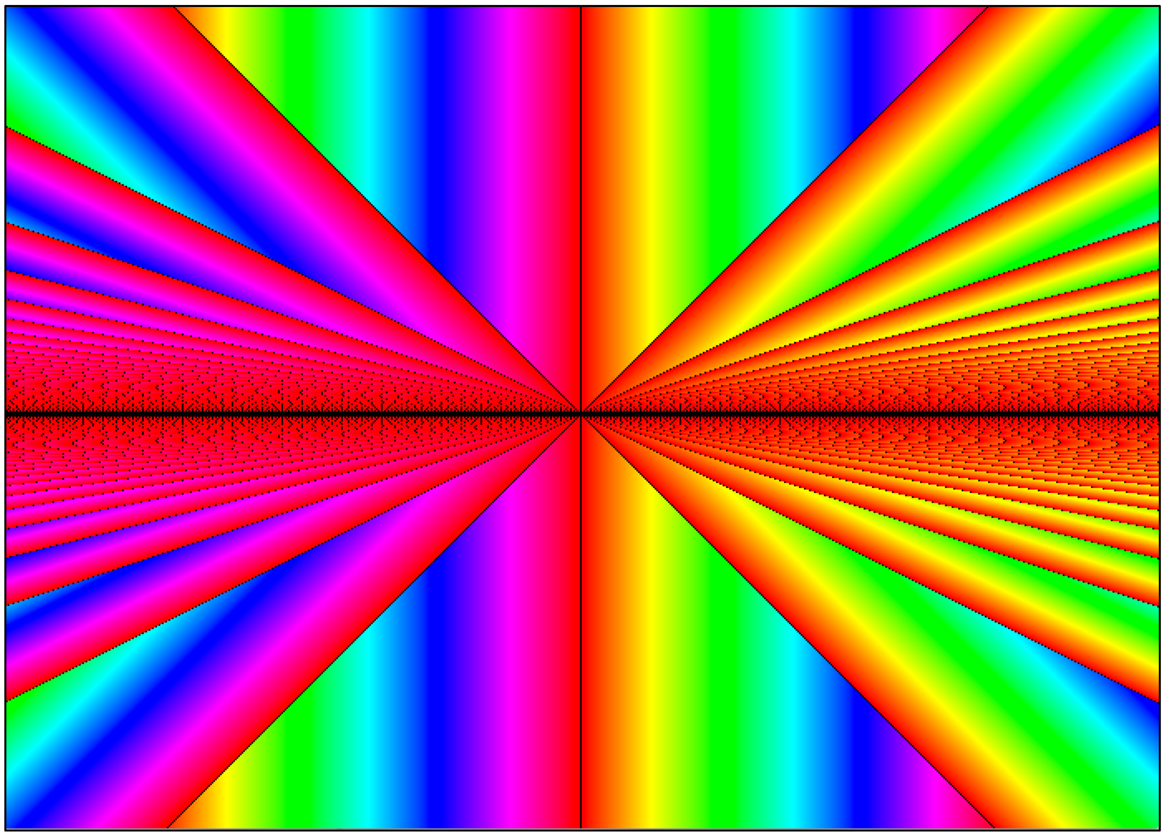
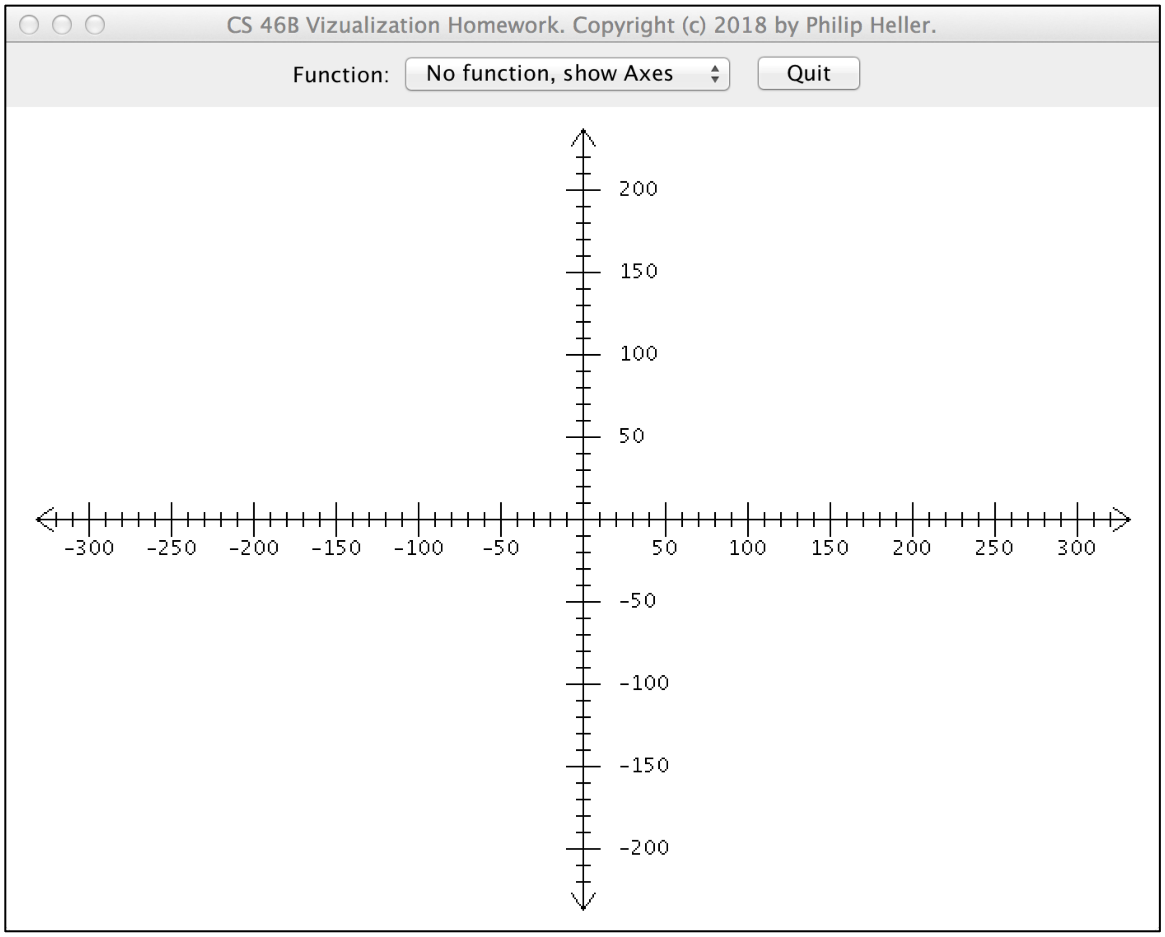
CS 46B
Fall 2019
Homework 6: Visualizing Functions of (x,y)
Due 11:55 PM on Thursday October 17



