Bővítsük a GIMP-et

Miután megismerkedtünk a GIMP legfontosabb képességeivel, a használat során bizonyára felmerült bennünk a kérdés: tud-e ennél többet a program, vagy lehet-e valamilyen módszerrel a gyakran ismétlődő műveletek végrehajtását felgyorsítani. Természetesen a készítők már a kezdet kezdetén gondoltak erre a lehetőségre, így a válasz mindkét kérdésre igen!

A *GIMP* kétféle megoldást is kínál a bővítésre, mindkettőt egy-egy programnyelven keresztül. A korábbi változatokban a ***scheme*** programozási nyelvet használhatjuk, amelynek azonban kissé nehezen érthető a szintaxisa. A következő oldalakon az egyszerűbb megoldással fogunk foglalkozni, mégpedig a ***Python*** programnyelven készített beépülő modulok készítésével.

A modulok elkészítéséhez azért is célszerű a ***Python*** nyelvet választani, mert véleményem szerint ez az egyik legkönnyebben elsajátítható nyelv, mely egyszerűsége ellenére is alkalmas összetett programok elkészítésére. A nyelv egyik nagy előnye az is, hogy könnyedén illeszthető más programkönyvtárakhoz, így napjainkban már használhatunk például ***GTK+*** vagy ***Qt*** alapú grafikus felületet is a ***Python*** programjainkban.

Az alábbiakban a nyelv alapelemeit szeretném bemutatni, mindössze olyan mélységben, amennyire a ***GIMP*** modulok készítéséhez szükségünk lesz rá.

Minden programnyelvben találhatunk programvezérlési szerkezeteket, mint amilyenek a ciklusok és a feltétel vizsgálatra szolgáló utasítások. A ***Python***-ban egy számláló ciklus az *1. listán* látható módon készíthető el. Szintén az 1. listán látható az elöl tesztelő ciklus és a feltételvizsgálat formailag helyes megadása is. Ha jól megfigyeljük a listát azonnal látható a formátum egyszerűsége. A ciklus vagy vizsgálat kezdetét a `:` karakter jelzi, majd az utasítások következnek.

Az utasítások beírásánál figyelni kell arra, hogy a ***Python*** értelmező a tabulátorok vagy szóközök alapján határozza meg az utasításblokk végét. Tehát a második listán látható utasítások mindegyike helytelen, mert nem kezdtük beljebb az utasításblokkba tartozó sorokat. Aki írt már egyszerűbb vagy összetettebb programokat, annak nem lesz nehéz megszoknia a ***Python*** formai követelményeit sem, hiszen már az áttekinthetőség kedvéért sem írjuk folyamatosan, mindenféle tagolás nélkül a sorokat egymás alá.

A ***Pythonban*** természetesen lehetőségünk van a korszerű programozási nyelvekben megszokott függvények és osztályok használatára is. A *3. listán* látható egy függvény, mely megfelel a nyelv formai követelményeinek. A függvényünk nem csinál mást, csupán kiírja az első paraméter értékét, majd az első és a második paraméter összegét. A ***Pythonban*** nem szükséges sokat foglalkozunk a változók típusával, elegendő csak arra figyelni, hogy azonos típusú változókkal végezzünk műveleteket. Alapvetően három típussal dolgozhatunk. Számokkal, szövegekkel és listákkal. Természetesen nem adhatunk össze egy számot egy szöveggel és listával sem.

1. lista

```
# számláló ciklus
for a in range(1,10):
        utasítások

# elöl tesztelő ciklus
i = 0
while i<10:
        i = i + 1

# feltételvizsgálat
if FELTÉTEL:
        utasítások
else:
        utasítások
```

2. lista

```
# számláló ciklus helytelen
for a in range(1,10):
print a

# elöl tesztelő ciklus helytelen
i=0
while 1:
print i
if i==10:
break
```

3. lista

```
# Függvény megadása
def fuggvenyneve(parameter1, parameter2):
        utasításblokk
        print parameter1
        print parameter1 + parameter2
```

