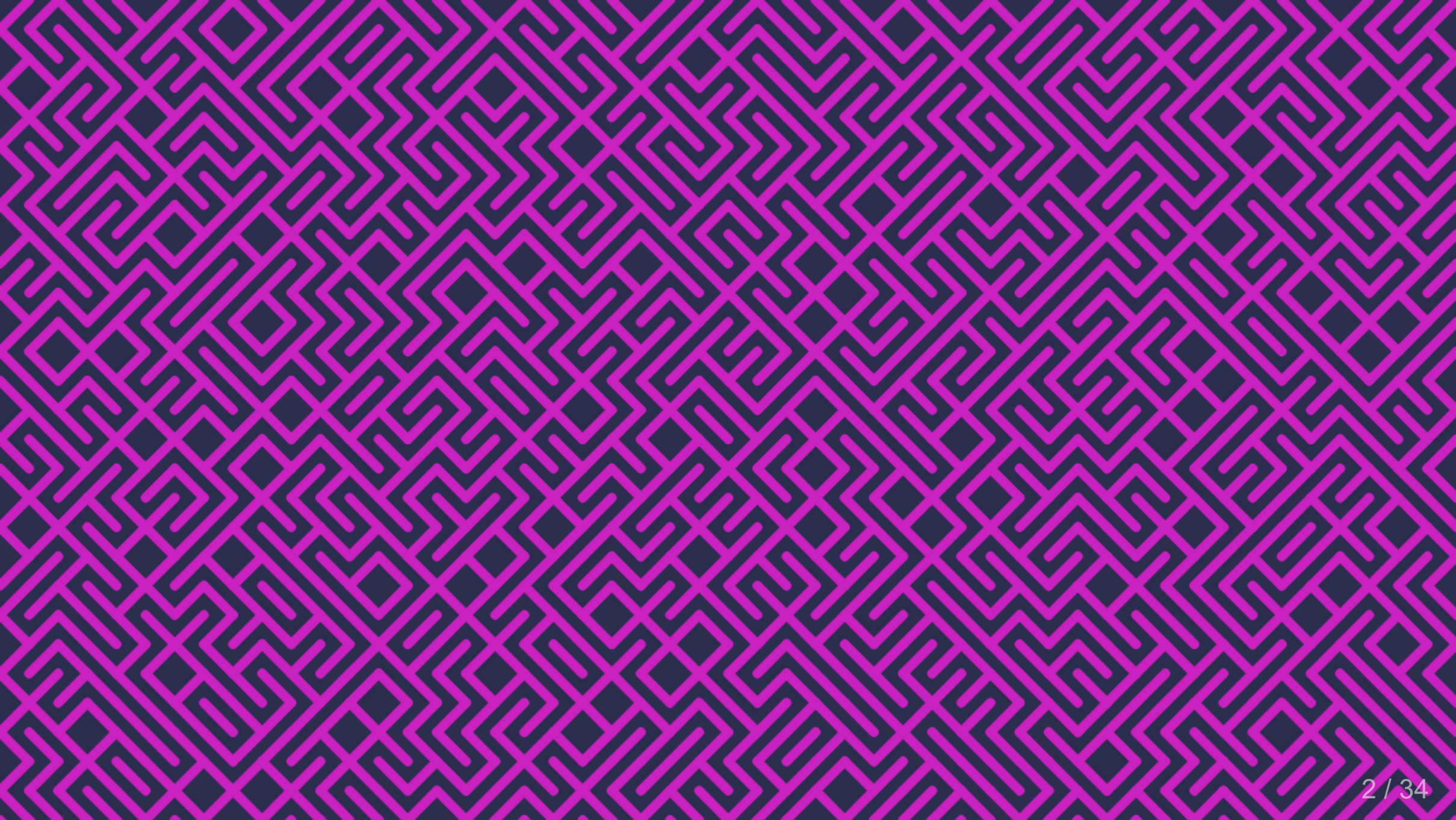
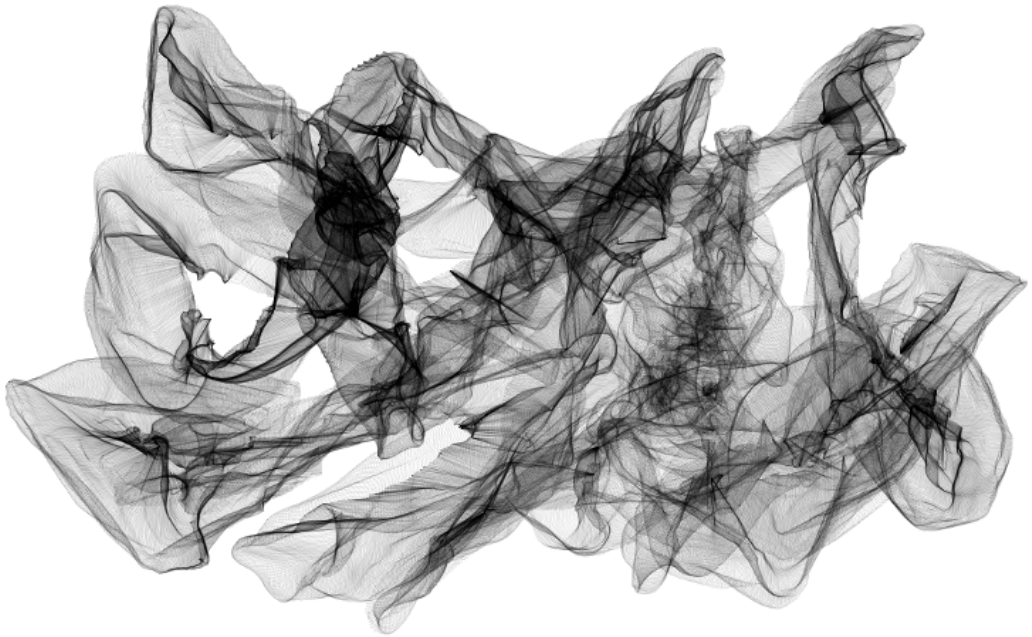
