Real-Time Sensing and Artificial Intelligence for Live Performative Arts

Live Performers Meeting, March 2024, M'Hamid El Ghizlane, Morocco.

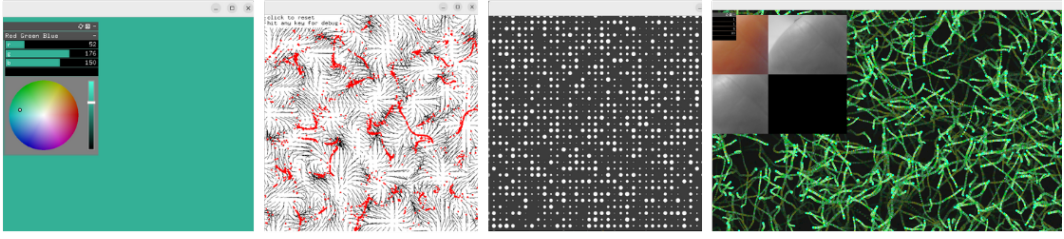
March 12, 2024

Federico Corradi - federico.corradi@gmail.com

FC
Federico Corradi

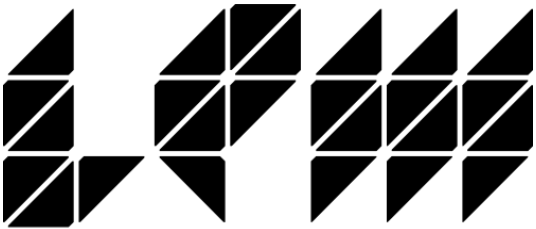
Real-Time Sensing and Artificial Intelligence for Live Performative Arts

Date	Lectures	Notes
Session 1	Creative coding, OpenFrameworks, the basic of programming, interactivity.	OFX docs, slides, code examples
Session 2	OpenFrameworks, creating add-ons in OFX, and sensors.	OFX docs, slides, OFX addons
Session 3	Neural networks basics, working with data, neural networks in OFX.	OFX docs, slides, code examples
Session 4	Using Neural Networks for data analysis, interpretation, and augmentation.	NN models, slides, OFX addons



https://openframeworks.cc/

Workshop logistics



Contents, objectives, and details:
https://avnode.net/learnings/real-time-sensors-and-artificial-intelligence-for-live-performative-arts/

Logistics:

- **Place: here!**
- **When**, 1.5h per session
- **Material:** slides, code (open-source).
- The coding examples, materials, add-ons, neural network models, will be made available on-line.

Tools: Laptops with Windows, Linux, or MacOSX

Outline

Get inspired: OFX creator (Zachary Lieberman)

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Zachary Lieberman is an American new media artist, designer, computer programmer, and educator.

Zachary Lieberman

Born	1977 (age 46–47)
Nationality	American
Education	Parsons School of Design, Hunter College
Known for	interactive art, programming, video, digital art
Website	http://thesystemis.com/

Early life and education [edit]

Born in 1977,[1] Lieberman holds a B.A. in Fine Arts from Hunter College and both a B.F.A. and M.F.A. in Design and Technology from Parsons School of Design.[2]

Work [edit]

Lieberman's work has appeared in numerous exhibitions around the world, including Ars Electronica, Futuresonic, CeBIT, and the Off Festival.

He collaborated with artist Golan Levin on the interactive audiovisual project "Messa Di Voce".[3]

With Theo Watson and Arturo Castro, he created openFrameworks, an open source C++ library for creative coding and graphics.[4]

Lieberman has held residencies at Ars Electronica Futurelab, Eyebeam,[5][6] Dance Theater Workshop, and the Hangar Center for the Arts in Barcelona. In 2013, he co-founded the School for Poetic Computation, a hybrid of a school, residency and research group in New York City.[7]

His work uses technology in a playful way to break down the fragile boundary between the visible and the invisible. His art work focuses around computer graphics, human-computer interaction, and computer vision.[*citation needed*]

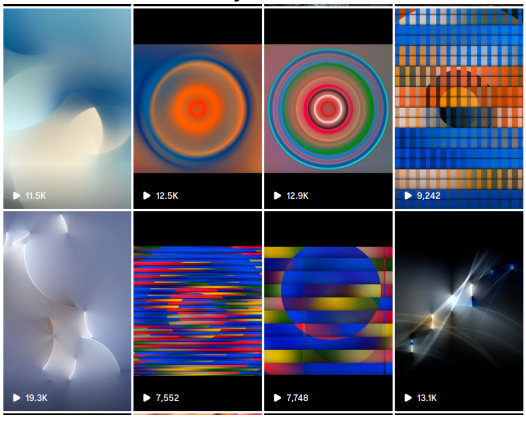
He teaches graphics programming classes at Parsons School of Design.[8]

Awards & distinctions [edit]

