Drag-and-Guess: Drag-and-Drop with Prediction

Takeshi Nishida
Department of Computer Science,
The University of Tokyo
tnishida@ui.is.s.u-tokyo.ac.jp

Takeo Igarashi
Department of Computer Science,
The University of Tokyo / PREST JST
takeo@acm.org

ABSTRACT
Drag-and-guess is an extension of drag-and-drop that uses predictions. As the user starts dragging an object, the system predicts the drop target and presents the result. If the target is hidden in a closed folder or beneath other windows, the system makes it temporarily visible to free the user from manual preparation. The user can accept the prediction by releasing the mouse button and the object flies to the target, or reject it by continuing the dragging operation, thereby switching to traditional drag-and-drop seamlessly.

Keywords: Drag-and-drop, Drag-and-guess, Prediction

INTRODUCTION
We demonstrate an extension of traditional drag-and-drop that uses predictions, called drag-and-guess (DnG) (Figure 1). As the user starts dragging an object, the system predicts the drop target and responds by revealing the predicted result. If the target is not visible (e.g., hidden in nested hierarchical folders or outside the area visible on the screen), the system automatically makes the target location temporarily visible. If the prediction is correct, the user can accept it by releasing the mouse button, when the object automatically drops on the target. If the prediction is wrong, the user can easily discard it by continuing the dragging operation. This makes the predicted result disappear and the user switches to a traditional drag-and-drop operation seamlessly.

DnG can be useful in various situations. While distributing e-mails in an inbox to nested folders, the system can predict the appropriate target folder based on its content. While editing a spreadsheet, the system can predict the target based on any regularity observed in previous drag-and-drop operations. In opening a document with an application program using drag-and-drop, the system can find a probable drop target based on the document file type and the usage frequency of the application.

DRAG-AND-GUESS
Figure 2 shows the behavior of DnG. DnG begins, as with drag-and-drop, when the user grabs an object. The system presents the predicted result, and shows a line connecting the cursor and the predicted target to indicate active binding. If the target is not visible, the system makes it temporarily visible. For example, if the target is hidden in a contracted path within a tree, the system automatically opens the path to the target. To accept the prediction, he simply releases the mouse button and the object flies to the target. This process is presented to the user as an animation; the user can start the next operation before the animation terminates.

UIST '06, October 15-18, 2006, Montreux, Switzerland

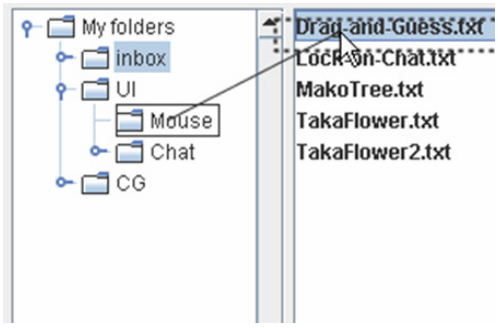

Figure 1 Drag-and-guess in action.

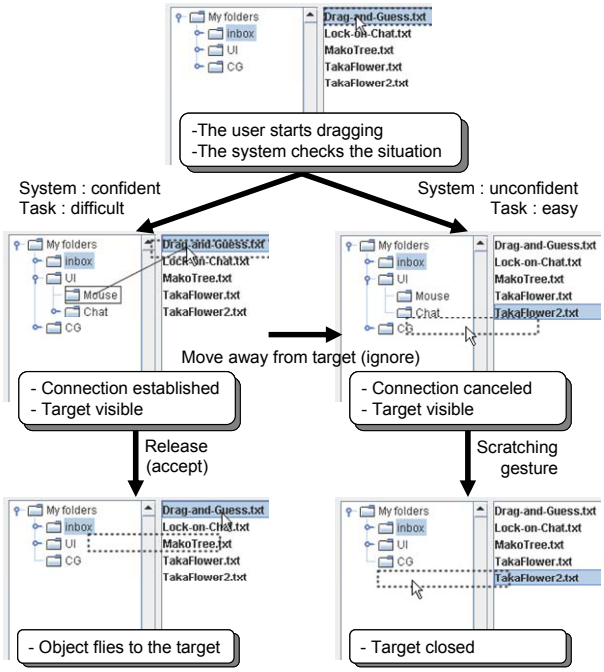

Figure 2 Behavior of drag-and-guess

If the prediction is incorrect and the user wants to drop the object somewhere else, he can do so by continuing the drag operation and ignoring the prediction. The connection disappears when the cursor moves away from the predicted target, allowing the user to switch seamlessly to a standard drag-and-drop operation.

