Programming Education on an Electronic Whiteboard Using Pen Interfaces

Taro Ohara, Naoki Kato, Masaki Nakagawa
Dept. of Computer Science, Tokyo Univ. of Agri. & Tech.
Naka-cho 2-24-16, Koganei, Tokyo, 184-8588, Japan
e-mail: sylph@hands.ei.tuat.ac.jp

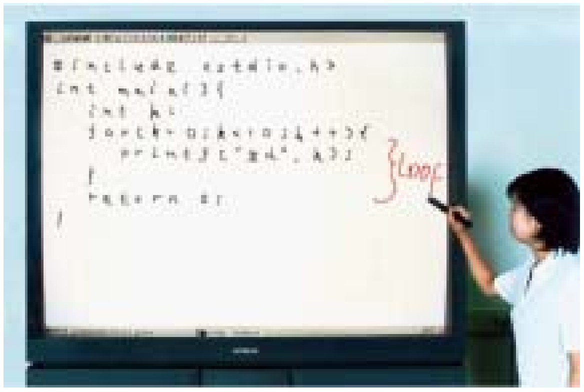
This paper describes a system to teach programming on an interactive electronic whiteboard that combines the merits of classroom lectures using a black/white board and those of computer processing. Using this system, a teacher can write a program on the board, explain it, make the system recognize it and run the program in front of the class while keeping the attention of the students focused on the board. The system allows input data to be entered by writing input parameters on the board.

1. Introduction

Conventionally, a teacher teaches programming by writing a program on a white/black board and explaining its logic and structure. Students copy the program and its explanation in their notebooks. Then, they are often required to study the program after the class by running it on a computer. Problems may arise at this stage, however, because the program written by the teacher on the board may have contained some errors, the student may make some mistakes in copying the program, and so on.

In order to solve this problem, we have proposed a system for an interactive electronic whiteboard [1]. Using this system, the teacher can verify the correctness of a program that he/she has written immediately, show its execution, and explain how the output is changed when some part of the program is modified, without losing the familiarity and advantages of lectures using chalk and blackboard (Fig. 1).

This paper describes the latest version of our programming education system on an electronic whiteboard system.



```
#include <stdio.h>
int main(){
   int k;
   for(k=0;k<10;k++){
      printf("%d",k);
   }
   return 0;
}
```

Fig.1 Electoronic whiteboard and programming education system

2. Design of the Programming Education System

2.1 Design Goals

The specifications of the programming education system are based on the design criteria of the user interface for the electronic whiteboard [2, 3] as follows:

(1) Operability from arbitrary standing position of the user.
(2) Easy operability with a single electronic marker.
(3) Natural extension of the desktop GUI.
(4) Simplicity of displayed contents.
(5) Maximize space for contents while minimizing space for control.

The first of the specifications allows the teacher to operate the board without having to cross the surface or stretch hands from side to side or from edge to edge. The second specification ensures that the teacher can operate the board with a single electronic marker without needing other markers, keyboard, mouse, etc. The third specification ensures consistency with the desktop environment. The fourth criterion allows the teacher to operate the board without confusion so that the students can understand the contents easily. The fifth criterion is to make sure that the surface of the whiteboard is utilized for education as much as possible: buttons, menus, etc. should not hide the contents unless absolutely necessary.

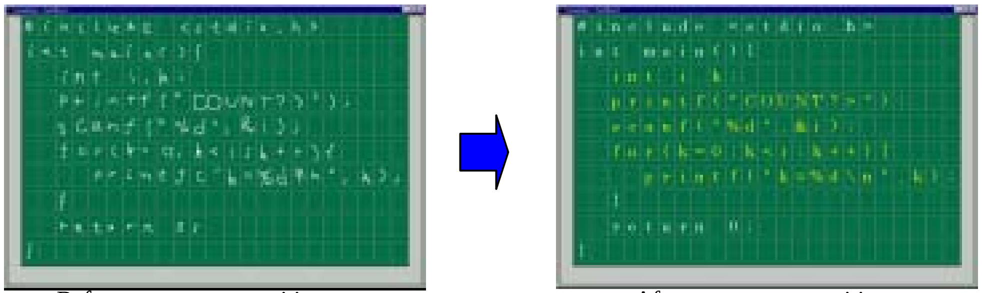
2.2 Functional Design

The following functions are necessary for the programming education system.

