Computer Programming Fundamentals

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/

QUIZ 4 TODAY

DUE 11AM TOMORROW (SATURDAY)

NO MAKE UP QUIZZES

ASSIGNMENT 6
DUE FRIDAY 11/19

questions?

1D CELLULAR AUTOMATA

WHERE WE ARE:
CODE WALK THROUGH

PUTING IT ALL TOGETHER

IN main

```java
public static void main(String[] args) {
    CellularAutomata1D CA = new CellularAutomata1D();
    CA.iterate(3);
}
```

```
   *
  * *
 *   *
* * * *
```

IN main

```java
public static void main(String[] args) {
    CellularAutomata1D CA = new CellularAutomata1D();
    CA.iterate(20);
}
```

```
                 *
                * *
               *   *
              * * * *
             *       *
            * *     * *
           *   *   *   *
          * * * * * * * *
         *               *
        * *             * *
       *   *           *   *
      * * * *         * * * *
     *       *       *       *
    * *     * *     * *     * *
   *   *   *   *   *   *   *   *
  * * * * * * * * * * * * * * * *
 *                               *
* *                             * *
```

IN main

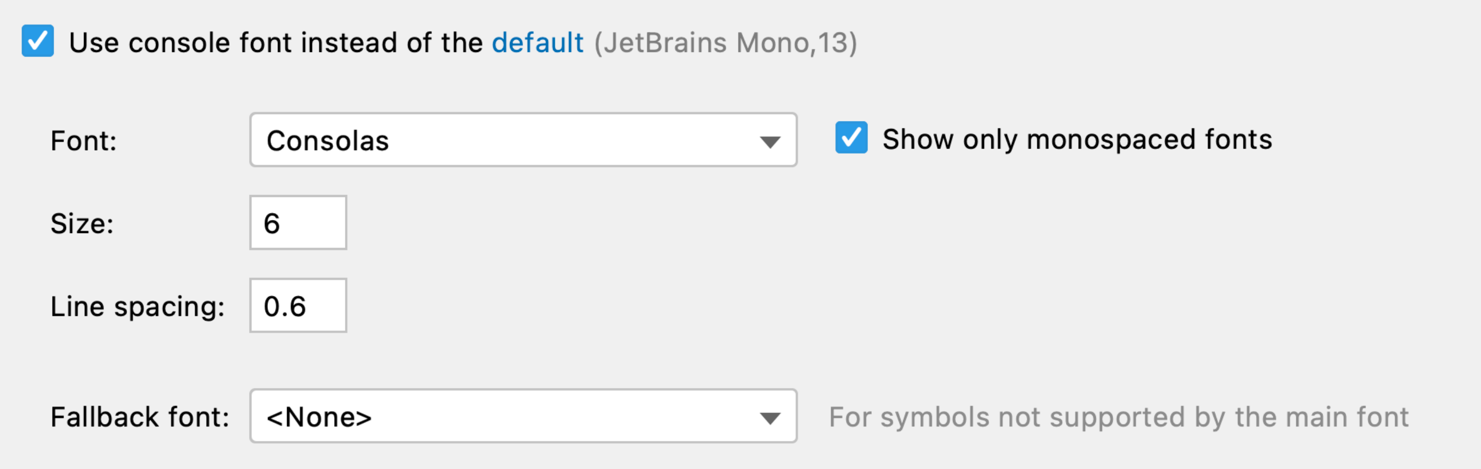
```java
public static void main(String[] args) {
    CellularAutomata1D CA = new CellularAutomata1D();
    CA.iterate(35);
}
```

```
                               *
                              * *
                             *   *
                            * * * *
                           *       *
                          * *     * *
                         *   *   *   *
                        * * * * * * * *
                       *               *
                      * *             * *
                     *   *           *   *
                    * * * *         * * * *
                   *       *       *       *
                  * *     * *     * *     * *
                 *   *   *   *   *   *   *   *
                * * * * * * * * * * * * * * * *
               *                               *
              * *                             * *
             *   *                           *   *
            * * * *                         * * * *
           *       *                       *       *
          * *     * *                     * *     * *
         *   *   *   *                   *   *   *   *
        * * * * * * * *                 * * * * * * * *
       *               *               *               *
      * *             * *             * *             * *
     *   *           *   *           *   *           *   *
    * * * *         * * * *         * * * *         * * * *
   *       *       *       *       *       *       *       *
  * *     * *     * *     * *     * *     * *     * *     * *
 *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

questions?

EDIT YOUR CONSOLE FONT IN IntelliJ FOR BETTER VISUALIZING

IntelliJ—>Preferences
Editor—>Color Scheme—>Console Font



☑ Use console font instead of the default (JetBrains Mono,13)

Font: Consolas ☑ Show only monospaced fonts

Size: 6

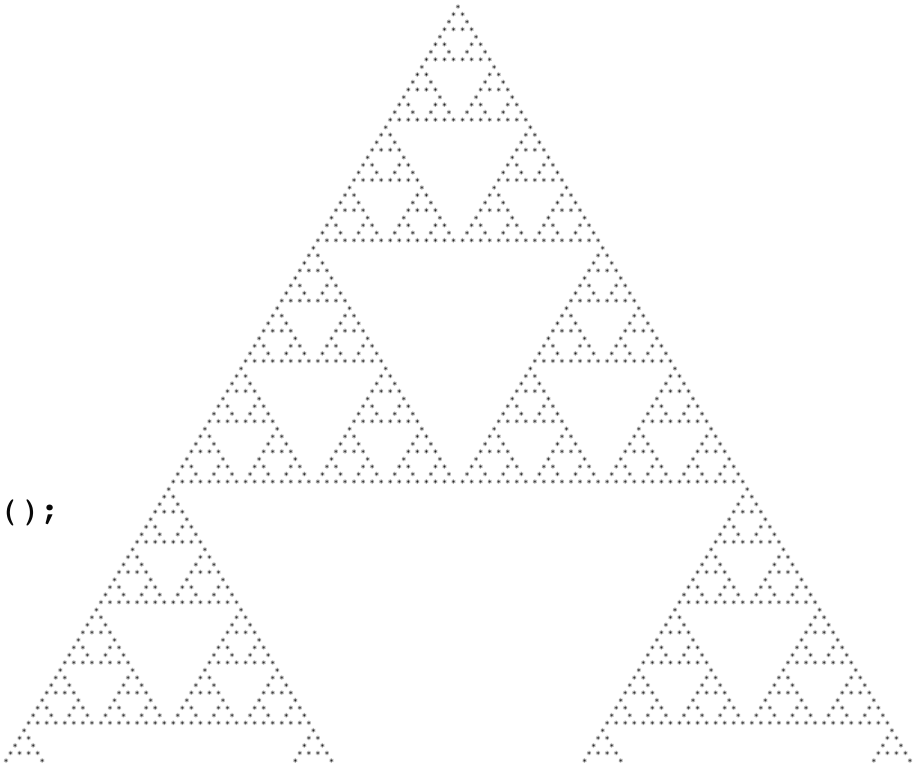
Line spacing: 0.6

Fallback font: <None> For symbols not supported by the main font

CHANGE SIZE & ITERATIONS

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    currentStates[size/2] = ALIVE;
}


public static void main(String[] args) {
    CellularAutomata1D CA = new CellularAutomata1D();
    CA.iterate(100);
}
```