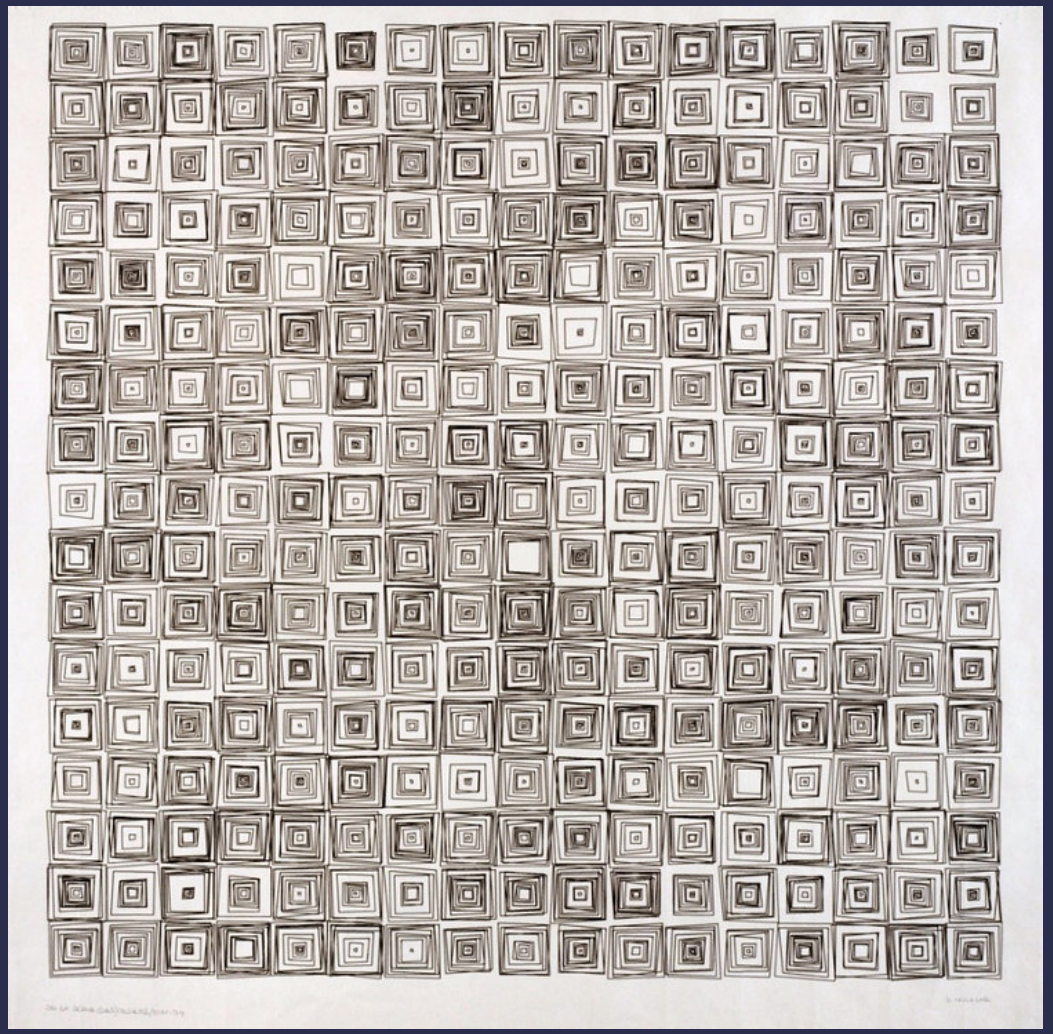
Programming for Artists



