International Journal on Advanced Science Engineering Information Technology

Vol.8 (2018) No. 4-2
ISSN: 2088-5334

An Educational System Design to Support Learning Transfer from Block-based Programming Language to Text-based Programming Language

Soyul Yi[#1], Youngjun Lee[#2]

[#]*Dept. of Computer Education, Korea National University of Education,*
250 Taeseongtapyeon-ro, Cheongju-si, 28173, South Korea
E-mail: [1]soyulyi@knue.ac.kr, [2]yjlee@knue.ac.kr

***Abstract*— In programming education, novices normally learn block-based programming languages first, then move on to text-based programming languages. The effects of learning transfer on learning two or more languages in programming education has had positive results. However, block-based and text-based programming languages have different figurations and methods, which can occur cognitive confusion or increase cognitive overload for learners. Thus, it is necessary to develop an educational system that supports learning transfer. We suggest using the following design principles: utilization of advanced organizers, problem-solving-based learning content, and simple and intuitive user interface and screen layout. Two types of screen composition modes are presented: training mode and practice mode. Future research must implement and apply this design in the educational field to verify its effectiveness.**

***Keywords*— Scratch 2.0; python; block-based programming language; text-based programming language; educational program system; learning transfer.**

I. Introduction

As computational thinking is increasingly emphasized in education, South Korea and many other countries around the world are providing programming and coding education in computing-related subjects, such as computing and informatics, to improve students' computational thinking. Various methods have been implemented in computing-related subjects, such as unplugged activity, block-based and text-based programming language education, and physical computing education [1].

In programming education, novices are usually taught block-based programming languages like Scratch 2.0, Entry, and Kodu before moving on to text-based programming languages like C, C++, Java, and Python. Although there is some variation, it is usually recommended that unplugged activities and block-based programming languages be taught to elementary school students. Middle school students are generally introduced to block-based programming languages and physical computing education, and text-based programming languages are reserved for high school students [2].

Block-based and text-based programming languages include the same basic logic, but like all other programming languages, there are some differences. Although block-based programming languages have more limitations than text-based programming languages, they are used to teach novices because they present several advantages, such as being very intuitive and having no syntax error.

Therefore, it is necessary to develop an educational system that allows students to form a cognitive scheme as novices by learning to the programme through block-based programming languages before proceeding to text-based programming languages.

II. Material and Method

A. Related Works

1) Learning Transfer: The transfer of learning, which is the effect of prior learning or experience on new learning or performance [3], was initially introduced as the transfer of practice by E. Thorndike and R. S. Woodworth [4]. Learning transfer is the effect of prior learning on performance or new learning.

There are three kinds of learning transfer: special, general, and mixed (see Table 1). Particular transfer refers to a process that helps students better accomplish a specific task whereas general transfer refers to an experience that helps students complete different types of tasks. Mixed transfer, which involves applying concepts of a general principle to different tasks, is broader than specific transfer and narrower than general transfer [5]. Generally, when one is taught a block-based language before learning a different programming language, mixed transfer of learning takes place.

TABLE I
THREE KINDS OF LEARNING TRANSFER [5]

Type of Transfer	Description	Example
Specific Transfer	Specific behaviors (or procedures or facts) in A are like those required in B.	Latin has some similar words and verb conjugations as Spanish, so learning Latin will help you learn Spanish.
General Transfer	There is nothing in common between A and B, learning A is a mind-enriching experience.	Latin improves the mind so learning Latin should help you solve logic problems.
Mixed Transfer	The same general principle or strategy is required in A and B.	Learning how to pronounce printed words helps you pronounce words in Latin and Spanish.

2) Educational Programming Language: The two types of educational programming languages are visual-based and text-based. Visual-based programming languages (VPLs) allow students to program by manipulating elements visually rather than by specifying them textually. There are several VPLs, one of which is a block-based programming language [6].

