Designing Interactive Systems II - Summer Semester 2010

Assignment 2: Windows (not the Microsoft kind)

Due: Monday, May 10, 2010 @10:00 AM

This assignment is to be completed in *groups of two*.

This is the first in a multi-week assignment where you will build your own window system + toolkit in Java.

Your first task is to create the "desktop" for your window system. Write a `WindowSystem` class that puts up an empty desktop when created. You can subclass from the `GraphicsEventSystem` class, passing in the size of the desktop, which will create and draw the empty rectangle for you. The `GraphicsEventSystem` constructor accepts two `int` arguments for the width and height of the desktop.

- `GraphicsEventSystem(int i, int j);`

Extend your `WindowSystem` class to handle drawing in a resolution independent way, using an "abstract coordinate system" with values between 0.0 and 1.0. To draw a line in the abstract coordinate system, implement a method

- `void drawLine(float StartX, float StartY, float EndX, float EndY)`

that accepts float values ranging from 0.0 to 1.0. You may find the following methods in `GraphicsEventSystem` useful:

- `void handlePaint()`
- `void drawLine(int inStartX, int inStartY, int inEndX, int inEndY)`
- `void setColor(Color inColor)`

You will need to override the `handlePaint()` in your `WindowSystem` class, and do all of your drawing in there. The `handlePaint()` method is inherited from `GraphicsEventSystem`, and currently does nothing.

Then, create a `SimpleWindow` class (the name "Window" is already taken by Java). Your `SimpleWindow` class doesn't need to have much functionality yet, *but* you will need to augment your `WindowSystem` class so that it has the ability to keep track of a collection of `SimpleWindow` objects.

Finally, create a test program, `MyApp`, that uses your `WindowSystem` class. Your test program should display a single line from (0.2, 0.3) to (0.8, 0.7) on your "desktop".

Hints and Tips:

- Implement a private method that converts between the "desktop" and the "abstract coordinate system". Be sure that you preserve the aspect ratio of your desktop.
- You should also be concerned with how to organize your basic data structures – drawing and event handling will take place in an upcoming assignment.
- Do not use any Java Swing API's in any of your code. While all of the features that you will be implementing in your window system are already provided by Java Swing, the purpose of this assignment is for you to learn how window systems like Java Swing are implemented. Your code should depend only on `GraphicsEventSystem`. You may use any basic data structures that Java provides, such as Points, Arrays, etc, however.
- Do not decompile `GraphicsEventSystem`. We're not giving you the source code for this class precisely because your solution should not depend on how it is implemented.

Testing Your Understanding

Answer the following questions:

1. How does your `WindowSystem` keep track of `SimpleWindows`? If you used a data structure, specify which one (e.g., array, linked list, hash table).
2. Justify your specific design and/or choice of data structures, in particular how it would affect the following:
   - adding/removing windows
   - drawing windows in front-to-back order
   - finding a specific window given an arbitrary (x, y) (desktop) coordinate
   - overall code complexity

Keep in mind that there is no single correct answer here. What we're more interested in seeing is whether or not you have gone through the thought process, not whether you have found the "perfect" solution.

Extra Credit

For extra credit, write a (separate) Java Swing program that contains a custom widget that extends `JComponent`. This widget should simply draw a horizontal line whose length can be adjusted by holding down the mouse button and dragging up (to make the line longer) or dragging down (to make the line shorter). The purpose of this exercise is to understand the relationship between event handling and drawing, which will help you in upcoming assignments.

Sun has some good documentation on this here:

http://java.sun.com/docs/books/tutorial/uiswing/14painting/

Submission

Email a **ZIP archive** of your assignment to `dis2_submissions@cs.rwth-aachen.de` before the due date. The ZIP-file should be named ⟨lastnamegroupmember1_lastnamegroupmember2⟩.zip where you fill in your last names. The subject of your email should be "`DIS2 Assignment 2`"; be sure to use this *exact* subject line as it will be used to filter assignment submissions for grading.

Your assignment archive should include your source code and everything necessary to compile and run your program. Be sure to document your source code. Include a short **PDF document** `README.pdf` that contains:

- the names and email addresses of all group members
- a short description of your design
- non-obvious things you did in your code (if any)
- anything that you think makes your design particularly optimized and/or elegant

Be prepared to discuss your solution in the next lab.

Grading

The assignment will be graded on the following rough scale:

- 1.0 - exceptional work that clearly went above and beyond what was given on the exercise
- 2.0 - exercise was completed satisfactorily as per the assignment specification
- 3.0 - exercise was completed, but has some problems
- 4.0 - incomplete exercise
- 5.0 - little or no effort was put into the exercise

Late assignments will *not be graded*.