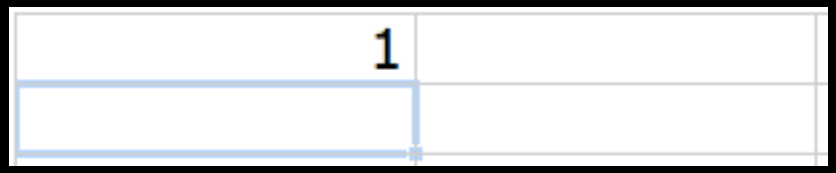
10.2 Diffusion and Cellular Automata

Simulating Motion: Cellular Automata

- If all we have to work with is a grid of cells (spreadsheet), how can we simulate a random walk?

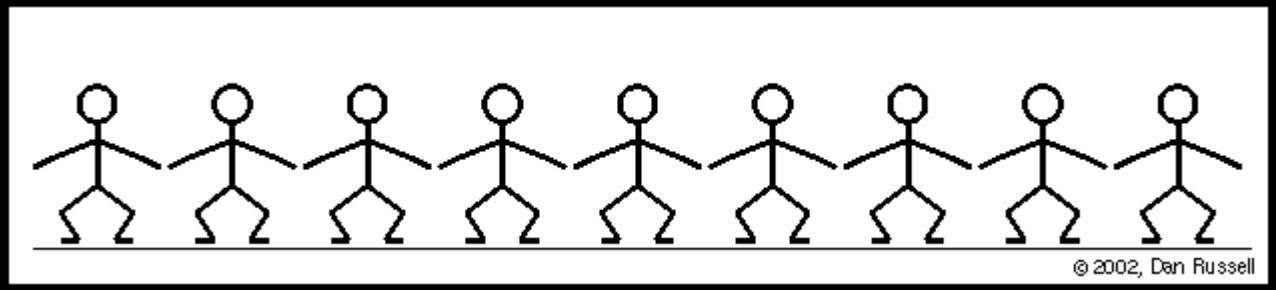
- Moving a value from one cell to an adjacent cell is simulated by changing the contents of the cells over time:



time *t*

time *t* +1

The Wave

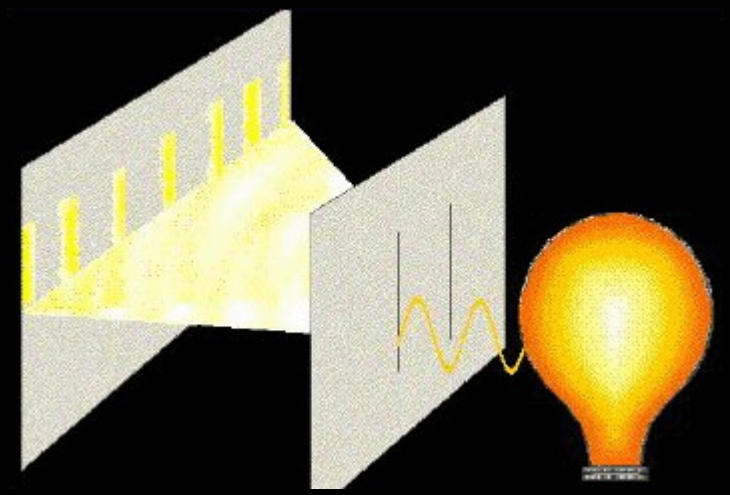


Aside: The Wave/Particle Duality

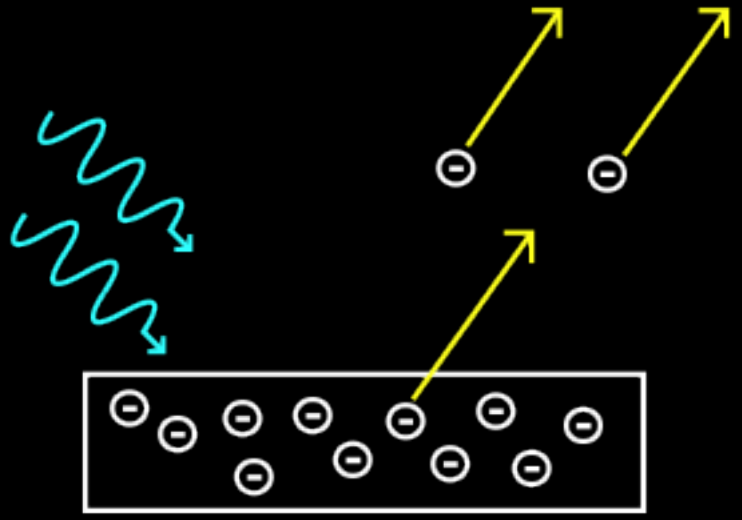
- We tend to think of “true” motion as movement of an object from one location to another, and the wave phenomenon as an illusion of motion.

