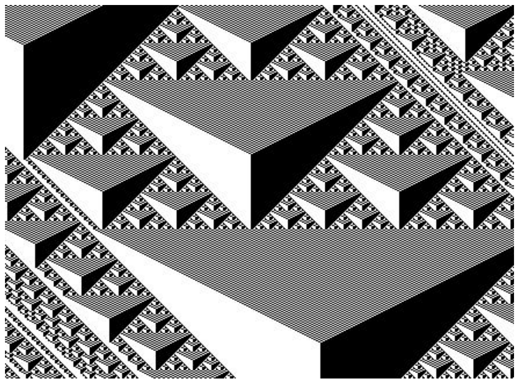
Impact Lab 2023: Programming Fundamentals
Lecture 5: Cellular Automata

Summer 2023

School of Computing and Data Science

Wentworth Institute of Technology



Topics for Today

- **1D Cellular Automata**
- 2D Cellular Automata
- Object Communication

What is Cellular Automation?

As an example of 1D and 2D arrays, we're going to talk about Cellular Automation (CA). This represents a single simulation in which the states of cells on a board are determined by specific rules.

Noita's material simulations and interactions are entirely based on cellular automation

What is Cellular Automation?

As an example of 1D and 2D arrays, we're going to talk about Cellular Automation (CA). This represents a single simulation in which the states of cells on a board are determined by specific rules.

The best known example of CA is **Conway's Game of Life.**

John Conway developed this cellular automation in 1970. The "game" itself is a zero player game that is entirely determined by the initial configuration of the play area.

We'll learn the rules and how to create this "game" in **1D** and **2D!**

What is Cellular Automation?

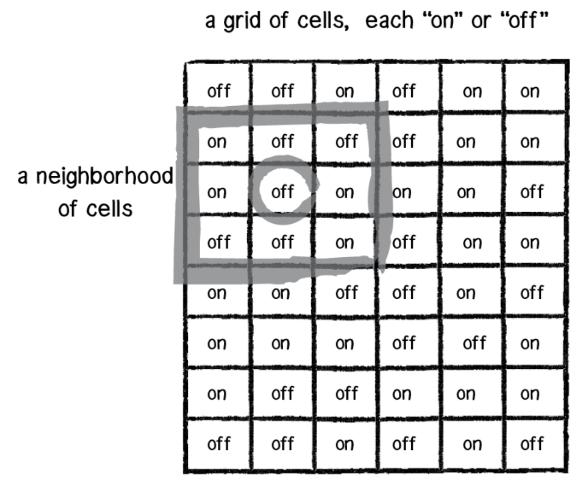
A cellular automation is a model of a system of “cell” objects with the following characteristics:

The cells live on a **grid**
(1D, 2D, 3D, or more)

Each cell has a **state.**
The number of states is usually finite. (e.g. on or off, 0 or 1, alive or dead, etc.)

Each cell has a **neighborhood.**
This can be defined in many ways depending on the type of simulation.

What is Cellular Automation?



A Bit of History

The development of CA started at Los Alamos National Laboratory in the 40’s with Stanislaw Ulam and John von Neumann.

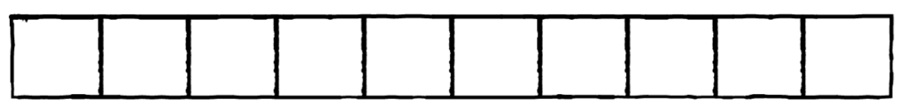
Ulam was interested in crystal growth and von Neumann in self replicating robots.

The most significant work was done by Stephan Wolfram in 2002 with his 1280 page book (*A New Kind of Science*) that explored CA in problems in physics, biology and chemistry.

Elementary CA

What is the simplest CA that you can think of (based on the things we need to make a CA)?

1D Grid
2 States (0 or 1)
3 Cell Neighborhood



Elementary CA

What is the simplest CA that you can think of (based on the things we need to make a CA)?

1D Grid
2 States (0 or 1)
3 Cell Neighborhood

1	0	1	0	1	1	1	0	0	1

Elementary CA

What is the simplest CA that you can think of (based on the things we need to make a CA)?

1D Grid
2 States (0 or 1)
3 Cell Neighborhood

1	0	1	0	1	1	1	0	0	1

Elementary CA

What is the simplest CA that you can think of (based on the things we need to make a CA)?

1D Grid
2 States (0 or 1)
3 Cell Neighborhood

1	0	1	0	1	1	1	0	0	1

What about the edge?

