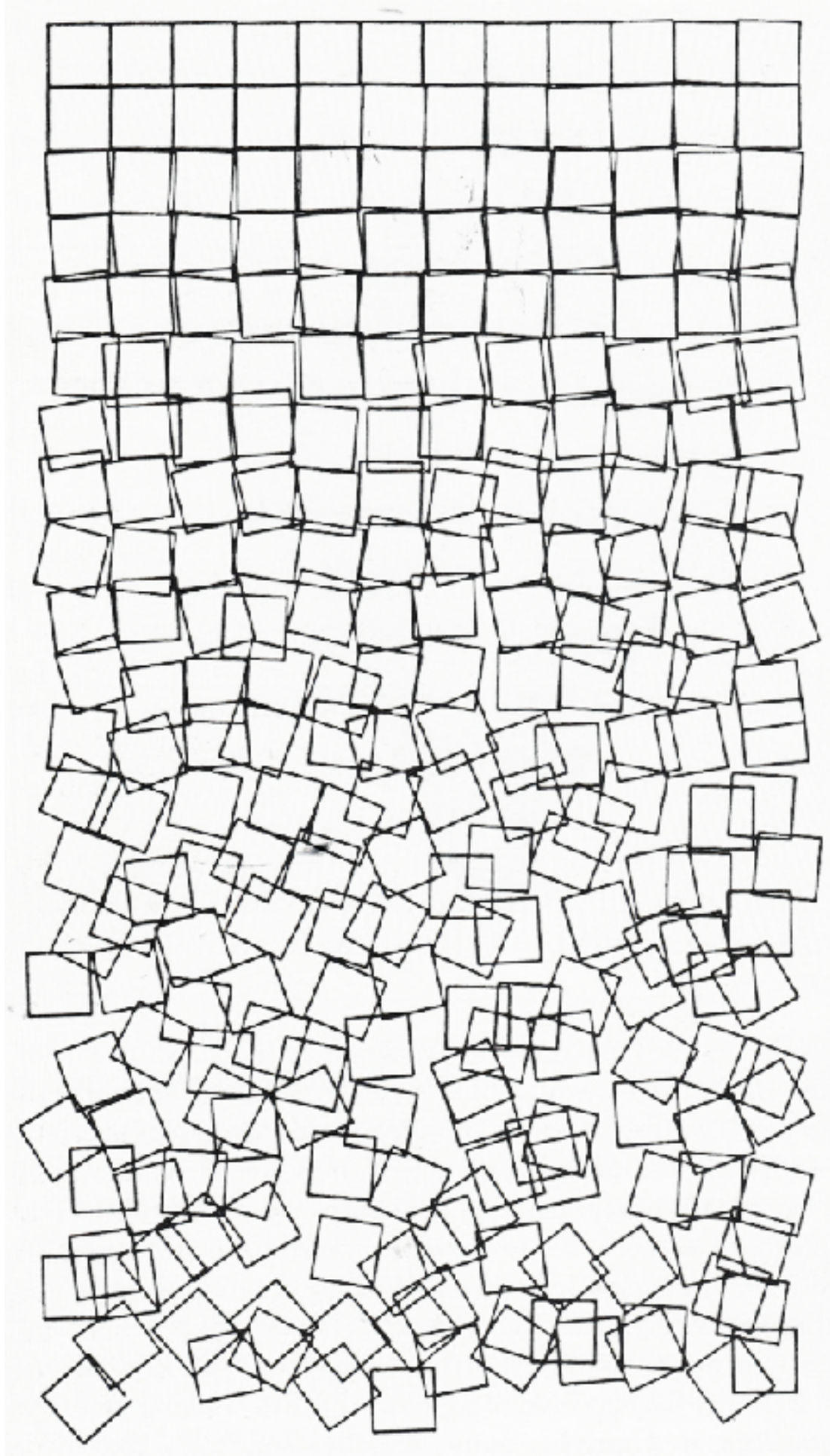
Module 08

Randomness and noise.

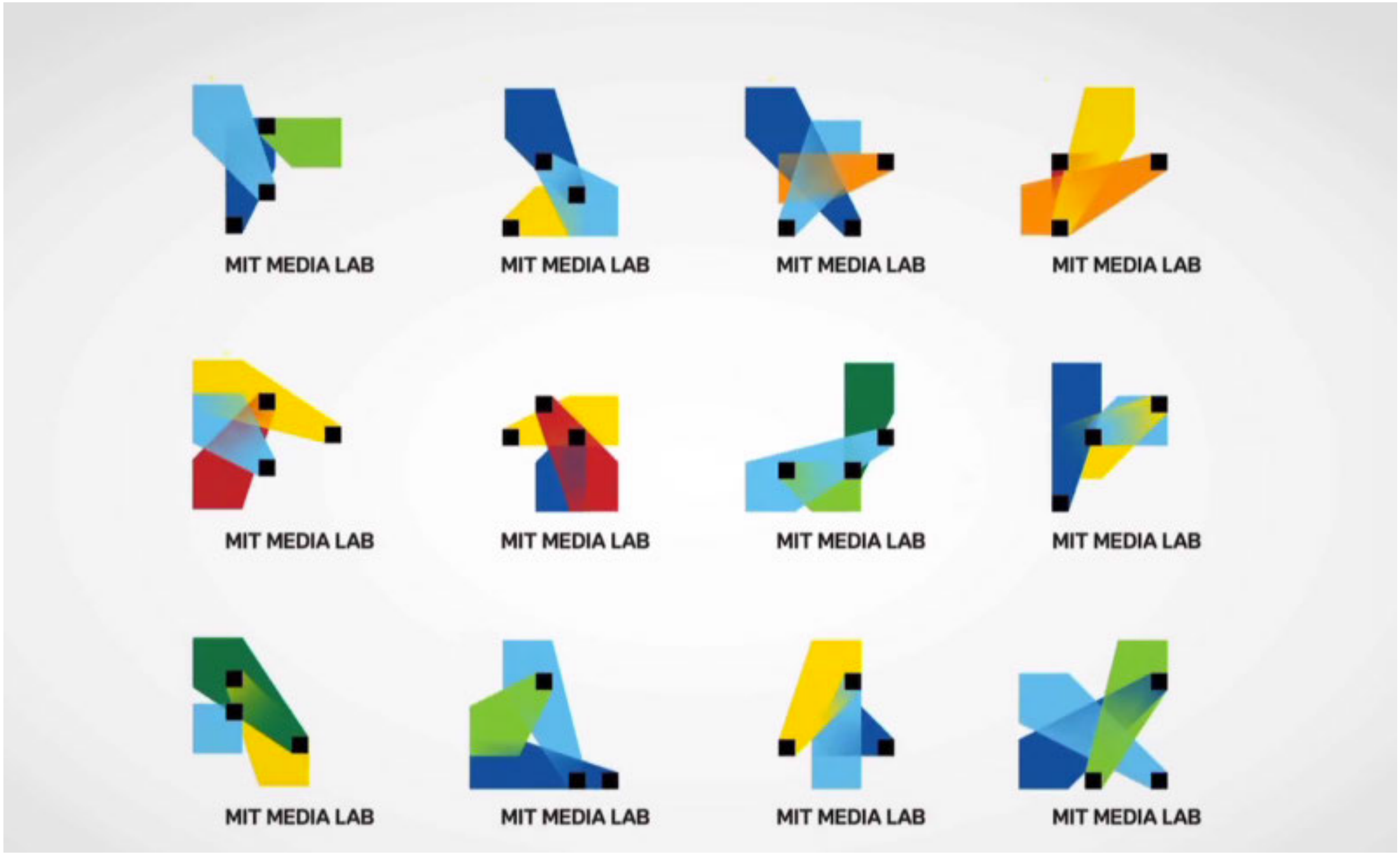
“Maybe the greatest novelty here is the ability of the computer not only to follow ant complex rule of organization but also to introduce an exactly calculated dose of randomness.”