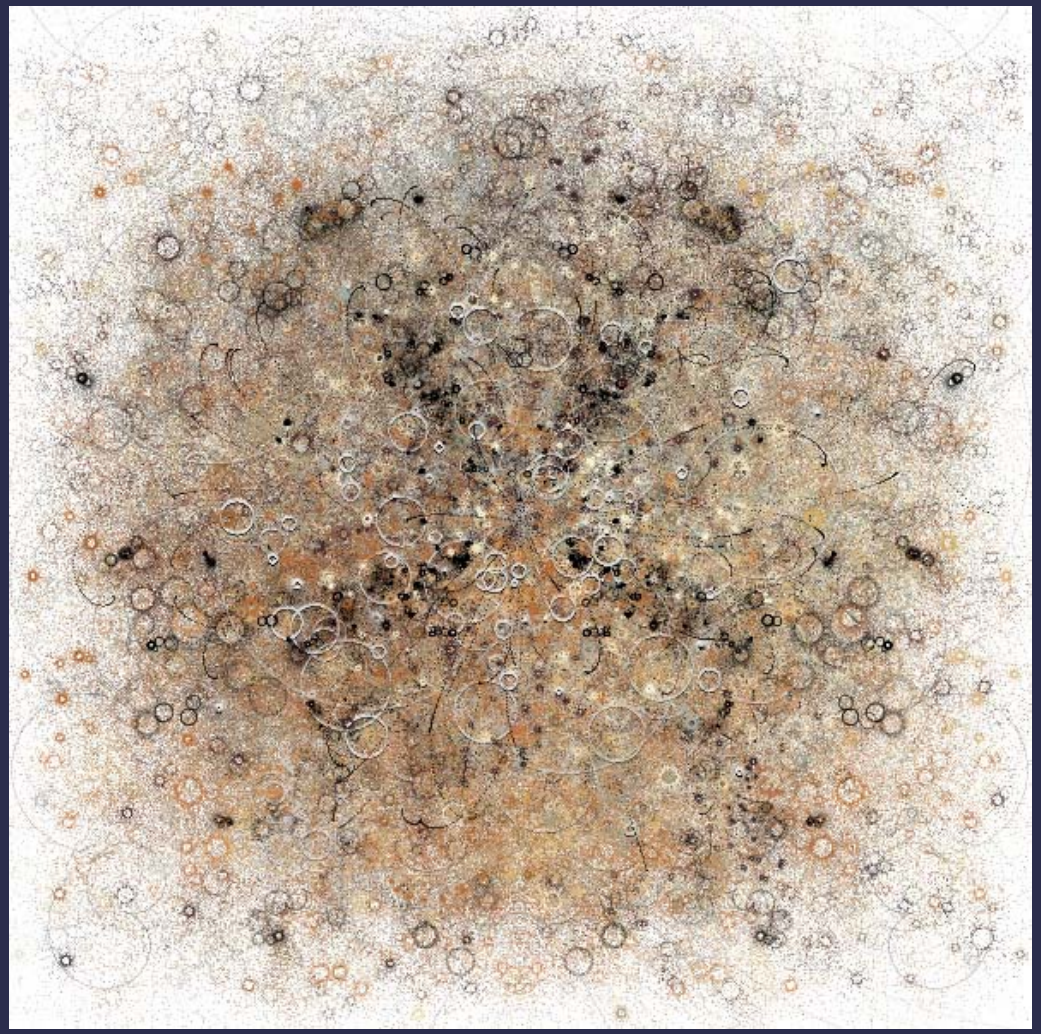


Drawing with particles by 010101101 (2009)

Cartesian Creatures by David Dessen (2007)

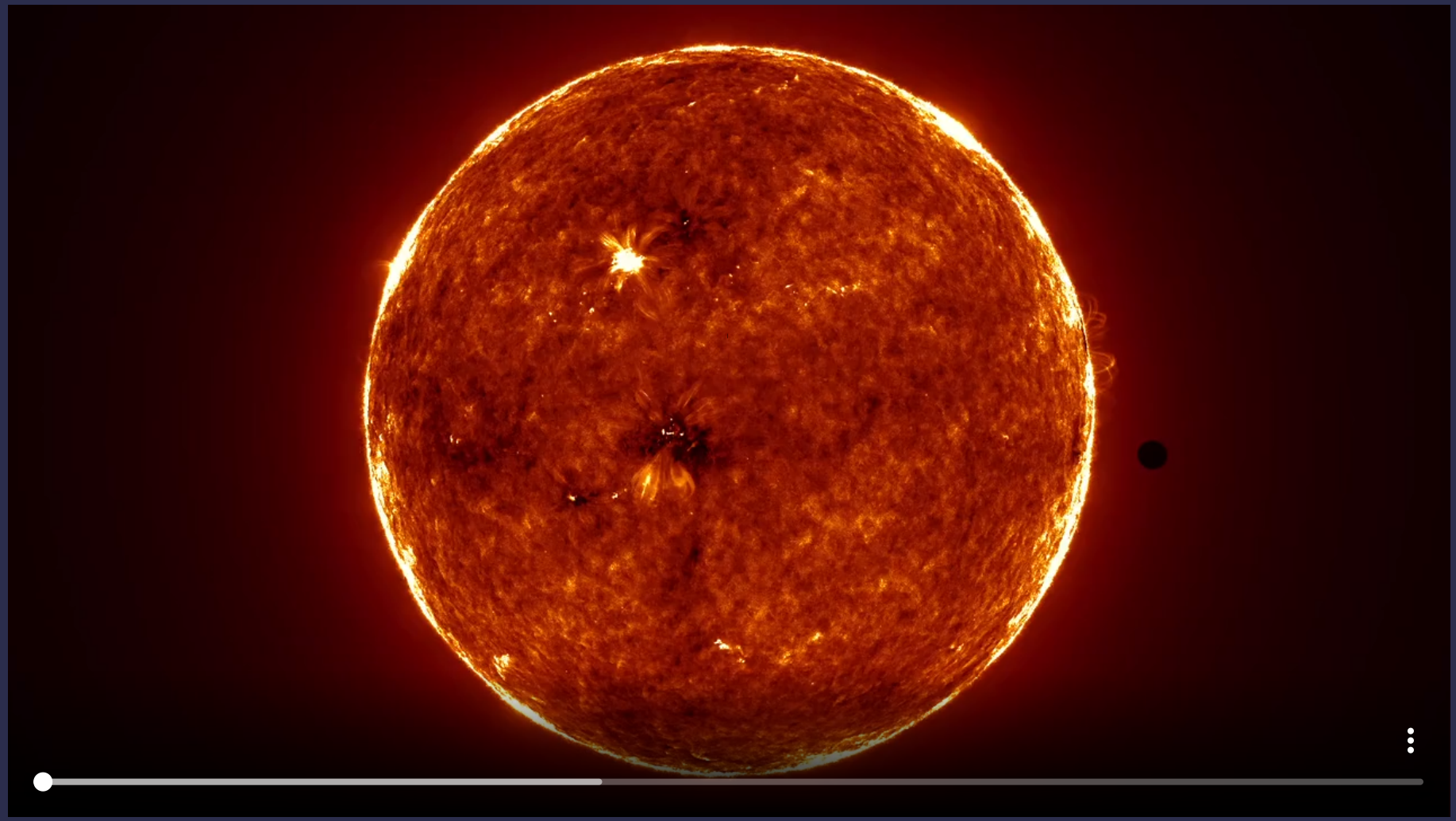


(Dés)Ordres by Vera Molnár (1974)

