programming for artists & designers

Daniel Berio

Session Summary

- Module Overview
  - Module Purpose
  - Programming, art, design?
  - Classroom Culture
  - Module structure + how you are assessed
- Introduce ourselves
  - Procedural drawing group activity
- Break & Troubleshooting openFrameworks (20 mins)
- Lab Session
  - Intro to C++/openFrameworks
  - Intro to 2D Graphics
  - Lab Activities & more troubleshooting!

About me: Daniel Berio

Computational artist, researcher

Background as a graffiti artist

PhD in Goldsmiths, computer models of graffiti

Dealing with computer graphics research:

- Mostly vector graphics
- Fonts, calligraphy, graffiti
- Some robotics (drawing/painting robots)

www.enist.org
@colormotor
daniel.berio@gold.ac.uk

module purpose

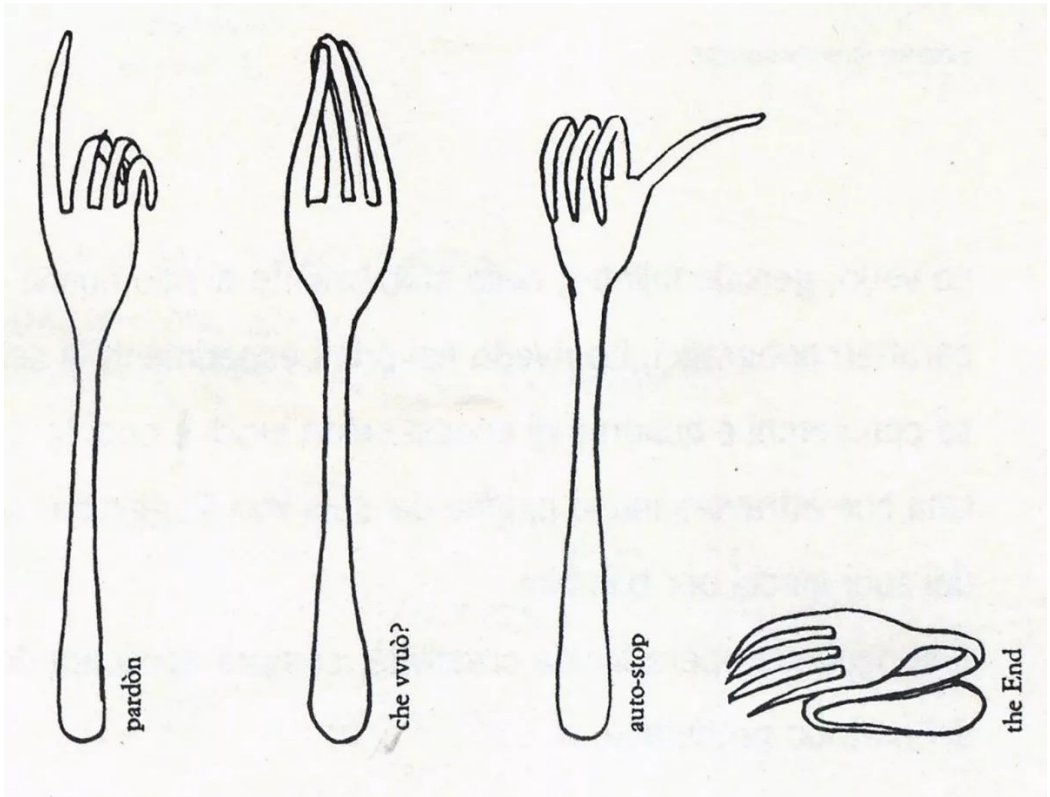
- provide a fundamental understanding of code and modern computer literacy
- introduce you to a range of techniques for making creative software and computer graphics in C++
- develop your artistic or creative practice & develop your visual communication skills

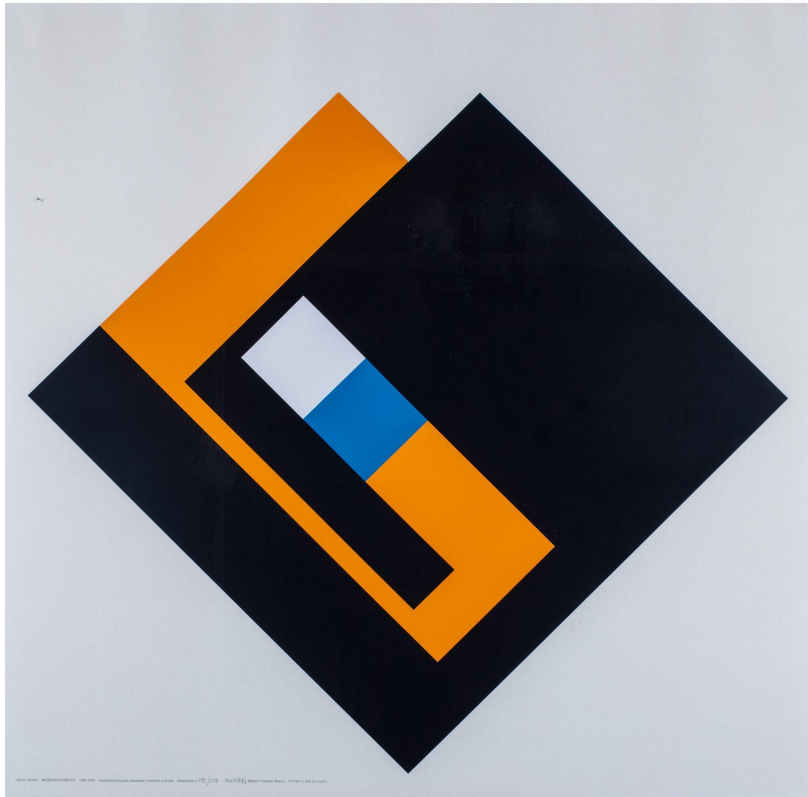
OpenFrameworks

code as creative medium

In a news article, a famous art critic asked this question regarding "programmed art": will we have art made by machines? This sentence only denotes ignorance regarding the problem, because it is like asking: will we have art made by brushes? or by pencils?

Bruno Munari, Design e Comunicazione Visiva, 1968





Code as a “meta-medium"

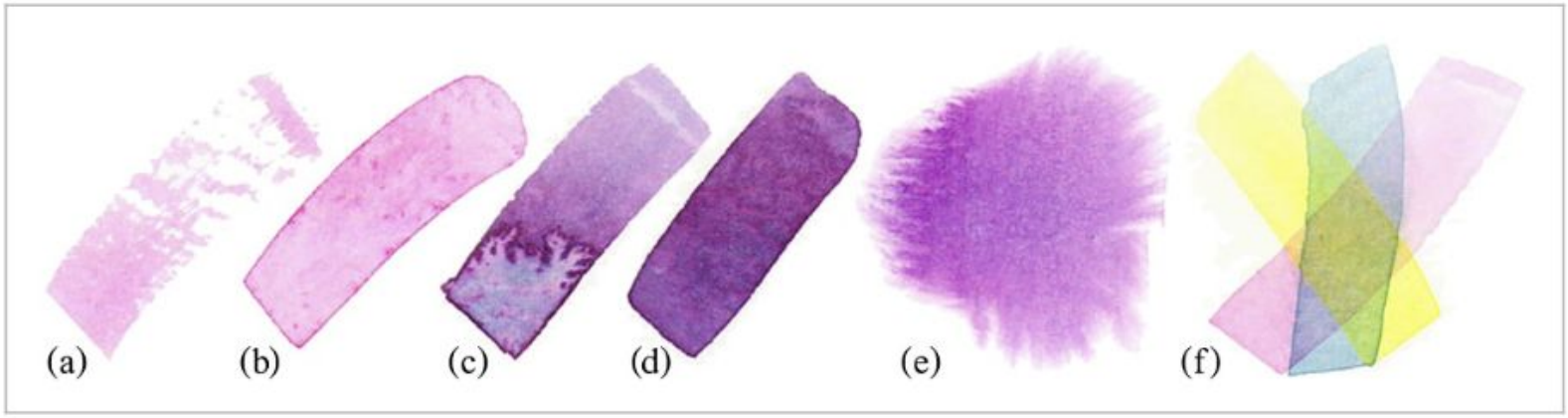
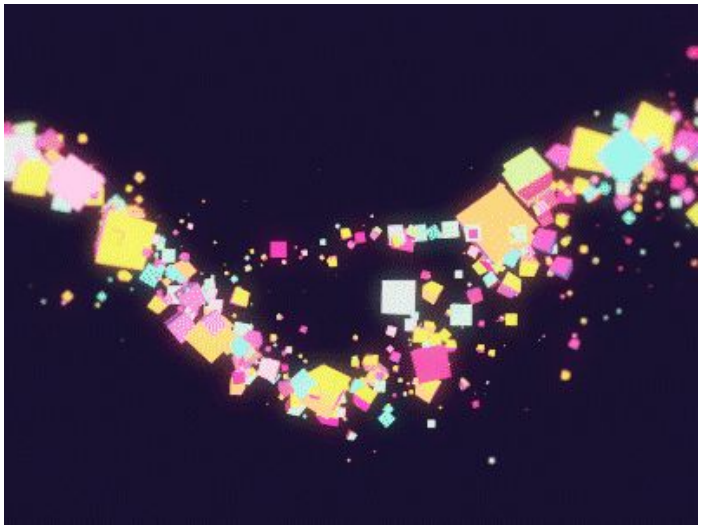


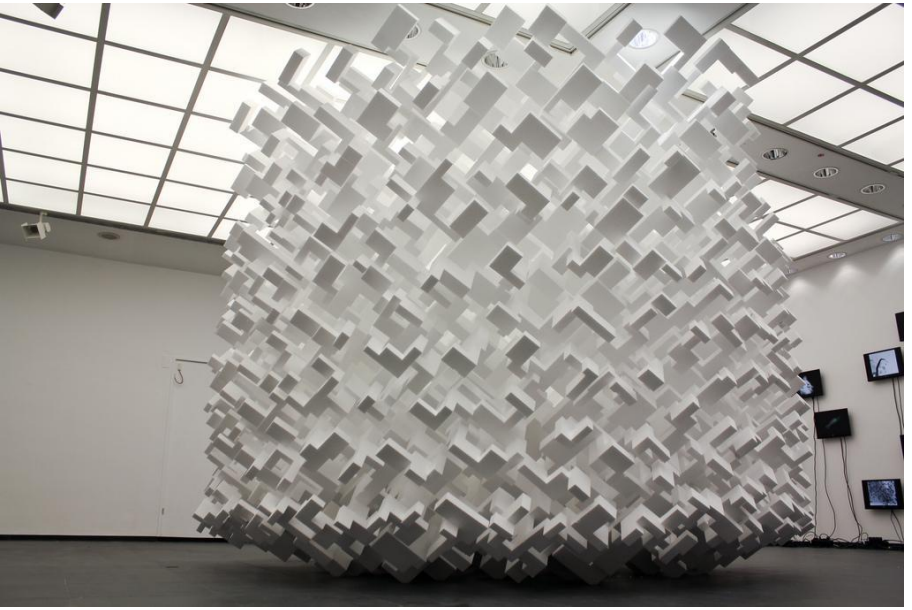
Figure 4 Open in figure viewer | PowerPoint

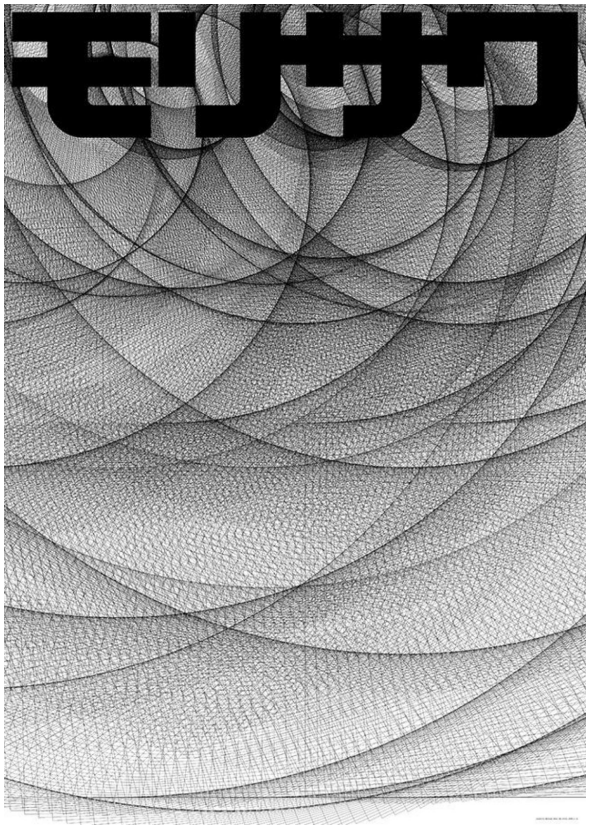
Different watercolour effects: (a) dry brush, (b) edge darkening, (c) backruns, (d) granulation, (e) flow effects and (f) glazing 17.

https://onlinelibrary.wiley.com/doi/full/10.1002/cav.1435



Code as a “meta-medium"





http://www.emohr.com/

Working with a computer allows
“High speed visual thinking.”

manfred mohr

une esthétique programmée

démonstrations sur matériel électronique · Benson traceur de courbes

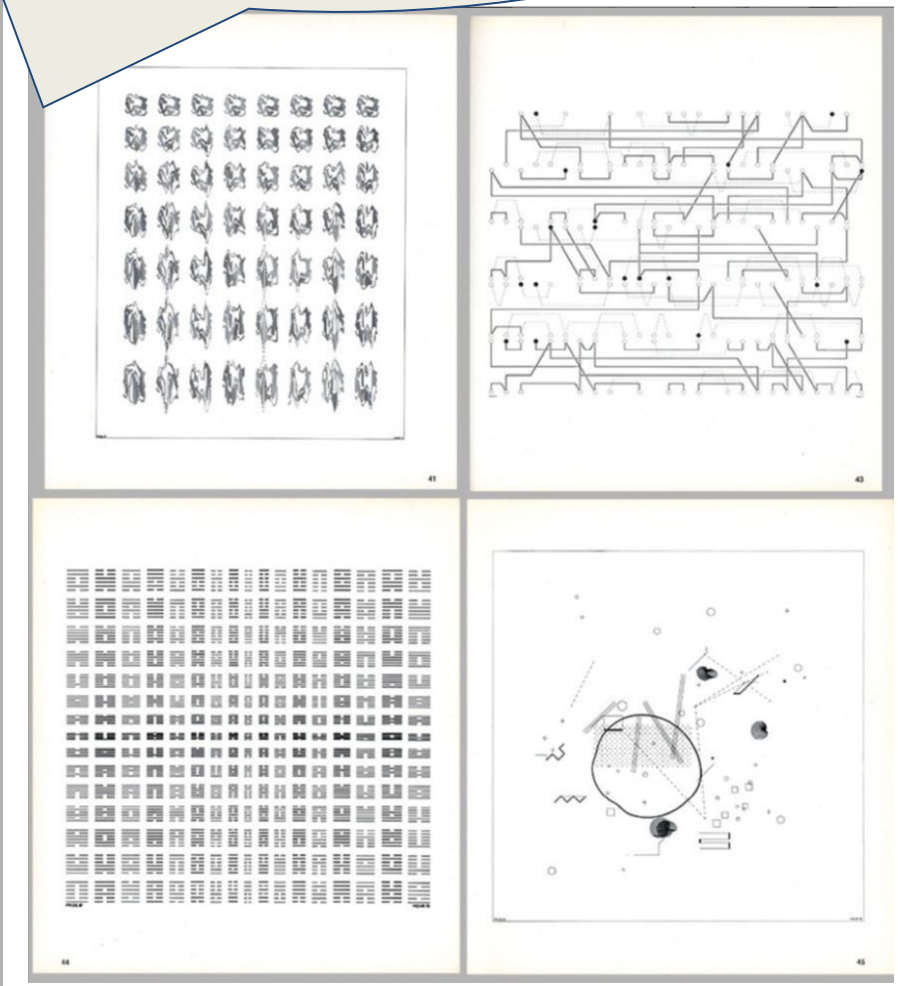
la section animation recherche confrontation du musée d'art moderne de la ville de Paris vous invite au vernissage de l'exposition de manfred mohr « une esthétique programmée » le mardi 11 mai 1971 de 18 heures à 22 heures, jusqu'au 6 juin au musée d'art moderne de la ville de Paris, 11, avenue du Président-Wilson Paris 16e, valable pour la durée de l'exposition