Elementary CA

The most important part of CA - **Time**

1	0	1	0	1	1	1	0	0	1

Generation 1

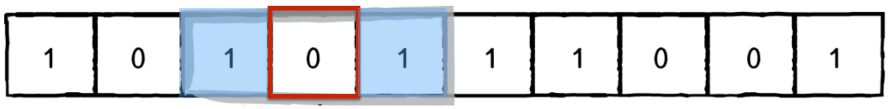
?	?	?	?	?	?	?	?	?	?

Generation 2

?	?	?	?	?	?	?	?	?	?

Elementary CA

We have a description of or neighborhood around a cell, let's use that to determine the next generation!



Notice that our center cell (the 0) determines which cell gets updated in the next generation.

Elementary CA

Thera are many ways to come up with rules that determine the next generation.

When you blur an image in Photoshop, it uses cellular automata like rules!

For this 1D CA, we can look at all possible configurations of the three cells and come up with rules for each one!

How many rules do we need?

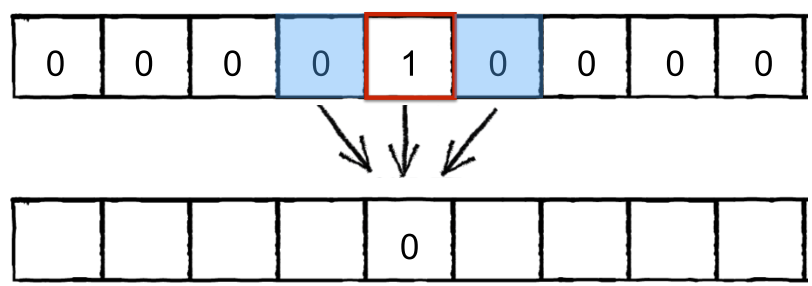
Elementary CA - Rules

8 Configurations:

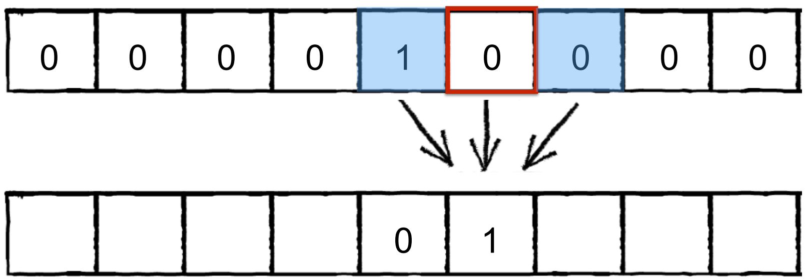
0 0 0   0 0 1   0 1 0   0 1 1   1 0 0   1 0 1   1 1 0   1 1 1

