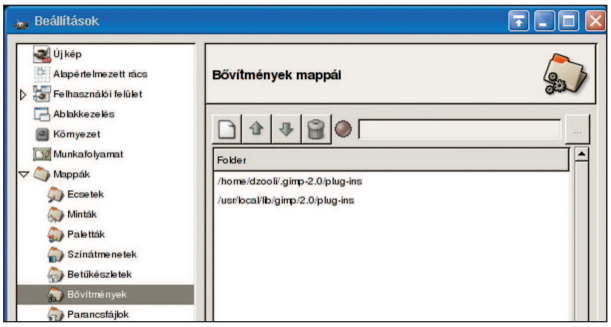
Bővítőmodulok segítséggel

A cikksorozat utolsó részében megvizsgáljuk annak lehetőségét, hogy hogyan írhatjuk át saját igényeinknek megfelelően a már elkészült GIMP bővítményeket.

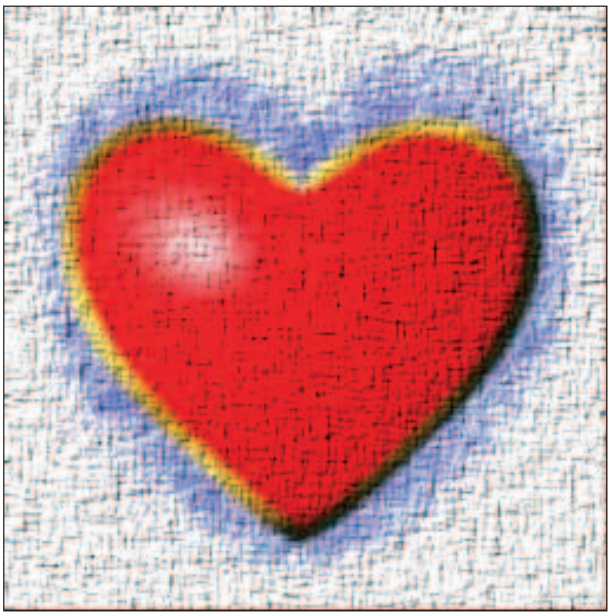
Tekintettel arra, hogy már néhány modul rendelkezésre áll ***Python*** nyelven is, így amennyiben megfelelő kiindulási alapot találunk, apróbb változtatásokkal könnyedén hozhatunk létre újabb hatásokat. A meglévő bővítményeket tároló könyvtár a ***GIMP*** eszköztár ablakában a ***File->Beállítások*** párbeszédablakban tekinthető meg. Válasszuk ki a bal oldali listában a ***Mappák->Bővítmények*** elemeket és nézzük meg a jobb oldali részen megjelenő adatokat. Ez a könyvtár az *1-es képen* látható esetben a ***/usr/local/lib/gimp/2.0/plug-ins***.
A következő lépés, hogy kiválasztjuk a számunkra megfelelő kiindulási alapot képező bővítményt. A ***GIMP***-ben hozzunk létre egy új képet vagy nyissunk meg egy már meglévőt és a ***Python*** menüben válasszunk egy modult. Nézzük meg, hogy melyiknek az eredménye áll legközelebb az elképzeléseinkhez. Az alábbi példában a ***Clothify*** modult egészítjük ki elképzeléseinknek megfelelően mégpedig úgy, hogy a mintázatot színes plazma aláfestéssel tesszük látványossá.
Tekintsük át, milyen változtatásokra van szükség a modul regisztrációs részében. Az első listán látható az eredeti ***Clothify*** modul bejegyzését végző kódrészlet.

```
register(
  "python_fu_clothify",
  "Make the specified layer look like it is
  ➥printed on cloth",
  "Make the specified layer look like it is
  ➥printed on cloth",
  "James Henstridge",
  "James Henstridge",
  "1997-1999",
  "<Image>/Python-Fu/Alchemy/_Clothify",
  "RGB*, GRAY*",
  [
    (PF_INT, "x_blur", "X Blur", 9),
    (PF_INT, "y_blur", "Y Blur", 9),
    (PF_INT, "azimuth", "Azimuth", 135),
    (PF_INT, "elevation", "elevation", 45),
    (PF_INT, "depth", "Depth", 3)
  ],
  [],
  python_clothify)
```