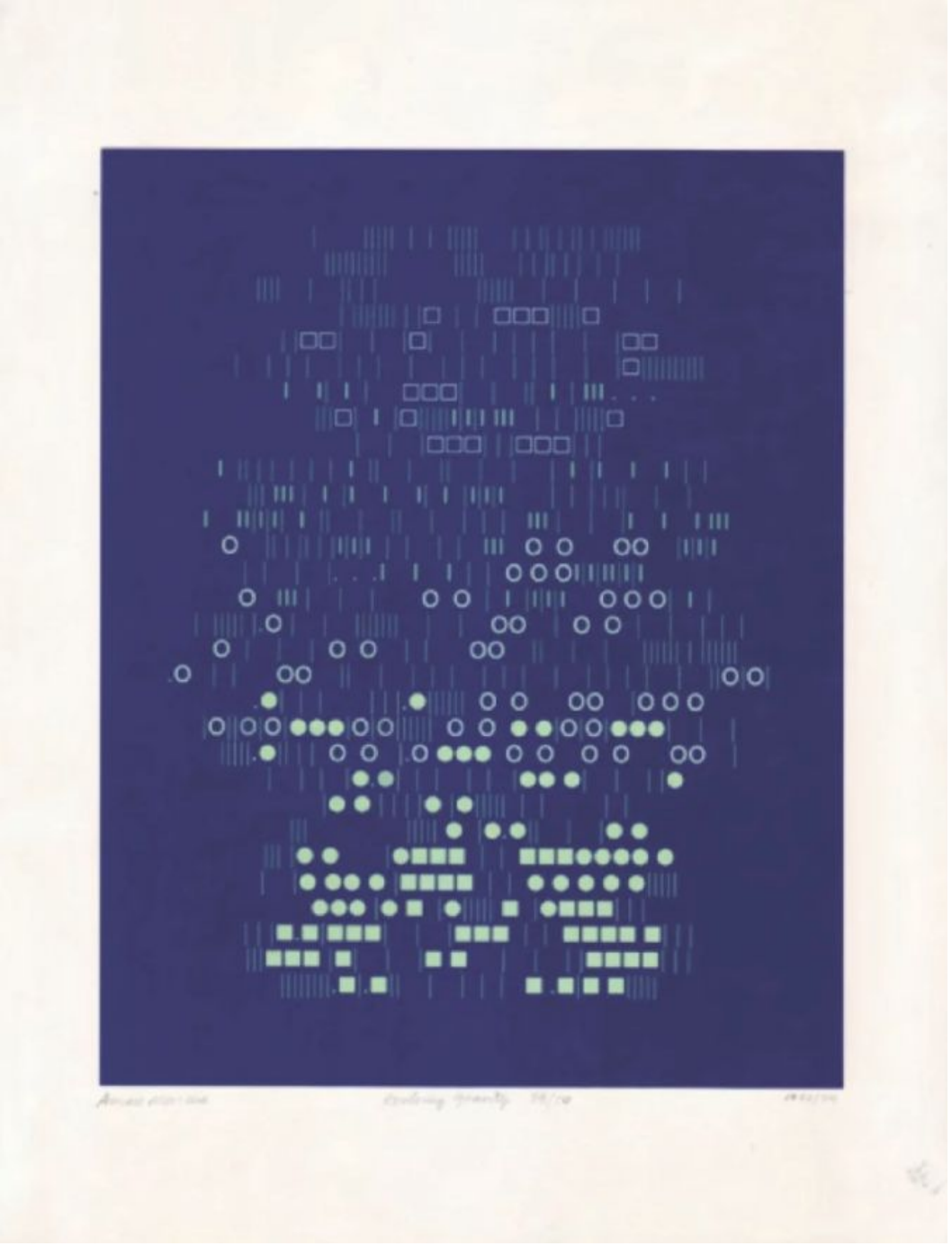
50 years ago today

Photos at opening & invitation of historic show,
1st one-person show of digital art in a museum:

Manfred Mohr - Computer Graphics,
Une Esthétique Programmée
Curated by Pierre Gaudibert
ARC - Musée d'Art Moderne de la Ville de Paris
Paris, MAM/ARC, 11. May - 6. June 1971

Show consisted of a long wall with 20+ pen plotter drawings, Benson flat bed plotter, large guest book panel on wall & catalog of show with 3 contributed texts: http://emohr.com/paris-1971/





Aaron Marcus. *Evolving gravity* 1972/74.

"Creative coders" are artists, designers, architects, musicians, and poets who use computer programming and custom software as their chosen media.

Code as a creative medium, Golan Levin and Tega Brain

Programming for Artists & Designers

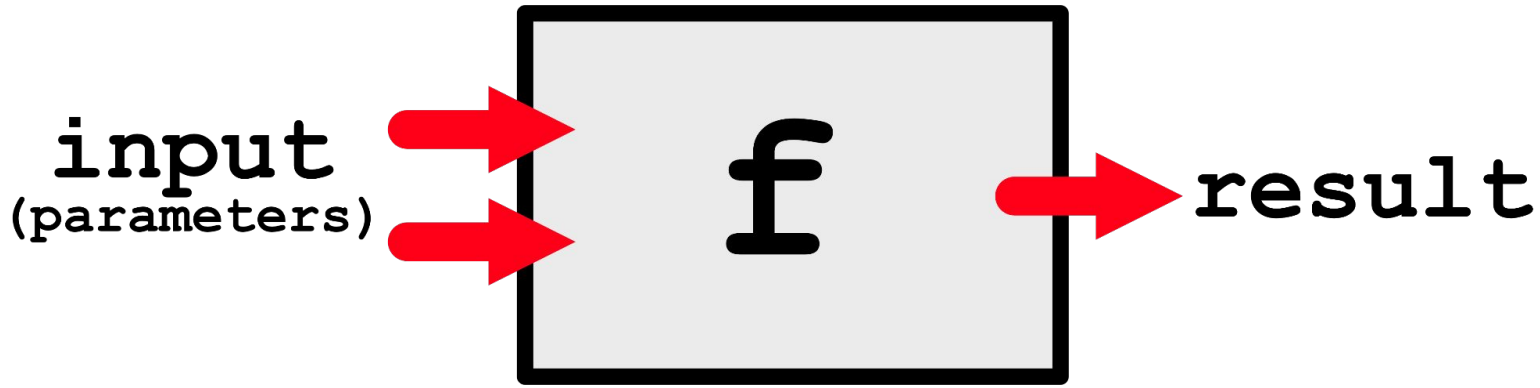
Computational Art

or

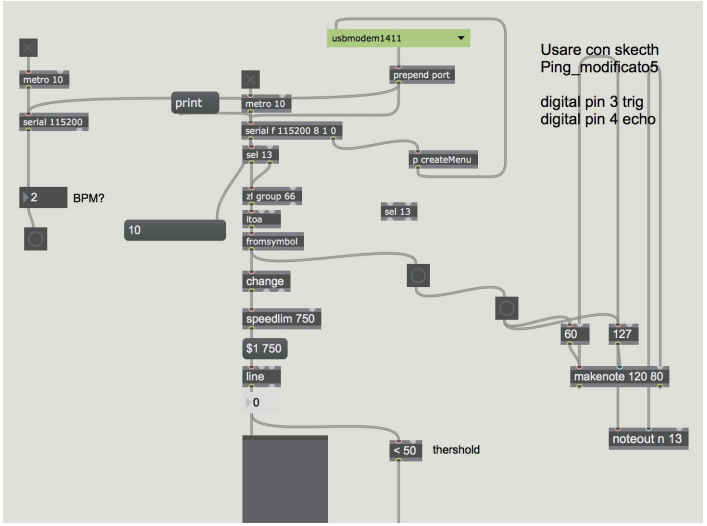
Creative Coding

What is programming or coding?

- Writing instructions/procedures that a computer can follow and execute
- These instructions are usually expressed through **functions**
- You can think of a function as a box that takes in input parameters, performs some actions and gives back a result



- In some (visual) programming languages (e.g Max) that you will see during the course, this is quite literal!



- But here we will define/use functions as text, e.g.

```
result = myFunction(parameter1, parameter2, parameter3);
```

  - Has its advantages and its disadvantages

```
float ofMap(float value, float inputMin, float inputMax, float outputMin, float outputMax, bool clamp) {

  if (fabs(inputMin - inputMax) < std::numeric_limits<float>::epsilon()){
    return outputMin;
  } else {
    float outVal = ((value - inputMin) / (inputMax - inputMin) * (outputMax - outputMin) + outputMin);

    if( clamp ){
      if(outputMax < outputMin){
        if( outVal < outputMax )outVal = outputMax;
        else if( outVal > outputMin )outVal = outputMin;
      }else{
        if( outVal > outputMax )outVal = outputMax;
        else if( outVal < outputMin )outVal = outputMin;
      }
    }
    return outVal;
  }

}
```

- Coding is like swimming, you need to **do it** to learn, can’t just read

PREPARATION

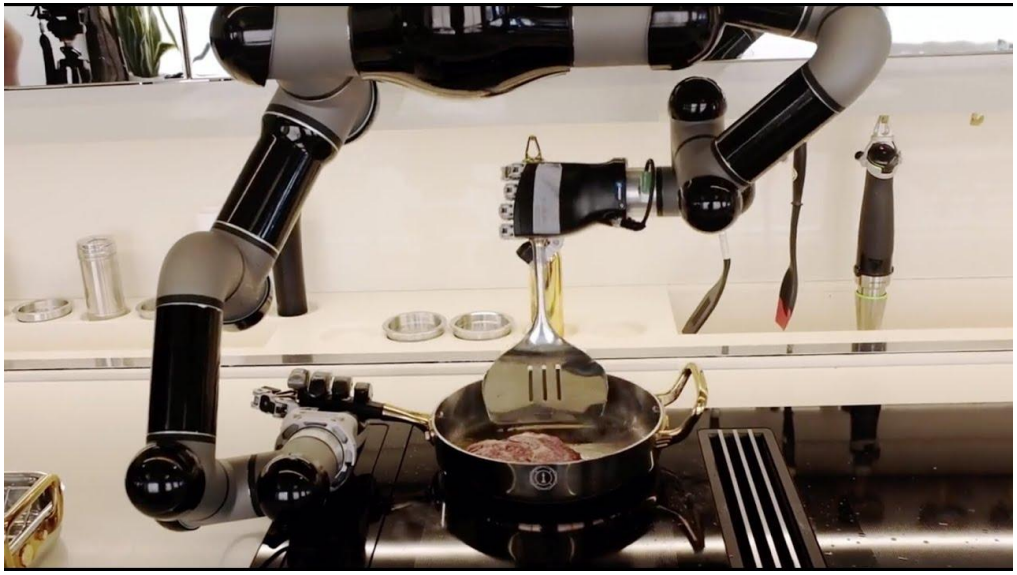
HOW TO PREPARE LASAGNE ALLA BOLOGNESE


```
fineness = 2.0; // 2 mm slices
carrot = chop(carrot, fineness, 0); // horizontally
carrot = chop(carrot, fineness, 1); // vertically
```

To prepare **lasagne alla Bolognese**, start with the meat sauce. Prepare the vegetable broth and ensure to keep it warm. Then take the pancetta bacon, cut it into strips first 1 and then chop it well 2. Separately, finely chop the carrots, onions, celery and set aside 3.

In a saucepan, pour a drizzle of oil and the pancetta bacon. Use a ladle to spread it well 4 and let it brown for a few minutes. Then add the chopped vegetables 5 and cook for 5-6 minutes 5. Then add the minced meat 6. Stir and raise the heat. Let the meat brown slowly, it must be well browned to seal the juices and be soft.

Deglaze with red wine 7 and let it evaporate completely before adding the tomato puree 8. The Bolognese sauce must cook for two hours. When it starts boiling again, you can add some hot broth, one or two ladles 9, and let the sauce cook for another couple of hours at least.



How to Paint Van Gogh's *Sunflowers* (F. 457)

Tack a blank canvas of the required size onto the drawing board.

Tape down a one-inch border around the canvas with masking tape. This leaves the necessary space to stretch the canvas onto wooden strainers later.

Tack up another *Sunflowers* painting from the shop for reference.

Mix quick-dry medium into the titanium white.

Prepare the palette with large amounts of lemon yellow, medium yellow, cadmium red orange and titanium white, smaller amounts of rose madder, burnt sienna, forest green, and cerulean blue.

Using leftover dirty paint anywhere in the yellow-to-red spectrum, sketch out the overall composition. Start with assigning the location of the vase's opening, the edge of the table, and then outline the shape of the vase. Draw six ovals with smaller inner circles where the six round sunflower blossoms should be. Mark out the ovals of the other flower blossoms, and the dried leaves near the vase.

Paint the first layer of the background: using a ¼ in (0.6 cm) brush, lightly combine two dabs titanium white and one dab lemon yellow. Make sure not to mix the two pigments together too much, and to leave traces of pure pigment in the mixture. Load this onto the brush. In a '井'-character pattern (that is, two horizontal lines followed by two vertical lines), fill in the background. Work lightly. Do not pull too hard or the effect of each thick brushstroke will be lost. Be sure that each brushstroke is visible. Be light and at ease. Do not grip the brush tightly. Remember to keep loading pure pigment onto the brush.

After the whole background is painted, use a ⅛ in (0.3 cm) brush to paint over the whole background again with the same two pigments and the '井'-character pattern, this time loading more paint onto the brush with every stroke. Make it brighter in the centre, darker at the edges of the painting.

Repeat the same two-layer process with the table, using long horizontal strokes. This time use equal amounts of lemon yellow and medium yellow on the ¼ in brush. Add in a few green strokes for depth.

Paint the upper part of the vase using medium yellow and cadmium red orange, mixing in some burnt sienna and rose where needed. Paint the lower part with lemon yellow, medium yellow, and white. Remember to paint the vase as a rounded volume.

With the same brush, pick up amounts of light yellow and medium yellow, fill in the large blossoms in very short strokes, turning each stroke towards the centre of the blossom. Add darker yellows as necessary to round out the volume of the blossoms.

Taking a palette knife, pick up equal amounts of two yellow pigments with the back tip of the knife. Dab or dot the paint in a circular pattern with light strokes pulling outward from the centre of each blossom. Progressively add darker yellows, orange, and sienna in order to round out each blossom. Work lightly and don't scrape into the wet paint.

With the ⅛ in brush, paint in the blossom petals in profile.

Using a third ⅛ in brush, load with dark green pigment and freehand draw each leaf. Make it look natural, like a leaf.

Using the same brush, fill in the centre of each flower with three strokes of green, sienna, and a dab of cerulean blue.

