FUNHOUSE MIRRORS

Lab 3

**What You’ll Learn in this Lab:**

* More practice with number bases (binary and hexadecimal)
* How to set individual pixels in an image
* How to manipulate a portion of an image (vs. the whole image)
* How to mirror, grow, and shrink an image
* How to layer images on one another
* How to call functions within functions (again)

**Getting Ready:**

* Read Sections 6.1 and 6.2 in the Python textbook and Chapters 2 and 3 of CSI
* Bring the following to lab:
  + Lab 2 Worksheet, ready to turn in at beginning of lab!!
  + CSIS-110 Lab Manual
  + Both textbooks
  + Something to write with



**Details**

*This section of each lab will involve practicing and experimenting with various aspects of our computer systems – editing, printing, mail, techniques for making your life easier, and saving your projects.*

Thisactivity will give you some useful (and fun) practice with binary numbers and powers of 2, which will make many activities easier for you later in the semester. **Open** the following link by clicking it along with the <control> key. This is a speed game for converting from base 10 (decimal) to/from base 2 (binary). Read the instructions, and play the game! When you reach a score of 6000, you’re done! Don’t stop the game – just pause it.

<http://forums.cisco.com/CertCom/game/binary_game_page.htm?site=celc>



***Show your winning score to your instructor for signature.***

We have mostly been working with colors that are in existing images. Sometimes it is useful to choose or create colors “from scratch”, either when creating an image or modifying pixels in some way. The JES function “pickAColor()” allows you to see all the possible 24-bit colors and their RGB values. Start JES and type the following in the command area:

**myColor = pickAColor()**

In the color choose window, under the “RGB” tab, choose a color that you would consider “purple”, and click “OK”. In the command area, type the command **print myColor**.



***What are the decimal R, G, and B values for myColor? What are those values in 6-digit hexadecimal form, using 2 hex digits for each color?***

**Enhancing your Computer Science Knowledge**

*This section of each lab will involve problems and software that will augment your understanding of concepts that are fundamental to Computer Science.*

You will now write a function in Python that works as follows: given any picture, any X and Y co-ordinates, and any color, the function will change the pixel at location (X, Y) to whatever color you give it. Here are some examples of how you should be able to CALL (use) your function from the command area, assuming you already have variables called “myPic”, “yourPic”, and “aColor” (don’t type these now, they’re just examples to help you understand the function):

**setPixel (myPic, 100, 200, aColor)**

**setPixel (yourPic, 50, 300, aColor)**

Copy the file “lab3SetPixel.py” from the csis110/common/lab3 folder into your csis110 folder. Then, in JES, Open that file. It contains the shell of a function that will do what we want. Add your names to the top of the file. Your job is to complete the function and test it. The easiest way to test it is to use an all-white picture as the image, and set one of its pixels to “myColor” so you can see the results easily: First, type the command **myPic = makePicture (pickAFile())** and choose the file “640x480.jpg” from the Media Sources folder. Once you’ve completed the “setPixel” function, save the file, load it into Python, then test it by typing the following command:

**setPixel (myPic, 150, 250, myColor)**

If it’s hard to see the individual pixel, then use the “Zoom” feature of the Picture Explorer to get a larger view of the pixel. If it doesn’t work as it should, then fix the error(s), save, load, and test again!

***Print your copy of “lab3SetPixel.py” and attach the printout to the back of the worksheet.***

**Extending and Expanding:**

*This section of each lab will involve using Python to answer questions and solve problems.*

Copy the file “lab3Mirrors.py” from the csis110/common/lab3 folder into your csis110 folder. Then, in JES, Open that file and load the program. These are essentially Programs 20 and 21 from your Python text. Add your names to the top of the file. Using your (by now) favorite Python command, **myPic = makePicture (pickAFile()),** choose any picture you like from the Media Sources, or download something from the Internet. Ideally, you should choose a picture that is not too uniform: there should be easy-to-see differences between the four main quadrants of the picture, for example.

**PYTHON POINTER #1:**

As you learned in previous labs, it can take a while to process all pixels of a large picture, so don’t pick too large a picture unless you want to spend a lot of time waiting for results.

Type the following sequence of commands:

**mirrorVertical (myPic)**

**mirrorHorizontal (myPic)**



***Which quadrant of the picture is repeated 4 times?***

Now use the following sequence of commands to reset the picture and mirror it differently:

**myPic = makePicture (pickAFile())**

**mirrorHorizontal (myPic)**

**mirrorVertical (myPic)**



***Now which quadrant of the picture is repeated 4 times?***

Now alter one or both of the functions you were given so that the bottom right quadrant of the image is repeated 4 times when you mirror the picture both horizontally and vertically. As always, repeat the following steps until your program works as it should: fix, save, load, and test.