- 2018: Maryland Institute College of Art, William O. Steinmetz '50, Designer-in-Residence.[9][10]
- 2010: AOL, 25x25 artist grant
- 2010: Golden Nica in Interactive Art for the project Eyewriter, Prix Ars Electronica. Shared with James Powderly, Tony Quan, Evan Roth, Chris Sugrue (US) and Theo Watson (UK).[11]
- 2010: Number 36, of the "100 Most Creative People in Business", Fast Company[12]
- 2009: Artist's Grant, New York State Council on the Arts
- 2008: Honorary Mention at Ars Electronica in the Interactive Art category, for the OpenFrameworks project, shared with Theo Watson[13]

Zachary Lieberman - Wikipedia

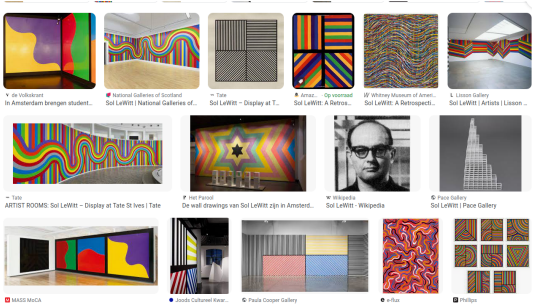
Zachary Lieberman



Zachary Lieberman - instagram

FC

Get inspired: art by instructions (Sol LeWitt)



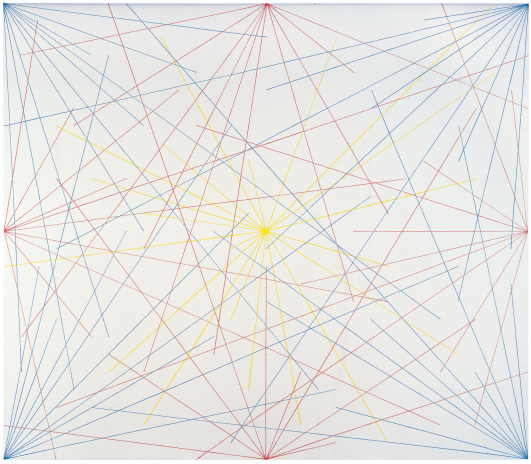
Sol LeWitt

Solomon "Sol" LeWitt (September 9, 1928 – April 8, 2007) was an American artist linked to various movements, including conceptual art and minimalism.

Conceptual art: *artist as programmer*

LeWitt helped revolutionize the definition of art in the 1960s by stating that the idea for an artwork is more important than its form. Each of his impermanent wall drawings consists of a set of the artist's instructions, with the actual execution carried out by someone else. LeWitt compared his instructions to musical scores, which are realized in a new way every time they're played. Similarly, it's possible for LeWitt's wall drawings to take slightly different forms, depending on how his directions are implemented.

Get inspired: art by instructions (Sol LeWitt)



Sol LeWitt, Wall Drawing 273, September 1975; The Doris and Donald Fisher Collection at the San Francisco Museum of Modern Art; ©The LeWitt Estate / Artists Rights Society (ARS), New York; photo: Ian Reeves

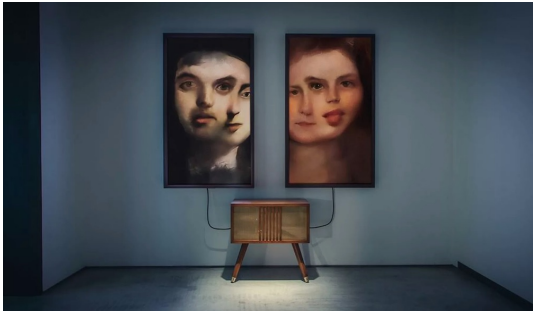
Conceptual art: *artist as programmer*

Sol LeWitt did many wall drawings during his career. Now that he is no longer alive, other people have to follow his instructions to *install* his works. Let's see if we can follow a set of Sol LeWitt's wall drawing instructions to create our own drawing.

- Take your white paper with the grid of 3" x 3" squares.
- Read the instructions for LeWitt's Wall Drawing below.
- Using red, yellow, and blue markers, pencils, or crayons, follow the steps for the Seventh wall.

Sol LeWitt's Wall Drawing 273, September 1975 Graphite and crayon on seven walls
A 6" (15 cm) grid covering the walls. Lines from corners, sides, and center of the walls to random points on the grid. 1st wall: Red lines from the midpoints of four sides;
2nd wall: Blue lines from the four corners;
3rd wall: Yellow lines from the center;
4th wall: Red lines from the midpoints of four sides, blue lines from four corners;
5th wall: Red lines from the midpoints of four sides, yellow lines from the center;
6th wall: Blue lines from four corners, yellow lines from the center;
7th wall: Red lines from the midpoints of four sides, blue lines from four corners, yellow lines from the center.
Each wall has an equal number of lines. (The number of lines and their length are determined by the draftsman.)

Get inspired: AI art movement (Mario Klingemann)



AI art movement

Klingemann's work **Memories of Passersby**. He generates portraits in real-time using neural networks. It is a computer system hidden inside of an antique-looking piece of furniture, which looks like a cross between a midcentury modern cabinet and an old-fashioned radio.