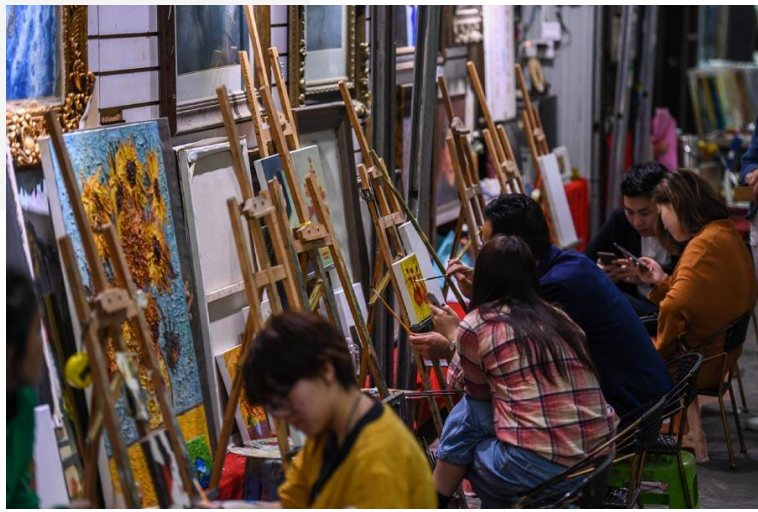
Using the same brush, but adding a trace of lemon yellow, paint in the stems.

Using the palette knife, dab white highlights on the vase, drag yellow highlights onto the green leaves, dab white highlights on buds. Using the ⅛ in brush with dark dirty paint on it, use the dirty paint on the palette and mix in dark green to get an off-black, then outline some of the vase, table, some leaves, and some blossoms.

Sign 'Vincent'.

Hang to dry.

Repeat.

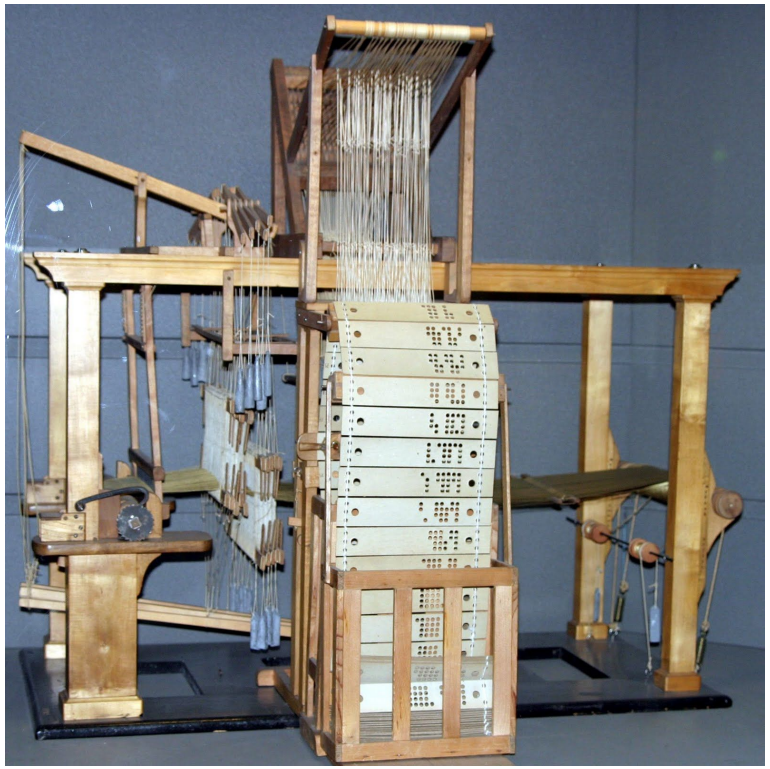


Dafen village, China

https://worksthatwork.com/5/the-reality-of-the-fake

https://en.wikipedia.org/wiki/Non-photorealistic_rendering

Jacquard Looms (1700s)

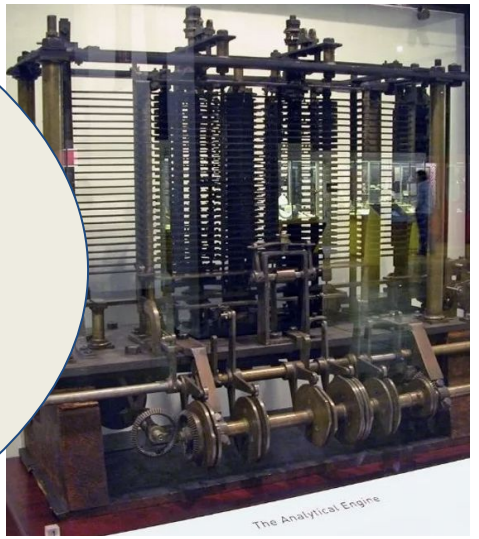


Hugely important in the history of computing, the idea of punch cards influenced the inventors of early computers such as Charles Babbage.

1850's - Ada Lovelace

"Mother of programming" starts working very young with Charles Babbage on his "Analytical Engine" (first general purpose computer)

"...the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."



Carlucci Aiello (2016) The multifaceted impact of Ada Lovelace in the digital age
https://www.sciencedirect.com/science/article/pii/S0004370216300224

Now, fast forward to today with creative coding…
But here is a nice timeline of programming languages starting from Ada and Charles

Programming for *Artists & Designers*

processes or procedures
for *creative making*

Who is nervous about coding? Who is new to coding?

Classroom Culture

A note regarding the "Impostor Syndrome"

- This is **NOT** a computer science class it is a **creative coding class**
  - Goal is not technical excellence but creativity
- If you have a strong technical background:
  - an opportunity to nurture the creative side of your work while **improving** your techniques.
- If you don't:
  - also an an opportunity to nurture the creative side of your work while **learning** techniques.
- Avoid bragging about your expertise, instead put it to practice and help/inspire your peers!
- We have a huge variety of technical, artistic and cultural backgrounds and we can all learn loads from each other!
- This is a place of learning: You will make mistakes but mistakes are a great way to learn coding

learning from each other

Mixing different abilities in the classroom

Maker Culture

- We share code (with appropriate citation)
- We study other people's code!
- We work in pairs in class
- We teach each other how to use tools
- Classes are arranged for collaboration
- Hatchlab vs. studio space

Don’t hesitate to slow me down!

Could you cover that example again?

Could you repeat what you just said? I do have an accent!

Can you do another example?

Can you explain that another way?

module structure

- Programming for Artists & Designers
  - Weekly topics / syllabus
  - Learning resources
  - Assessment structure

http://learn.gold.ac.uk

Referred to as learn.gold or the VLE (Virtual Learning Environment)

Conceptual Arc: Procedure & Translation*

**Translation: translating from one medium into another, from one form or process into another.*

Making Computational Creative Tools
Generative & Procedural Art
(static/linear output/media)

Making Live Computational Art
Real-time & Interaction
(live computational output/media)

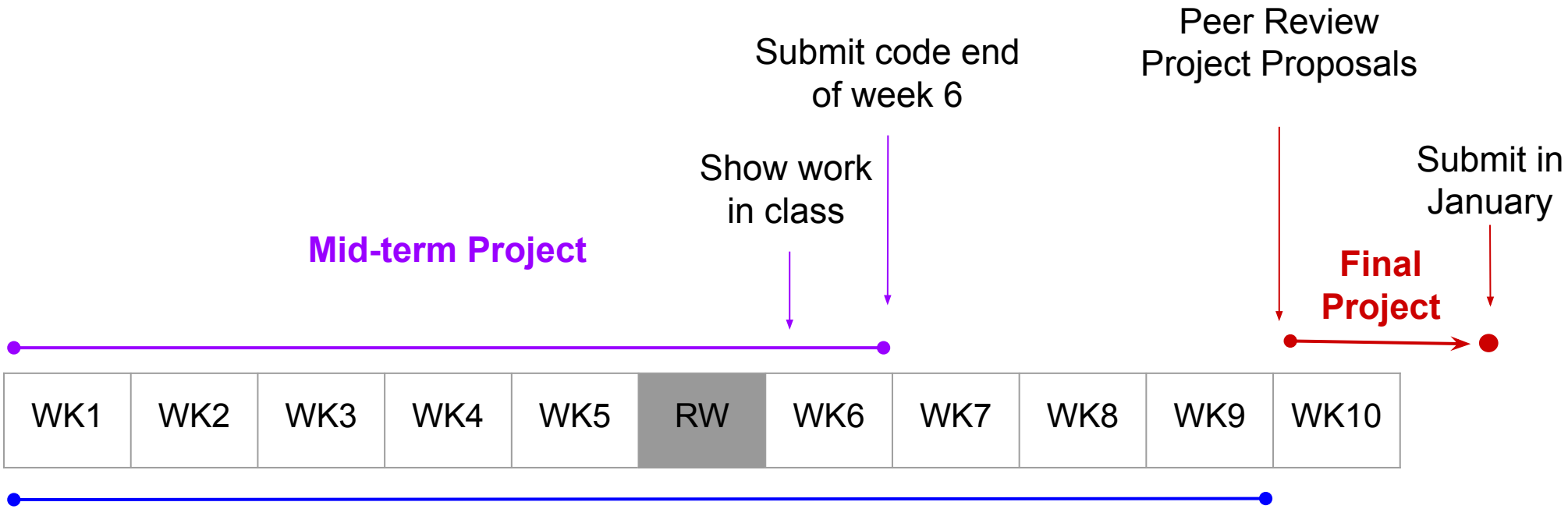
WK1	WK2	WK3	WK4	WK5	RW	WK6	WK7	WK8	WK9	WK10

Weekly Continual Assessment: Weekly coding topics & creative inspiration
Mid-term Project: Static output
Final Project: Personal choice

Topics: Computer graphics, digital media, print-making / fabrication, generative art, interactive art, sound, visual communication.

Our weekly sections serve as a quick preview of what's coming up.

How you are assessed



Weekly Continual Assessment:

- *Quiz*, to help test your knowledge of the core coding & technical fundamentals.
- Creative Coding prompt with a *code submission*, to practice coding & to do creative ***sketches*** with code.

Weekly Continual Assessment: **10%**

Mid-term Project: **10%**

Final Project: **80%**

more information

The mid-term project starts today?

“Use the topics, and code provided within the first half of the term and your own weekly experiments as a starting place. You could reflect on and **refine** one of those experiments or combine two to make your work.“

please read the prompt this week

what does sketching mean to you?

Weekly sketching share forum

wooclap

Mini comfort break?

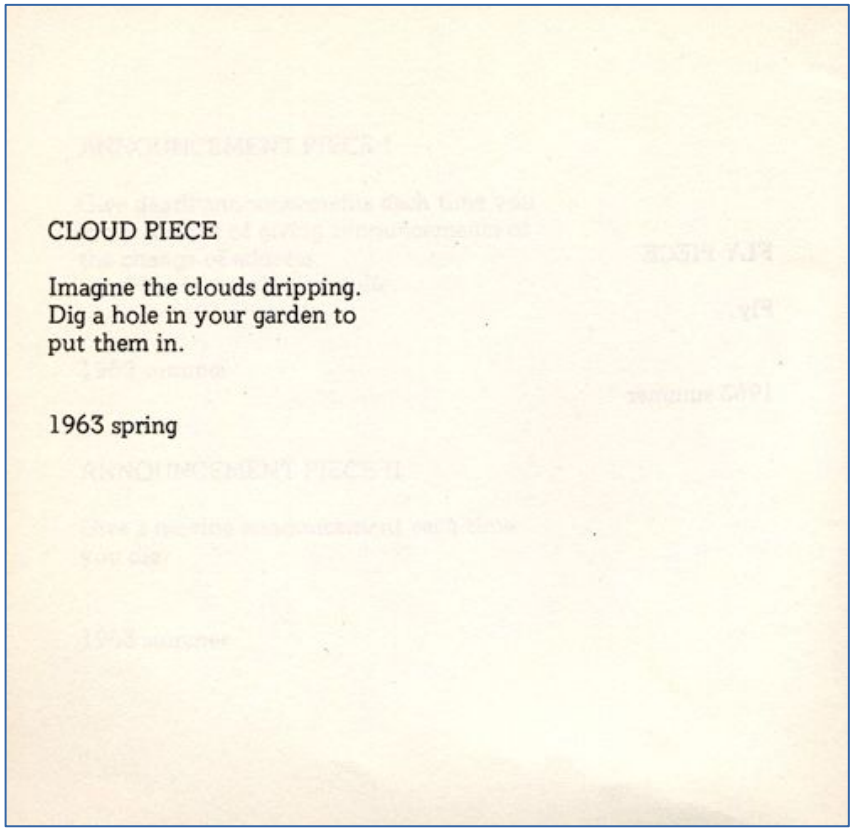
(10 mins)

Programming

processes or procedures

instructions, process, procedure

- procedural art
  - instructions and diagrams as a form of art

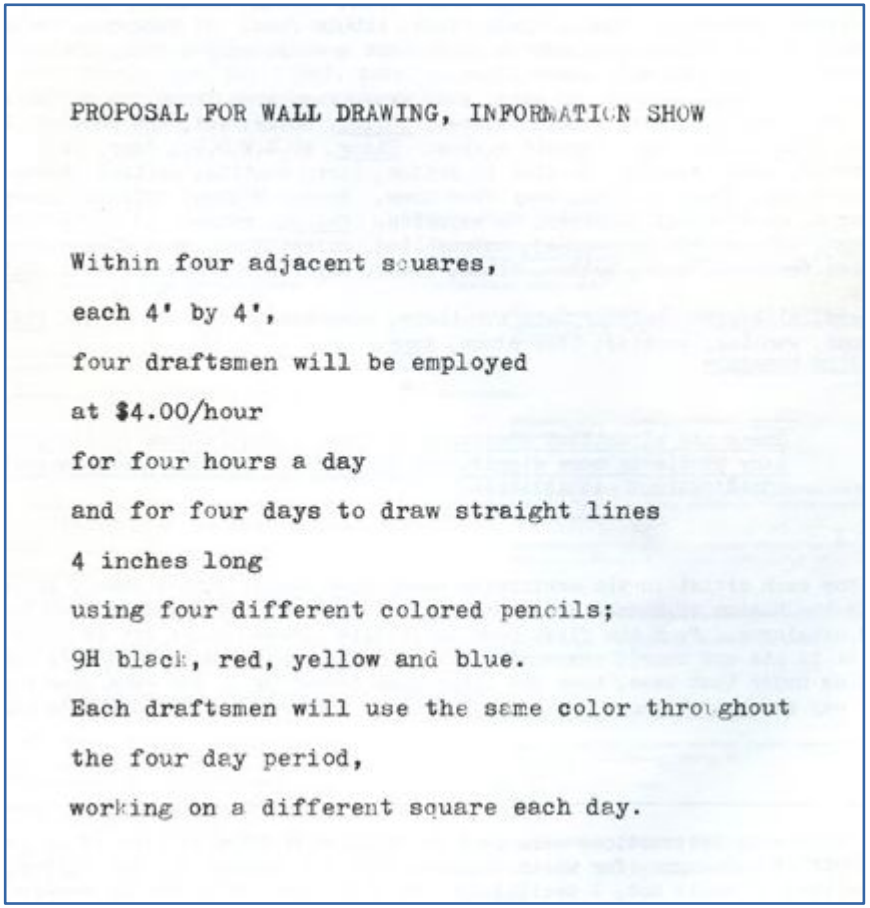


CLOUD PIECE

Imagine the clouds dripping.
Dig a hole in your garden to
put them in.

