A Gambas bevetésen
Egy egyszerű képnézegető fejlesztése

Az előző számban olvashattuk, hogyan is épül fel és miért lehet jelentős ez a BASIC alapú fejlesztőkörnyezet. Ebben a cikkben egy egyszerű képnézegető alkalmazás készítésébe vágunk bele, természetesen az eddig tárgyalt Gambas nevű eszköz segítségével.

Azonban, nem szorítkozzunk csak azon elemekre melyek egy kép megjelenítéséhez szükségesek, előfordulhatnak feleslegesnek tűnő dolgok. A cikk célja most, hogy az olvasó betekintést nyerjen az eszköz használatába, ami pedig nem mindig a legoptimálisabb megoldásokról szól.

Mint már olvashattuk az előző számban, a ***Gambas*** igen barátságos környezetet nyújt a fejlesztéshez. Egy projekt elindítása gyerekjáték a varázsló igénybevételével. Lássuk mit is szeretnénk elkészíteni.

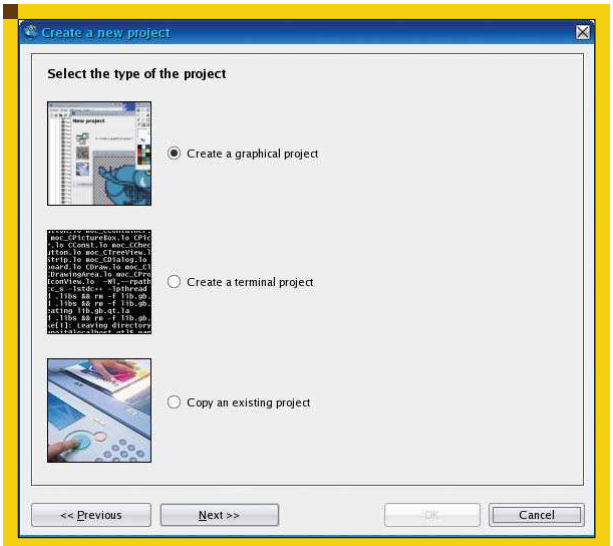
A ***Gambas*** segítségével létre fogunk hozni egy képnézegető alkalmazást, mely alkalmas lesz a fejlesztőkörnyezet által kezelt képformátumok közül ***jpg***, ***png***, ***gif*** és ***bmp*** képek megjelenítésére. Ha ez menni fog, akkor kiegészítjük az alkalmazásunkat egy pár aprósággal, hogy ezzel többet tudjunk meg a eszközök tárházában található dolgokról. Nem egy profi programra törekszünk csak ízlelgetjük kicsit a ***Gambast***.

Vágjunk hát bele. Indítsuk el a környezetet, majd válasszuk a ***„New project...”*** menüt az új projekt létrehozásához. Egy grafikus felületű alkalmazást fogunk készíteni, tehát válasszuk a ***Create graphical project*** pontot (*2. ábra*), melynek a ***Gambas*** által felhasznált neve (első mező) ***„kepnezegeto”*** lesz (*4. ábra*). Itt hasznos dolog elkerülni az ékezetek használatát.

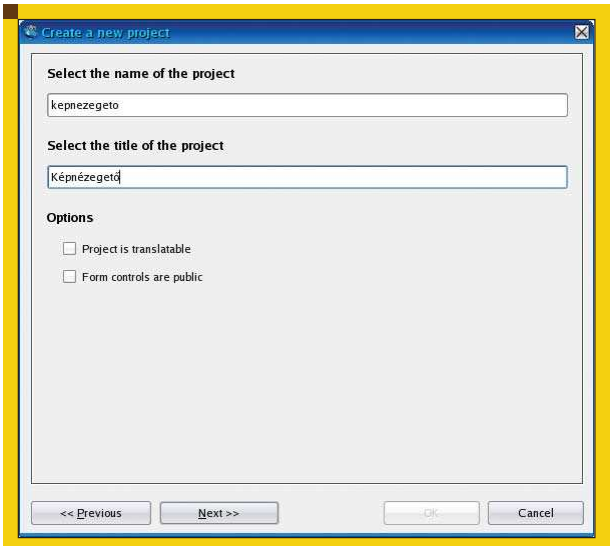
A második mezőben a projekt címét adhatjuk meg. Itt már el lehet ereszteni a kemény ékezeteket is. A projektet – mint minden más fejlesztőkörnyezet esetén – célszerű egy különálló könyvtárba menteni, hogy az egy alkalmazáshoz tartozó dolgok mind egy helyen legyenek tárolva (*3. ábra*). Nem kötelező létrehoznunk kézzel, külön könyvtárat, a program minden esetben készít egy, a projekt nevével megegyezőt. Ha minden rendben ment, akkor végre elénk tárul a ***Gambas IDE***, ahol a kis kék ***Gambas*** hüvelykujja segítségével jelzi nekünk, hogy minden feltétel teljesült ahhoz, hogy a munkát megkezdhessük (*5. ábra*).

