Fundamentals of Computer Science I (CS151.01 2006F)

Drawing Images with Script-Fu

Summary: We explore techniques for making and transforming drawings.

Contents:

Exercises

Exercise 0: Preparation

a. Create a new file for your procedures for this lab (say, `my-drawings.scm`). Open that file in DrScheme.

b. Start the GIMP and open a Script-Fu console.

c. Load your new file and `/home/rebelsky/Web/Courses/CS151/2006F/Examples/drawing.scm`.

Exercise 1: Simple Drawings

As you may recall from the reading, we can build sequences of points by making a list of `point` values and draw those sequences with the `connect-the-dots` procedure.

a. Define a sequence of points that might describe a tree. If you don’t like figuring out the points yourself, you can use the following.

```
(define tree (list (point 15 30) (point 18 14) (point 5 18) (point 8 8) (point 20 0) (point 30 2) (point 35 19) (point 23 16) (point 23 30)))
```

b. Create a new image and call it slate.

```
(define slate (create-image 400 400))
(show-image slate)
```

c. Set the foreground color to a shade of green and the brush to one of the smaller simple brushes (say a `"Circle (03)"` or `"Circle Fuzzy (05)"`).

d. Draw the tree image using `connect-the-dots`.

```
(connect-the-dots slate tree)
```

Exercise 2: Your Own Image

Create your own interesting simple image using a sequence of no more than twelve points.

Exercise 3: Translation

Recall that there are two ways to translate (move) a drawing, `htrans` (and its friends `htrans-point` and `htrans-points`) and `vtrans` (and its friends `vtrans-point` and `vtrans-points`).

a. Draw a copy of the tree (or your image) translated 20 to the right

```
(connect-the-dots slate (htrans-points 20 tree))
```

b. Draw a copy of the tree (or your image) translated 15 down with

```
(connect-the-dots slate (vtrans-points 15 tree))
```

c. Draw a copy of the tree (or your image) translated 30 to the right and 40 down with

```
(connect-the-dots slate (vtrans-points 40 (htrans-points 30 tree)))
```

d. What do you expect to happen if the first parameter to `htrans-points` is negative? (For example, what if it's -5?)

e. Check your answer by translating the tree (or your image) -5 to the right, but don't print it.

```
(htrans-points -5 tree)
```

f. Draw the translated tree.

g. What do you expect to happen if the first parameter to `vtrans-points` is negative? (For example, what if it's -10?)

h. Check your answer.

Exercise 4: Scaling Drawings

Recall that there are three ways to scale (resize) a drawing, `scale` (and its friends `scale-point` and `scale-points`), `hscale` (and its friends `hscale-point` and `hscale-points`), and `vscale` (and its friends `vscale-point` and `vscale-points`).

a. Draw a copy of your tree (or other drawing), scaled by 3 with

```
(connect-the-dots slate (scale-points 3 tree))
```

b. Draw a copy of your tree (or other drawing), scaled horizontally by 1.5 with

```
(connect-the-dots slate (hscale-points 1.5 tree))
```

c. Draw a copy of your tree (or other drawing), scaled vertically by 2 with

```
(connect-the-dots slate (vscale-points 2 tree))
```

d. Draw a copy of your tree (or other drawing), scaled horizontally by 2 and vertically by 3. (We leave it up to you to figure out how.)

e. Draw a copy of your tree, scaled vertically by 1.5 and translated horizontally by 20.

f. What do you expect to happen if you horizontally scale the tree (or other drawing) by 1/2 (0.5)?

g. Check your answer experimentally.

h. What do you expect to happen if you vertically scale the tree (or other drawing) by -1?

i. Check your answer experimentally. (If you don't see the image, you may want to check the points in the result and translate them if necessary.)

Exercise 5: Multiple Transformations

At times, you will find it convenient to build your own compound transformations. How? One possibility is to do something like what we've done above - apply one transformation, then the next, then the next. For example, suppose we want to translate the image right 20, scale it by 0.5, and then translate it down 5. We can write

```
(define transform
  (lambda (points)
    (vtrans-points 5 (scale-points 0.5 (htrans-points 20 points)))))
```

However, we can also use our friendly helpers `map`, `l-s`, and `compose`.

```
(define transform
  (l-s map (compose (l-s vtrans-point 5) (l-s scale-point 0.5) (l-s htrans-point 20))))
```

Is one better than the other? Some find the former clearer. Some find the latter clearer. We mention both because you will need to choose one or the other (or something similar) for the following problems.

a. Write (but do not execute) instructions for translating a drawing right by 5 and then scaling by 2.

b. Write (but do not execute) instructions for scaling a drawing by 2 and then translating right by 5.

c. Do you expect the results of the two preceding transformations to be the same or different?

d. Confirm or reject your previous answer experimentally.

