

# **today**

Due: Ex 3

Random and Transform + Generative Design

Introduce Midterm: Due Wed, Feb 10

Student Presentations

Reading: Ch 4-6 on p.60, optional Ch 14

# **Wednesday, Feb 3**

Due: Component 1, 2, 3 of Midterm project

Push Pop

Lab to work on midterm project

Student Presentations

# **Random**

“... the physical world is usually idiosyncratic. ... We can simulate the unpredictable qualities of the world by generating random numbers.” - Casey Reas

# random ( ) function

The random() function always returns a **floating**-point value, so be sure the variable on the left side of the assignment operator (=) is float like this:

To generate a pseudo random number between 0 and high and assign it to f

```
float f;  
f = random(high);
```

To generate a pseudo random number between low and high

```
f = random(low, high);
```

```
void draw ( ) {  
    float r= random(0, 100);  
}
```

a variable named r will randomly use a value between 0 and 100, each time the program draws.

check out: sketch\_4\_7\_random\_circle.pde

```
//Basic Random Circle

void setup() {
  size(640, 360);
}

void draw() {
  float x=random(0, width);
  float y=random(0, height);

  ellipse(x, y, 100, 100);
}
```

check out: sketch\_4\_7\_random\_circle.pde

```
//Basic Random Circle

void setup() {
  size(640, 360);
}

void draw() {
  float x=random(0, width);
  float y=random(0, height);

  ellipse(x, y, 100, 100);
}
```

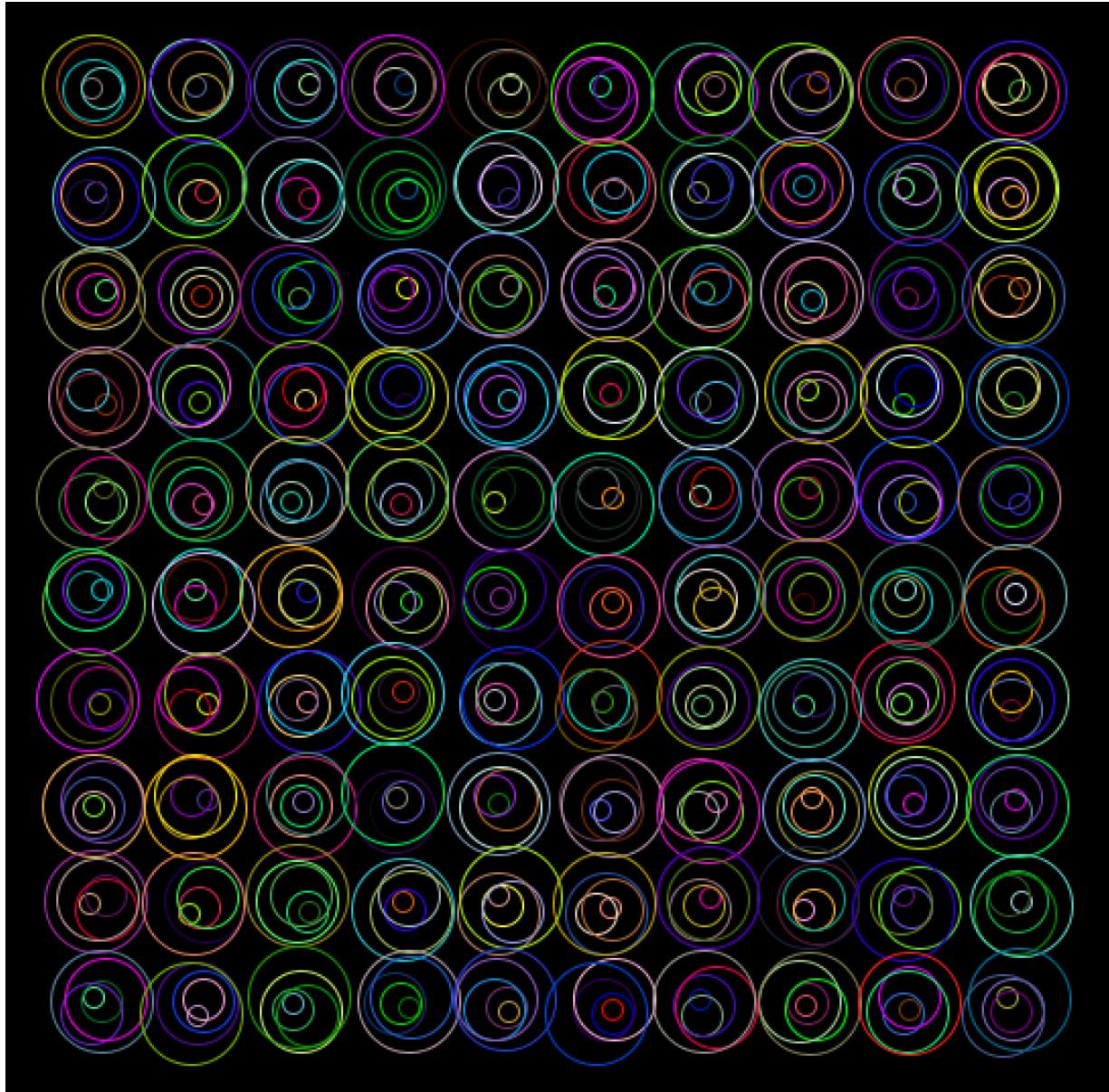
also check out: sketch\_4\_2\_Variable\_Random.pde  
random x position for a drawing circles

