Teaching Computer Programming in Elementary Schools: A Pilot Study

Janet Mei-Chuen Lin, Long-Yuen Yen, Mei-Ching Yang, and Chiao-Fun Chen
Department of Information and Computer Education
National Taiwan Normal University
Taipei, Taiwan
mjlin@ice.ntnu.edu.tw

Abstract. In this study eighty-one ten- to twelve-year-old students learned to use Stagecast Creator, HANDS, and Visual Basic to create simple games and animations. Data were gathered and analyzed to understand students' attitudes toward learning the three programming packages, the difficulties they encountered in writing programs and getting programs to work, and how parents felt about their children's programming learning experience. Results of this study reveal that children can learn and they enjoy learning computer programming. Most parents also support teaching of computer programming in elementary schools.

Keywords: Computer Programming, IT Education, Elementary Schools

1. Introduction

It is stated in the National Educational Technology Standards (NETS) for students (ISTE, 1998) that six broad categories of technology foundation standards are to be introduced, reinforced, and mastered by Pre-K -12 students. The six categories are: (1) basic operations and concepts, (2) social, ethical, and human issues, (3) technology productivity tools, (4) technology communications tools, (5) technology research tools, and (6) technology problem-solving and decision-making tools. The ACM Model Curriculum for K-12 Computer Science (ACM, 2003) augments the NETS-S model by emphasizing the components of algorithmic thinking and other fundamental elements of computer science. In the four-level ACM model it is specified that *Level 1* (recommended for grades K-8) should provide elementary school students with foundational concepts in computer science by integrating basic skills in technology with simple ideas about *algorithmic thinking*. More specifically, "Children learn about algorithmic problem solving whenever they discover a collection of steps that can be carried out to accomplish a task. These steps should accommodate unusual contingencies (using conditional, or "if" statements) and repetitions (using loops, or "while" statements) (ACM, 2003, p.12).

National Research Council's Committee on Information Technology Literacy supports this view (National Research Council, 1999). The committee articulates the role of algorithmic thinking and programming as an essential element of understanding information technology. It is pointed out that one can customize, extend, and enhance prepackaged applications using basic programming concepts. Besides, some of the capabilities that are at the heart of fluency with information technology

rely explicitly or implicitly on programming knowledge. Programming concepts can even be applied to non-information technology problems.

Despite the increasingly recognized importance of programming instruction at pre-college levels, it is rarely integrated into K-9 computer curriculum in Taiwan or other countries. Students in Taiwan's elementary and junior high schools typically learn Window applications such as Word, PowerPoint, FrontPage, Flash and PhotoImpact. Although a small number of schools have been teaching Lego Mindstorms in their programming clubs or summer camps in recent years, very few students have had the chance of learning it.

We believe that computer programming should be taught more often in elementary schools. This paper reports the experiments that we conducted during the summer of 2004, in which we taught 81 students to program using three programming packages. In the remainder of this paper Section 2 is a description of the instructional materials used in the experiments and how we collected research data; in Section 3 we present results of the experiments; and Section 4 is the concluding remarks.

2. The Experiments

During the summer of 2004 we conducted experiments at three elementary schools. A total of 81 students participated voluntarily. At each school the experiment lasted two weeks, three hours a day and five days a week. First we introduced students to Stagecast Creator for nine hours, then HANDS for six hours, and Visual Basic for the remaining fifteen hours. With each programming package students learned how to create simple games and animations. One author of this paper served as the instructor and the remaining three as teaching assistants and observers. Students worked individually at their own computers most of the time, but they were occasionally teamed up to do larger projects. In the following we describe in detail subjects of the experiments, the instructional materials used, and how we collected research data.

2.1 The Subjects

The 81 students who participated in this study were from three schools, including 33 from School A, 19 from School B, and 22 from School C. All three schools are located in the metropolitan districts of Taipei. At the time of the experiments, nineteen (23%) of the subjects were about to enter the fifth grade, 58 (72%) to enter the sixth grade, and the rest (4 students, or 5%) the seventh grade. Fifty-four (67%) of the subjects were male and 27 (33%) were female.

