Pen Computing for Air Traffic Control

Stéphane Chatty
Centre d'Études de la Navigation Aérienne
7 avenue Édouard Belin
31055 TOULOUSE CEDEX
France
chatty@cena.dgac.fr

Patrick Lecoanet
Centre d'Études de la Navigation Aérienne
Orly Sud 205
94542 ORLY AÉROGARES
France
lecoanet@cena.dgac.fr

ABSTRACT
Modernizing workstations for air traffic controllers is a challenge: designers must increase efficiency without affecting safety in any way. Air traffic control is a time-intensive and safety-critical activity, and thus interaction efficiency and low error rates are crucial. Classical interaction techniques have been used in prototype workstations, but the resulting efficiency is not always satisfactory. This leads designers to consider more advanced interaction techniques. This paper reports on the design and a preliminary evaluation of the first prototype of project IMAGINE, which represents the second generation of graphical interfaces for air traffic control. This prototype, GRIGRI, uses a high resolution touch screen and provides mark based input through the screen. The use of gestures, as well as the use of multi-modal techniques, make interaction faster, and closer to the controllers' habits.

Keywords
Air traffic control, gesture recognition, mark-based input, pen computing, touch-screen, direct manipulation, prototyping

INTRODUCTION
Air traffic control (ATC) is a fascinating application domain for human-computer interaction research, but it is also very challenging to interface designers. They face many difficulties, especially since safety requirements are essential. First, there are the requirements for the interface itself: low error rates are necessary for obvious reasons and interaction efficiency is required to avoid fatigue and to improve safety during periods of time-intensive work. Second, air traffic controllers and their management are understandably cautious about changing tools that affect hundreds of lives, and tend to resist new technologies. Other reasons include the need for finely tuned transition plans (the job must be done, even when the system is being changed), the cost of learning new procedures, and the cost of developing high quality hardware and software for a narrow market. Finally, even access to end users is sometimes a problem. In France, for instance, the number of air traffic controllers is often considered too low to allow for secondary tasks such as designing future systems. The consequence is that rather old technologies and designs are still used in air traffic control centers. Direct manipulation graphical user interfaces have been commonly available for ten years. But the air traffic control rooms of most advanced countries are still equipped with systems designed in the 60s and 70s, that are more and more expensive to maintain.

However, the need for new interfaces for air traffic control is getting more and more perceptible. Air traffic has been growing rapidly for the past ten years, and does not appear to be stabilizing. The increased workload has so far been balanced by a growth of the number of controllers (sometimes), and by growing individual workloads (most often). If the traffic keeps growing, more computerized systems and more efficient tools will be necessary. Despite the difficulties mentioned earlier, most air traffic control agencies embarked on projects based on graphical interfaces. The past ten years have been spent applying technologies and techniques from other application domains, identifying the specifics of ATC, and developing solutions when necessary.

France has such a plan, which is reaching its final development stage, and is aimed at replacing the current air traffic control workstations within the next few years. This plan, called PHIDIAS, makes heavy use of classical interaction techniques, and most of its original features are to be found in its hardware or low-level software components. PHIDIAS provides a standard WIMP (windows, icon, mouse, pointer) interface on two screens, one of which is very large. A lot of work has been spent on identifying and solving the software issues raised by such a large screen, and especially the management of many graphical objects moving independently. If those technical issues have now been solved, a number of design choices made in PHIDIAS can still be improved, if only because hardware, interaction techniques, and our knowledge of ATC have improved over time. This article describes the design of GRIGRI[1], the first prototype in project IMAGINE, which is a step toward the future generation of interfaces for air traffic control. GRIGRI implements design choices very remote from current designs, and finds its inspiration in pen computing. Contrasting with the notepads that made pen computing popular a few years ago, GRIGRI does not use handwriting recognition, but only the recognition of simple gestures, drawn with a pen or a fingertip. Those gestures (marks) are performed directly on the surface of the screen, thanks to a high resolution touch-screen. GRIGRI also uses direct manipulation for some functions and makes use of sam-

[1]*grigri* is a colloquial French word for *scribble*

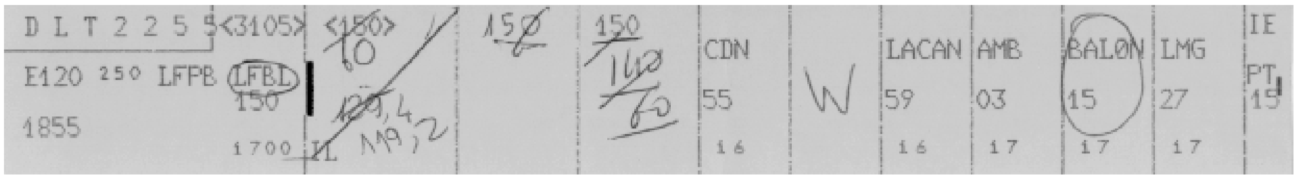


Figure 1: An annotated flight strip

pled sounds to improve the confidence of users and to make some interaction modes more explicit. Finally, the layout of GRIGRI was designed to encourage two-handed interaction.

