Programmierung eines GIMP-Plugin

Was ist GIMP ?

- GNU Image Manipulation Program
- Bildbearbeitungssoftware
- Bildkonvertierer
- Open Source
- Erweiterbar durch Plugins

Mögliche Programmiersprachen für Plugin-Entwicklung

- Scheme (Script-Fu)
- C
- Python
- Perl

Plugin Registrierung

- PDB (Procedural Database) speichert alle bekannten Plugins
- GIMP sucht beim Start nach Plugins und aktiviert diese
- Plugins können sich in der PDB registrieren (genauer: können Funktionen des Plugin registrieren)
- Plugins können andere Plugins aufrufen

Plugin-Schnittstelle

- init - wird aufgerufen wenn GIMP startet
- quit – wird aufgerufen wenn GIMP endet
- query – wird aufgerufen, damit sich das Plugin registrieren kann
- run – wird aufgerufen, um das Plugin auszuführen

Plugin Parameter

```
static void run (const gchar      *name,
                 gint              nparams,
                 const GimpParam  *param,
                 gint             *nreturn_vals,
                 GimpParam       **return_vals);
```

- name - bezeichnet Funktion, für die das Plugin aufgerufen wurde (Plugin kann mehrere Funktionen implementieren)
- nparams, param – Eingabeparameter
- nreturn_vals, return_vals – Rückgabewerte
- Typ und Anzahl der Paramter/Rückgabewerte werden bei der Registrierung der Funktion des Plugin angegeben

Plugin Parameter (2)

```
static GimpParamDef args[] = {
  {
    GIMP_PDB_INT32,
    "run-mode",
    "Run mode"
  },
  {
    GIMP_PDB_IMAGE,
    "image",
    "Input image"
  },
  {
    GIMP_PDB_DRAWABLE,
    "drawable",
    "Input drawable"
  }
};
```

- Je nach Art der registrierten Funktion (Load/Save/Tool/Image) sind bestimmte Parameter mindestens vorgeschrieben
- Im Beispiel 3 Paramter:
  - Interaktive oder nicht (run-mode)
  - Aktives Bild (image)
  - Aktive Zeichenfläche in diesem Bild (drawable)

Begriffe

- Bild (**image**) besteht aus verschiedenen Ebenen (layer), Masken und Auswahlmarkierungen
- Ebenen, Masken, etc. sind letztlich nur Rechtecke mit Pixeln und werden als Zeichenfläche (**drawable**) zusammengefaßt
- Zeichenfläche ist in Kacheln (**tile**) im Speicher organisiert

Beispiel 1: Script-Fu

- Script-Fu wird standardmäßig von GIMP unterstützt
- Plugin ist Scheme-Code
- Skript wird gespeichert als:

  `~/.gimp-2.3/scripts/test.scm`

Beispiel 1: Script-Fu

```
(define (test image drawable radius fill color caption)
; place your code here
)

(script-fu-register "test"            ; function name
            _"<Image>/Script-Fu/Test" ; menu label
            "just a test"         ; description
            "author"          ; name of author
            "copyright notice"        ; copyright information
            "04/04/2007"      ; creation date
            "RGB*"            ; accepted image formats
            SF-IMAGE "Image" 0
            SF-DRAWABLE "Drawable" 0
            SF-ADJUSTMENT _"Radius" '(8 0 100 .2 1 1 1)
            SF-TOGGLE _"Fill" TRUE
            SF-COLOR _"Color" '(0 128 255)
            SF-STRING _"Caption" "some text"
            )
```

Beispiel 2: Python

- Installation von GIMP-Python erforderlich
- Plugin ist Python-Code
- Skript wird gespeichert als:

  `~/.gimp-2.3/plug-ins/simple.py`

Beispiel 2: Python

```
#!/usr/bin/python

from gimpfu import *

def test(img, drawable, feather_in, opacity_in):
   gimp.message("Test bestanden")
   width = drawable.width
   height = drawable.height
   pdb.gimp_rect_select(img, 0, 0, width, height, CHANNEL_OP_REPLACE, TRUE, feather_in)
   pdb.gimp_edit_bucket_fill(drawable, FG_BUCKET_FILL, MULTIPLY_MODE, opacity_in, 0, 0,
0, 0)
   pdb.gimp_selection_none(img)

register(
   "python_fu_test",
   "Zeigt einen Dialog an",
   "Hilfe: Zeigt einen Dialog",
   "author",
   "copyright",
   "04/04/2007",
   "<Image>/Python/Test",
   "*",
   [
    (PF_INT, "feather_in", "Weicher Rand", 100),
    (PF_INT, "opacity_in", "Deckkraft", 50)
   ],
   [],
   test)

main()
```

Beispiel 3: C

- Wird standardmäßig von GIMP unterstützt
- Plugin wird als normales C-Programm geschrieben, z.B. `test.c`
- Wird übersetzt und installiert mit:

```
gimptool-2.0 --install test.c
```

