Tossing Objects in a Desktop Environment

David Rogers, Feimo Hou, Chandelle Vuolo, Benjamin B. Bederson
Computer Science Department
University of New Mexico
Albuquerque, NM 87131
(drogers, feimo, bederson@cs.unm.edu)

ABSTRACT
We have implemented an interface which supports tossing as an alternative to screen-wide dragging of icons, and have explored how this basic mechanism may be generalized. We discuss its advantages and limitations, and report the results of our user testing. Preliminary results show that tossing is intuitive and is perceived to be faster than drag and drop.

KEYWORDS
Tossing, animated interfaces, user interfaces, animation.

INTRODUCTION
Animated interfaces have been proposed as a way of giving the user a better understanding of actions taken by objects on the screen. They allow intuitive transfer of information from the system to the user [1][5], and help the user to navigate large information spaces [2][3][4]. The SELF system [1] incorporates animation techniques from cartoons in order to maintain the user's understanding of the interface, and give entities a more consistent presence. Animated icons [5] utilize animation to give the user information about use and properties of a specific icon. Pad++, a multi-scale interface [2][3], incorporates animated zooming into data as a method of exploring large information spaces.

A widespread interaction that users perform with current interfaces is dragging objects and dropping them onto other objects, expecting certain actions to take place as a result. For example, dropping an object onto a trash can icon causes that object to be "thrown away". Note that *the action of dragging the icon has no semantic meaning in the drag and drop action*, it is merely a way of getting two objects into a physical relationship which *itself* denotes an action, and tells the system that an action is required. We feel there is a more natural method of achieving this same interaction, which allows the user to indicate which objects are to interact. In our system, the user makes a simple gesture to indicate a task, and the objects are animated to complete that task. In addition, tossing has the potential to be much faster to execute than drag and drop because the act of moving the mouse long distances is eliminated.

TOSSING
We have implemented *tossing*, a gesture which supplements drag and drop. A *toss* is made with a pointing device while holding a graphical object. It is analogous to throwing a piece of paper into a trash can or a file into an "in basket". In our interface, this gesture allows the user to initiate an action which is completed by algorithms and information resident in the system. It is our hope that use of computation by the interface will make the user's task simpler, as it utilizes computational resources which are currently not exploited by other interfaces.

IMPLEMENTATION
A *basket* is any item on the screen which can receive a tossed object. Receiving an object involves performing some action on that object, i.e. erasing a file, copying a file, or writing a file to disk. Our interface is designed to allow users to create baskets of their own, and to create actions to be performed on received objects. A *ball* is any item on the screen which can be tossed. Note that an object can be both a basket and a ball.

The main difficulty in tossing is deciding which screen object is the intended target of the toss. We have implemented a *Basket Manager* for this purpose, which tracks all potential targets, and chooses the most likely target for a toss. A ball, target pair is then passed to the *Animator* routine, which forces an animation path for the ball to end in the target. Thus, the user never misses completely, but he may throw something to the wrong target. Note that if one basket is active, the user will never miss, no matter where the object is tossed.

The problem of tossing an object to an incorrect target

Petitioner Samsung Ex-1024, 0001

would render tossing useless if it were used in a situation in which an incorrect target choice by the system produced undesired or dangerous results. The drag and drop mechanism is successful because it is explicitly directed, and there is little chance of a missed target. If we are to consider tossing as a useful alternative to drag and drop, we must demonstrate situations in which tossing is accurate enough to be acceptable, and where its ease of use allows it to be helpful.

The default tossing environment, in which there is one possible target on the screen, has 100% accuracy, so the question is one of preference. If the user likes tossing, s/he could use it effectively with one target.

A more general situation has several active targets, but allows the user to place them so as to maximize accuracy. Or, many baskets may occupy the screen, but only some of them may be active at any moment. During a session, a user may change which basket is active according to the specific task being perfomed. Additionally, the activation of targets could be made automatic, so that each application was able to activate targets with which it frequently interacted, or that the user found most useful to have active for that application. In a word processing application, perhaps the printer would be the active target, while in an archiving application, tossing would activate a tape drive.

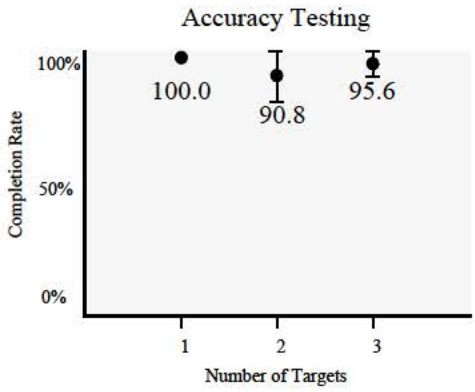
Due to both hardware and software limitations, tossing has limited accuracy. It works best in situations with a single target, or where each of several targets are all satisfactory. An example of this can be found in the solitaire game on the Macintosh Performa. Here, a card may be tossed, and the system will choose the best placement for the card from the possible legal moves. Users often don't care which of the legal moves are executed, so tossing is an excellent interface for this game.

USER TESTING

We conducted testing in which people tried the system with various numbers of targets, and then were asked to perform two accuracy tests, one with two targets active, and one with three targets active. These targets were placed in the corners of the screen, as shown in the figure at the beginning of this paper. The two target test used the upper right and lower right targets, and the three target test used all three targets. Users were also asked for specific data on ease of use, and finally for general comments about the system. Of the users tested, two had used the system before.

Some comments from users were -"I would like it for default type tasks." "Easier than dragging it all the way across a screen, particularly for some of the larger screens" "Actually much more efficient than dragging - when it works" "Takes skill with the mouse ..."

Users found that a toss for a specific target need not be directed perfectly towards the target. Once users understood how the system worked, they made tosses which



were in the general direction of a target, but were exaggerated in order to create interesting animations, or to insure selection of a particular target. For example, a toss to the lower left of the screen selected the lower right corner target in the two target demo, and created an interesting animation in the process. Apparently, users got better at tossing as they used the system, for the three target test shows an increase in accuracy over the two target test.

Physically manipulating the mouse to indicate a toss is an action that comes more easily to some than others. Some people found the system intuitive and easy to use and considered it a good idea, and some had a difficult time with the mouse, or didn't understand what the system was doing with their gestures. In general, however, the users felt tossing worked well for one target, but found that the system did not always detect a toss gesture.

CONCLUSION

We have implemented the general action of tossing, and have explained how it may be used in several specific and general interface interactions. It seems that the tossing mechanism could serve as a useful option for novice users in current interfaces, and could be used more extensively by expert users who are able to gesture more accurately.

This interface was written entirely in tcl/tk and is available by anonymous ftp from ftp.cs.unm.edu.

REFERENCES

[1] Bay-Wei Change, David Ungar, Animation: from Cartoons to the User Interface. UIST '93.

[2] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface, *Proceedings of 1993 ACM SIGGRAPH Conference*, 57-64.

[3] Bederson, Hollan, Perlin, Meyer, Bacon, & Furnas. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics, Journal of Visual Languages and Computing, in press.

[4] Lamping, Rao, & Piroli A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies, CHI'95, 401-408.

[5] Baecker, Small, Mander Bringing Icons to Life, CHI'91, 1-6.