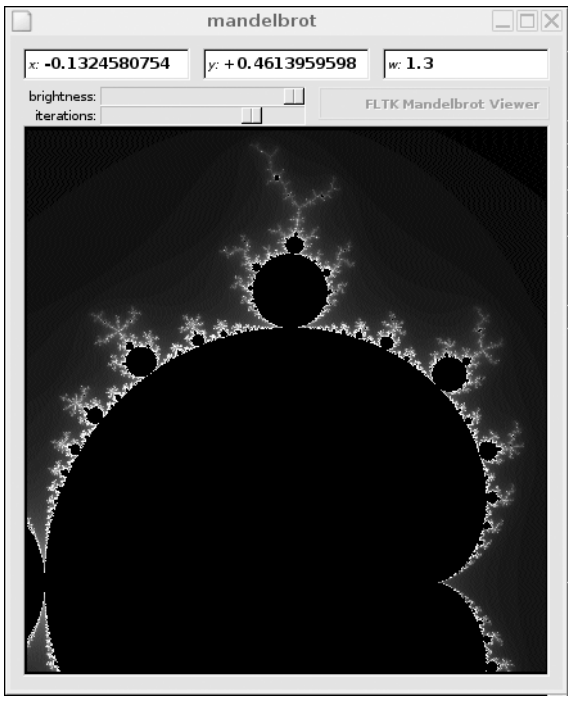

FIGURA 1 L'applicazione "mandelbrot", esempio di potenza e semplicitàdi fltk

L'FLTK, leggero e performante toolkit fortemente multipiattaforma, calza perfetto per utilizzi come quelli esposti, anche se non disdegna ambiti più complessi. Ne è un esempio EDE (http://ede.sourceforge.net/), completo ambiente desktop costruito su una versione modificata della libreria FLTK.
È imminente il rilascio di una nuova versione (2.0) che non ne snatura le caratteristiche, ma anzi estende ed evolve l'idea originale.

FLTK è un toolkit grafico per interfacce utenti, disponibile per Unix/Linux, Windows, MacOS

Caratteristiche e vantaggi

Abbiamo già accennato alla spiccata natura multipiattaforma del toolkit. Difficilmente si riuscirà a trovare una combinazione di sistema operativo ed ambiente di lavoro per la quale FLTK non sia disponibile. Gli stessi sorgenti della libreria sono pronti per essere compilati sotto Linux, Windows, MacOSX, Os2 ed all'interno degli ambienti Borland C++, VCnet, MinGW, DevC++ e Cygwin. Esistono inoltre pacchetti binari in forma nativa per tutti i sistemi citati, oltre che per tutte le maggiori distribuzioni Linux.

Nel 99% dei casi le modifiche necessarie al porting da una piataforma all'altra si riducono all'inclusione dei corretti header file.

```
#if !WIN32
# include <fltk/x.h>
#else
# include <windows.h>
# include <fltk/win32.h>
#endif
```

Detto ciò non ci dilungheremo sulle procedure di installazione, essendo possibile nella maggior parte dei casi sistemare il tutto con un doppio click (o con un "apt–get install" se siete più fortunati).
Naturalmente installando dai sorgenti si potrà godere di una migliore ottimizzazione, oltre che di una nutrita serie di utili esempi d'uso (nella directory "test").
Ulteriori caratteristiche vincenti dell'FLTK sono senza dubbio la sua leggerezza e la sua granularità, il cui effetto, come già detto, è di permetterne l'inclusione statica all'interno del

binario definitivo. Per dare un'idea la classica applicazione Hello Word, compilata staticamente, pesa meno di 200Kb. Resta comunque possibile la compilazione di una versione condivisa della libreria (.dll, .so o simili).
Il pacchetto contiene inoltre FLUID, un essenziale ma efficace modellatore visuale, tramite il quale è possibile realizzare o modificare rapidamente le interfacce "a colpi di mouse", demandando all'applicativo tutta la scrittura del codice relativo alla costruzione e alla gestione degli oggetti grafici.
Fin dall'inizio del suo sviluppo FLTK ha offerto un ottimo supporto alle librerie OpenGL.
Negli esempi allegati è possibile apprezzare i sorprendenti risultati ottenibili con una minima quantità di codice (**Figura 1**).
La maggior parte delle funzionalità infatti si possono ottenere semplicemente derivando la finestra principale dall'oggetto GlWindow ed implementando una propria funzione draw().
Il listato sottostante illustra un esempio (parziale) di codice per l'utilizzo di OpenGL.

```
class myWindow : public fltk::GlWindow {
  void draw();
};

void myWindow::draw() {
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
  for (int i=0; i<sides; i++) {
    double ang = i*2*M_PI/sides;
    glColor3f(float(i)/sides,float(i)/sides,float(i)/sides);
    glVertex3f(cos(ang),sin(ang),0);
  }
  glEnd();
}
```

La libreria comprende inoltre una propria implementazione di funzioni di accesso grafico diretto, pulita e comprensibile, per tutti i casi in cui le OpenGL non fossero disponibili (o necessarie).

