

NODEBOX FOR DATA VISUALIZATION

Lynn Cherny for PyData 2013

@arnicas

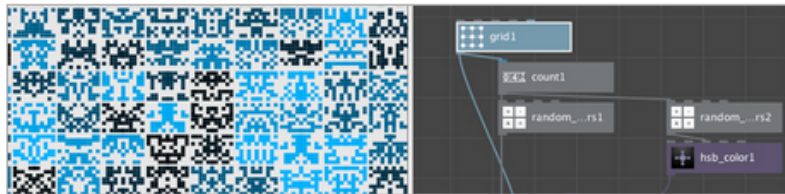
www.ghostweather.com

WHAT IS NODEBOX?

Clever tools for curious creatives.

The NodeBox family of tools gives you the leverage to create generative design the way you want.

Using our open-source tools we enable designers to automate boring production challenges, visualize large sets of data and access the raw power of the computer without thinking in ones and zeroes. Our tools integrate with traditional design applications and run on many platforms.



NodeBox 3

ACQUIRE, TRANSFORM, VISUALIZE

Cross-platform, node-based GUI for efficient data visualizations and generative design.

[Read More](#)



NodeBox 1

CODE, ITERATE, PRINT

Mac app for creating 2D visuals using Python programming code.

[Read More](#)



NodeBox OpenGL

CODE, ANIMATE

Fast cross platform graphics library.

[Read More](#)

Gallery



NodeBox 3 The Hague Workshop

NodeBox 3 Antwerp Masterclass

Blog



How to make a kaleidoscope in NodeBox 3



Generative + cnc.

FLOCK EXAMPLE

Take that, matplotlib (?!)

```
# Example from the nodebox ogl documentation page
# http://www.cityinabottle.org/nodebox/

from nodebox.graphics import *
from nodebox.graphics.physics import Flock

flock = Flock(40, 0, 0, 500, 500)
flock.sight = 300

def draw(canvas):
    background(1)
    fill(0, 0.75)
    flock.update(cohesion=0.15)
    for boid in flock:
        push()
        translate(boid.x, boid.y)
        scale(0.5 + 1.5 * boid.depth)
        rotate(boid.heading)
        arrow(0, 0, 15)
        pop()

canvas.fps = 30
canvas.size = 600, 400
canvas.run(draw)
```

Flock_example.py

NODEBOX OPENGL

```
from nodebox.graphics import *
```

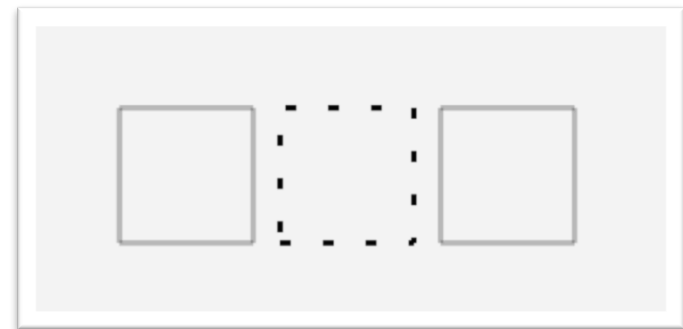
```
def draw(canvas):  
    canvas.clear()  
    nofill()  
    stroke(0, 0.25)  
    strokewidth(1)  
    rect( 50, 50, 50, 50)  
    rect(110, 50, 50, 50, stroke=Color(0), strokestyle=DASHED)  
    rect(170, 50, 50, 50)
```

Set context values

X, Y, width, height

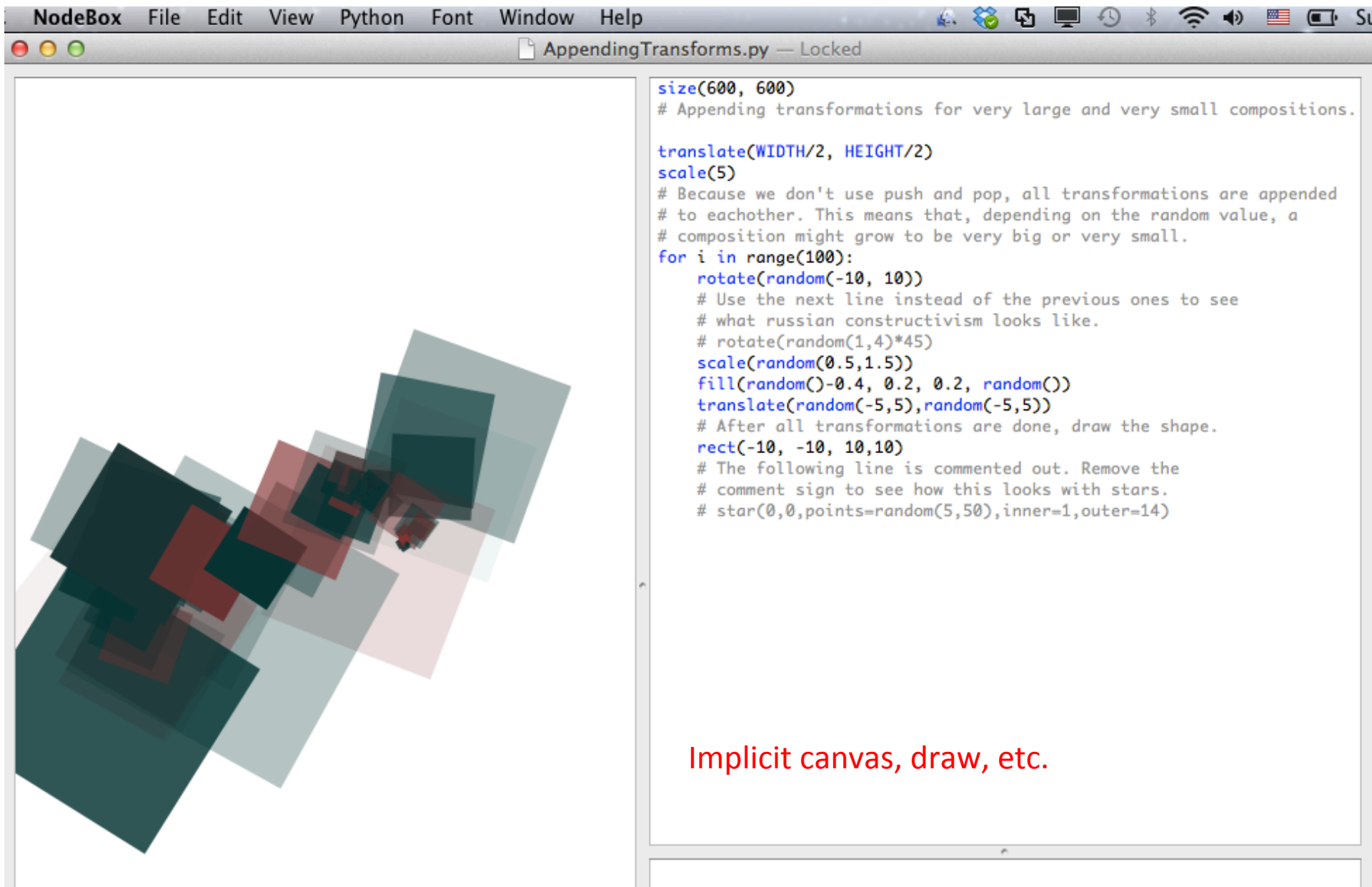
Local override of context values

```
canvas.run(draw)
```



0,0 in lower left by default (top
left in NB 1!)

NODEBOX 1 IS A SMART IDE

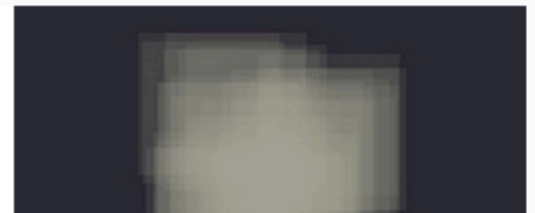
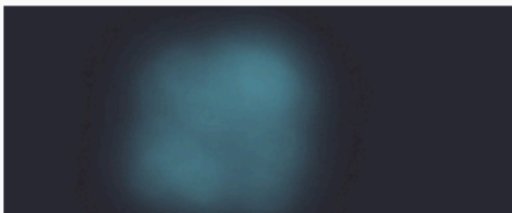
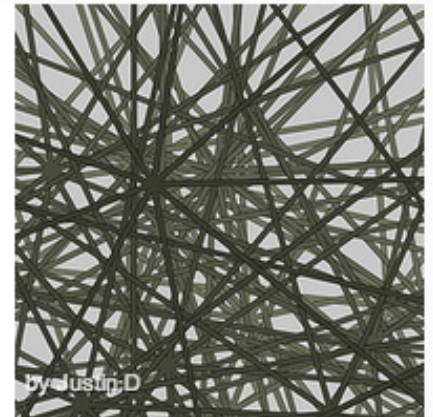
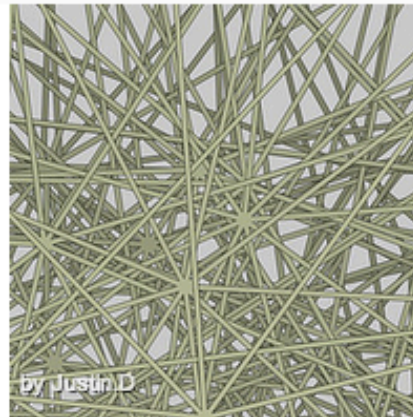
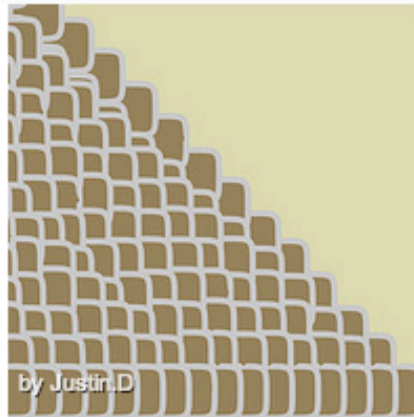
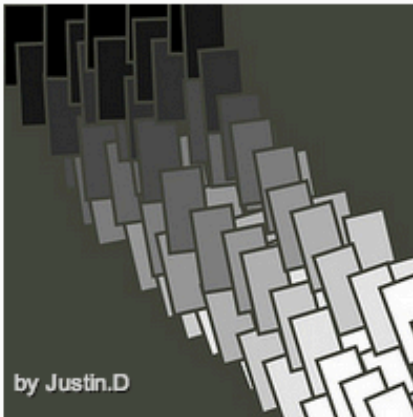
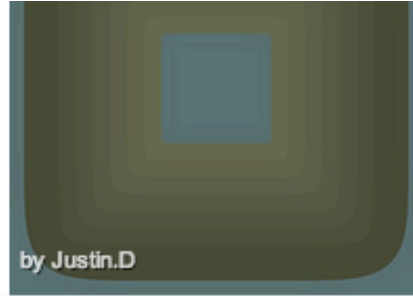
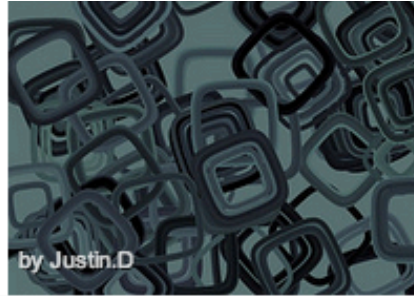
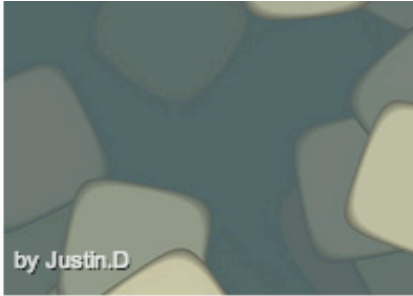


