Felhasználói felület fénysebességgel (2. rész)

Sorozatunk előző részében megalkottunk egy egyszerű, de működő képnéző programot az FLTK (Fast Light Toll Kit) segítségével. Most ezt bővítjük tovább…

Legyen az első feladatunk a többképkijelölés lehetőségének a megteremtése. Azt, hogy a fájlválasztóban ezt meg tudjuk tenni, nagyon könnyen elérhetjük: a `kep_open_cb` függvényben (***nezoclass.cxx***) a `New Fl_File_Chooser` értékei között írjuk át `Fl_File_Chooser::SINGLE` bejegyzést `Fl_File_Choosr::MULTI`-ra. Így a kiválasztó már megengedi, hogy több fájlt jelöljünk ki. Hátravan azonban még ezek kezelése a programunkban.
Először is el kell döntenünk, hogy hol és hogyan tároljuk ezeket a fájlokat (eddig nem volt rá szükség, mivel azonnal megnyitottuk őket, és csak a képet tároltuk). Számomra a legegyszerűbbnek az tűnik, hogy a fájlokat – elérési útjukkal együtt – egy karaktermutatókból álló tömbön keresztül érjük el. Legyen ez egy globális változó, hogy mindenhonnan el lehessen érni. Ezenkívül pedig két változót használunk a tömb méretének, valamint az aktuális tömbindexnek a tárolására. A ***nezo.cxx*** fájlban definiáljuk, a ***nezoclass.cxx***-ben pedig utalunk rá:

```
char **fajlok;
int siz=20;
int anum;
```

illetve

```
extern char **fajlok;
extern int siz;
extern int anum;
```

A `nezo.cxx` `init()` függvényében tárhelyet foglalunk az induló tömbnek, és a tömb első elemének (`NULL`) kezdőértéket adunk:

```
::fajlok=(char**)malloc(sizeof(char*) * ::siz);
::fajlok[0]=(char *)0;
```

Ha már a ***keptar.cxx*** fájlt szerkesztjük, módosítsuk még egy helyen – így többet már nem is kell hozzányúlnunk ezen részen belül. Ha úgy állítjuk be a programot, hogy indításkor a parancssorban meg lehessen adni egy képfájl nevét, amit a logó helyett megnyit, lehetőségünk lesz rá, hogy programunkat alapértelmezett képnézőként állítsuk be más programok vagy akár a teljes grafikus felületünk számára. Ez könnyen megoldható (szebb lenne, ha részletesebben elemezné a parancssort; de mivel a legrosszabb, ami ebből adódhat, az, hogy ha a fájl nem létezik, nem nyitja meg; vagy nem kép, esetleg nem jól adtuk meg a parancssorban, nincs rá igazán szükség). Ha van parancssori érték, azt megpróbálja megnyitni, ha nincs, marad a logó. A `main` függvényt kell módosítanunk a `Kep_Nez_Wnd()` utáni és a `show()` előtti részben:

```
if(argc>1){
    ::kep_nez_wnd->loadkep(argv[1]);
    }
else {
    ::kep_nez_wnd->loadkep("logo.jpg");
    }
```

Térjünk vissza a több fájl betöltése témára! A `kep_open_cb` visszahívandó függvényt fogjuk tovább módosítani. Az első tudnivaló: az `Fl_File_Chooser` elemből a `count()` tagfüggvénnyel lehet lekérdezni, hogy a felhasználó hány fájlt választott ki. Ha ez nulla, akkor a Mégsem gombbal, vagy az ablakkezelőn keresztül csukta be az ablakot, egyébként a fájlok számát mutatja. Az értékeket a `value(int)` tagfüggvény adja vissza, egytől sorszámozva – tehát erre kell szerveznünk egy ciklust. Íme a kód (az `Fl::wait()` utáni részt kell eszerint módosítani):

```
if((j=kep_opener->count())){
    for(i=0;i<j;i++){
    fajlok[anum]=strdup(kep_opener->value(i+1));
    anum++;
    if(anum>=siz){
        ➥fajlok=(char**)realloc(fajlok,
        ➥sizeof(char *) * 2 * siz;
        siz= 2 * siz;
        }
    }
    ::kep_nez_wnd->actu=anum-j;
    ::kep_nez_wnd->loadkep(kep_opener->
    ➥value(1));
    ::kep_nez_wnd->redraw();
}
```

