

Teaching Strategy for Algorithmic Problem-Solving

SOOJIN JUN[†], SEUNGBUM KIM[†], WONGYU LEE^{††}

Computer Science Education, Graduate School[†],

Dept. of Computer Science Education, College of Education^{††}

Korea University

Anam-Dong, Sungbuk-Gu, Seoul

KOREA

soojin.jun@inc.korea.ac.kr, seungbum.kim@inc.korea.ac.kr, lee@comedu.korea.ac.kr

Abstract: - Many instructors claim that the teaching process of programming education improves problem-solving abilities. However, if the instructor focuses on teaching programming grammar (syntax) or on memorizing well-known algorithms' process maps, such as sorting and searching, students' cognitive loads could increase and this does not enhance their problem-solving abilities. Therefore, this paper proposes a sample curriculum and problems that students should solve in each theme to learn programming skills and algorithmic problem solving. The programming course was based on methods, such as storytelling, simulations, and games to motivate students. We also present a teaching strategy to design a programming teaching course. The teaching strategy process is founded on basic steps of algorithm creation. Then, the Class-Responsibility-Collaborator card model was used as a hands-on activity for program design. We used an educational programming environment in our experiment in which the students can easily implement their programs. Our experiment ran for elementary school students for 12 hours. We demonstrate the examples of students' problem-solving activities that were discovered and solved by teacher or learner during the programming scenario. Learners' interests, satisfactions, and achievements in learning programming with the teaching strategy had considerable positive results.

Key-Words: - Algorithmic problem-solving, Teaching strategy, Squeak etoys, Elementary school students, Novice

1 Introduction

The revolution in information technology has resulted in innovations that have increasingly visible effects on human life. There is high interest in computer education within the school curriculum and growing investments for this field. We also need effective methods and tools to teach computer education content, including programming, an essential part of the computer science curriculum [1].

It is important to learn algorithmic thinking in computer science education for novices or elementary students [1]. Educational programming language is a basic tool of computer education [2]. Students are also interested in using a computer [3]. However, it is difficult for the student to write computer programs in the beginning. They first need to learn problem-solving strategies.

Some researchers states that computer programming was fertile ground for the development of problem solving abilities [4]. However, the

approach in traditional programming education has limited research in supporting systems to improve programming skills or developments of programming languages [2][5][6][7][8][9]. Little research has been conducted on teaching and learning methods that consider students' interests and motivation for enhancing the problem solving ability [4][10][11]. Most software engineering students program games, as exercises, to learn a programming language or operating system [12].

This paper suggests a teaching method based on edutainment, such as storytelling, simulating, and game programming, not practicing grammars or memorizing programming skills to learn programming for elementary school students.

In this paper, Squeak etoys is used to keep students' interests high and to implement their program more easily. Squeak etoys is an Educational Programming Language (EPL), not a high-level language. Squeak is a programming environment designed to teach programming skills to children.

Working with Squeak provides children with programming abilities whilst they play [13].

Activities based on the CRC (Class-Responsibility-Collaborator) card model [14][15] are used for the student to analyze and describe the process and the algorithm of a problem systematically and more easily before they program on a computer.

Moreover, they could learn programming easily and amusingly using Squeak toys. Although these programming contents are a middle school curriculum, our experiments are run for elementary school students. Our approach is convenient for younger students, as well as secondary students.

Finally, this paper evaluates if these teaching methods and approach to problem solving are effective in schools.

2 Backgrounds

2.1 Squeak-toy Environment

Squeak is a programming environment designed to teach programming skills to children. Working with Squeak provides children with programming abilities while playing. It is a media-rich authoring environment with a simple powerful scripted object model for many kinds of objects created by end-users. It runs on many platforms and is free and open source [16].

Having a look at the use of Squeak for learning computer programming, we see that it provides principles mentioned by Arthur Chickering and Stephen C. Ehrmann in 1996 [17]. Squeak toys provides (1) immediate feedback. The instructor is able to check if students have learned what they were taught. It contacts students and the instructor easily. They can spend their class time full of (3) enjoyment [3], which is very important in education [19]. They enjoy seeing their programs work. The student can easily change the program flow and correct it if they make a mistake. This provides (4) effectiveness. Squeak allows this with its visual environment. Students do not have to write the program code directly, but drag-drop the object in an easy way. Programming code elements are defined as objects. Students only use their mouse to move the objects. Using objects lets the student become familiar with (5) object-oriented programming, a good thing for their future computer science education [18].

