Modifying Pixels in a Matrix

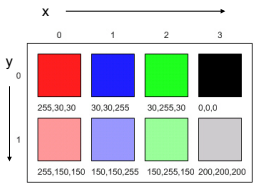
Barb Ericson
Oct 2010

Learning Goals

- Understand at a conceptual and practical level
  - How to loop through a two-dimensional array using a nested loop?
    - How do you get the pixel for a x and y location?
  - How to use print statements to trace execution
  - How to simplify a problem
    - To make it easier to figure out the algorithm
  - Revisit method overloading and returning a value

What is a two-dimensional array?

- The pixels in a picture are really stored in a two-dimensional array
  - Each pixel has a x value (horizontal location)
  - Each pixel has a y value (vertical location)
  - pictureObj.getPixel(x,y) returns the pixel at that location



Example Two-Dimensional Arrays

- Maps
  - That street is in D-5
- Battleship
  - Try I-5
    - Hit or miss
- Chairs at a theater or game
  - Row C seat 20

Nested Loop

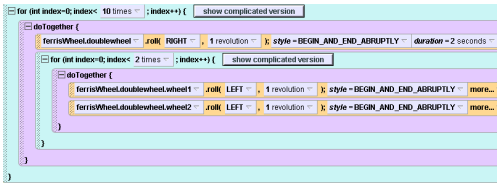
- How would you get all the pixels in a picture using their x and y values
  - From left to right and top to bottom?
  - x=0 and y=0, x=1 and y=0, x=2 and y=0, …
  - x=0 and y=1, x=1 and y=1, x=2 and y=1, …
  - x=0 and y=2, x=1 and y=2, x=2 and y=2, …
- We need to have one loop inside another
  - The outer loop counts y from 0 to height - 1
  - The inner loop counts x from 0 to width - 1

Alternative Nested Loop

- How would you get all the pixels in a picture using their x and y values
  - From top to bottom and left to right?
  - x=0 and y=0, x=0 and y=1, x=0 and y=2, …
  - x=1 and y=0, x=1 and y=1, x=1 and y=2, …
  - x=2 and y=0, x=2 and y=1, x=2 and y=2, …
- We need to have one loop inside another
  - The outer loop counts x to width - 1
  - The inner loop counts y from 0 to height - 1

Alice Nested Loop

- In Alice we used a nested loop to turn the ferris wheel. The outer loop rolled the large double wheel to the right and the inner loop rolled the two smaller wheels to the left.



Example using nested loops

```
public void clearBlueNested()
{
  Pixel pixel = null;

  // loop through the columns (x direction)
  for (int x = 0; x < getWidth(); x++)
  {
    // loop through the rows (y direction)
    for (int y = 0; y < getHeight(); y++)
    {
      // get pixel at the x and y location
      pixel = this.getPixel(x,y);

      // clear the blue
      pixel.setBlue(0);
    }
  }
}
```


Tracing clearBlueNested

pict.clearBlueNested()

- Starts execution of the clearBlueNested method on the picture called pict

Pixel pixel = null

- Set up a variable to refer to the current pixel, but there isn't a current pixel to start (set to null)

for (int x = 0; x < getWidth(); x++)

- Loop with x starting at 0, stop when x equals the width of the picture and increment by 1 each time. This will loop through all the columns.
  - Notice that you don't have to use this.getWidth() the *this* is implied.

Tracing clearBlueNested - continued

for (y = 0; y < getHeight(); y++)

- Loop with y starting at 0, stop when y equals the height of the picture and increment by 1 each time. This will loop through all the rows.

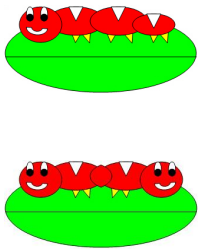
pixel = getPixel(x,y);

- set the variable pixel to refer to the pixel at the current x and y location. The first time through the loop this is (0,0). The second time it is (0,1). The third time it is (0,2). This keeps on till we process all the pixels in the first column from top to bottom and then we move to the second column. We keep processing pixels till we are done with all of them.

pixel.setBlue(0) // set the blue value to 0

Vertical Mirroring

- What if we want to pretend to place a mirror in the middle of the picture
  - We would see the left side of the picture mirrored on the right side



Thinking Through Vertical Mirroring

- If we just think of a number at each x and y location instead of a color
  - The mirroring would look like this:
- Can you write the algorithm to do this?