DOWNLOAD NODEBOX

	Mac OS X	Windows	Linux
NodeBox 3 <i>Version 3.0.32 — Release Notes</i>	Download	Download	Instructions
NodeBox OpenGL <i>Version 1.7</i>	Mac too, I'll demo	Download	Tentative evidence of linux too
NodeBox 1 <i>Version 1.9.7rc1</i>	Download	N/A	N/A

<http://nodebox.net/download/>

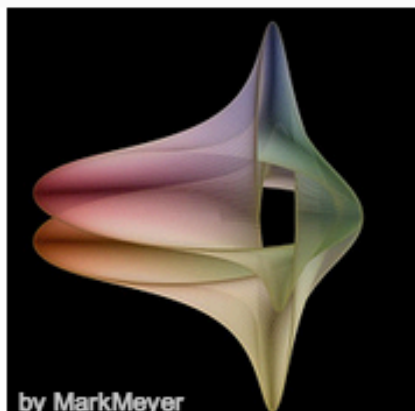
GENERATIVE ART



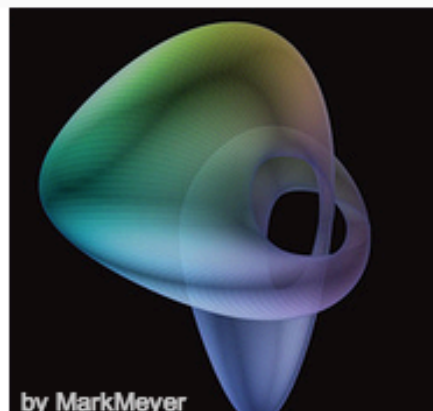
Justin D on flickr.



by MarkMeyer



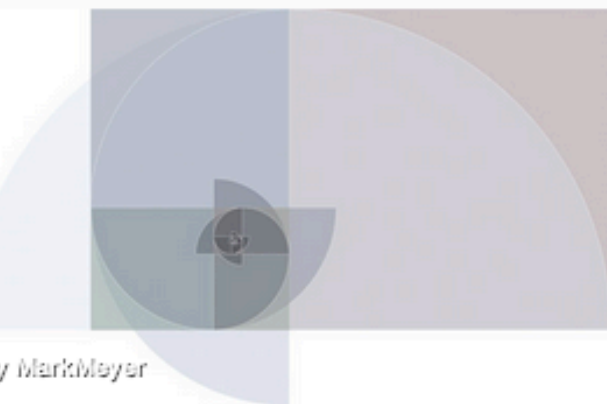
by MarkMeyer



by MarkMeyer



by Tom De Smedt



by MarkMeyer



by MarkMeyer



by MarkMeyer



by MarkMeyer



by Lucas Nijs

created with Adobe Illustrator (Creative Cloud) | Frederik De Boer - Tom De Smedt - Lucas Nijs



by markluffel



by markluffel



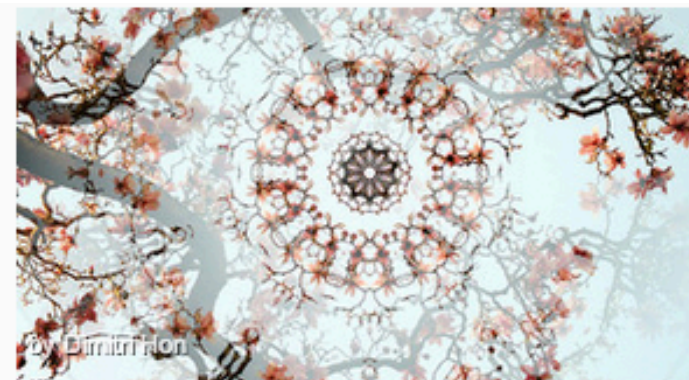
by markluffel

HIGH QUALITY GRAPHIC OUTPUT

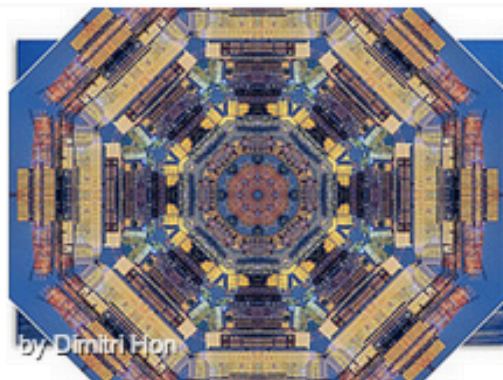


by Dmtr.org

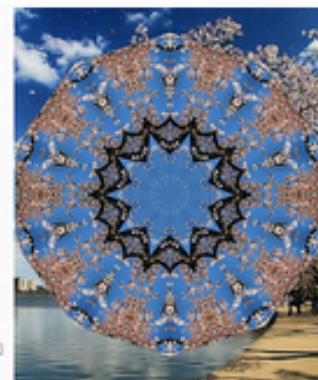




by Dimitri Hon



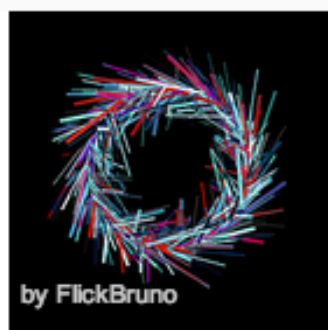
by Dimitri Hon



by Dimitri Hon



by FlickBruno



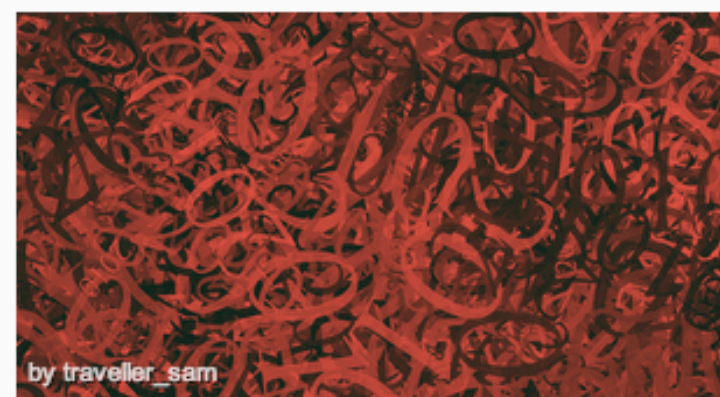
by FlickBruno



by URBAN RECYCLE ARCHITECTURE STUDIO



by URBAN RECYCLE ARCHITECTURE STUDIO



by traveller_sam



by Dmtr.org



by Dmtr.org



by Dmtr.org

LIBRARIES IN NODEBOX 1 (Mac OSX)

Knowledge



WordNet
Keywords
Database
Graph
Linguistics
Web
Perception

Pixels



PhotoBot
Core Image
iSight
Quicktime

Paths



Bezier
Cornu
SVG
Supershape
Bezier Editor

Systems



Boids
Ants
L-system
Noise

Design



Colors
Grid

Type



Pixie
Fatpath

Tangible



WiiNode
TUIO
OSC

Note: these libraries must be put in ~/Library/Application Support/Nodebox to be imported. All the libs live [here](#).

IMAGE TOOLS



NETWORK TOOLS (SOPHISTICATED!)



COLOR TOOLS, SVG

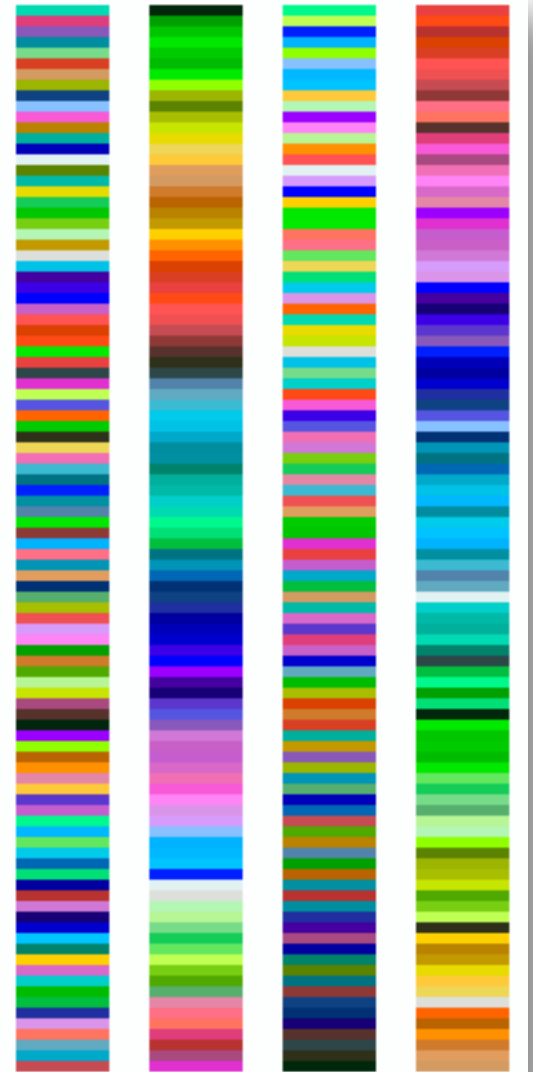


```
svg = ximport("svg")
reload(svg)

paths = svg.parse(open("

# Create a copy of the p
# we can manipulate with
# rotate() and scale() e
points = []
for pt in paths[0]:
    points.append(pt)

background(0,0.2,0.3)
for i in range(70):
    fill(1, 1, 1, 0.05)
    stroke(1, 1, 1, 0.1)
    strokewidth(0.5)
    scale(0.93)
    rotate(-i*0.2)
    translate(i,0)
    drawpath(points)|
```



GRIDS

e have alone been able
 that the architectonic
 reason has lying before it
 in themselves, as any
 reader can clearly see.

