

Algorithmic Art

Summary: We explore two techniques for algorithmically generating interesting images: Random drawing and Color grids.

Contents:

- Exercises
 - Exercise 0: Preparation
 - Exercise 1: Random Art
 - Exercise 2: Splats
 - Exercise 3: New Kinds of Random Drawings
 - Exercise 4: Repeated Blobs
 - Exercise 5: Color Grids
 - Exercise 6: Specifying Components
 - Exercise 7: Anonymous Components
 - Exercise 8: Making Grids from the Console

Exercises

Exercise 0: Preparation

For the second half of this laboratory, you will need to have a copy of `grid.scn` in your GIMP scripts folder.

a. Open a new terminal window.

b. Type

```
ln -s /home/rebelsky/Web/Courses/CS151/2006F/Examples/grid.scn .gimp-2.2/scripts/grid.scn
```

c. Start GIMP and verify that the Xtns/Script-Fu menu contains a Glimmer submenu.

d. Make you own copy of `random-art.scn`.

e. Start DrScheme and open your copy of `random-art.scn`.

f. In the GIMP, open a Script-Fu console and load `random-art.scn`.

Exercise 1: Random Art

a. Read through `random-art.scn` to make sure that you know what all of the procedures do.

- b. Create an image (say 200 wide by 100 high) and name it `image`.
- c. Enter the following sequence of commands a few times to generate a few “random” lines.

```
(set-fgcolor (random-color))  
(random-brush)  
(random-line image 200 100)
```

Exercise 2: Splats

- a. Read through the code to determine what `splat` does.
- b. Use `splat` to generate a few additional lines.
- c. As you may have noted, `random-art.scm` contains two different procedures for generating a random color in a limited range, `random-blue` and `random-grey`. Make sure that you understand the strategy used for each.
- d. Update `splat` to that it only generates grey lines. Draw a few more lines.
- e. Update `splat` so that it only generates blue lines. Draw a few more lines.
- f. Update `splat` so that it only uses circular brushes.
- g. Pick a few favorite brushes and update `splat` so that it selects between those brushes.

Exercise 3: New Kinds of Random Drawings

- a. Write a procedure, `(random-ellipse img width height)`, that selects an unpredictable ellipse in `img`. (The parameters `width` and `height` represent the width and height of the image. You can use them or ignore them.)
- b. Test your procedure with a sequence of commands like the following:

```
(define img (create-img 200 100))  
(random-ellipse img 200 100)  
(stroke img)
```

- c. In addition to selecting a random ellipse, we might also want to select a random foreground color, a random background color, and even a random brush. Rather than retyping a sequence of commands, we might encapsulate them into a procedure, which we might call `splat`.

Write a procedure, `(blob img width height)` that

- chooses a random foreground color (perhaps from a certain group of colors),
- chooses a random background color (perhaps from a certain group of colors),
- chooses a random brush (perhaps from a certain group of brushes).
- selects a random ellipse (using `random-ellipse`),
- fills the ellipse with the background color, and

- strokes the ellipse.

d. Test your procedure by drawing a few blobs on the screen.

Exercise 4: Repeated Blobs

In the previous exercise, you wrote a procedure, `blob` and then called it a few times. In practice, most programmers don't like to type the name of a procedure again and again and again. What's the solution? A new procedure that repeatedly calls `blob`.

Write and test a procedure, `(blobs img width height times)` that draws a blob on the image `times` times.

Exercise 5: Color Grids

a. Set the current brush to a medium circle.

```
(set-brush "Circle (09)")
```

a. From the Xtns menu, select Script-Fu, then Glimmer, and finally `Color Grid`. A dialog box should appear. Enter 100 for the width and height, and 10 for the horizontal and vertical spacing. Observe the image that appears.

b. What do you expect to happen if you use 8 for the horizontal and vertical spacing? Confirm or reject your prediction experimentally.

c. Set the current brush to a larger fuzzy circle.

```
(set-brush "Circle Fuzzy (15)")
```

d. What do you expect to happen if you again use 8 for the horizontal and vertical spacing? Confirm or reject your prediction experimentally.

e. What do you expect to happen if you use `func3` for the red component (continuing to use `func2` for green and `func3` for blue)? Confirm or reject your prediction experimentally.

f. Try one or two other brushes and one or two other arrangements of functions.

Exercise 6: Specifying Components

a. With DrScheme, create a new file on your desktop, `components.scm`.

b. Add the following function to that file.

```
(define func4
  (lambda (x y)
    (modulo (+ (* 3 x) (* 5 y)) 256)))
```

c. Make a link to that file in your Script-Fu scripts folder by opening a new terminal window and typing the following command:

```
ln -s /home/username/Desktop/components.scm .gimp-2.2/scripts/components.scm
```

(You only need to do this once, but it makes sure that the GIMP knows about the file.)

d. Refresh scripts by selecting Xtns->Script-Fu->Refresh Scripts. (You will need to refresh scripts whenever you change `components.scm`.)

e. Verify that you can now use `func4` in building a Color Grid.

f. Add the following procedure to `components.scm`.

```
(define func5
  (lambda (x y)
    (modulo (* x (abs (sin y))) 256)))
```

g. Verify that you can use this procedure. (If you can't remember that you need to save the file and to refresh scripts.)

h. Add a few of your own "interesting" component functions.

Exercise 7: Anonymous Components

As you may have noted from the previous exercise, it can be a bit of a pain to remember to save and refresh every time you define a new component function. As the reading suggests, we can instead use anonymous functions.

a. Open the Color Grid dialog box, use the following for the red, green, and blue components, and then display the grid.

- `(lambda (x y) (modulo (* x 5) 256))`
- `(lambda (x y) (modulo (* y 5) 256))`
- `(lambda (x y) 0)`

b. Try some functions of your own design. Remember that each function has the form `(lambda (x y) expression)` and that the expression should return a value in the range `[0..255]`.

Exercise 8: Making Grids from the Console

Some people (myself included) prefer to generate their grids from the console. As you may recall from the reading, the `color-grid` procedure has seven parameters:

- the width of the image
- the height of the image
- the horizontal spacing between points
- the vertical spacing between points
- the red component function

- the green component function
- the blue component function

For example, here's the command that we might use for the first grid we drew (100x100, 10x10 grid)

```
(color-grid 100 100 10 10 func1 func2 func3)
```

- Try that example.
- Try varying the parameters (e.g., using a different width, height, spacing, or function).
- Try using lambda expressions for the functions. For example

```
(color-grid 100 100 8 9 (lambda (x y) (modulo (* x y) 256)) (lambda (x y) (modulo (* x 5) 256)) (lambda (x y) (trunc (* 255 (abs (sin (* x y)))))))
```

Copyright © 2006 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.