Az eFltk bemutatása (2. rész)

Alapvető elemek és ablakok.

Egy grafikus elemkészlet megismerésénél fontosnak tartom, hogy megvizsgáljuk a legtöbb elem alapját képező osztályokat, mint amilyen az ***eFltk***-ban a `Fl_Widget`. Amint az később is látható lesz majd, ebben az eszközkészletben az elemek elnevezése bizonyos megszokásokat mutat. Mivel az ***eFltk*** őse az ***Fltk***, így az abban megtalálható elnevezéseket követték a szerzők, vagyis az `Fl_` karakterek után következik az elem működésére utaló név. Kezdjük tehát az egyik legfontosabb elemmel, hiszen ennek megismerése a legtöbb elem esetén használható osztályfüggvényeket biztosít számunkra. Ez az elem az `Fl_Widget` nevet viseli.

Az osztály létrehozására alkalmas konstruktor legkevesebb négy, de legfeljebb öt paramétert vár. Az első kötelezően meghatározza az elem kezdeti helyzetét és méretét, és ötödikként megadhatjuk még az elem feliratát meghatározó karakterláncot.

Az első néhány függvény a grafikus elem aktív állapotára vonatkozik (`active(); active_r(); activate(); deactivate()`). Ha egy elem aktív állapotban van, akkor képes az események kezelésére és azokat meg is kapja az eseménykezelő ciklusban. Az `active()` és az `active()` függvények a lekérdezésre szolgálnak (az `_r` itt a rekurzióra utal vagyis az `active_r()` függvény csak akkor tér vissza igaz értékkel, ha az adott elem és annak minden szülője is képes fogadni az eseményeket.

Az elemek feliratának igazítását az `align()` eljárással állíthatjuk be. Ahogyan a grafikus tervezőben is láthatjuk a feliratokat többféle módon igazíthatjuk a vezérlőn belül. Ezeket az igazítási módokat az `FL_ALIGN_` kezdetű változók határozzák meg.

Az egyik legfontosabb függvény a `callback()` mellyel meghatározhatjuk az eseménykezelő függvényt. Általában egy eljárásról vagyis visszatérési érték nélküli `void` típusú függvényről van szó, amint az alábbi listán is látható.

```
void call_back(Fl_Widget* w, void* data)
{
        eseménykezelő_utasítások();
}
```

A függvénynek két paramétere van, az elsőben kapja meg, hogy melyik grafikus elem hívta meg az adott eseménykezelő eljárást, míg a másodikban az adott elemhez tartozó felhasználói adatokat kapjuk. Így hasonló elemeket az eseménykezelő eljárásban az adatai alapján tudunk megkülönböztetni. Lássunk egy példát arra is, hogy hogyan tudjuk beállítani az eseménykezelő eljárást. A példában egy gomb lenyomásához tartozó eljárást állítjuk be:

```
button1->callback((Fl_Callback*)call_back, NULL);
```

Vagyis látható, hogy típuskonverziót kell végeznünk, hogy beállíthassuk az esemény kezeléséhez készített eljárásokat és függvényeket. Az elemek színének beállítására szolgál a `color()` osztályfüggvény. Paramétereként egy `Fl_Color` típusú értéket vár, melyet előállíthatunk az `fl_rgb_color()` függvény segítségével. Ez utóbbi függvénynek paraméterként a kívánt vörös, zöld és kék összetevőket kell átadni, visszatérési értéke pedig a megfelelő típusú szín lesz.

Az `image()` és a `deimage()` osztályfüggvények `Fl_Image` típust várnak, és meghatározható segítségükkel, hogy az elem aktív- és inaktív állapotában milyen képet jelenítsen meg a grafikus elem.

A `label()` metódus segítségével adhatjuk meg az elem címkéjének feliratát, míg a `labelcolor()` (szintén `Fl_Color` típusú paraméterrel) a címke színét határozza meg.

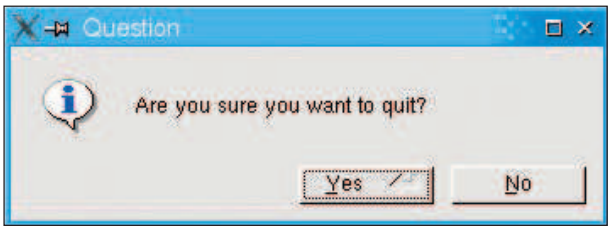
A `show()` és a `hide()` függvények segítségével megjeleníthetjük vagy elrejthetjük az adott grafikus elemet, míg a `visible()` és a `visible_r()` függvényekkel lekérdezhetjük a láthatóságát meghatározó információkat. Itt szintén a `_r` változat szolgál a szülők állapotának lekérdezésére.