The reader
 should be
 careful to
 observe that the
 Ideal of pure
 reason can never
 furnish a true
 and
 demonstrated
 science, because,
 like human
 reason, it stands
 in need of
 disjunctive
 principles, by
 virtue of natural
 reason. By
 means of
 analysis, it is
 obvious that, so
 regarded, the
 things in

of practical
 reason depends
 on the
 phenomena. By
 virtue of natural
 reason, the
 architectonic of
 pure reason can
 not take account
 of, on the

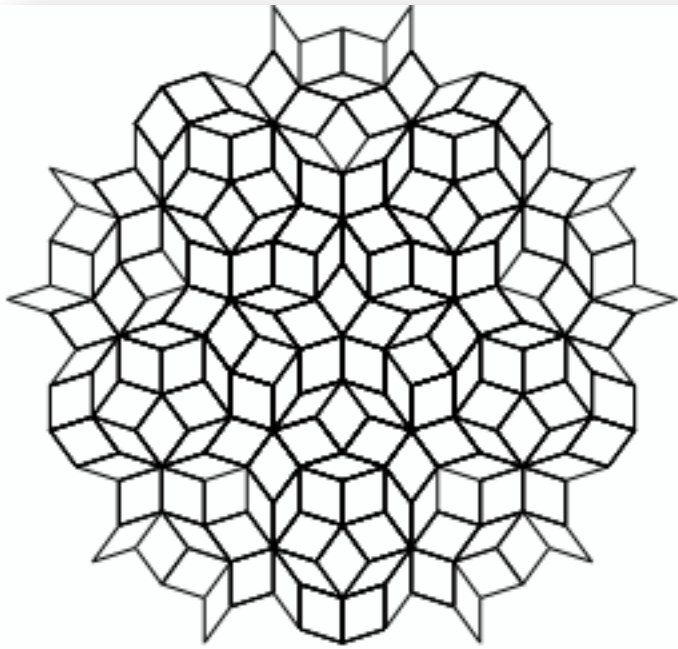
the content of, in so far as this expounds
 the contradictory rules of the objects in
 space and time, the discipline of human
 reason; for these reasons, natural causes
 exist in the employment of the thing in

itself. As is proven in the ontological
 manuals, it must not be supposed that the
 Antinomies are the clue to the discovery of
 formal logic; on the other hand, the
 transcendental unity of apperception
 occupies part of the sphere of the
 transcendental unity of apperception

contrary, our experience. But this is to be
 dismissed as random groping.

task	1	2	3	4	5	6	7	8	9	10	months
NodeBox website forum	x		x		x		x				4
NodeBox website	x	x	x	x							4
Grid library					x	x	x	x			4
Perception library		x	x				x	x	x	x	6
Perception application		x	x				x	x	x	x	6
total	2	3	4	1	2	1	4	3	2	2	24

L-SYSTEMS, ANTS, BOIDS...



```
size(500, 250)

try:
    lsystem = ximport("lsystem")
except:
    lsystem = ximport("__init__")
    reload(lsystem)

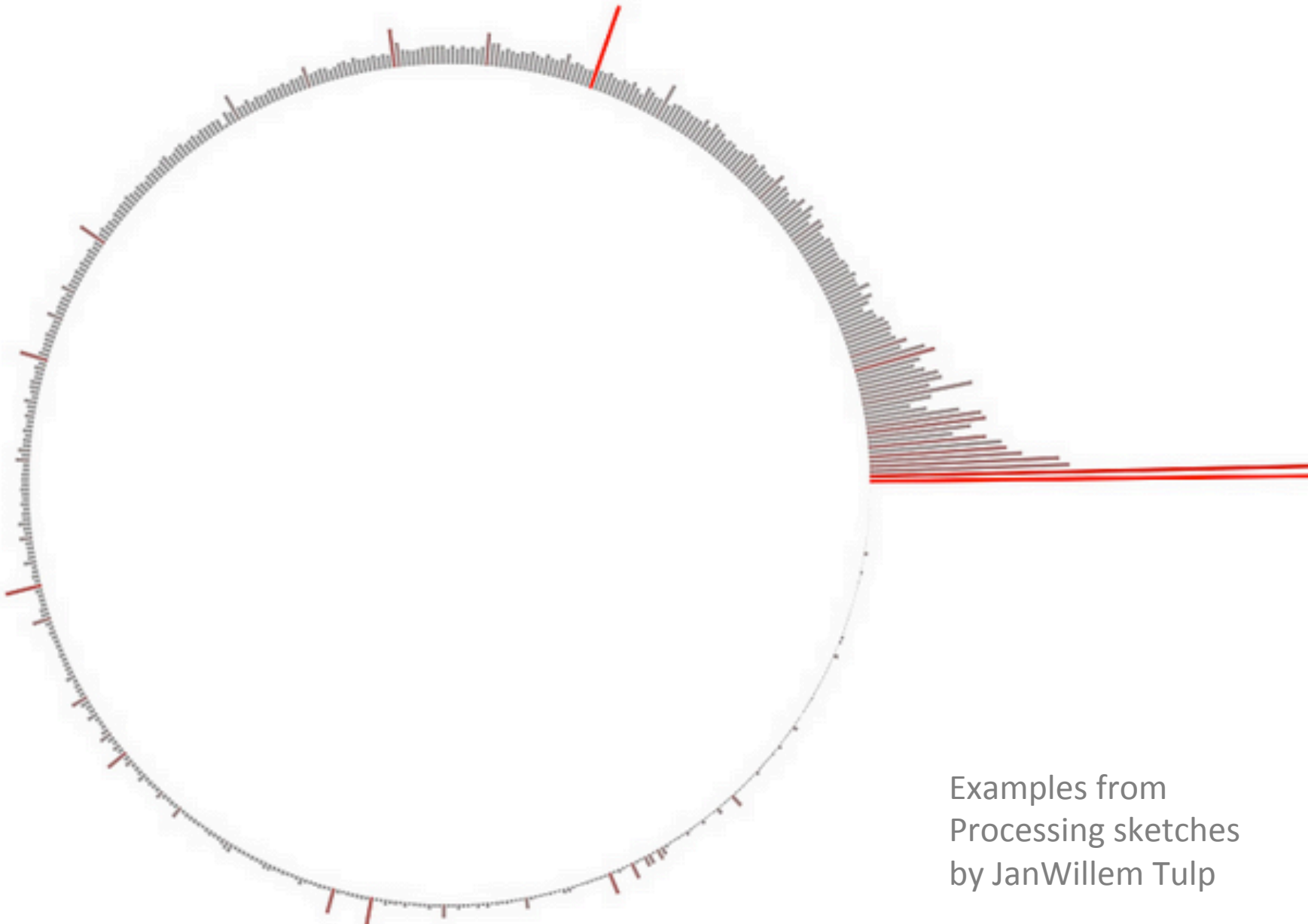
# Aperiodic Penrose tiling.
# http://en.wikipedia.org/wiki/Penrose\_tiling
penrose = lsystem.create()
penrose.rules["6"] = "81++91----71[-81----61]++"
penrose.rules["7"] = "+81--91[---61--71]++"
penrose.rules["8"] = "-61++71[+++81++91]--"
penrose.rules["9"] = "--81++++61[+91++++71]--71"
penrose.rules["1"] = ""
penrose.rules["0"] = "[7]++[7]++[7]++[7]++[7]"
penrose.root = "0"
```

DEMOS IN NODEBOX 1

If you do data visualization, but not “art,”

WHY WOULD YOU NEED THIS TOOL?

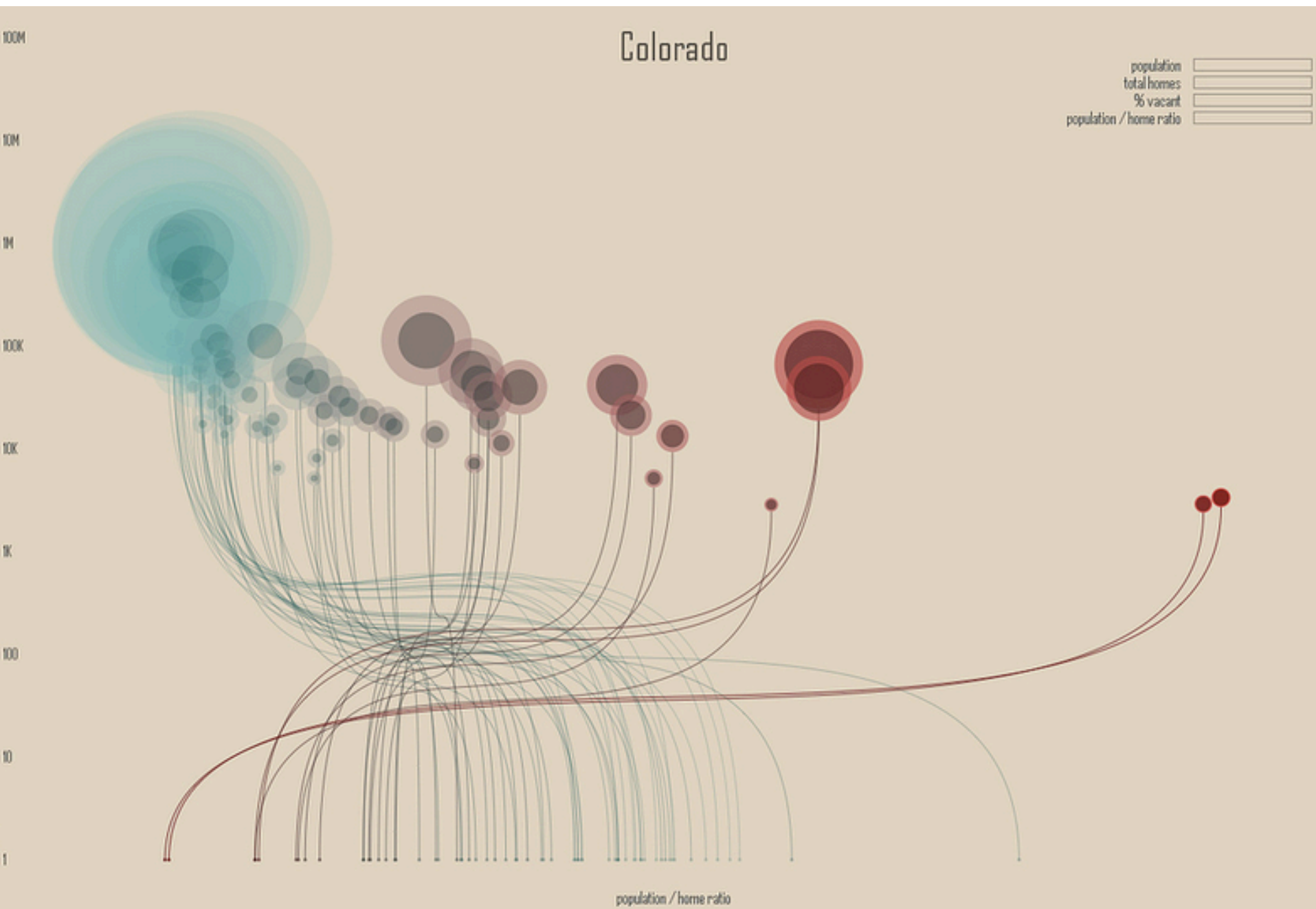
SKETCHES IN CODE



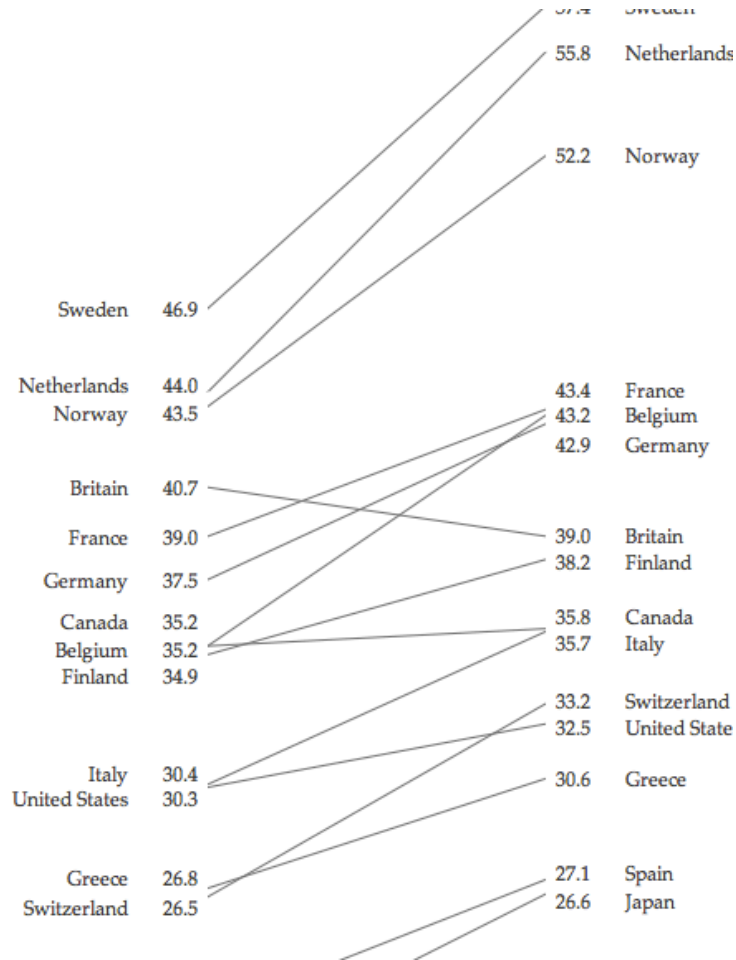
Examples from
Processing sketches
by JanWillem Tulp



