POLITECNICO
MILANO 1863

IMAGE AND SOUND
ISPG
PROCESSING GROUP

CREATIVE PROGRAMMING AND COMPUTING

Tools

PROGRAMMING WAYS…

- Compiled languages:
  - Compilation and running are in two separated phases

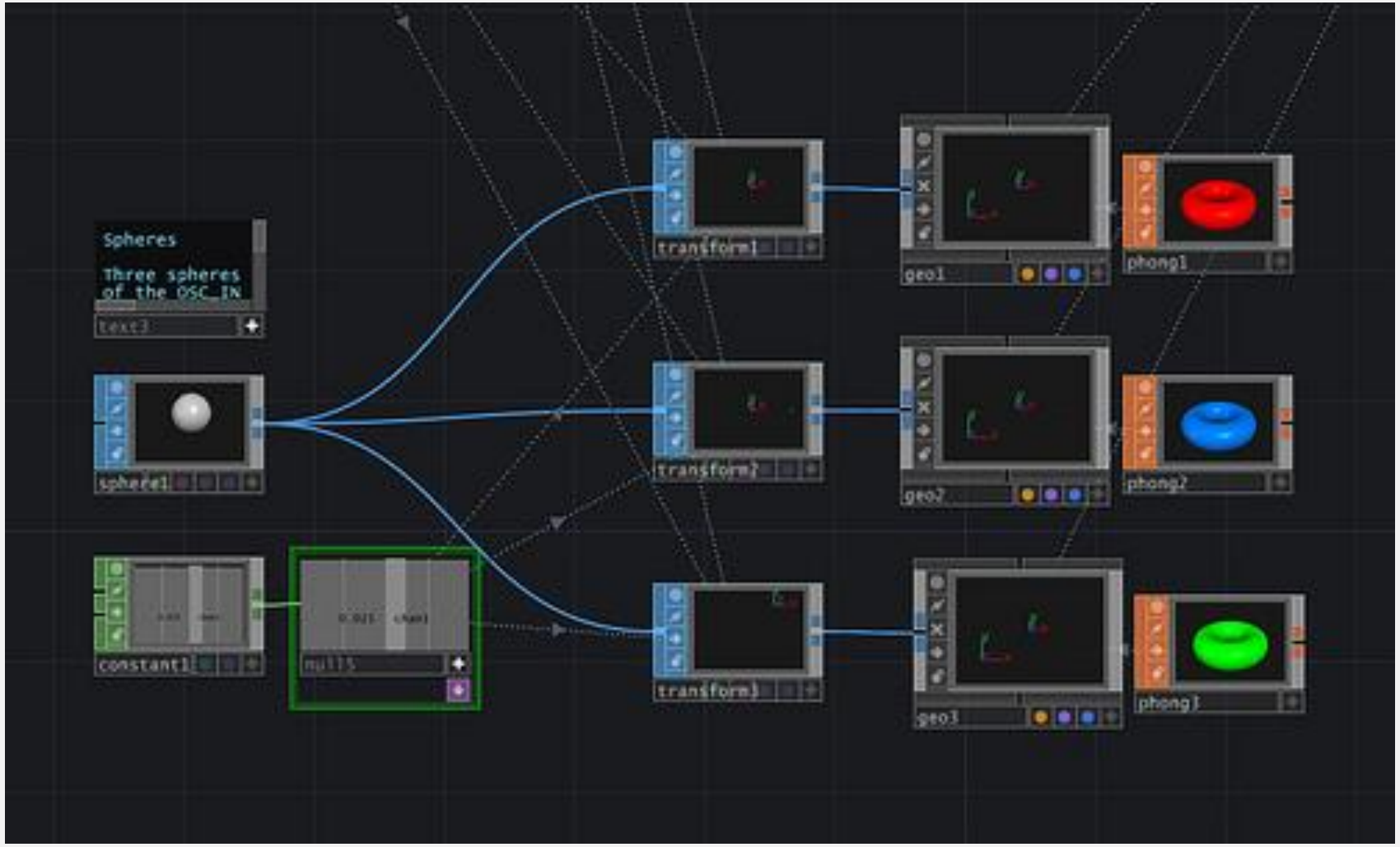
- Live coding languages (as SuperCollider):
  - Interpretation and running can be done in conjunction
  - It possible to upgrade the program during the execution and influence the execution
  - Very useful in live performances and interactive performances

Wobble Bass

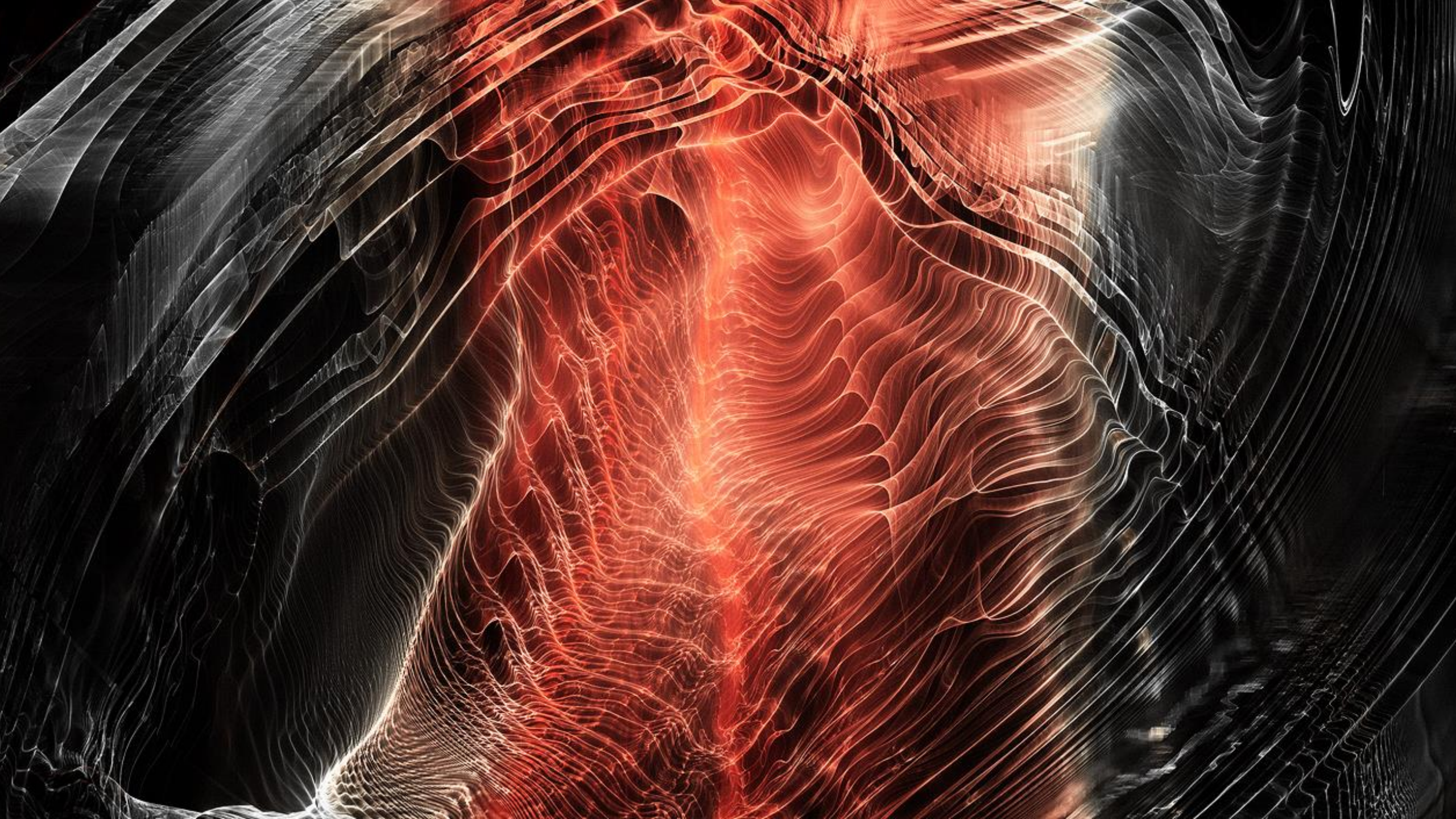
- Visual programming languages (as TouchDesigner): :
  - Not based textual coding, but on the linking of functional blocks

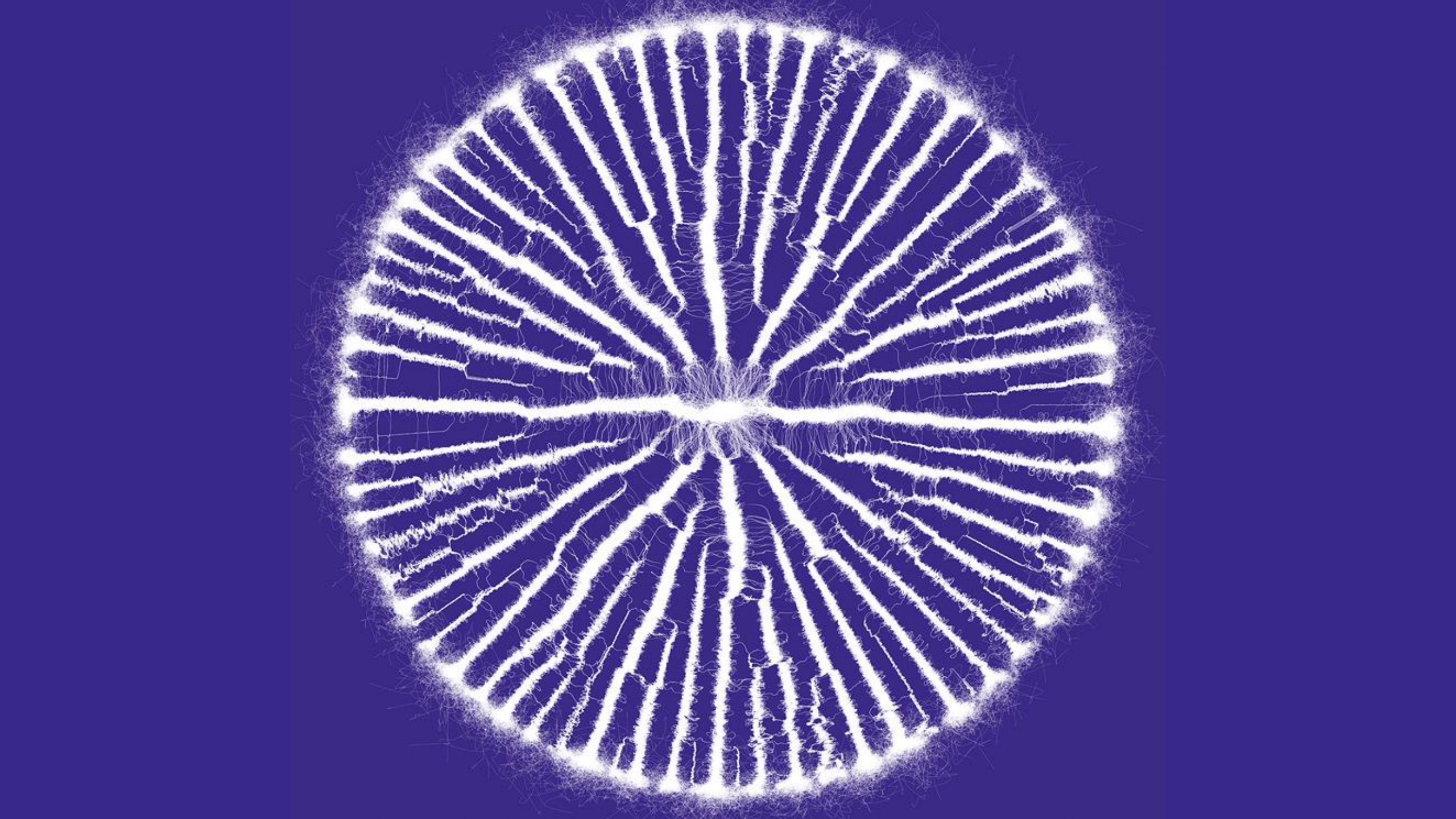
PROGRAMMING WAYS…

- **Visual programming languages: touchDesigner**



PROCESSING





Pointillism by Daniel Shiffman

INTRODUCTION

- Processing is a computer language originally conceived by Ben Fry and Casey Reas, students at the time (2001) at MIT

- A simple language for use by designers and artists so that they could experiment without needing an extensive knowledge of computer programming

- Processing is a Java-based language
- Even if Processing is not Java can integrate any pieces of Java code

- It is basically used for computer graphic, digital image elaboration, video processing, interaction design …and some sound processing

INTRODUCTION

- Processing sketches can be very easily exported to be run as standalone applications.
- You just need to use the command Export Application in the File menu.
- Applications can be exported for different operating systems, such as Linux, Windows and MacOsX.

- Furthermore, there is the possibility of developing applications for Android mobile Devices, to translate your code in JavaScript or Python or Raspberry Pi

BASIC ELEMENTS

Variables

Variable types has to be decleared

```
int myAge = 35
```

TYPE	SIZE	DESCRIPTION
boolean	1 bit	True or false
char	16-bits	Keyboard characters
byte	8 bits or 1 byte	0–255 numbers
int	32 bits	Integer numbers
float	32 bits	Real fractional numbers
color	4 bytes or 32 bytes	Red, Green, Blue, and Transparency
String	64 bits	Set of characters that form words

```
names or newNames or newPeopleNames
```

Name style convention

BASIC ELEMENTS

Casting

Casting transforms a type in another type

```
float dist = 3.5;
int x = int(dist);
```

Final value = 3

Decimals are omitted

```
float dist = 3.5;
String s = str(dist);
```

Final Value is the string 3.5

BASIC ELEMENTS

OPERATOR	USE	DESCRIPTION
+	op1 + op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2

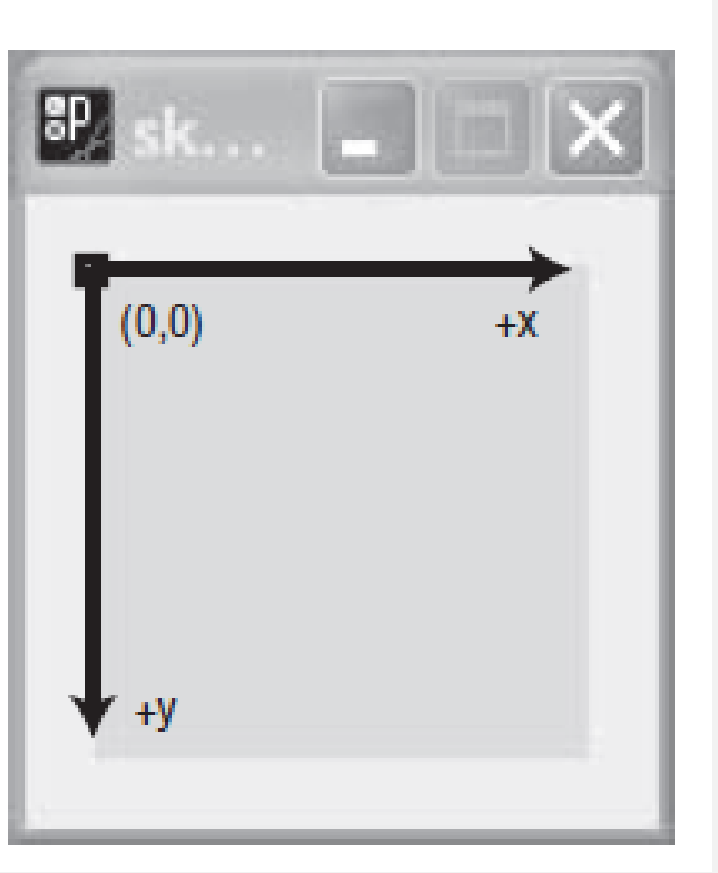
Math and logical Operators

OPERATOR	USE	RETURNS TRUE IF
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal
&&	op1 && op2	op1 and op2 are both true, conditionally evaluates op2
||	op1 || op2	either op1 or op2 is true, conditionally evaluates op2

PREPARE THE CANVAS

- Once the program starts a canvas will be open in a new window
- The size of the Canvas:
  - *size(width,height)* -> width and height are number of pixel
  - *fullScreen()*
  - *width* returns the width of the canvas
  - *height* turns the height of the canvas

- The background colour:
  - *background(R,G,B)* – Red, green, blue



CONTROLLING THE FLOW

- The code is divided in two sections setup() and draw()
- The **setup** section is used to define initial environment properties (e.g. screen size, background color, loading images/fonts, etc.) and
- The **draw** section for executing the drawing commands (e.g. point, line, ellipse, image, etc.)