Mint észrevehető, a tömb méretét – ha betelne – a kétszeresére nagyítjuk. A fájlelérési utak és -nevek karakterei számára a `strdup` foglal memóriát. Ezt fel kell szabadítani, ha a listát töröljük; a felszabadítás a `kep_close_cb` függvénnyel történik. Mielőtt elfelejteném: a `Kep_Nez_Wnd` meghatározásánál hozzá kell adni az `int actu;` adattagot, mivel ezt fentebb használjuk. A legegyszerűbb, ha a megadott címről letöltjük a fájlokat, és az itt leírtakat csak magyarázatként használjuk. Ezzel a programnak már több fájlt meg tudtunk adni.
A következő lépés az, hogy meg is tudjuk őket nézni. A `kep_open_cb` új változata az újonnan kiválasztott képek közül az elsőt jeleníti meg (`kep_opener->value(1)`), és ezt is teszi pillanatnyilag láthatóvá. Mivel azt szeretnénk, ha lépkedni tudnánk a képek között, valamilyen eseménykezelést kell megvalósítanunk. Eddig is cél volt, hogy külön kezelőelemek nélkül, gyorsbillentyűk segítségével tudjuk használni a programot, most eszerint folytatjuk. Ennek megfelelően a `Kep_Box` leíró tagfüggvényét kell bővíteni, mivel ez kezeli az eseményeinket. Adjunk hozzá a PG_DN, PG_UP, HOME és END billentyűkhöz is a kódot a kapcsolón belül. A forrásszöveg alapján egyértelmű, hogy a hatásukra mit csinál.
Szintén itt látható egy olyan kódrészlet, ami következő célunk megvalósításához tartozik. A fenti léptetőgombok ugyanis

kikapcsolják az önműködő lejátszást. Tekintsük át ennek a megvalósítását! Az FLTK-ban az időzített feladatok elvégzésére az Fl osztály `add_timeout(double time, Fl_Timeout_Handler callback)` függvényét; eltávolításra a `remove_timeout(callback)`; újraindítására pedig a `repeat_timeout(double time, callback)`; függvényeit használhatjuk. Sejthető, hogy az első a programhoz egy időzített eseménykezelőt ad hozzá. Ez úgy működik, hogy a megadott másodperc után (mivel ez egy `double` érték, tetszőleges tört is megadható) meghívja a megadott függvényt. Elvileg egy mutatót is átadhat értékként, ezt itt nem használjuk, alapértelmezett értéke `NULL`, így nem kell megadnunk. A `remove_timeout(callback)` a megadott függvényt távolítja el az időzített feladatok közül. A `repeat_timeout` célja az, hogy magán az időzített függvényen belül meg lehessen hívni, azért, hogy a megadott idő múlva a rendszer újra meghívja. Érdekes megoldása az FLTK-nak, hogy ilyen esetben az idő mérése akkor indul el, amikor a függvény visszatér, azaz (elvileg) független attól, hogy a függvény mennyi munkát végez, vagy hogy az adott esetben milyen bonyolult a végzendő feladat. Az előző részből már ismert `menu_popup[]` tömbhöz két pontot adunk ***Lejátszás indul*** és ***Megáll*** szöveggel. Ezek visszahívandó függvényeiben indítjuk, illetve töröljük az időzített függvényt (`hajto_tocb`), amelyben elvégezzük a képcserét, a pillanatnyi tömbindex állítását, valamint az időzítő újraindítását (`repeat_timeout`). Így már működik az önműködő képváltás!

Van még néhány új képessége képnézőnknek az előző változathoz képest (ezért a változatszám fel is ugrik 0.4-re, mert annyival jobb). Ezek közül még egyet ismertetnék részletesen, a többi sorsát a forrásszövegre utalva az olvasóra bízom.

Ha a kép nagyobb, mint amit az ablakunkban meg tudunk jeleníteni, kicsinyítünk rajta – így azonban a részletek nem látszanak. Mármost megvan a lehetőségünk arra, hogy a képet nagyítsuk, de akkor jelenleg a kép kívánt részét csak a görgetősávokkal tudjuk behozni az ablak látható részébe. Ez nem kényelmes, ezért módosítunk rajta. Egyfelől a kurzormozgató billentyűk (nyilak) segítségével görgetni tudjuk a képet, ehhez a `Kep_Box` leíró tagfüggvényéhez kell a megfelelő bejegyzéseket hozzáadni, másrészről pedig azt szeretnénk, ha a képet az egérrel megragadva húzni lehetne az ablakon belül (mint például az Acrobat Readerben vagy a Gimpben a kijelölések esetében).

A dolgok egyszerűbbé tétele érdekében készítettem néhány tagfüggvényt a `Kep_Nez_Wnd` osztályhoz, ezek: a bal scroll, a jobb scroll, a fel scroll és a le scroll. Mindegyik egy egész értéket fogad, és annyival görgeti a képet a megfelelő irányba, amennyivel az lehetséges. Ha nullát adunk át, akkor nagyjából a kép 1/3 részével görget tovább (ezt használjuk a gombokkal való görgetésnél). Szintén kezeli, ha többet akarunk görgetni, mint amennyit lehetne. Így már egyszerűbb megírni az eseménykezelő részt. A gombokkal való mozgatás ezután egyértelmű.