1963 spring

Yoko Ono (1963)

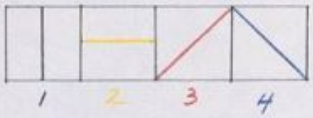


PROPOSAL FOR WALL DRAWING, INFORMATION SHOW

Within four adjacent squares,
each 4' by 4',
four draftsmen will be employed
at $4.00/hour
for four hours a day
and for four days to draw straight lines
4 inches long
using four different colored pencils;
9H black, red, yellow and blue.
Each draftsmen will use the same color throughout
the four day period,
working on a different square each day.

Sol LeWitt (1970)

DIAGRAM



This is a diagram for the Sol LeWitt wall drawing number 49. It should accompany the certificate if the wall drawing is sold or otherwise transferred but is not a certificate or a drawing.

CERTIFICATE

This is to certify that the Sol LeWitt wall drawing number 49 evidenced by this certificate is authentic.

A wall divided vertically into fifteen equal parts, each with a different line direction and color, and all combinations.

Red, yellow, blue, black pencil
First Drawn by: Chris Hansen, Nina Kayem, Al Williams
First Installation: Jewish Museum, New York, NY. June, 1970

This certification is the signature for the wall drawing and must accompany the wall drawing if it is sold or otherwise transferred.

Certified by Sol LeWitt
Sol LeWitt

 Date

Sol LeWitt

Artists & Designers

creative making

Group Activity

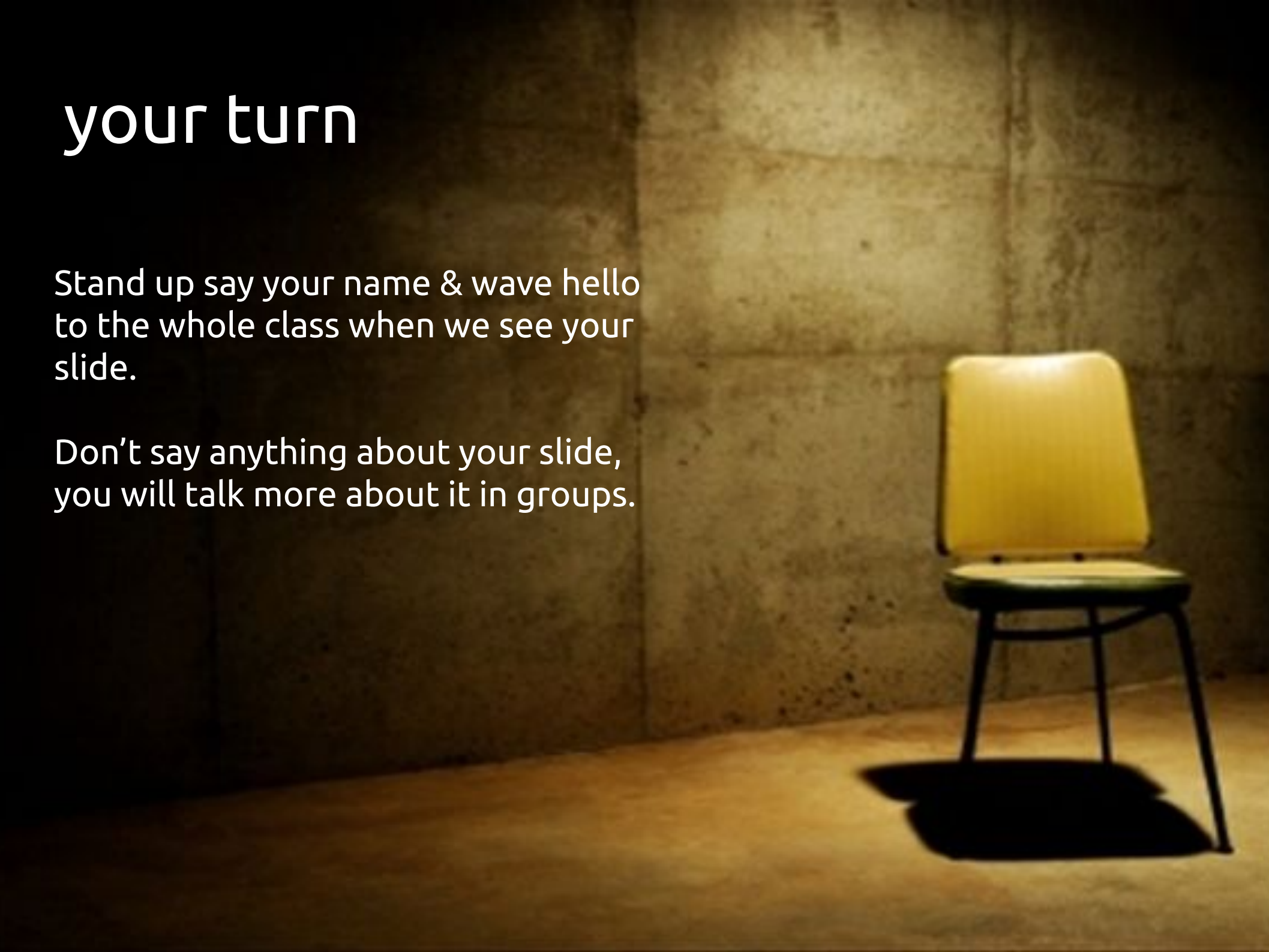
We are all going to see each other's slides now as a way of introduction.

But first, stand up, move around and form a group of 4 - 5 people.

your turn

Stand up say your name & wave hello to the whole class when we see your slide.

Don’t say anything about your slide, you will talk more about it in groups.



Group Activity

Next, 30 mins to work in your group.

5 mins to introduce yourselves:

1) What type of maker are you? Talk a bit about what you showed.

Do the group activity:

“A Score for Drawing”.

What do we mean by score?

Long Break
&
Troubleshooting installs

15mins

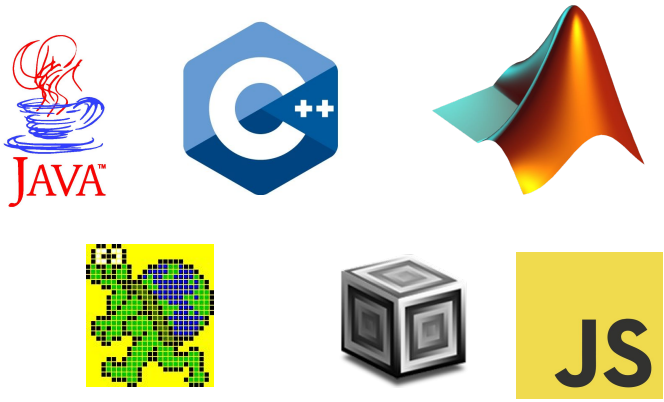
A quick intro to C++ and openFrameworks

(next week -> more in-depth intro to C++)

Why so many programming languages?

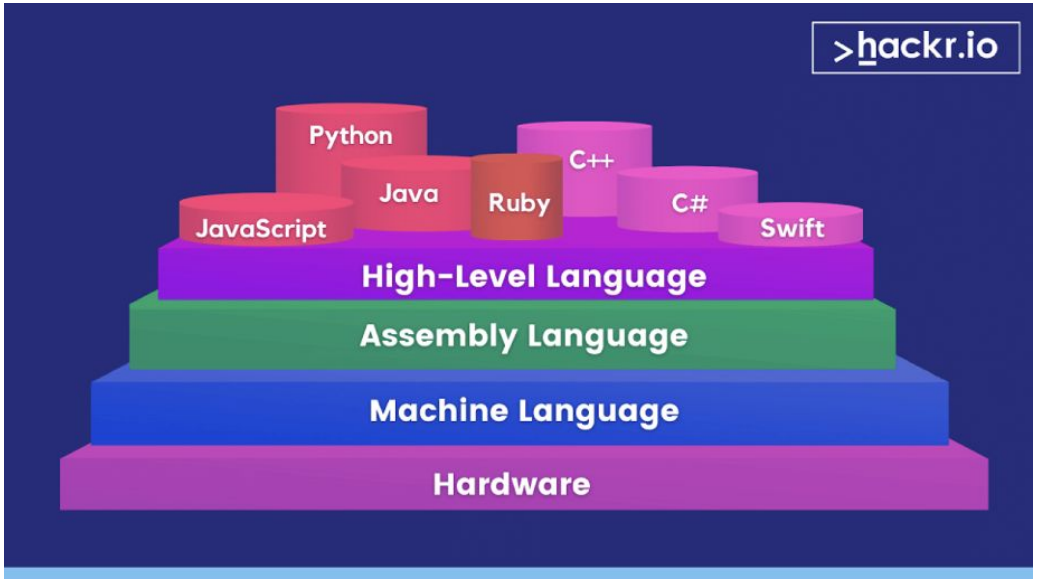
- Most are not **ideal** for all uses

- Many are designed for specific tasks:
  - Educational - LOGO, Basic, Scheme
  - Musical – Max, Supercollider, CSound
  - Network - javaScript, PHP
  - Scientific - MatLab, Fortran
  - Micro-chip - picBasic, Assembler
  - “Esoteric” - Brainfuck, ArnoldC

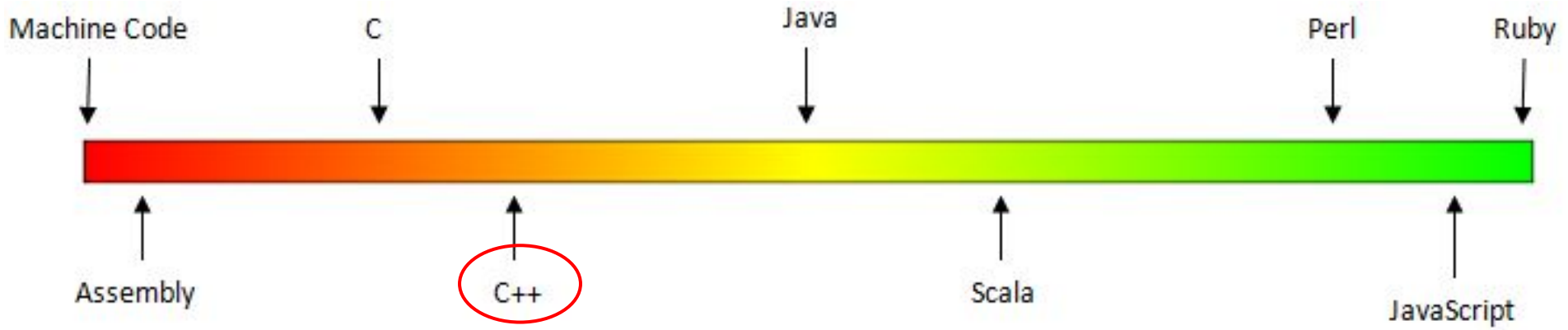


MAX6

“General purpose” high level languages



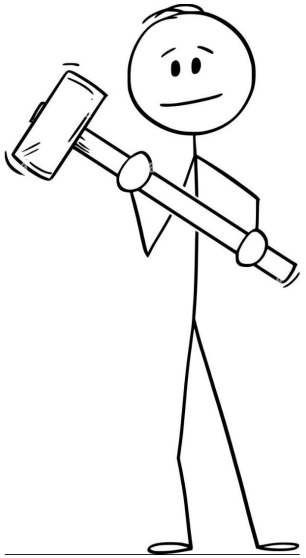
- a vocabulary of about 30 terms that sound more familiar:
  - while, for, if/else, return



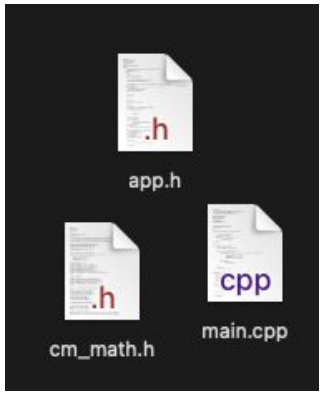
Faster, more “foreign” language

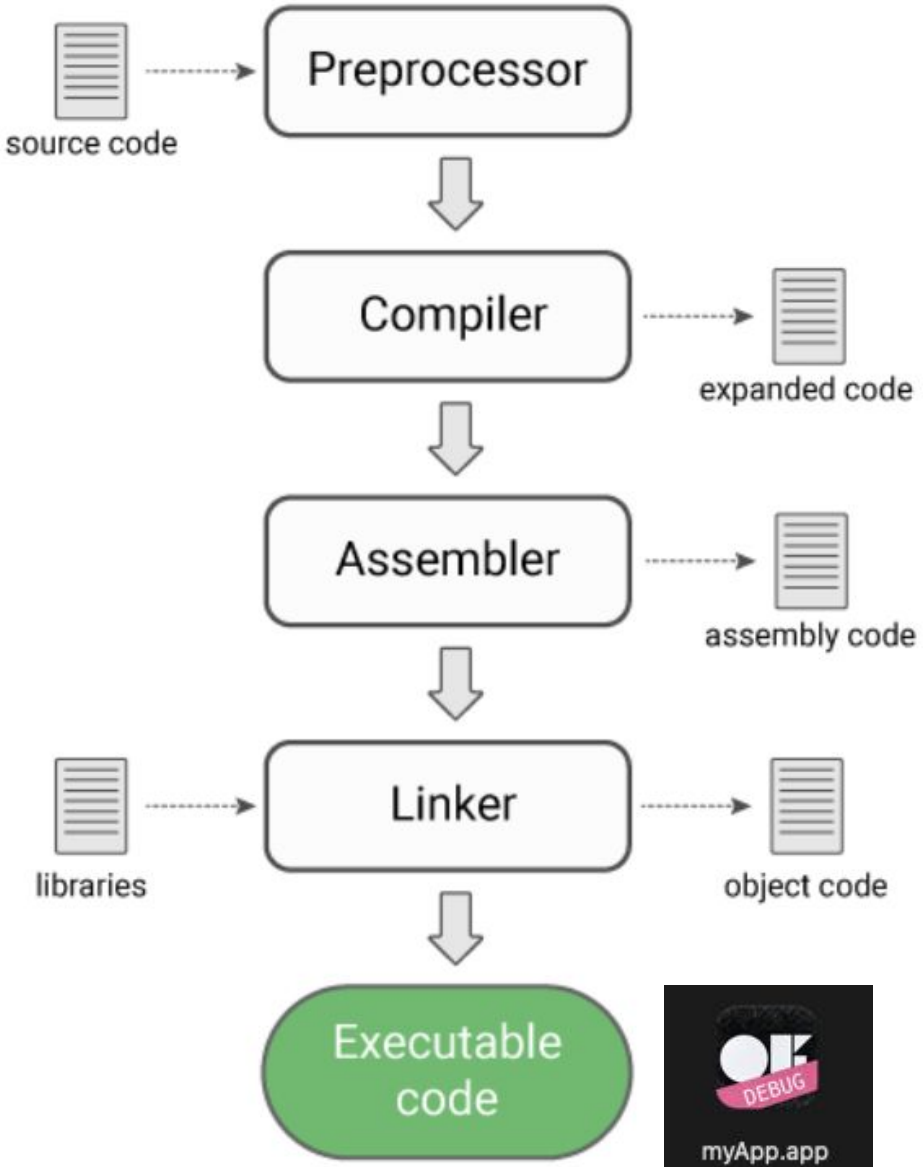
Slower, more “familiar” language

- C/C++ are general purpose languages
  - C++ is a superset of C
  - Designed to solve a variety of tasks
- Java, Javascript, PHP, and many other languages have a similar syntax (e.g. curly braces `{ . . . }` delimiting “blocks”)
  - C++ is “lower level”, i.e. closer to the metal and thus typically faster



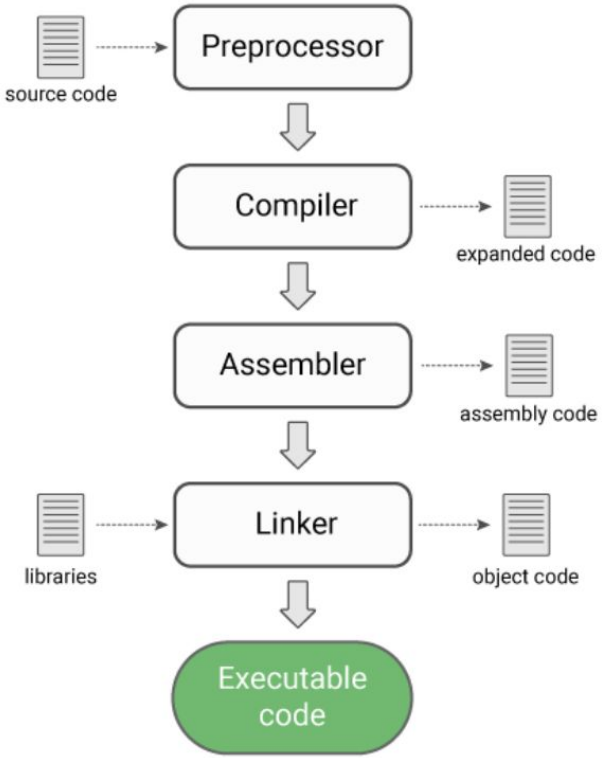
How does C++ work?





