ndonmRseas

nsenRdamso

esdnRsaonm

nsnasemdRo

emnosandsR

Randomness

dasnnmseoR

anmdRonsse

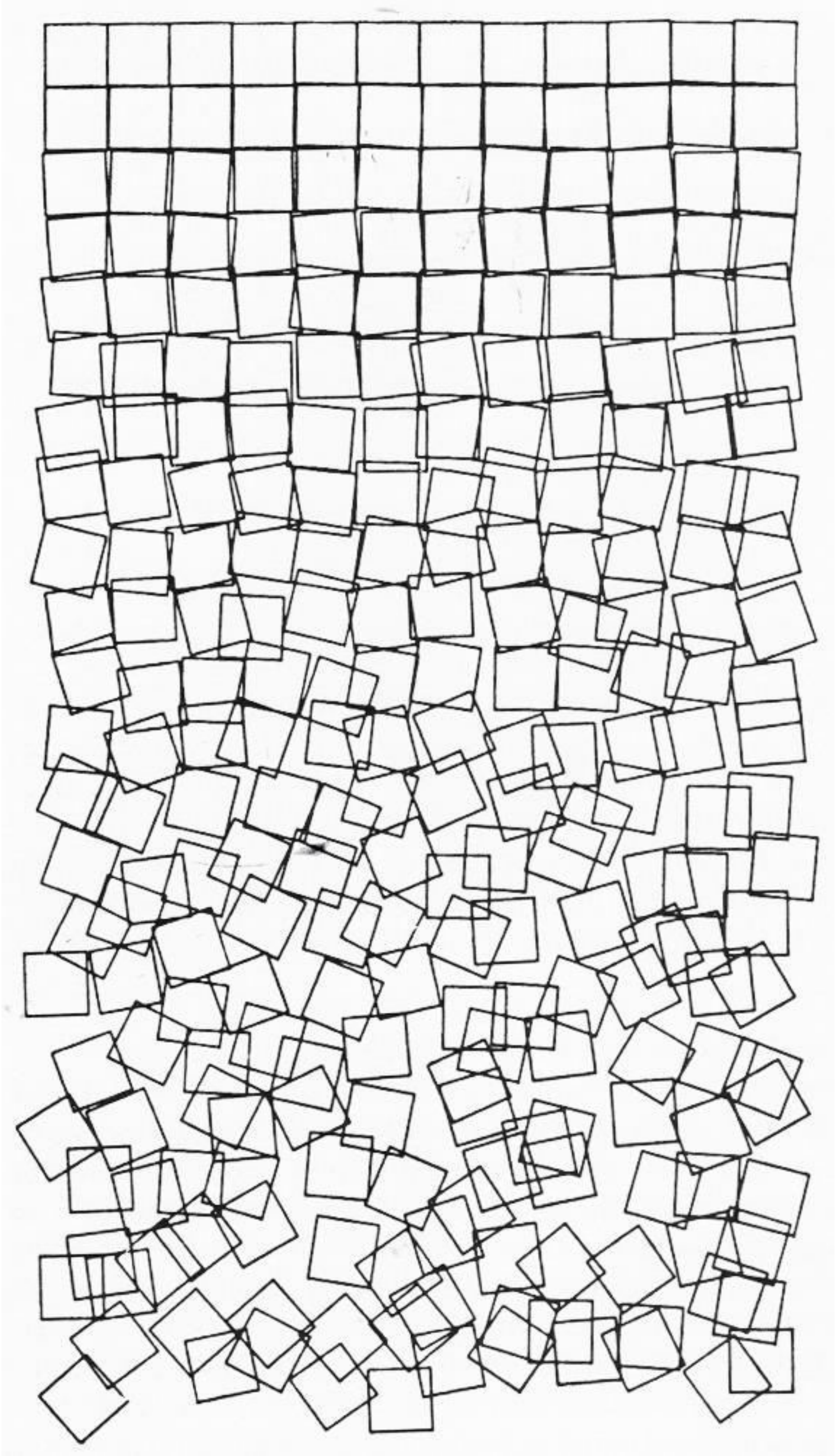
sRedmnaons

nndaeRsams

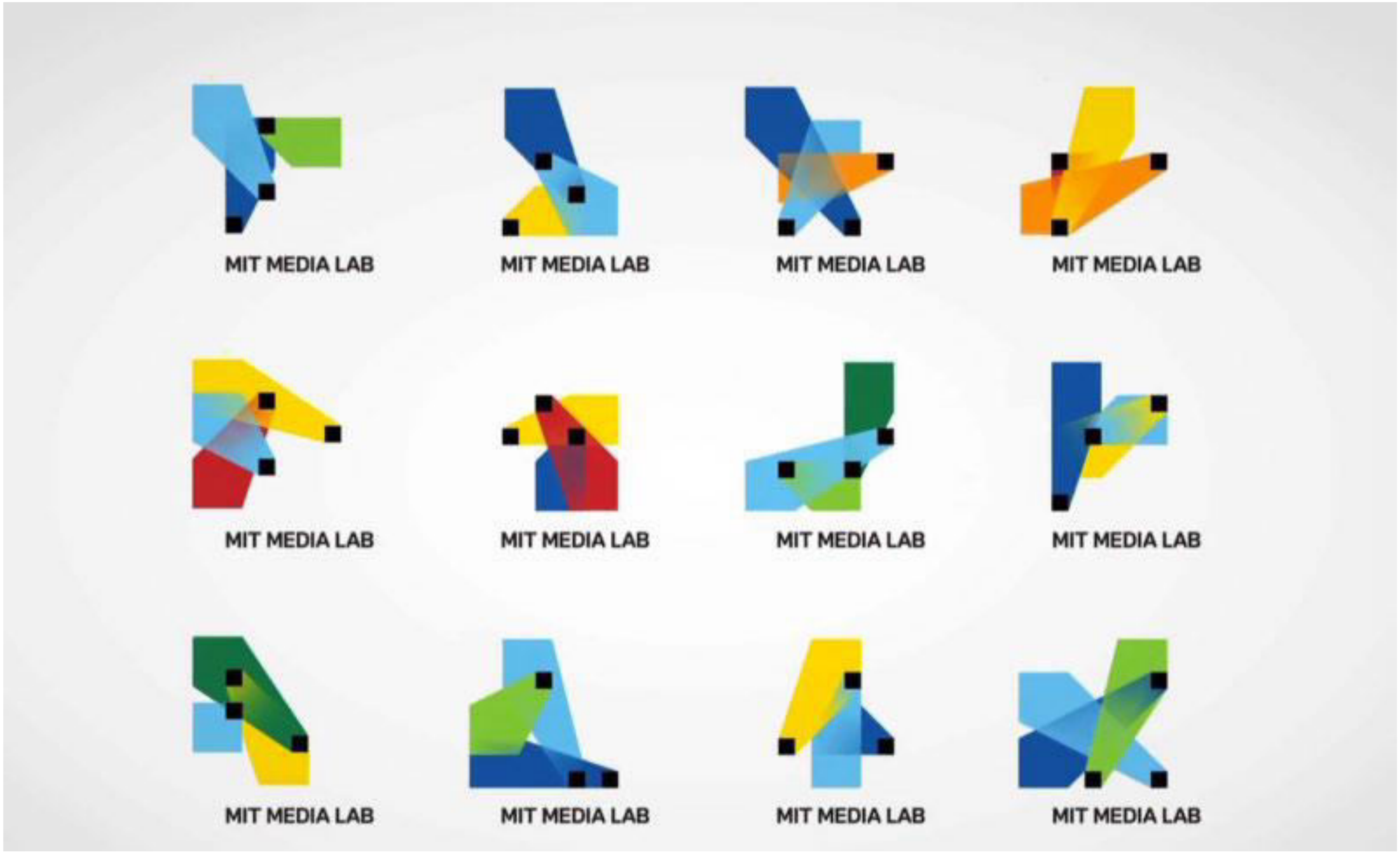
Module 09
CS 106 Winter 2019

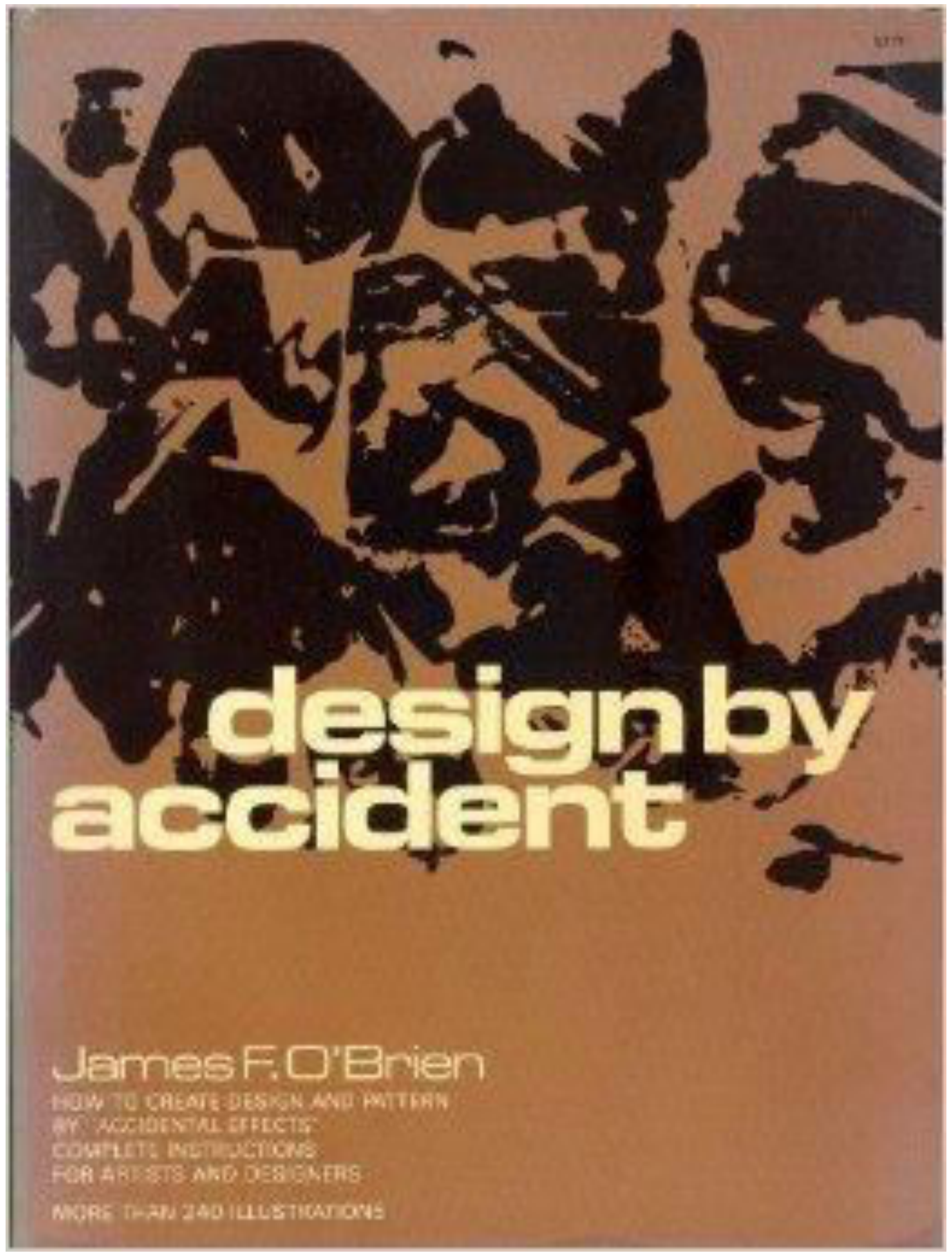
“Maybe the greatest novelty here is the ability of the computer not only to follow any complex rule of organization but also to introduce an exactly calculated dose of randomness.”

— E.H. Gombrich


Georg Nees, *Gravel Stones* (1971)