a more complex example:

sketch\_4\_2\_Random\_CirclePossibilities.pde

```
//GOAL: random position, radius, color and alpha  
  
float xPosition; //circle position  
float yPosition;  
  
float circleRa; //circle radius  
  
float circleR; //circle color in RGB mode  
float circleG;  
float circleB;  
  
float circleAlpha; //circle alpha
```

# sketch\_random\_concentric\_circle\_pattern.pde



# sketch\_random\_concentric\_circle\_pattern.pde

```
void setup () {
    size (800, 800);
    background (0);
    ellipseMode(CENTER);
    noFill();
    stroke(255);
    strokeWeight(1);
    noLoop();
}

void draw () {

    for (float i=0; i<10; i++) {
        float x=50;
        x=x+x*i;
        for (float j=0; j<10; j++) {
            float y=50;
            y=y+y*j;

            for (float k=0; k<5; k++) {
                float ir=10; //inner most circle radius
                float spacing=10; //radius difference between each circle
                float r;

                r=ir+spacing*k;
                ellipse (x, y, r, r);
            }
        }
    }
}
```

```
void setup () {
    size (800, 800);
    background (0);
    ellipseMode(CENTER);
    noFill();
    stroke(0);
    strokeWeight(1);
    noLoop();
}

void draw () {

    for (float i=0; i<10; i++) {
        float x=50;
        x=x+x*i;
        for (float j=0; j<10; j++) {
            float y=50;
            y=y+y*j;
            for (float k=0; k<5; k++) {
                float ir=10;
                float spacing=10;
                float r;

                float rr=random(0, 255);
                float g=random(0, 255);
                float b=random(0, 255);

                stroke(rr, g, b);

                r=ir+spacing*k;
                ellipse (x+random(0, 10), y+random(0, 10), r, r);
            }
        }
    }
}
```

# better random - Perlin noise

Returns the Perlin noise value at specified coordinates.