This article is organized as follows. The first section is devoted to a description of the ATC task and the current environment of controllers. The second section identifies several drawbacks in the use of classical user interface techniques. The third section describes related work on gesture recognition and pen based computing. The following sections describe the design choices implemented in GRIGRI. The last section reports on preliminary evaluations of the system.

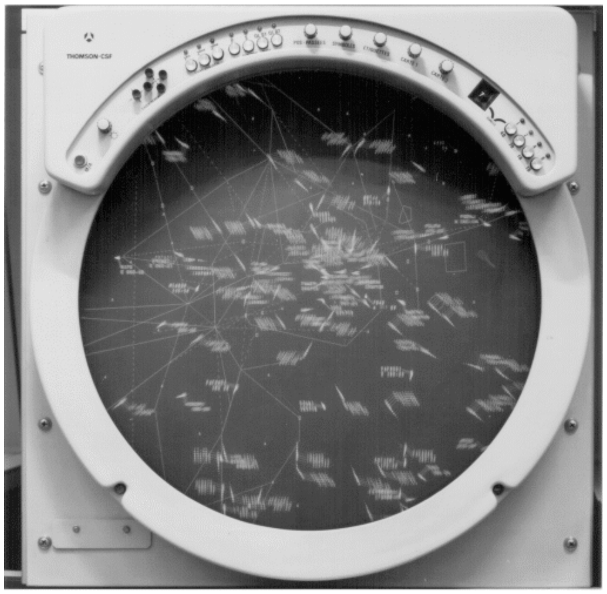


Figure 2: The current radar screen

THE AIR TRAFFIC CONTROL TASK

Air traffic control is a domain that encompasses a number of different tasks. We will focus here on the basic task of en-route air traffic controllers. These controllers mainly guide and monitor aircraft that follow airways, at a reasonable distance from airports. Airspace is usually divided in a number of sectors, each of which is managed by a team of two or three air traffic controllers. The most popular task is performed by so-called radar controllers, who monitor aircraft through radar screens and give vocal orders to pilots through radio links.

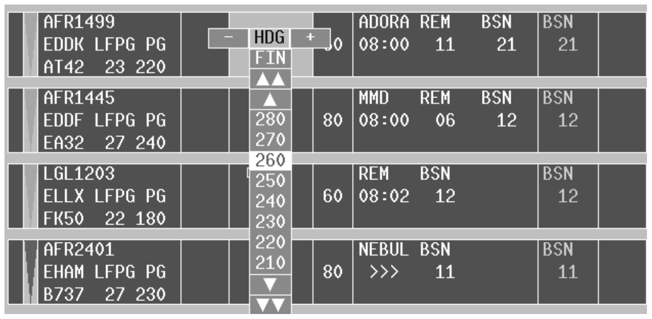
Today's radar screens are usually circular and monochrome. They provide a top view of airspace, and represent each aircraft with an icon showing its current position, smaller icons showing a few of its past positions, a line representing its speed, and a label containing its call-sign, speed, and altitude (see figure 2). This display enables radar controllers to identify problems, solve them in real time, and monitor the implementation of the solution. In this process, they also benefit from data about flight plans, that are printed on semi-rigid paper strips, called flight strips (see figure 1).

The flight strips are printed by the computing systems a few minutes before the estimated entry of an aircraft in the sector. The planning controller, who assists the radar controller by planning his work and managing the communications with other sectors, grabs the strips as soon as they are printed. Each strip is then inserted in the set of current strips, and the planning controller signals possible problems through voice, gestures, or layout conventions. The strips, which feature the call-signs of aircraft, the altitude requested by their pilots, and the planned route, are then used by the radar controller, mainly as a reminder. Strips are gathered in groups according to the current situation, and annotated with the orders given to pilots, as shown on figure 1. Most annotations are codified and documented in manuals. Finally, when an aircraft leaves for another sector, the corresponding strip is thrown away, as a sign that the radar controller may forget it.

LIMITS OF TRADITIONAL INTERACTION

When the plan for modernizing the French ATC workstations was launched in 1986, one of the goals, as in other countries, was to take advantage of new technologies in order to alleviate the workload of air traffic controllers. This goal had immediate consequences, such as the use of color screens to code information useful to radar controllers. It also had indirect but major consequences. A number of ATC experts placed hopes on intelligent tools that would help radar controllers solve problems. But such tools need data that are not available to the ATC system in its current state. For instance, the system does not know which controller is in charge of a given aircraft, because transfers are performed through procedures based on voice. Consequently, the idea of helping controllers quickly turned into the need for making them provide more data to the system. This is why they were interested in turning paper strips into a graphical interface: the information previously contained in annotations would then be known to the system.