Insieme al toolkit è disponibile FLUID, un programma che permette di creare in pochi minuti applicazioni complete di interfaccia grafica.

Ad uso di quanti ne volessero fare un utilizzo commerciale riportiamo tra i vantaggi il tipo di licenza scelta. FLTK è rilasciato infatti secondo una delle politiche più permissive: sotto GNU–LGPL (Lesser General Public Licence, ovvero la più diffusa licenza open–source destinata alle librerie), modificata però in modo da permetterne l'inclusione statica anche all'interno di codice proprietario. In sostanza è quindi utilizzabile liberamente e senza troppe preoccupazioni in qualsiasi scenario commerciale o professionale.
Naturalmente va tenuto conto che, una volta deciso per l'uso del toolkit, sarà necessaria una accorta scelta anche per tutte

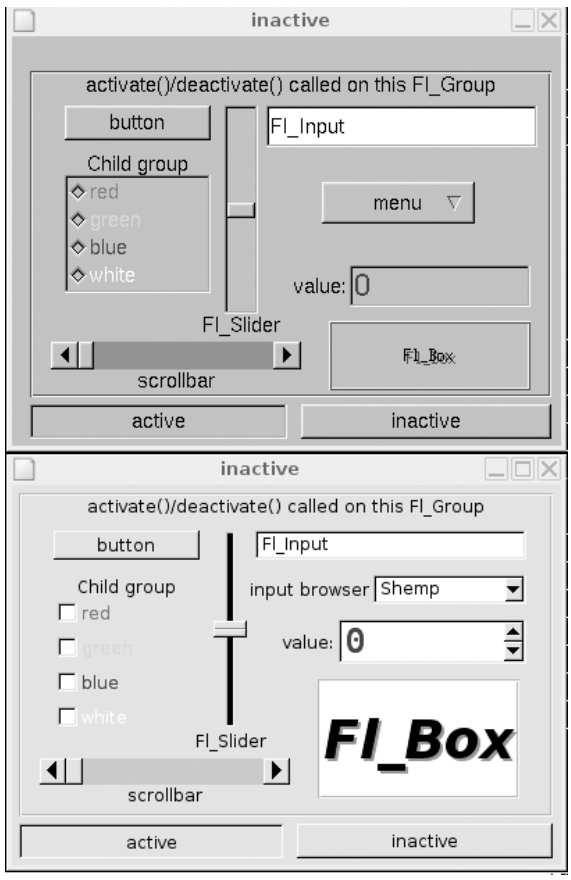

FIGURA 2 Confronto estetico tra la versione 1.1.6 (sopra) e la futura fltk2

le funzionalità aggiuntive necessarie al progetto in sviluppo (a meno naturalmente che questo non si esaurisca nella GUI). Risulta infatti evidente che utilizzare FLTK dovendo poi linkare il codice, ad esempio, alla API WIN32 vanifica tutti i discorsi fatti fin qui, almeno per quanto riguarda leggerezza e portabilità del codice. Non ultimo tra i vantaggi: il progetto è attivo. È infatti imminente, il rilascio della versione 2 giunta ormai ad un livello di stabilità sufficiente. Andiamo quindi ad illustrarne le maggiori novità.

Novità

Togliamo subito un dubbio: come si deduce dal cambio di major number, la nuova versione non conserva retrocompatibilità a livello di sorgente (ed ovviamente tanto meno a livello di binario). Secondo gli sviluppatori comunque "è tutto sommato molto simile: l'80% di tutti i cambiamenti si risolvono sostituendo "FL_..." con "fltk::...".
L'ultima osservazione si riferisce all'introduzione del namespace fltk, che permette una maggiore pulizia complessiva oltre che una migliore aderenza ai paradigmi della programmazione ad oggetti.
Nella **Taballa 1** vengone illustrate le differenze del classico codice Hello Word, che visualizza una semplice finestra. In entrambi i casi la funzione "run()" implementa il loop