Squeak also allows (6) active learning, because students learn by doing [18]. The interesting point is

that the method defined below will enable productive (7) collaborative learning between students and instructor. Thus, it provides for (8) joint problem solving.

The advantages are not limited to the ones mentioned here. There is more about using Squeak for computer programming education. Thus, it seems using Squeak is appropriate for teaching algorithmic thinking to middle school students enjoyably and allows them to apply a real 'good practice'[17].

2.2 Curriculum of Programming in school

Middle school's model curriculum for computer science describes that Programming Languages (PL) introduce the student to basic issues associated with program design and development.

The focus of this unit is to establish an appreciation of the work being done by software [1]. Student learning objects of the PL are as follows.

The student will be able to:

1. Code, test, and execute a program.
2. Convert a word program to code using top-down design
3. Select appropriate data type
4. Write structured program code
5. Describe the changes occurring in RAM, as code executes

We developed and applied variable activities based on "Code, test, and execute a program" and "Convert a word program into code using top-down design" of this curriculum.

2.3 Problem-based learning and programming

Programming is a process of problem solving. Therefore, students can experience new algorithms by being involved in the entire context, rather than simply be given a programming problem.

Problem-based learning (PBL) is an active learning strategy. Students are engaged by these active PBL in real world problems. The PBL also allow them to connect the learner with real world and to provoke their high level thinking skill [18]. Therefore, they can achieve their highest potential in their fields [20]. Strategies to aid the PBL for programming are various [12][15][21][22].

A CRC (Class-Responsibility-Collaborator)-card as one of the strategies describes the properties that certain kinds of objects of interest in the problem domain have in common. An object can be a tangible

thing, person, place, event, or (abstract) concept. A class should have a single and well-defined purpose.. It should be named by a noun, noun phrase, or adjective that applicably can state the abstraction. The class name is written across the top of the CRC-card. The goal of the CRC-card is to develop, discuss and evaluate object-oriented models. Students establish these objects for exploring scenarios of their programming. When a command is sent to an object, the player tries to achieve the request by playing the equivalent responsibility, which is listed on the CRC-card. This role-play using CRC card gives learners a 'live impression' of the functionality of an object-oriented program [15].

3 Methodologies

3.1 Design of Curriculum for programming learning

First, we developed a curriculum including storytelling, simulation, and game programming to improve programming skills. This enables students to understand programming easily, interestingly and progressively from basic concepts.

Table 1 Programming Learning Curriculum

Days (12h.)	Types	Themes	Programming skills
1 st	Story	Drive Objects	Interface, Making objects and Scripts, Loop statements
2 nd	Story	Rabbit vs. turtle race	Test(Conditional statements) and all above
3 rd	Simulation	Collecting candies	Controlling scripts of the above
4 th	Simulation	Following a train	Top-down design at the above
5 th	Game	Free Project	All of the above
6 th	Game	Saving a baby butterfly from the spider	All of the above

Moreover, students touch various algorithmic solutions by analyzing complete project solutions that the teacher offers with the foundation of such an educational process. With such a method, students can become interested in learning programming and the interest increases through game source analysis. They can also learn a source maker's approach and philosophy to solve a problem [23].

Table 1 shows the curriculum for the experiment. We let the children analyze and make familiar stories or interesting games. Moreover, they programmed together with pair for effective learning [24]. This course was run for six classes, two hours a day; 12 hours in all. The teacher provided students problems in the 1st to 4th class by telling a story or showing a model project. Thus, we applied our new teaching strategies to support students' algorithmic problem solving. Then, the students made a free project after designing it themselves in the 5th class. Their programming skills and problem solving were evaluated with a test in the last class.

Table 2 shows the problems that the students should solve in each theme. Programming problems can let students acquire valuable solutions by touching on new algorithmic situations in given themes, not following a given assignment exactly. It should also include problem situations in which students can think. Therefore, the teacher should provide the student with encouraging problems related to a guided project, not just following it, in this curriculum.

Table 2 Problems as themes

Days (12h.)	Themes	Problems
1 st	Drive Objects	Drive your object with a joystick.
2 nd	Rabbit vs. turtle race	Control the race between the rabbit and the turtle by changing instances.
3 rd	Collecting candies	Collect all candies, using a conditional statement in the field.
4 th	Following a train	Let the back train follow the front train only on the rail.
5 th	Free Project	Make a game of your choice.
6 th	Saving a baby butterfly from the spider	Do not let the spider leave the spider's thread; save the little butterfly from the spider.