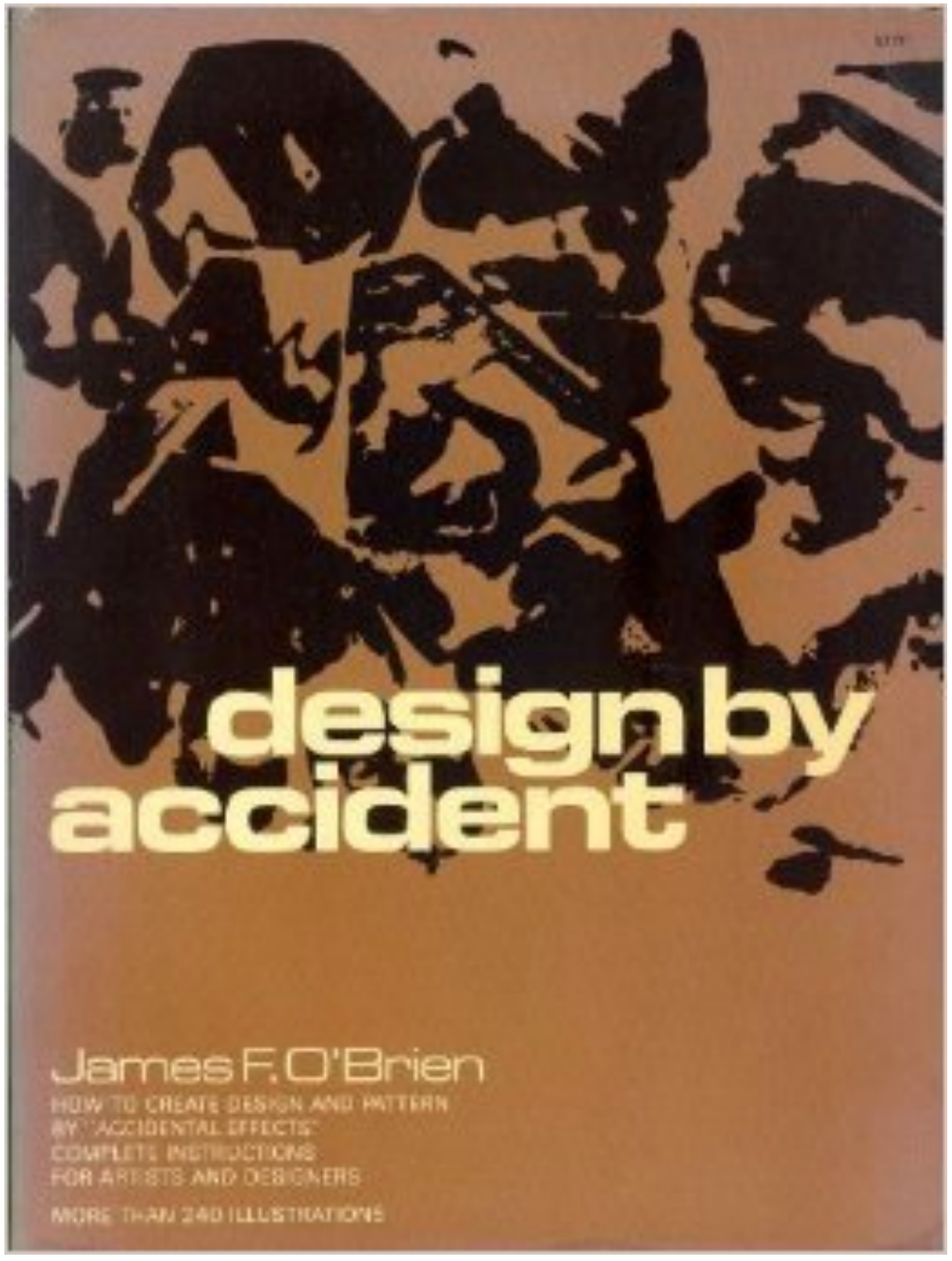
— E.H. Gombrich

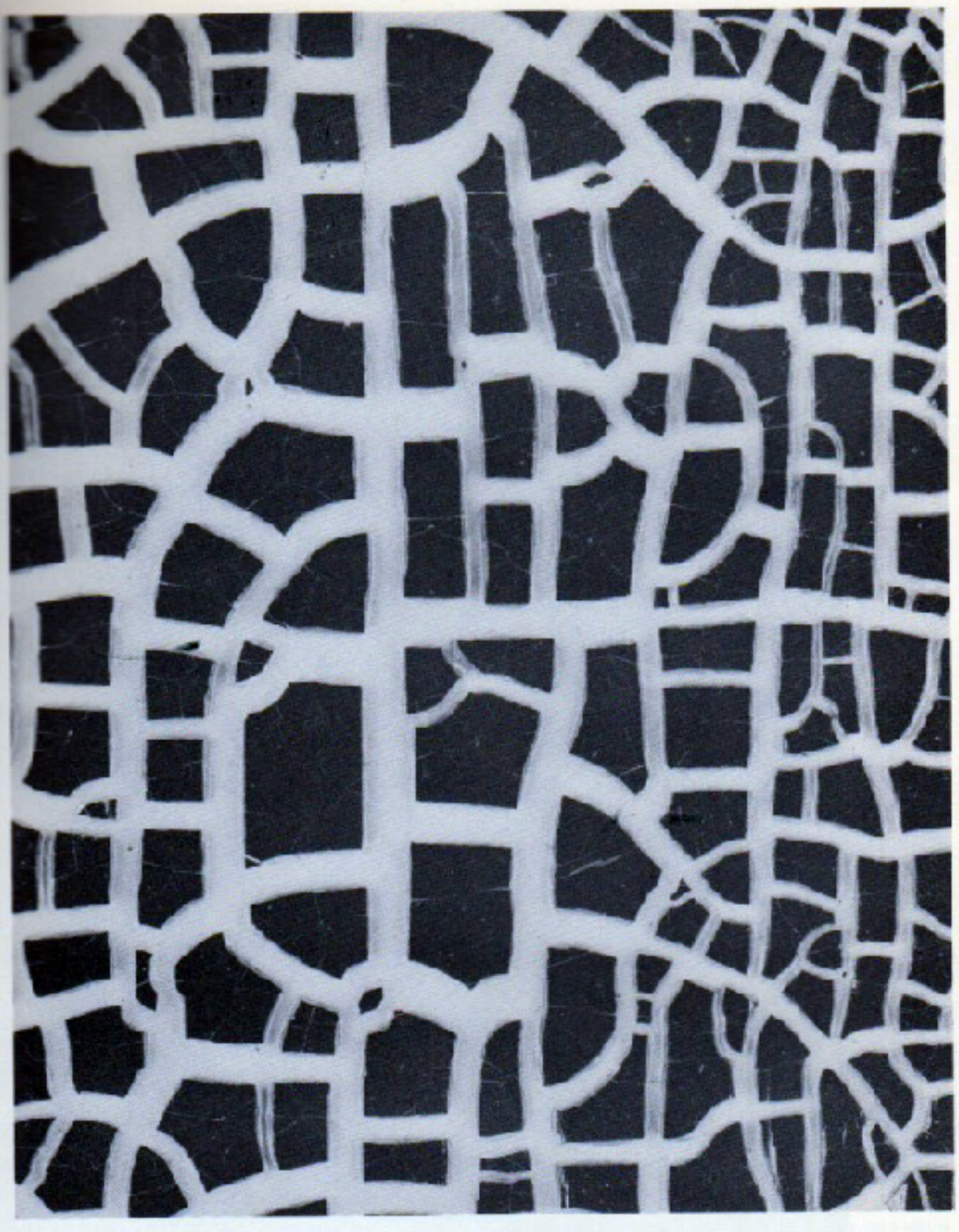


Georg Nees, *Gravel Stones* (1971)