Baloldalon a projekt ablakban kezdjük a munkát. Kattintsunk jobb egérgombbal a ***Forms*** ágra, a projekt fájában és adjunk hozzá a struktúrához egy formot a ***New*** menüpont, ***Form..*** pontjának segítségével (*6. ábra*). A felbukkanó ablakban adjuk meg a form

1. ábra Új projekt létrehozása

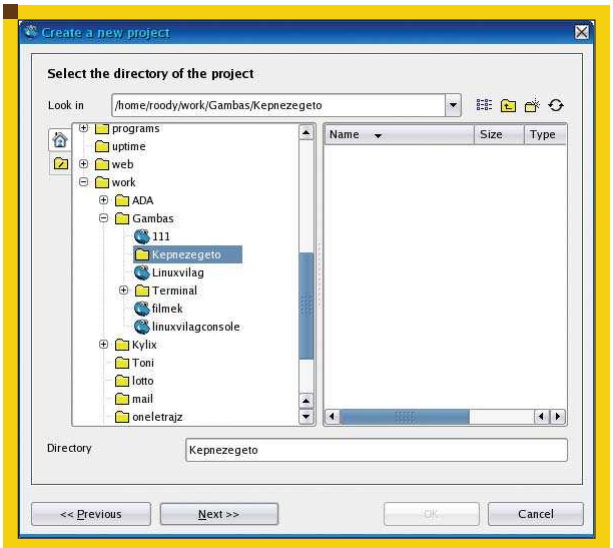


2. ábra Grafikus felületű programot készítünk

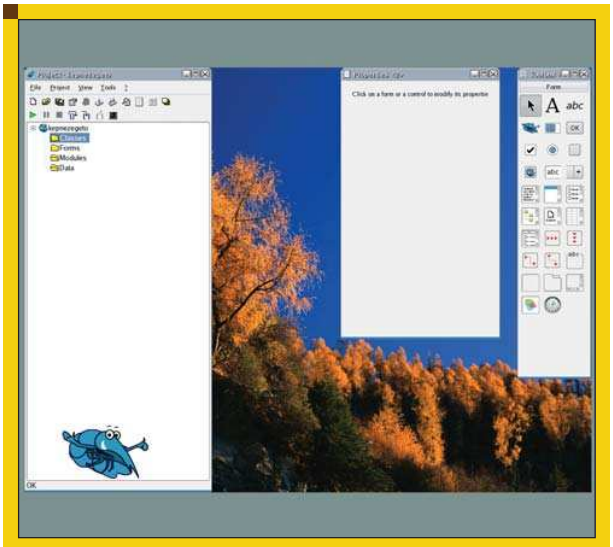


3. ábra Nevezzük el az alkalmazást

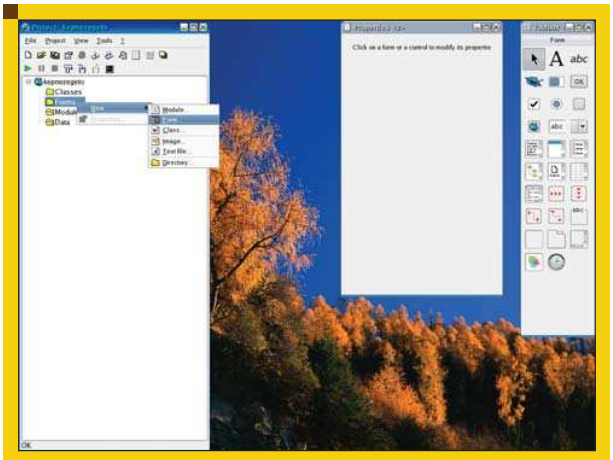
nevét: ***frmKep***, ezen kívül mást nem kell tennünk. Megjelenik a formunk mely minden munkánk *„ágyát”* fogja képezni.
Keressük meg a ***Properties*** ablakot, miután tettünk egy kattintást a formunkra. Ezentúl csak egyszerű jelöljük ki azt a komponenst, melynek tulajdonságát látni szeretnénk! A tulajdonságait egy elemnek az ***F4*** billentyű segítségével is előhívhatjuk.
Nevezzük el a formunkat. Legyen a ***Name*** mező értéke: ***frmKep***. Itt az *„frm”* az objektum típusára utal. Hasznos szokás. Érdemes megszokni, a későbbi módosítások, vagy a kód újrafelhasználását elősegítve, hogy beszédes változókat, elnevezéseket alkalmazunk. Adjuk meg a form szélességét (***Width***), valamint magasságát (***Height***) képpont, de az egérrel is módosíthatunk a méreteken. Ez bármely komponensre igaz. Az ablak fejlécében pedig szerepeljen majd a *„Képnézegető”* felirat, ez legyen hát a ***Text*** értéke (*7. ábra*).
Mivel talán az egyik legkedveltebb komponens a vizuális elemek között a lista, ezért a kényelmes használat érdekében próbáljuk megjeleníteni a könyvtár tartalmát egy listában, melynek segítségével navigálhatunk a későbbiekben. Adjunk hozzá egy ***ListBoxot*** a formunkhoz. Neve legyen ***lstFileok***. Természetesen akárhogy elrendezhetjük a formunkon a dolgokat. Törekedjünk az arányos elrendezésre. Az emberi szem mindig jobban kedveli az egyenlő arányokat, mint az aszimmetriát. Most pedig hagyjuk el egy kicsit a vizuális fejlesztést egy egyszerű kód létrehozásának erejéig.
