Programming Languages (CS302 2007S)

Scripting the GIMP with Script-Fu

Summary: GIMP distinguishes itself from Photoshop, in part, by making itself *scriptable* in a Scheme-like language called *Script-Fu*. The availability of Script-Fu means that you can make GIMP by entering textual commands and by writing your own programs. In this reading, we consider the basics of scripting the GIMP.

Contents:

Introduction: Scripting

In the early days of interactive computing (and even in the early days of personal computing), almost all interaction with the computer was textual. That is, the computer would present a *prompt*, you’d type commands in response to that prompt, and the computer would respond to those commands.

In the late 1970’s and early 1980’s, a new mode of interaction was developed; one that involved different *windows* for different activities, *icons* you could click on to initiate activities (e.g., start new programs, print the current document), *menus* from which you could select other activities, *dialog boxes* which prompted you for information, and a *pointing device* you could use to select icons and menu items.

This new *graphical user interaction* paradigm made computers usable to a much wide group of people, and it remains the primary paradigm with which most people now interact with computers.

However, there are still times in which it is more useful to enter commands by typing them than by clicking on things. For example, if you have to draw a series of dots in a predictable series of sizes, it may be easier to write something like

```
(set-brush-size 5)
(click-at 10 20)
(set-brush-size 10)
(click-at 10 30)
(set-brush-size 15)
(click-at 10 40)
```

than to bring up the brush tool, open the size dialog, enter 5, close the dialog, click at the point (10,20), open the size dialog again, enter 10, and so on and so forth. It's also probably much more accurate. (Think of how hard it is to click at a precise point in the window.)

Because there are certain advantages to being able to enter commands textually, many applications now include *scripting languages* in which you can enter these commands. Most scripting languages also let you write small (or large) programs. Microsoft's Visual Basic for Applications (VBA) is one of the most popular scripting languages, but there are many others.

Script-Fu: GIMP's First Scripting Language

Because an important philosophical goal of open-source software is to empower the user to make changes, most open-source applications provide a scripting language. (How's that for a bold claim?) When it was first released, GIMP supported one scripting language, Script-Fu, a variant of Scheme. (Although GIMP now supports other scripting languages, we will discuss only scripting in Script-Fu.)

In general, you interact with Script-Fu much like you interact with DrScheme in the interactions window. You type expressions, Script-Fu evaluates those expressions, and then Script-Fu shows the results. However, in addition to showing some results in the Script-Fu interactions window, Script-Fu may also update the state of GIMP (drawing things in the current window, changing the current tool, etc.).

It is best to begin to learn about scripting by entering commands interactively (just as we entered Scheme commands interactively). In this reading and the corresponding lab, we will consider some of the basic commands you might enter. In the next reading, we'll also consider how (and why) you write your own procedures.

Practicum: Opening the Script-Fu Console

To interact with Script-Fu, you'll need to open the *Script-Fu Console* (GIMP's equivalent of the DrScheme interaction pane). The process for opening the console is fairly straightforward:

- Start GIMP by typing `gimp` in a terminal window. (Alternately, click the GIMP icon, if you have one.)
- Select Console from the Script-Fu menu available from under the Xtns menu in the primary GIMP palette.

You should see a window appear with some text at the top and an entry box at the bottom. This is the console for interacting with Script-Fu, which is based on a *variant* of Scheme called SIOD.

Detour: Simplifying Script-Fu

While Script-Fu has many great advantages to the experienced programmer, it is a bit complex for the beginning programmer. Hence, for this class, we will use a library of routines that simplify Script-Fu and GIMP.

When working in the MathLAN, you should load this library with

```
(load "/home/rebelsky/Web/Courses/CS302/2007S/HOG/gimp.scm")
```

If you're working on your home computer, you'll need to make your own copy of the library, which I'd recommend that you save on the desktop. The command you type then depends on your computing platform.

- Macintosh OS X users will use
  `(load "/Users/`*`username`*`/Desktop/gimp.scm")`
- Linux users will use something like
  `(load "/home/`*`username`*`/Desktop/gimp.scm")`
- Microsoft Windows XP users will use something like
  `(load "/Documents and Settings/`*`username`*`/Desktop/gimp.scm")`
  As strange as it seems to Windows users, Script-Fu wants forward slashes rather than backslashes and does not want the `C:`.

Creating and Loading Images

Behind the scenes, the GIMP assigns a numeric identifier to each image and permits an arbitrary number of layers for each image, each with its own identifier. Our experience is that the numbers are somewhat confusing, so we do our initial work with single-layer images whose numeric identifiers are encapsulated.

You can create a new image with `(create-image` *`width height`*`)`. This procedure returns an encapsulated image value.

You can load an existing image with `(load-image` *`filename`*`)`. This procedure returns an encapsulated image value.

By default, when you create or load an image, the image is not shown. You can show the image with `(show-image` *`image`*`)`.

For example, to create an image 200 pixels wide and 100 pixels high, we might write

```
(define img (create-image 200 100))
(show-image img)
```

Colors

You can set the foreground color with `(set-fgcolor` *`color`*`)` and the background color with `(set-bgcolor` *`color`*`)`. For this purpose, color is a list of three integers in the range 0-255, representing the red, green, and blue components. For example, you might write the following to get a nice shade of blue for the foreground color.

```
(set-fgcolor (list 0 127 255))
```

We have also defined a wide range of color constants. You can find the whole list in the variable `colors`. We can get the same color as above with

```
(set-fgcolor SLATE_BLUE)
```

Selecting Brushes

You can set the current brush with the `set-brush` command. You can find a list of all possible brushes with `(list-brushes)`.

Drawing by Selecting

One of the common ways to draw in GIMP is by selecting an area and then either filling or stroking the selected areas.

You can select areas with any of the following procedures:

- `(select-rectangle img operation left top width height)`
- `(select-ellipse img operation left top width height)`

The operation is one of `ADD`, `SUBTRACT`, `REPLACE`, and `INTERSECT`.

You can also select the whole image with `(select-all img)` and clear the selection with `(select-none img)`.

Once you have selected an area, you can fill it with `(fill-fgcolor` *`img`*`)` or `(fill-bgcolor` *`img`*`)`. You can also stroke it with `(stroke` *`img`*`)`, which uses the current brush.

Other Drawing Operations

The procedures above provide most of the basic drawing operations available in our extended Script-Fu. One other important procedure is `(line` *`img x1 y1 x2 y2`*`)`, which draws a line from the point (x1,y1) to the point (x2,y2).

You will learn about other drawing operations in the lab.