Az eFltk bemutatása (1. rész)

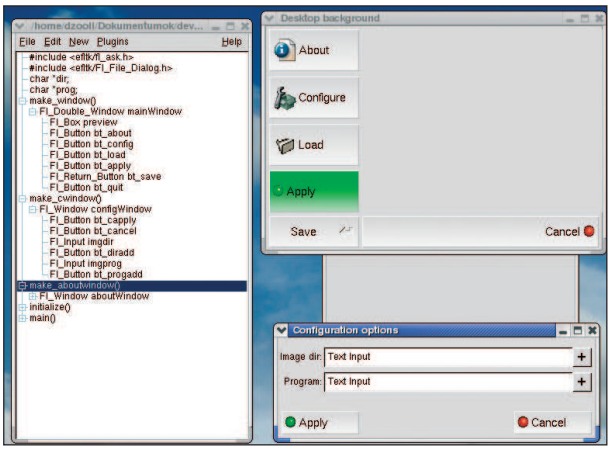
Egy nagyon jól használható grafikus eszközkészletet (widget-set) szeretnék bemutatni. A neve eFltk, ami egyben utal az elődjére is.

Első nekifutásra talán egy kis történelmi háttérrel kell kezdeni az ismerkedést és a grafikus felületek általános működésének tanulmányozásával. Általánosságban elmondható, hogy egy grafikus felület fő működése a megjelenítés után az események feldolgozásából áll. A mai grafikus felületek mind esemény-vezérelt módon működnek és ez igaz a ***Win32***, a ***Mac OS***, az ***Xwindow*** megjelenítésére egyaránt. A program fő tevékenysége az események figyelése és továbbítása a grafikus felület egyes elemei felé.

Az ***eFltk*** sem működik másképpen, azonban hosszú fejlődési időszak után tisztultak le a körvonalak és alakult ki az az eszközkészletet, amiről most szó lesz. A készlet elődjét ***Bill Spitzak*** kezdte el fejleszteni, aki korábban a ***NeXT*** grafikus felületének fejlesztésével foglalkozott 1987-ben. A korai változatot *„views”*-nek nevezték el, de sajnos zárt forráskódja miatt ma nem sokat tudhatunk róla. A következő állomás a ***Forms*** eszközkészletet megismerése és jó tulajdonságainak átvétele volt, majd kiegészítések váltak szükségessé az ***OpenGL*** és ***GLX*** kiegészítések használatához. Ebben az időszakban született meg valójában az ***Fltk*** első változata, ami alapját képezi jelenlegi témánknak.

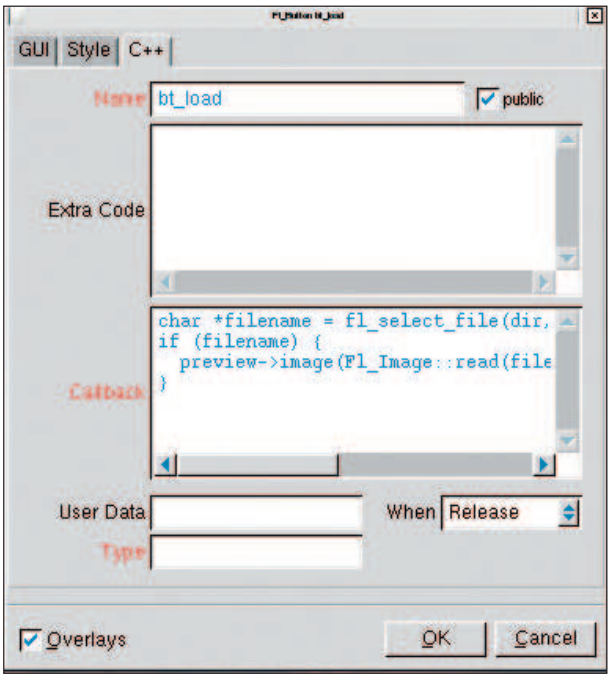
Magát az ***eFltk*** készletet jelenleg hárman fejlesztik egy nagyobb projekt részeként. A munka célja egy általánosan használható grafikus munkakörnyezet kialakítása, amely az ***eFltk***-ra épül és ebből következően gyors működésű és kis helyigényű alkalmazások összessége. A készülő munkakörnyezet tartalmaz, munkaasztal kezelést, ikonok megjelenítése lehetséges, része egy ablakkezelő, egy beállító alkalmazás, egy naptár, óra és egyéb segédprogramok. A környezet neve ***Equinox Desktop Environment*** vagy rövidebben ***ede***. A linkek között megtalálható az eredeti változat és a bővített változat is, amely már tartalmaz a tálcára elhelyezhető ikonokat is.