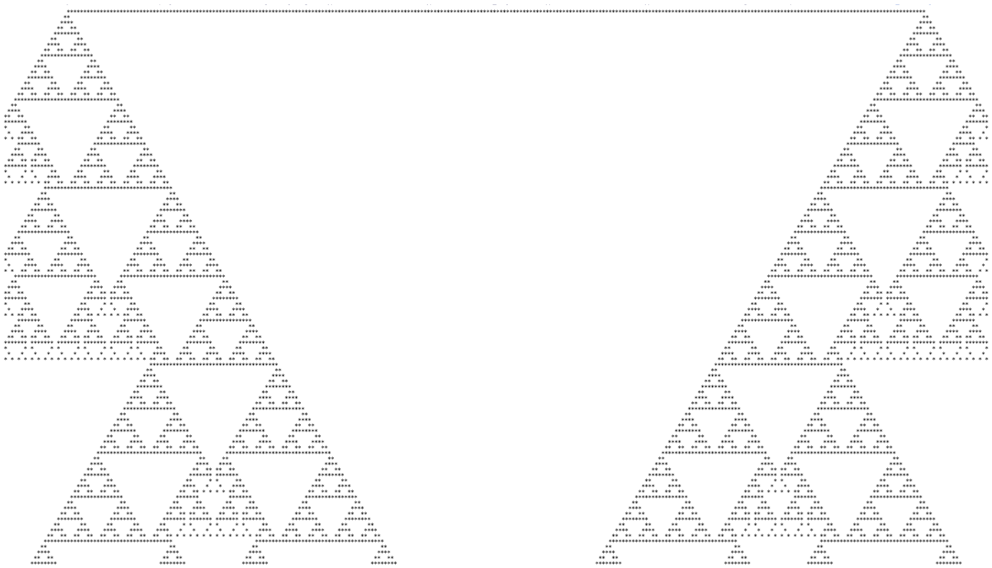
DIFFERENT STARTING CONDITIONS

IN CONSTRUCTOR

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    currentStates[size/2] = ALIVE;
}
```

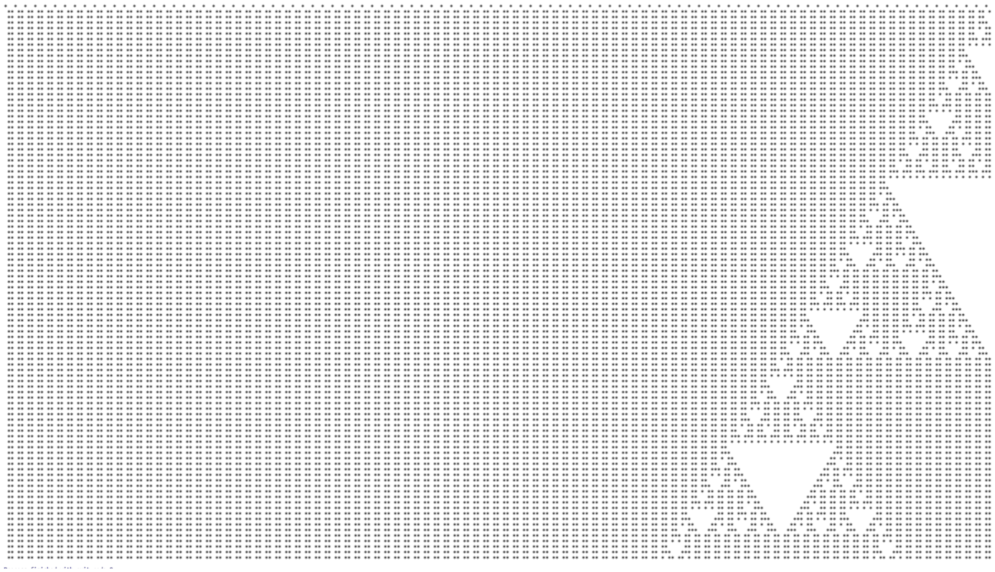
DIFFERENT STARTING STATES

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    for (int i=20;i<size-20;i++) {
        currentStates[i] = ALIVE;
    }
}
```



