Modifying Pictures Using Loops

Barb Ericson
Georgia Institute of Technology
Oct 2010

Learning Goals

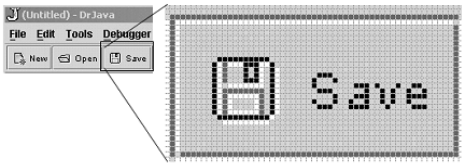
- Understand at a conceptual and practical level
  - How to manipulate digital pictures?
  - How to work with one and two-dimension arrays
  - How to write object methods
  - Review iteration using for-each, while, and for loops
  - What the scope is for a variable name

Digital Pictures

- Represented by pixels (picture elements)
  - With a red, green, and blue value stored for each pixel with 8 bits per color (a range of 0 to 255)
- Stored in .jpg (JPEG) files
  - International standard
  - With lossy compression
    - Lossy means not all data is stored
      - But what is lost isn't that important
    - Compression means made smaller
- Other formats for storing digital pictures are GIFF and BMP

Pixels – Picture Elements

- Small dots of color that make up digital pictures

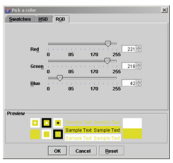


Color Objects

- There is a class defined in Java that represents color
  - The Color class in the package java.awt
  - To use the class you must either
    - import java.awt.Color;
    - Use the full name java.awt.Color
- You can create a color object by giving the red, green, and blue values for it
  - Color colorObj = new Color(255,10,125);

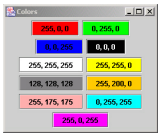
Mixing red, green, and blue

- You can use a ColorChooser to see how red, green, and blue mix to make all the colors
  - ColorChooser.pickAColor()
  - Try to make different colors
    - Yellow
    - Green
    - Pink
    - Black
    - White
    - Brown



Predefined Colors

- The Color class has defined class constants for many colors
  - Color.RED, Color.GREEN, Color.BLUE, Color.BLACK, Color.WHITE, Color.YELLOW, Color.GRAY, Color.ORANGE, Color.PINK, Color.CYAN, Color.MAGENTA
  - Or you can use all lowercase names
    - Color.red, Color.blue, Color.black, …



Pictures have lots of Pixels

- How can we refer to each pixel?
  - pixel1, pixel2, pixel3, pixel4, pixel5, …
- Do we really want to name each one?
  - There are 640 x 480 = 307,200 pixels
- How do we deal with lots of data of the same type?
  - Use an array
  - Like a list but of a fixed size and with all the data contiguous in memory

What is an Array?

- Storage for a sequence of items
  - Of the same type
- You can access items by using an index
  - int x = arrayRef[index];
- The index starts at 0
  - The first item is at index 0
  - The last item is at index (length – 1)
- Arrays know their length (have a public length field)
  - arrayObj.length

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

0	1	2	3
8	3	2	6

Declaring and Creating an Array

- Declare an array using
  - *Type*[] *name* or *Type name*[];
  - Pixel[] pixelArray = null;
  - double grades[];
- Create an array using
  - new *Type*[*numberOfElements*];
  - new int[5];
  - int numArray = new int[5]; // all values are 0
- Initialize array elements using
  - double[] gradeArray = {80, 90.5, 88, 92, 94.5};
  - int grade = gradeArray[2] // grade = 88

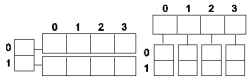
Two-Dimensional Arrays (Matrix)

- Pictures are actually two-dimensional arrays
  - Have a horizontal (x or column) and vertical (y or row) value for each pixel

	0	1	2	3
0	15	12	13	10
1	9	7	43	23
2	6	13	15	16

2D Arrays in Java

- Elements can be accessed with
  - arrayRef[x][y] or arrayRef[col][row]
  - arrayRef[y][x] or arrayRef[row][col]
- It depends how the data was initialized
  - We use arrayRef[x][y] in our Picture class



Manipulating a Picture

- To manipulate a picture we need to manipulate the pixels that make up the picture
  - Change the red, green, or blue values at the pixel
- Pixel is a class that we created at Georgia Tech
  - Each pixel object has a red, green, and blue value that ranges from 0 to 255

What Data does a Picture Object Have?