4. lista

```
#!/usr/bin/env python

from gimpfu import *

# A fo fuggveny, ami a munkat vegzi
def anigif(image, drawable, neww, newh,
filename):
    newimage = pdb.gimp_file_load(filename,
    ➥filename)
    newlayer = newimage.active_layer
    newlayer.scale(neww, newh, 0)
    pdb.gimp_selection_all(newimage)
    pdb.gimp_edit_copy(newimage.active_layer)
    mylayer = gimp.Layer(
            ➥image, "from " + filename,
            ➥neww, newh,
            ➥image.base_type, 100, NORMAL_MODE)
    image.add_layer(mylayer, 0)
    mylayer.set_offsets((image.width-neww) / 2,
            ➥(image.height-newh) / 2)
    image.active_layer = mylayer
    pdb.gimp_edit_paste(mylayer, 1)
    pdb.gimp_floating_sel_anchor(
       image.floating_selection
    )

# Modul bejegyzese
register(
    "python_fu_anigif",
    "Animated GIF creator",
    "Create animated GIF image from single
    ➥images",
    "Zoltan Fabian",
    "Zoltan Fabian",
    "2004",
    "<Image>/Python-Fu/GifAnim",
    "RGB*",
    [
       ( PF_INT, "neww", "New width", 100),
       ( PF_INT, "newh", "New height", 100),
       ( PF_FILE, "filename", "Filename", "")
    ],
    [], anigif)

main()
```

5. lista

```
register (  modul_neve, "modul leírása",
➥"modul bővebb leírása",
➥"készítő_neve", "copyright információk",
➥"dátum",
➥"modul_helye_a_menüben", "KÉPTÍPUSOK",
➥paraméterek[], visszatérési_értékek[],
➥a_modul_fő_függvénye
 )
```

Mindössze ennyi elegendő a modulok készítésének megértéséhez, tehát a most egy példán keresztül bemutatom kedves olvasóimnak, hogyan is lenne célszerű elkészíteni egy *GIMP* kiegészítő modult. A *4. listán* látható a példa, igyekeztem minél rövidebben használható rutint írni, de természetesen a továbbiakban bővíteni is fogjuk a programunkat. A *GIMP* alapértelmezésben a megadott könyvtárakban keresi a kiegészítő modulokat, tehát a beállítások párbeszédablakában meg kell határoznunk a modulokat tartalmazó könyvtárakat. Amennyiben ezt a lépést kihagyjuk, a program a felhasználó saját könyvtárában lévő ***.gimp-2.0/plug-ins*** könyvtárban vizsgálja az állományokat. A ***Python*** nyelven készült modulok írásakor két fontos dolgot kell megjegyeznünk. Az egyik, hogy a `from gimpfu import *` sornak szerepelnie kell a modul forrásában, mert így használhatjuk a *GIMP* beépített lehetőségeit. A másik fontos dolog pedig, a `register` függvény meghívása a megfelelő paraméterekkel. A függvény paraméterezése az *5. listán* olvasható.
A paraméterek közül bővebb magyarázatra szolgál a modul helyét meghatározó `'modul_helye_a_menüben'` paraméter. Ez a paraméter határozza meg, hogy a *GIMP*-ben milyen menüpontokon keresztül találjuk meg a kiegészítő modult. A megfelelő formátum az alábbi példákból jól látható:

```
"<Image>/Python-Fu/AnimGif"
vagy
"<Toolbox>/Xtns/Python-Fu/AnimGif"
```

A *GIMP* indulásakor alapértelmezésben nincs megnyitva semmilyen kép, tehát amennyiben a *GIMP* eszköztárának menüpontjai között szeretnénk találkozni az elkészített modullal, akkor a második megoldás szerint a az eszköztárban az ***Xtns*** menüben a ***Python-Fu*** pontot választva találhatjuk meg az ***AnimGif*** modult, melynek forrásával a 4-es listán ismerkedhettünk meg. A másik fontos paramétere a `register` függvénynek, a `KÉPTÍPUSOK` nevet viseli. Ezzel a paraméterrel határozhatjuk meg, hogy a *GIMP* milyen képekre engedélyezze a modul használatát. A típus lehet *'RGB*'*, *'GRAY*'* vagy *'INDEXED'* de természetesen a * karakter helyett használhatjuk a pontos értéket is, mely meghatározza az elfogadható színmélységet. Például, amennyiben csak 24-bites képekre használható a modul, akkor a megfelelő érték az `RGB24` lesz. Több értéket megadhatunk a karakterláncban, vesszővel elválasztva.
Moduljaink készítése során előfordulhat, hogy a modulnak paramétereket szeretnénk átadni a *GIMP*-ből. Meg kell jegyezni, hogy minden modul fő függvényének első