- But at the most fundamental level, energy and matter behave like both a particle (object) *and* a wave….

Aside: The Wave/Particle Duality



Double-slit experiment (Young 1801): Light must be a wave, because interference produces more than two bands on screen.



Photoelectric effect (Thomson 1899): Light must consist of particles, because light waves knock electrons out of a metal plate.

Cellular Automata

- A cellular automaton (plural automata) is a dynamical model discrete in space, time, and state (possible values, typically just 0 and 1).
- CA has a grid of cells, and a set of transition rules describing what happens to a cell based on its own state and the state of its neighbors.
- Simplest CA are one-dimensional….

1D Cellular Automata

Sample rule: Look at a cell and its left and right neighbors. Change the cell according to the following table:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

ExCellular Automata

- Each row is the state of the CA at a particular time step.

- First row has random values:

  = IF(RAND()>0.5, 1, 0)

- Subsequent rows have rule:

```
=IF( OR (AND( A1=1,B1=1,C1=1),
         AND (A1=1,B1=0,C1=0),
         AND (A1=0,B1=0,C1=0)),  0,  1)
```

ExCellular Automata

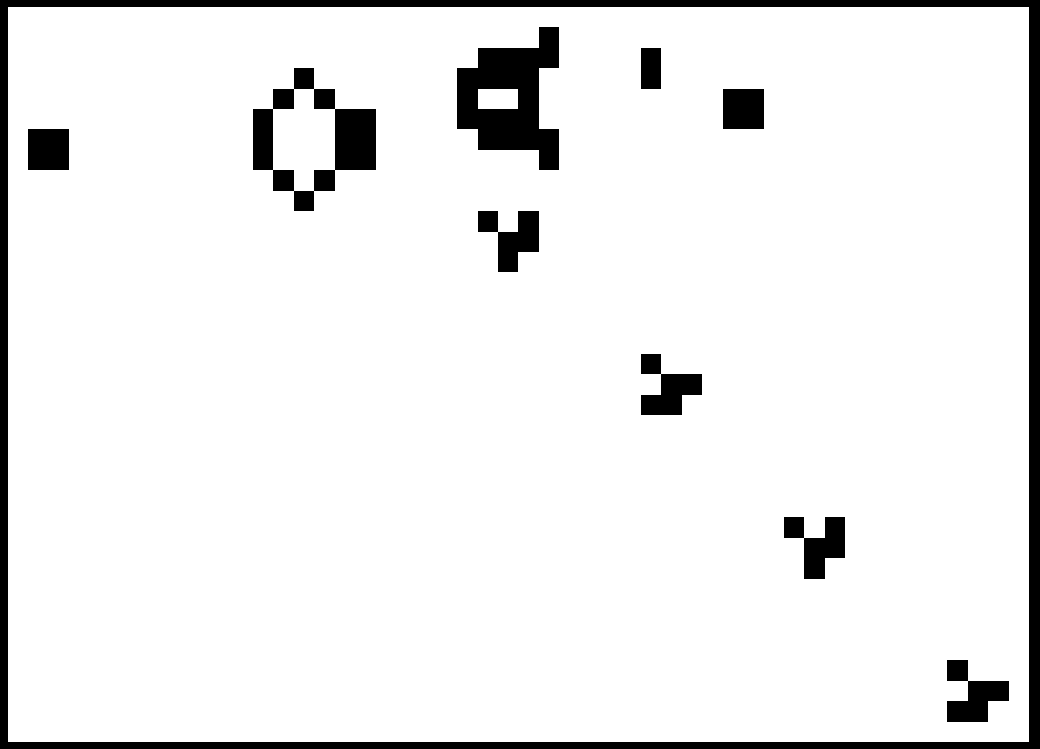
- Drag second row, second column rightward to next-to-last
- Keep first and last columns (boundary conditions) constant
- Drag second row down 60 or 70 rows
- Use Conditional Formatting / Color Scales to color cells based on values
- Zoom out to view whole display
- Hit F9 key to restart with new initial conditions
- Experiment with other rules:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0

Diffusion: Cellular Automata in Nature

Diffusion: Cellular Automata in Nature

2D Cellular Automata



Diffusion

- Diffusion refers to any process in which a substance or property spreads from one region to another: transfer of heat, spread of dye in a liquid, motion of ants outward from a nest, etc.