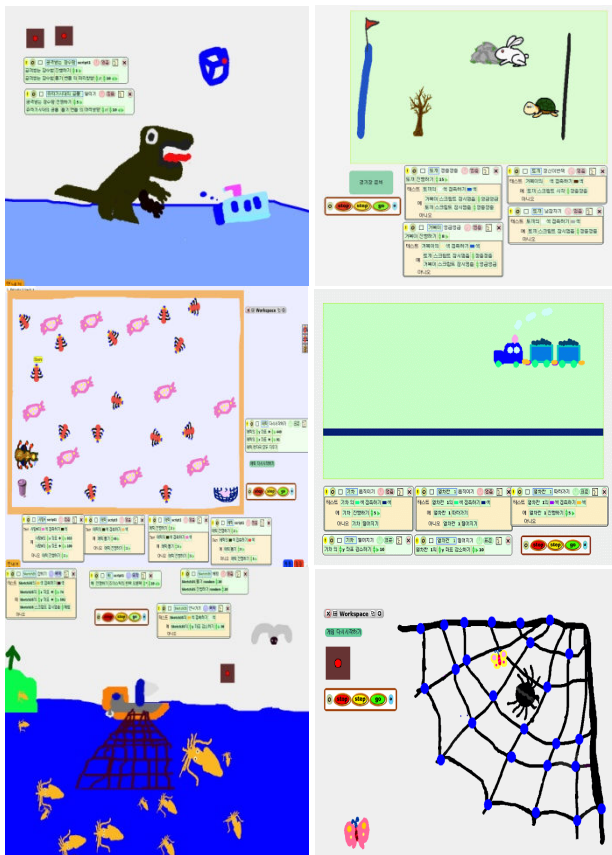


Fig. 1 Projects examples

Figure 1 shows projects' examples. Students can learn programming and enhance their problem-solving ability by making valuable edutainment projects by themselves.

3.2 Teaching strategy for algorithmic problem solving

Understanding the problem and communicating a design to peers is not easy for many students. When students try to design and code at the same time, they frequently make the situation more complex than it would be if they first designed a solution in a natural language and then subsequently faced the issue of coding [21]. Therefore, these hands-on activities (or unplugged) have been developed for computer science novices [25].

Students need abilities to analyze and to design problems to implement programs. Analyzing programs can allow learners see new ideas and functions naturally. Designing programs can support learners to understand the structure of programming.

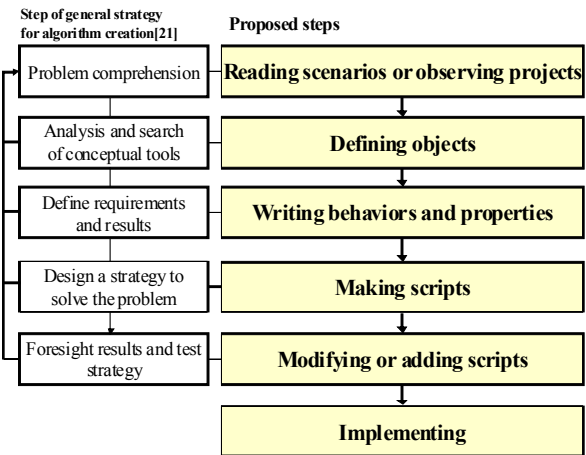


Fig. 2 Teaching steps for algorithmic problem solving

In this paper, we suggest a new teaching strategy of programming learning for algorithmic problem solving based on the ‘General strategy for algorithm creation’ of Jorge Vasconcelos [26]. Figure 2 shows the teaching steps in a class.

We transformed and applied activities in our class to design object-oriented programs. The activities for program design were based on CRC (Class-Responsibility-Collaborator) Card Modeling, a simple, yet powerful object-oriented analysis technique [14].

CRC-cards have been described as a tool to teach object-oriented thinking to programmers [15][27]. It is a simple informal tool for collaborative object-oriented modeling. The power of the CRC-card approach lies in its associated role-play activities.

Scenario role-play is usually used in the social sciences to evaluate human behavior in specific, yet hypothetical situations. Role-playing is an effective way to simulate or explore hypothetical situations, since the characters and scripts can be easily varied [15].

In object-oriented software development, the characters are the objects in our system and the scenarios are situations of system usage. Scenario role-play can be used in the very early stages of software development, before any code is written (or even designed) [28]. That makes it appealing for novices, as a means to become familiar with the object-oriented way of thinking, without needing to bother about syntactic details of programming or design languages [15].

