Pictures

Objectives

Learning about pictures and how they are represented in a program

Understand that picture files are made up of pixels and how pixels are represented.

Understand how Java uses evaluation and substitutions to interpret your code.

Understand what a literal is and how to differ from a variable or an expression.

- Media is a term for audio-visual materials used to convey ideas.
- Media computation is the programmatic manipulation of media.
- In this module, we will be focusing on static media, such as images and text, and interactive media, such as interactive stories.

Working with Images

- Digital images allow your applications to be more aesthetically pleasing, or even just to function (especially if you're building an image manipulation application).
- To understand how images are manipulated programmatically, it is important first to understand how we see color, and how color is represented in code.
- Our eyes have special cells in them called "cones" that can sense red, green and blue, and "rods" that can sense intensity in terms of black, white and shades of grey. All of the colors we see are actually just how our brain interprets different combinations of inputs from rods (intensity) and cones (red, green and blue).
- Computers encode colors using a very similar approach:
  - Red, Green, and Blue are each assigned to an intensity on a scale of 0 to 255.
- Observation...
  - This scale is not arbitrary. Recall that a byte is 8 bits long, meaning it can hold 2^8 pieces of information, or 256 different numbers. This means that a color can be stored in only 3 bytes.
- Example
  - You can make colours using RGB yourself.
  - Here is a simple applet that shows you the variations:
    - http://www.javaview.de/vgp/tutor/color/PaColorDemo.html

- Digital images, including the interactive programs displayed on your screen, the images captured by digital cameras, and the images you create in programs like Photoshop, are encoded by breaking them up into many tiny *pixels*, each of which is assigned a color.

- Each image is a grid of very very tiny pixels, so small that to our eyes they look smooth.
- Digital cameras record the color of the light at each pixel and your monitor displays images using pixels.
- In fact, if your screen is set to 1280x800, that means that there are 1280 pixels across your screen, and 800 going down.
- When you draw something with any program, you are actually specifying a series of pixels that should have their color changed to the current color.
- We can do much more than drawing lines when we work with media. For example, using this information, you can change the colors of an image or create a collage.
  - Here are some neat collages that students in media computation classes have created.
- To make media manipulation easier, we are going to use some files:
  - Picture, ColorChooser, FileChooser, Pixel, Color, and so on.
  - These classes work together to make pictures possible. Each plays a different role, and the division of labor should make sense as you get to know them. They are a working example of how object-oriented programming can simplify a programming task (in this case, the task of manipulating images).
- Picture, ColorChooser, FileChooser, and Pixel are not parts of the Java language, but are from a library provided by Georgia Tech (other classes that we use such as java.awt.Color are part of the libraries provided by Sun, to complement the Java language).
- Let's explore some of these classes more. First, let's create a Picture.
- To create your first picture object, you will use the same syntax we used previously: Picture myPic = new Picture().
- In the case of a Picture object, a filename is required as an argument. One option is to "hard-code" the filename as a *literal*.
- For example, you could type Picture mypic = new Picture(“C:/Users/saryta/Pictures/night.jpg"), assuming you had an image file in that location.

Expressions and Evaluation

- Literals can be used anywhere you might use a variable name or some other code that needs to be evaluated.
- For example, the most common way to initialize a variable is by assigning a literal to it, which you would type by hand into your program (e.g., String fileName = C:/Users/saryta/Pictures/night.jpg ").
- Literals can also be passed into methods (System.out.println("Hello World")) or used in expressions (3 + 4 contains two literals, 3 and 4)
- Expression
  - (definition) An expression is a small piece of code that produces a value. For example, 3 + 4 is an expression. Its value is 7. Every expression also has an implicit type (in the case of 3 + 4, the type is int).
- Expressions can include variables, methods, and so on.
  - For example, suppose that I've created a new Picture object and associated it with the variable myPic. The method call myPic.getWidth() is an expression. Its value is an int. An expression may include other expressions (as in myPic.getWidth() / 2, which contains the expressions myPic.getWidth() and 2). 2 is an expression because an expression may be a simple literal (3 is an expression, as is "hello"). myPic is an expression, of type Picture.
- When java sees an expression, it *evaluates* it and uses the result to complete the calculation. For example, consider the code 3+4. This can be evaluated as 7.
- Evaluation

  - (definition) When a program is run, pieces of it are *evaluated* and then the resulting values are substituted in. Evaluation is the process of interpreting and acting on a piece of code (such as an expression).