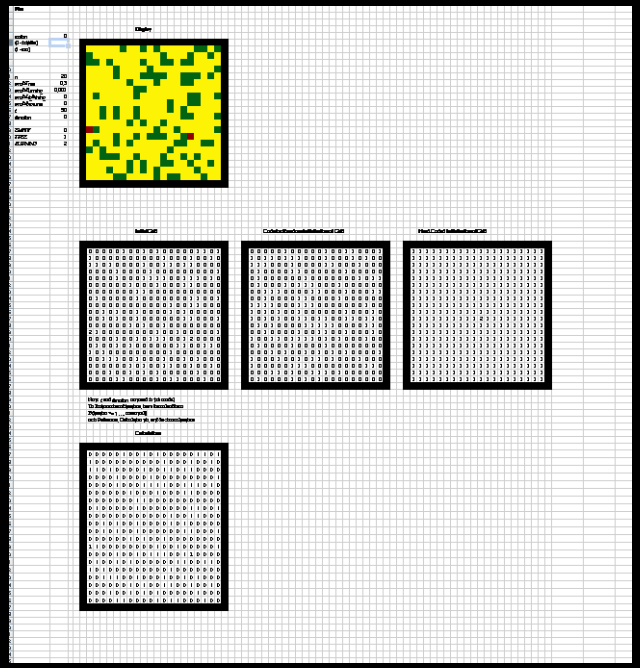
Diffusion has traditionally been modeled using differential equations / systems dynamics – averages over large regions of individuals.

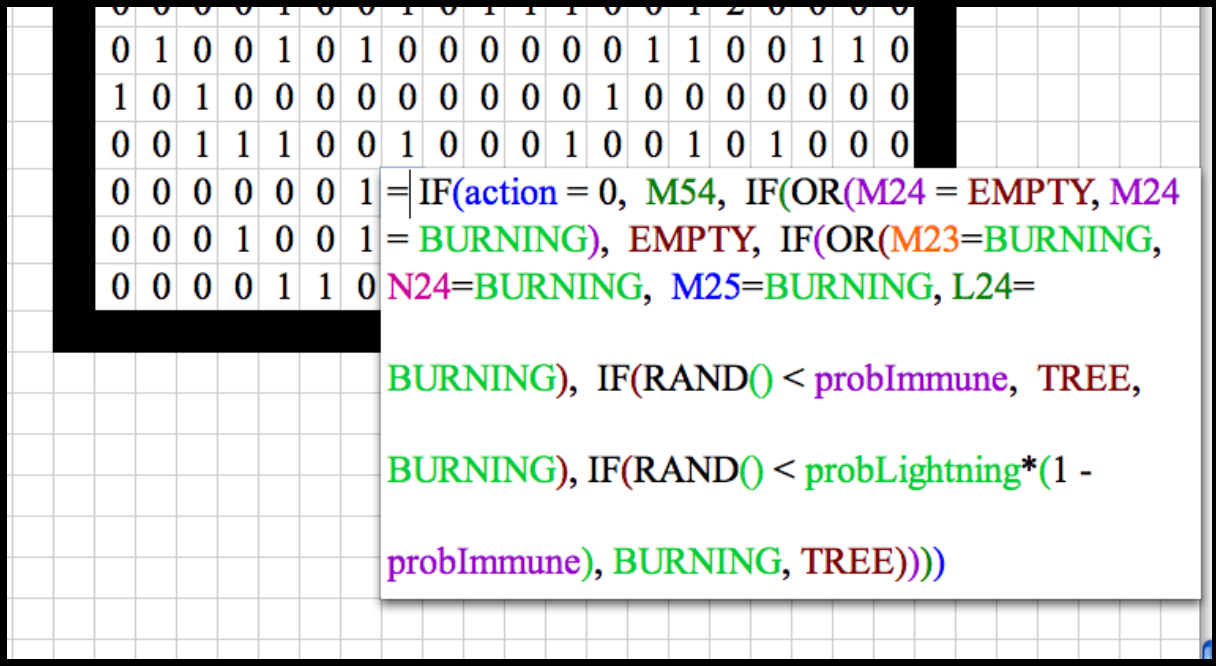
Modern advances in computing power allow us to use cellular automata to model diffusion at the level of individual ants, molecules, etc.