principale, e può essere proficuamente sostituita da un ciclo custom che richiami la funzione "check()".
Una delle più frequenti critiche mossa al toolkit, peraltro non sempre fondata, è relativa al suo scarso appeal estetico. Gli sforzi sulla nuova versione si sono quindi orientati anche per ottenere un risultato complessivamente più gradevole all'occhio (**Figura 2**), con una migliore resa dei font e un feeling più naturale e più simile a quello delle interfacce più diffuse. Per restare in tema di caratteristiche "superflue" citiamo l'introduzione di temi e skin, tramite cui è possibile agire sulle caratteristiche estetiche di più elementi in un colpo solo.
Sono stati inoltre completamente ridisegnati una serie di widget (cioè di elementi di base come bottoni, box, riquadri, ecc.) tra cui spicca il nuovo file chooser, ed aggiunti alcuni oggetti utili, tra i quali la ShapedWindow (finestra di forma arbitraria) e i Mutex, con tutte le funzioni necessarie alla loro gestione.
La nuova versione introduce inoltre il supporto ad unicode (UTF–8), tramite le relative funzioni di encode, decode, conversione, ecc.
È stata migliorata la gestione degli shortcut di tastiera, ora assegnabili tramite una semplice funzione. Anche la gestione dei timer, prima implementata in modo un po' confuso, è stata migliorata e razionalizzata. La gestione dei menù, già notevole nella versione precedente, è stata ulteriormente migliorata.
È importante infine sottolineare come tali risultati siano stati raggiunti senza tradire la filosofia di base da cui il progetto prende le mosse. In particolare la libreria non perde niente sul fronte della leggerezza e della pulizia.

Fluid

FLUID (Fast Light User Interface Designer) è il già apprezzato tool di modellazione visuale fornito con il toolkit. A parte una generale migliore pulizia estetica e funzionale (e la sparizione della, poco utile, toolbox), l'utilizzo di fluid resta sostanzialmente identico a quanto già visto nelle precedenti versioni. Ci permettiamo una piccola critica relativa ad una personale aspettativa: anche nella nuova release fluid non presenta alcun meccanismo di undo.

La portabilità di questo toolkit lo dimostra il fatto che è presente anche su AmigaOS 4

Tale difetto, naturalmente è aggirabile tramite frequenti salvataggi, ma la scomodità resta. Continua inoltre a non essere presente alcun tipo di ausilio alla digitazione del codice (syntax highlighting, completamento automatico e simili), tale mancanza è comunque decisamente marginale considerando la destinazione d'uso.
A parte ciò resta uno strumento leggero e veloce, perfettamente coerente con la filosofia del base del progetto. Il funzionamento è decisamente intuitivo, e perciò eviteremo di scendere nei dettagli, sottolineando invece qualche caratteristica saliente. Il tool si presta ad una doppia modalità di utilizzo, permettendo

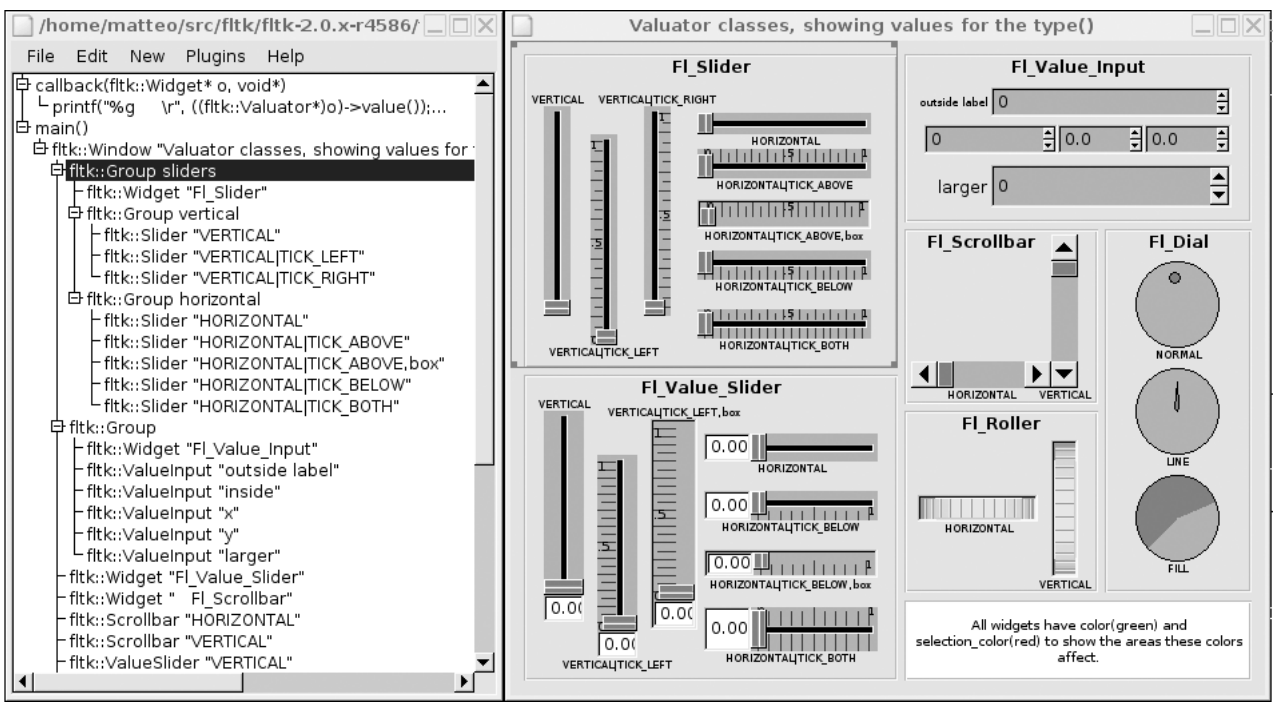

FIGURA 3 Sviluppo di una finestra in FLUID, a sinistra è visibile la gerarchia degli oggetti

