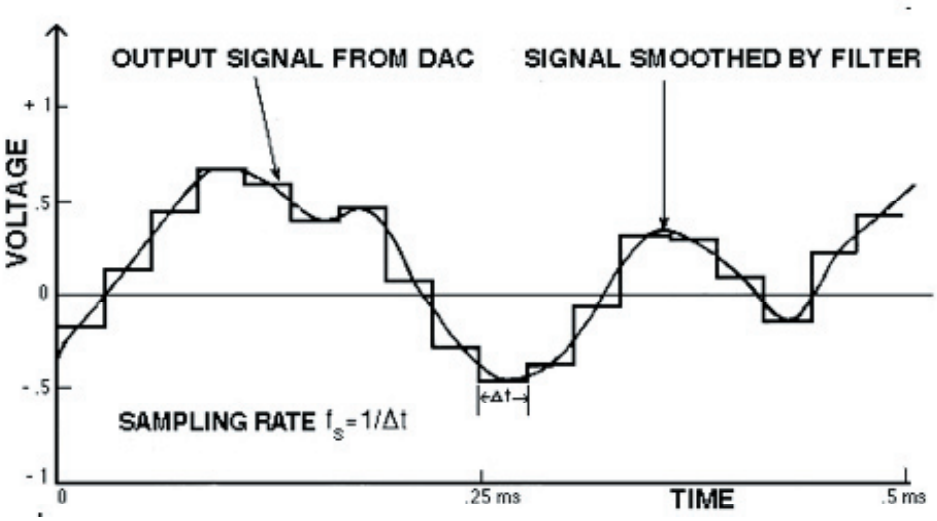
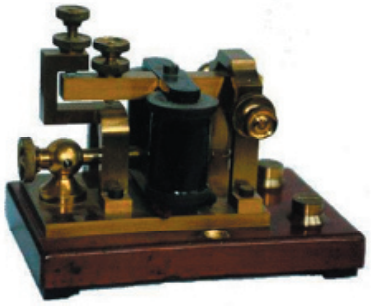
Representation of Information

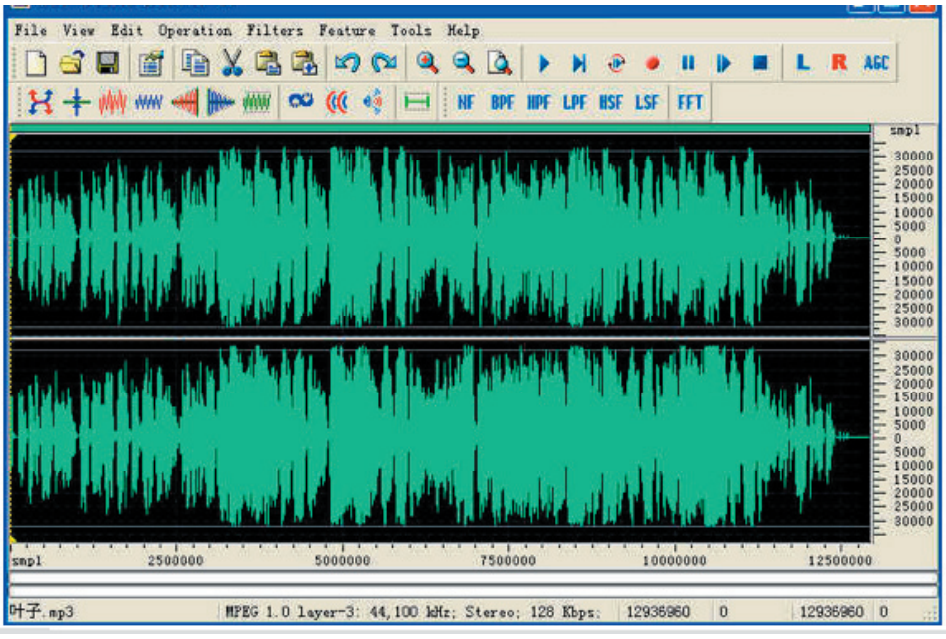
Jennifer Boriss
Fall 2006

How do you represent sound?

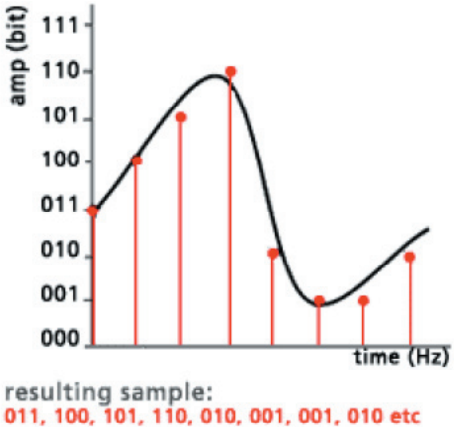


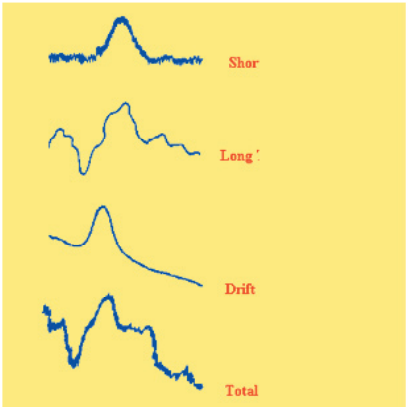
A: ·–	B: –···	C: –·–·
D: –··	E: ·	F: ··–·
G: ––·	H: ····	I: ··
J: ·–––	K: –·–	L: ·–··
M: ––	N: –·	O: –––
P: ·––·	Q: ––·–	R: ·–·
S: ···	T: –	U: ··–
V: ···–	W: ·––	X: –··–
Y: –·––	Z: ––··	

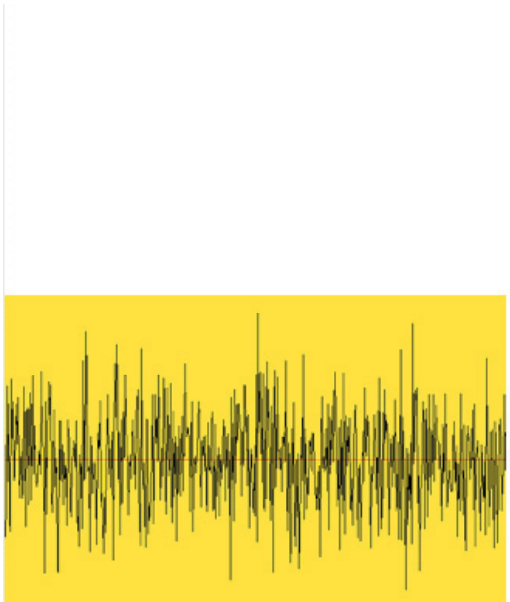


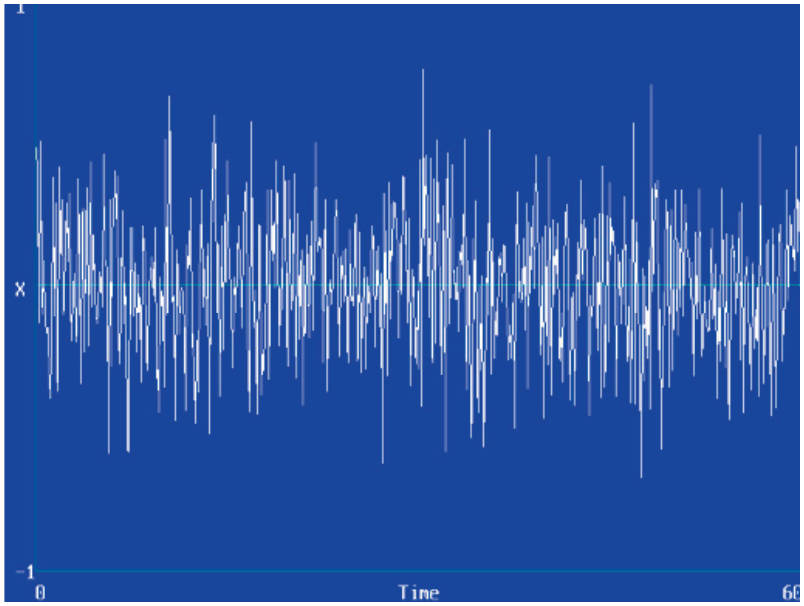


Before a continuous, time-varying signal such as **sound** can be manipulated or analyzed with a digital computer, the signal must be *acquired* or *digitized* by an *analog-to-digital (A/D) converter*.[1]
The A/D converter repeatedly measures or samples the instantaneous voltage amplitude of an input signal at a particular sampling rate, typically thousands or tens of thousands of times per second (Figure A.1). The digital **representation** of a signal created by the converter thus consists of a sequence of numeric values representing the amplitude of the original waveform at discrete, evenly spaced points in time.









Representation of Information - from BSA
Mondays, 9am, A14 block in Harrow

*check blackboard

From the 2nd-10th, introductory projectory - will last until the 16th or the 10th, with presentation.

Corrections from book -

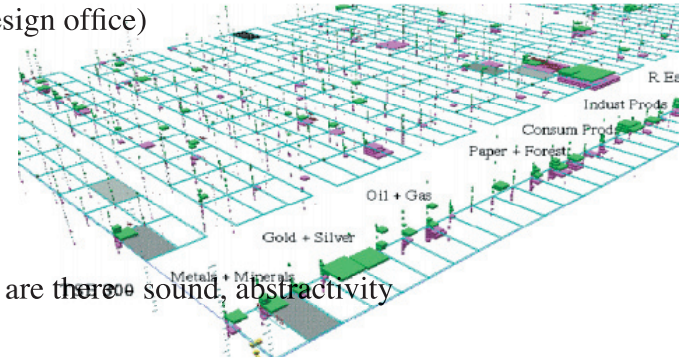
Group presentations of ideas are 30/10
Project idea agreement - 06/11
o8/

Three weeks of initial project. Then a performance report.

Reading list - recommends “Cause and Effect” by Meadows
Edward Tuffty - Envisioning Information - (What do we mean by representation, graphical representation)
Morgan - “See what I Mean”

Get some sort of visual notebook - A3?
Show how you developed in book

Prof is available on Tuesdays from 11am-1:30pm in J2.7 (design office)

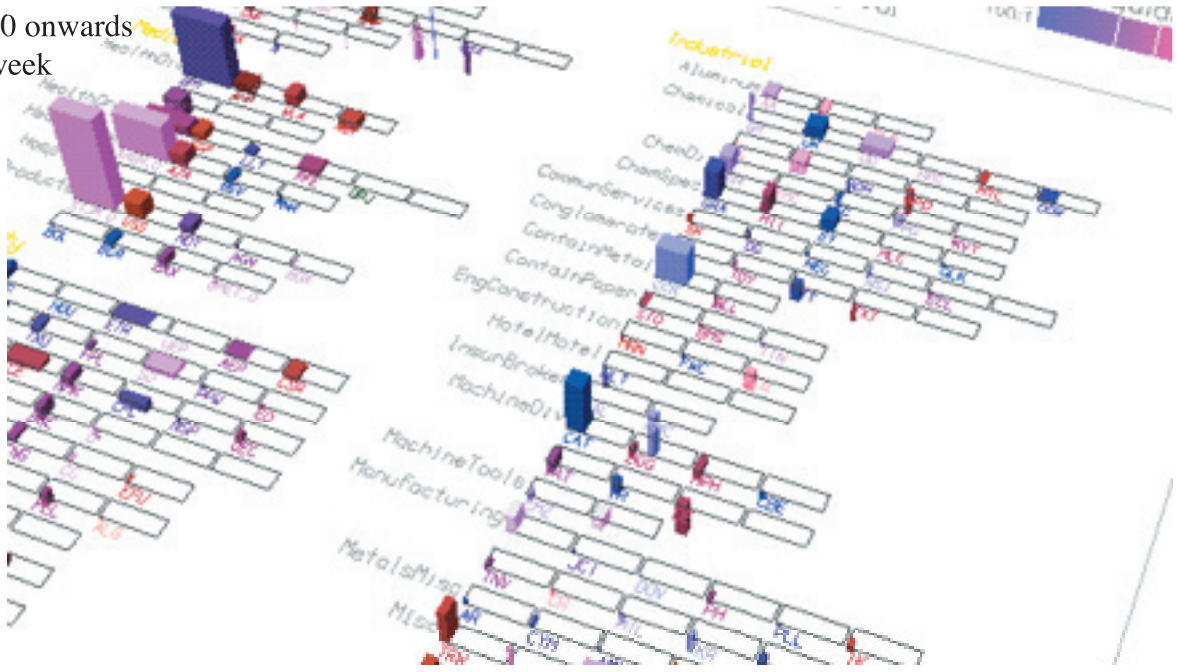


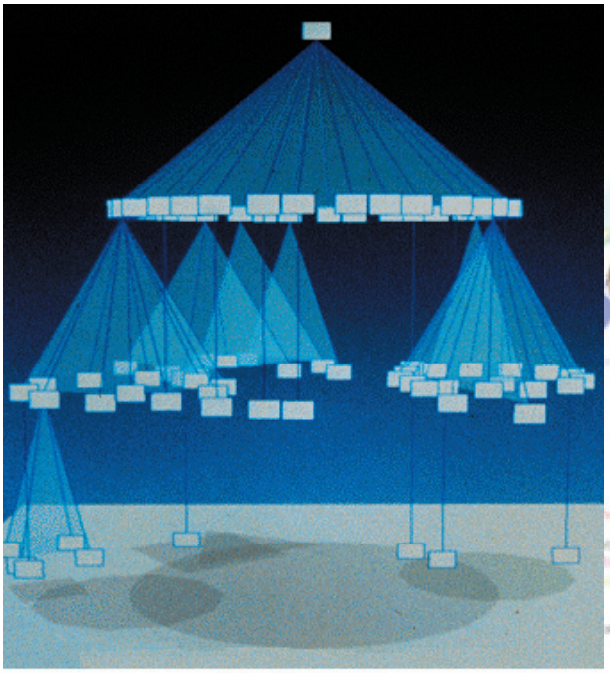
personal email: jkiayani@yahoo.co.uk
university email: kiayanj@winmin.ac.uk
mobile: 07971786940

for next week - interrogate a couple people in the module
think about: what kind of information representation system are there? sound, abstractivity

9-1 this module runs
11-1 (am I available?)
prof will be here at 9:30 onwards
come in at 1030 next week

8/2/06





A 3D Cone Tree

)ata represented in this form provides faster navigation ınd allows more information to be seen at one time. The Cone Tree shows the structure of an entire hierarchy all at ınce, and the cones spin around to retrieve occluded data. *Image courtesy of Palo Alto Research Center; Brian Tramontana, photographer.)*

sound information -

http://brainop.media.mit.edu/Archive/Metois/Thesis0.html

MUSICAL SOUND INFORMATION
Musical Gesture and Embedding Synthesis
Abstract
Chapter 1

