Creative Coding Workshop #1

Yuji Ogawa
(Hitotsubashi University / University of Ibadan)

What is Creative Coding?

Creative Coding:

Creative coding is a type of computer programming in which the goal is **to create something expressive** instead of something functional.

Why creative coding is worth to learn?

1. Coding without any visual/sound is **SOOOOOO boring**, at least for me.

2. The art of coding should exist not only for hard-core engineers, but also for **artists/designers/architects/ hobbyist**.

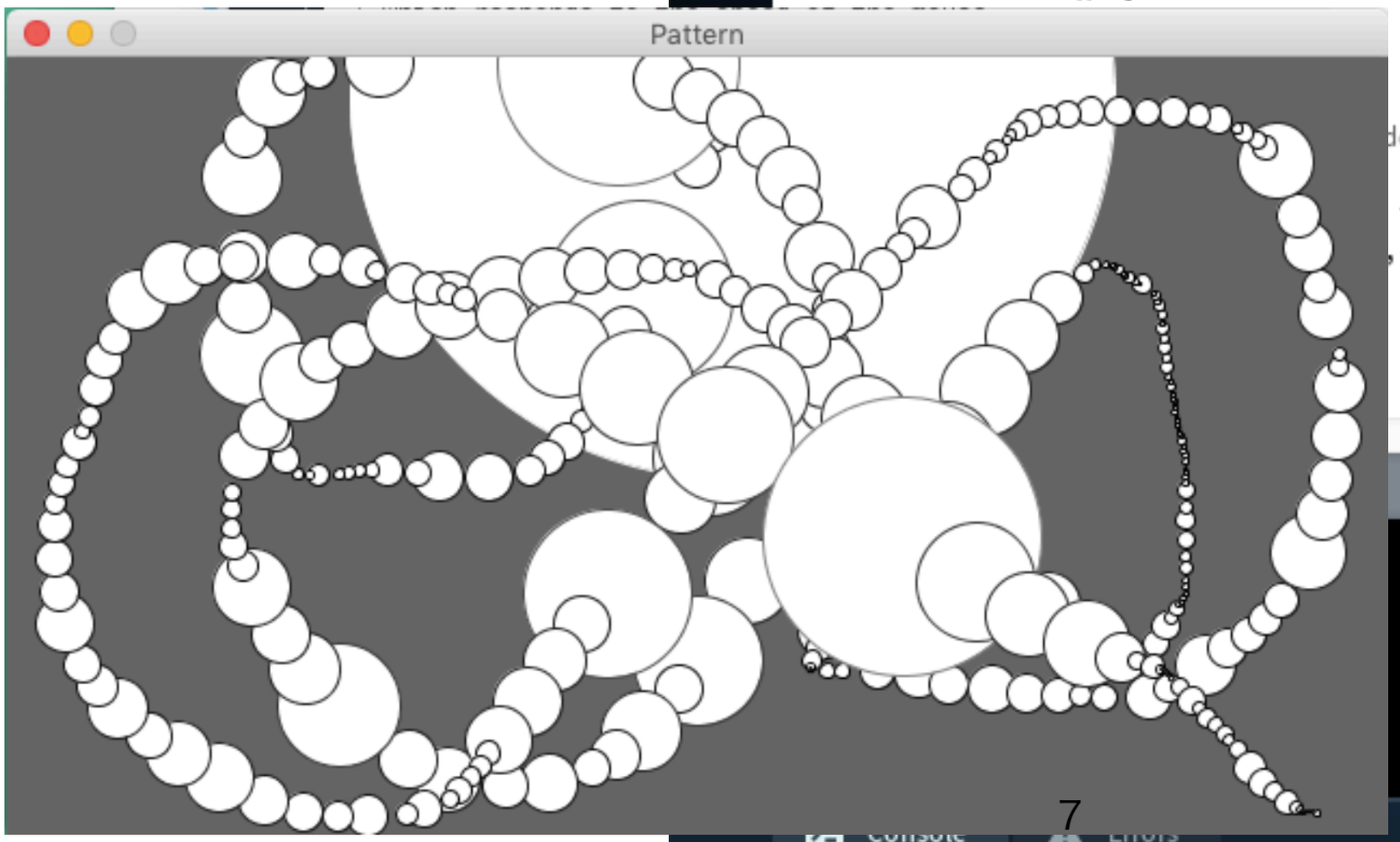
3. The expression that machine produces causes **different affection and impression** for us.

Popular tools for Creative Coding

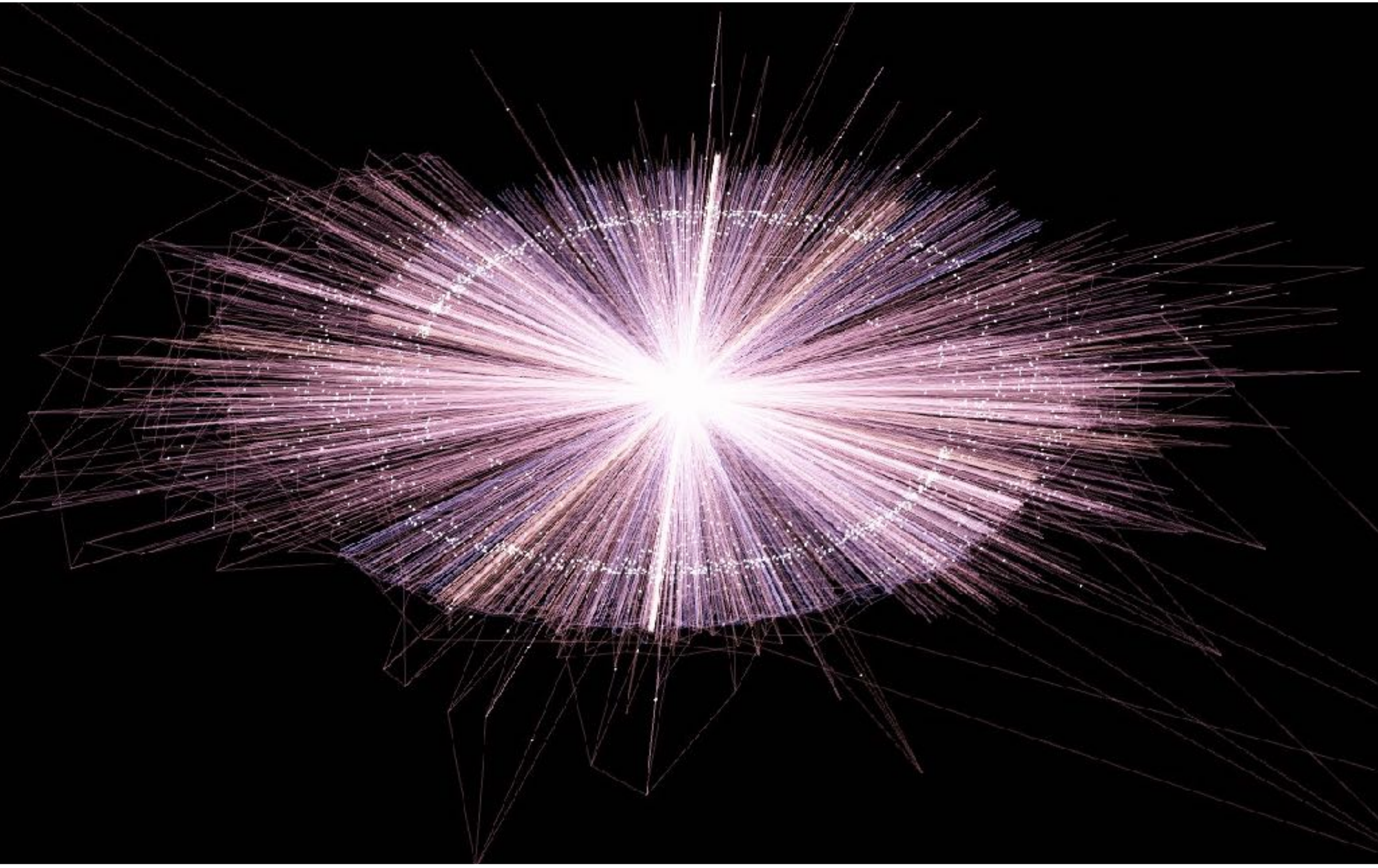
Processing
(2003– / Java)

An open-source graphical library and integrated development environment (IDE) / playground built for the electronic arts, new media art, etc.
The basic language is **Java**.

