

HCI-631 Project 1: Fitts' Law Tester

Due: Wednesday, 13 February at 10:30am

Goal

This project will involve creating a simple program that will collect data of user mouse movement and clicks and compute the coefficients for Fitts' Law.

Included Files

AbstractFittsLawTester.java
RoundButton.java

Project Overview

Fitts' Law predicts movement-time as a function of distance (to the center of the target) and required accuracy (the size of the target).

$$T = A * \log_2(D/S + 0.5) + B$$

Where T is time, D is the distance to the center of the target and S is the size (diameter) of the target.

In this assignment you will collect data in order to calculate the coefficients **A** and **B**. To complete this project you need to provide a class named `FittsLawTester` which will belong to a package called `sau_i_pr1`. This class should be a subclass of the `AbstractFittsLawTester` class provided to you. `AbstractFittsLawTester` has two abstract methods that you must implement in your subclass:

```
public abstract void createGUI();  
public abstract void mousePressed(MouseEvent e);
```

In addition to the methods from the `AbstractFittsLawTester` class, your class must have a constructor with the following signature:

```
public FittsLawTester(int trials)
```

This constructor takes a single argument – the number of trials (target hits) to run, and should call the constructor of its superclass.

You should implement a main method that will accept 0 or 1 arguments. The only argument you should accept is an integer with the number of trials to run the test. If no arguments or an invalid first argument is passed you should use a default value of 6 trials. If more than 1 argument is passed you should use only the 1st argument and ignore the rest. For example:

```
java sau_i_pr1.FittsLawTester 15 hello
```

should start the program and set the number of trials to 15.

Note that the initial click to start (see Figure 1a) does **not** count as a trial.

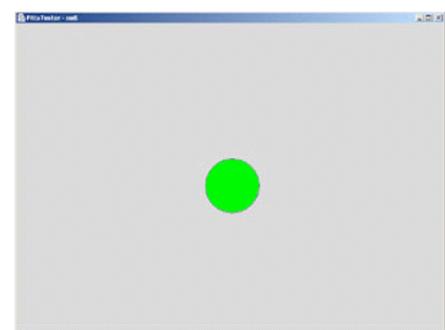


Figure 1a. Initial Screen

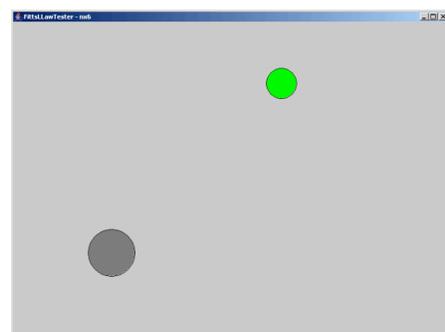


Figure 1b. In a trial

The `createGUI()` method should create a frame and show one green round button (using the provided `RoundButton` class). The size of the frame should be set to the `MAX_WIDTH` and `MAX_HEIGHT` constants from `AbstractFittsLawTester`. The frame should be made visible by calling its `setVisible()` method. The target should be positioned at the center of the content area of the frame and be the maximum target size allowed (note that the constant defines the radius, not the diameter). In order to find the center of the content pane you will need to use the `getBounds()` method. **Note** that the bounds are only updated after a panel is made visible. Assume that the user will not resize the frame.

When the user presses the green button for the first time, it should seem to disappear and two buttons should appear, placed at two random positions on the screen and with two random sizes within given limits. (Note that in order to position components at an absolute location you must set the layout manager of their container to `null` and call their `setBounds()` method.) One of the buttons should be green and accept user presses. The other button should be gray and only serves as a distracter. Each trial begins with the apparent appearance of the two new buttons and ends when the green button is pressed, at which point the buttons should be moved and resized to begin the next trial. This process continues until the desired number of trials is completed.

Note that the first trial starts after you click on the green button for the first time (“Click Here”). This is also true when starting another set of trials.

`AbstractFittsLawTester` is defined as a `MouseListener` so you must implement the `mousePressed()` method. This method is automatically called whenever the user clicks on the target. In response to the mouse click, it should measure the time it took to reach the target (in seconds), generate a new size and position for the next target and store all the data in the `timeData`, `distanceData`, `sizeData` members of `AbstractFittsLawTester`. The target radius should be randomly generated in the range of `MIN_RADIUS` to `MAX_RADIUS` (inclusive). The new target and distracter must be positioned **completely inside** the content area of the frame and **must not overlap**. You will also need to keep count of the number of targets hit. The first target hit should start the timer (that is, the first time captured is between the first and second target). You will need to use the constants provided in `AbstractFittsLawTester` to randomly generate new sizes and positions.

After the last trial, you should call the `calculateCoefficients()` method and print its result to the standard I/O. You should then reset your counter, reposition the target at the center, and have the user click on the target to start a new set of trials.

Remember, you may implement as many other helper methods and classes as you like. Make sure that they take advantage of object-oriented programming and are well documented. **This project will be graded for good programming style.**

Resources

This project should be done in Java 2 SDK version 1.5, using the Swing user interface toolkit which can be downloaded from <http://java.sun.com/j2se/1.5.0/download.html>. You may also do the project with earlier versions of the Java 2 development environment. You can also find documentation for the Java toolkit and a Java tutorial on the java website at <http://java.sun.com>. See the course syllabus for more information on learning about Java and for Java development environments.

The source code for the `AbstractFittsLawTester` and `RoundButton` classes can be downloaded from the Blackboard site (under Assignments/Project 1). In addition, a sample executable can be downloaded from that location.

Turning Your Program In

The program is due Wednesday, 13 February at 10:30am. You should turn in your assignment by email to the instructor using a message whose subject contains the string “**631 project1 turnin for**” and then your name. What you turn in should take the form of **a single** "zip" file (as an attachment to your email message) which contains the source code (.java file) for your `FittsLawTester` class and any needed supporting classes. Do not send multiple attached files (i.e., one attachment for each source file). Again, be certain to include the string “631 project1 turnin for” and then your name, in the subject of your message.

Grading

Your program will be compiled and run on my machine against test cases unknown to you. If your program performs properly, is well structured, and is copiously documented you will receive 100 points.

The tentative detailed grading breakdown for this assignment is as follows:

Basics:

Turned in and compiles	35 points
No crashes	15 points
Good commenting	10 points
Window close works	5 points
Correct distance computed	10 points
Correct size computed	5 points
Various test cases	5-10 points each

Test cases will be designed to test base functionality, command-line parameters, as well as boundary conditions. Comments in your code should document each class, method, variable, and parameter, as well as provide comments needed to understand the logic within the body of the code. Please use JavaDoc style comments where appropriate. Format your code to be readable (e.g., use indentation) and to make it easy to find methods and other declarations (e.g., via white space and/or explicit separators).

Note 1: You will be graded on the quality of your code and commenting. This will include factors such as modularity, sensible method and variable names, and overall clarity. McConnell’s book *Code Complete* is an excellent resource here. Also useful is <http://particletree.com/features/successful-strategies-for-commenting-code>

Note 2: You may not change any of the classes that you are provided with. Before testing your code, the original `AbstractFittsLawTester` and `RoundButton` will be copied over to the directory with your files, so any modifications you made may cause your code not to work and for points to be deducted.