1. ábra A bővítményeket tároló könyvtár

Számunkra tulajdonképpen csak a párbeszédablak felépítéséhez szükséges változók definíciója érdekes, amelyet a (PF_ kezdetű sorokban határozott meg a bővítmény készítője.
A kiegészítésünkhöz még két változóra lesz szükség. Az egyik a plazma hatás örvénylésének mértékét adja majd meg, míg a másik a létrejövő rétegek átlátszatlanságát. Az átlátszatlansággal tudjuk majd befolyásolni, hogy milyen mértékben érvényesül a mintázat és mennyiben marad meg az eredeti hátteret képező kép. Az örvénylés meghatározására szolgáló változó neve legyen `turbulence`, típusa lebegőpontos. Az átlátszatlanság meghatározását egy csúszka segítségével végezzük, melynek értéke nullától százig változhat. A programban majd lebegőpontos értékként használjuk, de a bejegyzést végző kódrészletnél ennek nincs jelentősége. A változónk neve legyen `opacity`.
A második listán látható sorokat írjuk tehát az eredeti kódban a szögletes zárójel bezárása elé, de ne felejtsük el az eredetiben a sor végén vesszővel jelezni a ***Python*** értelmező számára a folytatást.
Látható az eredeti részletben, hogy a menüben hol helyezkedik majd el a bővítmény. Ezt a sort is javítsuk ki valamilyen más névre. Jelen esetben a ***Python-Fu->Alchemy->Plasma Clothify*** nevet választottam. A modul fő eljárásának a nevét is megváltoztattam, így a bejegyzést végző ***register*** eljárásban is az utolsó paraméterben a megváltozott nevet kell megadni. A 2-es listán látható a kódrészlet a leírt változtatások elvégzése után. Az első paraméterben megadott nevet is ki kell cserélnünk valamilyen más azonosítóra,



2. ábra Az eredeti bővítmény

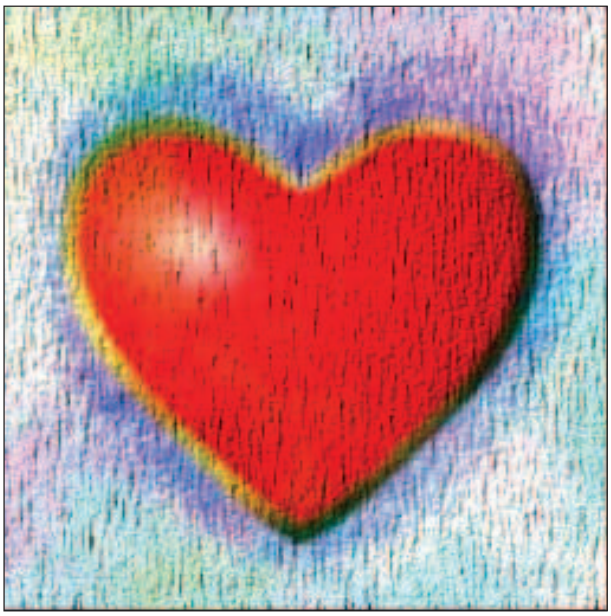
hiszen a *GIMP* ezen név alapján tárolja a saját bővítmény-adatbázisában az új eljárást. A bővítmény neve az új változatban `python_fu_plasmaclothify` lesz.

```
    # eredeti részlet
    (PF_INT, "elevation", "elevation", 45),
    (PF_INT, "depth", "Depth", 3),

    # örvénylő hatás beállítására
    (PF_FLOAT, "turbulence", "Plasma turbulence",
    ➥0.5),
    # rétegek átlátszatlansága (alapértelmezés 20%)
    (PF_SLIDER, "opacity", "Layers opacity", 20,
    ➥(0, 100, 1))
],
[],
python_pclothify)
```

A változók és a párbeszédablak beállítása után következhet az érdemi munka.
Lényegében a változtatás annyiból áll majd, hogy még egy plusz réteget adunk a képhez, melyen a plazma hatást létrehozzuk. A hatások elkészítése után a rétegeket a képre lapítjuk, így érjük el a végleges eredményt. Az eredeti bővítménnyel ellentétben most nem egy új képet hozunk létre hanem a *GIMP* által átadott képen dolgozunk. Az eredeti modulban minden műveletet az `img` változó által tárolt képen hajtunk végre. Első dolgunk tehát az legyen, hogy ahol hivatkozást találunk erre a változóra, azokon a helyeken javítsuk ki `timg` névre a megadott változót. Hogy miért pont `timg`-re? Ha jobban megvizsgáljuk a bővítmény fő eljárását, észre fogjuk venni, hogy az első paraméter neve ***timg***. Ez az a változó, melyben a GIMP átadja a modul részére a feldolgozásra szánt képet. Tehát a kívánt eredmény elérése érdekében nekünk a program által adott képen kell dolgoznunk, vagyis azon, melyet a `timg` változóban kapunk.

