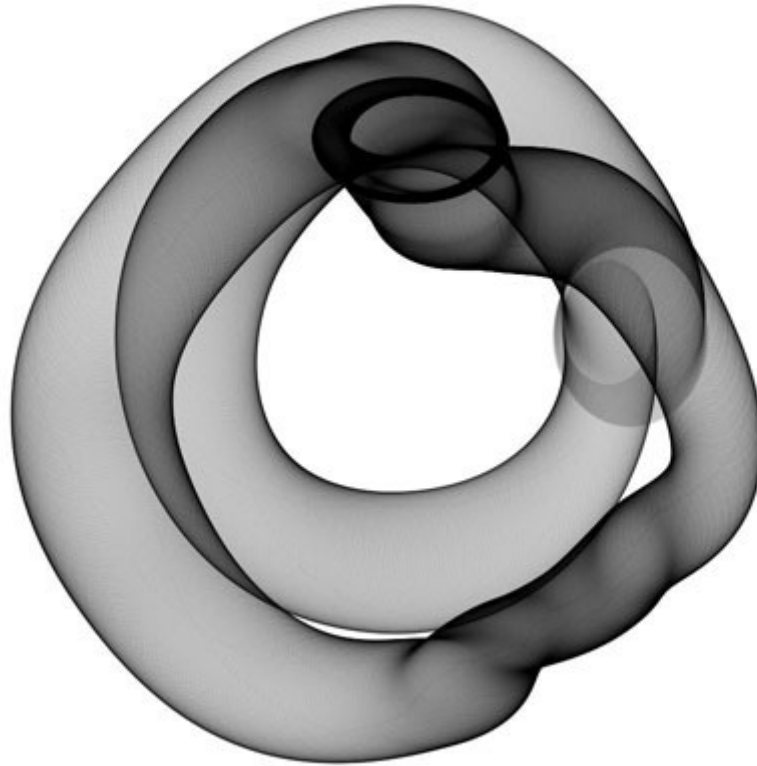


Animation and Generative Art



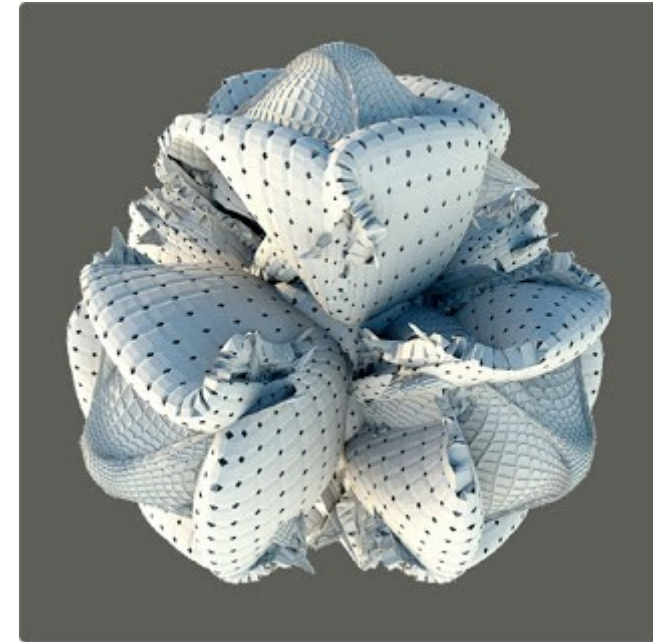
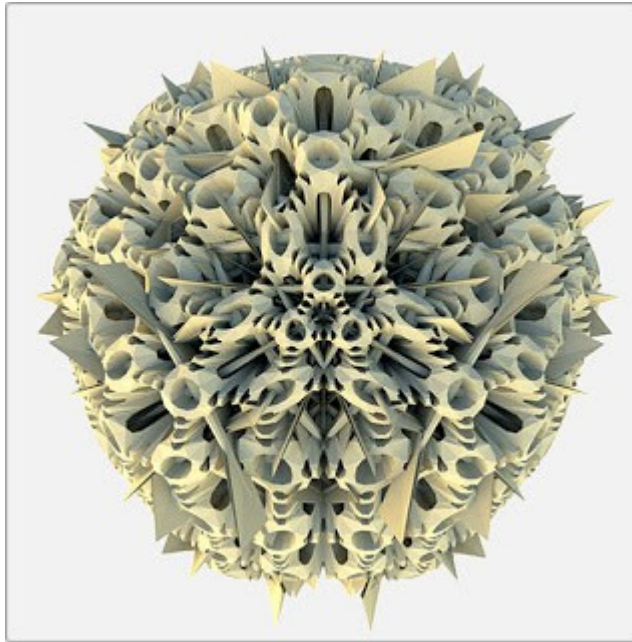
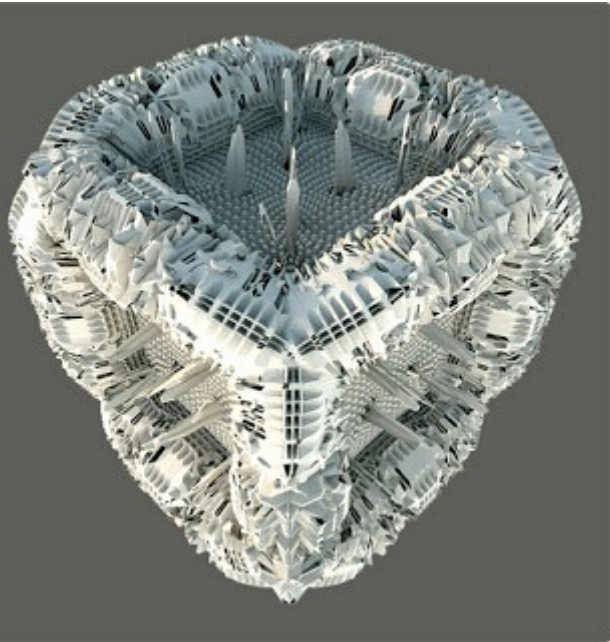
"The machine makes the music, but I created the machine... I don't know where responsibility lies in that situation".

