AP® Computer Science A Picture Lab Student Guide

The AP Program wishes to acknowledge and thank Barbara Ericson of the Georgia Institute of Technology, who developed this lab and the accompanying documentation.

A7: Mirroring part of a picture

Sometimes you only want to mirror part of a picture. For example, Figure 10 shows a temple in Greece that is missing a part of the roof called the pediment. You can use the explorer tool to find the area that you want to mirror to produce the picture on the right. If you do this you will find that you can mirror the rows from 27 to 96 (inclusive) and the columns from 13 to 275 (inclusive). You can change the starting and ending points for the row and column values to mirror just part of the picture.

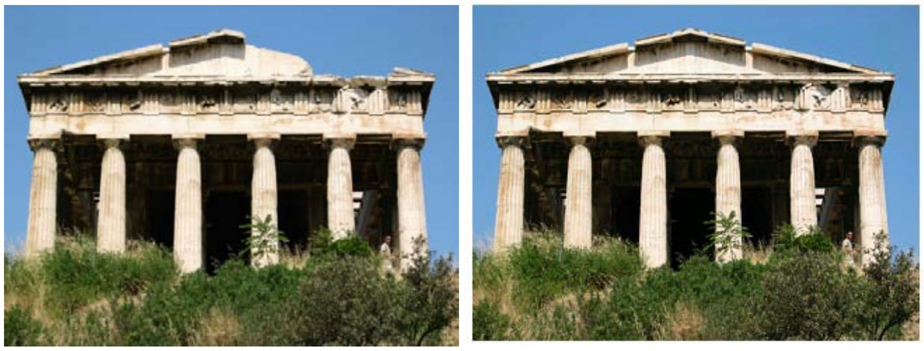

Figure 10: Greek temple before (left) and after (right) mirroring the pediment

To work with just part of a picture, change the starting and ending values for the nested `for` loops as shown in the following `mirrorTemple` method. This method also calculates the distance the current column is from the `mirrorPoint` and then adds that distance to the `mirrorPoint` to get the column to copy to.

```
public void mirrorTemple()
{
  int mirrorPoint = 276;
  Pixel leftPixel = null;
  Pixel rightPixel = null;
  int count = 0;
  Pixel[][] pixels = this.getPixels2D();

  // loop through the rows
  for (int row = 27; row < 97; row++)
  {
    // loop from 13 to just before the mirror point
    for (int col = 13; col < mirrorPoint; col++)
    {

      leftPixel = pixels[row][col];
      rightPixel = pixels[row]
                       [mirrorPoint - col + mirrorPoint];
      rightPixel.setColor(leftPixel.getColor());
    }
    }
  }
```

You can test this with the `testMirrorTemple` method in `PictureTester`.

How many times was `leftPixel = pixels[row][col];` executed? The formula for the number of times a nested loop executes is the number of times the *outer loop* executes multiplied by the number of times the *inner loop* executes. The outer loop is the one looping through the rows, because it is outside the other loop. The inner loop is the one looping through the columns, because it is inside the row loop.

How many times does the outer loop execute? The outer loop starts with `row` equal to 27 and ends when it reaches 97, so the last time through the loop `row` is 96. To calculate the number of times a loop executes, subtract the starting value from the ending value and add one. The outer loop executes 96 – 27 + 1 times, which equals 70 times. The inner loop starts with `col` equal to 13 and ends when it reaches 276, so, the last time through the loop, `col` will be 275. It executes 275 – 13 + 1 times, which equals 263 times. The total is 70 * 263, which equals 18,410.

Questions

1. How many times would the body of this nested `for` loop execute?
```
for (int row = 7; row < 17; row++)
  for (int col = 6; col < 15; col++)
```
2. How many times would the body of this nested `for` loop execute?
```
for (int row = 5; row <= 11; row++)
  for (int col = 3; col <= 18; col++)
```

Exercises

1. Check the calculation of the number of times the body of the nested loop executes by adding an integer `count` variable to the `mirrorTemple` method that starts out at 0 and increments inside the body of the loop. Print the value of `count` after the nested loop ends.
2. Write the method `mirrorArms` to mirror the arms on the snowman (“`snowman.jpg`”) to make a snowman with 4 arms. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.
3. Write the method `mirrorGull` to mirror the seagull (“`seagull.jpg`”) to the right so that there are two seagulls on the beach near each other. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

How image processing is related to new scientific breakthroughs

Many of today's important scientific breakthroughs are being made by large, interdisciplinary collaborations of scientists working in geographically widely distributed locations, producing, collecting, and analyzing vast and complex datasets.

One of the computer scientists who works on a large interdisciplinary scientific team is Dr. Cecilia Aragon. She is an associate professor in the Department of Human Centered Design & Engineering and the eScience Institute at the University of Washington, where she directs the Scientific Collaboration and Creativity Lab. Previously, she was a computer scientist in the Computational Research Division at Lawrence Berkeley National Laboratory for six years, after earning her Ph.D. in Computer Science from UC Berkeley in 2004. She earned her B.S. in mathematics from the California Institute of Technology.