Mario Klingemann

Outline

This workshop approach: immersive and generative art

OpenFrameworks

- Data pre-processing
- Rendering, shaders
- Simulation engine (e.g., particles)
- If, then, loops

Sensors

- Cameras
- Radars
- Video outputs
- Midi in/out
- Wearable sensors

Neural Networks

- Style transfer
- Pose estimation
- Object detection
- Generative AI
- AR/VR (mixed reality)

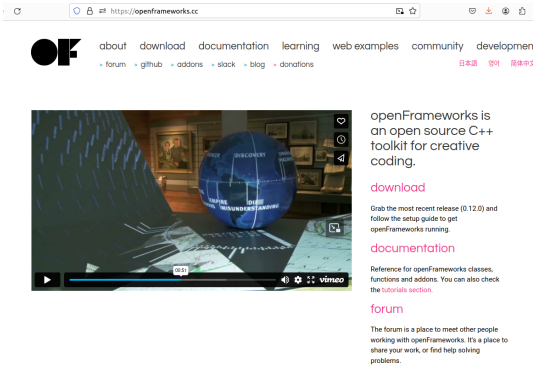
Immersive and generative art: the sky is the limit!

Generate designs in real-time, multiple outcomes tailored to the context, audience, rich in content, work in constant evolution, unique experiences.

Convivial: Immersive Art and Experiential Design
Probable universe - example

Outline

OpenFrameworks



openframeworks.cc
C++ Tutorials

OFX

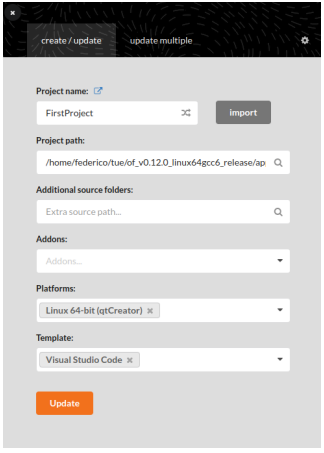
OpenFrameworks is an open-source C++ creative toolkit design for artists.

- C++ is fast
- C++ is efficient
- C++ is popular
- C++ is industry proven

Install OFX

Download from website, available for Linux, MacOSX, Windows. openframeworks.cc
Don't forget to download the Integrated Development Environment (IDE).

Let’s create our first project

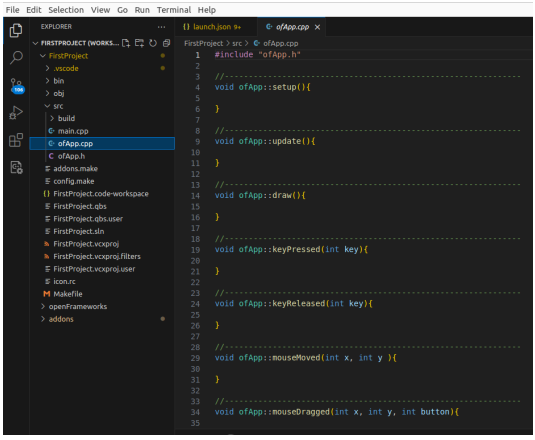


OFX project generation

- Open Project Generator
- Type a name, select a folder
- Platform might differ on MacOSX (XCode), Windows (VStudio), Linux (qtCreator)
- Linux users: click on the gear top-left advanced options, then template visual studio code appears.

*lldb if error with cppdbg in json

Project structure



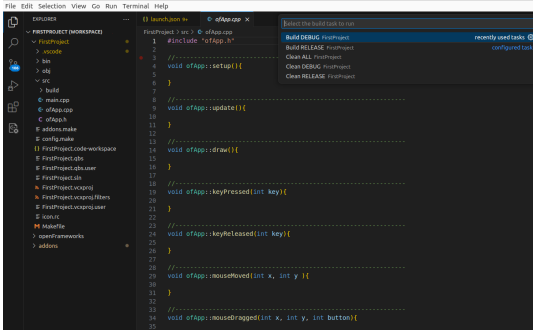
OFX project structure

- bin/
- bin/data/
- src/
- other project files

Visual Studio Interface

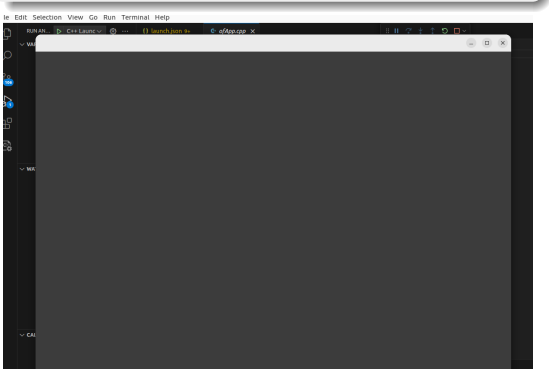
- Play (compile and run)
- Debug/Release
- Source codes
  - main.cpp
  - ofApp.cpp
  - ofApp.hpp

