Fehér Krisztián

Alkalmazásfejlesztés a Felületépítő segítségével

„Taníts úgy számlálni napjainkat, hogy bölcs szívhez jussunk!”
(Zsoltárok 90,12.)

Szerzői magánkiadás, 2024.

ISBN 978-615-6184-37-5
Nyomdai és kiadói munkák: Fehér Krisztián
Elérhetőség: feher.konyvek@gmail.com

Tartalomjegyzék

1 Előszó

1.1 A kiadvány célja és szerkezete

Ennek a kiadványnak a központi témája közel 20 éves múltra tekint vissza. Egy meglévő programcsomag felhasználásáról szól, amit ráadásul én magam fejlesztettem.

A programírás ismétlődő részei nagyon fárasztóak és időigényesek tudnak lenni, ami ellen mindig is kerestem a megoldást.

Még zsenge egyetemista koromban készítettem egy vizuális fejlesztőrendszerként használható kódgenerátort egy BASIC nyelvhez. A célom nem holmi programozási bravúr elérése volt (ami amúgy sem felelt volna meg a valóságnak), hanem a saját programozási tevékenységeim megkönnyítésére szerettem volna eszközöket írni.

A dolog sikerült, 2 évvel később pedig átírtam az egészet C nyelvre, teljesen a win32API-hoz igazítva. Ezt 2008-ban fejeztem be.

Az alkalmazást 2016-ban újrafordítottam friss fordítóval, mivel az újabb Windows rendszerek néhol összevesztek a korábbi EXE fájlokkal.

2023 végére elvégeztem egy újabb felülvizsgálatot és újrafordítást, így született meg a Felületépítő V1.0.1. Legfrissebb verziója ezen kiadvány lezárásakor a V1.0.2.

Kisebb javításokat ugyanis még annak ellenére el szoktam végezni a programcsomagon, hogy az érdemi fejlesztés, a fő funkciók fejlesztése már befejezettnek tekinthető.

Ez a könyv ennek a programcsomagnak a használatát mutatja be, az első használatba vételtől egészen a programok éles használatának bemutatásáig.

Figyelem: A Felületépítő egy hobbi projekt, ezért a frissítése nem garantált, továbbá a használata is szigorúan csak saját felelősségre történhet!

1.2 Követelmények

Ez a kiadvány 'A Windows keményvonalas programozása' c. kiadványom szerves folytatása is egyben, ezért aki nem járatos a Microsoft Windows API C nyelven történő programozásában, annak erősen ajánlható az első rész beszerzése és áttanulmányozása is. 'A Windows keményvonalas programozása' kiadvány áttanulmányozása mindenképpen kell ahhoz, hogy értelmezhető legyen ennek a kiadványnak minden része, lévén azok itt nem kerülnek ismételten bemutatásra.

Ebből következik az is, hogy ez a kiadvány nem mutatja be a Windows API programozását, csak utal rá.

Kellemes és hatékony programozást kívánok!

Fehér Krisztián

2 Ismerkedés a Felületépítővel

2.1 Letöltés

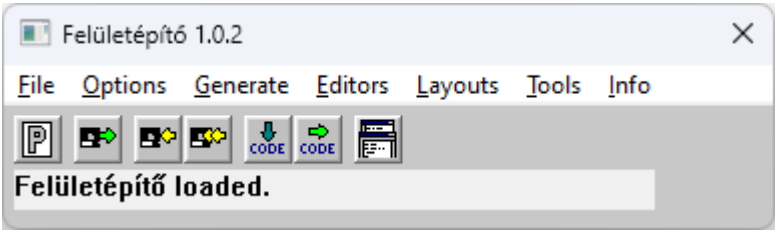
A Felületépítő a github-ról tölthető le, egy becsomagolt ZIP fájlban, a melynek mérete mindössze pár száz kilobájt, annak ellenére, hogy a főprogramon kívűl még 6 segédprogramot és erőforrásfájlokat is tartalmaz.

A letöltési hely pontos címe:

https://github.com/zeuszultra/feluletepito

Letöltés után csomagoljuk ki a ZIP fájlt egy tetszőleges könyvtárba a gépünkön, majd indítsuk el a benne található FEPITO.EXE alkalmazást!

A következő ablak fogad minket:



Fontos: a Felületépítő és a segédprogramjainak a kódjai közvetlenül hívnak Windows függvényeket, ezért előfordulhat, hogy egyes érzékenyebb víruskereső programok megpróbálják blokkolni a Felületépítő egyes műveleteit, például a projektek elmentését. Ilyen esetben a Felületépítőt és a segédprogramokat adjuk a kivételek közé a víruskereső programban!

2.2 A felület megismerése

A programcsomag mindegyike angol nyelvű felülettel rendelkezik. A menüpontok, dialógusablakok stb. mind angol nyelvűek. Ennek ellenére nagyon egyszerűen és tömören van minden megfogalmazva, átlagos angol nyelvtudás elegendő az eredményes használathoz.

2.2.1 Menüpontok áttekintése

Minden menüpont használatát részletesen is ismereteni fogom, itt most csak egy áttekintés erejéig futunk át a lehetőségeken. A Felületépítő menüpontjai a következőek.

File: minden fájlművelet elérhető a menüből, ide tartozik a projektek, layout-ok (lásd később), vagy ablaktervek kezelése is.

Options: általános és speciális funkciók elérhetősége. Az általános beállítások, vagyis 'General settings', fura módon soha nem készültek el, talán majd egyszer...

Generate: a kódgenerálás menüje, ez a Felületépítő titkos fegyvere.

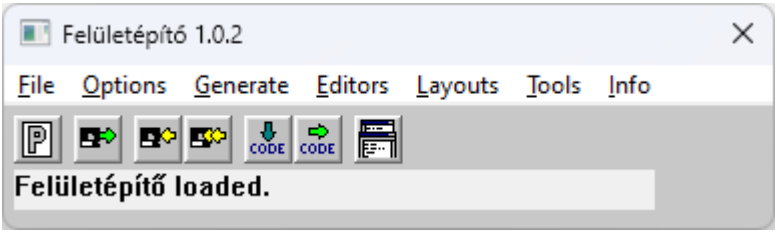
Editors: különböző szerkesztőablakok elérhetősége, ezekkel dolgozunk a projektjeinkben.

Layouts: a szerkesztőablakok különböző elrendezéseinek gyors elérése, vagy elmentése.

Tools: különálló segédprogramok elindítása, ezek is a programcsomag részei.

Info: néhány alapvető információ megjelenítése a Felületépítőről és az aktuális projektről.

2.2.2 Ikonsor áttekintése



Az egyes ikonok rendeltetése az alábbi:

 Projekt létrehozása.

 Projekt betöltése.

 Projekt elmentése.

 Projekt elmentése más néven.

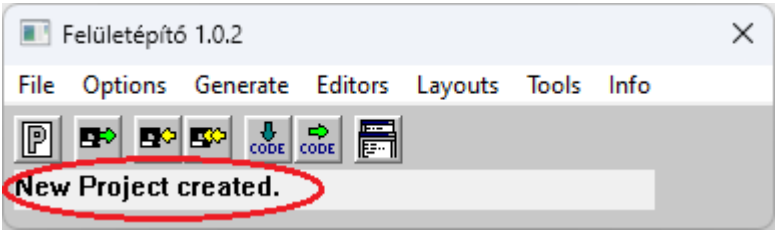
 Kódgenerálás indítása.

 Generált kód elmentése .C forrásfájlban.

 Kódnézet gyors aktiválása.

2.2.3 Állapotsor

Minden fontosabb művelet befejezését a Felületépítő üzenet dialógusablakkal jelzi, vagy az állapotsorban, ami közvetlenül az ikonsor alatt helyezkedik el.



2.2.4 Segédprogramok áttekintése

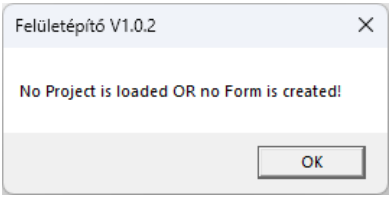
A Felületépítő az alábbi, saját, de különálló segédprogramokkal rendelkezik, ezeket is mind én írtam (részletes bemutatásukat lásd később):

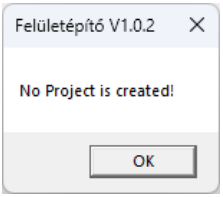
- Menu Resource Creator: menü erőforrás generáló
- Icon Editor: ikonszerkesztő
- Package Creator: telepítőkészlet készítő
- Color Explorer: színkeverő
- Quick Launcher: batch alkalmazásindító.

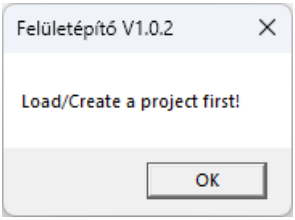
2.2.5 Projekt létrehozása

A Felületépítő használatához létre kell hoznunk, vagy meg kell nyitnunk egy projektet, erre figyelmeztetéseket is kapunk, ha ezt nem

tettük meg és egy programfunkciót el akarunk érni. (Kivételek ez alól az Options, Tools és az Info menüpontok.)



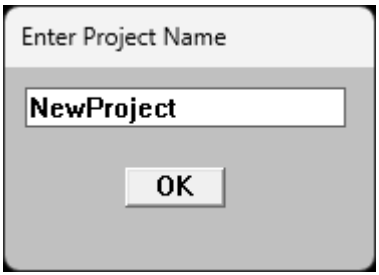




Új projekt létrehozásához Kattintsunk a File – Create New Project menüpontra,vagy a  ikonra az ikonsorban!