A grafikus elem mérete és helyzete beállítható a `size()` és a `position()` eljárások segítségével, míg ezen információk lekérdezésére a `w()` (szélesség), a `h()` (magasság), `x()` (x-koordináta) és az `y()` (y-koordináta) függvények alkalmasak.

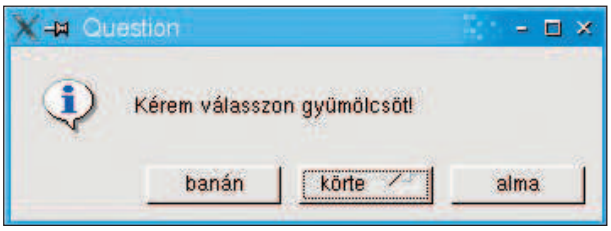
A `resize()` eljárással a méretet és a helyzetet egyszerre állíthatjuk be, az első két paraméterben átadva az új helyzetet meghatározó értékeket, majd a harmadik és a negyedik paraméterben az új méretet adjuk meg.

A `redraw()` és a `redraw_label()` eljárások segítségével ismételten kirajzolhatjuk a grafikus elemet vagy annak címkéjét, míg a `damage()` függvény visszaadja, hogy szükség van-e újbóli kirajzolásra. Amennyiben az elemet valami eltakarta az idők során, akkor a `damage()` függvény nullától különböző értéket ad vissza.

Fontos még megismerni a `tooltip()` függvényt is, hiszen ennek segítségével hozhatjuk létre vagy kérdezhetjük le a grafikus elemhez rendelt gyorstippet. Amennyiben az eljárásnak egy karakterláncot adunk át paraméterként, akkor

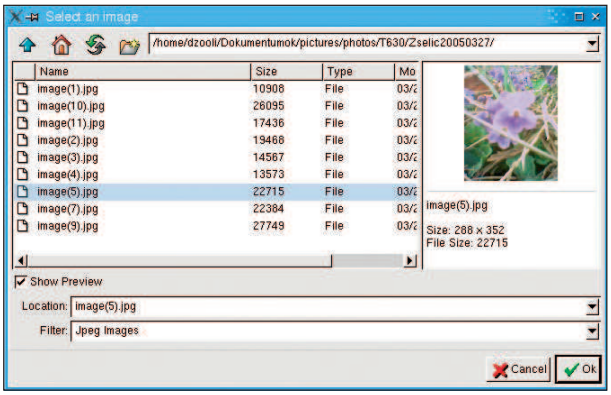


1. ábra eFltk kérdezőablak



2. ábra eFltk többszörös választás

3. ábra eFltk beviteli ablak



4. ábra Állományválasztó

a gyorstipp beállításáról gondoskodik, míg ha nem adunk paramétert, akkor annak lekérdezéséről. Ez utóbbi megoldás a legtöbb ***eFltk***-ban használatos osztályfüggvényre igaz. Hasznos lehet még a `window()` osztályfüggvény, melynek segítségével lekérdezhetjük a grafikus elemhez tartozó `Fl_Window` típusú objektumot. Egy valódi ablak esetében azonban ügyelni kell arra, hogy a függvény nem az adott ablakhoz tartozó mutatót adja vissza (hiszen annak segítségével hívtuk meg), hanem az aktuális ablak szülőablakát kapjuk visszatérési értékként.

A gyakorlatban majd látni fogjuk, hogy valójában az objektum-orientált megközelítés mennyire leegyszerűsíti a programozók dolgát. Nem kell más csak egy jó referencia és egy ábra az osztályhierarchia felépítéséről és máris tisztában vagyunk a lehetőségeinkkel és eldönthetjük, hogy valóban erre az eszközkészletre van-e szükségünk. Javaslatom szerint, ahol kicsi és gyors programokra van szükség, ott mindenképpen érdemes fontolóra venni az ***eFltk*** használatát.

Így röviden áttekintve a fontosabb osztályfüggvényeket, egy nagyon hasznos lehetőségre szeretném felhívni a kedves olvasók figyelmét. Az ***eFltk***-ban és már az elődjében is találkozhatunk előre elkészített párbeszédablakokkal. Következzen ezek áttekintése, megismerése és alkalmazása. Az ismerkedés után már megpróbálhatjuk elkészíteni az első programunkat, ami ugyan kezdetben nem csinál majd semmi hasznosat, de azt legalább felhasználói azonosító és jelszó bekérése után teszi. Itt láthatjuk majd igazán az előre elkészített párbeszédablakok használatát.