Könnyítsük meg a képnézegetőnk használatát egy nagyon egyszerű fájlböngésző megvalósításával. Keressük meg a kódablakot és hozzunk létre egy ***fileokat_frissit*** eljárást egy



4. ábra Adjuk meg a projekt könyvtárának nevét



5. ábra Kezdhetjük a fejlesztést



6. ábra Form hozzáadása a projekthez

1. Lista Végigmegyünk az sKonyvtar paraméterben megadott könyvtár tartalmán

```
FOR EACH sFile IN Dir(sKonyvtar)

  IF Stat(sKonyvtar &/ sFile).Hidden THEN
  CONTINUE
  ENDIF

  IF IsDir(sKonyvtar &/ sFile) THEN
    lstFileok.Add("!" & sFile,"[" & sFile & "]")
  ELSE
    lstFileok.Add(sFile,sFile)
    ENDIF
  NEXT
```

sKonyvtar string paraméterrel. Ez az eljárás fogja végezni fájlok listába olvasását:

```
PUBLIC SUB fileokat_frissit
➥(sKonyvtar AS String)

END SUB
```

Először is deklaráljunk egy `sFile` nevű, `string` típusú változót. Ebben fogjuk majd tárolni az éppen vizsgált könyvtár elemeit.

```
DIM sFile AS String
```

Töröl a listát az `lstFileok.Clear` paranccsal. Megnézzük, hogy a paraméterként kapott könyvtár a gyökér-e. Ha igen, akkor nem tesszük be a lista elejére a visszaugráshoz szükséges két pontot.

```
lstFileok.Clear

  IF sKonyvtar <> "/" THEN
  lstFileok.Add("..","..")
  ENDIF
```

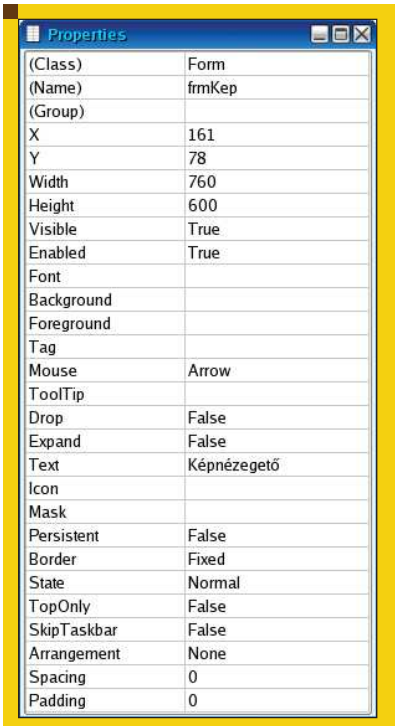
Egy listához elemet hozzáadni a `listanév.Add()` parancs segítségével tudunk, mely y két paramétert vár. Egy kulcsot, mely a listában való azonosítást segíti elő, valamit a listában szereplő szöveget. Esetünkben kulcsnak és szövegnek egyaránt megfelel a könyvtár- illetve fájlnév.
A