- The most basic C++ program

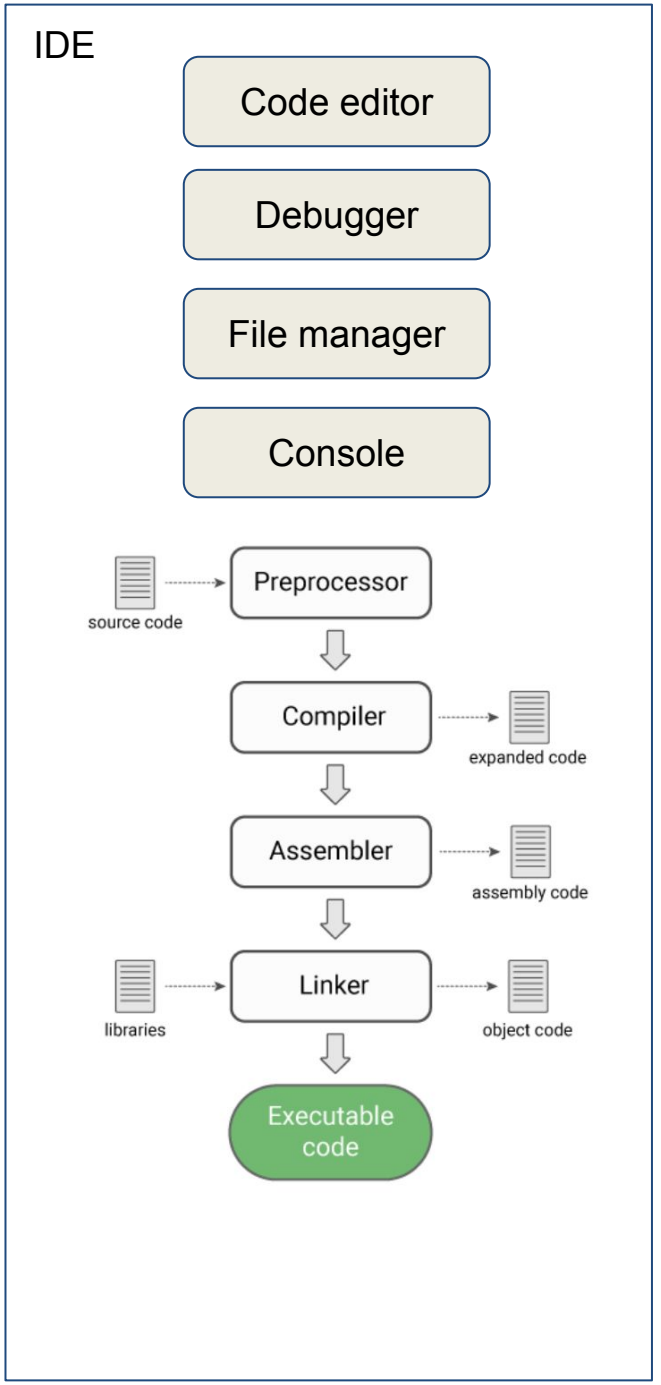
```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]) {
  // This is the C++ way of outputting to the console
  cout << "Hello, C++ world!" << endl;
  return 0;
}
```

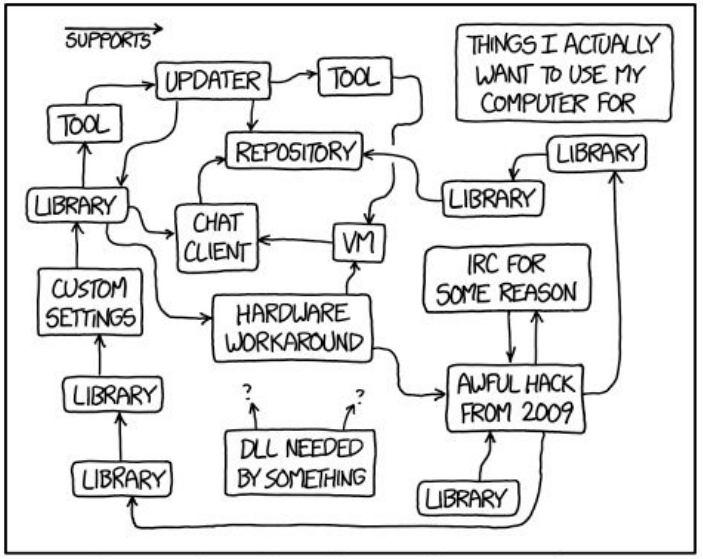


- Prints “Hello, world!” to the console
- `main` is the “entry point” for most C/C++ programs
  - Program flow begins from there

[Code example]

- Managing all these steps can be daunting
- **Integrated Development Environments** (IDEs) make it easier
  - Allows you to edit, compile, run and debug code in a specifically designed environment
  - Organize multiple source code files, libraries and resources in a “project”
  - Examples of an IDE for different platforms are:
    - XCode
    - Visual Studio
      - (**NOT** Visual Studio Code/VSCode)
    - QTCreator





- C++ is an extremely powerful and versatile language
- BUT More complex C++ projects can require managing many **libraries**
  - E.g for generating graphics/sound, specialized math operations, handling input/output devices, but also for some standard functionalities of C++
- **Daunting task also with an IDE!**
  - Platform/version compatibility issues
  - Different syntax conventions
- **OpenFrameworks (OF) comes to the rescue!**
  - So we can focus on creative applications rather than “admin”

FAQ

- Is it a program?
- Is it a programming language?
- Is it a library?
- What's a framework?

what is OF ?

- a toolkit for artists/designers working with interactive design and media art
- provides a simple and intuitive **framework** for experimentation
  - A structured environment that facilitates integrating different libraries/technologies
- Not a library. It works as a glue, wrapping together several C++ libraries commonly used for creative coding

OF history

Zach Lieberman

Arturo Castro

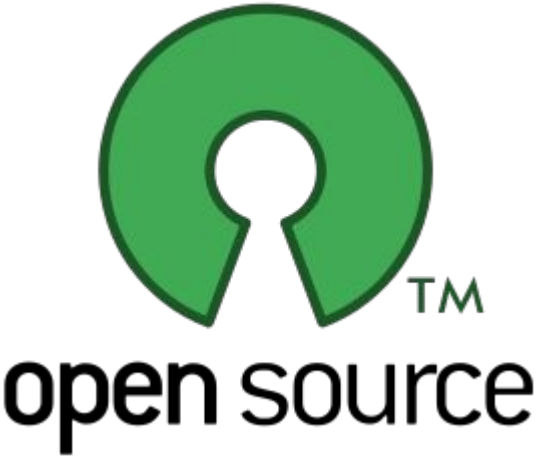
Theo Watson

Credit to:

- hundreds of other contributors (on Github)
- Processing was a great influence
  - MIT Media Lab / Parsons School of Design

free vs. open source

GNU Project



- open source: way of developing software
- free software: social movement

https://en.wikipedia.org/wiki/GNU_General_Public_License

ofx + IDE

- directory structure
  - src
  - core addons + extra addons
  - examples
- project generator demo
- where your code should live
  - *path to openFrameworks*/**apps/myApps**
- resources:
  - books / documentation / forum / ofBook

an empty project

- main.cpp
- ofApp.cpp
- ofApp.h

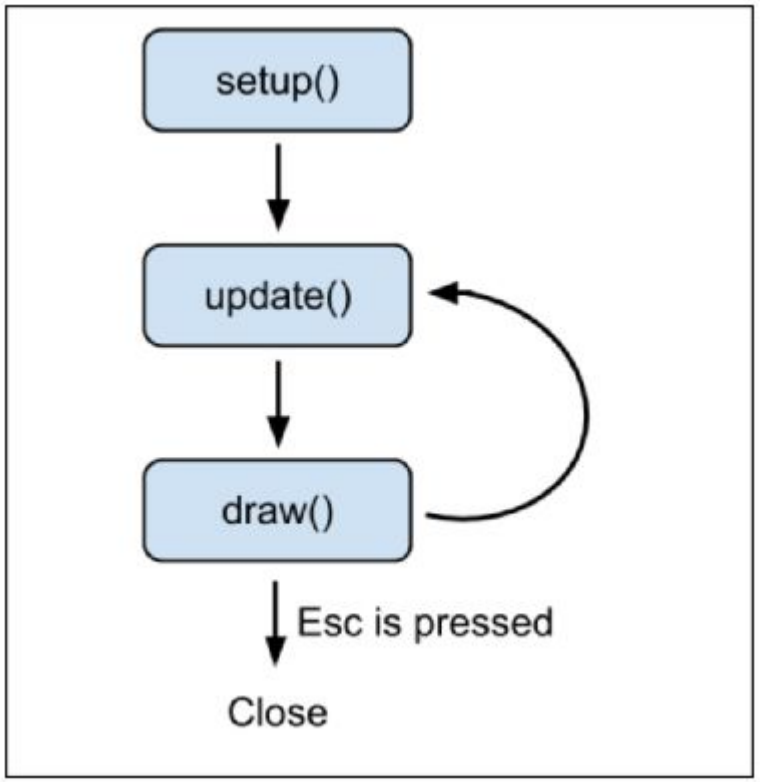
Printing to the console

```
cout << "Hello World" << endl;
```

Helpful for ***debugging***

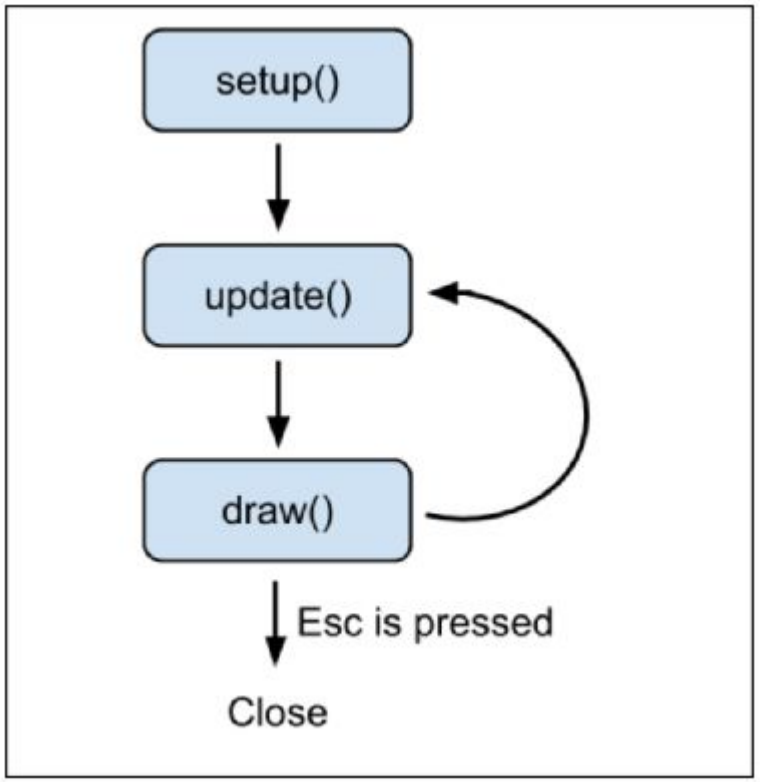
Debugging means finding & solving problems in code

setup() vs. update() vs. draw()
& events



setup() vs. update() vs. draw()
& events

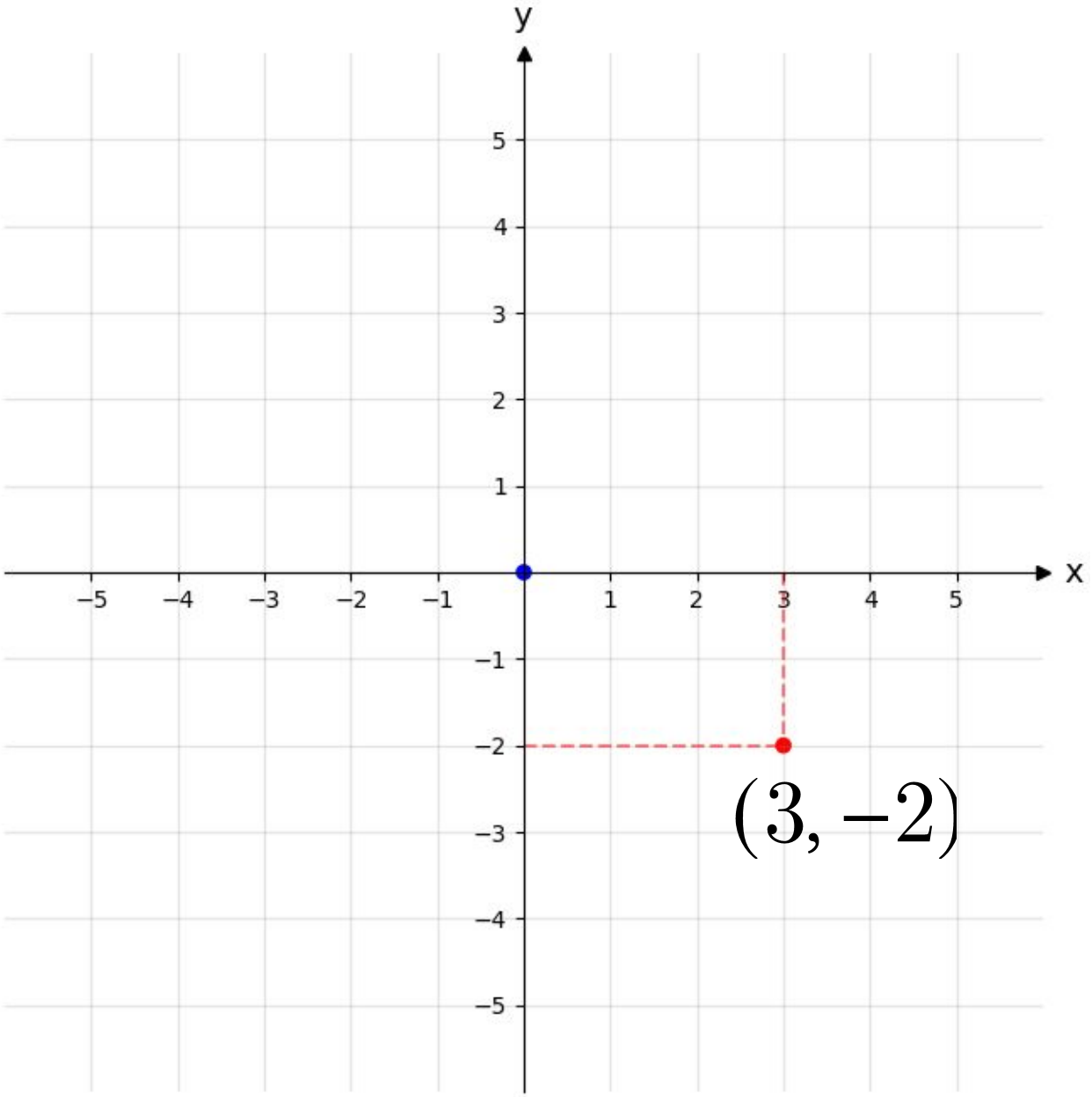
```
//--------------------------------
void ofApp::setup(){
}

//--------------------------------
void ofApp::update(){
}

//--------------------------------
void ofApp::draw(){
  ofBackground(255, 0, 0);
  ofSetColor(255);
  ofFill();
  ofDrawCircle(200, 200, 100);
}
```




```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(255, 0, 0);
  fill(255);
  noStroke();
  circle(200, 200, 200);
}
```