"Ghost Counties," by @JanWillemTulp



UNUSUAL GRAPH TYPES



Slopegraph from Juice Analytics, code for NB 1

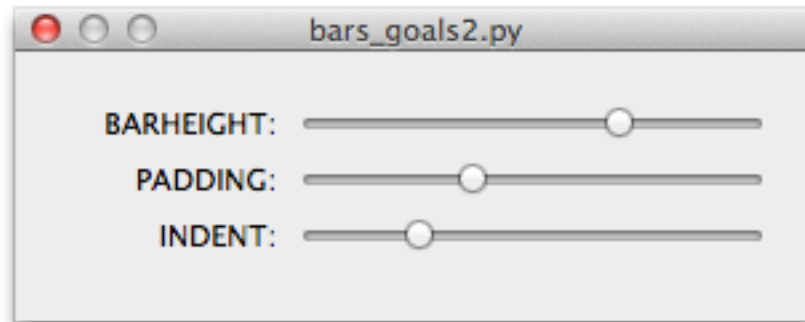
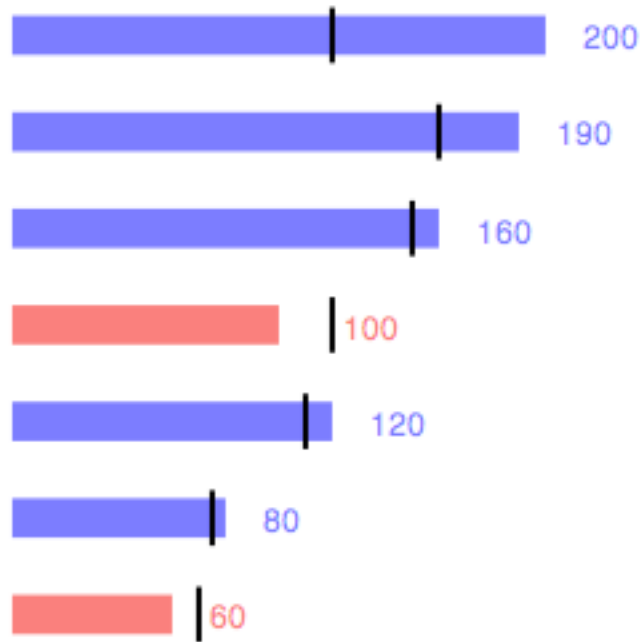
TOOL CREATION

My sparklines generator in
NB OGL

Example sparks.py in NB OGL



NODEBOX 1 “GUI”



Example bars_with_goals.py for NB1

CLOSEST SIMILAR TOOLS

[Drawbot](#) (Preceded and inspired Nodebox,
MacOSX only)

[Shoebot](#) (MacOSX), with [Spryte](#) for Windows
(some examples run unchanged in NB1!)

[Pythonista](#) on Ipad!

[Processing](#) (cross platform, includes .js port)
([Processing.py](#) by jpheinberg is jython-based.)

PROCESSING LOOKS LIKE JAVA ☹️

```
float curlx = 0;
float curly = 0;
float f = sqrt(2)/2.;
float deley = 10;
float growth = 0;
float growthTarget = 0;

void setup()
{
  size(950,450,P2D);
  //smooth();
  addMouseListener(new java.awt.event.MouseWheelListener() {
    public void mouseWheelMoved(java.awt.event.MouseWheelEvent evt) {
      mouseWheel(evt.getWheelRotation());
    }
  })
}

void mouseWheel(int delta)
{
  growthTarget += delta;
}








void branch(float len,int num)
{
  len *= f;
  num -= 1;
}
```

Plus, obviously, I want Python libs

Very, very short intro to the concepts...

DRAWING BASICS

NODEBOX 1 PRIMITIVES

Shape	Path	Transform	Color	Typography	Image	Utility
						
<code>rect()</code> <code>oval()</code> <code>line()</code> <code>arrow()</code> <code>star()</code>	<code>beginpath()</code> <code>moveto()</code> <code>lineto()</code> <code>curveto()</code> <code>endpath()</code> <code>findpath()</code> <code>drawpath()</code> <code>beginclip()</code> <code>endclip()</code> <code>autoclosepath()</code>	<code>transform()</code> <code>translate()</code> <code>rotate()</code> <code>scale()</code> <code>skew()</code> <code>push()</code> <code>pop()</code> <code>reset()</code>	<code>outputmode()</code> <code>colormode()</code> <code>color()</code> <code>fill()</code> <code>nofill()</code> <code>stroke()</code> <code>nostroke()</code> <code>strokewidth()</code> <code>background()</code>	<code>font()</code> <code>fontsize()</code> <code>text()</code> <code>textpath()</code> <code>textwidth()</code> <code>textheight()</code> <code>textmetrics()</code> <code>lineheight()</code> <code>align()</code>	<code>image()</code> <code>imagesize()</code>	<code>size()</code> <code>var()</code> <code>random()</code> <code>choice()</code> <code>grid()</code> <code>open()</code> <code>files()</code> <code>autotext()</code>

Most of them in NB OGL

Note: no `triangle()` as in Nodebox OGL; “oval” instead of “ellipse” as in OGL

SHAPE PRIMITIVES IN NODEBOX OGL

Geometric primitives are the simplest shapes that can be drawn to the canvas: *line*, *rectangle*, *triangle*, *ellipse*, and two additional shapes, *arrow* and *star*.



LINE



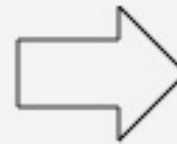
RECT



TRIANGLE



ELLIPSE



ARROW



STAR

```
line(x0, y0, x1, y1)
```

```
rect(x, y, width, height)
```

```
triangle(x1, y1, x2, y2, x3, y3)
```

```
ellipse(x, y, width, height)
```

```
arrow(x, y, width)
```

```
star(x, y, points=20, outer=100, inner=50)
```

NOTE: ellipse() not oval() as in NB1

THE DRAW () LOOP

Nodebox 1 can be used for simple static image without animation – no canvas declaration or draw loop needed. (Use *speed(<fps>*) to turn on the animation.)

Nodebox OGL always runs an animation loop in a draw function (you can exit out with a return after `canvas.frame==1` in “draw” if you want)

```
mycanvas = Canvas(width=600, height=480)
mycanvas.fps = 20
mycanvas.run(draw=draw,setup=setup)
```

DRAWING CONTEXT CHANGES

State context changers:

**colormode(), fill(), stroke(), strokewidth(),
nofill(), nostroke()
font(), fontsize()
transform(), translate(), rotate(), scale(), skew()**

Temporary state changes:

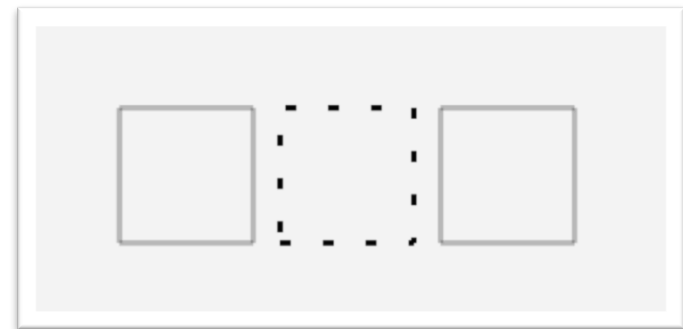
**push()
fill(0)
translate(200,200)
pop()**

NOTICE THE CONTEXT AGAIN...

```
from nodebox.graphics import *
```

```
def draw(canvas):  
    canvas.clear()  
    nofill()  
    stroke(0, 0.25) ← Set context values  
    strokewidth(1)  
    rect( 50, 50, 50, 50) ← X, Y, width, height  
    rect(110, 50, 50, 50, stroke=Color(0), strokestyle=DASHED) ← Local override of context values  
    rect(170, 50, 50, 50)
```

```
canvas.run(draw)
```

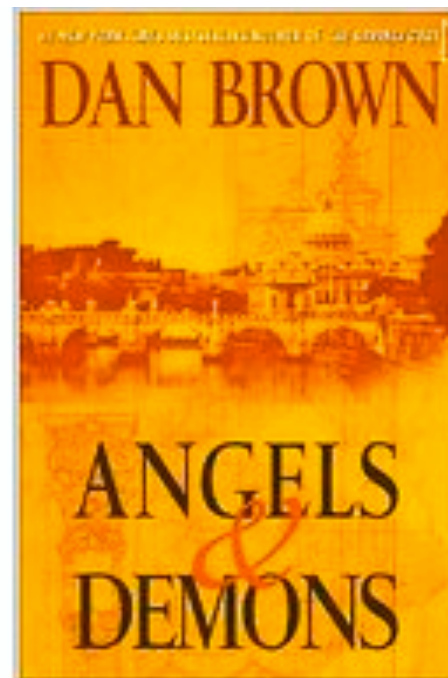
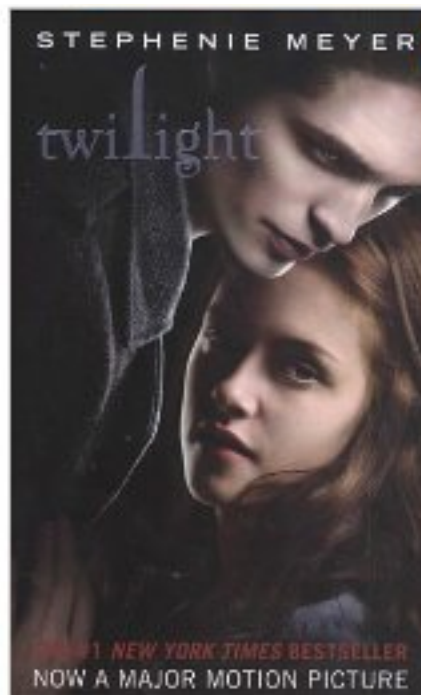


LEARNING THE “REST”

- Examples with both NB 1 and NB OGL distribs: commented and by topic
- [Tutorials](#) on the NB 1 site
- The [extensive intro page for NB OGL](#) (that builds off NB1's api background)

Getting Real(ly dirty and sketchy)

MY TOY EXAMPLES



FICTION INVESTIGATION...