- Sean Booth

Generative Art

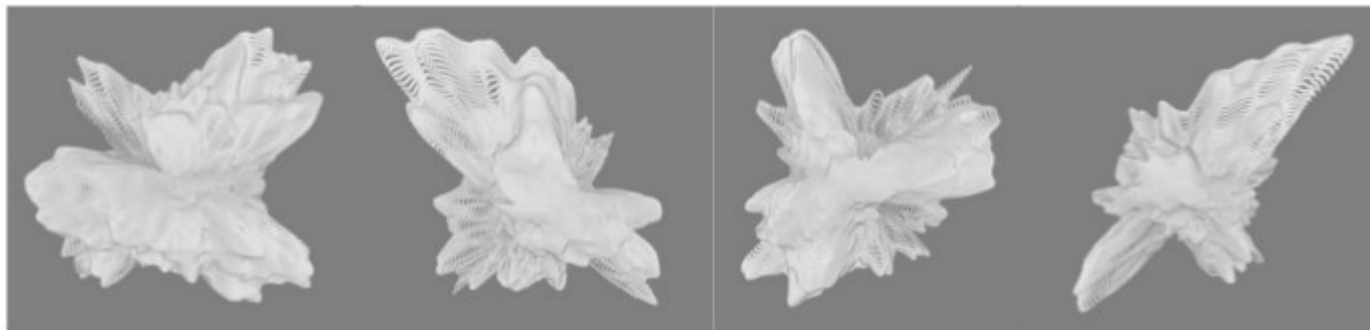
“Generative art refers to any art practice where the artist creates a process, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is then set into motion with some degree of autonomy contributing to or resulting in a completed work of art.”

- Philip Galanter



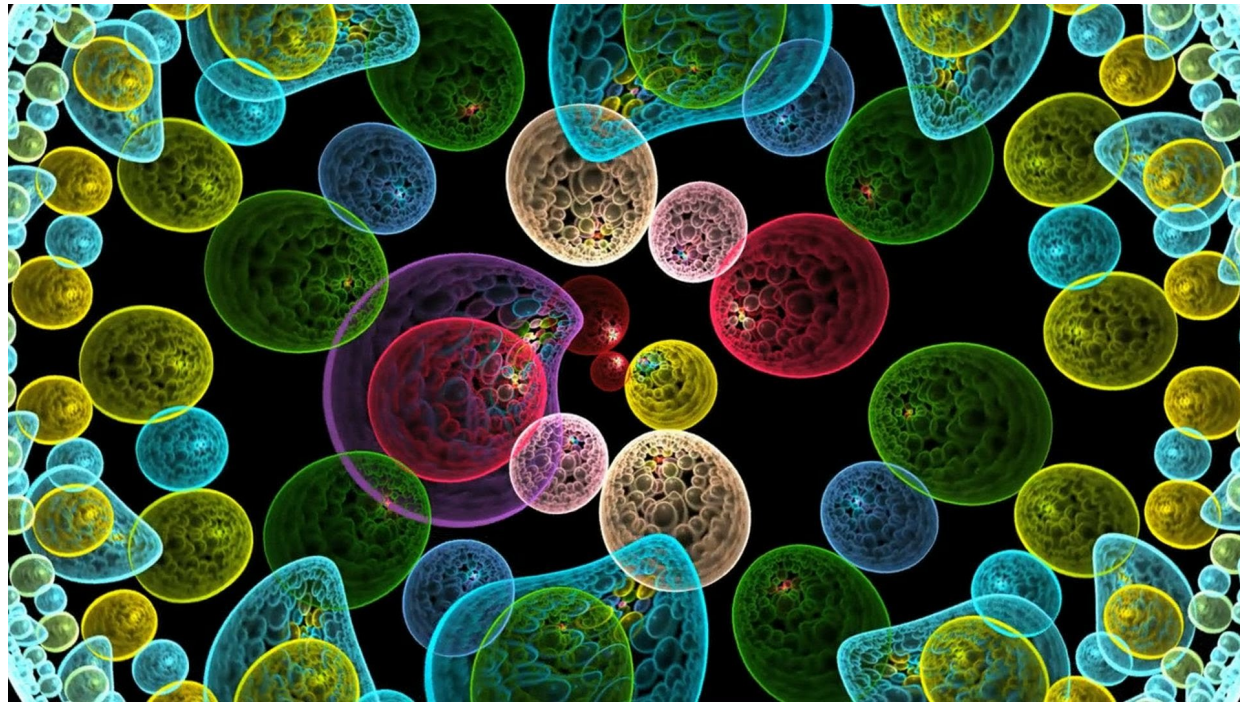
Programming vs. Art

- Generative Art: neither programming, nor art with the traditional definition...
- **Programming**: the interface between man and machine. A logical exercise with clearly defined goals
- **Art**: emotional and subjective subject without a clear definition
- **Generative Art**: the point where the above meet. Strict, cold processes used subversively in order to create illogical, unpredictable, expressive results