Az egérrel való mozgatáshoz a leíró függvényt ki kell bővítenünk. A külső kapcsolóhoz hozzá kell adnunk az egérgomb-lenyomás, -vonszolás (lenyomva mozgatás) és -felengedés eseményeit. Ezek a következők: `FL_PUSH`, `FL_DRAG`, `FL_RELEASE`. Az első és utolsónál azt is megnézzük, hogy a bal gombról (`event_button1()`) van-e szó. A vonszolásnál erre nincs szükség, ugyanis az FLTK-ban egy elem csak akkor kapja meg a vonszolás eseményt, ha előtte a gomblenyomás eseményt „elvette”. Azaz, ha azt adnánk vissza az eseménykezelőből, hogy az egérgomb esemény nem érdekel bennünket (mint `Kep_Box`-ot), akkor a vonszolást meg se kapnánk. Ha az egyes egérgomb okozta az eseményt, akkor lenyomásnál beállítjuk a fog nevű, jelen esetben jelzőként (flag) működő változót, hogy benne vagyunk a mozgatási folyamatban, és tároljuk az esemény helyét. A vonszolás eseménynél ezután az új esemény koordináták alapján eldöntjük, hogy merre kell mozdulni, és meghívjuk a megfelelő scroll tagfüggvény(eke)t.

Még néhány szó arról, mivel bővült a program az előző részhez képest:

- Teljes képernyős mód – az FLTK beépített `Fl_Window::fullscreen()` módja segítségével.
- A pillanatnyi fájlnév kiírása az ablak címsorába az `Fl_Window::label()` tagfüggvény segítségével.
- Egyszerű segítségablak biztosítása az `Fl_Help_View` elem segítségével, html formátumú állomány beolvasásával.
- Nagyításnál az előző méret képközepének az ablak közepén tartása, ha a kép már nagyobb az ablaknál.

A képnéző program további fejlődését nyomon követhetik a ➲ http://demo.hmvhely.hu/ címen, ahová igyekszem feltenni az új változatot, valamint a 45. CD-mellékleten, a Magazin/FLTK könytárban.

Végezetül további kedvcsinálónak még néhány projekt, amiket az FLTK segítségével írnak: címeiket az FLTK honlapjáról (➲ http://www.fltk.org) kiindulva találhatjuk meg.

Elemek:

- `Fl_Menu_Bar` – a hagyományos, felül elhelyezett legördülő menük megvalósításához.
- `Fl_Button` – és alfajai a legkülönbözőbb nyomógombokhoz.
- `Fl_Preferences` – a Windows registryhez hasonló, szövegszerkesztővel is kezelhető adatbázis jellegű, beállítási adatok tárolására tervezett elem. Ezt használja például az `Fl_File_Chooser` a kedvencek eltárolására.
- `Fl_Roller` – a régi hangkártyákhoz vagy a legtöbb CD-olvasó elején látható forgatógombhoz hasonló jellegű elem, ami az egér segítségével vonszolással forgatható.
- `Fl_Tooltip` – a lebegő segítségekhez.

Programok, projektek:

- `Fl_Photo` – kép- és digitális kamerakezelő program. Elég sok külső könyvtár is szükséges hozzá, viszont jelenleg is hevesen fejlesztik, a 0.4 változattól rövid idő alatt eljutott a 0.9-ig. Lehet, hogy a cikk megjelenése idején már kész lesz az 1.0 is.
- Teljes ablakkezelő FLTK-ban írva: a ***Bill Spitzak*** által írt `flwm`, valamint a jelenleg is fejlesztett Equinox. Ehhez az utóbbihoz sok segédprogram is készen van már, viszont az FLTK fejlesztési változatát, a 2-est használják.
- Imview képnéző és -analizáló program (csak nézéshez nem igazán kényelmes, viszont sok mást tud).
- 3D-planetárium, valamint sok-sok kisebb segédprogram és az FLTK-ba még be nem vett elem.

Véleményem szerint érdemes megnézni az FLTK-t, mert egyrészt rendkívül logikusan megírt eszközkészlet, kicsi, gyors, rugalmas; emellett Linux/Unix, valamint Windows és Mac alatt egyaránt használható, másrészt felhasználási szerződése teljesen nyitott, még üzleti célú programot is fejleszteni lehet vele.

Havránek Ferenc
Automatikamérnökként dolgozik. Kedvtelései közé tartozik mindenféle kétkerekű járművön (kerékpár és motor) való közlekedés. Ezenkívül szívesen tölti idejét programozással, nemcsak PC-s, hanem egyéb környezetben is, például mikrovezérlő programokat ír.