***Call your instructor over to see the results of your new mirroring code, and make sure he/she signs on the line on the worksheet.***



***Print your copy of “lab3Mirrors.py” and attach the printout to the back of the worksheet.***

**Calling Functions with Finch:**

Today your finch wants to learn how to sing, dance, and act at once. This can be done by putting the actions needed in a function. However, the finch wants to learn to dance, sing, and act in that order. To do this you must create a program, called “stage”, where you create three functions, dancing(), singing(), and acting(), each with its own set of actions and then call them from a single function, performance(). The built-in functions saySomething (for singing); setLED and buzz (for acting), and setWheelVelocities (for dancing) will help with this task. Open the file “lab3Stage.py” from the common folder (or Blackboard) and save it to your own csis110 folder. Complete the code and have fun with Will the Finch! Don’t forget that you may need to use the “sleep” function to help Will perform. Help on all these Finch functions is available in Chapter 13 of the “Help” menu in JES.



**Call your professor over so they can see the Finch’s performance, and don’t forget to have them sign your worksheet.**

****

**Print a copy of “stage.py” and attach it to the back of your worksheet.**

**Scaling Images:**

Now we (that means you) will play (i.e., work) with scaling pictures up and down. Copy the file “lab3Scaling.py” from the csis110/common/lab3 folder into your csis110 folder. Then, in JES, Open that file. The “grow” and “shrink” functions are essentially Programs 82 and 83 from your Python text, generalized to work with any picture. Add your names to the top of the file. Again using your favorite Python command, **myPic = makePicture (pickAFile()),** choose any picture you like from the Media Sources, or download something from the Internet. Ideally, you should choose a picture of a person or animal, for the most entertaining effects later on.

You should test the “grow” and “shrink” functions just to see what they do to your picture:

**small = shrink (myPic)**

**repaint (small) # has “myPic” changed?**

**big = grow (myPic)**

**repaint (big) # has “myPic” changed?**

**PYTHON POINTER #2:**

Recall from the text that Python treats integers and real numbers differently! For example, 3/8 is 0, but 3.0/8.0 is 0.375.

Now we have one more function to write for this lab: In the same file with “shrink” and “grow”, write a function called “layer” that will take any picture as input, and create a 3-layered image with the scaled-up picture on the bottom, the original picture centered on top of that, and the scaled-down picture centered on top of that (see example on next page).

HINT 1: You can do this the easy way (1) or the hard way (2). (1) The easy way is to use, in your “layer” function, the built-in function “copyInto”. There will be no loops in your “layer” function. (2) The hard way is to do anything else.

HINT 2: Before testing your code, remove the lines “show (target)” from the “grow” and “shrink” functions. You no longer need to see those results, just your final result.

Here is an example result of “layer (myPic)”, assuming “myPic” refers to the picture “church.jpg” from the Media Sources folder (you should use a different picture, perhaps the same one you used earlier in today’s lab, or whatever):



***Call your instructor over to see the results of your “layer” function, and make sure he/she signs on the line on the worksheet.***



***Print your copy of “lab3Scaling.py” and attach the printout to the back of the worksheet.***

Before you log out, you should make sure you have shared your work with the lab partner(s) whose account you are not using. To do this, you can **email** all of the relevant files to your partner. (There are other ways to do it too, of course, so do it however you like.)

**Reflection:**

*The discussion questions in this section of each lab are meant to make you think critically and creatively about some of the things you did earlier in the lab. Your answers to these questions must not be written on the lab worksheet, but on separate sheets of paper attached to the end of your lab worksheet. Your answers must be typed (not handwritten), and you will be graded on all aspects of your answer (correctness, use of proper terminology, readability, use of complete sentences only, etc.). In general you are expected to write at least one or two paragraphs in answer to each question.*

Image files and other data files have often been shared over the internet using services such as Napster, BitTorrent, Kazaa, Gnutella, LimeWire, etc. Suppose you make your living creating things that can be represented digitally (pictures like those you created today, music, movies, computer programs, books, etc.). How would you feel if someone who bought something you spent a year creating were to make it available on a service such as Napster or LimeWire, for free? Discuss your reactions to this scenario, including your thoughts on whether you think this is a good or bad thing, what you might say to the person who made your work available for free, what you think should happen (if anything), and other aspects of the situation.

As we know, color pixels are represented by a 3-byte (24-bit) number. This doesn’t allow us to capture all possible colors, but for most purposes it’s “good enough”. Another example of when something approximate was considered “good enough” occurred with the infamous Y2K bug. It was a good example of how short-sighted assumptions about information representation can have costly and wide reaching effects. Software and hardware afflicted with the Y2K bug was expected to run incorrectly or even crash after December 31, 1999. So in anticipation of this problem, companies invested millions of dollars hiring computer scientists/engineers to fix it. Because the extent of the problem was not completely known, prior to the new year many people bought bottled water, firewood, and kept cash reserves at home in case things such as the water, power, and banking systems broke down.

Do a search on the internet to learn more about the Y2K bug. In your own words describe what you learn. Some questions you want to try to answer include: What does Y2K stand for? What is the Y2K bug and what caused it? What did people and companies do to prepare for it? What was the extent of the problem on Jan 1, 2000 and the weeks and months following that?