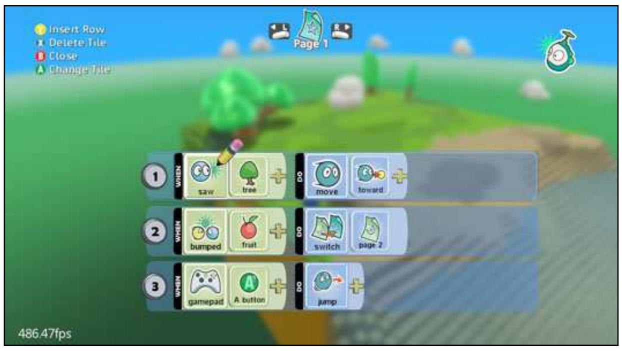


Fig. 1 Kodu (Visual Programming Language)

3) Block-based Programming Language: Block-based programming languages provide an easy-to-use interface and intuitive commands. This relieves the learner's cognitive overload in the programming process. Also, there are no grammatical errors due to text input because it is based on a coding scheme that frames building blocks. Therefore, it is easy for novices to make a program using this language.

Because of these advantages, education often includes block-based programming languages, and research on the effect of an educational programming language is ongoing [7]. Examples of a block-based programming language are Entry and Scratch 2.0, the most widely used programming language in the world.

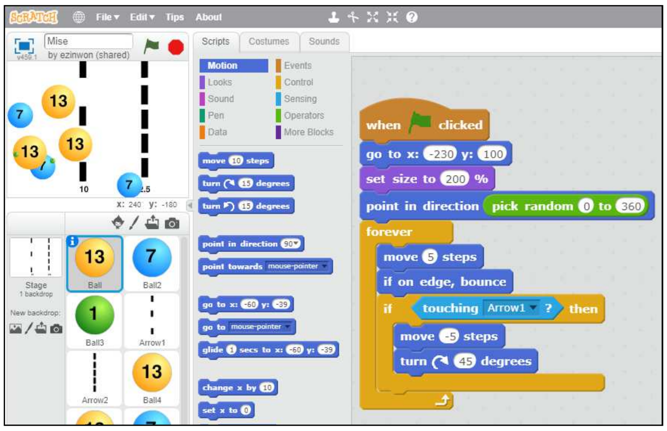
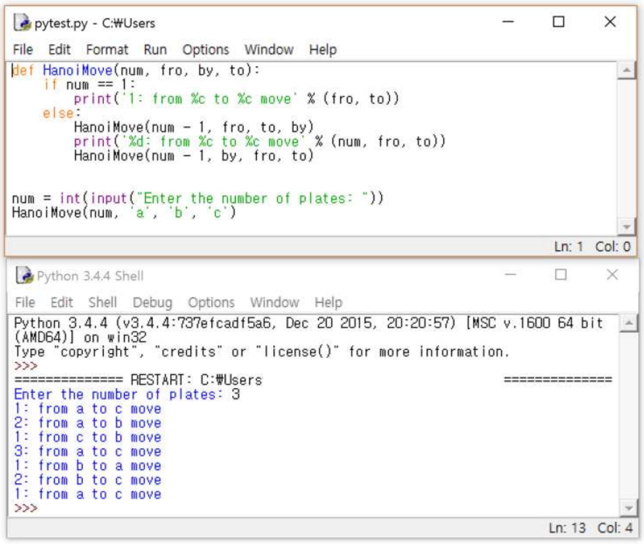


Fig. 2 Scratch 2.0 (Block-based Programming Language)

4) Text-based Programming Language: Text-based programming languages require entering the command syntax necessary for programming. Although understanding language structure and learning to use the correct syntax are lengthy processes, text-based languages present fewer restrictions on complicated programming [8]. As programming skills improve, the time spent programming can be decreased. Programming processes, including debugging, can improve logical and analytical thinking [9], [10].