Generative Art



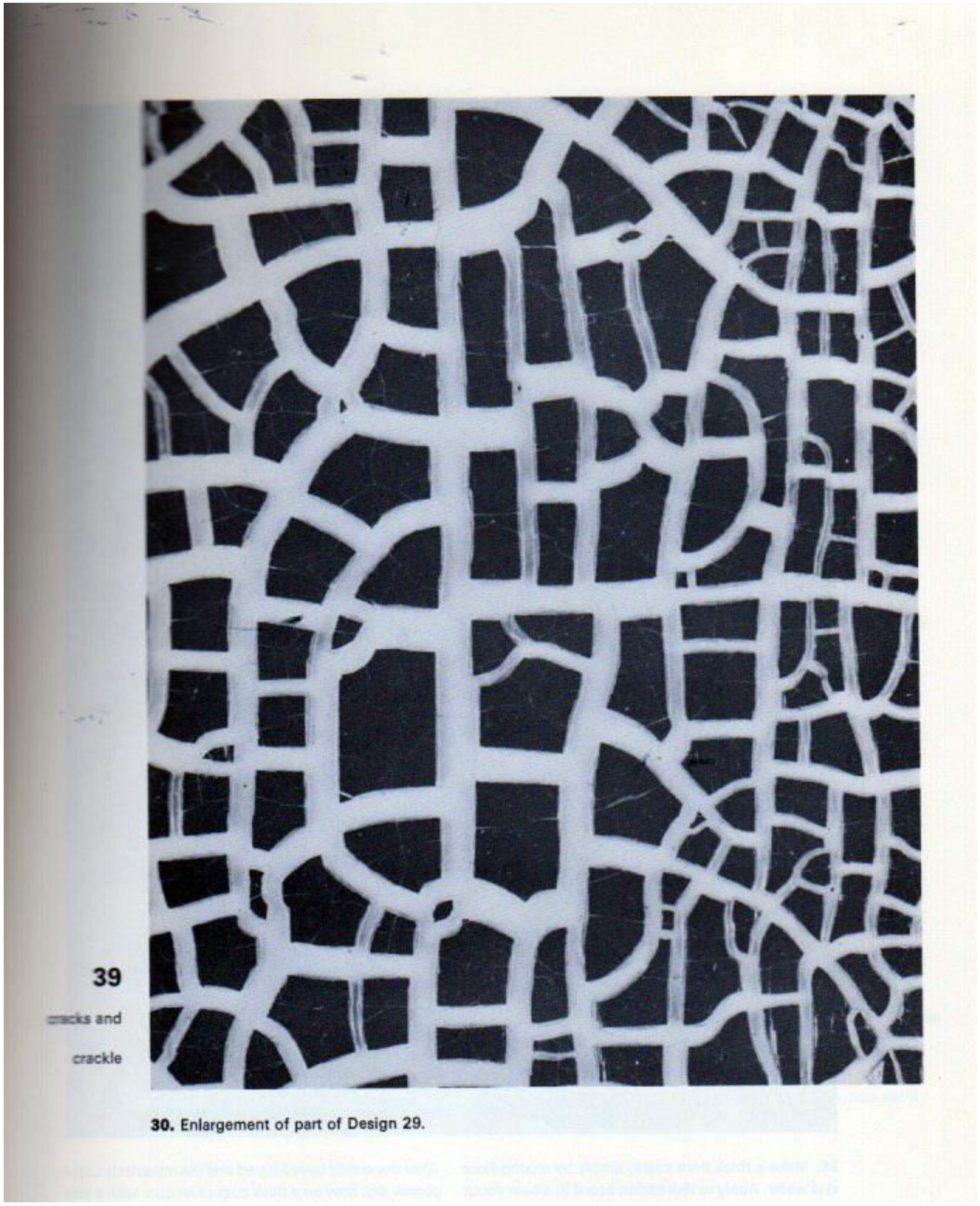


design by accident

James F. O'Brien

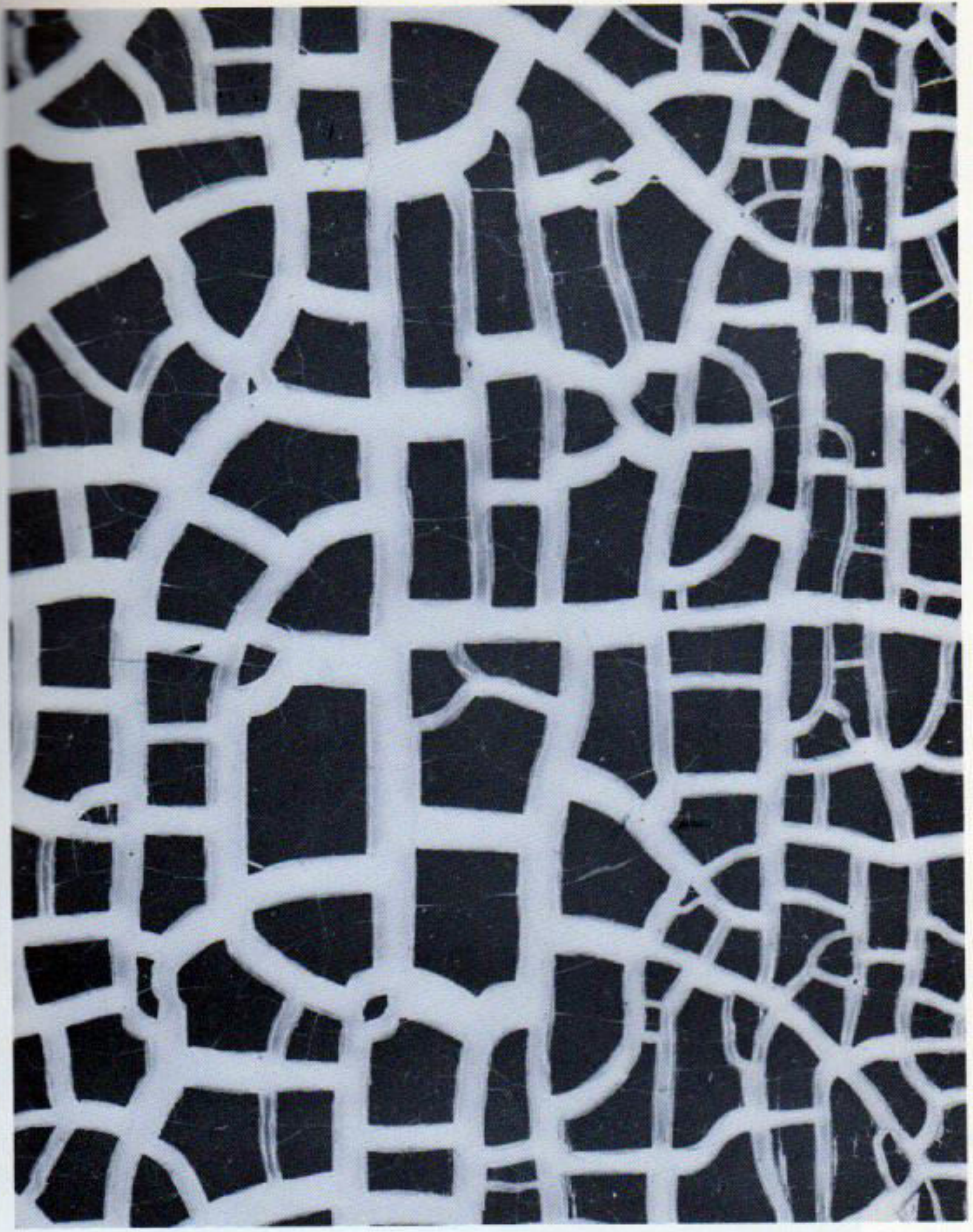
HOW TO CREATE DESIGN AND PATTERN BY "ACCIDENTAL EFFECTS" COMPLETE INSTRUCTIONS FOR ARTISTS AND DESIGNERS

MORE THAN 240 ILLUSTRATIONS



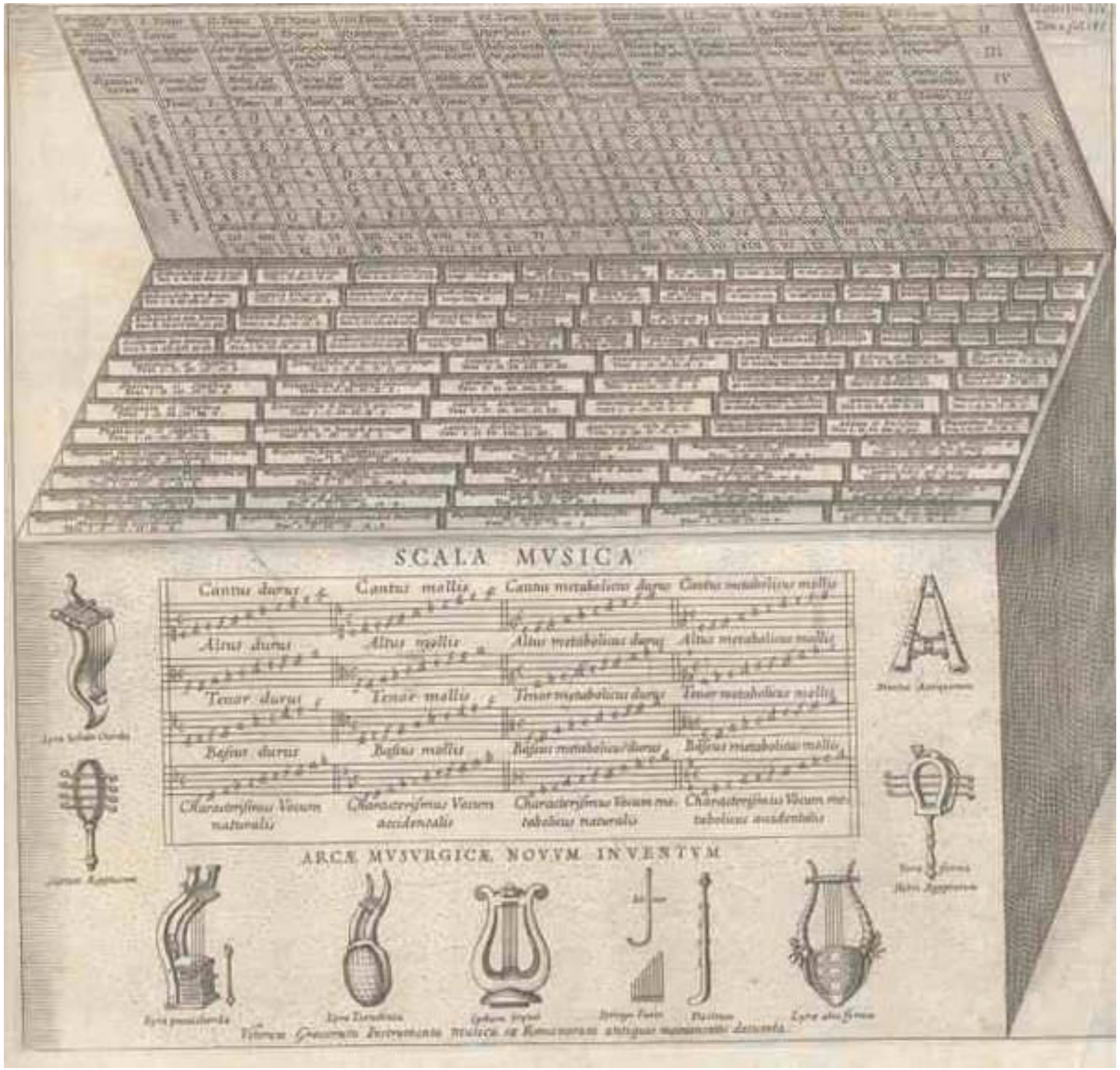
39

cracks and crackle



30. Enlargement of part of Design 29.

Athanasius Kircher (1602–1680)



Arca Musarithmica

Sol Lewitt (1928–2007)

Plate 1. Within a twenty inch square area, using a black, hard crayon, draw ten thousand freehand lines, of any length, at random.