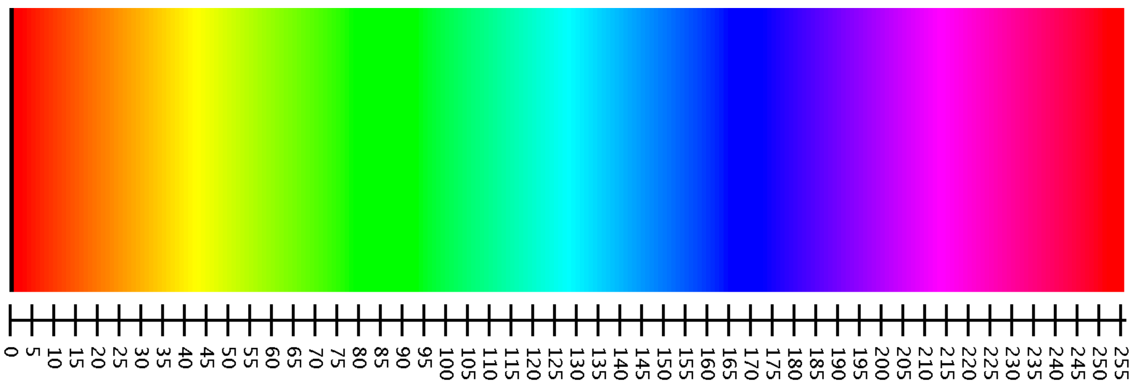
Visualization is the art of depicting large sets of data as pictures, so that patterns can be recognized. In this assignment you'll use an existing visualization application to draw pictures of mathematical functions of 2 variables.

The visualization app initially looks like this:



Imagine the main app area as representing the (x,y) plane, with x ranging from -350 to +350, and y ranging from -200 to +200. Each dot on the screen (pixel) represents an (x,y) point in this 2-dimensional space. The app computes a user-specified function of each (x,y) point to generate a number. Then the app converts each of those numbers to a color, and "paints" the corresponding pixel.

To convert numbers (generated by functions) to colors, the app uses the following algorithm:
0 ➔ Black
1 ➔ Red
2 – 255 ➔ Rainbow blending: low values are red, blending to orange, to yellow, green, blue, purple, magenta, then back to red (See figure below).
>255 ➔ Cycle through the rainbow again and again and again.



The app manages all of this for you. So if you're confused by any of this, don't worry. Your job is to provide classes that compute various functions of (x,y). The visualization part is just to make it fun.

Create a new Eclipse Java project. Under "src" create a package called "func". Import your starter code files into that package.

Your starter files are listed below. Some of them will have compiler errors until you complete your assigned code.