Processing



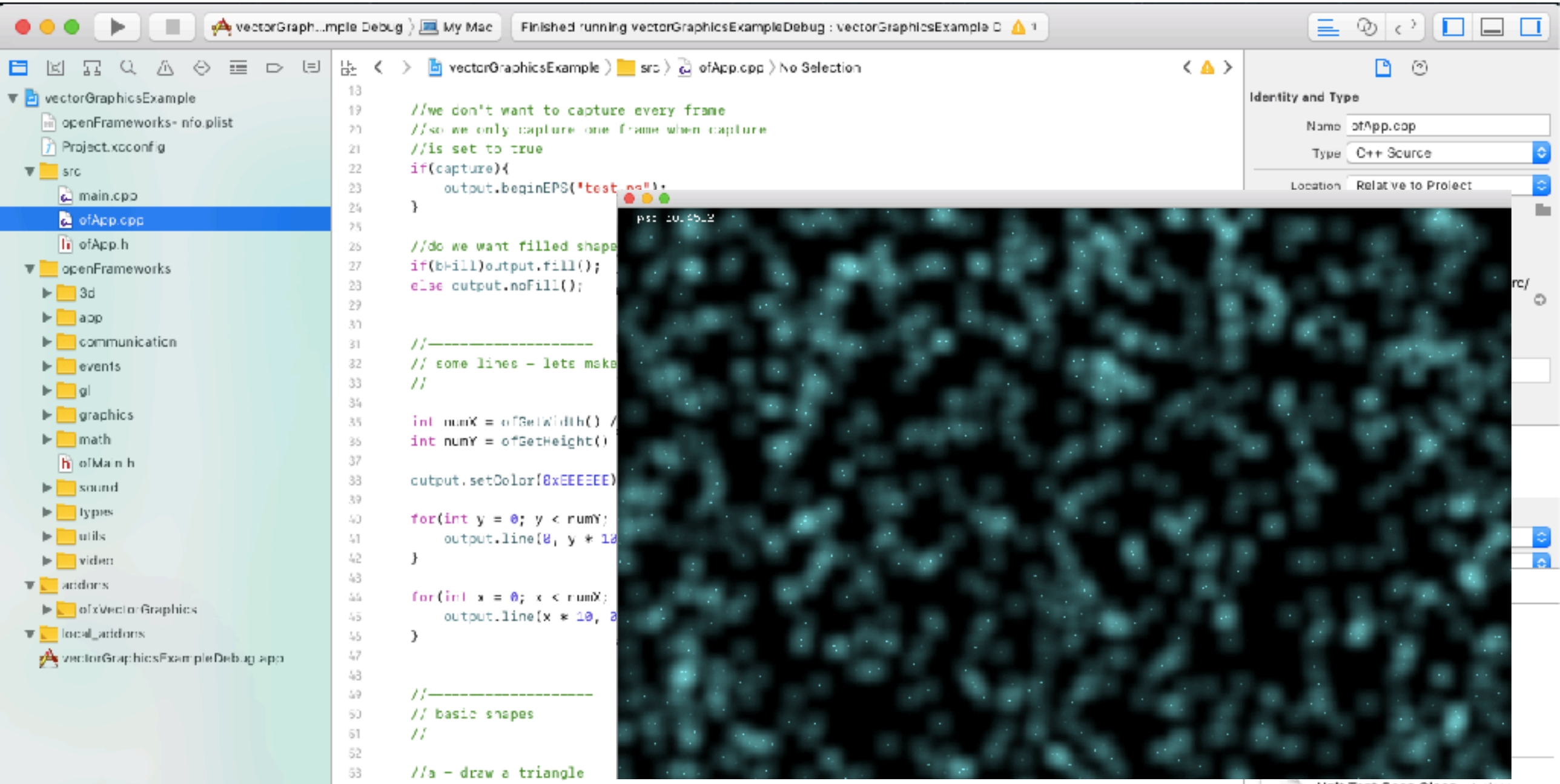

```
NoC_2_5 | Liquid | Mover

23      }
24
25      float m = 0.1*movers[i].mass;
26      PVector gravity = new PVector(0, m);
27      movers[i].applyForce(gravity);
28
29      movers[i].update();
30      movers[i].display();
31      movers[i].checkEdges();
32    }
33    fill(0);
34    text("click mouse to reset", 10, 30);
35  }
36
37
38  void mousePressed() {
```



openFrameworks
(2005– / C++)

An open source toolkit designed to assist the creative process by providing a simple and intuitive framework for experimentation. OpenFrameworks is written in **C++** and built on top of **OpenGL**.

OpenFrameworks



Cinder
(2010– / C++)

An open-source programming library designed to give the **C++** language advanced visualization abilities. Cinder, combined with the speed provided by C++, makes the library more appropriate **for heavily abstracted projects**, including art installations, commercial campaigns and other advanced animation work.

CubeMapping › My Mac | Finished running CubeMapping : CubeMapping | 1

CubeMapping › assets › env_map.vert › No Selection

- CubeMapping
  - assets
    - env_map_es2.frag
    - env_map_es2.vert
    - env_map.frag
    - env_map.jpg
    - env_map.vert
    - sky_box_es2.frag
    - sky_box_es2.vert
    - sky_box.frag
    - sky_box.vert
  - Blocks
  - Headers
    - Resources.h
    - CubeMapping_Prefix.pch
  - Source
    - CubeMappingApp.cpp
  - Resources
    - CinderApp.icns
    - Info.plist
  - Frameworks
    - IOKit.framework
    - IOSurface.framework
    - AVFoundation.framework
    - CoreMedia.framework
    - Linked Frameworks
    - Other Frameworks
  - Products
    - CubeMapping.app

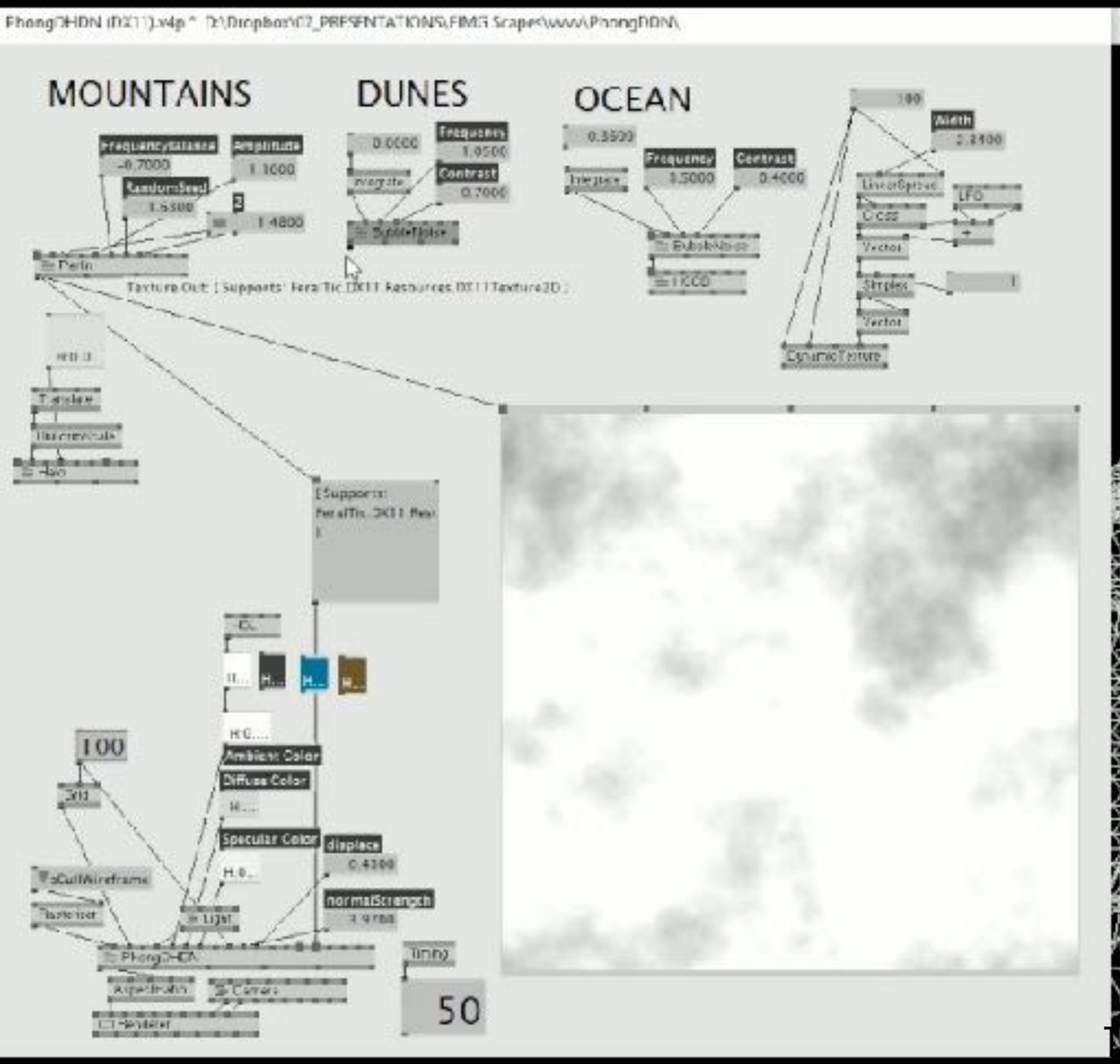
```
#version 150

uniform mat4    ciModelView;
uniform mat4    ciModelViewProjection;
uniform mat3    ciNormalMatrix;
uniform mat4    ciViewMatrix;
uniform mat4    ciViewMatrixInverse;

in vec4     ciPosition;
in vec3     ciNormal;

out highp vec3  NormalWorldSpace;
out highp vec3  EyeDirWorldSpace;

void main( void )
{
    vec4 positionViewSpace = ciModelView * ciPosit
    vec4 eyeDirViewSpace = positionViewSpace   vec
    EyeDirWorldSpace = vec3( ciViewMatrixInverse *
    vec3 normalViewSpace = ciNormalMatrix * ciNorma
    NormalWorldSpace = normalize( vec3( vec4( norma
    gl_Position = ciModelViewProjection * ciPositi
}
```