Generative Art







39

cracks and

crackle

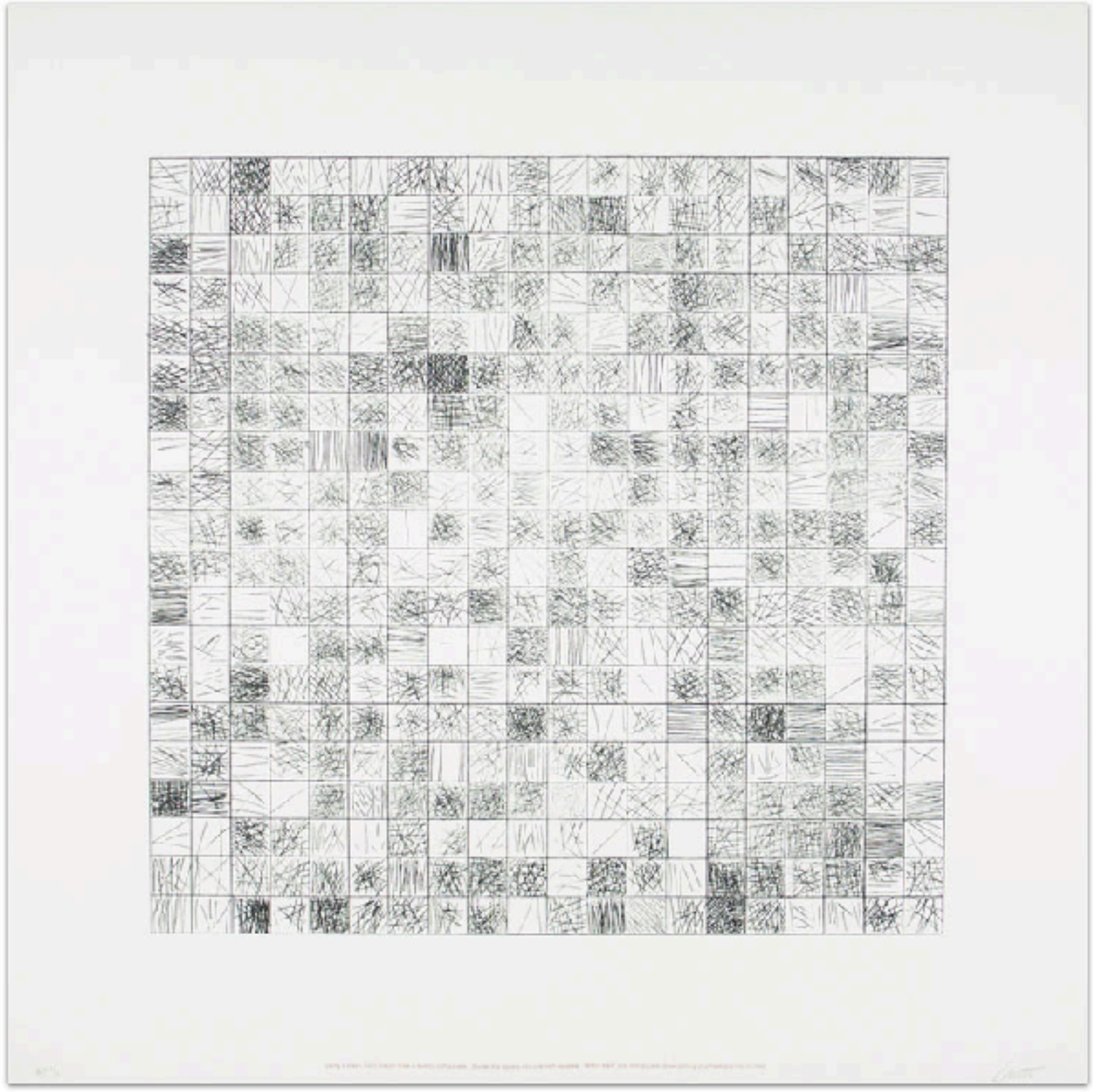
30. Enlargement of part of Design 29.

