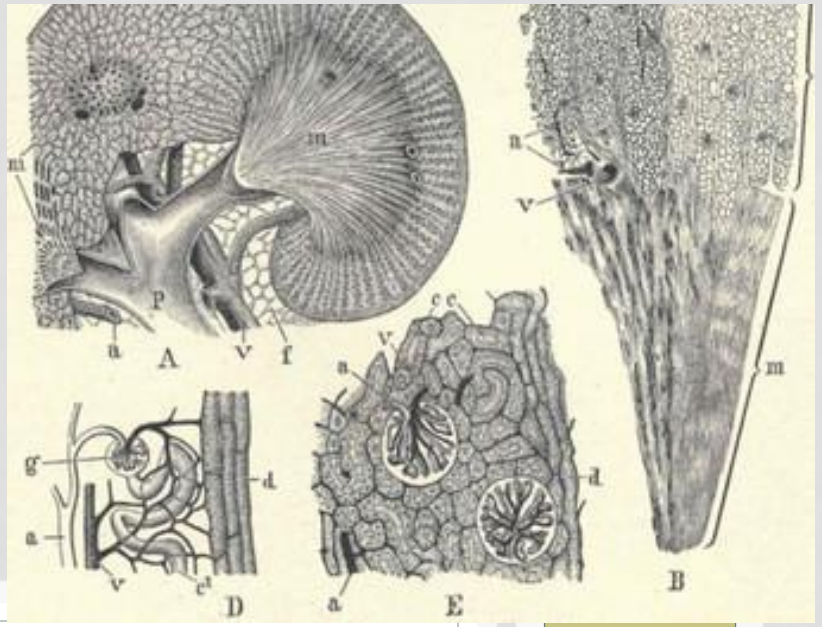
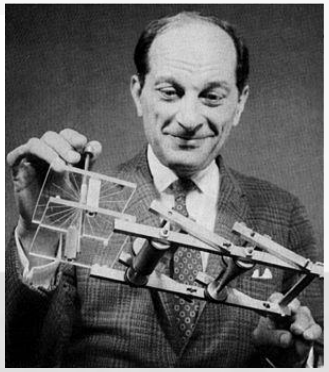


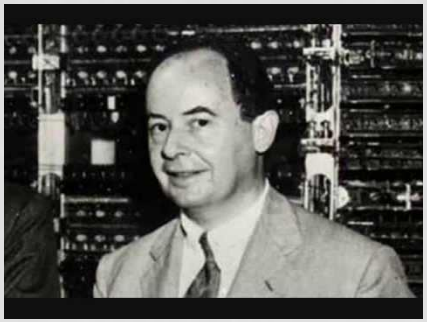
CELLULAR AUTOMATA
+
REACTION-DIFFUSION SYSTEMS

REFERENCE: HTTP://WWW.KARLSIMS.COM/RD.HTML

PART I: CELLULAR AUTOMATA



- History:
  - Discovered by Stanislaw Ulam and John Von Neumann in 1940
  - Used in Mathematics, Physics, Cellular Biology
  - (imo) kind of a cool toy project.



- Basic Idea:
  - A rectangular grid of cells
  - Each cell can either be alive (1) or dead (0)
  - Simple state transitions determine the state of this cell in the next generation.
    - Usually based on the state of the neighbors.
    - Von-Neumann Neighborhood (4 neighbors)
    - Moore Neighborhood (8 neighbors)
    - Usually "wrapped" (a toroidal universe)

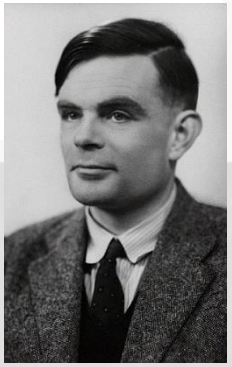
EXAMPLES

- **Conway's Game of Life** is the classic example
  - 1 => 0 if less than 2 live neighbors (under-population)
  - 1 => 0 if more than 3 live neighbors (over-population)
  - 0 => 1 if exactly 3 live neighbors (birth)
  - State remains the same otherwise.
  - https://bitstorm.org/gameoflife/
- A good example of **emergent behavior**
- **Wireworld** is another cool one
  - 4 states: 0=empty, 1=head, 2=tail, 3=wire
  - 1 => 2
  - 3 => 1 if there are 1-2 neighbors that are in state 1.
  - 2 => 3
  - https://en.wikipedia.org/wiki/Wireworld

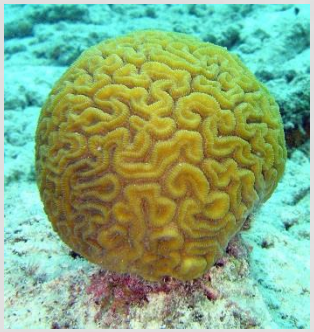
IMPLEMENTATION

- Creating a (partial) copy is important
  - See why?
- Visualizing
  - One pixel per cell
  - Maybe scale up?

