EECS 1710
Programming for Digital Media

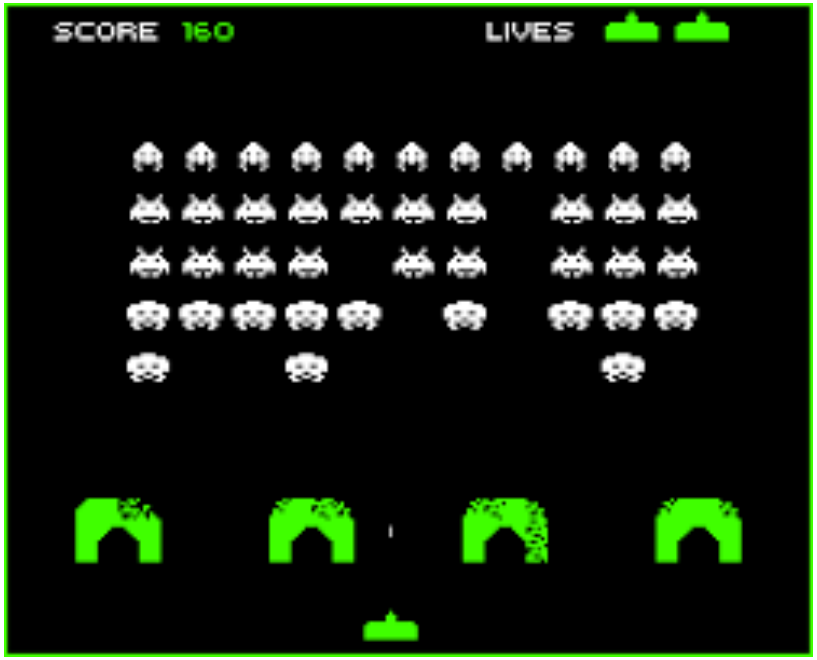
Lecture 12b :: Arrays [3]

multi-dimensional arrays

YORK UNIVERSITÉ UNIVERSITY

Arrays

- Very useful for storing/managing ***state*** (data) in a program
  - Examples:
    - a set of number guesses (number guessing game)
    - the letters in a hangman game
    - a set of shapes/lines drawn to the screen
    - inventory (items collected in a RPG = role playing game)
    - where you are currently on a game board
    - storing moves made or last things drawn (for undo)



Trivia question:
What games are these?
What might relate to *state* here?

Inventory (as an array)

- Common way to store a collection of “things” or “items” that may be used in a game context or similar, is to store them in an array
- Imagine we have a character within a game that can “collect” items (to be used in later parts of the game)
- We will store these in the character’s “inventory”
  - The inventory will be an array of String variables – each entry holding a String “description” of the item

Inventory example

```
final int MAX_ITEMS = 10;
String [] inventory = new String[MAX_ITEMS];
int numItems = 0;


inventory[numItems++] = “banana”;
inventory[numItems++] = “stick”;
inventory[numItems++] = “BFG”;
inventory[numItems++] = “abomb”;
inventory[numItems++] = “magic potion”

// output inventory
println(“You currently have ” + numItems + “ items:”);
for (int i=0; i<numItems; i++) {
        println(inventory[i]);
}
```

Audio & Images (as Arrays)

- Important area of scientific computing:

  Digital Signal Processing (DSP)

- We think of Audio and Images as “signals”
  - Audio: 1D (sound samples over time)
  - Images: 2D (pixel/colour samples over space)

Recap

- Declare and Initialize array (primitives)

```
int[] numbers = { 1, 2, 3 };
double[] decimals = { 1.1, 3.2, -4.842 };
```

- Declare, then initialize array separately (primitives)

```
int[] numbers;
numbers = new int[3];
for (int i=1; i<=3; i++ ) { numbers[i] = i; }
```

- Declare and initialize (non-primitive array)

```
String[] names = { "joe", "jane", "bob" };
String[] words = new String[3];
for (int i=0; i<3; i++ ) { words[i] = names[i]; }
```

YORK UNIVERSITÉ UNIVERSITY

Audio & Images, as Arrays

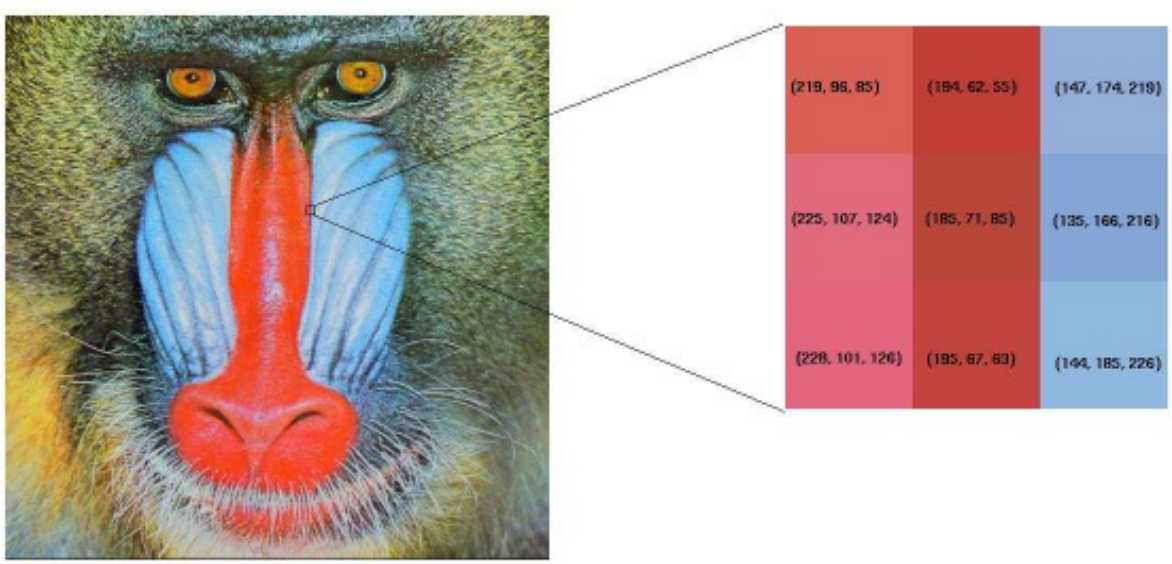
- Audio: 1D (sound samples over time)