- The code in the setup section will be executed only one time at the beginning
- Once the setup is executed the program will keep continue to cycle on draw section unless some run-control operator will modify the running

```
void setup(){
  size(300,100);
}

int i=0;
void draw(){
  line(i,0,i,100);
  i++;
}
```

CONTROLLING THE FLOW

- The speed of the draw loop is controlled by frameRate()
  - *frameRate(60)* -> 60 frame per second

- *noLoop()* placed in the draw section will freeze the flow
- *loop()* placed in the draw section will unfreeze the flow

Loop

```
boolean blocked = false;
float x=0;
float y=0;

void setup(){
  size(400,400);
  background(255,255,255);
  frameRate(10);
}


void draw(){
  fill(255,20,0, 200);
  circle(x,y,10);

  x=random(0,width);
  y=random(0,height);
}

void mousePressed(){

  if (blocked==true){
    blocked = false;
    loop();
  }
  else if (blocked==false){
    blocked = true;
    noLoop();
  }
}
```

GRAPHICS

- point() makes a point (i.e., a dot).
  - It takes two integer numbers to specify the location's coordinates, starting from the upper-left corner
  - *point(20, 30)*

- line() draws a line segment between two points
  - It takes four integer numbers to specify the beginning and end point coordinates.
  - *line(10,10, 120,200)*

- rect() draws a rectangle
  - It takes as parameters four integers to specify the x and y coordinates of the starting point and the width and height of the rectangle
  - *rect(10,10,100,50)*

GRAPHICS

- ellipse() draws an ellipse.
  - It takes four integers to specify the center point and the width and height.
  - *ellipse(30,30,50,20)*

- arc() draws an arc.
  - It takes four integers to specify the center point and the width and height of the bounding box and two float numbers to indicate the beginning and end angle in radians. For example,
  - *arc(50,50,70,70,0,PI/2)*

Arc

- curve() draws a curve between two points.
  - It takes as parameters the coordinates of the two points and their anchors
  - *curve(first anchor x, first anchor y, first point x, first point y, second point x, second point y, second anchor x, second anchor y)*

GRAPHICS

- bezier() draws a Bezier curve between two points
  - It takes as parameters the coordinates of the two points and their anchors
  - *bezier(first anchor x,first anchor y,first point x,first point y, second point x,second point y,second anchor x,second anchor y)*

- vertex() produces a series of vertices connected through lines
- It requires a beginShape() and an endShape() command to indicate the beginning and end of the vertices.

Curves

```
beginShape();
for(int i=0; i<20; i++){
  int y = i%2;
  vertex(i*10, 50+y*10);
};
endShape();
```

ATTRIBUTES

- Attributes modifies the appearance of geometrical shapes
- The are placed before the code of the shape in the code
- They have effect until a new attribute of the same type is called

```
fill(155,0,0);
rect(10,10,100,100);
```

- fill() will fill a shape with a color.
  - It takes either one integer number between 0 and 255 to specify a level of gray (0 being black and 255 being white) or three integer numbers between 0 and 255 to specify an RGB color.
  - It can take also the alpha value a the last parameter
  - *fill(255,0,0,150)*

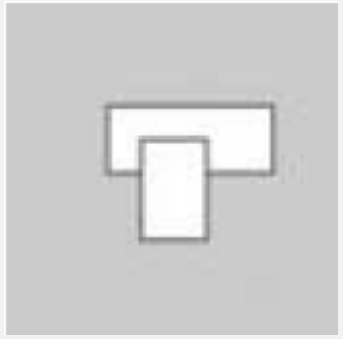
- noFill() will not paint the inside of a shape

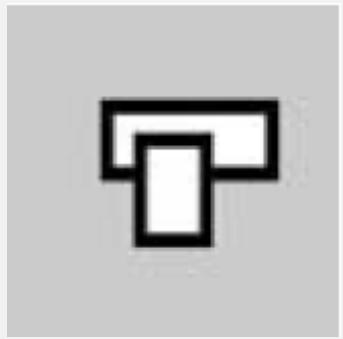
ATTRIBUTES

- stroke() will paint the bounding line of a shape to a specified gray or color.
- It takes either one integer number between 0 and 255 to specify a level of gray (0 being black and 255 being white) or three integer numbers between 0 and 255 to specify an RGB color.

- noStroke() will draw no boundary to the shape.

- strokeWeight() will increase the width of the stroke.
  - It takes an integer number that specifies the number of pixels of the stroke's width





CONDITIONS AND LOOPS

- *if … then … else …* in Processing

```
if( condition )
        …;

    else
      …;
```

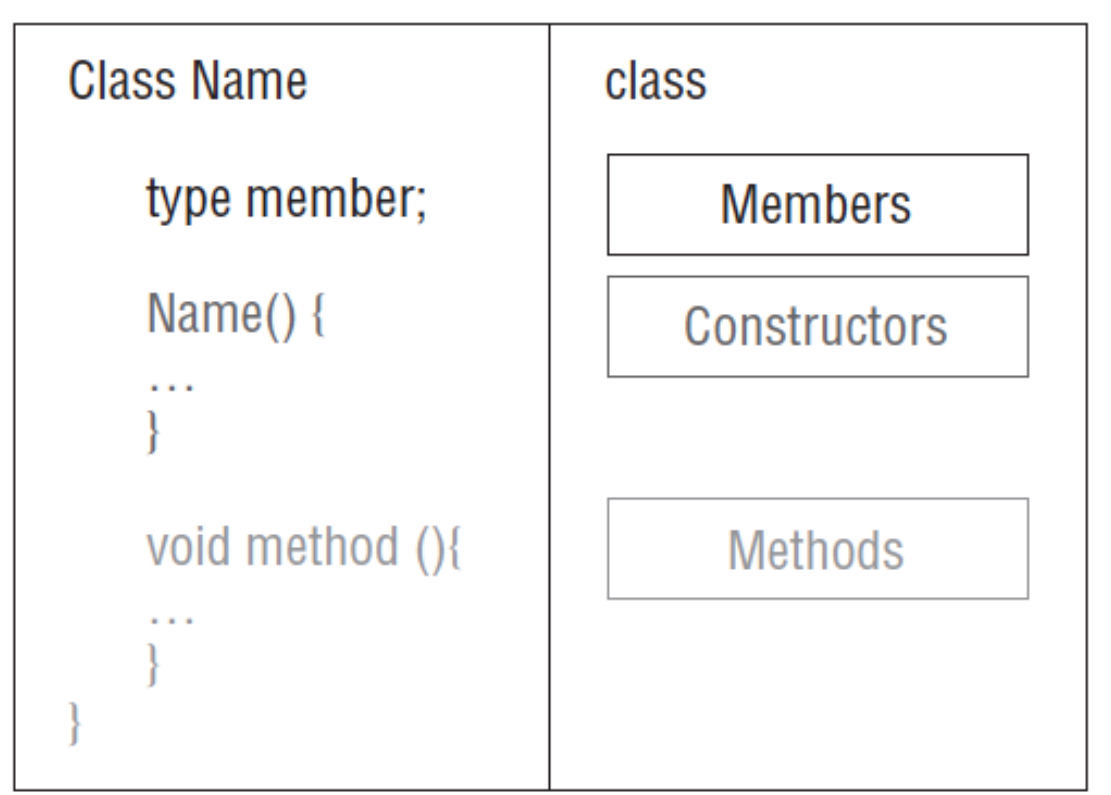
- Two loops (for, while)

```
for(int i=0; i<10; i=i+1){
     println(i);  // will printout the value of i
}
```

```
int i=0;
while(i<10){
     println(i);  // will printout i
     i++;
}
```

CLASSES AND OBJECTS

- Processing is a OO language




```
class MyPoint {
    float x, y;  // members of class

    // Constructor
   public MyPoint(float xin, float yin){
       x = xin;
       y = yin;
    }
    //Method1
    public void plot(){
        rect(x,y,5,5);
    }
    // Method2
    void move(float xoff, float yoff){
        x = x + xoff;
        y = y + yoff;
    }
}
```

CLASSES AND OBJECTS

- Create an Object

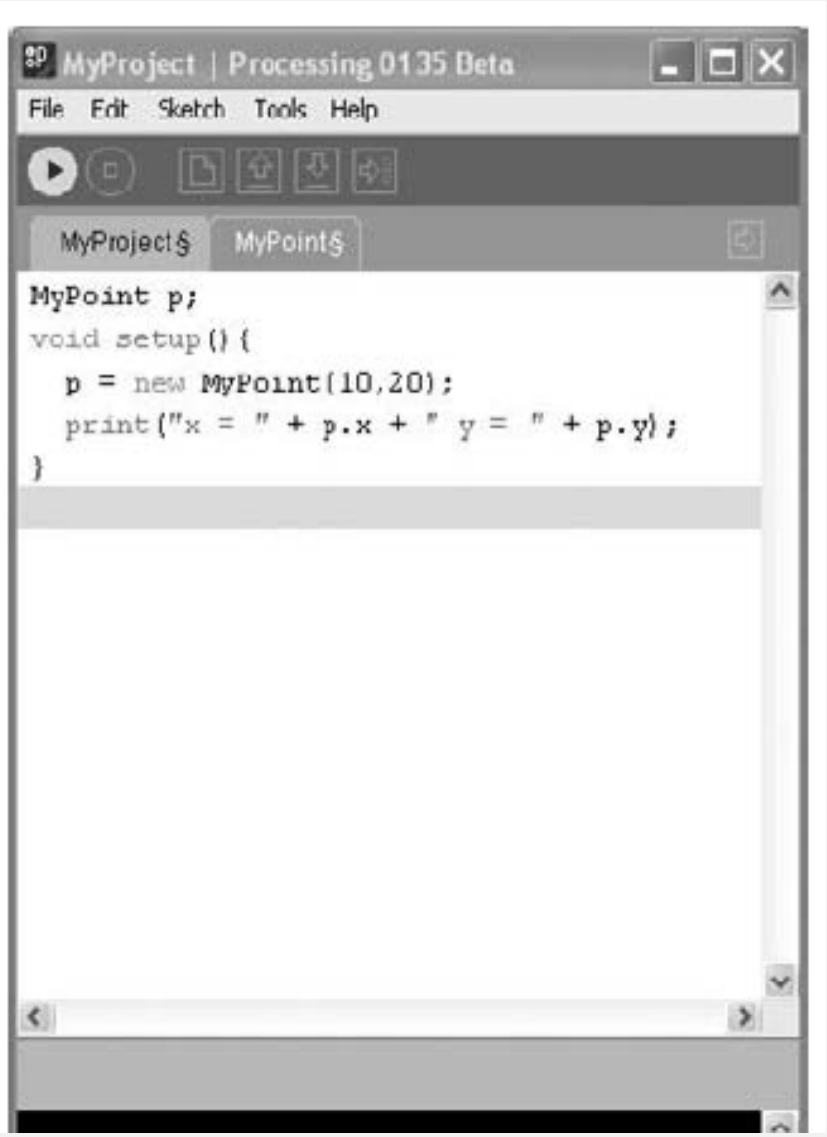
```
MyPoint p = new MyPoint(10, 20);
```

- Call attributes and methods

```
p.x or p.y or p.plot();
```

- Each class has to be in a new tab with the same name of the class

PointClass



ANIMATIONS

- It possible to easily create animation in Processing
- Each "draw" loop is a frame
- For each frame
  - We can use background to clean the canvas at each frame
  - We readraw static object in the same position
  - We readraw moving object in the position
- The higher is the frame rate, the more fluid is the animation

PointAnimation

DATA STRUCTURES

Array

```
String [] names = {"Kostas", "Ivan", "Jeff", "Jie Eun"};
float []  temperatures = {88.9, 89.1, 89.0, 93.4, 95.2, 101.2};
int [] num_of_transactions = new int[50];
boolean [] isOff;
```

```
int twoDArray[][] = new int[5][100];
```

Initialization

Few java-like attributes for arrays

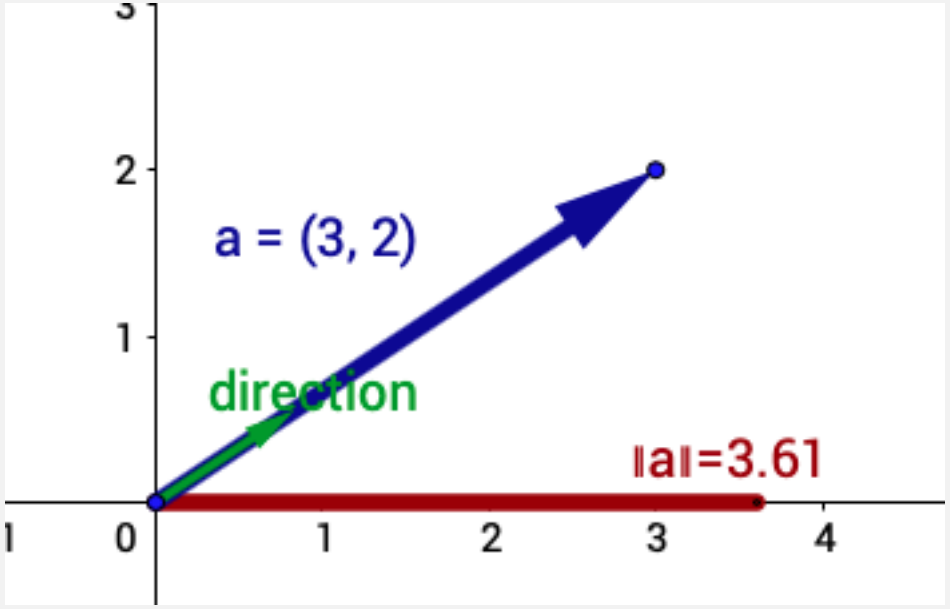
- *array_length*: return the number of elements of an array
- *sort(array)*: sorts the elements in alphabetic order
- *append(array, data)*: expand an array by one element and adds data value
- *subset(array, offset, length)*: creates a subset of an array at offset for a specific length
- *expand(array, size)*: expands an array by a total size position retaining the existing data values

DATA STRUCTURES

Processing includes some useful predefined Class/Data Structures

PVector

- *PVector(x, y, z)*
- A class to describe a two- or three-dimensional vector, specifically a Euclidean (also known as geometric) vector
- A vector is an entity that has both magnitude and direction
- *mag():* Calculate the magnitude of the vector
- *heading():* Calculate the angle of rotation for this vector
- *rotate(angle):* Rotate the vector by an angle
- n*ormalize():* Normalize the vector to a length of 1
- …



FONTS

- createFont() create a font to use in the graphic canvas
  - It takes the name of a font and the size. The name of the font should correspond to one of those that installed in the computer
  - It return a PFont object

```
PFont myFont = createFont("Times", 32);
textFont(myFont);
text("P",50,50);
```

- textFont() load the created font
  - It takes the PFont object

- text() display the text object
  - It takes the string to display and the coordinates of the position

IMAGES

- loadImage() will load and display an existing image
  - It takes the location of the image
  - Returns a PImage object

- image() will display a Pimage
  - It takes the coordinates of the left-upper corner and eventually width and height

```
PImage myImage = loadImage("c:/data/image.jpg");
image(myImage, 0, 0);
```