Regarding the participating students' prior computer knowledge, most of them were familiar with word processing applications (94%), image editing applications (86%), presentation software (78%), and web page design tools (73%). Some of them were familiar with spreadsheet software (31%) and animation design tools (40%). Only three students had learned computer programming before, in Java Script, Logo, and Lego Mindstorms respectively.

2.2 The Instructional Materials

The programming packages selected for use in our experiments are Stagecast Creator, HANDS, and Visual Basic. They represent three different techniques of teaching computer what to do, ranging from one that does not require any code to be

typed in, to second one that requires programmers to type in English-like statements, and to third one with which programs need to be coded in rigid syntax.

Stagecast Creator, from Stagecast Software, Inc. (http://www.stagecast.com), is a languageless programming system for constructing simulations and games. It eliminates the need for any syntactic language during program construction in favor of a combination of two technologies: programming by demonstration (PBD) and visual before-after rules (Smith, Cypher, & Tesler, 2000). Programming with Stagecast Creator involves creating characters and associating with each character a set of visual rules. These rules allow the user to demonstrate to the computer how a character is supposed to perform actions in different situations. The computer records the user's actions and can then reexecute them later on different inputs. Figure 1(a) shows a *Sim* created with Stagecast Creator. It features the Buddha, Monkey King and Pigsy in the famous ancient Chinese story *Journey to the West*. Figure 1(b) shows five visual before-after rules associated with the Monkey King character. The five rules specify that Monkey King can move left, right, up, down and throw a peach. Figure 1(b) also shows that Monkey King can have many different appearances.

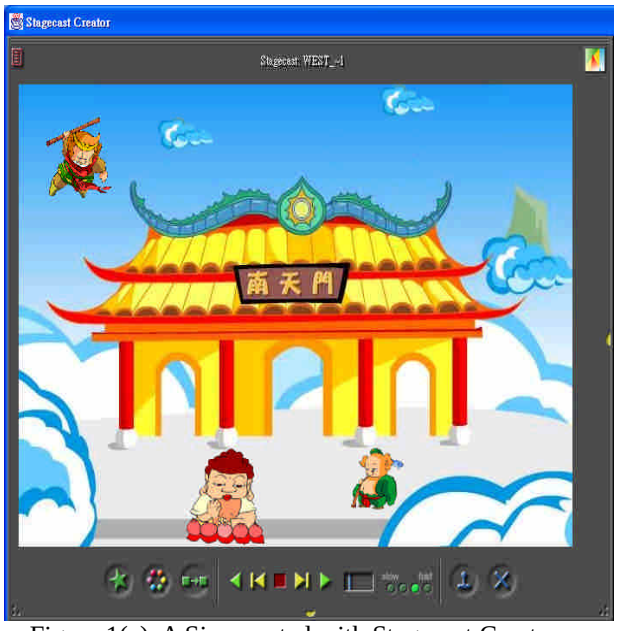


Figure 1(a). A Sim created with Stagecast Creator

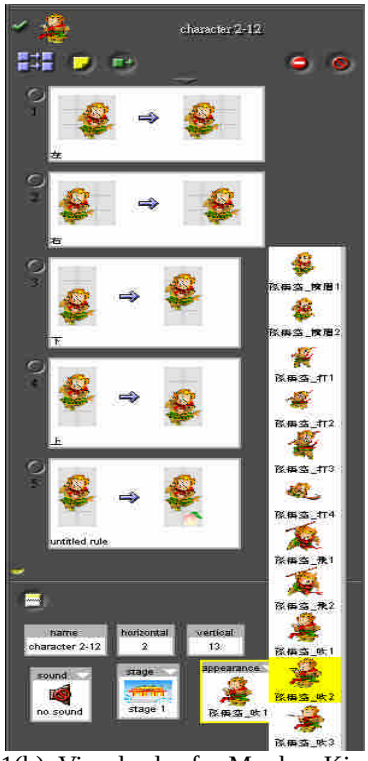


Figure 1(b). Visual rules for Monkey King

HANDS (Human-centered Advances for the Novice Development of Software) (http://www-2.cs.cmu.edu/~pane/research.html) is a programming system for children. The developers' goal is to provide an easy entry into creating interactive graphical programs, and to scale well so that it is possible to create more elaborate programs (Pane, 2002; Pane, Myers, & Miller, 2002). In HANDS, an agent named Handy sits in the upper left corner of the screen, manipulating information on cards. The cards can be drawn from a pile of new cards at the top right and displayed on a board at the center of the table. Figure 2(a) shows a board with ten cards, which are given pictures of Snow White, the prince, the witch and the seven dwarves. HANDS let programmers use English-like statements to describe interactions between characters. The programmer inserts code into Handy's thought bubble by clicking on Handy's