Ennyi bevezető után azonban tekintsük át az ***eFltk*** legfontosabb tulajdonságait. Elsősorban fontosnak tarom megemlíteni, hogy a készlet minden gond nélkül használható ***Win32***, ***Unix***, ***Linux*** és ***Mac OS X*** környezetben a megfelelő (***MSVC++***, ***GNU C***, ***MinGW***) fordítóprogramok képesek lefordítani készletet és a vele készült alkalmazásokat is. Tehát egy több-platformos grafikus eszközkészletről van szó, ami ***C++*** nyelven készült. Objektum-orientált működéséből adódóan azt gondolhatnánk, hogy a sebessége nem túl látványos, én azonban az elkészült alkalmazásokat és a példákat még így is gyorsabbnak találtam a ***GNOME*** vagy a ***GTK+*** alkalmazásoknál. Ezt talán annak köszönheti, hogy az eseménykezelő ciklusa egy egyszerűsített sémát követ. Az eseményeket mindaddig adja tovább a vezérlőknek, amíg azt valamelyik le nem kezeli vagy amíg van még vezérlő a grafikus elemek között. Nincs szükség annak meghatározására, mely eseményeket melyik vezérlők fogadhatják, hiszen mindegyik megkapja és vagy kezeli vagy visszautasítja. Sok-sok feltételvizsgálatot megtakaríthatunk ilyen módon, főleg ha egy bonyolultabb alkalmazásról van szó.

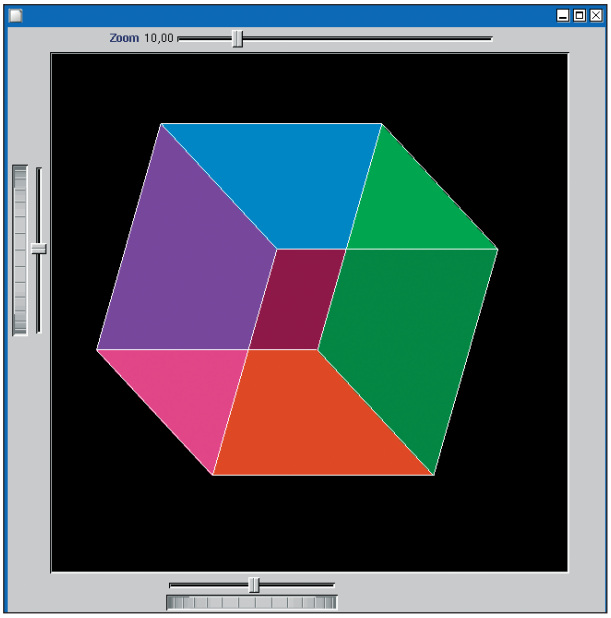


1. ábra Az eFluid felületkészítő

Az ***eFltk***-ban továbbá használhatunk ***OpenGL*** és ***GLX*** segítségével történő megjelenítést, nagyon sokféle vezérlő áll rendelkezésünkre és a fejlesztést nagyban megkönnyíti a grafikus fejlesztőkörnyezet az ***efluid***. Erről az alkalmazásról láthatunk egy képet az *1. ábrán*. Az alkalmazás nagyban felgyorsítja a fejlesztés menetét, hiszen gyorsan megtervezhetjük az alkalmazásunk grafikus felületét, majd a külső rutinok elkészítésével és használatával függetleníthetjük a grafikus megjelenítést a program többi részétől. Természetesen egyszerűbb alkalmazások esetében erre nincsen szükség és ilyen programokat akár szövegszerkesztő használata nélkül is készíthetünk az ***efluid***-



2. ábra Kódszerkesztő az eFluid-ban



3. ábra OpenGL alkalmazás eFltk-ban

ban. A program ugyanis lehetővé teszi, hogy az események kezelésére szolgáló kódrészleteket is a tervező felületen írjuk be a megfelelő helyre. A 2. *ábrán* látható egy egyszerű alkalmazás éppen a kódrészlet készítése közben.