The resulting value will always be between 0.0 and 1.0

Perlin noise is a random sequence generator producing a more natural, harmonic succession of numbers than that of the standard random() function. It was developed by Ken Perlin in the 1980s and has been used in graphical applications to generate procedural textures, shapes, terrains, and other seemingly organic forms

```
float xPercentage = 0;

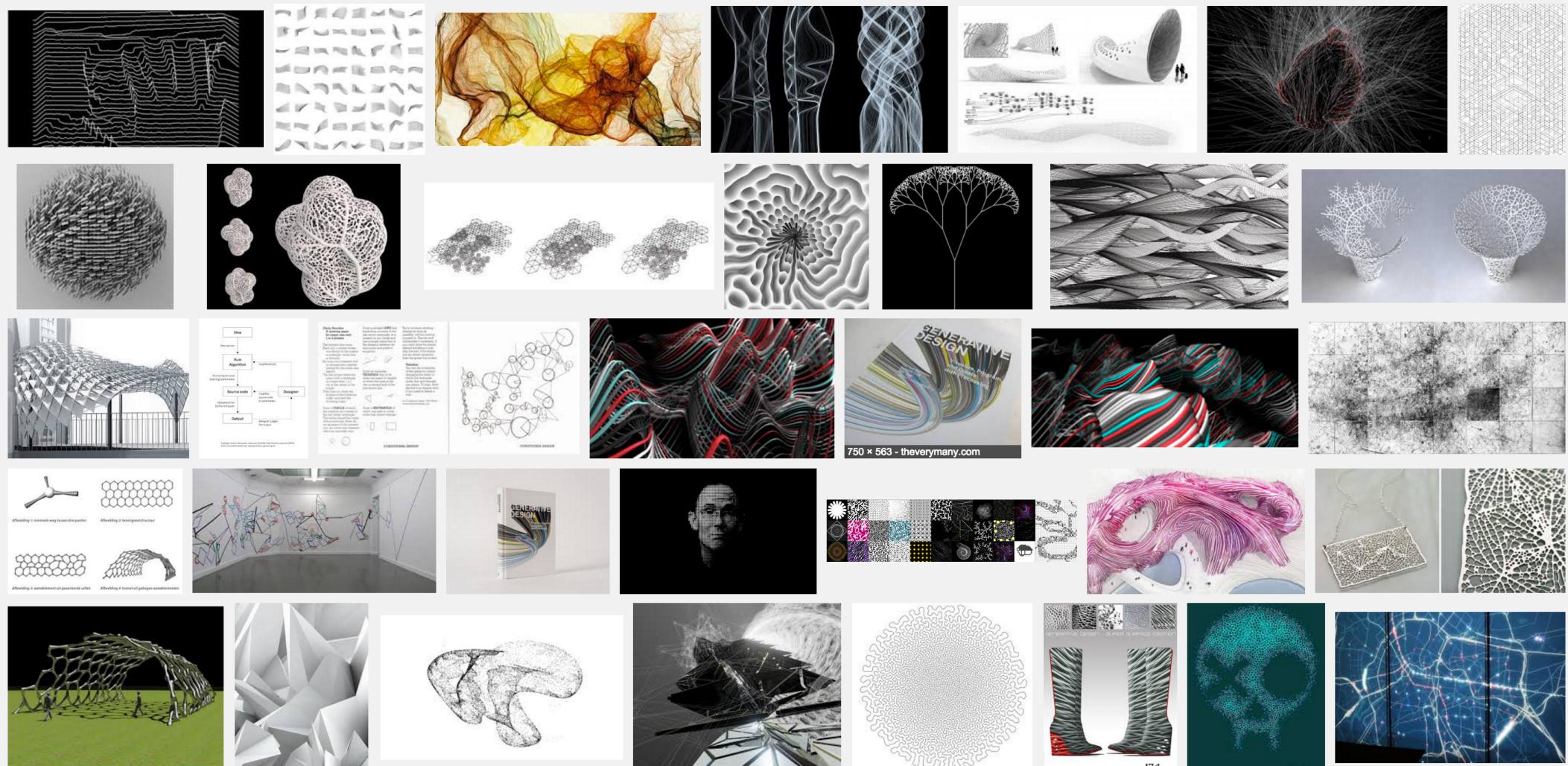
void draw() {
  background(200);
  xPercentage += .01;
  float x = noise(xPercentage) * width; //adds noise to xPercentage value
  line(x, 0, x, height);
}
```