Python, Processing, RUR-PLE, LOGO, C, Visual Basic, and others are text-based programming languages that can be used in education. The Computer Science Teachers Association (CSTA) in the United States recommends using Python as a text-based programming language in education.


pytest.py - C:₩Users
File Edit Format Run Options Window Help
Ln: 1 Col: 0
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Ln: 13 Col: 4

```
def HanoiMove(num, fro, by, to):
    if num == 1:
        print('1: from %c to %c move' % (fro, to))
    else:
        HanoiMove(num - 1, fro, to, by)
        print('%d: from %c to %c move' % (num, fro, to))
        HanoiMove(num - 1, by, fro, to)


num = int(input("Enter the number of plates: "))
HanoiMove(num, 'a', 'b', 'c')
```

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:₩Users                                ==============
Enter the number of plates: 3
1: from a to c move
2: from a to b move
1: from c to b move
3: from a to c move
1: from b to a move
2: from b to c move
1: from a to c move
>>>
```

Fig. 3 Python (Text-based Programming Language)

B. Effects of Two or More Languages in Programming Education on Learning Transfer

Because young novices often learn basic programming logic through block-based languages before proceeding to text-based languages [1], [2], it is important to examine the effects of using two or more languages on learning transfer in programming education. Choi, Kim, and Cho (2016) found that those learning Processing after learning Scratch 2.0 demonstrated higher interest, lower cognitive overload, higher immersion, and more confidence than the control group [11]. Similarly, Park and Cho (2012) found that students in an introductory programming course who learned Scratch 2.0 showed significantly more improvement regarding problem-solving ability, programming ability, and instructional satisfaction than those the control group [12]. In a recent study, So and Kim (2016) demonstrated effective learning transfer in students learning Python after learning Scratch 2.0.

TABLE II
CORRELATION BETWEEN PROGRAMMING EVALUATION FACTORS

	Learning achievement of block-based language	Project achievement of block-based language	Learning achievement of text-based language
Learning achievement of block-based language	.598*		
Project achievement of block-based language	.597*	.572*	
Learning achievement of text-based language	.681**	.662**	.635*

$**p < .01$, $*p < .05$

Based on researches, we conclude that block-based programming facilitates positive learning transfer to text-based programming learning. It is commonly thought that when learning two or more programming languages, learning block-based programming languages before learning text-based languages results in high interest, low cognitive overload, and positive learning transfer effect.

III. Results and Discussion

When someone who learned a block-based programming language begins learning a text-based programming language, they already have a basic programming logic. They can maximize learning transfer, naturally shifting from block-based programming without learning the basic logic or skills of text-based programming. Several strategies are needed to develop and implement an educational system that cultivates effective learning transfer from block-based to a text-based programming language.

A. *Differences between Block-based Programming Languages and Text-based Programming Languages*

The main differences between block-based and text-based programming languages are found in figurations and methods, which are further explained in the figures below.

1) *Difference in Figurations*

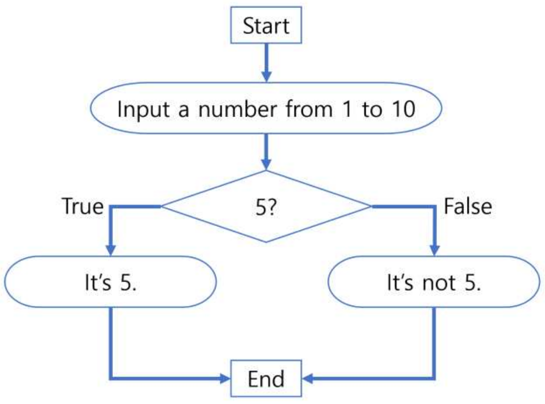
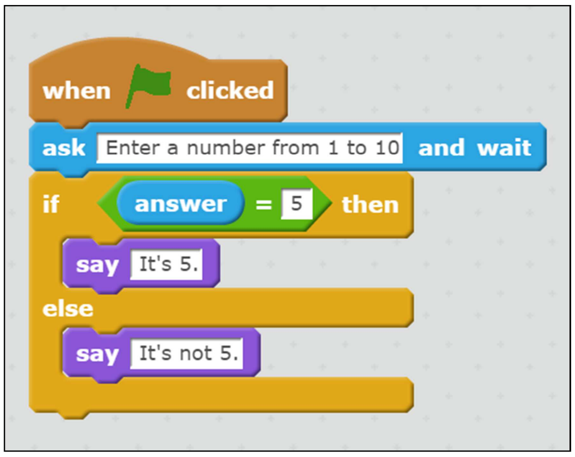
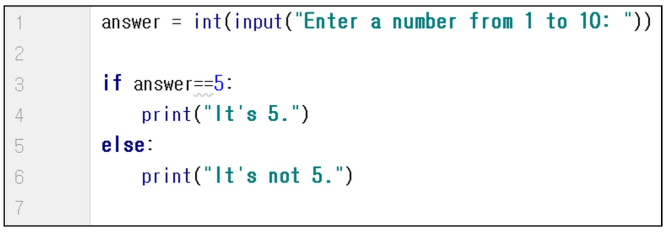


Fig. 4 If~Else statement algorithms