Before we draw:

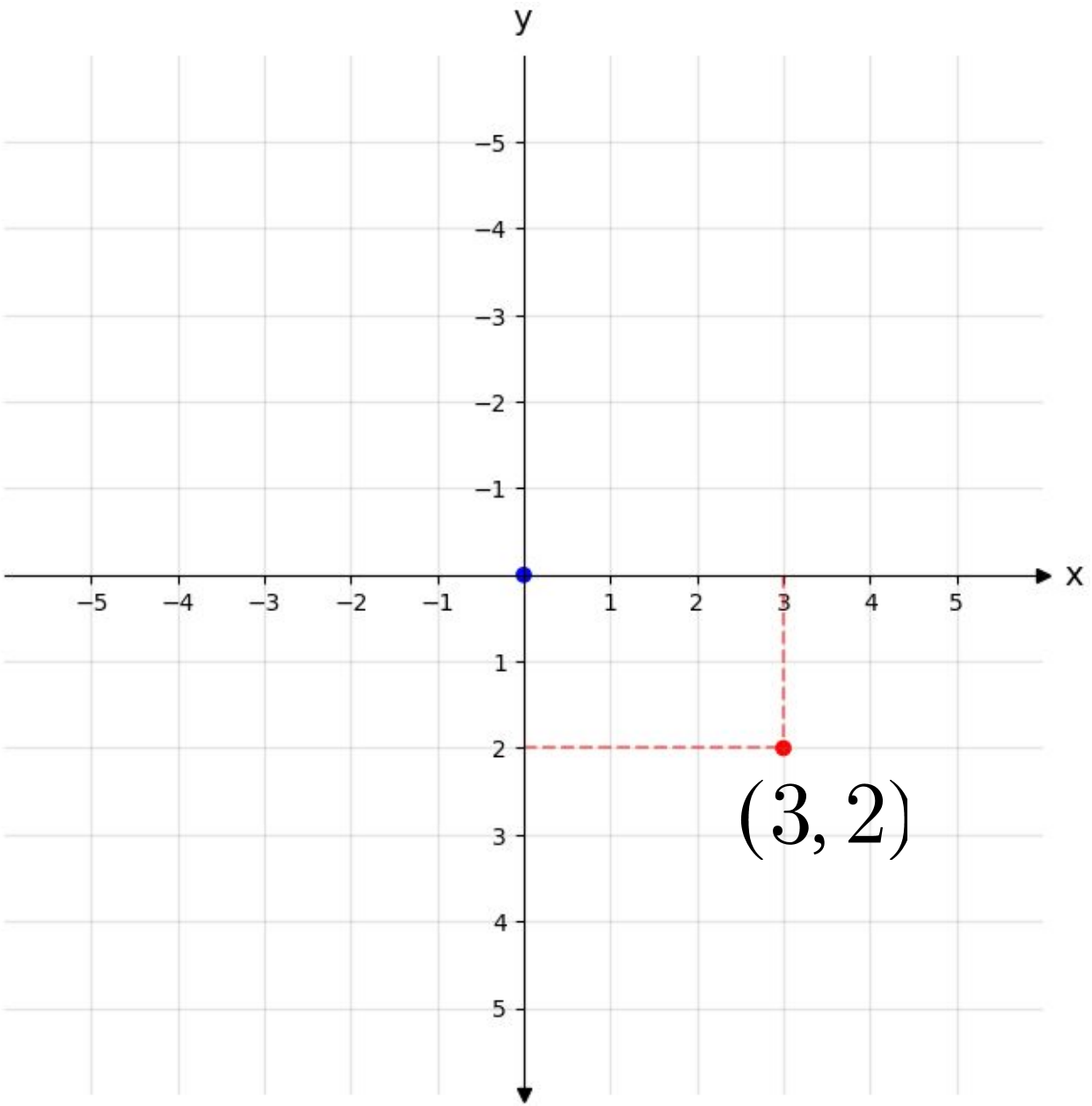
Cartesian coordinates



René Descartes

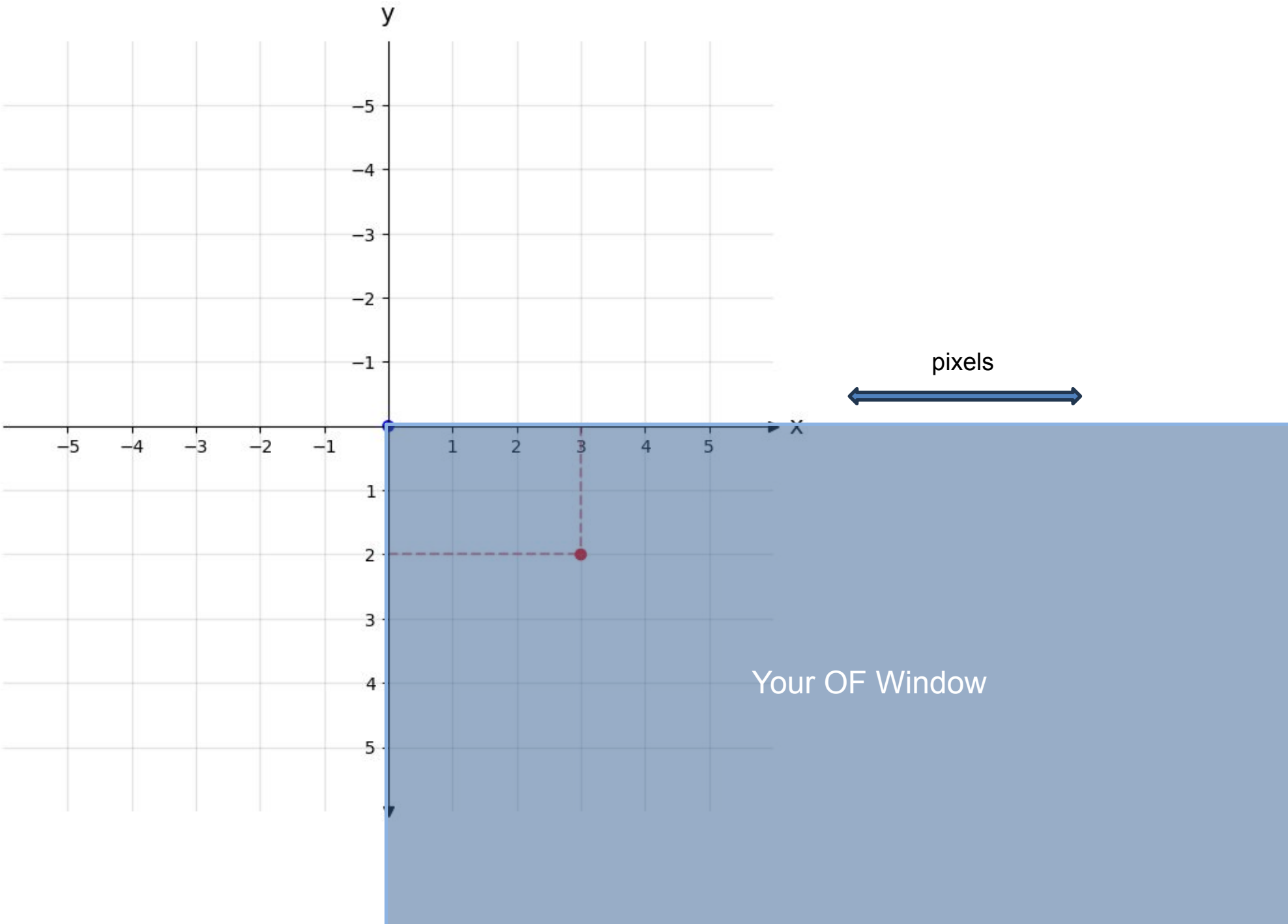
- Coordinate system with ”Y” pointing up
- Blue point at (0, 0) is the “Origin”

“Window” coordinates

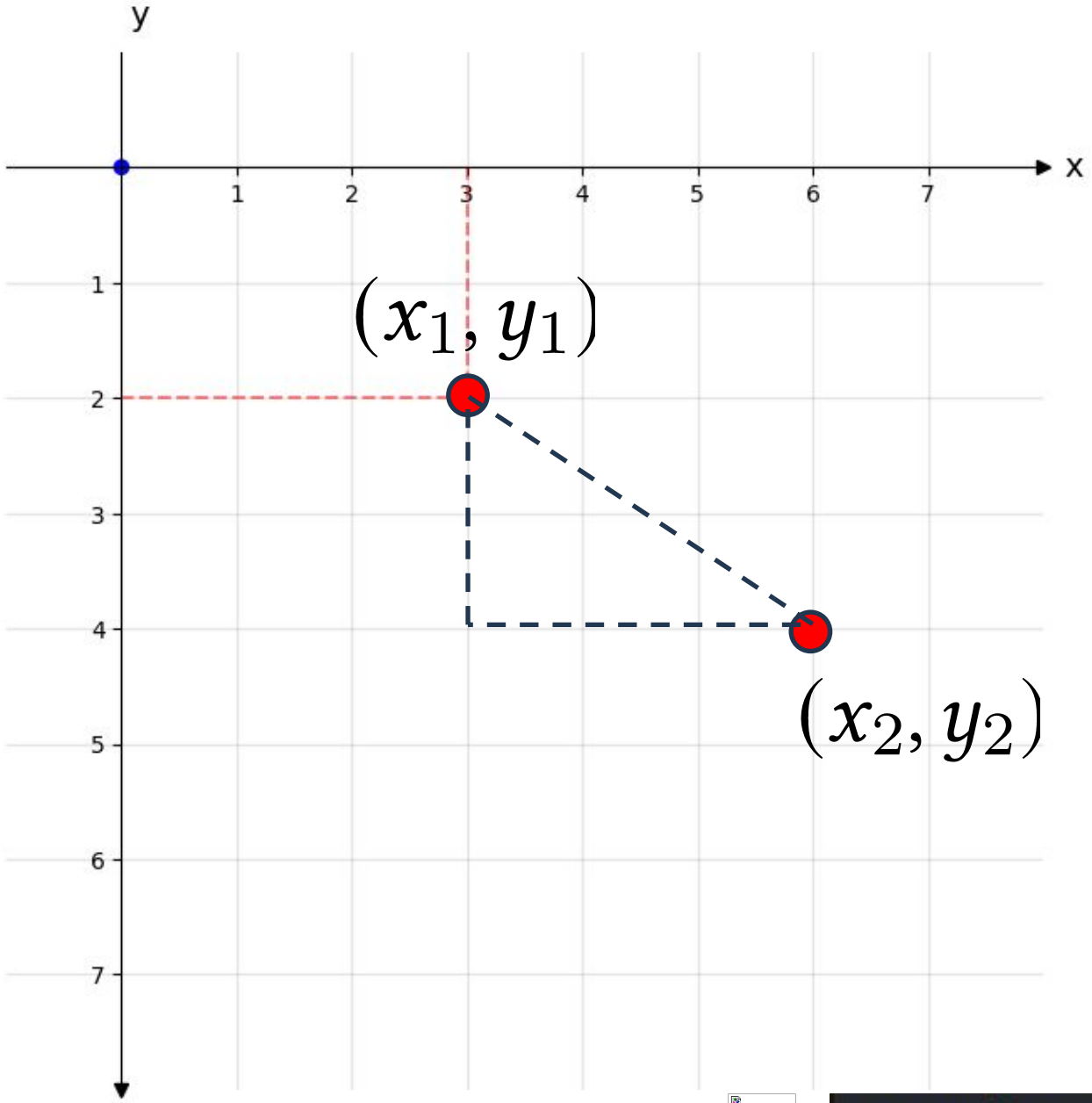


- Coordinate system with ”Y” pointing down
- Convention for 2d graphics
  - And OE when it starts up

“Window” coordinates



Measuring distance



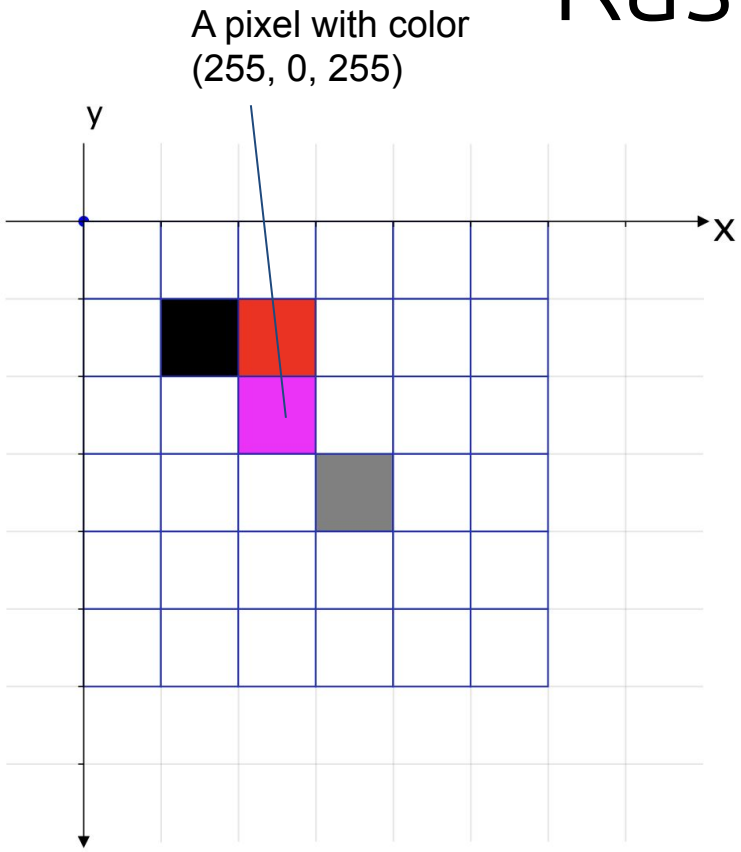
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

