Program 9: Mouse and Keyboard Events

CS 617 Winter 2013

1 Goals.

1. To use mouse, keyboard, and window events.
2. To use some new graphics classes.
3. To use an inner class.
4. To build a graphics application without GUI controls.

2 Points and Polygons

Definition: Bounding box:
The smallest possible rectangle with vertical and horizontal sides that surrounds a 2D figure.

2.1 Tools:

You will need the Point class in the package java.awt.geom. Its data members are an x-coordinate and a y-coordinate, both ints.

- Constructor: `Point(int x, int y)`
- `int getX()` and `int getY()`
- `String toString()` will be useful for debugging.

You will also need the Polygon class in java.awt.

3 Instructions:

The P9 controller class. This class will not use any widgets – only a Frame and a Panel, and possibly dialog boxes.

Setup.

- Define a JFrame class named P9. Implement one Viewer panel to paint on (a Canvas).
- Your model can be inside the P9 class. It will consist of two Color variables, a Polygon variable, and two Point variables. All are initially null.
- Create a static final color palette (an array of colors) with at least 4 colors that look good together. Two or more of these colors should be your own creations, two can be standard Java colors. Give them names and assign a key code to each one.
- Use one one color for the background of your Canvas. Prepare to display a menu of the other three colors.
- Use an inner class to define at least one listener.

Action. There will be four stages of execution.

1. Color selection.
   - Display a menu of at least 3 colors, with their key-codes. Do this by drawing a string of text in the Canvas panel or by using a tabbed layout with the instructions on a separate tab.
   - Tell the user to type the key codes of two colors. Implement a KeyListener to capture those codes. Save them for later use.
2. Create a Polygon.
   - Tell the user to click the mouse 4 times in the graphics window.
   - Capture each mouse click using a MouseListener. Use the x and y to create a Point, and add that Point to your Polygon.
   - Display a black circle 5 to 10 pixels wide CENTERED on the point that was clicked.
   - After the 4th point is created, display the polygon. Fill it with the first color the user picked from the menu.
3. Create a Line.
   - Tell the user to click the mouse somewhere inside the polygon and drag it to a point outside the polygon. Display a black circle centered on each point.
   - Use a dialog box to display an error comment if the first click is outside the polygon. Keep trying until you get a good click.
   - Use a dialog box to display an error comment if the user does not drag the mouse to a point outside the polygon. Keep trying until you get a good click-drag sequence.
   - When you have a good click-drag, display a line between the two points. Use the second color the user selected.
4. Do one or more of the extra credit actions or say goodbye and thank the user for participating. Implement a window listener to show this message when the window is closed.

Due: March 30

- Turn in one zip file that contains your code and at least 8 screen shots that cover each stage of execution.
- **Extra credit, after parts 1–3 are completed:**
  (a) Activate the cursor keys. Let the user use the cursors to move the line. Move 3 pixels each time a cursor is typed. Don't move the Polygon. (Very easy: just change the x or y coordinates of both points that define the line.)
  (b) Activate the cursor keys. Let the user use the cursors to move the entire polygon. Don't move the line. Move 3 pixels each time a cursor is typed. (A little harder; you have to change the Points inside the Polygon.)
  (c) Let the user click on a point of the polygon and drag it. A click is a "hit" if the click is inside the circle that surrounds the point. When the mouse is released, move that corner of the polygon (This is more difficult because you need to find out how close the click is to each point of the polygon, then change the Point inside the Polygon.)