1.4	**3.5**	**12**	**4**	**0.6**	**-3.5**	**-10.3**	**…**

1.4	3.5	12	4	0.6	-3.5	-10.3	…

t

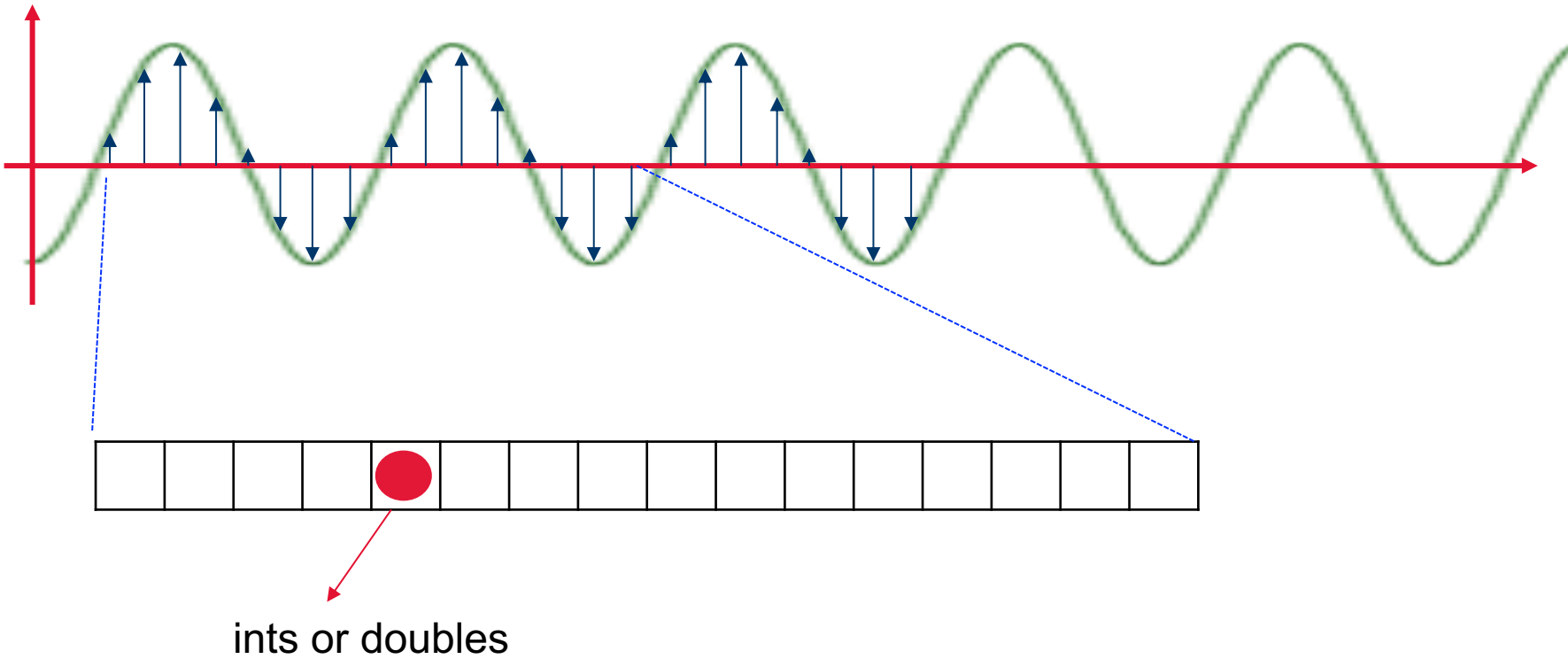
- Images: 2D (pixel/colour samples over space)