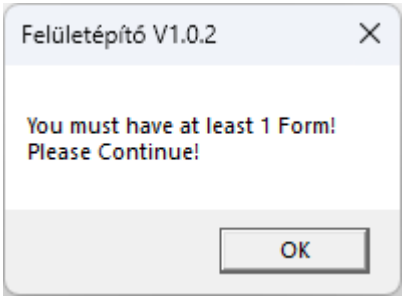
Ekkor egy dialógusablak jelenik meg, amiben meg kell adnunk a leendő projektünk nevét. Ha ez megvan, kattintsunk az OK gombra!

(Fontos, hogy ezt a folyamatot mindenképpen végig kell vinni, megszakítani nem lehet ezt a lépéssorozatot!)



Ezt követően ki kell választanunk a projekt helyét valamelyik adattárolónkon.

Ajánlott, bár nem kötelező a Felületépítő 'Projects' mappájába menteni. Ha a 'Mégse' gombra kattintunk a 'Mentés' dialógusablakban, akkor megjelenik az alábbi üzenet és visszakerülünk az 'Enter Project Name' párbeszédablakhoz.

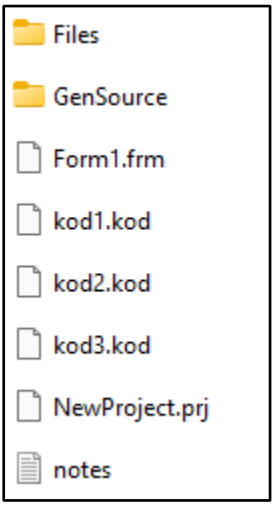


A Felületépítőből elérhető 'Fájl megnyitás', ill. 'Fájl mentés' típusú dialógusablakok jelenleg nem teszik lehetővé a könyvtárak kezelését, ezért új projekt könyvtárak létrehozását ezt megelőzően, ill. ettől függetlenül kell elvégeznünk.

2.2.6 A projektek mappaszerkezete

Minden projekt két könyvtárat tartalmaz és néhány fájlt. A 'Files' könyvtárba a projekthez tartozó külső adafájlokat, például bitképeket helyezhetünk el. A 'GenSource' mappába kerülnek a majdani generált forráskódok, a Felületépítő valódi kimenetei.

Az .FRM kiterjesztésű fájlok az ablaktervek adafájljai, kezdetben egy automatikusan létrejön.



A .KOD típusú fájlok az egyes forráskódokat tartalmazó szerkesztőablakok tartalmát mentik el.

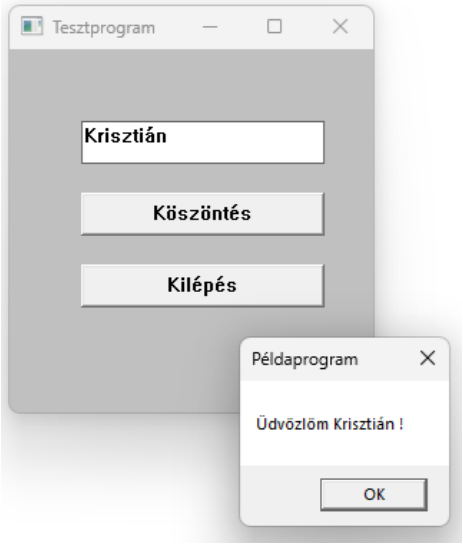
A projektjeinkhez megjegyzéseket, jegyzeteket írhatunk egy külön ablakba, ezek kerülnek a NOTES.TXT szövegfájlba. Ennek a tartalma nem kerül semmilyen formában a majdani generált forráskódokba, csupán segít rendszerezni és könnyen elérhetővé tenni a projekttel kapcsolatos ötleteinket, megjegyzéseinket.

Ezen a ponton már megismerkedtünk picit a Felületépítővel, itt az ideje, hogy a segítségével elkészítsük egy egyszerű ablakos alkalmazás forráskódját!

3 A Felületépítő élesben

Ebben a fejezetben egy olyan kis alkalmazás kódját fogjuk elkészíteni, ami azonnal fordítható álapotú lesz.

Az alkalmazás egy szöveges beviteli mezőbe írt nevet felhasználva ír majd ki egy üdvözlő szöveget.

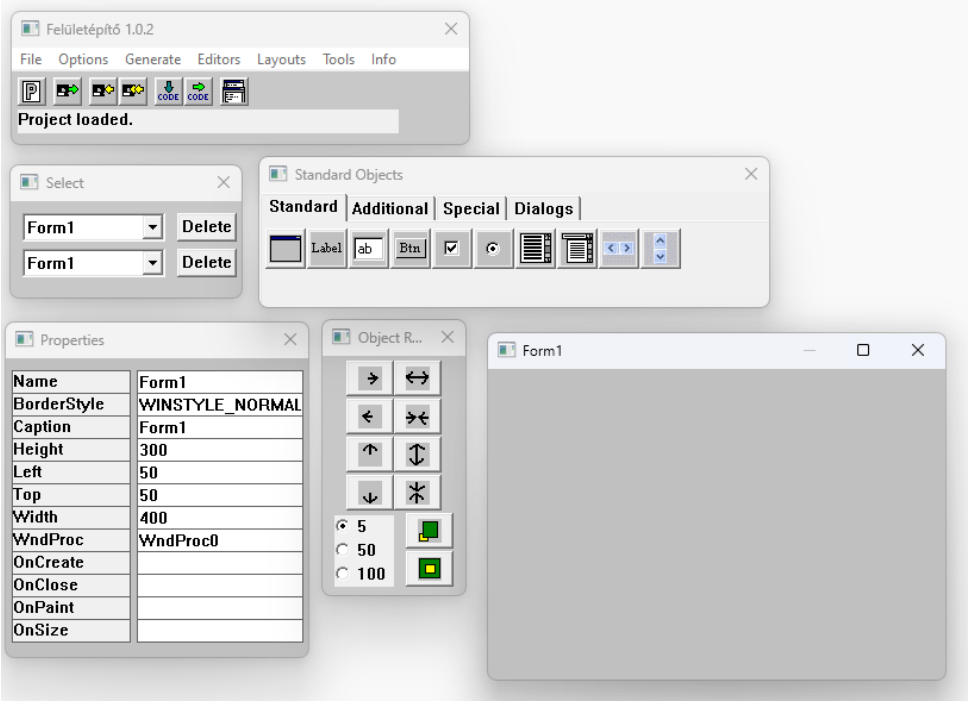


Első lépésként hozzunk létre egy új projektet!

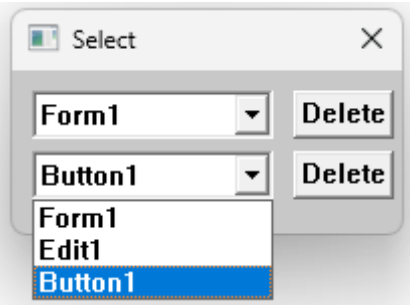
3.1 Alkalmazás megtervezése és layout-ok használata

Váltsunk át 'Design' layout-ra, majd helyezzünk el a tervezőablakban egy beviteli mezőt és két nyomógombot! Ezeket az elemeket a 'Toolbox' ablak 'Standard' füle alatt találjuk.

Valahogy így fest a végeredmény…



A ‘Select’ ablak legördülő listájából válasszuk ki a ‘Form1’ objektumot, azaz magát a leendő alkalmazásablakot!



Az ablak paramétereit így állítottam be a Caption, Height és Width mezők segítségével. A ‘Caption’ lesz az ablak címsorának felirata.

Természetesen más beállítások is lehetségesek, nekem ezek voltak szimpatikusak.

Ebben a példában a ‘Name’ tulajdonságokat változatlanul hagyom, mivel összesen 4 objektummal lesz dolgunk a tervezés során, ez pedig azért még kezelhető mennyiség.

A következő finomhangolandó elem az ‘Edit1’ objektum lesz.

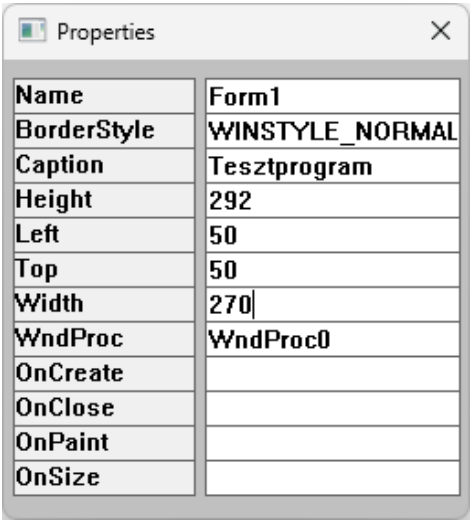
A MaxLength tulajdonságot 32-re állítjuk, azaz maximum 32 darab karaktert fog fogadni a beviteli mező.

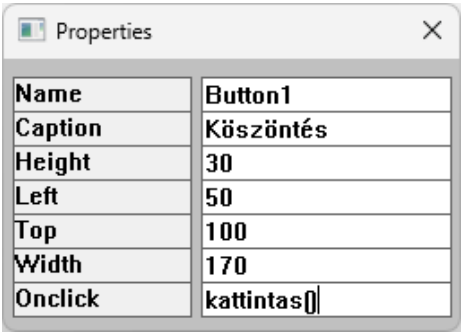
Az alapértelmezett szöveg az ‘Írja ide a nevét!’ mondat lesz, hogy teljesen egyértelmű legyen a felhasználás módja.