(1) Handwritten character recognition

With a marker, writing is the easiest and simplest way to input program text. Without pattern recognition, however, handwriting is just pen-trace patterns and cannot be processed as program text. Therefore, handwritten character recognition [4] is necessary.

To facilitate character recognition, we provide a grid of character input frames to write program text. The character recognition engine accepts the handwritten text, recognizes it within the context of the programming language and outputs a code sequence of program text. We employ a lazy recognition scheme i.e., recognition after all the text is written, because recognizing each character immediately after it is written interrupts the writing of a program. The lazy recognition approach also allows the use of context to help pattern recognition. After program recognition, handwritten character patterns are replaced by font patterns (Fig. 2).



```
#include <stdio.h>
int main(){
  int i,k;
  printf("COUNT?>");
  scanf("%d",&i);
  for(k=0;k<i;k++){
    printf("k=%d\n",k);
  }
  return 0;
}
```

Before program recognition　　　After program recognition

Fig.2　Handwritten program recognition

(2) Editing

Input, insertion and deletion of program text are the most necessary editing functions. According to the design specifications, the user must be able to perform these operations with a single marker. We implement the input function by allowing the user to write a character pattern in any empty frame. Insertion and deletion can be performed by tapping a marker between lines and dragging right or left (insertion or deletion of character frames within a line), or down or up (insertion or deletion of lines). When a marker is dragged to the right, new frames appear along with the dragging and the user can write characters in them. When the marker is dragged to the left, the characters on the right move over the dragged characters which are deleted (Fig. 3).

Insertion and deletion of lines can be done similarly by shifting lines downward and making new lines or shifting lines upward.

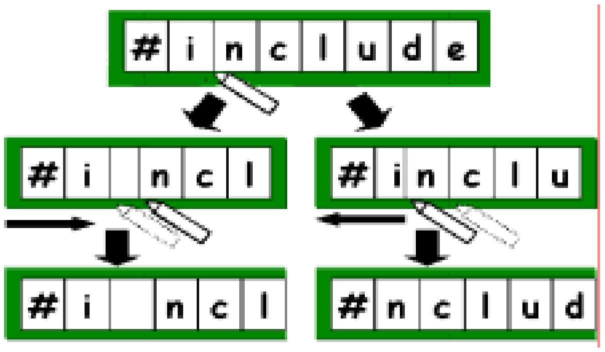


Fig.3　Insertion and deletion of text by sliding character frame

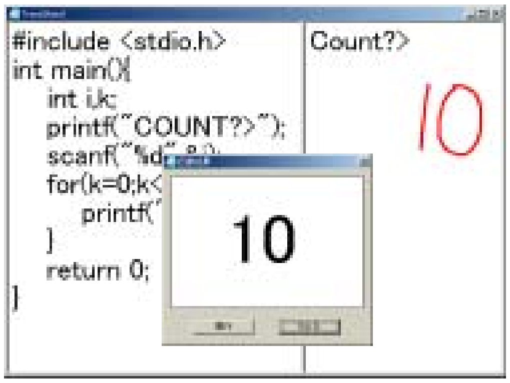
(3) Program execution

A program thus edited can be executed by an interpreter. Using an interpreter rather than a complier, the program execution can be started without the compilation step. The output of the program being interpreted is not displayed directly, but a pipe with the system is created and the output is displayed within the execution window of the system. The input is dealt similarly. Data can be inputted by writing in the execution window and the recognized code is sent to the program through the pipe.

We deliberately avoided the use of a keyboard for input in our system. A keyboard is neither easy to use for a standing teacher nor easy to observe for the students facing the teacher. When the teacher writes some input in the execution window, its recognition result is displayed in a pop-up window. The teacher can confirm the result of recognition, correct it if necessary, and then make the program continue its execution by pushing the execution button (Fig. 4).