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

1	2	3	2	1
6	7	8	7	6
11	12	13	12	11

What is the Vertical Mirror for this?

- Try the solve the problem for small samples
- If you can't solve it on a small sample
  - You can't write a program to solve it

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

	0	1	2
0			
1			
2			

Mirror Vertical Algorithm

- Loop through all the rows (y starts at 0, increments by 1, and is less than the picture height)
  - Loop with x starting at 0 and x less than the midpoint (mirror point) value
    - Get the left pixel at x and y
    - Get the right pixel at width – 1 - x
    - Set the color for the right pixel to be the color of the left pixel

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

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

Mirror Vertical Algorithm to Code

- We are going to need the midpoint

  int midpoint = this.getWidth() / 2;
- Loop through the rows (y values)

  for (int y = 0; y < this.getHeight(); y++) {
  - Loop through half the x values (0 to midpoint)

    for (int x = 0; x < midpoint; x++) {
    - Set right pixel color to left pixel color

```
Pixel leftPixel = this.getPixel(x, y);
Pixel rightPixel = this.getPixel(this.getWidth() - 1 - x, y);
rightPixel.setColor(leftPixel.getColor());
```

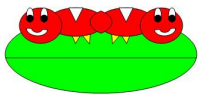

Mirror Vertical Method

```
public void mirrorVertical() {
 {
   int width = this.getWidth();
   int mirrorPoint = width / 2;
   Pixel leftPixel = null;
   Pixel rightPixel = null;

   // loop through all the rows
   for (int y = 0; y < getHeight();
        y++)
   {
     // loop from 0 to the middle
     for (int x = 0; x < mirrorPoint; x++)
     {
       leftPixel = getPixel(x, y);
       rightPixel = getPixel(width - 1 - x, y);
       right Pixel.setColor(
               leftPixel.getColor()) ;
     }
   }
 }
```

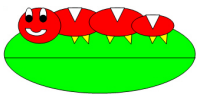

Trying Mirror Vertical

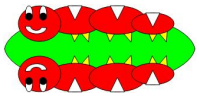
- Create the picture
  - Picture p1 = new Picture(
    FileChooser.getMediaPath("caterpillar.jpg");
- Invoke the method on the picture
  - p1.mirrorVertical();
- Repaint the picture
  - p1.repaint();



Mirror Horizontal

- What about mirroring around a mirror held horizontally in the vertical center of the picture?
  - Like a reflection in a lake?





Thinking Through Mirror Horizontal

- Again think of a number at each x and y location
  - Instead of a color
  - And try it with a small sample
- How can we write a nested for loop to do this?

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

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

What is the Horizontal Mirror for this?

- Try to solve the problem for several small sample problems
- See if you can come up with the algorithm to solve it
  - Test it more small samples

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

Mirror Horizontal Algorithm

- Get the vertical midpoint
  - Picture height / 2
- Loop through all the x values
  - Loop from y=0 to y < vertical midpoint
    - Get the top pixel
      - At x and y
    - Get the bottom pixel
      - Height - 1 - y
    - Set the bottom pixel's color to the top pixel color

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
1	2	3

Mirror Horizontal Exercise

- Write the method to mirror the top half of the picture to the bottom half.
  - This is a motorcycle redMotorcycle.jpg
- How about mirroring bottom to top?
- How would you mirror on a diagonal line?

Mirroring part of a picture

- Use the explorer to figure out the part to mirror

Mirror temple method

```
/**
 * Method to mirror part of the temple picture around a
 * vertical line at a mirror point
 */
public void mirrorTemple()
{
  int mirrorPoint = 276;
  Pixel leftPixel  = null;
  Pixel rightPixel  = null;
  // loop through the rows
  for (int y = 27; y < 97; y++)
  {
    // loop from 13 to just before  the mirror point
    for (int x = 13; x < mirrorPoint; x++)
```

Mirror temple - continued

```
  {
     leftPixel = getPixel(x, y);
     rightPixel = getPixel(mirrorPoint +
                      (mirrorPoint - x), y);
     right Pixel .setColor(leftPixel .getColor());
   }
  }
}
```


Testing the mirrorTemple method

```
> String fileName =
  "C:/intro-prog-java/mediasources/temple.jpg";
> Picture picture = new Picture(fileName);
> picture.explore();
> picture.mirrorTemple();
> picture.explore();
```


How many pixels were changed?