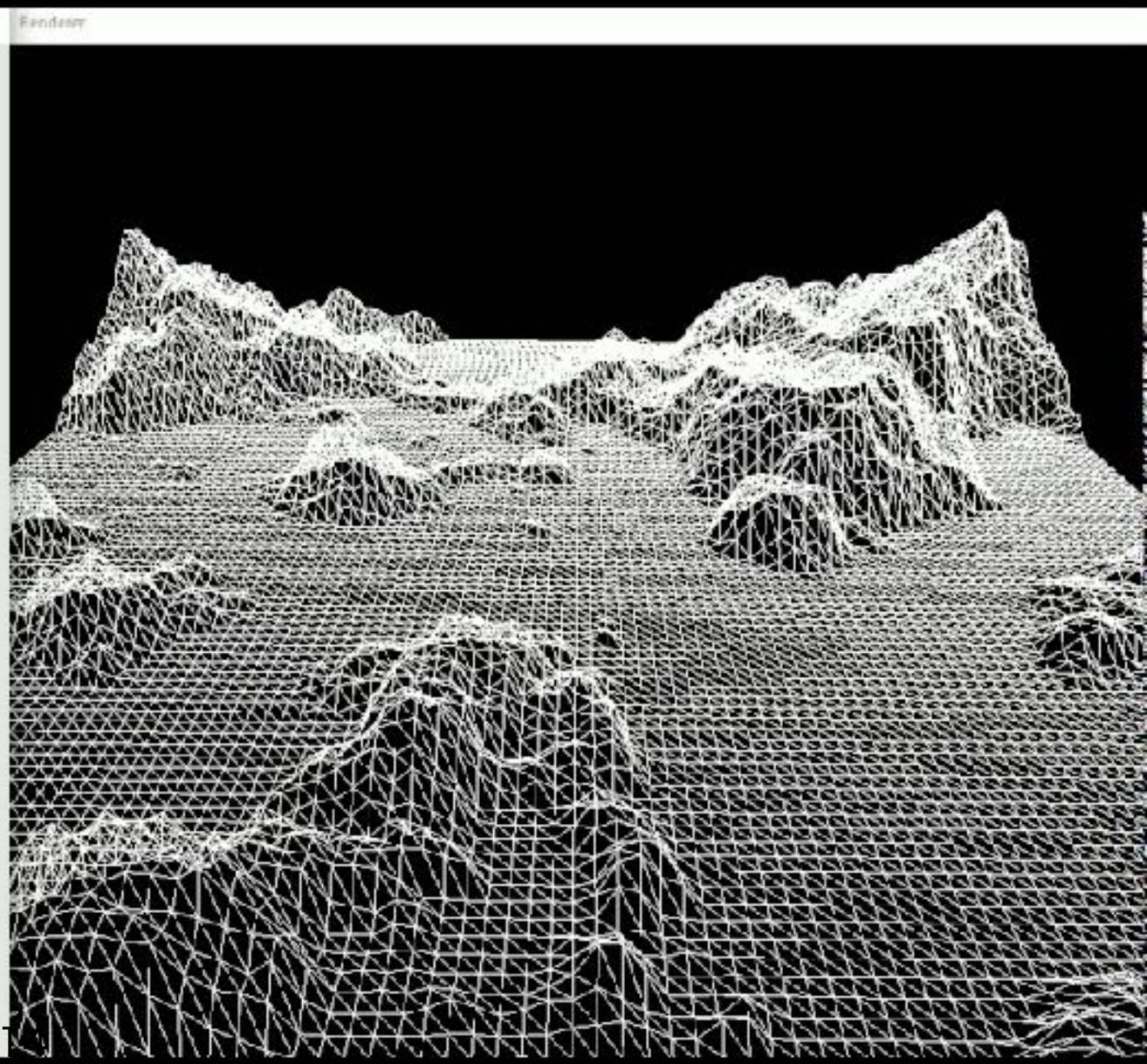
vvvv
(1998- / dataflow programming)

a general purpose toolkit with a special focus on **real-time video synthesis** and programming large media environments with physical interfaces, real-time motion graphics, audio and video.
It only runs on **Windows OS.**

VVVV

A MULTIPURPOSE TOOLKIT

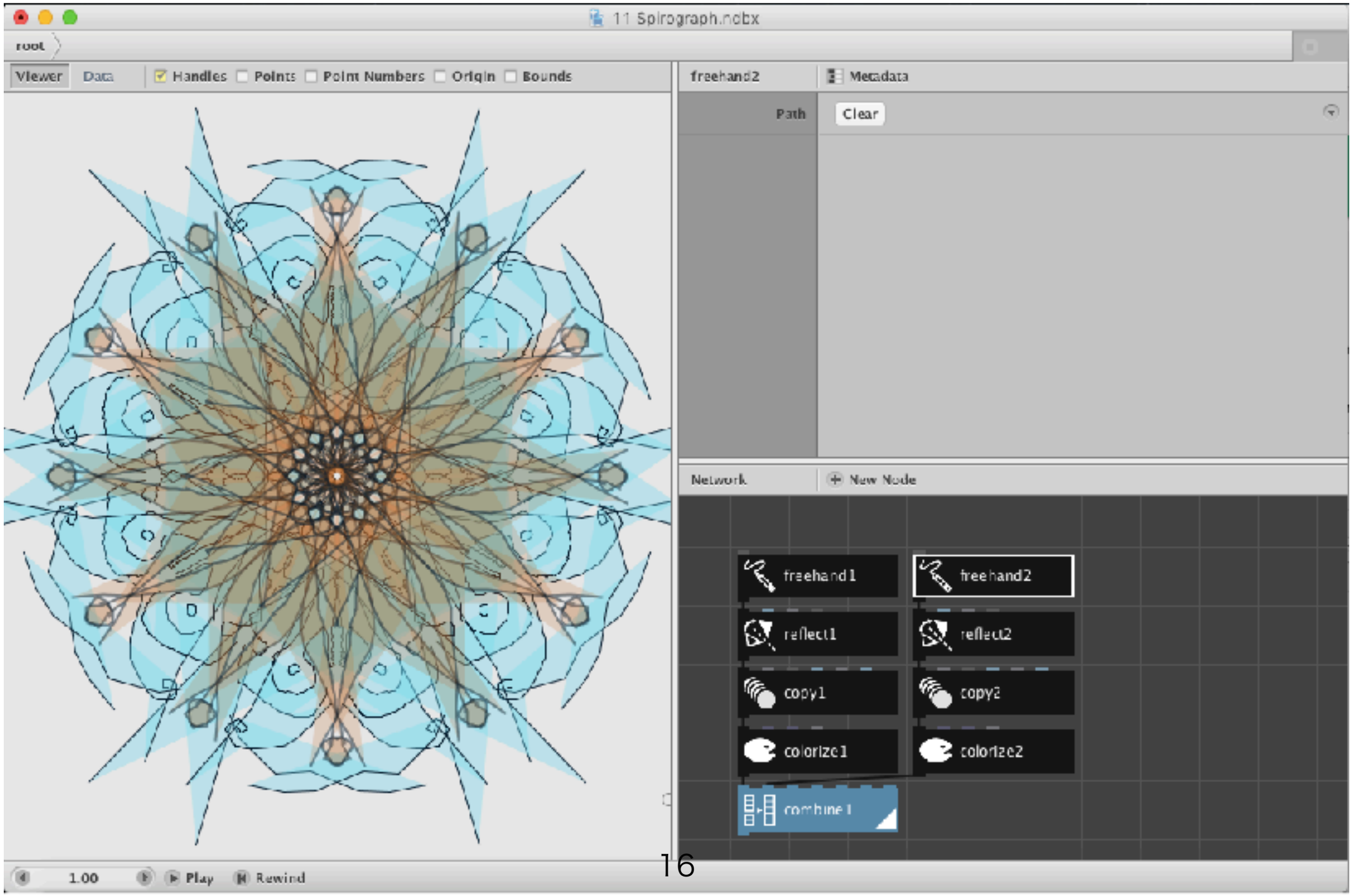




NodeBox
(1998– / Python or dataflow)

Using its node-based interface, NodeBox makes **generative design easily without coding.** The result is like Processing.