Diffusion & Cellular Automata

We will study diffusion via cellular automata.

With Excel, this becomes difficult:





Cellular Automata in Matlab

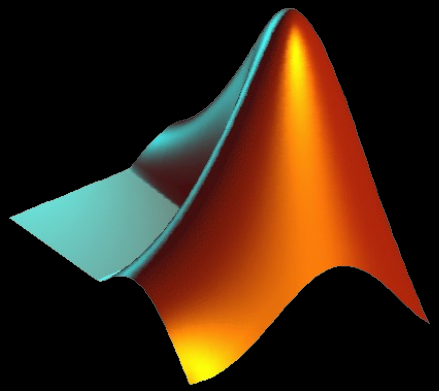
So we will use Matlab to study CA:

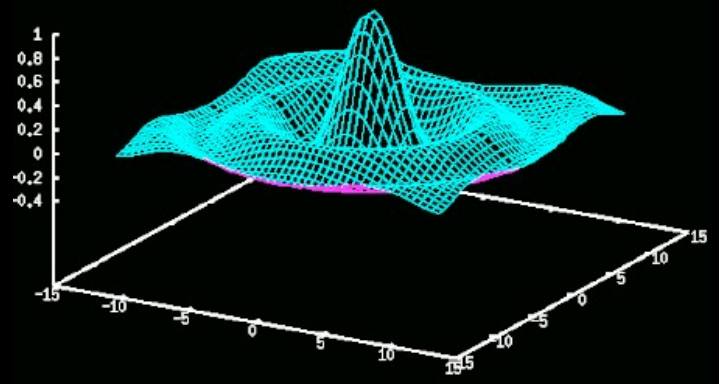
Industry standard

Performs operations on entire table at once (no click-and-drag)

Programs look much like pseudocode in textbook

Free, open-source version called Octave



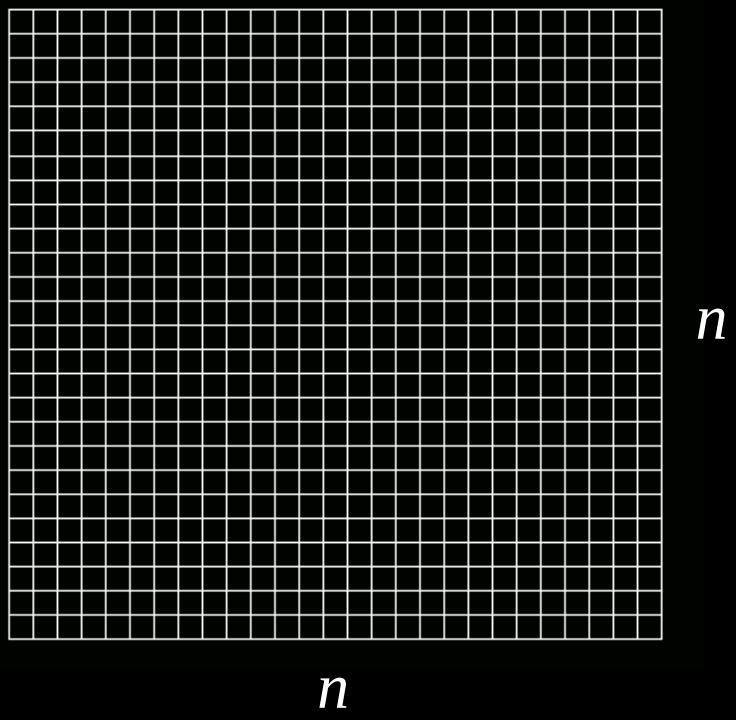


Application: Spreading of Fire

Initializing the System

"Forest" is an n x n grid (lattice) of cells.