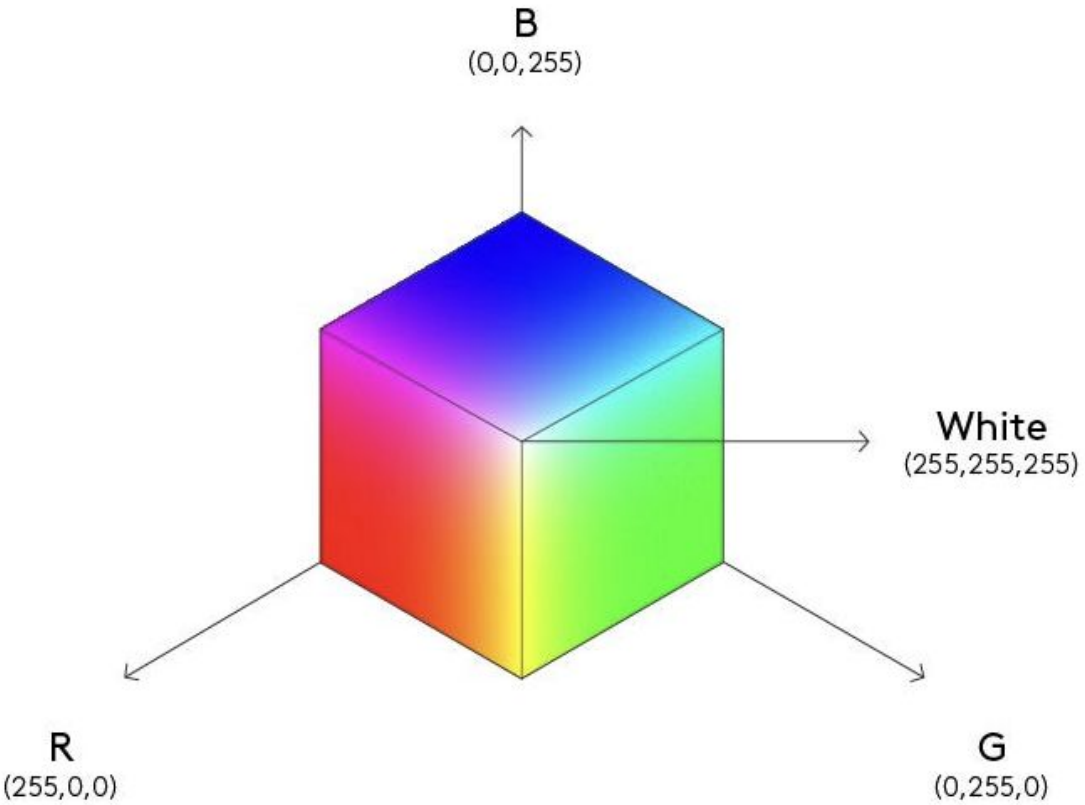
```
d = sqrt((x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1));
```

demo 

```
d = ofDist(x1, y1, x2, y2);
```

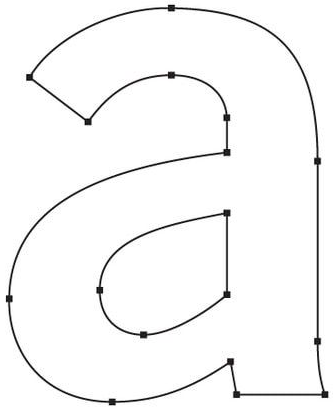
Raster graphics



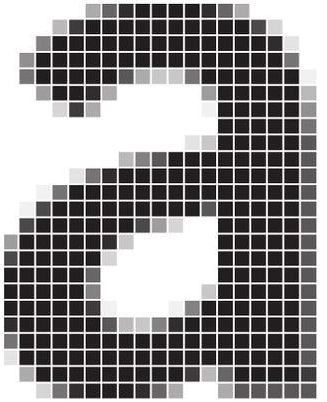


- Images on the computer screen are **approximated** by an arrangement of colored pixels over a grid (raster)
- The color of each pixel can be described with a red, a green and a blue component (RGB).
- Specifying graphics in this way can be inconvenient:
  - Storage increases with size of the image
  - Information about shapes is lost (just “pixels”)
    - E.g. a circle is just a approximation made of “little squares”
  - When increasing size information is lost (aliasing)
  - We are stuck in the screen, projections or print (if size is large enough)

Vector graphics



VECTOR



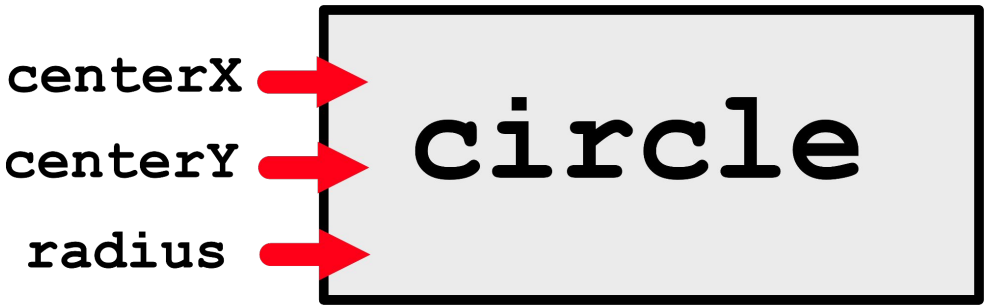
RASTER

- The way we draw in OpenFrameworks (well, mostly)
- In vector graphics, we specify shapes through their parameters, e.g.:
  - A circle can be defined with it's center (x, y) coordinates and its radius
  - A line with a pair of coordinates (x1, y1) and (x2, y2)
  - Etc…
- These shapes are then displayed by converting them to pixels, a process known as rasterization.
  - Very fast and is taken care of by a library called OpenGL
- **Advantages**:
  - More memory efficient for "graphic images"
  - No loss of quality with scale
  - We have information on how to reproduce each shape
  - Suitable for guiding a drawing machine, fabrication, extrusion in 3d software

OpenGL®

Vector graphics

- Back to our functions as a magic “box”



  - The functions makes a circle appear on the screen

- In OF this would be something like

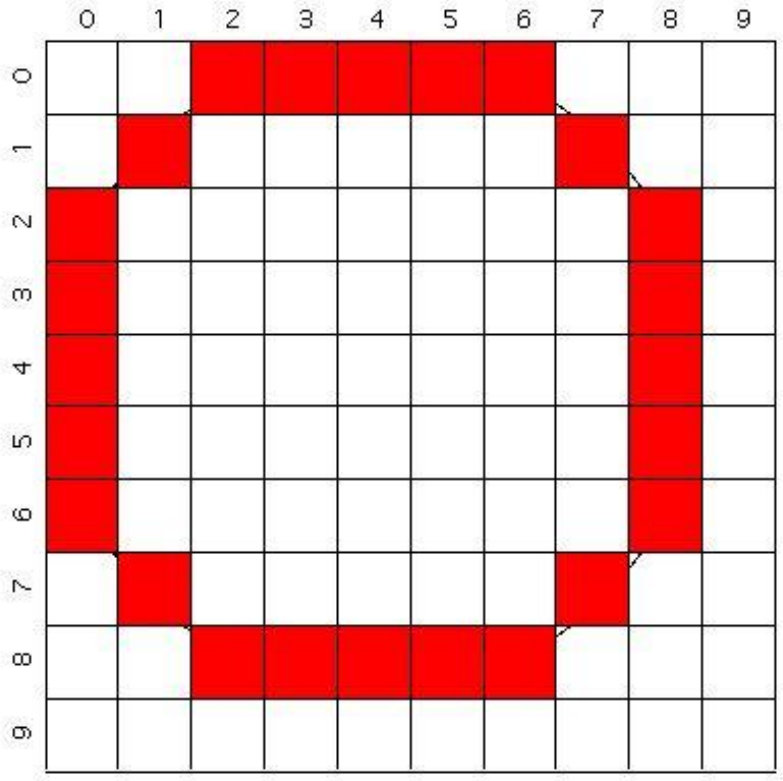
```
ofDrawCircle(centerX,centerY,radius);
```

when does oFx draw the elements in `draw();`

```
void ofApp::draw()
{
   ofBackground(255);
   ofDrawRectangle(100,100,100,100);
   ofDrawEllipse(100,100,50,50);
   ofDrawTriangle(100,100,50,50,20,20);
}
```

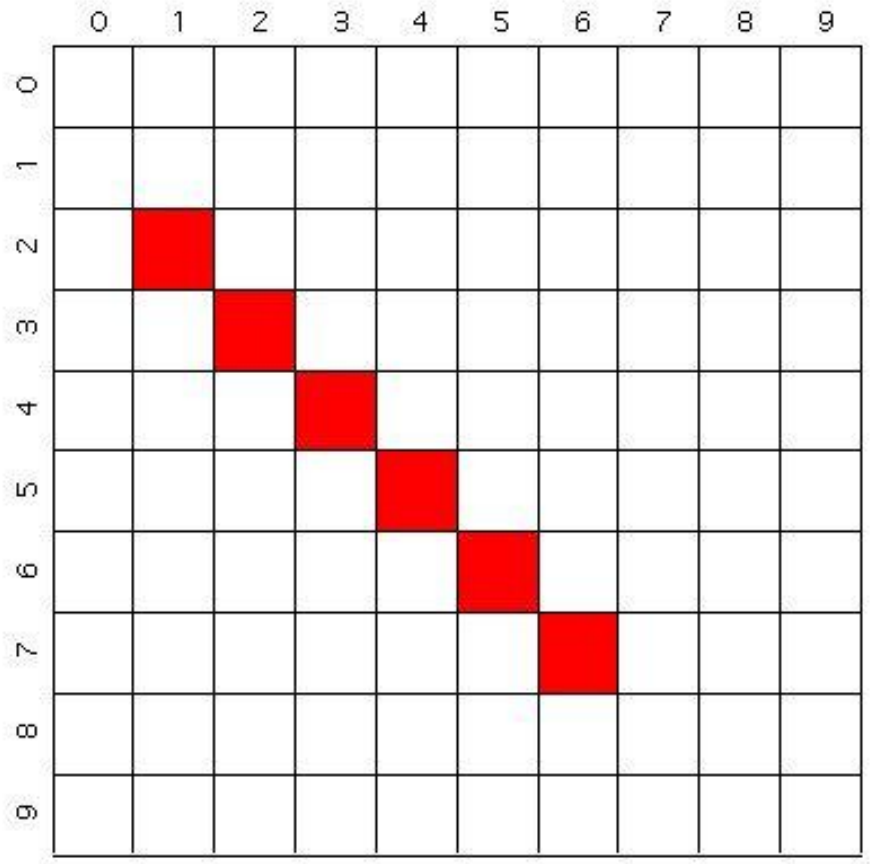
circle

ofDrawCircle(centerX,centerY,radius);



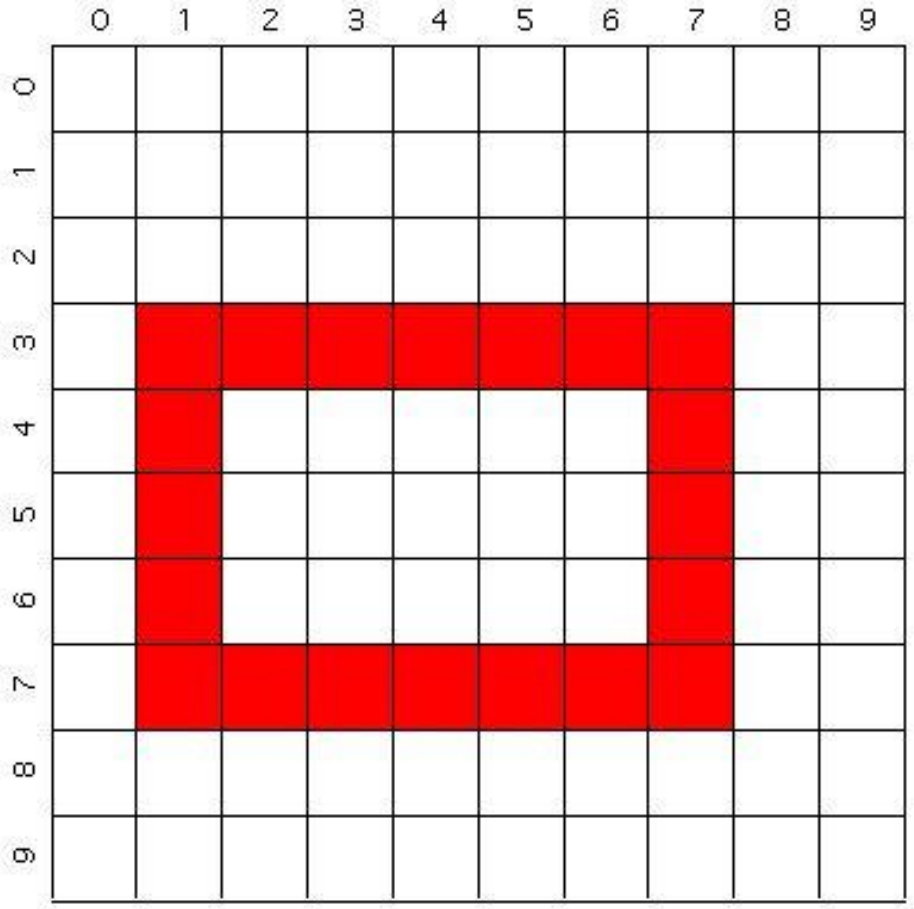
line

ofDrawLine(fromX, fromY, toX, toY);



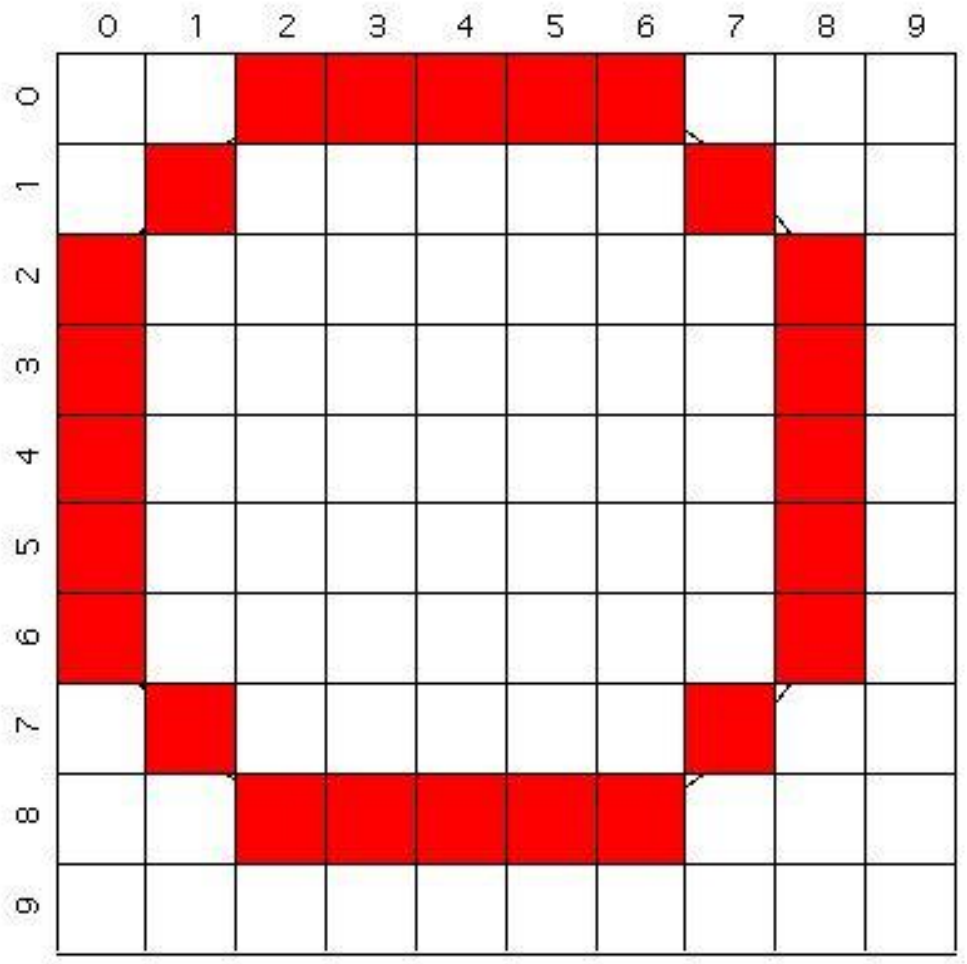
rectangle

ofDrawRectangle(topLeftX, topLeftY, width, height);