- Now suppose that you wrote int result = 3+4. Java will *evaluate* 3+4 to 7 and store the resulting value in result.
- Any time you assign something to a variable, pass something into a method, or otherwise manipulate information in a program, you are asking Java to evaluate expressions.
- For example result was replaced with its value (7) in the code above. Any time you use a variable, its value will eventually be substituted in.
- Similarly, an expression such as 3/4 will be replaced with its value (0).
- The same is true of a method invocation. FileChooser.pickAFile() is replaced with the String value it returns (the location of the file) in the code above (new Picture(FileChooser.pickAFile()).
- Only literals are unchanged by evaluation. A literal such as 3 or "Hello" will be evaluated as itself.
- An expression can be made up of literals or expressions.
- If an expression contains multiple sub expressions, Java will evaluate each of them and then evaluate the result.
- In fact, for Java to run your code, it must eventually substitute a literal or a reference to an object for each variable, method invocation, or expression that you use.
- This is done by evaluating code progressively, typically working from left to right and from innermost to outermost material.
- For example (3/4) is evaluated after size in size = size*(3/4). When possible, you want to ensure that your code will be correct regardless of order of evaluation. In the case of size, for example, a more correct statement of intent might be size = size*(3.0/4.0).
- Example

  - x = (3+4)+((5+6)/7);
  - This piece of code is an expression that contains several other expressions.
  - Java will evaluate this one expression at a time, from left to right.

- First, it will evaluate (3+4), which is 7.

  - Next it will evaluate (5+6), which is 11.
  - Replacing those expressions with their values, we have 7 + (11/7).
  - Since 7 evaluates to itself, Java will next evaluate the expression (11/7).
  - This will be 1, since we ignore the fractional part in integer arithmetic.
  - Finally, Java evaluates 7 + 1, which returns the value 8, an integer.
  - Since x = 8, Java stores 8 in x.

Did I get this?

- Suppose you are trying to solve the following problem: Two brothers went shopping and came home with 12 books. One had bought 3 books for himself. How many did the other brother buy?

- This problem can be encoded by assigning a variable to each brother, representing the number of books bought. In one case, we know the number: int joshBought = 3; In the other case we do not: int kenBought; A simple calculation can tell us how many books Ken bought: kenBought = 12 - joshBought . Now we can print the answer using System.out.println("Ken bought: " + kenBought + " books.");
- How many literals are used in this code? ____________
- What will 12 – joshBought be evaluated as? __________
- How many expressions are used in 12 – joshBought? __________

Did I get this?

- How many literals are used in this code?
  - A literal is anything that requires no interpretation. A literal can be used in many settings, including an expression or as a method parameter. A literal can be text, numeric, or boolean. Here, 3, 12, "Ken bought ", and "books" are all literals.
  - A literal can be the value assigned to a variable, but it can also be used in an expression or passed into a method.
- What will 12 – joshBought be evaluated as?
  - First, figure out what the value of joshBought is. Next, do the math.
- How many expressions are used in 12 – joshBought?
  - any piece of code that Java can evaluate is an expression. This includes even small pieces of code such as a single literal or variable name. Here, 12, joshBought, and 12-joshBought are literals.

- Evaluation order is important to consider when you are casting.
- When casting, it is a good idea to always use parentheses to specify which expression should have its type changed.
- Java will evaluate the expression, and then change the type of the expression.

Learn by doing

- What will the value of dollars * 3 be, if dollars = 1.50?
- Consider the expression (double)(3/4). Why do you think this leads to a different result from ¾?

Our answer

- What will the value of dollars * 3 be, if dollars = 1.50?
  - The value of this expression is a double, 4.50 (or 4.5). dollars are evaluated as a double, The value of this expression is a double, 4.50 (or 4.5). dollars are evaluated as a double, the math.
- Consider the expression (double)(3/4). Why do you think this leads to a different result from ¾?
  - It's important to understand that programs do only, and exactly, what you tell them. They don't know what you meant. One thing that you have to specify, either implicitly or explicitly, is what order things happen in. (double)(3/4) says divide three by four and then convert it to a double.