INTERACTION

- Mouse and keyboard activities can be captured in real-time
- *mouseX* and *mouseY:* return the actual mousecoordinates
- *mousePressed():* is a callback called when a mouse button is pressed
- *mouseDragged():* is a callback called when a mouse is dragged
- *mouseMoved():* ia a callback called when the mouse in movement
- *mouseReleased():* is a callback called when a mouse button is released
- *keyPressed():* is a callaback called when a key is pressed on the keyboard

PointAnimationMouse

ImagesMouse

```
void setup(){

}

void draw(){

}

void mousePressed(){

}

void keyPressed(){

}
```

CAMERAS

- It easily possible to capture the video from the built-in camera of the computer
- Use the class *Capture* in the standard library *import processing.video.**
- Some controls:
  - *available()* Returns "true" when a new video frame is available to read
  - *start()* Starts capturing frames from the selected device
  - *stop()* Stops capturing frames from an attached device
  - *read()* Reads the current video frame

```
  String[] cameras = Capture.list();

  if (cameras.length == 0) {
    println("There are no cameras available for capture.");
    exit();
  } else {
    println("Available cameras:");
    for (int i = 0; i < cameras.length; i++) {
      println(cameras[i]);
    }

    // The camera can be initialized directly using an
    // element from the array returned by list():
    cam = new Capture(this, cameras[0]);
    cam.start();
  }
}

void draw() {
  if (cam.available() == true) {
    cam.read();
  }
  image(cam, 0, 0);
  // The following does the same, and is faster when just drawing the image
  // without any additional resizing, transformations, or tint.
  //set(0, 0, cam);
}
```

Camera

AUDIO

- Processing Standard library also includes sound elaboration
- It allows to generate sounds (sound synthesis)
- It allows to load, play and elaborate audio files
- It allows to capture and elaborate audio in real-time

Configuration

Sound

I/O

AudioIn

Sampling

SoundFile
AudioSample

Effects

LowPass
HighPass
BandPass
Delay
Reverb

Noise

WhiteNoise
PinkNoise
BrownNoise

Oscillators

SinOsc
SawOsc
SqrOsc
TriOsc
Pulse

Envelopes

Env

Analysis

Amplitude
FFT

Voice_intensity

Re-think the Net by O3LAB

MATERIALS

- **References**
  - https://processing.org/tutorials/
  - https://processing.org/reference/environment/
  - https://processing.org/reference/

- **Further readings**
  - Casey Reas and Ben Fry. **Processing: A Programming Handbook for Visual Designers, Second Edition**, The MIT Press, 2014
  - https://processing.org/reference/libraries/

WEKINATOR

INTRODUCTION

- Open source software originally created in 2009 by Rebecca Fiebrink
- It is a black box containing several Machine Learning algorithms

- Inputs are OSC messages
- Output are OSC messages

Anything that can output OSC can be used as a controller

Anything that can be controlled by OSC can be controlled by Wekinator

- Allows users to build new interactive systems without writing code

Download

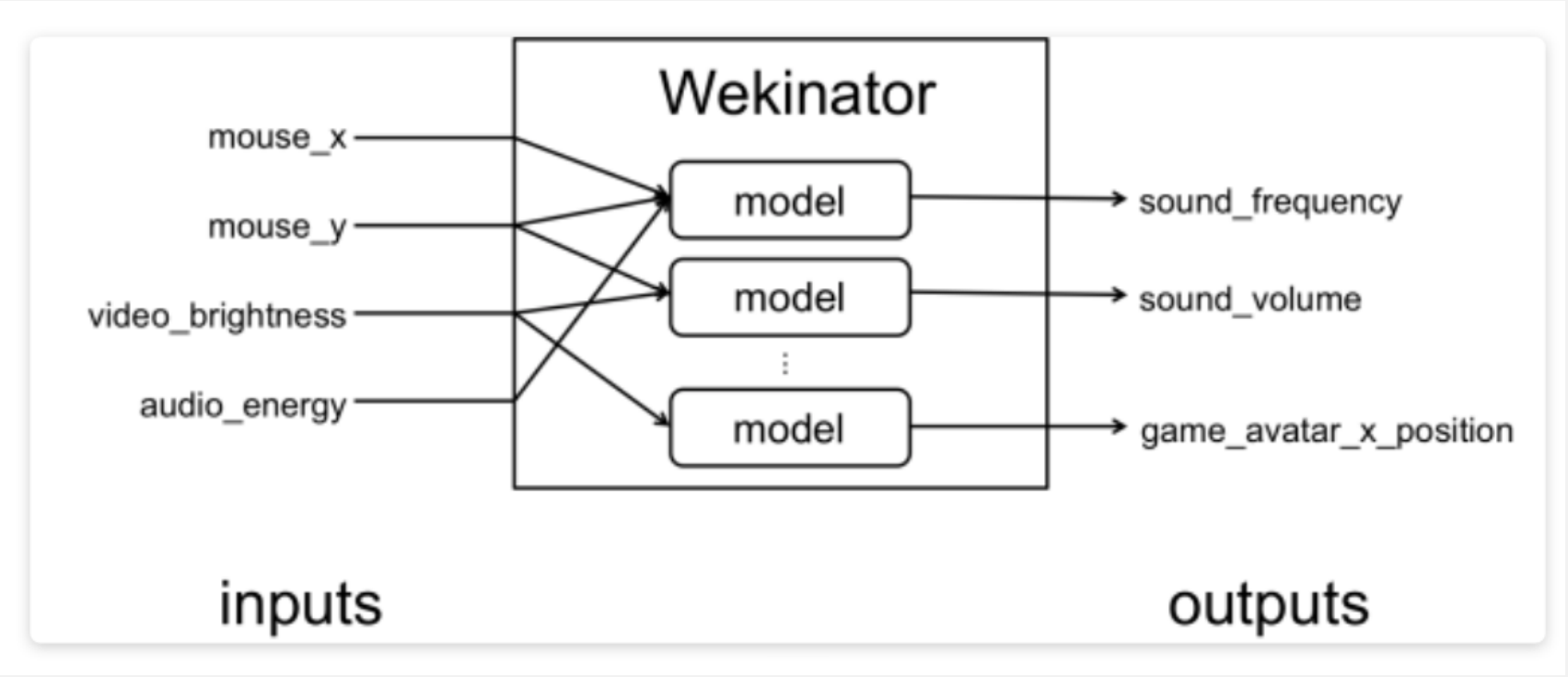
http://www.wekinator.org/downloads/

INTRODUCTION

- Creation of new musical instruments
- Create mappings between gesture and computer sounds. Control a drum machine using your webcam! Play Ableton using a Kinect!
- Creation of gesturally-controlled animations and games
  - Control interactive visual environments created in Processing, OpenFrameworks, or Quartz Composer, or game engines like Unity, using gestures sensed from webcam, Kinect, Arduino, etc.
- Creation of systems for gesture analysis and feedback
  - Build classifiers to detect which gesture a user is performing. Use the identified gesture to control the computer or to inform the user how he's doing.
- Creation of real-time music information retrieval and audio analysis systems
  - Detect instrument, genre, pitch, rhythm, etc. of audio coming into the mic, and use this to control computer audio, visuals, etc.
- Creation of other interactive systems in which the computer responds in real-time to some action performed by a human user (or users)

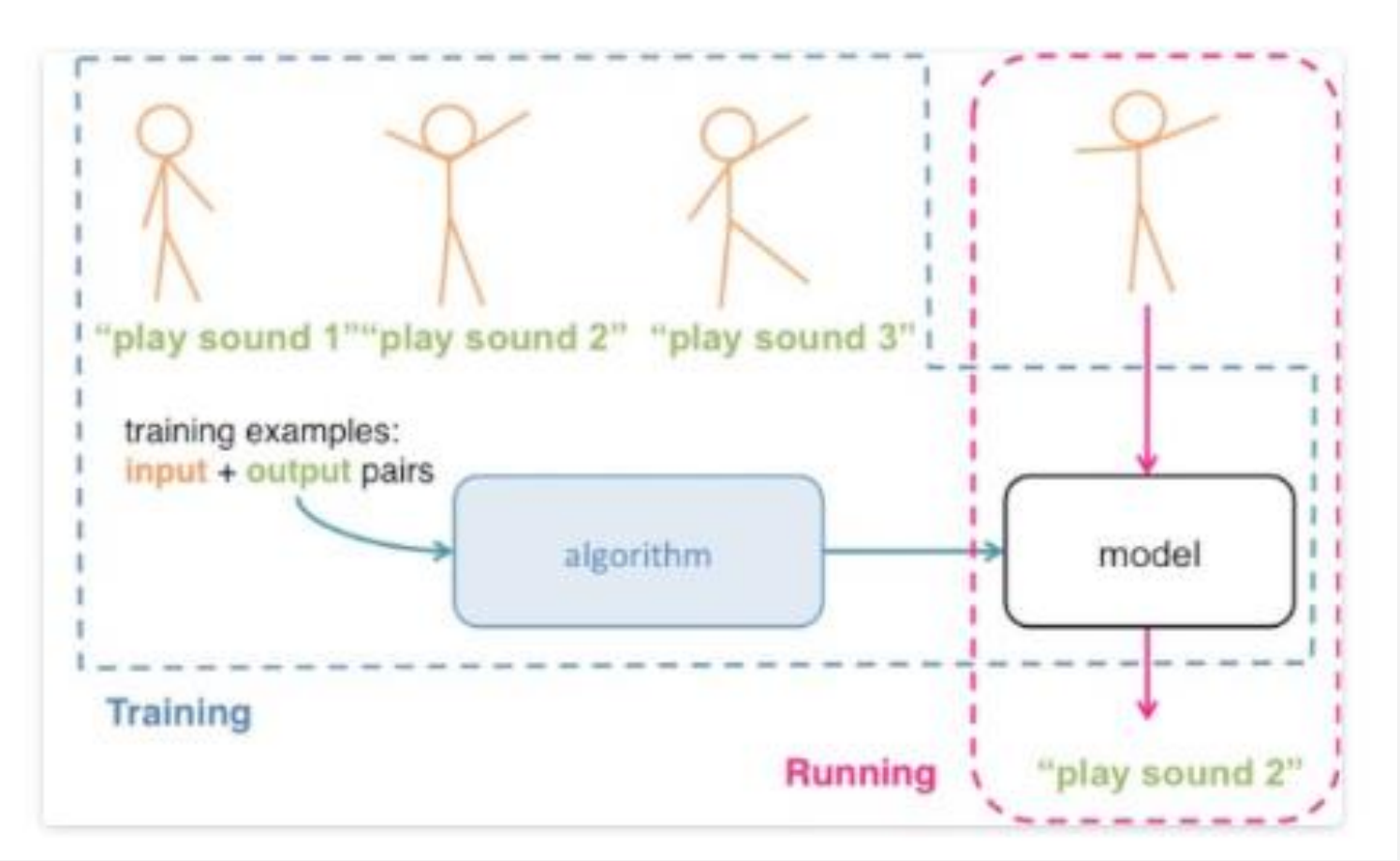
INTRODUCTION

- Inputs could be sent from any **real-time input**, such as game controllers, webcam input, motion tracking, audio input, sensors connected to Arduino, etc.
- Outputs could be sent to any **real-time process**, such as music (Max/MSP, ChucK, PD, SuperCollider, Ableton/Max4Live…), animation (Processing, OpenFrameworks, …), games (Unity3D), robots or physical actuators (e.g. using Arduino), etc.



INTRODUCTION

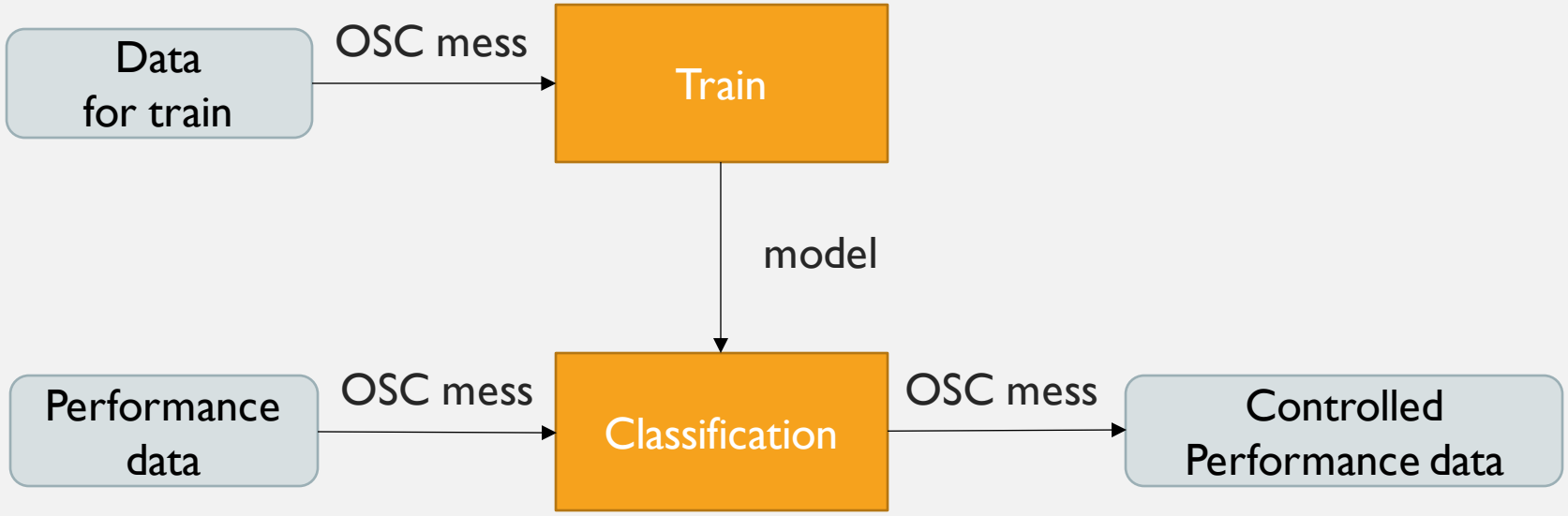
- Wekinator builds these models using **supervised machine learning algorithms**



OSC

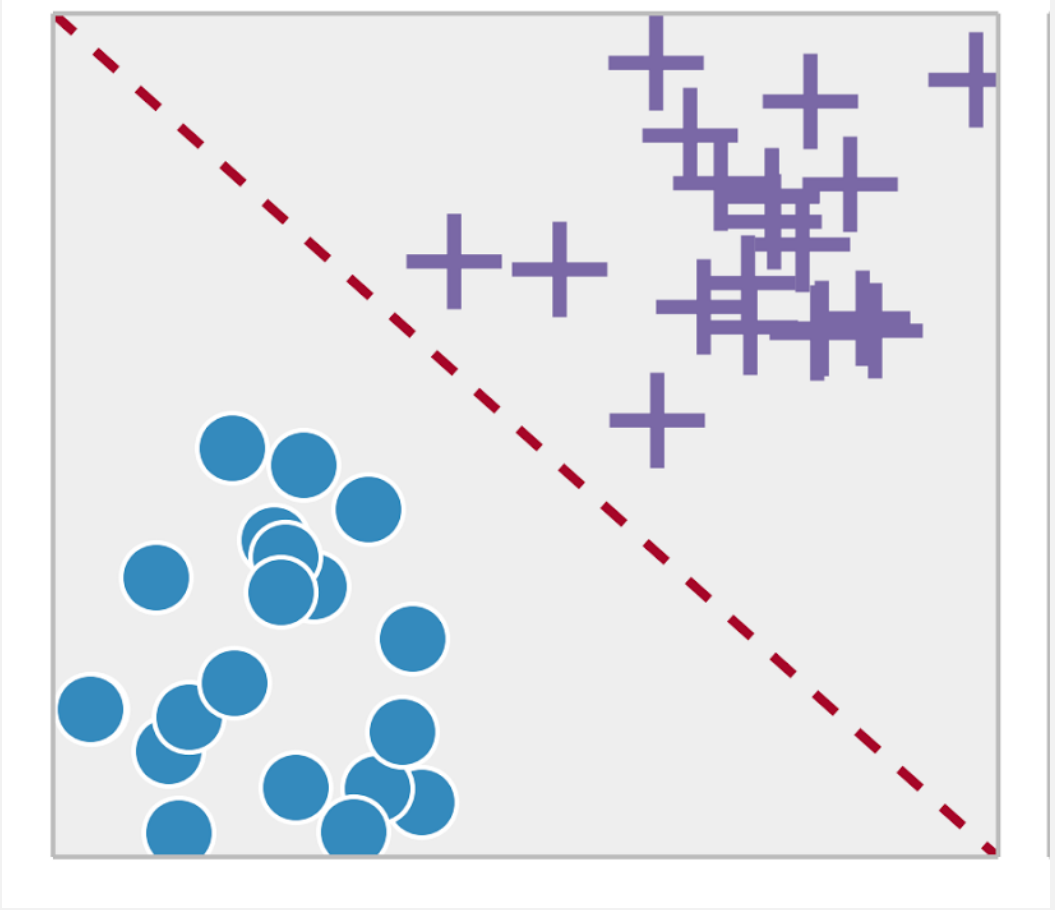
- Open Sound Control (OSC) is a protocol for real-time communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology.
- Network-based system utilizes common UDP/TCP transport mechanisms
- A message is composed of an
  - OSC **Address Pattern** followed by an
  - OSC **Type Tag String** followed by
  - zero or more OSC **Arguments**