Generative Art

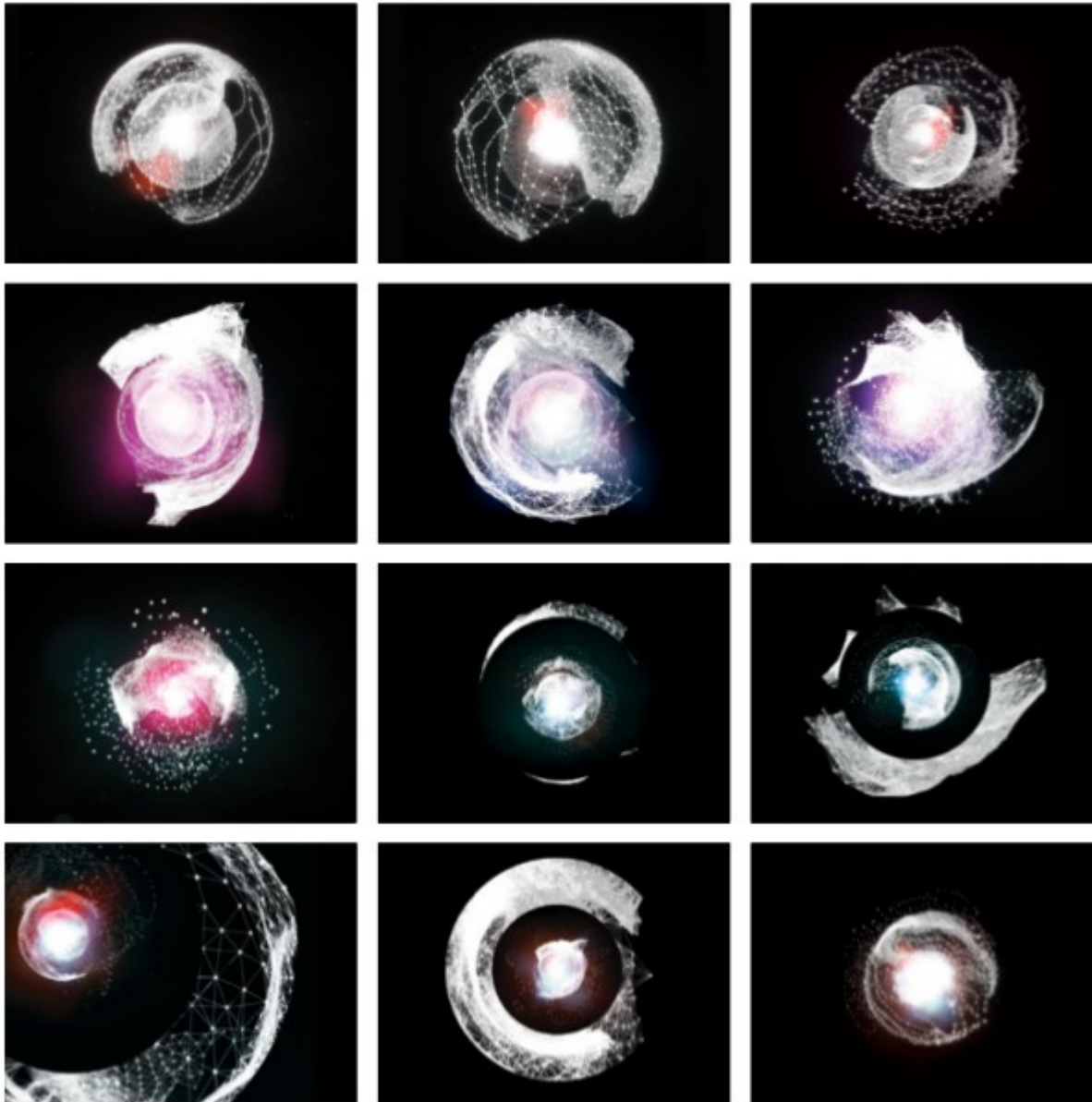
- Not something with build with plans, materials and tools
- Something that is grown and is cultivated like a tree or flower
- It's a property that emerges from simple functions: logical decisions and mathematics
- It's the art of the creation of the organic from the mechanical



the artist as a gardener







Mechanical & Organic

not necessarily on opposite poles

we find both attractive

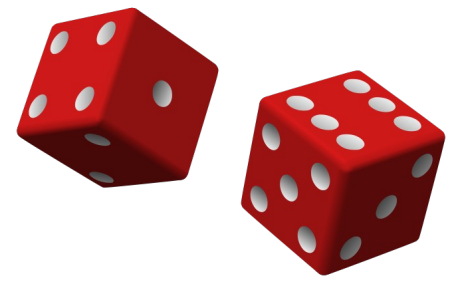
generative art uses the mechanical to produce the organic






Life in 2050

Generative Art


main characteristics



- Main characteristic: the artist does not control the system completely
- It's not the tools we use but how we use them -> Some degree of autonomy must exist
- We don't need a computer to create generative art. Generative Art in other fields:
 - painting
 - narrating (collaborative storytelling)
 - dance (Merce Cunningham – Beach Birds) 
 - music (John Cage / Mozart) 
 - architecture 
 - poetry



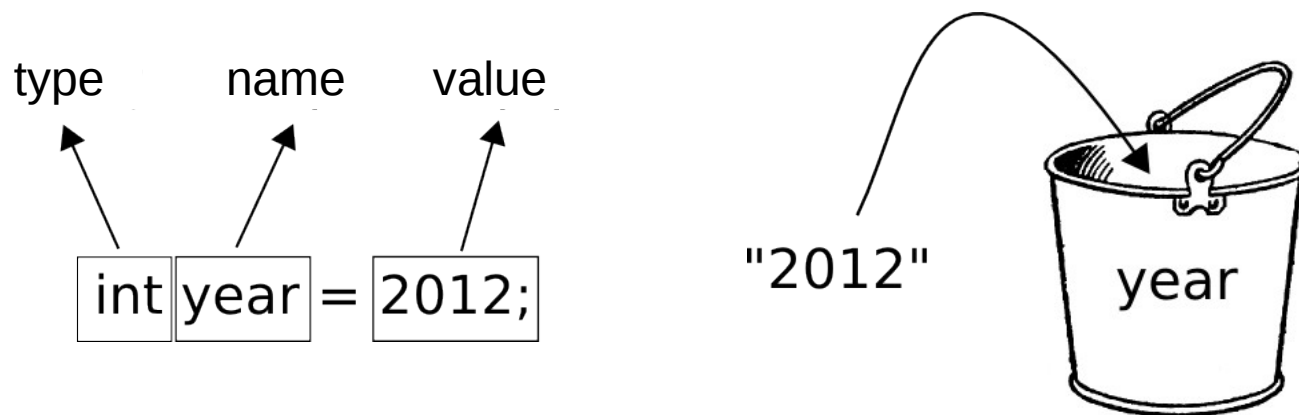
Why Generative Art;