Lássuk tehát az ***eFltk***-ban használható párbeszédablakokat szép sorjában. Ahhoz, hogy használatba vehessük ezt a lehetőséget be kell illesztenünk a megfelelő fejléc állományt. Ezt az alábbi makró segítségével tehetjük meg:

```
#include <efltk/fl_ask.h>
```

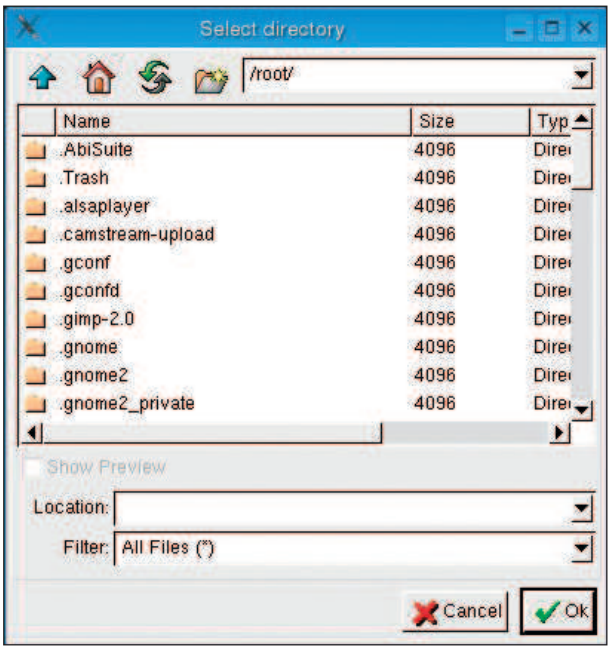
Az első, és legegyszerűbb az eldöntendő kérdések megjelenítésére alkalmas IGEN/NEM párbeszédablak. Az ablakot elővarázsolhatjuk egy egyszerű `fl_ask()` hívással. A függvénynek csak a kérdés szövegét kell megadnunk paraméterként, majd a visszatérési érték alapján eldönthetjük, hogy mit választott a felhasználó. Amennyiben ez az érték 1, akkor a felhasználó vagy az ENTER billentyűvel vagy a ***Yes*** gombra kattintva válaszolt, míg 0 visszatérési érték esetén a ***No*** gombot alkalmazta a felhasználó. Ez utóbbi eset megfelel az 'ESC' billentyű lenyomásának. Az *1. ábrán* látható egy ilyen párbeszédablak.

A következő ilyen párbeszédablak az `fl_alert()` függvény segítségével érhető el és egy egyszerű figyelmeztető üzenet megjelenítésére alkalmas. A függvény paramétere az üzenet szövege.

Több lehetőséget is felajánlhatunk a kedves felhasználóknak a programok használata során, mégpedig az `fl_choice()` függvény segítségével. A függvény első paramétere a kérdés szövege, majd a további három paraméter határozza meg a választások szövegét. Így összesen három lehetőséget ajánlhatunk fel a felhasználóknak. Az alapértelmezett választás a középső gomb, melyet szintén az ENTER billentyűvel választhatunk ki. Ebben az esetben a függvény visszatérési értéke 1 lesz. Az ESC billentyű hatására a visszatérési érték 0, és szintén kiválasztható a jobb oldali gomb segítségével. A bal oldali gomb lenyomásával a visszaadott érték 2 lesz. A *2. ábrán* láthatjuk a többszörös választás megjelenítésére szolgáló ablak képét.

A következő egyszerű párbeszédablak arra szolgál, hogy a felhasználótól adatokat kérjünk be. Itt egyszerűen szöveges adatokat olvashatunk be, az `fl_input()` függvény segítségével. A függvény első paramétere a beviteli mező felirata, második paramétere az alapértelmezett érték. Amennyiben a felhasználó megszakítja a bevitelt, a visszatérési érték `NULL` lesz. A *3. ábrán* látható egy egyszerű adatbeviteli párbeszédablak.

Jelszavak bevitelére alkalmas az `fl_password()` függvény. Paraméterezése megegyezik az `fl_input()`-nál megadottakkal, de a felhasználó által leütött karakterek helyén * karaktereket jelenít meg az ***eFltk***.