Ezután az alapvető változtatás után már tulajdonképpen nincsen sok dolgunk. A plazma hatást a *GIMP* bővítményei között találjuk, tehát az alábbi lépésekben összefoglalható minden szükséges változtatás. Elsősorban, miután az eredeti változat szerint elkészült a két ruhaszerű mintázatot megjelenítő réteg, következhet a harmadik réteg létrehozása. A harmadik rétegen jelenik meg a plazma hatás. Hozzunk létre egy új réteget a képen a `gimp.Layer()` objektum elkészítésével. Itt már a létrehozásnál adjuk meg a felhasználó által beállított átlátszatlanság értéket az `opacity` változó segítségével. Mivel a `gimp.Layer()` objektum létrehozásakor a hatodik paraméter az átlátszatlanság, így nincsen nehéz dolgunk. Az új réteg megjelenítési módja legyen `HARDLIGHT_MODE`, hogy jobban láthatóvá váljon a plazma mintázat a modul futtatása után. A példában használt réteget tároló változó a `layer_alma` nevet kapta. A réteg létrehozása után következhet a plazma hatás elkészítése. A *GIMP*-ben minden bővítményt vagy eljárást elérhetünk más bővítményekből is, tehát a nyugodtan meghívhatjuk a `pdb.plug_in_plasma()` eljárást. Az eljárás négy paramétert vár. Az első legyen a kép, melyhez az új réteg tartozik. Ez jelen esetben megegyezik a *GIMP* által átadott `timg` képpel. A második paraméter az a réteg, melyen a plazma hatás megjelenik. Példánkban ez a `layer_alma` réteg. A harmadik paraméter a véletlenszámok létrehozásához használandó alapérték. A bővítmény elején olvassuk be a ***Python*** rutinok között található `time` gyűjteményt, melyben megtalálható a `time()` függvény. Az éppen aktuális idő elegendő véletlenszerűséget biztosít a plazma hatás elkészítéséhez, tehát nyugodt szívvel használhatjuk az eljárás harmadik paramétereként. A negyedik szükséges paraméter határozza meg a hatás szabálytalanságát. Ez utóbbi értéket korábban örvénylésnek neveztük és a modul beállítására szolgáló ablakban a felhasználó állítja be.
Az új réteget természetesen az eredeti képhez is hozzá kell adni, erre szolgál a `timg.add_layer()` függvény.
Az utolsó lépésben már csak annyi a teendő, hogy a képen lévő rétegeket a `timg.flatten()` eljárás meghívásával egy réteggé „lapítjuk”. Végeredményben ez a pár lépés elegendő lenne a kívánt eredmény eléréséhez, azonban néhány apróbb módosításra még szükségünk van.
Elsősorban arról van szó, hogy az eredeti modulban a hozzáadott rétegek egyáltalán nem átlátszóak. Ez számunkra nem túl sikeres választás, hiszen ebben az esetben a háttérként szolgáló kiindulási képet már az első hozzáadott réteg teljesen el fogja takarni. A megoldást jelentő változtatás az első réteg létrehozására használt `gimp.Layer()` hívás jelenti. Ha megvizsgáljuk ennek az eljárásnak a paramétereit látható, hogy a hatodik paraméter – az átlátszatlanság mértéke –, állandó 100-as értéket kap. Ahogyan az általunk létrehozott rétegnél megoldottuk, itt is ki kell javítani ezt az állandót a felhasználó által beállított értékre. Cseréljük ki tehát a megadott értéket az `opacity` változóra.
A második rétegnél szintén hasonló problémába ütközhetünk, azonban nem találunk az első réteg létrehozásához hasonló hívást. Tulajdonképpen a második réteg az első másolataként jön létre, amit láthatunk a `layer_two = layer_one.copy()` sorban. A második rétegnél különféle változókat is beállítunk, mint például a `layer_two.mode = MULTIPLY_MODE` sorban. Itt állíthatjuk be az átlátszatlanságot



3. ábra Az új modul eredménye

is, arra azonban vigyázni kell, hogy a rétegnek lebegőpontos számot kell megadnunk. A `layer_two.opacity = float(opacity / 100)` sorban végezzük el ezt a beállítást. Utolsó lépés, hogy az első réteg megjelenítési módját (hetedik paraméter a `gimp.Layer()` hívásnál) szintén `MULTIPLY_MODE`-ként határozzuk meg, így jobban kiemeljük a végeredményben a réteg hatását.