NodeBox


11 Spirograph.ndbx
root
Viewer
Data
Handles
Points
Point Numbers
Origin
Bounds
freehand2
Metadata
Path
Clear
Network
New Node
freehand1
freehand2
reflect1
reflect2
copy1
copy2
colorize1
colorize2
combine1
1.00
Play
Rewind

supercollider
(1998– / supercollider)

An language and IDE for **real-time audio synthesis** and algorithmic composition, which provides a framework for acoustic research, algorithmic music, interactive programming and **live coding.**

SuperCollider

Modal Space.scd (~/Documents/SuperCollider/examples/demonstrations) - SuperCollider IDE

Modal Space.scd

```
(
// modal space
// mouse x controls discrete pitch in dorian mode
var scale, buffer;
scale = FloatArray[0, 2, 3.2, 5, 7, 9, 10]; // dorian scale
buffer = Buffer.alloc(s, scale.size,1, {|b| b.setnMsg(0, scale) });
{
    var mix;
    mix =
    // lead tone
    SinOsc.ar(
        (
            DegreeToKey.kr(
                buffer.bufnum,
                MouseX.kr(0,15),        // mouse indexes into scale
                12,                     // 12 notes per octave
                1,                      // mul = 1
                72                      // offset by 72 notes
            )
            + LFNoise1.kr([3,3], 0.04)  // add some low freq stereo detuning
        ).midicps,                      // convert midi notes to hertz
        0,
        0.1)

    // drone 5ths
    + RLPF.ar(LFPulse.ar([48,55].midicps, 0.15),
        SinOsc.kr(0.1, 0, 10, 72).midicps, 0.1, 0.1);

    // add some 70's euro-space-rock echo
    CombN.ar(mix, 0.31, 0.31, 2, 1, mix)
}.play
)
```

Help browser — Home — Find in page...

Browse | Search | Indexes ▼ | TOC ▼

The stuff within the curly brackets is what will get executed each time you reuse, or evaluate the Function. Note that this is written like an equation, i.e. f = {...}. This is not an equation in the mathematical sense, it's what's called an assignment. Basically it allows me to name the Function I've created, by storing it in a variable called f. A variable is just a name representing a slot in which we can store things, such as a Function, a number, a list, etc. Execute the following lines one at a time and watch the post window:

```
f = { "Function evaluated".postln; };
f;
```

Both times it should say 'a Function'. Now whenever we want to refer to our Function we can just use the letter f. That's in fact what makes it reusable! Otherwise we'd need to type the Function in every time.

So how do we reuse it? Execute the following lines one at a time and watch the post window:

```
f = { "Function evaluated".postln; };
f.value;
f.value;
f.value;
```

Our Function is an object, (i.e a thing that does something or represents

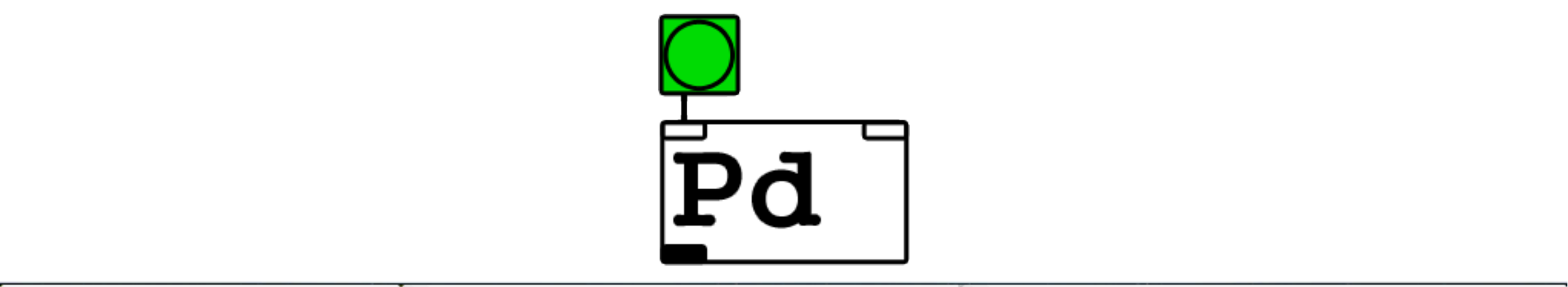
Post window — Auto Scroll

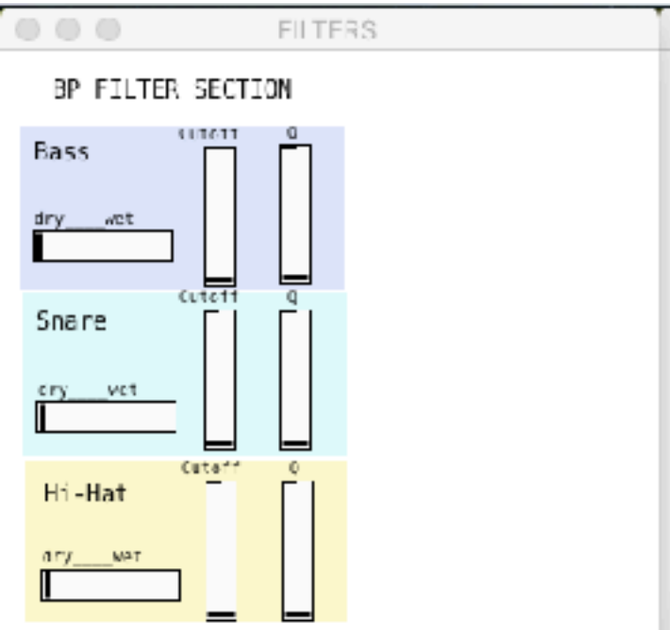
```
Shared memory server interface initialized
-> Synth('temp__0' : 1000)
-> Synth('temp__1' : 1001)
-> Synth('temp__2' : 1002)
-> Synth('temp__3' : 1003)
-> Synth('temp__4' : 1004)
-> an EventStreamPlayer
-> Synth('temp__5' : 1036)
```

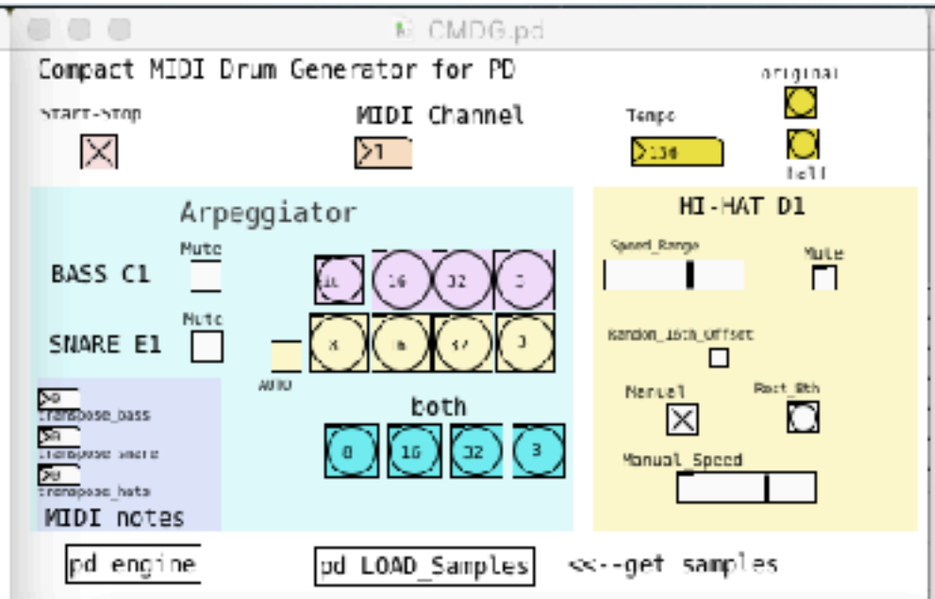
Interpreter: Active Server: 0.86% 1.37% 34u 1s 2g 112d 0.0dB

