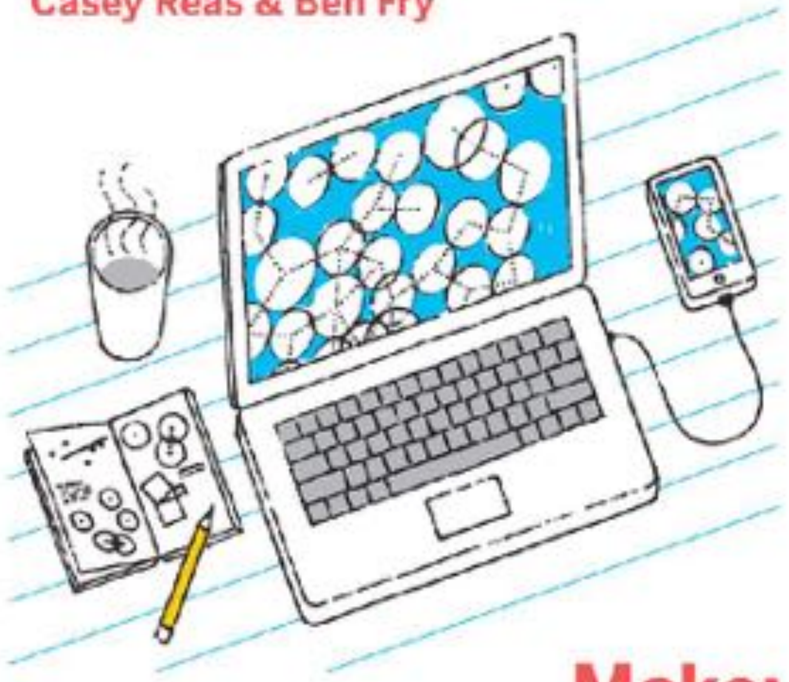
all code illustrations taken from the processing.org website +

Make: PROJECTS

A HANDS-ON INTRODUCTION TO MAKING INTERACTIVE GRAPHICS

Getting Started with Processing

Casey Reas & Ben Fry



O'REILLY

Make: makezine.com

Open Source/Graphics Programming

Make: PROJECTS

Getting Started with Processing

Learn computer programming the easy way with Processing, a simple language that lets you use code to create drawings, animation, and interactive graphics. Programming courses usually start with theory, but this book lets you jump right into creative and fun projects. It's ideal for anyone who wants to learn programming, and serves as a simple introduction to graphics for people who already have some programming skills.

Written by the founders of Processing, this book takes you through the learning process one step at a time to help you grasp core programming concepts. Join the thousands of hobbyists, students, and professionals who have discovered this free and educational community platform.

- Quickly learn programming basics, from variables to objects
- Understand the fundamentals of computer graphics
- Get acquainted with the Processing software environment
- Create interactive graphics with easy-to-follow projects
- Use the Arduino open source prototyping platform to control your Processing graphics

Casey Reas is a professor in the Department of Design Media Arts at UCLA and a graduate of the MIT Media Laboratory. Reas's software has been featured in numerous solo and group exhibitions in the U.S., Europe, and Asia.

Ben Fry, a designer, programmer, and author based in Cambridge, Massachusetts, received his doctoral degree from the MIT Media Laboratory. He worked with Casey Reas to develop Processing, which won a Golden Nica from the Prix Ars Electronica in 2005.

Make: makezine.com

O'REILLY

US $19.99 CAN $24.99
ISBN: 978-1-449-37980-3
51999
9 781449 379803

4/Variables

A variable stores a value in memory so that it can be used later in a program. The variable can be used many times within a single program, and the value is easily changed while the program is running.