* Introduction
    o Quick History
    o Musical Sound Representation: The Issues
        + Musical Intentions versus Musical Gestures
            # Musical Intentions
            # Musical Gestures
        + Machine Listening versus Instrument Modeling
            # Machine Listening
            # Instrument Modeling
    o Motivations and Overview
        + Music and Media - Motivations
        + Overview

Chapter 2

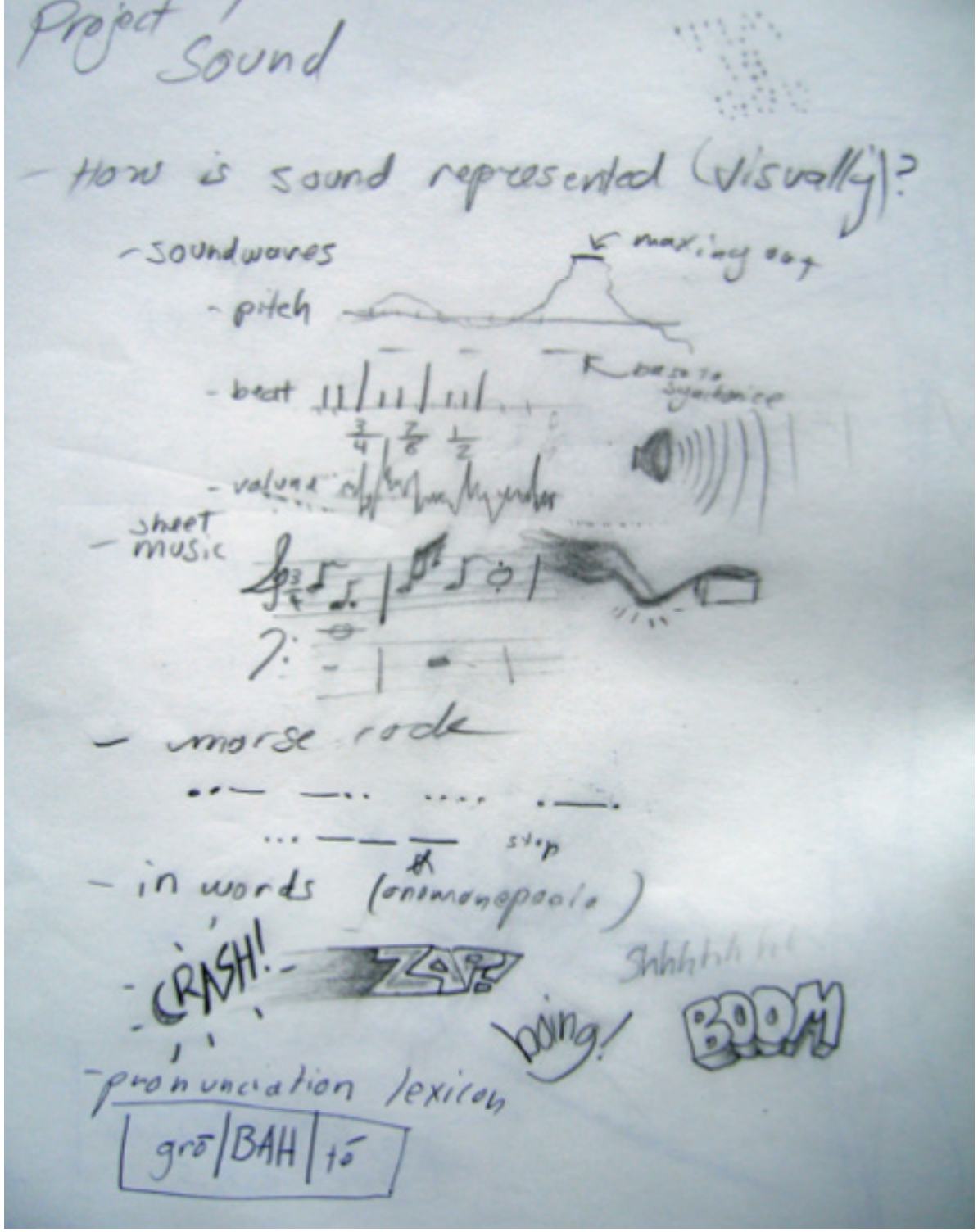
* Background
    o Timbre and Musical Expression
        + Timbre-based composition
            # Some examples
            # A lack of structure for timbre
        + Suggested structures for timbre
            # The “top-down” school
            # The “bottom-up” approach
    o Linear System Theory: Notions and Assumptions
        + Assumptions behind spectral analysis
        + Deterministic / non-deterministic processes
        + Wold’s decomposition
    o Entropy, Information and Redundancy
        + Entropy
        + Mutual information and redundancy
        + The case of a sampled strict sense stationary stochastic process
    o Nonlinear Dynamics and the Embedding Theorem
        + Dynamical Systems
        + The Embedding Theorem

Chapter 3

* Machine Listening - Real-time Analysis
    o Perceptual Components of a Musical Sound

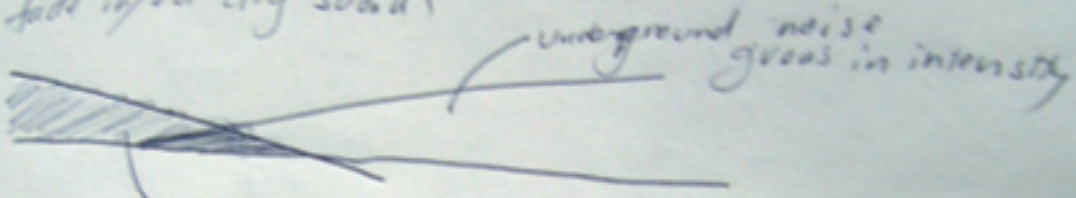
Project 1 — Sound

- How is sound represented (visually)?



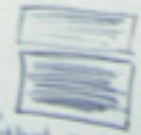
Sound design

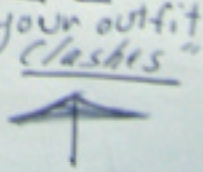
fade in/out city sounds



noise – people also describe visual images as having noise

"jarring"
"soft"





words can describe sound and image – how did they begin?

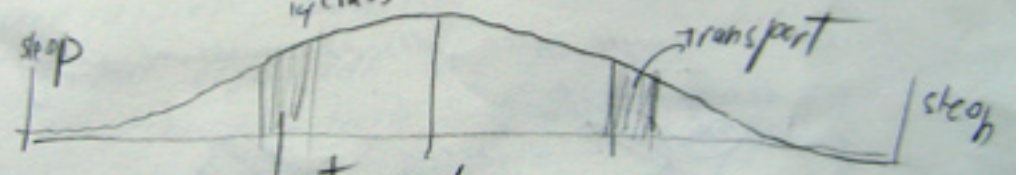

City rhythm —
can't get out of your head
tube noise
Sound as image —
- can a story be told with just sound?
just image =
just sound =
percussion behind sound?
What clips can be percussive?
- brushing teeth
- clip clop of underground
- feet in the hallway
- washing hands
- music listened to, played at club
- typing / mouse clicks

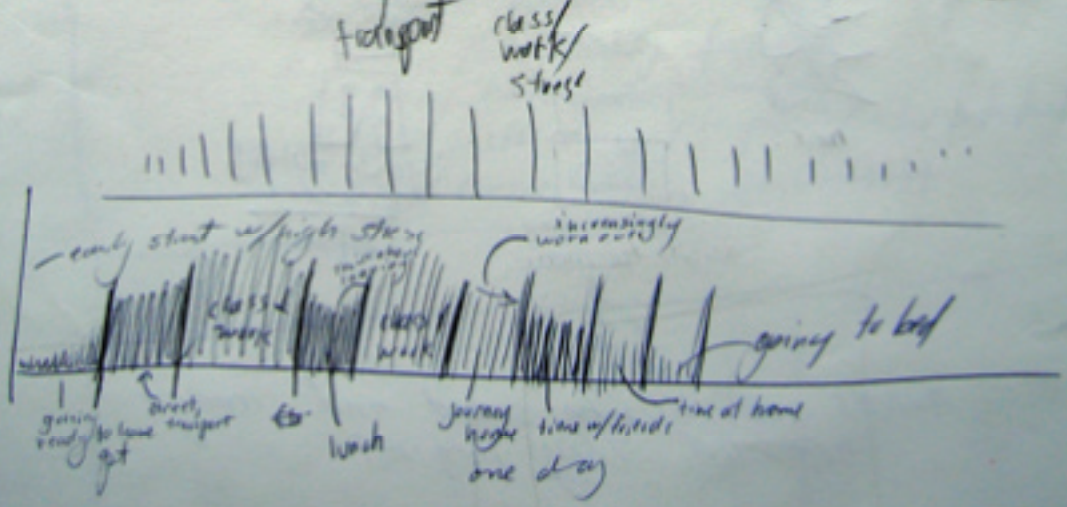
A journey in sand

A day –

alarm

1 wake up
2 brush teeth
3 wash face
4 bathroom
5 shower
6 dress
7 tea
8 eggs
9 door
10 lift
11 street
12 bus/tube
13 halls
14 class
15 walls
16 work
17 bus/tube
18 friends
19 go out
20 street
21 lift
22 door
23 TV/internet
24 dinner
25 wash
26 bed



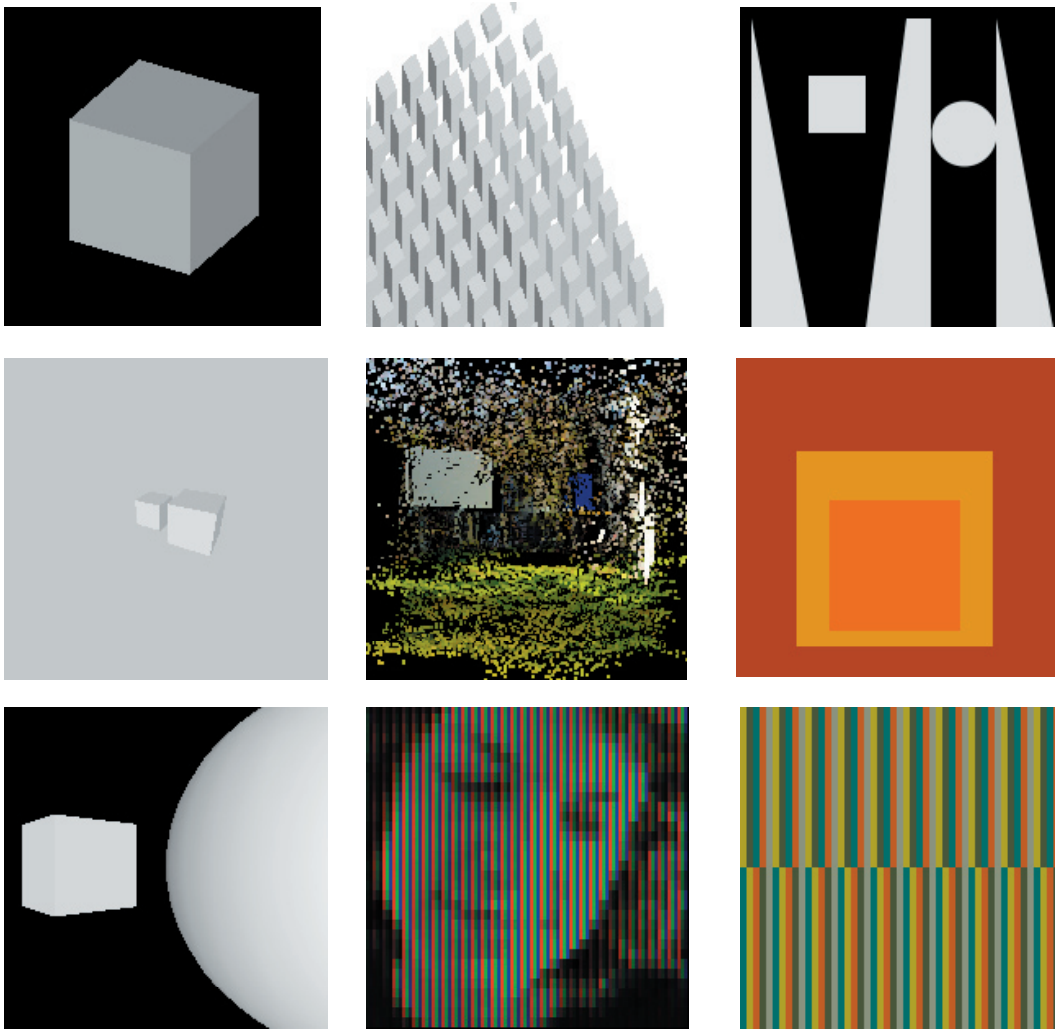


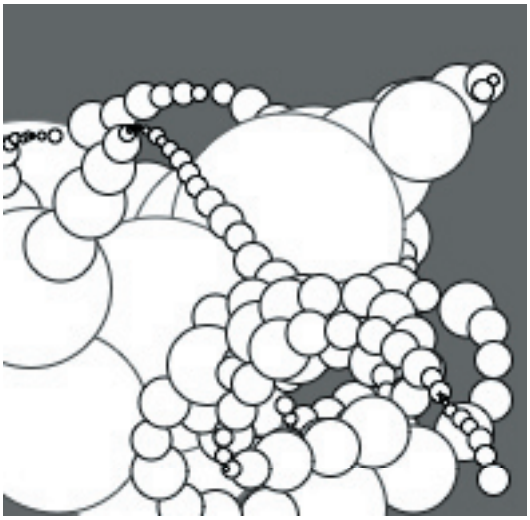
Data Visualization Notebook

Representation of Informatiokn

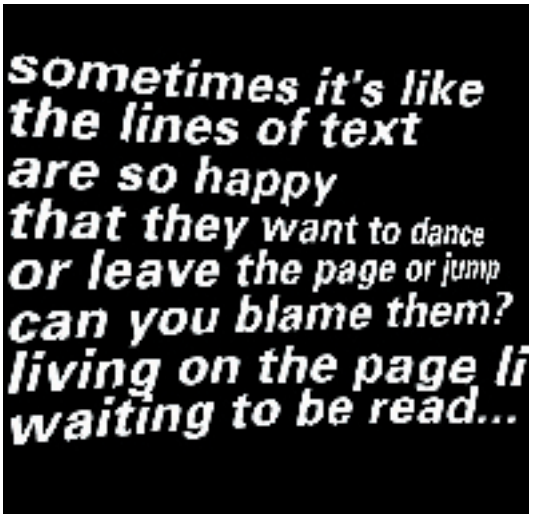
Jennifer Boriss

Processing is an open source programming language and environment for people who want to program images, animation, and sound. It is used by students, artists, designers, architects, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is developed by artists and designers as an alternative to proprietary software tools in the same domain.





