Práctica 3: Estela de ratón

David Burian[1]
Jindra Pikora[2]

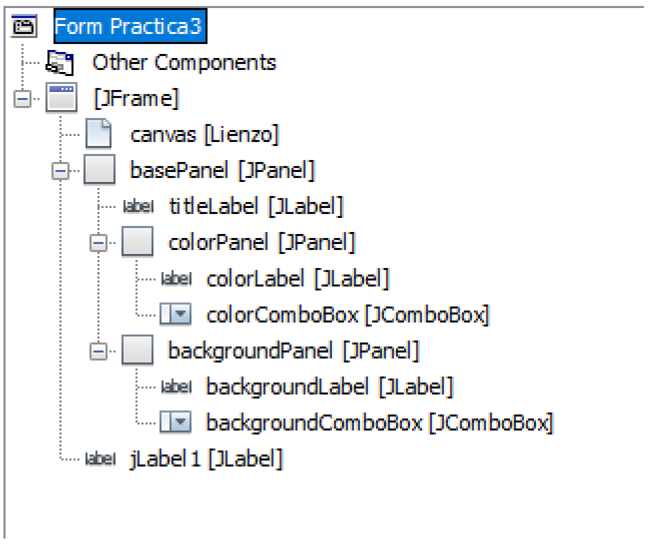
1 david.burian@me.com
2 jindrich.pikora@gmail.com

Introduction

Our goal was to design application, which demonstrates the movement of a mouse across the application window using 5 dots. The application was to allow user to change colors both the outlined path and the background of the canvas from 3 or more different colours.

Development

The application window consists of two areas.
First of which (basePanel) is where the name of the aplication resides (titleLabel), as well as two panels for controlling the colour of the mouse path (colorPanel) and the colour of the canvas background (backgroundPanel). In both colorPanel and in backgroundPanel there are labels which indicate what the combo boxes below the labels change.
The main feature of the application, the canvas, is in the second area of the application window.



```
Form Practica3
├─ Other Components
└─ [JFrame]
   ├─ canvas [Lienzo]
   ├─ basePanel [JPanel]
   │  ├─ titleLabel [JLabel]
   │  ├─ colorPanel [JPanel]
   │  │  ├─ colorLabel [JLabel]
   │  │  └─ colorComboBox [JComboBox]
   │  └─ backgroundPanel [JPanel]
   │     ├─ backgroundLabel [JLabel]
   │     └─ backgroundComboBox [JComboBox]
   └─ jLabel1 [JLabel]
```

Picture 1: Structure of main application window

Both combo boxes are used to change color and both offer the same selection of colours. Whereas colorComboBox changes the colour of a mouse path, the backgroundComboBox changes the background color of the canvas.
The path is painted using Graphics2D class. To paint straight onto the canvas, we used JPanel as the base class of the canvas class – Lienzo – and put the code to paint the mouse path into the inherited method paintComponent. Moreover the Lienzo class reacts to the mouse movement events such as mouseMove and mouseDragged and memorizes the locations of the mouse, which will be later used when drawing the path.