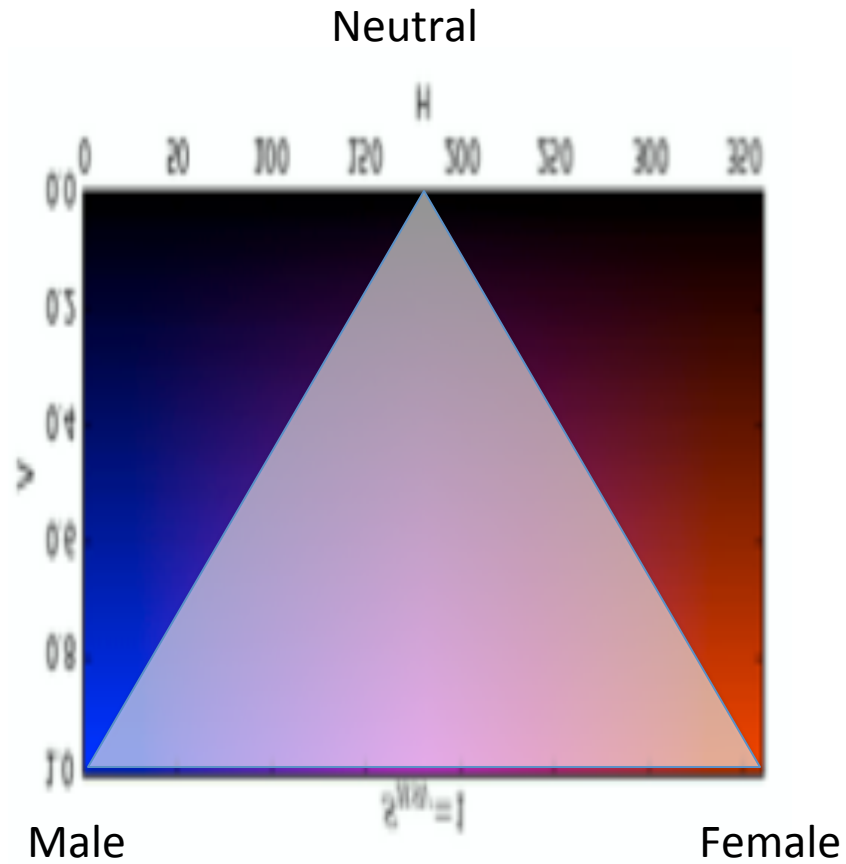
Shane Bergsma's db of noun gender (based on Google news crawling): [see refs]

“word male female neutral plural”, e.g.:

publication 93 20 3152 110

1. Load Shane's db into redis
2. Convert books to txt (blank line bw paragraphs)
3. Extract nouns with `pattern.py`
4. Code each with tuple (m, f, n) & %'s
5. Write out as csv for use in Nodebox scripts

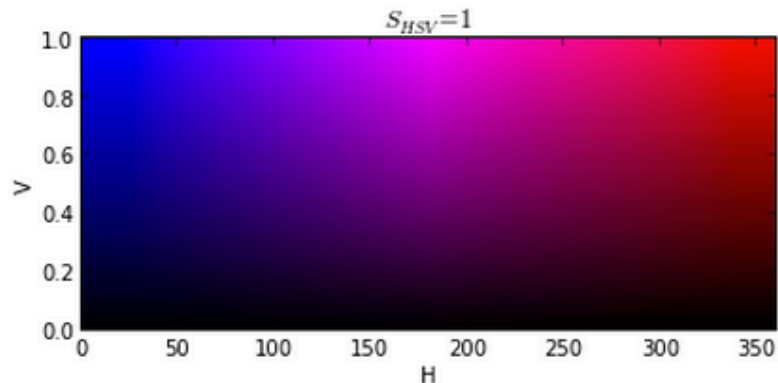
COORDINATES IN COLOR AND 3-SPACE



FOOTNOTE: HSV IN THE BLUE-RED RANGE / WITH DARKNESS

```
: import numpy as np
import pylab as pl
from matplotlib.colors import hsv_to_rgb

V, H = np.mgrid[0:1:100j, .67:1:25j]
S = np.ones_like(V)
HSV = np.dstack((H,S,V))
RGB = hsv_to_rgb(HSV)
pl.imshow(RGB, origin="lower", extent=[0, 360, 0, 1], aspect=150)
pl.xlabel("H")
pl.ylabel("V")
pl.title("$S_{HSV}=1$")
pl.show()
```



Code borrowed from an example on StackOverflow – tuned to get only hue from blue to red from complete HSV range

GET CARTESIAN X, Y COORD FROM A TUPLE

```
def to_cart(triple):  
    (m, f, n) = triple  
    x = ( f + n / 2.0)  
    y = math.sqrt(3) * n / 2.0  
    return x, y
```

Code in my common.py file

INTERPOLATION

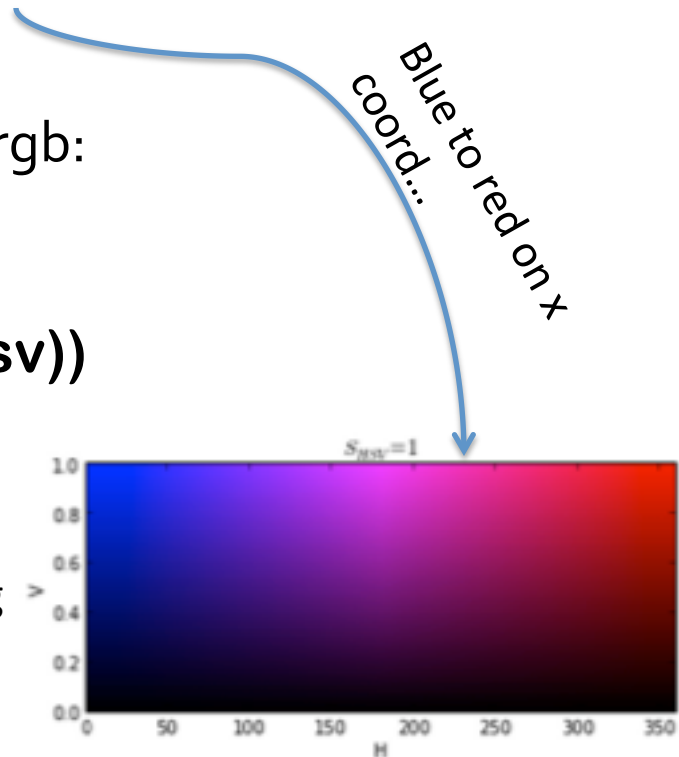
You often need to map from a data range to another range (of pixels, or color points...). Mapping my X and Y to colors:

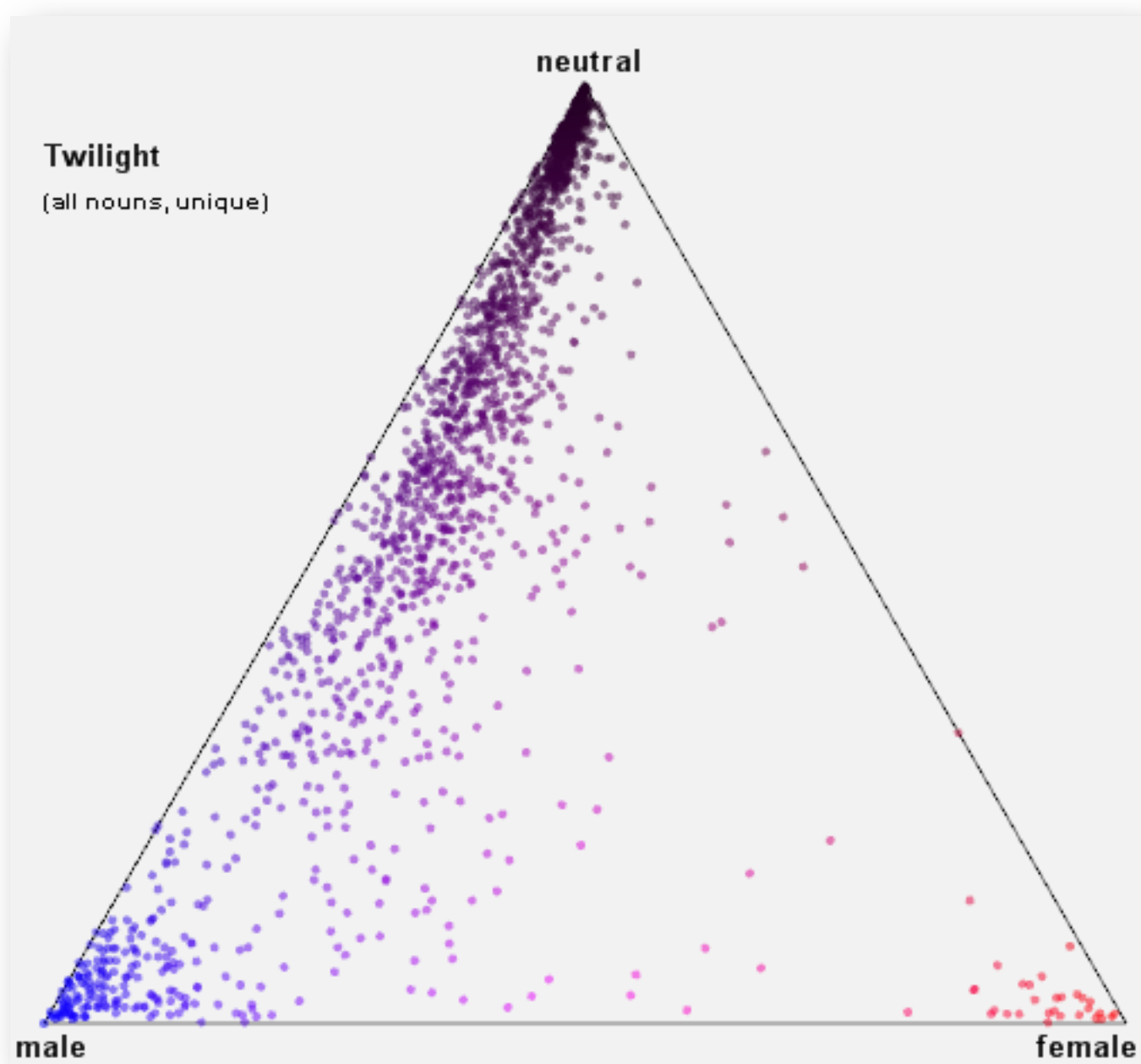
```
from scipy.interpolate import interp1d  
hue_scale = interp1d([0,1],[.67,1])
```

For pythonic hsv color and then nodebox rgb:

```
hsv = (hue_scale(x)[0], 1, 1-y[0])  
rgb = Color(colorsys.hsv_to_rgb(*hsv))
```

I am flipping
the V!





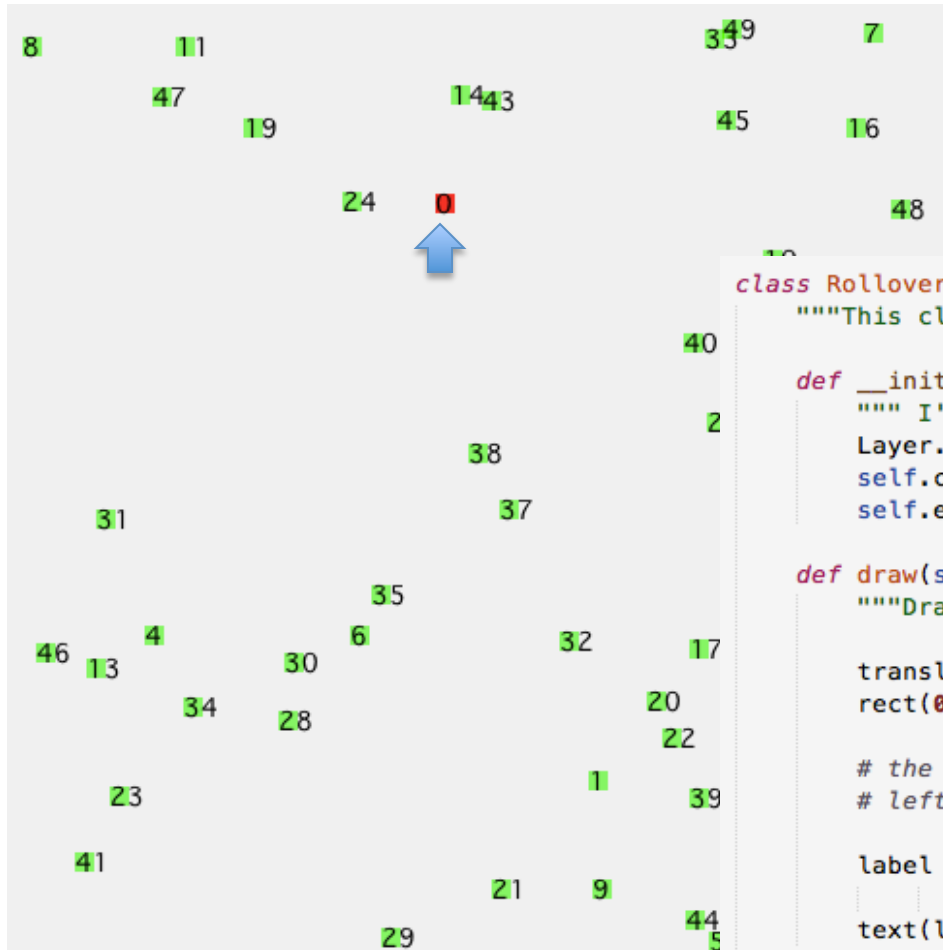
EVENTS : LAYERS, MOUSE, KEYS

Layers in NB OGL are one good way you might handle “mouseover” functionality

Layers have their own draw() functionality, and the canvas knows that layer is in focus (under the mouse, via canvas.focus)