TABELLA 1 La classica applicazione "Hello world" nella due versioni fltk ed fltk2

```
// hello word in FLTK2
   #include <fltk/Window.h>
   #include <fltk/Widget.h>
   #include <fltk/run.h>
   using namespace fltk;

   int main() {
     Window *w=new Window(300,180);
     Widget *b=new Widget(20,40,260,100,"Hello");
     w->end();
     w->show();
     return run();
   }
```

```
// hello word in FLTK
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Box.H>


int main() {
  Fl_Window *w=new Fl_Window(300,180);
  Fl_Box *b=new Fl_Box(20,40,260,100,"Hello");
  w->end();
  w->show();
  return Fl::run();
}
```

di lavorare sia in modo visuale, tramite mouse, che tramite una rappresentazione gerarchica ad albero (**Figura 3**).
Quest'ultima in particolare presenta una visione molto funzionale, e spesso molto congeniale alla logica della programmazione ad oggetti. Naturalmente le due modalità sono sincronizzate, e possono essere utilizzate indifferentemente. La costruzione del codice rispecchierà però lo schema ad albero, che va quindi considerato lo strumento primario.
Nei casi più semplici, ad esempio per la creazione di un frontend, è possibile affrontare lo sviluppo di tutto il progetto all'interno del tool, limitando allo stretto necessario l'editing diretto del codice all'interno delle callback (fork/exec o system che siano). Nei casi più complessi invece è possibile sfruttare la logica della programmazione ad oggetti, tramite inclusione o ereditarietà, evitando in questo modo l'editing di codice complesso negli angusti spazi offerti dal tool.
Nonostante l'intera libreria sia naturalmente votata all'uso del C++, Fluid è comunque in grado di generare codice C. La differenza, durante la costruzione, si limita a non includere tra oggetti grafici alcuna classe, limitandosi all'inserimento di semplici funzioni.
Per il file di lavoro di FLUID vale un discorso analogo a quello dei sorgenti: non è compatibile con quello della precedente versione.

Conclusioni

Con questa nuova release FLTK guadagna sicuramente sotto vari punti di vista, non ultimo quello estetico. Il supporto unicode e le altre migliorie ne estendono l'utilizzo anche al di fuori dell'ambiente embedded, finora campo di utilizzo elettivo. Il tutto senza tradire le ottime caratteristiche di leggerezza.
Vista anche la quantità di variabili in gioco, il confronto con altri prodotti analoghi va effettuato con una certa cautela. All'interno della stessa categoria troviamo infatti prodotti molto differenti in quanto a filosofia di base. FLTK non è infatti assimilabile, a causa della sua elevata portabilità (ma anche del tipo di licenza), ai toolkit nativi delle varie piattaforme (Microsoft, Apple, ma anche Motif e simili), e non può sicuramente competere con GTK e QT, che non si propongono come librerie esclusivamente grafiche, ma includono invece tutte le funzionalità necessarie alla sostituzione di una API nativa.

FLTK è stato progettato deliberatamente per essere piccolo, in modo che lo si potesse linkare staticamente alle proprie applicazioni

L'unico prodotto approssimativamente confrontabile è wxWidgets, ma anche in questo caso le differenze sono tali da non poter parlare di reale concorrenza.
Anche dopo l'uscita della release definitiva della 2.0 la versione 1.1.x continuerà ad essere sviluppata e supportata. La maggiore garanzia di stabilità può infatti continuare a renderla preferibile, almeno per il momento, per i progetti più tecnici e laddove non siano indispensabili le ultime caratteristiche. La compilazione statica in particolare rende possibile senza alcun problema la coesistenza delle due release.
Un'ultima nota relativa agli utilizzatori di MacOS che dovranno attendere ancora un po', non essendo, al momento della stesura di questo articolo, ancora disponibile una release completa della versione 2.0 per questo sistema.

Matteo Lucarelli

Ingegnere aeronautico, e programmatore dall'età di 11 anni, lavora come consulente nella progettazione e prototipazione di sistemi innovativi di monitoraggio e sicurezza. Quando possibile preferisce lavorare con strumenti open source (e rilasciare il codice sotto GPL).