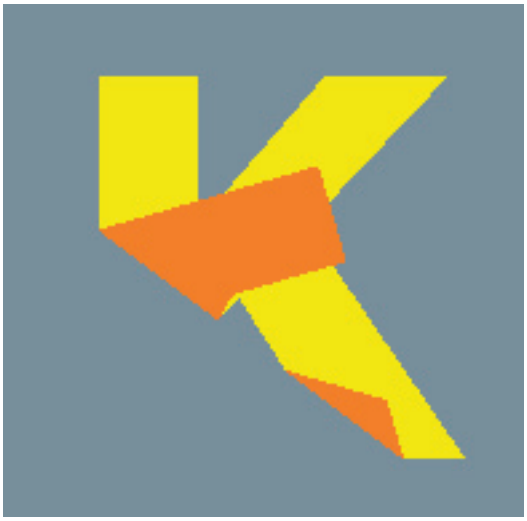
Pattern - responds to the speed of the mous



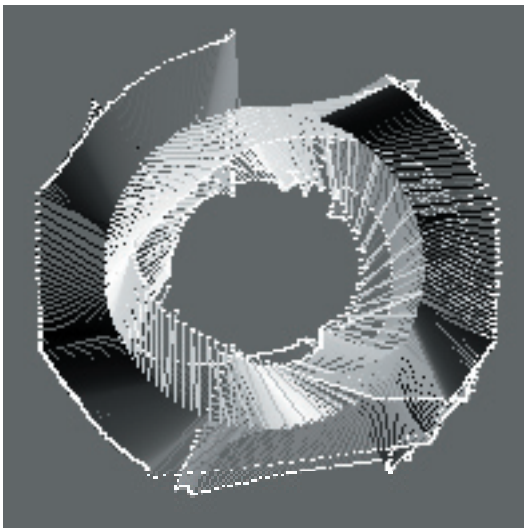
kinetic typography

Click - user presses

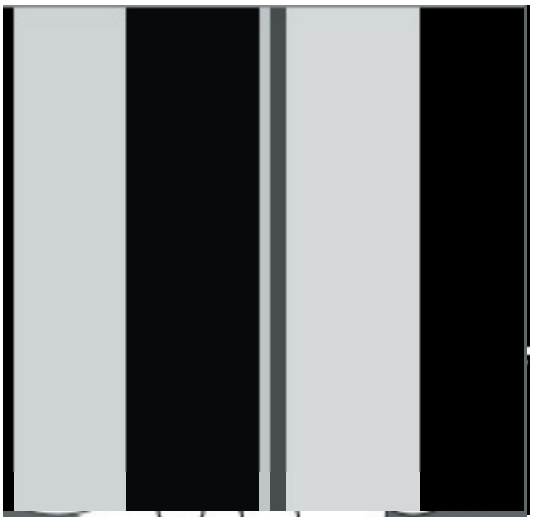
Constrain - Move the mouse across the screen to move the circle



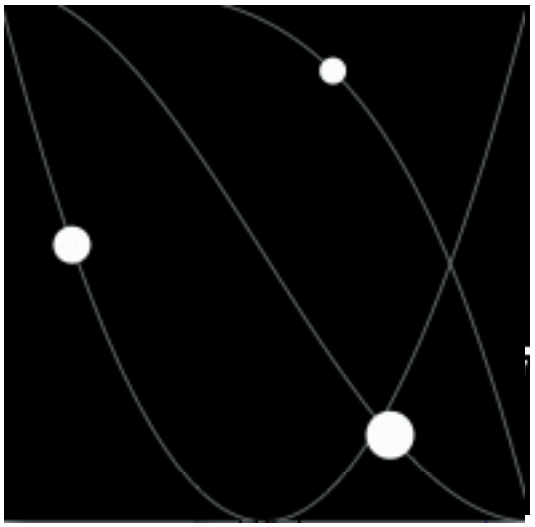
Letter K - Move the mouse across the screen to fold the “K”



Customtool - create unique tools to draw with.

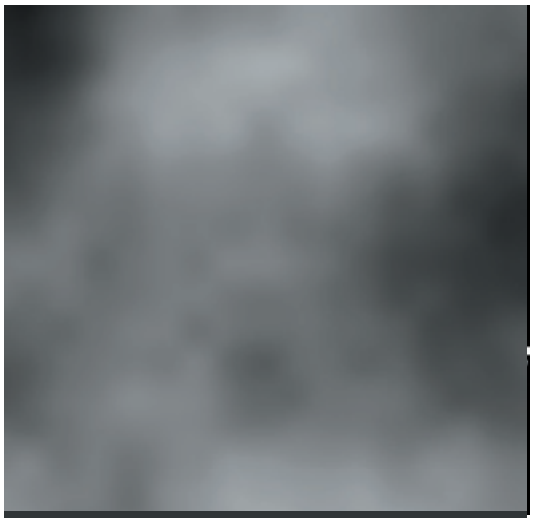


Keyboard - press letter keys to create forms in time and space

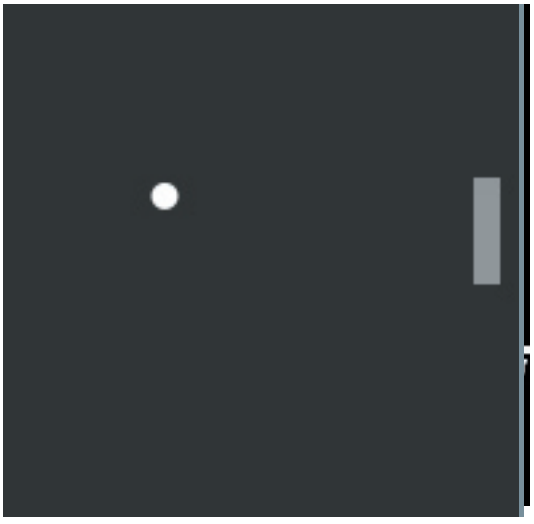


Shapes may be moved along function curves by setting their position to be the values input and returned

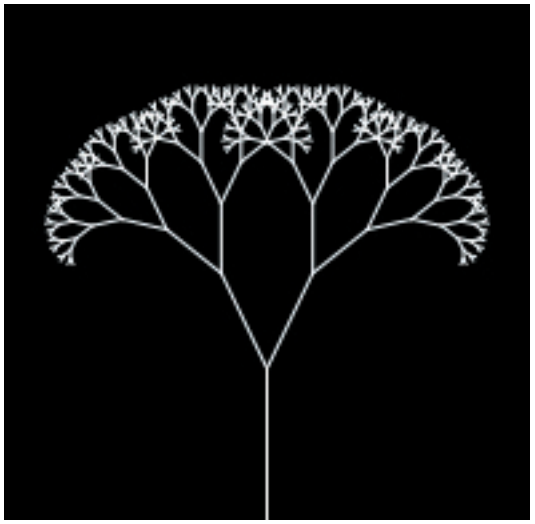
Keyboard functions - keyboard function keyPressed() is called whenever * a key is pressed



Using 3D noise to create simple animated texture.



simple collision detection



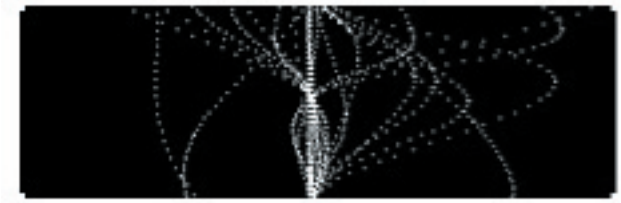
Renders a simple tree-like structure via recursion

The beta software for Processing 1.0 was released 20 April 2005 and can be downloaded here. Bug fixes are being made as we head toward the 1.0 release. Processing is free to download and available for GNU/Linux, Mac OS X, and Windows. Please help in releasing version 1.0!

Processing is an open project initiated by Ben Fry (Broad Institute) and Casey Reas (UCLA Design | Media Arts). Processing evolved from ideas explored in the Aesthetics and Computation Group at the MIT Media

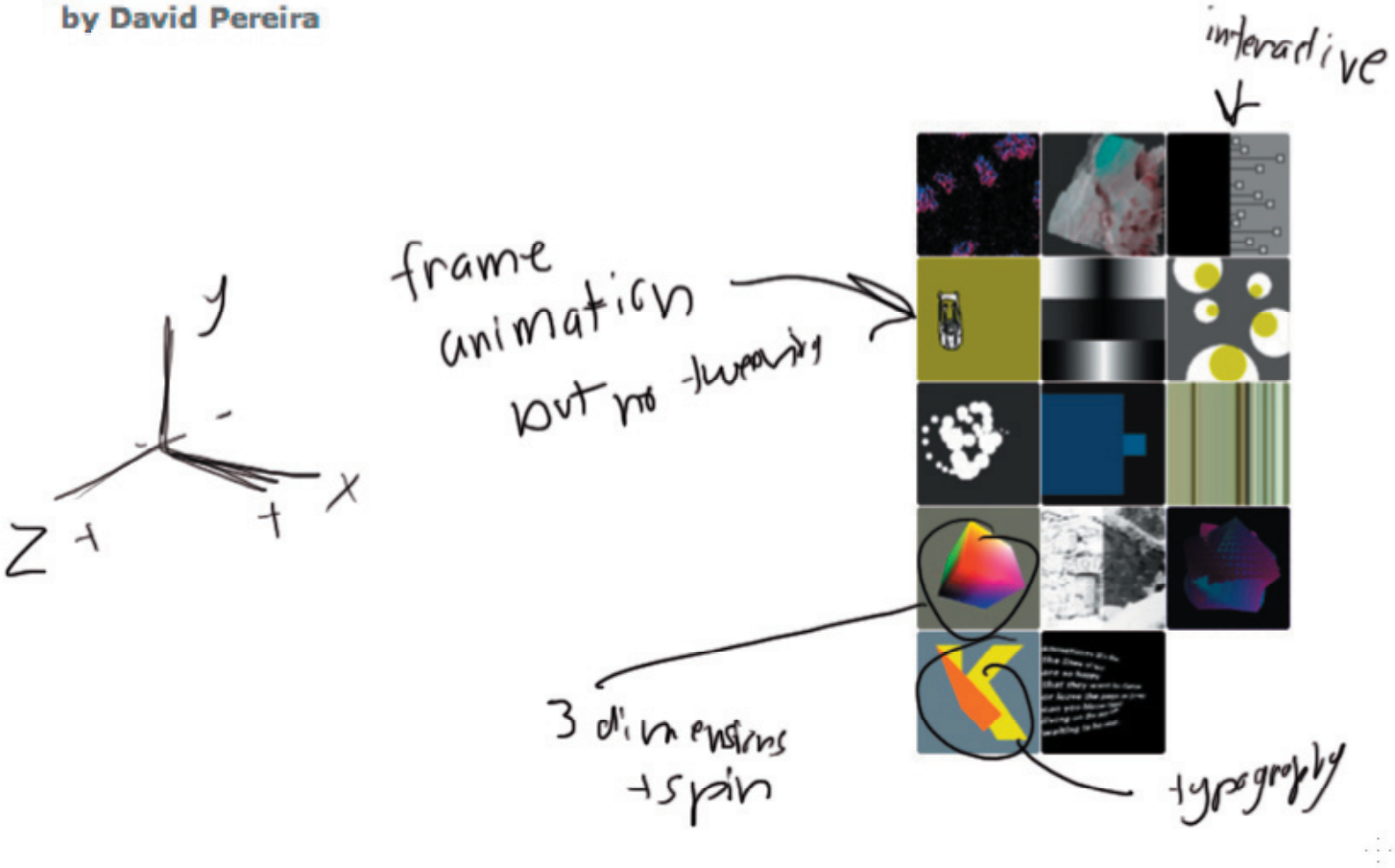
Spies by Lisa Strausfeld and James N. Sears