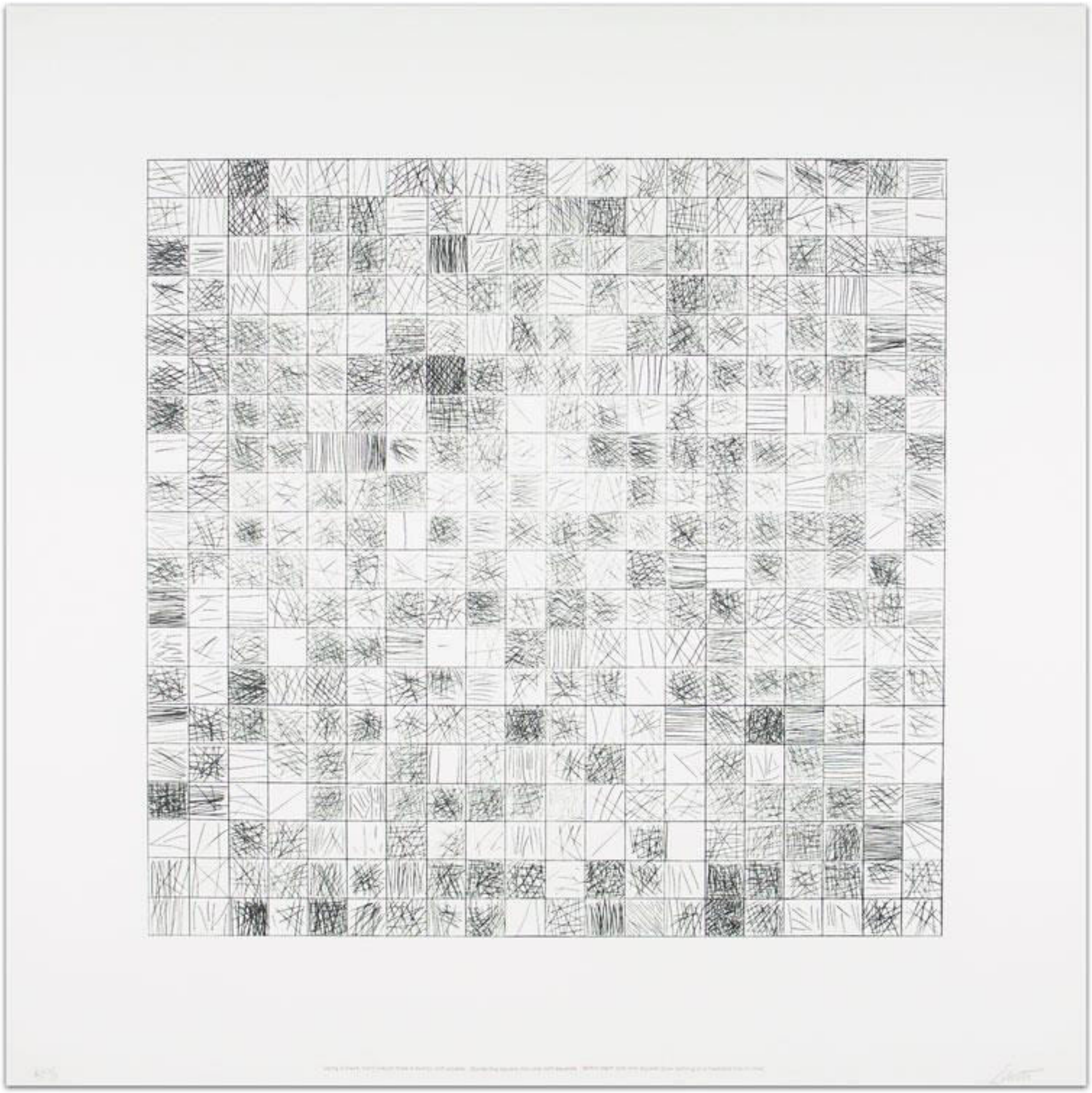
Plate 2. Within a twenty inch square area, using a black, hard crayon, draw ten thousand straight lines, of any length, at random.

Plate 3. Using a black, hard crayon draw a straight line of any length. From any point on that line draw another line perpendicular to the first. From any point on the second line draw another line perpendicular to that line. Repeat this procedure.

Plate 4. Using a black, hard crayon, draw four contiguous ten inch squares, forming one twenty inch square, divided into quarters. Within the first square (upper left) draw one line, one inch long. Within the second square (upper right) draw ten lines, each one inch long. Within the third square (lower left) draw one hundred lines, each one inch long. Within the fourth square (lower right) draw one thousand lines, each one inch long. All lines should be drawn at random, and straight.

Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.

Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.



Random

noise()

noiseDetail()

noiseSeed()

random()

randomGaussian()

randomSeed()

```
float random( float lo, float hi ) { ... }
```

Return a random number at least as big as lo but smaller than hi.

Get a different answer every time!

```
float random( float lo, float hi ) { ... }
```

Return a random number at least as big as lo but smaller than hi.

Get a different answer every time!

```
float random( float hi )
{
  return random( 0, hi );
}
```

Random integers

`int( random( N ) )`

Choose a random integer from the set 0, 1, …

Random integers

```
int( random( N ) )
```

Choose a random integer from the set 0, 1, … **N-1**

(The int() function always rounds down)

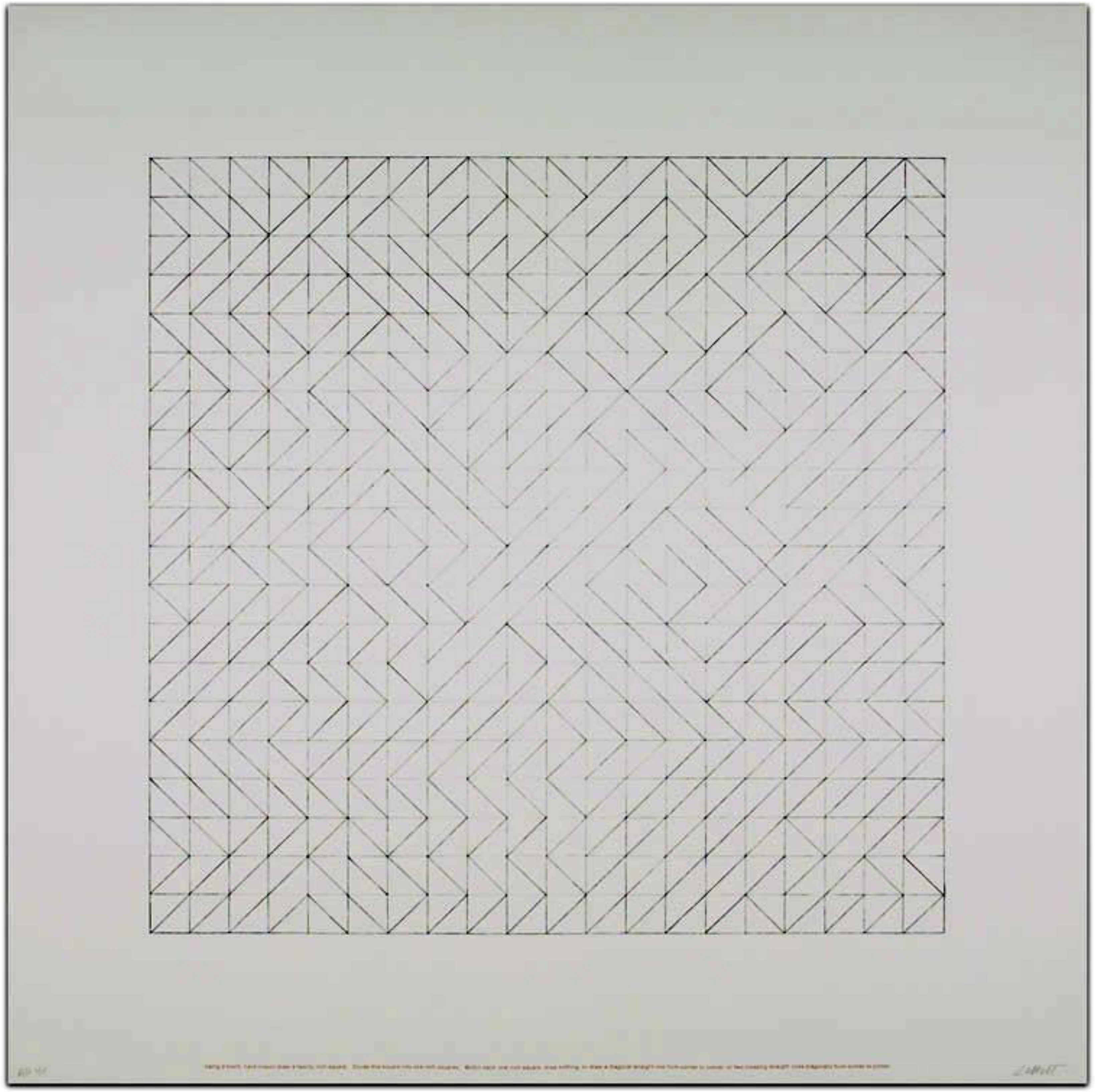
Flipping a coin

Write a function that simulates flipping a fair coin.