5. ábra Könyvtárválasztó

Az `fl_message()` megegyezik a korábban bemutatott `fl_alert()` ablakkal, de ebben az esetben nem figyelmeztetést jelző felkiáltójelet, hanem egy információt jelző 'i' ikont láthatunk a párbeszédablakban.
A párbeszédablakok ikonját megváltoztathatjuk, ha az `fl_message_icon()` függvény által visszaadott Fl_Widget objektumban a képet kicsréljük egy általunk beolvasott képre. Ahogyan korábban már láttuk, erre szolgál az `Fl_Widget->image()` osztályfüggvény.
Természetesen az üzenetek betűkészletének megváltoztatására is van lehetőség, mégpedig az `fl_message_font()` függvénnyel. A függvény két paramétere a betűkészlet (`FL_HELVETICA`, `FL_HELVETICA_BOLD`, `FL_HELVETICA_ITALIC`, `FL_TIMES`, `FL_TIMES_BOLD`, `FL_TIMES_ITALIC`, `FL_COURIER`, `FL_COURIER_BOLD`, `FL_COURIER_ITALIC`) és a betűméret. Érdemes megjegyezni, hogy a betűkészlet és az ikonok megváltoztatása csak a változás után megjelenített párbeszédablakokra van hatással.
A párbeszédablakok sorában a leginkább összetett az állományok kiválasztására alkalmas állományválasztó ablak. Megjeleníthető az `fl_select_file()` függvény meghívásával, melynek három paramétert adhatunk át. Az első paraméter a kiindulási könyvtár, a második az állományok szűrésére alkalmas minta, a harmadik paraméter az ablak címe. A *4. ábrán* látható egy ilyen állományválasztó ablak, melyben éppen JPG képet keresünk a könyvtárakban.
Természetesen az eFltk készítői nem csupán egyes állományok kiválasztására szolgáló párbeszédablakot készítettek, hanem a meglévő eszközök segítségével több állományt is kijelölhetün. Erre szolgál az `fl_select_files()` függvény, melynek paraméterei megegyeznek az előbbiekben bemutatott `fl_select_file()` paramétereivel egyezik meg.
A függvény visszatérési értéke egy karakterlánc lista vagy NULL, amikor nem választunk ki egyetlen állományt sem. Amennyiben a felhasználói programban könyvtár kiválasztására van szükség, használhatjuk az `fl_select_dir()` függvényt. Első paramétere a kiindulási útvonal, a másodikban adhatjuk meg az ablak feliratát. Amennyiben a felhasználó kiválaszt egy könyvtárat a függvény visszatérési értéke ennek elérési útja lesz.
Mindegyik állomány- és könyvtár választó párbeszédablakról elmondható, hogy ha nem választunk ki semmit, vagyis a felhasználó az Esc billentyűvel vagy a '***Mégsem***' gomb használatával bezárja az ablakot, akkor az előbbi függvények NULL értékkel térnek vissza.
Az állományok kiválasztására szolgáló ablakok működését néhány előre definiált változóval befolyásolhatjuk.
Az `fc_initial_w` egész érték határozza meg, hogy milyen széles legyen a párbeszédablak, míg az `fc_initial_h` a magasságot befolyásolja. A képek előnézetének megjelenítését az `fc_initial_preview` logikai típusú változó megfelelő beállításával kapcsolhatjuk be vagy ki.
Most lássunk egy egyszerű példát az alkalmazásunk indítása előtti felhasználó azonosításra. Az előző rész alapján a `main()` függvényben jelenítjük meg az alkalmazás fő ablakát, azonban célszerű az ellenőrzést még az ablak megjelenítése előtt megtenni. A korábbi példában a főprogram így kezdődött.

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

  make_window();
  window->show();
}
```

Az ellenőrzés legyen a lehető legegyszerűbb, lássunk is hozzá. Első lépésben ellenőrizzük a felhasználónevet a `make_window()` előtti sorokban, majd amennyiben azt helyesen adja meg a felhasználó, következhet a jelszó bevitele.

```
const char* pass=NULL;
const char* azon=
   fl_input((const char*)"Kerem adja meg
   ➥a nevet:",(const char*)NULL);
if (!strcmp(azon, (const char
➥*)"rendszergazda")) {
   pass=fl_password((const char *)"A jelszot
   ➥is kerem:", "");
   if (strcmp(pass, (const char*)"jojelszo")) {
      fl_alert("Nem jo jelszo! Kilepes!");
      abort();
   }
} else {
   fl_alert("Nem jo azonosito! Kilepes!");
   abort();
}
```

Amint látható ismét nem túl bonyolult a megoldás, köszönhetően annak, hogy az ***eFltk***-ban a készítők nagy tapasztalattal rendelkező programozók, akik tisztában vannak egy grafikus elemkészlet készítése során felmerülő igényekkel.

Fábián Zoltán