Sol Lewitt

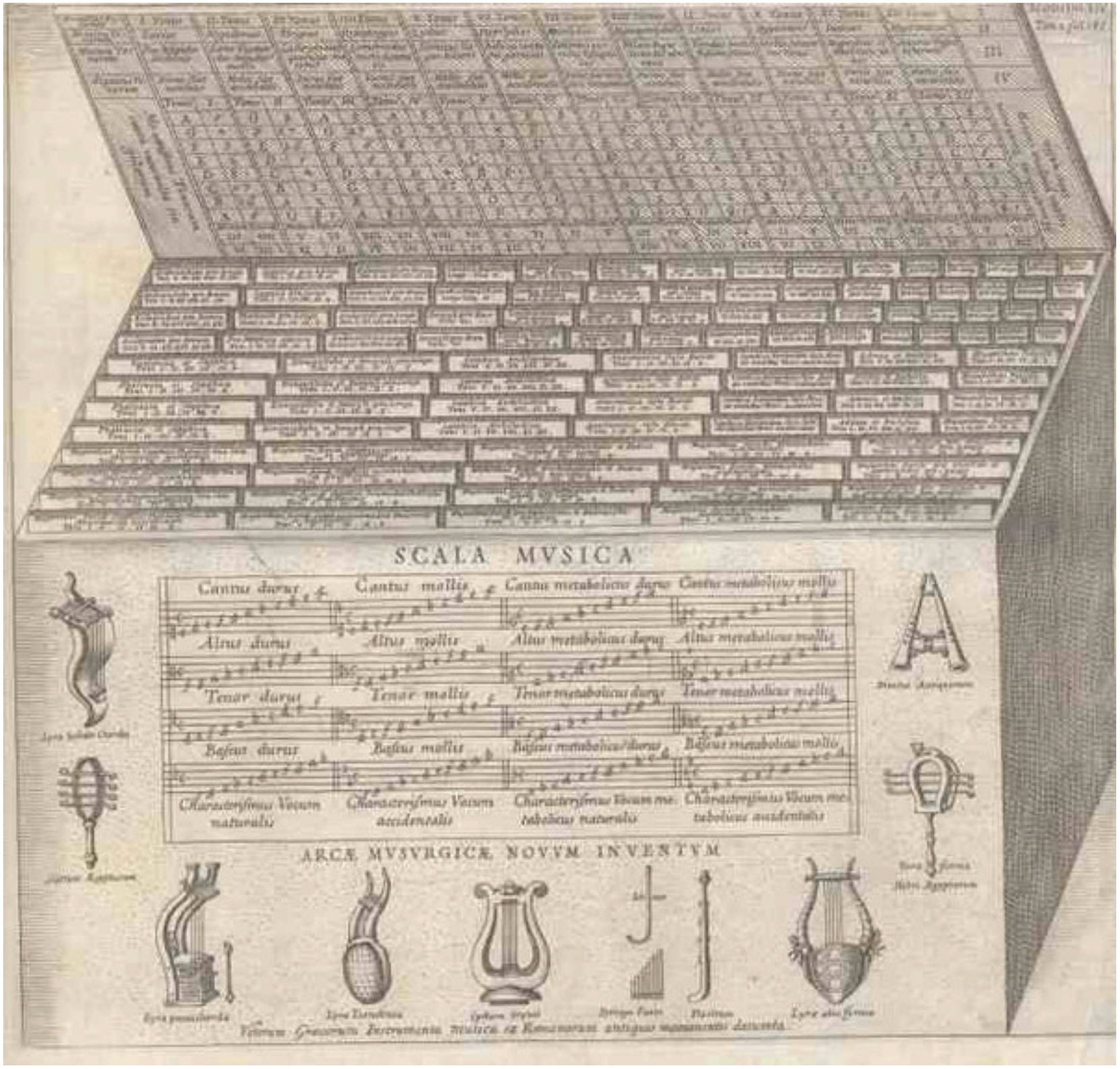
Plate 5. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing or a freehand line or lines.

Sol Lewitt

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.



Athanasius Kircher (1602–1680)



Arca Musarithmica

Random

noise()

noiseDetail()

noiseSeed()

random()

randomGaussian()

randomSeed()

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**

Flipping a coin

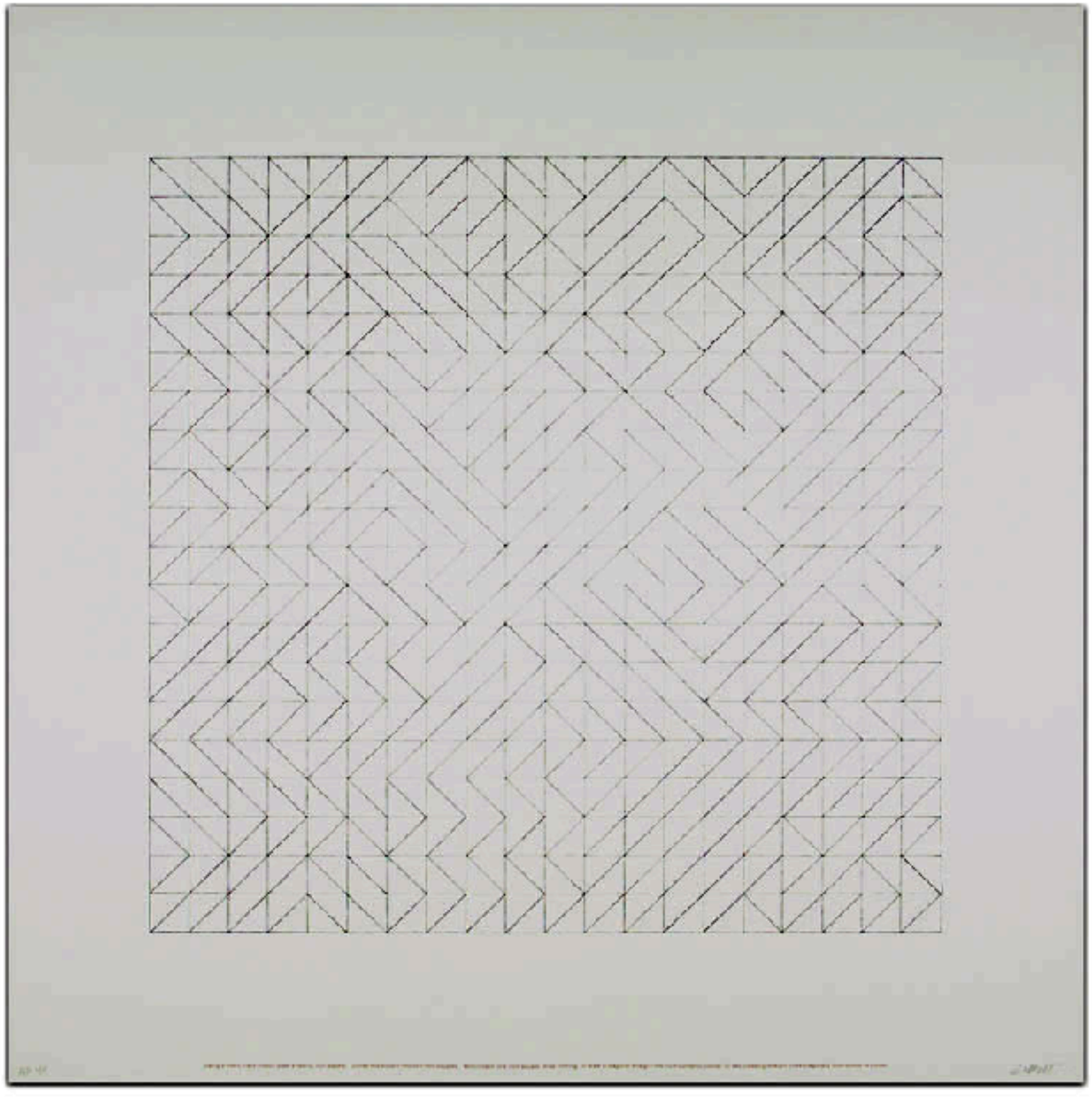
Write a function that simulates flipping a fair coin.