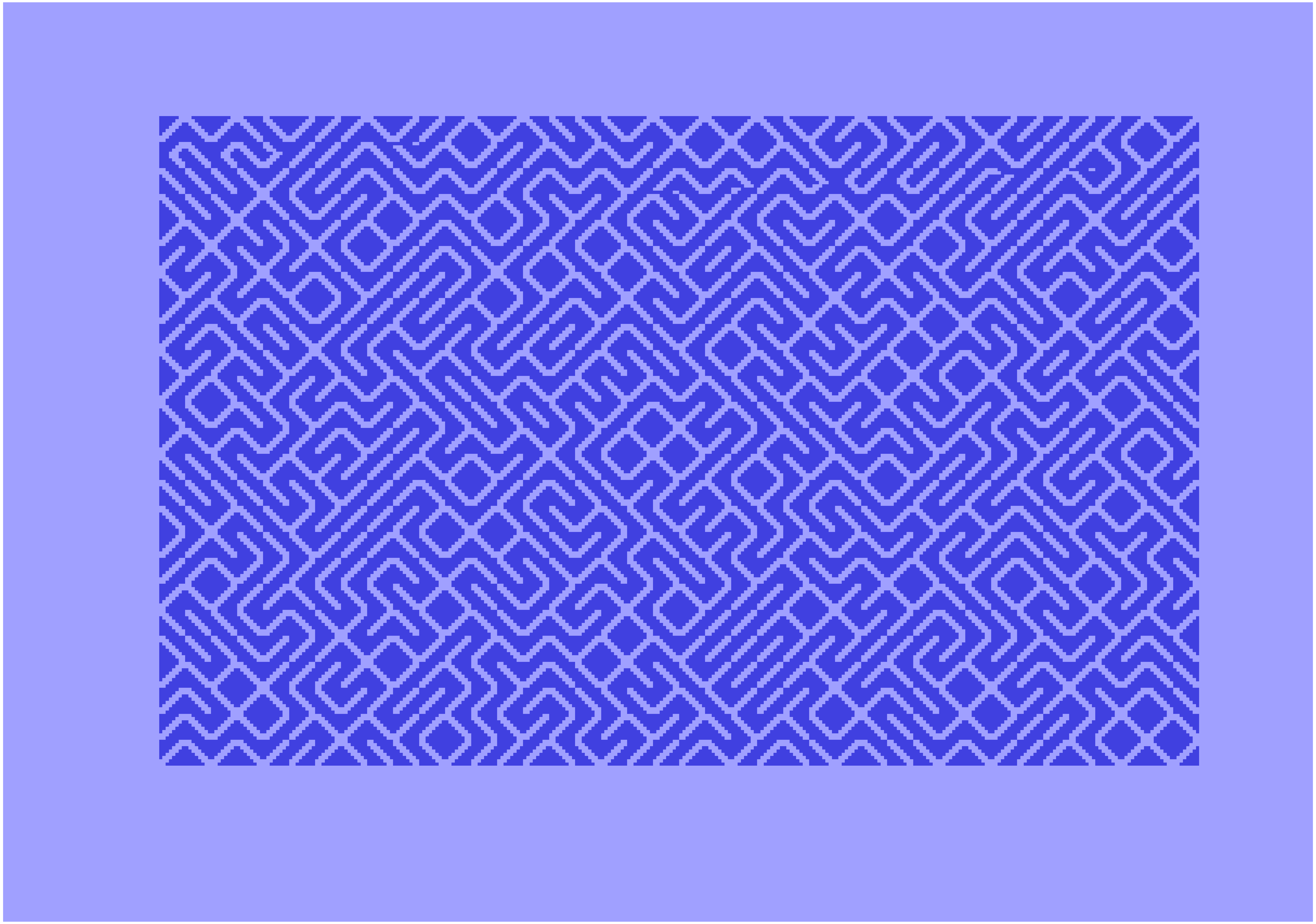
Flipping a biased coin

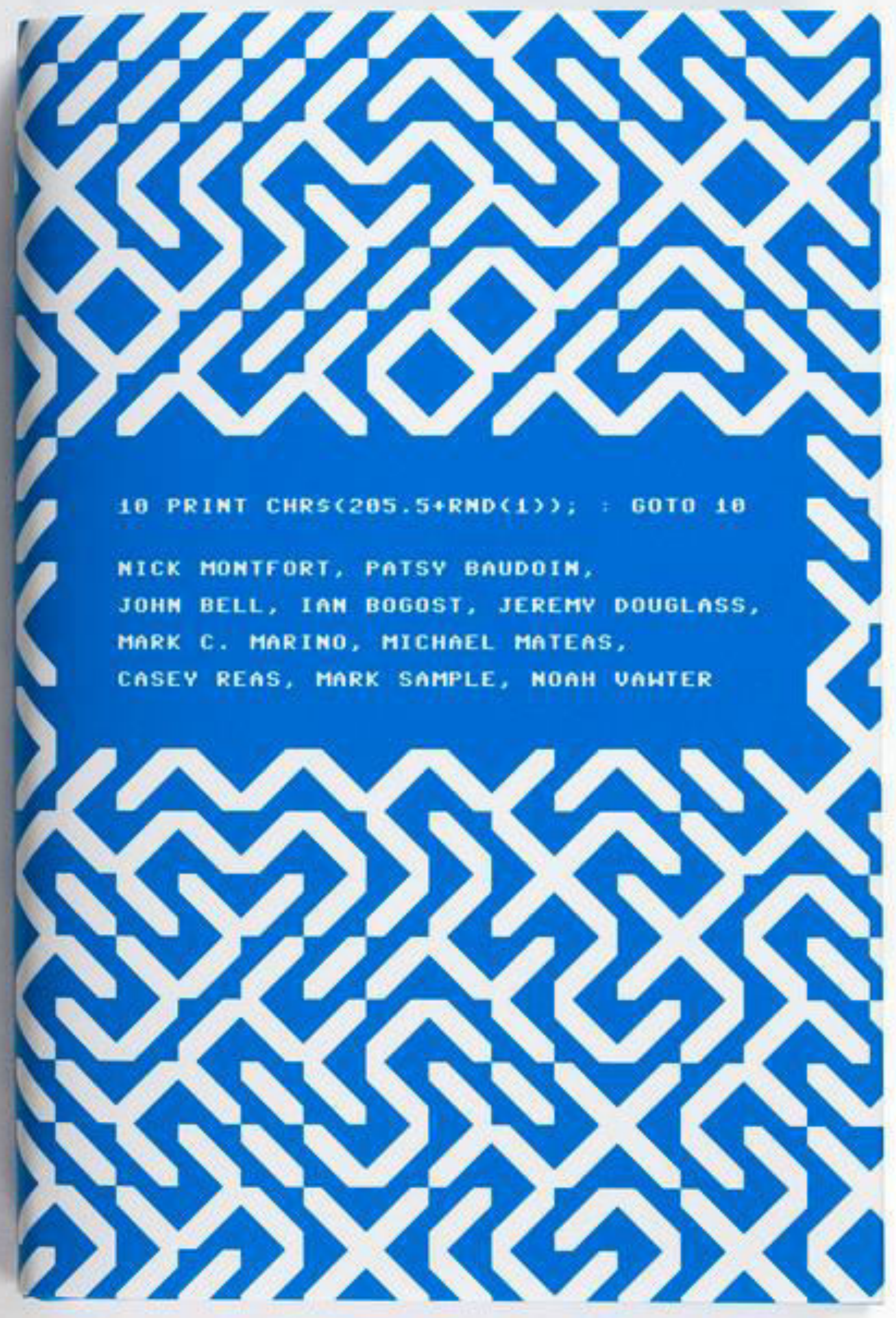
What if we wanted to get heads 75% of the time and tails 25% of the time?