Each cell is either empty (0) has a tree (1) or has a burning tree (2).



- *probTree* = probability of tree in cell
- *probBurning* = probability that the tree is burning

Cell Initialization Algorithm (Pseudocode)

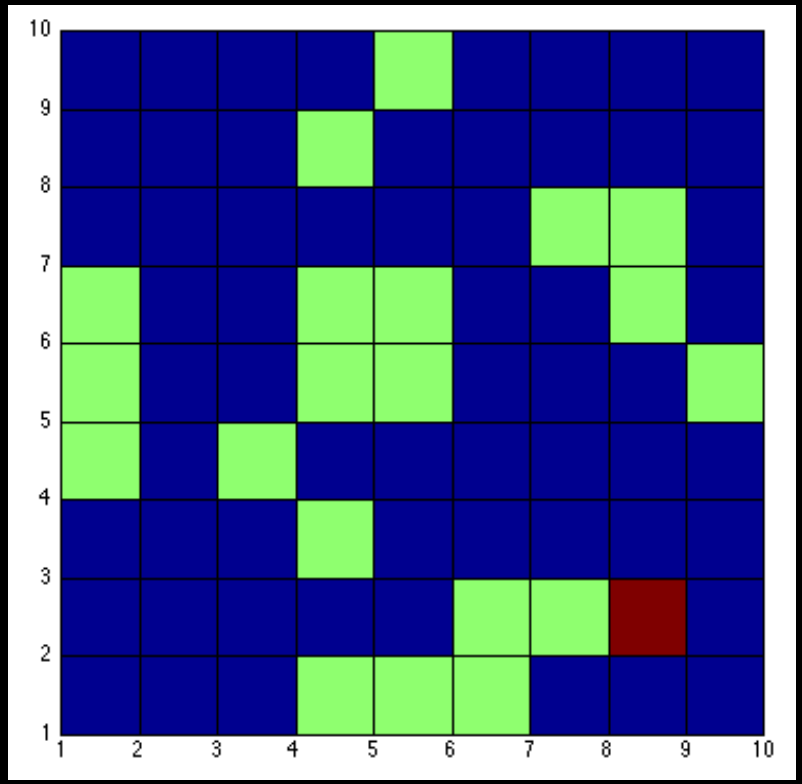
```
if a random number is less than probTree
    if another random number is less than probBurning
     assign 2 to the cell   // tree is burning
    else
     assign 1 to the cell   // tree is not burning
else
    assign 0 to the cell        // no tree
```

Cell Initialization (Matlab)

```
grid = zeros(n); % N x N grid of zeros

rand1 = rand(n);  % N x N random in [0,1)

rand2 = rand(n);

grid(rand1 < probTree) = 1;

grid(rand1 < probTree &  rand2 < probBurning) = 2;
```

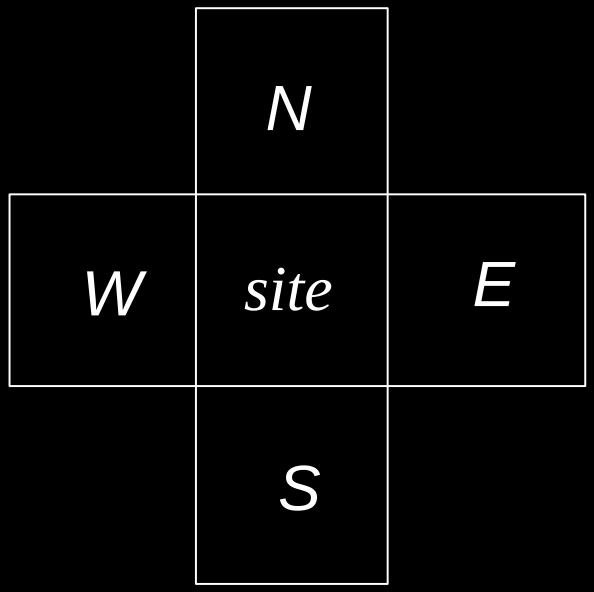
Cell Initialization (Matlab)

```
pcolor(grid)

axis square
```



Updating Rules

We determine a cell's new state at time $t + 1$ based on its current state (at time t) and the state of the neighbors in its von Neumann neighborhood (N, S, E, W):