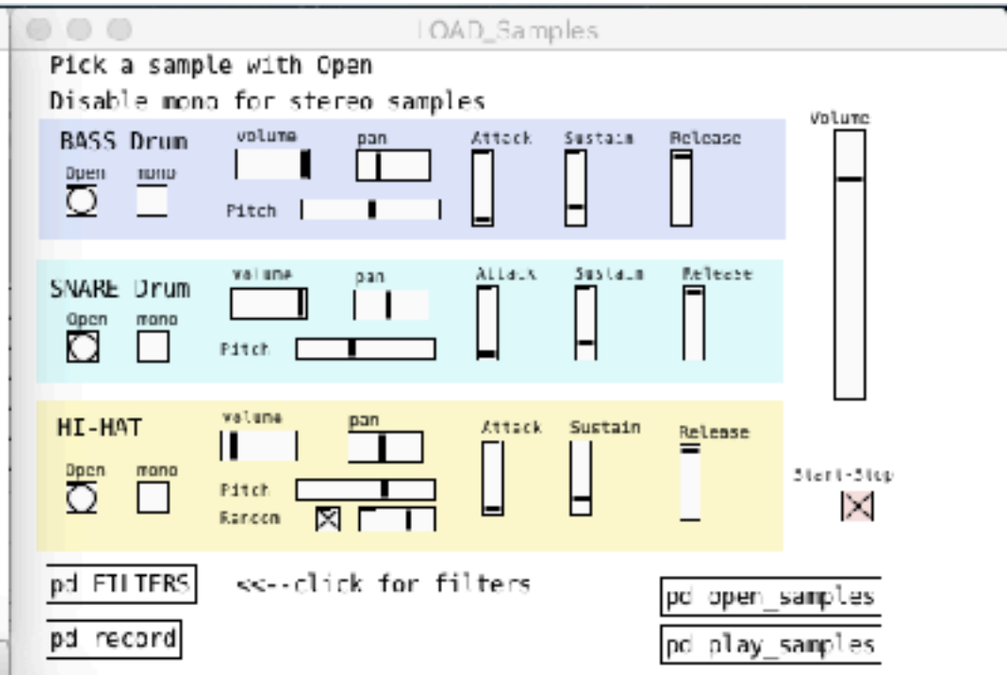
Pure-Data/Purr-Data
(1996– / dataflow programming)

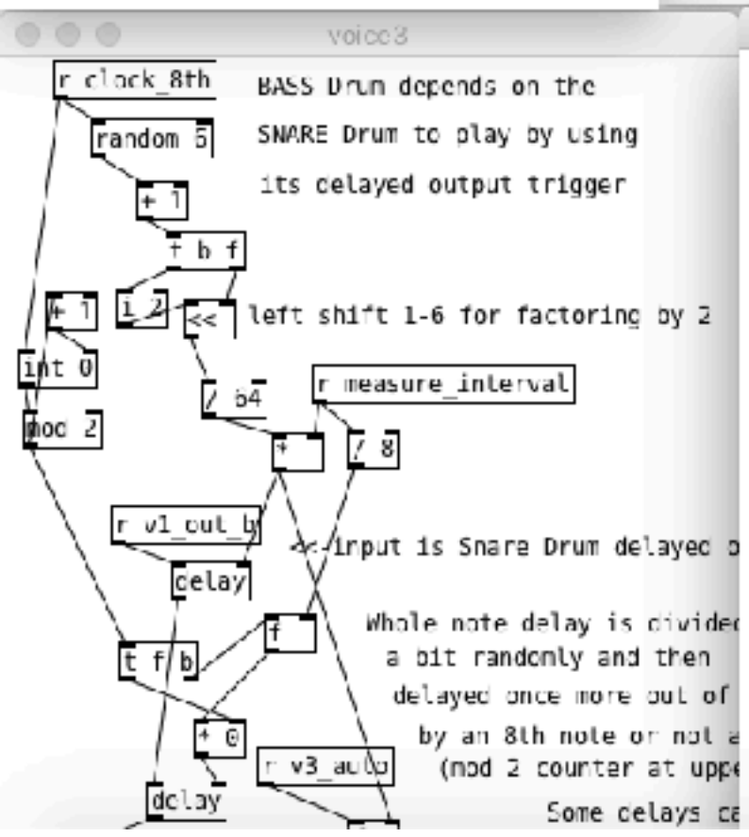
A visual programming language developed by Miller Puckette, for creating **interactive computer music** and multimedia works.
→Non-opensource version is called **Max**.

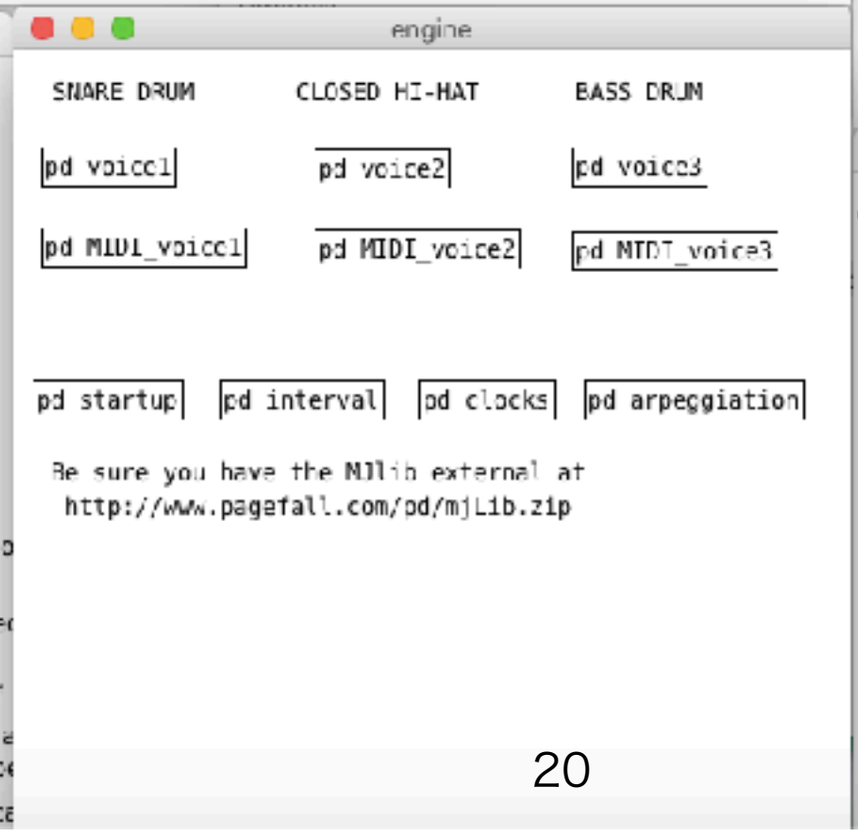

Pd

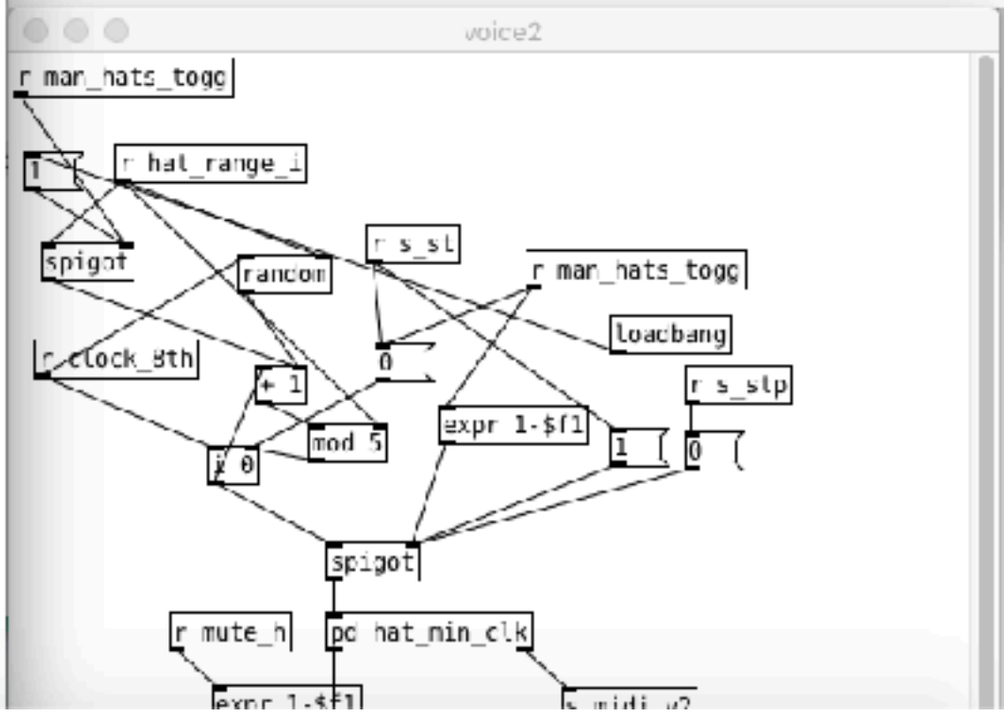

FILTERS
BP FILTER SECTION
Bass
Cutoff
Q
dry
wet
Snare
Cutoff
Q
dry
wet
Hi-Hat
Cutoff
Q
dry
wet


CMDG.pd
Compact MIDI Drum Generator for PD
START-STOP
MIDI Channel
Tempo
original
Arpeggiator
BASS C1
Mute
SNARE E1
Mute
Auto
both
MIDI notes
HI-HAT D1
Speed_Range
Mute
Random_pitch_offset
Manual
Root_Pch
Manual_Speed
pd engine
pd LOAD_Samples
<<--get samples