Her current research focuses on human-computer interaction (HCI) and computer-supported cooperative work (CSCW) in scientific collaborations, distributed creativity, information visualization, and the visual understanding of very large data sets. She is interested in how social media and new methods of computer-mediated communication are changing scientific practice. She has developed novel visual interfaces for collaborative exploration of very large scientific data sets, and has authored or co-authored many papers in the areas of computer-supported cooperative work, human-computer interaction, visualization, visual analytics, image processing, machine learning, cyberinfrastructure, and astrophysics.

In 2008, she received the Presidential Early Career Award for Scientists and Engineers (PECASE) for her work in collaborative data-intensive science. Her research has been recognized with four Best Paper awards since 2004, and she was named one of the Top 25 Women of 2009 by Hispanic Business Magazine. She was the architect of the Sunfall data visualization and workflow management system for the Nearby Supernova Factory, which helped advance the study of supernovae in order to reduce the statistical uncertainties on key cosmological parameters that categorize dark energy, one of the grand challenges in physics today.

Cecilia Aragon is also one of the most skilled aerobatic pilots flying today. A two-time member of the U.S. Aerobatic Team, she was a medalist at the 1993 U.S. National Championships and the 1994 World Aerobatic Championships, and was the California State Aerobatic Champion.

Glossary

1. Abstract class — You cannot create an object of an abstract class type. But, you can create an object of a subclass of an abstract class (as long as the subclass is not also an abstract class).
2. Abstract method — An abstract method cannot have a method body in the class where the method is declared to be abstract.
3. Algorithm — A step-by-step description of how to solve a problem.
4. AWT — The Abstract Windowing Toolkit. It is the package that contains the original Graphical User Interface (GUI) classes developed for Java.
5. Binary number — A binary number contains only the digits 0 and 1. Each place is a power of 2 starting with 2^0 on the right. The decimal number 6 would be 110 in binary. That would be $0 * 2^0 + 1 * 2^1 + 1 * 2^2 = 6$.
6. Bit — A **b**inary dig**it**, which means that it has a value of either 0 or 1.
7. Byte — A consecutive group of 8 bits.
8. Column-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first column is stored before all the data for the second column and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the columns and the inner arrays represent the rows.
9. Digital camera — A camera that can take digital pictures.
10. Digital picture — A picture that can be stored on a computer.
11. Inheritance — In Java, a class can specify the parent class from which it inherits instance variables (object fields) and object methods. Even though instance variables may be inherited, if they are declared to be private they cannot be directly accessed using dot notation in the inheriting class. Private methods that are inherited can also not be directly called in an inheriting class.
12. Inner loop — In a nested loop (a loop inside of another loop) the loop that is inside of another loop is considered the inner loop.
13. Interface — A special type of class that can only have public abstract methods in it and/or static constants.
14. Lossy compression — Lossy compression means that the amount of data that is stored is much smaller than the available data, but the part that is not stored is data that humans would not miss.
15. Media computation — A method of teaching programming by having students write programs that manipulate media: pictures, sounds, text, movies. This approach was developed by Dr. Mark Guzdial at Georgia Tech.
16. Megapixel — One million pixels.
17. Nested loop — One loop inside of another loop.
18. Outer loop — In a nested loop (a loop inside of another loop) the loop that is outside of another loop is considered the outer loop.
19. Package — A package in Java is a group of related classes.
20. Pixel — A picture (abbreviated **pix**) **el**ement.
21. RGB model — Represents color as amounts of red, green, and blue light. It sometimes also includes alpha, which is the amount of transparency.

22. Row-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first row is stored before all the data for the second row, and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the rows and the inner arrays represent the columns.
23. Subclass — A class that has inherited from another class.
24. Superclass — A class that another class has inherited from.
25. UML —Unified Modeling Language. It is a general purpose modeling language used in object-oriented software development.

References

Dann, W., Cooper, S., & Ericson, B. (2009) *Exploring Wonderland: Java Programming Using Alice and Media Computation*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson B. (2006) *Introduction to Computing and Programming in Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2009) *Introduction to Computing and Programming in Python: A Multimedia Approach*. (2nd ed.). Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2010) *Problem Solving with Data Structures using Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Quick Reference

```
DigitalPicture Interface
Pixel[][] getPixels2D()         // implemented in SimplePicture
void explore()                  // implemented in SimplePicture
boolean write(String fileName)  // implemented in SimplePicture
```

```
SimplePicture Class (implements Digital Picture)
public SimplePicture()
public SimplePicture(int width, int height)
public SimplePicture(SimplePicture copyPicture)
public SimplePicture(String fileName)
public Pixel[][] getPixels2D()
public void explore()
public boolean write(String fileName)
```

```
Picture Class (extends SimplePicture)
public Picture()
public Picture(int height, int width)
public Picture(Picture copyPicture)
public Picture(String fileName)
public Pixel[][] getPixels2D()        // from SimplePicture
public void explore()                 // from SimplePicture
public boolean write(String fileName) // from SimplePicture
```

```
Pixel Class
public double colorDistance(Color testColor)
public double getAverage()
public int getRed()
public int getGreen()
public int getBlue()
public Color getColor()
public int getRow()
public int getCol()
public void setRed(int value)
public void setGreen(int value)
public void setBlue(int value)
public void setColor(Color newColor)
```

```
java.awt.Color Class
public Color(int r, int g, int b)
public int getRed()
public int getGreen()
public int getBlue()
```