I've picked these rules very specifically…

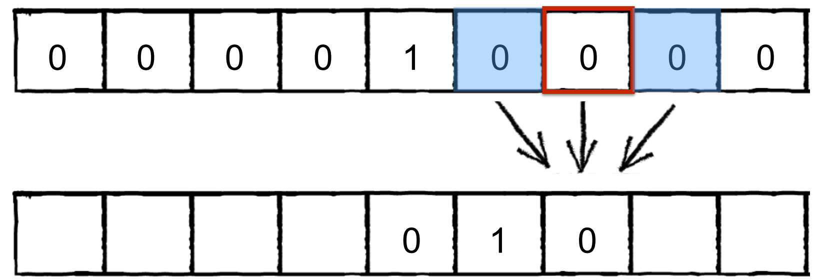
Elementary CA - Handwritten Example



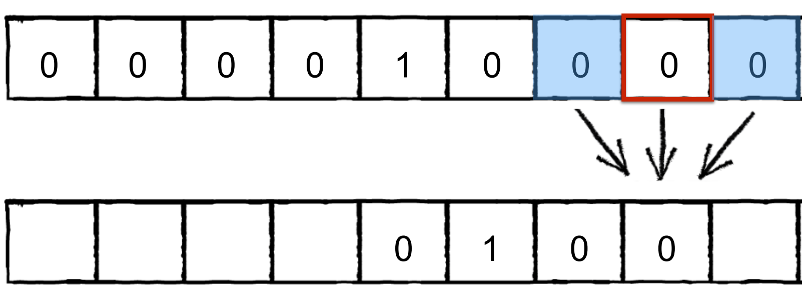
Elementary CA - Handwritten Example



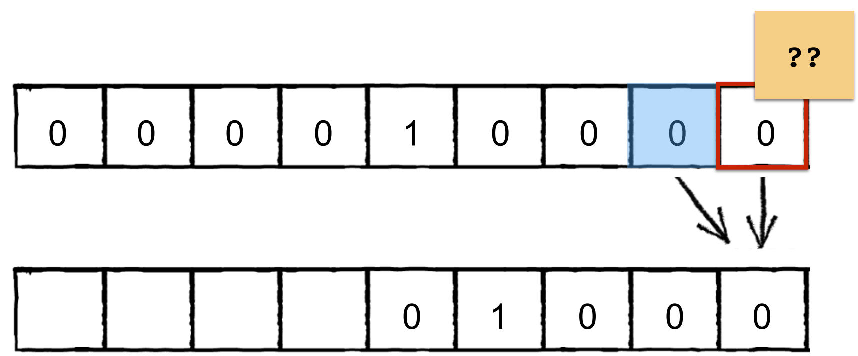
Elementary CA - Handwritten Example



Elementary CA - Handwritten Example

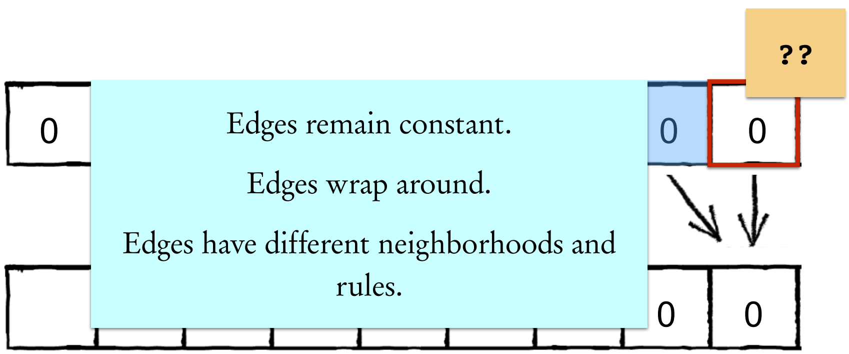


Elementary CA - Handwritten Example



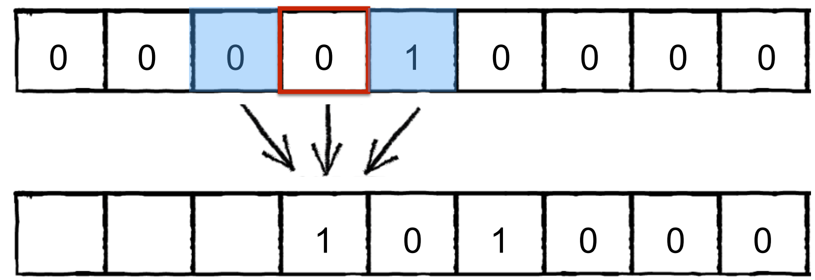
We'll just assume that if we go off the grid, the edge value is zero