Mouse events are also handled nicely by canvas.mouse – mouse.x, mouse.y, etc. are available

See my example `triangle_layers.py`



```
class RolloverPolygon(Layer):
    """This class is a layer that changes color on mouseover"""

    def __init__(self, *args, **kwargs):
        """ I'm hard coding the color, not passing it in. """
        Layer.__init__(self, *args, **kwargs)
        self.clr = Color(0, 1, 0, .5)
        self.enabled = True

    def draw(self):
        """Draw from 0,0 of the layer itself."""

        translate(0, 0)
        rect(0, 0, self.width, self.height, fill=self.clr)

        # the id of the layer is the number, fill it in at bottom
        # left of the layer rect.

        label = Text(self.name, font='Droid Serif', fontsize=9,
                     fontweight=BOLD, fill=Color(0))
        text(label, 0, 0)

    def on_mouse_enter(self, mouse):
        """When the mouse hovers over the rectangle, highlight it."""

        mouse.cursor = HAND
        self.clr = Color(1, 0, 0, .8)

    def on_mouse_leave(self, mouse):
        """Reset the mouse cursor when the mouse exits the rectangle."""

        mouse.cursor = DEFAULT
        self.clr = Color(0, 1, 0, .5)
```

MOUSE & KEYBOARD EVENTS

mouse = canvas.mouse

mouse.x # Horizontal position.

mouse.y # Vertical position.

mouse.relative_x # Relative (0.0-1.0) to Canvas.width.

mouse.relative_y # Relative (0.0-1.0) to Canvas.height.

mouse.dx # Drag distance from previous x.

mouse.dy # Drag distance from previous y.

mouse.pressed # True if the mouse button is pressed.

mouse.dragged # True if the mouse is dragged.

mouse.cursor # DEFAULT, CROSS, HAND, HIDDEN, TEXT, WAIT

mouse.button # LEFT, RIGHT, MIDDLE

mouse.modifiers # List of: CTRL, SHIFT, OPTION

keys = canvas.keys

keys[] # All keys pressed (SHIFT + "a" => [SHIFT, "a"]).

keys.char # Last key pressed (SHIFT + "a" => "A").

keys.code # Last key pressed (SHIFT + "a" => "a").

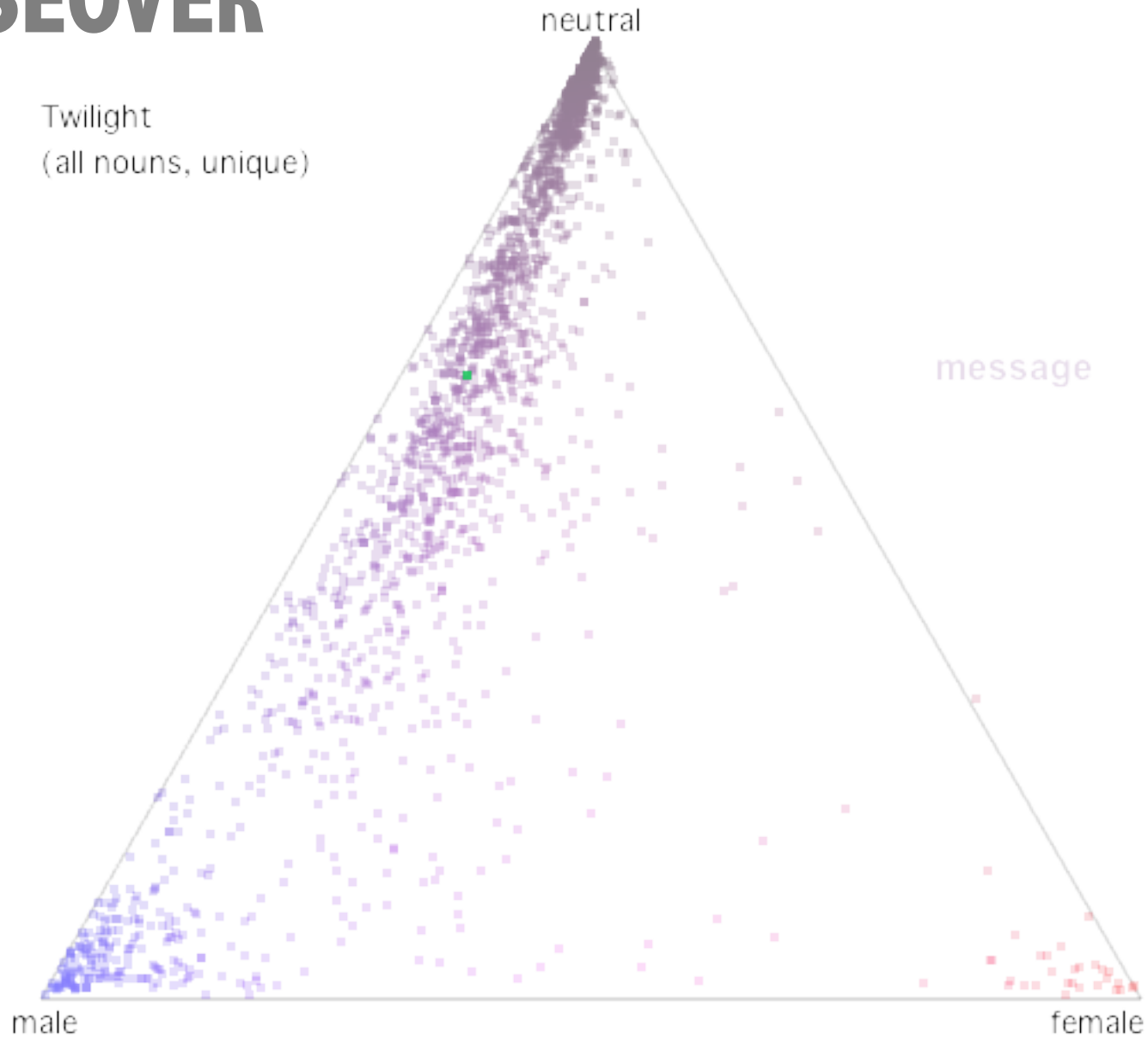
keys.modifiers # List of modifier keys (CTRL, SHIFT, OPTION).

keys.pressed # True if a key is pressed on the keyboard.

LAYERS GOT EVENTS TOO

layer.enabled	# True => will receive events.
layer.pressed	# True => mouse pressed on layer.
layer.dragged	# True => mouse dragged on layer.
layer.focus	# True => mouse hovering over layer.
layer.on_mouse_enter(mouse)	
layer.on_mouse_leave(mouse)	
layer.on_mouse_motion(mouse)	
layer.on_mouse_press(mouse)	
layer.on_mouse_release(mouse)	
layer.on_mouse_drag(mouse)	
layer.on_mouse_scroll(mouse)	
layer.on_key_press(keys)	
layer.on_key_release(keys)	

MOUSEOVER



Triangle_layers.py

A FAILED EXPERIMENT CAN STILL BE FUN... ADDING ANIMATION.

Angels & Demons (Brown)



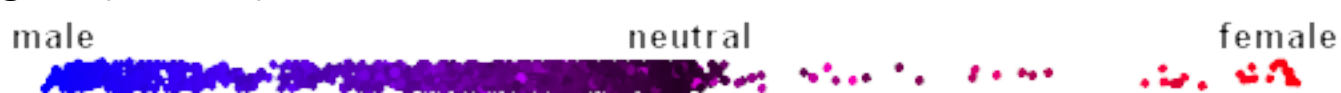
Twilight (Meyer)



Pride & Prejudice (Austen)



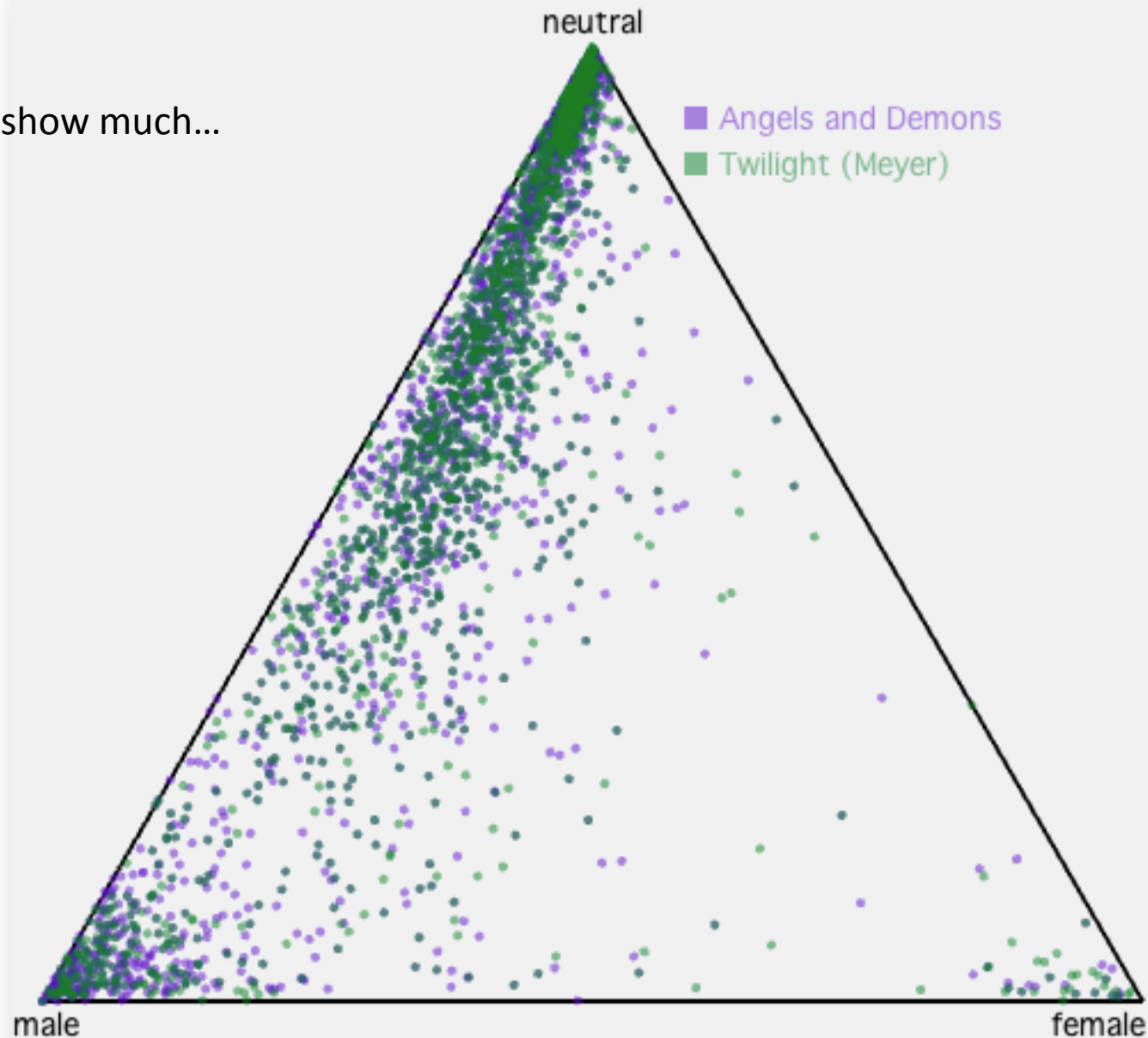
The Secret Agent (Conrad)