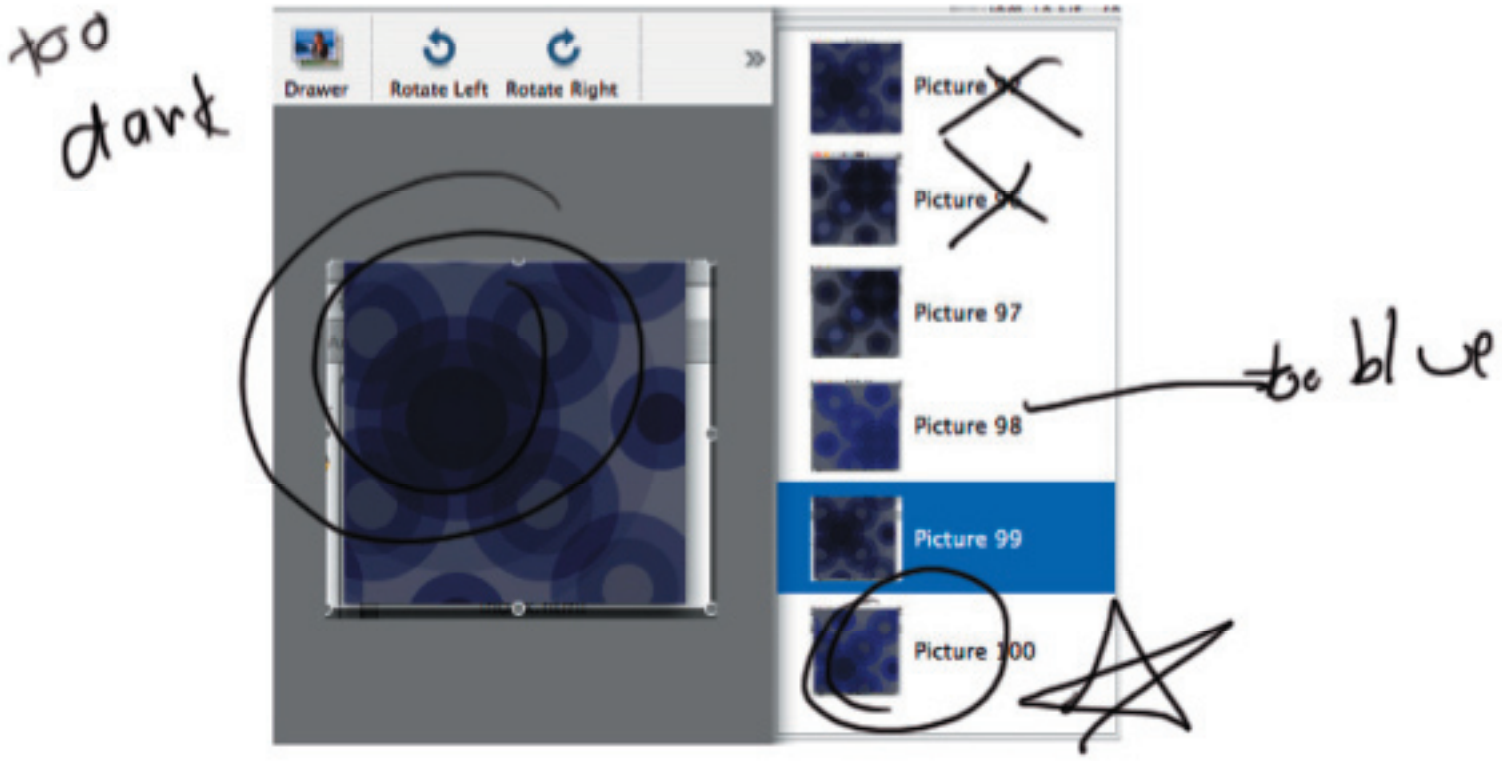
by Lisa Strausfeld and James N. Sears

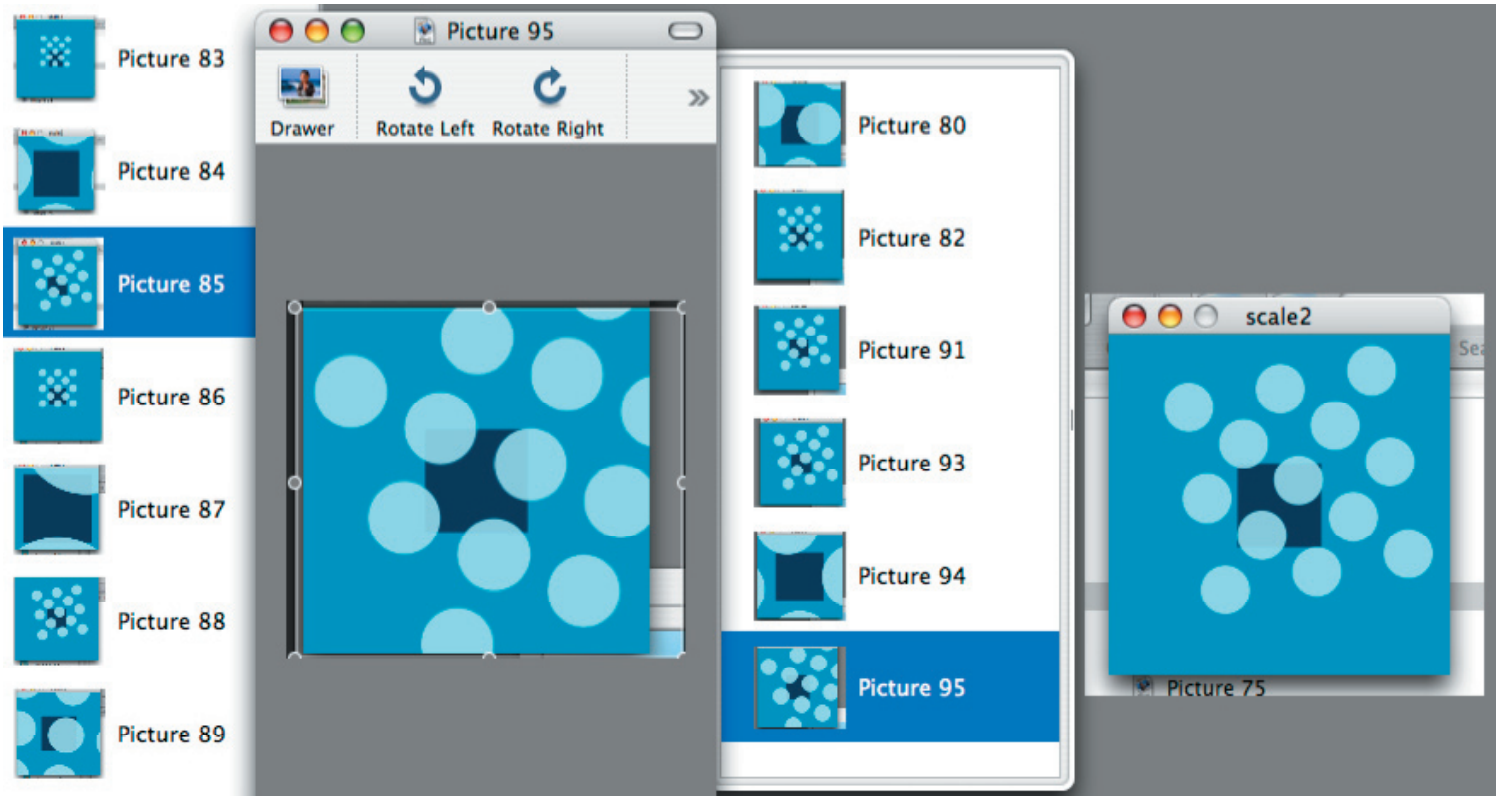


Mutualism

by David Pereira




too dark
Drawer
Rotate Left
Rotate Right
Picture 97
Picture 98
Picture 99
Picture 100
to blue

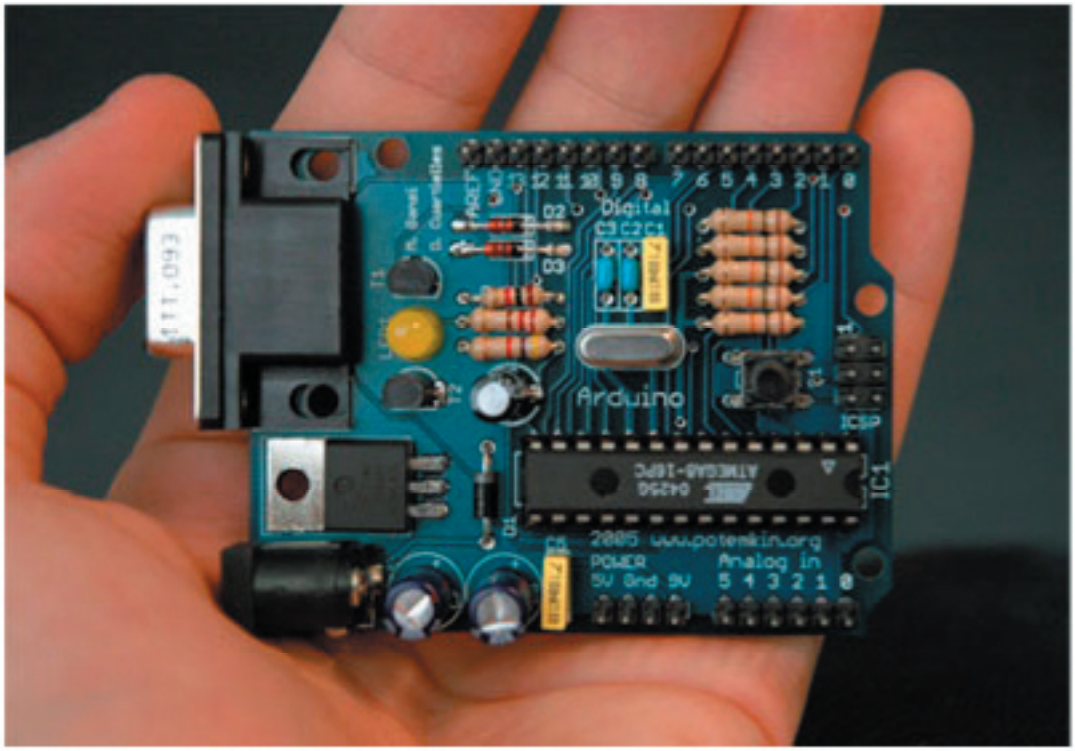

Picture 83
Picture 84
Picture 85
Picture 86
Picture 87
Picture 88
Picture 89
Picture 95
Drawer
Rotate Left
Rotate Right
Picture 80
Picture 82
Picture 91
Picture 93
Picture 94
Picture 95
scale2
Picture 75

Related projects to Processing:

Arduino is an open-source physical computing platform based on a simple i/o board, and a development environment for writing Arduino software. The Arduino programming language is an implementation of Wiring, itself built on Processing.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

Arduino received an Honory Mention in the Digital Communities section of the 2006 Ars Electronica Prix. Credits



Related projects to Processing:

Blender is the open source software for 3D modeling, animation, rendering, post-production, interactive creation and playback. Available for all major operating systems under the GNU General Public License.

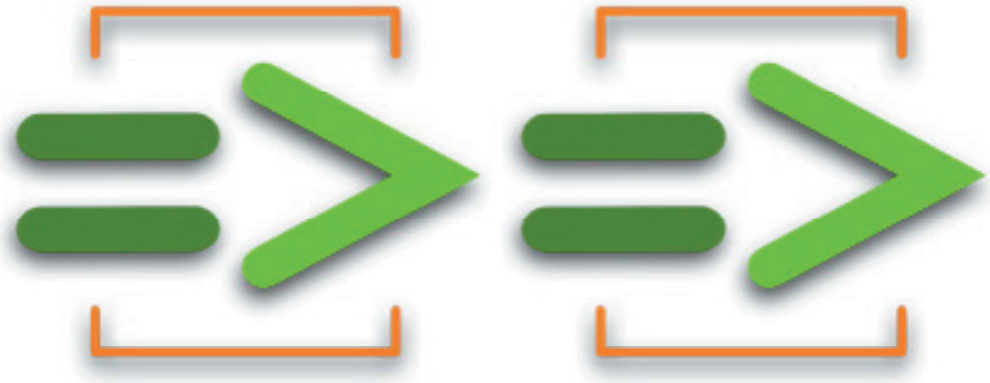
Old Man's Face
Grzegorz Rakoczy (grzybu)

Man
Vincent Girès

Related projects to Processing:

ChucK : Strongly-timed, Concurrent, and On-the-fly Audio Programming Language

authors: [chuck team]
date: 2002 - present
version: 1.2.0.7 (dracula)
(what's new?)

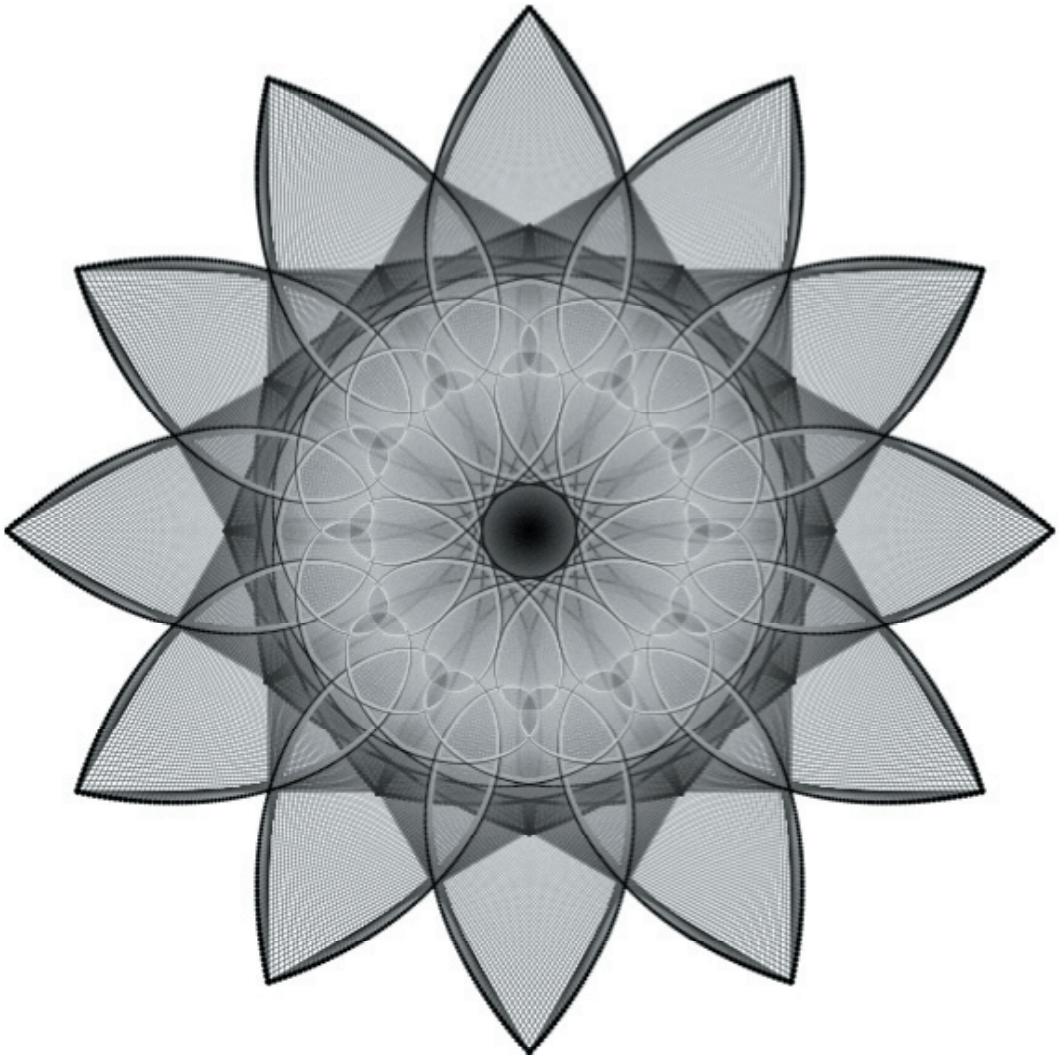


Welcome to ChucK!

what is it? : ChucK is a new (and developing) audio programming language for real-time synthesis, composition, and performance - fully supported on MacOS X, Windows, and Linux. ChucK presents a new time-based, concurrent programming model that's highly precise and expressive (we call this strongly-timed), as well as dynamic control rates, and the ability to add and modify code on-the-fly. In addition, ChucK supports MIDI, OSC, HID device, and multi-channel audio. Furthermore, the language is designed to favor readability and flexibility over raw performance. It's fun and easy to learn, and offers composers, researchers, and performers a powerful programming tool for building and experimenting with complex audio synthesis programs, and real-time interactive control.

Related projects to Processing:

Context Free Art



```
startshape flower6

rule flower6 {
        12* {r 30} petal6 { }
        12* {r 30} petal6 {flip 90}
}

rule petal6 {
        SQUARE { s 1 0.001 }
        CIRCLE { x -0.5 s 0.01 b -1 }
        petal6 [ x 0.5 r 120.21 s 0.996 x 0.5 b 0.005 ]
}
```

Related projects to Processing:

Design By Numbers

News

- August 2003: This month the beta version of DBN 4 will be released. DBN 4 is the 4th (and final re-write) of DBN by Jessica Rosenkrantz. Stay tuned.
- Recent contribution by Prof. Warren Sack at UC Berkeley to implement LOGO-style graphics [info].
- Answers to many common questions can be found in the DBN FAQ
- From winter of 2002, John Maeda has taken over development of DBN. Please expect a major revision by the end of summer of 2003 that is reduced to greater simplicity.
- Casey Reas and Ben Fry have released an advanced system for learning programming called Processing which is a production-class environment for creating systems in Java with color, flexible screen-size, realtime three-dimensional graphics, and a host of other features.

Related projects to Processing:

Eclipse - an open development platform

Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform. **New to Eclipse?**

Download Eclipse

Eclipse is used for ...