Project compilation

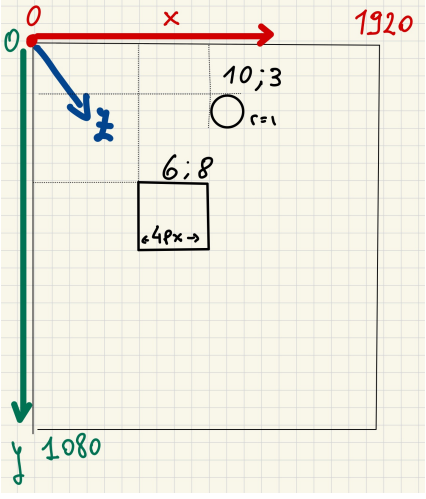


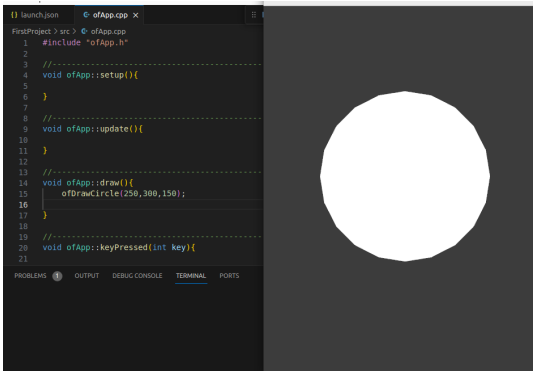
OFX project compilation

- Terminal -> Run Build Task
- Compile in Debug or Release
- Stand-alone application in **bin/**



Geometric shapes and colors

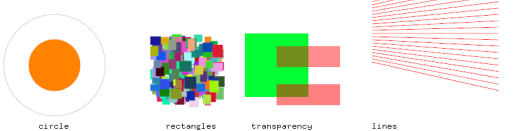




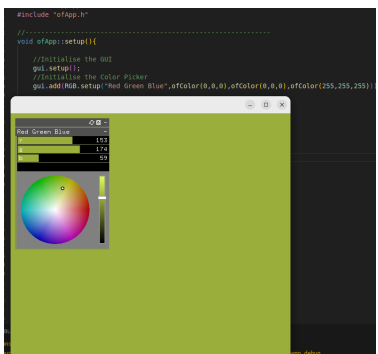
- Documentation https://openframeworks.cc/documentation/
- Many examples in **examples/graphics/**

Geometric shapes and colors

graphics example



- **examples/graphicsExample**
- ofSetColor (0-255)
- **myApp/rgbDemo**



The basic of programming: variables

Variables

Variables are containers for storing data values.

- int stores integer whole number
- float stores floating point number with decimals
- bool stores values with two states (true,false or 0,1)
- sting store text such as "Ciao". Strings are surrounded by double quotes.

Variable scopes

The scope of a variable is the "area" of the program in which the variable can be used. Variables can be local (exists within a function) or global (exists across functions).

- Global variables are defined in the ofApp.h and initialized in ofApp::setup()
- Local variables are defined within the function, where needed (e.g. ofApp::update)

The basic of programming: functions

Functions

A function is a block of code which runs only when it it called. You can pass data (parameters). Functions are used to perform certain actions and they are important for reusing code.

- Perform actions
- Get input data
- Can spit out data
- Use or create your own

OFX App functions

- ofApp::setup()
- ofApp::update()
- ofApp::draw()
- ofApp::keyPressed()
- ofApp::mouseMoved()
- ...

OFX Example

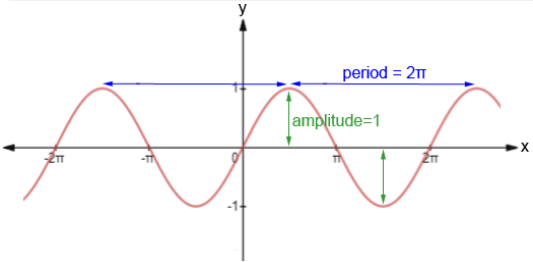
Let's look at this example together.

- **myApps/drawCross/**

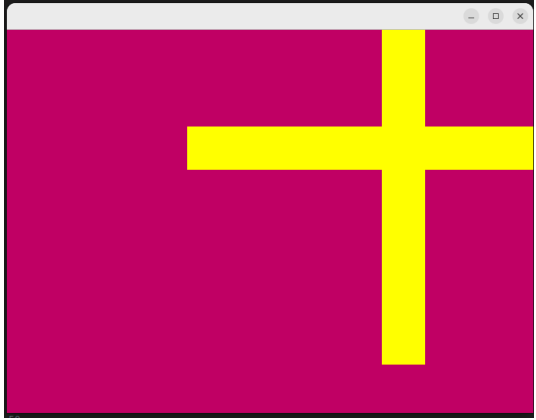
Some important functions

OFX functions

- ofBackground
- ofMap
- ofGetWidth
- sin(), ofRandom(), ofNoise()
- ofGetElapsedTimeMillis()




```
13  //--------------------------------------------------------------
14  void ofApp::draw(){
15      // some more interesting ofx functions
16      ofBackground(ofMap(ofGetMouseX(), 0, ofGetWidth(), 0,  255), 0 , 100);
17
18      drawCross(ofGetMouseX(), ofGetMouseY(), 600);
19  }
```