A ‘Button1’ objektum segítségével fogjuk megjeleníteni az üdvözlő szöveget.

A gomb feliratát a ‘Caption’ tulajdonsággal állítjuk be. Az ‘OnClick’ tulajdonságnál kell megadnunk a gombra történő kattintáshoz rendelt függvényünket/metódusunkat, ami jelen esetben a ‘kattintas()’ elnevezésű függvény lesz.

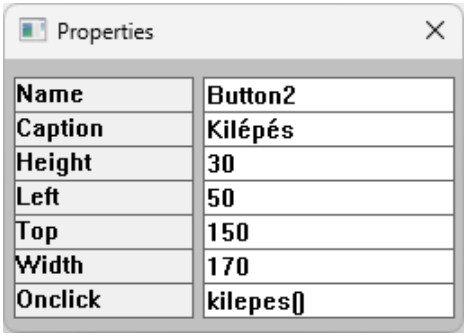
Magát a függvényt is megírjuk majd hamarosan.





Végezetül a ‘Button2’ nyomógomb feliratát állítjuk be ‘Kilépés’-re és ehhez a nyomógombhoz a ‘kilepes()’ függvényt rendeljük hozzá.

Ezzel készen is volnánk, legalábbis, ami a tervezés első fázisát illeti.



Most legeneráljuk az alkalmazás forráskódját. Ehhez kattintsunk a főablak 

gombjára! Az eredményt azonnal meg is tekinthetjük. Ehhez kattintsunk az ‘Editors’ menüben a ‘Generated code’ menüpontra. Azonnal meg is jelenik az ablak, benne a generált kóddal.

Nagyon fontos: ennek az ablaknak a tartalmát soha ne módosítsuk kézzel, mert minden változtatásunk elvész, amint újabb kódgenerálást kezdeményezünk! A generált kóddal kapcsolatos változtatásokat (másik gombfeliratot szeretnénk stb.) mindig a vizuális tervezőeszközökkel végezzük!

A generált kódban látható, hogy a tervezés során létrehozott objektumok adatai, azonosítói létrejöttek.

```
#define OBJ_ID100 100
#define OBJ_ID101 101
#define OBJ_ID102 102
HWND Form1;
HWND Edit1;
HWND Button1;
HWND Button2;
```

Az ablak eseménykezelő metódusában pedig:

```
 case WM_CREATE:
 /*Initializing*/;
 Edit1=CreateWindow(TEXT("edit"),TEXT("Írja ide a nevét!")
  ,WS_CHILD | WS_VISIBLE | WS_BORDER | ES_AUTOHSCROLL,50,50,170,30
  ,hwnd,(HMENU)(OBJ_ID100),((LPCREATESTRUCT)lParam)->hInstance,NULL);
 SendMessage(Edit1,EM_SETLIMITTEXT,32,0);
 Button1=CreateWindow(TEXT("button"),TEXT("Köszöntés")
  ,WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_MULTILINE,50,100,170,30
  ,hwnd,(HMENU)(OBJ_ID101),((LPCREATESTRUCT)lParam)->hInstance,NULL);
 Button2=CreateWindow(TEXT("button"),TEXT("Kilépés")
  ,WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_MULTILINE,50,150,170,30
  ,hwnd,(HMENU)(OBJ_ID102),((LPCREATESTRUCT)lParam)->hInstance,NULL);
```

Természetesen magának az ablaknak is annak rendje s módja szerint létrejön a kódja.

```
wndclass0.style=CS_HREDRAW | CS_VREDRAW;
 wndclass0.lpfnWndProc=WndProc0;
 wndclass0.cbClsExtra=0;
 wndclass0.cbWndExtra=0;
 wndclass0.hInstance=hInstance;
 wndclass0.hIcon=LoadIcon(NULL,IDI_APPLICATION);
 wndclass0.hCursor=LoadCursor(NULL,IDC_ARROW);
 wndclass0.hbrBackground=(HBRUSH)GetStockObject(LTGRAY_BRUSH);
 wndclass0.lpszMenuName=NULL;
 wndclass0.lpszClassName=TEXT("WIN0");

 if (!RegisterClass(&wndclass0))
  {MessageBox (NULL,HIBA_00,TEXT("Program Start"),MB_ICONERROR);return
0;}

Form1=CreateWindow(TEXT("WIN0"),
 TEXT("Tesztprogram"),
```

```
WINSTYLE_NORMAL,
50,
50,
270,
292,
NULL,
NULL,
hInstance,
NULL);
```

A generált kód azonban ebben a formában még nem lenne fordítható, ugyanis még nem készítettük el a 'kilepes()' és 'kattintas()' eseménykezelő függvényeinket.

Ezt most gyorsan pótoljuk. Váltsunk át az egyik kódszerkesztő nézetre, mondjuk a 'Horizontal coding' layout-ra!

A 'Main Declarations' ablakba beírjuk a függvényeink prototípusát, tehát:

```
void kattintas(void);
void kilepes (void);
```

A 'MainCode' ablakba a saját függvényeink törzseit írhatjuk be. A kilepes() függvény a könnyebbik eset, mivel itt egyszerűen elküldjük az ablaknak saját maga részére a WM_CLOSE üzenetet, ami kezdeményezi az ablak és ezzel az egész alkalmazás bezárását.

A picit nehezebb dió az üdvözlő szöveg kiírása. Modális dialógusablakokat a MessageBox Windows API függvénnyel írathatunk ki. A generált kód tartalmaz két segédfüggvényt is, amelyekkel egyszerűbben használhatjuk ezeket.

```
void ShowMessage(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo)
{MessageBox(kuldo,uzenet,cim,MB_OK);}
```

```
int QuestionBox(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo)
{return MessageBox(kuldo,uzenet,cim,MB_YESNO);}
```

Számunkra a ShowMessage segédfüggvény lesz az érdekes most. Arra kell csak figyelni, hogy az 'Edit1' szöveges beviteli mező tartalmát 16bites karakterként (wide char) kérdezzük le, mert a ShowMessage is ezt várja. Átírhatnánk a ShowMessage-et ANSI-ra is, ami a 'MessageBoxA' Windows API függvényt használja, de most nem bonyolítjuk túl ezt a dolgot.

```
void kattintas(void)
{
  wchar_t nev[33], uzenet[64];
  wcscpy_s(uzenet,L"Üdvözlöm ");

  GetWindowText(Edit1,nev,sizeof(nev)/2);
  wcscat_s(uzenet, nev);
  wcscat_s(uzenet, L" !");

  ShowMessage(uzenet, TEXT("Példaprogram"), Form1);
}

void kilepes(void)
{
  SendMessage(Form1,WM_CLOSE,0,0);
}
```