Flipping a biased coin

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

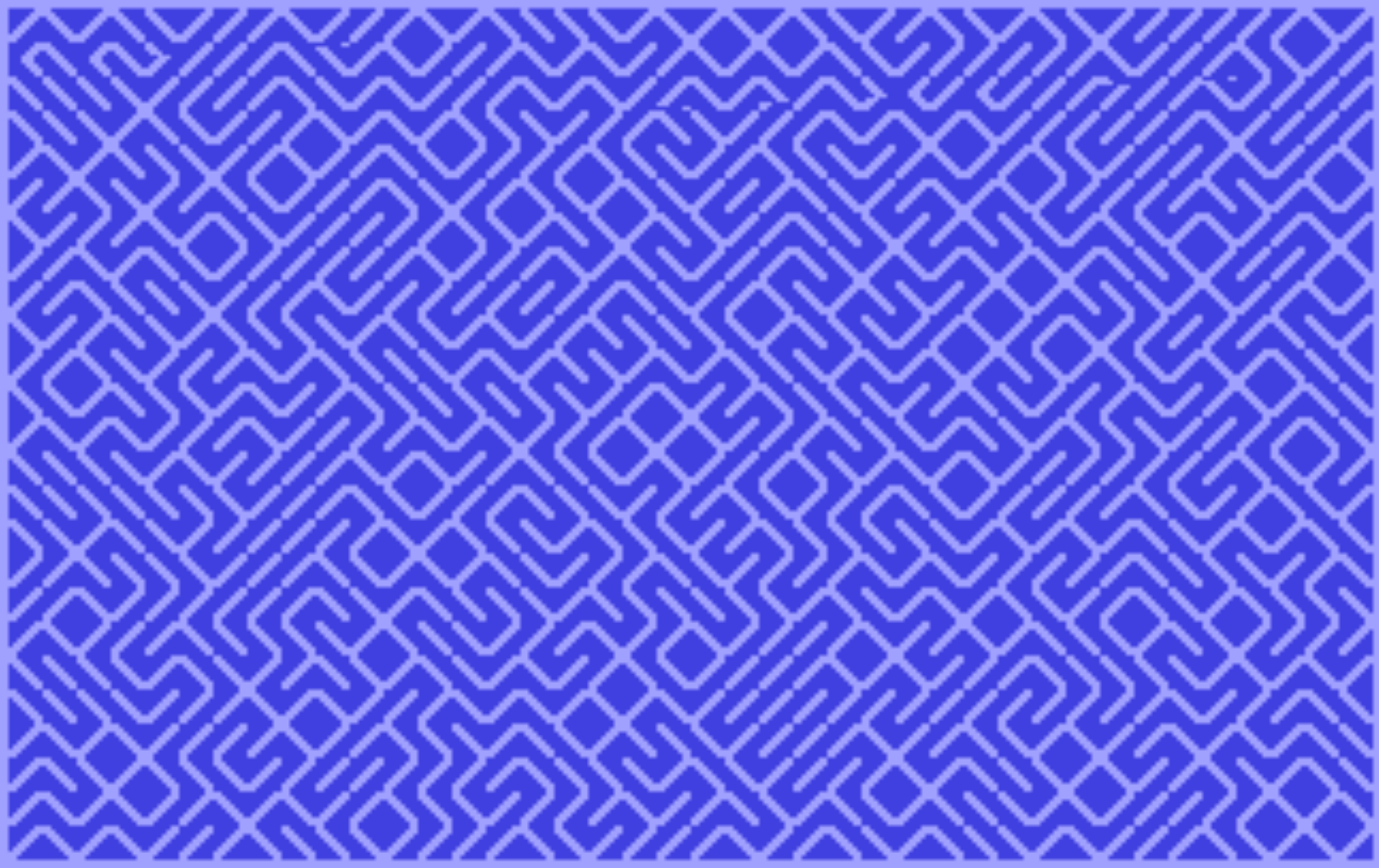
Plate 6. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing, or draw a diagonal straight line from corner to corner, or two crossing straight lines diagonally from corner to corner.