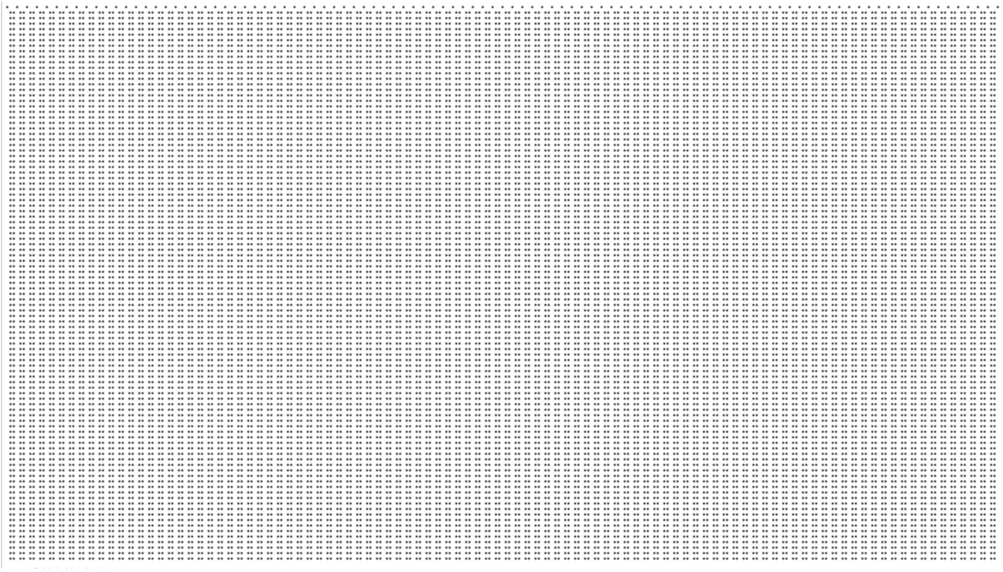
DIFFERENT STARTING STATES

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    for (int i=0;i<size;i=i+3) {
        currentStates[i] = ALIVE;
    }
}
```



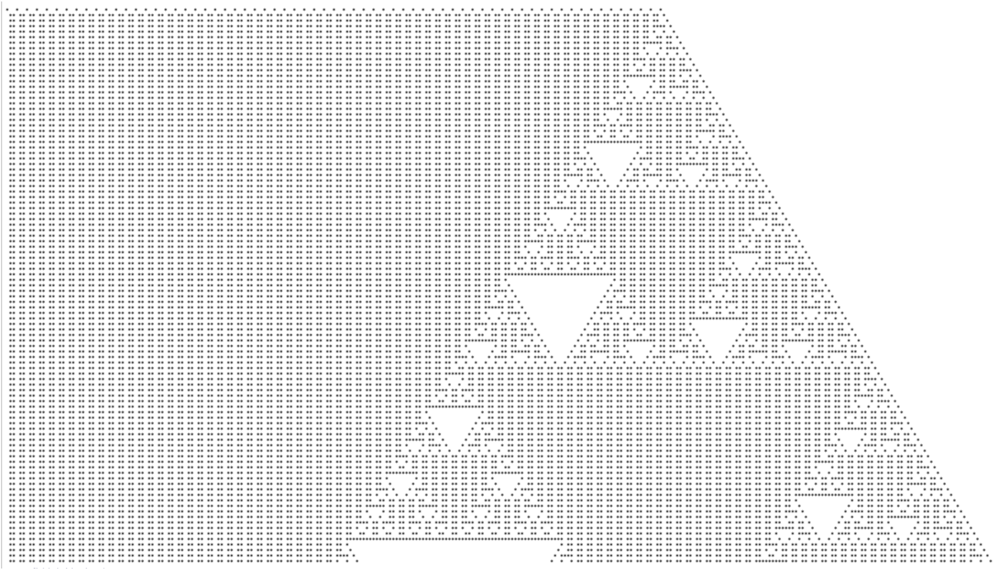
DIFFERENT STARTING STATES

