Meta

- Project: *SVG Digital Halftoning*
- Organization: Portland State Univ.
- Mentor: Bart Massey bart.massey@gmail.com
- Student Name: Morgan Gangwere morgan.gangwere@gmail.com
- Post feedback to morgan.gangwere@gmail.com

The recommended text, *Modern Digital Halftoning (2nd ed)*, was not easily get-able by myself without a hefty ($200!) cost and long delay. As a result, though it may be a fantastic reference, its guidance is likely out of reach. A loaned copy would be appreciated as my university doesn't seem to have a non-reference copy in very bad condition.

Project Original description

Halftoning is the process of taking color or grayscale image to monochrome or reduced-color by using dither patterns. The classic case is photograph reproduction for print newspapers.

Typical halftoning algorithms produce bitmaps as output. However, there are cases when scalable halftone patterns would be useful, and SVG is the obvious choice for scalable graphics. A halftone generator written in C and Python and integrated into Inkscape as a tool would be a useful addition for digital artists.

Observed Halftoning Processes

I looked through a selection of print media with varying requirements and found these major styles of halftoning patterns:

- Dot-grid halftoning: This process involves creating angled grids of color, typically offset at various angles for the CMYK channels. Using circles for color splotches gets a "pretty good" version of the image. For grayscale images, it commonly (as opposed to hue) of the area the dot represents.
- Stipple halftoning: This process is famous in the "stippled picture" portraits within the New York Times and other print news sources. A fundamentally stochastic process, this method works well for low-density images that start as grayscale images.
- Flat-Flood halftoning: This "halftoning" is more of an art style, where solid spaces of color are expressed by a solid CMY (less K) block, with one channel having holes or one being comprised of dots to represent the added tint to the other channels. Typically more common with digital printing mechanisms, this process does produce a very "cartoon" or "comic book" style.
- Offset grid halftoning: Contrasted to dot-grid halftoning, this process appears more often in newsprint for small run operations--my university paper uses this method in their paper--where a constant grid is offset at a constant distance. Typically, this grid is hexagonal (for better density) and uses "target" shaped dots (a central dot with a ring around it).
- The book "Modern Digital Halftoning" makes mention of a line halftone. This process was made famous by the AT&T logo (which was nothing more than a shaded sphere run through the process) producing the infamous "Death Star" look.

Previous open source implementations

There are a handful of current FOSS halftoning implementations. A cursory search across the Internet reveals a few implementations that are limited in their speed and efficiency. The most commonly referenced one I can find is from GitHub, from a StackOverflow discussion: https://github.com/philgyford/python-halftone

This implementation is naive: It doesn't work well when handling color halftones due to (what I believe to be) simmply rotating the channel's image by nearly 45° in the case of a K channel to produce a "halftone". While it approaches a solution, I feel this could be improved upon signifigantly.

The largest issue with this method is that it produces a raster, not vector result. However, the algorithm could be a good starting point.

Another implementation is intended not for traditional print; however, is built for the CNC market: https://github.com/mattvenn/cad/tree/master/tools/drillpic2

Drillpic2, like the philgyford implementation, uses PIL to read pixel data from the input file. This is used to produce a "halftone" intended to be drilled into material (or drawn, etc.) via G-Code (CNC machine language) to generate coasters and other nicknacks. This method of "halftoning" also includes a few novel scan- line style implementations, which may have artistic (if not other) merits.

I refer to this as a "Scanline" or "Drillpic2" halftone.

Current Inkscape workflows for halftoning.

Long-time inkscape users have found a variety of ways to get the halftone effect they are looking for. One method uses the "Create Tiled Clones" tool to do the dirty work: The source work is chosen (typically a group) which handles a naive single-channel method (or multi-channel if the work is willing to be done) that does look "halftone-like" but which leaves something to be desred. An example of this technique is presented at http://istarlome.deviantart.com/art/halftone-effect-in-inkscape-85648573

This effect is used by some users to create laser-cut halftones: https://milwaukeemakerspace.org/2014/03/laser-cut-halftone-photo/

I refer to this form as a *Cloned Copies Halftone* or CCH.

The fine folks over at EvilMadScientist have created an external to inkscape tool called StippleGen: http://wiki.evilmadscientist.com/StippleGen . StippleGen uses Processing to implement the work of Adrian Secord of New York University; this tool, being external to Inkscape, is a bit of a bear to set up: It requires multiple libraries written in Java. This setup is built for the EggBot and other hardware projects built by the EvilMadScientist folks.

This approach uses a Vonroi density pattern to determine dot placement, but also includes a novel appoach called "TSP" -- Travelling Salesman Problem. TSP images are built from a dense Vonroi diagram and edge detection.

Finally, there are a variety of other implementations that have been built in the past, with the GIMP having a "newsprint" type filter for some time now. This filter is commonly used to generate the halftone for a single channel, then this is imported into Inkscape and vectorized.

Deliverables

I will build an inkscape plugin OR external tool that:

- Produces three kinds of halftones:
    - Dot-grid halftones (size of halftone = intensity, traditional)
    - Scanline halftones (a la drillpic2's implementation)
    - Stochastic halftoning (the "stippled picture" effect)
    - Non-rotated (overlaid) CMYK halftoning.
- Enables RGB, CMYK color separation
- Can customize
    - DPI
    - Per-channel angle offset
    - Dither and dot placement patterns)
    - Global/Per-channel color cutoffs

Midway deliverable does CMYK dot-grid halftoning, ability to choose per-channel cutoffs and dot size/density.

Stretch deliverable handles at least two more (scanline, stochastic) halftoning, plus an "inkscape to tool" bridge that glues our external tool (tentatively called "halftone-gen") should we go that route.

Python as a language choice

I feel a purely Python implementation

- requires few non-optimized libraries
- does not require compilation for two Windows platforms
- does not require the myriad compilation steps for Linux distributions
- keeps the source relatively clean.

Disadvantages:

- It's not going to be as fast as C
- It can't link against big fancy vector libraries like Boost
- Have to be able to drag all dependencies along with you (or use `pip`).

Deadlines

Calendar week 22 is week 1, the week of the 29th.

- Week 1: Get nice and familiar with the inkscape plugin API. Decide if it's better to write an SVG-producing tool or an Inkscape plugin.
- Week 3: Simple dot-grid ("CCH") halftone process (fixed parameters).
- Week 4: Adjustable parameters on CCH method, thresholds. *I am potentially speaking at Open Source Bridge in Portland, OR.*
- Week 5: Per-channel (CMYK) halftoning (with adjustable angles, etc.)
- Week 7: Stochastic dither ("Stippled Picture" halftone), Drillpic2 (scanline) algorithm.
- Week 8: Late-game optimizations, parameterization of all algorithms.
- Week 10: Documentation, writeup, A fully packaged solution for Windows, Linux.
- Week 11 (week of the 14th): Buffer time. I expect CMYK cloned-copy to slip.

About me

I'm a student from the University of New Mexico who spends my summers in the greater Seattle area. I've been writing software for much of my life, developing for Windows, Linux, Android and everything in between.