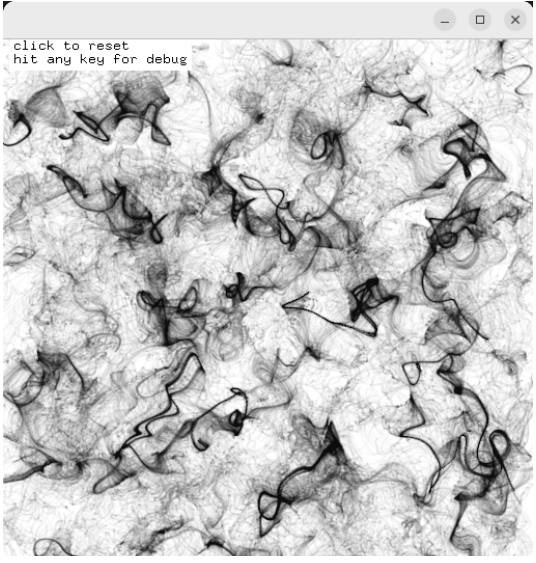
Some important functions

OFX functions

Advanced example

math/noiseField2dExample

- ofNormalize
- ofNoise

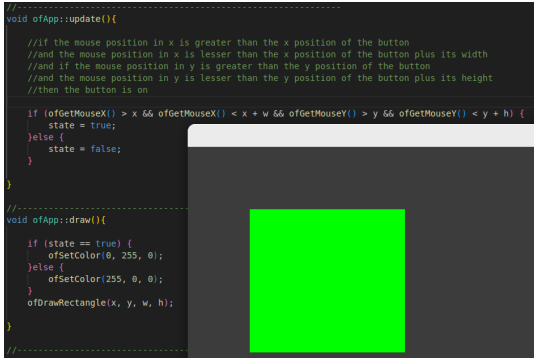


The basic of programming: conditions

Conditions

An action/function is performed if a condition is met (true or false). Similarly to mathematical formalism.

- Less than: $<$
- Less or equal than: $<=$
- Greater than: $>$
- Greater than or equal to: $>=$
- Equal to: **==**
- Not equal to: $!=$
- Logical and: &&
- Logical or: ||



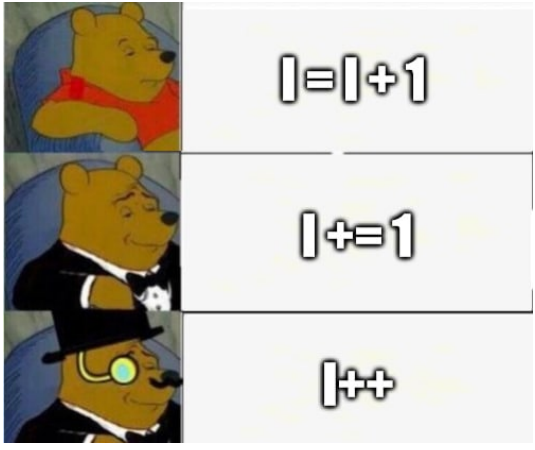
When is the rectangle green? What other color is possible, and under which condition?
myApps/ifCond/

Animate geometric shapes

Conditions

It is useful to increment or decrement a variable. The increment operator ++ adds 1 to its operand, and decrement operator – subtracts 1 from its operand.

- $x = x + 1;$ is the same as $x++;$ and as $x += 1;$
- $x = x - 1;$ is the same as $x--;$ and as $x -= 1;$



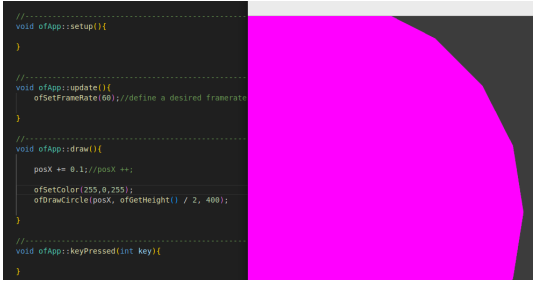
myApps/incrementExample/

Animate geometric shapes

Conditions

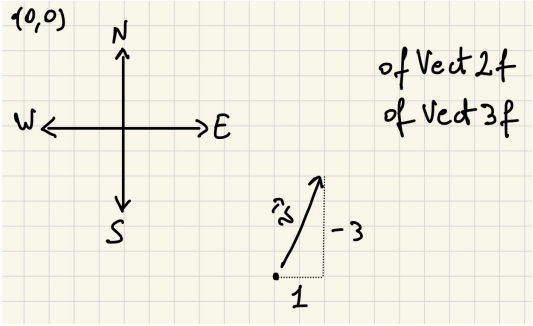
It is useful to increment or decrement a variable. The increment operator ++ adds 1 to its operand, and decrement operator – subtracts 1 from its operand.