Néhány mondat erejéig tekintsük át, hogy milyen kiegészítések találhatók az elődhöz képest az ***eFltk***-ban. Mindamellett, hogy az általános vezérlőelemeket megtaláljuk (gombok, listák, választógombok, görgetősávok, menü, felbukkanó menü és további vezérlőelemek), fontos megemlítenem, hogy a készlet alkalmas adatbázis kapcsolatok létrehozására is. Lehetőségünk van ***MySQL*** és ***ODBC*** adatbázisokhoz kapcsolódni, majd az egyes vezérlőelemek értékét a lekérdezések eredményéből meghatározni. Nagyon egyszerűen bővíthetjük további adatbázis kapcsolatokkal a meglévő készletet és találhatunk az adatbázis rekordjainak szerkesztésére alkalmas párbeszédablakot is.

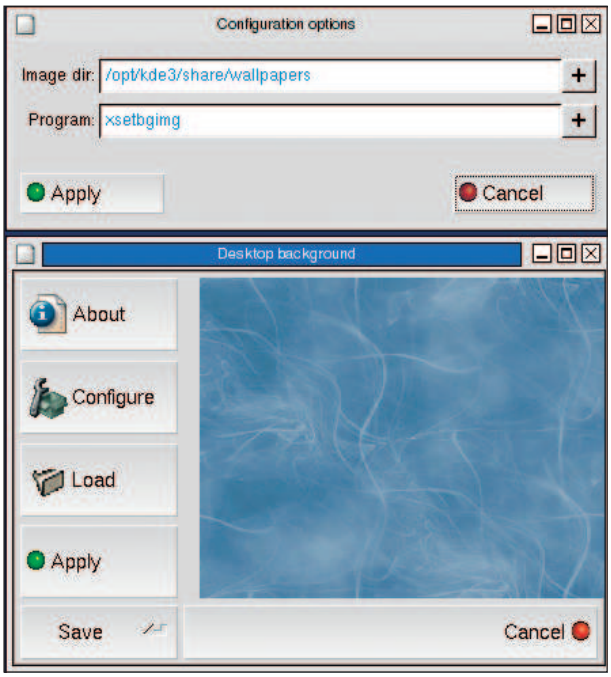
Egy jól használható eszközkészletből nem maradhat ki a hálózati kapcsolatok kezelésére szolgáló osztály sem, melyekből az ***eFltk***-ban többet is megtalálhatunk. Lehetőségünk van ***IMAP***, ***FTP*** és általános hálózati kapcsolat kezelésére a meglévő osztályok segítségével. Könnyedén elkészíthetjük az alaposztályból a számunkra megfelelő hálózati protokoll kezelésére alkalmas leszármazottakat is.

A mai grafikus felületekből nem maradhat ki a képek kezelése sem. Találhatunk beépített képeket, melyeket például alkalmazhatunk gombok vagy menüelemek látványosabbá tételére de a készlet alkalmas ***PNG***, ***BMP***, ***GIF***, ***JPEG*** és ***XPM*** képek betöltésére és megjelenítésére. A képek adatait közvetlenül is elérhetjük, ami lehetővé teszi, hogy képfeldolgozó alkalmazásokat készítsünk. Amennyiben nincs szükségünk különleges képkezelő műveletekre, a képek kezelését végző rutinokat ki is hagyhatjuk a kód szerkesztéséből így kisebb programokat készíthetünk. A későbbiekben vizsgálandó, *4. ábrán* látható program mindössze 77 kilobyte, amelyből a képek 43 kilobyte-ot foglalnak el a programban.

Említésre méltó, hogy a fejlesztők gondoltak a konfigurációs állományok kezelésére is, találhatunk olyan osztályokat, melyekkel a megszokott ***INI*** állományok szerkezetéhez hasonló beállításokkal dolgozhatunk. Az osztály egyszerűsíti a beállítások tárolását és betöltését, nagyban megkönnyítve ezzel mindennapi programozási feladataink ellátását.

Szintén az ***eFltk*** jól használhatóságát bizonyítja az is, hogy az előre elkészített osztályok között találunk naptárat, gyors képkezelésre alkalmas memóriában módosítható (offscreen) bitképeket, állomány választó párbeszédablakot, szövegszerkesztőt és a színek kiválasztására alkalmas párbeszédablakot is.