x

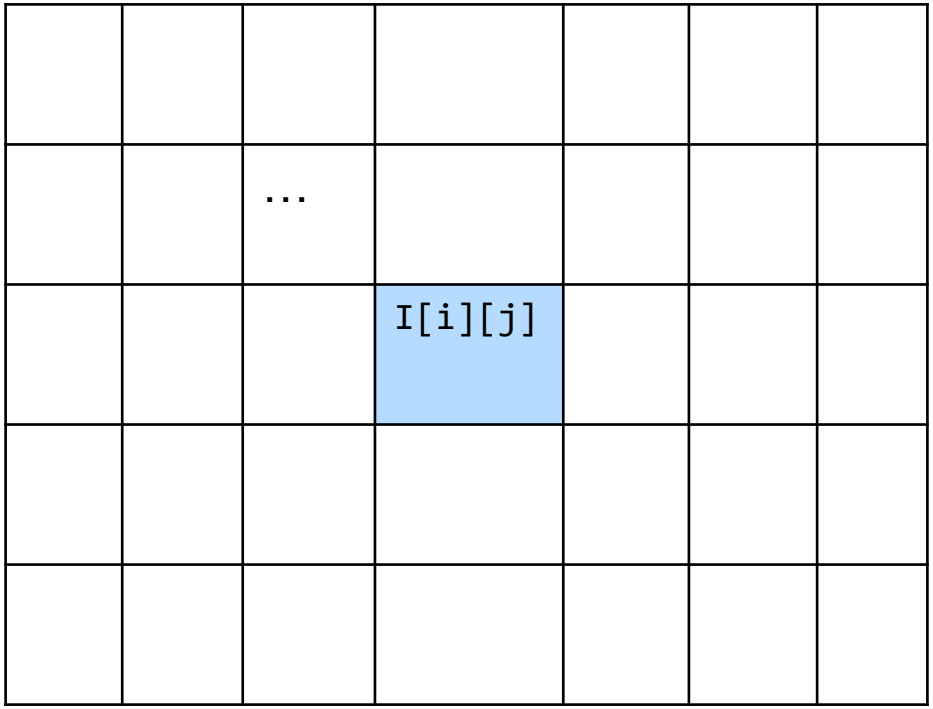
y

Digitized Sound (1D array of voltages)



Digitized Images (2D array of “colours”)

```
int [][] I ;
```



We will look at 2D arrays in more depth later (when we discuss pixel arrays – when working with Images

Multidimensional Arrays (2D +)

```
String[][] names = {
            {"Mr. ", "Mrs. ", "Ms. "},
            {"Smith", "Jones"}
};


// Mr. Smith
println(names[0][0] + names[1][0]);


// Ms. Jones
println(names[0][2] + names[1][1]);
```

2D Integer Array

- Declaring

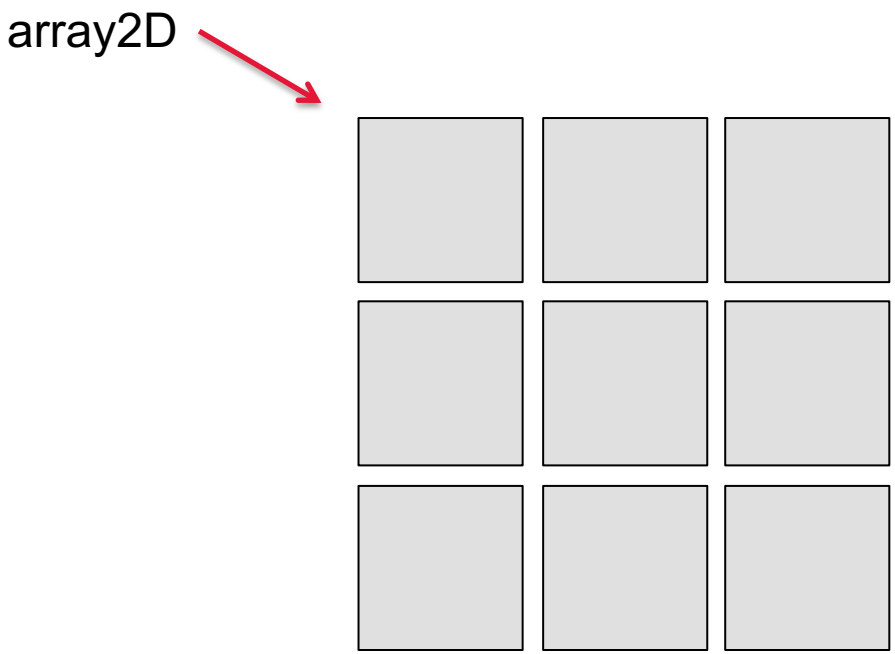
```
final int M = 3;
final int N = 3;

int [][] array2D = new int[M][N];    // empty size MxN
                                     // initializes all
                                     // values to 0


int [][] array2D = {                 // initialized with
                      { 1, 2, 3},    // specific values
                      { 4, 5, 6},
                      { 7, 8, 9}
                    };
```