- $x = x + 1$; is the same as $x++$; and as $x += 1$;
- $x = x - 1$; is the same as $x--$; and as $x -= 1$;



myApps/incrementExample/

Animate geometric shapes: directional vectors

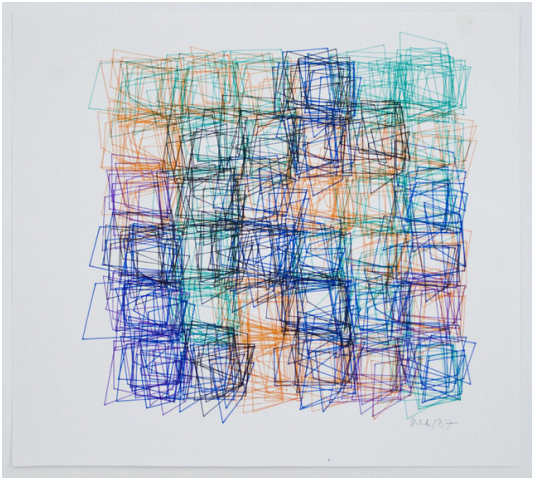



```
//--------------------------------------------------------------
void ofApp::setup(){
    //Random initial position
    pos.x = ofRandom(0, ofGetWidth());
    pos.y = ofRandom(0, ofGetHeight());
    //Random initial velocity
    vel.x = ofRandom(-30, 30);
    vel.y = ofRandom(-30, 30);
}

//--------------------------------------------------------------
void ofApp::update(){
    ofSetFrameRate(60);
    //we add velocity to the position for everyframe
    pos += vel;
    //we invert the velocity (direction) when the dot hit a wall
    if ((pos.x > ofGetWidth()) || (pos.x < 0)) {
        vel.x *= -1;
    }

    if ((pos.y > ofGetHeight()) || (pos.y < 0)) {
        vel.y *= -1;
    }
}

//--------------------------------------------------------------
void ofApp::draw(){
    ofDrawCircle(pos.x, pos.y, 40);
}
//--------------------------------------------------------------
```

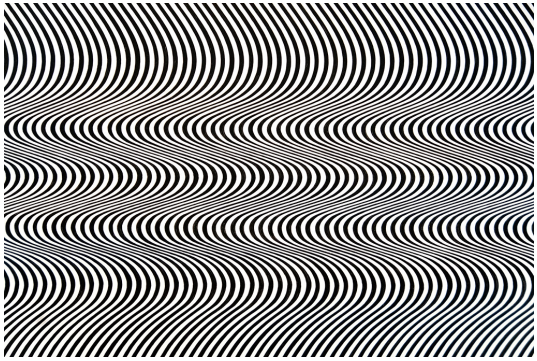
myApps/boucingBall/

https://natureofcode.com/f

Animate geometric shapes: the power of for loops



Vera Molnár



Bridget Riley - The Curve

Vera Molnar's Computer Paintings

"For information to exist, it must always be instantiated in a medium. Conceiving of information as a thing separate from the medium instantiating it is a prior imaginary act that constructs a holistic phenomenon as an information/matter duality"

"Molnar opened a space of mediation between the computational realm of information processing and the material practice of painting; in so doing, she directly challenged the duality between information and materiality that Hayles condemned"

"Information needs to undergo a certain amount of analogizing before humans can experience it, a task that today is routinely, and more or less invisibly, performed by interfaces"

Guillermet, A. *Representations* 2020.

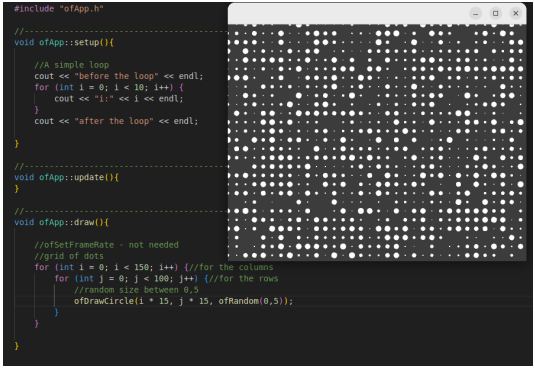
Animate geometric shapes: for loops

For Loops

When you want to repeat a block of code.

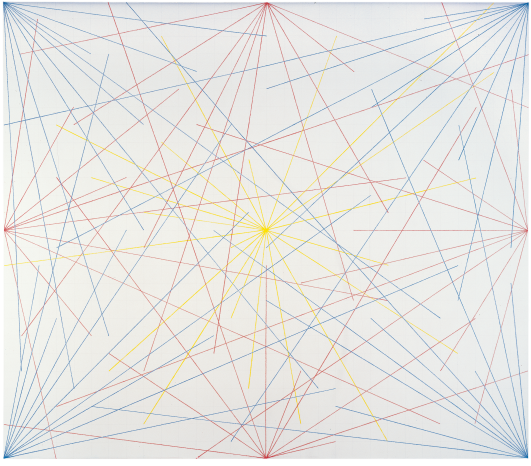
```
for (st1; st2; st3){
    ... my code
}
```

- st1 is executed one time, before the execution of the code block.
- st2 defines the condition for executing the code block.
- st3 is executed every time, after the code block is executed.