FOR EACH változó IN Dir(könyvtár) … *NEXT*

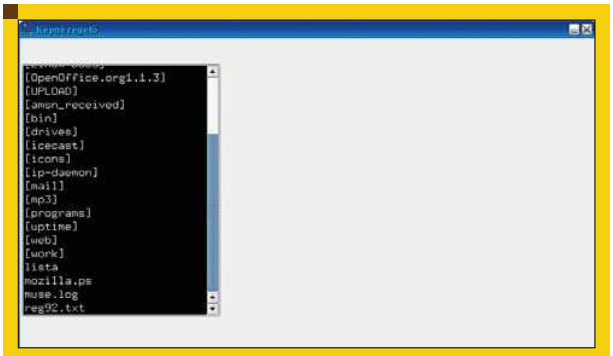
konstrukció segítségével végigszaladunk a könyvtár elemein. A ciklusmagban az aktuális elem az `sFile`-ban lesz. A rejtett állományokat figyelmen kívül hagyjuk (`CONTINUE`), valamint ha valamely elemet könyvtárnak titulál a rendszer, akkor azt [és] jelek közé tesszük. Egy állományról információkat a `Stat()` függvénnyel kérdezhetünk le. Egy egyszerű függvény az `IsDir()` pedig elmondja, hogy a paraméter könyvtárat takar vagy sem. A két függvény fájlnevet vár paraméterként, akár elérési úttal együtt. Stringeket az ***&*** jel segítségével fűzhetünk össze.
A listával megvolnánk viszont szeretnénk, hogy minden induláskor már ott legyen a könyvtár tartalma.
Az egyes komponensek eseményeit, rájuk jobb gombbal kattintva tudjuk elérni: ***Event*** menüpont. Tegyük ezt a formunkkal és válasszuk ki az ***Open*** nevű eseményt, mely a form megnyitásakor hajtódik végre.
Az esemény kódja legyen `fileokat_frissit(System.Home)`. Ez fogja garantálni, hogy induláskor a listában az éppen programot futtató tulajdonos ***home*** könyvtára fog szerepelni. A `System` objektum több ilyen hasznos tulajdonságot is hordoz. Lehetőség van még – a fentihez hasonló módon – a ***host***, ***domain***, ***nyelv*** valamint karakterkészlet és még pár rendszerparaméterek lekérdezésére.

```
PUBLIC SUB Form_Open()
  fileokat_frissit(System.Home)
END
```

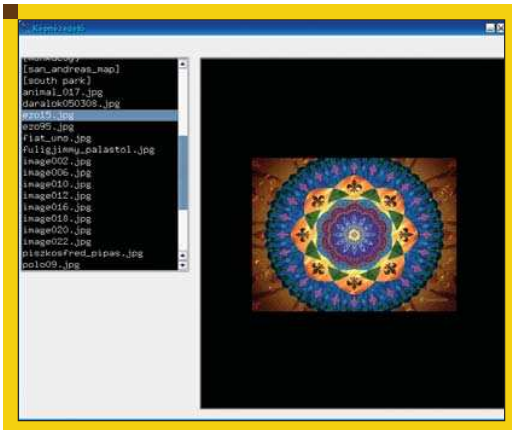
Az ***F5*** segítségével ellenőrizhetjük az eddigi eredményeket (*8. ábra*). Megnyomásával egyetemben a ***Gambas*** lefordítja a projektünket, jelezve az esetleges hibákat. Az ***F7*** segítségével csak szintaxis ellenőrzést végezhetünk, futtatás nélkül. ***Ctr + Alt + M*** kombinációval pedig kész futtatható állományt készíthetünk a projekt könyvtárában. De azért ettől még messze vagyunk.
Szeretnénk a lista tartalmát böngészni is az egér segítségével. Készítsünk hát a listához egy `Click` eseményt. Defini-