```
when [flag] clicked
ask [Enter a number from 1 to 10] and wait
if <(answer) = [5]> then
    say [It's 5.]
else
    say [It's not 5.]
```

Fig. 5 If~Else statement, Scratch 2.0 code



```
answer = int(input("Enter a number from 1 to 10: "))

if answer==5:
    print("It's 5.")
else:
    print("It's not 5.")

```

Fig. 6 If~Else statement, Python code

Figure 5 is coded with Scratch 2.0 and Figure 6 is coded with Python, both of which implement the algorithms of Figure 4 about the conditional statement If~Else. The two codes have the same logic and solution, but they have different figurations.

The Scratch 2.0 code puts condition blocks and combines execution blocks on the If~Else block. It is very intuitive to write a program, and it is very clear where to place the command blocks to do the programming. Python code, unlike the Scratch code, needs to declare variables, and all typing in text also requires indentation and use of colons. Students who learn programming only with Scratch will be confused by these differences in figuration when they do program in Python.

2) Difference in Methods

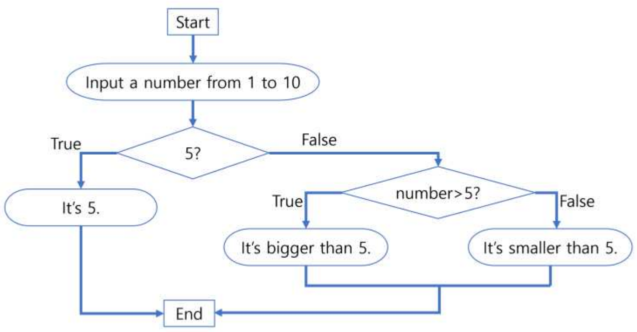
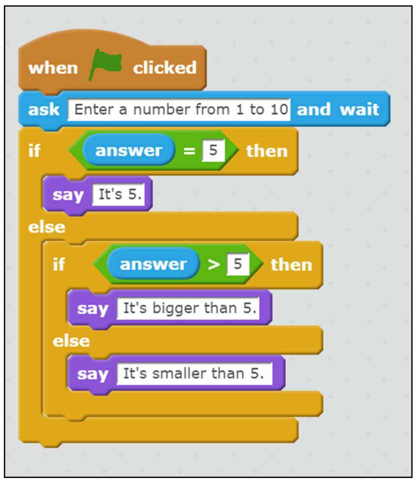
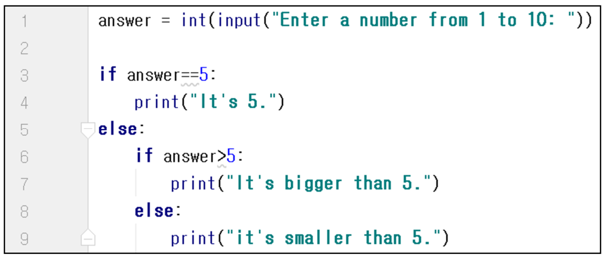


Fig. 7 Nested If statement algorithms