Mivel csak a harmadik réteg elkészítése után van szükségünk a rétegek összevonására, az eredeti kódban szereplő `timg.flatten()` hívásokat megjegyzésbe tettem. Természetesen ki is lehet törölni ezeket a sorokat, de az érthetőség és a változtatások követése szempontjából érdemes a megjegyzést használni. Így később is tudjuk majd, milyen változtatásokat végeztünk a programban és nem veszítjük szem elől a bővítmény lényegét alkotó sorokat.

A harmadik listán látható a teljes bővítmény listája a korábban már tárgyalt bejegyzést elvégző `register` hívás nélkül. A második képen az eredeti modul által előállított képet tekinthetjük meg. A hármas képen az új változat által készített képet vehetjük szemügyre.

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

import time
import math
from gimpfu import *

def python_pclothify(timg, tdrawable, bx=9, by=9,
  azimuth=135, elevation=45, depth=3,
  turbulence=0.5, opacity=20):
     bx = 9 ; by = 9 ; azimuth = 135
     elevation = 45 ; depth = 3
     width = tdrawable.width
     height = tdrawable.height

# eredetileg új képen dolgozott a bővítmény.
# nekünk mindent az aktuális képen kell elvégezni
# img = gimp.Image(width, height, RGB)

  # atlatszóság beállítása a megadott
  # értékre és szorzás mód
     layer_one = gimp.Layer(timg, "X Dots", width,
      height, RGB_IMAGE,
      opacity, MULTIPLY_MODE)
     timg.disable_undo()
     pdb.gimp_edit_fill(layer_one, BACKGROUND_FILL)
     timg.add_layer(layer_one, 0)
     pdb.plug_in_noisify(timg, layer_one,
        0, 0.7, 0.7, 0.7, 0.7)
     layer_two = layer_one.copy()
     layer_two.mode = MULTIPLY_MODE

     # beállítjuk az átlátszóságot
     layer_two.opacity = float(opacity / 100)

     layer_two.name = "Y Dots"
     timg.add_layer(layer_two, 1)
     pdb.plug_in_gauss_rle(timg, layer_one, bx,
     ➥1, 0)
     pdb.plug_in_gauss_rle(timg, layer_two, by,
     ➥0, 1)
#     erre nincs szükség
#     img.flatten()
     bump_layer = timg.active_layer
     pdb.plug_in_c_astretch(timg, bump_layer)
     pdb.plug_in_noisify(timg, bump_layer,
        0, 0.2, 0.2, 0.2, 0.2)
     pdb.plug_in_bump_map(timg, tdrawable,
        bump_layer, azimuth,
        elevation, depth, 0, 0,
         0, 0, TRUE, FALSE, 0)

# új réteg és a plazma bővitmény használata
     layer_alma = gimp.Layer(timg, "Plasma",
        width, height, RGB_IMAGE,
        opacity, HARDLIGHT_MODE)
     pdb.plug_in_plasma(timg, layer_alma,
        int(time.time()), turbulence)
# réteget hozzáadjuk a képhez
     timg.add_layer(layer_alma, 1)
  # kép végső lapítása
     timg.flatten()

# az új képet nem használtuk,
# így törölni sem kell
#     gimp.delete(img)
```

Gyakorlatilag ez utóbbi tíz hónap alatt megtárgyaltunk minden olyan fontosabb dolgot, amire a *GIMP*-ben lehetőségünk adódik, így a sorozatnak nincsen hivatalos folytatása. Természetesen otthon vagy munkahelyén mindenki tovább folytathatja az ismeretek alkalmazását és bővítését, ebben a továbbiakban elektronikus levelezés útján a szerző is szívesen nyújt segítséget.

Fábián Zoltán