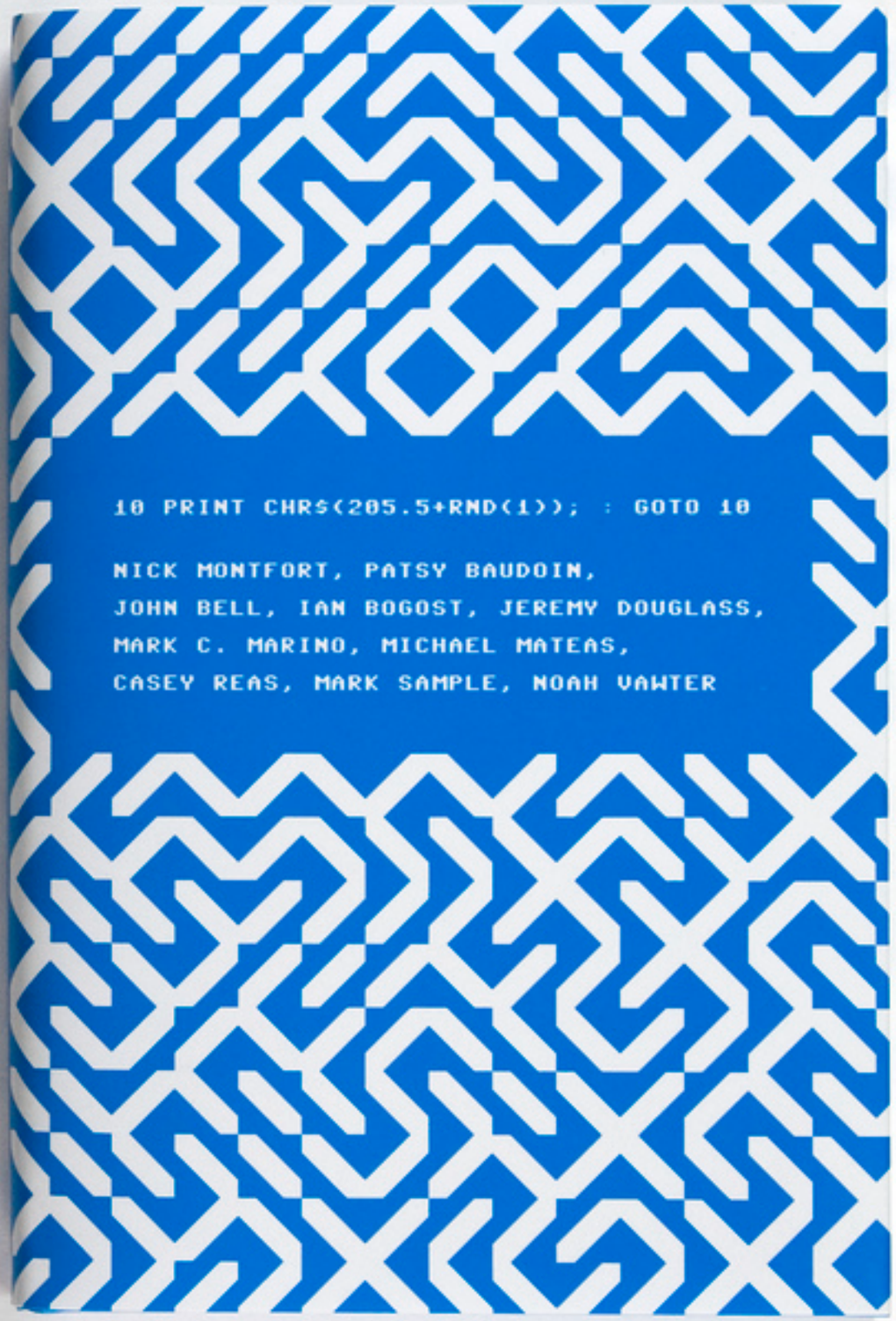
Plate 6. Using a black, hard crayon draw a twenty inch square. Divide this square into one inch squares. Within each one inch square, draw nothing, or draw a diagonal straight line from corner to corner, or two crossing straight lines diagonally from corner to corner.



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

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




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

Is this sequence of digits random?

943702770539217176293176752384674818467669405132000568 12

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420198938095257201065485863278865936153381827968230301952035301852968995773622599413891249721775283479131515574857242454150695950829533116861727855890750983817546374649393 19

