ISSUES AND EXPERIENCE IN DESIGNING TWO-HANDED INTERACTION

Stéphane Chatty

Centre d'Études de la Navigation Aérienne
7 avenue Edouard Belin
31055 TOULOUSE CEDEX
FRANCE
+33 62 25 95 42
chatty@dgac.fr

ABSTRACT

Considering that direct manipulation interfaces using a pointing device could be more efficient with the addition of a second pointing device, we are designing and implementing two-handed interfaces for air traffic controllers. This paper describes the interaction styles we imagined for such interfaces, and some issues raised by their implementation.

KEYWORDS: two-handed interaction, interaction style, multimodal interaction, air-traffic control

INTRODUCTION

Though their construction is still a matter of research, graphical interfaces are now widely used. Most of them make use of a pointing device that users manipulate with their dominant hand. This has lead to the introduction of a number of interaction styles centered around that pointing device: buttons, menus, point-and-click, drag-and-drop, etc. Such interaction styles allow interface designers to build systems that are easy to use. However, we believe that the efficiency of such interfaces can be improved. In the real world, we perform many tasks with both hands, because it is more efficient. Because of these natural skills, drawing pictures with a MacDraw-like system is sometimes frustrating: a significant part of the time is spent in moving the mouse around to select tools, locking objects so that they do not move when manipulating them, etc. This is very similar to handcrafting with one hand behind one's back. It is interesting to note that keyboard short-cuts are a way for us to use our non-dominant hand when drawing, and to avoid unnecessary movements with the dominant one. This provides grounds for exploring two-handed interaction, as suggested by Buxton [2].

Apart from drawing tools, a number of application domains could benefit from such interfaces. Among these are the domains where users are well-trained professionals, whose attention is focused at the task they are performing. Air-traffic control is such a domain. At CENA, we are developing two-handed interfaces for controllers in order to test that belief with experiments and measurements. This paper describes the design and implementation issues raised by such interfaces, and our solutions to these issues.

INTERACTION STYLES

Today, the interfaces provided to air-traffic controllers essentially consist of a presentation of the situation in air-space: it is the so-called "radar image", which is composed of maps and a number of symbols and vectors representing way-points, aircraft and their past positions, speed, etc. Though there are significant differences among national systems, one can consider that there is no real interaction with the radar image. A number of countries are working on new interfaces that allow controllers to enter information in the system with modern interaction techniques. At CENA, we are also exploring the hypothesis that controllers might be able to plan their work by manipulating aircraft future trajectories. This is why we are investigating efficient techniques for interaction with curves and objects moving along them, including two-handed input.

Before examining two-handed interaction techniques, one point has to be made. There probably is no task for which two-handed input should be the only way to perform operations: there will always be situations in which one hand is used for another task, such as holding a sheet of paper. This means that all systems based on two-handed input should be usable with one hand only. Obvious design rules suggest that one-handed and two-handed actions for the same operation should be similar, and that one should be easily inferred from the other. We suggest that this requirement is most easily met when using paradigms from the real world. In our opinion, interfaces based on such paradigms just need to be extended for two-handed input according to their paradigm. For instance, we have a good intuition of what happens if we pick an object with one hand and drag it; similarly, something predictable should happen if we pick an object with both hands and stretch it. The interaction styles we use follow that rule.

A simple way to smoothly extend one-handed interfaces consists of adding a second pointing device that can be used in the same way as the first. This allows users to save a considerable amount of time when pressing buttons or selecting tools: for

instance, the non-dominant hand can select tools while the dominant one rests on the object which is being manipulated. Such interfaces can still be used with one hand: they are just more efficient with both hands. Xerox PARC's Toolglass [1] is a sophisticated version of that: having the tools located on a transparent palette that can be moved around allows users to keep their focus on the object of interest. In our air traffic control interface, such simple parallel interactions are used to modify several global parameters of the radar image, while interacting with it: users can change the zoom factor, or even the time (we allow controllers to look at future situations).