- to imitate the complexity of systems (ex. financial, political, social)
- to imitate nature (ex. the relation between objects in an ecosystem, see electric sheep) 
- for smart objects
- to break the model of linear translation
- to distance ourselves from the role of the artist as sole responsible - *"to feel like gods"*
- for artworks that work independently, are fluid and never repeat themselves



variables

- we run one command at a time so we need a place to store values until we need them again
- a layer of abstraction:
 - no need to know the location of a value in memory
 - I just need to know its name



- variables are at the heart of dynamic programs

using variables

- defining the type:

```
float x;
```

- assigning a value:

```
x = 400;          // a number
```

```
x = y * 2;        // the result of an operation
```

```
x = abs(y);       // the return value of another function
```

- reading the value:

```
cout << x;
```

```
ofSetLineWidth(x);
```

```
ofSetColor(x/2);
```



variable types



- types:

int	integer	int
float	floating point	float
char	character	char
boolean	a binary value	boolean

- how to name variables:

- use camel case (myVar, centerOfRect, etc)
- don't use spaces or special characters except “_”
- don't start names with numbers
- start with minuscule, except classes

variables & memory allocation

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 or 4 bytes	1 wide character



scope

ofApp.h

```
int r;
```

ofApp.cpp

```
void ofApp::setup() {  
    r = 5;  
}  
  
void ofApp::update() {  
    r = 10;  
}  
  
void ofApp::draw() {  
    ofCircle(100, 100, r);  
}
```



scope cont.

ofApp.h

```
int r;
```

ofApp.cpp

```
void ofApp::setup() {  
    r = 5;  
}
```

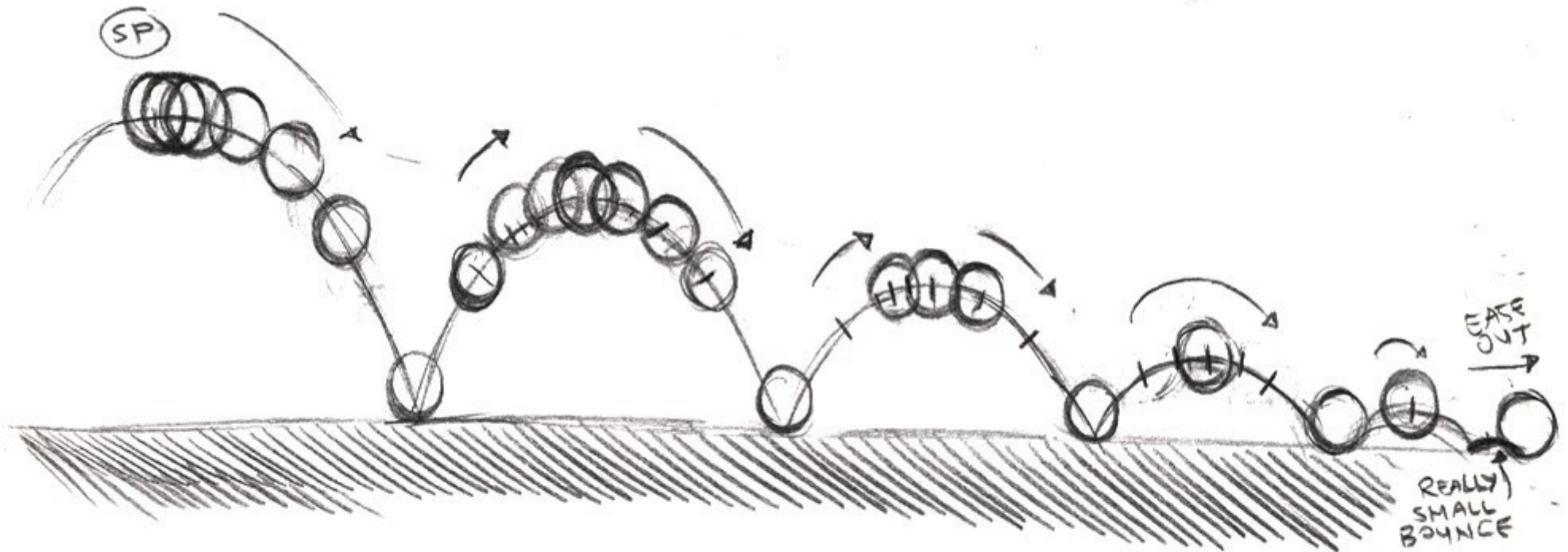
```
void ofApp::update() {  
    r = 10;  
    int x = 20;  
}
```

```
void ofApp::draw() {  
    ofCircle(x, 100, r);  
}
```

// **WRONG!** 'x' is unknown

simple animation

and its shortcomings



conditional statements

```
if ( ????? )  
{  
    // then run this code  
}
```


if within draw()

- placing if statements within draw()
 - we run some commands some of the time and not others
 - different behaviours depending on conditions

```
void ofApp::update( )  
{  
    if(?????)  
    {  
    }  
}
```

boolean statements

- Socrates is dead TRUE
- donkeys fly FALSE
- 15 greater than 20 FALSE
- 5 equals 5 TRUE



relational operators

they allow us to evaluate the relationship between values

>	greater than	(6>5)	TRUE
<	smaller than	(6<6)	FALSE
>=	greater or equal to	(6>=6)	?
<=	smaller or equal to	(6<=6)	?
==	equal to	(6==5)	?
!=	not equal to	(6 != 6)	?