Accordingly, we make students learn using steps of activities to analyze and design games and

simulations: concept of objects, behavior of the objects, scripts etc. The details of each step follow.

Step 1. Reading scenarios or observing projects

Students read scenarios or demonstrate projects of the stories, the simulations, or the games. They also explain (speaking or writing) their processing. Then, the children can understand an outline of the stories.

For example, if they program a race between a rabbit and a turtle, the children can understand the logical situation by reading and explaining the story.

Step 2. Defining objects

Children grasp the main objects in the stories. For instance, although all items (rabbit, turtle, tree, rock etc.) are objects, students know that they have to command only the rabbit object and the turtle object in practice.

Step3. Writing behaviors and properties of objects

After finding the objects, children write the object's behaviors and properties: "the rabbit is running" and "If the rabbit meets a rock, he sleeps".

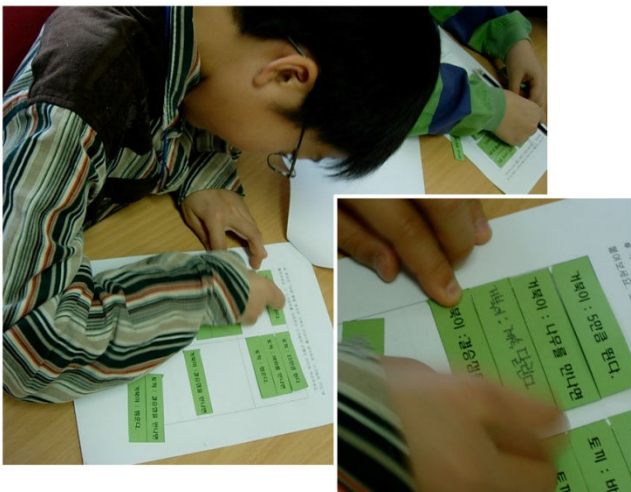


Fig. 3 Activities to Make Scripts

Step4. Making scripts using scripts card

Children compound scripts (tiles), pieces that a teacher gave them, or make scripts through hands-on activities using tangible things, such as cards and Post-it notes, as in Figure 3.

Therefore, the concept is to make scripts, not to write scripts. In addition, if they are novices or young programming students, script components prepared beforehand could be used. They could write the scripts on cards or Post-its, as they become proficient in making scripts. For example, they can compound

commands of "a turtle: forward by 5", "a turtle: If he meets a tree", "a rabbit: forward again" and so on.

Step5. Modifying or adding scripts (Optional)

The children are able to modify scripts using new tiles and cards and to add new ideas to the completed script cards. Students can predict the results of the scripts by modifying or adding lines.

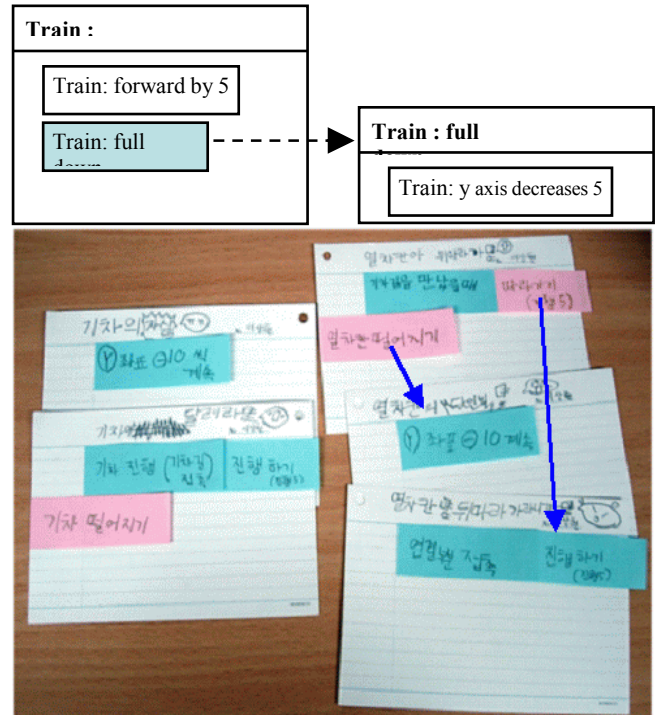


Fig. 4 Activities for Top-down design

Step6. Top-Down design (Optional)

They can use script tiles exactly. In addition, they should make new scripts for specific commands to implement some behaviors of the objects. This is motive to let them design top-down. For instance, "train: forward by" can use scripts of Squeak category, however, "train: full down." should compound two or more scripts on a new script card (see figure 4)

Step 7. Implementation with Squeak etoys

The children implement programs with the script cards using Squeak etoys. They can eventually make simulations or games.

