

Class 11: GIMP Internals (3): The Procedural Database

Held: Thursday, 18 April 2013

Summary: We consider how to write functions that are more naturally callable by other PDB functions.

Related Pages:

- EBoard.

Overview:

- Some procedures are documented at: <http://developer.gimp.org/api/2.0/libgimp/libgimp-gimp.html>
- Assignment: Write an “interesting” function or set of functions.

Overview:

- Writing functions, revisited.
- Return values.
- Dealing with arrays.
- Calling user-defined PDB functions.

Extending the PDB with New Procedures

- In extending the GIMP, we’ve been writing procedures that get called when they get selected from a menu item.
- But our research focus is really scripting the GIMP.
- And, as we’ve seen, many of the menu-driven GIMP plugins take parameters that aren’t naturally for scripting situations.
- So, we’re going to consider how to write procedures that aren’t directly accessed via menu items.
- The changes are simple:
 - Just list the parameter types you find useful.
 - Return the values you want.
 - Don’t include a menu item.

Returning Values

- So, how do we return other values?
- It’s similar to returning the status. We just make the array bigger.
- We need to set the type and the corresponding data value.
- Some types: `GIMP_PDB_INT32`, `GIMP_PDB_FLOAT`, `GIMP_PDB_STRING`
- The corresponding data values `data.d_int32`, `data.d_float`, `data.d_string`
- So, to make 3.14 the first return value, we’d say

```
results[1].type = GIMP_PDB_FLOAT;  
results[1].data.d_float = 3.14;
```

Arrays

- Forthcoming

Calling PDB Functions

- We know how to call your functions in Scheme. But how do we call them in C?
- Unfortunately, the compiler is not smart enough to know about potential functions.
- So there's a really weird calling convention, using `gimp_run_procedure` or `gimp_run_procedure2`.
- Signatures:

```
GimpParam* gimp_run_procedure (const gchar *name, gint *n_return_vals, ...);  
GimpParam* gimp_run_procedure2 (const gchar *name, gint *n_return_vals,  
                                gint n_params,  
                                const GimpParam *params);
```

- So, suppose we have a function, `square`, that takes one integer as a parameter. We can call it with

```
GimpParam *returned;  
int nreturned;  
returned = gimp_run_procedure ("square", &nreturned; GIMP_PDB_INT32, 5, GIMP_PDB_END);
```

- Or

```
GimpParam newparams[] = { { GIMP_PDB_INT32, 5 } };  
returned = gimp_run_procedure ("square", &nreturned, 1, newparam);
```

- How do you get info on the parameters to that procedure? `gimp_procedural_db_proc_info`
- How do you find out what procedures are available? `gimp_procedural_db_query`

Copyright © 2012 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.