“==” VS. “=”

```
if (x == y)  
{ }
```

// does x equal y?

```
x = y;
```

// make x equal to y

Conditional statements - if

```
if ( raining ) // evaluates statement
{
    // if it's raining it runs the following code
    - take umbrella
    - take car
}
```



how would we draw a rectangle when **mouseX** is past the mid-point of the x-axis and a circle when it's before?

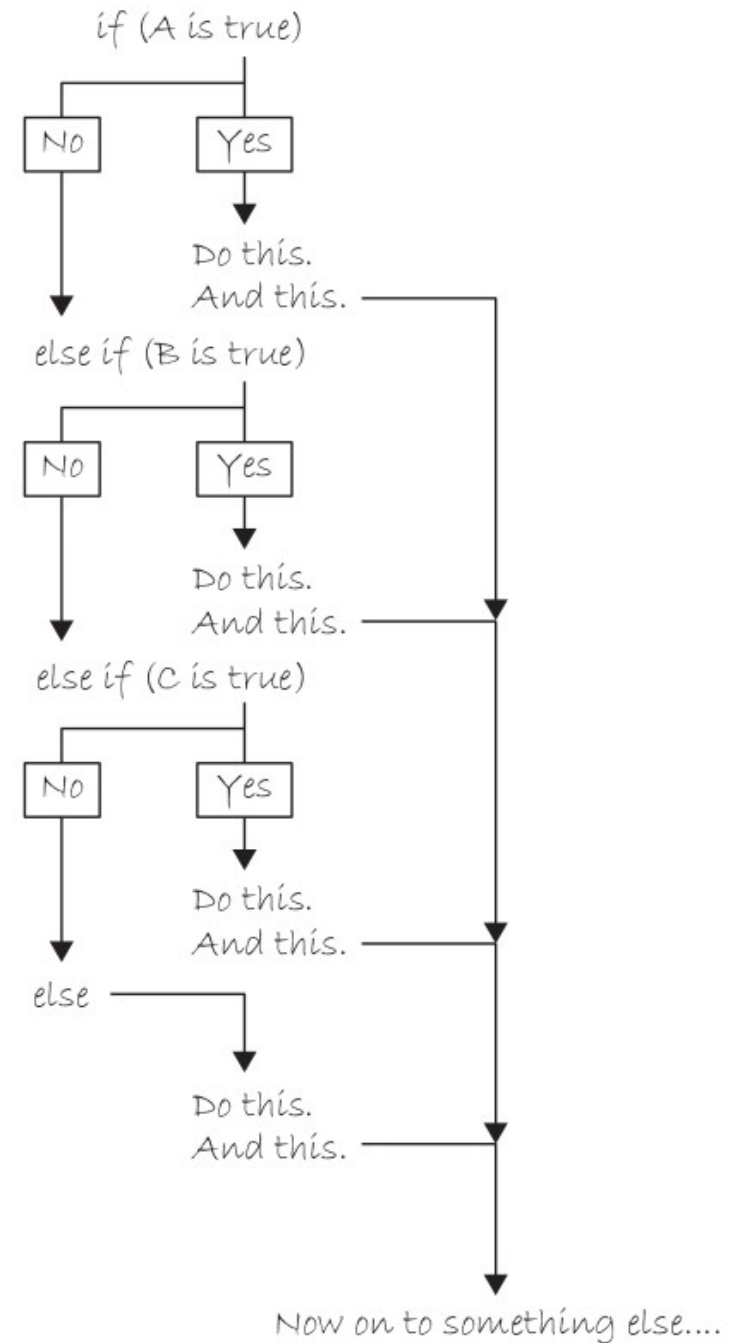
Conditional statements - `if` / `else if`

```
if ( raining )  
{  
    - take car           // executes this if raining  
}  
else if ( sunny )  
{  
    - take bicycle       // this if sunny  
}
```