```
CellularAutomata1D() {
    size = 301;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    for (int i=0;i<size;i=i+3) {
        currentStates[i] = ALIVE;
    }
}
```



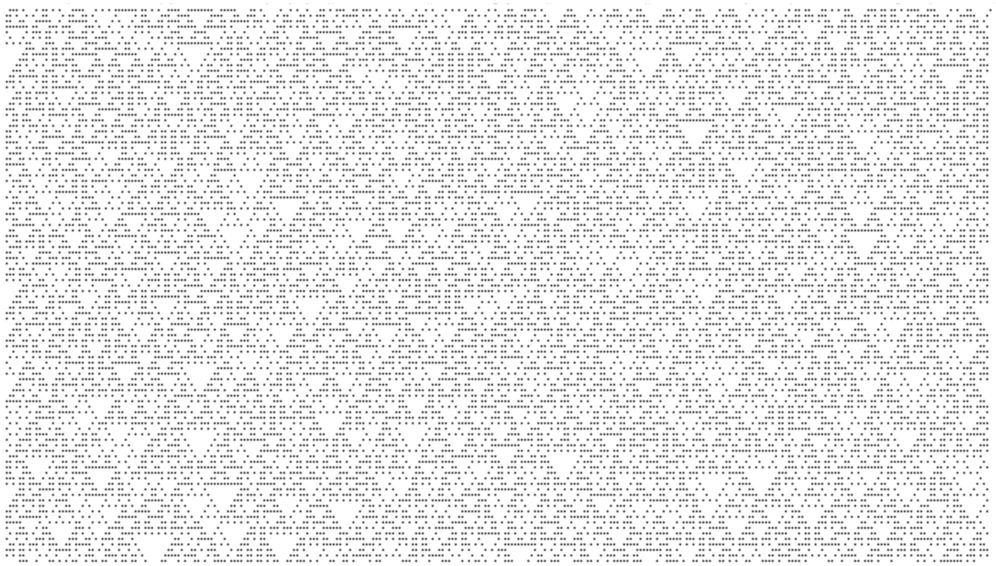
DIFFERENT STARTING STATES

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i] = DEAD;
    }
    for (int i=0;i<size-100;i=i+3) {
        currentStates[i] = ALIVE;
    }
}
```