Exercise 6: A Complex Drawing

The file `td.scm` contains a fairly complex drawing represented as a list of lists. Here it is

```
(define td
  (list
   (list (point 65 27) (point 60 28) (point 40 22) (point 25 25)
         (point 15 40) (point 30 60) (point 45 65) (point 52 70)
         (point 50 80) (point 35 85) (point 18 80))
   (list (point 62 42) (point 40 37) (point 31 40)
         (point 31 50) (point 50 58) (point 60 70)
         (point 55 85) (point 50 90) (point 30 88) (point 18 80))
   (list (point 65 27) (point 64 32) (point 43 30)
         (point 64 40) (point 62 42))
   (list (point 51 23) (point 52 24) (point 52 23))
   (list (point 57 24) (point 58 25) (point 58 24))
   (list (point 52 18) (point 55 25) (point 57 18))
   (list (point 38 89) (point 35 103) (point 45 102))
   (list (point 46 89) (point 48 98) (point 53 97))
   (list (point 64 32) (point 63 35) (point 62 32))
   (list (point 60 32) (point 59 35) (point 58 32))
   (list (point 56 31) (point 55 35) (point 54 31))
   (list (point 56 36) (point 57 33) (point 58 37))
   (list (point 60 38) (point 61 35) (point 62 39))
   (list (point 68 33) (point 72 31) (point 70 34)
         (point 85 33) (point 80 38) (point 85 38)
         (point 95 40) (point 90 41) (point 100 45)
         (point 92 44) (point 95 47) (point 85 45)
         (point 87 47) (point 78 45) (point 80 48)
         (point 72 42) (point 73 45) (point 65 37))
   (list (point 50 80) (point 43 73) (point 45 82))
   (list (point 43 83) (point 39 77) (point 35 85))
   (list (point 35 85) (point 31 78) (point 29 83))
   (list (point 30 25) (point 33 15) (point 32 5)
         (point 43 12) (point 40 15) (point 41 17)
         (point 37 19) (point 39 21) (point 33 24))
   (list (point 28 28) (point 18 20) (point 10 10)
         (point 5 20)  (point 8 20)  (point 9 25)
         (point 12 25) (point 15 28) (point 27 29))
   (list (point 20 32) (point 2 35) (point 0 48)
         (point 2 55)  (point 0 60) (point 8 62)
         (point 8 60) (point 6 60) (point 8 59)
         (point 8 57) (point 6 57) (point 8 56)
         (point 8 54) (point 5 54) (point 9 40)
         (point 9 43) (point 12 42) (point 15 40))
  ))
```

Make your own copy of that file and load it into DrScheme.

a. Set the foreground color to black and the brush to something small (say `"Circle (01)"`).

b. Draw the first component with

```
(connect-the-dots slate (car td))
```

c. Draw the second component with

```
(connect-the-dots slate (cadr td))
```

d. Draw a few random components with

```
(connect-the-dots slate (list-ref td (random (length td))))
```

e. When you are confident that you know what the figure is (or when you're sick of drawing random components), draw the whole figure with

```
(draw slate td)
```

f. Draw some scaled and/or translated versions of the image.

g. What do you expect to have happen if you execute the following command once?

```
(draw slate (vary 4 (scale 3 td)))
```

h. Check your answer experimentally.

i. What do you expect to have happen if you execute the previous command ten or so times?

j. Check you answer experimentally.

Exercise 7: Algorithmic Drawings

The reading provides two procedures that algorithmically generate a list of points, `(spiral` *`n`*`)` and `(zig-zag` *`n`*`)`.

a. Draw some simple spirals of size 45, 90, 180, and 360.

b. Draw zig-zags of size 10, 20, 30, and 40.

c. Draw some translated and scaled zig-zags and spirals.

Exercise 8: Varying Images

As you may recall from the reading, `(vary-points amt points)` takes a list of points and varies each coordinate of each point by a value between `-amt` and `amt`.

a. What do you expect to happen if we make a “size 20” zig-zag, with each point varied by at most 3? E.g.,

```
(connect-the-dots slate (vary-points 3 (zig-zag 20)))
```

b. Check your answer.

c. What do you expect the result of the following to be?

```
(set-brush "Circle (03)")
(set-fgcolor BLACK)
(connect-the-dots slate (vary-points 3 (zig-zag 20)))
(set-fgcolor RED)
(connect-the-dots slate (vary-points 3 (zig-zag 20)))
(set-fgcolor BLUE)
(connect-the-dots slate (vary-points 3 (zig-zag 20)))
```

d. Check you answer.

e. What do you expect to happen if we make a “size 180” spiral, with each point varied by at most 2?

f. Check your answer.

Exercise 9: Algorithmic Drawing, Revisited

Write a procedure to algorithmically generate some other simple figure. For example, you might generate a regular polygon of three or more sides. (How? Pick some radius (say 10), divide the circle into *n* equal angles, and, for each angle, generate the point `(point (* radius (sin angle)) (* radius (cos angle)))`.

For Those With Extra Time

If you find that you have spare time, write a `rotate-point` procedure (and its friend `rotate-points`). How do you rotate a point?

- Find the radius by taking the square root of the sum of the squares of the x and y coordinates.
- Find the "angle" to that point using `acos`, `asin`, or `atan`.
- Add the rotation angle to the angle you just found.
- Let the new x be the radius times the cos of the new angle.
- Let the new y be the radius times the sin of the new angle.