These goals, as well as the set of available techniques, have had a strong influence on the design that was chosen for the new workstations. The screen gained even more importance and its size became critical. In addition to the radar image which covers a 50 cm wide round screen in the current system, the screen also has to display the new "electronic strips", as well as the potential intelligent tools. Air traffic controllers, afraid of missing important information that would be hidden, are generally opposed to overlapping windows. It was thus decided to use screens as large as possible, namely 50 cm wide square screens, featuring 2048x2048 pixels, developed by Sony for the US FAA. But those screens are so large that users must be seated far from them if they want to see the whole contents. This immediately excluded any possibility of direct interaction on the screen, and led designers to using a mouse and classical interaction techniques: menus, buttons, double clicks, etc. Studies were then focused on information coding (colors, icons, etc) and on the software techniques needed to efficiently display and move many objects on a large screen.



AFR1499 EDDK LFPG PG AT42 23 220	0	ADORA REM BSN 08:00 11 21	BSN 21
AFR1445 EDDF LFPG PG EA32 27 240	80	MMD REM BSN 08:00 06 12	BSN 12
LGL1203 ELLX LFPG PG FK50 22 180	60	REM BSN 08:02 12	BSN 12
AFR2401 EHAM LFPG PG B737 27 230	80	NEBUL BSN >>> 11	BSN 11

Figure 3: Manipulating a flight strip through menus.

On this screen, the main interactive areas are the fields of flight strips and aircraft labels on the radar image. Users may manipulate them to store a new heading or altitude or select an aircraft and display its route. All these manipulations require the designation of an aircraft or a strip, and the input of a command and its other parameters. Several commands are possible for nearly every active field (label, field of a label, part of a strip). Among the classical interaction techniques, menus seemed to be the most appropriate. But in order to minimize the number of items without having to use cascading menus, it was decided to associate a different menu with every field. For instance, figure 3 shows the menu associated to the "heading" field of a strip.

The system that we just described is still being developed. However, several design choices have been criticized by future users. For instance, the manipulations on electronic strips are considered slow and painful. It has been observed that with today's system, a controller writes an annotation on a strip every 17 seconds [6]. With such usage frequencies, manipulation times and the degree of attention needed are very important. But manipulating menus is relatively slow, and users have to pay attention to what they are doing. Even worse, before selecting an item in a menu, a controller has to click in a zone which is sometimes very small. This increases manipulation times according to Fitt's law. In the case of flight strips, the decrease in performance is noticeable, compared to manual manipulations that are often performed in parallel with other tasks, without paying much attention. With no simple solution to such problems, a number of users ask for the status quo, and want to keep paper strips until the time when alternative tools will make them really obsolete.

Even if electronic strips are discarded, manipulation times will still be an issue for radar images, and for every tool provided by the ATC system. This leads us to exploring possible solutions to that issue, by exploring new interaction techniques. Gesture recognition and mark-based input through a digitizing tablet or a touch screen look like a promising direction. Other techniques are also explored, among which are speech recognition and computer vision. But gesture recognition seemed mature enough to be proposed to air traffic controllers for a medium-term implementation.

RELATED WORK

Gesture recognition deals with the movements performed with one's hand, or any part of the body, with or without an instrument (pen, glove, etc). The input parameters are the successive positions of the tip of the pen, those of fingertips, or the successive angles of articulations, for instance. The devices used are pointing devices (mouse, digitizing tablet, touch-screen), digital gloves and position locators, or video cameras. Some of the techniques developed for 2D gestures performed with a pen can be transposed to more complex situations like 3D gestures performed with a glove [1]. Among those techniques, the most popular is Rubine's algorithm [12], which incrementally computes geometrical features of gestures, and uses statistical methods.

2D gestures, or marks, have been used in several ways in graphical interfaces. In addition to obvious applications to text input, gestures can be used to issue commands to the system. Kurtenbach has studied the issues raised by mark-based input, on the user's as well as on the system's side [9, 8]. Pie menus [3], although apparently close to traditional menus, are close to mark-based input, in that they use the orientation of gestures performed by the user. T-Cube even chains pie menus, thus associating each command to a broken line [13]. In T-Cube, menus are only displayed if the user hesitates while issuing a command. Unistrokes [7] fit between handwriting recognition and mark-based input. It models each letter with a simplified letter composed of only one stroke, in the same way as shorthand.

Pen computers, or notepads, which use the techniques mentioned above, have been popular at the beginning of the decade [10]. One of the first widely available environments was GO Corp's PenPoint. Apple's Newton uses handwriting recognition and mark-based input, combined with high quality feedback. But handwriting recognition accuracy often determines how such systems are accepted by users [5], and it is still difficult to achieve a very high accuracy, especially in countries where cursive writing is dominant, like France. Graffiti, a commercial variation of Unistrokes by Palm Computing, improves the accuracy and thus the usability of such systems at the cost of learning a simplified alphabet.

Finally, it is interesting to note that French air traffic controllers have been familiar with touch-screens for decades. The Digitatron, introduced in the 60s, is a touch sensitive alphanumeric screen with a low resolution. It gives access to a videotext system, used for infrequent operations such as checking which military zones are activated or modifying a flight plan.