6. lista

```
PF_INT
PF_FLOAT
PF_STRING
PF_INTARRAY
PF_FLOATARRAY
PF_STRINGARRAY
PF_COLOR
PF_REGION
PF_IMAGE
PF_LAYER
PF_CHANNEL
PF_TOGGLE
PF_BOOL
PF_FONT
PF_FILE
PF_BRUSH
PF_PATTERN
PF_GRADIENT
```

két paraméterében a *GIMP* átadja az éppen aktuális rajzolásra használt felületet és az aktuális képet, tehát a fő függvénynek legalább két paraméterrel kell rendelkeznie a definícióban is. E két alapértelmezett paraméteren kívül a `paraméterek[]` paraméterben adhatunk meg továbbiakat. Ahogyan a 4. listán is látható, a szögletes zárójelben (python lista) vesszővel elválasztva felsoroljuk a szükséges paramétereket. A paraméterekről is háromféle információt szükséges a *GIMP* tudtára adni, mégpedig a paraméter típusát, a párbeszédablakban megjelenő szöveget és a paraméter alapértelmezett értékét. A leggyakoribb paraméterek típusa a *6. listán* olvasható. A négyes listán látható példában könnyen nyomon követhető a paraméterek meghatározásának módja.

Végül a modul használatba vételéhez a regisztráció elvégzése után meg kell hívni a `main()` függvényt, mely a *GIMP* beépített modulkezelő függvénye, tehát semmi más tennivalónk nincs a modul működtetéséhez.

Tehát amikor a *GIMP* elindul, végignézi az elérési utakat ahol modulokat és egyéb kiegészítőket tárolhatunk, majd a saját készítésű moduljainkat megtalálva meghívja az abban található `register` függvényt. Ezzel készen is van a modul alaphelyzetbe állítása és a megadott néven, a megfelelő menüpontokon keresztül megtalálhatjuk a *GIMP* menüjében.

A programozási alapismeretek elsajátítása után, térjünk rá első modulunk tárgyalására. A modul célja, hogy az animált *GIF* képek elkészítését megkönnyítsük. A kiegészítő modulok használata nélkül minden képet, melyet a végeredményen látni szeretnénk, meg kellene nyitnunk, majd az átméretezés után mindent kijelölni és a vágólapra helyezni. A másolás után átváltunk az készülő képre, majd ott beillesztjük a másolt képrészletet és új rétegként létrehozzuk a következő képkockát. Ez a folyamat ugyan csak néhány billentyűlenyomást igényel, kis gyakorlással már észre sem vesszük az mozdulatokat, de mégis célszerűnek tartom egy modulba foglalni az ismétlődő műveleteket.

1. táblázat *Változók és metódusok*

Változók	
`bpp`	A réteg színmélysége
`height`	Magasság
`width`	Szélesség
`image`	A kép, melyhez a réteg tartozik
`is_floating_selection`	Nem 0 az értéke, ha a réteg egy kiválasztott rész
`mask`	A rétegmaszk, ha létezik
`mode`	Kirajzolás módja
`name`	A réteg neve
`opacity`	A réteg átlátszósága