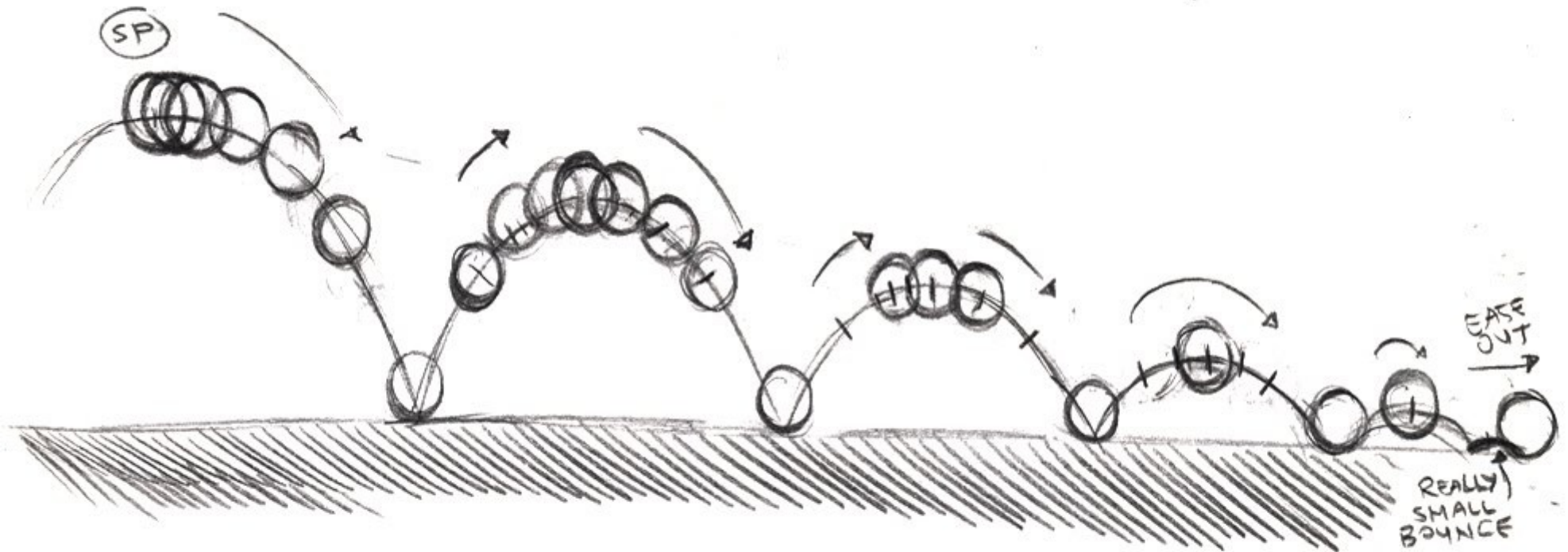
Conditional statements - `if` / `else`

```
if ( raining )
{
    - take car                // executes this if raining
}
else if ( sunny )
{
    - take bicycle           // this if sunny
}
else
{
    - stay home              // this for every other case
}
```


the
if/else if
path



simple animation



bouncingBall
simplePhysics

▶ kinect physics tutorial

▶ Swayi - A public interactive installation

! && ||
(not - and - or)

```
if ( !raining )           // if it's not raining
{
    - take bicycle
}
```

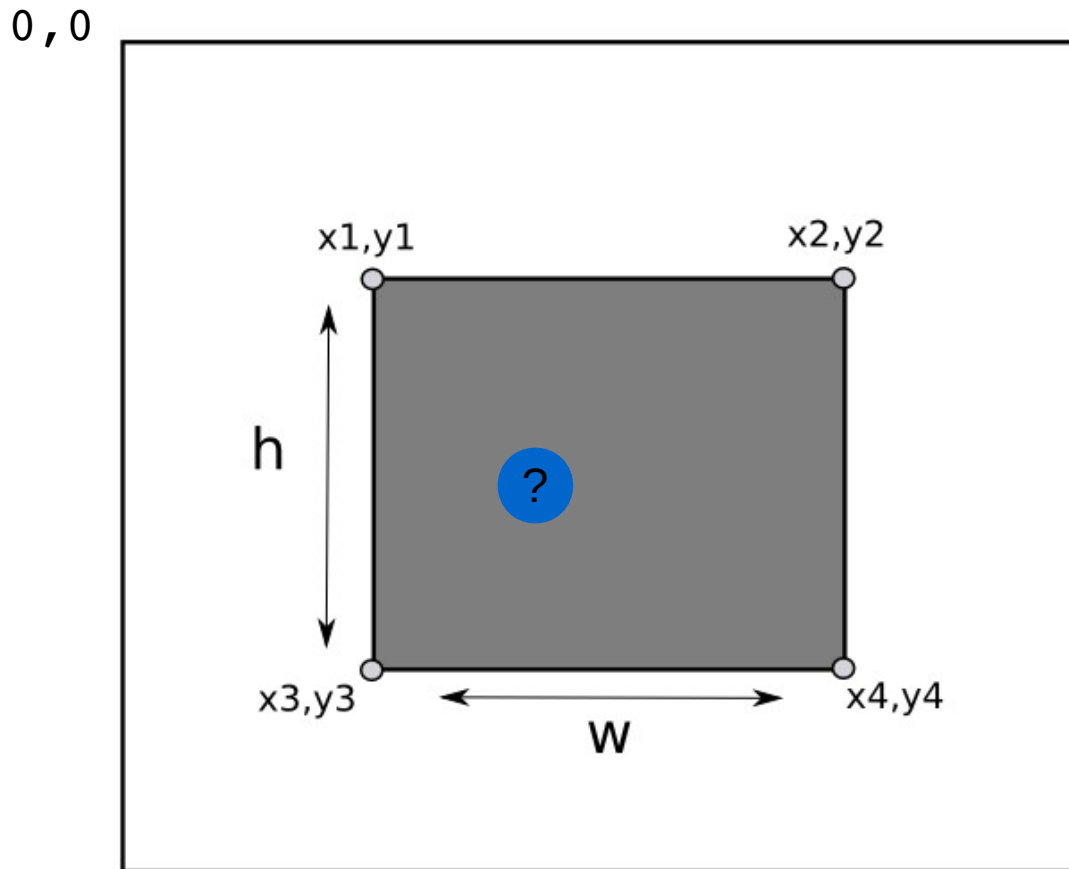
```
// if it's raining and you can drive
if ( raining && you have license )
{
    - take car
}
```

```
if ( raining || snowing ) // if it's raining or snowing
{
    - wear jacket
}
```




what is a button?