(4) Handwriting annotations

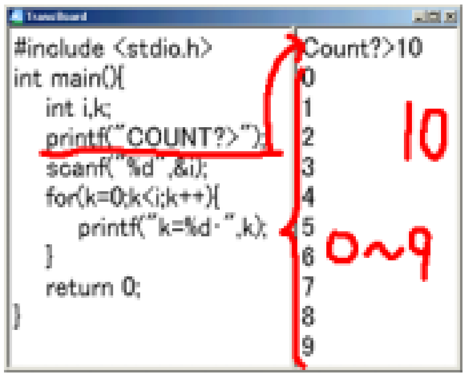
A great advantage of using a pen is that one can write almost anything freely. It is very useful for the teacher to be able to write annotations on the program that he/she is explaining instead of merely showing the program. Therefore, our system provides an annotation capability on the source and execution windows with the electronic marker in the same way as on a blackboard (Fig. 5).



```
#include <stdio.h>
int main(){
   int i,k;
   printf("COUNT?>");
   scanf("%d
   for(k=0;k<
      printf(
   }
   return 0;
}
```

```
Count?>
```

Fig.4 Execution of program



```
#include <stdio.h>
int main(){
   int i,k;
   printf("COUNT?>");
   scanf("%d",&i);
   for(k=0;k<i;k++){
      printf("k=%d ",k);
   }
   return 0;
}
```

```
Count?>10
0
1
2
3
4
5
6
7
8
9
```

Fig.5 Handwriting annotation

(5) Screen scroll

Since the electronic whiteboard has a limited size, and some programs may be too large to show within the display area, the scroll function is essential. We will describe this in detail in the next section, as it is an issue of user interface.

The standard scroll bar for a window is displayed on the bottom right-hand side of the window. This is convenient for operating in the traditional desktop environment, but on an interactive electronic whiteboard with an electronic marker, it is hard to use. The teacher has to stretch his or her hands from side to side and hide the board by his or her body. This violates one of our basic design goals.

Therefore, scrolling the screen is implemented without using the traditional scroll bar. Scroll area is located around the input area. By touching the marker on any place in this area and dragging it to arbitrary direction, the screen can be scrolled in that direction.

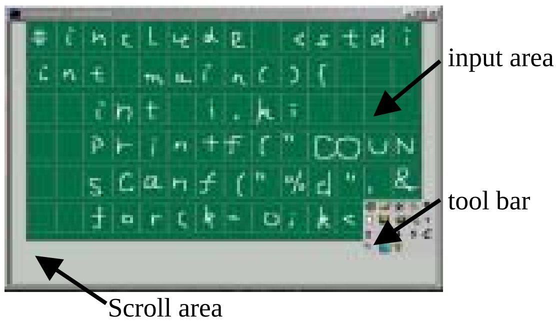
2.3 Design of the User Interface

Design of the user interface is also based on the specifications mentioned earlier in 2.1.

(1) Screen interface

The screen of the programming education system consists of a handwriting input area and a screen scroll area (Fig. 6). The scroll area is located around the input area except for the upper part of the window. When the user taps with the marker somewhere in the scroll area, a tool bar is displayed and operations can be performed from there.

A user can write a program in the handwriting input area.



```
#include <stdi
int main(){
    int i,k;
    printf("COUN
    scanf("%d",&
    for(k=0;k<
```

Fig.6 Screen layout of the programming education system

(2) Tool bar

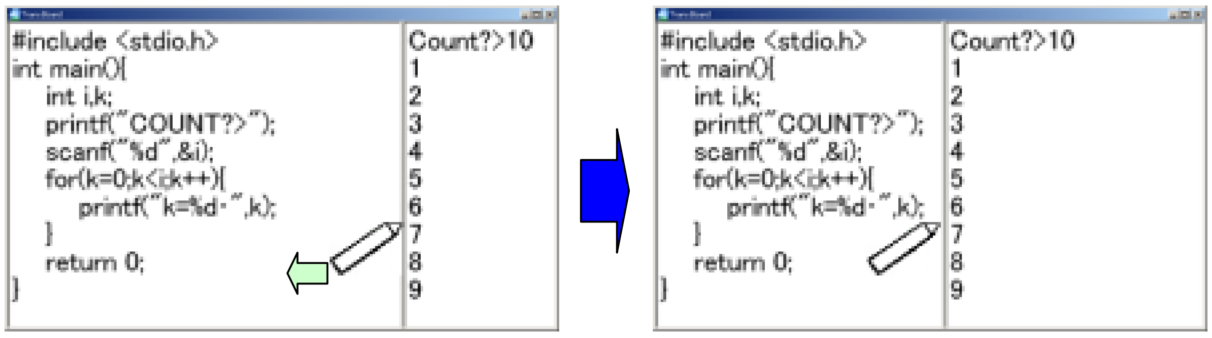
Since operations in the tool bar are not so frequently performed, the tool bar is displayed only when necessary. When the user taps somewhere in the scroll area with the electronic marker, the tool bar appears there. Moreover, only the buttons used often are displayed, others are hidden. When the functions associated with the hidden buttons are required, the number of buttons displayed can be changed by dragging the scroll area in the tool bar with the marker.