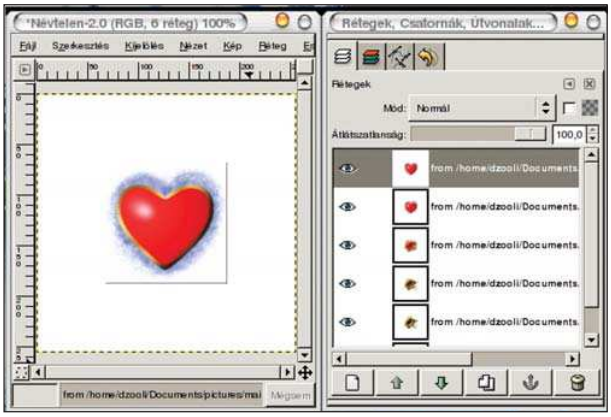
Metódusok	
`add_alpha()`	alfa csatorna létrehozása
`copy()`	másolat a rétegről
`create_mask(típus)`	rétegmaszk létrehozása
`resize(w,h,x,y)`	méret változtatása [w,h] méretre
`scale(w,h,közép)`	átméretezés
`set_offsets(x,y)`	eltolás a [0,0] ponthoz képest
`translate(x,y)`	eltolás az aktuális helyzethez képest

Tekintsük át a 4. listán látható programot. A program elején látható, hogy a forrást a *Python* értelmező fogja végrehajtani és az is, hogy a *GIMP* modulkiegészítőit is használni szeretnénk. A következő fontos dolog, maga a műveleteket végrehajtó függvény. Az `anigif` függvény a *GIMP*-től megkapja az aktuális rajzterületet és a képet, majd a `register` eljárás meghívásakor bejegyzett egyéb változókat.

Első lépésként betölti a `filename` paraméterben kapott képet, mégpedig a *GIMP* beépített függvényének segítségével. A betöltés után a képről megállapítja az aktív réteget (ez maga a kép lesz, amennyiben nem *Photoshop* vagy *GIMP* formátumú képet töltünk be) és azt átméretezi a kívánt méretre. A szükséges méreteket szintén a modul indulásakor megjelenő párbeszédablakban állítjuk be, majd a `neww` és `newh` paramétereken keresztül a fő függvényben használhatjuk. Az átméretezés után szintén a GIMP beépített függvényadatbázisából választjuk ki a megfelelő függvényt, melynek segítségével az aktív réteget teljes egészében kijelöljük és a vágólapra másoljuk.

A következő lépés célszerűen az lenne, hogy a célképen a beillesztendő képnek létrehozunk egy új réteget. A *GIMP* minden működése elérhető a *Python* felületen keresztül is, így létrehozhatunk új réteget is a `gimp.Layer` objektum létrehozásával. Az objektum létrehozásához hét paraméterre van szükség, melyek sorrendben a következők: első a kép, melyhez a réteg tartozik. A második a réteg neve, amely