LOAD_Samples
Pick a sample with Open
Disable mono for stereo samples
BASS Drum
volume
pan
Attack
Sustain
Release
Open
mono
Pitch
SNARE Drum
Open
mono
Pitch
HI-HAT
Open
mono
Pitch
Random
volume
pan
Attack
Sustain
Release
Volume
Start-Stop
pd FILTERS
<<--click for filters
pd record
pd open_samples
pd play_samples


voice3
r clock_8th
BASS Drum depends on the
random 5
SNARE Drum to play by using
+ 1
its celayed output trigger
t b f
+ 1
1 2
<<
left shift 1-6 for factoring by 2
int 0
/ 64
r measure_interval
mod 2
+
/ 8
r v1_out_b
<--input is Snare Drum delayed
celay
Whole note delay is divide
t f b
a bit randomly and then
delayed once more out of
+ 0
by an 8th note or not
r v3_audio
(mod 2 counter at upp
delay
Some celays ca


engine
SNARE DRUM
CLOSED HI-HAT
BASS DRUM
pd voice1
pd voice2
pd voice3
pd MIDI_voice1
pd MIDI_voice2
pd MIDI_voice3
pd startup
pd interval
pd clocks
pd arpeggiation
Be sure you have the Mllib external at
http://www.pagefall.com/pd/mjLib.zip


voice2
r mar_hats_togg
r hat_range_i
spigot
random
r s_sl
r mar_hats_togg
loadbang
r clock_8th
+ 1
0
r s_slp
mod 5
expr 1-$f1
spigot
r mute_h
pd hat_min_clk

Examples of Media Art (deleted for IP matter)

Processing's system

Basic syntax

- Mainly, Processing uses **Java** language to code.
- There are several modes which allows you to code with different language: processing.py, processing.js, p5.js…
- Just like **Arduino**, it is constituted with two main functions: setup() and draw().

void setup()

- executed **one time** when you run the code.

e.g.
Instantiation of class (c = new PVector(v);)

size(), background(), noLoop(), blendMode(),
frameRate(), fill(), stroke(), strokeWeight()...etc.

void draw()

- executed at every frame (default=30)

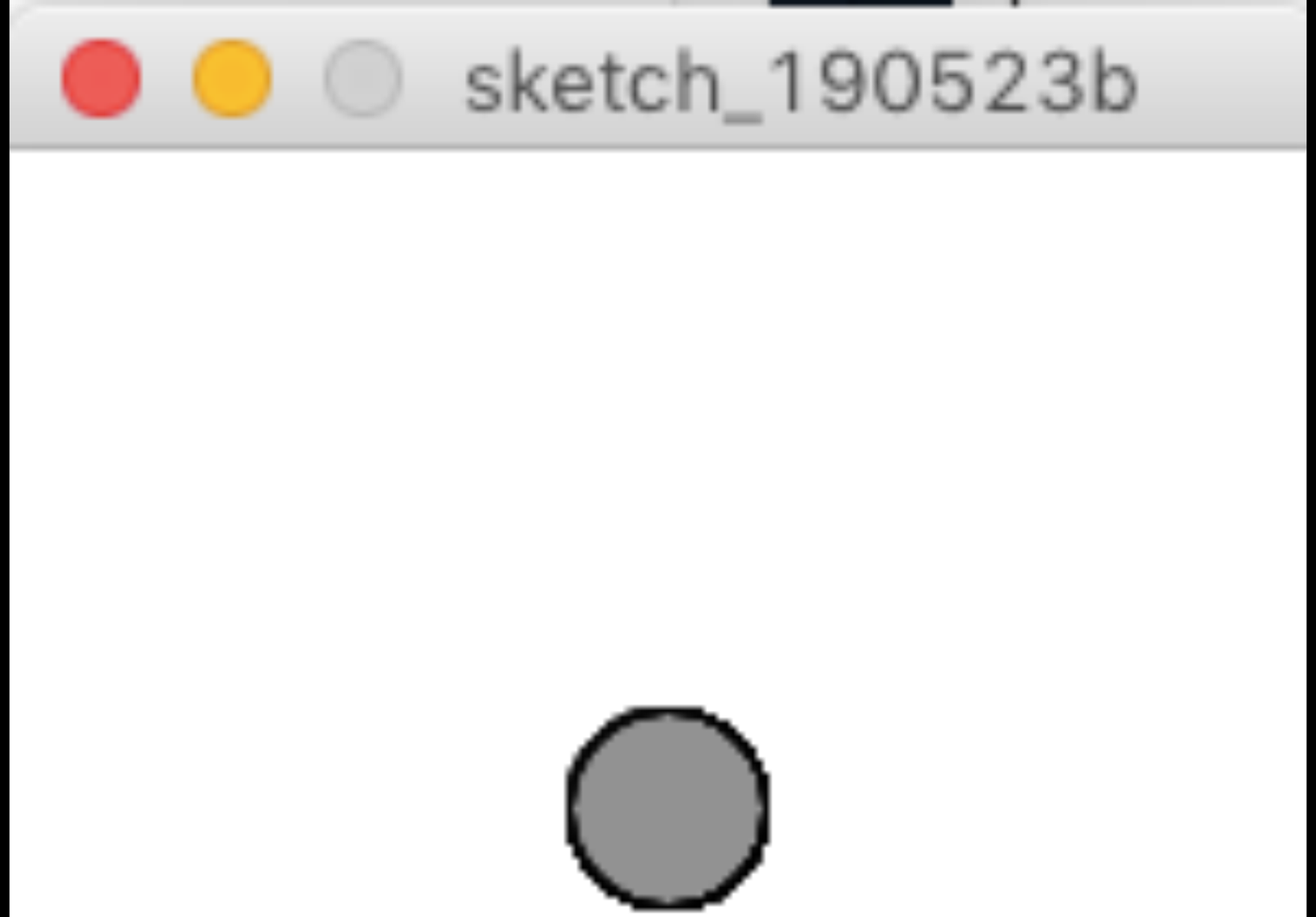
ellipse(), rect(), line(), point(), box(), sphere()...etc

- If you want to make it run only one time, use noLoop().

Example 1:
Simple Motion Simulation

Example #1.1

```
void setup () {
 size(200, 200); // define size of window
 background(255); //background draw
 stroke(1); //color of stroke
 strokeWeight(1); //width of stroke
 fill(140, 20); //define the color of object
}

void draw () {
 ellipse(width/2, height/2, 30, 30);
}
```

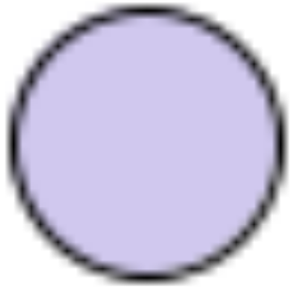

sketch_190523b

Example #1.2

```
float xpos = 0;
float ypos = 0;
float xspeed = 8;
float yspeed = 6;

void setup () {
  size(200, 200);
  frameRate(30);
  fill(100, 40, 200,
  70);
  smooth();
}
```

```
void draw () {
  background(255);
  ellipse(xpos, ypos, 30, 30);
  xpos = xpos + xspeed;
  ypos = ypos + yspeed;
  if (xpos > 200 || xpos < 0) {
    xspeed = xspeed * -1;
  }
  if (ypos > 200 || ypos < 0) {
    yspeed = yspeed * -1;
  }
}
```