- We could put print statements in the code
  - to see the x and y values, but that would take a long time to print out each value
- We can also just keep a count
  - And print the value of the count after the loop ends
- We can calculate the number of times a loop will run using end – start + 1 = number of times
  - numOuter = 96 – 27 + 1 = 70
  - numInner = 275 – 13 + 1 = 263
  - Total for a nested loop is numInner * numOuter
    - 70 * 263 = 18,410

Copying Pixels to a New Picture

- Need to track the source picture x and y
  - And the target picture x and y
- We can use a blank picture
  - As the target picture
- Several blank pictures are available
  - 640x480.jpg
  - 7inX95in.jpg

1	2
3	4

1	2		
3	4		

Copy Picture Algorithm

- Copy a picture to the 7 by 9.5 inch 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

```
// 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());
```


Copy Method

```
public void copyKatie()
 {
  String sourceFile =
    FileChooser.getMediaPath("KatieFancy.jpg");
  Picture sourcePicture = new Picture(sourceFile);
  Pixel sourcePixel = null;
  Pixel targetPixel = null;

  // loop through the columns
  for (int sourceX = 0, targetX=0;
      sourceX < sourcePicture.getWidth();
      sourceX++, targetX++)
  {
```


Copy Method - Continued

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


Trying the copyKatie Method

```
> String fileName = FileChooser.getMediaPat
    h("7inx95in.jpg");
> Picture targetPicture = new Picture(fileName);
> targetPicture.show();
> targetPicture.copyKatie();
> targetPicture.show();
```

Tracing copyKatie

- The first two lines create the sourcePicture
- Next we declare variables we will need in the loop
  - sourcePixel and targetPixel
- Next the outer loop declares and initializes variables for keeping track of the x values and loops while the sourceX is less than the width of the sourcePicture
  - sourceX and targetX
- The inner loop declares and initializes variables for keeping tack of the y values and loops while the sourceY is less than the height of the sourcePicture
  - sourceY and targetY
- In the loop set the color in the target pixel to the color in the source pixel

Copying Pixels to a New Picture

- What if we want to copy the target to a different location in the source
  - Than 0,0
  - Say startX and startY
- What is an algorithm that will do this?

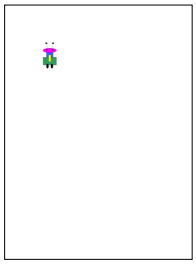
	0	1
0	1	2
1	3	4

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

Copy to Position Exercise

- Write a method copyRobot to copy
  - robot.jpg
  - To location
    - 100, 100 in 7inx95in.jpg
- Test with

String file = FileChooser.getMediaPath ("7inx95in.jpg");
Picture p = new Picture(file);
p.copyRobot();
p.show();



Cropping

- We can copy just part of a picture to a new picture
  - Just change the start and end source x and y values to the desired values
  - Use pictureObj.explore() to find the x and y values
  - What are the x and y values to get the face of the girl in KatieFancy.jpg?

Copy Face Method

```
public void copyKatiesFace()
 {
  String sourceFile =
    FileChooser.getMediaPath("KatieFancy.jpg");
  Picture sourcePicture = new Picture(sourceFile);
  Pixel sourcePixel = null;
  Pixel targetPixel = null;

  // loop through the columns
  for (int sourceX = 70, targetX = 100;
     sourceX < 135; sourceX++, targetX++)
    {
```


Copy Face Method - Continued

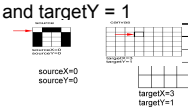
```
    // loop through the rows
    for (int sourceY = 3, targetY = 100;
        sourceY < 80; sourceY++, targetY++) {
    {
      // set the target pixel color to the source pixel color
      sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
   }
 }
```


Testing Copy Katie's Face

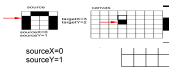
```
> String fileName = FileChooser.getMediaPat
  h("7inx95in.jpg");
> Picture targetPicture = new Picture(fileName);
> targetPicture.copyKatiesFace();
> targetPicture.show();
```

How does that work?