Elementary CA - Handwritten Example



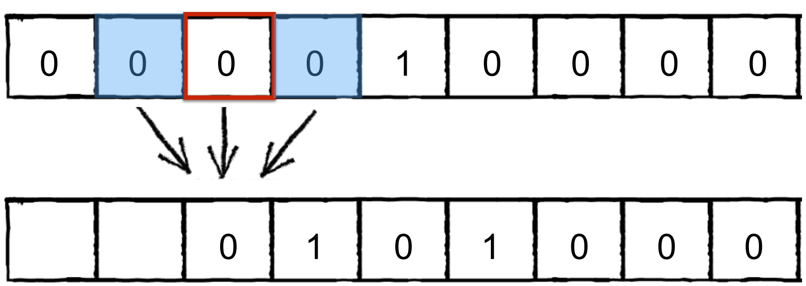
Wentworth
Computing & Data Science

Elementary CA - Handwritten Example



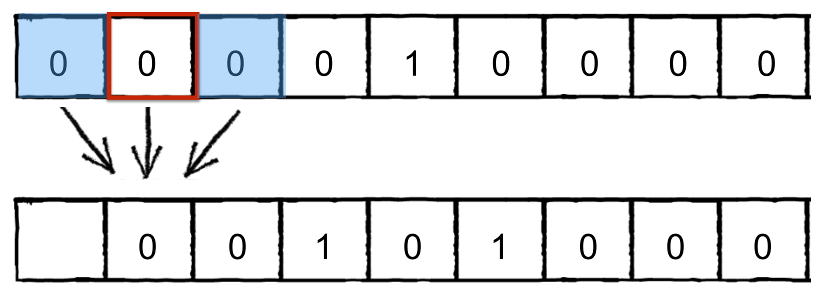
Wentworth
Computing & Data Science

Elementary CA - Handwritten Example



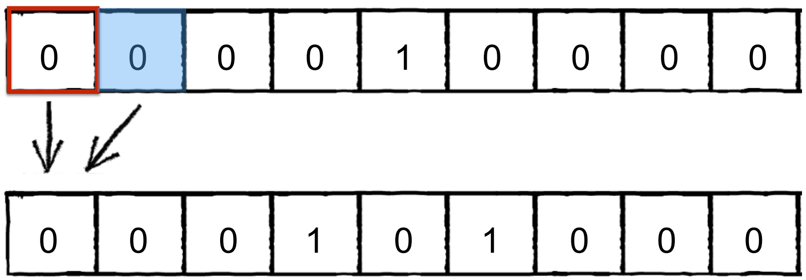
Wentworth
Computing & Data Science

Elementary CA - Handwritten Example

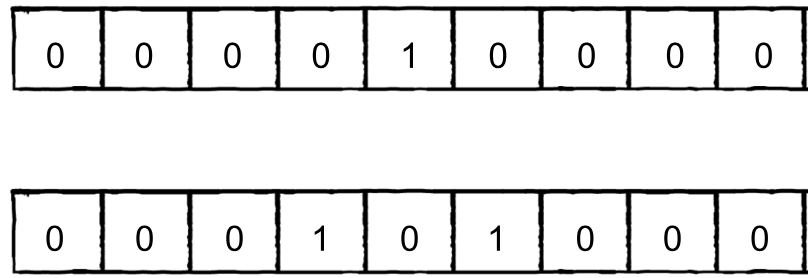


Wentworth
Computing & Data Science

Elementary CA - Handwritten Example



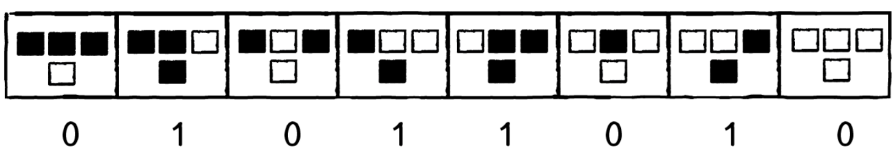
Elementary CA - Handwritten Example



So, we now have a complete first generation. Congratulations, you've done your first CA simulation!

That was a bit tedious though...

Elementary CA - Visual Example



The rules set that we used is called Rule 90. On a computer we can represent 1s as filled cells and 0s as empty cells...

Topics for Today

- 1D Cellular Automata
- **2D Cellular Automata**
- Object Communication

The Game of Life

Two-Dimensions!

More Complex than 1D:
Each neighborhood is eight tiles, which leads to more configurations but also more useful applications

In 1970 Martin Gardner wrote an article in *Scientific American* that documented John Conway's "Game of Life", describing it as "recreational" mathematics. He recommended getting out a chess board and some checkers and playing for yourself.

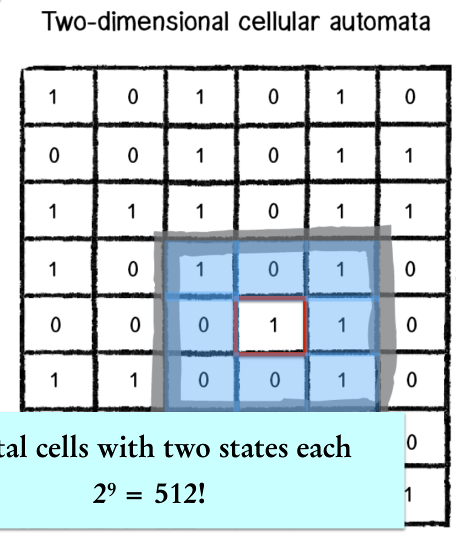
These days, the "Game of Life" is more of a computational cliché.

The Game of Life

Instead of a line of cells, we have a two-dimensional matrix of cells.

Each cell only has a value of 0 or 1 (dead or alive).

The neighborhood is also large, eight total cells.