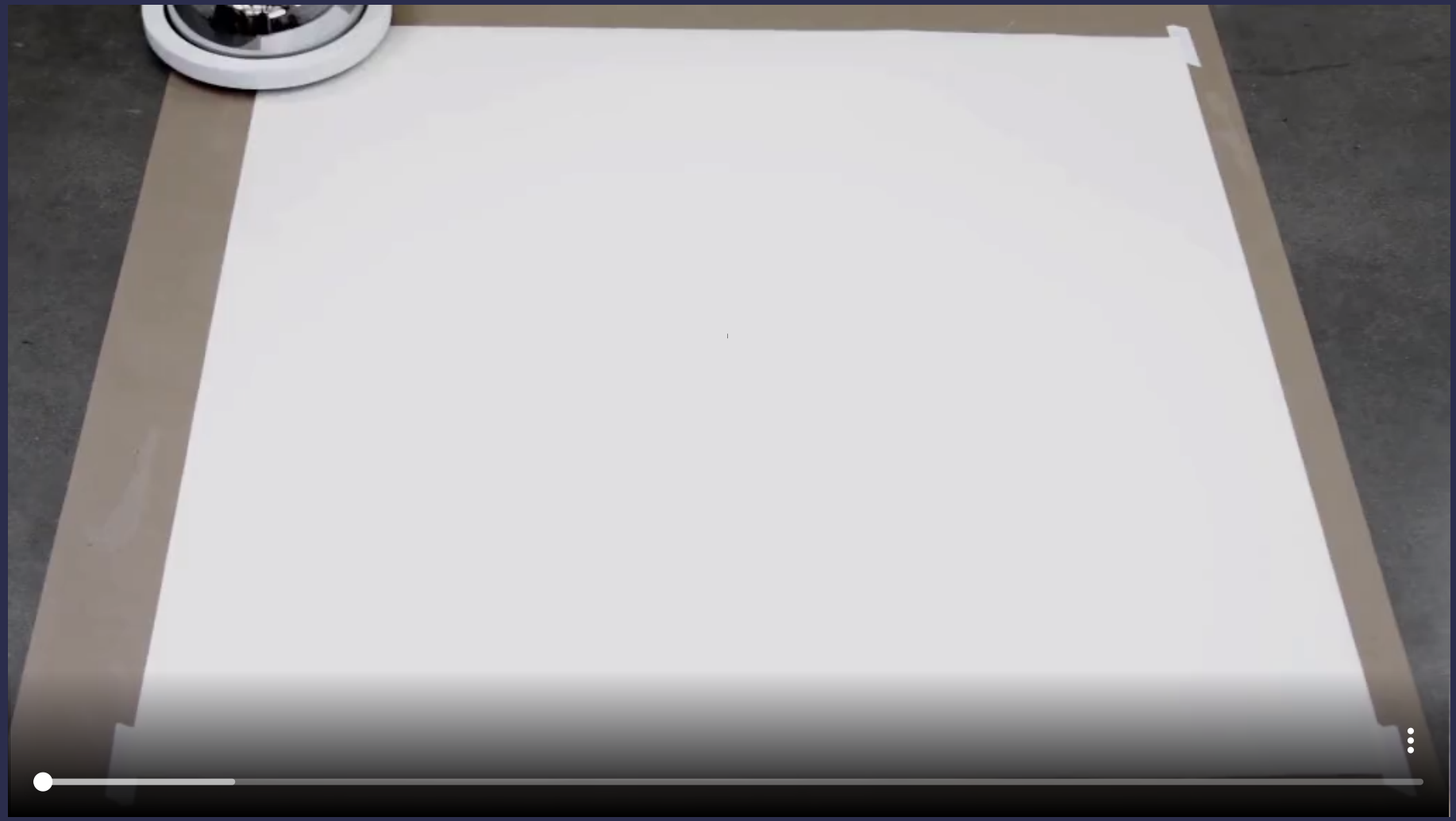


Bubble Chamber by Jared Tarbell (2003)

Nude Portrait by Robbie Barrat (2018)



Sol by Robert Hodgin



Mechanical Parts by Matthias Dörfelt (2013)

What is code?

- A series of instructions for the computer
- A precise way of explaining how to do something
- Written in a programming language
- Facilitate communication between human and machine, rather than human and human

```
"use strict";
function setup() {
  createCanvas(500,500);
  background(100);
  colorMode(HSB);
  noLoop();
}

let tilesX = 10;
let tilesY = 10;
function draw() {
  let tileWidth = width/tilesX;
  let tileHeight = height/tilesY;
  let [coloursLeft, coloursRight] = shakeColours(tilesY);
  noStroke();
```

Code as an artistic tool

- Like a paintbrush which can be used to paint a wall
- Add interactivity
- Artists, designers, architects use software to realise artistic vision – learning code gives more flexibility
- Generative art - “taking strict, cold, logical processes and subverting them into creating illogical, unpredictable, and expressive results. […] Generative art is about creating the organic using the mechanical.”