A FIRST PROTOTYPE

As mentioned earlier, the management of flight strips is a good candidate for mark-based input. The annotations written on paper strips are made of numbers, and of simple and codified marks. Moreover, annotations are frequent, and the expected gain in time provides a good motivation. This led us to build a demonstrator in order to check whether gesture recognition was useful in this context. The proposed interface was based on the proposed format for electronic flight strips, itself very close to real flight strips. The recognized marks, made with a pen on a digitizing tablet, were of two types. First, it was possible to sort, group and shift strips, just as it is done with paper strips. Then, the usual annotations performed on paper strips were mimicked: highlighting of values by underlining them, modification of values by striking them off and writing the new ones, input of direct routes by drawing arrows between beacon names.

This prototype of flight strip management was soon enriched with a simplified radar image manipulated with gestures, so as to check how well the technique could be applied to interfaces where the use of gestures was not as obvious as for flight strips. This radar image was built with no button or menu, and every command was performed through gestures. There were gestures for zooming and panning, and for several operations on aircraft and beacons.

The results of this pilot study were very encouraging, as well as full of lessons. The lessons were related to the handling of errors, the acceptance of the system by users, and the type of applications that could easily take advantage of gesture recognition.

Errors

Error rates were a major concern, because high error rates would have meant that the technology could not be applied to air traffic control in its current state: errors make interaction slow and painful when they are noticed, and are dangerous when unnoticed. Although we did not carry serious evaluations at that stage of the project, the results were comforting. Our worst concern was interpretation errors: one gesture recognized as another gesture. But such errors were rare. More frequent were interpretation failures: gestures that could not be related to any gesture class. Those failures were especially numerous during the first contacts with the system. The absence of adequate feedback could induce users into errors and decrease their confidence in the system.

Acceptance

During this pilot study, the way the prototype was accepted by users was important: previous experience showed that full studies are useless if air traffic controllers are not confident about the system's potential usefulness. Comfortingly, the acceptance of the prototype by the first air traffic controllers who tried it was good. They found most gestures natural and fast enough, even when they had no equivalent in the current ATC system. The gestures proposed to interact with the radar image were especially well accepted, mainly because of the low precision that is required to perform operations. However, the physical setup, composed of a digitizing tablet and a normal workstation screen, was frustrating to users. They would have preferred a more direct interaction with the contents of the screen.

Potential applications

The use of gestures to interact with the radar image and the contents of flight strips was appreciated. However, the interface proposed for manipulating the strips themselves was the cause of problems. Users, who took pleasure in being able to use their pen again to annotate strips, were disappointed to be unable to move them with their bare fingers. Moving strips with a pen was far less natural. Moreover, the gestures for moving strips often caused confusion with gestures for annotating them. For instance, downward moves were interpreted as downward arrows when made close to a number representing a flight level. Consequently, we decided to discard that interaction technique.

We used the lessons learned with this first prototype when developing the prototype described in the rest of this article. First, we investigated the set of available hardware allowing direct interaction on the screen, either by projecting images on a digitizing tablet (like Rank Xerox EuroPARC's Digital Desk [14]), or by using touch-screens. Second, finding no simple solution for moving flight strips around, we decided to concentrate on radar images, leaving flight strip management for future research. Finally, we paid more attention to feedback during interaction.

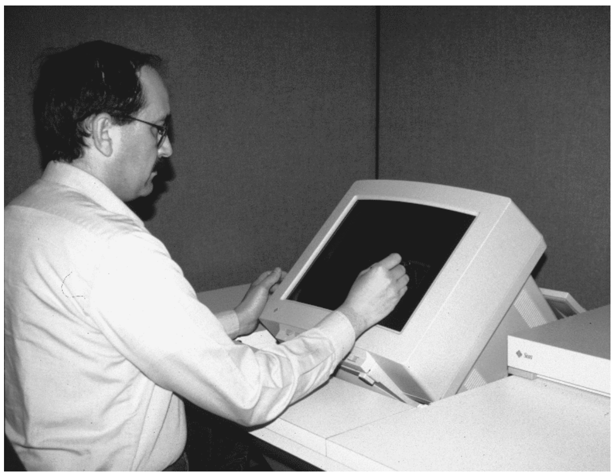


Figure 4: The physical setup of GRIGRI

THE PROTOTYPE GRIGRI

Using our first results, we developed a second version of the system, devoted to more systematic evaluations with air traffic controllers. This version, described below, offers a more complete interface and set of functions than the first prototype, so as to allow more realistic tests.

The physical setup of our prototype uses a touch-screen. It includes a color screen built in the surface of a desk with an angle of approximately 30 degrees (see figure 4). The screen is covered with a high resolution touch sensitive layer, that can be used with a pen or a fingertip. It is used to display a radar image and a bar of controls, as shown on figure 5. Depending on their type, commands are issued through the bar of controls, through gestures on the radar image, or through longer direct manipulation interactions.