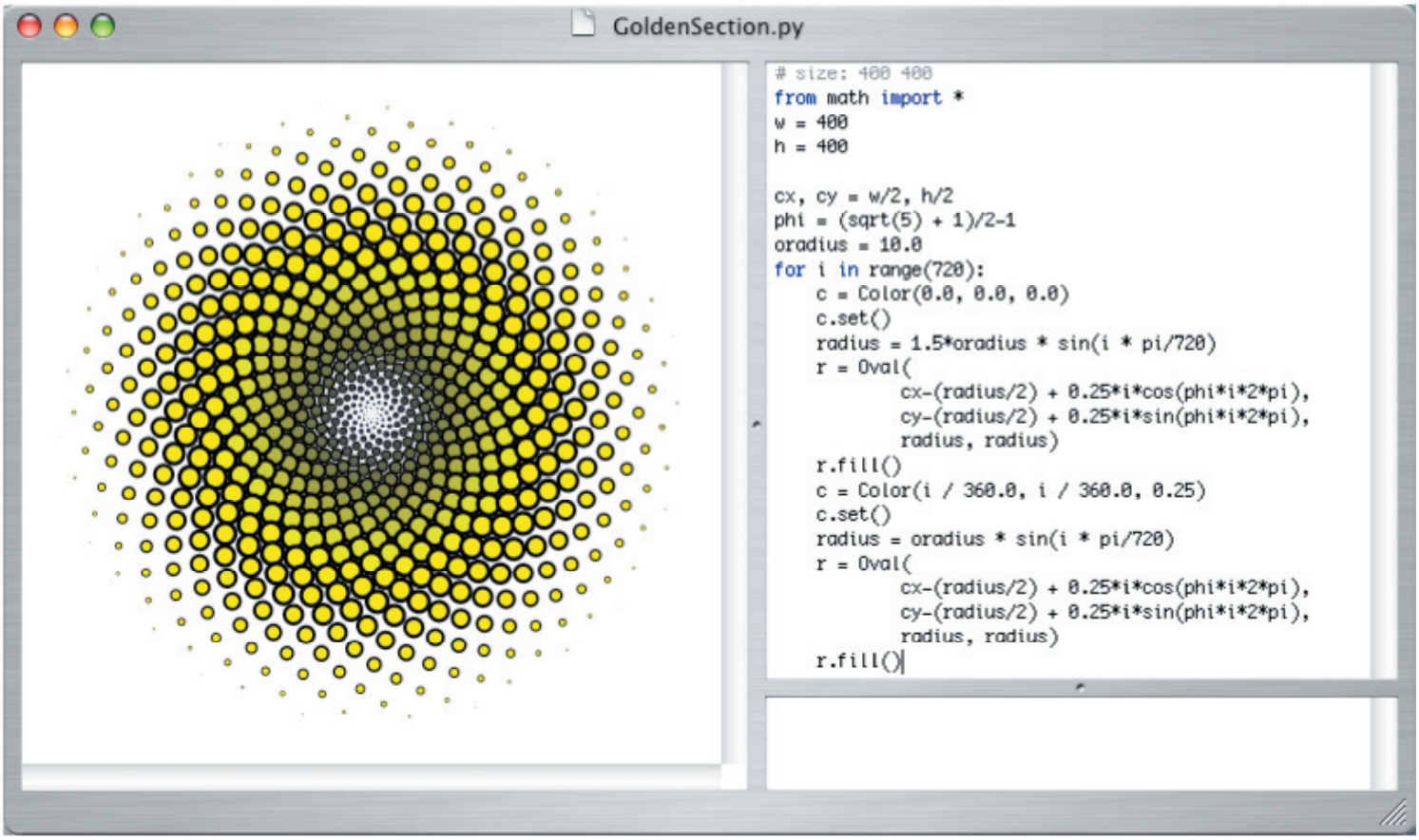
Enterprise Development

Embedded + Device Development

Rich Client Platform

Application Frameworks

Language IDE



DrawBot

DrawBot is a simple application for MacOSX that allows you to write simple Python scripts that generate two-dimensional graphics. The builtin graphics primitives are currently pretty braindead, it currently only supports rectangles, ovals and (bezier) paths and polygons.

DrawBot is written in Python. The binary download is fully self-sufficient (ie. it doesn't need a Python install around), but if you use or build it from the source you'll need Python.framework 2.3 as well as PyObjC 1.0 (*** 1.0b1 does _not_ work, in contrast to what the readme says).

About Isadora®

Isadora is a graphic programming environment for Macintosh (with a Windows version now in public beta) that provides interactive control over digital media, with special emphasis on the real-time manipulation of digital video.

Because every performance or installation is unique, Isadora was designed not to be a "plug and play" program, but instead offers building blocks that can be linked together in an almost unlimited number of ways, allowing you to follow your artistic impulse.

Because we use Isadora to teach Troika Ranch's Live-I Workshops, it was designed to be easy to learn for those new to the world of real-time interaction. The titles and values of every actor are visible to the user (though they can be hidden if you desire). Help is available for all actors and their inputs and outputs from within the program. Isadora is accompanied by a complete manual that includes numerous tutorials and example files to get you started.

Isadora was designed by composer and media artist Mark Coniglio, and as such, it reflects over 10 years of practical experience with realtime interaction.

Jitter

A Brilliant Collection of Video, Matrix, and 3D Graphics Objects for Max

Remove Budc

Overview

Jitter extends the Max/MSP programming environment to support realtime manipulation of video, 3D graphics and other data sets within a unified processing architecture.

Because Jitter, like Max/MSP, is generic in nature, it offers unlimited possibilities for creative exploration. Whether you are interested in video processing, interactive art, teaching new media, or data visualization, Jitter offers both high and low level tools for working in exciting new ways.

Jitter abstracts all data as multidimensional matrices, so objects that process images can also process audio, volumetric data, 3d vertices, or any numerical information you can get into the computer. Jitter's common representation simplifies the reinterpretation and transformation of media. And with Jitter 1.5, many types of data can be processed on the GPU, leveraging the massively parallel computing power of today's latest graphics cards.

- **Video**
- **Graphics**
- **Ease of Use**
- **Matrices**
- **More Details**
- **System Requirements**
- **Pricing**
- **Upgrading to Jitter 1.5**
- **Frequently Asked Questions Page**
- **Download**
- **Jitter in Education**

Computer Vision for Artists

Myron is the cross-platform, cross-language, open source, video capture and computer vision plugin. One core C++ object gets cross-compiled as a handful of high level language "wrapper" libraries. The wrapper for Java and Processing is called JMyron. The wrapper for Macromedia Director is called WebCamXtra. The aim of the project is to keep computer vision free and easy for the new media education and arts community.

About Pure Data

The Pure Data Portal

PD (aka Pure Data) is a real-time graphical programming environment for audio, video, and graphical processing. It is the third major branch of the family of patcher programming languages known as Max (Max/FTS, ISPW Max, Max/MSP, jMax, etc.) originally developed by Miller Puckette and company at IRCAM. The core of Pd is written and maintained by Miller Puckette and includes the work of many developers, making the whole package very much a community effort.

Pd was created to explore ideas of how to further refine the Max paradigm with the core ideas of allowing data to be treated in a more open-ended way and opening it up to applications outside of audio and MIDI, such as graphics and video.

The Python Programming Language

Python® is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

Wiring is an open source programming environment and electronics i/o board for exploring the electronic arts, tangible media, teaching and learning computer programming and prototyping with electronics. It illustrates the concept of programming with electronics and the physical realm of hardware control which are necessary to explore physical interaction design and tangible media aspects.

vvvv : a multipurpose toolkit

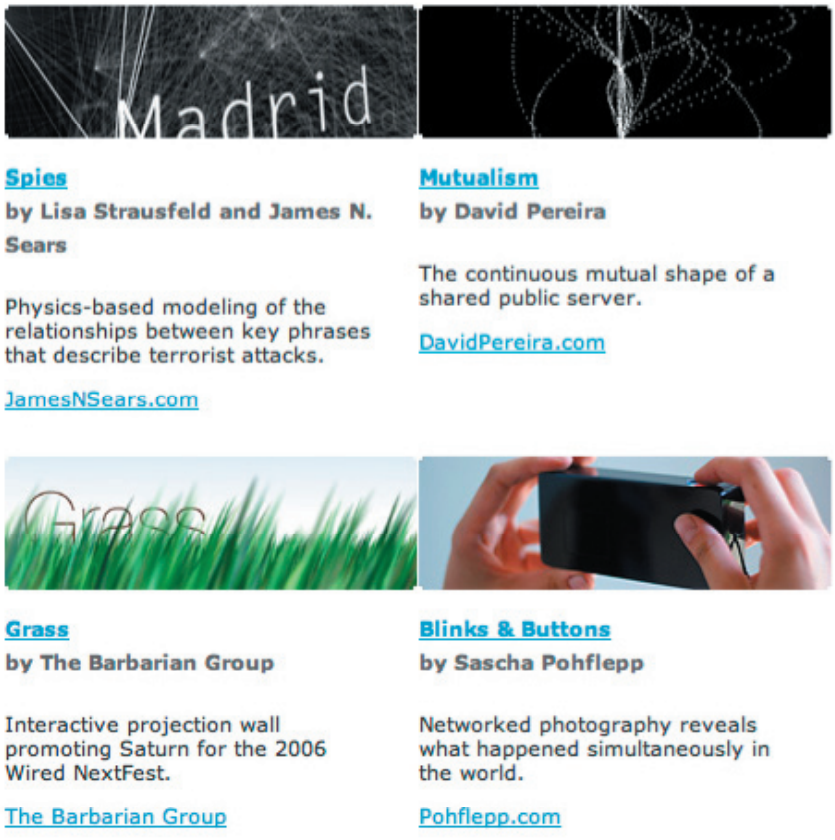
vvvv is a toolkit for real time video synthesis. It is designed to facilitate the handling of large media environments with physical interfaces, real-time motion graphics, audio and video that can interact with many users simultaneously.

Abstract:

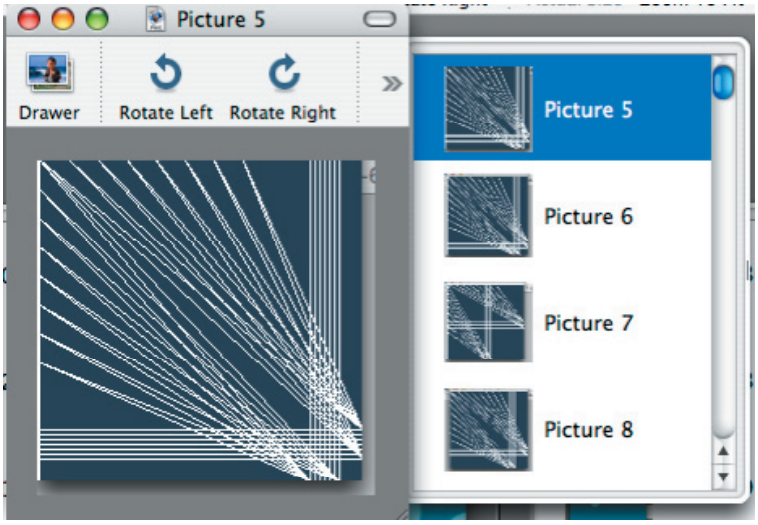
Representation Information (RI) is a term encompassing all information required to access the information stored within a digital object. The term can be applied to all levels of abstraction and refers to both the structural and semantic composition. The use of RI is often recursive: using one element of RI in a meaningful manner requires further RI. This recursion continues until the contents of the original object are displayed in a form the user can understand. A collaborative model for storing, maintaining, accessing, and using Representation Information is required to assist the development of long term digital curation strategies.

There is No Such Thing as Information Design

One of the ways we increase the returns we obtain from understanding the world is by adapting our vocabulary to reflect our increasing knowledge. Humanity never would have progressed beyond building the most primitive of mechanical engines if careful thinkers had not learned to distinguish between energy, force, work, and power. These words are still used loosely in everyday speech, but professional mechanical designers and physicists use them carefully and with well-defined meanings. Professionals in the field of information-related design will have to learn to be equally careful. We must not allow sloppy thinking to muddy the deep waters we find here at the meeting of art, psychology, and electronic technology. Things are difficult enough as it is. .... http://www.taskz.com/ucd_no_info_design_indepth.php



```
rect(20, 175, 5, 5);
rect(30, 175, 5, 5);
rect(40, 175, 5, 5);
rect(50, 175, 5, 5);
rect(60, 175, 5, 5);
rect(70, 175, 5, 5);
rect(80, 175, 5, 5);
rect(90, 175, 5, 5);
rect(100, 175, 5, 5);
rect(110, 175, 5, 5);
rect(120, 175, 5, 5);
rect(130, 175, 5, 5);
rect(140, 175, 5, 5);
rect(150, 175, 5, 5);
rect(160, 175, 5, 5);
rect(170, 175, 5, 5);
```



line -
first x, firsty, second x, second y

rectangle: // The first two parameters to rect() are c
// and the second two are the width and height

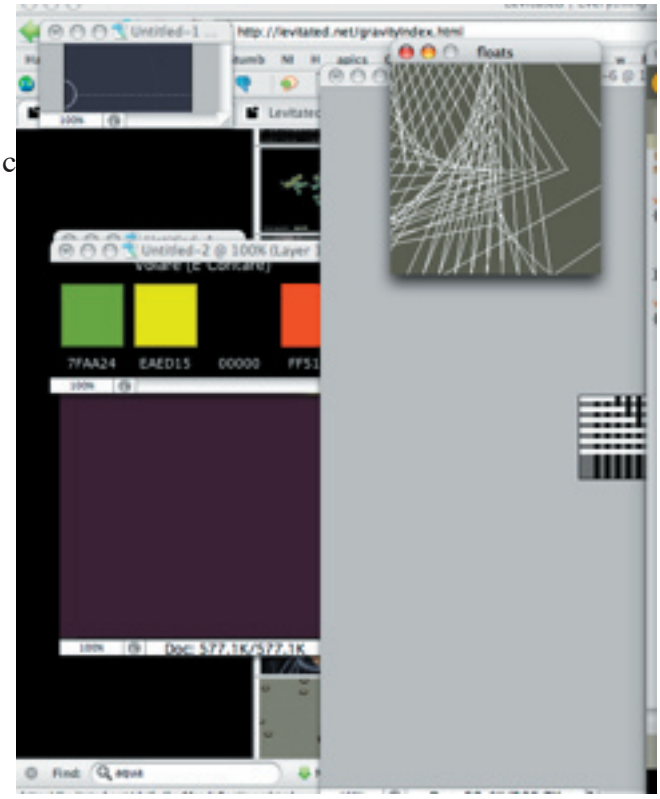
draw_target

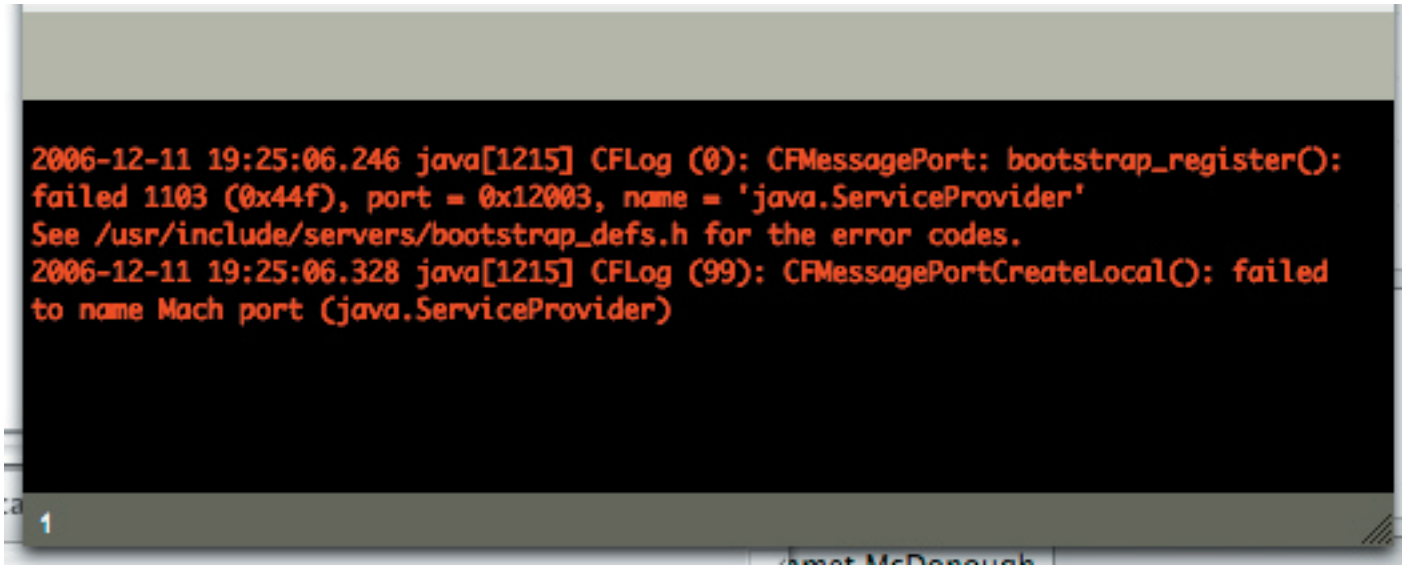
```
  draw_target(68, 34, 200, 10);
```
position x, position y, size, number of rings