- It knows the picture width
  pictureObj.getWidth()
- It knows the picture height
  pictureObj.getHeight()
- It knows how to return an array of pixels
  Pixel[] pixelArray = *pictureObj*.getPixels()
- It knows how to return a pixel at a location (x,y)
  Pixel p = pictureObj.getPixel(0,0);

Pixel Objects

- Each pixel has a red, green, and blue value
  - getRed(), getGreen(), getBlue()
  - setRed(int v), setGreen(int v), setBlue(int v)
- Each pixel knows the location it was in the picture object
  - getX(), get(Y)
- You can also get and set the color at the pixel
  - Color currColor = pixelObj.getColor()
  - pixelObj.setColor(Color theColor)

Modifying the Pixels in a Picture

- We can get a 1D array of pixel objects from a picture
  - Pixel[] pixelArray = this.getPixels();
  - This gets all the pixels in the first row followed by all the pixels in the second row and so on.
- We can use a for-each loop to loop through all the pixels in a picture
  - Like looping through an Alice list

Increase Red Algorithm

- How do we increase the amount of red in a digital picture?
  - By twice the original amount?
- Loop through all the pixels in the picture
  - Get a 1D array of pixels in the picture
  - Loop through all the pixels
    - Get the current red at the current pixel
    - Set the red at the current pixel to twice the original value

Increase Red Method

```
public void increaseRed()
{
  Pixel[] pixelArray = this.getPixels();
  int value = 0;

  // loop through all the pixels  in the array
  for (Pixel pixelObj : pixelArray)
  {
    // get the current red value
    value = pixelObj.getRed();

    // double the red
    value = value * 2;

    // set the red value of the current pixel to the new value
    pixelObj.setRed(value);
  }
}
```

Testing the Increase Red Method

```
String fName =
   "mediasources/caterpillar.jpg";
Picture pict = new Picture(fName);
pict.explore();
pict.increaseRed();
pict.explore();
```


How it works

- When we execute pict.increaseRed()
  - The **this** keyword refers to the same picture object as pict
  - We get a one dimensional array of pixels from the current picture and refer to it as pixelArray
  - The first time through the loop pixelObj will refer to the first pixel in the picture
    - pixelArray[0] or getPixel(0,0)
  - The last time through the loop pixelObj will refer to the last pixel in the picture
    - pixelArray.length – 1 or getPixel(328,149)

Challenge

- Write a method to decrease the amount of red in a picture
  - Call it decreaseRed()
  - Divide the current red value by 2 for each pixel in the picture
- Write a method to set all the blue values in a picture to 0
  - Call it clearBlue()

Loop Exercise

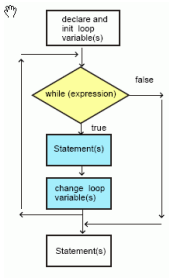
- Ask a person to clap 12 times
  - How does s/he know when to stop?
  - What changes each time s/he claps?
- If you are following a recipe that asks you to stir the ingredients 50 times how would you do this?
- What if you were trying to break a sit-up record
  - How would you know if you did break it?

Loops often need Counters

- If you want to do something x times you often need a counter
  - That starts at 0
  - And you add 1 to it each time you finish doing the thing you are repeating
  - When the counter reaches the number you are trying to do you stop the loop
    - What is the value of the counter the last time the statements of the loop are executed?

While Loops

- In Java one way to repeat a block of statements while an expression is true is to use a while loop
- Create a counter and set it to the start value
- Check that the counter is less then the stop value
- If it is less than execute the statements in the loop
- Add one to the counter and go back to check that the counter is less than the stop value



Total the Numbers from 1 to 100

- What if you want to add all the numbers from 1 to 100?
  - You will need something to hold the total
    - What type should it be?
    - What value should it start out with?
  - You will need something that counts from 1 to 100
    - And add that value to the total
    - Stop when you get to 100
    - What type should it be? What value should it start with?

While Loop Syntax

- Adding up the numbers from 1 to 100

```
int total = 0;
int num = 1;
while (num <= 100)
{
    total = total + num;
    num = num + 1;
}
System.out.println(total);
```


While Loop Syntax

- Adding up the numbers from 1 to 100

```
int total = 0;  // declare and initialize the total
int num = 1; // declare and init the number

while (num <= 100) // do while num <= 100
{
    total = total + num; // add num to total
    num = num + 1;     // increment the num
}
System.out.println(total);  // print the total
```


Parts of a While Loop

- Adding up the numbers from 1 to 100

```
int total = 0;
int num = 1;
while (num <= 100)
{
    total = total + num;
    num = num + 1;
}
System.out.println(total);
```

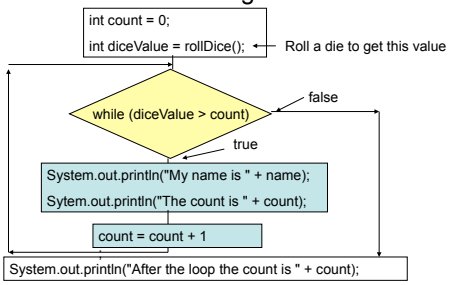
Declaration and initialization of variables

This test is done each time and when It is true the loop body will be executed

This is the body of the loop. It Starts with a '{' and ends with a '}'. If there is just one statement In a loop body the '{' and '}' aren't needed.