- We start by copying from sourceX=0, sourceY=0 to targetX = 3 and targetY = 1

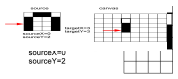


- Increment both the sourceY and targetY and copy again

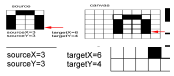


How does that work? - continued

- We continue down the column, incrementing both the source and target index variables

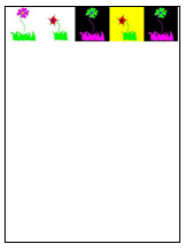


- When we finish one column we increment both the source and target X values and continue with next column until we copy every pixel



Creating a Collage

- You can create a collage by copying several pictures to a blank picture
  - You can use flower1.jpg and flower2.jpg
  - You can even negate the picture or clear blue on it before you copy it



Create Collage Method

```
/**
 * Method to copy flower pictures to create a
 * collage.
 * All the flower pictures will be lined up near the
 * top of the current picture
 */
public void copyFlowersTop()
{
  // create the flower pictures
  Picture flower1Picture = new Pictur
      e(FileChooser.getMediaPath(
    "flower1.jpg"));
  Picture flower2Picture = new Pictur
      e(FileChooser.getMediaPath(
    "flower2.jpg"));

  // declare the source and target pixel variables
  Pixel sourcePixel = null;
  Pixel targetPixel = null;

  // copy the first flower picture to the top left
  // corner
  for (int sourceX = 0, targetX = 0;
       sourceX < flower1Picture.getWidth();
       sourceX++, targetX++)
  {
    for (int sourceY = 0, targetY = 0;
         sourceY < flower1Picture.getHeight();
         sourceY++, targetY++)
    {
      sourcePixel = flower1Picture.getPixe
        l(sourceX,sourceY);
      targetPixel  = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
```


Create Collage Method - cont

```
// copy the flower2 picture starting with x =
// 100
  for (int sourceX = 0, targetX = 100;
       sourceX < flower2Picture.getWidth();
       sourceX++, targetX++)
  {
    for (int sourceY = 0, targetY = 0;
         sourceY < flower2Picture.getHeight();
         sourceY++, targetY++)
    {
      sourcePixel = flower2Picture.getPixe
        l(sourceX,
                                  sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }

// copy the flower1 negated to x = 200
  flower1Picture.negate();
  for (int sourceX = 0, targetX = 200;
       sourceX < flower1Picture.getWidth();
       sourceX++, targetX++)
  {
    for (int sourceY = 0, targetY = 0;
         sourceY < flower1Picture.getHeight();
         sourceY++, targetY++)
    {
      sourcePixel = flower1Picture.getPixe
        l(sourceX,
                                  sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
```


Create Collage Method - cont

```
// clear the blue in flower 2 picture and add at
// x=300
flower2Picture.clearBlue();
for (int sourceX = 0, targetX = 300;
     sourceX < flower2Picture.getWidth();
     sourceX++, targetX++)
{
  for (int sourceY = 0, targetY = 0;
       sourceY < flower2Picture.getHeight();
       sourceY++, targetY++)
  {
    sourcePixel = flower2Picture.getPixe
      l(sourceX,sourceY);
    targetPixel = this.getPixel(targetX,targetY);
    targetPixel.setColor(sourcePixel.getColor());
  }
}

// copy the negated flower 1 to x=400
for (int sourceX = 0, targetX = 400;
     sourceX < flower1Picture.getWidth();
     sourceX++, targetX++)
{
  for (int sourceY = 0, targetY = 0;
       sourceY < flower1Picture.getHeight();
       sourceY++, targetY++)
  {
    sourcePixel = flower1Picture.getPixe
      l(sourceX,
                                sourceY);
    targetPixel = this.getPixel(targetX,targetY);
    targetPixel.setColor(sourcePixel.getColor());
  }
}
```


That is one long method!

- Isn't there a better way to do that?
  - Much of the code is repeated
- You can make a new method with just the repeated code
  - Use parameters to pass in the values that change
    - Picture to copy
    - X location to start the copy at in the target

General Copy Method

```
public void copyPicture(Picture sourcePicture,  int xStart)
{
 Pixel sourcePixel = null;
 Pixel targetPixel = null;

 // loop through the columns
 for (int sourceX = 0, targetX = xStart;
     sourceX < sourcePicture.getWidth();
     sourceX++, targetX++)
 {
  // loop through the rows
  for (int sourceY =0, targetY = 0;
      sourceY < sourcePicture.getHeight();
      sourceY++, targetY++)
```


General Copy Method - continued

```
  {
   sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
   targetPixel = this.getPixel(targetX,targetY);
   targetPixel.setColor(sourcePixel.getColor());
  }
 }
}
```


