Fundamentals of Computer Science I: Media Computing (CS151.02 2007F)

Programming with the GIMP Tools

Summary: We consider another model of image, one that involves giving instructions based on the core tools associated with the GIMP.

Contents:

- Introduction: Models of Drawing
- An Example
- GIMP Context
- Selecting Regions
- Working with Selections
- A Few Other Useful Tools

Introduction: Models of Drawing

As we've noted in the past, in order to write algorithms, one must also have a representation for the manipulated data and a collection of primitive procedures with which to manipulate those data. For example, for colors, we have built-in procedures to create colors (e.g., `rgb.new`), extract components (e.g., `rgb.red`), and to modify them in fixed ways (e.g., `rgb.darker`).

But both the representation and the collection of related procedures are motivated by a deeper issue, the *model* we use to think about the data. For example, we have seen two simple ways to think about images: We can think of images purely as grids of pixels or we can think of images as collections of positions and colors, which can be rendered on a grid of pixels in a variety of ways.

But there are certainly other ways to think about images. We might also describe an image in terms of the sequence of actions used to create the image. For example, most people who draw with the GIMP do so by selecting appropriate tools (in sequence) and using those tools and pointing device to draw the image. We might therefore represent images by giving sequences of commands that a human might carry out, but that a computer might also carry out. It is this model that we consider in the reading.

In particular, in the GIMP, as in most painting programs, artists and designers create new images by selecting appropriate tools and using those tools to draw. Once a base image is created, they may also modify that image using a variety of filters. Since we've already explored ways to write filters, we'll take a step backwards and look at the basic tools for creating images.

An Example

If we were using Scheme to communicate with other humans (a scary concept, isn't it?), we might write something like the following to tell someone how to draw a simple variant of our ubiquitous smiley face.

```
(define smiley (image.new 200 200))
(image.show smiley)
; Draw the primary circle
(image.select-ellipse! smiley selection.replace
                       10 10 180 180)
(envt.set-fgcolor! color.yellow)
(image.fill! smiley)
(envt.set-fgcolor! color.black)
(envt.set-brush! "Circle (09)")
(image.stroke! smiley)
; Draw the eyes
(image.select-ellipse! smiley selection.replace
                       50 60 30 20)
(image.select-ellipse! smiley selection.add
                       120 60 30 20)
(envt.set-fgcolor! color.white)
(image.fill! smiley)
(envt.set-fgcolor! color.black)
(envt.set-brush! "Circle Fuzzy (07)")
(image.stroke! smiley)
(image.select-ellipse! smiley selection.replace
                       60 60 10 20)
(image.select-ellipse! smiley selection.add
                       130 60 10 20)
(envt.set-fgcolor! "light steel blue")
(image.fill! smiley)
; Smile
(image.select-ellipse! smiley selection.replace
                       40 60 120 100)
(image.select-ellipse! smiley selection.subtract
                       40 45 120 100)
(envt.set-fgcolor! color.white)
(image.fill! smiley)
(envt.set-fgcolor! color.red)
(envt.set-brush! "Calligraphic Brush#3")
(image.stroke! smiley)
; Get ready to show
(image.select-nothing! smiley)
```

What’s going on here? The model for much of the drawing in the GIMP is that you select portions of the image and then either fill them or stroke (trace the edges of) them. We draw the face by selecting appropriate sections, colors, and brushes, and then stroking or filling, as appropriate. We discuss each of these operations in the following sections.

GIMP Context

Before looking at the commands that place digital ink on the canvas, As the example above suggests, before putting digital ink in the screen, it is important to set contextual information that guides how the GIMP interprets drawing actions. The most important contexts for drawing are the *foreground color*, the *background color*, and the *brush*.

When you draw in a painting program, the program often assumes that you want to draw using selected brush. As is the case with physical brushes, different digital brushes give very different effects. For example, a square brush across the image will give you a very different kind of line than will a fuzzy circle.

In DrFu, there are a few primary procedures for working with brushes.

- `(brush.list)` gives a list of the names of all the available brushes.
- `(brush.list` *`name`*`)` gives a list of the names of all the available brushes that include *name*. For example, `(brush.list "circle")` lists all the brushes whose name includes the word `"circle"`.
- `(envt.set-brush! "`*`brush-name`*`")` sets the working brush.

We recommend that you explore the strengths and weaknesses of each brush for a variety of situations.

You've already encountered the foreground and background colors, which you set with `envt.set-fgcolor!` and `envt.set-bgcolor!`. While these procedures work best with RGB colors, you can also use them with color names. (Behind the scenes, both procedures convert the name to an RGB color.)

Selecting Regions

As the example suggested, a lot of simple drawing can be done by selecting interesting regions and then stroking or filling those regions. There are two basic operations for selecting regions, `image.select-ellipse!` and `image.select-rectangle!`. The two procedures take exactly the same list of parameters:

- `image`, the image in which to select a region;
- `operation`, the way in which we want to update the selection;
- `left`, the left edge of the selected region;
- `top`, the top edge of the selected region;
- `width`, the width of the selected region; and
- `height`, the height of the selected region.

For ellipses, `left` and `top` represent the left edge and top edge of the rectangle that bounds the ellipse.

Except for `operation`, all of the remaining parameters should be obvious. The operation stems from an interesting, but useful, design decision in the GIMP: Often, we build more complex images by selecting more complex regions. However, if our only building blocks for describing selections are rectangles and ovals, we need appropriate ways to combine those building blocks. That's where the operation comes into play. The operation can be

- `selection.replace`, in which case the old selection is replaced by the new selection;
- `selection.add`, in which case the old selection is extended by the new selection, even if the two seletions are non-contiguous;
- `selection.subtract`, in which case any pixels in the the old selection that are also in the new selection are de-selected; and

- `selection.intersect`, in which case only pixels that are in both the old and new selections remain selected.

There are also a few procedures that permit one to select more broadly:

- `(image.select-nothing! image)` deselects everything.
- `(image.select-all! image)` selects everything in the image.

Working with Selections

Once you've made an appropriate selection, what can you do with that selection? Only a few things. You can fill the interior of the image using the current foreground color with `(image.fill! image)`. You can trace the exterior of the image with `(image.stroke! image)`. You've seen examples of both commands in the example above. The third possibility is that you can "clear" the selection with `image.cleaer-selection!`. In simple images, clearing a selection results in the selection getting filled by the background color. In layered images, clearing the selection may reveal images from lower layers.

Because designers may not want all of the changes to an image to appear piecemeal, the GIMP does not automatically update the displayed version of the image after one of these commands. The updates only become visible after either (a) you click on the image or (b) you call `(envt.update-displays!)`.

A Few Other Useful Tools

While a lot of drawing can be done with selection, stroking, and filling, it is also useful to have other tools available. DrFu provides two other simple GIMP commands that focus on simple kinds of drawings.

`(image.blot!` *`image col row`*`)` draws a single spot at the specified position using the current brush and foreground color.

`(image.draw-line!` *`image c1 r1 c2 r2`*`)` - draw a line from (*c1*,*r1*) to (*c2*,*r2*) using the current brush and foreground color.

Neither of these procedures can easily be simulated with the selection approach to drawing, so they provide a useful addition to the algorithmic designer's toolbox.