PART II: REACTION-DIFFUSION SYSTEMS



- Based on Turing's "Chemical Basis of Morphogenesis" [1952]
  - http://www.dna.caltech.edu/courses/cs191/paperscs191/turing.pdf
  - [tangent: Turing's life]
- Attempts to simulate the chemical basis for patterns in nature
  - Fingerprints
  - Coral folding
  - Zebra stripes
  - …



BASIC IDEA (GROSS SIMPLIFICATION)

- A system containing two chemicals / organisms: A and B
  - A is gradually converted to B.
  - B eventually dies off (at a rate of **kill-rate**)
  - A is regularly injected into the system (**feed-rate**)
- Both A and B diffuse to neighboring cells
  - Uses a Laplacian operator
  - This is the big difference with cellular automata
- We simulate on a toroidal rectangular grid
  - Moore Neighborhood (7 neighbors)
  - Wrap-around
  - Each cell holds two values: A and B (generally 0.0 – 1.0)

MATH!

$$A' = A + \left(D_A * \nabla^2 A - AB^2 + f * (1 - A)\right) * \Delta t$$

$$B' = B + (D_B * \nabla^2 B + AB^2 - \quad (k + f) * B) * \Delta t$$

- Key:
  - A' and B' are the *new* values of A & B after update
    - They should generally be *very close* to [0.0 – 1.0]
    - I had to clamp them – occasionally they would go outside range.
  - A and B are the *current* values of A & B before update
  - D_A and D_B are the diffusion rates for A and B [0.0 – 1.0]
  - f = feed-rate for A [0.0 – 1.0]
  - k = kill-rate for B (0.0 – 1.0)
  - Δt is a rate (smaller = slower, bigger = faster)
  - $\nabla^2 A$ and $\nabla^2 B$ are the 2-d Laplacian operator (next slide)
- The "breakdown" on http://www.karlsims.com/rd.html is very helpful -- I'll repeat and add to it here…

2D LAPLACIAN

- A scalar which looks at the current concentration of either A or B in a small neighborhood.
- Calculates a new value which is slightly more like the neighbors.
- Pass this **convolution kernel** over all cells.
  - Make sure you do it on a copy (just like CA)
  - If calculating $\nabla^2 A$, just use the A values.
  - For example if the cells held…
  - $0.05 * 0.8 + 0.2 * 0.6 + 0.05 * 0.2 + 0.2 * 0.1 - 1.0 * 0.5 + 0.2 * 0.3 + 0.05 * 0.4 + 0.2 * 0.2 + 0.05 * 0.1 = \mathbf{-0.185}$

Laplacian convolution kernel

	x - 2	x - 1	x	x + 1	x + 2
y - 2					
y - 1		0.05	0.2	0.05	
y		0.2	-1.0	0.2	
y + 1		0.05	0.2	0.05	
y + 2					

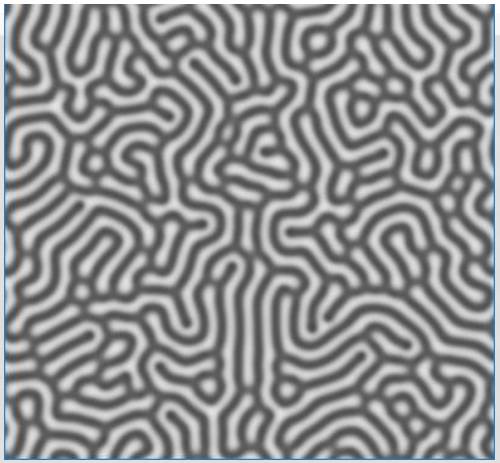
Sample cell neighborhood

0.8	**0.6**	**0.2**
0.1	**0.5**	**0.3**
0.4	**0.2**	**0.1**

VISUALIZING THE RESULTS

- I just used A (A=0=black, A=1=white)
- Many other people use a gradient
  - https://pmneila.github.io/jsexp/grayscott/
- You could also use A and B for two channels (red and green, maybe).
- Some people also use a 3d-effect
  - https://www.youtube.com/watch?v=8dTmUr5qKvI

MY RESULT #1 & #2



- Result #1
  - DA = 1.0
  - DB = 0.5
  - f = 0.055
  - k = 0.062
  - Initially, A=1,B=0 everywhere except a small patch where A=0, B=1.
  - https://youtu.be/I5PuXBvgLmc
- Result #2
  - Just like #1, but I load an image, setting A=red, B=(1.0-red)
  - https://youtu.be/8H_9DdIIU_k

NON-UNIFORM VALUES

- You can get much more interesting results if you vary the 4 parameters non-uniformly over the image
  - Often by loading an image. Perhaps with:
    - E.g. Keep all other values as before, but make the feed-rate vary per-pixel using the red channel of an image.
- Result
  - Feed-rate = “bill.png”; kill-rate = “giraffe.png”
  - diffA = 0.9; diffB = 0.15;
  - Seed = checkerboard (30px)
  - https://youtu.be/F2cGd5QhyuA