

Writing Script-Fu Procedures

Summary: Now that you've written basic commands for the GIMP using Script-Fu, you can put them together into your own procedures.

Contents:

- Introduction
- A Sample Procedure
- Parameterizing
- Telling DrScheme About Procedures

Introduction

As you may recall from your first experiences with Scheme, we started our Scheme programming by writing simple sequences of commands in the DrScheme interactions pane and then progressed to grouping those commands into procedures. Grouping commands simplified our work because we could now type a single command rather than a series of commands. It also simplified our work because we could *parameterize* the sequence of commands, making them work with a variety of values.

It is, of course, natural to do something similar within the GIMP. In particular, we will build general procedures to draw a variety of things. (In future readings, we will build general procedures that also modify existing images.)

As you might expect, instead of typing instructions one-at-a-time within the Script-Fu console, we can

1. group the commands within a Scheme procedure;
2. put the procedure definition in a file;
3. load the file in the Script-Fu console; and
4. execute the procedure.

Unfortunately, the GIMP includes no editor that corresponds to the definitions window in DrScheme. Hence, we will write our GIMP/Script-Fu programs in DrScheme, but run them in the GIMP. (*Warning:* If you try to run them in DrScheme, it will complain.)

A Sample Procedure

For example, here is a simple procedure I wrote that draws the letter F and then puts a "No" sign around it.

```
;;; Procedure:
;;; no-failure
;;; Parameters:
;;; none
;;; Purpose:
;;; Creates a new image with the universal sign for "No Fs"
```

```

;;; Produces:
;;; Nothing. You call this only for the side effect of
;;; drawing a new image.
;;; Preconditions:
;;; Must be called from within Script-Fu.
;;; Postconditions:
;;; A new image appears on the screen.
(define no-failure
  (lambda ()
    ; Draw everything on white
    (set-bgcolor WHITE)
    (let ((image (create-image 256 256)))
      ; Draw the silly letter F in blue
      (set-fgcolor BLUE)
      (set-brush "Circle (11)")
      (line image 96 40 176 40) ; Top stroke
      (line image 96 40 96 216) ; Down stroke
      (line image 96 112 148 112) ; Middle stroke
      ; Draw a nice red circle.
      (set-fgcolor RED)
      (set-brush "Circle (07)")
      (select-ellipse image REPLACE 8 8 240 240)
      (stroke image)
      (select-nothing image)
      ; Now we're ready to draw the slash. Basic math tells us that
      ; the endpoints are offset by radius/(sqrt 2) from the center
      ; of the circle.
      (let* ((center 128)
             (radius 120)
             (offset (/ radius (sqrt 2))))
        (line image
              (- center offset) (- center offset)
              (+ center offset) (+ center offset)))
      ; And display the image
      (show-image image))))

```

We can run it by loading the file and then running the procedure in the Script-Fu console.

```

(load "/home/rebelsky/Web/Courses/CS151/2007S/Examples/no-failure.scm")
(no-failure)

```

Parameterizing

One disadvantage of `no-failure` is that it always draws an image of the same size. What if we want a bigger or smaller image? One possibility is to make the image size a parameter to `no-failure` and then make the various numbers depend on the size. Since our image is naturally square, the width and height should probably be the same. Here's what I've come up with:

```

;;; Procedure:
;;; new-no-failure
;;; Parameters:
;;; side, an integer
;;; color, an RGB list
;;; Purpose:

```

```

;;; Creates a new image with the universal sign for "No Fs".
;;; The image has width side and height side.
;;; The F in the image has the specified color.
;;; Produces:
;;; Nothing. You call this only for the side effect of
;;; drawing a new image.
;;; Preconditions:
;;; Must be called from within Script-Fu.
;;; Postconditions:
;;; A new image appears on the screen.
(define new-no-failure
  (lambda (side color)
    (let ((image (create-image side side))
          (unit (/ side 32)))
      ; Draw everything on white
      (set-bgcolor WHITE)
      ; Draw the silly letter F in the specified color
      (set-fgcolor color)
      (gimp-brushes-set-brush "Circle (11)")
      (line image (* unit 12) (* unit 5)
             (* unit 20) (* unit 5))
      (line image (* unit 12) (* unit 5)
             (* unit 12) (* unit 27))
      (line image (* unit 12) (* unit 14)
             (* unit 18) (* unit 14))
      ; Draw the nice red circle.
      (set-fgcolor RED)
      (set-brush "Circle (07)")
      (select-ellipse image REPLACE
                     unit unit
                     (- side unit unit) (- side unit unit))
      (stroke image)
      (select-nothing image)
      ; Now we're ready to draw the slash. Basic math tells us that
      ; the endpoints are offset by radius/(sqrt 2) from the center
      ; of the circle.
      ; direction.
      (let* ((center (/ side 2))
             (radius (- center unit))
             (offset (/ radius (sqrt 2))))
        (line image
              (- center offset) (- center offset)
              (+ center offset) (+ center offset)))
      ; Display the image
      (show-image image)
      ; And return the image
      image)))

```

Telling DrScheme About Procedures

With the Script-Fu that you know up to this point, all the cool Script-Fu commands you write are only available to people who are able to use the Script-Fu console. It would also be nice to add Script-Fu commands to the GIMP's various menus. Fortunately, you only need to follow a few simple steps.

1. Write Scheme code that tells the GIMP to add a procedure to the menu. Here, you need to tell the GIMP what parameters your procedure takes so that it can ask the user for those parameters.

2. Put the Scheme code in a place the GIMP can find it. Typically, you'll put your Scheme code in `/home/username/.gimp-2.2/scripts`. (Yes, the period in the `.gimp-2.2` is important.) You'll only have to do this once per script. My experience suggests that you should end the file name with `.scm` rather than `.ss`.

3. Tell the GIMP that you've added the script by selecting **Refresh** from the **Script-Fu** submenu of the **Xtns** menu. (You should only have to do this when you add or modify the script in the middle of a session. In the future, the GIMP should load your script automatically.)

Steps 2 and 3 are straightforward, so let's consider the first step in more detail. You add a procedure with the `script-fu-register` procedure. That procedure takes a large number of parameters.

- *name*, the procedure name, given as a string;
- *menu-item*, the full path to the menu item, given as a string; this should start with `<Image>` for the image menu or `<Toolbox>/Xtns/` for the toolbox extensions menu;
- *description*, a short description, given as a string;
- *author*, given as a string;
- *copyright*, given as a string;
- *date*, the date last modified, given as a string;
- *image-type*, which restricts the kind of image to be used; (I recommend that you start with "RGB" but try other options later); and
- descriptions of desired parameters for your procedure.

For each parameter for your procedure, you'll need three additional parameters to `script-fu-register`:

- *param-type*, the parameter type (can be SF-VALUE, SF-COLOR, SF-TOGGLE, SF-IMAGE, SF-DRAWABLE, or more);
- *prompt-text*, a message to print in the dialog box that will appear when someone selects your menu command;
- *default-value*, the default value.

For SF-IMAGE and SF-DRAWABLE, the value should be 0; the GIMP will fill them in with the current image and layer. For SF-VALUE, the default value should be in quotation marks, even when it's a number. For SF-COLOR, the color should be a list of three integers (as you explored in the previous lab).

For example, here's what I might use for my `no-failure` procedure.

```
(script-fu-register
  "new-no-failure"
  "<Toolbox>/Xtns/Script-Fu/Sample/No Failure"
  "Draws a \"No Failure\" logo"
  "Samuel A. Rebelsky"
  "Copyright (c) 2001-2006 Samuel A. Rebelsky. All Rights Reserved"
  "Monday, 23 October 2006"
  "RGB"
  SF-VALUE "Side Length" "256"
  SF-COLOR "Color" (list 0 0 255))
```

I've put this complete example in the file `no-failure-menu.scm`.

Note that in order for it to work, your `gimp.scm` file must now also be in the GIMP scripts directory. We'll consider how to get it there in the lab.

If you want your scripts in the menu associated with each image, rather than with the toolbox, you use a different path (the second parameter).

```
"<Image>/Script-Fu/Sample/Forbidden Sign"
```

Copyright © 2007 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.