CS 106A, Lecture 18
Practice with 1D and 2D Arrays
HW5: ImageShop

We are here

Practice

2D Arrays

Arrays

Midterm!

The River of Java
Plan for Today

• Recap: 2D Arrays and Images
• Practice: Shrink
• Practice: Histogram
• Practice: Tic-Tac-Toe
Plan for Today

• Recap: 2D Arrays and Images
• Practice: Shrink
• Practice: Histogram
• Practice: Tic-Tac-Toe
The Matrix

Image used under “fair use” for educational purposes.
2D Arrays ("Matrices")

\[
\begin{pmatrix}
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3
\end{pmatrix}
\]

WELCOME..... TO THE MATRIX!!!!!!

2D Arrays

`type[][] name = new type[rows][columns];`

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a[0][0]</td>
<td>a[0][1]</td>
<td>a[0][2]</td>
<td>a[0][3]</td>
<td>a[0][4]</td>
</tr>
<tr>
<td>1</td>
<td>a[1][0]</td>
<td>a[1][1]</td>
<td>a[1][2]</td>
<td>a[1][3]</td>
<td>a[1][4]</td>
</tr>
</tbody>
</table>
int[][] a = new int[3][4];
int[] firstRow = a[0];
Summary: 2D Arrays

- Make a new 2D array
  
  \[ \text{type}[][] \text{name} = \text{new type}[\text{rows}][\text{columns}]; \]

- Get and set values using bracket notation

  \[ \text{name}[\text{row}][\text{col}] \quad // \text{get elem at row},\text{col} \]

  \[ \text{name}[\text{row}][\text{col}] = \text{value}; \quad // \text{set elem at row},\text{col} \]

- Get the number of rows and columns

  \[ \text{arr.length} \quad // \# \text{ rows} \]

  \[ \text{arr[0].length} \quad // \# \text{ columns} \]

- Iterate over a 2D array using a double for-loop

  \[
  \text{for} \ (\text{int row} = 0; \text{row} < \text{arr.length}; \text{row}++) \ { \ \\
  \quad \text{for} \ (\text{int col} = 0; \text{col} < \text{arr[0].length}; \text{col}++) \ { \ \\
  \quad \quad \text{// do something with arr[row][col];} \ \\
  \quad } \ \\
  \} \ \\
  \} \]
Limitations of 2D Arrays

• Unlike 1D arrays, you cannot compare 2D arrays with `Arrays.equals`. You must use `Arrays.deepEquals`.

```java
int[][][] a1 = ...;
int[][][] a2 = ...
if (Arrays.deepEquals(a1, a2)) {
    ...
}
```

• A 2D array does not know how to print itself:

```java
int[][][] a = new int[rows][cols];
println(a);                        // [I@8cf420
println(Arrays.toString(a));      // [I@6b3f44,[I@32c2a8]...
println(Arrays.deepToString(a));  // [[0, 1, 2, 3, 4], [1, 2, ...
Images

Images are just grids (2D arrays!) of pixels! Pixels are just integer values from 0-255.
Images as 2D Arrays

We can get a GImage as a 2D array of pixels.

GImage img = new GImage("res/daisy.jpg");
int[][][] pixels = img.getPixelArray();
int pixel = pixels[0][0]; // top-left pixel
Example: Pointillism

Pointillism is an art style where many small dots of color are combined to make a larger image.
Red, Green and Blue in one int?

Images *encode* the R, G, and B values of a pixel into a single integer between 0 and 255. You can convert between this **pixel value** and the individual **RGB values**.

```java
int[][][] pixels = image.getPixelArray();
int px = pixels[0][0];
int red = GImage.getRed(px);
int green = GImage.getGreen(px);
int blue = GImage.getBlue(px);
```
Creating New Pixels

Images *encode* the R, G, and B values of a pixel into a single integer between 0 and 255. You can convert between this *pixel value* and the individual *RGB values*.

You can also create pixels with your own RGB values.