Exercise

- Have students walk through this flowchart



Changing from For-each to While

- In a for-each loop something needs to keep track of the current pixel
  - And change each time through the loop
  - To sure that we have gone through all of the pixels
- We can loop through all elements in an array by starting with index 0, then index 1, and so on till index (length – 1)
  - And get the pixel at the current index value

Decrease Red Algorithm

- Get the array of Pixel objects from the current picture
- Declare a variable to hold the red value
- Declare the index variable and set it to 0
- Declare a variable to refer to the current pixel
- Loop while index is less than the length of the array
  - Get the pixel at the index value
  - Get the current red value from the pixel
  - Divide the current red value by 2
    - Or multiply by 0.5 and change back to integer
  - Set the red for the current pixel to the changed value
  - Increment the index

Loop Algorithm to Code

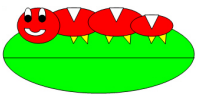
- How to write (code) the loop?
  - Use a while loop with a counter for the index starting at 0
    int index = 0;
  - Add a variable to refer to the current pixel
    Pixel pixelObj = null;
  - Loop while the index is less than the length of the array
    while (index < pixelArray.length)
  - Get the current pixel from the array of pixels for the current index
    pixelObj = pixelArray[index];

Loop Algorithm to Code - Continued

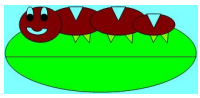
- Get the red value at the pixel
  value = pixelObj.getRed();
- Multiply by 0.5 and convert back to integer
  value = (int) value * 0.5;
- Set the pixel red value
  pixel.setRed(value);
- Add one to (increment) the index
  - index = index + 1;

decreaseRed Method Result

- Before method



- After method



Decrease Red Method

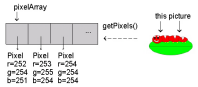
```
public void decreaseRedWhile()
{
 Pixel[] pixelArray = this.getPixels();
 Pixel  pixel  = null;
 int value = 0;
 int index = 0;

 // loop through all the pixels
 while(index < pixelArray.length)
 {
   // get the current pixel
   pixel = pixelArray[index];
   // get the value
   value = pixel.getRed();

   // decrease the red value by 50%
   value = (int)   (value * 0.5);

   // set the red value of the current
   // pixel to the new value
   pixel.setRed(value);

   // increment the index
   index = index + 1;
 }
}
```


Tracing the code

- If you test this with
  > String fName =
  "C:/intro-prog-java/mediasources/caterpillar.jpg";
  > Picture picture = new Picture(fName);
  > picture.explore();
  > picture.decreaseRedWhile();
  > picture.explore();
- How does that work?
  - picture.decreaseRedWhile() means execute the *decreaseRedWhile* method in the Picture class passing in the picture of the caterpillar as *this*

Tracing the code – continued

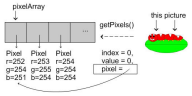
- Pixel[] pixelArray **= this**.getPixels();
  - Set *pixelArray* to refer to a one dimensional array of pixels from the current picture



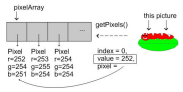
- Declare some variables we will need in the loop: index, value, and pixel

Tracing the code - continued

- Start the loop. Index is 0 which is less than the length of the array.
  - Set pixel to the Pixel object at the current index in the array

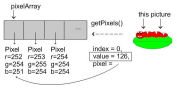


  - Set value to the red value in that pixel

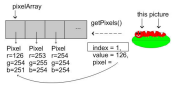


Tracing the code - continued

- Set value to the result of multiplying the current value by 0.5 which returns a double result and throw away the fractional part by casting to integer
  - value = (int) (value * 0.5);

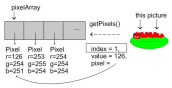


- Set the red in the pixel to the new value
  - pixel.setRed(value);

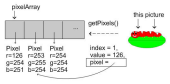


Tracing the code - continued

- Increment the index by 1
  - index = index + 1

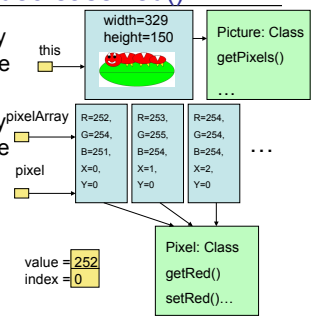


- Check if index is less than the array length and if so repeat the body of the loop
  - This time for the 2nd pixel



Memory Map of decreaseRed()

- What does memory look like the 3rd time through?
- What does memory look like the 4th time through?
- How about the last time through?