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 ) { ... }
```

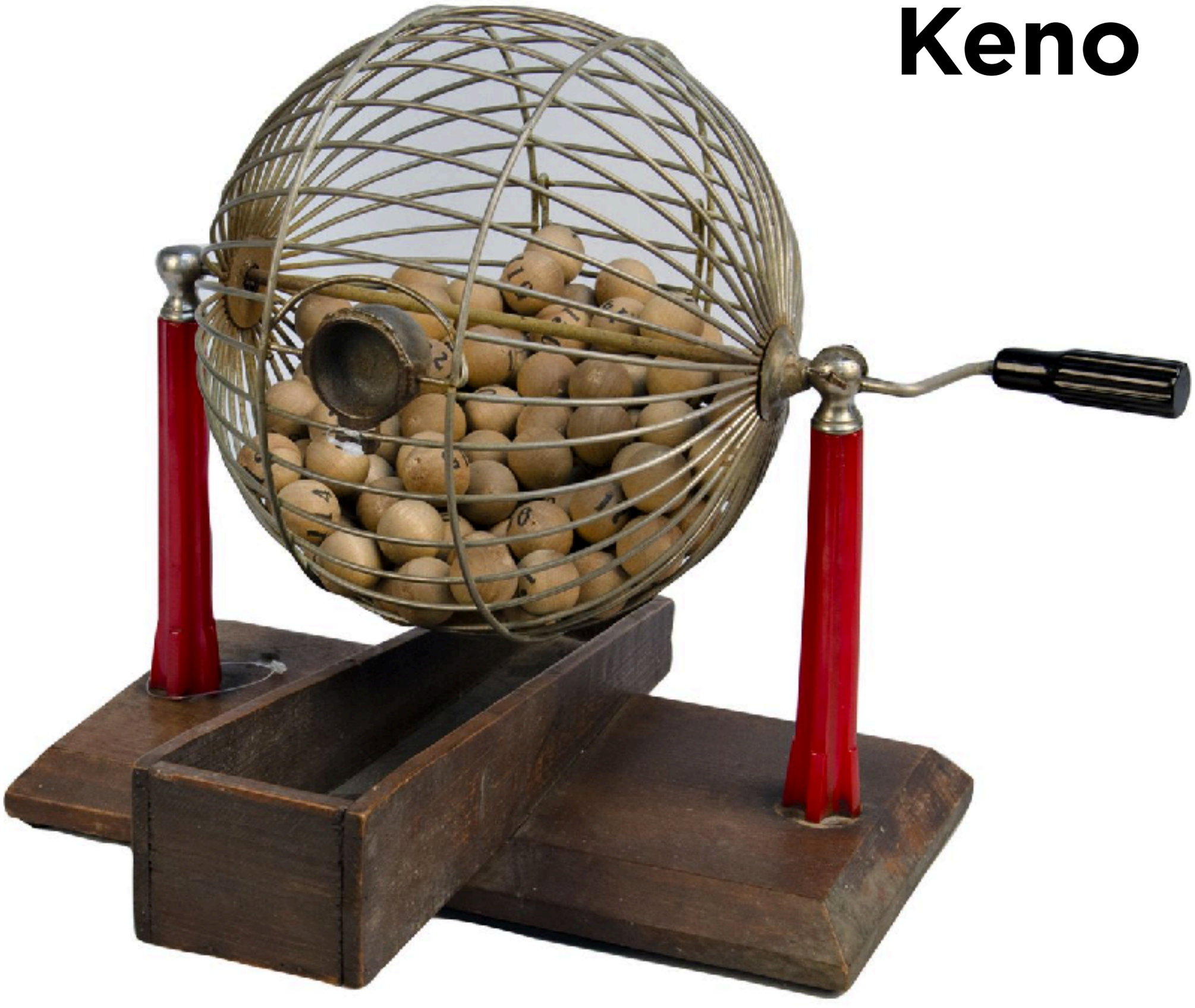
Reset the internal state of Processing's PRNG based on the passed-in `seed`**. A given seed will always produce the same sequence of random numbers.**

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
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
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

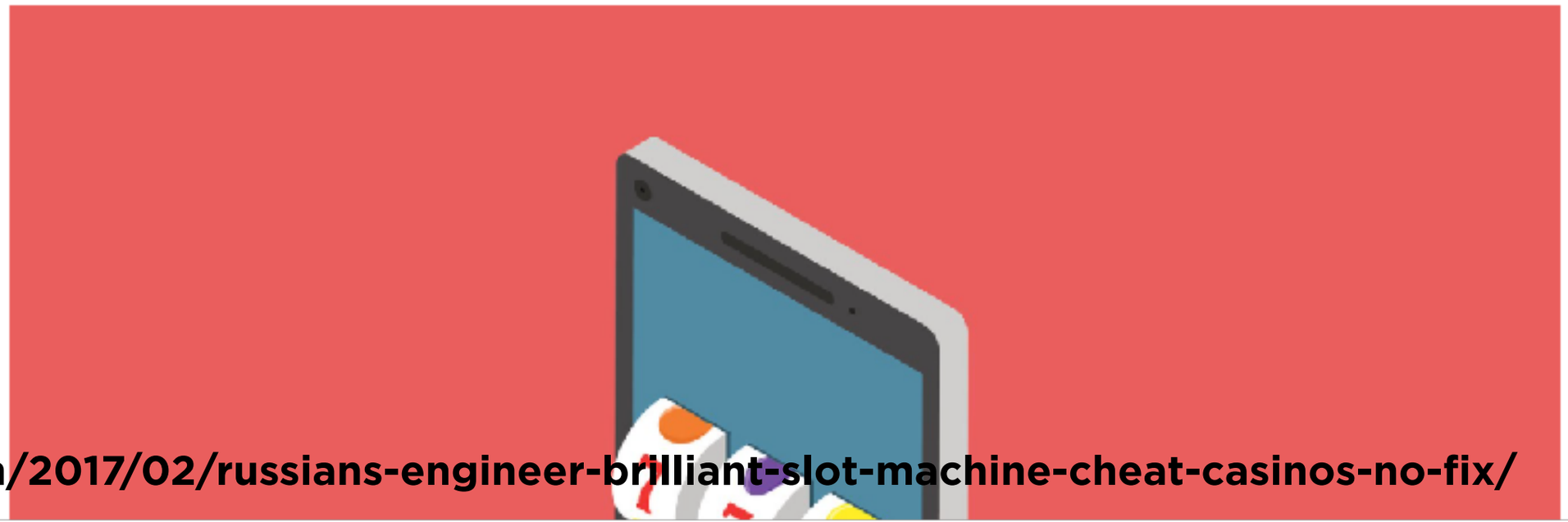
Partial solutions:

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

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

BRENDAN I. KOERNER SECURITY 02.06.17 7:00 AM

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



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

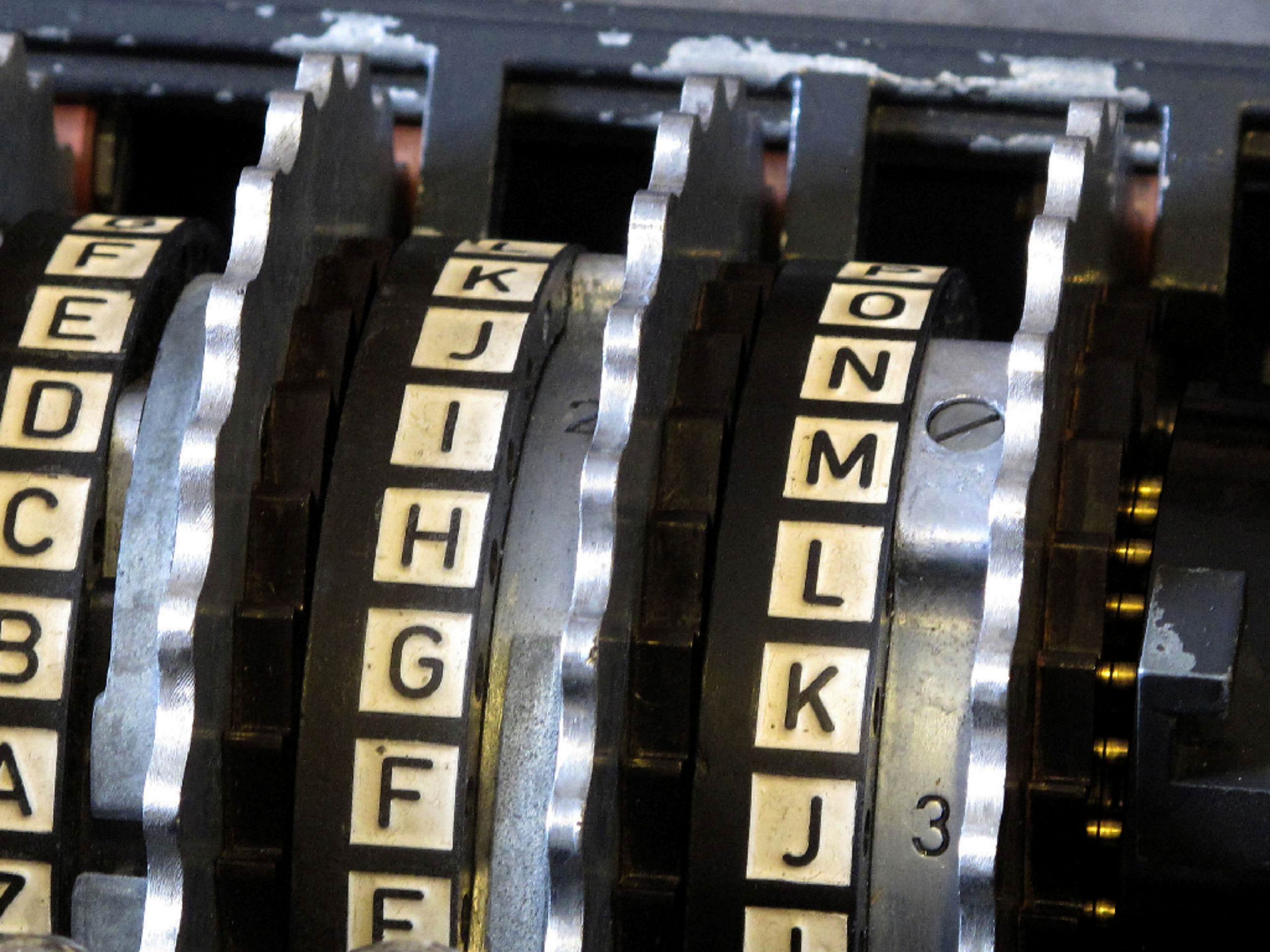
Ronald Dale Harris is a computer programmer who worked for the Nevada Gaming Control Board in the early 1990s and was responsible for finding flaws and gaffes in software that runs computerized casino games. Harris took advantage of his expertise, reputation and access to source code to illegally modify certain slot machines to pay out large sums of money when a specific sequence and number of coins were inserted.

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

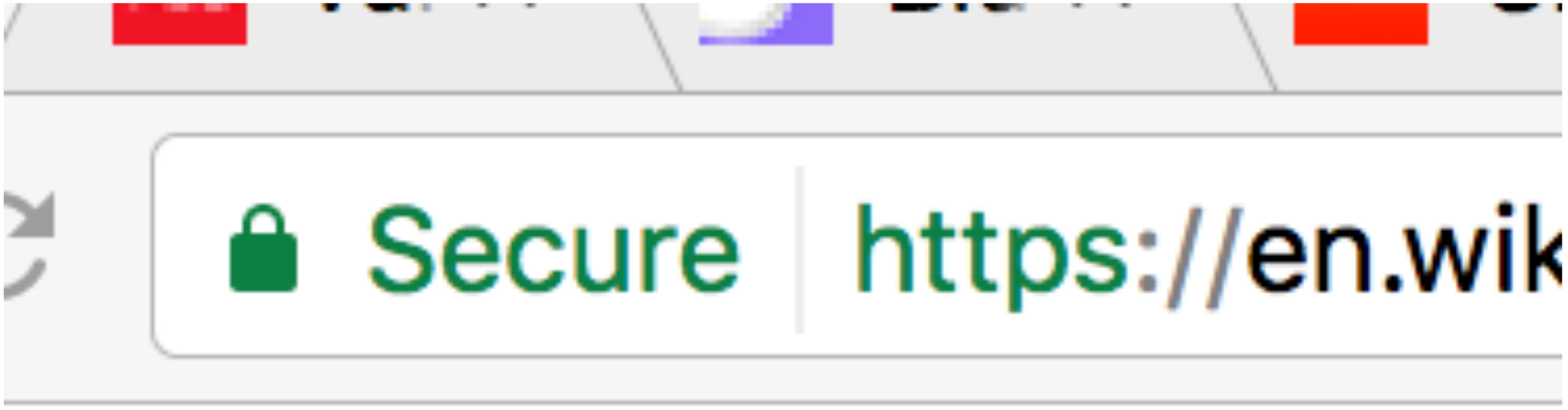
Modern cryptographic protocols often require frequent generation of random quantities. Cryptographic attacks that subvert, or exploit weaknesses in, this process are known as **random number generator attacks**.

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

Enigma Machine


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

Transport Layer Security (TLS/SSL)



Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

...The problem in the running code was discovered in 1995 by Ian Goldberg and David Wagner...

Early versions of Netscape's Secure Socket Layer (SSL) encryption protocol used pseudo-random quantities derived from a PRNG seeded with three variable values: the time of day, the process ID, and the parent process ID.

...The problem in the running code was discovered in 1995 by Ian Goldberg and David Wagner...

RISK ASSESSMENT —

How the NSA (may have) put a backdoor in RSA's cryptography: A technical primer

Here are the basics on backdoors in security systems.

NICK SULLIVAN - 1/5/2014, 7:00 PM