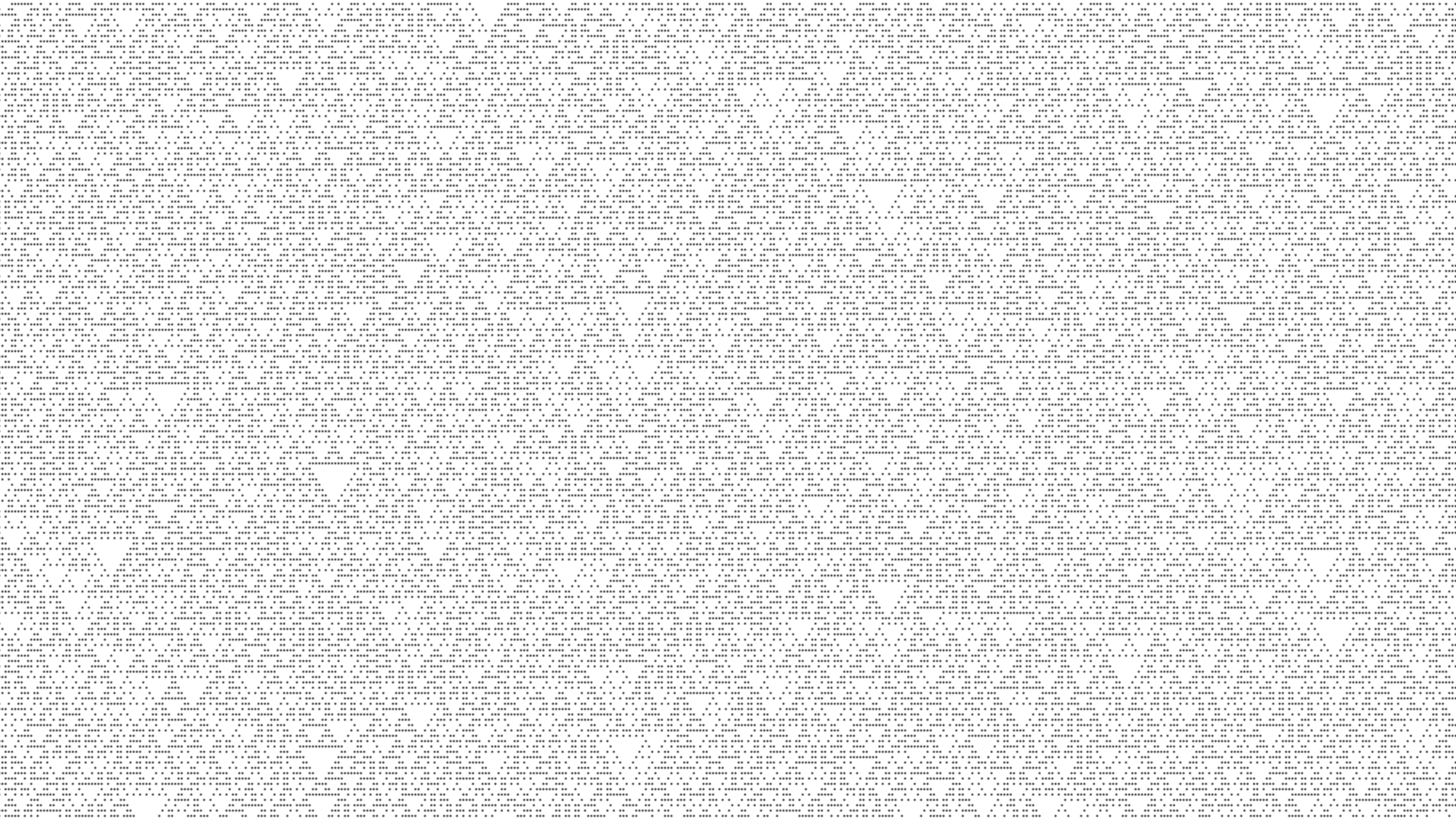


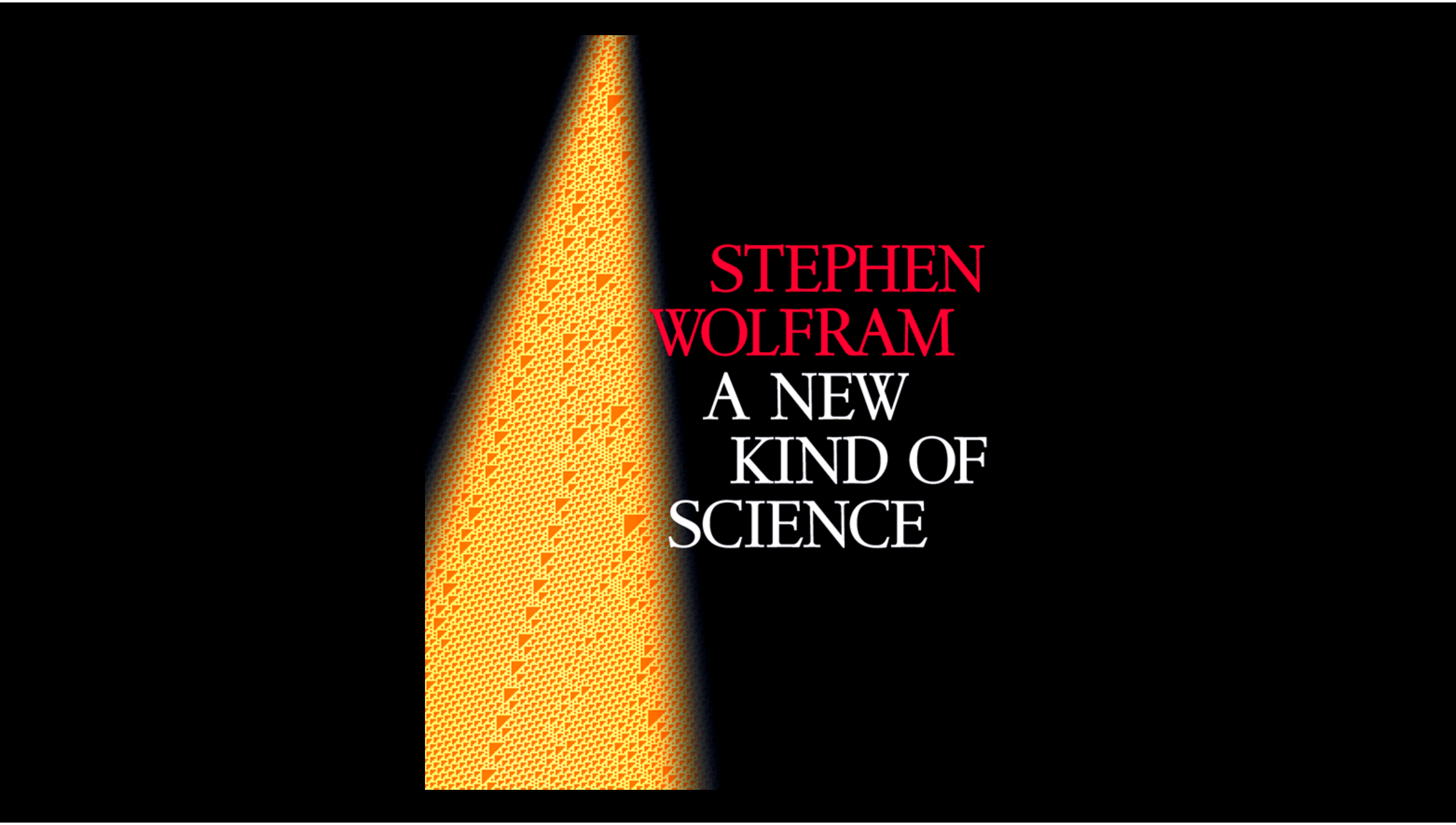
DIFFERENT STARTING STATES

```
CellularAutomata1D() {
    size = 300;
    currentStates = new int[size];
    nextStates = new int[size];
    for (int i=0;i<size;i++) {
        currentStates[i]=(int)(Math.random()*2);
    }
}
```



size = 500
iterations = 300





STEPHEN
WOLFRAM

A NEW
KIND OF
SCIENCE

ELEMENTARY CELLULAR AUTOMATA

https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

OUR RULE: 90

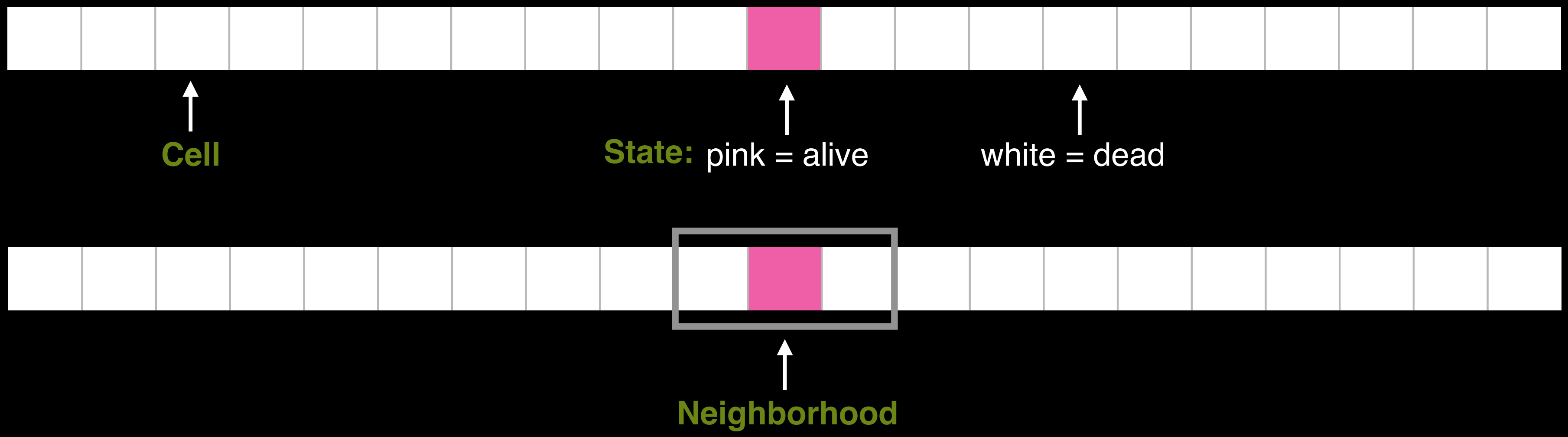
CELLULAR AUTOMATA IN NATURE

source internet - www.aquaportail.com

questions?