Most generáltassuk le a kódot még egyszer, íme az eredmény. Alább vastagon kiemeltem azokat a kódrészeket, amiket a szerkesztés során érintettünk.

```
/*
Created by Feluletepito
*/
```

```
#include <windows.h>
#include <commdlg.h>
#include <commctrl.h>
#include <richedit.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

/*global variables*/
#define HIBA_00 TEXT("Error:Program initialisation process.")
#define IDC_STATIC -1
#define LVM_SELECTED 2
#define WINSTYLE_NORMAL (WS_OVERLAPPED | WS_SYSMENU |
WS_THICKFRAME | WS_MAXIMIZEBOX | WS_MINIMIZEBOX)
#define WINSTYLE_DIALOG (WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU)
#define WINSTYLE_NONESIZEABLE (WS_OVERLAPPED | WS_SYSMENU |
WS_MINIMIZEBOX)
#define WINSTYLE_CAPTIONONLY (WS_DLGFRAME)
#define WINSTYLE_NOCAPTON (WS_POPUP)

FILE *file1;
HINSTANCE hInstGlob;
int SajatiCmdShow;
POINT MousePos;

void ShowMessage(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo);
int QuestionBox(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo);

void kattintas(void);
void kilepes (void);

#define OBJ_ID100 100
#define OBJ_ID101 101
#define OBJ_ID102 102
HWND Form1;
HWND Edit1;
HWND Button1;
```

```
HWND Button2;

LRESULT CALLBACK WndProc0(HWND,UINT,WPARAM,LPARAM);
char szClassName[]="WindowsApp";

int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,PSTR
szCmdLine,int iCmdShow)
{static TCHAR szAppName[]=TEXT("StdWinClassName");
 HWND hwnd;
 MSG msg;
 WNDCLASS wndclass0;
 LoadLibrary(TEXT("RICHED32.DLL"));
 LoadLibrary(TEXT("COMCTL32.DLL"));
 SajatiCmdShow=iCmdShow;
 hInstGlob=hInstance;

 wndclass0.style=CS_HREDRAW | CS_VREDRAW;
 wndclass0.lpfnWndProc=WndProc0;
 wndclass0.cbClsExtra=0;
 wndclass0.cbWndExtra=0;
 wndclass0.hInstance=hInstance;
 wndclass0.hIcon=LoadIcon(NULL,IDI_APPLICATION);
 wndclass0.hCursor=LoadCursor(NULL,IDC_ARROW);
 wndclass0.hbrBackground=(HBRUSH)GetStockObject(LTGRAY_BRUSH);
 wndclass0.lpszMenuName=NULL;
 wndclass0.lpszClassName=TEXT("WIN0");

 if (!RegisterClass(&wndclass0))
  {MessageBox (NULL,HIBA_00,TEXT("Program Start"),MB_ICONERROR);return
0;}

Form1=CreateWindow(TEXT("WIN0"),
 TEXT("Tesztprogram"),
 WINSTYLE_NORMAL,
 50,
 50,
 270,
```

```
 292,
 NULL,
 NULL,
 hInstance,
 NULL);

 ShowWindow(Form1,SajatiCmdShow);
 UpdateWindow(Form1);
 while(GetMessage(&msg,NULL,0,0))
  {
   TranslateMessage(&msg);
   DispatchMessage(&msg);
  }
  return msg.wParam;
}

LRESULT CALLBACK WndProc0(HWND hwnd,UINT message,WPARAM
wParam,LPARAM lParam)
{int s1;
 HDC hdc;
 PAINTSTRUCT ps;
 RECT rect;

 switch(message)
 {
  case WM_CREATE:
  /*Initializing*/;
  Edit1=CreateWindow(TEXT("edit"),TEXT("Írja ide a nevét!")
   ,WS_CHILD | WS_VISIBLE | WS_BORDER | ES_AUTOHSCROLL,50,50,170,30
   ,hwnd,(HMENU)(OBJ_ID100),((LPCREATESTRUCT)lParam)->hInstance,NULL);
  SendMessage(Edit1,EM_SETLIMITTEXT,32,0);
  Button1=CreateWindow(TEXT("button"),TEXT("Köszöntés")
   ,WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_MULTILINE,50,100,170,30
   ,hwnd,(HMENU)(OBJ_ID101),((LPCREATESTRUCT)lParam)->hInstance,NULL);
  Button2=CreateWindow(TEXT("button"),TEXT("Kilépés")
   ,WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_MULTILINE,50,150,170,30
   ,hwnd,(HMENU)(OBJ_ID102),((LPCREATESTRUCT)lParam)->hInstance,NULL);
```

```
return 0;
case WM_HSCROLL:
 switch(LOWORD(wParam))
 {;
 }
return 0;
case WM_VSCROLL:
 switch(LOWORD(wParam))
 {;
 }
return 0;
case WM_NOTIFY:{
 return 0;}
case WM_LBUTTONUP:
  MousePos.x=LOWORD(lParam);
  MousePos.y=HIWORD(lParam);
return 0;
case WM_MBUTTONUP:
  MousePos.x=LOWORD(lParam);
  MousePos.y=HIWORD(lParam);
return 0;
case WM_RBUTTONUP:
  MousePos.x=LOWORD(lParam);
  MousePos.y=HIWORD(lParam);
return 0;
case WM_COMMAND:
  MousePos.x=LOWORD(lParam);
  MousePos.y=HIWORD(lParam);
 switch(LOWORD(wParam))
 {/*Processing commands*/;
   case OBJ_ID101:kattintas();break;
   case OBJ_ID102:kilepes();break;
 }
 return 0;
case WM_SIZE:
 return 0;
case WM_PAINT:
```

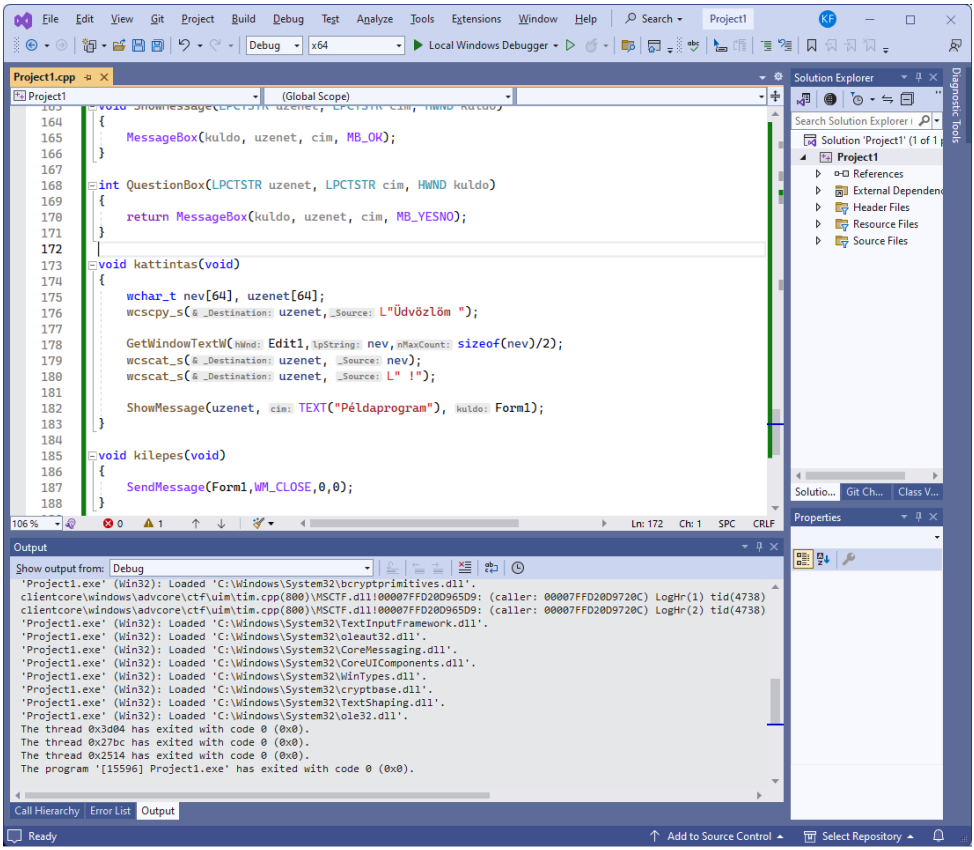
```
  hdc=BeginPaint(hwnd,&ps);
  EndPaint(hwnd,&ps);
  return 0;
 case WM_CLOSE:
  DestroyWindow(hwnd);
  return 0;
 case WM_DESTROY:
  PostQuitMessage(0);
  return 0;
 }
return DefWindowProc(hwnd,message,wParam,lParam);
}

void ShowMessage(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo)
{MessageBox(kuldo,uzenet,cim,MB_OK);}

int QuestionBox(LPCTSTR uzenet,LPCTSTR cim,HWND kuldo)
{return MessageBox(kuldo,uzenet,cim,MB_YESNO);}

void kattintas(void)
{
   wchar_t nev[33], uzenet[64];
   wcscpy_s(uzenet,L"Üdvözlöm ");


   GetWindowText(Edit1,nev,sizeof(nev)/2);
   wcscat_s(uzenet, nev);
   wcscat_s(uzenet, L" !");

   ShowMessage(uzenet, TEXT("Példaprogram"), Form1);
}

void kilepes(void)
{
   SendMessage(Form1,WM_CLOSE,0,0);
}
```

A kódot közvetlenül ki is másolhatjuk a ‘Generated Code’ ablakból, de a 

gombra kattintva a projekt GenSource könyvtárába elmenthetjük külön forrásfájlként is, ennek neve fixen MAIN.CPP lesz.

Fontos, hogy a generált kód ANSI kódolású, ezért az ékezetes betűk esetében utólag szükség lehet az érintett sztringek “finomhangolására” is.

A kód ezután például a Visual Studio Community Edition egy Windows Desktop projektjének kódszerkesztőjébe bemásolva azonnal fordítható és kipróbálható.



3.2 Vizuális eszköztár

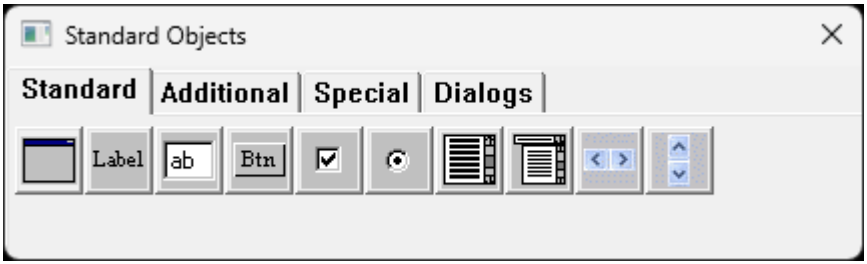
A leendő alkalmazás grafikus felületének kialakítását számos segédeszköz segíti. Ezeket tekintjük most át közelebbről.