TRAIN AND USE A MACHINE

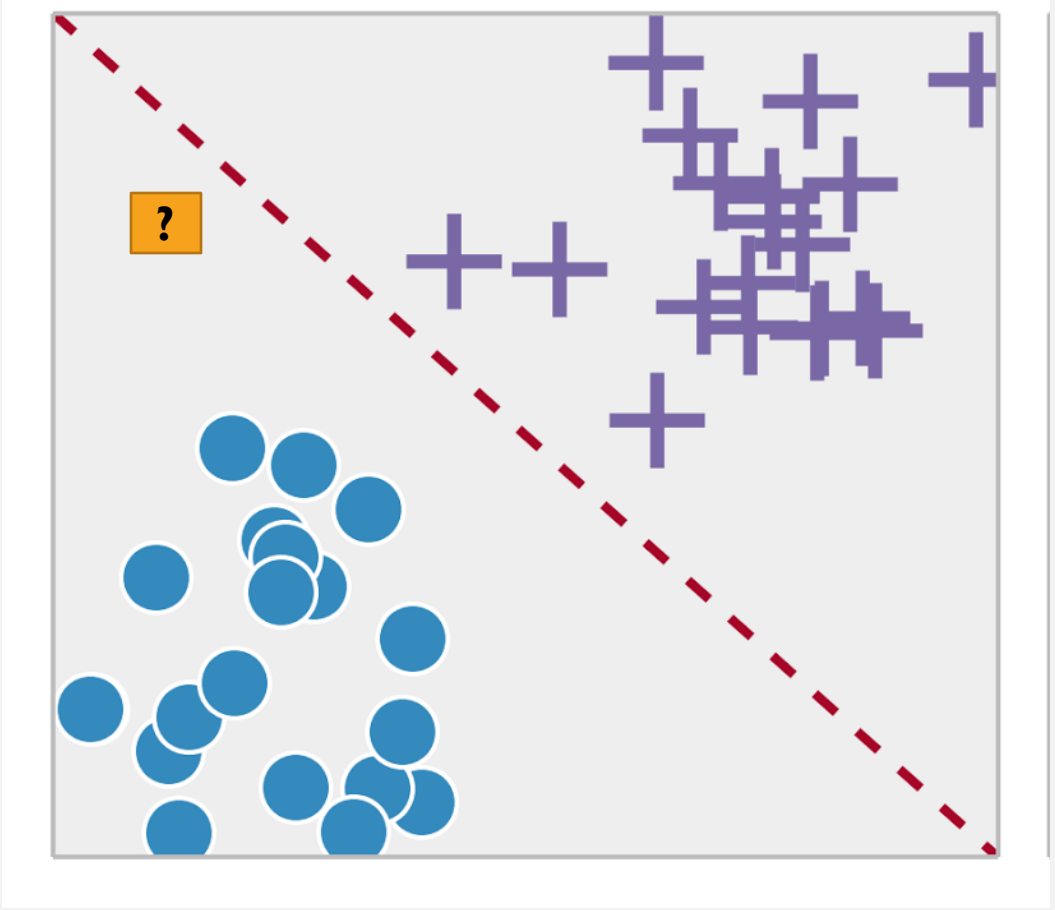


CLASSIFICATION

Train

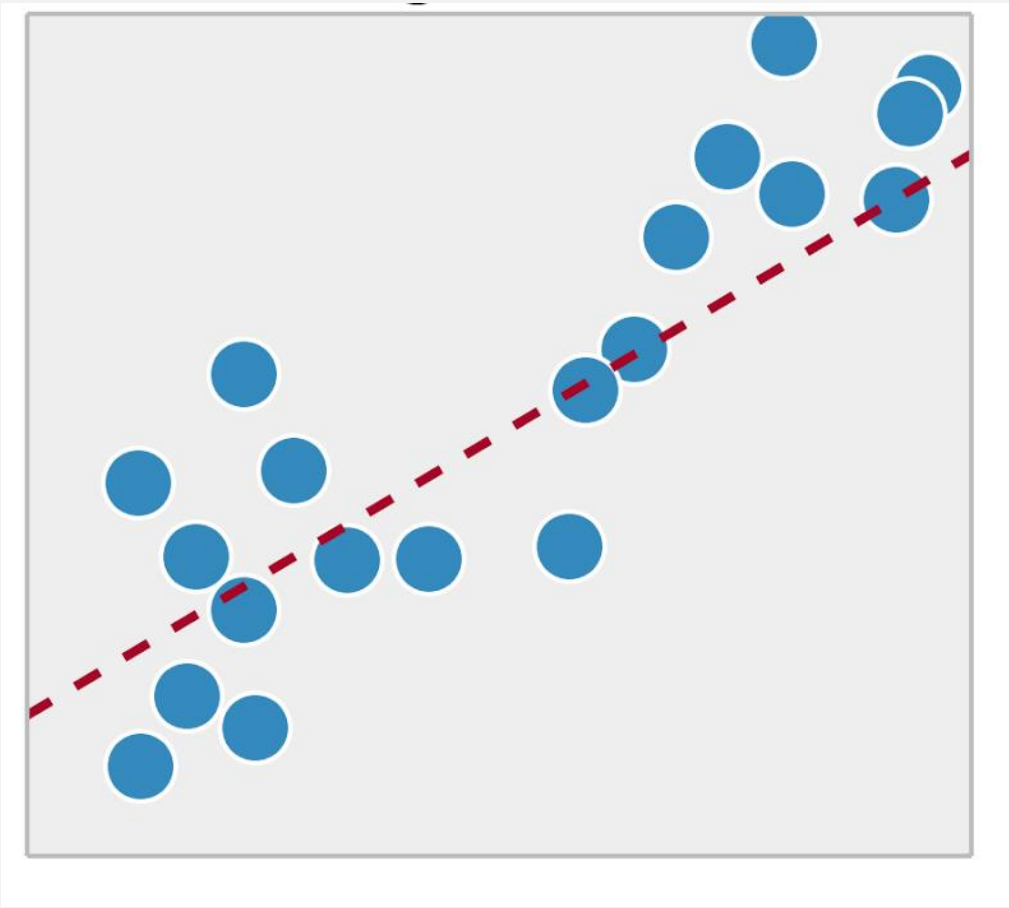


Performance

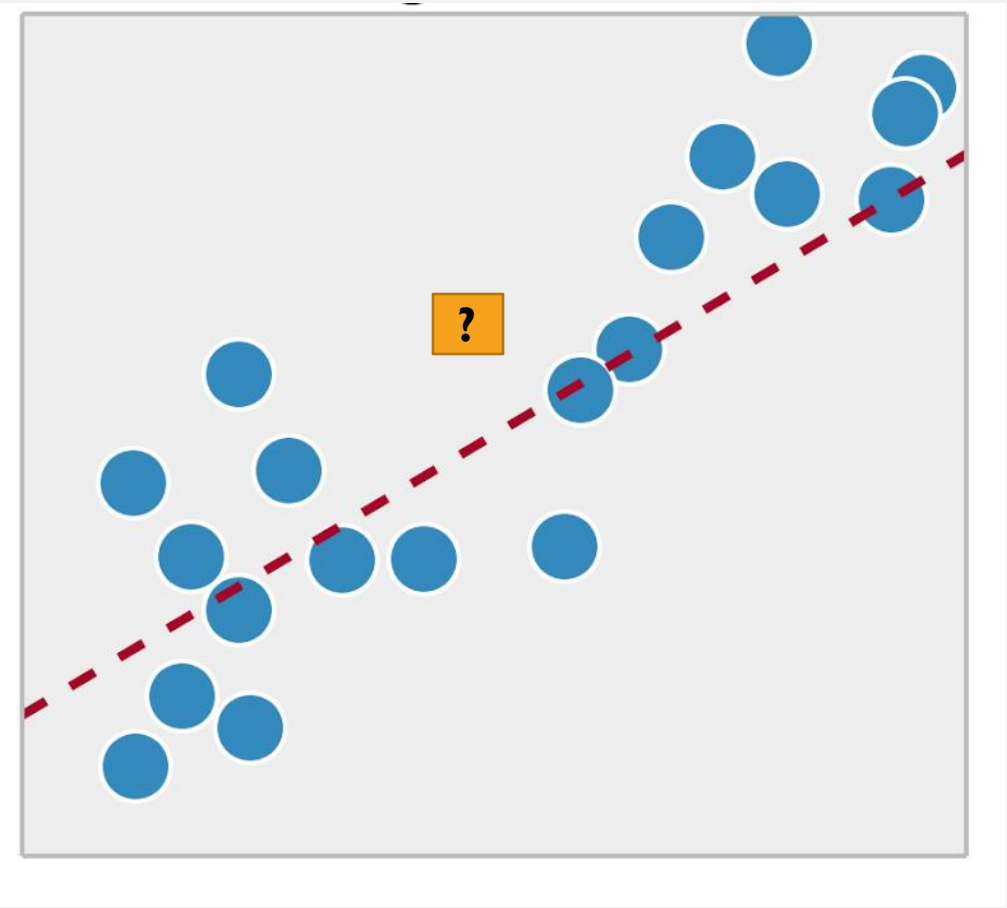


REGRESSION

Train



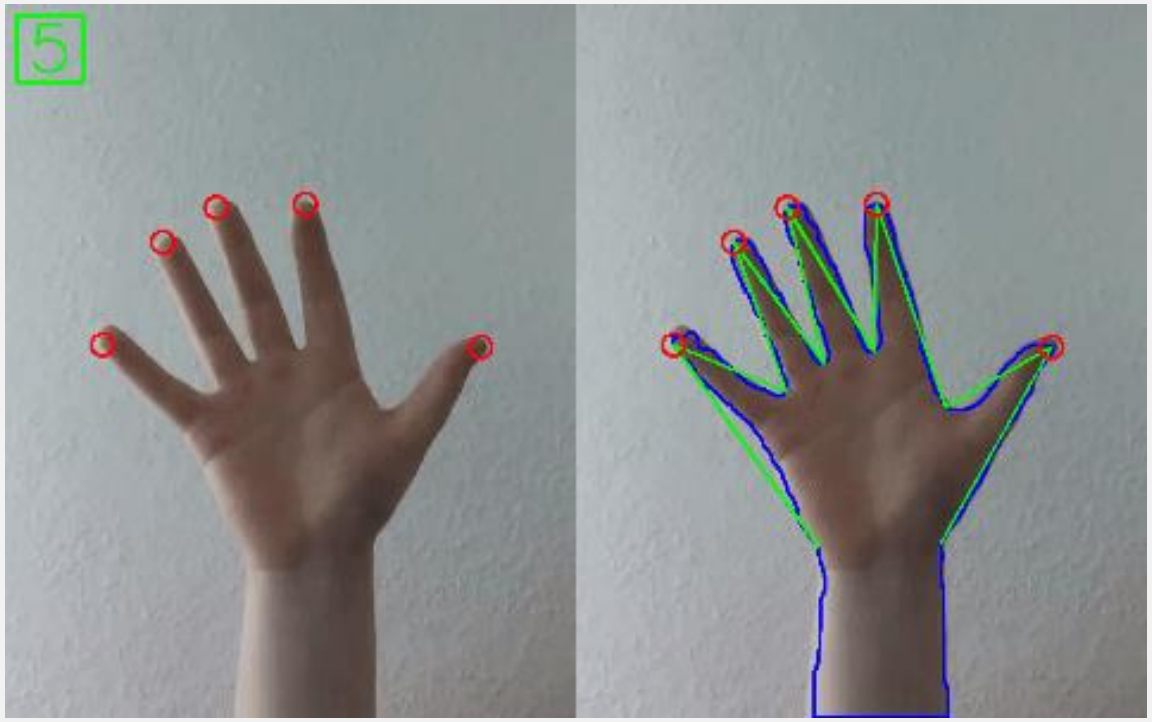
Performance



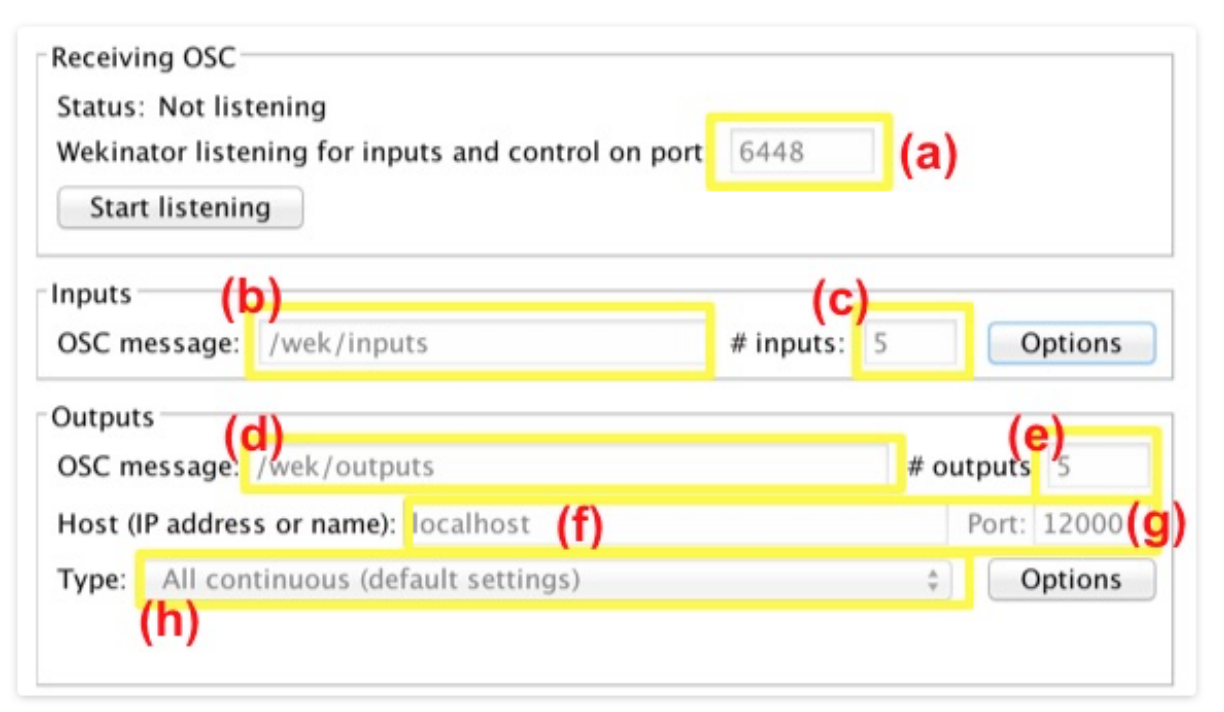
DYNAMIC TIME WARPING

- In time series analysis, dynamic time warping (DTW) is a measure of the similarity between two temporal sequences, which may vary in speed
- Let's consider the evolution of data of the performance along the time –> time series
- Gestures or movement in general are time series
- Give a set of time time-series representing the training data, we can use DTW to compare them we the performance data in real-time