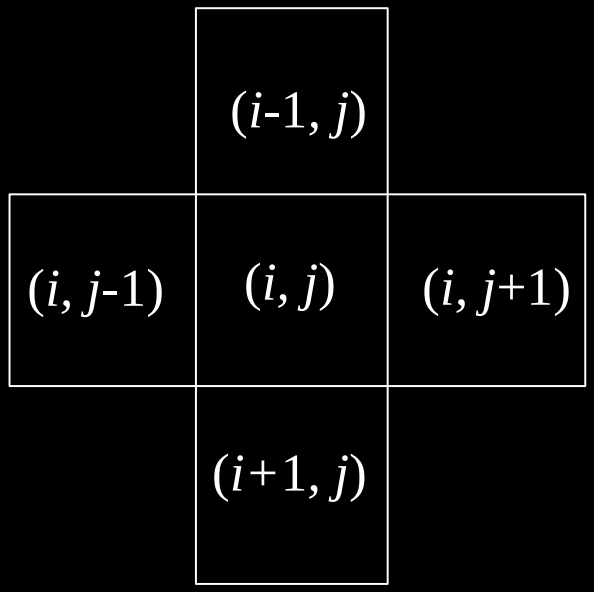
Updating Rules

- If cell is empty, it stays empty at $t + 1$
- If a cell has a burning tree, the tree is gone at $t + 1$
- If a tree is adjacent to a burning tree, it catches fire with a probability (1 – *probImmune*)
- A tree may also catch fire by being struck by lightning: *probLightning*

Applying a Function to Each Grid Point

- i indexes row; j indexes column

Problem: what to do at boundaries?

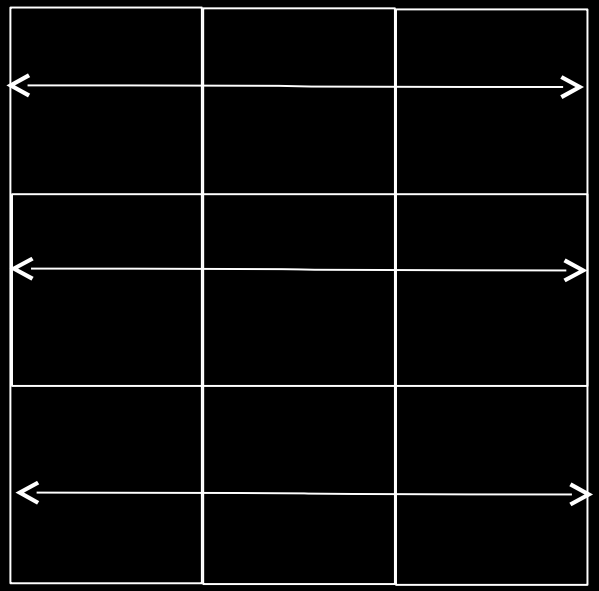


Absorbing Boundary Conditions

Cells beyond perimeter are inactive (nothing west of column 1, nothing east of column *n*, nothing north of row 1, nothing south of row *n*).

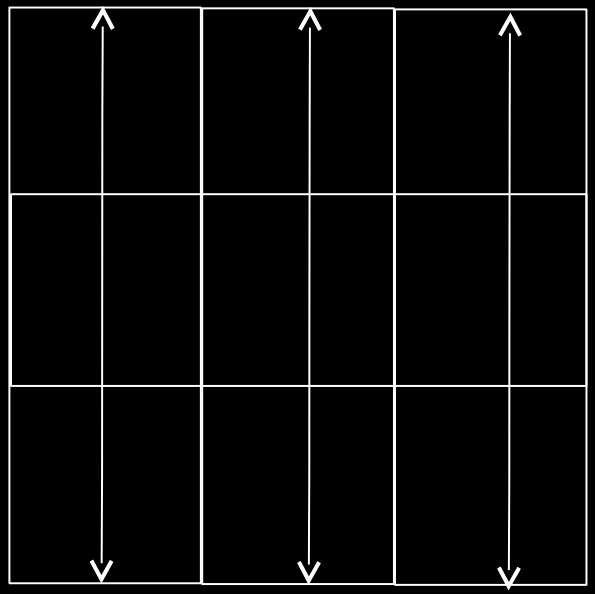
Periodic Boundary Conditions

Column 1 and column n are neighbors.



