

CSC151.02 2014S, Class 22: Images as Functions from Position to Colors

Overview

- Preliminaries.
 - Upcoming work.
 - Admin.
 - Questions.
- Topics.
 - Models of images.
 - Iterating over positions: Images as functions from position to color.
 - Examples.
 - Blends and other positionally-computed images.
 - Drawing simple shapes.
 - Some strange computations.
- Questions.
- Lab.

Preliminaries

Upcoming Work

- Work on homework 5.
- Reading for Wednesday: Building Data Structures with Heterogeneous Lists
- Writeup: Exercise 4
 - Subject: CSC 151.02 Writeup 14: Iterating Positions (OPTIONAL NAMES)

Admin

- Newish lab partners. (I didn't pay much attention to the computer's assignment, so you may find out that you have familiar partners.)
- I've forgotten the students who asked for individual or small group tutors. Please send me an email if you're interested.
- Our computer system is (still) acting up. I have not received any clear indications as to when it will be healthy again.
- Congratulations to Ajuna for her 120 in bowling yesterday. (It beats my average.)
- I've finished getting all of yesterday's eboard into place.
- A few of you are frustrated that your indentation changes when you cut and paste into Google docs. If you use Chromium for accessing Google docs, cut and paste *should* work.

- Extra credit:
 - Tuesday Town Hall, noon in JRC 101, 7:30 p.m. in Harris.
 - Tuesday Wellness event, 5:30-8:00 pm.
 - Wednesday Royce Wolf concert.
 - Thursday CS Extras at 4:30 p.m. in 3821: New CS Curriculum.
- Other good things to do
 - Go see Martin Estrada's show in Smith
 - Go to tonight's Faulconer talk - Works with cars and video

Questions on the Homework

What should our solution to problem 3 look like?

```
;;; Procedure:
;;; dwelling
;;; Parameters:
;;;   width, a positive integer
;;;   height, a positive integer
;;; Purpose:
;;;   ...
(define dwelling
  (lambda (width height)
    (let ([scene (image-new width height)])
      ; Standard instructions
      (context-set-fgcolor (irgb 128 64 196))
      (image-select ellipse! scene REPLACE 10 10 (- width 20) 20)
      (image-fill! scene)
      ...
      ; The required two-way decision
      (if (< width 100)
          ...
          ...)
      ; The required three-way decision
      (cond
        [((< width height)
          ...]
        [(> width height)
          ...]
        [else
          ...]))
      ; The required one-way decision
      (when (= width 42)
        (image-show-text! scene "You got the answer!"))
      scene))))
```

What do you want for the three examples?

Code. We'll run the code.

Models of images

- Course model
 - Problem solving in a formal environment
 - Idea: Different basic operations lead to different solutions
- How might you make a 100x100 image with a 50x50 black outlined circle.
 - Approach one: Use drawings as values
 - Scale the drawing unit circle by 50
 - Shift it horizontally by 50 and vertically by 50
 - Scale the drawing unit circle by 48
 - Shift it horizontally by 50 and vertically by 50
 - Recolor it white
 - Group the two circles
 - Render it with something like (image-show (drawing->image asdfasd 100 100))
 - Approach two: Use the Gimp tools
 - (context-set-bgcolor "white")
 - (define image (image-new 100 100))
 - (image-select-ellipse image REPLACE 25 25 50 50)
 - (context-set-fgcolor! "black")
 - (context-set-brush! "2. Hardness 100" 3)
 - (image-stroke! image)
 - (image-select-nothing! image)
 - (image-show image)

Iterating over positions: Images as functions from position to color

- Images are just a bunch of pixels, each with a column and a row.
- We can write a function that assigns a color based on the column and the row
- (image-compute (lambda (x y) INSTRUCTIONS-TO-GIVE-COLOR)) WIDTH HEIGHT)

Examples

A simple color blend

```
> (image-show (image-compute (lambda (x y)
                               (irgb x 0 (- 256 x)))
                               256 256))
```

Something a bit more geometric

```
> (image-show (image-compute (lambda (x y)
                              (if (< x y)
                                  (irgb 255 0 0)
                                  (irgb 0 0 0)))
              256 256))
```

Something very strange. Yay irgb!

```
> (image-show (image-compute (lambda (x y)
                              (* 100 x y))
              256 256))
```

Something equally strange.

```
> (image-show (image-compute (lambda (x y)
                              (* 2 x y))
              256 256))
```

Questions

Lab

Samuel A. Rebelsky, rebelsky@grinnell.edu

Copyright (c) 2007-2014 Janet Davis, Samuel A. Rebelsky, and Jerod Weinman. (Selected materials are copyright by John David Stone or Henry Walker and are used with permission.)



This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.