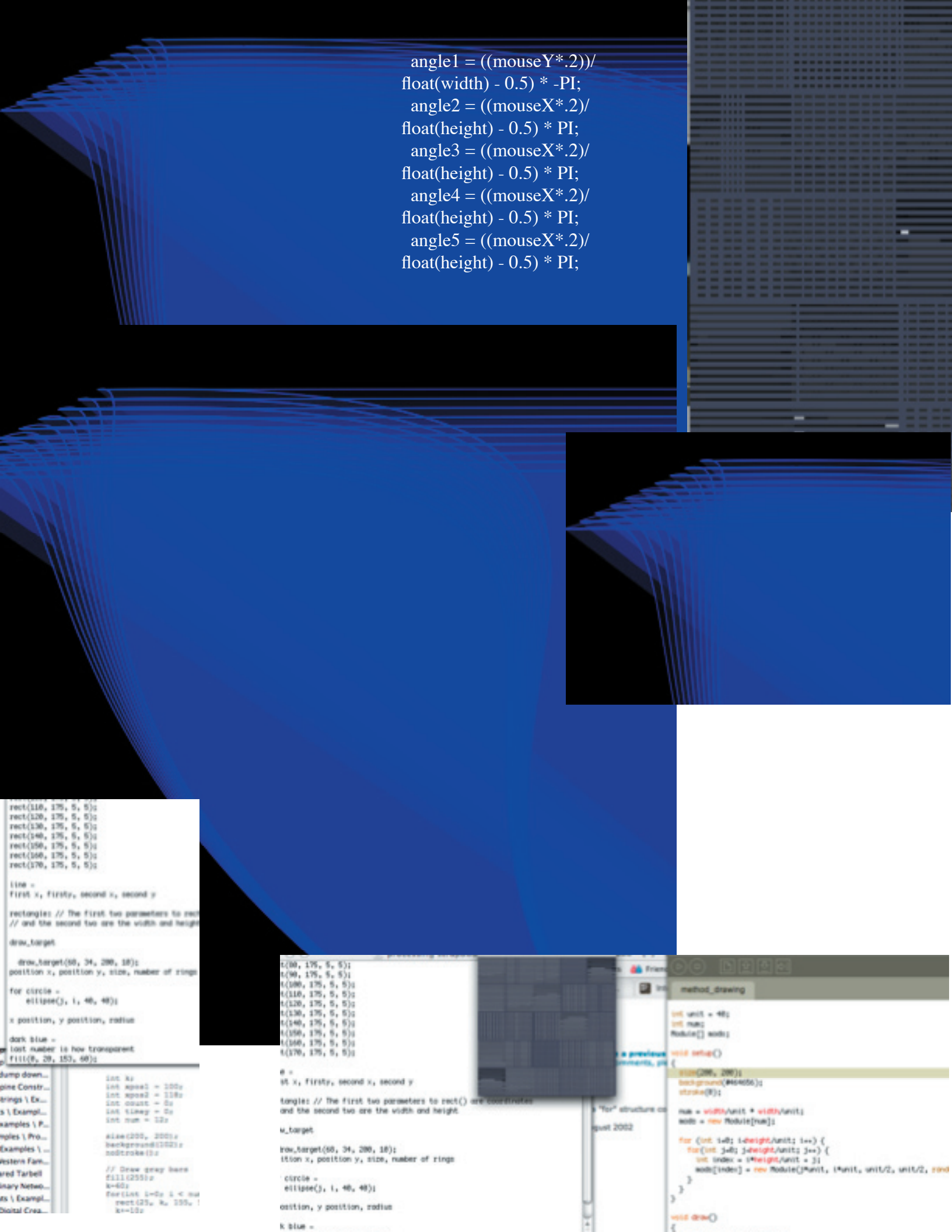
for circle -
```
    ellipse(j, i, 40, 40);
```

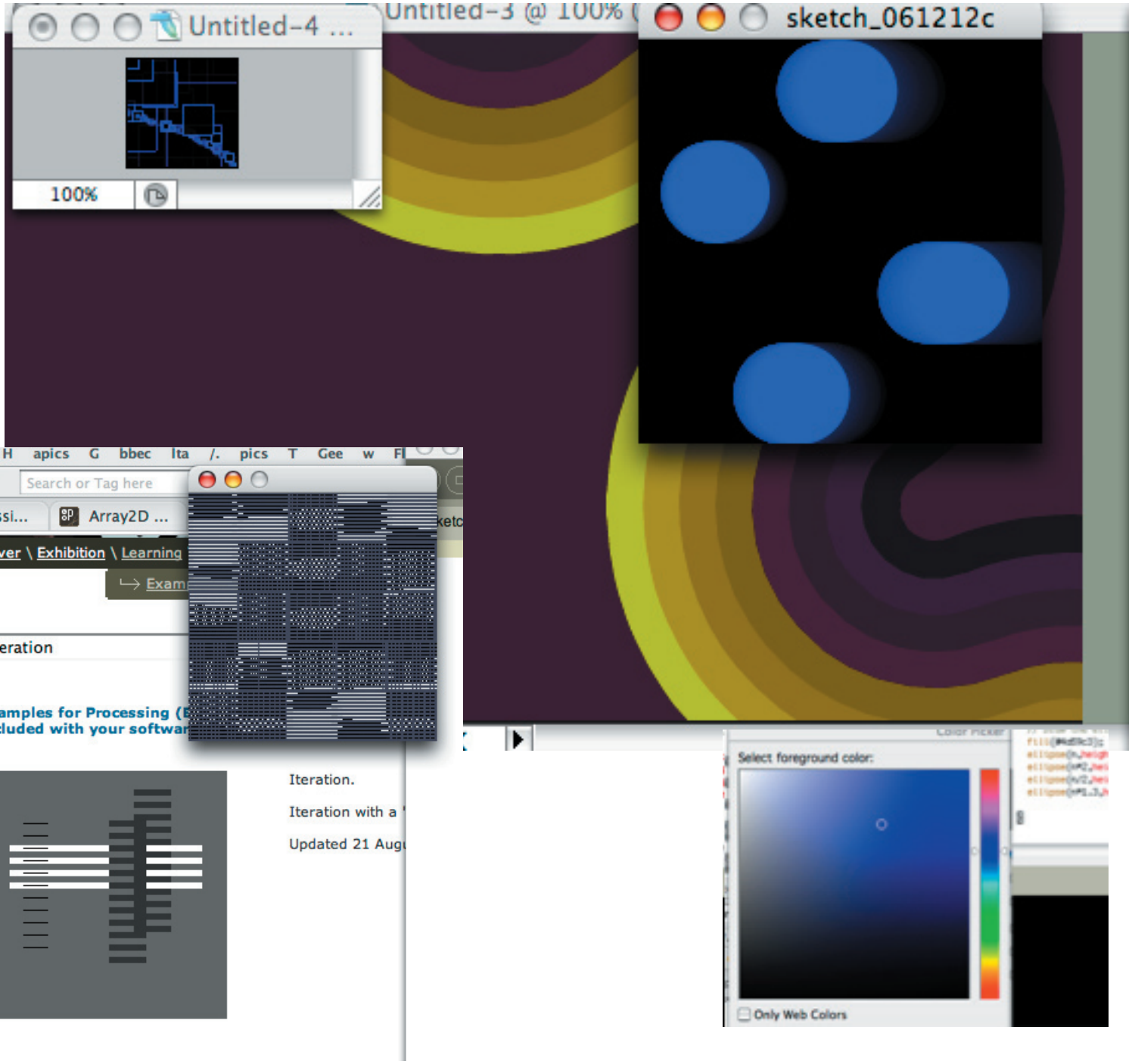
x position, y position, radius

dark blue -
last number is how transparent
```
fill(0, 20, 153, 60);
fill(#0000CC, 90);
stroke(#0000FF);
```



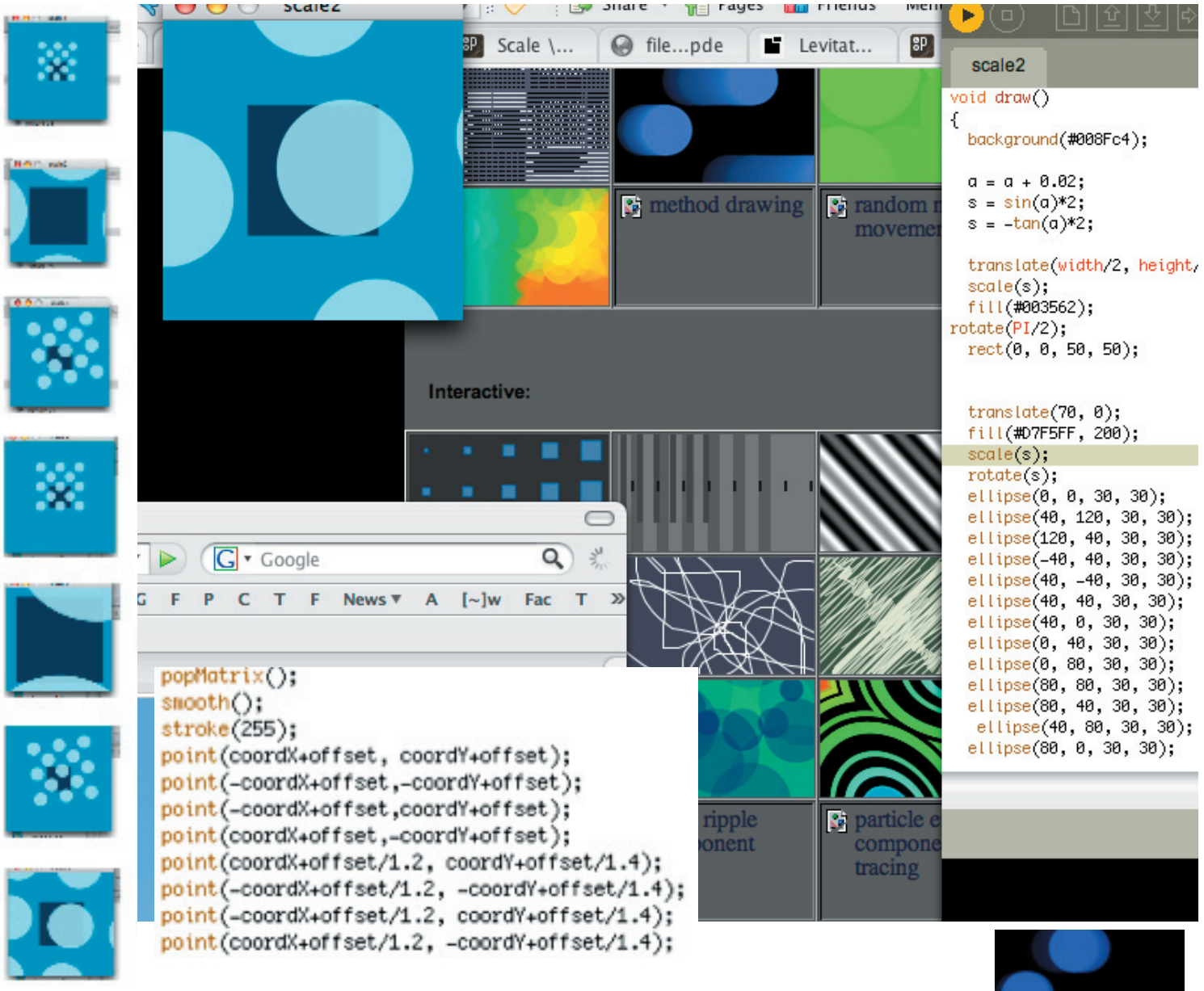



angle1 = ((mouseY*.2))/
float(width) - 0.5) * -PI;
angle2 = ((mouseX*.2)/
float(height) - 0.5) * PI;
angle3 = ((mouseX*.2)/
float(height) - 0.5) * PI;
angle4 = ((mouseX*.2)/
float(height) - 0.5) * PI;
angle5 = ((mouseX*.2)/
float(height) - 0.5) * PI;

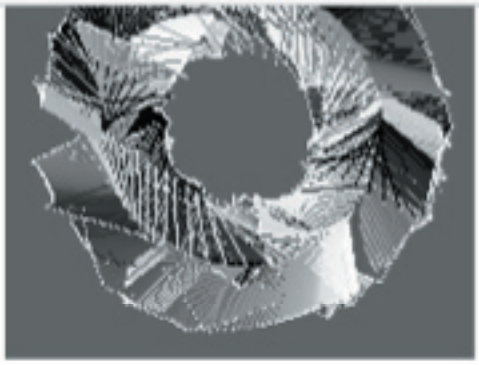


<td height="22"><img src="icons/interactive/gradient_sketch.jpg" width="133" height="78" alt="gradient sketch" border="0"></td>
<td height="22"><img src="icons/interactive/hand.jpg" width="133" height="78" border="0" alt="segmented motion"></td>
<td height="22"><img src="icons/interactive/pulsing.jpg" width="133" height="78" border="0" alt="duplicate inputs"></td>