7. ábra A form tulajdonságai



■ *8. ábra* A fileok működő listája



■ *9. ábra* Végre a képek is látszanak

2. Lista Az lstFileok nevű listánk, egérkattintáshoz tartozó eseménykezelője

```
PUBLIC SUB lstFileok_Click()

  DIM sJelolt AS String
  DIM sFile AS String

  sFile = lstFileok.Item.Text

  sJelolt = Mid(lstFileok.Item.Text,2,Len
  ➥(lstFileok.Item.Text)-2)

  IF sFile=".." THEN
    fileokat_frissit(File.Dir(File.Dir
    ➥($sKonyvtar &/ sFile)))
    RETURN
  ENDIF

  IF IsDir($sKonyvtar &/ sJelolt) THEN
    fileokat_frissit($sKonyvtar &/ sJelolt)
    RETURN
  ELSE
    mutat($sKonyvtar &/ sFile)
    RETURN
  ENDIF
END
```

álunk egy `sFile` és egy `sJelolt` sztringet. Az előzőbe beolvassuk az éppen kijelölt elemet a listából. Az `lstFileok.Item.Text` értéke felel meg ennek. Ugyanezt az értéket betesszük az `sJelolt` változóba is, de az utolsó és első karakterek nélkül. Ezt a jó öreg Basic függvény, a `Mid()` segítségével fogjuk megtenni.

Ez a függvény szerintem szakembereknek igen ismerős. Három paramétert vár. Az első egy sztring a második pedig egy karakterpozíció, melytől kezdve a harmadik paraméter értékével megegyező számú elemet kapunk vissza a sztringből.

Ezt a „csonkított” értéket fogjuk használni arra, hogy megvizsgáljuk könyvtárra kattintottunk-e vagy sem.

A csonkításra a `[` és `]` miatt van szükség. Figyelnünk kell először is arra, hogy ha „..” lett kijelölve, azaz a felhasználó ugrani szeretne egy könyvtárat, akkor az megfelelően működjön. Nem kell sokat gondolkoznunk. Megint egy egyszerű megoldást kínál a *Gambas*. Létezik egy osztály `File` névvel és egy `Dir()` ***függvénnyel*** mely visszaadja a paraméterbeli állomány vagy könyvtár szülőkönyvtárát, ha a paraméter teljes elérési út.

Ha az `sJelolt` könyvtárnak bizonyul, akkor az eddigi elérési úthoz (`$sKonyvtar`) a kijelölt könyvtárat hozzáadva, a `fileokat_frissit()` eljárással beolvassuk a könyvtár tartalmát a listába, ha nem könyvtár akkor megpróbáljuk megjeleníteni a fájl tartalmát a `mutat()` eljárással. Ezt a következőkben fogjuk tárgyalni. Mivel mindegyik feltétel kizárja a másikat, ezért a feltételhez tartozó eseményeket követően a `RETURN` kulcsszót alkalmazzuk, hogy visszatérjünk az eljárás hívópontjához.

Miután végre szabadon böngészhetünk a lista segítségével állományaink között, ideje hogy végre képeket is tudjunk megjeleníteni. Mivel biztos hogy lesznek képek melyek nem férnek majd el az ablakunkban, adjunk a formunkhoz egy `ScrollView` nevű komponenst. Ennek segítségével nagyobb képeket görgetősávval tudunk majd megtekinteni. Legyen az új komponens neve: `scrlNezet`.

A `ScrollView` általában is alkalmas nagyobb tartalmak, görgetősávos megjelenítésére. Ez a komponens fogja tartalmazni a képet. `Gambasban` képeket `PictureBox` segítségével is megjeleníthetünk. Rakjunk hát egyet az `scrlNezet` közepére. Nevezzük el `picKepnek`.