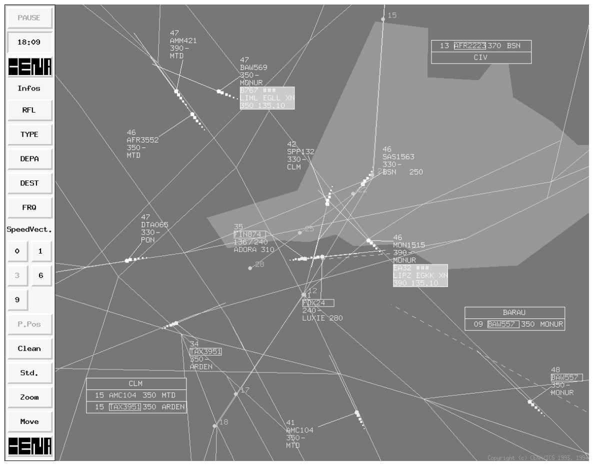


Figure 5: The screen of GRIGRI

Contents of the screen

For the reasons explained earlier, GRIGRI only features a radar image, and no flight strips. Our purpose is that evaluations yield results about mark-based input only, without introducing perturbations due to design choices in strip management. For similar reasons, the proposed radar image is not much different from radar images offered by PHIDIAS and its earlier prototypes. Controllers are familiar with their design, and we did not want to add novelties that would have changed their perception of the system. The radar image is thus composed of a plain background on which appear airways and restricted zones, represented by lines and polygons of a different color. Aircraft nearly have the same representation as on today's radar screen: icons for the current and past positions, a segment for speed. Labels are similar to those of PHIDIAS: they contain several text fields displaying the call sign of the aircraft, its speed, its flight level, etc. The label can be further enriched on demand, by displaying data from the flight plan of the aircraft: destination airport, route, etc. Users can also obtain graphical representations of routes on the display. Finally, the bar of controls on the side of the screen provides a number of global commands.

Global commands

Most commands offered by air traffic control interfaces are related to a given aircraft. Only a few commands are global to the radar image, and make it possible to change parameters of the display: filters to display only certain categories of flights, zoom factor, panning, length of speed vectors (expressed in minutes of flight), etc. Though we plan to return to real hardware buttons and knobs for a number of those parameters, we offered the corresponding controls in the form of software buttons, grouped in a bar on the side of the screen. The size of those buttons is large enough to allow fast manipulations with fingertips, without having to use a pen. We laid them on the side of the screen in order to study whether users would be tempted to use their non-dominant hand for pressing them. With this design, we hoped to observe a specialization of hands, the non-dominant one dealing with secondary tasks, and the dominant one being dedicated to fine manipulations on aircraft representations.

Mark-based commands

Most commands concerning a given aircraft are simple operations, for which one needs to specify the aircraft, the operation to be performed, and one or two additional parameters (field to be highlighted or modified, type of warning, etc.). When those parameters can have many values, special interactions will be necessary. But in other cases, the number of possibilities is low enough to code each combination of operation and parameters by a mark. For instance, controllers highlight the flight level or the heading of an aircraft to register the fact that the valued was confirmed to the pilot. This highlighting operation can be designed in two ways. It is possible to go for a unique, generic command, by assigning a specific behaviour to every field representing a value. It is the choice made in PHIDIAS: users click on the field they want to highlight, thus making a menu appear, and choose the appropriate item in the menu.

With mark-based input, another choice can be made: do not specialize fields, but have several marks for the same operation, and have each mark code for a field. For instance, instead of having a mark for the command “modify a field”, that can be applied to any field, let us have a mark for “modify speed”, a mark for “modify flight level”, and so on, with the meaning of a mark being independent from the location where it is issued. This design choice yields a growth of the number of commands that must be learnt (about 15 different marks). But it dramatically lowers the constraints imposed on users, because gestures can be started anywhere on the label, and their sizes may vary much more. This choice allows us to expect a better efficiency in interaction, at the expense of some training, which is acceptable for professional users such as air traffic controllers.

However, even if we were confident that specific gestures were a good solution, we also implemented generic commands as an alternative way of performing operations. Those generic commands (modify, highlight) are interpreted according to the field on which they are performed. Doing this, we allowed users to choose the interaction style they preferred, thus getting some hints about whether our reasoning was right. The set of all possible gestures is shown in figure 6.