<td height="22"><img src="icons/interactive/wreath.jpg" width="133" height="78" border="0" alt="coordinate offset trajectories"></td>
</tr>
<tr align="center">
<td height="22"><img src="icons/interactive/blinds.jpg" width="133" height="78" border="0" alt="colored blinds double input"></td>
<td height="22"><a href="circle_draw/applet/index.html"><img src="icons/interactive/circle_draw.jpg"

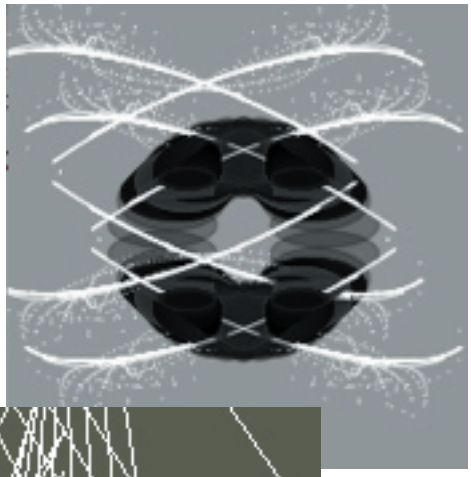


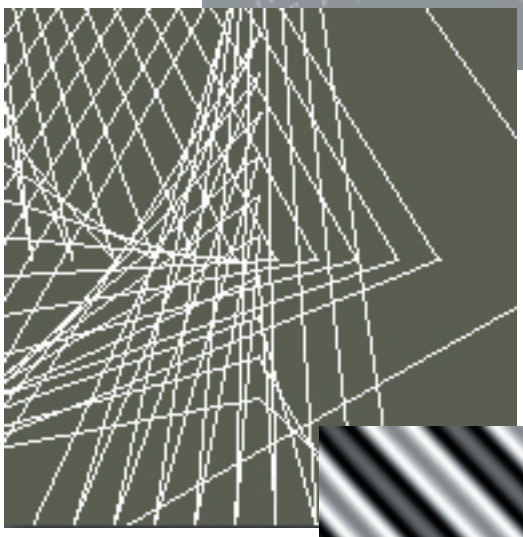
```
void setup()
{
  size(200, 200);
  colorMode(RGB, width);
  a = width;
  b = width;
  direction = false;
  frameRate(30);
  smooth();
}

void draw()
{
```



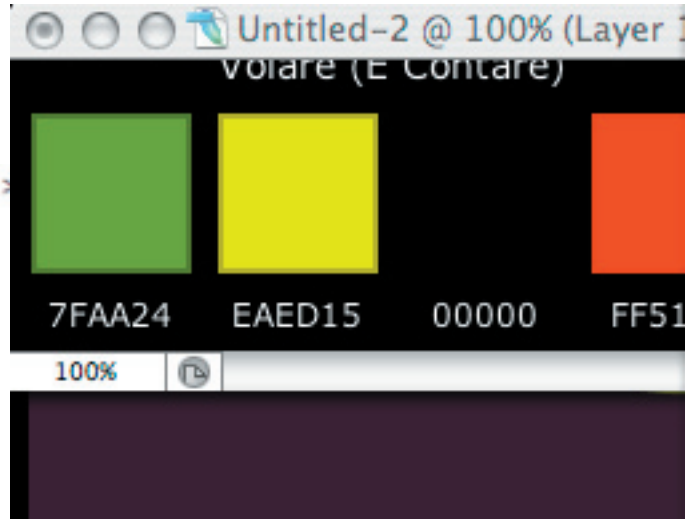
it is possible t

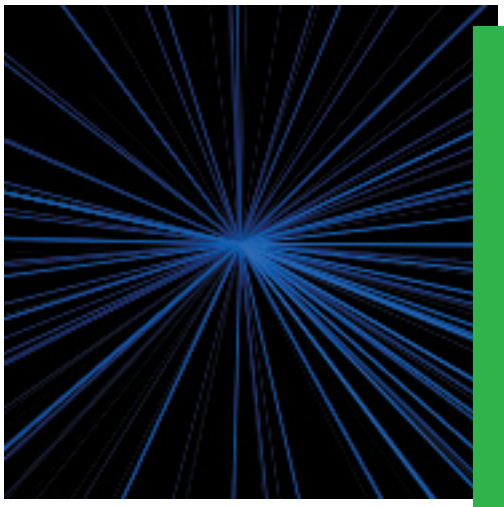
Created 23 Oc

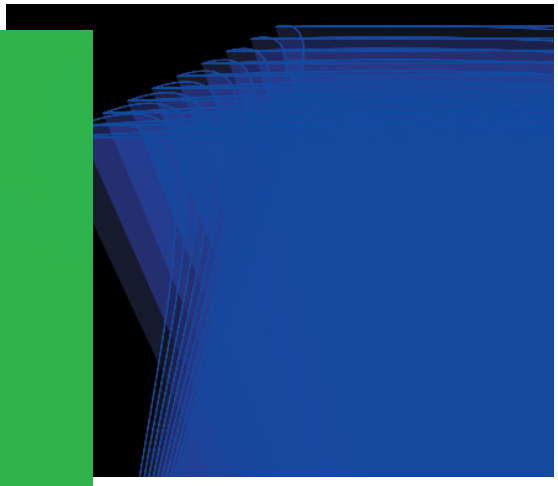
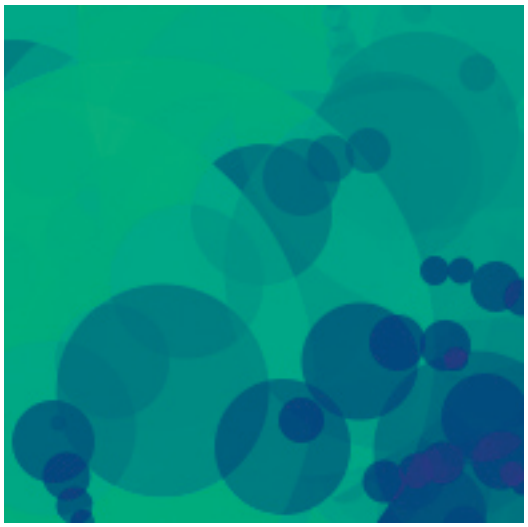
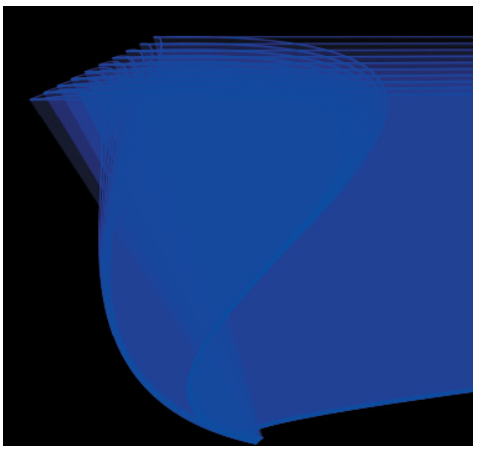
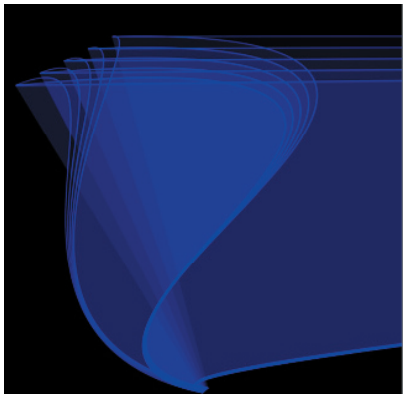
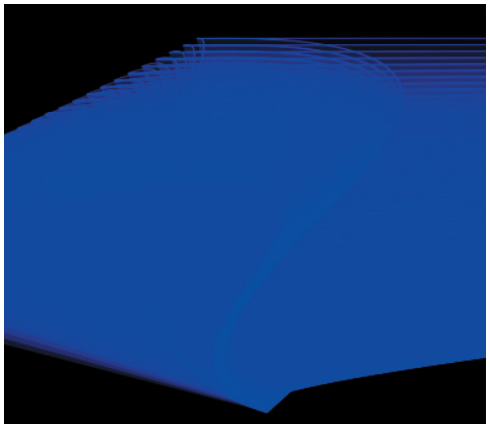
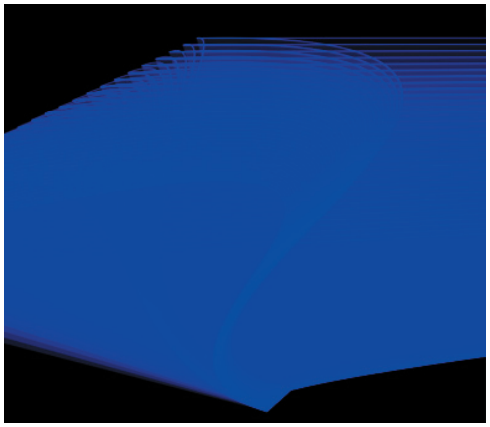


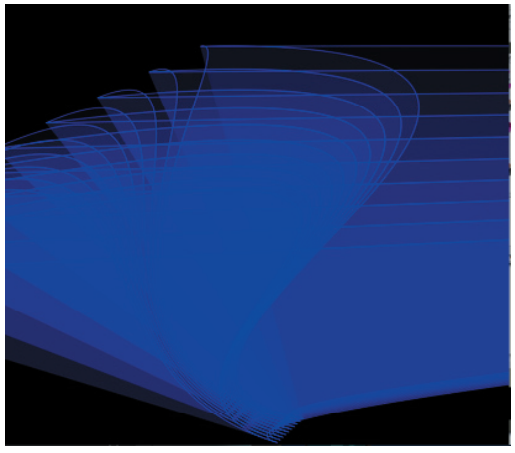


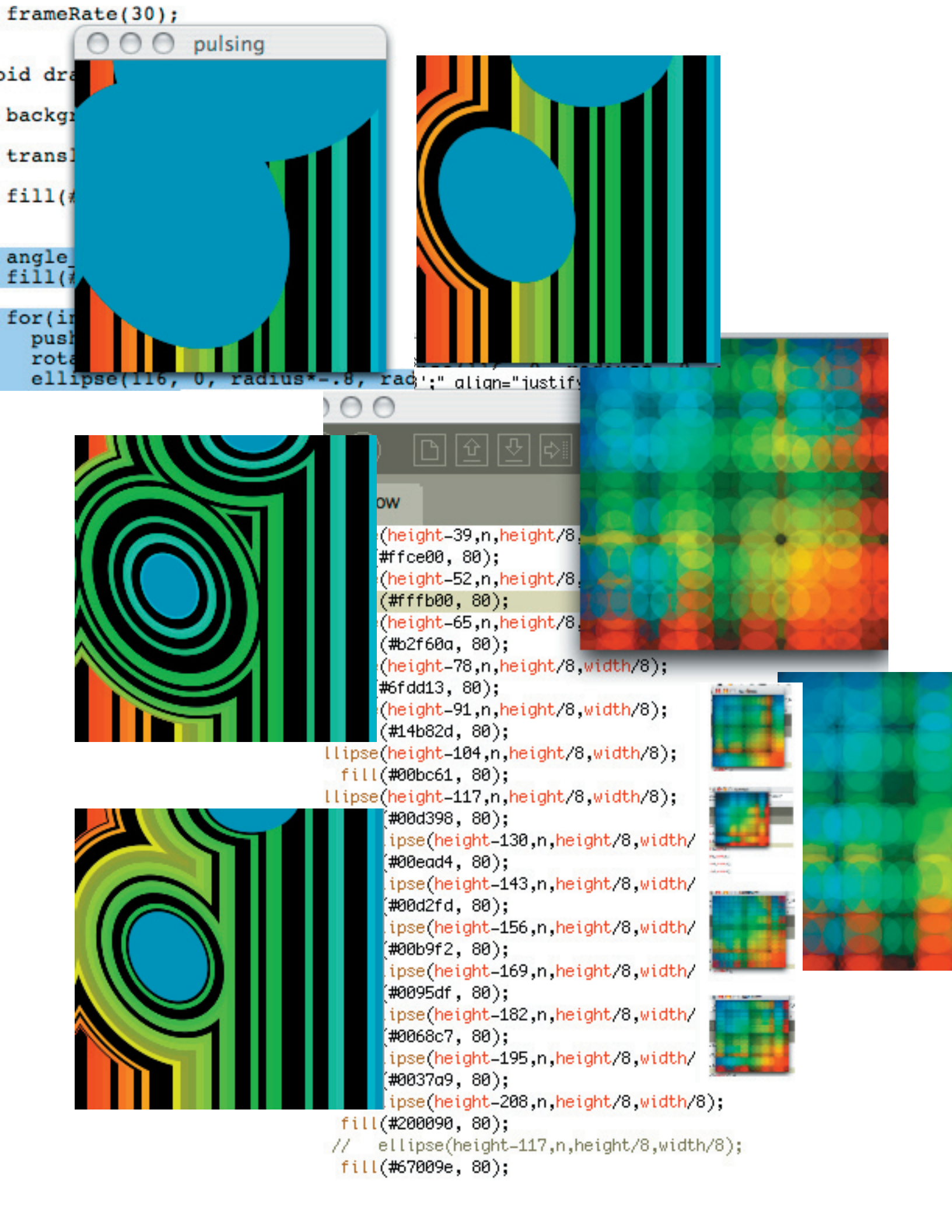
```
int dots = 1000;
float[] dX = new float[dots];
float[] dY = new float[dots];

float l_0 = 0.0;
float h_0 = 0.0;

float legX = 0.0;
float legY = 0.0;
float thighX = 0.0;
float thighY = 0.0;

float l = 30.0; // Length of the 'leg'
float h = 60.0; // Height of the 'leg'

float nmx, nmy = 0.0;
float mx, my = 0.0;

float offset;

int currentValue = 0;
int valdir = 1;

void setup()
{
  size(200, 200);
  noStroke();
  fill(102);
  rect(0, 0, width, height);
  offset = width/2;
}

void draw()
{
  // Smooth the mouse
  nmx = mouseX;
  nmy = mouseY;
  if((abs(mx - nmx) > 1.0) || (abs(my - nmy) >
    mx = mx - (mx-nmx)/20.0;
```

