Object Pointing: A Complement to Bitmap Pointing in GUIs Summary

Svetlana Slavova
sds797@mail.usask.ca

ABSTRACT

This work is a summary of the paper, presented in [1].

INTRODUCTION

Pointing is a ubiquitous technique that is used in the user interfaces to select targets. The Fitts' law specifies that the selection time can be improved if the distance to the target is decreased and/or if the width of the target is increased. Different approaches, based on the Fitts' law, such as drag-and-pop and semantic pointing, are proposed by researchers, in order to facilitate target acquisition, assuming that the "screen cursor is a tool for pixel selection". This pointing technique is called bitmap pointing.

PROBLEM DEFINITION

The bitmap (BMP) pointing corresponds to the real-world object acquisition – it requires the user to complete the selection movement, even in the cases, in which the destination item is obvious and unambiguous. As a result, the BMP pointing sends more information to the device, than actually is needed, and the unnecessary data is simply ignored.

The main question is: How to make the pointing easier than in the real-world? This would decrease the selection time and would increase performance, respectively. As a result, the amount of unnecessary information, sent to the system, during movement from one object to another, would be reduced.

APPROACH

The authors of [1] suggest that the user interface can be augmented with an additional cursor, called Tim. Within an object, the Tim cursor moves in parallel with the input device, for example the mouse. However, when the Tim cursor reaches the boundary of the current item, it simply jumps to the closest one, which is located in the same direction as the direction of the movement of the mouse. In order to achieve a correct jump action, not only the position of Tim cursor is taken into account, but also its current velocity and acceleration. Moreover, if there is no object in the direction, the additional cursor does not leave the object.

This approach, called object pointing (OP), allows none of the cursors to be displayed to the user, since an object is highlighted all the time, and by moving the mouse, the user is able to change the selected item. Furthermore, the distance to the target is reduced, which suggests that the selection time will decrease.

EVALUATION

Experiment setup

Two types of experiments are conducted – in 1D space and in 2D space. The former consists of three models – BMP pointing, OP with visible Tim cursor, and OP without Tim cursor displayed. The latter contains two models – BMP pointing and OP without Tim. The observed variables are the movement time between two object selections, the amplitude of the mouse (physical area and screen area), and the pointing difficulty.

Results

As expected, the OP technique reduces the amount of sent information to the system. The results confirm that object pointing is faster than the BMP technique. The advantage of this approach increases even more, when the pointing becomes more complex, due to high density of the objects or high level of index of difficulty. In addition, the footprint of the input device is reduced, which is beneficial, in case of a restricted movement area of the input device.

CONCLUSIONS

The paper, presented in [1], proposes a promising interaction technique – object pointing, which facilitates item selection. This new approach makes the target acquisition easier and faster than the BMP mode, since an object is selected all the time, and the focus jumps from the current item to the next one, depending on the movement direction of the input device, as well as on its velocity and acceleration.

REFERENCES

1. Guiard, Y., Blanch, R., Lafon, M. Object Pointing: A Complement to Bitmap Pointing in GUIs. *Proceedings of Graphics Interface 2004*, London, Ontario, May 17-19, 2004, 9-16.

Object Pointing

Implementation Description

Svetlana Slavova
sds797@mail.usask.ca

ABSTRACT
This paper is a description of my implementation of the object pointing technique, presented in [1].

DESCRIPTION

The program represents an implementation of object pointing, which introduces additional cursor in the user interface, called Tim. The environment consists of two 2D objects, represented as dark-blue boxes. When one of the objects is selected, its color changes to red. Target selection is realized by the Tim cursor. When the Tim cursor is within an object, it follows the movement of the mouse. However, when the Tim cursor reaches the boundary of the object, it jumps to another object, which is located in the same direction, as the direction of the mouse movement. If there is no item in that direction, the Tim cursor does not leave the current object. The Tim cursor is visualized as a white dot and a green concentric circle around it, which helps the user to locate the cursor.

IMPLEMENTATION DETAILS

The program is implemented in Java, using Java Swing. Two classes are developed, as follows:

- *ObjectPointing*. This is the starting point of the application. The class is responsible for the creation of the program's frame. It adds to the frame the mouse surface, which is represented by an object of class *MousePanel*, described below.
- *MousePanel*. This class represents the mouse surface. It handles the mouse events, reacts, according to the performed actions, and updates the user interface. The main components of the class are the following:
  - Method *drawObjects* – it displays the background of the user interface, as well as two objects, which are potential targets;
  - Method *highlightObject1* – it displays the left object of the user interface in red;
  - Method *highlightObject2* – it displays the right object of the user interface in red;
  - Method *drawTimCursor* – it displays the Tim cursor on particular coordinates as a white dot and a concentric circle in green;
  - Class *MyMouseMotionAdapter* – it handles the movements of the mouse. Every time when the mouse is moved, method *mouseMoved* of the class is called. It updates the mouse coordinates and invokes method *paintComponent*, which repaints the user interface;
  - Method *paintComponent* – it contains the main logic of the program. It calculates where to display the Tim cursor, analyzes the object to be highlighted, as well as invokes the methods that draw the objects. The *paintComponent* method is called when the user interface has to be repainted. The previous and the current coordinates of the mouse are taken into account, in order to calculate the direction of the mouse movement and the movement length. The algorithm of the method is presented on the diagram below.