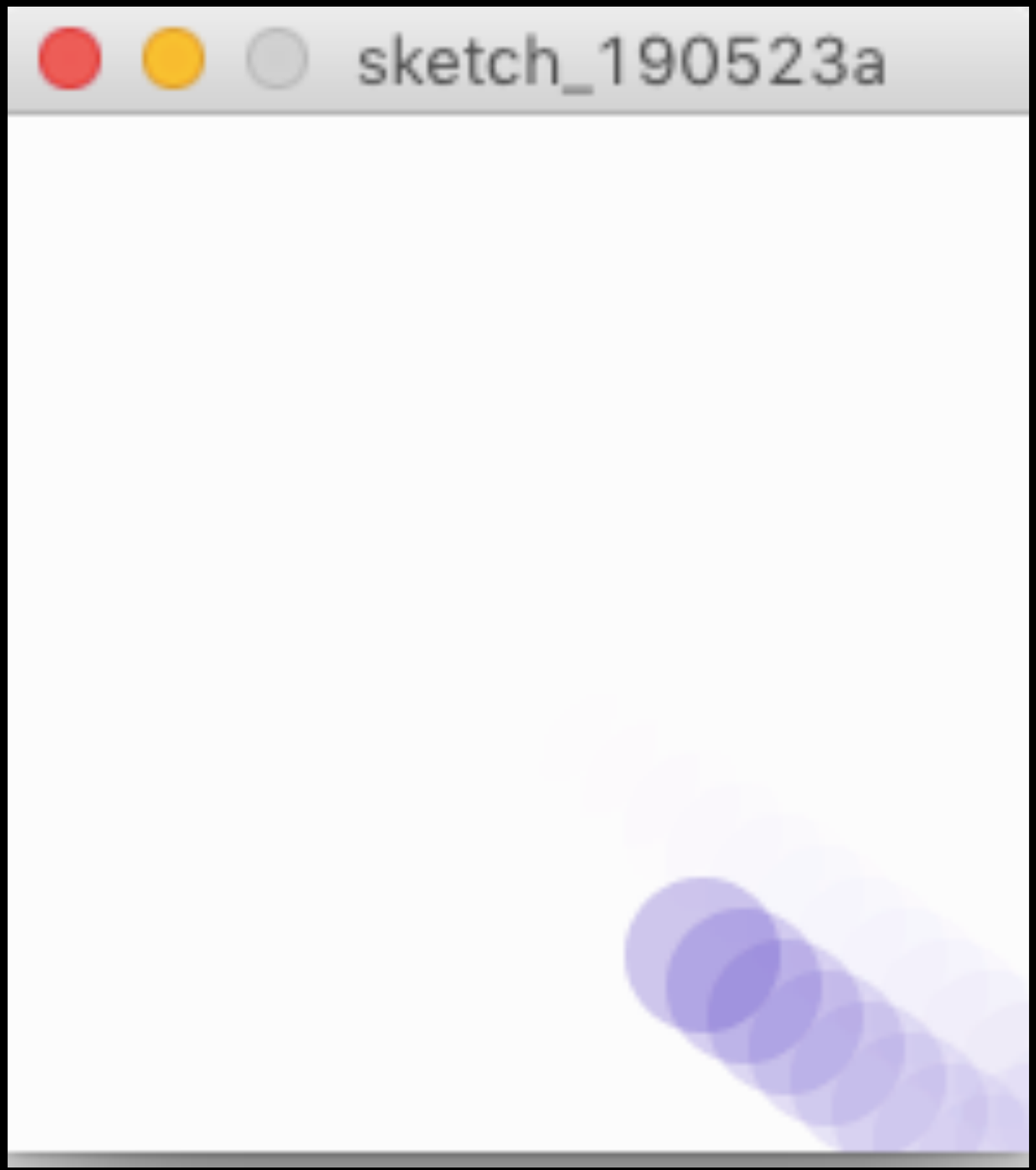

sketch_190523c



Example #1.3

```
float xpos = 0;
float ypos = 0;
float xspeed = 8;
float yspeed = 6;
float radius = 30;

void setup () {
  size(200, 200);
  frameRate(30);
  fill(100, 40, 200,
  70);
  smooth();
  noStroke();
}
```

```
void draw () {
  background(255);
  fill(255, 40)
  rect(0, 0, width, height);
  fill(100, 40, 200, 70);

  ellipse(xpos, ypos, radius,
radius);
  xpos = xpos + xspeed;
  ypos = ypos + yspeed;
  if (xpos > width || xpos < 0) {
    xspeed = xspeed * -1;
  }
  if (ypos > height || ypos < 0) {
    yspeed = yspeed * -1;
  }
}
```


sketch_190523a

Example #1.4 (1)

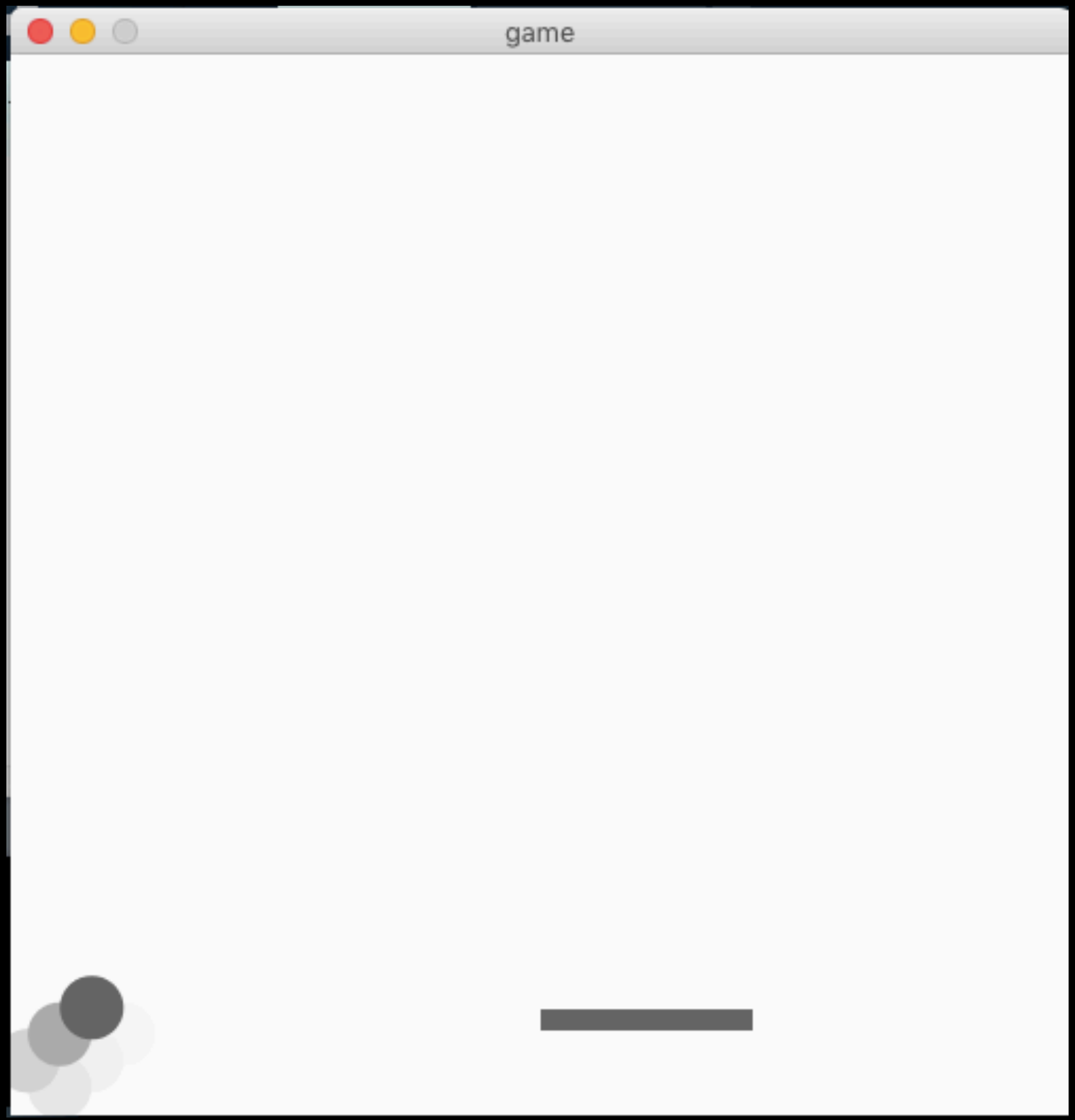
```
float xpos = 100.0;
float ypos = 100.0;
float radius = 15.0;
float speedX = 8.0;
float speedY = 6.0;
int directionX = 1;
int directionY = -1;
float racket = 250;

void setup() {
  size(500, 500);
  smooth();
  noStroke();
  ellipseMode(RADIUS);
  frameRate(60);
}
```

Example #1.4 (2)

```
void draw() {
  fill(255, 120);
  rect(0, 0, width, height);
  fill(100);
  ellipse(xpos, ypos, radius, radius);

  xpos += speedX * directionX;
  if ((xpos > width-radius) || (xpos < radius)) {
    directionX = -directionX;
  }
  ypos += speedY * directionY;
  if ((ypos > height-radius) || (ypos < radius)) {
    directionY = -directionY;
  }
[…]
```

Example #1.4 (3)

```
rect (racket, 450, 100, 10);//drawing racket
  if (keyPressed) {
    if (keyCode == RIGHT) {
      racket = racket + 20;
      if (racket>width-100)racket = width-100;
    }
    if (keyCode == LEFT) {
      racket = racket - 20;
      if (racket<0)racket = 0;
    }
  }
  if ((racket < xpos) && (racket+100 > xpos) && (ypos <
450) && (ypos > 440) ) {
    directionY = -directionY;
  }
  speedX+=0.001;
  speedY+=0.001;
}
```


game

Example 2:
PVector & OOP
Motion simulation

class PVector

- This class is already installed.
  - A PVector contains **(x, y) coordinate**. By using two,

it allows to simulate motion.