picture. Figure 2(b) contains a portion of code in Handy's thought bubble. With one statement we set Bashful's attributes, including his position on the screen, the direction he moves, etc. We have also written a statement that would set Snow White's direction to 90 when A is typed. Another statement specifies that 10 should be added to Snow White's energy and subtracted from dwarf's energy when Snow White collides into any dwarf. When the play button is pressed, Handy begins manipulating cards according to the instructions in his thought bubble.

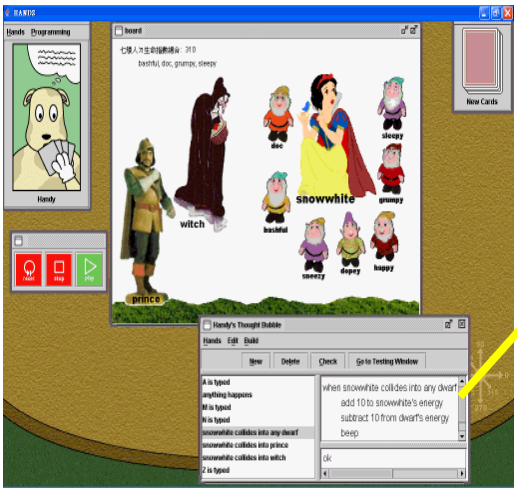


Figure 2(a). A program created with HANDS

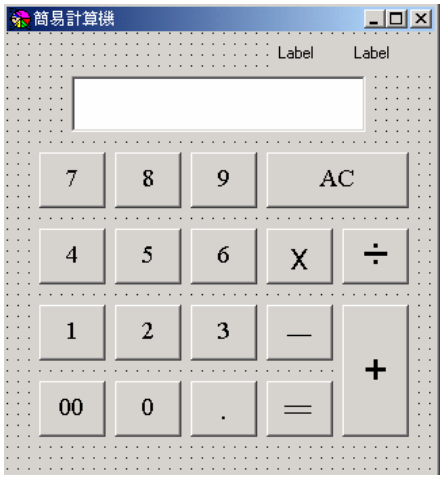
```
bashful x:535 y:191 back:bashful.gif kind:dwarf
speed:0 direction:330 energy:50;
…

when A is typed
   set snowwhite's direction 90
end when
…
when snowwhite collides into any dwarf
     add 10 to snowwhite's energy
     subtract 10 from dwarf's energy
     beep
end when

when anything happens
     set leastDwarf's back to CardWithLeast
          energy of all dwarfs
     set total's back to Sum the energy of all dwarfs
end when
```

Figure 2(b). Code in Handy's thought bubble

Visual Basic (http://www.microsoft.com) is designed to create Windows applications with graphical user interfaces. Programmers place custom controls such as text boxes, command buttons, labels and check boxes within forms that the user uses to control the application. Each custom control has a set of properties, including its name, size, caption, its position on the screen and many more. These properties can be altered either at design-time or at run-time. A control also may have one or more events associated with it. Users interact with application by triggering these events. When an event is triggered, the code for that event is executed. Figure 3 shows a calculator application and its two event handling functions, one for the 'click' event associated with the "**9**" button and the other for the 'click' event associated with the "**=**" button. As can be seen from Figure 3, VB allows Chinese characters to be used as identifier names. Comments may also be written in Chinese. During the experiments we encouraged students to name identifiers in Chinese as much as possible to avoid typing errors.



```
Private Sub Command9_Click()  ‘按下按鈕 9 時，將 9 併入數字框中
    數字框.Text =  數字框.Text & "9"
End Sub

Private Sub  等於_Click()
    If  運算符號.Caption = "+" Then
        數字框.Text = Val(第一個數.Caption) + Val(數字框.Text)
    ElseIf  運算符號.Caption = "-" Then
        數字框.Text = Val(第一個數.Caption) - Val(數字框.Text)
    ElseIf  運算符號.Caption = "*" Then
        數字框.Text = Val(第一個數.Caption) * Val(數字框.Text)
    ElseIf  運算符號.Caption = "/" Then
        數字框.Text = Val(第一個數.Caption) / Val(數字框.Text)
    End If
End Sub
```