Megvannak a vizuális elemek a képek megjelenítéséhez. Írjuk meg hát a `mutat()` eljárást. Ez az eljárás fogadja a listánk eseménykezelőjétől a fájlokat. Szűrjük ki, amit nem tudunk megjeleníteni. A `File.Ext()` segítségével kinyerjük a paraméterből a megadott fájl kiterjesztését. Ezt a `Lower()` függvénnyel egységesen kisbetűsre alakítjuk, hogy leegyszerűsítse számunkra az ellenőrzést. A kiterjesztésből látni fogjuk, hogy milyen állománnyal van dolgunk. A paraméterként kapott képet egy `Picture` objektumban fogjuk tárolni. A kódhoz hozzáadunk egy globális `pKep`, említett típusú objektumot. Miután inicializáltuk, a `pKep.Load()` segítségével betöltjük a képet és a

```
picKep.Picture = pKep
```

3. Lista A mutat() eljárásunk első változata

```
PUBLIC SUB mutat(sKep AS
➥String)

  DIM fArany AS Float
  DIM sKit AS String

  sKit =
Lower(File.Ext(sKep))

  IF sKit=jpg" OR sKit =
  ➥"png" OR sKit="gif" OR
  ➥sKit="jpeg" OR
  sKit="bmp" THEN
    pKep = NEW Picture
    pKep.Load(sKep)
    picKep.Picture = pKep
    Nyujtas
  ENDIF

END
```

4. Lista A Kép nyújtása menüpont eseménykezelője

```
PUBLIC SUB mnuElfer_Click()
  mnuElfer.Checked = NOT
  ➥mnuElfer.Checked
  picKep.Stretch =
  ➥mnuElfer.Checked
  Nyujtas
END
```

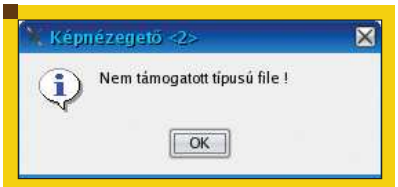
fogja biztosítani, hogy meg is jelenik. Ezután a `nyujtas()` eljárás fogja a beállításoknak megfelelően megjeleníteni a képet.
Adjunk lehetőséget a felhasználónak, hogy váltogathasson két nézetfajta között (kicsire nyújtott vagy teljes méret). Tegyük ezt menü segítségével! Egy ablakban a menüt úgy tudjuk szerkeszteni, hogy az adott formra jobb gombbal kattintunk majd innen kiválasztjuk a ***Menu editor*** pontot.
A ***Name*** mezőbe várja a ***Gambas*** a menü elnevezését, mely szerint a kódban is hivatkozni fogunk rá, a ***Caption*** mező pedig a megjelenítendő menüpont látható nevét fogja tartalmazni. A menürendszer hierarchiáját a nyilak segítségével dolgozhatjuk át. Adjunk a menünkhöz egy ***Képnézegető*** (`mnuKepnezegeto`) elemet, ez lesz a főmenü. Alá pedig vegyünk fel még kettőt: ***Kép nyújtása*** (`mnuElfer`), illetve ***Kilépés*** (`mnuKilepes`). Az `mnuElfer` menüpont esetében jelöljük ki a ***Checked*** boxot is. Erre azért van szükség, mert ez a menü kétállásúként fog működni. Segítségével fogja a felhasználó szabályozni, hogy a képet csak picire nyújtva, vagy teljes méretben szeretné látni.
Kattintsunk most a ***Kép nyújtása*** menüre. Ekkor megjelenik a hozzátartozó esemény eljárása. Írjuk be:

```
mnuElfer.Checked = NOT
➥mnuElfer.Checked
```