Better Collage Method

```
public void copyFlowersBetter()
{
 // create the flower pictures
 Picture flower1Picture =
   new Picture(FileChooser.getMediaPath("flower1.jpg"));
 Picture flower2Picture =
   new Picture(FileChooser.getMediaPath("flower2.jpg"));

 // copy the first flower picture
 this.copyPicture(flower1Picture,0);
```


Better Collage Method - continued

```
 // copy the flower2 picture starting with x = 100
 this.copyPicture(flower2Picture,100);

 // copy the flower1 negated to x = 200 in the canvas
 flower1Picture.negate() ;
 this.copyPicture(flower1Picture,200);

 /* clear the blue in flower 2 picture and add at x=300 in the canvas */
 flower2Picture.clearBlue();
 this.copyPicture(flower2Picture,300);

 // copy the negated flower 1 to x=400
 this.copyPicture(flower1Picture,400);
}
```


Challenge

- Create your own collage
  - Copy at least two different pictures to the collage
  - Do at least 3 different picture manipulations to the pictures
    - Reduce red
    - Negate
    - Clear blue
  - Mirror the collage

Adding more Parameters

```
public void copyPicture(Picture sourcePicture,
                        int xStart,
                        int yStart)
{
 Pixel sourcePixel = null;
 Pixel targetPixel = null;

 // loop through the columns
 for (int sourceX = 0, targetX = xStart;
     sourceX < sourcePicture.getWidth();
     sourceX++, targetX++)
 {
```

Adding more Parameters - continued

```
    // loop through the rows
    for (int sourceY = 0,
        targetY = yStart;
        sourceY < sourcePicture.getHeight();
        sourceY++, targetY++)
    {
      sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```


Method Overloading

- Notice that you have two copyPicture methods
  - One takes a Picture object and an integer
  - One takes a Picture object and two integers
- This is called method overloading
  - Having methods with the same name
  - But with a different number, kind, or order of parameter types

Blend Pictures

- If we want to blend two pictures we need to take 50% of the color from one picture and add it to 50% of the color from the other picture
  - So 50% of the red, green, and blue from each

Blend Pictures Algorithm

- Create the two pictures to blend
- Copy the pixels from the first part of picture1
  - First 150 columns from picture1
    - X loops from 0 and stops when equal to 150
    - Y loops from 0 and stops when equal to the picture1 height
- Copy the rest of the columns from picture1 blended with the pixels from picture2
  - Set the color to a new color that is a combination of half of the red from each pictures, half of the green from each picture and half of the blue from each picture
  - X loops from 150 to the width of picture1
  - Y loops from 0 to the height of picture1
- Copy the rest of the pixels from picture2

Blend Pictures Method

```
public void blendPictures()
 {
  // create the sister pictures
  Picture katiePicture =
    new Picture(FileChooser.getMediaPath("KatieFancy.jpg"));
  Picture jennyPicture =
    new Picture(FileChooser.getMediaPath("JenParty.jpg"));

  // declare the source and target pixel variables
  Pixel katiePixel = null;
  Pixel jennyPixel = null;
  Pixel targetPixel = null;

  /* declare the target x and source x since we will need
   * the values after the for loop
   */
  int sourceX = 0;
  int targetX = 0;
```


blendPictures() continued

```
  // copy the first 150 pixels of katie to the canvas
  for (; sourceX < 150; sourceX++, targetX++)
  {
    for (int sourceY=0, targetY=0;
        sourceY < katiePicture.getHeight();
        sourceY++, targetY++)
    {
      katiePixel = katiePicture.getPixel (sourceX,sourceY) ;
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(katiePixel.getColor());
    }
  }

 /* copy 50% of katie and 50% of jenny till the end of katie's width */
  for (; sourceX < katiePicture.getWidth();
      sourceX++, targetX++)
  {
```

blendPictures() continued

```
for (int sourceY=0,targetY=0;
       sourceY < katiePicture.getHeight();
       sourceY++, targetY++)
  {
    katiePixel = katiePicture.getPixel(sourceX,sourceY);
    jennyPixel =
      jennyPicture.getPixel(sourceX - 150,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(
        new Color((int)   (katiePixel.getRed() * 0.5 +
                  jennyPixel.getRed() * 0.5),
              (int) (katiePixel.getGreen() * 0.5 +
                  jennyPixel.getGreen() * 0.5),
              (int) (katiePixel.getBlue() * 0.5 +
                  jennyPixel.getBlue() * 0.5)));
  }
 }
```


blendPictures() continued