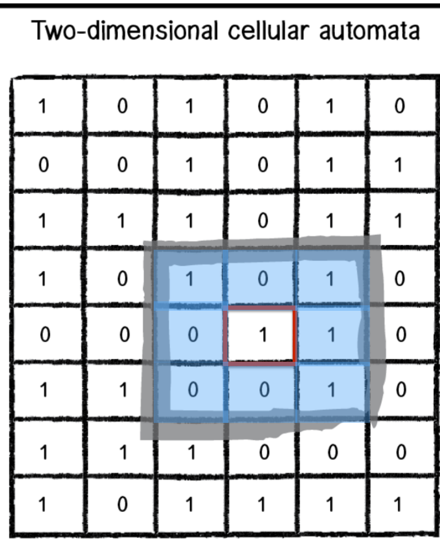


9 total cells with two states each
$2^9 = 512$!

The Game of Life

Obviously we don't want to define 512 rules, so what do we do?

Let's define some rules based on the general look of the neighborhood...



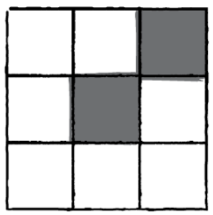
The Game of Life - Rules

Death of a Cell:
Overpopulation: four or more alive neighbors
Loneliness: one or fewer alive neighbors

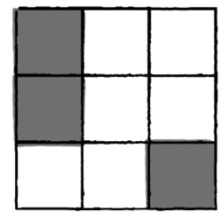
Birth of a Cell:
If a cell is dead it comes back to life if it has exactly 3 alive neighbors

Stasis:
Stay Alive: Exactly two or three live neighbors
Stay Dead: Anything other than three live neighbors

The Game of Life - Rules Example



The Game of Life - Rules Example



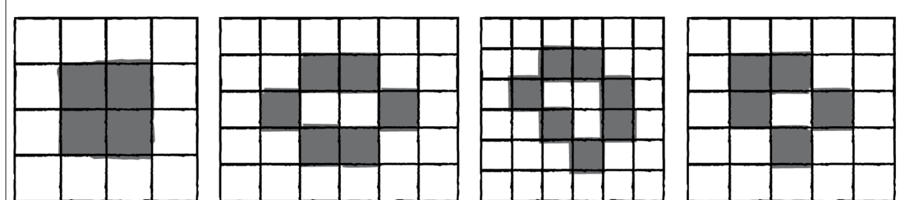
The Game of Life - Notes

For the 1D CA, we can "stack up" the generations, but that's not very viable for 2D.

The typical way to present the "Game of Life" is to treat each generation like a single frame in a movie. Instead of seeing all the generations at once, we see them over time!

There are a few interesting structures that we will see in the simulation: some that don't change, some that oscillate back and forth between two states and some that move across the grid...