To recognize gestures, GRIGRI uses Rubine's classifying algorithm. This algorithm can classify gestures composed of a single stroke, and involves an individual training period. For each class of gestures, each user provides about fifteen examples, using a tool described later in this article. During this training, the geometrical features of each gesture are extracted, and are used to produce a dozen features that are significant of the class. Later, when using the system, the same features are extracted from every gesture drawn, and compared to those of the defined classes. The classifying

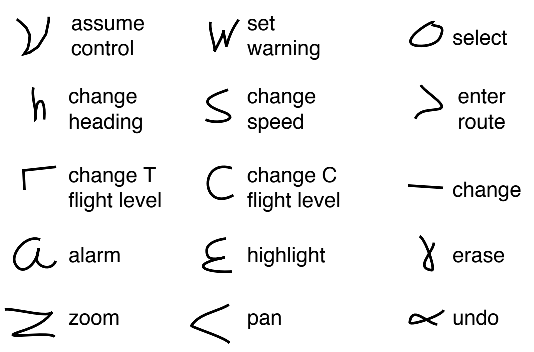


Figure 6: The set of gestures recognized.

algorithm identifies the class whose features are closest, or yields a recognition failure if the confidence ratio is too low. Classification is very fast (a few milliseconds), and response times are not perceptible by users. Having noticed that recognition failures had to be quickly and clearly signaled to users, we had to devise appropriate feedback. Considering that the visual channel was already very busy, we added sound capacities to GRIGRI. Two sampled sounds are used: one signals recognition failures, and the other signals gestures that are correct but meaningless at the place where they are performed. However, we did not use sound for successful operations yielding an immediate visual feedback, so as to restrict the use of sound to situations where it is necessary.

Long interactions

Some commands of GRIGRI involve the input of numerical or geometrical data, and cannot be implemented as a simple gesture or a button click. In the current version of GRIGRI, there are four of those "long interactions": heading input, warning input, flight level or speed input, and zooming. Heading and alarm input are triggered by a gesture on the label of an aircraft. Then begins an input sequence using classical techniques of direct manipulation: rubber band for headings, and palette for alarms. Input of flight levels and zoom are more complex.

To enter a flight level, one needs to specify an aircraft, the operation itself, and the new value. These data are entered as follows: Like other operations, the command begins with a gesture on the label of the appropriate aircraft. Then GRIGRI opens a window that allows the user to enter the new flight level. In many regions of airspace, only a few values are acceptable (330, 350, 370, 390, for instance), and PHIDIAS proposes to enter the new value through a menu, allowing itself to propose a default value. However, a number of air traffic controllers have to deal with many more possible flight levels, which makes this solution less acceptable. What we implemented in GRIGRI is the recognition of hand written numbers, in order to compare the efficiency of the two techniques. The input window has its own gesture classifier, independent from the one which is used in the radar image. It recognizes the ten digits (drawn with a single stroke) as well as gestures for correction and validation. The input window is fairly large, so as to let users write at their preferred size. In order to avoid masking a large portion of the radar image for tool long, this window is semi-transparent, after Xerox's see-through tools [2] (see figure 7).

Changing the zoom factor involves a more classical type of interaction. The user first needs to go into zoom mode; this is done with a gesture, or by pressing a button in the bar of controls on the side of the screen. Then, the image is stretched or condensed by moving the pen outward or inward; the system goes back to normal mode when the user lifts the pen again. The two ways of starting the operation have been implemented because we hoped to identify whether users preferred to use a button or a gesture. We also hoped to induce users into using forms of two-handed interaction: the bar of controls is located on the side of the non-dominant hand, while the pen is usually held in the dominant hand. This is a situation where using two hands may be useful: the non-dominant hand rests on the border of the screen and presses buttons on demand, while the other hand stays where the attention of the user is focused, and keeps holding the pen. Even though it is impossible to perform parallel actions with current touch-screens (which might be a problem, as described in [4]), we wanted to assess whether a purely sequential operation was possible.

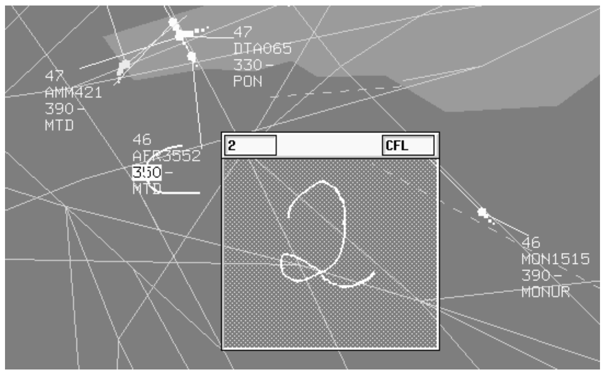


Figure 7: Entering a flight level

Finally, conscious that long interactions place the system in modes that should be made explicit to the user, we decided to use sounds. When the user starts a long interaction, or when the action is terminated, the system produces a fairly long sound of the kind which is usually associated to transitions, teleportations in science fiction movies, for instance. Though we have not yet planned to test the usefulness of such sounds, we believe they have a significant role in the perception users have of the results of their actions. However, air traffic controllers are still very cautious about sound, and this will need to be studied more closely.