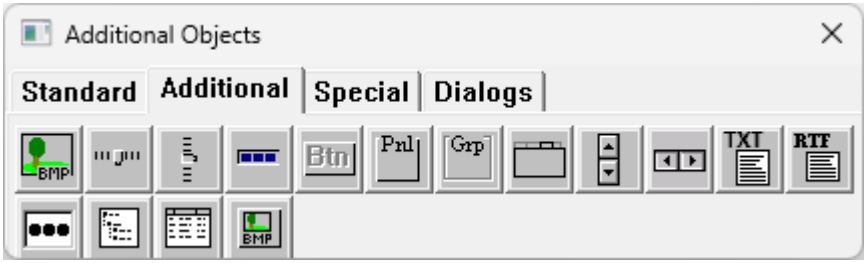
3.2.1 Toolbox ablak

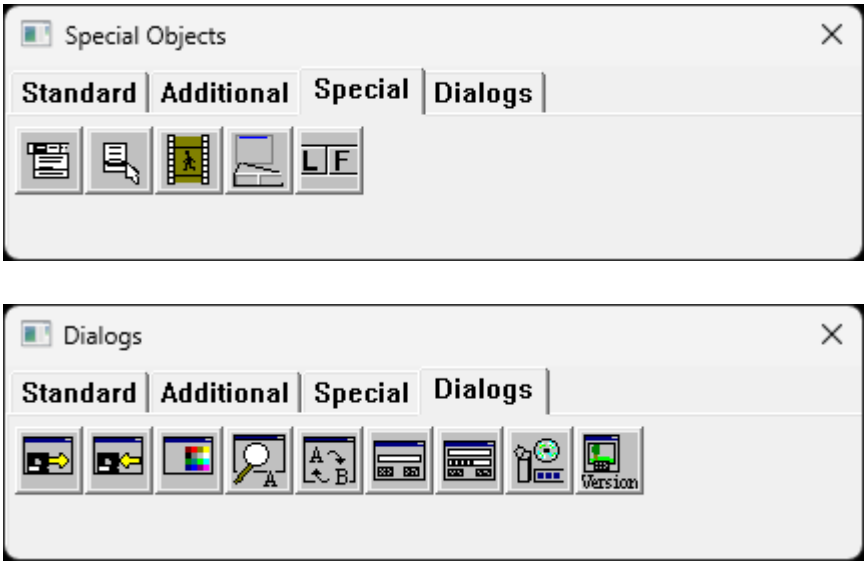
Az egyes Windows vezérlők, ill. speciális objektumok négy csoportba felosztva érhetőek el:

- Standard: a leggyakrabban előforduló vezérlők.
- Additional: speciálisabb, több beállítást lehetővé tevő vezérlők.
- Special: Menük és néhány speciális objektum.
- Dialogs: dialógusablakok, ill. előre létrehozott ablaktervek.

A legtöbb vezérlő konkrét vizuális megjelenéssel is rendelkeznek, mások inkább csak kódszinten jelennek meg.

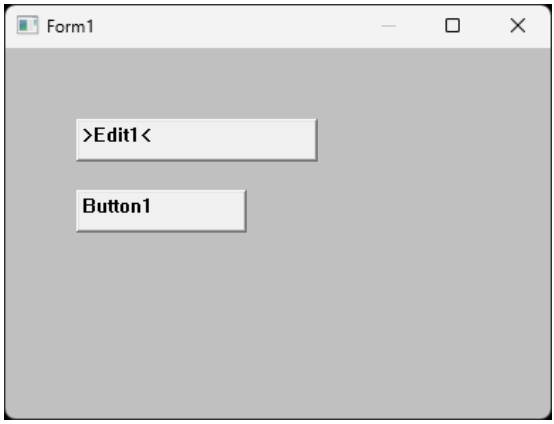






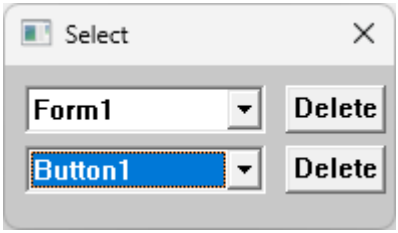
3.2.2 Form Designer

Az egyes vezérlők a kiválasztásuk után a ‘Form Designer’-re kerülnek, mely pontosan felveszi a leendő alkalmazásablak méreteit. A tervezőablak a Windowsban megszokott módon, egyszerűen átméretezhető, a megfelelő értékek azonnal frissülnek a ‘Properties’ ablakban.



3.2.3 Select ablak

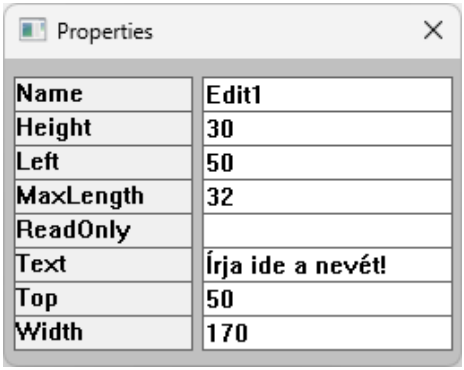
Az egyes vezérlőket ebben a dialógusablakban választhatjuk ki. Megjegyzendő, hogy egy elemet a 'Form designer'-en is kiválaszthatunk bármikor, egyszerű rákattintással.



3.2.4 Properties ablak

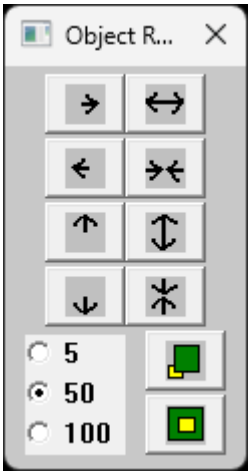
Az aktuálisan kiválasztott vezérlőelem tulajdonságait ellenőrizhetjük és akár módosíthatjuk is. A kódgenerálás az itt beírt értékeket is felhasználja. A tulajdonságok listája nélkülözi a teljességet, de a legfontosabbak elérhetőek itt.

Az Edit típusú szövegbeviteli mező esetében a következő tulajdonságokat érhetjük el itt:



3.2.5 Object Resizer

Ezzel az ablakkal a Form designer-en elhelyezett vezérlők áthelyezése és átméretezése végezhető el 5, 50 és 100 pixeles léptékekkel. Minden esetben előtte ki kell választani a Form designer-en a változtatni kívánt objektumot.



Ugyanitt változtatható meg az esetlegesen átlapolt, egymást takaró objektumok fizikai sorrendje is.

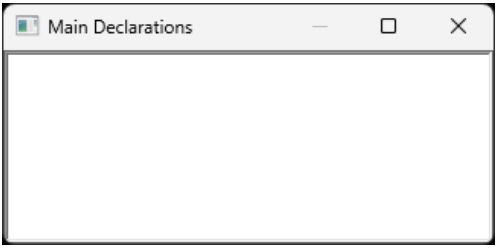
3.3 Kódszerkesztő ablakok

Az alkalmazás egyedi, nem generálható, hanem kézzel megírandó kódját szintén megírhatjuk a Felületépítővel, bár ez nem kötelező, mivel a végső kódot úgyis külső fordítóval kell elvégezni.

Kevesebb kód desetén azonban könnyebb és hasznos is, ha egy helyen van minden kód, amit később is módosíthatunk.

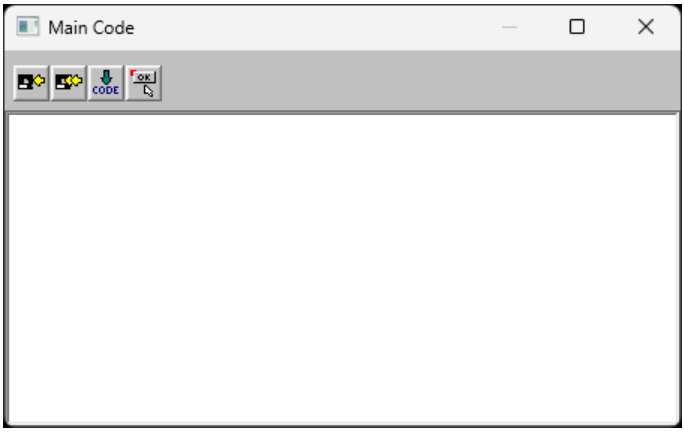
3.3.1 Deklarációk kezelése

A deklarációk, függvény prototípusok megadása ebben az ablakban történhet.



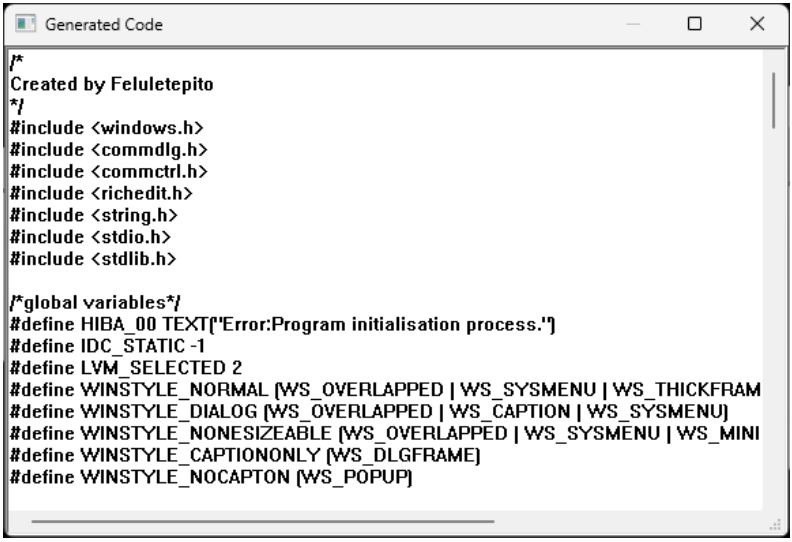
3.3.2 Fő kód szerkesztése

A függvénytörzsek megírását ebben az ablakban lehet elvégezni. Az ablakban szintén közvetlenül elérhető a Projekt mentése, a kódgenerálás, valamint az átváltás a Design ablak elrendezésre.



3.3.3 Generált kód megtekintése

A Felületépítő által megírt kód ebben az ablakban érhető el. Szerkeszteni itt nem érdemes, mivel minden egyes kódgenerálás első lépésben törli a tartalmát.



3.3.4 Jegyzetek

Saját magunk számára jegyzeteket, gondolatokat, ötleteket is rögzíthetünk a projektünkhöz. Ennek a 'Notes' ablaknak a tartalma semmilyen hatással nem bír a létrehozandó konktér programkódra, tényleg csak jegyzetelésre való.