2D Integer Array

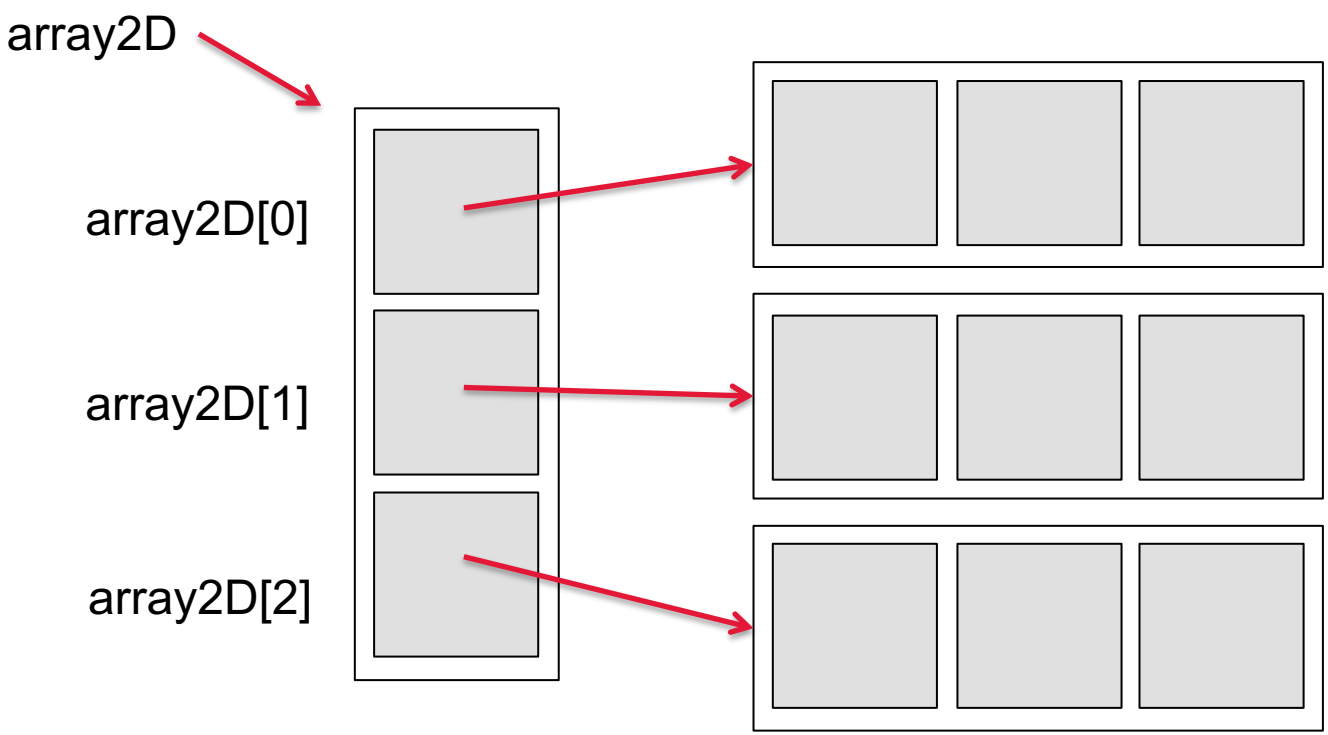
- Finding dimensions?

```
// …

// output = ??
println("array2D.length    = " + array2D.length);
println("array2D[0].length = " + array2D[0].length);
println("array2D[1].length = " + array2D[1].length);


// ??
println("array2D[0][O].length = " + array2D[0][0].length);
```