# **What is Generative Design?**

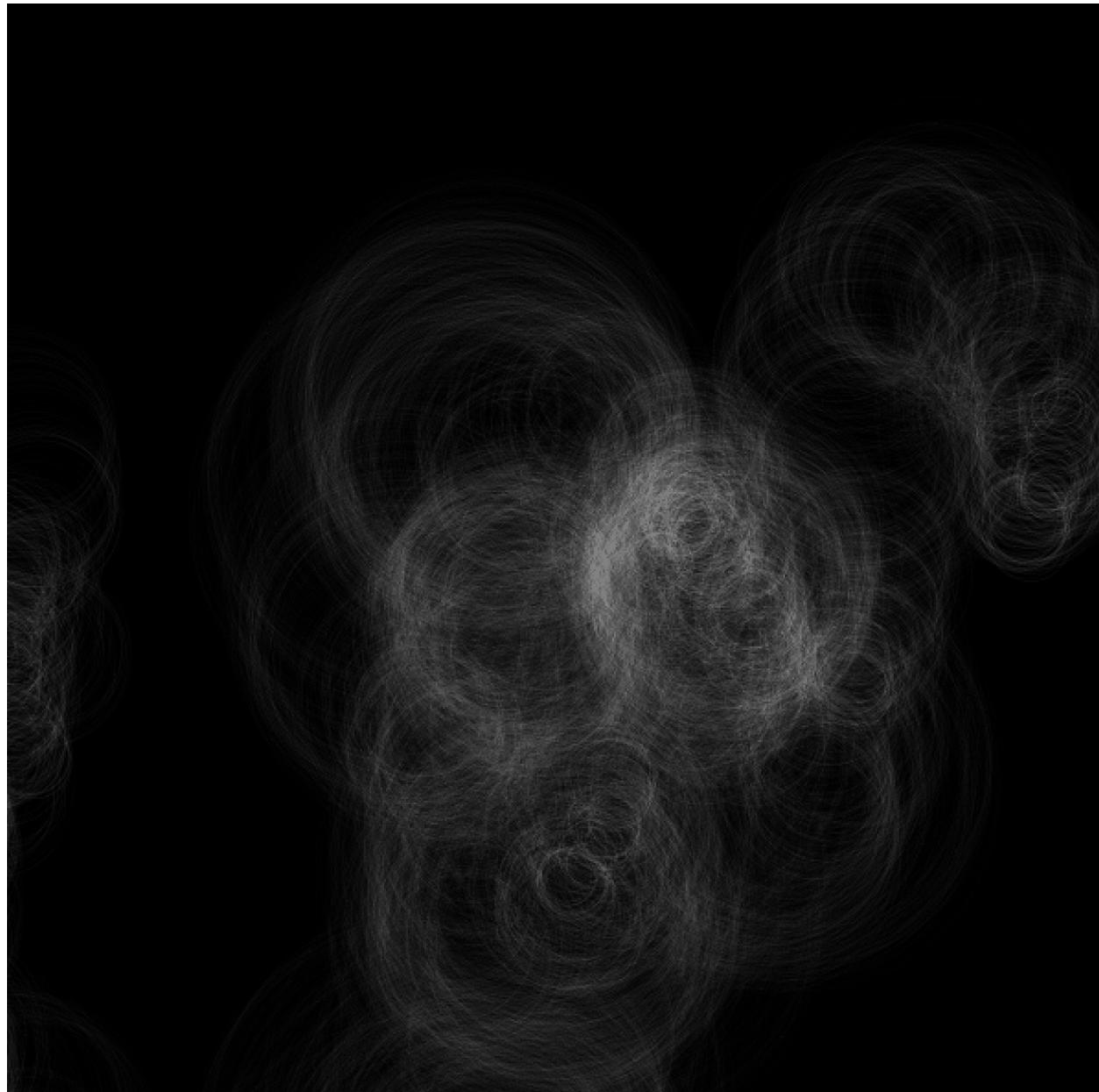
So new that no formal definition exists!

Involving variable(s) that evolve on its own according to a set of defined algorithm.

# What does it look like?



an example of generative design: sketch\_gen1.pde



```
float x, y;
float oX, oY;           //original x and original y
float r=50, theta;     //r is radius of the circles, theta is the angle of rotation

void setup() {
    size(600, 600);
    background(0);
    strokeWeight(0.1);

    oX=width/2;          //start generating in the middle of the canvas
    oY=height/2;
}

void draw() {
    if (oX>width) {           //step back 20, if goes off to the right of width
        for (int i=0; i<20; i++) {
            oX=oX-oX*i;
        }
    }

    for (theta = 0; theta < 2* PI; theta += 0.01) {           //stippled circles
        x = oX + cos(theta)*r;
        y = oY + sin(theta)*r;
        point(x, y);
    }

    oX = oX +random(-2, 2);
    oY = oY +random(-5, 5);
    r=r+random(-2, 2);
    float alpha=random(-50, 50);
    stroke(255, alpha);

    if (r<10) {           //reduce alpha in small circles
        alpha=20;
        r=50;
    }
}
```