4 Assessments

4.1 Experiment

We create an environment to show that our curriculum and teaching strategy using Squeak etoys are very useful and effective methods for algorithmic problem solving for students in learning programming.

We show examples of problem solving in a free project and an assignment from a teacher. A post-test on problem-solving ability for given situations was conducted and the ability to apply the programming skills ('saving a baby butterfly' of figure 5): naming of objects and scripts, loop statements, conditional statements and controlling scripts was assessed.

We surveyed the children on how satisfied they were with the programming activities and how much they maintained their interest in programming.

This experiment was performed over 12 hours (three weeks: 2 days a week for two hours), for 3rd grade ~6th grade students (N=13) of a Korean elementary school. None of the students had learned a programming language before.

A pre-test was used as a measure of interest in programming and basic computer literacy. Although they learned a programming language for the first time, their interest in programming was over the mid-point. Their abilities using computer was similar to each other and they were familiar with basic computer skills, such as using the Internet, and OA applications.

4.2 Solving Algorithmic Problems

4.2.1 Problem solving in assignments

For example, in the game "Saving a little butterfly from the spider" students used various methods to solve the problem.

Example: If I try to make sure that the spider won't go out of the cobweb, how do I do this?

Students' solution examples:

Solution 1: After the inside of the Cobweb is colored with different colors from the background, if a spider meets a color of the outside background, let it turn back to the inside of the web.

Solution 2: After the outside of the Cobweb is colored with different colors from the background, if a spider meets the color of the outside background, let it turn back to the inside of the web.

Solution 3: Cobweb is colored with a different color surround. If the spider meets that color, it turns back to the center.

Solution 4: The cobweb background is filled with a different color. When the spider sees it, it goes around

in random movements. When it does not see the color, it turns around.

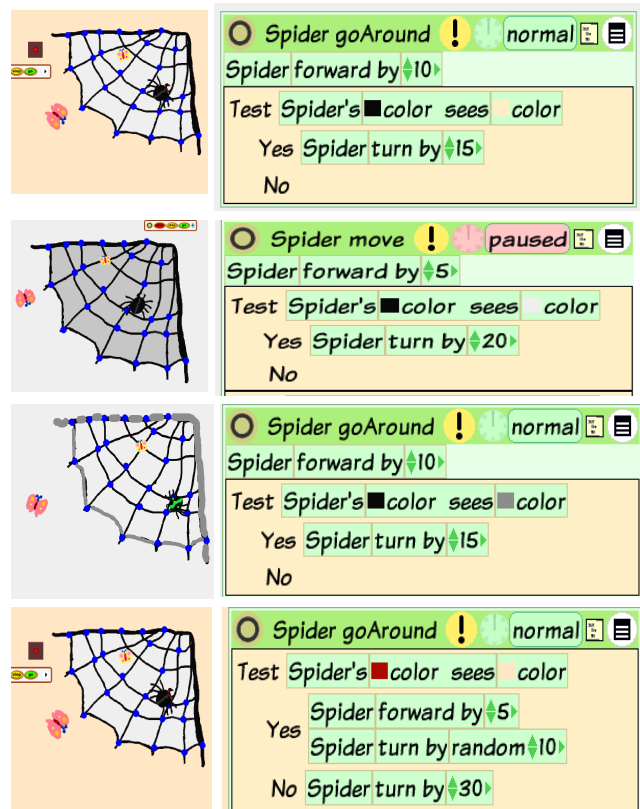


Fig. 5 Examples of problem solving in an assignment

4.2.2 Problem solving in students' own projects

When a learner creates a project, s/he faces problems. Thus, s/he should solve the algorithmic problems to implement what s/he wants. The students solved or implemented the following problems or situations they had never previously learned or exercised.

Table 3 Example 1 of student's own project

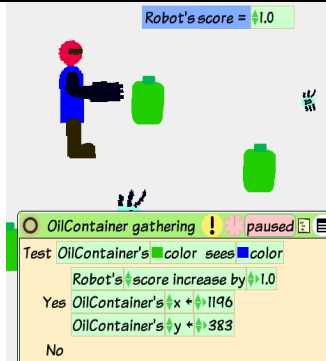
Squeak eToys' tiles for programming	Original codes of the tile codes
	self forward: 20 atRandom. self turn: 20 atRandom
	(self color: (Color r: 1.0 g: 0.71 b: 0.129) sees: (Color r: 0.935 g: 0.935 b: 0.935) ~ false ifTrue: [self setY: self getY - 30])

Example 1: 3rd grade student's project

Problem: If I stop them going up to the water surface, although squid swim freely, how do I do this?