Another way to use two pointing devices is to combine their actions. We often use our non-dominant hand to hold objects while performing precise operations on them. We also use it in coordination with the dominant hand to provide additional strength, or to manipulate objects that are more precisely moved when held from two distant points. Traditional interfaces have replaced the second hand by some kind of magic: when we move one end of a segment, the other end is held by an invisible hand. What we suggest here is to disable that magic when two hands are at work: the non-dominant hand can hold the end of the segment, and if we finally decide to move the whole segment, it is ready for action. The only necessary adjustment is a tolerance for its imprecise movements. We used that kind of movement combination (which we called hold-and-pull) in our interface, and it seems very promising.

IMPLEMENTATION ISSUES

In order to develop two-handed interfaces, we enriched a user interface toolkit with two-handed input capabilities. This lead us to identify a number of technical issues raised by such interfaces, and to consider their implementation as a special case of multimodal interaction. We extended Whizz, a toolkit aimed at describing the behaviour of direct manipulation interfaces [3]. Behaviours are attached to graphical objects and described as data-flow graphs, with sources such as clocks, the user's actions or active values, and filters that perform computations such as projections onto trajectories or numeric operations. Graphical objects have slots that can be connected to the output of such filters: when an event occurs in a source, it is processed through the flow graph, and the graphical objects react accordingly.

One of the novelties of two-handed input that has an impact on interface construction is the possibility to perform parallel actions. During a long action such as a drag, information about the object being dragged and the visual feedback have to be stored. There is usually no mechanism for that, and programmers store that information as global variables. The introduction of parallelism makes such techniques obsolete: there can be more than one action, and hence more than one feedback at a time. To support such parallel interaction, toolkits will have to provide a specific mechanism. When such mechanisms are present, building pure parallel actions is straightforward. For instance, with Whizz, deforming a segment with two hands is done by connecting the two mice to the two ends of the segment (figure 1).

Once actions can be performed in parallel, the next issue is

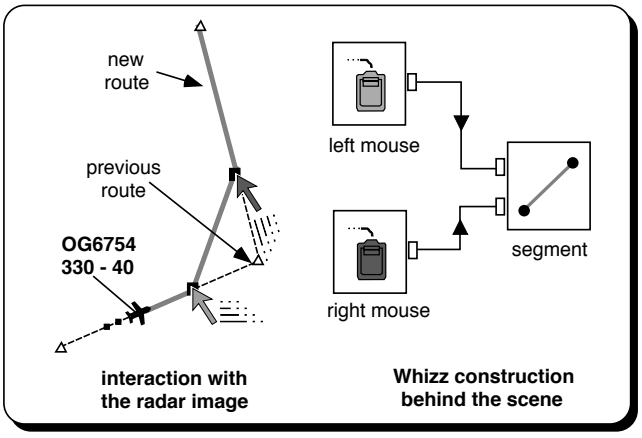

Figure 1: Changing an aircraft route.

the combination of actions, as in hold-and-pull interactions. When combining movements, one has to manipulate events that do not occur at exactly the same time. This is a serious issue when using a sequential programming language. It is very similar to the notion of fusion for multimodal interfaces mentioned in [4]: events issued at different times from different sources have to be combined to produce a significant action. It appears that two-handed interfaces are a special case of multimodal interfaces, using two identical channels, and with fusion performed at the lexical level. In order to support that, we are adding to Whizz special filters for combining asynchronous flows of events. This will allow programmers to decide that an object can be moved only if it is pulled with both hands, for instance. With such tools, we expect to be able to explore new kinds of combined interactions.

CONCLUSION

We have presented in this paper the design and implementation issues raised by the introduction of two-handed interaction in air-traffic controllers' interfaces. We also explained the relations between two-handed interaction and other kinds of multimodal interactions, and we described the extensions we made to the direct manipulation toolkit Whizz in order to support such interactions. Future work will include tests with real users, in cooperation with human factors specialists. We hope that these tests will confirm our opinion that two-handed interaction can improve the efficiency of certain kinds of users.

REFERENCES

1. E. Bier et al. Toolglass and magic lenses : the see-through interface. In *Proceedings of the ACM SIGGRAPH*, 1993.

2. W. Buxton and B. Myers. A study in two-handed input. In *Proceedings of the ACM SIGCHI*, 1986.

3. S. Chatty. Defining the behaviour of animated interfaces. In *Engineering for Human-Computer Interaction*, North-Holland, 1992.

4. L. Nigay and J. Coutaz. A design space for multimodal systems: concurrent processing and data fusion. In *Proceedings of the ACM SIGCHI*, 1993.