- Application: gesture recognition
- It is possible to control a system by gesture

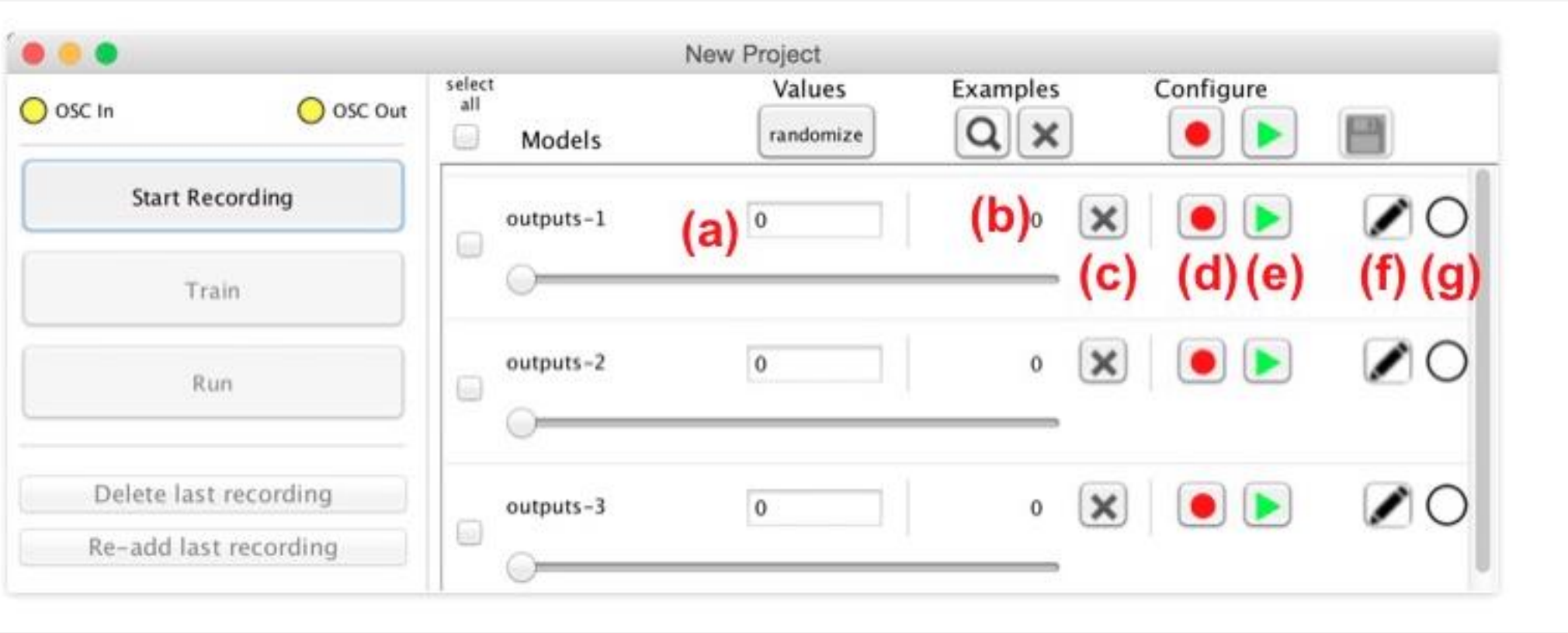


TRAIN AND USE A MACHINE



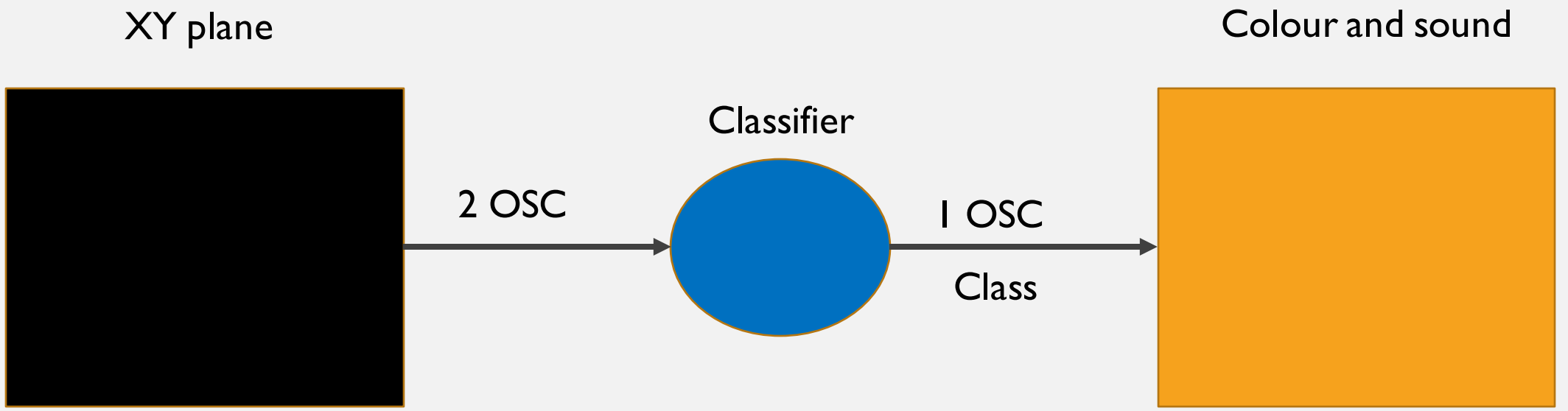
The port number your inputs are being sent to (a)
The message name you're using to send inputs (b)
The number of inputs (c)
The message name to use for sending outputs (d)
The number of outputs (e)
The host name and port number to send outputs to (f, g)
The type of each output (read about output types below) (h)

TRAIN AND USE A MACHINE



- manually set the value of each output using its text box and/or slider (a)
- view the number of examples currently recorded for that model (b)
- delete all examples for a model (c)
- disable recording of new examples for that model (d)
- disable computation for that model (e)
- edit the model (f)

EXAMPLE 1 - CLASSIFICATION

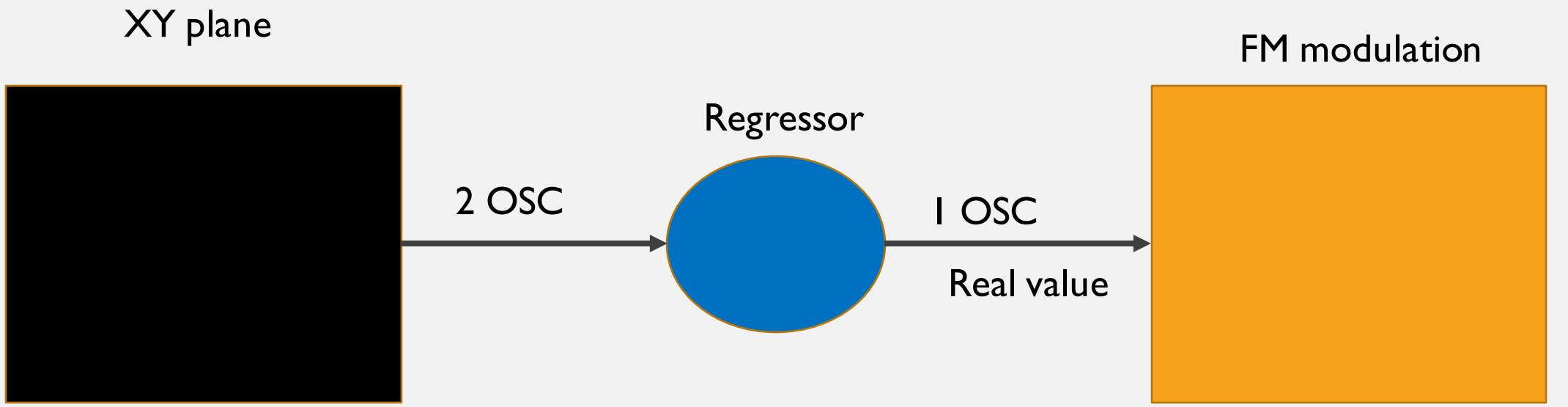


Able to place point in the XY plane

Based on the incoming value colour and sound changes

Linear_Polynomial_Classification

EXAMPLE 1 - REGESSION



Able to place point in the XY plane

Based on the incoming value
colour and sound changes

Linear_Polynomial_Regression

SUPERCOLLIDER - REVIEW

INTRODUCTION

SuperCollider is an environment and programming language for

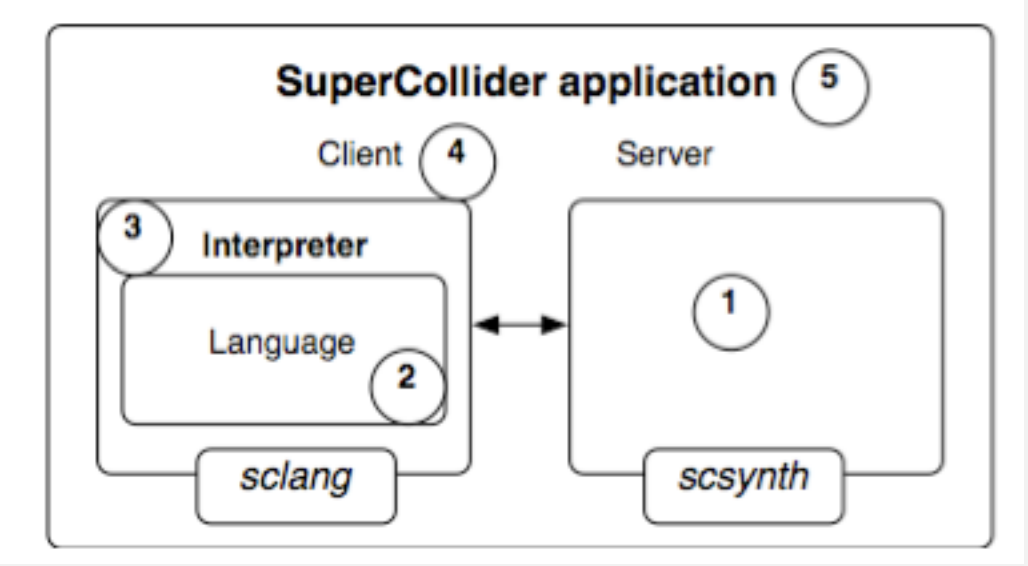
- Creative coding
- Real time audio synthesis
- Sound Design
- Interaction Design
- Algorithmic composition
- Instrument building
- Other ….

- object-oriented language
- Client/server-based framework

INTRODUCTION

SuperCollider is composed by:

- an audio server
- an audio programming language
- an interpreter for the language, i.e. a program able to interpret it
- the interpreter program as a client for the server
- the application including the two programs and providing mentioned functionalities



WHY SUPERCOLLIDER ?

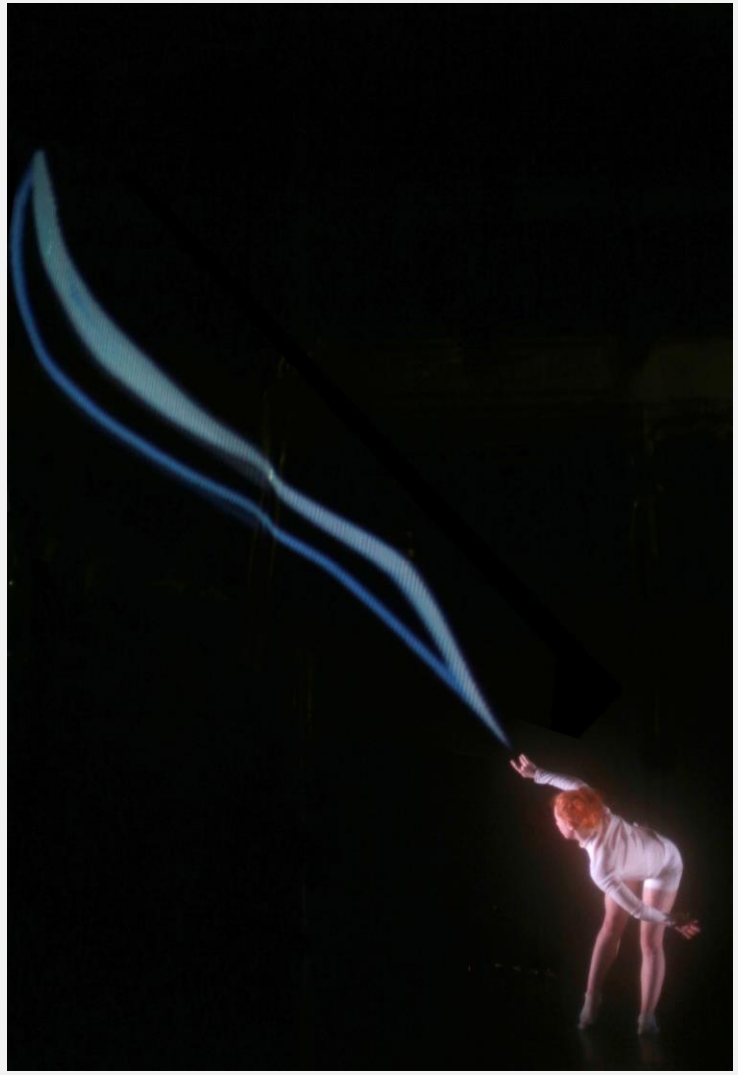
- Client/server Architecture -> it works over networking protocols
- It's open source –> easily extendable in SP or C++
- Many extensions to integrate it and interact with many platforms and frameworks
  - Open Framework
  - Processing
  - SuperColliderAU
  - Ableton Live
- Able to communicate with many tools via OSC protocol
  - Max/MSP
- Easily used as language for creative coding and signal processing

It's FREE

WHAT CAN WE DO WITH SC?

- In conjunction with The Wire magazine, a unique collection: 22 pieces by artists from around the world, each piece created with just 140 characters of code
- https://archive.org/details/sc140/01_nathanielvirgo.mp3

- NEURAL NARRATIVES 2: POLYTOPYA by Pablo Palacio
- https://vimeo.com/126729369



WHAT CAN WE DO WITH SC?