a button



$x2 = ?$

$y2 = ?$

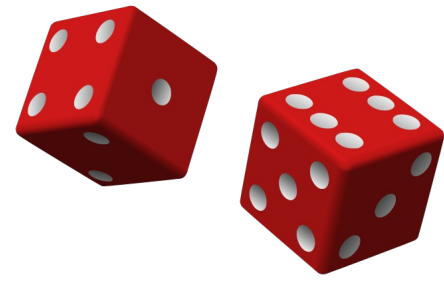
$x3 = ?$

$y3 = ?$

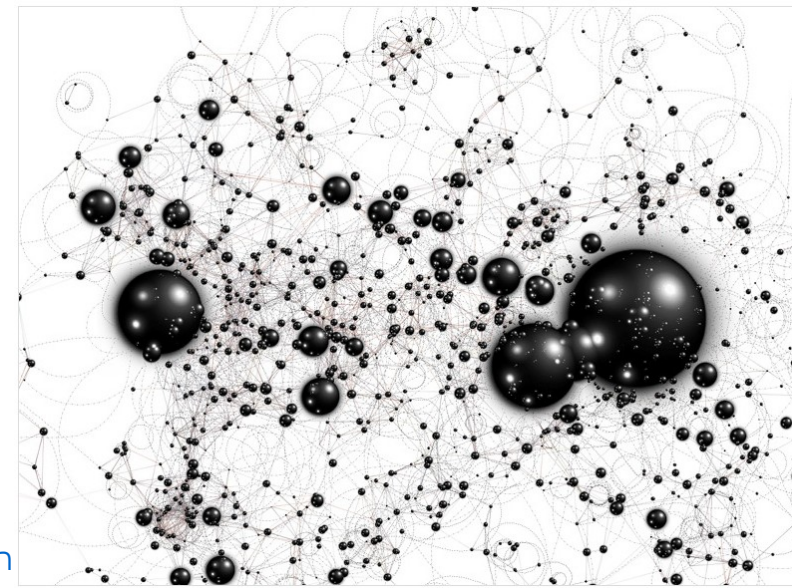
$x4 = ?$

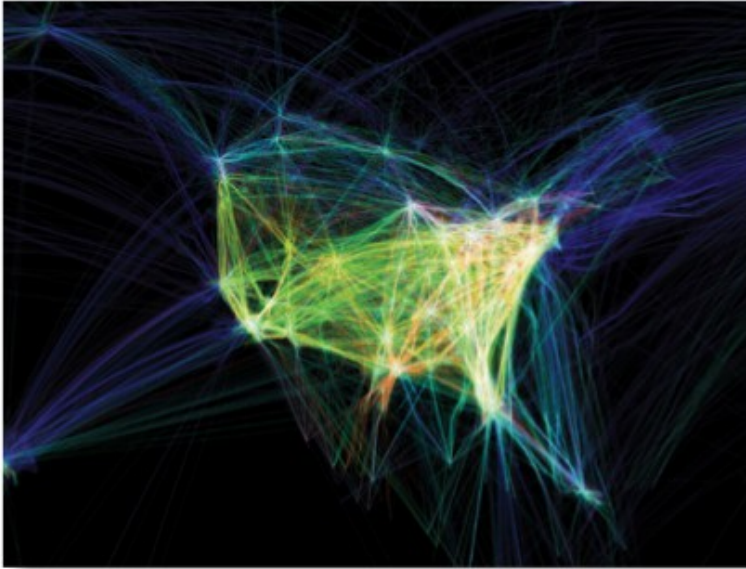
$y4 = ?$

random

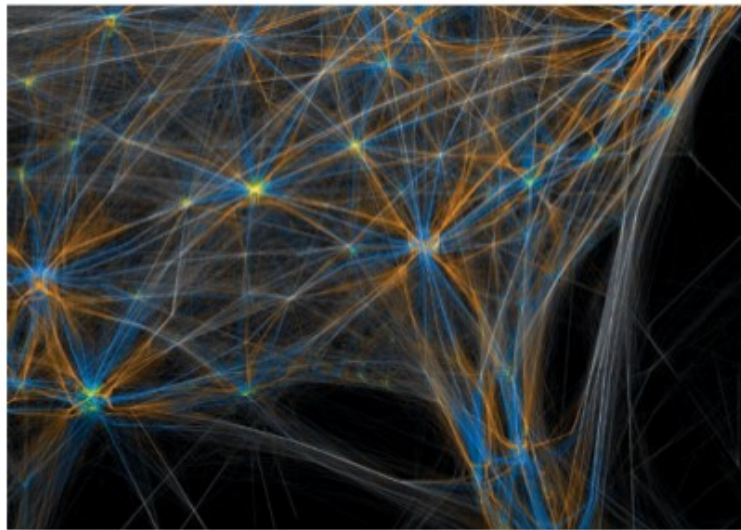


- Generative Art plays with randomness
 - the artists gives the reigns to the user or to an algorithm
- **But**: what distinguishes generative artworks are the elements that the artists chose to still control
- It's not about the *number* but the *range* of random/stochastic decisions that are allowed





We express our uniqueness, but we also want to express chaos and allow properties out of our control to emerge.

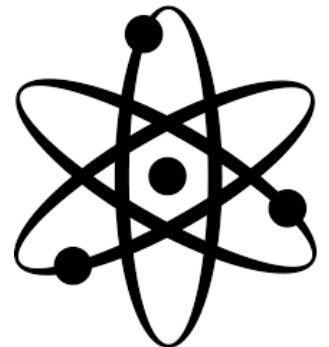


Generative Artists are artists of chaos

Aarron Koblin's "Flight Patterns"

random vs. chaotic

- **chaotic systems**
 - complex systems often exhibit chaotic behavior
 - they exhibit non-linear dynamics (butterfly effect)
 - predictions tricky - but are deterministic (cause/effect)
 - they have a sense of history
 - example: weather, dice, lottery, economy
- **true random/stochastic systems** have no history. Can be analysed statistically but not predicted precisely.