Ezzel biztosítjuk, hogy kétállásúként fog működni a menüpont.
A `picKep.Stretch` tulajdonságot állítsuk be a `menuElfer.Checked` értékre. Így már a menüpont hatalma alá került a `picKep` ***megjelenítési*** tulajdonsága. Majd hogy az új beállítások érvényesüljenek hívjuk meg a `nyujtas()` eljárást. Ez a kódsorozat fogja biztosítani a kép helyes elrendezését az ***scrlNezetben***. Ha kisebb a képünk mint a `ScrollBox` és a ***Kép nyújtása*** be van jelölve, akkor próbáljuk középre helyezni a képet, amennyire csak lehet, egyébként legyen a kép pozíciója a `srclNezethez` képest a 0,0 helyen. A ***nyujtas()*** eljárás végül is csak a helyes megjelenítés miatt készült, újabb elemek alkalmazására nem mutat példát, ezért külön nem is tárgyaljuk. A mellékletben természetesen megtalálható.
Adjunk még hozzá pár dolgot az alkalmazáshoz, csak hogy tudjuk hogyan is kell őket alkalmazni. Legyen mondjuk a lista alatt egy `TextBox` mely a megjelenített fájl nevét tartalmazza. A ***mutat()*** eljárásban állítjuk be az értékét, ha már biztosak vagyunk benne, hogy képpel van dolgunk. Legyen `TextBoxunk` neve `txtNev`. Ennek tartalmát a `txtNev.Text`-en keresztül érhetjük el. Szintén a lista alatt szerepeljen egy `TextArea` komponens. Ebben fogunk a képről – szintén a `mutat()` eljárásban – információkat megjeleníteni. Az információs doboz neve legyen `txaInfo`. Tartalmát ugyancsak a `txaInfo.Text` tulajdonságon keresztül érhetjük el. Mivel képünket a `pKep` objektumban tároljuk, ezért innen fogjuk az információkat szerezni. `pKep.Width` a kép szélessége, `pKep.Height` a magassága, `pKep.Depth` pedig színmélysége. Ezek után már gyerekjáték feltölteni az információs dobozt.
Az `Str$()` függvény segít nekünk, hogy egyéb típusokat szöveggé konvertáljunk, mivel a `pKep.Depth`, `pKep.Width` és `pKep.Height` értékek mind számok, és közvetlenül számot szöveggel nem fűzhetünk össze. Figyelmeztessük a felhasználót, ha nem jó állománnyal próbálkozik! A `message()` függvény nagyon is alkalmas erre, mely egy figyelmeztető ablakot jelenít meg. Paraméterekben meg lehet adni, hogy milyen gombokkal rendelkezzen az ablak. Ha csak egy szöveget adunk meg paraméterként, akkor az egy ***OK*** gomb kíséretében jelenik meg (*10. ábra*). A függvény a választott gombok alapján visszatérési értéket is ad.
Csak hogy alkalmazásunkon tanulhassunk még pár dolgot, lássuk hogyan is működik a ***Gambas*** időzítő komponense. Ennek segítségével egy egyszerű animációt fogunk készíteni.

5. Lista A mutat() eljárás módosítása

```
PUBLIC SUB mutat(sKep AS
➥String)

  ...

  IF sKit="jpg" OR sKit =
➥"png" OR sKit = "gif" OR
➥sKit="jpeg" OR sKit="bmp"
➥THEN
    ...
    txtNev.Text = sKep
    txaInfo.Text = "Szélesség:
    ➥" & Str$(pKep.Width) &
    ➥"\nMagasság: " &
    Str(pKep.Height) & "\nSzín
    ➥bitek: " & Str(pKep.Depth)
  ELSE
    txtNev.Text = ""
    message("Nem támogatott
    ➥típusú file !")
  ENDIF

END
```

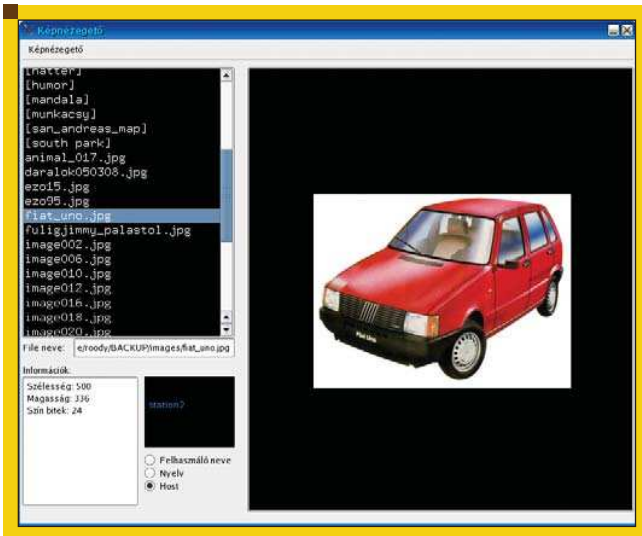


10. ábra Egy egyszerű figyelmeztetés

6. Lista Az időzítő eseménykezelője, mely minden 500. ms-ban végrehajtódik

```
PUBLIC SUB tIdozito_Timer()

  DIM sSzoveg AS String

  IF rbFelhasznalo.Value THEN
    sSzoveg = System.User
  ENDIF

  IF rbNyelv.Value THEN
    sSzoveg = System.Language
  ENDIF

  IF rbHost.Value THEN
    sSzoveg = System.Host
   ENDIF

  drwRajz.Clear
  drwRajz.Refresh
  Draw.Begin(drwRajz)
  Draw.Font.Name = "sans"
  Draw.Font.Size = 8
  Draw.ForeColor =
  ➥Int(Rnd(&HFFFFFF))
  Draw.Text(sSzoveg,Int
➥(Rnd(drwRajz.Width-Len
➥(sSzoveg)*6)),
  Int(Rnd(drwRajz.Height-10)))
  Draw.End
END
```