The primary reason we use variables is to avoid repeating ourselves in the code. If you are typing the same number more than once, consider making it into a variable to make your code more general and easier to update.

```
int x;   // Declare x as an int variable
x = 12;  // Assign a value to x
```

```
int x = 12; // Declare x as an int variable and assign a value
```

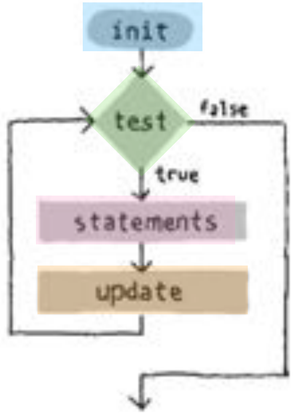
Inside the parentheses are three statements, separated by semicolons, that work together to control how many times the code inside the block is run. From left to right, these statements are referred to as the *initialization* (*init*), the *test*, and the *update*:

The *init* typically declares a new variable to use within the *for* loop and assigns a value. The variable name *i* is frequently used, but there's really nothing special about it. The *test* evaluates the value of this variable, and the *update* changes the variable's value. Figure 4-1 shows the order in which they run and how they control the code statements inside the block.

The *test* statement requires more explanation. It's always a *relational expression* that compares two values with a *relational operator*. In this example, the expression is "i < 400" and the operator is the < (less than) symbol. The most common relational operators are:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

The update generally increments the variable generated by the *init*, and *test*ed for in the relational expression




```
for (init; test; update) {
  statements
}
```

The *for* loop is different in many ways from the code we've written so far. Notice the braces, the { and } characters. The code between the braces is called a *block*. This is the code that will be repeated on each iteration of the *for* loop.

Figure 4-1. Flow diagram of a for loop.

Use a (single) *for loop* to set the *x* and *y positions* of several different shapes randomly

Use a (single) *for loop* to set the *x* and *y positions* of several different shapes randomly

Use an *embedded for loop* to set the *x* and *y position* of several different shapes in a grid

Example 5-2: The *setup()* Function

To complement the looping *draw()* function, Processing has a function called *setup()* that runs just once when the program starts:

```
void setup() {
  println("I'm starting");
}

void draw() {
  println("I'm running");
}
```

When this code is run, the following is written to the Console:

```
I'm starting
I'm running
I'm running
I'm running
...
```

Example 5-3: *setup()*, Meet *draw()*

The following example puts it all together:

```
int x = 280;
int y = -100;
int diameter = 380;

void setup() {
  size(480, 120);
  smooth();
  fill(102);
}

void draw() {
  background(204);
  ellipse(x, y, diameter, diameter);
}
```

Use a (single) *for loop* to set the *x* and *y positions* of several different shapes randomly

Use an *embedded for loop* to set the *x* and *y position* of several different shapes in a grid