- **The three faces of Supercollider**

- **General purpose OOP Language**
  - it can be seen as an efficient OOP scripting language for signal processing

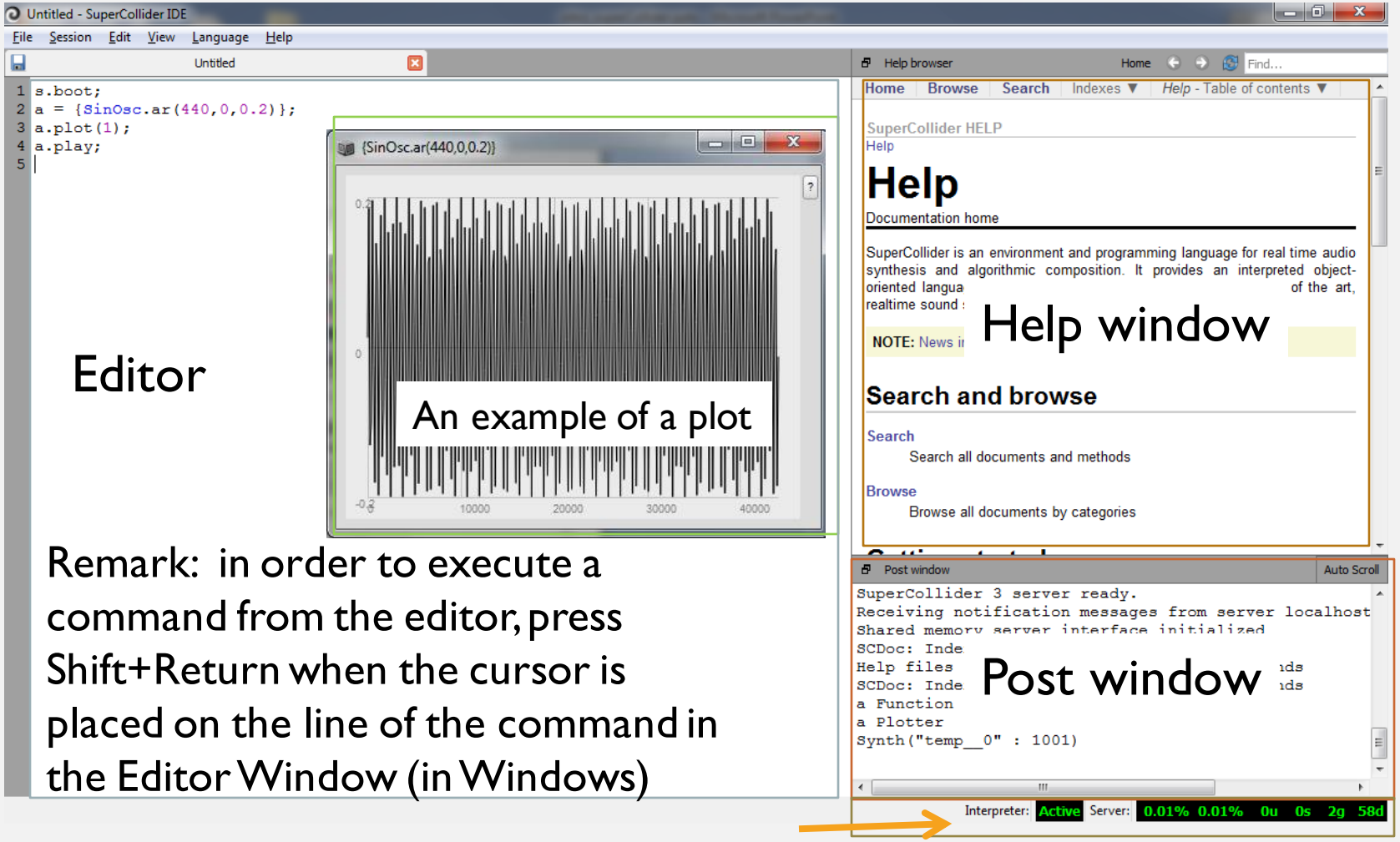
- **Powerful tool (language) for sound synthesis**
  - build instruments di interact with internal or external controllers
  - design sounds and effects

- **OOP Language for electronic music composition**
  - the evolution of the composition is controlled by the script it self

RUN THE MACHINES

- “Hello World !!!”
- To execute the code
  - Select and [shift+Enter] Mac OSX
  - Select and [ctrl+Enter] Windows
  - Select and [ctrl+c] Linux
- To stop an execution
  - [cmd+.] Mac OSX
  - [alt+.] Windows
- To run the server
  - Server.default = s = Server.internal.boot
- To clear the post window
  - [shift+cmd+p] Mac OSX
  - [ctrl+alt+p] Windows

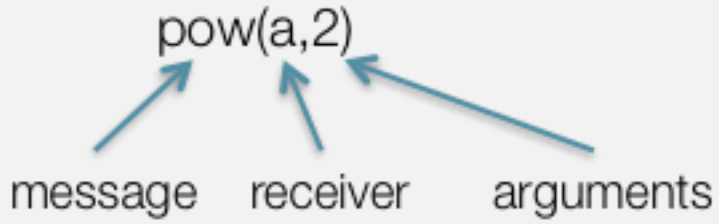
IDE



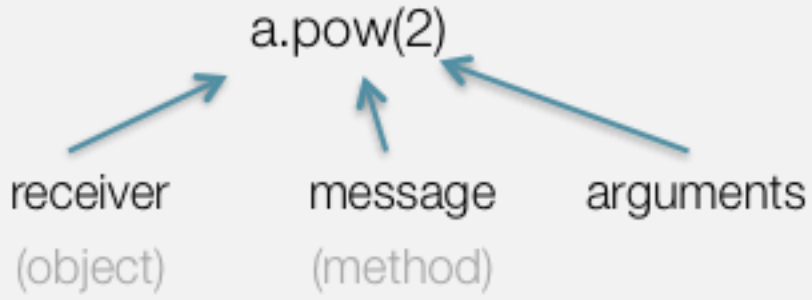
Remark: in order to execute a command from the editor, press Shift+Return when the cursor is placed on the line of the command in the Editor Window (in Windows)

THE LANGUAGE

- Supercollider is an Object-Oriented language
- It follows the receiver-massage-arguments paradigm
- Messages and Arguments
  - A lowercase word followed by a list of arguments and express and action
  - A message needs to be connected to a Receive



  - All objects can take this form
  - Numbers, functions, arrays and string can take this form



THE LANGUAGE

- Each rows ends with ;
- Part of the code that has to be executed as a block should be in the (…) environment
  - For (…) in the Post Windows the result of the last operation is shown

Ex. 0

- **Variables**
  - A single letter variable name do not need to be declared - they are global
  - Otherwise you should use var - they a valid only within an execution
  - a = 10; var value = 10;

- **Functions**
  - "A Function is an expression which defines operations to be performed when it is sent the 'value' message or another trigger massage".
  - Functions are enclosed in {…} environment
  - When you call the method 'value' on a Function it will evaluate and return the result of its last line of code

Ex. 1

UNITE GENERATORS (UGEN)

- There are many primitive building blocks, like types of tone generators or filters
- UGen are classes
- These are connected together in a processing graph to make more complicated synthesisers and sound processors.
- Most UGens have just one output, an audio stream or some sort of control signal

Ex. 2

- Many UGens handle common message:
  - ar: signal
  - kr: control signal

- Many UGens have common arguments:
  - mul: output will be multiplied by this value
  - add: this value will be added to the output.

SYNTH

Ex. 3

```
(
   SynthDef(\s1,
         { arg freq = 440, out = 0;
            Out.ar(out, SinOsc.ar(freq, 0, 0.2));
         }).add;
)

Synth(\s1);
```


- **SynthDef** defines a new Synth
  - .add add the definition of the synth (class) to the server
  - Synth(\s1) create a new instance (object) of the Synth called /s1

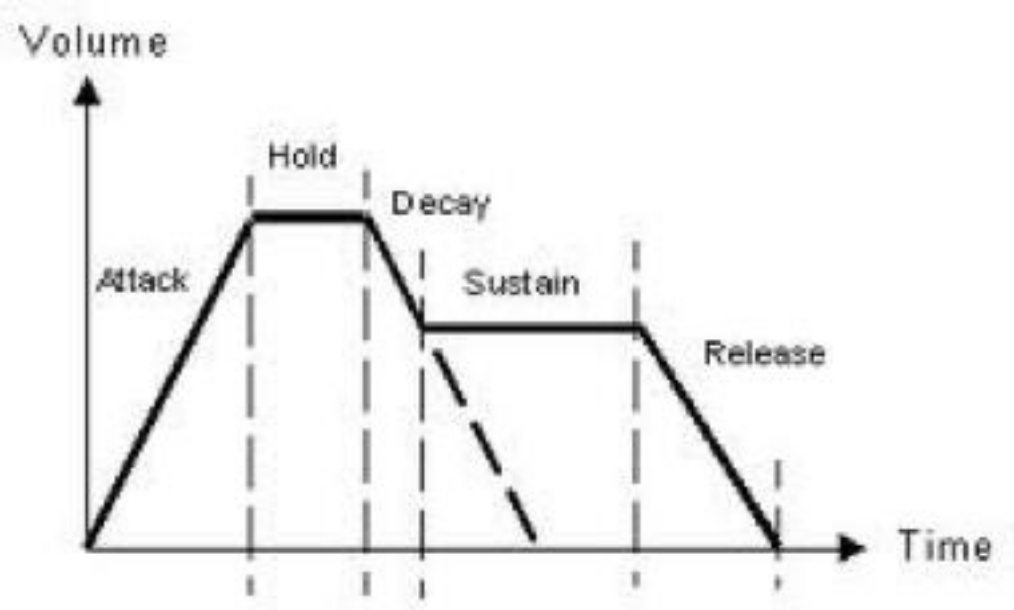
- the output can be multichannel

```
Out.ar([out1,out2], SinOsc.ar([freq1,freq2], 0, 0.2));
```

ENVELOPES

- Most of the time, we'll want to make sounds that go on for a limited time, and stop of their own accord.
- In general, we want total control over how parameters of a sound (like volume or frequency) vary over time.

Ex. 4



The volume envelope

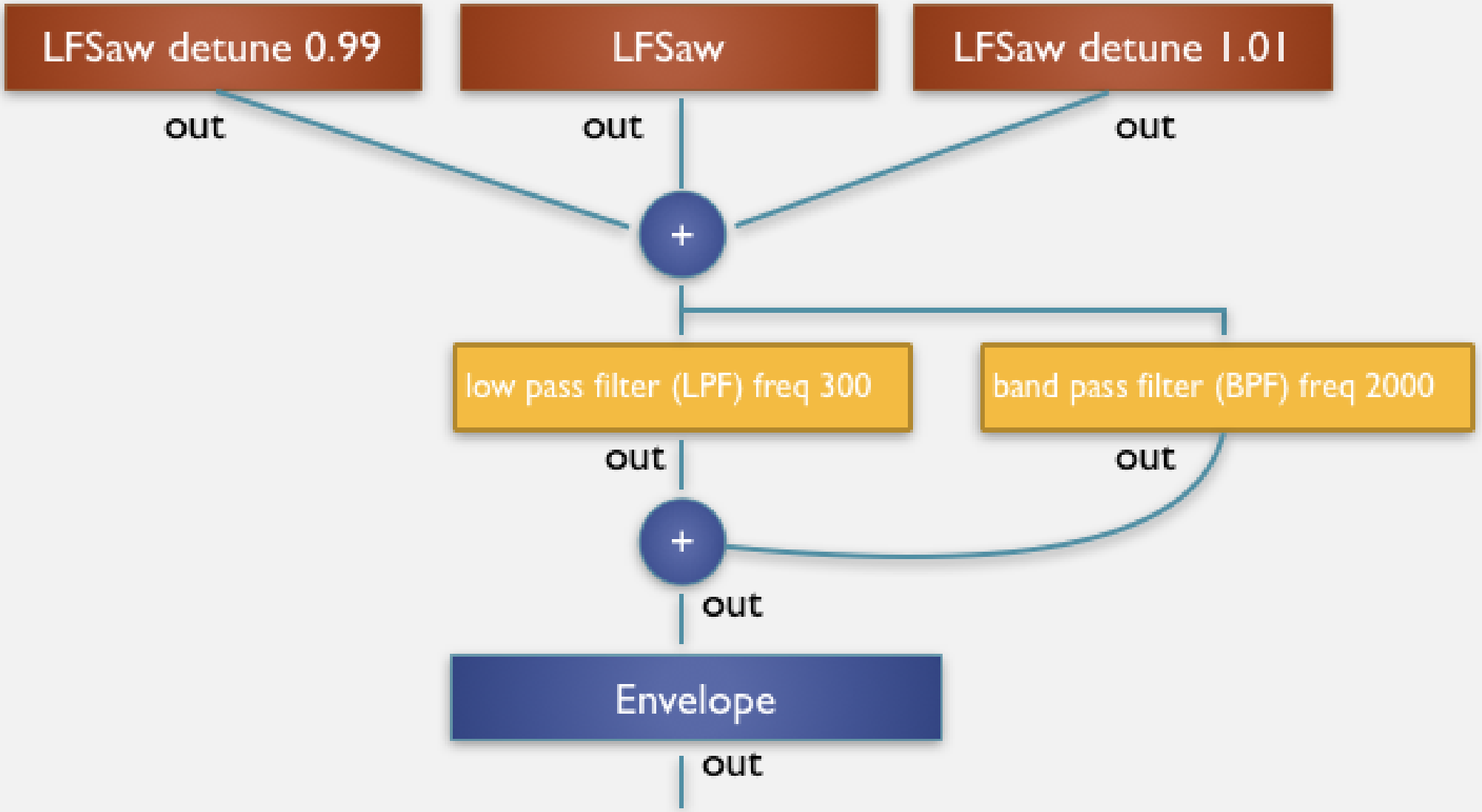
THE BELL

Ex. 5

- Given the model of the bell:
  - f = [0.5,1,1.19,1.56,2,2.51,2.66,3.01,4.1];
  - a = [0.25,1,0.8,0.5,0.9,0.4,0.3,0.6,0.1];
  - a1*sin(freq*f1)+a2*sin(freq*f2)+…

- 1) synthesise the sound bell
- 2) design the envelope in order to produce as most similar sound as possibile