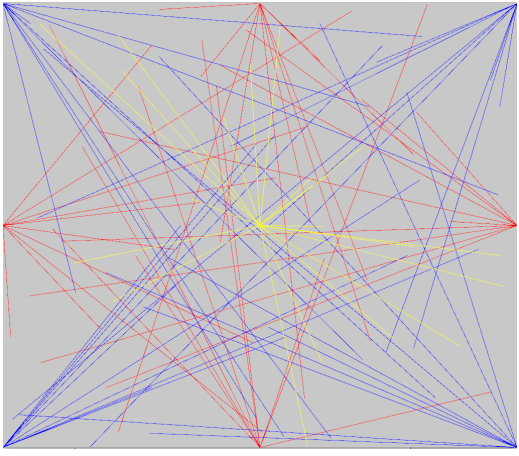


myApps/forLoops/

Animate geometric shapes: for loops



Sol LeWitt



myApps/solLeWitt

Classes and arrays

Class

Everything in C++ is associated with classes, along with its attributes and methods (also known as functions and variable members).

- Declare attributes and methods in the class.
- To create a class use class keyword.
- public or private specifiers tells if the member of the class are accessible from outside the class.
- The constructor is a special method that is automatically used when an object of a class is created.

```
class MyCar {
  private:
    int speed;  // Car's speed

  public:
    // Constructor
    MyCar(int initialspeed=0);
    // Increase the speed
    void accelerate(int more);
    // Apply brakes
    void brake(int less);
    // Change the speed
    void changeSpeed(int newSpeed
    );
    // Get the current speed
    int getSpeed();
```

Classes and arrays

Object

An object is created from a class.

- To create an object, specify the class name, followed by the name of the object.
- Access the class attributes and methods with **.**

```
MyCar car(50);  // Create a MyCar
    object
cout << "Speed: " << car.getSpeed
   () << " km/h" << endl;
car.accelerate(100); //use class
   function
```

Classes and arrays

Class methods

Methods are functions of the class.

- Methods can be defined inside or outside the class.
- OFX defines those outside the class. Usually a class is defined in ofApp.h and its methods in ofApp.cpp.
- To define methods outside the class, we use the class name ofApp followed by the *scope resolution* :: operator, followed by the name of the function.
- We must include ofApp.h in the class file ofApp.cpp using #include "ofApp.h"

```
\\ofApp.h
class ofApp : public ofBaseApp{
    public:
        int myFunction(int num);
}
```

```
\\ofApp.cpp
#include "ofApp.h"
int ofApp::myFunction(num){
    //some function code
}
```

Classes and arrays

Class methods

Methods are functions of the class.

- Methods can be defined inside or outside the class.
- OFX defines those outside the class. Usually a class is defined in ofApp.h and its methods in ofApp.cpp.
- To define methods outside the class, we use the class name ofApp followed by the *scope resolution* :: operator, followed by the name of the function.
- We must include ofApp.h in the class file ofApp.cpp using #include "ofApp.h"

```
\\ofApp.h
class ofApp : public ofBaseApp{
    public:
        int myFunction(int num);
}
```

```
\\ofApp.cpp
#include "ofApp.h"
int ofApp::myFunction(num){
    //some function code
}
```

by why? to organize! and use many objects with one class.

Arrays I

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring multiple variable for each value.

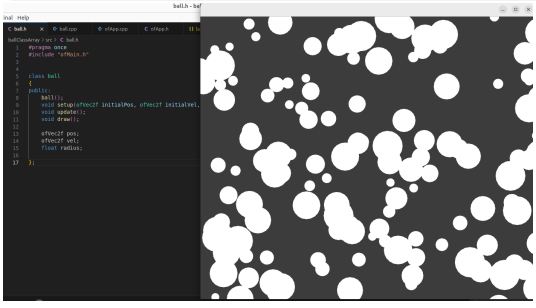
- To declare arrays, define the type of variable, the name of the array, followed by a square bracket and define how many elements it should store.
- You access an array element by referring to the index number inside the square bracket
- In $C++$, we count from 0 (i.e. zero is the first element).

```
int myNum[10]; // init array of
    10 int
myNum[0] = 1;
myNum[9] = myNum[0]+1;
cout << myNum[9] << endl;
```

Arrays II

Arrays

Use for loops to access all elements of your array.



```
car myBall[100]; // 100 balls
!
for(int i=0; i<100;i++){
    myBall[i].start();
}
```

myApps/ballClassArray/

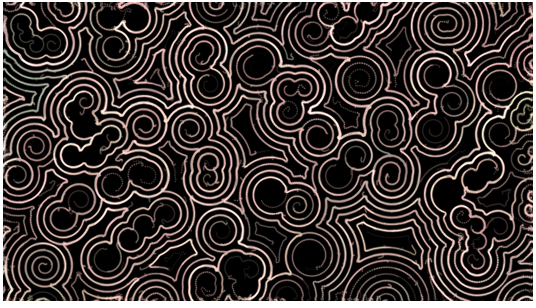
This is our first particle system.

Particle system

Particle System

A particle system is a computer graphic techniques that uses graphic objects to simulate certain kinds of natural phenomena.