This design is a result of iteration cycles of prototyping and pilot studies. We revised the system to establish the binding to the predicted target only when it is confident of the result and the task is difficult to perform manually.

BENEFITS

DnG improves drag-and-drop in a number of ways. First, it saves the user from tedious pointing tasks. If a prediction is correct, the user has only to hold the mouse button on the object, confirm the target, and release the button. This is much faster than moving the cursor carefully to the target location, especially when the target is remote or very small.

Second, it saves the user from the tedious tasks of making the target location visible, such as opening multiple folders in a tree, rearranging windows to make both source and target visible, scrolling the view in the window, and so on.

Compared to other possible interaction techniques using predictions, DnG has certain advantages. First, it does not introduce additional widgets or consume extra screen space. Whereas interactive techniques such as showing predicted results as buttons [5] require additional space, DnG presents only temporary visual feedback.

Second, the target is displayed in its original context, which makes it easier for the user to understand its location. For example, if a predicted result is shown on a button as the name of the destination folder then the user cannot distinguish between multiple folders with the same name but in different locations.

Third, DnG supports a smooth transition to standard drag-and-drop. This is very important to compensate for prediction errors. Finally, DnG is designed to be a common front-end interface. This frees the user from having to learn how to use a specialized interface for each application, while enjoying the benefits of predictions.

DEMONSTRATION

We implemented various prototypes to demonstrate the effectiveness of our technique: filing (Figure 1), spreadsheet (Figure 3) and spider solitaire (Figure 4). The filing prototype uses a random prediction with a controllable accuracy for user study purposes. The spreadsheet prototype predicts the target cell based on the patterns in the operation sequence and the spider prototype predicts the target column based on simple heuristics.

RELATED WORK

Many techniques have been proposed to improve target selection. Semantic pointing [3] dynamically adjusts the control display gain to make it easier to catch the target. Drag-and-pop [2] temporarily moves the potential target towards the cursor, as the user starts dragging an object. Delphian Desktop [1] predicts the target using the linear relationship between the peak velocity and the distance to the target. The cursor jumps to the predicted location when the system detects the peak velocity.

One shortcoming of these techniques is that the benefits are decreased when the potential targets are densely located. In addition, it requires considerable effort to make the target visible in advance. DnG overcomes these problems by using the knowledge of the application and the grabbed object.

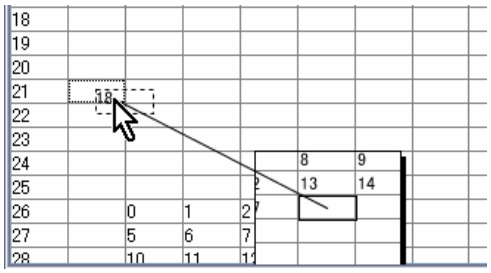

Figure 3 Table with drag-and-guess. The system shows a thumbnail when the target is out of view.

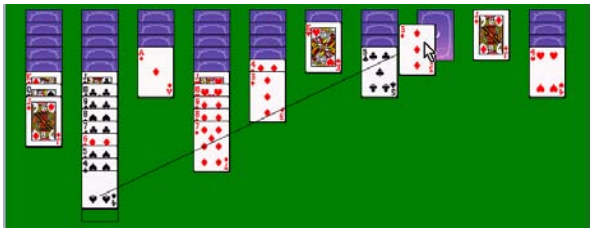

Figure 4 Spider solitaire with drag-and-guess

Fold-and-drop [4] allows the user to leaf through overlapping windows while performing DnD. Windows are folded like paper in response to the cursor passing over. This reduces advance manual preparation; however, it still requires considerable effort to find the desired window within a stack of windows.

CONCLUSIONS AND FUTURE WORK

We proposed an extension of drag-and-drop that uses predictions. We discussed how it would facilitate the operation and implemented various prototypes. We plan to conduct a user study to confirm our findings.

REFERENCES

1. Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. Predictive interaction using the delphian desktop. In *Proc. UIST 05*, 133–141, 2005. ACM Press.
2. Patrick Baudisch, Edward Cutrell, Dan Robbins, Mary Czerwinski, Peter Tandler, Benjamin Bederson, and Alex Zierlinger. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. Interact 2003*, 57–64, 2003.
3. Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. CHI 04*, 519-526, 2004.
4. Pierre Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. UIST 04*, 193–196, 2004.
5. Richard B. Segal and Jeffrey O. Kephart. Mailcat: an intelligent assistant for organizing e-mail. In *Proc. AGENTS 99*, 276–282, 1999.