

DATT 2040

Math Code Art

Grids (questions from lecture 1)

In our first lab we will be thinking about grids.

- How can we draw a grid through code?
- How can we make a simple and highly defined structure more interesting?
- How far can we deconstruct a grid structure before it stops resembling a grid?

To lab bring.

- Yourself
- Pen and paper

Reading to do before lab (at the very least skim through and look at the pictures):

<https://www.rightclicksave.com/article/an-interview-with-vera-molnar>

And this [Processing overview](#)

In today's lab we will be starting with simple games where we create rules.
At first we will make open ended rules – like the instructions by Sol Lewitt.

We will then introduce ourselves to the Processing environment, build a simple grid structure and experiment with how these rules can be applied. We will ask questions such as:

- How can we make a simple and highly defined structure more interesting?
- How far can we deconstruct a grid structure before it stops resembling a grid?

This [Processing overview](#) is a handy learning companion

General lab format

9.30 - Arrival and 15 minutes of set up/wake up. Lab delivery will start at 9.45 so make sure you are here and ready to go for 9.45.

9.45 - lab content delivery.

10.30 - break.

10.45 - Working on lab projects and weekly sketch assignments. Make sure to use this time to work on projects and understand the material from lecture and lab. I am here to chat, explain things and support you so **ask questions and ask for help!**

Exercise 1.1

On a sheet of paper write down a simple set of drawing instructions.

You have 3 minutes.

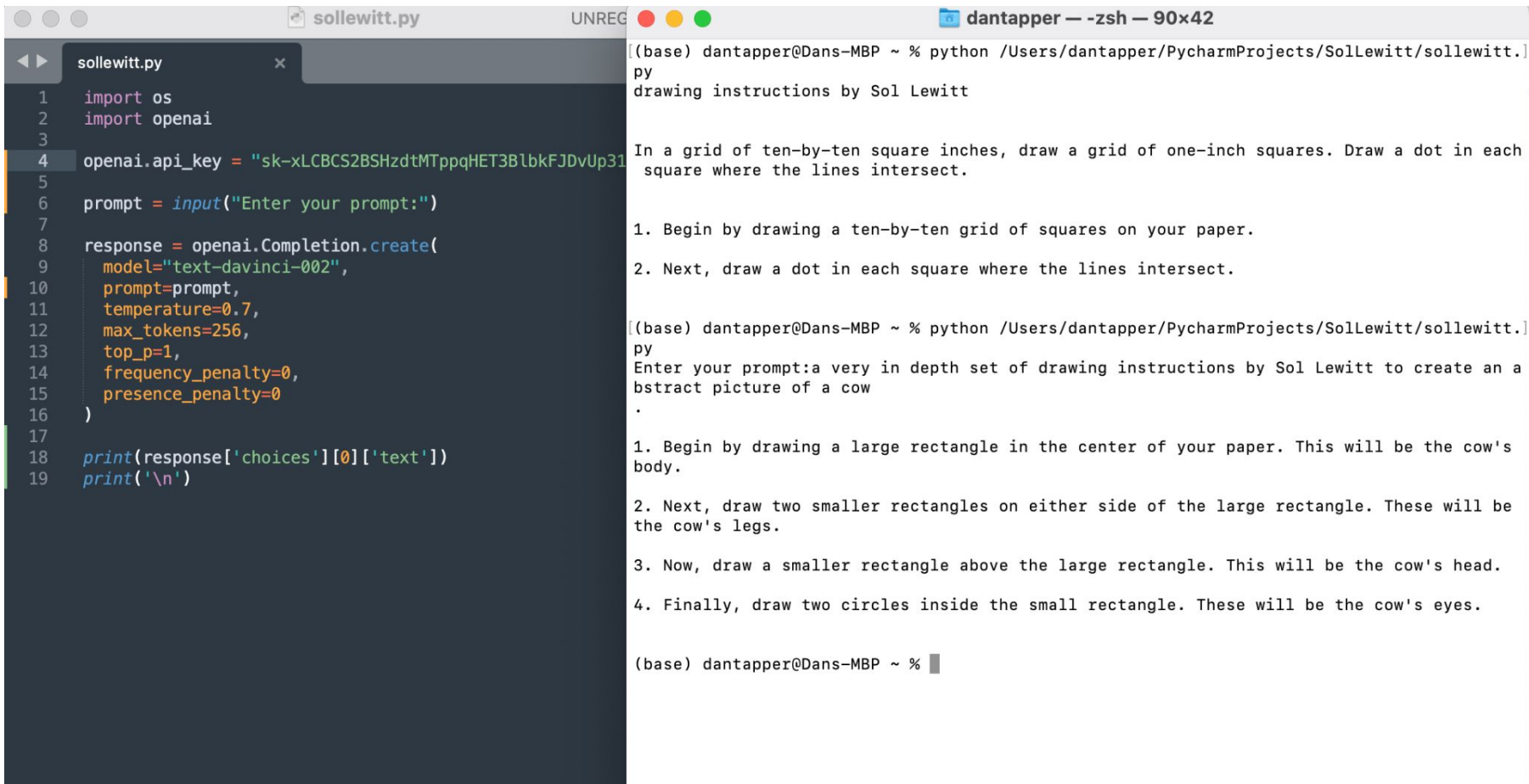
Examples from Sol Lewitt:

“Bands of lines 12 inches (30 cm) wide, in three directions (vertical, horizontal, diagonal right) intersecting.”