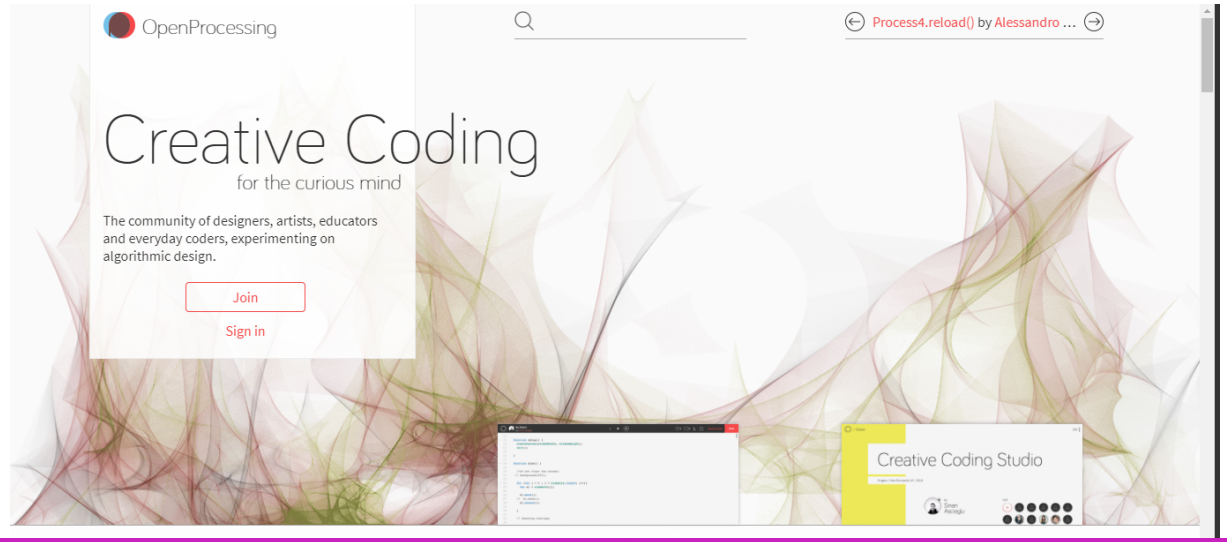
Organic vs Mechanical

> From the standpoint of Taoist philosophy natural forms are not made but grown, and there is a radical difference between the organic and the mechanical. Things which are made, such as houses, furniture, and machines, are an assemblage of parts put together, or shaped, like sculpture, from the outside inwards. But things which grow shape themselves from within outwards—they are not assemblages of originally distinct parts; they partition themselves, elaborating their own structure from the whole to the parts, from the simple to the complex.

- Alan Watts, 1958

Getting Started

- Search for “openprocessing” and open link
- Sign up
- Create first sketch
- Change the background to pink
  - `background('pink')`
- Stop it from looping
  - `noLoop();`