A *Gambas* lehetőséget ad a programozónak, hogy alakzatokat rajzoljon egy bizonyos felületre. Ez a bizonyos felület pedig a `DrawingArea` komponens. Adjunk hát hozzá egyet a formhoz, a neve pedig legyen `drwRajz`. Hogy ne maradjon ki a repertoárból egy másik igen gyakori komponens a `RadioButton` adjunk hozzá 3 darabot a formunkhoz. Valamint szükségünk lesz még egy `Timer` objektumra is. A Timerünknek adjunk körülbelül 500-as intervallumot, ez ms-ban értendő. Valamint, hogy elkerüljük a meglepetéseket, állítsuk az `Enabled` tulajdonságot `True`-ra.



11. ábra A kész alkalmazásunk

A `Timer` `Timer` nevű eseménye van még hátra. Ez fogja megjeleníteni az apró animációt.
Először is ellenőrizzük, hogy mely `RadioButton` van megjelölve, ennek függvényében választunk egy megjelenítendő szöveget `System.User`, `System.Host` vagy `System.Language`. Ezután töröljük a rajzfelületet: `drwRajz.Clear`. A törlést a `drwRajz.Refresh` parancs követi. Azért szükséges, hogy a felületet frissítse, azaz ténylegesen végrehajtódjon a törlés. A rajzolást megelőzi a `Draw.Begin(drwRajz)` parancs, azaz kiválasztjuk a felületet, amelyre alkotunk. A `Draw` osztály igen széleskörű. Rajzolhatunk segítségével vonalat, négyszöget, ellipszist, szöveget és így tovább. `Draw.FillColor`, `Draw.FillStyle` adják meg az alakzatok kitöltési színét illetve stílusát. Rajzoláshoz tartozik még szorosan a `Color` osztály, melynek elemei a gyakoribb színek szóbeli megnevezése, például: `Color.White`.
A véletlenszám generátor általában törtet ad vissza, ezért minden generált számot egésszé kell alakítanunk, ha azt a helyzet úgy kívánja. Tehát az *Int()* lecsonkolja a törtet egy egészre, `Rnd(x)` pedig [0..x] intervallumon generál véletlen valós számot. A rajzolást a `Draw.End` parancs zárja le.
Azért, hogy nagyjából teljes legyen az alkalmazás, a ***Kilépés*** menü eseményébe írjuk be: `frmKep.Close`. Ezzel biztosítjuk a menün keresztüli kilépést a programunkból.

Kaptunk végül egy igen egyszerű képnézegető alkalmazást (*11. ábra*), és egy kis lökést a ***Gambas*** megismeréséhez, ezek mellett remélem sok kedvet is a további használathoz, és a számos lehetőség felfedezéséhez. Remélhetőleg a példa sokaknak segítségére szolgál a ***Gambashoz***. Sajnos mindenre nem lehet kitérni egy cikkben, de ha ezek a dolgok könnyen mentek, akkor a továbbiakban sem lesz gond. Miért is lenne? Hiszen mint láttuk, egyszerű és ésszerű parancsokkal, felépítéssel várja a fejlesztőkörnyezet a kezdő és haladó programozót egyaránt. A ***BASIC*** nyelv már magában egyszerű, a parancsokat szinte már előre lehet tudni. Használatát ezért is ajánlanám kezdő programozók számára, bár az igen jó felépítésű súgó még nem igazán van kidolgozva tartalmilag, sok a hiányos, dokumentálatlan elem. Ennek ellenére igen hatékony eszköz, ha időt szeretnénk megtakarítani a fejlesztéssel. Minden érdeklődőnek sok hasznos, értékes időtöltést kívánok!

Radics Péter
(peter.radics@gmail.com)
Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, web-design, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.