  - function: **x.add(), x.sub(), x.mult(), x.div(),**

x.limit()…etc

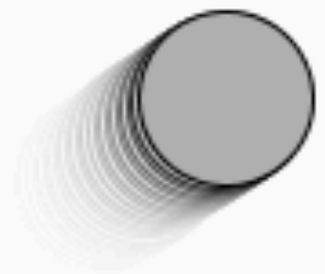
Example #2.1 (1)

```
class Mover {

  PVector location;
  PVector velocity;

  Mover() {
    location = new
PVector(random(width),
random(height));
    velocity = new
PVector(random(-2, 2),
random(-2, 2));
  }

  void update() {
    location.add(velocity);
  }

  void display() {
    stroke(0);
    fill(175);
    ellipse(location.x,
location.y, 48, 48);
  }

  void checkEdges() {
    if (location.x > width) {
      location.x = 0;
    } else if (location.x < 0) {
      location.x = width;
    }
    if (location.y > height) {
      location.y = 0;
    } else if (location.y < 0) {
      location.y = height;
    }
  }
}
```

Example #2.1 (2)

```
Mover mover;

void setup() {
  size(640, 360);
  mover = new Mover();
}

void draw() {
  fill(255, 40);
  noStroke();
  rect(0, 0, width, height);
  mover.update();
  mover.checkEdges();
  mover.display();
}
```

ex3



Example #2.2 (1)

```
class Mover {

  PVector location;
  PVector velocity;
  PVector acceleration;
  float topspeed;

  Mover() {
    location = new
PVector(random(width), random(height));
    velocity = new PVector(random(-2,
2), random(-2, 2));
    topspeed = 10;
  }

  void update() {
    PVector mouse = new PVector(mouseX,
mouseY);
    PVector dir = PVector.sub(mouse,
location);

    dir.normalize();
    dir.mult(0.8);
    acceleration = dir;

    velocity.add(acceleration);
    velocity.limit(topspeed);
    location.add(velocity);
  } void display() {
    stroke(0);
    fill(175);
    ellipse(location.x, location.y, 48, 48);
  }

  void checkEdges() {
    if (location.x > width) {
      location.x = 0;
    } else if (location.x < 0) {
      location.x = width;
    }
    if (location.y > height) {
      location.y = 0;
    } else if (location.y < 0) {
      location.y = height;
    }
  }
}
```

Example #2.2 (2)

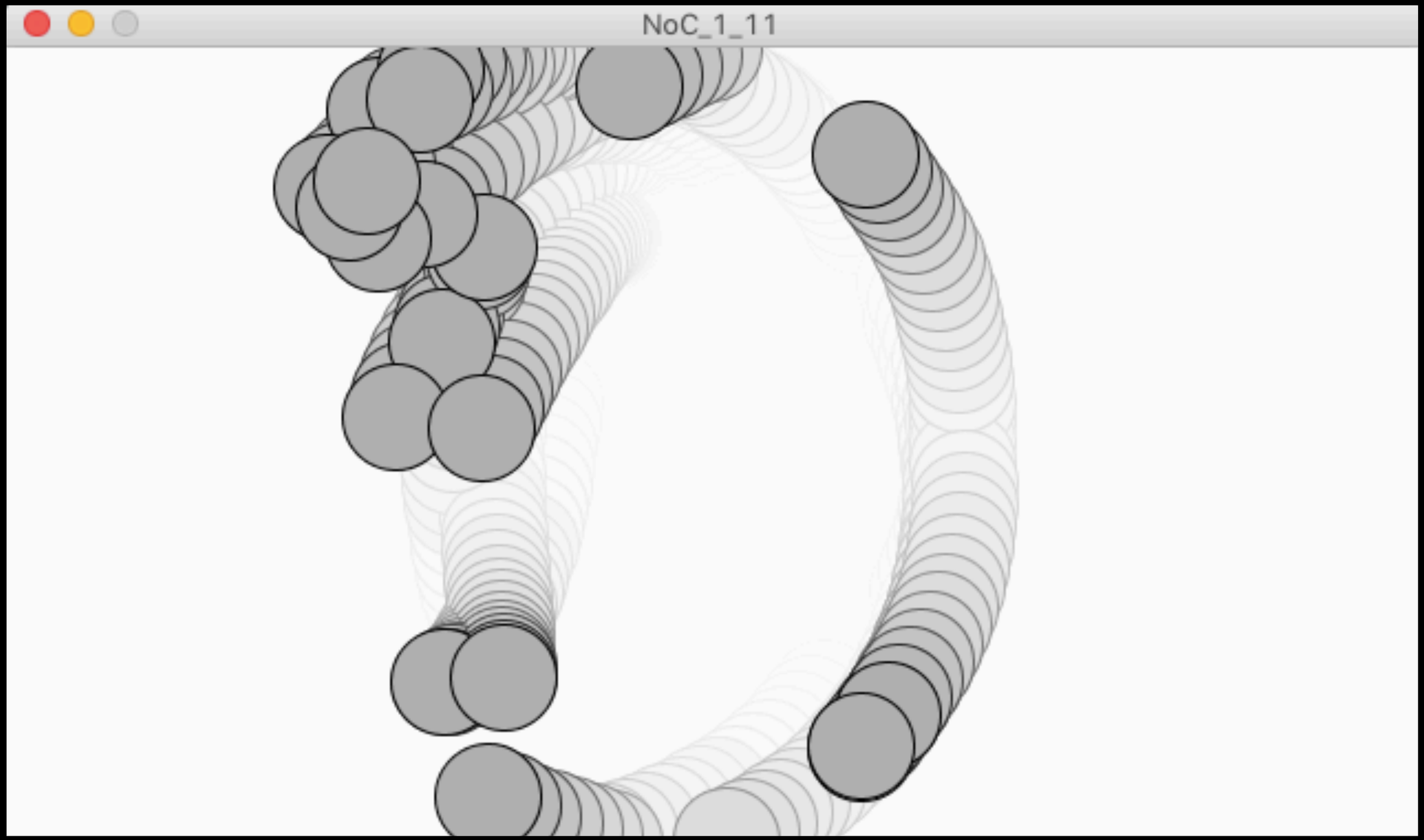
```
Mover mover;

void setup() {
  size(640, 360);
  mover = new Mover();
}

void draw() {
  noStroke();
  fill(255, 40);
  rect(0, 0, width, height);

  mover.update();
  mover.checkEdges();
  mover.display();
}
```

Example #2.3

```
Mover[] movers = new Mover[20];

void setup() {
  size(640, 360);
  for (int i = 0; i < movers.length; i++) {
    movers[i] = new Mover();
  }
}

void draw() {
  //background(255);
  noStroke();
  fill(255, 40);
  rect(0, 0, width, height);
  for (int i = 0; i < movers.length; i++) {
    movers[i].update();
    movers[i].checkEdges();
    movers[i].display();
  }
}
```

Example #2.4 (1)

```
Mover m;
Attractor a;

void setup(){
  size(640, 360);
  //fullScreen();
  background(255);
  smooth();
  frameRate(60);
  m = new Mover();
  a = new Attractor();
}

void draw(){
  //background(255);
  /*noStroke();
  fill(255, 40);
  rect(0,0, width, height);
  */
  PVector force = a.attract(m);
  m.applyForce(force);
  m.update();

  a.display();
  m.display();
}
```


NoC_1_11

Example #2.4 (1)

```
class Mover {

  PVector location;
  PVector velocity;
  PVector acceleration;

  float mass;

  Mover() {
    location = new PVector(400, 50);
    velocity = new PVector(1, 0);
    acceleration = new PVector(0, 0);
    mass = 1;
  }

  void applyForce(PVector force) {
    PVector f = PVector.div(force,
mass);
    acceleration.add(f); //kasokudo +
force
  }

void update() {
    velocity.add(acceleration);
    location.add(velocity);
    acceleration.mult(0);//clearing the
acceleration each time
  }

  void display() {
    stroke(2);
    fill(215);
    ellipse(location.x, location.y, 16,
16);
  }

  void checkEdges() {
    if (location.x > width) {
      location.x = 0;
    } else if (location.x < 0) {
      location.x = width;
    }
if (location.y > height) {
      velocity.y *= -1;
      location.y = height;
    }
  }
}
```

Example #2.4 (2)

```
class Attractor {
  float mass;
  PVector location;
  float G;

  Attractor() {
    location = new
PVector(width/2, height/2);
    mass = 20;
    G = 0.4;
  }

  void display() {
    stroke(0);
    fill(175, 200);
    ellipse(location.x,
location.y, mass*2, mass*2);
  }

  PVector attract(Mover m) {
    PVector force =
PVector.sub(location,
m.location);
    float distance =
force.mag();
    distance =
constrain(distance, 5.0, 25.0);
    force.normalize();
    float strength = (G * mass *
m.mass) / (distance * distance);
    force.mult(strength);
    return force;
  }
}
```

Example #2.4 (3)

```
Mover m;
Attractor a;

void setup(){
  size(640, 360);
  background(255);
  m = new Mover();
  a = new Attractor();
}

void draw(){
   /*noStroke();
  fill(255, 40);
  rect(0,0, width, height);
  */
  PVector force = a.attract(m);
  m.applyForce(force);
  m.update();

  a.display();
  m.display();
}
```

example2_4