Author Nick Sullivan worked for six years at Apple on many of its most important cryptography efforts before recently joining CloudFlare, where he is a systems engineer. He has a degree in mathematics from the University of Waterloo and a Masters in computer science with a concentration in cryptography from the University of Calgary. This post was originally written for the CloudFlare blog and has been lightly edited to

https://arstechnica.com/security/2014/01/how-the-nsa-may-have-put-a-backdoor-in-rsas-cryptography-a-technical-primer/

The U.S. National Institute of Standards and Technology has published a collection of “deterministic random bit generators” it recommends as NIST Special Publication 800-90. One of the generators, Dual_EC_DRBG, was favoured by the National Security Agency.

The U.S. National Institute of Standards and Technology has published a collection of “deterministic random bit generators” it recommends as NIST Special Publication 800-90. One of the generators, Dual_EC_DRBG, was favoured by the National Security Agency.

In 2013, Reuters reported that documents released by Edward Snowden indicated that the NSA had paid RSA Security $10 million to make Dual_EC_DRBG the default in their encryption software, and raised further concerns that the algorithm might contain a backdoor for the NSA.

Withdrawal Date:	**June 2015**
Withdrawal Note:	**NIST Released Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators** **June 25, 2015** **NIST announces the completion of Revision 1 of NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators. This Recommendation specifies mechanisms for the generation of random bits using deterministic methods. In this revision, the specification of the Dual_EC_DRBG has been removed. The remaining DRBGs (i.e., Hash_DRBG, HMAC_DRBG and CTR_DRBG) are recommended for use. Other changes included in this revision are listed in an appendix.**

Withdrawal Date:	**June 2015**
Withdrawal Note:	**NIST Released Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators** **June 25, 2015** **NIST announces the completion of Revision 1 of NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators. This Recommendation specifies mechanisms for the generation of random bits using deterministic methods. In this revision, the specification of the Dual_EC_DRBG has been removed. The remaining DRBGs (i.e., Hash_DRBG, HMAC_DRBG and CTR_DRBG) are recommended for use. Other changes included in this revision are listed in an appendix.**

- 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.