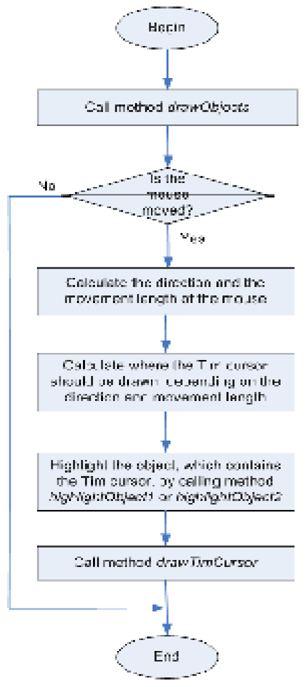
Begin
Call method drawObjects
Is the mouse moved?
No
Yes
Calculate the direction and the movement length of the mouse
Calculate where the Tim cursor should be drawn depending on the direction and movement length
Highlight the object, which contains the Tim cursor, by calling method highlightObject1 or highlightObject2
Call method drawTimCursor
End

Figure 1. Algorithm of method *paintComponent*

HOW TO COMPILE & RUN

- In order to compile the program, type the following command in the terminal window:

javac ObjectPointing.java

- In order to run the program, type:

  java ObjectPointing

  or

  Double-click on jar file

  ObjectPointing2D

USER INTERFACE

When the application is started, a window, shown on figure 2, is displayed. Once the mouse is located in the window of the program, one of the objects is highlighted, as shown on figure 3. When the Tim cursor is located within an object, it follows the movement of the mouse. However, when the Tim cursor leaves the boundary of the object, it highlights the item that is located in the same direction, as the direction of the mouse movement (figures 4 and 5). Figure 6 shows that if there is no object in the direction of the mouse movement (in this case object on the right side of the red object), the Tim cursor does not leave the current item.

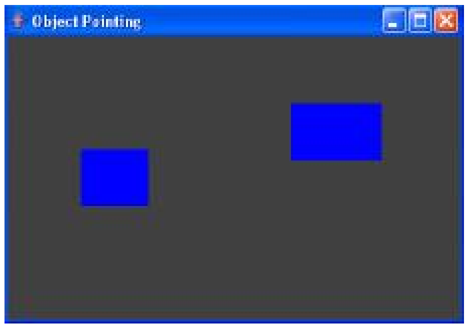

Figure 2. Initial window of the application

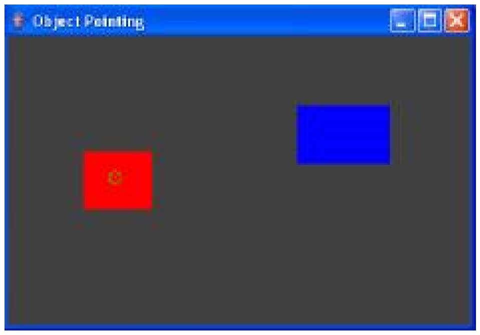

Figure 3. Tim cursor is within an object

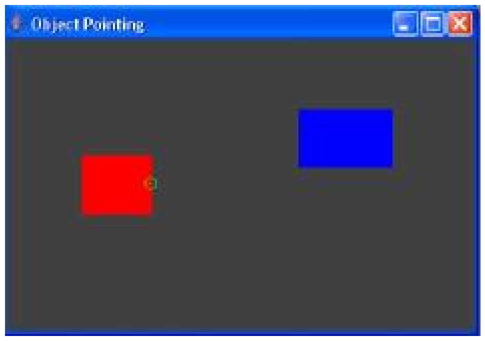

Figure 4. The Tim cursor reaches the boundary of the object

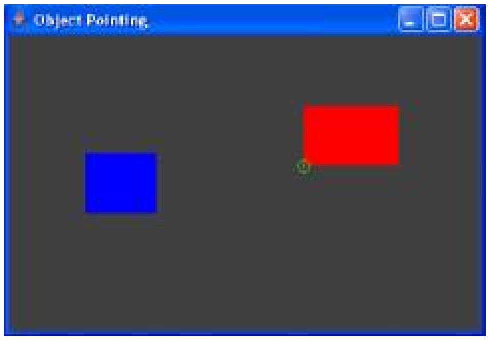

Figure 5. The Tim cursor highlights the object that is in the same direction, as the mouse movement

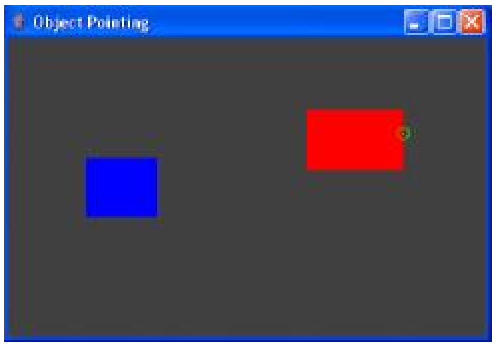

Figure 6. The Tim cursor does not leave the object, if no object is located in the direction of the mouse movement

REFERENCES

1. Guiard, Y., Blanch, R., Lafon, M. Object Pointing: A Complement to Bitmap Pointing in GUIs. *Proceedings of Graphics Interface 2004*, London, Ontario, May 17-19.