Figure 3. A calculator application created with Visual Basic and some of its event

The material used for teaching Stagecast Creator was based on the 18-lesson tutorial provided on Stagecast Software's Web site. We translated the 18 lessons into Chinese and added a project (Betty's Kayaking) at the end to show how the features introduced in the 18 lessons can be used to create a more elaborate project.

The material for teaching HANDS centered around an example similar to that designed by Pane, Meyer, and Miller (2002), except that we change the scene of honey bees gathering nectar into that of Snow White and the seven dwarfs. We also showed students how the Betty's Kayaking project could be done using HANDS. The purpose was to let the students see for themselves that different programming packages can be used to implement same applications.

Visual Basic's teaching material was composed of nine examples. It started with a simple example of moving a picture around the screen at the command of arrow keys and progressed gradually to harder ones such as the calculator application shown in Figure 3. With each example we introduced the use of a couple or so custom controls as well as one or more syntactic features. In other words, new VB features were introduced only when they were needed in an example. The syntactic features covered include declaration of variables, assignment statements, arithmetic, logic and Boolean expressions, control structures (if-else, select-case, for-next), and built-in functions such as rnd, randomize, abs, and val.

2. 3 Data Collection

Data were collected during the experiments through classroom observation and questionnaire surveys.

(1) Students were asked to fill out questionnaires in which we asked such questions as how the students liked each of the three programming packages, how easy/difficult each programming package was to learn and use, and which package they would prefer to use again in the future.
(2) Parents were asked to answer a questionnaire in which we asked them what they had observed about their children's attitudes toward learning programming, how much time their children had spent practicing programming at home, and what their opinions are on teaching programming in elementary schools.
(3) Three of the authors served as observers during lab hours to take note of students' learning activities. The main purpose was to identify the common problems that students had in writing programs and getting programs to run correctly.

3. Results and Discussion

In this section we summarize some of the more important findings.

3.1 Students' feelings about the three programming packages

- When asked about how much they liked each programming package immediately after they learned it, the students gave 4.18 (out of 5) to both Stagecast Creator and Visual Basic, whereas HANDS received a rating of 3.52.
- After the students have learned all three programming packages, they were asked to pick their favorite programming package. Forty-one students (56.9%) went for Visual Basic, 26 students (36.1%) chose Stagecast Creator, and the remaining 5 students (6.8%) went for HANDS.
- As to how difficult each programming package was to learn and use (ranging from 1 for "very easy" to 5 for "very difficult"), Stagecast Creator was considered the easiest to learn, scoring 1.6, whereas HANDS and Visual Basic were considered a little more difficult, scoring 2.43 and 2.46 respectively.
- With regard to the expressive power of the three programming packages, 48 students (66.7%) answered that they could realize their programming ideas most easily with Visual Basic. Stagecast Creator came second, chosen by 20 students (28%). Only four students (5.6%) thought that HANDS was the most expressive.
- The question "Which programming package(s) would you use if you want to program again?" produced the following answers: Visual Basic was selected by 64 students (88.9%), Stagecast Creator by 61 students (84.7%), and HANDS by 44 students (61.1%).

It can be seen from above figures that HANDS turns out to be less well received than the other two packages by students. We think there are at least two reasons for it. First, not like Stagecast Creator and Visual Basic, HANDS is not a commercial product and it seems that nobody has been maintaining the software in recent years. We found that HANDS programs sometimes exhibited unexpected behavior, which might be due to uncaught bugs in its compiler. Second, since we were not able to find any user manual for HANDS, sometimes not even us could solve the programming problems encountered by students. This not only frustrated learners but might also have shaken their confidence in the software.