```
 // copy the rest of Jenny
 sourceX = sourceX - 150;
 for (; sourceX < jennyPicture.getWidth();
      sourceX++, targetX++)
 {
 for (int sourceY = 0, targetY = 0;
      sourceY < jennyPicture.getHeight();
      sourceY++, targetY++)
   {
     jennyPixel = jennyPicture.getPixel(sourceX,sourceY);
     targetPixel  = this.getPixel(targetX,targetY);
     targetPixel.setColor(jennyPixel.getColor());
   }
 }
}
```


Trying out blendPictures()

```
> String fileName =
    FileChooser.getMediaPat
  h("640x480.jpg");
> Picture picture = new Picture(fileName);
> picture.blendPictures();
> picture.show();
```


For Loop Syntax

- The general for loop syntax is:
  for (init area ; continuation test; change area) {
- Each part of this loop is optional
  - But the semicolons are required
    - You can only have 2 semicolons
- We didn't put anything in the init area since we wanted to keep using the values of sourceX and targetX

Challenge

- Write a method to blend two pictures together
  - And only use 25% of one picture's color added to 75% of the other picture's color
- Also blend the entire pictures together
  - Best to use two pictures of the same size

Left Rotation

- How can you copy one picture onto another so that the first picture is rotated to the left 90 degrees?

Left Rotation

- First simplify the problem by thinking about how to copy when each pixel has just a number at it
- Can you come up with an algorithm for this?

	0	1	2
0	1	2	3
1	4	5	6

	0	1
0	3	6
1	2	5
2	1	4

Left Rotation

- Try out your algorithm on another example
  - Does it work?
- Can you translate it into code?

	0	1	2	3
0	5	6	7	8
1	1	2	3	4

	0	1
0		
1		
2		
3		

Left Rotation

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

	0	1	2
0	1	2	3
1	4	5	6

	0	1
0	3	6
1	2	5
2	1	4

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

Left Rotation Method

```
public void copyKatieLeftRotation()
 {
   String sourceFile =
     FileChooser.getMedi aPath("KatieFancy.jpg");
   Picture sourcePicture = new Picture(sourceFile);
   Pixel sourcePixel = null;
   Pixel targetPixel = null;

   // loop through the columns
   for (int sourceX = 0;
       sourceX < sourcePicture.getWidth();
       sourceX++)
   {
```


Copy Katie Left Rotation

```
    // 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);
      targetPixel = this.getPixel(sourceY,
              sourcePicture.getWidth() - 1 - sourceX);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```

Testing Left Rotation

```
String fileName = FileChooser.getMediaPath(
                  "7inX95in.jpg");
Picture picture = new Picture(fileName);
picture.show();
picture.copyKatieLeftRotation();
picture.repaint();
```


Challenge

- Create a general left rotation method that
  - Works on any picture
    - Call the method on the picture to be rotated left
  - Returns a new picture that is just the right size to hold the rotated picture
    - Make the new picture's width the same as the old picture's height and the new picture's height the same as the old picture's width
      - You can use new Picture(width,height) to create a picture of a given width and height

Scaling

- You can make a picture smaller
  - Faster to download on the web
    - Increment the source x and y by a number larger than 1
      - Don't use all the source pixels in the target picture
- You can make a picture larger
  - Show more detail
    - Copy the same source x and y to more than one target x and y
      - Use source pixels more than once in target

Scaling Down a Picture

- jakita.jpg is 768 pixels wide and 768 pixels high

- If we copy every other pixel we will have a new picture with width and height (768 / 2 = 384)

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	2
8	10

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

Scaling Down Method

```
public void copyJakitaSmaller()
{
  Picture jakitaPicture =
    new Picture(FileChooser.getMediaPath("jakita.jpg"));
  Pixel sourcePixel = null;
  Pixel targetPixel = null;

  // loop through the columns
  for (int sourceX = 0, targetX=0;
      sourceX < jakitaPicture.getWidth();
      sourceX= sourceX + 2, targetX++)
  {
```

Scaling Down Method - Continued

```
  // loop through the rows
  for (int sourceY=0,   targetY=0;
      sourceY < jakitaPicture.getHeight();
      sourceY= sourceY + 2,   targetY++)
  {
    sourcePixel  = jakitaPicture.getPixel(sourceX,sourceY);
    targetPixel = this.getPixel(targetX,targetY);
    targetPixel.setColor(sourcePixel.getColor());
  }
 }
}
```