“A 6-inch (15 cm) grid covering each of the four black walls. White lines to points on the grids. twenty-four lines from the center, twelve lines from the midpoint of each of the sides, twelve lines from each corner. (The length of the lines and their placement are determined by the drafter.) ”

“On four black walls, white vertical parallel lines, and in the center of the walls, eight geometric figures (including cross, X) within which are white horizontal parallel lines. The vertical lines do not enter the figures.”

AI Sol Lewitt (generated using [GPT3](#))



The image shows a PyCharm IDE window with two panes. The left pane displays a Python script named `solle Witt.py`. The script imports `os` and `openai`, sets an API key, and uses `openai.Completion.create` to generate drawing instructions based on a user prompt. The right pane shows the terminal output of the script, which displays the generated drawing instructions for a cow.

```
solle Witt.py
1 import os
2 import openai
3
4 openai.api_key = "sk-xLCBCS2BShZdtMTppqHET3BlbkFJDvUp31
5
6 prompt = input("Enter your prompt:")
7
8 response = openai.Completion.create(
9     model="text-davinci-002",
10    prompt=prompt,
11    temperature=0.7,
12    max_tokens=256,
13    top_p=1,
14    frequency_penalty=0,
15    presence_penalty=0
16 )
17
18 print(response['choices'][0]['text'])
19 print('\n')
```

```
((base) dantapper@Dans-MBP ~ % python /Users/dantapper/PycharmProjects/SolLewitt/solle Witt.py
drawing instructions by Sol Lewitt

In a grid of ten-by-ten square inches, draw a grid of one-inch squares. Draw a dot in each
square where the lines intersect.

1. Begin by drawing a ten-by-ten grid of squares on your paper.
2. Next, draw a dot in each square where the lines intersect.

((base) dantapper@Dans-MBP ~ % python /Users/dantapper/PycharmProjects/SolLewitt/solle Witt.py
Enter your prompt:a very in depth set of drawing instructions by Sol Lewitt to create an a
bstract picture of a cow
.

1. Begin by drawing a large rectangle in the center of your paper. This will be the cow's
body.

2. Next, draw two smaller rectangles on either side of the large rectangle. These will be
the cow's legs.

3. Now, draw a smaller rectangle above the large rectangle. This will be the cow's head.

4. Finally, draw two circles inside the small rectangle. These will be the cow's eyes.

((base) dantapper@Dans-MBP ~ %
```

Exercise 1.2

Exchange your drawing instructions with the person nearest to you.

You will have 5 minutes to draw your best interpretation of their instructions.

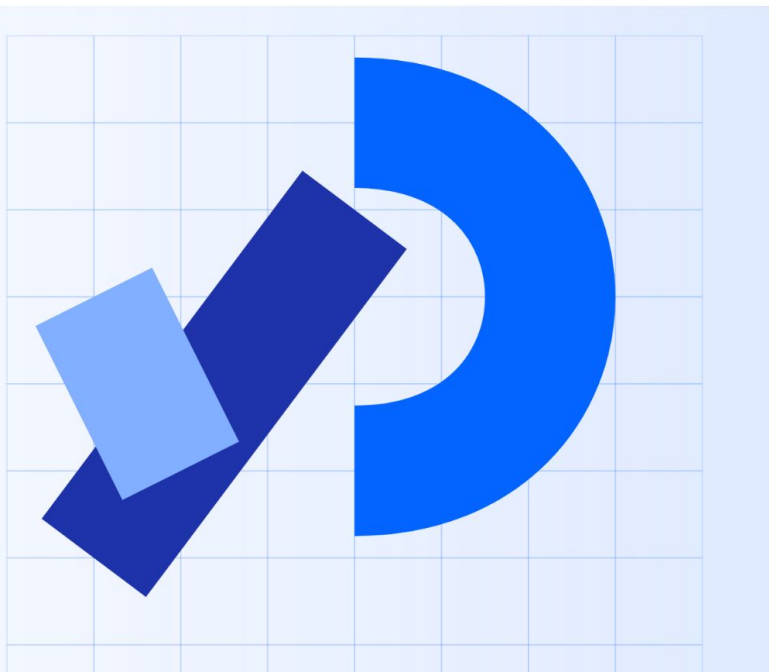
If you finish within 5 minutes take time to think about how you might use code to translate these rules.