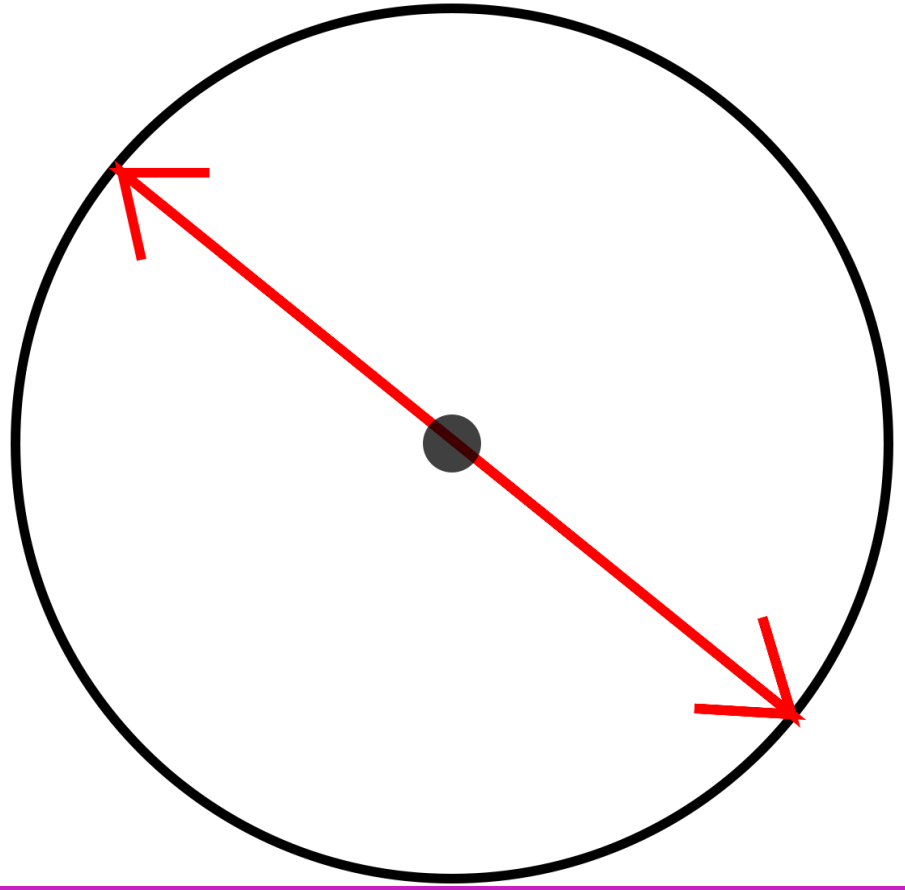


Canvas

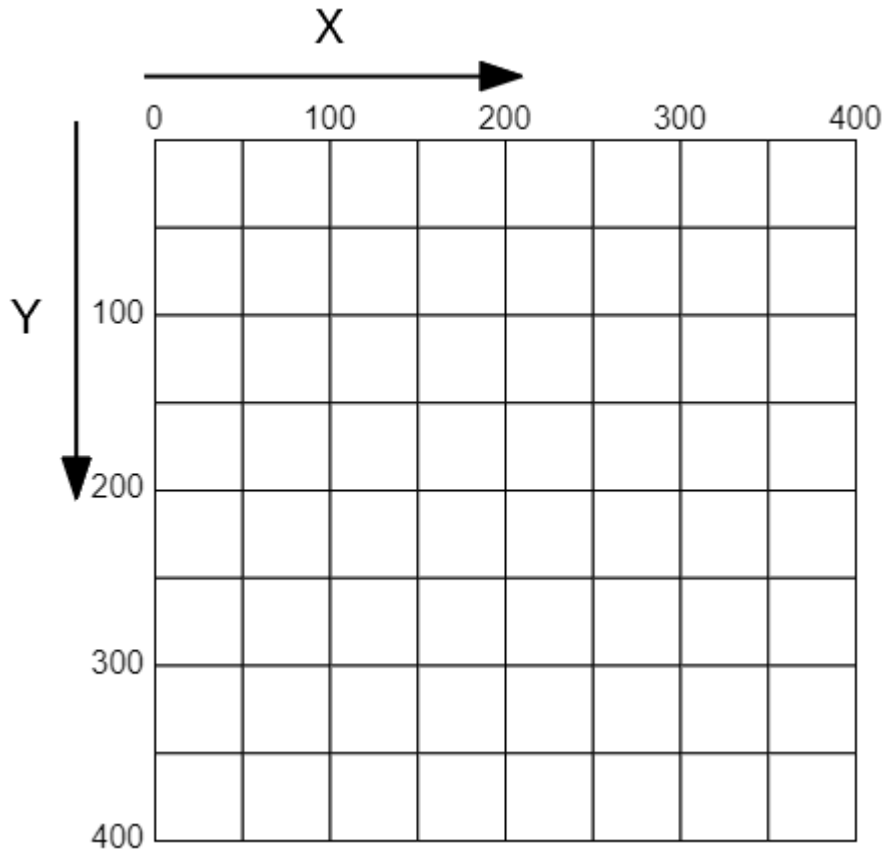
- We need canvas to draw
- We need to know how big the canvas is (width & height)
- `createCanvas(width,height)`
- Width and height are pixels

Circles

- Two pieces of information
- Location
  - x and y coordinates
- Diameter
  - In pixels



Coordinate system (computer)

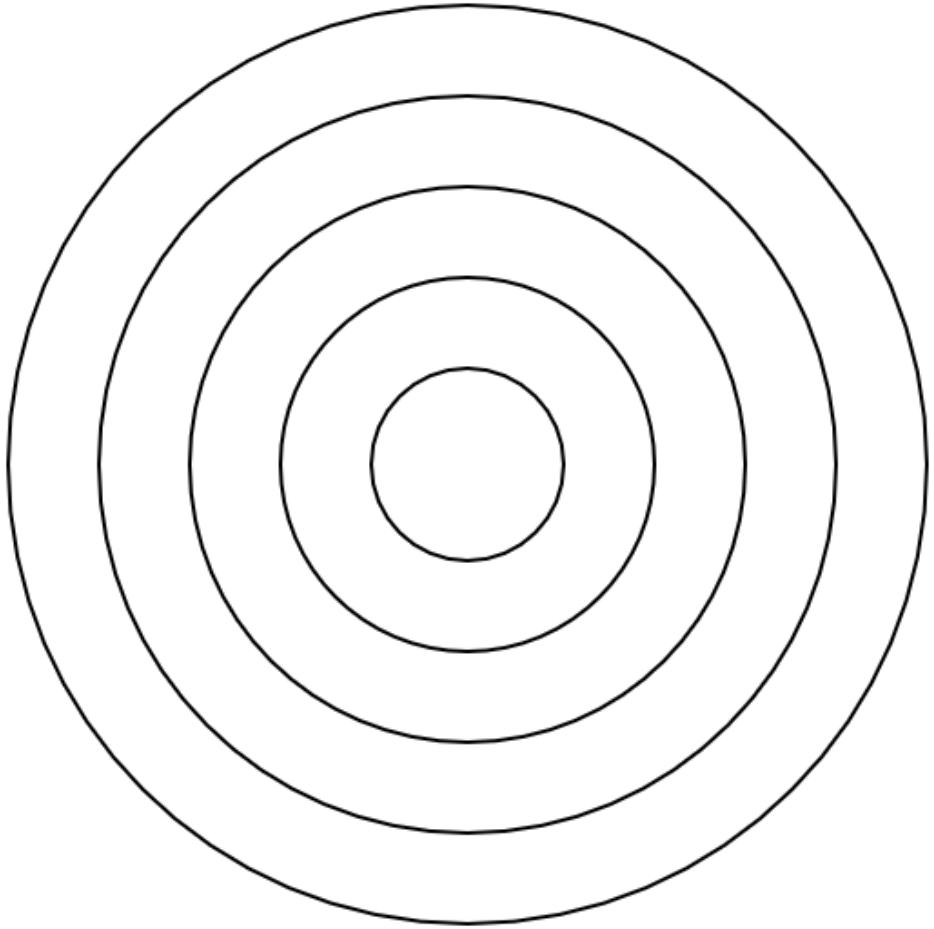


Draw some circles

- Canvas is measured in pixels, which are TINY!
- Try drawing some circles: ellipse(x, y, diameter)
- Examples:

```
ellipse(0,0,200);
ellipse(150,120,50);
ellipse(180,200,20);
```

Challenge 1 - Concentric circles



Saving and forking

- You need to save your work, it's not saved automatically!
- Forking work just means making a copy, you can do this with your own work or other people's
- If you are working on an idea and want to explore a different direction without losing the original, fork your work

Other shapes

- Rectangle:

```
rect(x,y,width,height)
```

  x and y refer to top left corner
- Line:

```
line(startX, startY, endX, endY)
```

- Check p5 reference for more shapes
- http://p5js.org/reference

Some grammar for code

- Read from top to bottom
- Semi colon at end of line - like a full stop
- Case sensitive, the following **wouldn't** work (notice the capital E)