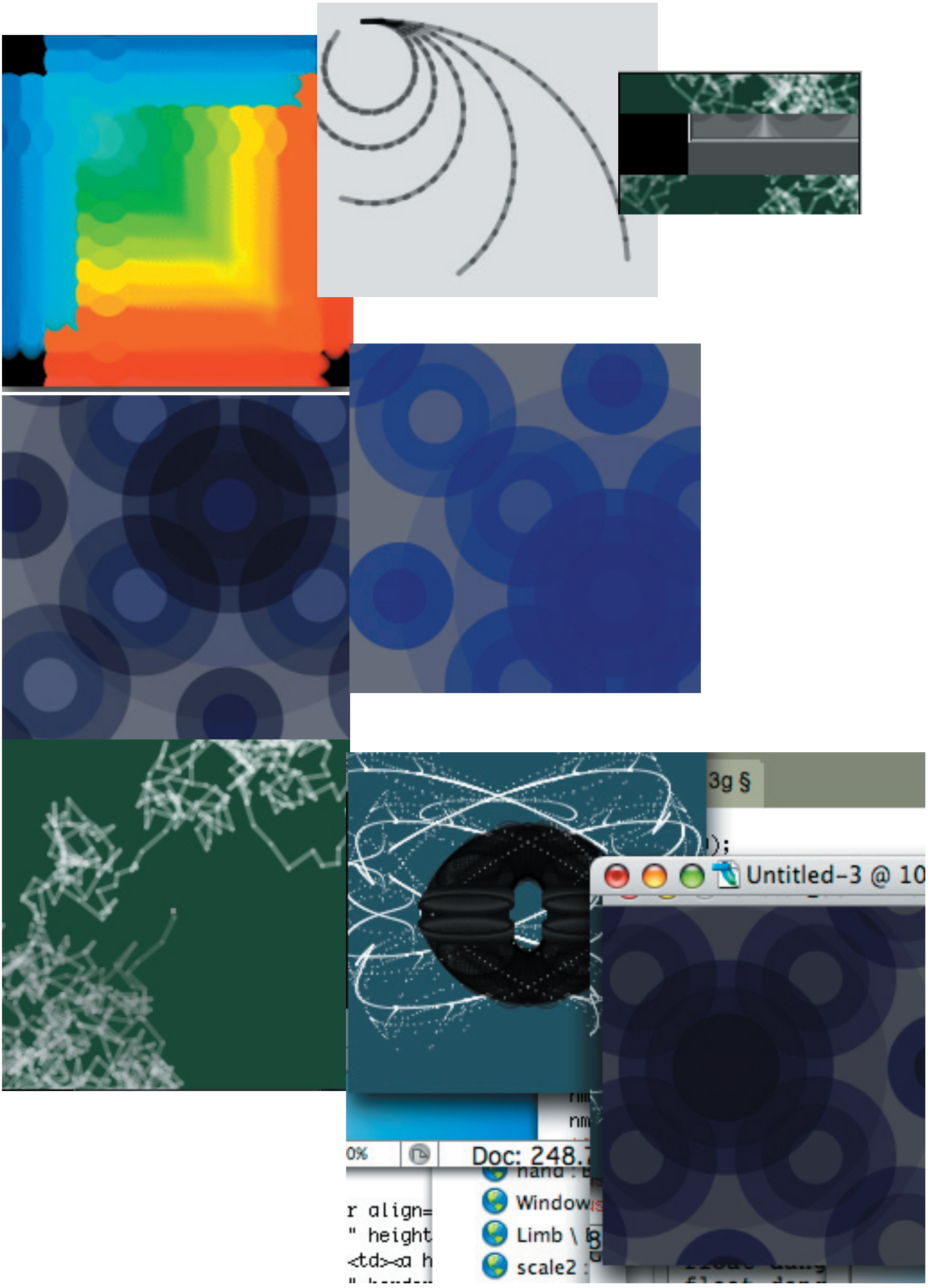


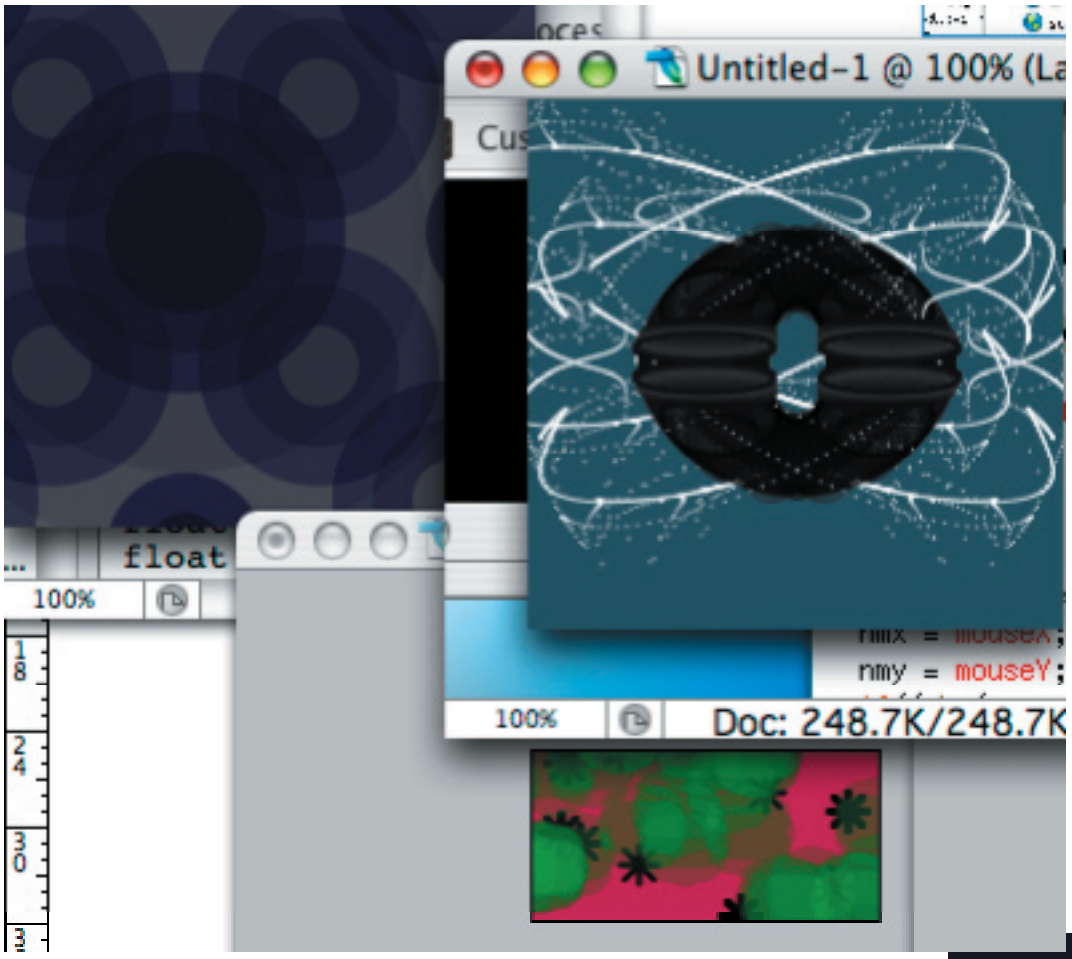


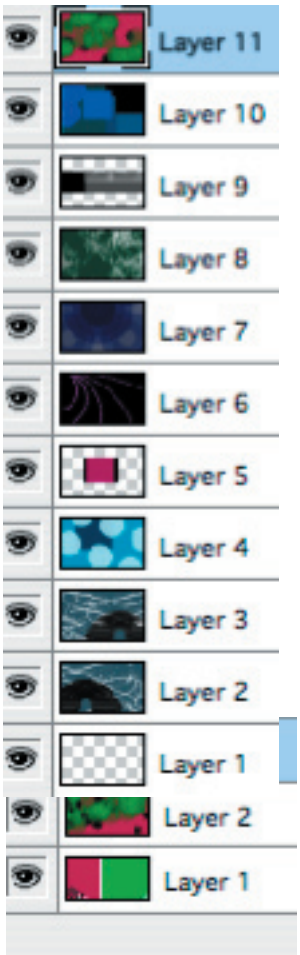









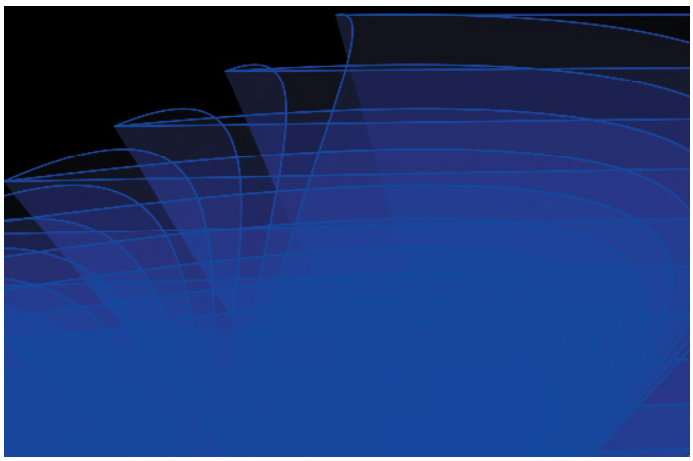

frameRate(30);
pulsing
ellipse(116, 0, radius*-.8,
(height-39,n,height/8,
#ffce00, 80);
(height-52,n,height/8,
(#fffb00, 80);
(height-65,n,height/8,
(#b2f60a, 80);
(height-78,n,height/8,width/8);
#6fdd13, 80);
(height-91,n,height/8,width/8);
(#14b82d, 80);
llipse(height-104,n,height/8,width/8);
fill(#00bc61, 80);
llipse(height-117,n,height/8,width/8);
(#00d398, 80);
ipse(height-130,n,height/8,width/
(#00ead4, 80);
ipse(height-143,n,height/8,width/
(#00d2fd, 80);
ipse(height-156,n,height/8,width/
(#00b9f2, 80);
ipse(height-169,n,height/8,width/
(#0095df, 80);
ipse(height-182,n,height/8,width/
(#0068c7, 80);
ipse(height-195,n,height/8,width/
(#0037a9, 80);
ipse(height-208,n,height/8,width/8);
fill(#200090, 80);
// ellipse(height-117,n,height/8,width/8);
fill(#67009e, 80);

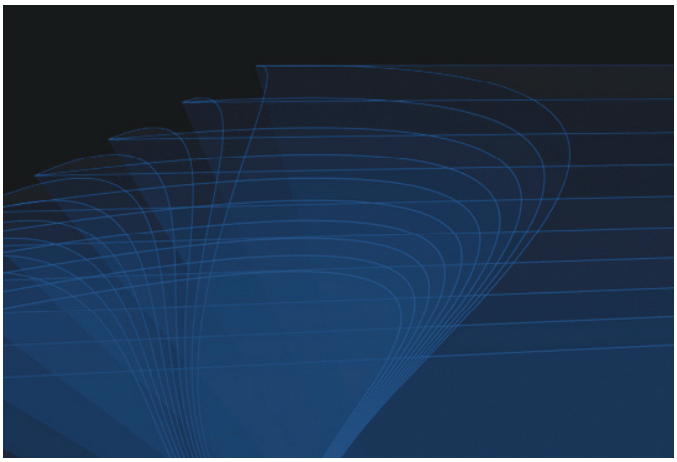

3g §
Untitled-3 @ 10
Doc: 248.
Windowis
Limb \
scale2 :

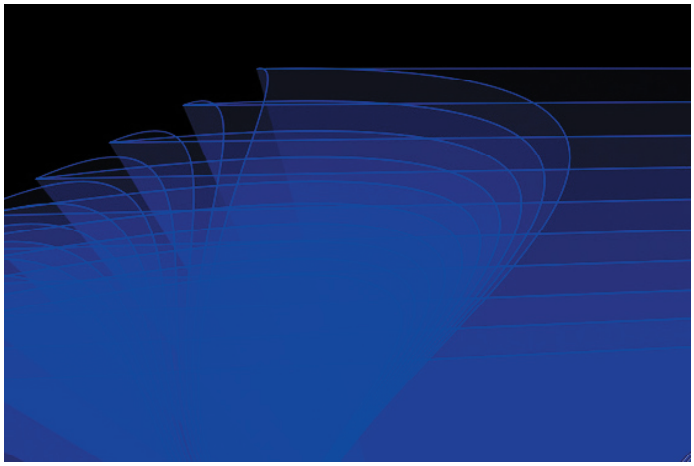

Untitled-1 @ 100% (La
float
100%
nmy = mouseY;
100%
Doc: 248.7K/248.7K

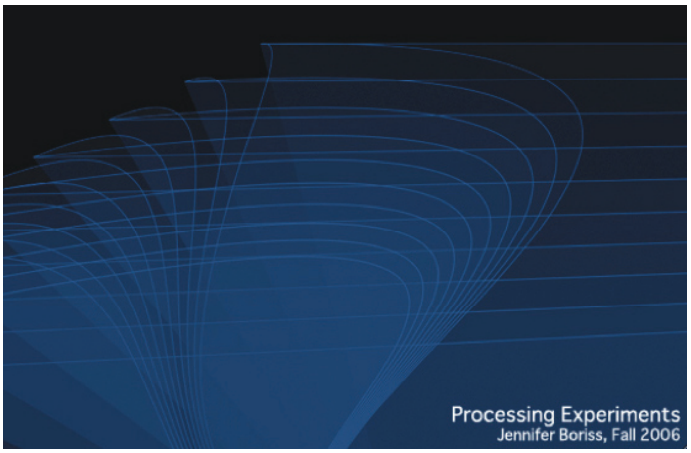

Layer 11
Layer 10
Layer 9
Layer 8
Layer 7
Layer 6
Layer 5
Layer 4
Layer 3
Layer 2
Layer 1
Layer 2
Layer 1

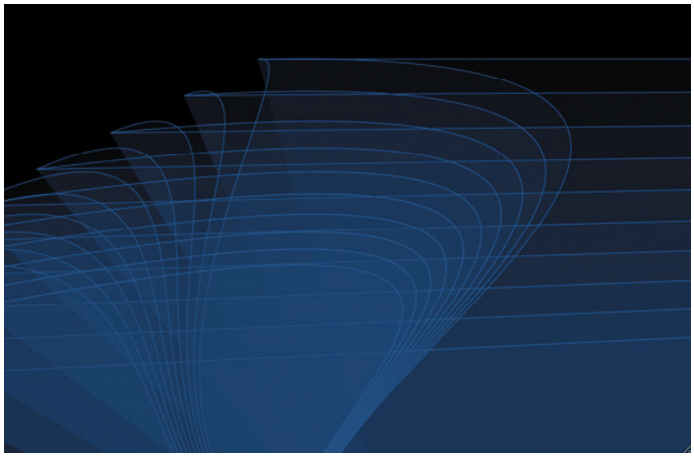

sketch_061214

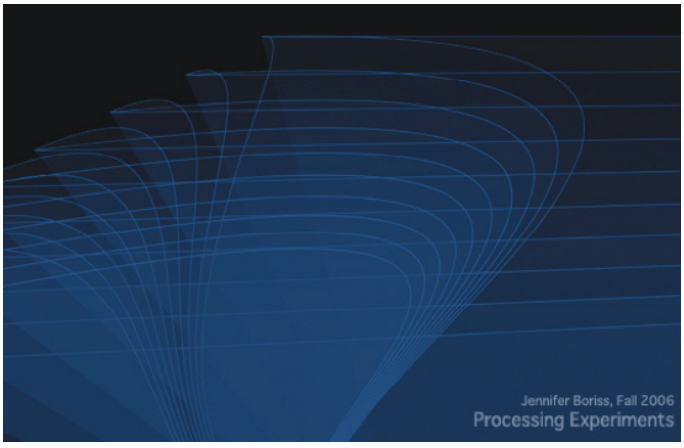







Processing Experiments
Jennifer Boriss, Fall 2006




Jennifer Boriss, Fall 2006
Processing Experiments