Plate 6. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing, or draw a diagonal straight line from corner to corner, or two crossing straight lines diagonally from corner to corner.



```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```




10 PRINT CHR$(205.5+RND(1)); : GOTO 10
NICK MONTFORT, PATSY BAUDOIN,
JOHN BELL, IAN BOGOST, JEREMY DOUGLASS,
MARK C. MARINO, MICHAEL MATEAS,
CASEY REAS, MARK SAMPLE, NOAH VAWTER

10print.org

3.141592653589793238462643383279502884197169399375105820974944
59230781640628620899862803482534211706798214808651328230664709
38446095505822317253594081284811174502841027019385211055596446
22948954930381964428810975665933446128475648233786783165271201
90914564856692346034861045432664821339360726024914127372458700
66063155881748815209209628292540917153643678925903600113305305
48820466521384146951941511609433057270365759591953092186117381
93261179310511854807446237996274956735188575272489122793818301
19491298336733624406566430860213949463952247371907021798609437
02770539217176293176752384674818467669405132000568127145263560
82778577134275778960917363717872146844090122495343014654958537
10507922796892589235420199561121290219608640344181598136297747
71309960518707211349999998372978049951059731732816096318595024
45945534690830264252230825334468503526193118817101000313783875
28865875332083814206171776691473035982534904287554687311595628
63882353787593751957781857780532171226806613001927876611195909
21642019893809525720106548586327886593615338182796823030195203
53018529689957736225994138912497217752834791315155748572424541
50695950829533116861727855889075098381754637464939319

Most random number generators are like the digits of π: completely deterministic, but *hard to predict.*

These are called **Pseudorandom Number Generators** (PRNGs).