Download Processing

[Download](#)[Documentation](#)[Learn](#)[Teach](#)[About](#)[Donate](#)

Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

[Download](#)[Reference](#)[Donate](#)

Processing Overview

Processing overview

P.0.0

Processing—an overview

This book uses the programming language Processing as a way of introducing the concept of generative design; its basic features and functions will be presented on the following pages. This chapter lays no claim to completeness, which would be far beyond the realm of the book. A very good and detailed introduction to programming with Processing can be found in the book *Processing: A Programming Handbook for Visual Designers and Artists* by Casey Reas and Ben Fry.

→ W.201
Processing book

The Processing project was initiated in the spring of 2001 by Ben Fry and Casey Reas and has continually grown and been developed since then with a small group of collaborators. The main goal of Processing is to provide visually oriented people with simple access to programming.

Processing programs are called “sketches.” Hence, Processing can be understood as an environment for the quick creation of digital sketches. The main folder where user-created programs are stored is called the “sketchbook” folder.

Processing is an open-source project; users can download it for free and use it for their own projects. Since Processing uses textual notation that consists of up to ninety-five percent Java syntax (with Java considered one of the main standards of the software industry), it is easy to transfer the code and examples to the majority of other textual development environments. Processing is cross-platform, which means the same source code can be used on all operating systems for which a Java platform exists (e.g., Mac OS, Windows, and Linux) and can also be integrated into websites.

→ W.202
Programming language Java

An ever-growing, vital, and supportive online Processing community exists that actively exchanges ideas on the Processing website. The website also contains online references of all language elements and an index of supplementary libraries.

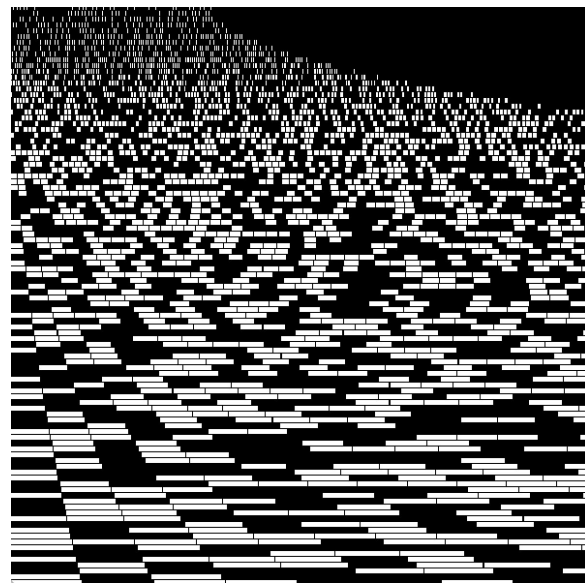
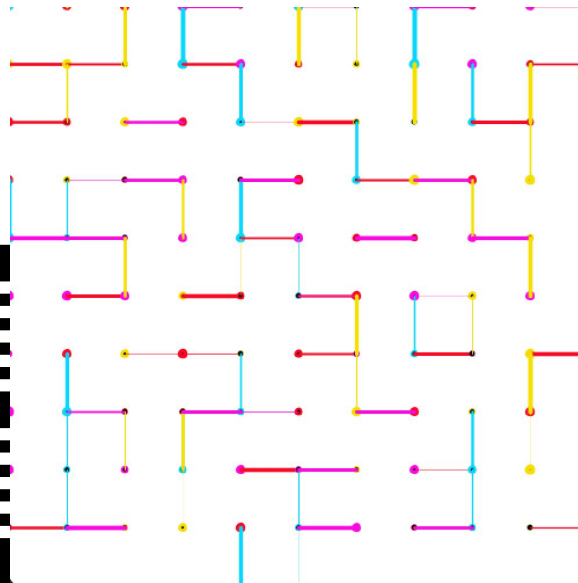
→ W.203
Processing Website

Several of these external libraries are used for the programs in this book. We also developed our own library for this book. The range of functions of the generative design library also includes the support of graphic tablets, the ability to export to Adobe’s ASE color palette format, and the transformation of Cartesian coordinates into polar coordinates. The complementary libraries found in the “libraries” folder must be installed for all the programs in this book to work.

→ www.generative-gestaltung.de
→ Generative Design Library

Palette exporting
→ Ch. P.1.2
Color palettes

Creative Grids



Homework

Make a creative sketch that is based around the grid structure we built in lab. Try not to spend longer than 45 minutes on the sketch, it's interesting to improvise with your immediate ideas.

If you have lots of ideas you can try making multiple versions.