- Initialize particles.
- Simulation stage; update parameters and simulates how particle evolve.
- Rendering stage.



addons/ofxReactionDiffusion

Vectors in C++

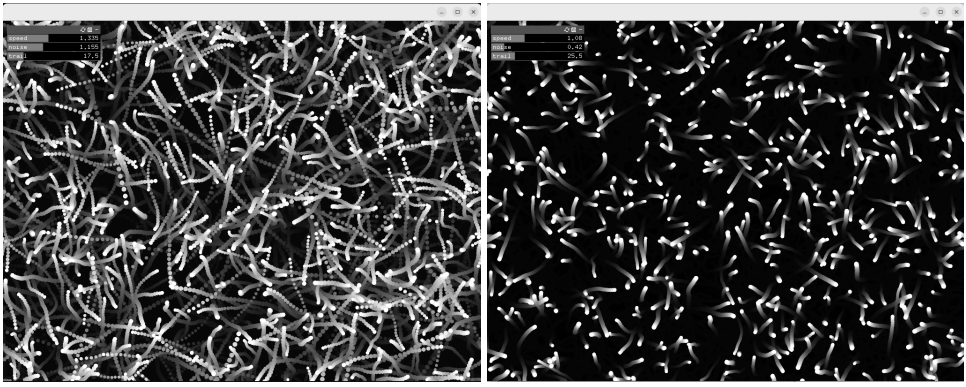
Vectors

Resizeable arrays.

- Not confuse with euclidian vectors.
- They are used like arrays, and access using [].
- Declared them using vector<float>myFloat
- .assign() assign vector content.
- .push_back() add an element at the end.
- .size() return size.
- .clear() clear content.

```
// Declare a vector of float
std::vector<float> myFloat;
// Assign values to the vector
myFloat.assign({1.1, 2.2, 3.3});
// Access elements using []
cout << "Element 1: " << myFloat
    [1] << endl;
// Add an element at the end
myFloat.push_back(4.4f);
// Get the size
int size = myFloat.size();
// Clear the vector
myFloat.clear();
// Check size
cout << "Size now: " << myFloat.
    size() << endl;
```

Particle system: putting all together



myApp/particleSystem

Outline

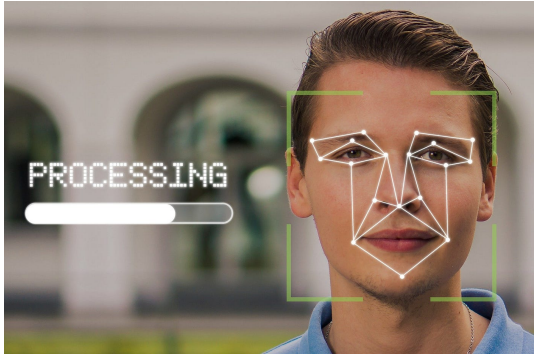
Computer vision and interactivity

Computer Vision

Computer vision is the field of study that enables computers to interpret and process visual data from the real world, akin to human vision.

- OFX uses OpenCV.
- OpenCV is very powerful!
- OpenCV is open-source.

https://opencv.org/



Background subtractions

Step 1

Capture an image of the empty scene

Step 2

Live video from the cam

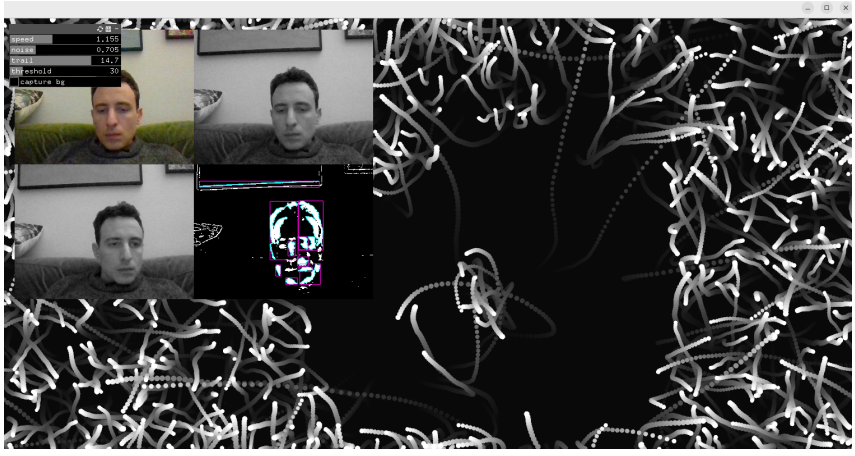
Step 3

We compare the difference between the captured image and the image from the live cam. This is the ***background subtraction*** technique.

Step 4

From the subtracted silhouette we define contours as vectorial ***blobs***

Interactive particle system



myApps/computerVisionInteractive

That’s all for this session

Summary

- OpenFrameworks (project, folder, compilation, IDE setup)
- The basic of programming:
  - Shapes and colors, variables, functions, conditions, animate shapes, loops, arrays, vectors, c++ classes
- Particle system
- Computer vision and interactivity

Next Time

We will learn to create add-ons in OFX and look at sensor data for more interactivity, and how to handle data from Cameras, Radars, Kinect 360, and other sensors.