```
void randomSeed( int seed ) { ... }
```

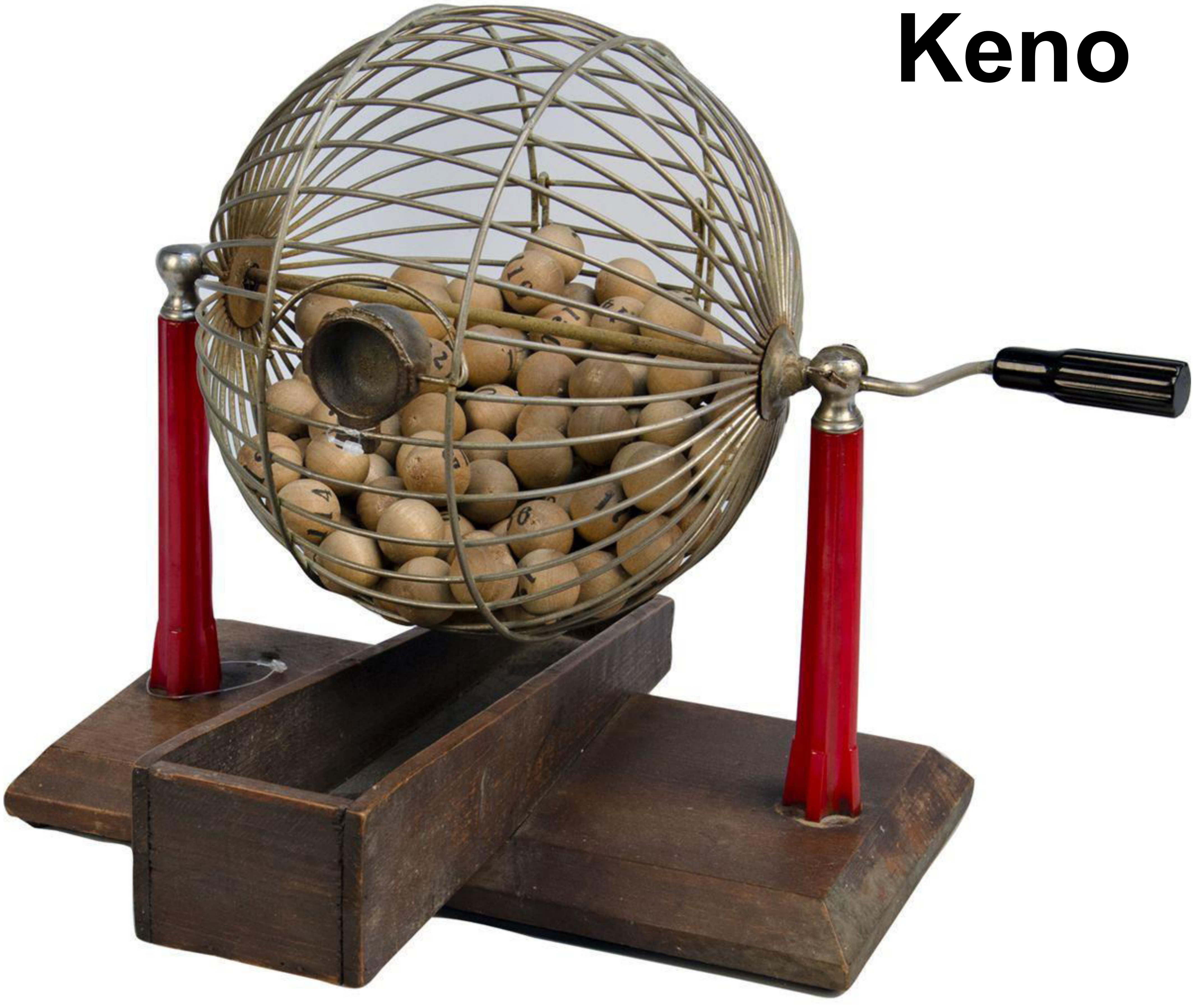
Reset the internal state of Processing’s PRNG based on the passed-in seed. A given seed will always produce the same sequence of answers to a given sequence of calls to random().

Pseudorandom number generators are a double-edged sword.

The good: we can always “replay” a sequence of pseudorandom numbers.

The bad: pseudorandom numbers ***are not actually random.***

Keno



Round №
106280
0:29
Jackpot
0.00
KENO2 InBet
54 45 37 8 20 14 12 9. 3 39 74 29 46 77 48 80 68 70 21 32
54

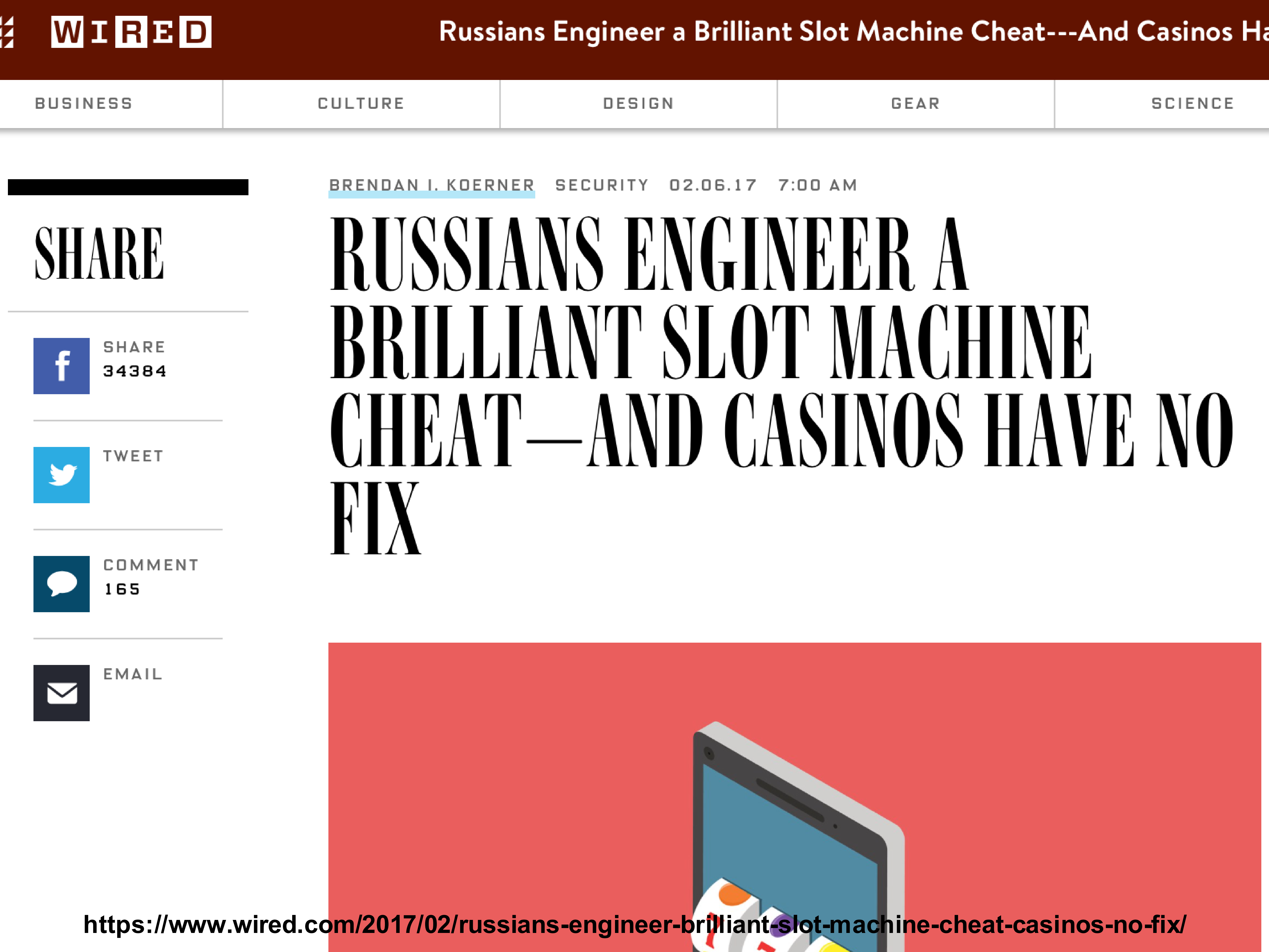
In April 1994, Daniel Corriveau won $620,000 CAD playing keno. He picked 19 of the 20 winning numbers three times in a row. Corriveau claims he used a computer to discern a pattern in the sequence of numbers, based on chaos theory. However, it was later found that