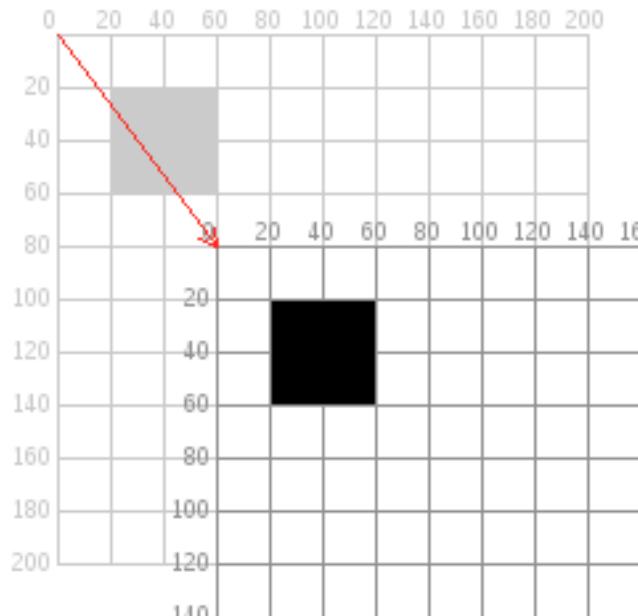
# **Transform**

translate

rotate

scale

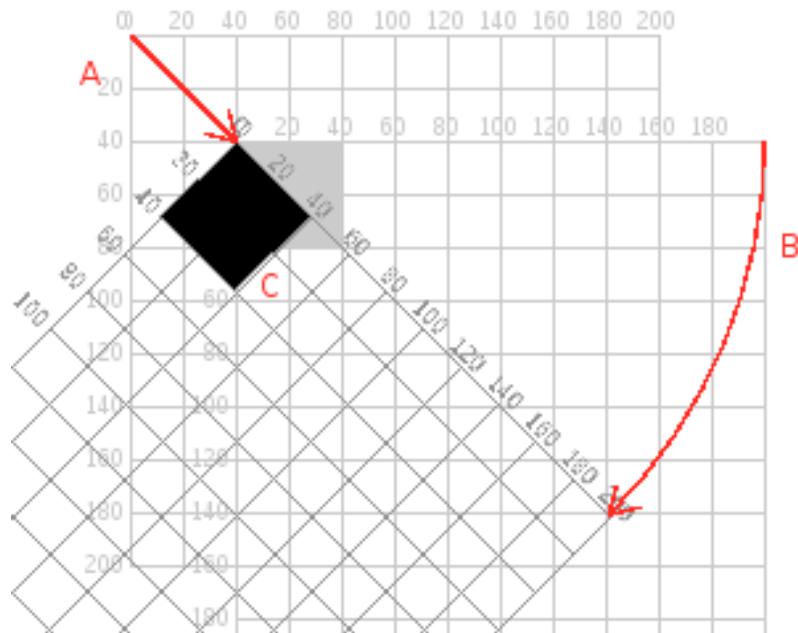
# translate



```
translate(60,80);  
rect(20,20,40,40);
```

# translate

## then rotate



```
translate(40,40);  
rotate(radians(45));  
rect(0,0,40,40);
```

Where is the center of rotation?

## sketch\_14\_1\_basic\_translate\_rotate\_scale.pde

```
void setup() {  
    size(500, 500);  
    background(0);  
    stroke(250);  
    fill(255);  
}  
  
void draw() {  
  
    translate(100, 100); //move the origin from (0,0) to (10,10)  
    //translate (mouseX,mouseY);  
    rotate (PI/4); //rotate the rectangle 45 degrees clockwise  
    //float rad = radians(45);  
    //rotate(rad);  
    scale (1.5); //scale up 1.5 times  
  
    rect(0, 0, 100, 50);  
}
```

The order of operation matters!

**What happens when you draw  
another shape after the coordinates  
have been**

translated

rotated

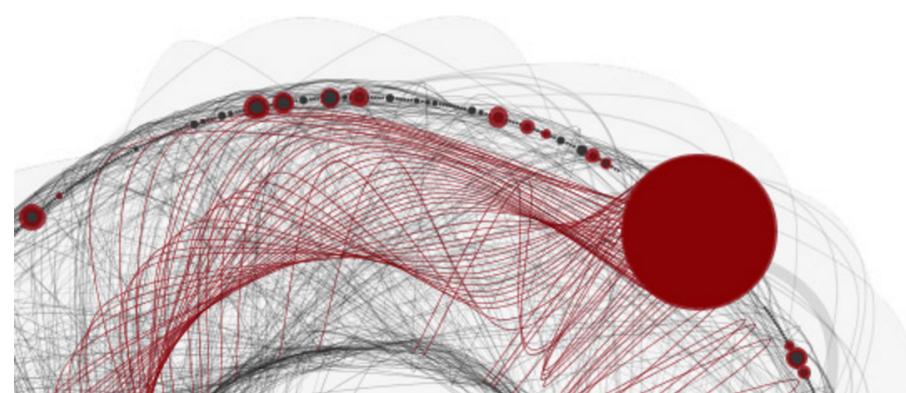
scaled

?