Clearing the blue value

- You can even set all the blue to 0 in the whole picture
  - Which makes the white areas turn yellow

clearBlue method

```
public void clearBlue()
{
 Pixel[] pixelArray = this.getPixel
    s();
 Pixel  pixel  = null;
 int index = 0;

 // loop through all the pixels
 while (index < pixelArray.length)
 {
  // get the current pixel
  pixel = pixelArray[index];
  // set the blue on the pixel to 0
  pixel.setBlue(0);

  // increment index
  index++;
 }
}
```


Testing clearBlue

```
> String fName =
    "C:/intro-prog-java/mediasources/caterpillar.jpg";
> Picture picture = new Picture(fName);
> picture.explore();
> picture.clearBlue();
> picture.explore();
```


Faking a Sunset

- If you want to make an outdoor scene look like it happened during sunset
  - You might want to increase the red
    - But you can't increase past 255
  - Another idea is to reduce the blue and green
    - To emphasize the red
    - Try to reduce the blue and green by 30%

Faking a Sunset Algorithm

- Reduce the blue and green by 30%
  1. Get the array of pixels from the picture
  2. Set up an index to start at 0
  3. Loop while the index is less than the length of the array
     1. Get the pixel at the current index from the array of pixels
     2. Set the blue value at the pixel to 0.7 times the original value
     3. Set the green value at the pixel to 0.7 times the original value
     4. Increment the index and go back to step 3

Faking a Sunset Method

```
public void makeSunset()
{
 Pixel [] pixelArray = this.getPixels();
 Pixel pixel = null;
 int value = 0;
 int i = 0;

 // loop through all the pixels
 while (i < pixelArray.length)
 {
  // get the current pixel
  pixel = pixelArray[i];
  // change the blue value
  value = pixel.getBlue();
  pixel.setBlue((int) (value * 0.7));

  // change the green value
  value = pixel.getGreen();
  pixel.setGreen((int) (value * 0.7));

  // increment the index
  i++;
 }
}
```


Testing makeSunset

```
String file =
    "c:/intro-prog-java/mediasources/beach-smaller.jpg";
Picture pictureObj = new Picture(file);
pictureObj.explore();
pictureObj.makeSunset();
pictureObj.explore();
```

Variable Scope

- In makeSunset we used i for the index
  - Programmers often use short names like i to stand for an index
- Scope – where a variable name is understood and can be used
  - Variables declared inside of a method have scope only inside that method
    - The same name can be used by different methods
  - Variables declared in the interactions pane are only known in the interactions pane

For Loops

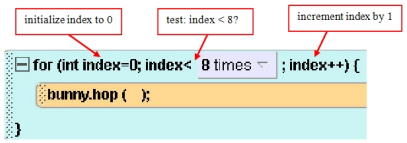
- Programmers like shortcuts
  - Especially those that reduce errors
  - And mean less typing
- We have been using a while loop with an index
  - We had to declare the index variable and initialize it before the loop
    - If you forget this there will be a compiler error
  - We had to increment the index in the loop
    - If you forget this it will be an infinite loop
- The shortcut for this is a for loop

For Loop Syntax

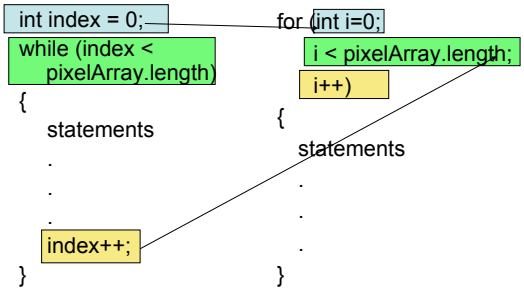
- for (initialization area; continuation test; change area)
  - Initialization area
    - Declare variables and initialize them
  - Continuation test
    - If true do body of loop
    - If false jump to next statement after the loop
  - Change area
    - Change the loop variables
      - Increment or decrement them

For loop in Alice

- We used a for loop to make a bunny hop 8 times



Comparison of While and For Loops



clearBlue() using a For Loop

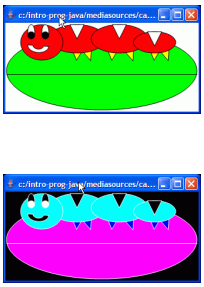
```
public void clearBlueFor()
{
 Pixel [] pixelArray = this.getPixels();

 // loop through all the pixels
 for (int i=0; i < pixelArray.length; i++)
    pixelArray[i].setBlue(0);
}
```

