Data Structures & Algorithms

Lecture 7

February 13

Benjamin Carle

Vassar College

Announcements

Copying Pixels to a New Picture

• What if we wanted to copy (part of) one picture into another?
• How would we do that?
Copying Pixels to a New Picture

- We would need to copy each pixel from the source to the target image.
- To do this, we would need to track the source picture x and y
  - And the target picture x and y

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

Copy Picture Algorithm

- Copy a picture to a 960 x 720 blank picture
- Create the target picture object
- Invoke the method on the target picture
- Create the source picture object
- Loop through the source picture pixels
  - Get the source and target pixels
  - Set the color of the target pixel to the color of the source pixel

Copy Algorithm to Code

- Loop through the source pixels

```java
// loop through the columns
for (int sourceX = 0, targetX = 0;
     sourceX < sourcePicture.getWidth();
     sourceX++, targetX++)
{
    // loop through the rows
    for (int sourceY = 0, targetY = 0;
         sourceY < sourcePicture.getHeight();
         sourceY++, targetY++)
    {
```
Copy Algorithm to Code – Cont

- Get the source and target pixels
  
  `sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
  targetPixel = this.getPixel(targetX,targetY);`

- Set the color of the target pixel to the color of the source pixel
  
  `targetPixel.setColor(sourcePixel.getColor());`

```
public void copySpecific()
{
    Picture sourcePicture = new Picture("/piperdd.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;

    // loop through the columns
    for (int sourceX = 0, targetX = 0;
        sourceX < sourcePicture.getWidth();
        sourceX++, targetX++)
    {
        // loop through the rows
        for (int sourceY = 0, targetY = 0;
            sourceY < sourcePicture.getHeight();
            sourceY++, targetY++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
            targetPixel = this.getPixel(targetX,targetY);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
```

Copying A Specific Picture

- We can use this technique to copy a specific picture.
- In this case the picture "piperdd.jpg"

```
```

Trying the copySpecific Method

- Create a blank picture object
  
  - `Picture pl = new Picture(960, 720);`

- Invoke the method on this picture object
  
  `- pl.copySpecific();`

- Show the picture
  
  `- pl.show();`
Result of copySpecific Method

- The `copySpecific()` method copies the picture into the upper left hand corner of the new image.
- What if we wanted to put it somewhere different?

Copy to an Upper Left Location

- How would you copy a picture to a location in another picture (like 100, 100)?
  - Specified as the upper left corner
- You still copy all the source pixels
  - But the target x and y start at the specified location

Copy to Position Exercise

- Write a method `copyRobot` to copy
  - `robot.jpg`
  - To location
    - 100, 100 in blank picture
- Test with
  ```java
  Picture p = new Picture(640, 480);
p.copyRobot();
p.show();
  ```
Copy to Position Exercise

• Hint – you have an offset in each direction:
  – offsetX – how far to the right you want to start
  – offsetY – how far down you want to start.
• When copying the pixels, you use the offsets to figure how far over and down to copy the pixels.

Cropping

• We can copy just part of a picture to a new picture
  – Just change the start and end values for source x and y values to the desired range of pixels.
    • Instead of starting at (0,0) and going to (width-1,height-1).
  – Use pictureObj.explore() to find the x and y values.
  – What are the x and y values to get the face of the dog in piperdd.jpg?

public void copySpecificFace()
{
    Picture sourcePicture = new Picture("../piperdd.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;

    // loop through the columns
    for (int sourceX = 190, targetX = 100; sourceX < 440; sourceX++, targetX++)
    {
        // loop through the rows
        for (int sourceY = 30, targetY = 100; sourceY < 240; sourceY++, targetY++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
            targetPixel = this.getPixel(targetX,targetY);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
Testing Copy specific Face

• Create a picture object
  - `Picture p1 = new Picture(640, 480);`
• Invoke the method
  - `p1.copySpecificFace();`
• Show the picture
  - `p1.show();`

What makes a Good Method?

• Ideally, a method should do one and only one thing.
  – That is, accomplish some task.
  – Also, the name should tell you what the method does.

What makes a Good Method?

• A method can call other methods to do some of its work
  – This is known as *procedural decomposition*.
  – In essence, farming out sub-problems for other methods to handle.
• For example, remember when we re-wrote the sunset method to call methods to
  – Decrease green
  – Decrease blue
  to get the (increased red) sunset effect.

What makes a Good Method?

• We shouldn’t have the same code appearing in several methods.
  – Instead, put the code to do that task in its own method, and call that method when we need that code.
  – Like the methods to change the amount of red, green and blue.
    • Make three methods to do those color changes.
  – We should make simple, general methods that are reusable.
What makes a Good Method?

- A method should be in the class that has the data the method is working on.
  - That is methods that work on Picture objects go in Picture.java, etc.

was the last method general?

- No.
- We wrote in the name of a specific file to copy from in the method
- What if we want to copy from a different picture?
  - We would need to
    - change the method
    - or
    - make another method

General Copy Algorithm

- Create a method that copies pixels from any source picture
  - Passing the picture as a parameter.
  - Giving a start x and y and end x and y for the source picture
    - If the start x and y and end x and y cover the entire picture then the whole picture will be copied
    - If the start x and y and end x and y are part of the picture then cropping will occur
  - Copying to the current picture object with a target start x and target start y
    - If the start x and y are 0 then it copies to the upper left corner

General Copy Algorithm

- Loop through the x values between xStart and xEnd
- Loop through the y values between yStart and yEnd
- Get the pixel from the source picture for the current x and y values
  - Get the pixel from the target picture for the targetStartX + x and targetStartY + y values
  - Set the color in the target pixel to the color in the source pixel
public void copy(Picture sourcePicture, int startX, int startY,
    int endX, int endY, int targetStartX, int targetStartY)
{
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    // loop through the x values
    for(int x = startX, tx = targetStartX;
        x < endX;
        x++, tx++)
    {
        // loop through the y values
        for(int y = startY, ty = targetStartY;
            y < endY;
            y++, ty++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(x, y);
            targetPixel = this.getPixel(tx, ty);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}

// Rewrite Methods Exercise

- Type the copy method in Picture.java
- Rewrite copySpecific() and copySpecificFace() methods to use the new copy method
- Run the methods to make sure they still work
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

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

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());
        }
    }
}
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.copySpecificRightRotation();
  p.show();
  ```
- 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
 Scaling Down the a Picture

- **passionFlower.jpg** is 640 pixels wide and 480 pixels high
- If we copy every other pixel we will have a new picture with
  - width = \( \frac{640}{2} = 320 \)
  - height = \( \frac{480}{2} = 240 \)

```
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());
        }
    }
}
```

 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 < 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 < 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

 Trying Copy Flower Smaller

- Create a new picture half the size of the original picture (+ 1 if odd size)
  - Picture p1 = new Picture(320, 240);
- Copy the flower to the new picture
  - p1.copyFlowerSmaller();
- Show the result
  - p1.show();
More general

- Can we make scale down more general?
- Change the method to operate on the source instead of the target.
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)
Scaling Up Exercise

• Write a method `scaleUp()` to scale up a picture by 2.
• Save the result to a file using 
  – `pic.write("file");`