3.4 Kódgenerálás

A kódgenerálás bármikor kedzeményezhető a 

gombra történő kattintással. A generált kódot pedig egyszerűen exportálhatjuk a 

nyomógombbal. Generált kód elmentése .C forrásfájlban. Ennek neve MAIN.CPP lesz és a projekt GENSOURCE mappájában jön létre.

3.5 Alkalmazás lefordítása

A Felületépítővel generált kódból külső fejlesztőeszközökkel szállított fordítóprogramok segítségével készíthetünk valódi, futtatható Windows alkalmazást.

Ilyen eszközök például a Visual Studio Community Edition, mint fentebb utaltam rá, vagy a Bloodshed Dev C++ is.

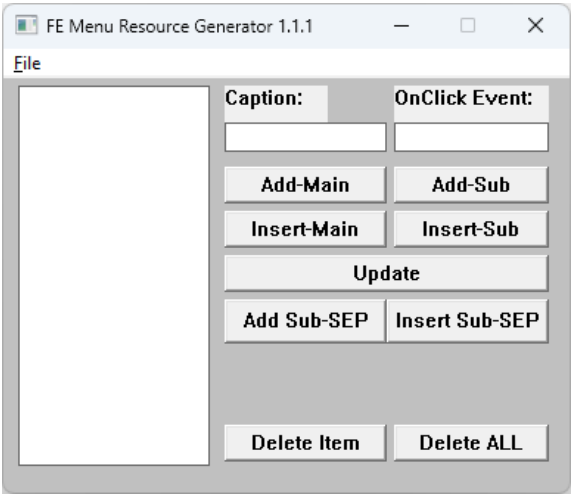
4 Segédprogramok bemutatása

A Felületépítő különböző segédprogramjai méginkább megkönnyítik az alkalmazásfejlesztés egyes lépéseit. Egyeseknek már csak "történeti" jelentősége van és ezért hagytam meg.

FIGYELEM: az egyes segédprogramok újra lettek ugyan fordítva, de működésük nem lett teljeskörűen leteszteltve, ezért előfordulhat, hogy egyes funkciók nem működnek megfelelően. Az esetlegesen szükséges javítások kiadása nem garantált!

4.1 Menü erőforrások készítése

Az egyik eszköz, amivel rengeteg gépelést tudunk megspórolni, a menü erőforrás fájl készítő segédprogram, az 'FE Menu Resource Generator'.



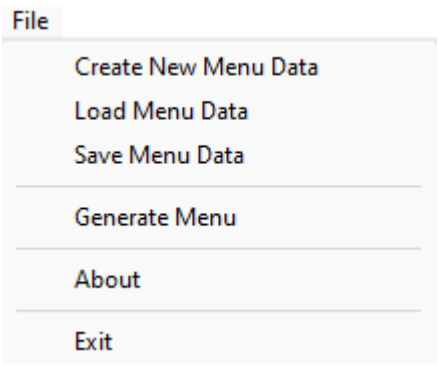
Mindössze annyi dolgunk van, hogy felépítsünk egy menü szerkezetet, megadva az egyes menüpontok nevét és a

kiválasztásukkor meghívandó függvények nevét, majd a 'File' – 'Generate Menu' menüpont kiválasztásával elkészíthetjük és külön fájlokba exportálhatjuk az összes kódot, amit a végső alkalmazás forráskódjában felhasználhatunk.

Ezenkívül, mintegy mini-projektekként, az egyes menüszerkezeteket el is menthetjük és később vissza is tölthetjük. Így komplexebb igényeket is ki tud szolgálni az alkalmazás.

Fontos, hogy csak kétszeres mélységű menüszerkeszet kezelése lehetséges ezzel az eszközzel.

Menüpontokat az 'Add-Main', ill. 'Insert-Main' gombokkal tudunk létrehozni. Az 'Add Sub-SEP' gombbal a menün belüli vízszintes elválasztó vonalakhoz szükséges kódot is tudunk generáltatni.

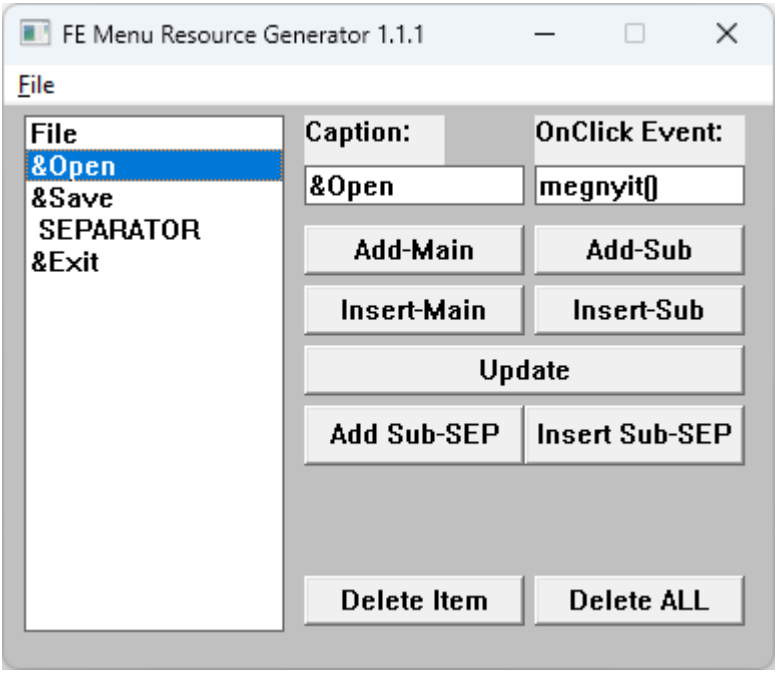


A '-Sub' feliratú gombokkal almenüket (2. szerkezeti mélység) tudunk kezelni.

Meglévő menüpontokat frissíteni is tudunk, ehhez előbb a listában ki kell választanunk a megváltoztatni kívánt elemet, majd az 'Update' gombbal végezhetjük el a frissítést.

Egyes elemet, vagy az összeset a 'Delete' és 'Delete ALL' feliratú gombokkal tudunk.

Az alábbi képen például egy klasszikusnak mondható Fájl menü szerkezetét láthatjuk, a segédprogrammal szerkesztve.



A fenti kis projektünket először el kell mentenünk, mielőtt a kódgenerálást elvégezhetnénk.

Ha minden megvan, az alábbi fájlok jönnek létre (a projekt neve ebben a példában: 'MYMENU'):

mymenu.h	2024. 02. 26. 22:07	C/C++ Header	1 KB
MYMENU.kod	2024. 02. 26. 22:07	KOD fájl	1 KB
mymenu.mnd	2024. 02. 26. 22:07	MND fájl	1 KB
MYMENU.rc	2024. 02. 26. 22:07	Resource Script	1 KB

Az .MND fájl maga a menü projekt fájlunk.

A MYMENU.H fájlt include-olnunk kell a leendő alkalmazásunk fő kódjában. A header fájl tartalma:

```
#define MENUTITLE_0 "&File"
#define IDM_MENU0 500
#define MENUTITLE_1 "&Open"
#define IDM_MENU1 501
#define MENUTITLE_2 "&Save"
#define IDM_MENU2 502
#define MENUTITLE_4 "&Exit"
#define IDM_MENU4 504
```

A MYMENU.RC fájlt a fő kódban az ablak létrehozásakor kell megadnunk. Az erőforrás fájl tartalma:

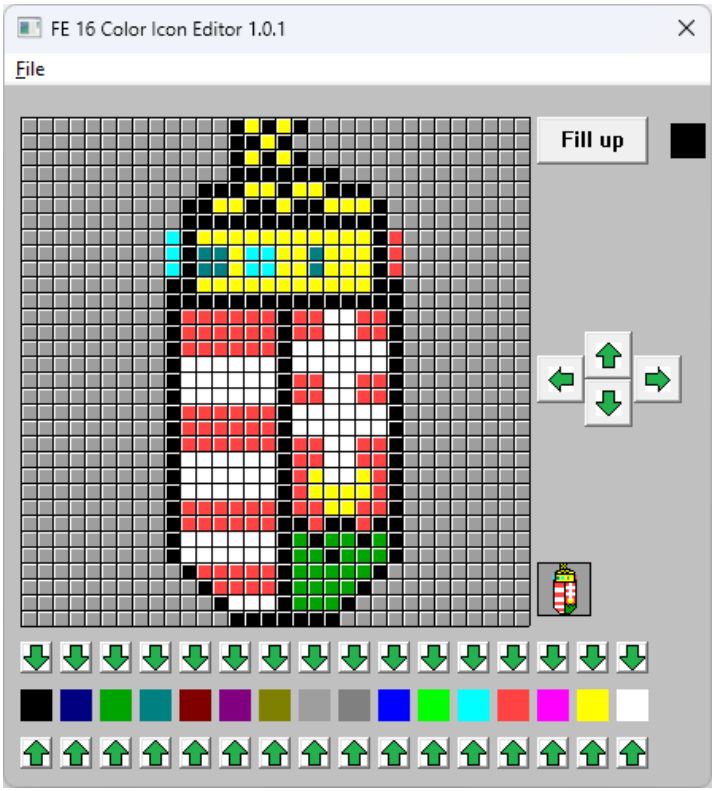
```
#include "MYMENU.h"

MYMENU MENU DISCARDABLE
BEGIN
 POPUP MENUTITLE_0
 BEGIN
  MENUITEM MENUTITLE_1, IDM_MENU1
  MENUITEM MENUTITLE_2, IDM_MENU2
  MENUITEM SEPARATOR
  MENUITEM MENUTITLE_4, IDM_MENU4
 END
END
```

A MYMENU.KOD a menüpontok kiválasztásához használatos eseménykezeléshez nyújt előre elkészített segédkódokat:

```
case IDM_MENU1:megnyit();break;
case IDM_MENU2:elment();break;
case IDM_MENU4:kilepes();break;
```

4.2 Ikonok létrehozása

16 színű ikonokat mint erőforrásokat az 'FE 16 Color Icon Editor' segédprogrammal hozhatunk létre.



