

# Multidimensional Arrays

# Arrays

137	42	314	271	160	178
0	1	2	3	4	5

- An array stores a **sequence** of multiple objects.
  - Can access objects by index using `[]`.
- All stored objects have the same type.
  - You get to choose the type!
- Can store *any* type, even primitive types.
- Size is fixed; cannot grow once created.

# Basic Array Operations

- To create a new array, specify the type of the array and the size in the call to **new**:

***Type*** [] ***arr*** = **new** ***Type*** [***size***]

- To access an element of the array, use the square brackets to choose the index:

***arr*** [***index***]

- To read the length of an array, you can read the **length** field:

***arr*** . **length**

**YO DAWG, I HEARD YOU LIKE ARRAYS**



**YO DAWG, I HEARD YOU LIKE ARRAYS**

**SO I PUT AN ARRAY IN YOUR ARRAY SO  
YOU CAN INDEX WHILE YOU INDEX**

# Multidimensional Arrays

- You can create **multidimensional arrays** to represent multidimensional data.

```
int[][] a = new int[3][5];
```

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

# Multidimensional Arrays

- You can create **multidimensional arrays** to represent multidimensional data.

```
Type [][] a = new Type[rows][cols];
```

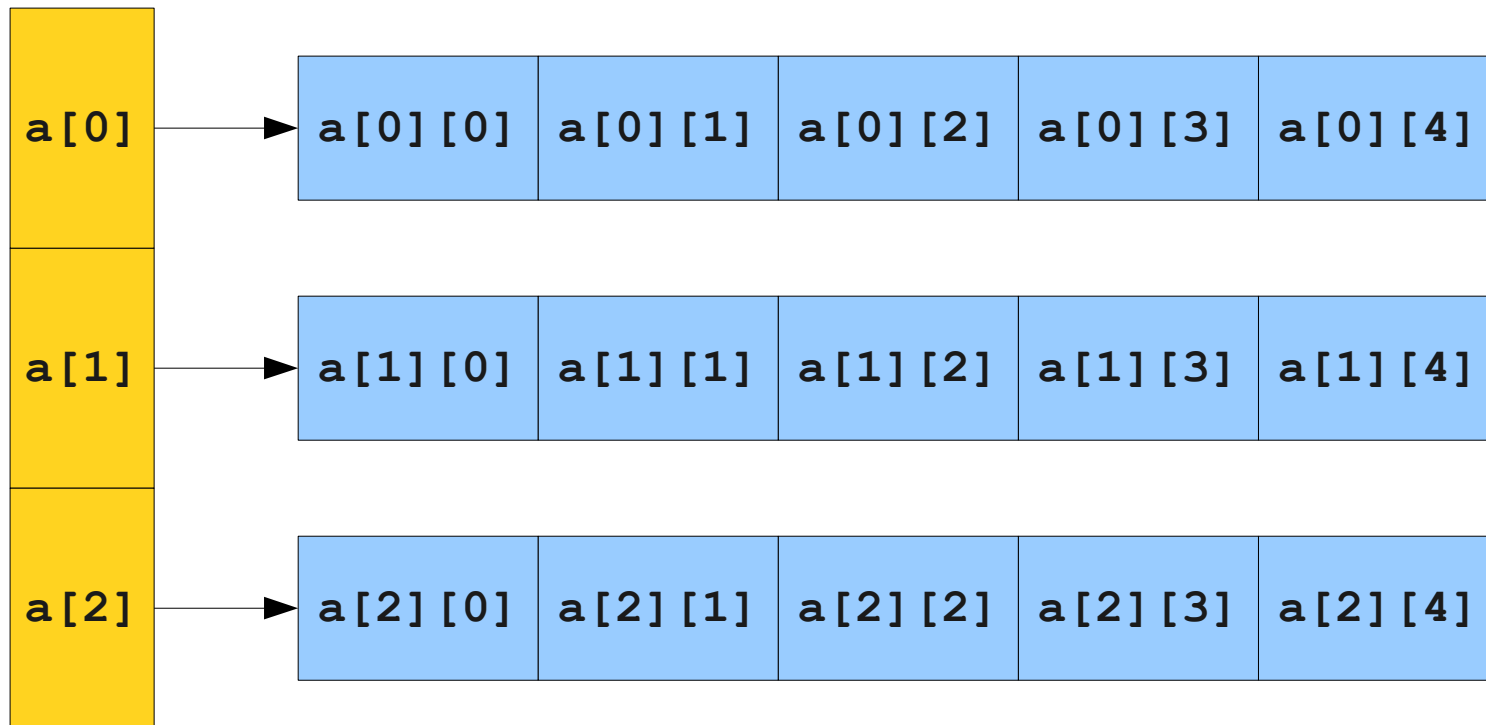
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

# Interpreting Multidimensional Arrays

- There are two main ways of intuiting a multidimensional array.
- **As a 2D Grid:**
  - Looking up `arr[row][col]` selects the element in the array at position (`row`, `col`).
- **As an array of arrays:**
  - Looking up `arr[row]` gives back a one-dimensional consisting of the columns in row `row`.



a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]



# Loops and Multidimensional Arrays

- The canonical way to loop over a multidimensional array is with a double **for** loop:

```
Type[][] arr = /* ... */  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        /* ... access arr[row][col] ... */  
    }  
}
```

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0



# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0



# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	0	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	0	0	0	0	0
3	0	0	0	0	0

# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	0	0	0	0	0
3	0	0	0	0	0

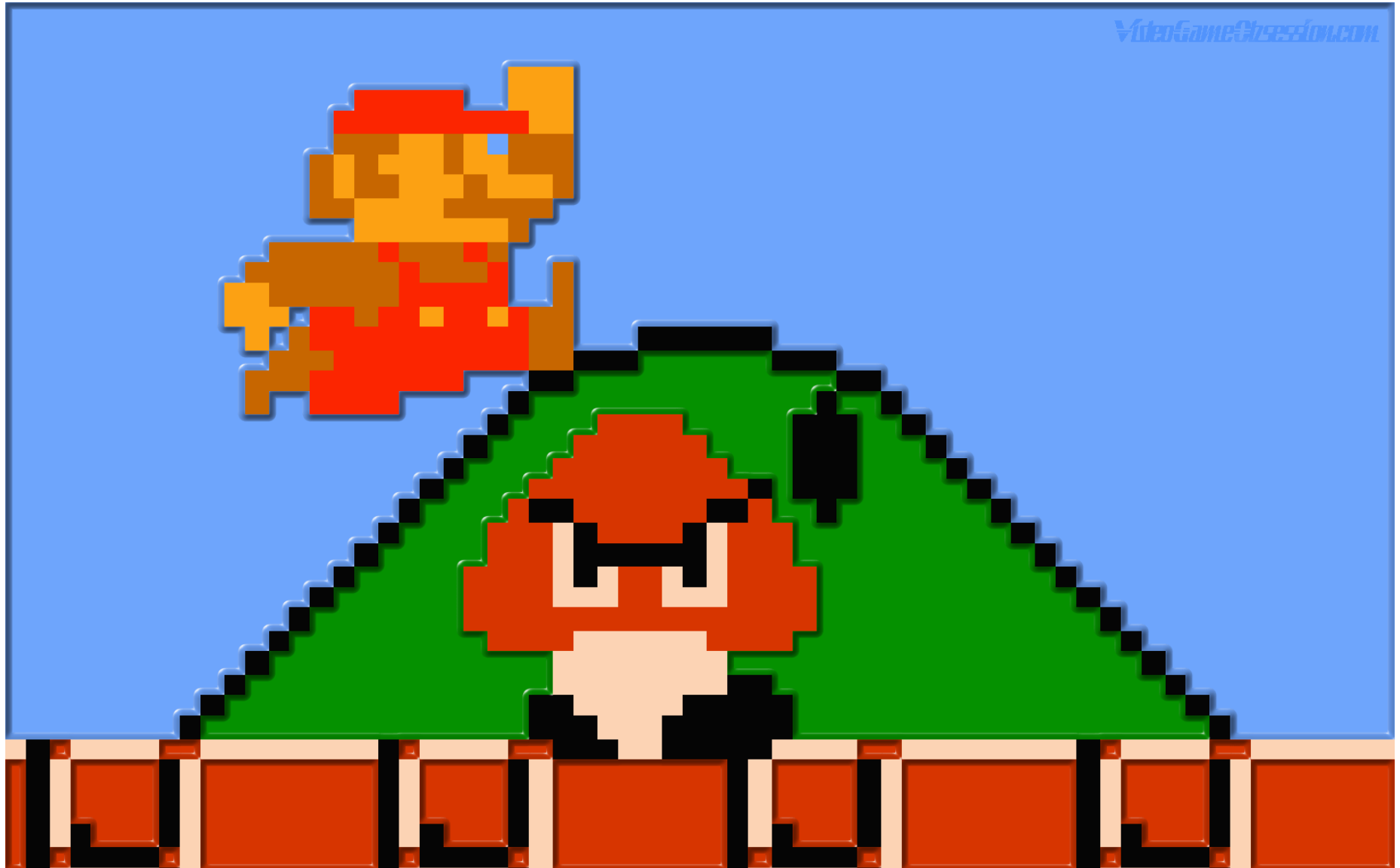


# Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];  
for (int row = 0; row < arr.length; row++) {  
    for (int col = 0; col < arr[row].length; col++) {  
        arr[row][col] = row + col;  
    }  
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7

# Working with Images



# Representations of Color

- The human eye has three different types of color receptors that pick up colors (close to) red, green, and blue.
- Computers usually represent color as **RGB triplets**:
  - Describe the intensity of the red, green, and blue components of the color.
  - Values range from 0 (min) to 255 (max), inclusive.



# Early Color Photographs



This picture was taken in 1911 by  
**Сергей Михайлович Прокудин-Горский**  
(Sergei Mikhailovich Prokudin-Gorskii)

# Creating GImageS

- It is possible to directly create a **GImage** by specifying the RGB values of each pixel in the image.
- To do so:
  - Create an `int[][]` two-dimensional array to hold the pixel values.
  - Use `GImage.createRGBPixel` to convert the RGB triplets to `int`.
  - Construct a `new GImage` from the array.

**Time-Out for Announcements!**

# An Interesting Read

“Intel's Sharp-Eyed Social Scientist”

<http://www.nytimes.com/2014/02/16/technology/intels-sharp-eyed-social-scientist.html?from=homepage>



# Assignment 5 Demo

# Announcements

- Assignment 4 due right now.
  - Due Friday with one late period, Monday with two.
- Assignment 5 (**Array Algorithms**) out today, due **Friday, February 28** at 3:15PM.
  - Play around with arrays, sound processing, and image processing!
  - Send secret messages to your friends!
  - Compose music!
  - Fix broken family photos!
- YEAH hours (assignment review hours) this Sunday, February 23 from 7PM – 8PM in Hewlett 200.

Back to CS106A!

# Manipulating Images

- You can extract an array of pixels from a **GImage** by calling

*image*.getPixelArray()

- You can then create a new image by changing the pixel values.
  - Changing these pixel values doesn't change the underlying image; the picture stored in a **GImage** is immutable.
- Can read color components with **GImage.getRed**, **GImage.getBlue**, and **GImage.getGreen**.

Midterms Available Outside