Think about elements such as grid spacing, the sizing of objects, the position, color and orientation of objects.

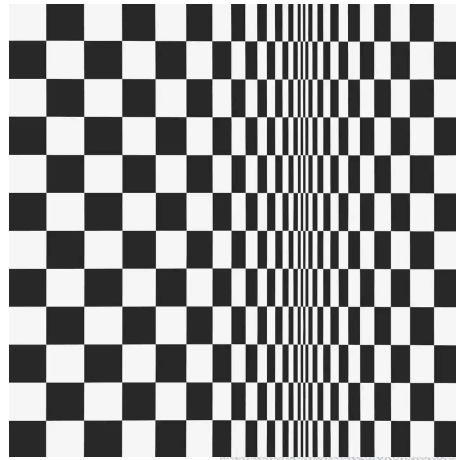
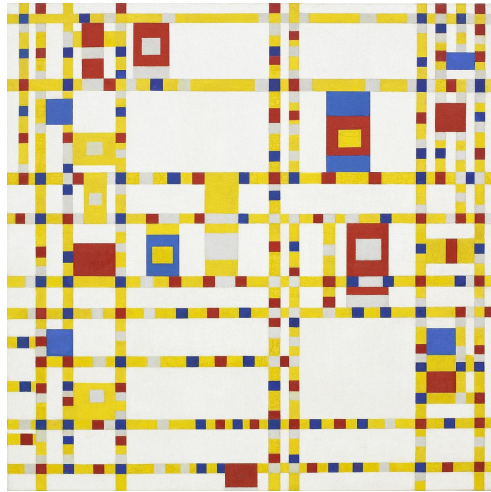
Experiment with if, else statements or [switch](#) (a more convenient way of selecting between multiple statements).

Be as creative as you like. If your final result looks radically different from a grid that's OK.

Comment your work as much as possible.

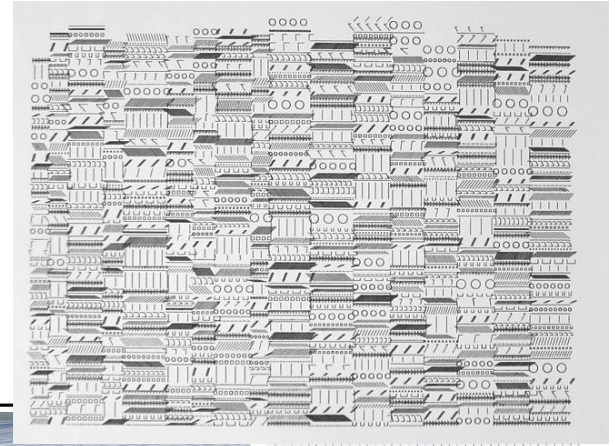
[Submit](#) a PDF linking to your uploaded sketch on Github.

Ideas



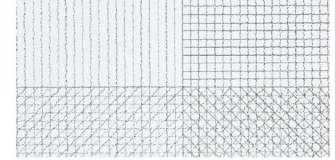
Wall Drawing 47

A wall divided into fifteen equal parts, each with a different line direction, and all combinations. June 1970 Black pen... [Read More](#)



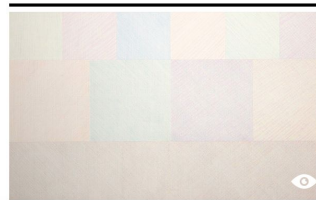
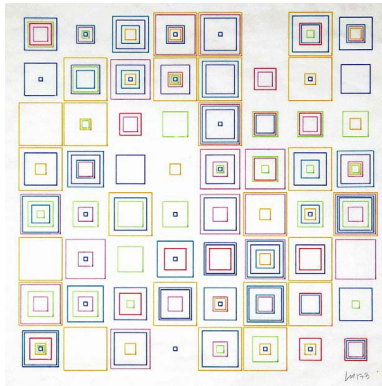
Wall Drawing 51

All architectural points connected by straight lines. June 1970 Blue snap lines LeWitt Collection, Chester, Connecticut ... [Read More](#)

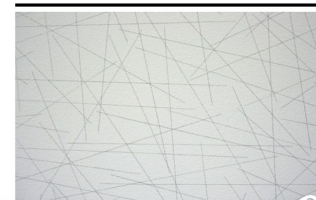


Wall Drawing 56

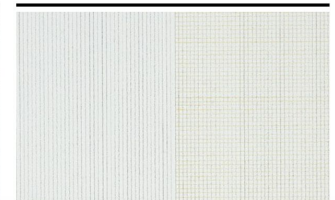
A square is divided horizontally and vertically into four equal parts, each with lines in four directions superimposed p... [Read More](#)



Wall Drawing 85



Wall Drawing 86



Wall Drawing 87