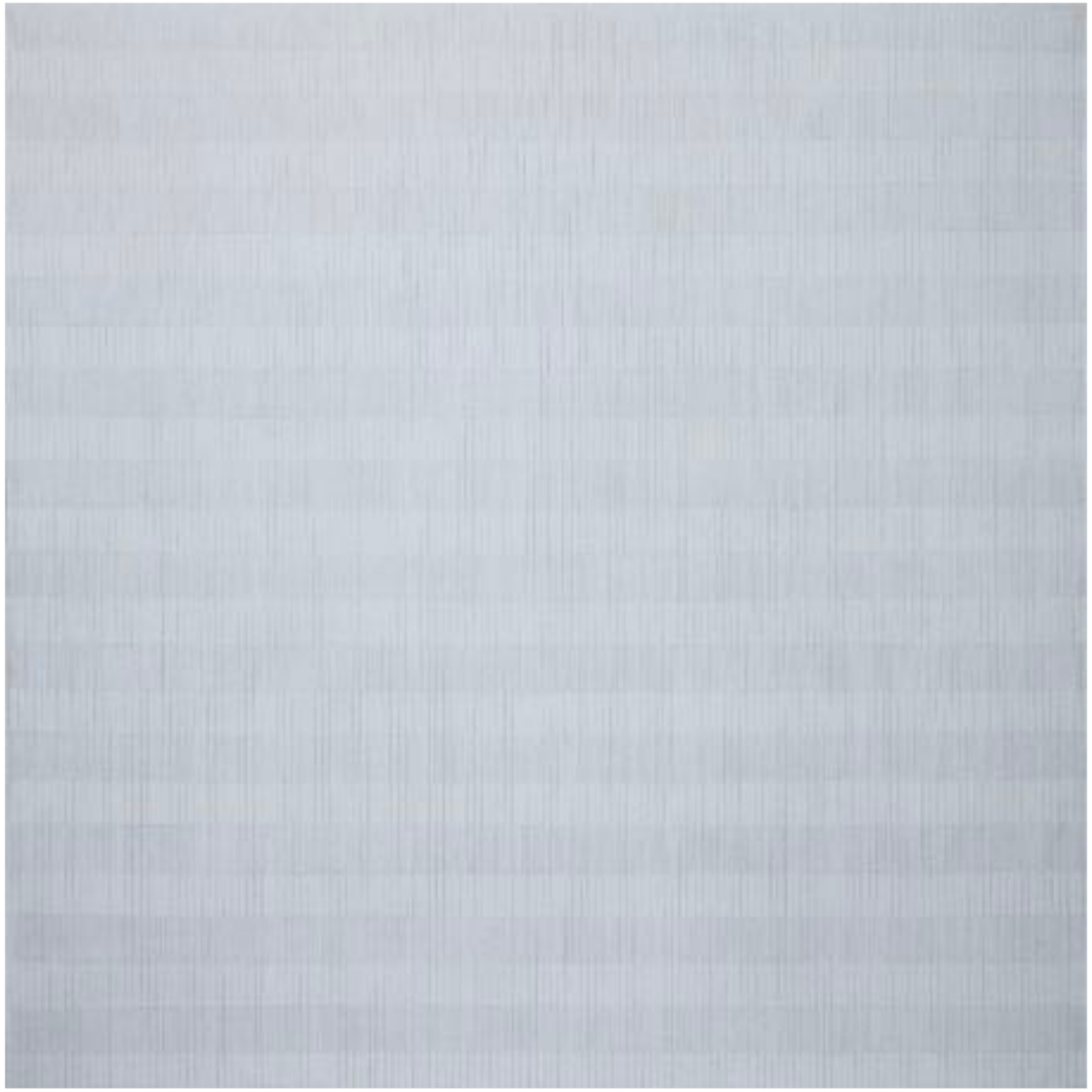
Test out your for loop inside of setup and draw. Explore *frameRate()*

Draw an ellipse that follows the mouse using *mouseX* + *mouseY*

Draw an ellipse that follows the mouse using *mouseX* + *mouseY*

Use *mouseX* + *mouseY* with and *embedded for loop* to create a responsive grid

Agnes Martin at SFMOMA
https://www.sfmoma.org/artist/Agnes_Martin



When I first made a grid I happened to be thinking of the innocence of trees and then this grid came into my mind and I thought it represented innocence, and I still do, and so I painted it and then I was satisfied. I thought, this is my vision.[14]