the sequence was easy to predict because the casino was using an inadequate electronic pseudorandom number generator. In fact, the keno machine was reset every morning with the same seed number, resulting in the same sequence of numbers being generated. Corriveau received his winnings after investigators cleared him of any wrongdoing.

en.wikipedia.org/wiki/Montreal_Casino

Partial solutions:

1. Set an initial seed based on the current time.

3. Generate random numbers continuously, not just when needed.



WIRED Russians Engineer a Brilliant Slot Machine Cheat---And Casinos Ha

BUSINESS CULTURE DESIGN GEAR SCIENCE

BRENDAN I. KOERNER SECURITY 02.06.17 7:00 AM

RUSSIANS ENGINEER A BRILLIANT SLOT MACHINE CHEAT—AND CASINOS HAVE NO FIX

SHARE

SHARE 34384

TWEET

COMMENT 165

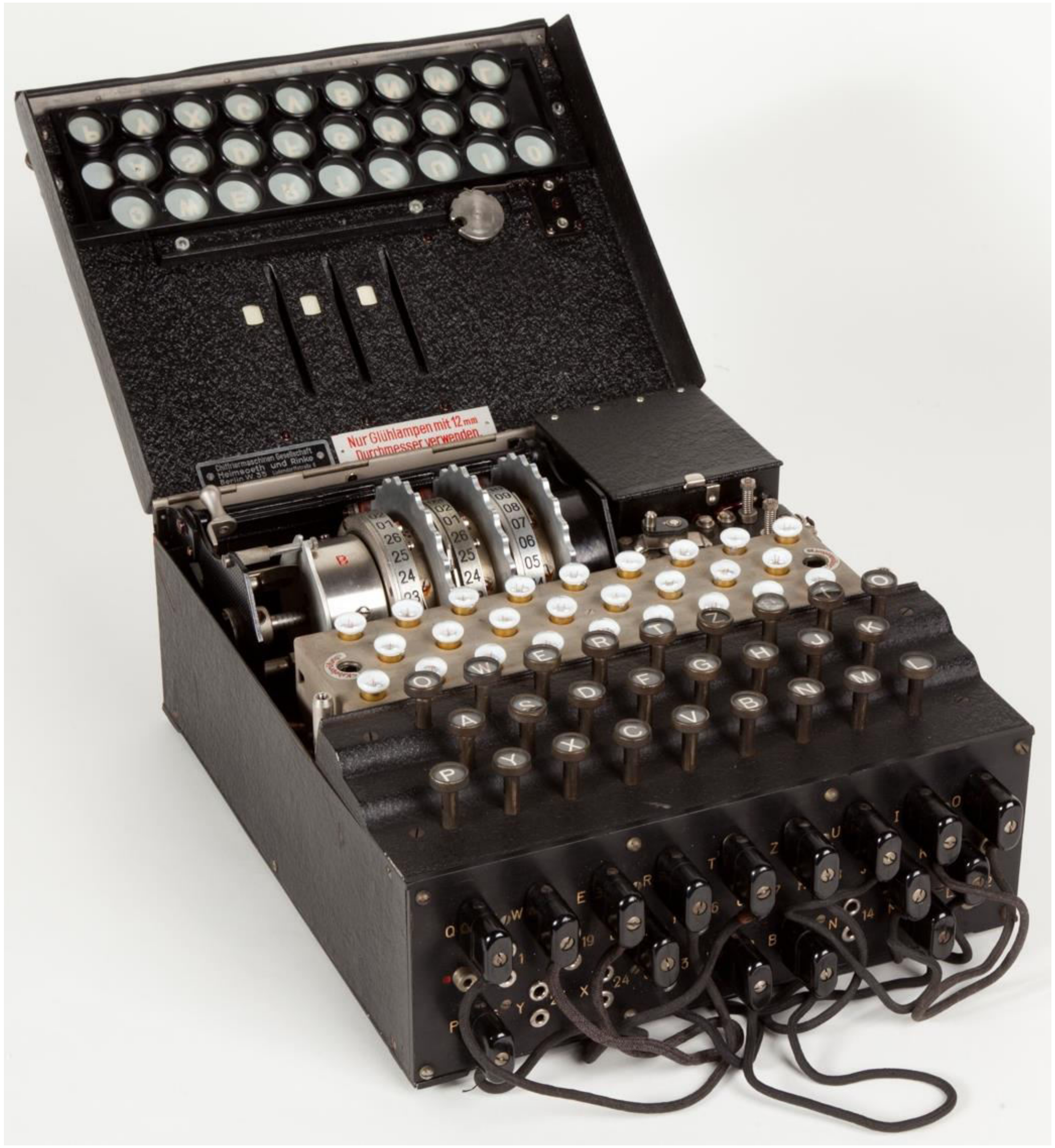
EMAIL

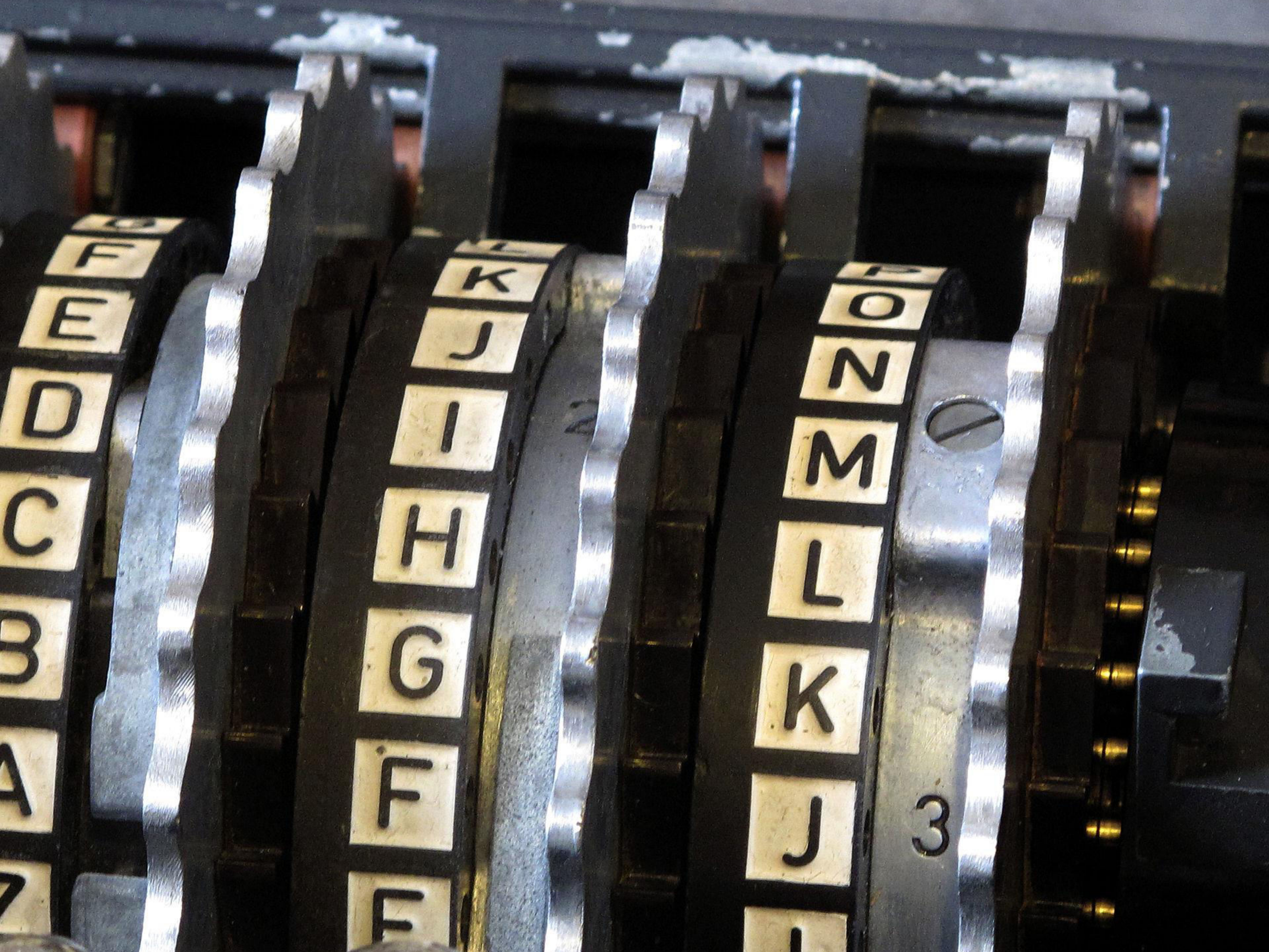