A program kezelőfelülete nagyon egyszerű.

Az ablak alsó részén, a lefelé mutató gombokra történő kattintással egy-egy, a Windows által előre definiált színt választhatunk ki rajzolási színnek.

A felfelé mutató nyilakkal minden színt külön-külön lecserélhetünk egy egyedi színre.

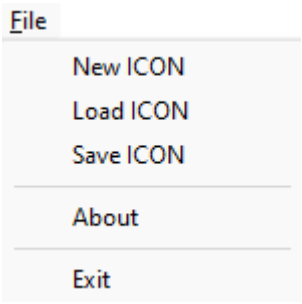
Az aktuálisan kiválasztott szín az ablak jobb felső sarkában is látható lesz.

Ugyanitt, a 'Fill up' feliratú gomb segítségével az atuálisan kiválasztott színnel az egész ikont kitörölhetjük.

A kissé lentebb található négyirányú gombokkal az egész ikon kép tartalmát eltolhatjuk a gombbal megegyező irányba, a felszabaduló sorok/oszlopok pedig az aktuális rajzolási színnel lesznek kitöltve.

Magát az ikont az ikon nagyított képére történő kattintással tujduk megrajzolni, képkockánként. Szerkesztés közben, a jobb alsó sarokban valós időben frissül az ikon valós méretű előnézeti képe.

Az ikonokat elmenthetjük és megnyithatjuk a 'File' menü segítségével.



Fontos, hogy itt nincsen köztes projekt fáll, a segédprogram az ikonokat közvetlenül tudja megnyitni és menteni is!

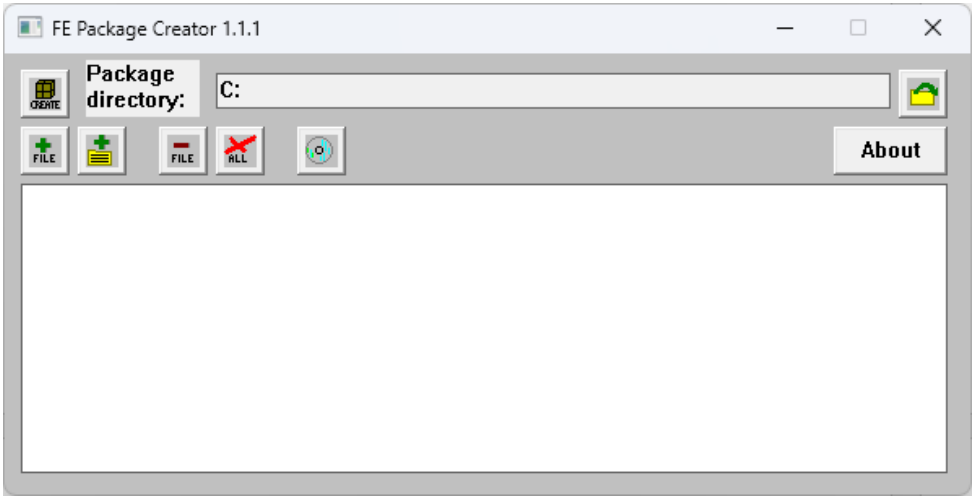
4.3 Csomagkészítés

Az ‘FE Package Creator’ segítségével egyfajta telepítőprogramot hozhatunk létre, amit a Felületépítő INSTALLER.EXE segédprogramjával tudunk felhasználni. Lényegében egy egyszerű teleptőkészletet hozhatunk létre.

Az ‘FE Package Creator’ segítségével ki tudjuk választani a csomagba elhelyezendő fájlok listáját, majd a ‘CREATE’ gombbal létrehozhatjuk magát a csomagot.

Ilyenkor létrejön egy PACKAGE.DAT nevű fájl a megadott célkönyvtárban, ami a listában megadott fájlokat tartalmazza, titkosítás, vagy tömörítés nélkül.

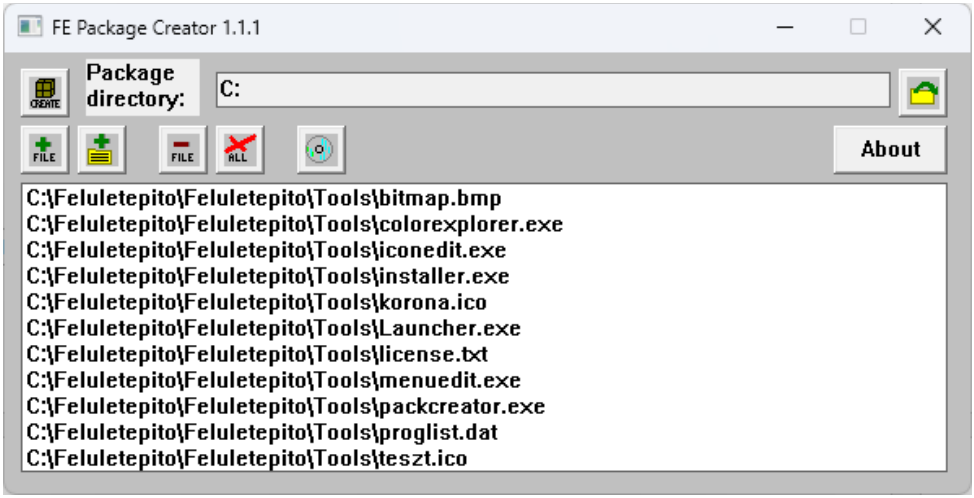
A másik fájl a PACKAGE.INF fájl lesz, ami a telepítő segédprogram számára tartalmazza a .DAT fáj tartalmának tételes felsorolását.



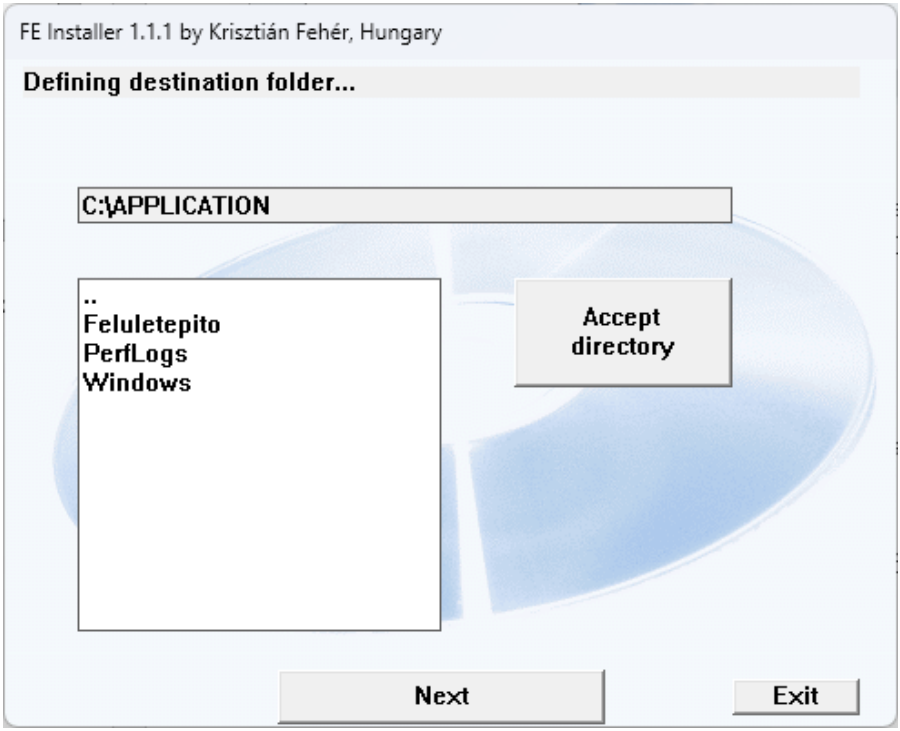
A csomag helye megadható.

A kis CD ikon segítségével az ún. telepítési vezérlőfájlokat hozhatunk létre. (Régi verzió maradvány, jelenleg már nem funkcionál, ignorálható.)

Az alábbi példában a Felületépítő saját segédprogramjaiból állítunk össze egy telepítőkészletet.



Az INSTALLER.EXE fájllal egy könyvtárba elhelyezve kezdeményezhetjük a telepítést.

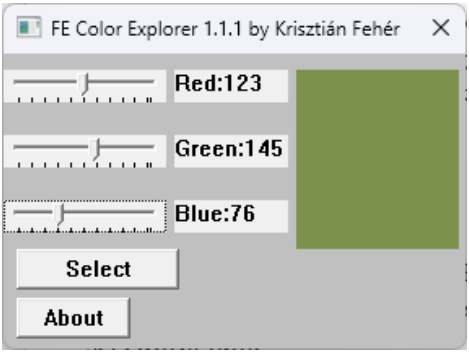


A Felületépítő ‘TOOLS’ mappája tartalmaz egy LICENSE.TXT nevű fájlt is. Ezt a teleptőprogram be tudja olvasni, így egyedi liszensz szövegeket stb. tudunk megjeleníteni a felhasználónak a telepítés elindítása előtt.