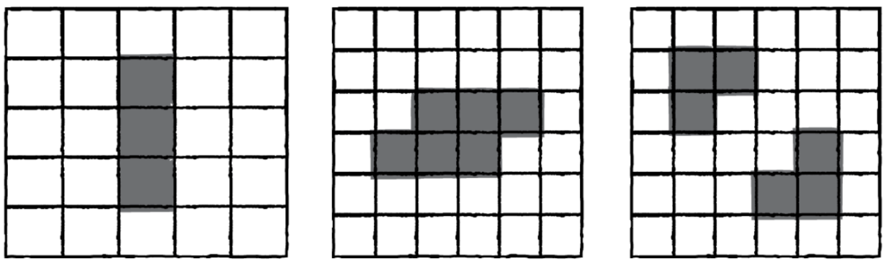
The Game of Life - Notes



block beehive loaf boat

These are stationary

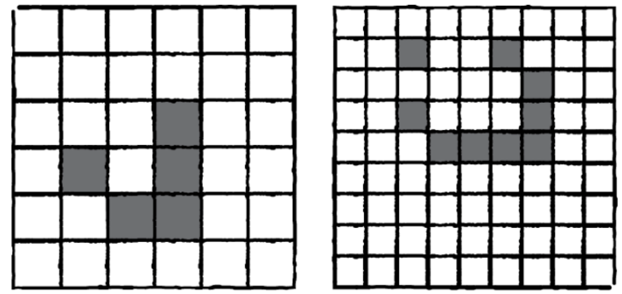
The Game of Life - Notes



blinker toad beacon

These will oscillate between two states

The Game of Life - Notes



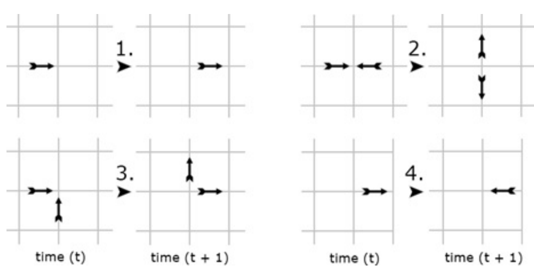
glider lightweight spaceship

These will move across the grid

More Than Just Game of Life

CA isn't just for simple game like behavior

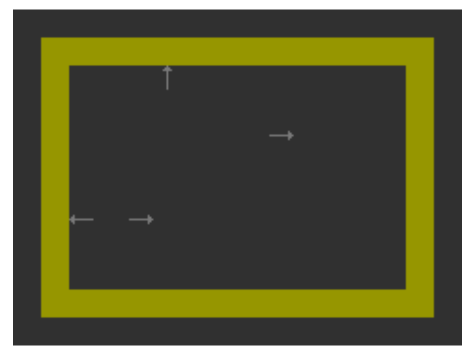
Fluid Dynamics - HPP Model:



More Than Just Game of Life

CA isn't just for simple game like behavior

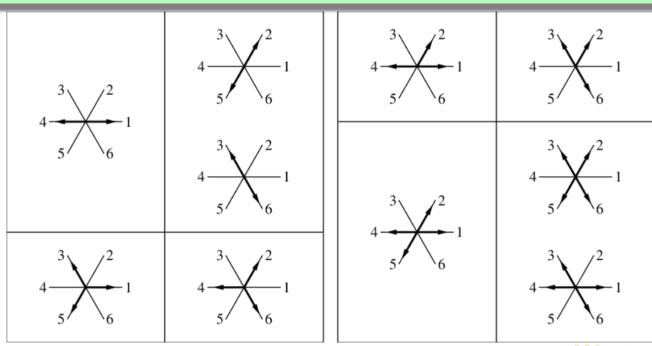
Fluid Dynamics - HPP Model:



More Than Just Game of Life

CA isn’t just for simple game like behavior

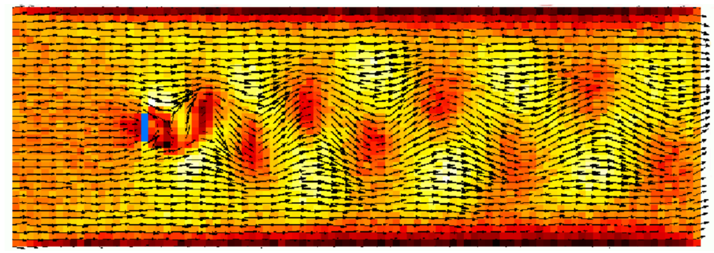
Fluid Dynamics - FHP Model:



More Than Just Game of Life

CA isn’t just for simple game like behavior

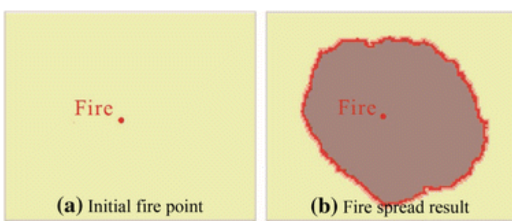
Fluid Dynamics - FHP Model:



More Than Just Game of Life

CA isn’t just for simple game like behavior

Forest Fire Spread:



More Than Just Game of Life

CA isn’t just for simple game like behavior

Many, Many More:

- Sand Dune Movement
- Snow Accumulation
- Disease Transmission
- Sound Wave Propagation
- Heat Diffusion
- Traffic Simulations

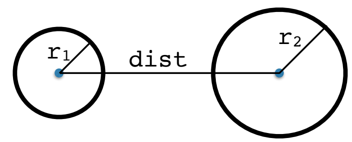
Topics for Today

- 1D Cellular Automata
- 2D Cellular Automata
- **Object Communication**

Object Communication

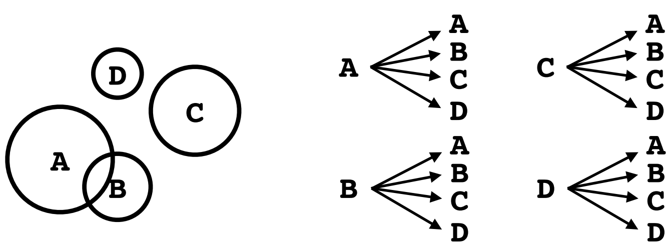
We can already tell if an object and the mouse interact, but what about multiple objects?

For circles, it's nice and easy, but for other objects it can be very complicated.



More Nested Loops

If I have many bubbles, I need to check every pairwise combination of bubbles.



So, we pick a bubble, loop though every other bubble, and check if they touch.