

## Laboratory: Programming with the GIMP Tools

**Summary:** In this laboratory, you will explore how one instructs the GIMP to create some simple images.

### Contents:

- Preparation
- Exercises
  - Exercise 1: Simple Faces
  - Exercise 2: Drawing Faces, Revisited
  - Exercise 3: Drawing Houses
- For Those With Extra Time
  - Extra 1: Animation

## Preparation

Create a new 200x200 image called canvas.

## Exercises

### Exercise 1: Simple Faces

Here are the instructions for drawing a simple face from the reading, modified to work as a procedure.

```
;;; Procedure:
;;; draw-smiley!
;;; Parameters:
;;; image, an image
;;; Purpose:
;;; Draw a variant of a smiley face in the given image.
;;; Produces:
;;; [Nothing of import; Called for the side effect]
;;; Preconditions:
;;; image is at least 200 by 200
;;; Postconditions:
;;; image now contains something that might be deemed to be a
;;; smiley face.
(define draw-image!
  (lambda (image)
    ; Draw the primary circle
    (image.select-ellipse! image selection.replace
      10 10 180 180)
    (envt.set-fgcolor! color.yellow)
    (image.fill! image)
    (envt.set-fgcolor! color.black)
    (envt.set-brush! "Circle (09)")
    (image.stroke! image)
```

```

; Draw the eyes
(image.select-ellipse! image selection.replace
  50 60 30 20)
(image.select-ellipse! image selection.add
  120 60 30 20)
(envt.set-fgcolor! color.white)
(image.fill! image)
(envt.set-fgcolor! color.black)
(envt.set-brush! "Circle Fuzzy (07)")
(image.stroke! image)
(image.select-ellipse! image selection.replace
  60 60 10 20)
(image.select-ellipse! image selection.add
  130 60 10 20)
(envt.set-fgcolor! "light steel blue")
(image.fill! image)
; Smile
(image.select-ellipse! image selection.replace
  40 60 120 100)
(image.select-ellipse! image selection.subtract
  40 45 120 100)
(envt.set-fgcolor! color.white)
(image.fill! image)
(envt.set-fgcolor! color.red)
(envt.set-brush! "Calligraphic Brush#3")
(image.stroke! image)
; Get ready to show
(image.select-nothing! image)))

```

a. Copy these instructions into your definitions pane. Read through the instructions to see what image you predict they will draw.

b. In the interactions pane, create a new 200x200 image called `canvas` and use `draw-smiley!` to draw in that image. If things have gone well, you should see a new image with a smiling face.

If the image is not visible, try clicking on the window or typing `(envt.update-displays!)`

c. In your interactions window, write instructions for drawing a nose onto the image. Your instructions might look something like the following:

```

(image.select-rectangle! canvas selection.replace 90 90 20 20)
(image.select-ellipse! canvas selection.intersect 80 85 30 30)
(envt.set-fgcolor! color.green)
(image.fill! canvas)

```

If the instructions don't seem to create a new drawing, try clicking on the window or typing `(envt.update-displays!)`.

d. Add the instructions for drawing the nose to the definition of `draw-smiley!`.

e. It is possible to clear an image by selecting everything and then clearing the selection. Do so now.

```
(image.select-all! canvas)
(image.clear-selection! canvas)
(envt.update-displays!)
```

f. Sometimes it's more fun to see an image drawn bit-by-bit. DrFu allows you to pause a bit between actions, as when you want to show each step in the creation of a drawing. In particular, the procedure `(usleep n)` pauses for  $n$  microseconds. Hence, if you want to pause for a half a second, you might write `(usleep 500000)`.

Insert instructions to sleep for half a second and to update the displays immediately after each call to `image.fill!` and `image.stroke!` and redraw the image.

## Exercise 2: Drawing Faces, Revisited

- Write a procedure, `(image.draw-face! image)`, that draws a face different than the sample we gave above. You should try a different shape of face, different brushes, and additional facial features.
- What do you expect to happen if you try to draw this face in a 100x100 image or a 300x300 image?
- Check your answer experimentally.
- If you predicted or encountered results that you consider undesirable, summarize how you might fix them. (Don't fix them, just indicate how you might fix them.)

## Exercise 3: Drawing Houses

- Write a procedure, `(image.draw-house! image left bottom width height)`, that draws a house in `image`. The house should be the specified height and width, and its lower-left corner should be at `(left,bottom)`.
- Using `image.draw-house!`, make a small village.

## For Those With Extra Time

### Extra 1: Animation

As our step-by-step face drawing suggested, it is possible to do simple (but not incredibly attractive) animations by drawing something, updating the displays, pausing for a bit, drawing something else, updating the displays, and so on and so forth.

If we are to generalize the idea of animation, we might think about writing a procedure, `(draw-frame! image frame-number total-frames)`, that draws one frame in the sequence. For example, here's a simple picture that draws a circle whose radius and position depend on the frame number.

```

(define draw-frame!
  (lambda (image frame-number total-frames)
    (image.select-ellipse! image selection.replace
      0 0
      (* 5 (+ frame-number 4))
      (* 11 (+ frame-number 1)))
    (envt.set-fgcolor! (rgb.new (modulo (* 10 frame-number) 256)
      (modulo (* 15 frame-number) 256)
      (modulo (* 20 frame-number) 256)))
    (image.fill! image)))

```

We could show a sequence of frames with instructions something like the following:

```

(draw-simple-frame! canvas 0 10)
(envt.update-displays!)
(usleep 100000)
(image.select-all! image)
(image.clear-selection! image)
(draw-simple-frame! canvas 1 10)
(envt.update-displays!)
(usleep 100000)
(image.select-all! image)
(image.clear-selection! image)
...
(draw-simple-frame! canvas 8 10)
(envt.update-displays!)
(usleep 100000)
(image.select-all! image)
(image.clear-selection! image)
(draw-simple-frame! canvas 9 10)

```

Of course, whenever we write code that repetitious, we consider whether it would be better to write a procedure that does the repetition for us. Fortunately, the DrFu library contains such a procedure.

`(image.animate! image draw-frame num-frames)` creates a simple animation by calling *draw-frame* on each frame number from 0 to *num-frames*-1.

- a. Test `image.animate!` with the `draw-simple-frame!` procedure above.
- b. Write your own, more interesting, `draw-frame!` procedure and use it to create an animation.

Copyright © 2007 Janet Davis, Matthew Kluber, and Samuel A. Rebelsky. (Selected materials copyright by John David Stone and Henry Walker and used by permission.) This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. 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.