Zoom Panel 1.0 Requirements Specification

1. Scope

1.1 Overview

The Zoom Panel component provides a SWING panel that performs zoom (and other kind of transformations) for another JComponent. It will transform the graphics of the original component to a certain zoom factor and will receive the mouse events and delegate to it according to the zoom transformation. This panel will also provide scrollbars, so that the zoomed component can be scrolled nicely.

1.2 Logic Requirements

1.2.1 JComponent

The component will provide a concrete JComponent. This component will receive a different JComponent, for which it will perform the zoom. It will add the component to itself, so that when this zoom component is made visible, the zoomed component will also be visible.

This component should be implemented in such way that the zoomed component will not be aware that it is zoomed.

1.2.2 Zoom transformation

The component must support at least a natural zoom transformation. Other transformations are required also. They should not be provided only if they are really hard to be supported. The AffineTransform class should be used.

For instance, if the zoom is 0.5, then newX=0.5*x and newY=0.5*y.

This refers to painting on the zoom component's graphics using the paintComponent(..) method of the zoomed component, while applying a transformation to the graphics. Any other strategy is welcome as long as it does not require more resources (time and memory) than this one. Note that this component should make sure the zoomed component is never required to paint itself (to avoid a useless overhead). The zoomed component will only paint on the graphics provided by the zoomed component.

The component must support changing the transformation through API, during runtime.

1.2.3 Mouse events

The zoom component will receive all the mouse events, apply the zoom transformation and delegate them to the zoomed component. The events should execute immediately. They shouldn't be placed in the AWT event queue, in order to keep the order in which the events occurred.

Note that the zoomed component should receive the events as if it was not zoomed.

1.2.4 Scroll bars

The zoom component must provide scrollbars, if the zoomed component is not viewed entirely (while zoomed, of course). The scrolling will be performed relative to the zoom component, not the zoomed component.

The component should provide a way to set the view port through API.

1.2.5 No zoom

If the zoom factor is 1.0 (or the transformation is the identical transformation), this component should not receive the events, as this implies a useless step. It should just let the zoomed component receive the events. In addition, there should be no transformation applied to the graphics object, to avoid possible slowdown due to useless calculations.

1.2.6 Background

The zoom component will normally have a fixed size (though its size can change). It should be able to draw the zoomed component at least in the top left corner inside the view port, in case the zoom is very low. If the zoomed component is not able to draw beyond it's preferred size, or actual size, this component should fill the background with a configurable color.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool as the zoom panel for the diagrams.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5

2.1.4 Package Structure

com.topcoder.gui.panels.zoom

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The scroll bars - all the properties related to them
- The view port
- The transformation

These are required only through API, not configuration file.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

None.

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.