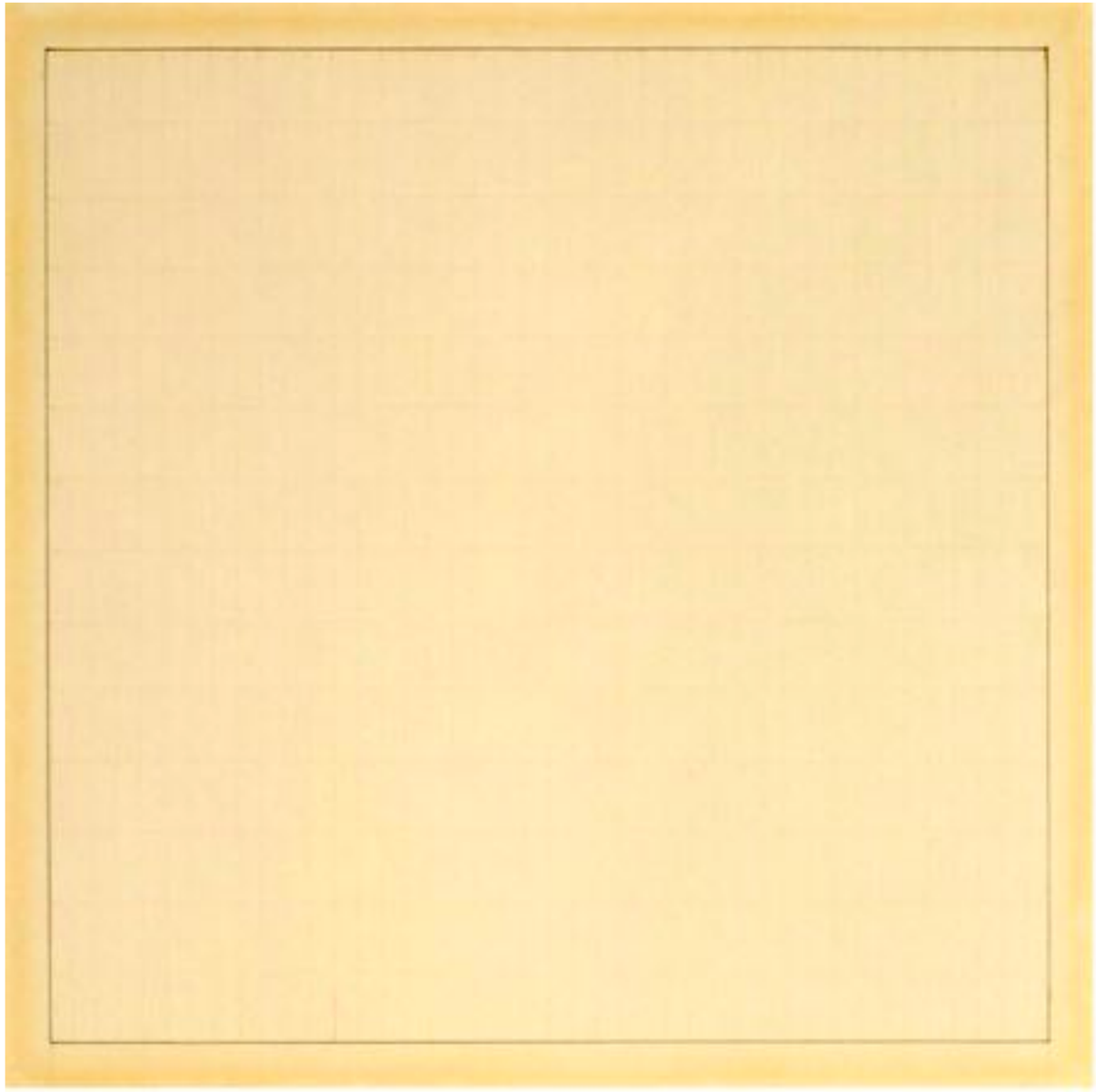
14. Agnes Martin, interview by Suzan Campbell, May 15, 1989, transcript in Archives of American Art, The Smithsonian Institution, Washington, D.C.

Agnes Martin, The Tree (1964) // Oil and graphite on canvas, 72 x 72 inches
https://www.guggenheim.org/arts-curriculum/topic/grids

Agnes Martin, I Love the Whole World, 1999
https://www.tate.org.uk/art/artworks/martin-i-love-the-whole-world-al00193

Left: White Flower II, 1985. Acrylic and graphite on canvas, 72 x 72 inches (182.9 x 182.9 cm).
Right: Fiesta, 1985. Acrylic and graphite on canvas, 72 1/16 x 72 1/8 inches (183 x 183.2 cm).
https://www.guggenheim.org/arts-curriculum/topic/series