2D CELLULAR AUTOMTATA

A 1D CELLULAR AUTOMATON



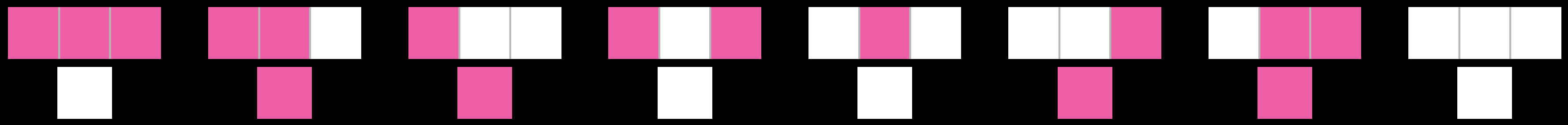
Cell

State: pink = alive

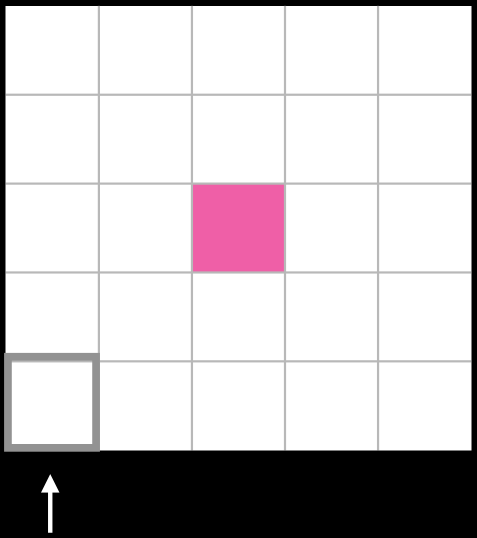
white = dead

Neighborhood

Rule:

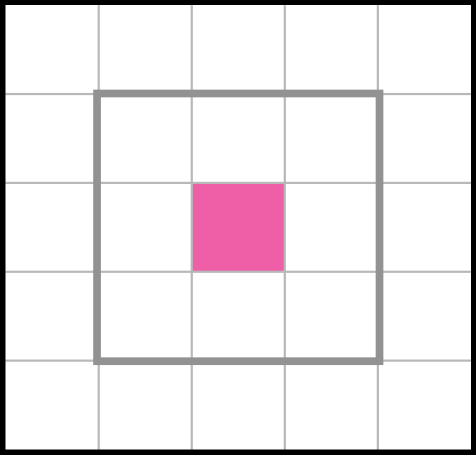


A 2D CELLULAR AUTOMATON

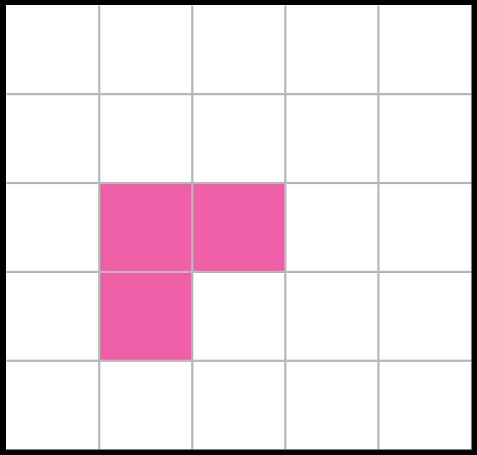


Cell

State: pink = alive
white = dead



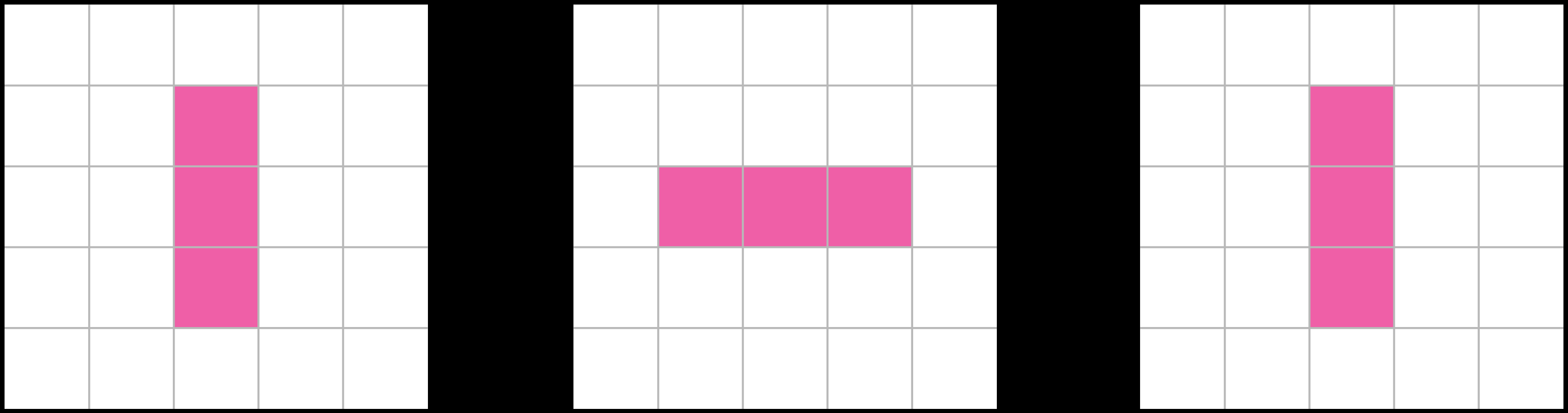
Neighborhood



Game of Life Rule:
if alive and 2 or 3 neighbors are alive, stay alive

if dead and 3 neighbors are alive, come alive

A 2D CELLULAR AUTOMATON



questions?

LET’S BUILD ONE

CREATE A CellularAutomata2D.java CLASS

2D CELLULAR AUTOMATA IN CODE

```
public class CellularAutomata2D {

}
```

DOWNLOAD BasicPanel.jar
ADD AS A LIBRARY

EXTEND BASIC PANEL

```
public class CellularAutomata2D extends BasicPanel {

}
```

INSTANCE VARIABLES

```
public class CellularAutomata2D extends BasicPanel {
    int size;
    int[][] currentStates;
    int[][] nextStates;
    final int ALIVE = 1;
    final int DEAD = 0;
}
```

CONSTRUCTOR

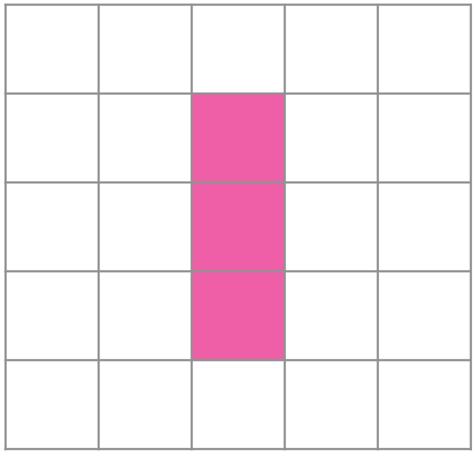
```
public class CellularAutomata2D extends BasicPanel {
    int size;
    int[][] currentStates;
    int[][] nextStates;
    final int ALIVE = 1;
    final int DEAD = 0;

    CellularAutomata2D() {
        size = 50;
        run = true;
        currentStates = new int[size][size];
        nextStates = new int[size][size];
   }
}
```

CONSTRUCTOR

```
public class CellularAutomata2D extends BasicPanel {
    int size;
    int[][] currentStates;
    int[][] nextStates;
    final int ALIVE = 1;
    final int DEAD = 0;

    CellularAutomata2D() {
        size = 50;
        run = true;
        currentStates = new int[size][size];
        nextStates = new int[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                currentStates[i][j] = DEAD;
                nextStates[i][j] = DEAD;
            }
        }
        currentStates[size/2-1][size/2] = ALIVE;
        currentStates[size/2][size/2] = ALIVE;
        currentStates[size/2+1][size/2] = ALIVE;
    }
}
```



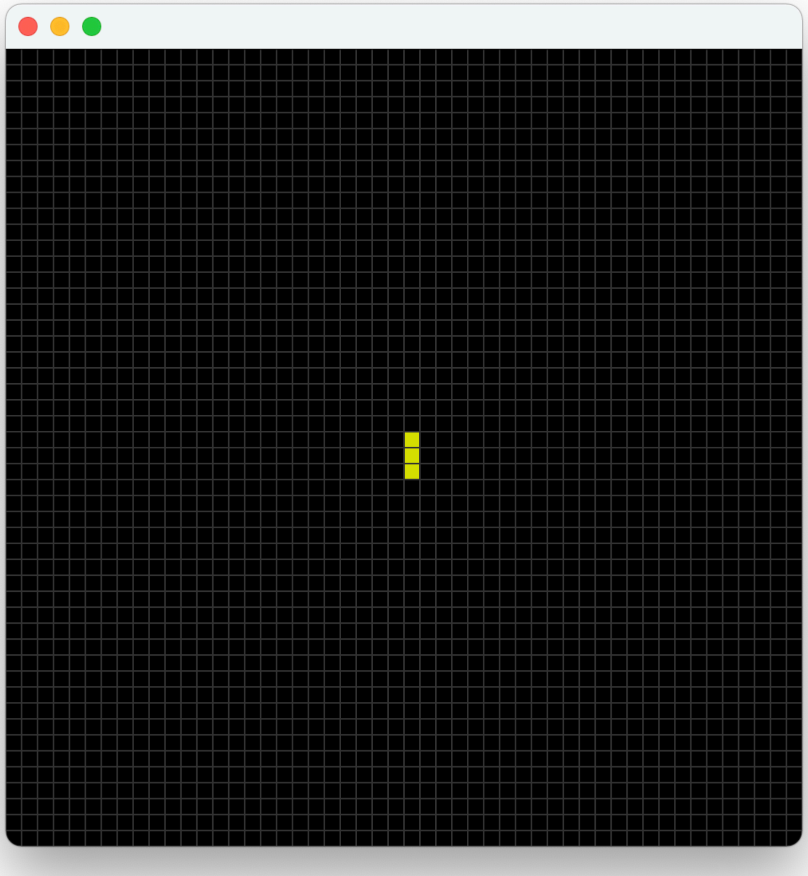
questions?