3.2 Parents' observations about their children's programming learning experience

- Forty-eight parents (87.3% of the respondents) noticed that their children looked forward to going to the programming class every morning.
- Fifty parents (90.9%) reported that their children spent more than an hour everyday working on programming assignments at home. Some students, according to their parents, even spent the entire afternoon and evening trying to add as many sophisticated features to their programs.
- Thirty-six parents (65.5%) said that their children were eager to show them their programming projects and all of these parents were very impressed with their children's work.
- Forty parents (72.7%) reported that their children had mentioned that they wished

to learn more about programming in the future.

- Twenty-five parents (45.5%) said their children had talked about difficulties they had come across while learning to program, which included difficulty in understanding English interface, slow typing speed and lack of time to write more programs.
- Forty-three parents (78.18%) strongly or moderately supported teaching of computer programming to their children in elementary schools. Reasons given by these parents include "it encourages children to do logical thinking", "it develops children's potential and sparks their interest in computer science", and "it enables children to have a better understanding of what is going on inside packaged applications and computer games they use everyday." The only parent (1.8%) that was against teaching computer programming in elementary school wrote, "it's useless for elementary school students to learn the basics of computer programming if there will not be a follow-up in the junior high schools."

3.3 Our Observations

- Although students in Taiwan start learning English from the third grade up, their command of English at the fifth or the sixth grade is not good enough to handle the English interfaces of the three software packages. Other than not being able to understand many of the English commands and messages displayed by the compiler, they also tended to misspell English words in their code.
- Some students hastened to create rules or writing code without trying to make sense of what they were doing first. These children typically resorted to the trial and error technique until they happened to get something right. Therefore, their program might work, but there usually were extraneous rules or code in their programs.
- Though we did spend time explaining different kinds of programming errors, namely compile errors, run-time errors and logical errors, most students had difficulties identifying sources of errors when their programs did not function correctly.
- Students tended to spend more time than necessary on interface design when we actually wanted them to do more thinking on program logic. For example, some of them would try all kinds of Chinese fonts for a VB text box before s/he started writing event handling functions for that text box. Some others would spend time drawing a pair of fancy glasses for Buddha instead of creating useful rules for the character. As a result, they usually had insufficient time left for coding.
- Students enjoyed working in teams. They discussed design ideas, suggested improvement on each other's work, and helped each other debug programs. Even so, they preferred having a computer to themselves during lab hours rather than a team sharing one computer.
- Students were proud of themselves when they presented their own or the team's work in front of the class. Though some were shy, most of them showed confidence and pride when they explained to the class their design ideas and how they managed to get it to work. They also learned to appreciate others' work through these presentations.

4. Conclusions

The study reported in this paper reveals that computer programming is not as

inaccessible and incomprehensible to children as many educators think it is. The results of our experiments indicate that children can learn and they enjoy learning computer programming. Most parents also express positive opinions about their children's programming learning experience. What we found from this research should help to answer an often-raised question of "Why teach computer programming in elementary schools" with a counter-question—"why not?" The important issues to be addressed next seem to be the establishment of a set of criteria that would help teachers select suitable programming packages for their computer courses and the development of more quality programming packages so that teachers can have more to choose from.

5. References

1. International Society for Technology in Education (ISTE) (1998). National Educational Technology Standards for Students. Available online from: http://www.iste.org/standards/.
2. National Research Council (1999). Being fluent with information technology. Washington, DC: National Academy Press. Available online from: http://www.nap.edu/catalog/6482.htm.
3. Pane, J.F. (2002). A programming system for children that is designed for usability. Unpublished doctoral dissertation, Pittsburgh, PA: Carnegie Mellon University.
4. Pane, J.F., Myers, B.A. & Miller, L.B. (2002). Using HCI techniques to design a more usable programming system. *IEEE 2002 Symposia on Human Centric Computing Languages and Environments*. 198-206.
5. Smith, D. C. (1994). Kidsim: Programming agents without a programming language. *Communications of the ACM, 37*(7), 55-67.
6. Smith, D. C., Cypher, A., & Schmucker, K. (1996). Making Programming Easier for Children. *Interactions*, September/October 1996, 59-67.
7. Smith, D. C., Cypher, A., & Tesler, L. (2000). Novice programming comes of age. *Communications of the ACM, 43*(3), 75-81.
8. Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C. & Verno, A. (2003). A model curriculum for K-12 computer science: Final report of the ACM K-12 task force curriculum committee. New York, NY: The Association for Computing Machinery.