4.4 Színek kikeverése

Az ‘FE Color Explorer’ egy RGB színek kikeverésére, az egyes komponensek pontos meghatározására használható apró, egyszerű segédprogram.

A ‘Select’ gobmra kattintva a Windows előre definiált színeit is elérhetjük és ki is választhatjuk.

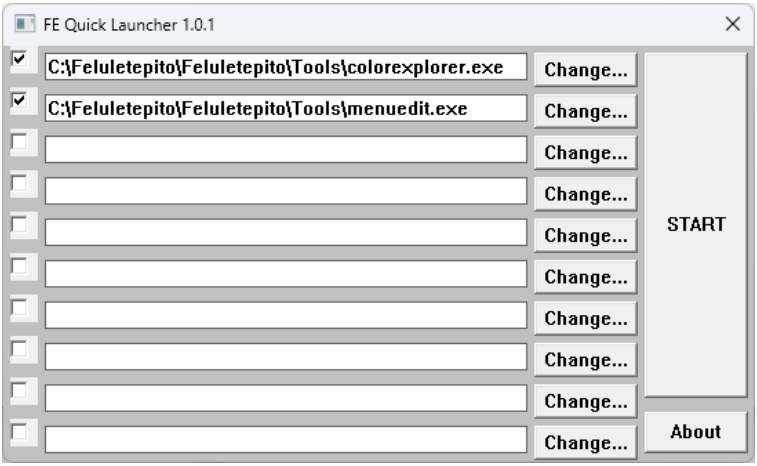


4.5 Kedvenc alkalmazások kötegelt elindítása

Az ‘FE Quick Launcher’ segédprogram segítségével 10 alkalmazás indítását végezhetjük el a ‘START’ feliratú gombra történő kattintással.

Persze ehhez előbb ki kell választani az elindítandó alkalmazásokat (‘CHANGE’ feliratú gombok).

A listát a program automatikusan elmenti, így elég csak egyszer beállítani. Egyes alkalmazások indítását deaktiválhatjuk anélkül is, hogy törölnénk az elérési útvonalát. Ehhez egyszerűen csak ki kell venni a hozzá tartozó pipát a bal oldalon.



5 Haladó funkciók használata

A Felületépítő rendelkezik néhány speciális funkcióval is, melyek segítenek még rugalmasabbá tenni a munkát.

5.1 Gyorsított ablakszerkesztés

Előfordulhat olyan igény, ami bizonyos vezérlők nagy mennyiségű létrehozását igényli oly módon, hogy ezek elrendezése egyenletes legyen.

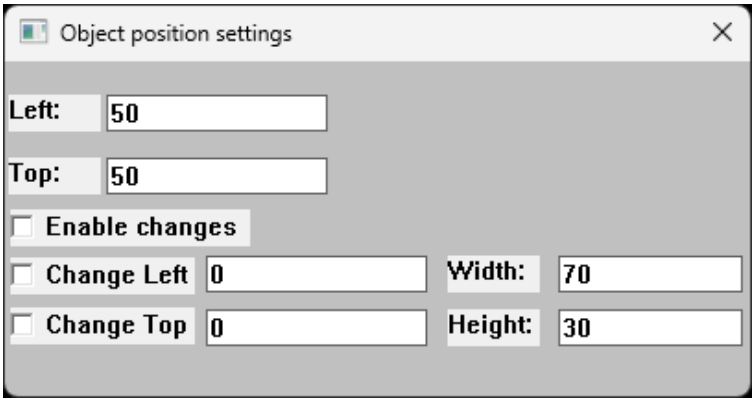
Ez megoldható dinamikus módon, programkódból is, ám elvégezhetjük ezzel az 'Object position settings' párbeszédablak használatával is, amit az 'Options' menüben érhetünk el..

A 'Left' és 'Top' értékek a létrehozandó új objektumok kezdőpozícióját adják meg.

A 'Change Left' és 'Change Top' a vízszintes és függőleges pozíció léptetést fogja eredményezni. Ha például 100 pixelt adunk meg a 'Change Left' értékeként, akkor minden új vezérlő 100 pixellel jobbra eltolva lesz pozicionálva.

A 'Width' és 'Height' értékek az új objektumok alapértelmezett méreteit adják meg.

Az ebben az ablakban megadott beállításokat a Felületépítő csak akkor használja, ha előtte bejelöljük az 'Enable changes' jelölőnégyzetet.



5.2 Saját layout létrehozása

A File menü 'Save Layout' és 'Load Layout' menüpontjainak segítségével saját ablakelrendezéseket is kimenthetünk későbbi újrafelhasználásra. Ezek a fájlok .LAY kiterjesztésű bináris adatfájlok.

Akár az előredefiniált (a Layouts menüből elérhető) elrendezéseket is felülírhatjuk a Felületépítő LAYOUTS mappájában, ily módon teljesen egyedi, igényeinknek megfelelő ablakelrendezéseket létrehozva.

5.3 Ablaktervek egyéni kimentése

A 'Form Designer' tartalma szintén egyedileg elmenthető, a 'File' menü 'Export Form' és 'Import Form' menüpontjainak segítségével.

Importálás esetén a betöltött ablaktartalom hozzáadódik a projektünkhöz és teljes mértékben integrálódik abba.

Ez a funkció lehetővé teszi, hogy külön-külön is készíthessünk ablakterveket, amelyeket aztán egy projektbe lehet integrálni.

A szerző eddig megjelent könyvei

A szerző saját kiadású könyvei:

- Kliens-szerver programozás tömören 2. rész (2023)
- Kliens-szerver programozás tömören (2023)
- Sensor API programozás tömören (2023)
- Többszálú win32 programozási alapismeretek (2022)
- JavaScript és AJAX fénysebességgel (2022)
- Programozz C nyelven! (2022) – ÚJ KIADÁS
- Direct2D programozás dióhéjban (2022) – ÚJ KIADÁS
- Kibervédelmi tesztalkalmazások programozása (2022) – ÚJ KIADÁS
- Androidos alkalmazásfejlesztés Kotlin nyelven 3. rész (2022)
- Automatizált és manuális szoftvertesztelési alapismeretek (2021)
- Játékprogram egy hétvége alatt! 2. rész (2021) – ÚJ KIADÁS
- Multi-GPU grafika CUDA alapokon 2. rész (2021)
- Játékprogram egy hétvége alatt! 2. rész (2021)
- Játékprogram egy hétvége alatt! 1. rész (2021)
- A Windows keményvonalas programozása (2020)
- Androidos alkalmazásfejlesztés Kotlin nyelven 2. rész (2020)
- Multi-GPU grafika CUDA alapokon (2020)
- Programozz C nyelven! (2020)
- Számítógépvásárlási túlélőkönyv (2020)
- Így készíts térképet otthon! (2020)
- Androidos alkalmazásfejlesztés Kotlin nyelven (2020).
- CUDA programozási alapismeretek (2020)
- HTML5 Canvas grafika programozása (2020)
- Nyomtatók programozása (2020)
- Gamepad programozása PC-n (2020)
- Direct2D programozás dióhéjban (2020)
- Kibervédelmi tesztalkalmazások programozása (2020)
- Szupergyors rajzoló algoritmusok (2020)
- 3D programozás (2019)

A szerző saját kiadású ingyenes PDF kiadványai:

- Alkalmazásfejlesztés a Felületépítő segítségével (2024)
- Navigációs szoftverek és digitális térképek fejlesztése Androidra (2023)
- Androidos szoftverfejlesztés Flexben alapfokon (2023)

A szerző saját kiadású könyvei angol nyelven:

- Krisztián's Win32 API programming guide (2020)
- Multi-GPU graphics programming with CUDA (2020)
- Direct2D programming in a nutshell (2020)
- HTML5 Canvas pocket guide (2020)
- Programming cyber security test applications (2020)
- Printer programming basics (2020)
- CUDA programming basics (2020)
- Gamepad programming on PC (2020)
- Programming CUDA for extreme performance (2019)
- Lightweight 3D programming in a weekend (2019)

A szerzőnek a BBS-INFO Kiadó gondozásában megjelent könyvei:

- Mesterséges intelligencia avagy Pandora digitális szelencéje (Fehér Krisztián - dr. Kökényesi Bartos Attila - Bártfai Barnabás, 2020)
- Hogyan írj extrém gyors programot? - Bevezetés a CUDA programozásba (2019)
- Hackertechnikák - Útmutató valódi hacker módszerek biztonságos kipróbálásához (2018)
- Alkalmazásfejlesztés Android Studio rendszerben (2018)
- Direct2D - DirectX programozás egyszerűen (2018)
- Grafikus és játékalkalmazások programozása (2017)
- AJAX adatbázis-kezelés Javascript alapon (2017)
- Kezdő hackerek kézikönyve - Avagy informatikai támadások és kivédésük (2016)
- Szoftvertesztelési alapismeretek (2016)
- Android kézikönyv - avagy okostelefonok kezelése laikusoknak (Bártfai Barnabás – Fehér Krisztián, 2015)
- Navigációs szoftverek fejlesztése Androidra (2015)
- Androidos szoftverfejlesztés alapfokon (2014)

Látogassa meg a szerző weboldalát!
Információk a legújabb könyvekről,
hírek, érdekességek,
webshop!
https://feherkrisztian.com