DISPLAYING THE CA

DISPLAYING THE CA

- Display on screen
- Grid is size x size cells
- Square for each cell of dimension CELLSIZE
- ALIVE cells drawn in ALIVE_COLOR (green)
- DEAD cells drawn in DEAD_COLOR (black)
- Grid drawn in GRID_COLOR

- What is size of window?
- width = CELLSIZE * size
- height = CELLSIZE * size



ADD SOME VARIABLES FOR DISPLAY

```
public class CellularAutomata2D extends BasicPanel {
    int size;
    int[][] currentStates;
    int[][] nextStates;
    final int ALIVE = 1;
    final int DEAD = 0;
    final int CELLSIZE = 10;
    final Color ALIVE_COLOR = new Color(219, 224, 4);
    final Color DEAD_COLOR = Color.BLACK;
    final Color GRID_COLOR = new Color(50,50,50);
```

CONSTRUCTOR

```java
public class CellularAutomata2D extends BasicPanel {
    int size;
    int[][] currentStates;
    int[][] nextStates;
    final int ALIVE = 1;
    final int DEAD = 0;
    final int CELLSIZE = 10;
    final Color ALIVE_COLOR = new Color(219, 224, 4);
    final Color DEAD_COLOR = Color.BLACK;
    final Color GRID_COLOR = new Color(50,50,50);

    CellularAutomata2D() {
        size = 50;
        setSize(size * CELLSIZE, size * CELLSIZE);
        currentStates = new int[size][size];
        nextStates = new int[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                currentStates[i][j] = DEAD;
                nextStates[i][j] = DEAD;
            }
        }
        currentStates[size/2-1][size/2] = ALIVE;
        currentStates[size/2][size/2] = ALIVE;
        currentStates[size/2+1][size/2] = ALIVE;
    }
```

set the size of the panel/window

questions?

ADD A DISPLAY METHOD

```
void displayCurrentStates(Graphics g) {
    for (int i=0;i<size;i++) {
        for (int j = 0; j < size; j++) {

       }
    }
}
```

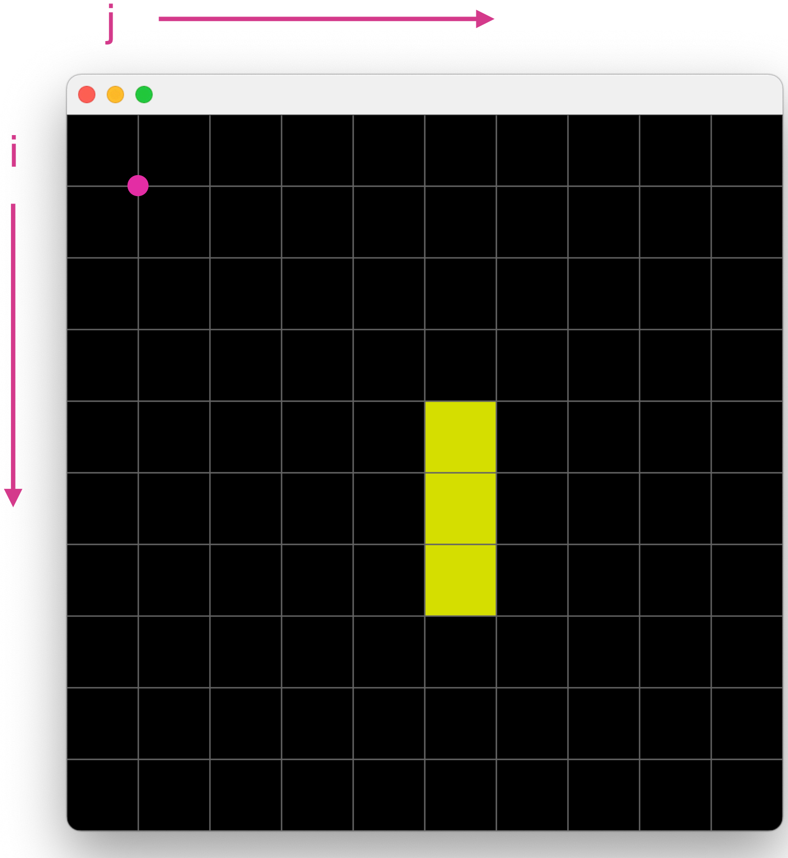
loop through 2D array

ADD A DISPLAY METHOD

```
void displayCurrentStates(Graphics g) {
    for (int i=0;i<size;i++) {
        for (int j = 0; j < size; j++) {
            if (currentStates[i][j] == ALIVE) {
                g.setColor(ALIVE_COLOR);
            }
            else {
                g.setColor(DEAD_COLOR);
            }
        }
    }
}
```

set color depending on cell state

DRAW RECTANGLE FOR EACH CELL



- Grid is size x size cells
- Square for each cell of dimension CELLSIZE

- What is x coordinate for each cell?
- Using variables and constants?
- j*CELLSIZE

- What is y coordinate for each cell?
- Using variables and constants?
- i*CELLSIZE

DRAW RECTANGLES

```
void displayCurrentStates(Graphics g) {
    for (int i=0;i<size;i++) {
        for (int j = 0; j < size; j++) {
            if (currentStates[i][j] == ALIVE) {
                g.setColor(ALIVE_COLOR);
            }
            else {
                g.setColor(DEAD_COLOR);
            }
            g.fillRect(j * CELLSIZE, i * CELLSIZE, CELLSIZE, CELLSIZE);
        }
    }
}
```