1. kép Animált GIF-et készítő modul



2. kép A modul használat közben

majd a GIMP réteg-kezelőjében is megjelenik. A harmadik és a negyedik határozza meg a kép méretét, míg az ötödik a réteg típusát. A réteg típusa lehet `RGB_IMAGE`, `RGBA_IMAGE`, `GRAY_IMAGE`, `GRAYA_IMAGE`, `INDEXED_IMAGE` vagy `INDEXEDA_IMAGE`. A hatodik paraméter a réteg átlátszatlanságának mértékét határozzuk meg. 100-as értéknél a réteg egyáltalán nem lesz átlátszó, míg nulla értéknél teljesen átlátszó réteget kapunk. A hetedik paraméter pedig a réteg kirajzolási módját adja meg, amit most `NORMAL_MODE`-ként határozunk meg, tehát a réteggel semmilyen műveletet nem kell végezni a kirajzoláskor.
Az *1. táblázatban* egy rövid összefoglaló található a `gimp.Layer` objektum változóiról és metódusairól.
Az új réteg létrehozása után már csak annyi dolgunk maradt, hogy azt a képhez hozzáadjuk és beállítjuk a megfelelő eltolást és átméretezést, hogy az cél-kép közepére kerüljön a beillesztett kép. A beillesztéshez ismét a *GIMP* egyik menüpontjához tartozó eljárását hívjuk meg, mégpedig a `pdb.gimp_edit_paste` metódust. Itt látható, hogy minden beépülő modul vagy elérhető függvény használható a *PDB (Plug-in DataBase)* adatbázison keresztül, tehát itt is érvényes a bővíthetőség és az újra felhasználhatóság elve. A réteg beillesztése után a fő függvényünkben az utolsó feladat már csak a kijelölés megszüntetése, vagyis a beillesztett tartalom rögzítése.
A modul további sorai tartalmazzák a *GIMP* adatbázisába történő felvételt (a `register` függvény segítségével) és a modul elindítását.
Végül az egyes rétegek beillesztése után még egy egyszerű feladatot kell megoldanunk, hogy az animált *GIF* kép ne tartalmazzon üres képkockát. A két legalsó réteget egyesíteni kell a réteg-kezelőben a *Merge Down* menüpont kiválasztásával. Mindezek után máris menthetjük *GIF* formátumba az elkészült képet. A *GIMP* érdeklődni fog a mentés formátuma iránt, itt válasszuk ki az animált *GIF* lehetőséget és készen is vagyunk első modulunk tesztelésével. Az *1-es képen* látható a modul beállító párbeszédablaka, amelynek felületét a `register` eljárás meghívásakor hozzuk létre.
A *2-es képen* a modul segítségével beillesztett rétegekből álló kép részletei tekinthetők meg.
Amint látható, nem túl bonyolult dolog egy *GIMP* modult elkészíteni, köszönhetően annak, hogy a *Python* nyelv támogatása is belekerült a *GIMP* nyelvezetébe. Felmerülhet azonban az a kérdés kedves olvasóimban, hogy honnan tudhatjuk meg, milyen lehetőségek rejlenek a háttérben, milyen feladatokat végezhetünk el egy-egy rövid programsor segítségével.
Mivel a *GIMP* tartalmaz egy modul- és függvényböngésző párbeszédablakot, így könnyedén utána járhatunk a meglévő modulok képességeinek.
A *GIMP* fő ablakában található az ***Xtns -> Python-FU -> PDB Browser*** menüpont, melynek segítségével minden beépülő modulról vagy függvényről bővebb információkhoz juthatunk. Amint megnyitjuk ezt a párbeszédablakot, a rendelkezésre álló függvények listáját láthatjuk és kedvünkre válogathatunk közöttük. Egy függvényt kiválasztva az ablak jobb oldali részén annak rövid leírását, paramétereit és használatát segítő szöveges leírását olvashatjuk. Az ablakban lehetőségünk van a függvények közötti keresésre is, tehát igazán néhány óra olvasgatással hamarosan otthon érezhetjük magunkat a *GIMP* moduljainak körében.
A másik lehetőség az ismeretek bővítésére szintén a fő eszköztár menüjén keresztül érhető el. Az ***Xtns -> Python-Fu -> Console*** menüpontok kiválasztásával megjelenik egy szövegbevitelre alkalmas párbeszédablak, ahol írhatunk különféle *Python* nyelven értelmezhető parancsokat, melyeknek kimenete az ablak felső részét elfoglaló mezőben azonnal láthatóvá válik. Első lépésben mindig töltsük be a ***gimpfu*** modult, hogy a *GIMP* saját függvényei is elérhetővé váljanak. Ezt az `import gimpfu` utasítás begépelésével érhetjük el, ahogyan a modul írásakor is tettük. Ezután már a Python nyelvben használatos `dir` függvény alkalmazásával minden objektumról megjeleníthetjük a metódusokat és az objektum változóit. Bővebb leíráshoz juthatunk, ha a `import pydoc` utasítással a dokumentációt segítő modult is betöltjük majd a `pydoc.doc(VALAMI)` függvény használatával kérünk segítséget a kérdéses objektumról. A konzol azonban nem csak ismerkedésre alkalmas, hiszen segítségével a modulokat még a tényleges elkészítés előtt részleteiben is ellenőrizhetjük, továbbá kipróbálhatjuk a meglévő függvényeinket is.
Meg kell jegyeznem, hogy amikor a *GIMP*-et kiegészítjük valamilyen modullal, a regisztráció után más modulokban is használhatjuk a saját függvényeinket, tehát megtalálhatjuk a *PDB Browser* listájában is.
A következő részben folytathatjuk az ismerkedést bonyolultabb feladatok megoldásával, addig is bizonyára az érdeklődő olvasók sikerrel veszik a kezdeti nehézségeket.
Addig is mindenkinek kellemes ünnepeket és Boldog Új Évet kíván a szerző.

Fábián Zoltán