Agnes Martin, Untitled #5, 1977
https://www.sfmoma.org/artwork/FC.788



Agnes Martin, Petal, 1964
https://www.sfmoma.org/artwork/69.72

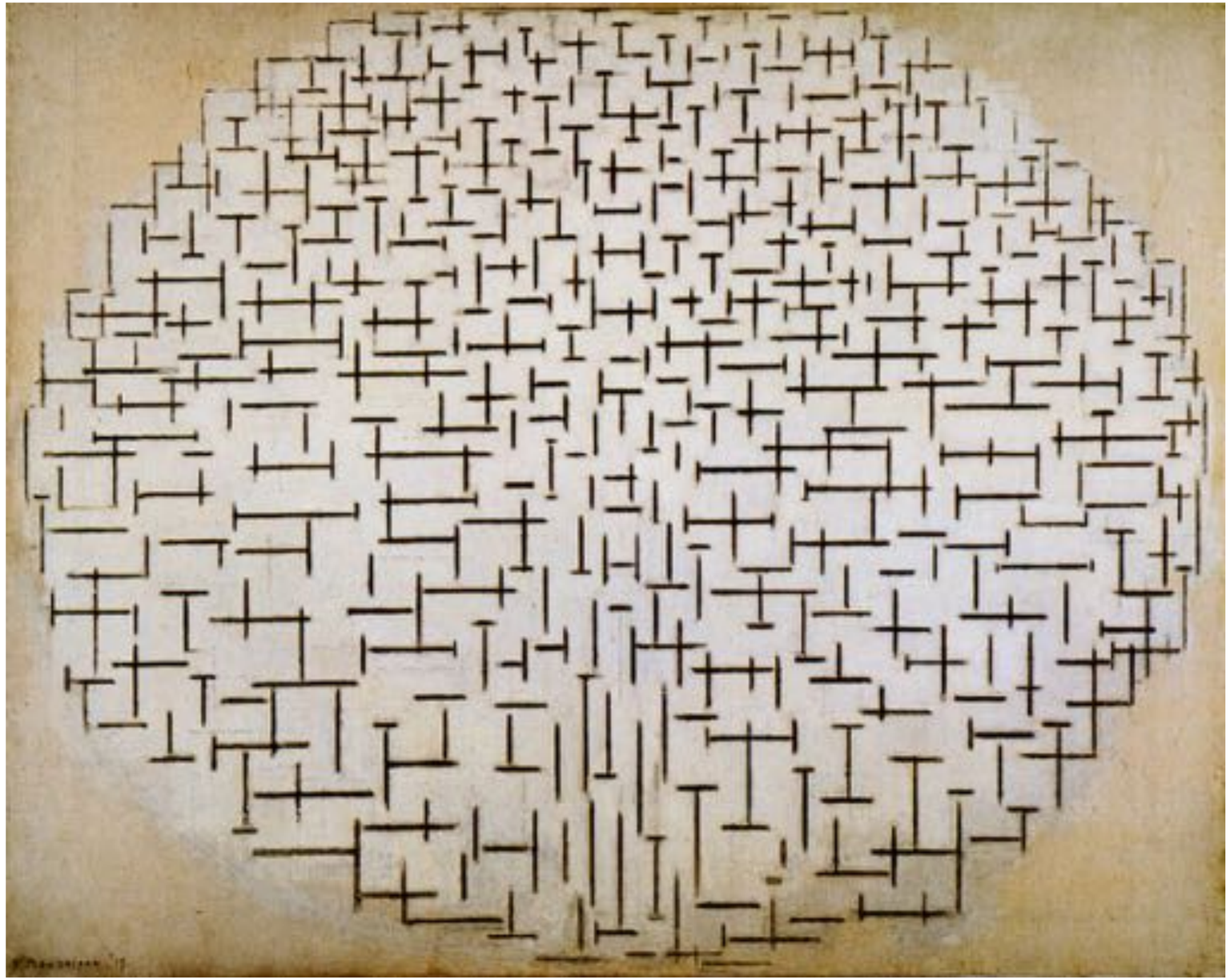
Homework 03 // Due 2018.09.24

Part I. Reproduce one of Agnes Martin's grid paintings/drawings
use at least one *for loop* in your work

Size: appropriate to the work and fits within 1280x720 pixels
(e.g. a square work would be 720x720 or 1280x1280 pixels)

Part II. Come up with a concept and generate your own grid inspired by the work of Agnes Martin
bonus: make your work interactive, responding to the mouse position

Size: appropriate to the work (see above size guidelines)



Piet Mondriaan, Pier and Ocean (Composition No. 10) (1915)
https://arthistoryproject.com/artists/piet-mondrian/pier-and-ocean-composition-no-10/