# **Midterm - Generative Design**

Project Guideline  
Credit when credit is due!

# OpenProcessing

[browse](#)[classrooms](#)[collections](#)[books](#)[go plus+](#)[jobs](#)[search](#)[login - sign up](#)

featured sketches

Visualizing text  
by Diana Langesketch  
by Aris BezasBiomechanics  
by Asher Salomon

A website to share Processing sketches

**share** your sketches with others**help** and **collaborate** with the community**improve** and **polish** your programming skills**follow** classes around the world teaching processing

Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is an open project initiated by Ben Fry and Casey Reas. It can be downloaded from [processing.org](http://processing.org).



Books are available for learning Processing at all levels.

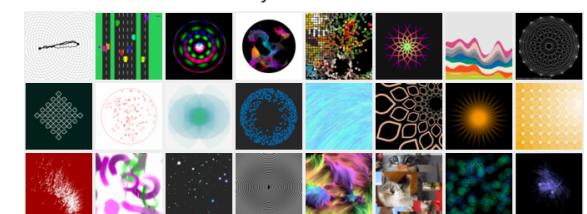
## Introducing Plus+ Membership!

Enjoy the next level for your sketches while supporting OpenProcessing

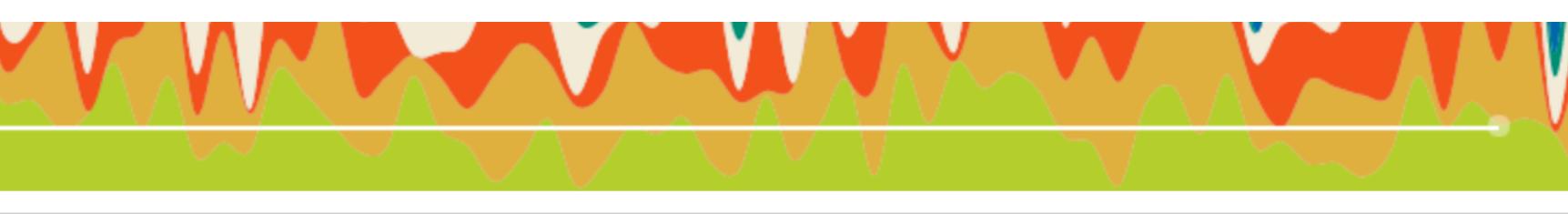
Bigger uploads, no ads, custom license & private sketches

[create new sketch](#)[upload from processing](#)

favorites of the last 7 days

[browse all-time favorites](#)

recent likes



## r-Stack Visualization

Add to Faves 2

Like

0

Streams are useful data representations for displaying changing different components in time. This example uses random data.

[Tweak!](#)  
[Download](#)  
[Embed](#)



code

tweaks (0)

about this sketch

visualization colors stack stream graph

diagram 200 views January 26, 2016

collection: [data visualization](#)

This sketch is running in HTML5 using [Processing.js](#).

license

[how to attribute?](#)

You should provide the text below when attributing this sketch:

*"Color-Stack Visualization" by Agoston Nagy,  
licensed under Creative Commons Attribution-  
Share Alike 3.0 and GNU GPL license.*

*Work: <http://openprocessing.org/visuals/?visualID=301695>*

*License:*

*<http://creativecommons.org/licenses/by-sa/3.0/>  
<http://creativecommons.org/licenses/GPL/2.0/>*

creative  
commons

gnu-gpl

# **today**

Due: Ex 3

Random and Transform + Generative Design

Introduce Midterm: Due Wed, Feb 10

Student Presentations

Reading: Ch 4-6 on p.60, optional Ch 14

# **Wednesday, Feb 3**

Due: Component 1, 2, 3 of Midterm project

Push Pop

Lab to work on midterm project

Student Presentations