- Let's explore evaluation order further.

```
double size = 3.0 + 4.0;
size = size*(3/4);
```

- If you were to put this into a project in Netbeans you should see 0.0 as the result. This may seem like a surprising answer, but it illustrates the importance of evaluation order. At heart, a computer only knows how to do one thing at a time. Thus, when interpreting a program, it must choose an order in which to do things.
- In this case, Java chooses the order as follows:
  - Substitutes the current value of size (7.0) for size
  - Begins to evaluate the expression (3/4)
  - Evaluates 3 (as 3) and decides that it is an int
  - Evaluates 4 (as 4) and decides that it is an int
  - Evaluates 3/4 (the result is 0 for reasons discussed earlier)
  - Substitutes 0 for (3/4)
  - Decides that size is a double and casts 0 to double
  - Multiplies 7.0 * 0.0
  - Stores 0.0 in size
- By looking at concatenation, we can see a nice example of the impact of evaluation order. Remember that when Java adds things together, it sometimes changes their type.
- If you type "The result is " + 2 + 3, the result will be the string "The result is 23". You might have expected to get "The result is 5" instead because of 2+3. However, since the first object in this concatenation is a string ("The result is"), and it is evaluated first, 2 and 3 are interpreted as string objects as well.
- Observation...
  - As shown in the example above, you can concatenate strings and other primitives together to create another string object. The + operator is used for concatenation. For instance, you can have "This is " + "a new string." to concatenate two strings ("This is " and "a new string.") into a new string, "This is a new string." Moreover, you can also concatenate strings and primitive types to create a new string.
- Now, if you type 2 + 3 + " is the result". The result will be "5 is the result", because 2 and 3 are evaluated first and therefore interpreted as type int. When "is the result" part is encountered, then the int part is treated like a String object. Hence, it is equivalent to concatenating two strings: "5" + "is the result", where 5 comes from the addition of two ints (2 and 3).
- Kind of neat don’t you think?

Loading Images

- Let's return to the problem of loading an image file.
- Instead of passing a literal String into Picture(), we can pass an expression that, when evaluated, will return a String.
- The FileChooser's pickAFile() method will bring up a graphical user interface (GUI) allowing you to select a file, and then return the location/name of that file as a String.
- You could then store the result of pickAFile() in a variable for future use, and pass that variable in to new Picture().

- In that case, the variable will be evaluated and replaced with it's String value (the location of whatever file you selected).
- I have uploaded the Netbeans project onto D2L. Download and unzip the project and save it to your drive. Under File, you will find Open Project. Once the project is open, look for the folder mediaoneex and in there is the MediaOneEx1.java file and others each with their own main method.
- Run the files and find yourself a picture to load.
- Note: If you run the program and do not load an image you will get an error and your Netbeans might not shut down the application fully.
- There are three different ways to load a picture -- you can pass in a literal (String) specifying the name of the picture, pass in a variable which holds the name of a picture, or pass in the result of a method call to FileChooser.pickAFile().
- An example of each method follow.
- NOTE: In Netbeans, right click on the file and select run to execute the files.

Example 1: passing in a variable

```
String filename = “/Users/saryta/Pictures/night.jpg";
//    OR String filename = FileChooser.pickAFile();
Picture mypic = new Picture(filename);
mypic.show();
```

This is the code in MediaOnezex1.java

Example 2: passing in a literal

```
// replace "/Users/saryta/Pictures/night.jpg" with the path for YOUR file!
Picture mypic = new Picture("/Users/saryta/Pictures/night.jpg");
mypic.show();
```

This is the code in MediaOnezex2.java

Example 3: using FileChooser

```
  String fileName = FileChooser.pickAFile();
Picture myPic = new Picture(fileName);
//OR you can do this in one line
//Picture mypic = new Picture(FileChooser.pickAFile());
myPic.show();
```

This is the code in MediaOnezex3.java

- How do you decide which approach to use?
- There are advantages and disadvantages to each.

- If you use a literal, you save yourself the effort of using a GUI to pick an image file each time you want to test your code.
- If you use the FileChooser, you don't have to figure out the correct path to the image you wish to load, you just use the GUI to select it.
- Once you decide whether to use a literal or a method call, you also need to decide whether to store that information in a variable or simply pass it in to Picture().
- If you store it in a variable, you can access that information (the location of your file) for other purposes and use it over and over again. Otherwise, you will use it only once.

Summary

- This module is beginning to divide its learning goals among things that represent media literacy and understanding of how to programmatically manipulate media, *versus* your understanding of more general programming concepts.
- In the media arena, at this point, you should be comfortable with creating a Picture object and using it to load an image file from your desktop and show it on the screen.
- In addition, based on this and prior modules, you should be able to explain what a *pixel* is and how it is used to encode images.
- With regard to programming concepts, this module introduced the concept of evaluation and substitution, an activity that is progressively done by Java as it runs your code. It is important to understand how an expression, variable, or method invocation will be interpreted during evaluation. In the end, each will be replaced with its value (whether a literal or a reference to an object).