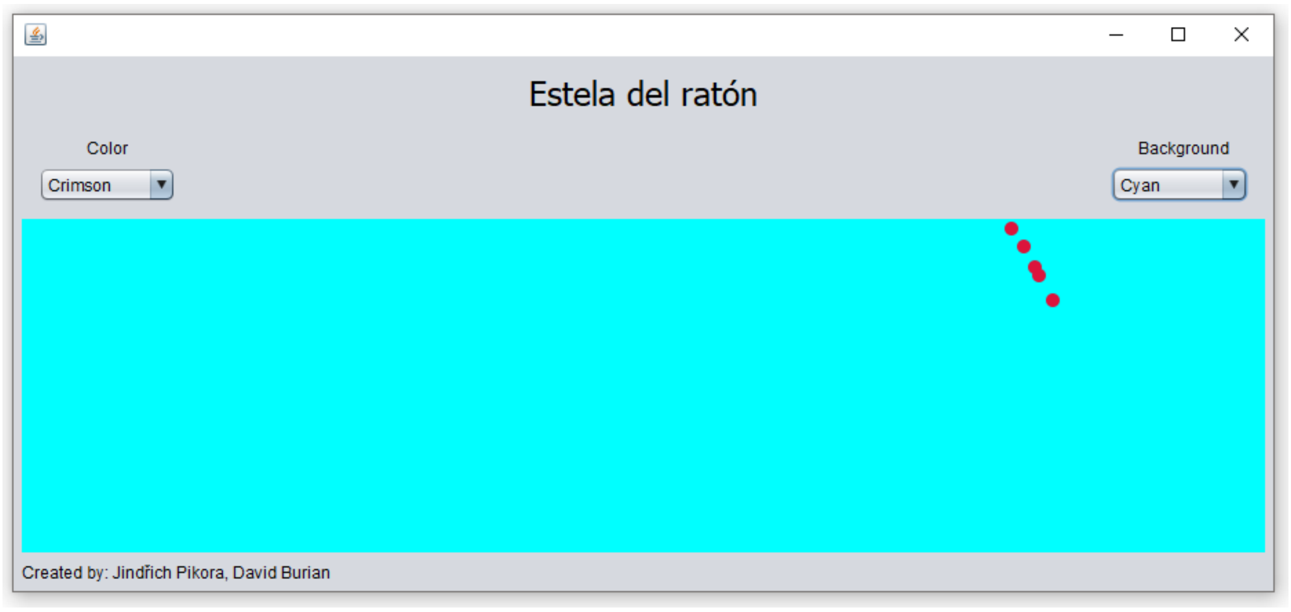
Evaluation of the user interface according to Shneidermans‘ and Plaisants‘ rules

1. Consistency

Thanks to the small number of controls, it was easy to achive consistent look when designing the application. Notice how both combo boxes are at the same height, giving them the same importance and not confusing the user which he/she should set first. The distance between both combo boxes also serves as a hint to the user, that the controls are unrelated in terms of function. By contrast combo boxes appear to be glued to the canvas, making it obvious what do they change.

2. Universal usability

To enable the user to set the colors of both path and background, we chose to introduce two combo boxes. That way user can freerly select and combine colors however she/he wants.



Picture 2: Design of the application window.

3. Informative feedback

Here the predefined combo boxes do the most of the work. Not only do they react to mouse hover events so that the user knows these controls are interactive, but also when clicked they display a list of options from which to select. When an option is selected, it is displayed in the combo box itself to represent the current state.
Because the reaction to changing the combo box state is instantaneous we chose not to tell the user what he/she has changed. Instead we directly execute the change, making it visible to the user.

4. Task flow

The task of changing colour in our application is really quick and therefore introduces little chance for the user to get confused as what he/she is supposed to do next. Label above the combo box clearly tells the user it´s purpose. Thanks to the interactiveness of combo box (described above) user is easily guided through the selection of a colour. Upon selecting a colour, the corresponding element is instantly updated and the state in the combo box is replaced, making it clear that the action is completed.

5. Preventing errors

To prevent user from entering a colour, which the program does not know about, combo boxes were used, giving the user just a predefined list of options to choose from.

6. Easy reversal actions

To change the colour back to the previous one, user just has to select the previous color from the drop-down list. If we would assume unexperienced user a „Back“ button could have been used. This would, however, introduce a little confusion to our design, creating two ways how to change color of an element. Therefore we decided to use only combo boxes.

7. Keep user in controlling

As the only purpose of our application is to change colours of two different elements, it is easy to give the user full control. Nevertheless if the designer gave the user full control, it would have been easy for the user to make mistakes. Combo boxes are good compromise. For more demanding users a color palette would be the perfect solution.

8. Short-term memory

Since named combo boxes require the user to remember little to no information, I would evaluate our design as short-term memory firendly.

Conclusion

We developed an application to demonstrate a movement of a mouse pointer over a canvas. Along the way we learned how to use Graphics2D class, how to paint directly on JPanel and how to use comboBoxes. To design easy-to-use interface, we followed the Shneidermans‘ and Plaisants‘ rules.

In the end we ended up with an application which functions properly and is easy-to-use at the same time.