Egyetlen hátránya az elemkészletnek az, hogy a dokumentáció nem teljes. Az ismert, változatlan részek dokumentumai megegyeznek az eredeti ***Fltk*** dokumentációval, így ezeket nem találjuk meg a html formátumú dokumentumok között, ami újdonság, arról pedig nagyon gyakran csak az osztályfüggvények és változók nevét tudhatjuk meg. Ez bizonyos szinten érthető, hiszen nem mindenkinek egyértelmű és megszokott, hogy fejlesztés közben dokumentálja is a készülő kódot. Ez időigényes feladat, ugyanakkor nagyban könnyíti az érthetőséget és az áttekinthetőséget. Személyes tanácsom, hogy fejlesztés során alkalmazzunk valamilyen önműködő dokumentáció készítő programot, mint például a ***Doxygen***. A program fejlesztése közben így a kóddal együtt készülhet a dokumentáció és egyetlen parancs végrehajtásával elkészíthetjük a ***HTML***, ***RTF***, ***LaTeX*** formátumban elérhető dokumentációt is. A ***Doxygen*** segítségével néhány szabály elsajátítása árán akár magyar nyelvű, képekkel és hivatkozásokkal gazdagított dokumentációt is készíthetünk.



4. ábra Egyszerű eFltk alkalmazás

A program ugyanakkor alkalmas az osztályhierarchia megjelenítésére is, melynek segítségével könnyen érthetővé tehetjük a forráskódokat és az általunk készített új osztályokat is.
Az ismerkedést folytassuk most a klasszikus *„Hello World"* programmal. A grafikus felület megtervezéséhez használjuk az *efluid* programot, és készítsünk egy ablakot, melyben egy gomb felirata legyen *„Hello World"*. A grafikus tervezőprogram képes előállítani a *C++* nyelvű forráskódot is, amit az 1-es listán láthatunk.
A programban találhatunk három függvényt, melyek közül a `make_window` építi fel a grafikus felületet. Amint láthatjuk a felület felépítése nem bonyolult, azonban mégis azt javaslom, hogy a generált kódhoz csak akkor nyúljunk szövegszerkesztővel, amikor már egészen biztosak vagyunk abban, hogy a felület nem fog változni. Nagyobb alkalmazások esetében az események kezelésére szolgáló függvényeket célszerű külön forráskód állományban tárolni, és a tényleges eseménykezelőkből ezeket a külső programrészeket meghívni. Így elérhetjük, hogy a felhasználói felület független legyen a működéshez szükséges kódrészletektől. Tekintsük át most az első listát.

```
#include "hello.h"
Fl_Window* window;

static void cb_Hello(Fl_Button*, void*) {
 window->hide();
}

Fl_Window* make_window() {
 Fl_Window* w;
 {Fl_Window* o = window
 = new Fl_Window(280, 80, "Hello eFltk");
 w = o;
 o->shortcut(0xff1b);
 {Fl_Button* o
 = new Fl_Button(25, 15, 230, 45, "Hello World");
 o->label_font(fl_fonts+1);
 o->label_size(24);
 o->callback((Fl_Callback*)cb_Hello);
 }
 o->end();
 }
 return  w;
}

int main (int argc, char **argv) {

 make_window();
 window->show();
 return  Fl::run();
}
```

A listán jól látható, hogy a program fő függvénye mindössze három sorból áll. Az első sorban elkészítjük az ablak grafikus felületét, majd megjeleníti az ablakot. Ez után kezdődik a program eseménykezelő ciklusa, ami addig fut, míg valamilyen ablak látható a képernyőn. Az ablak eltüntetéséről a gomb eseménykezelő függvénye gondoskodik mégpedig oly módon, hogy a fő ablakot elrejti. Ennek hatására a program befejezi futását.
A program lefordítására létezik egy egyszerű módszer: a parancssorban adjuk ki az

```
efltk-config –compile hello.cpp
```

parancsot. Így statikusan összekapcsolhatjuk a programot a szükséges rutinokkal és bármilyen ***Linux*** környezetben működésre bírhatjuk.
A következő részben folytatjuk az ismerkedést az elemkészlettel, addig is mindenkinek kellemes munkát és tanulást kívánok.

Fábián Zoltán (dzooli@freemail.hu)
26 éves, jelenleg oktatóként dolgozik, szabadidejében szívesen foglalkozik Blenderrel, programozással és elektronikai tervezéssel. Szereti a természetet, túrázást, úszást és a kellemes baráti társaságot.

KAPCSOLÓDÓ CÍMEK

Az Ede (Equinox Desktop Environment) bővített változata:
➲ http://dzooli.uw.hu/ede-1.0.2-dzooli.tar.bz2

Az eredeti fltk:
➲ http://www.fltk.org

Az eFltk honlapja:
➲ http://ede.sourceforge.net