WOBBLE BASS



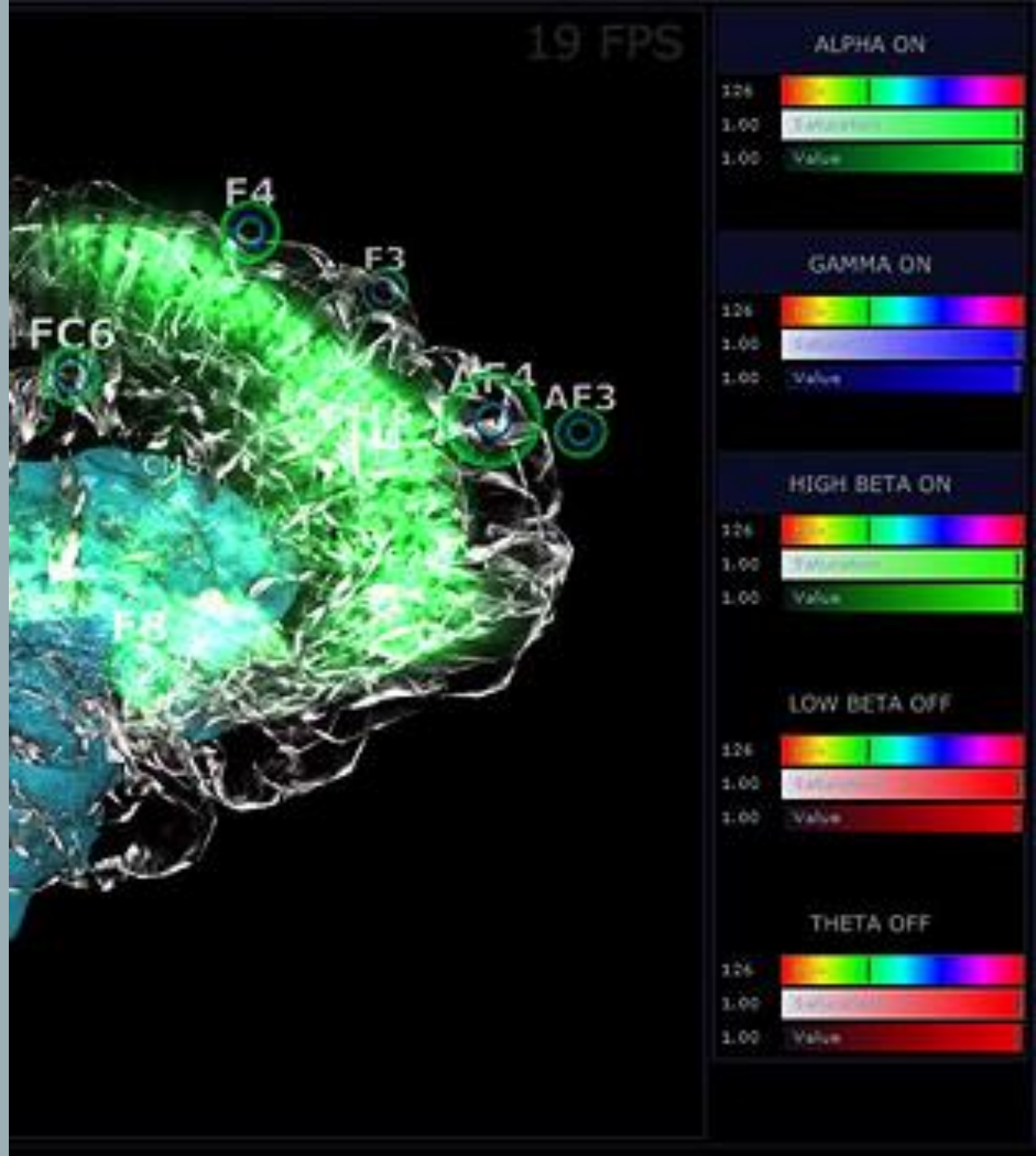
Ex. 6

MATERIALS

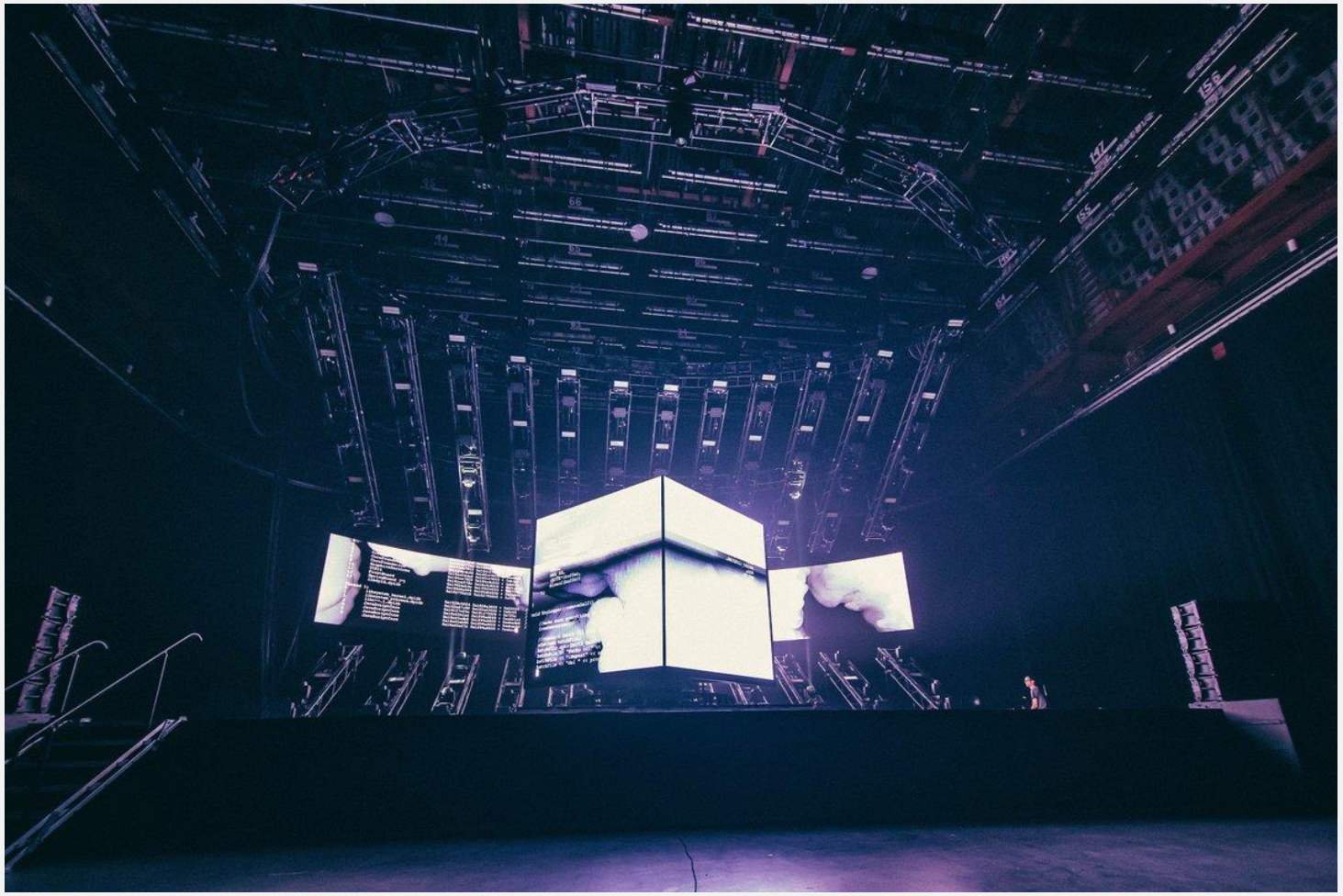
- **References**
  - Online tutorial by Scott Wilson and James Harkins (http://doc.sccode.org/Tutorials/Getting-Started/00-Getting-Started-With-SC.html)

- **Further readings**
  - Scott Wilson, David W. Cottle, Nick Collins, James McCartney, **The SuperCollider Book**, The MIT Press, 2011
  - Andy Farnell, **Designing Sound**, The MIT Press, 2010

TOUCHDESIGNER



WHAT YOU CAN DO WITH TD

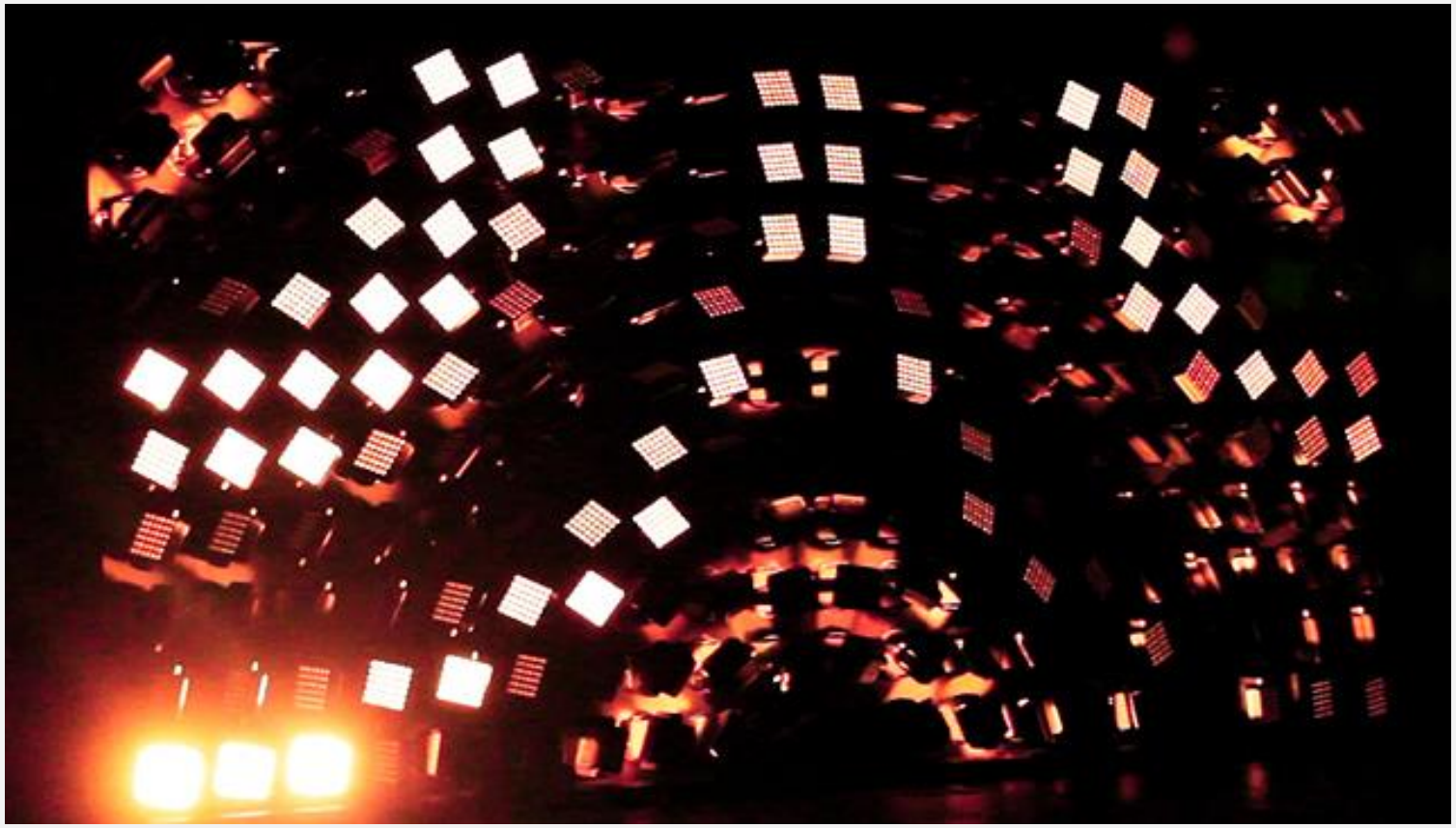


deadmau5 – Cube v3 - 2019 live

https://www.youtube.com/watch?v=1UcPqg211FY

https://www.youtube.com/watch?v=x5p2bI9lKks

WHAT YOU CAN DO WITH TD



Pixelux Studio's Kinetic Wall of Light System for Paris Couture Fashion Show - 2014

https://vimeo.com/107353028

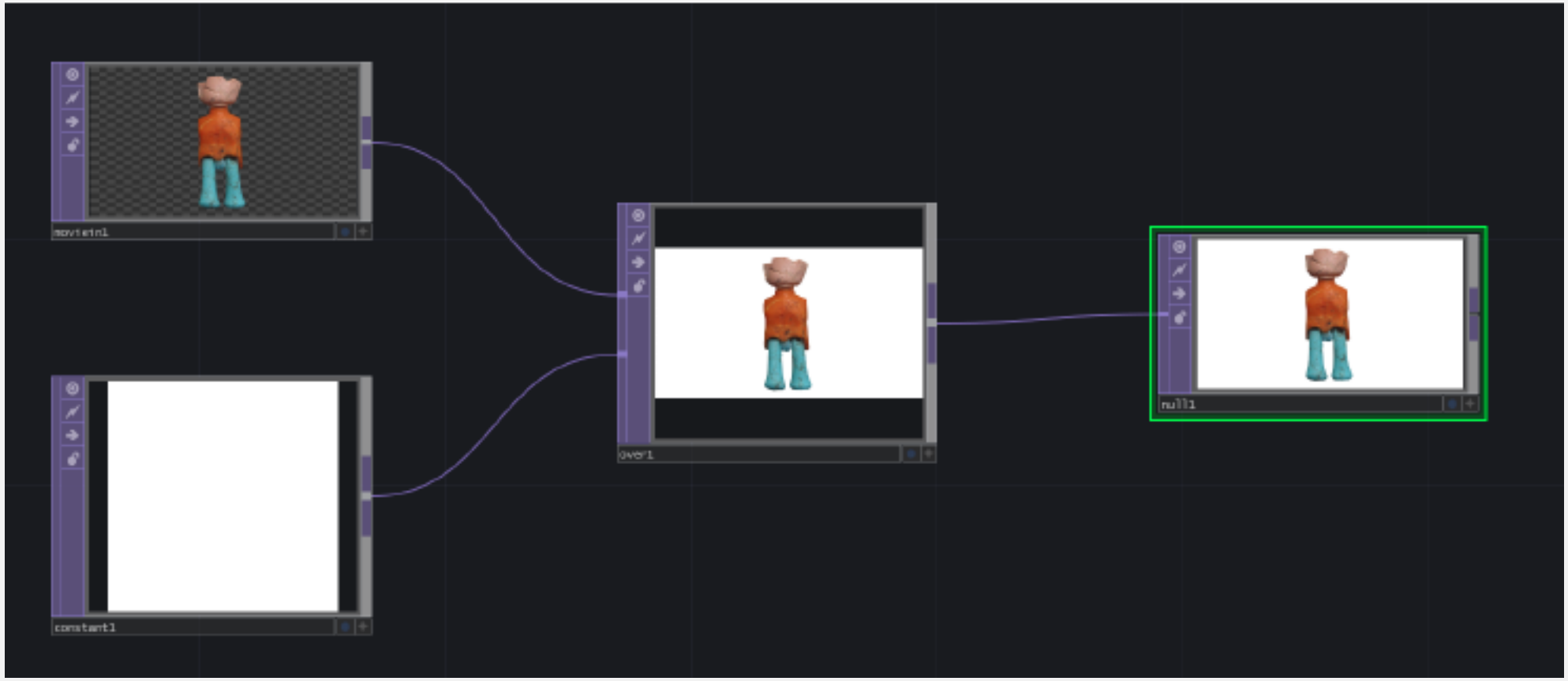
INTRODUCTION

- TouchDesigner is a visual, node-based programming language for:
  - Media management
  - Interaction Design
  - Computer Graphic
  - 3D mapping
  - Light and Video management
  - Digital Visual creation
  - Music and Sound management and elaboration
  - …

MAIN ELEMENTS

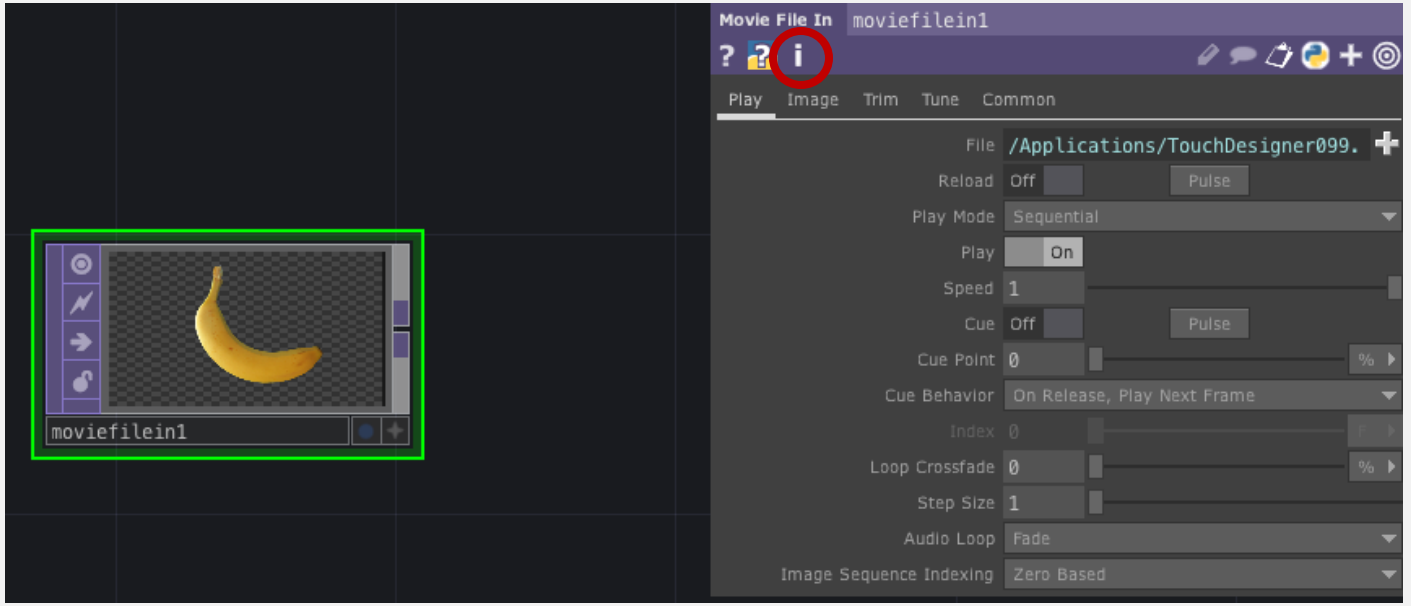
- Two main nodes: Operators and Components
- **Operators**
  - Each Operator has input (left) and outputs (right)
  - The inputs and outputs will also be ordered first to last, going from top to bottom
- **Components**
  - Components are capable of parent and child relationships, which flow from top to bottom.
  - component at the top of the signal chain is the parent, and the components below it are its children
- **Network**
  - Components and Operators are connected to compose a Network
  - In the network the flow goes from left to right

MAIN ELEMENTS



BASICS

- To create an Operator from scratch the two easiest methods are to hit 'Tab' on the keyboard, or to double click on the Network background
- Network. Right click on an Operator to reveal a menu with options that will be introduced slowly
- When an Operator is selected, "p" key opens the proprieties window
- Left-click on the 'i' of the proprieties window to get more detailed information about the selected operator.