Change to For Loop Exercise

- Edit makeSunset() and change it from using a while loop to using a for loop
  - Move the declaration of the index to the for loop initialization area
  - Move the index increment to the for loop change area
  - Execute the code to make sure it still works

Negating an Image

- How would you turn a picture into a negative?
  - White should become black
    - 255,255,255 becomes 0,0,0
  - Black should become white
    - 0,0,0 becomes 255,255,255



Negate Algorithm

- Subtract current value from 255 for red, green, and blue
  1. Get the array of pixels from the picture
  2. Declare variables to hold the current pixel and the red, green, and blue values
  3. Loop starting an index at 0 and incrementing by 1 and loop while the index is less than the length of the array
     1. Get the pixel at the current index from the array of pixels
     2. Set the red value to 255 – current red value
     3. Set the blue value to 255 – current blue value
     4. Set the green value to 255 – current green value
     5. Increment the index and go back to step 3

Negate Method

```
public void negate()
{
 Pixel[] pixelArray = this.getPixels();
 Pixel pixel = null;
 int redValue = 0, blueValue = 0;
 int greenValue = 0;

 // loop through all the pixels
 for (int i = 0; i < pixelArray.length; i++)
 {
  // get the current pixel
  pixel = pixelArray[i];

  // get the current red, green, and blue values
  redValue = pixel.getRed();
  greenValue = pixel.getGreen();
  blueValue = pixel.getBlue();

  // set the pixel's color to the new color
  pixel.setColor(new Color(
                 255 - redValue,
                 255 - greenValue,
                 255 - blueValue));
 }
 }
```


Changing to Grayscale

- Grayscale ranges from black to white
  - The red, green, and blue values are the same
- How can we change any color to gray?
  - What number can we use for all three values?
    - The intensity of the color
  - We can average the colors
    - (red + green + blue) / 3
  - Example
    - (15 + 25 + 230) / 3 = 90

Grayscale Algorithm

- Set color values to the average of the original values
  1. Get the array of pixels from the picture
  2. Declare variables to hold the current pixel and the red, green, and blue values
  3. Loop starting an index at 0 and incrementing by 1 and loop while the index is less than the length of the array
     1. Get the pixel at the current index from the array of pixels
     2. Calculate the average of the current values
        1. (redValue + greenValue + blueValue)/ 3
     3. Set the red value to the average
     4. Set the blue value to the average
     5. Set the green value to the average
     6. Increment the index and go to step 3

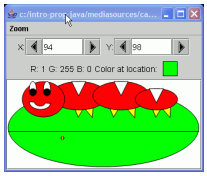
Grayscale Method

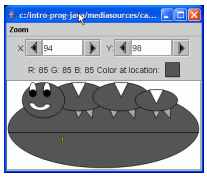
```
public void grayscale()
{
 Pixel[] pixelArray = this.getPixels();
 Pixel pixel = null;
 int intensity = 0;

 // loop through all the pixels
 for (int i = 0; i < pixelArray.length; i
    ++)
 {
  // get the current pixel
  pixel = pixelArray[i];

  // compute the intensity of the pixel
  // (average value)
  intensity = (int) ((pixel.getRed() +
                      pixel.getGreen() +
                      pixel.getBlue()) / 3);

  // set the pixel color to the new color
  pixel.setColor(new Colo
     r(intensity,intensity,intensity));
 }
}
```


Testing Grayscale

String file =
"c:/intro-prog-java/mediasources/caterpillar.jpg";
Picture pictureObj = new Picture(file);
pictureObj.explore();
pictureObj.grayscale();
pictureObj.explore();

Grayscale Result





Luminance

- We perceive blue to be darker than red, and green
  - Even when the same amount of light is reflected
- A better grayscale model should take this into account
  - Weight green the highest (* 0.587)
  - red less (* 0.299) and
  - blue the very least (* 0.114)

Grayscale with Luminance Exercise

- Create a new method grayscaleWithLuminance
- Using the new algorithm for calculating intensity
- intensity = (int) (red * 0.299 + green * 0.587 + blue * 0.114)

Testing Grayscale with Luminance

String file =
"c:/intro-prog-java/mediasources/caterpillar.jpg";
Picture pictureObj = new Picture(file);
pictureObj.explore();
pictureObj.grayscaleWithLuminance();
pictureObj.explore();

Summary

- Pictures have pixels
  - You can change the picture by changing the color of the pixels
- Arrays let you store and retrieve values of the same type using an index
- You can ask a picture for it's width, height, and an array of pixels
- You can get and set the color of a pixel
- You can use for-each, while, and the general for loop to iterate