```
when 🏴 clicked
ask Enter a number from 1 to 10 and wait
if answer = 5 then
    say It's 5.
else
    if answer > 5 then
        say It's bigger than 5.
    else
        say It's smaller than 5.
```

Fig. 8 Nested If~Else statement Scratch 2.0 code



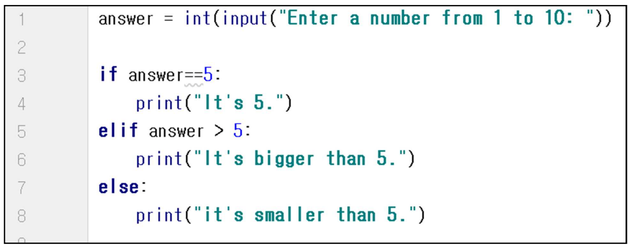
```python
answer = int(input("Enter a number from 1 to 10: "))

if answer==5:
    print("It's 5.")
else:
    if answer>5:
        print("It's bigger than 5.")
    else:
        print("it's smaller than 5.")
```

Fig. 9 Nested If~Else statement Python code

The difference in methods can create cognitive confusion between block-based and text-based programming languages. Figure 7 shows nested If statement algorithms. It can be coded like Figure 8 in Scratch or Figure 9 in Python.

As shown in Figure 5, a new If~Else block is placed in the Else part of the If~Else block. Similarly, in the Python code of Figure 6, the new If~Else statement is written in the Else section of the If~Else statement.



```python
answer = int(input("Enter a number from 1 to 10: "))