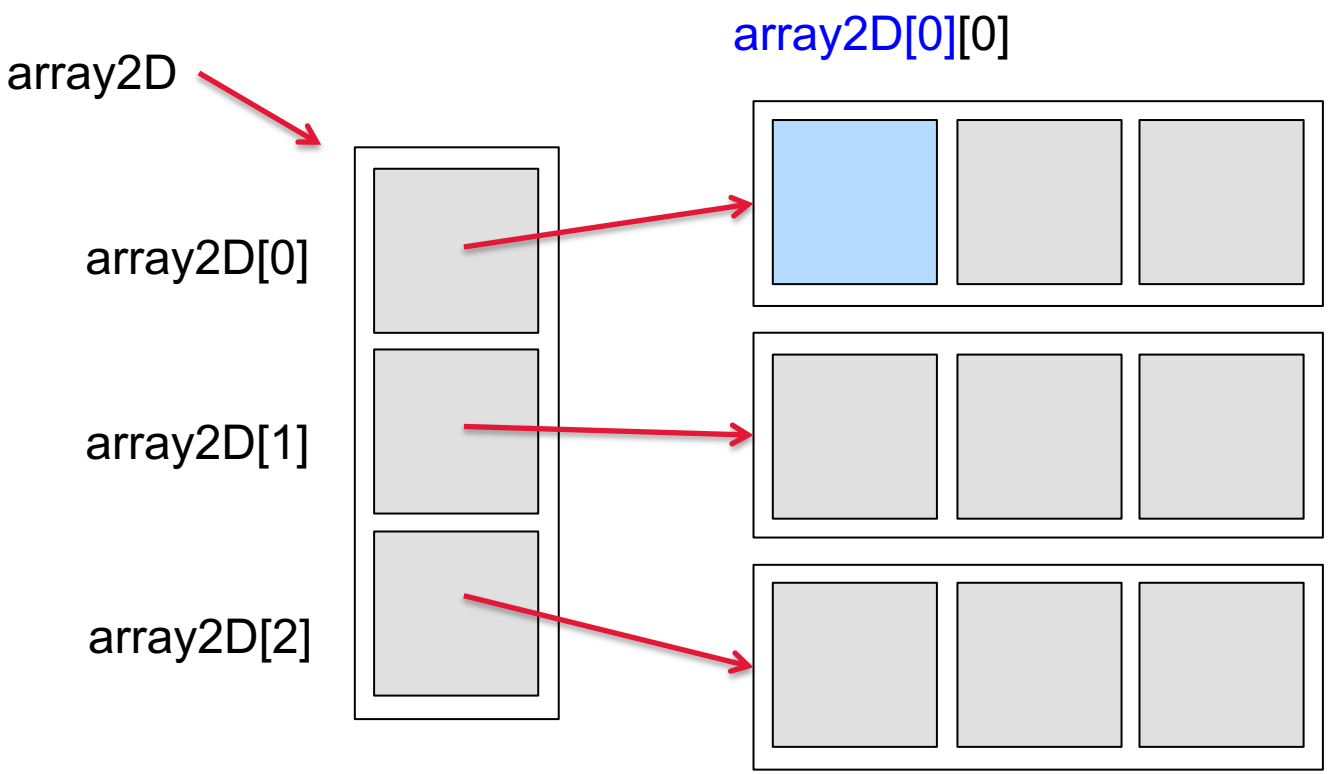
2D Array – in memory



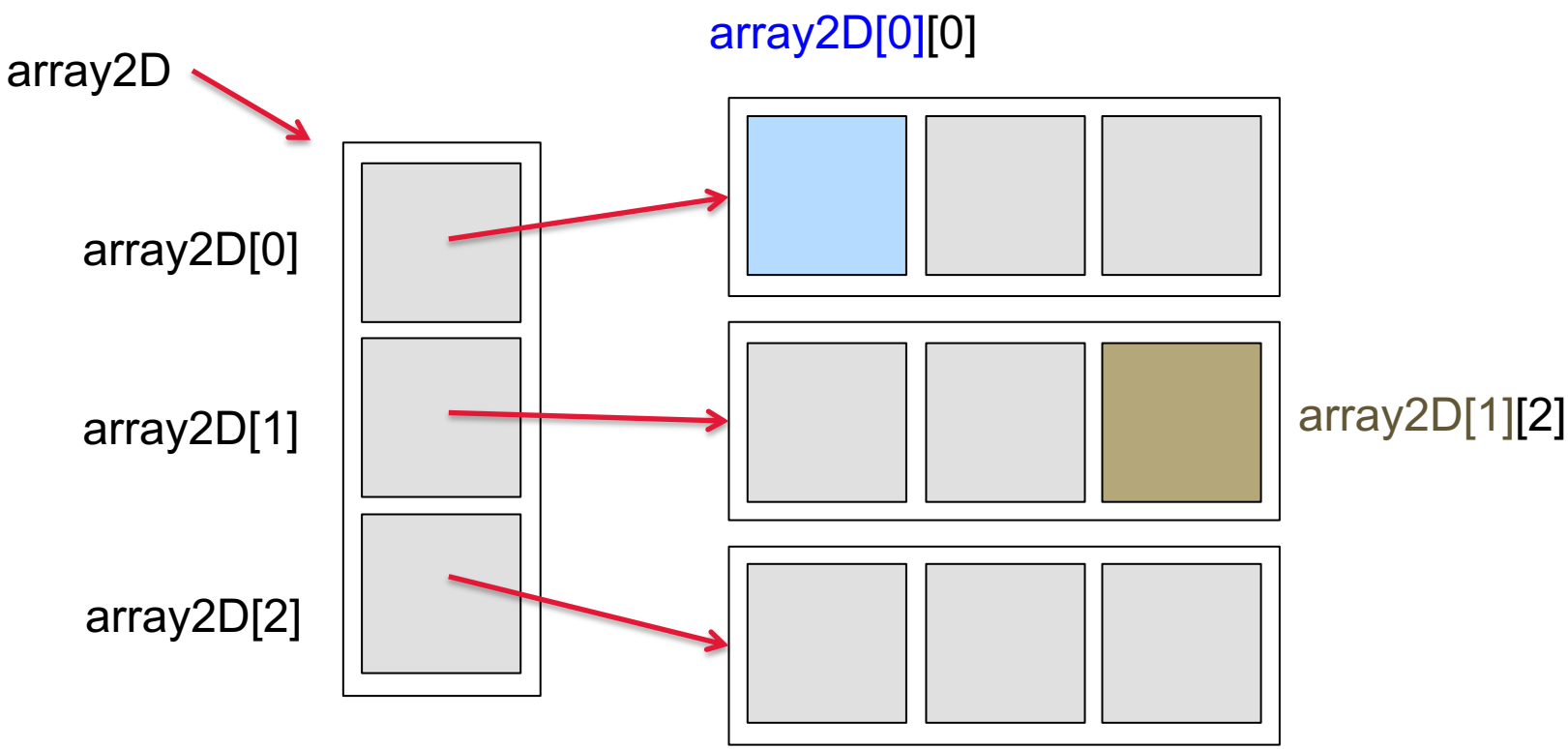
2D Array – in memory (actually, more like this)