- VizFrame.java – This is the visualization app. After you write your code, you can run this to look at pictures of your functions. You won't change this class except for uncommenting a few lines when instructed below. Graphical Java is beyond the level of 46B, so don't expect to understand all of this file. But read it anyway … it's an example of how to lay out a slightly large source file, and it shows the sort of comment you're expected to write in your own code. Mostly you won't be graded on your comments. But if you hope to succeed as a programmer – and *especially* if you hope to pass coding interviews – you have to write good comments and lay out your code neatly and consistently. Run this class as an app (you might have to tell Eclipse to ignore compiler errors). By making selections in the "Function" drop-down menu, you'll be able to visualize the axes or the addition function. More items will be added to the menu as you work through the assignment.
- VizPanel.java – Used by VizFrame.java. Don't change this code, but again get a feel for the comments and the layout.
- DoubleFunctionOfTwoInts.java – An interface that you will implement several times. The interface has 2 methods: `double fOfXY(int x, int y)`, which computes a function on its inputs, and `String getName()`, which returns the name of the function. (In your implementing classes, have `getName()` return any reasonable name.) Don't change the DoubleFunctionOfTwoInts.java code.
- AdditionFunction.java – A very simple implementation of the DoubleFunctionOfTwoInts interface. It might be a helpful guide when you write your own implementations. Don't change this code.

- WeirdComplexFunction.java – Don't change this code. It's another implementation of the interface, and it's just for fun. You can see what it does later once your own code works.
- FunctionGrader.java – This is the grader bot. It will tell you how you're doing even if some of your work is incomplete or doesn't compile. But all homework assignments must have *no compilation errors*. No matter what grade the bot gives you, if you submit work that doesn't compile then you get zero points.
- Complex.java – See below.

Your assignment is to write the following Java classes. They should all go in the `func` package. Write them in the following order. After you complete each one, uncomment the corresponding line in VizFrame.java, in the block at lines 26-31. This will make the function available in the drop-down menu in the app, and you'll be able to see a picture of the function.

- SubtractionFunction.java – An implementation of the interface. Its `fOfXY` method should return x minus y. Simple!
- ModFunction.java – Another implementation of the interface. Its `fOfXY` method should return x modulo y. Caution: anything % 0 is illegal, so if y is 0, change it to 1 before computing x%y. Before you visualize this function, try to guess what it looks like.
- HypotFunction.java – Yet another implementation of the interface. Its `fOfXY` method should return the hypotenuse of a right triangle whose legs have length x and y. Check that API page for the java.lang.Math class for an easy way to do this. Again, try to guess what the visualization will look like.

The next 2 interface implementations use complex numbers. Recall that a complex number has the form a+b*i*, where *i* is the square root of -1. Edit the starter file Complex.java, adding code where instructed by the comments. WARNING: it's easy to download this code from the web. Remember that you're not here to submit someone else's code; you're here to learn how to be great at writing your own code. Also, submitting someone else's code as if it were your own is plagiarism. That's a serious form of cheating, and the consequences are also serious. So instead, enjoy the process of writing code that does something cool.

After you complete Complex.java, write 2 more implementations of the DoubleFunctionOfTwoInts interface. After you write each class, uncomment the corresponding line at the beginning of VizFrame:

- ComplexNormFunction – The `fOfXY` method should treat its x and y inputs as the components of a complex number x+b*i*, and should return the norm of this complex number. Does this visualization look familiar?

- ComplexSquaredNormFunction - The `fOfXY` method should treat its x and y inputs as the components of a complex number x+b*i*, and return the norm of (x+b*i*)*(x+b*i*).

- Finally, uncomment line 31 in VizFrame.java. This will add WeirdComplexFunction to the drop-down menu. Don't worry if this takes a few minutes to compute. You won't lose points if this part doesn't work. But it's worth waiting for! The area of the inner black shape is finite but its boundary is infinitely long and its dimension is a fraction between 2 and 3!

To submit, export your code exactly as instructed in HW1/Lab1. Check your jar file to make sure its contents are correct. Upload when you are confident. Remember the rules:

1) You may only submit once.
2) You must submit by the due date and time.
3) Your submission must contain your sources. If it doesn't, you get zero points.
4) Your submission must compile without errors. If there are compiler errors, you get zero points.

Extra credit:

For this part of the assignment (and *only* this part), it's ok to work with other people.

Implement another function. Get creative. To visualize your function, edit VizFrame.java. After line 31 ("`new WeirdComplexFunction(),`"), add the following:

```
        new MyCreativeFunction(),
```

(or whatever you call your class). Then run VizFrame.java. Your function's name will appear as the last item in the pull-down menu.

Suggestion: get playful with complex numbers. Even simple functions can be surprising and delightful. (ComplexSquaredNorm totally surprised me … I had *no idea* it would look like that!)

To get the extra credit, take a screenshot of the graph of your function and upload it to assignment "HW5XC" on Canvas. You get 10 extra credit points for doing this, whether or not you think your image looks cool. If you work in a group, everyone in the group should upload under their own name.

For inspiration, check out the images in 46B_S18_HW4_Gallery.pptx in your assignment zip. These are a few of the submissions from the last few years.