https://www.wired.com/2017/02/russians-engineer-brilliant-slot-machine-cheat-casinos-no-fix/

Modern cryptographic protocols often require frequent generation of random quantities. Cryptographic attacks that subvert, or exploit weaknesses in, this process are known as **random number generator attacks**.

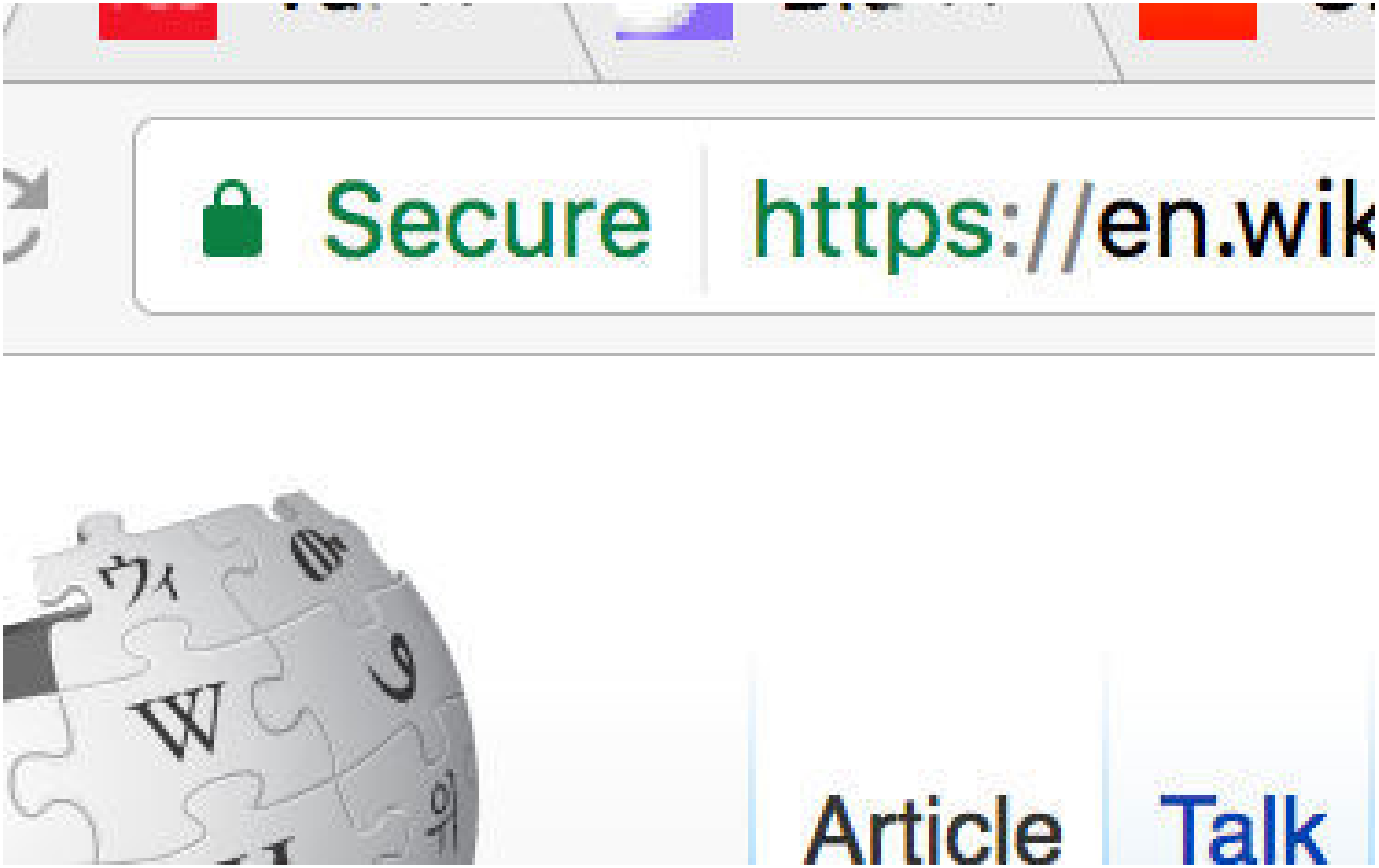
https://en.wikipedia.org/wiki/Random_number_generator_attack

Enigma Machine


Nur Glühlampen mit 12 mm Durchmesser verwenden
Chiffriermaschinen Gesellschaft Heimsoeth und Rinke Berlin W 35


F
E
D
C
B
A
K
J
I
H
G
F
O
N
M
L
K
J
2
3

Transport Layer Security (TLS/SSL)



Goals

- Understand how to use random() to generate unpredictable behaviour in Processing sketches.
- Understand how to use randomSeed() to control the generation of pseudorandom numbers.
- Understand the difference between random numbers and pseudorandom numbers.