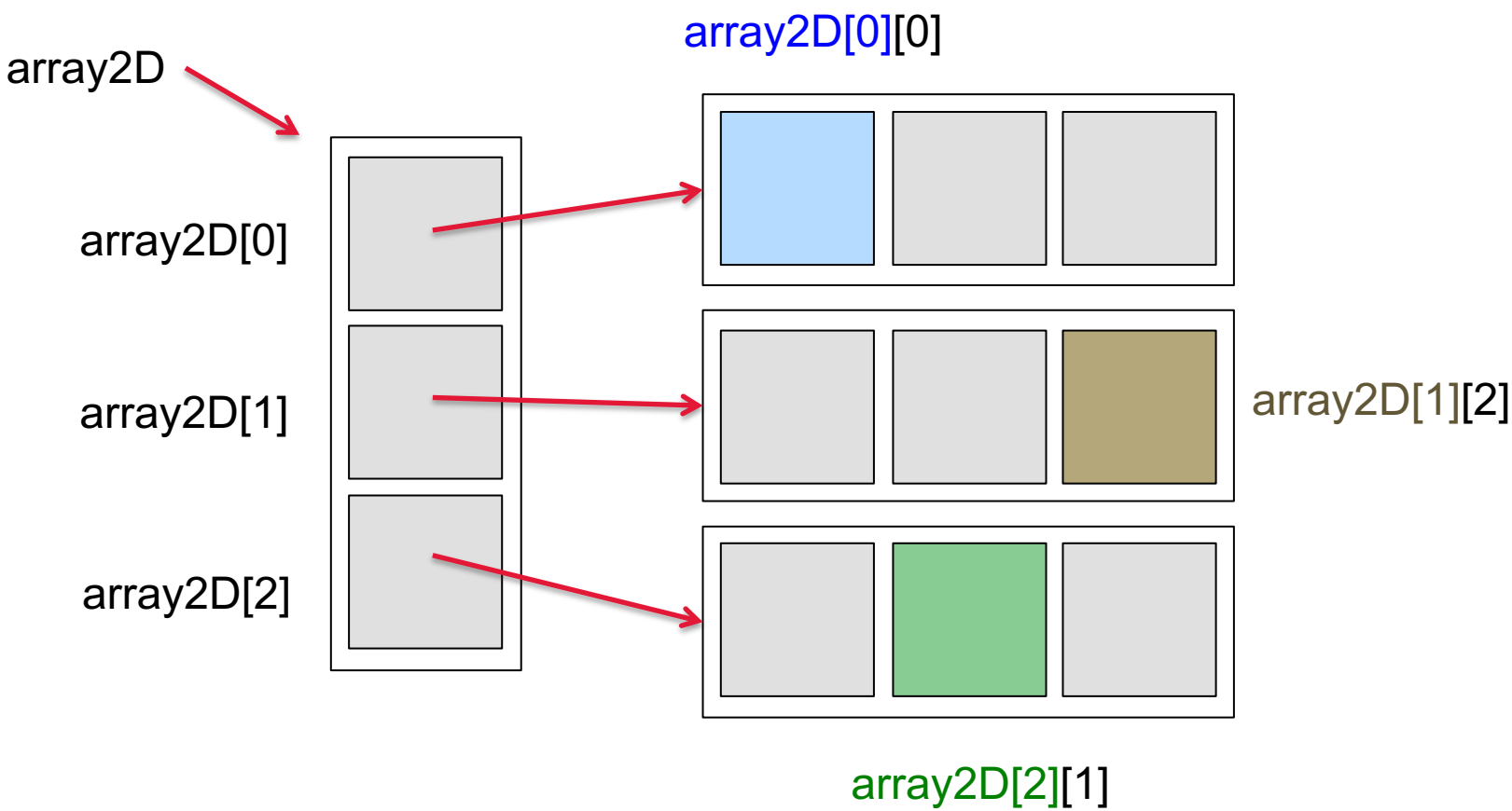
2D Array – in memory (actually, more like this)



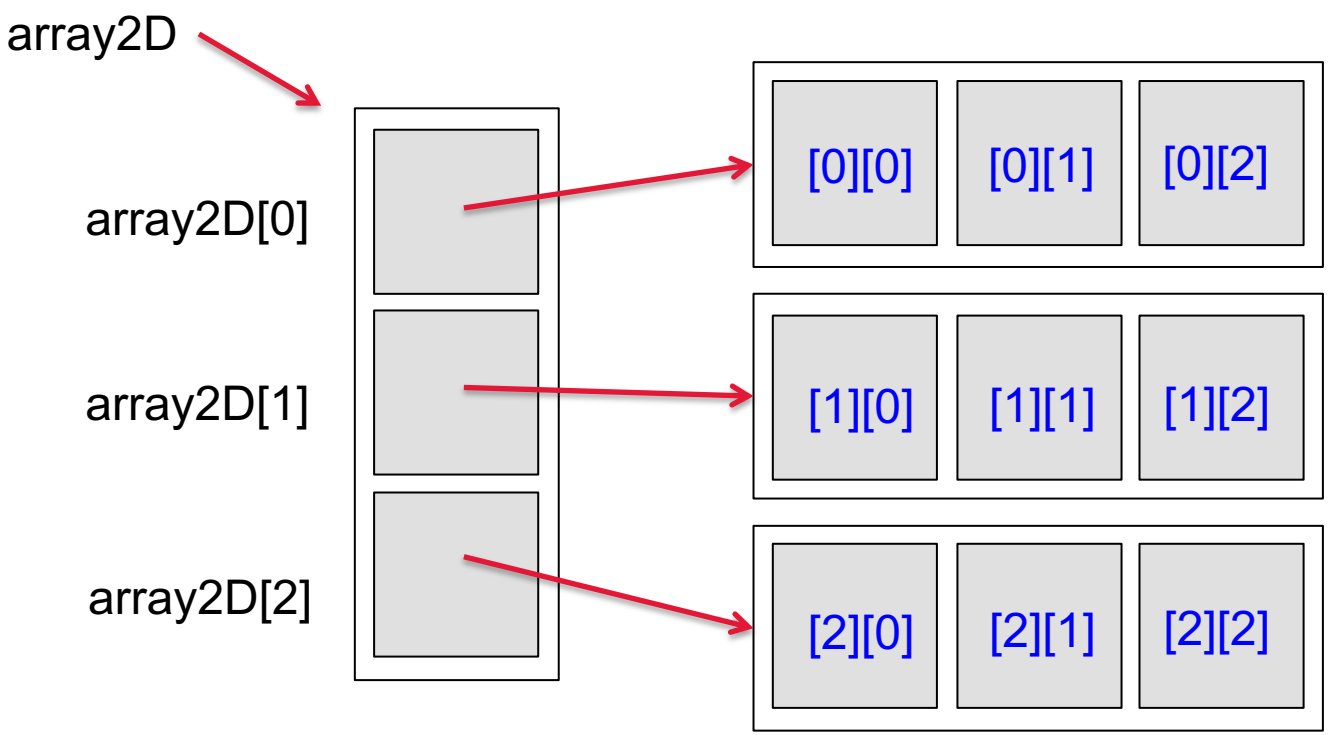
2D Array – in memory (actually, more like this)