Training

The algorithm we use for gesture recognition has a learning phase. Training has to be done for each user, even when the gesture set is fixed, which is the case for GRIGRI. This is partly because the dynamics of a gesture may vary, even if the result looks the same, and partly because complex gestures, such as digits, vary a lot. Although we could have used gesture editors that were already available, we decided to develop a new one, specific to our system, for several reasons. First, we wanted to make sure that the size and shape of gestures would not change between the learning phase and the evaluation phase, and we thought that the context, and especially the size of aircraft and labels, had a role. We also knew that the learning phase would be the first exposure of our users to GRIGRI, and we wanted to use it as a training period for them as well. Finally, we had noticed during the earlier experiments that gestures would vary when drawn near the edges of screen. Taking this into account, our training system makes users draw gestures in different locations of the screen, thus feeding the algorithm with all variations of gestures.

IMPLEMENTATION

GRIGRI was implemented on a Sun SPARCstation, with a software and hardware environment adapted to our needs. Only the screen of the workstation is accessible to users: the mouse was removed, and the keyboard is used by observers during tests. The screen is the standard screen of Sun workstations, attached to a custom made table. The input device is a resistive layer held between the screen and its frame, with a resolution of about 200 dots per inch.

Our system uses the X Input extension to the X Window System distributed by MIT, which had to be modified to suit our needs. With these extensions installed, the touch-screen replaces the mouse for every purpose. Gesture recognition is based on Rubine's algorithm, in its C version. The software layers that feed positions to it had to be modified to account for the slight unevenness of the screen that caused the pen to bounce. Rebounds were interpreted as the end of a mark and the start of a new one, which led to many interpretation failures. We introduced a time-based filter that removes most of those parasitic events.

GRIGRI was built using a hybrid prototyping environment developed at CENA. This environment is based on a public domain Scheme interpreter: time-intensive functions are programmed in C and exported as primitives of our environment, while most of the system is programmed in Scheme. This approach, similar to that of Tcl [11], combines the efficiency of C with the flexibility of an interpreted language, and proved very useful for prototyping. In GRIGRI, the most important primitives used are the gesture recognition algorithm and a graphical widget dedicated to animated interactive images.

EVALUATION

One of the main motivations in developing GRIGRI was to perform systematic evaluations with professional air traffic controllers in a semi-realistic environment. Those experiments, which are currently under way, have several phases. The first phase of tests relied on CENA staff as users. While not using the time of air traffic controllers, which is an expensive resource for us, those tests allowed us to identify and eliminate several design mistakes. They also helped us to design a training scheme for our new users. This first phase was sufficient to discover the drawbacks of our hardware setup. The worst problem was that the screen was too high above the working surface, thus provoking arm strain. This strain is aggravated by the fact that users cannot rest their wrists on the surface of the screen while interacting with the system, because that would confuse the touch-screen. Another problem is the thickness of the touch sensitive layer, which introduces parallax errors, that were identified as the cause of many out of context gestures: the user wants to highlight a field, but draws the corresponding mark on the background of the screen. Identifying this problem allowed us to adapt the measures that were taken during the next evaluations, and eventually led us to implementing a partial software correction. These defects did not prevent our first users from quickly getting used to the system. But this fast familiarization with the system also had a severe drawback: users immediately understood how the system worked, and soon wanted to use it like real pen and paper. This led them to increase their manipulation speed too much, and to produce many interpretation failures and errors, which was disappointing to us. We now prevent them from this disappointment by explaining the limits of the system at the first contact, before the first manipulation.

The second phase of the project, which is currently being carried out, is done with volunteer air traffic controllers. It is a first evaluation of how the system is accepted by controllers, as well as a means of getting data on how well the system works and how it is used. The obtained data will then be interpreted and used to improve the system, if it is considered promising. Other goals will then be set for the next evaluation phase, so as to understand better and better how this technology can be used in air traffic control systems. GRIGRI is evaluated by having it used in semi-realistic conditions by a radar controller. Simulated traffic is fed to the system and displayed on the radar image, while the controller is linked to pseudo-pilots through a simulated radio link. Measurements begin after one hour of simulation. What is observed is the way the radar controller interacts with the workstation. Some data is obtained by observing users and conducting interviews after every simulation. But the most objective data is obtained through the system itself, which is instrumented to output it. Among those data are the number of times every command is used, the frequencies of several types of errors, manipulation times, and even the trace of every gesture. Some errors which are difficult to detect automatically (errors on digits, for instance) are registered by an observer through the keyboard of the workstation.

