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

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
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)

canvas.run(draw)
NodeBox 1 is a smart IDE

Implicit canvas, draw, etc.
## Download NodeBox

<table>
<thead>
<tr>
<th></th>
<th>Mac OS X</th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NodeBox 3</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 3.0.32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Notes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Download]</td>
<td>[Download]</td>
<td>[Instructions]</td>
</tr>
<tr>
<td><strong>NodeBox OpenGL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 1.7</td>
<td></td>
<td>[Mac too, I’ll demo]</td>
<td>[Tentative evidence of linux too]</td>
</tr>
<tr>
<td></td>
<td>[Download]</td>
<td>[Download]</td>
<td></td>
</tr>
<tr>
<td><strong>NodeBox 1</strong></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Version 1.9.7rc1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[http://nodebox.net/download/](http://nodebox.net/download/)
GENERATIVE ART

Justin D on flickr.
HIGH QUALITY GRAPHIC OUTPUT
## Libraries in NodeBox 1 (Mac OSX)

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>Pixels</th>
<th>Paths</th>
<th>Systems</th>
<th>Design</th>
<th>Type</th>
<th>Tangible</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>PhotoBot</td>
<td>Bezier</td>
<td>Boids</td>
<td>Colors</td>
<td>Pixie</td>
<td>WiiNode</td>
</tr>
<tr>
<td>Keywords</td>
<td>Core Image</td>
<td>Cornu</td>
<td>Ants</td>
<td>Grid</td>
<td>Fatpath</td>
<td>TUIO</td>
</tr>
<tr>
<td>Database</td>
<td>iSight</td>
<td>SVG</td>
<td>L-system</td>
<td></td>
<td></td>
<td>OSC</td>
</tr>
<tr>
<td>Graph</td>
<td>Quicktime</td>
<td>Supershape</td>
<td>Noise</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linguistics</td>
<td></td>
<td>Bezier Editor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perception</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

paths = svg.parse(open("example.svg"))

# Create a copy of the paths
# we can manipulate with
# rotate() and scale() etc.
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)
```
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

<table>
<thead>
<tr>
<th>task</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>months</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeBox website forum</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>4</td>
</tr>
<tr>
<td>NodeBox website</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>4</td>
</tr>
<tr>
<td>Grid library</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Perception library</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>6</td>
</tr>
<tr>
<td>Perception application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>total</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
</tbody>
</table>

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.
L-Systems, Ants, Boids...

```python
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]++[7]"
penrose.root = "0"
```
DEMONS 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
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
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 😞

```java
float curlx = 0;
float curly = 0;
float f = sqrt(2)/2.;
float deley = 10;
float growth = 0;
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
{
background(255);
stroke(0);
curlx += (radians(360./height*mouseX)-curlx)/deley;
curly += (radians(360./height*mouseY)-curly)/deley;
translate(width/2,height/3*2);
line(0,0,0,height/2);
bran(x(height/4.,17);
growth += (growthTarget/10-growth+1.)/deley;
}
void mouseWheel(int delta)
{
growthTarget += delta;
}
void branch(float len,int num)
{
  len *= f;
  num -= 1;
}
```

Plus, obviously, I want Python libs

http://openprocessing.org/sketch/8941
Very, very short intro to the concepts...

DRAWING BASICS
**Nodebox 1 Primitives**

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

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.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>line(x0, y0, x1, y1)</td>
</tr>
<tr>
<td>RECT</td>
<td>rect(x, y, width, height)</td>
</tr>
<tr>
<td>TRIANGLE</td>
<td>triangle(x1, y1, x2, y2, x3, y3)</td>
</tr>
<tr>
<td>ELLIPSE</td>
<td>ellipse(x, y, width, height)</td>
</tr>
<tr>
<td>ARROW</td>
<td>arrow(x, y, width)</td>
</tr>
<tr>
<td>STAR</td>
<td>star(x, y, points=20, outer=100, inner=50)</td>
</tr>
</tbody>
</table>

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)

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

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**

```python
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("$S_{\text{HSV}}=1$")
pl.show()
```

*Code borrowed from an example on StackOverflow – tuned to get only hue from blue to red from complete HSV range*
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
You often need to map from a data range to another range (of pixels, or color points...). Mapping my X and Y to colors:

```python
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!
Twilight
(all nouns, unique)
**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, 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, 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 (0-1.0) to Canvas.width.
mouse.relative_y  # Relative (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)
Twilight
(all nouns, unique)
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
(all nouns, not unique)

Angels & Demons
(all nouns, NOT unique)
Twilight

Angels & Demons
Twilight
(uniqed)

Angels & Demons
(uniqed)
### Ratio of Nouns to Unique Nouns (As expected now)

<table>
<thead>
<tr>
<th></th>
<th>Nouns</th>
<th>Unique Nouns</th>
<th>Nouns/Unique</th>
<th>Words</th>
<th>Nouns/Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angels &amp; Demons</td>
<td>19546</td>
<td>2842</td>
<td>6.9</td>
<td>151610</td>
<td>7.8</td>
</tr>
<tr>
<td>Twilight</td>
<td>18252</td>
<td>1943</td>
<td>9.4</td>
<td>121205</td>
<td>6.6</td>
</tr>
</tbody>
</table>
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

Line_freqs.py
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!
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 sufficient.

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 surf.

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 whalermen 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)

Quote_bar_nodb.py

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
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

- Nodebox [flickr gallery](http://flickr.com)
- Running Nodebox 1 from command line: [http://nodebox.net/code/index.php/Console](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-06), 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, i.e., ~/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

Appendix: Nodebox1 at command line...

- Instructions and samples here: http://nodebox.net/code/index.php/Console
- Best to use a virtualenv again
<table>
<thead>
<tr>
<th><strong>Nodebox 1 (the original)</strong></th>
<th>Platform &amp; “style”</th>
<th>Status</th>
<th>URLs</th>
</tr>
</thead>
</table>
|  | 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](https://secure.nodebox.net/downloads/NodeBox-1.9.7rc1.zip)  
Github copy of svn source: [https://github.com/nodebox/nodebox-pyobjc](https://github.com/nodebox/nodebox-pyobjc) |

<table>
<thead>
<tr>
<th><strong>Nodebox 2 (the disappeared)</strong></th>
<th>Platform &amp; “style”</th>
<th>Status</th>
<th>URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac OSX – python visual programming “blocks”</td>
<td>GONE! Apparently was slow and confusing?</td>
<td>Home: <a href="http://Beta.nodebox.net">http://Beta.nodebox.net</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Nodebox 3 (the current beta)</strong></th>
<th>Platform &amp; “style”</th>
<th>Status</th>
<th>URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac and Windows – no IDE, no python exposed, all visual programming?</td>
<td>Not so interesting to me: I want to write python code.</td>
<td>Home: <a href="http://nodebox.net/node/">http://nodebox.net/node/</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Nodebox OpenGL (the incomplete)</strong></th>
<th>Platform &amp; “style”</th>
<th>Status</th>
<th>URLs</th>
</tr>
</thead>
</table>
|  | 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/](http://www.cityinabottle.org/nodebox/)  
Github code: [https://github.com/nodebox/nodebox-opengl](https://github.com/nodebox/nodebox-opengl) |