Solution: forwarding and turning squid has a random value, it makes them repeat, and when meeting air it decreases the y coordinate.

Table 4 Example 2 of student's project

Squeak eToys' tiles for programming	Original code for the tile codes
	<pre>(self color: (Color r: 0.129 g: 0.806 b: 0.0) sees: (Color r: 0.0 g: 0.0 b: 0.774)) ifTrue: [Robot set score: Robot get score + 1. self setX: 1196. self setY: 383]</pre>

Example 2: 6th grade student's project

Problem: How can I automatically count the number of containers of oil collected by the robot?

Solution: N is incremented by 1 each time a robot collects a container of oil using variable N.

As you see from the examples of tables 3 and 4, although sometimes students did not learn the concept about random, y coordinate, a variable and so on, they could naturally acquire the needs and uses of these. These programming courses automatically provide problems the students have to solve, even if the teachers do not provide the problems.

4.3 Programming skills

We surveyed how the students think about finding the object, writing behaviors and properties of objects, and making scripts for each stage: analyzing, designing and implement using Squeak. The survey design questions used a 5-point Likert scale (difficult: 1, easy: 5).

The children thought that finding suitable objects for programming was the easier than the next step, writing behaviors and properties (Fig. 6); the implementation step was the hardest. They felt that making the scripts at the design step was the hardest. However, they felt that it was easy again when it came to implementing the scripts.

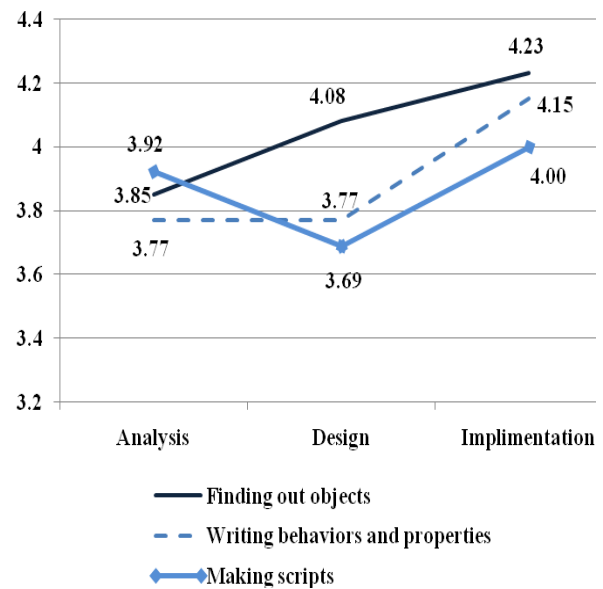


Fig. 6 The self-evaluation about learning during programming processing

We also asked them questions about the help doing the activities gave them to program using Squeak very well.

- Q1. It helped me to understand concepts of the scripts and the tiles: 76.9% - always
- Q2. When we made a program after making the script cards, it was easier: 84.6% - always

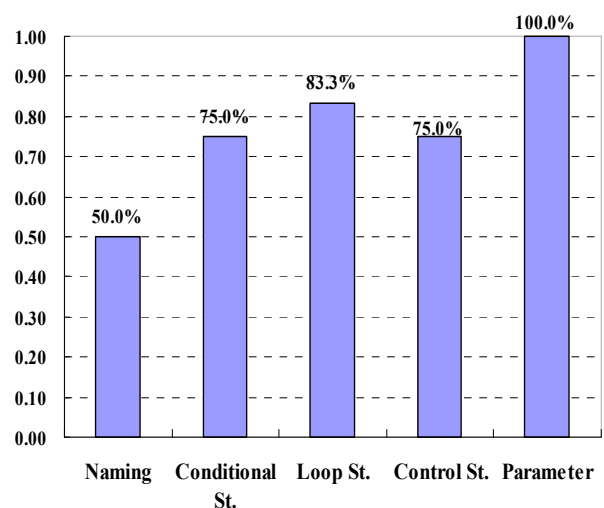


Fig. 7 The rate of students passing each skill

Figure 7 shows the success rate of the students for each of the programming skills test for 'the saving butterfly from a spider' project. Most of them (i.e. over 75% passed) could use the conditional statements,

loop statements, control statements, and parameters for solving the problems. Only naming caused problems (50% pass rate). Overall, students felt a high sense of achievement and acquired programming skills.