This second phase involved 12 volunteer air traffic controllers. The data obtained during this phase is currently being interpreted, and only rough data are available. The first results, however, are very encouraging. The perception of the system by users is generally good, and they consider it as a potential alternative to classical interaction techniques, if it can be made reliable. The figures are encouraging as well. Out of a total number of 5200 marks drawn by the 12 users, we observed about 5% of gestures out of context, 5% of interpretation failures, and 1.3% of interpretation errors. Of course, these numbers will have to be interpreted, and the nature of interpretation errors and their impact will be studied carefully. But they prove that mark-based input is an option that can be considered further in the context of air traffic control. We already have indications that these figures can be improved easily. For the first six users, for instance, parallax errors accounted for half of the marks drawn out of context. Those parallax errors have then been partially corrected by applying a geometrical transformation to the coordinates provided by the touch-screen. They would even probably disappear with a more appropriate device, using overhead projection for instance. Similarly, half of interpretation failures have been identified as caused by technical problems that we should be able to solve (rebounds still poorly handled, or insufficiently sampled gestures). After devising and applying solutions, we will still have to determine whether the resulting error rates are satisfactory, or if they can be further improved by other means. However, even with the current 90% accuracy, our users accepted the system well, probably because of the low cost of mistakes: redoing a mark is fast enough.

Although we have gathered a lot of data that will help us to improve the system (for instance by taking into account the probability of each command according to the context), our experiment will not allow us to evaluate all the design choices made in GRIGRI. For example, we will have to determine whether users discarded generic gestures (such as the straight line associated to "modify") because they forgot about their existence, as they put it, or because of other reasons. Similarly, there were too few zoom actions (two) to gather any evidence about two-handed interaction. Finally, we now need to measure manipulation times with PHIDIAS in order to be able to compare them with the ones we obtained with GRIGRI.

Whereas some expected results are missing, we also made unexpected observations, that will have an influence on future research. For instance, we discovered that controllers liked the fact that two of them were able to share a screen, talk over it, and interact with it one after another without having to exchange mice. This opens new perspectives in the management of the collaboration between a radar controller and a planning controller. Such properties of the system will be studied in future phases of experimentation. Doing this, we hope to progress in our understanding of mark-based input while proposing a system that will be more and more usable.

CONCLUSION

We have described in this article the reasons why we considered mark-based input as an interaction technique for air traffic control. We have explained the choices made while designing an experimental workstation based on that technology, and we have described the most significant technical issues encountered. Finally, we have given first evaluation results that make us confident regarding the possible applications of pen computing to air traffic control. Nevertheless, as we explained in the introduction of this article, there are many obstacles when designing interactive systems for a domain such as air traffic control. The techniques that we are studying have only been through the first of those obstacles. Then, if they prove worthwhile, they will have to be integrated in a full size workstation. They will also have to be tested with air traffic controllers under stress; the fatigue induced by their use will have to be measured; their impact on the collaboration between a radar controller and a planning controller will have to be studied. A number of social issues will also probably appear. It takes all that to apply new technologies to our environment where efficiency and safety are so important.

ACKNOWLEDGMENTS

The first version of GRIGRI was developed by Gwenael Bothorel, using previous work by Thomas Baudel (LRI, Orsay). The current version was built with Jean-Luc Dubocq and Frédéric Lepied. Christophe Mertz supervises the evaluations that are being carried. Thomas Baudel and Wendy Mackay provided helpful comments and helped with English grammar.

REFERENCES

1. T. Baudel and M. Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Communications of the ACM*, pages 28–35, July 1993.

2. E. Bier, M. Stone, K. Fishkin, W. Buxton, and T. Baudel. A taxonomy of see-through tools. In *Proceedings of the ACM CHI*, pages 358–364. Addison-Wesley, 1994.

3. J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs linear menus. In *Proceedings of the ACM CHI*, pages 95–100, 1988.

4. S. Chatty. Extending a graphical toolkit for two-handed interaction. In *Proceedings of the ACM UIST*, pages 195–204. Addison-Wesley, Nov. 1994.

5. C. Frankish. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of the ACM CHI*, pages 503–510. Addison-Wesley, 1995.

6. C. Garoff-Mercier and M. Chollet. Analyse de l'activité des contrôleurs du trafic aérien : utilisation des vecteurs d'information et des communications. Technical Report R90-07, Centre d'Études de la Navigation Aérienne, 1990.

7. D. Goldberg and C. Richardson. Touch-typing with a stylus. In *Proceedings of the ACM CHI*, pages 80–87, 1993.

8. G. Kurtenbach and T. Baudel. Hypermarks: issuing commands by drawing marks in Hypercard. In *CHI'92 Posters and Short Talks*, page 64, 1992.

9. G. Kurtenbach and W. Buxton. Issues in combining marking and direct manipulation techniques. In *Proceedings of the ACM UIST*, pages 137–144, 1991.

10. A. Meyer. Pen computing. A technology overview and a vision. *SIGCHI Bulletin*, 27(3):46–90, 1980.

11. J. K. Ousterhout. *Tcl and the Tk toolkit*. Addison-Wesley, 1994.

12. D. H. Rubine. *The automatic recognition of gestures*. PhD thesis, Carnegie Mellon University, 1991.

13. D. Venolia and F. Neiberg. T-Cube: a fast, self-disclosing pen-based alphabet. In *Proceedings of the ACM CHI*, pages 265–270, 1994.

14. P. Wellner. Interacting with paper on the Digital Desk. *Communications of the ACM*, 36(7), July 1993.