Beispiel 3: C

```
#include <libgimp/gimp.h>

/* Function prototypes */
static void query (void);
static void run (const gchar     *name,
                 gint             n_params,
                 const GimpParam *param,
                 gint            *n_return_vals,
                 GimpParam      **return_vals);

...

/* Setting PLUG_IN_INFO */
GimpPlugInInfo PLUG_IN_INFO =
{
    NULL,    /* init_proc */
    NULL,    /* quit_proc */
    query,   /* query_proc */
    run      /* run_proc */
};

MAIN()
```

Beispiel 3: C

```
static void query()
{
    static GimpParamDef params[] =
    {
    { GIMP_PDB_INT32, "run_mode", "Interactive, non-interactive" },
    { GIMP_PDB_IMAGE, "image", "(unused)" },
        { GIMP_PDB_DRAWABLE, "drawable", "Drawable to draw on" },
        { GIMP_PDB_COLOR, "fgcolor",  "Color to draw with"}
    };

    gimp_install_procedure(
    "my name",                  /* name */
    "blurb ??",                 /* blurb */
    "some helptext here",       /* help */
    "that's me",                /* author */
    "and i made it",            /* copyright */
    "what day is it ?",         /* date */
    "<Image>/Just/Simple",      /* menu_label */
    "RGB",                      /* image_types */
    GIMP_PLUGIN,                /* type */
    G_N_ELEMENTS(params),       /* n_params */
    0,                          /* n_return_vals */
    params,                     /* params */
    0                           /* return_vals */
    );
}
```

Beispiel 3: C

```
static void run (
    const gchar     *name,
    gint             n_params,
    const GimpParam *param,
    gint            *n_return_vals,
    GimpParam      **return_vals)
{
    gint x, y, w, h, n, idx;
    GimpPixelRgn srcrgn, dstrgn;
    guchar *src, *dst;

    /*
     * provide space for return value (note that this may NOT
     * be allocated on the stack(
     */
    static GimpParam values[2];

    /* the ID of the drawable */
    gint32 drawable_ID;

    /* will hold a pointer to the drawable later on,
     * the drawable is passed as a parameter in param */
    GimpDrawable *drawable;

...
```

Beispiel 3: C

```
    /* set n_return_vals to the number of return values, the status
     * code is not counted */
    *n_return_vals = 1;
    /* set the pointer to the return values */
    *return_vals = values;

    /* return value #0 is the status code and must always be returned */
    values[0].type = GIMP_PDB_STATUS;
/*  values[0].data.d_status = GIMP_PDB_EXECUTION_ERROR; */
    values[0].data.d_status = GIMP_PDB_SUCCESS;


    /* display a greeting */
    gimp_message("Hello world!");

    /* obtain the drawable of the image */
    drawable_ID = param[2].data.d_drawable;
    drawable = gimp_drawable_get(drawable_ID);

    /* obtain width and height */
    w = drawable->width;
    h = drawable->height;

    /* allocate memory for source and destination arrays */
    n = w * h;
    dst = g_malloc(n * 4);
    src = g_malloc(n * 4);


...
```

Beispiel 3: C

```
    /* setup pixel region for source */
    gimp_pixel_rgn_init(&srcrgn, drawable, 0, 0, w, h, 0, 0);
    gimp_pixel_rgn_get_rect(&srcrgn, src, 0, 0, w, h);

    /* fill the pixels with some data */
    idx = 0;
    for (y = 0; y < h; y++)
    {
        for (x = 0; x < w; x++)
        {
            dst[idx++] = x;
            dst[idx++] = y;
            dst[idx++] = src[idx];
        }
    }

    /* setup pixel region for destination */
    gimp_pixel_rgn_init(&dstrgn, drawable, 0, 0, w, h, 1, 1);
    gimp_pixel_rgn_set_rect(&dstrgn, dst, 0, 0, w, h);

...
```

Beispiel 3: C

```
    /* commit changes to the drawable */
    gimp_drawable_flush(drawable);
    gimp_drawable_merge_shadow(drawable->drawable_id, 1);
    gimp_drawable_update(drawable->drawable_id, 0, 0, w, h);

    /* make sure that changes become visible */
    gimp_displays_flush();

    /* release the drawable */
    gimp_drawable_detach(drawable);

    /* release the memory that was allocated with g_malloc */
    g_free(src);
    g_free(dst);
}
```

Referenzen

- http://developer.gimp.org/api/2.0/index.html
- http://docs.gimp.org/en/gimp-scripting.html
- http://www.jamesh.id.au/software/pygimp/
- http://www.gimp.org/docs/python/index.html
- http://registry.gimp.org/
- http://www.pro-linux.de/t_multimedia/gimp-rote-augen.html
- http://www.linux-user.de/ausgabe/2005/02/033-gimp-python/index.html