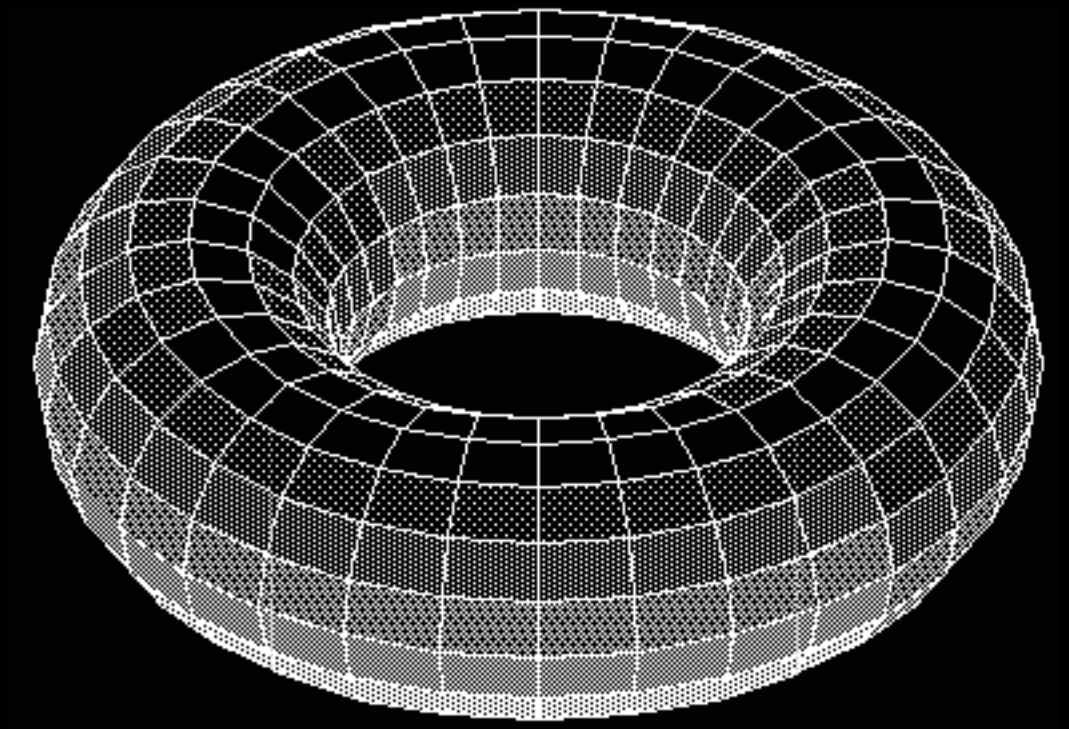
Periodic Boundary Conditions

Row 1 and row n are neighbors.



Periodic Boundary Conditions

This makes the space a torus.



Periodic Boundary Conditions In Matlab

Matlab's **`circshift`** function does this for us:

```
>> a = [1 2 3; 4 5 6; 7 8 9]

a =

     1     2     3
     4     5     6
     7     8     9

>> circshift(a, [0 1]) % W neighbors

ans =

     3     1     2
     6     4     5
     9     7     8
```

Translate Updating Rules to Matlab

- If cell is empty, it stays empty at $t + 1$

```
% no action necessary
```

Translate Updating Rules to Matlab

- If a cell has a burning tree, the tree is gone at $t + 1$

```
newgrid(oldgrid == 2) = 0;
```

Translate Updating Rules to Matlab

- If a tree is adjacent to a burning tree, it catches fire with a probability (1 – *probImmune*)

```
n = length(oldgrid); % length of one side
havetree = oldgrid == 1;
susceptible = rand(n) < (1 - probImmune);
nfire = circshift(oldgrid, [1 0])  == 2;
sfire = circshift(oldgrid, [-1 0]) == 2;
efire = circshift(oldgrid, [0 -1]) == 2;
wfire = circshift(oldgrid, [0  1]) == 2;
adjacent = nfire | sfire | efire | wfire;
newgrid(havetree & adjacent & susceptible) = 2;
```

Translate Updating Rules to Matlab

- A tree may also catch fire by being struck by lightning: *probLightning*

```
newgrid(havetree & (rand(n) < probLightning)) = 2;
```