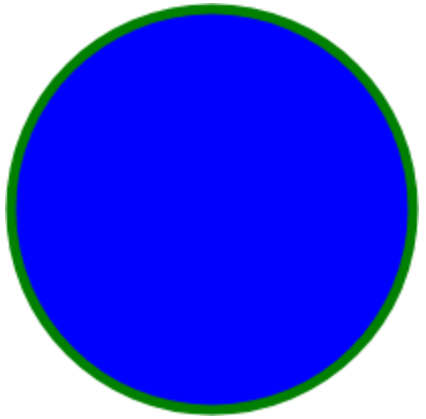
```
Ellipse(100,100,50);
```

- Commands to the computer look like this

```
ellipse(argument1, argument2);
```

- The arguments are information we give the command, e.g. how big to make circle or where to put circle

Colours



- There are many named colours, for example:
  - `‘darkolivegreen’`
  - `‘dimgrey’`
  - `‘dodgerblue’`
- You can change the fill colour of shapes:

```
fill('blue');
```

- You can change the stroke colour

```
stroke("green");
```

- More colours: colours.neilorangepeel.com

More Colour

- Grayscale – `fill(100)`
  - If one number provided for color command it will use grayscale
  - 0 is black, 255 is white
- RGB – `color(red,green,blue)`
- Hex codes – `color('#6495ED')`
- Colour picker: www.webfx.com/web-design/color-picker/

Randomness

- Computers can come up with random numbers
- This picks a number between 0 and 100 (includes decimals):

```
random(100);
```

- This picks a number between 100 and 200

```
random(100,200);
```

- This picks a random number between 0 and 1

```
random();
```

Random Colour

- Lists, in a computer program, are surrounded by square brackets

```
[1,2,3,4]
```

- We can have a list of named colours:

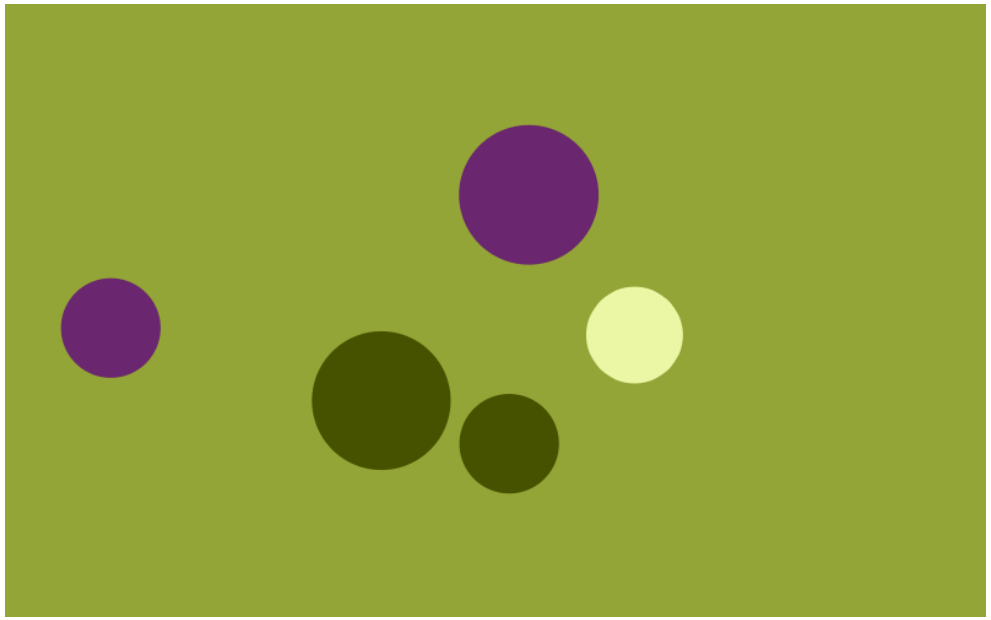
```
[‘red’, ‘green’, ‘blue’, ‘yellow’]
```

- We can use the random instruction to tell the computer to pick a random colour

```
random([‘darkcyan’, ‘olive’, ‘yellowgreen’])
```

- Try giving the circles random colours

Challenge 2 - Random circles



- Try finding your own colours to use
- Try having some things random and others not – for example you could pick a specific size for your circles
- See what happens if you change the range for the randomness

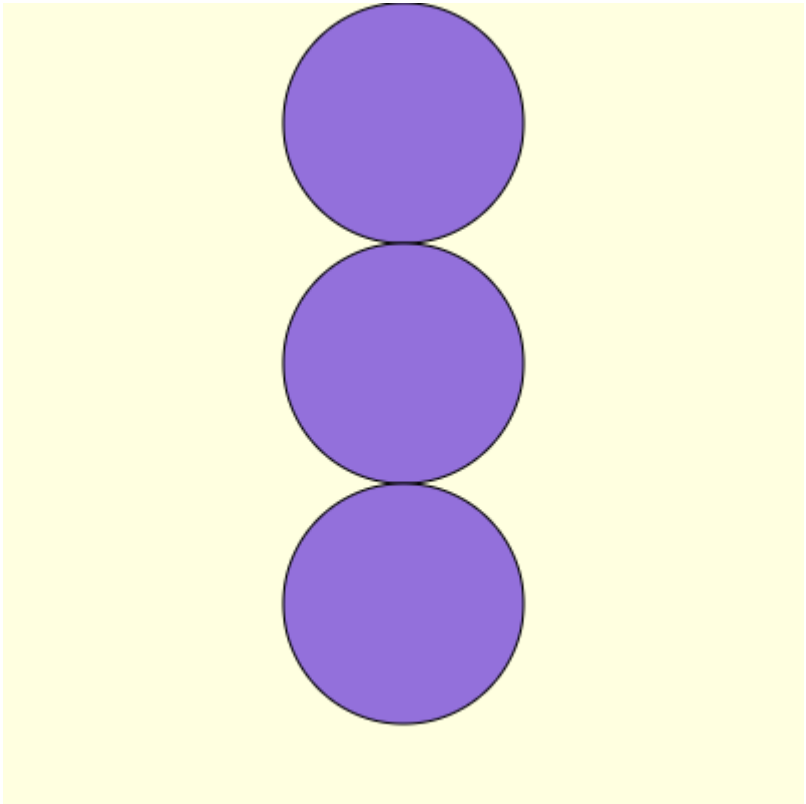
```
background('#94A537');
fill( random(['green', 'purple']) );
circle(random(800),random(500),random(100));
```