2D Array – in memory (actually, more like this)



2D Array – in memory (actually, more like this)



Iterating over 2D array

```
// assumes array is MxN -> print out


for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {

       print("\t" + array2D[i][j]);

   }
       print("\n\n\n");

}
```

Processing values in 2D array

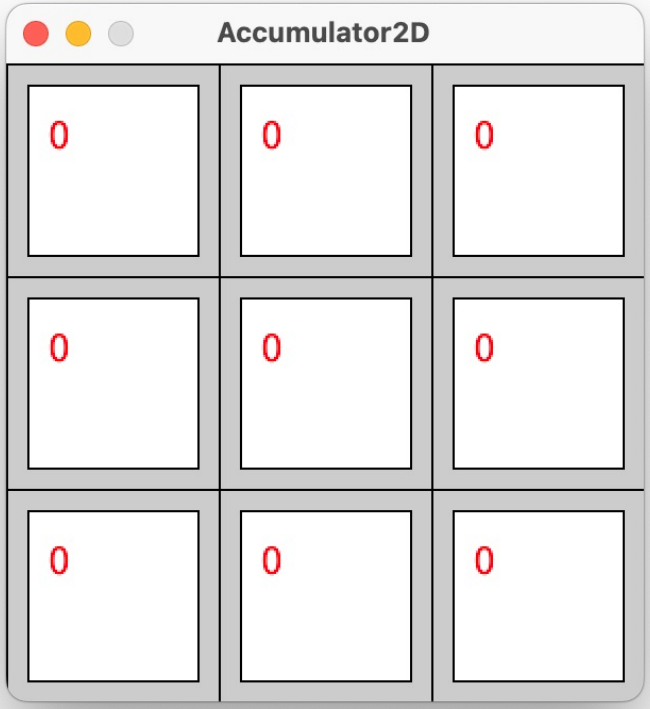
```
// e.g.
// sum values & find how many values in
// 2D array have a value larger than 10

int sum=0;
int numGreaterThan10=0;

for (int i = M - 1; i >= 0; i = i - 1) {
   for (int j = 0; j < N; j++) {

        print("\t" + array2D[i][j]);

        sum += array2D[i][j];

        if (array2D[i][j]>10) {
                numGreaterThan10++ ;
        }
   }
   print("\n\n\n");

}
```

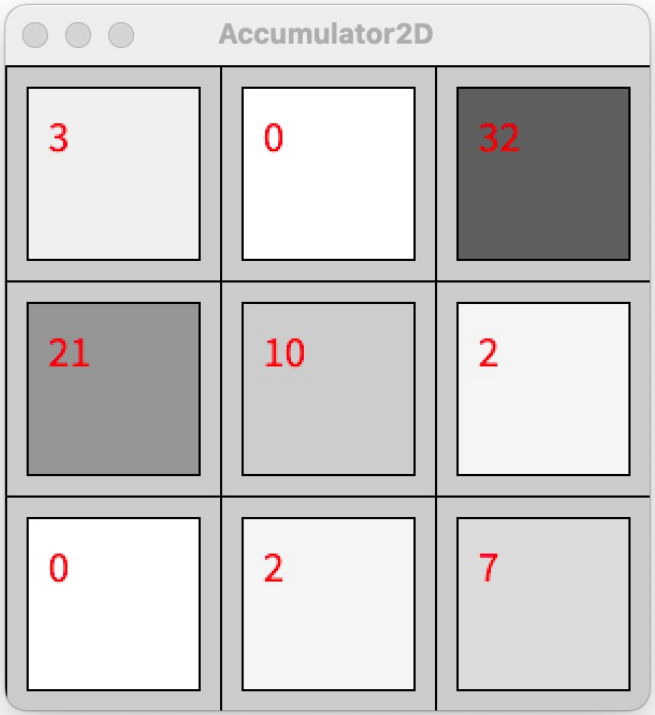
Accumulator2D (exercise)

Note this slide is modified from the one in the lecture (this exercise is re-framed to create a visual accumulator in processing (see next slide)