ellipse

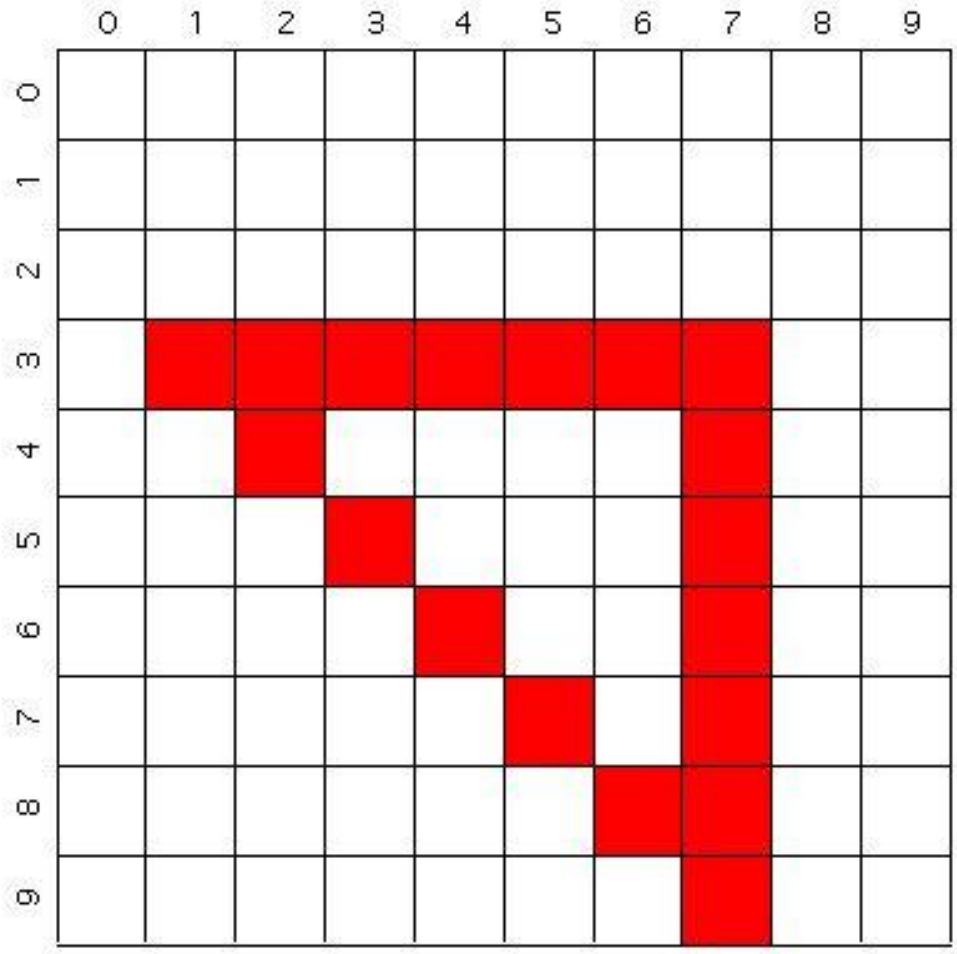
ofDrawEllipse(centerX,centerY,width,height);



- primitives2D

triangle

ofDrawTriangle(point1X, point1Y, point2X, point2Y, point3X, point3Y);

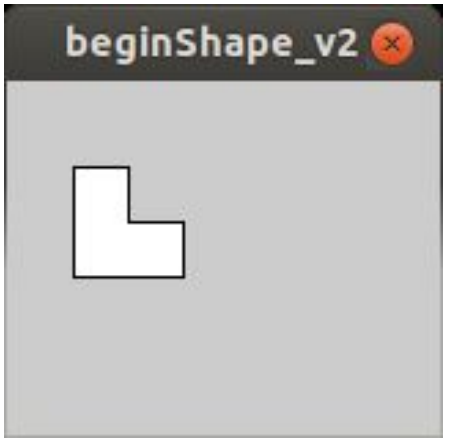


2D primitives

```
ofDrawLine(x1, y1, x2, y2);
ofDrawRectangle( x, y, w, h );
ofDrawTriangle(x1, y1, x2, y2, x3, y3);
ofDrawCircle(x, y, r);
ofDrawEllipse(x, y, w, h);
ofDrawCircle(x, y, r);
```

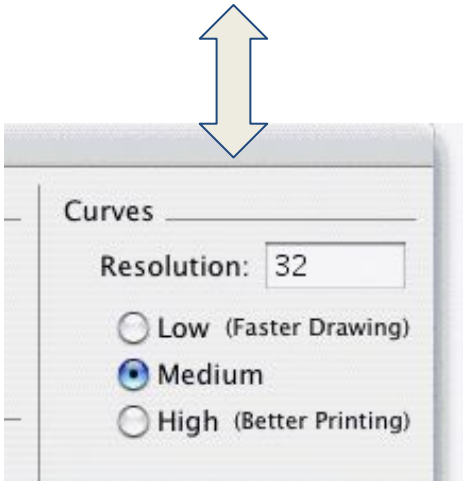
making your own shapes

```
ofBeginShape();
ofVertex(20, 20);
ofVertex(40, 20);
ofVertex(40, 40);
ofVertex(60, 40);
ofVertex(60, 60);
ofVertex(20, 60);
ofEndShape();
```



some more commands

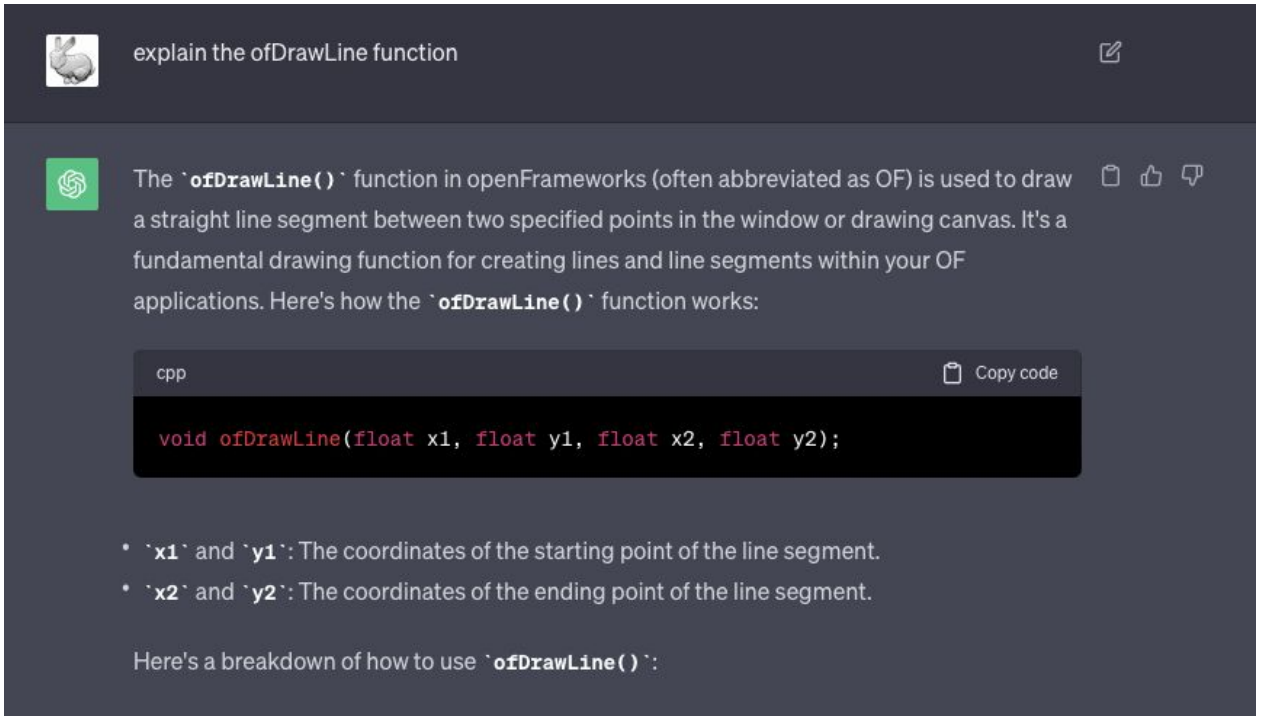
```
ofBackground( R, G, B );

ofSetColor(R, G, B, a);

ofFill();

ofNoFill();

ofSetLineWidth( aNumber );

ofSetCircleResolution( aNumber );
```



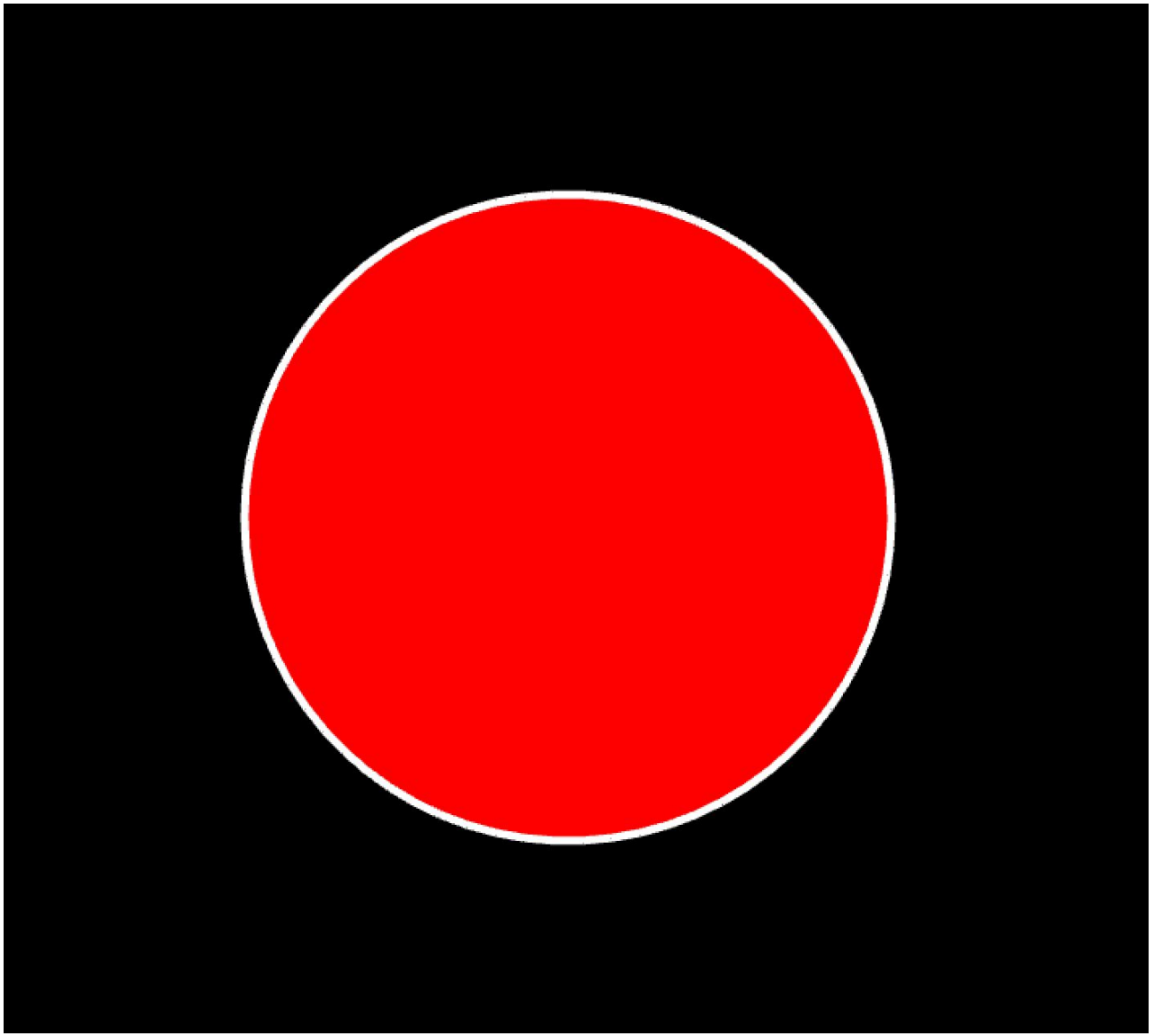
shapeOutline

Looking up documentation

- Use OF official documentation: https://openframeworks.cc/documentation/
- Or forum / ofBook
- Because many people use OF, ChatGPT can be also useful ☺



how do I get this?



shapeOutline

Explain Homework

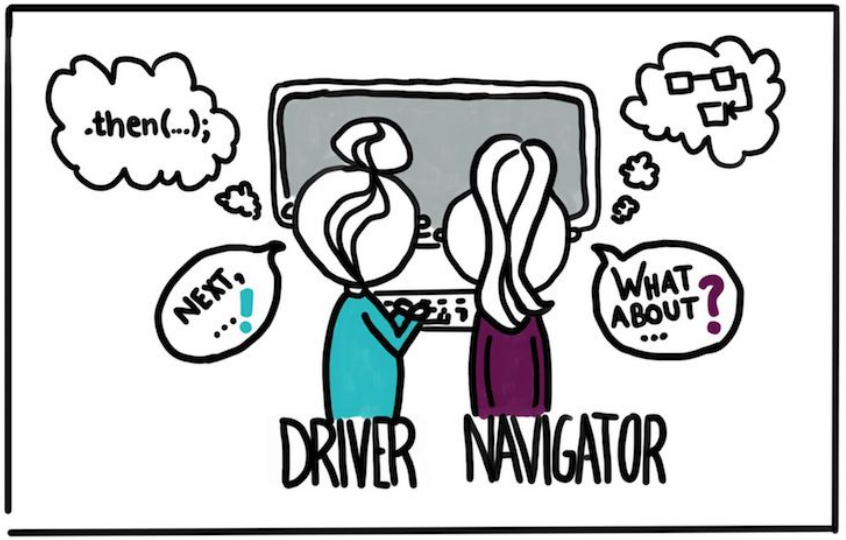
Lab assignment:

- openFrameworks troubleshooting
- Shape Stack

Home assignment:

- Instructions

Pair programming



Driver: write code, explain what you are doing, suggest ideas but focus on details of coding task

Navigator: reads instructions and look up commands, correct/review code, suggests ideas, keep a view on bigger picture

Only driver writes code, switch roles during session one or more times

Some examples of pair programming approaches:
https://martinfowler.com/articles/on-pair-programming.html

In industry often referred to as “Extreme programming”

We find Pair Programming to be one of the most effective ways to keep our developers productive, sharing knowledge and experience. It has certainly helped us build strong teams, reduce our defect rate and keep people happy. It is important to remember that both roles in the pair are equally important with specific expectations and responsibilities during pairing. Also, neither driver or navigator responsibilities last forever; it is encouraged to change roles several times during the course of a pairing session, and both wheels have to be in good shape.

ThoughtWorks®

But keep in mind, we are doing creative coding and not building a product!