```java
int r = ...  
int g = ...  
int b = ...  
int pixel = GImage.createRGBPixel(r, g, b);
```
Images as 2D Arrays

We can get a GImage as a 2D array of pixels, and modify it any way we want. Then, we can create a new GImage with the modified pixels.

GImage img = new GImage("res/daisy.jpg");
int[][][] pixels = img.getPixelArray();
... (modify pixels)
img.setPixelArray(pixels);  // update image

// or make a new GImage
GImage newImg = new GImage(pixels);
Modifying Image Pixels

• There are many cool image algorithms based around modifying individual pixels in an image: grayscale, brighten, normalize, remove red-eye...

grayscale

zoom
GImage Pixel Methods

```java
GImage img = new GImage("res/daisy.jpg");
```

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>img.getPixelArray()</code></td>
<td>returns pixels as 2D array of ints, where each int in the array contains all 3 of Red, Green, and Blue merged into a single integer</td>
</tr>
<tr>
<td><code>img.setPixelArray(array);</code></td>
<td>updates pixels using the given 2D array of ints</td>
</tr>
<tr>
<td><code>GImage.createRGBPixel(r, g, b)</code></td>
<td>returns an int that merges the given amounts of red, green and blue (each 0-255)</td>
</tr>
<tr>
<td><code>GImage.getRed(px)</code></td>
<td>returns the redness, greenness, or blueness of the given pixel as an integer from 0-255</td>
</tr>
<tr>
<td><code>GImage.getGreen(px)</code></td>
<td></td>
</tr>
<tr>
<td><code>GImage.getBlue(px)</code></td>
<td></td>
</tr>
</tbody>
</table>
Recap: Modifying Pixels

• **Extract** pixel RGB colors with `GImage.getRed/Blue/Green`.

  ```java
  int red   = GImage.getRed(pixels[0][0]);  // 0-255
  int green = GImage.getGreen(pixels[0][0]); // 0-255
  int blue  = GImage.getBlue(pixels[0][0]);  // 0-255
  ```

• **Modify** the color components for a given pixel.

  ```java
  red = 0;  // remove redness
  ```

• **Combine** the RGB back together into a single `int`.

  ```java
  pixels[0][0] = GImage.createRGBPixel(red, green, blue);
  ```

• **Update** the image with your modified pixels when finished.

  ```java
  image.setPixelArray(pixels);
  ```
Plan for Today

- Recap: 2D Arrays and Images
- Practice: Shrink
- Practice: Histogram
- Practice: Tic-Tac-Toe
Let’s write a program that can *shrink* an image to $\frac{1}{2}$ its original size.
Shrink
Shrink
Given a pixel \((x, y)\) in our smaller image, how do we know which pixel in our larger image should go there?
Given a pixel \((x, y)\) in our smaller image, how do we know which pixel in our larger image should go there?
Given a pixel \((x, y)\) in our smaller image, how do we know which pixel in our larger image should go there?
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];

for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}

image.setPixelArray(result);
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];
for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}
image.setPixelArray(result);
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];

for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}

image.setPixelArray(result);
Shrink

```java
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];

for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}

image.setPixelArray(result);
```
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];

for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}

image.setPixelArray(result);
int[][][] pixels = image.getPixelArray();
int[][][] result =
    new int[pixels.length/2][pixels[0].length/2];

for (int r = 0; r < result.length; r++) {
    for (int c = 0; c < result[0].length; c++) {
        result[r][c] = pixels[r*2][c*2];
    }
}

image.setPixelArray(result);
Plan for Today

• Recap: 2D Arrays and Images
• Practice: Shrink
• Practice: Histogram
• Practice: Tic-Tac-Toe
Given a file of temperatures (int), such as:

82
66
79
63
83

Write a **Histogram** program that will print a histogram of stars indicating the # of days with each unique temperature.

85: *****
86: ************
87: ***
88: *
Idea: Array of Counters

• For problems like this, where we want to keep count of many things, a frequency table (or tally array) can be a clever solution.
  – *Idea*: The element at index $i$ will store a counter for the digit value $i$.

– example: count of temperatures:

<table>
<thead>
<tr>
<th>index</th>
<th>...</th>
<th>55</th>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td></td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>


Plan for Today

• Recap: 2D Arrays and Images
• Practice: Shrink
• Practice: Histogram
• Practice: Tic-Tac-Toe
Let’s use 2D arrays to create a ConsoleProgram version of Tic-Tac-Toe.
Recap

• Recap: 2D Arrays and Images
• Practice: Shrink
• Practice: Histogram
• Practice: Tic-Tac-Toe

Next Time: More data structures