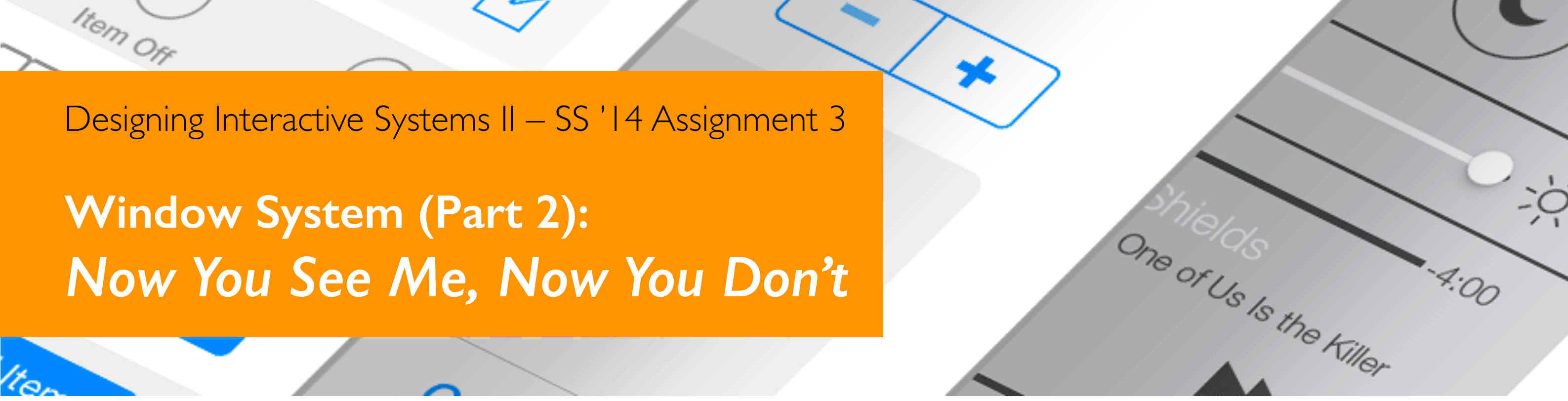

Designing Interactive Systems II – SS '14 Assignment 3

Window System (Part 2): *Now You See Me, Now You Don't*

Due: Monday, May 12th, 2014, **8:30 am**, in groups of 2 students via L2P.

Description

In this assignment, you will continue to build upon your Window System from the last assignment by adding features to draw windows, their window dressing and some basic event handling. As the primary goal of the exercise is for you to better understand the four-layered windowing system architecture discussed in the lectures, be sure to think about where each feature should go (Base Window System, Window Manager, or Application in this exercise). A significant portion of your grade for this assignment will be based on how faithfully you separate out the functionality into these windowing system layers.

Tasks

1. Window System

Start by expanding your `WindowSystem` class so that applications can make use of your windows, and so that your windows will be shown on the screen. To accomplish this, you will need to add the following features:

- Allow applications to create `SimpleWindows` that are associated with a `WindowSystem`.
- Display `SimpleWindows` on your "desktop" when they are created.

The drawing here should be kept very basic – windows should be drawn as simple, solid boxes in some color (e.g., grey) against the desktop, which should be drawn in some other color (e.g., black). Any fancy window dressing (should you choose to implement any) should be saved for the window manager (see next part).

You may find the following methods in `GraphicsEventSystem` useful:

- `void handlePaint()`
- `void drawLine(int inStartX, int inStartY, int inEndX, int inEndY)`
- `void drawRect(int inX, int inY, int inWidth, int inHeight)`
- `void fillRect(int inX, int inY, int inWidth, int inHeight)`
- `void setColor(Color inColor)`
- `void requestRepaint(Rectangle rect)`
- `void drawString(String str, int x, int y)`
- `Font getFont()`
- `void setFont(Font font)`

You will need to override the `handlePaint()` method in your `WindowSystem` class, and do all of your drawing in there. The `handlePaint()` method is inherited from `GraphicsEventSystem`, and currently does nothing. You can force the system to refresh a certain area of the desktop by calling `requestRepaint()`.

Again, we leave the design and implementation details up to you. For example, think about how developers coding an application will create a new window – will they call a method in `WindowSystem` which returns a new `SimpleWindow` object, or will they create `SimpleWindow` objects and then add them to a `WindowSystem`? There is no right answer here, although you may find some solutions are more elegant than others.

2. Window Manager

In this next part, you will implement a basic window manager that adds mouse input. In particular, you should implement a `WindowManager` class which adds a titlebar and a close button to all windows. The basic features your window manager should have are:

- Show a titlebar and close button for each window.
- Allow the user to move a window by dragging it around.
- Allow the user to close a window by clicking on the close button.

Again, you will have to make some design trade-offs here. For example, should the "window dressings" (titlebar, etc...) be drawn as part of the window or surrounding it? If they are drawn within the window, it makes moving the window easier, but take up valuable screen real-estate from the application; if they are drawn outside the bounds of the window, they will have to exist as separate `SimpleWindows`, and be moved together when the window is moved.

You may find the following methods in `GraphicsEventSystem` useful:

- `void handleMouseClicked(int x, int y)`
- `void handleMousePressed(int x, int y)`
- `void handleMouseReleased(int x, int y)`
- `void handleMouseMoved(int x, int y)`
- `void handleMouseDragged(int x, int y)`

Keep in mind that you may need to convert the coordinates before passing the events to higher layers.

Extend your `MyApp` program from last week so that it creates three windows and allows the user to drag them around and close them.

Testing Your Understanding

Answer the following questions:

1. Briefly describe and justify any design choices you made (if any) in your `WindowSystem` class.
2. Briefly describe and justify any design choices you made (if any) in your `WindowManager` class.

Extra Credit

Implement more bells and whistles to your window system/window manager. Sample "extra features" are given below, but feel free to use your imagination and come up with your own. Grading for extra credit will be heavily dependent on quality and creativity. Just doing all of the example features will not necessarily give you a lot of extra credit. Instead, we suggest that you pick only one improvement and think it through.

- Window re-ordering (e.g., clicking on the title-bar brings the window to the front).
- Fancier window dressing (border around the window, some decoration).
- Interactive window resizing by clicking on a window corner and/or edge and dragging.
- Minimizing a window to an icon.

Be sure to document which features you implemented in your **`README.pdf`**.

Submission

Submit your solution as ZIP archive (**`A3_GroupLetter.zip`**) in a group of two through the L2P classroom.

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 **PDF document** with the name **`README.pdf`** that contains:

- names and email addresses of all group members
- instructions on how to compile and run your source code
- answers to the questions (see Testing Your Understanding)
- non-obvious things you did in your code (if any)
- anything that you think makes your design particularly optimized and/or elegant

Make sure that your submission compiles and runs on the lab machines (JDK 8). Compile using `javac` on the command line only! **We will not start installing your favorite IDE or any other library you might want to use.**

Please note that assignments that do not meet the above submission criteria will *not* be graded. Be prepared to discuss your solution in the next lab. If you copy code without proper reference we will consider this as plagiarism.

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.