- Create a 2D integer array (initialized to 0's)
  - Think about a grid on the app window (e.g. 3x3 grid = 3x3 array), with each location represented by a rectangle (rect) to be drawn
- Let mouse clicks modify the array:
  - Get user to pick a location (click on location with mouse and use j=mouseX, i=mouseY)
  - Increment that location by 1
- Goal: array is to accumulate a value at locations that are chosen by user (e.g. could represent "hits" at a location)
  - Display text and colour to indicate the accumulation of clicks on that particular cell in the grid

Accumulator2D

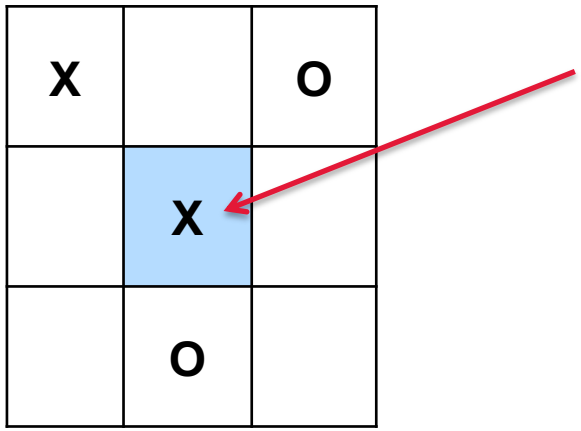




After several mouse clicks into each of the cells… darker cells have higher count of clicks

tic tac toe (exercise)

```
char[][] board ;
```



`board[i][j]`

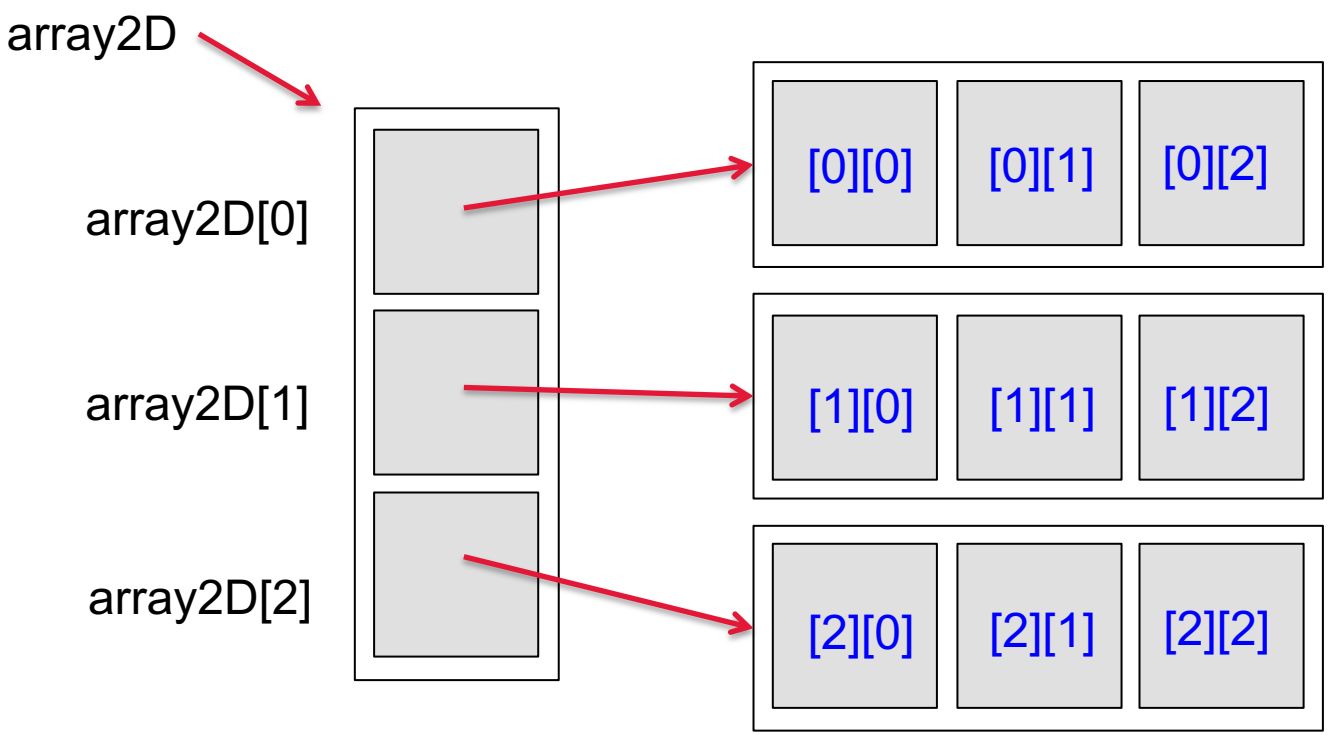
How would you modify the accumulator idea to record selections for a tic tac toe game??

[exercise given for lab4]

Get user to enter i,j (clicks) (fill location with ‘X’ or ‘O’)

Output (display/print) “board”

2D Array – in memory
print out 2D as a matrix?



Directionality is fairly arbitrary
(it relies 100% on how we index and loop over those indices)

```
// print in matrix form (0,0) at lower left

for (int i = M - 1; i >= 0; i = i - 1) {

    for (int j = 0; j < N; j++) {

        print("\t" + array2D[i][j]);

    }
    print("\n\n\n");

}
```

YORK UNIVERSITÉ UNIVERSITY

2D Array – in memory
print out 2D as a matrix?