Variables

- A way of saving a value to use later
- Like a box where you can store something
- Built in variables:
  - `width` - the width of your canvas
  - `height` - the height of your canvas
  - `windowWidth` - the width of the browser window the canvas is in
  - `windowHeight` - the height of the browser window

```
var randomWidth = random(100,500);
ellipse(100,100,randomWidth);
ellipse(50,50,randomWidth);
```

Worked Example



Functions

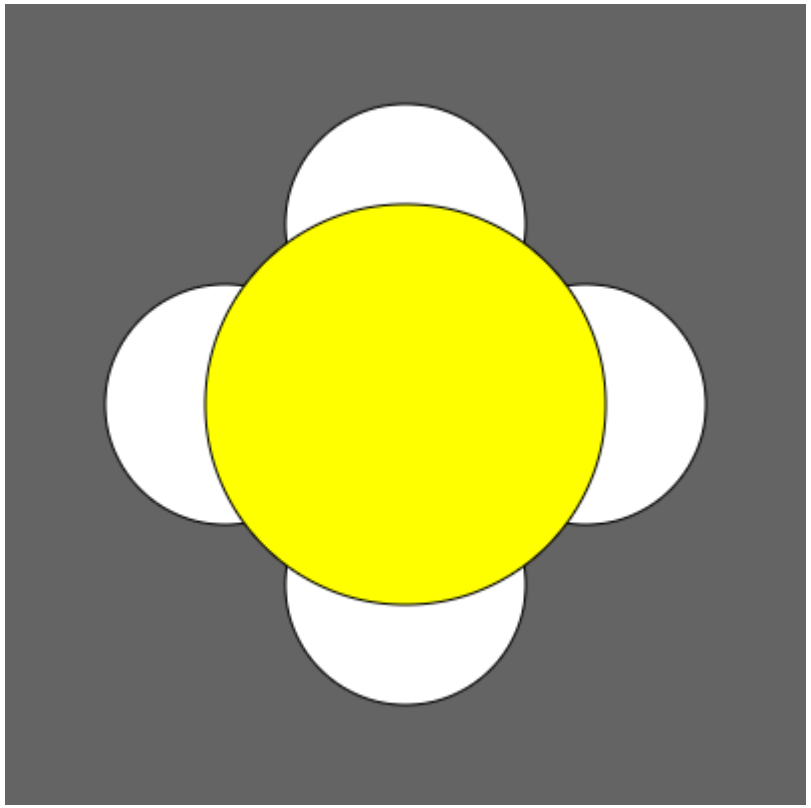
Grouping commands together

```
function randomCircle() {
  fill(random(['red','black']));
  ellipse(random(width),random(height),100);
}
```

Can give them 'arguments':

```
function randomCircle(diameter) {
  fill(random(['red','black']));
  ellipse(random(width),random(height),diameter);
}
```

Challenge 3 - Flower



- Try making this shape
- Try making its location random
- Try making its colour random
- Try putting it into a function

Loops

- Counts numbers
- We choose starting number
- We choose number it ends at
- We choose whether we count up, down, and how much we add or subtract each time
- Each time the loop counts it performs an action, the action can be based on the number the loop is at in the count

```
for (var size=200; size>0; size=size-50) {
  ellipse(200,200,size);
}
```

If this then that

- To choose whether or not to do something based on some condition we use an if-else statement
- This statement checks a condition and performs an action if that condition is met, otherwise (else) it does another action, it will look something like this:

```
if (random()>0.5) {
  doAnAction
}
else {
  doSomeOtherAction
}
```

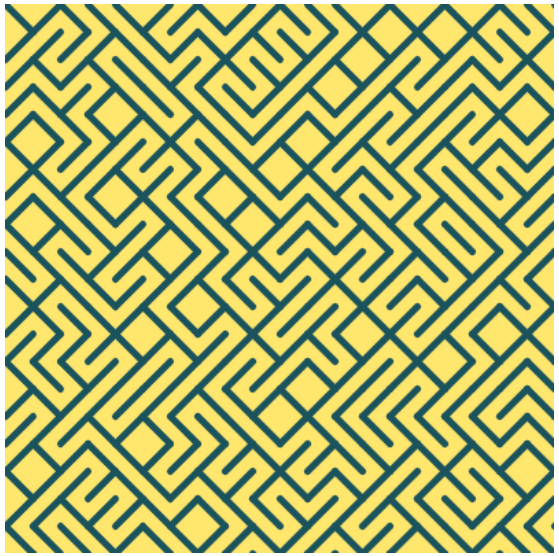
The above code will do each action about half the time

Drawing in a grid: Loop inside a loop

- One loops creates the "columns" or the x-values
- One loop creates the "rows" or the y-values
- The step is the height and width of each square in the grid

```
var step = 20;
for (var x=0;X<width;x+=step) {
  for (var y=0;y<height;y+=step) {
    rect(x,y,step,step);
  }
}
```

Final challenge



- Pattern created by randomly drawing either a left diagonal or right diagonal in a grid
- Play with the code and create variations

```
for (var x=0; x < width; x = x + 10) {
  for (var y=0; y < height;y = y + 10) {
    // Your line commands go here
  }
}
strokeWeight(3);
stroke("red");
line(startingX,startingY,endingX,endingY);
```