4.4 Interest in programming

Changes of interest in programming were also surveyed using the 5-point Likert scale (very much agree; 5, very much disagree; 1).

- Q1. I am interested in making programs.*
Q2. I like to learn a programming language.
Q3. I like to make programs.
Q4. I also want to learn other programming languages.

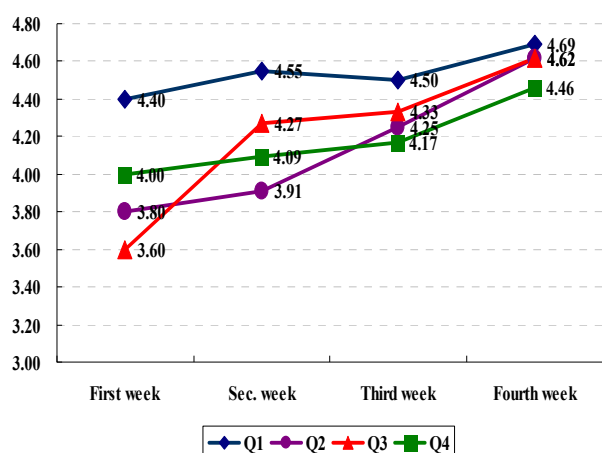


Fig. 8 Programming interest over time

We found out that the students' interest in programming was higher over time for all questions (Fig. 8). Scores for Q3 (Interests about making programs) were statistically significant with $p < 0.05$ between the first week and the last week. That is, we could see that children got more interested in making programs during the course.

5 Conclusions

Programming in computer science education is an important domain. Programming education is a form of problem solving that can improve the learner's cognitive skills, logical thinking, and reflective thinking [9].

Therefore, many researchers and instructors have developed teaching methods to help students learn computer science concepts, such as programming [2][29].

However, it is not easy to teach programming to young students. Students do not have an interest in programming; most programming languages are too difficult. Although there are some educational programming languages, there are few curricula for learning programming suited to young students.

Therefore, this study designed a curriculum of programming for algorithmic problem solving. Every class has a problem that the students should solve. Then, we proposed strategies based on steps within a general strategy for algorithm creation to support the design of algorithms for programming.

In this study, we also experimented with elementary school students using the methods and Squeak toys. Although the students did not learn basic concepts related to programming, they could naturally acquire needs and uses of these.

First, we can find that our activities to analyze algorithms help children to program what they want more easily, since they answer all are easy at the last, implementation stage, although they feel describing the behavior and properties and making the scripts are not easy at the design stage. They thought that these strategies helped their programming. Over 75% of students were assessed as passing the programming skills test, except for naming objects. We think that the students may often forget the naming of objects when they make programs, because they do not take sufficient care; they might think naming objects is not important, because the scripts are not complex yet. Especially, we can see that Squeak toys supported their easy programming.

Second, interest of the programming increased over time. Activities captured their interest, because approaches, such as CRC card activities for programming eased their hard workload. Therefore, we can state that our strategies helped them to like programming more.

In conclusion, we provided various problem situations, so that students could have experiences solving them in our curriculum. In addition, activities based on steps of algorithm creation and the CRC card model helped students program what they wanted. The students felt high a sense of achievement, and acquired programming skills. Their interest during learning programming increased.

References:

- [1] Tucker, Allen, (editor), Deek, F., Jones, J., McCowan, D., Stephenson, C., and Verno, A. A Model Curriculum for K-12 Computer Science:

- Final Report of the ACM K-12 Task Force Curriculum Committee. *Association for Computing Machinery (ACM)*, New York, October, 2003.
- [2] Caitlin Kelleher and Randy Pausch, Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, *ACM Comput. Surv.*, vol. 37, 2, pp. 83-137, 2005. [lowering]
- [3] Karuno, H. Konomi, S., *Creating, Connecting and Collaborating Through Computing Squeak* workshop experiences in Kyoto. Graduate Sch. of Informatics, Kyoto Univ., Japan, 2003. [Kyoto]
- [4] Karen Swan, LOGO Programming, Problem Solving, and Knowledge-based Instruction, Paper presented at *the Annual Meeting of the American Educational Research Association*, p.39, 1990.
- [5] Martin, J. L., "Is Turing a better language for Teaching Programming Than Pascal?" Honours Dissertation, University of Stirling, Department of Computer Science, 1996. [martin]
- [6] Pane, J. F., Nyers, B. A. "Usability Issue in the Design of Novice Programming Systems.". Technical Report, Carnegie Mellon University, 1996.
- [7] Moreno and Niko Myller and Erkki Sutinen and Mordechai Ben-Ari, Visualizing programs with Jeliot 3, AVI '04: *Proceedings of the working conference on Advanced visual interfaces*, pp. 373-376, 2004, ACM Press, New York, NY, USA, DOI=<http://doi.acm.org/10.1145/989863.989928>.
- [8] Handhausen, Christopher D., Brown, Jonathan L., "What You See Is What You Code: A 'Live' Algorithm Development And Visualization Environment For Novice Learners." Visualization and End user Programming Laboratory, School of Electrical Engineering and Computer Science, Washington State University. March 2006.
- [9] Park, W.G., Lee, J., Design of Programming Learning System for Children and Beginner, in processing *Korean Association of Information Education*, 2000.
- [10] Zaigham Mahmood, A framework for teaching introductory software development, *WSEAS Transactions on Computers*, Volume 8 Issue 8, August 2009.
- [11] Jun, S., Kim, S., Lee, W., Online Pair-Programming for Learning Programming of Novices, *WSEAS Transactions on Advances in Engineering Education*, Manuscript received Jul. 1, 2007; revised Sep. 11, 2007
- [12] Wikipedia, http://en.wikipedia.org/wiki/Game_programming
- [13] Vural, H., Jun, S.J., et al. Using Squeak for Teaching High School Students 'How Computers Think', *Korea Association of Computer Education*, 2006.
- [14] David M. Rubin, *Methodologies and Practices – White Paper: Introduction to CRC Cards*, Softstar Research, Inc., January 1998.
- [15] Jürgen Börstler, Teaching object oriented modeling with CRC-cards and role playing games, in *Proceedings WCCE 2005*, Cape Town, South Africa, Jul 4-7, 2005.
- [16] What is Squeak? <http://www.squeakland.org>
- [17] Chickering, Arthur and Ehrmann, Stephen C., "Implementing the Seven Principles: Technology as Lever", *AAHE Bulletin*, 1996.
- [18] Yang, H., Kuo, L., Yang, H., Yu, J., Chen, L., On-line PBL System Flows and User's Motivation, *WSEAS Transactions on Communications*, Issue 4, Volume 8, April 2009.
- [19] Stasko, J. T., "Using student-built algorithm animations as learning aids", In *Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education* (San Jose, California, United States, February 27 - March 01, 1997). J. E. Miller, Ed. SIGCSE '97. ACM Press, New York, NY, 25-29, 1997. DOI=<http://doi.acm.org/10.1145/268084.268091>
- [20] Duch, B. "The Power of Problem-based Learning." *About Teaching*, 1995. 47: pp.1-2.
- [21] Joseph Bergin, Joseph Bergin et al. *Non-Programming Resources for an Introduction to CS: A collection of resources for the first courses in Computer Science*, ITiCSE 2000 Working Group Reports, 2000.\
- [22] Shin, J.H., Kim, J.H., Programming Learning by Understanding of Game Programs, in processing *Korean Association of Information Education*, 2001.
- [23] Committee on Information Technology Literacy. "Being Fluent with Information Technology", Washington DC: National Academy Press, 1999.
- [24] Jun, S.J., Kim, S.B., et al. Pair-Programming in Online Programming Learning Environment, *IPSSJ symposium Series* Vol.2006, No.8, 2006.
- [25] Bell, T.C., Witten, I.H., and Fellows, M. *Computer Science Unplugged: Off-line activities and games for all ages*. 1998. available from <http://unplugged.canterbury.ac.nz>.

- [26] Jorge Vasconcelos, *Basic Strategy for Algorithmic Problem Solving*, 2007, at URL: <http://www.cs.jhu.edu/~jorgev/cs106/ProblemSolving.htm>
- [27] Beck, K., Cunningham, W. (1989). A Laboratory for Teaching Object-Oriented Thinking. Proceedings *OOPSLA'89*. 1-6.
- [28] Larman, C. *Applying UML and Patterns; An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Pearson Education International Upper Saddle River, NJ 07458, 2004.
- [29] Chin, Suk Kim . "A Practical Model For Improving Student Learning of A Programming Language". School of Business and Informatics, Australian Catholic University. *36th ASEE/IEEE Frontiers in Education Conference*. 2006