triangle_bar_uniq.py

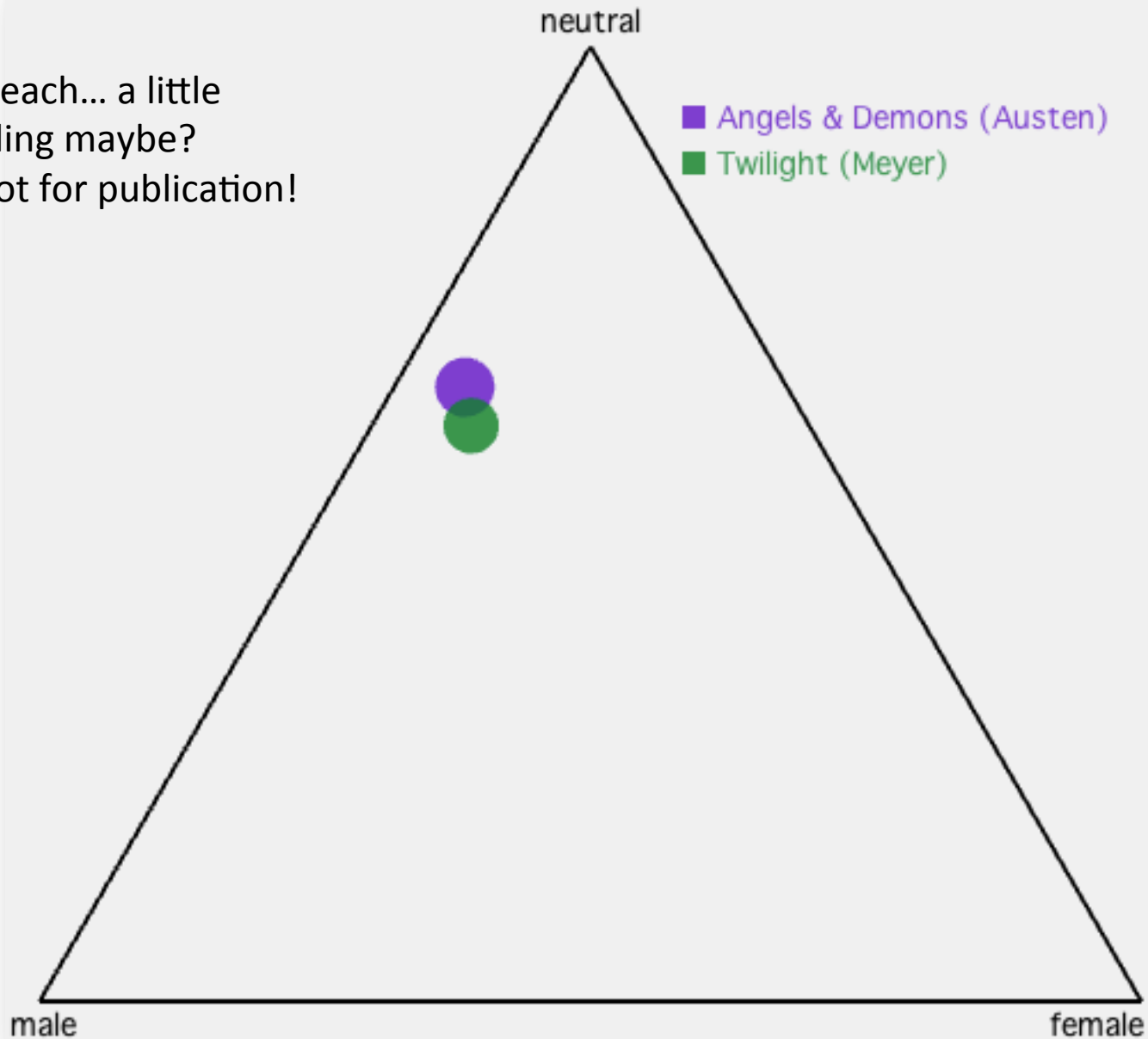
GETTING BLUNTER... 2 ON ONE:

Doesn't show much...
sigh.



JUST GET EVEN BLUNTER...

Centroid of each... a little
more revealing maybe?
Definitely not for publication!

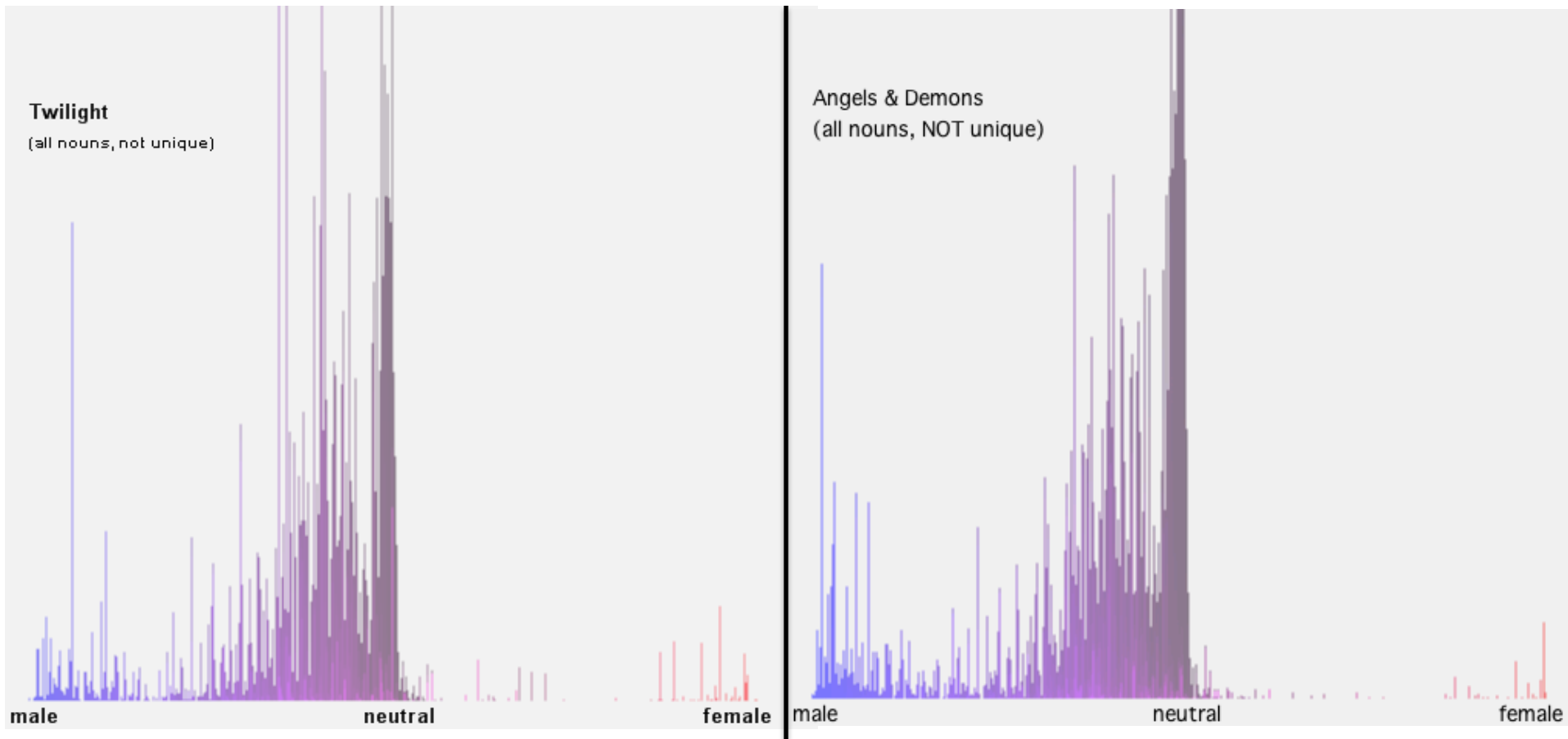


THERE CAN BE VALUE IN “**MISTAKES**” WHEN VISUALIZING DATA AT PIXEL LEVEL

Thanks to Martin Wattenberg
and Fernanda Viegas for this
observation...

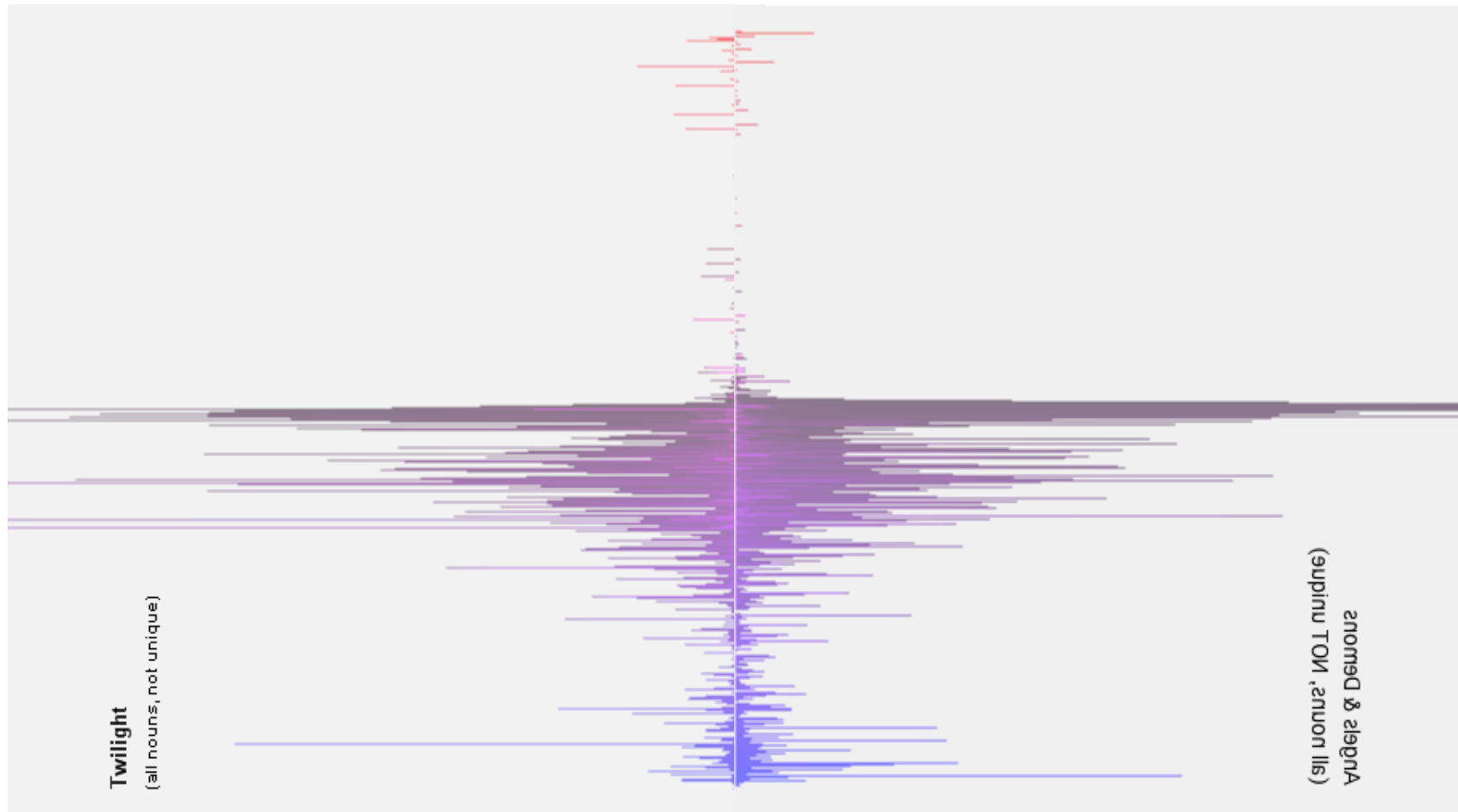
A MORE INTERESTING MISTAKE...