if answer==5:
    print("It's 5.")
elif answer > 5:
    print("It's bigger than 5.")
else:
    print("it's smaller than 5.")
```

Fig. 10 If~Elif~Else Python code

However, Python code could express an If~Elif~Else statement as seen in Figure 7, which is coded in a way that makes use of the Elif statement to check conditions sequentially in parallel. Students who learn only block-based programming languages like Scratch could experience cognitive confusions or cognitive overload when encountering the methods of text-based programming languages like Python, seen in Figure 7. They might not even infer this method because there are no Elif blocks in Scratch.

It might also be difficult for students to adapt to text-based programming language later if they do not understand these differences in method early on. Ironically, although the block-based programming language was developed and designed for programming education, these differences in methods may inhibit transferring experience from block-based languages to text-based language for learners.

B. Directions for the Designing Principle

1) Utilization of Advanced Organizers: According to Ausubel (1976), for learning to be meaningful, the teacher should provide a base that the learner can associate with the learning task, which is referred to as the advanced organizer. It is an organizer that plays an intermediary role in connecting content from one learning content to the next. It is sufficient for learning because it increases the motivation of learners, an internal variable of the learner, which could sustain interest [14]. By appropriately applying advanced organizers, students can more easily understand concepts and principles through the natural connection between prior knowledge of Scratch 2.0 and the new concepts of Python.

2) Problem Solving-based Learning Contents: Learning content should be focused on problem-solving. Various examples should be presented using simple grammar to compare how to solve problems in both languages [15]. Therefore, learning content should be related to the similarities of both Scratch 2.0 and Python. Likewise, it needs to consider the Game-based Bayesian Intelligent Tutoring System in the introductory course of training mode, because it could help the problem of learner losing motivation and enthusiasm when not being taught or interacted in a timely and can provide learners with a learning environment tailored to their individual needs [16].

3) Simple and Intuitive User Interface and Screen Layout: According to Kim's research (2013), the user interface (UI) should be visible and directly manipulatable, and graphic design should be of high quality [17]. Considering that most users who use this educational program are young novices, the UI should be intuitive and straightforward. Park and Kim's research suggests developing three steps of micro-learning contents: Intro (uploading video), Learning (adding interaction), and Organizing (organizing and summary). Although these steps are designed for micro-learning content, it also includes learning content [18]. This means that the learning content should be interactive and include various types of multimedia to maintain the learner's interest.

C. Design Proposal of the Screen Composition

Based on the above discussion, we propose screen compositions for two types of modes as follows. This is the core of the educational system.

1) Training Mode: The training mode should present a simple problem that can be implemented with Scratch. It should be structured in parts that can be practiced with Scratch and implemented in the same way with Python. Although not shown in Figure 8, a hint should appear when a mouse cursor hovers over the Scratch blocks that discusses the Python code to help the learner understand each function.

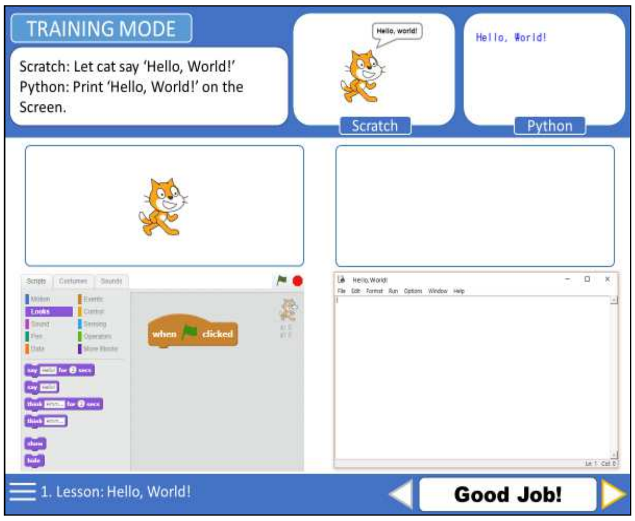


Fig. 11 Design proposal of the screen composition: Training mode

2) Practice Mode: In the practice mode, the learner can freely write a program with Scratch and Python, compare each, and practice the same result. The practice mode should provide a save function as well.

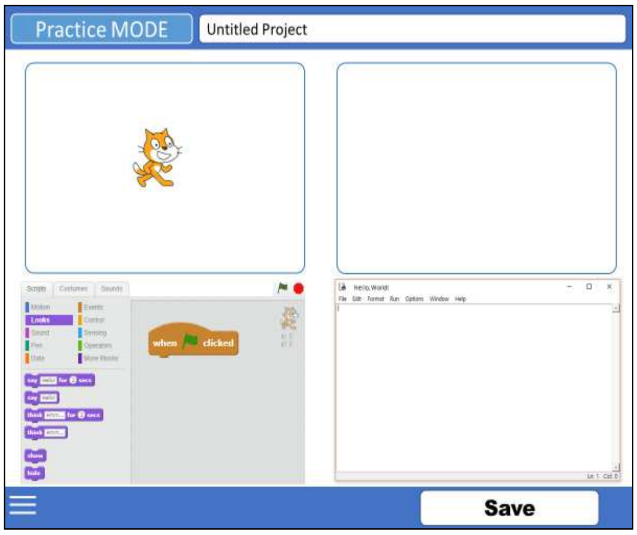


Fig. 12 Design proposal of the screen composition: Practice mode

IV. Conclusions

In this study, we designed and proposed an educational system that allows mixed learning transfer from block-based programming languages to text-based programming languages. We focused on Scratch 2.0 and Python.

Based on the literature, we found differences in figurations and methods between block-based programming languages and text-based programming languages that can result in cognitive confusions or cognitive overload for learners. Thus, we suggested a direction of designing principle: Utilization of advanced organizers, problem-solving based learning content, and simple and intuitive user interface and screen layout. Using this design, we developed and proposed two types of modes for the core of an educational system.