introducing random into our sketches

```
float num = random(5);
```

returns a number between 0 and 5

```
float num = random(3, 5);
```

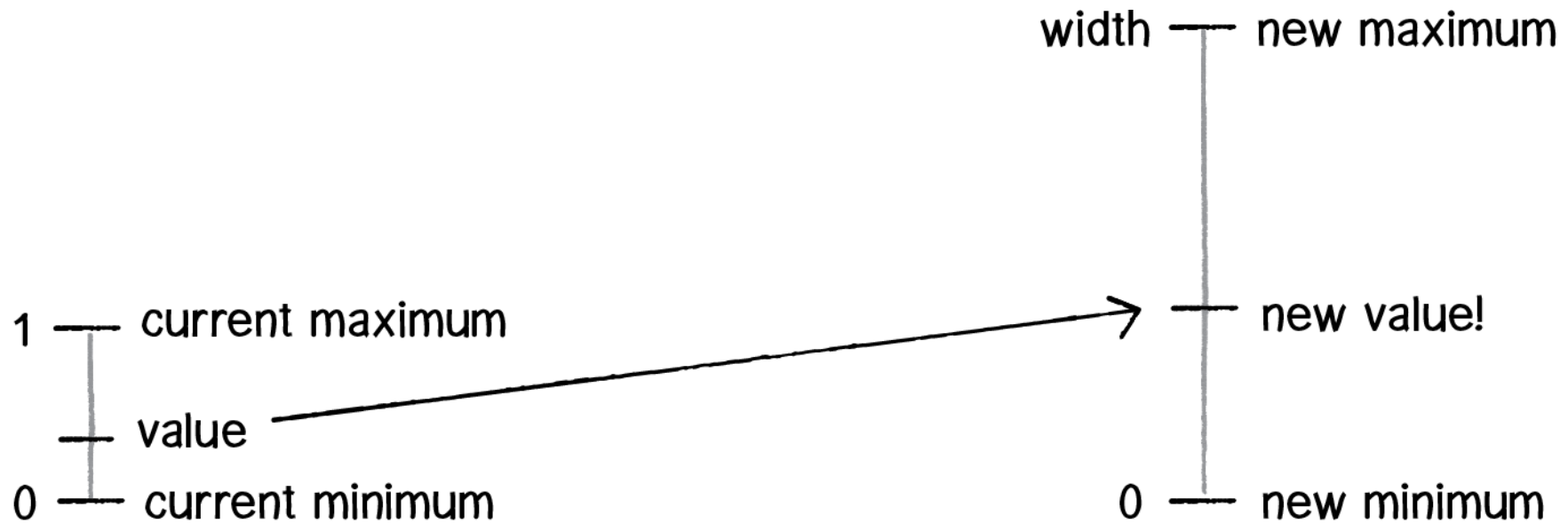
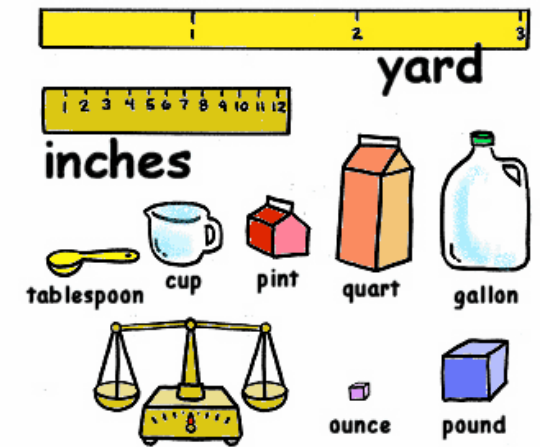
returns a number between 3 and 5 (but not 5)

Debugging

- we can print a variable's value using `println()`;
 - to print the variable number on screen:
`ofDrawBitmapString("mouseX: " + ofToString(mouseX), 10,15);`

ofMap ()

- convert one unit to another



new value= map(value, current min, current max, new min, new max)

2D transformations

commands

`ofPushMatrix();`

`ofPopMatrix();`

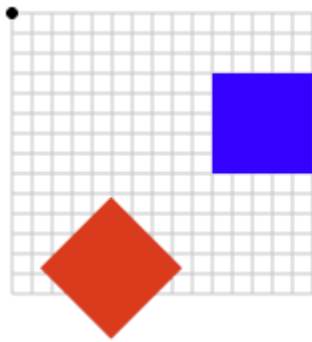
`ofTranslate(x, y);`

`ofRotate(degrees);`

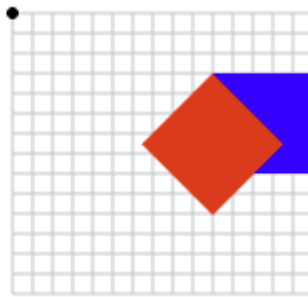
`ofScale(xAmt, yAmt);`



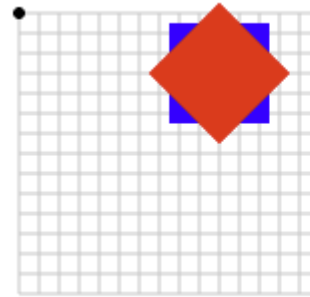
2D transformations visualized



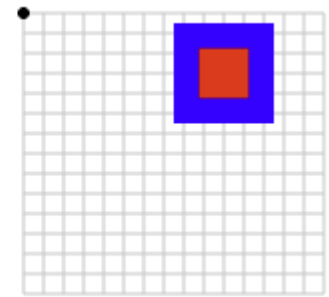
rotation before
translation



rotation from
top-left corner of
rectangle



rotation from
center of the
rectangle



scaling after
translation



make
translation
before rotation



change rect
mode to center
`ofSetRectMode()`

ACTION

SOLUTION

comments

“Any code of your own that you haven't looked at for six or more months, might as well have been written by someone else.”

(Eagleson's law)