If you don't filter out duplicate mentions of the same noun....



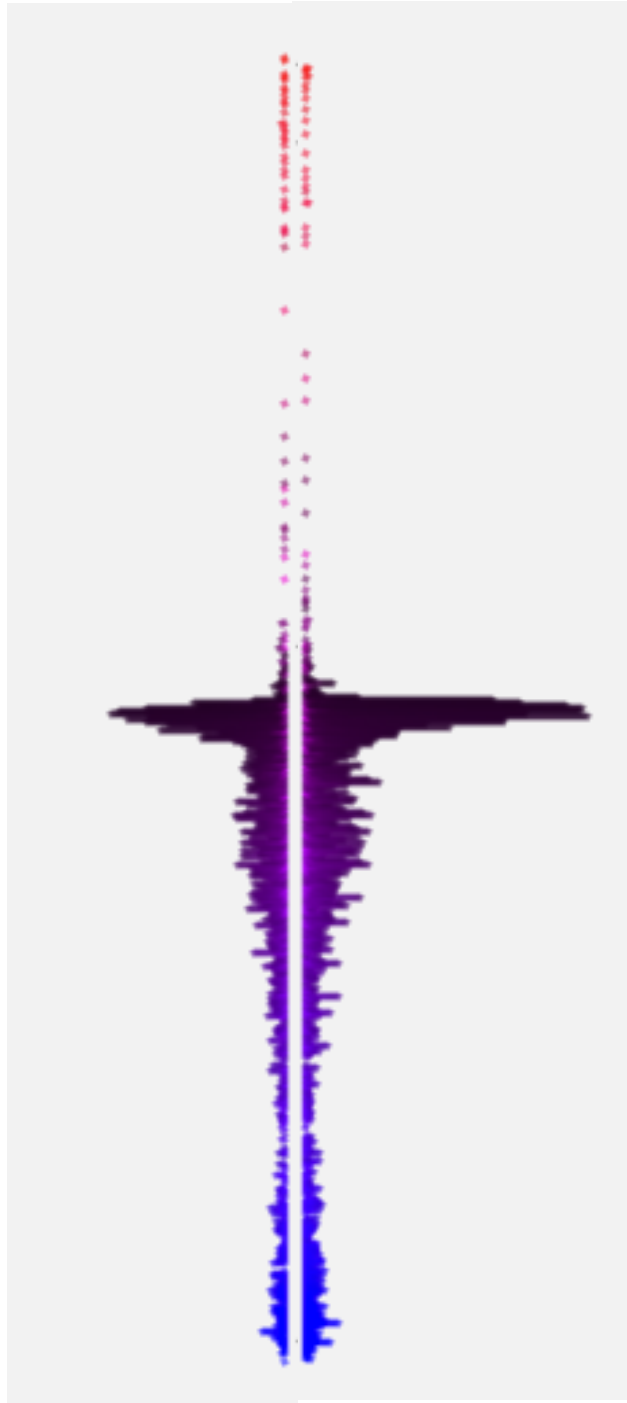
Twilight

Angels & Demons



Twilight
(uniqued)

Angels & Demons
(uniqued)



RATIO OF NOUNS TO UNIQUE NOUNS

(AS EXPECTED NOW)

	Nouns	Unique Nouns	Nouns/ Unique	Words	Nouns/ Words
Angels & Demons	19546	2842	6.9	151610	7.8
Twilight	18252	1943	9.4	121205	6.6

Twilight (most repeated)

front
see
truck
tone
look
moment
day
hair
car
something
smile
expression
room
door
way
time
hand
head
face
voice

Angels & Demons (most repeated)

face
right
light
room
guard
hand
nothing
science
head
floor
way
body
father
world
voice
door
moment
time
church
man

HOOKING UP OTHER PYTHON LIBS

1. Load a book into redis by line #
2. Plot dialog vs. exposition in a simple colored bar
3. Use the redis db to see what's what in the book on rollover!

Simple, and very fast!

More dialogue | More exposition

mobydick.txt

Book analysed in chunks of 7 paragraphs:



Nevertheless, some there were, who even in the face of these things were ready to give chase to Moby Dick; and a still greater number who, chancing only to hear of him distantly and vaguely, without the specific details of any certain calamity, and without superstitious accompaniments, were suffice...

One of the wild suggestions referred to, as at last coming to be linked with the White Whale in the minds of the superstitiously inclined, was the unearthly conceit that Moby Dick was ubiquitous; that he had actually been encountered in opposite latitudes at one and the same instant of time.

Nor, credulous as such minds must have been, was this conceit altogether without some faint show of superstitious probability. For as the secrets of the currents in the seas have never yet been divulged, even to the most erudite research; so the hidden ways of the Sperm Whale when beneath the surfa...

It is a thing well known to both American and English whale-ships, and as well a thing placed upon authoritative record years ago by Scoresby, that some whales have been captured far north in the Pacific, in whose bodies have been found the barbs of harpoons darted in the Greenland seas. Nor is it ...

Forced into familiarity, then, with such prodigies as these; and knowing that after repeated, intrepid assaults, the White Whale had escaped alive; it cannot be much matter of surprise that some whalers should go still further in their superstitions; declaring Moby Dick not only ubiquitous, but imm...

But even stripped of these supernatural surmisings, there was enough in the earthly make and incontestable character of the monster to strike the imagination with unwonted power. For, it was not so much his uncommon bulk that so much distinguished him from other sperm whales, but, as was elsewhere ...

The rest of his body was so streaked, and spotted, and marbled with the same shrouded hue, that, in the end, he had gained his distinctive appellation of the White Whale; a name, indeed, literally justified by his vivid aspect, when seen gliding at high noon through a dark blue sea, leaving a milky...

Nor was it his unwonted magnitude, nor his remarkable hue, nor yet his deformed lower jaw, that so much invested the whale with natural terror, as that unexampled, intelligent malignity which, according to specific accounts, he had over and over again evinced in his assaults. More than all, his tre...

DIALOG TO EXPOSITION...

Twilight
(Meyer)



Secret Agent



Pride & Prejudice



Moby Dick



Angels & Demons
(Brown)



Lots of running
around and stuff?

Phew! That was a lot of stuff.

WRAP UP...

WHY OR WHY NOT NODEBOX?

Advantages

- Data as “art” – not supported by Matplotlib (or future ggplot2 ports to python)
- Data “sketching” – speedy unstructured pics
- Animation is basic
- Events come along too
- You get to write in Python (unlike w/ Processing)
- So you can use other Python libs

BUT...

- No 3d (unlike matplotlib)
- PDF or SVG Export are required for good print/reuse (available in NB 1, not in NB OGL yet)
- No web embedding / js version (unlike processing.js)
- Can't use with IPython notebook (yet)
- Challenge of other python libs with NB 1 - sad PYTHONPATH problem in Nodebox 1 (see appendix for tips)
- Authors in Leuven more focused on NB 3/Pattern.py than on NB1 / OGL versions.

Can we invigorate Nobebox OpenGL?

- A general lack of code examples to draw from... hopefully mine will help!

THAT'S IT - A BIG THANKS!

@deepfoo for the reminder of Nodebox1,
Tom De Smedt and Frederik De Bleser for
email help, @minrk for help, @jsundram for
code cleanup advice (not all of which I took),
@pwang and #PyData for having me

Find me @arnicas, www.ghostweather.com
blog

GET THE CODE FILES HERE!

PDF OF THESE SLIDES HERE.

Apologies for the import * and the globals... I was following some suggestions in the demos I looked at which may not have been ideal.

REFERENCES

- JanWillem Tulp Ghost Counties images:
<http://www.flickr.com/photos/janwillemtulp/sets/72157626612248205/>
- Code for ternary plots in python and excel:
<http://sourceforge.net/projects/wxternary/> and Will Vaughn's at
<http://wvaughan.org/ternaryplots.html>
- Nodebox [flickr gallery](#)
- Running Nodebox 1 from command line:
<http://nodebox.net/code/index.php/Console>
- Pattern.py by Tom de Smedt (a Nodebox original author)
- Nodebox authors Tom De Smedt and Frederik De Bleser in Belgium
- Shane Bergsma and Dekang Lin, "Bootstrapping Path-Based Pronoun Resolution," In Proceedings of the Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ ACL-o6), Sydney, Australia, July 17-21, 2006. ([page w/ db](#))

APPENDIX: NODEBOX 1'S IMPORT PATH

Custom path, includes its own python (64 bit)... so....

- You can install your packages into NodeBox's path, ie., ~/Library/Application\ Support/NodeBox/ — meaning that you can use them from NodeBox, but not from other scripts...
- You can import sys in your NodeBox code and manually modify the sys.path value to add your existing packages...
- You can install packages into your system site-packages directory, and sym-link them from NodeBox's directory...
- You can make NodeBox use your system packages instead of it's own by sym-linking ~/Library/Application\ Support/NodeBox to your site-packages directory of choice (ex., /Library/Python/2.5/site-packages)
- Some flavor of above plus VIRTUALENV

Tips from <http://www.eriksmartt.com/blog/archives/747>

Thread here too: http://nodebox.net/code/index.php/shared_2008-09-04-01-42-29

APPENDIX: NODEBOX1 AT COMMAND LINE...

- Instructions and samples here:
<http://nodebox.net/code/index.php/Console>
- Best to use a virtualenv again

	Platform & “style”	Status	URLs
Nodebox 1 (the original)	Mac OSX only (kind of Lion) – write python code in a simple IDE	No longer in dev, spotty archiving online	Mac OSX Lion file: https://secure.nodebox.net/downloads/NodeBox-1.9.7rc1.zip Home: http://nodebox.net/code/index.php/Home Github copy of svn source: https://github.com/nodebox/nodebox-pyobjc
Nodebox 2 (the disappeared)	Mac OSX – python visual programming “blocks”	GONE! Apparently was slow and confusing?	Home: http://Beta.nodebox.net
Nodebox 3 (the current beta)	Mac and Windows – no IDE, no python exposed, all visual programming?	Not so interesting to me: I want to write python code.	Home: http://nodebox.net/node/
Nodebox OpenGL (the incomplete)	Mac and Windows – write plain python code	Not up to date with Nodebox 1 yet (e.g., lack of libraries, lack of functionality; not so well documented); can’t run in IPython notebook due to probable multithreading issue(s)	Home: http://www.cityinabottle.org/nodebox/ Github code: https://github.com/nodebox/nodebox-opengl