In future studies, it is necessary to implement this design and apply it in the educational field to verify its effectiveness. One of many things to consider when implementing this educational system is that it should be available online. Care should be taken when applying it to the education field as it should be suitable for all ages from elementary through secondary education. Additionally, its effectiveness should be demonstrated with statistically significant results.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. 2016R1A2B4010522).

References

[1] Yi, S.Y., Lee, Y.J.: The Development of Teachers' Training Course about Educational Programming Language to Enhance Informatics Teaching Efficacy for Elementary School Teachers. The Journal of the Korean Association of Computer Education, vol. 20(5), pp. 35-47. Seoul (2017)

[2] Shin, S.K., Bae, Y.K.: A Study on the Hierarchical Instructional System Design of Software Education by School System. Journal of The Korean Association of Information Education, vol. 19(4), pp. 533-544. Seoul (2015)

[3] Mayer, R. E.: Applying the science of learning. Pearson/Allyn & Bacon. Boston (2011)

[4] Thorndike, E. L., Woodworth, R. S.: The influence of improvement in one mental function upon the efficiency of other functions. Psychological Review, vol. 8(3) (1901)

[5] Mayer, Richard E., ed.: The Cambridge handbook of multimedia learning. Cambridge university press (2005)

[6] Seo, S.W., Nam, D.S., Lee, D.W.: The Effect of Computational Thinking Ability Using Text-based vs Visual-based Programming Language On Robot Programming Learning. Proceedings of the Korean Society of Computer Information Conference, vol. 18(2), pp. 457-462. Seoul (2010)

[7] An S.J., Seo, Y.M., Lee, Y.J.: A Review and Synthesis of Research in Educational Programming Language. Proceedings of the Korean Society of Computer Information Conference, vol. 20(1), pp. 139-142. Seoul (2012)

[8] Jeon, H.S., Jeong, J.K., Kim, S.S.: Problem Design & the Application of online judge to Basic C programming language Learning. Proceedings of the Korean Association of Computer Education Conference, vol. 18(1), pp. 291-294. Seoul (2014)

[9] Kwon, D.Y., Gil, H.M., Yeum, Y.C., Yoo, S.W., Kanemune, S., Kuno, Y., Lee, W.G.: Application and Evaluation of Object-Oriented Educational Programming Language "Dolittle" for Computer Science Education in Secondary Education. The Journal of Korean association of computer education, vol. 7(6), pp. 1-12. Seoul (2004).

[10] Heo, M.S., Kim, J.H., Lee, W.G.: Comparative Analysis of Programming Learning between Textual EPL and Visual EPL. Proceedings of the Korean Association of Computer Education Conference, vol. 13(1), pp. 123-127. Seoul (2009)

[11] Choi, Y.M., Kim, Y.C., Cho, S.H.: The Changing of Beginners Perception of the Programming after using Educational Programming Languages. Proceedings of the Korean Association of Computer Education Conference, vol. 20(2), pp.7-10. Seoul (2016)

[12] Park, J.S., Cho, S.B. The Effect of teaching Scratch in an introductory programming course. Journal of Digital Convergence, vol. 10(9), pp. 449-456. Seoul (2012)

[13] So, M.H., Kim, J.M.: Transference from learning block type programming to learning text type programming. The Journal of Korean association of computer education, vol. 19(6), pp. 55-68. Seoul (2016)

[14] Ausubel, D.P.: The psychology of meaningful verbal learning. Grune and Stratton, New York (1968)

[15] Park, M.S., Kim, J.H., Kim, T.Y.: Design of Multi-learning System of Programming Language-based Learning Transfer Theory. Proceedings of the Korean Association of Computer Education Conference, vol. 14(1). Pp. 211-216. Seoul (2010)

[16] Hooshyar, D., Ahmad, R.B., Wang, M., Yousefi, M., Fathi, M., Lim, H.: Development and Evaluation of a Game-Based Bayesian Intelligent Tutoring System for Teaching Programming. Journal of Educational Computing Research, 0735633117731872. (2017)

[17] Kim, H.J.: Development of Design Strategy of Content & User Interface for Digital textbook to achieve Smart Education: Through Comparative Analysis of Content & User Interface Design of e-textbook & Apple Digital textbook. Journal of Digital Design, vol. 13(1), pp. 161-171. Seoul (2013)

[18] Y. Park and Y. Kim, A design and Development of micro-Learning Content in e-Learning System. International Journal on Advanced Science Engineering Information Technology, vol. 8(1), pp. 56-61, 2018.

[19] Papert, S.: Mindstorms: children, computers, and powerful idea. Basic Books, New York (1980)

[20] Kim, J.H., Choe, H.J., Kim, T.Y.: The Effects of the Advance Organizer on Elementary School Students' Logical Thinking Ability and Self-Efficacy in Programming Class. JOURNAL OF The Korean Association of information Education, vol. 15(2), pp. 189-199. Seoul (2011)