BASICS

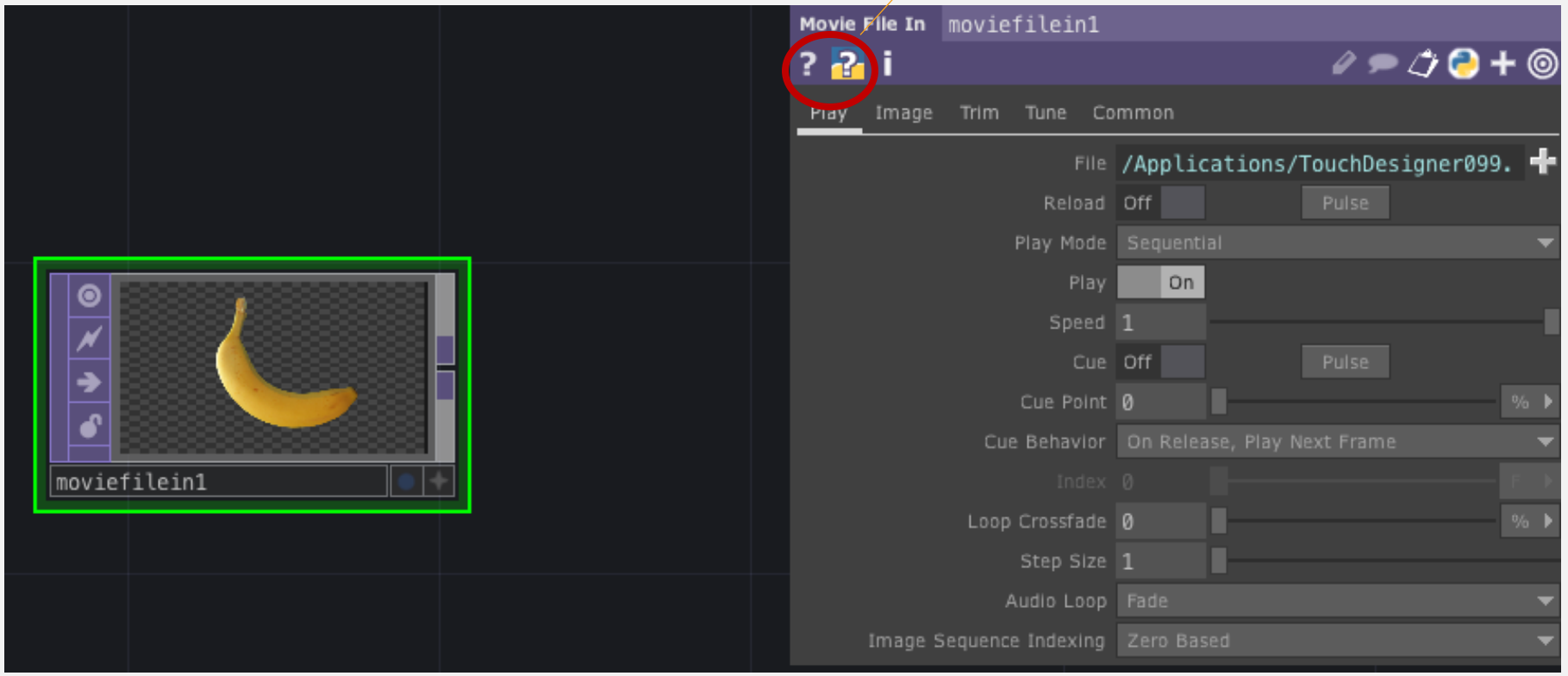
- All TouchDesigner projects are made of Networks. A Network is a group of Operators. Networks are encapsulated inside of components, such as a Container COMP, Base COMP, Geometry COMP, etc. Networks can be infinitely nested. The top level is called the 'root' level.

Path.toe

BASICS

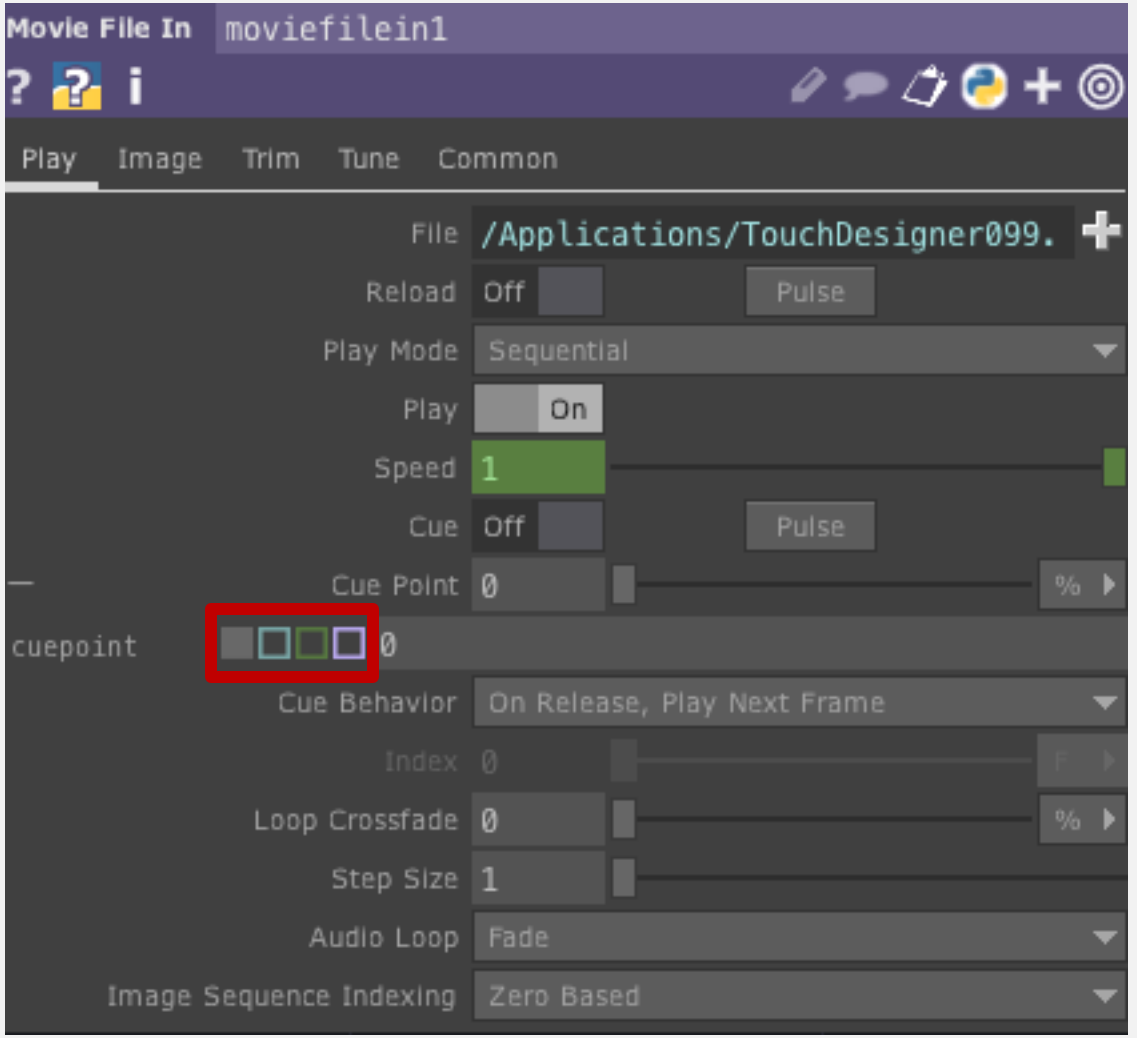
- TouchDesigner can highly interact with Python
- Python scripts can be used to modify operators
- Python scripts can be used to create operators

Allows to look at the python proprieties of the operator



USER INTERFACE

- Each parameter of an Operator has three modes:
  - Constant mode
  - Expression Mode
  - Export Mode
- **Constant mode** is the default for most parameters,
- **Expression mode** is used for Python, tscript, or mathematical operations and scripts.
- **Export mode** is used to directly reference CHOP channels
- Press the "+" at the left of a parameter



USER INTERFACE

- **Time line**
- The frame controls include ‘Start’ and ‘End’, which control the start frame and end frame of the timeline,
- ‘RStart’ and ‘REnd’, which control the loop start and loop end of the timeline
- FPS: Frames per second
- Tempo: Beat per Minutes and is related to music
- T Sig: Time signature and is related to music

PALETTE BROWSER

- The Palette Browser is the component library.
- The Palette Browser holds '.tox' files (or TouchDesigner Component files). These files contain a single Component Operator, that can hold a Network of other Operators.
- This means that a series of frequently used Operators, UI components, Python scripts, and more, can be created inside of a single Component Operator, saved as a '.tox' file
- In order to create personal components, select 'My Components' from the top portion of the browser. Then drag any component from the Network and drop it into the lower portion of the Palette Browser. It will then be added to 'My Components' repository.

OPERATORS

- **DATS**: Data Operators perform operations on data.
- They can edit, parse, create, send, and receive data in various forms: text strings, tables, Python scripts, XML, JSON, MIDI, Serial, OSC, etc.

- **SOPS:** The Surface Operators is a family of Operators are used for any and all 3D operations.
- This includes working with simple 3D geometry, particle systems architectural models, 3D characters, and more.

- **MATS:** Material Operators, or MATs, are used for materials and shaders for 3D geometry.

OPERATORS

- **TOPS**: Texture Operators are
  - 2D texture Operators that handle everything from movie playback
  - 3D geometry rendering, compositing,
  - hardware video inputs and outputs,
  - are used to process everything that will be output to a monitor, projector, or LED display.

- **CHOPS**: Channel Operators is a family of Operators that handle all channel operations including motion data, audio inputs, key-frame animations, hardware inputs (from Microsoft Kinect, Leap Motion, Oculus Rift, pen tablets, keyboards, mice, etc), DMX, MIDI, and OSC.

OPERATORS

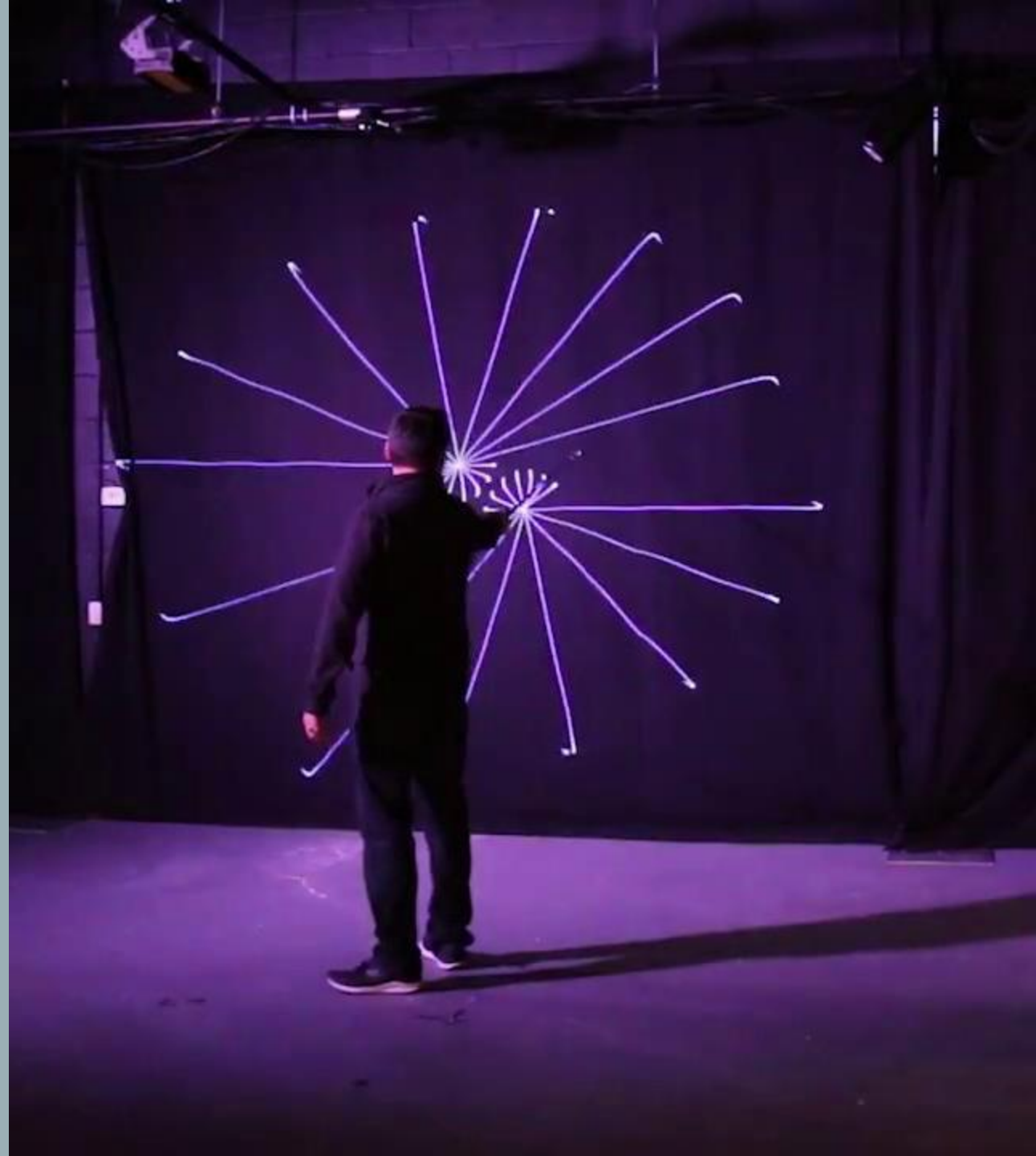
- **COMPS**: Contains four different types of components:
- **Object** components create, light, and view 3D scenes
- **Panel** components create UI components such as buttons, sliders, and window panes
- **Other** components include components that create keyframe animations, replicate other Operators, and create output windows. It includes the base component which is an empty container
- **Dynamics** components include physics modelling components

MATERIALS

- **References**
  - nVoid – Introduction to TouchDesigner (https://nvoid.github.io)

- **Further readings**
  - https://matthewragan.com/teaching-resources/touchdesigner/#course_python_intro
  - https://docs.derivative.ca/Category:Tutorials

OTHER TOOLS



TOOLS

- For Music
  - Cuck (https://chuck.cs.princeton.edu)
  - Soni Pi (http://sonic-pi.net)
  - Max MSP (https://cycling74.com)
  - Csound (http://www.csounds.com)
  - TidalCycles (https://tidalcycles.org/index.php/Welcome)
  - Pure Data (https://puredata.info)

TOOLS

- For Visual art and Interaction Design
  - Troikatronix Isadora (https://troikatronix.com)
  - OpenFrameworks (https://openframeworks.cc)
  - Cinder (https://libcinder.org)
  - MadMapper (https://madmapper.com)
  - Eyesweb (http://www.infomus.org/eyesweb_ita.php)
  - Fluxus (http://www.pawfal.org/fluxus/)
  - C4 (https://www.c4ios.com)
  - Polycode (http://polycode.org)
  - Vvvv (https://vvvv.org)
  - Tooll.io (http://tooll.io)
  - Cables (https://cables.gl)