(3) Execution window

When the teacher is explaining a program, it is useful if he or she can show the source program and its execution results, and annotate them by writing as shown in Fig.5.

The execution window displays the source program and its execution result within split areas of the window and allows the user to annotate over both the areas. The user can choose either or both areas to be displayed, and can change their proportions of the window as shown in Fig. 7.

Another alternative is to display the result of program execution in a window separate from the source program, but to allow the user to annotate over two or more windows requires restructuring of our system software for this system.



```
#include <stdio.h>
int main(){
    int i,k;
    printf("COUNT?>");
    scanf("%d",&i);
    for(k=0;k<i;k++){
        printf("k=%d-",k);
    }
    return 0;
}
```

```
Count?>10
1
2
3
4
5
6
7
8
9
```

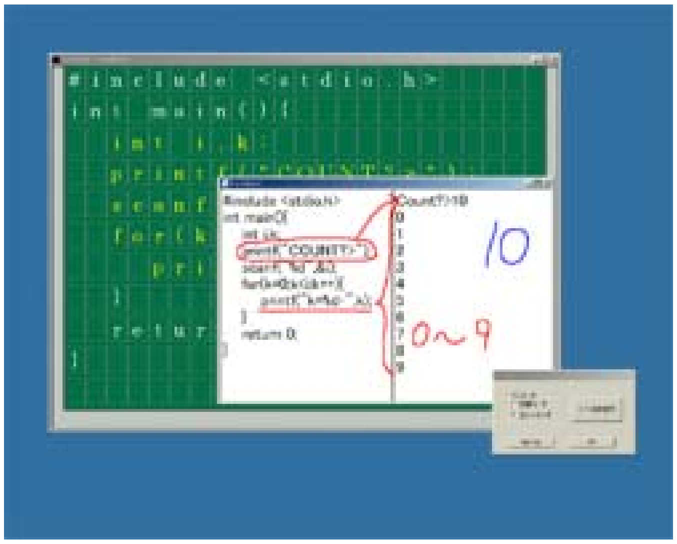
Fig.7 The splitter window displaying the source program and result

(4) Tool bar for the execution window

The tool bar containing buttons for operations on the execution window appears whenever the execution window is displayed because it occupies only a small area and these operations are used often when the teacher is working on the execution window.

3. Implementation of the Programming Education System

We have implemented the programming education system on the MS-windows ME using Visual C++ 6.0. Its appearance is shown in Fig. 8.



```
#include <stdio.h>
int main(){
    int i,k;
    printf("COUNT?>");
    scanf("%d",&i);
    for(k=0;k<i;k++){
        printf("k=%d ",k);
    }
    return 0;
}
```

```
Count?>10
0
1
2
3
4
5
6
7
8
9
```

Fig.8 Appearance of the Programming Education System

4. Conclusion

This paper described the design and implementation of our programming education system on an electronic whiteboard system. We are now planning to use the system for teaching a programming course in our university curriculum to evaluate it in actual use.

References

[1] J. Kanda, S. Sawada and M. Nakagawa: “Programming Education System on an Interactive Electronic Whiteboard,” Proceedings of the Fourteenth Symposium on Human Interface, Tokyo, pp.601-606 (1998.9)

[2] M. Nakagawa, T. Oguni and T. Yoshino: “Human Interface and Applications on IdeaBoard,” Proc. IFIP TC13 Int’l Conf. on Human-Computer Interaction, pp.501-508 (1997.7).

[3] M. Nakagawa, K. Hotta, H. Bandou, T. Oguni, N. Kato and S. Sawada: “A Revised Human Interface and Educational Applications on IdeaBoard,” CHI99 Video Proceedings and Video Program and also CHI99 Extended Abstracts pp.15-16 (1999.5).

[4] T. Fukushima and M. Nakagawa: ”On-line Writing-box-free Recognition of Handwritten Japanese Text Considering Character Size Variations”, Proc. of 15th ICPR, Vol.2, pp.359-363 (2000.9).