Trying copyJakitaSmaller

```
> Picture p =
      new Picture(FileChooser.getMediaPath("640x480.jpg"));
> p.copyJakitaSmaller() ;
> p.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)

	0	1	2
0	1	2	3
1	4	5	6

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

Scaling Up 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 0.5 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 0.5 each time through the loop, increment the target y by 1
        - » Copy the color from the source to target pixel

Method to scale up a flower

```
public void copyFlowerLarger()
{
  Picture flowerPicture =
    new Picture(FileChooser.getMediaPath("rose.jpg"));
  Pixel sourcePixel = null;
  Pixel targetPixel = null;

  // loop through the columns
  for (double sourceX = 0, targetX=0;
      sourceX < flowerPicture.getWidth();
      sourceX = sourceX + 0.5, targetX++)
  {
```


Method to scale up a flower - continued

```
   // loop through the rows
   for (double sourceY=0, targetY=0;
       sourceY <   flowerPicture.getHeight();
       sourceY = sourceY +0.5, targetY++)
   {
     sourcePixel =
       flowerPicture.getPixel((int)  sourceX,(int)  sourceY);
     targetPixel  = this.getPixel((int)  targetX,(int) targetY);
     targetPixel.setColor(sourcePixel.getColor());
   }
  }
}
```

Trying out the method

```
> String fileName =
   FileChooser.getMediaPat
  h("640x480.jpg");
> Picture picture = new Picture(fileName);
> picture.copyFlowerLarger();
> picture.show();
```


How does that work?

- This is similar to the copyPicture method
  - But we declare sourceX and sourceY to be of type double and increment by 0.5 each time
  - X and y values can't be double so we truncate by casting to integer
    - Throws away any fractional part
    - So a value of 0.5 results in 0
      - A value of 1.5 results in 1
- Would this algorithm work for scaling up by a factor of 3?

Making a more general scaling method

- You can create a blank new picture
  - By specifying the desired width and height
  - new Picture(width,height)
- You can then copy to the new picture
  - And return it
  - Be sure to save a reference to the returned picture when you call the method

General Scale Up Method

```
public Picture scaleUp(int numTimes)
{
  Picture targetPicture =
    new Picture(this.getWidth() * numTimes,
            this.getHeight()* numTimes);
  Pixel sourcePixel = null;
  Pixel targetPixel = null;
  int targetX = 0;
  int targetY = 0;

  // loop through the source picture columns
  for (int sourceX = 0;
      sourceX < this.getWidth();
      sourceX++)
  {
```


General Scale Up Method – cont 1

```
  // loop through the source picture rows
  for (int sourceY=0;
      sourceY < this.getHeight();
      sourceY++)
  {
    // get the source pixel
    sourcePixel = this.getPixel(sourceX,sourceY);

    // loop copying to the target y
    for (int indexY = 0; indexY < numTimes; indexY++)
    {
      // loop copying to the target x
      for (int indexX = 0; indexX < numTimes; indexX++)
      {
```


General Scale Up Method – cont 2

```
        targetX = sourceX * numTimes + indexX;
        targetY = sourceY * numTimes + indexY;
        targetPixel = targetPicture.getPixel(targetX,
                                    targetY);
        targetPixel. setColor(sourcePixel .getColor());
      }
    }
  }
  return targetPicture;
}
```

Trying out the scaleUp method

```
> Picture p =
   new Picture(FileChooser.getMediaPath("flower1.jpg"));
> p = p.scaleUp(2);  // change what p refers to
> p.explore();
```

Or create a new variable to refer to the returned picture

```
> String fileName = FileChooser.getMediaPat
  h("flower1.jpg") ;
> Picture origPicture = new Picture(fileName);
> Picture scaledPicture = origPicture.scaleUp(2);
> scaledPicture.show();
> origPicture.show();
```


Challenge

- Write a general scale down method
  - That returns the resulting picture
  - Remember that you can use
    - new Picture(width, height) to create a blank picture of a given width and height
    - And use the return statement to return a value from a method
    - You will need to save the result from calling the method

Summary

- Nested loops can be used to loop through a two-dimensional array
  - And keep track of the current x and y values
- Create several small versions of a problem
  - And solve those before you try to code a programming solution
  - Try to determine the general algorithm from the concrete small versions
  - Translate the algorithm into code