DATA STRUCTURES & ALGORITHMS

LECTURE 9

October 3

ANNOUNCEMENTS

TWO DIMENSIONS

ICLICKER QUESTION
ICLICKER QUESTION

What will be output by the following code?

```java
int[][] someNums = {{1,2,3},{4,5,6},{7,8,9}};
int sum = 0;
for(int i = 0; i < 3; i++)
    sum += someNums[i][i];
System.out.println(sum);
```

A. 12
B. 13
C. 15
D. 16
E. 45

LEFT ROTATION

• To rotate an image left 90 degrees still copy all the pixels
  • But they go to different locations in the target
  • Row values become column values
  • `target x = source y`
  • `target y = source width -1 – source x`

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

LEFT ROTATION ALGORITHM

• Create the target picture object
• Invoke the method on the target picture
• Create the source picture object
• Loop through the source x
  • Loop through the source y
  • Get the source pixel at the x and y values
  • Get the target pixel with the x equal the source y value and the y equal the source picture width – 1 minus the source x
  • Copy the color from the source pixel to the target pixel

```java
public void copySpecificLeftRotation()
{
    String sourceFile = ".../piperdd.jpg";
    Picture sourcePicture = new Picture(sourceFile);
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    int targetX, targetY = 0;
    // loop through the columns
    for (int sourceX = 0; sourceX < sourcePicture.getWidth(); sourceX++)
    {
        // loop through the rows
        for (int sourceY = 0; sourceY < sourcePicture.getHeight(); sourceY++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(sourceX, sourceY);
            targetX = sourceY;
            targetY = sourcePicture.getWidth() - 1 - sourceX;
            targetPixel = this.getPixel(targetX, targetY);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
```
TESTING LEFT ROTATION

- `Picture p = new Picture(640, 480);`
- `p.show();`
- `p.copySpecificLeftRotation();`
- `p.repaint();`

RIGHT ROTATION

- To rotate an image right 90 degrees still copy all the pixels
  - But they go to different locations in the target
    - Column values become row values
    - `target y = source x`
    - `target x = source height – 1 – source y`

RIGHT ROTATION EXERCISE

- Write the method to rotate the picture of the dog to the right instead of to the left
- Try out the method
  ```java
  Picture p = new Picture(640, 480);
  p.show();
  p.copySpecificRightRotation();
  p.repaint();
  ```
- Can you make the method more general?
  - To work on any picture?

SCALING

- You can make a picture smaller
  - Faster to download on the web
- You can make a picture larger
  - Show more detail
SCALING DOWN THE A PICTURE

- *passionflower.jpg* is
  - 640 pixels wide
  - 480 pixels high
- If we copy every other pixel we will have a new picture with
  - width = $640 / 2 = 320$
  - height = $480 / 2 = 240$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

SCALING DOWN ALGORITHM

- Create the target picture
- Invoke the method on the target picture
- Create the source picture
- Loop with source x starting at 0 and target x starting at 0 as long as $< \text{source width}$
  - Increment the source x by 2 each time through the loop, increment the target x by 1
  - Loop with source y starting at 0 and target y starting at 0 as long as $< \text{source height}$
  - Increment the source y by 2 each time through the loop, increment the target y by 1
  - Copy the color from the source to target pixel

```java
public void copyFlowerSmaller()
{
    Picture pic = new Picture(".../passionflower.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    // loop through the columns
    for (int sourceX = 0, targetX = 0;
        sourceX < sourcePicture.getWidth();
        sourceX += 2, targetX++)
    {
        // loop through the rows
        for (int sourceY = 0, targetY = 0;
            sourceY < sourcePicture.getHeight();
            sourceY += 2, targetY++)
        {
            // set the target pixel color to the source pixel color
            this.getPixel(targetX,targetY).setColor(pic.getPixel(sourceX,sourceY).getColor());
        }
    }
}
```
TRYING COPY FLOWER SMALLER

• Create a new picture half the size of the original picture (+1 if odd size)
  
  \[
  \text{Picture } \text{pl} = \text{new Picture}(320,240);
  \]

• Copy the flower to the new picture
  
  \[
  \text{pl.copyFlowerSmaller();}
  \]

• Show the result
  
  \[
  \text{pl.show();}
  \]

MORE GENERAL

• Can we make scale down more general?
• Change the method to operate on the source instead of the target.

```java
public void scaleDown() {
    Picture pic = new Picture(".../passionflower.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    int targetX, targetY = 0;
    // loop through the columns
    for (int sourceX = 0, targetX = 0;
        sourceX < pic.getWidth();
        sourceX += 2, targetX++)
    {
        // loop through the rows
        for (int sourceY = 0, targetY = 0;
            sourceY < pic.getHeight();
            sourceY += 2, targetY++)
        {
            // set the target pixel color to the source pixel color
            this.getPixel(targetX,targetY).setColor(pic.getPixel(sourceX,sourceY).getColor());
        }
    }
}
```

```java
public Picture scaleDown() {
    Picture pic = new Picture(".../passionflower.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    int targetX, targetY = 0;
    // loop through the columns
    for (int sourceX = 0, targetX = 0;
        sourceX < pic.getWidth();
        sourceX += 2, targetX++)
    {
        // loop through the rows
        for (int sourceY = 0, targetY = 0;
            sourceY < pic.getHeight();
            sourceY += 2, targetY++)
        {
            // set the target pixel color to the source pixel color
            this.getPixel(targetX,targetY).setColor(pic.getPixel(sourceX,sourceY).getColor());
        }
    }
}
```
TRYING SCALEDOWN

- Open a picture
  - String file = FileChooser.getMediaPath("passionflower.jpg");
  - Picture original = new Picture(file);
- Scale down and save in a new picture
  - Picture smaller = original.scaleDown();
- Show the result
  - original.show();
  - smaller.show();
THINKING THROUGH SCALING UP

- Copy each pixel in the source multiple times to the target
  - Source (0,0) Target (0,0)
  - Source (0,0) Target(1,0)
  - Source (1,0) Target(2,0)
  - Source (1,0) Target(3,0)
  - Source (2,0) Target(4,0)
  - Source (2,0) Target(5,0)
  - Source (0,0) Target(0,1)
  - Source (0,0) Target(1,1)

ICLICKER QUESTION

Which direction will the following code copy the array?

```java
for (x = 0; x < width/2; x++)
{
    for (y = 0; y < height; y++)
    {
        myArr[width-x-1][y] = myArr[x][y];
    }
}
```

A. Mirror top half to bottom
B. Mirror bottom half to top
C. Mirror left half to right
D. Mirror right half to left

SCALING UP EXERCISE

- Write a method `scaleUp()` to scale up a picture by 2
- Save the result to a file using
  ```java
  pic.write('file');
  ```