International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 9942-9947

Development of Additional Functions in Scratch for Learning the Fundamentals of Object-oriented Technology

Sung Wan Kim*
**Division of Computer and Mechatronics Engineering, Sahmyook University*
Seoul 01795, Korea.

**Corresponding author: Sung Wan Kim, Ph.D.*

Abstract

With the rapid increase in interest in software education, block-type coding tools such as Scratch are actively being employed as educational programming languages. Recently, studies have been introduced on the learning of object-oriented concepts using Scratch. However, because Scratch is not a programming language that supports object-oriented programming, it is not sufficient to facilitate consistent and natural learning for beginners learning object-oriented concepts. In this paper, we design and propose extended functions to express the concepts of classes, objects, and inheritance in Scratch, which are the most fundamental aspects of object-oriented technology, and we describe an application example. As a result of evaluating the usefulness of the proposed functions through expert evaluation, we found that it can be useful in software education for elementary school students.

Keywords: Object-oriented Programming, Scratch, Software Education, Educational Programming Language

INTRODUCTION

As software is recognized as a future engine of the national growth, there is a growing interest in software education around the world. At the core of software education is creative thinking for the digital age, or in other words, computational thinking [1]. Computational thinking refers to the ability to discover the core principles of problem situations, simplify them, and solve them logically and efficiently [2]. Recently, coding education has received considerable attention as a practical tool for developing computational thinking. A typical tool for such coding education is Scratch, an educational programming language capable of block-type coding programming.

On the other hand, object-oriented programming technology is widely adopted for large-scale software development, owing to its advantage of enabling flexible program development [2, 3]. Recently, research has been proposed regarding object-oriented concept learning using Scratch [4, 5]. Scratch is not a programming tool that supports object-oriented concepts, and hence, it is not possible to accurately define class definitions, object creation, and inheritance. In this study, we present and analyze existing research results for object-oriented learning for low school-age learners, such as elementary and middle school students. More fundamentally, we design additional function elements for Scratch to apply object-oriented concepts and introduce application examples. Finally, we verify the usefulness of the proposed approach through expert evaluation.

SCRATCH AND OBJECT-ORIENTED TECHNOLOGY LEARNING

Scratch Programming Tool

Scratch is a popular educational programming language. Stage, sprites, scripts area, and blocks are used as components for writing a program. The stage is the place where the program output is executed and rendered. A sprite is a basic unit for executing a program. There are various kinds of script blocks that are predefined and provided for setting the values of sprites or controlling operations, and one selectively combines these to complete the implementation of a specific sprite. The default script blocks consist of 10 types: Motion, Looks, Sound, Pen, Data, Event, Control, Sensors, Operations, and More Blocks. More Blocks, provided from Scratch 2.0 version, allow the creation of blocks that can perform desired new functions by combining default blocks. That is, it is possible to create a user-defined block. Backpack is a personal repository for reusing frequently used script blocks.

Learning Object-oriented Programming

Object-oriented programming is widely adopted for large-scale software development, because of its high efficiency in software development encompassing productivity, reusability, and maintenance [3, 4]. Learning object-oriented concepts is one way to recognize and solve problem situations, as well as a tool to improve personal cognitive abilities. Thus, it can play an important role in learning for students of a low school age [6]. In addition, because object-oriented programming is used

in actual software development sites, a coding education based on object-oriented concepts is required [7]. The essential characteristics of object-oriented programming can be considered as the understanding of classes and objects through data abstraction, inheritance, polymorphism, encapsulation, and dynamic binding [3, 8]. Thus, the most fundamental object-oriented concepts for elementary and middle school students' coding education can be surmised to be the understanding of objects, the relation between classes and objects, and inheritance.

It is effective to perform object-oriented programming through unplugged learning. In [4, 5], an approach to learning object-oriented concepts based on five steps was proposed, as shown in <Fig. 1>.

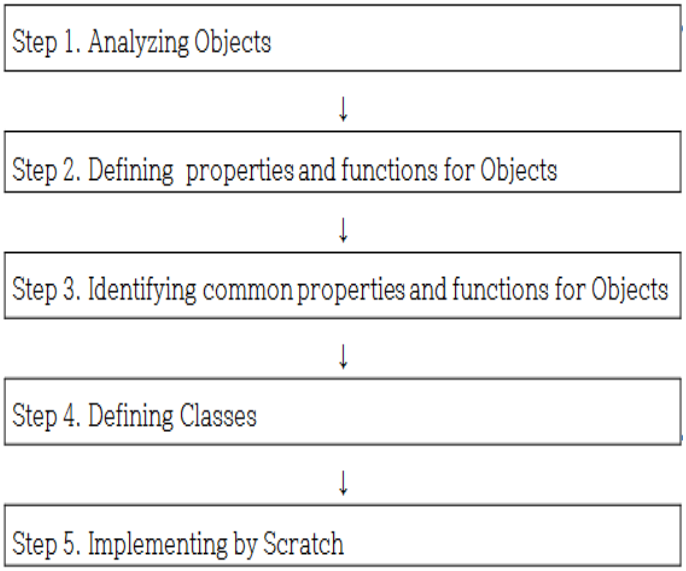

Figure 1. Five Steps for Learning Object-oriented Concepts

At each step, students conduct unplugged activities using a prepared activity form. A bottom-up approach is adopted to define a class after extracting the common properties and functions by analyzing individual objects. For example, in the class hierarchy of <Fig. 2>, we can see that the class Figure is defined, with color and position properties, and that the Bar, Ball, and Brick classes are represented as its child classes. In the final step, we actually implement the defined classes in Scratch. The left side of <Fig. 3> shows an implemented example of the class Figure which contains two properties created using More Blocks facilities. The right side of <Fig. 3> shows an example of the class "Bar", which includes two inherited properties from the class Figure, and an additional function "Move" of its own.

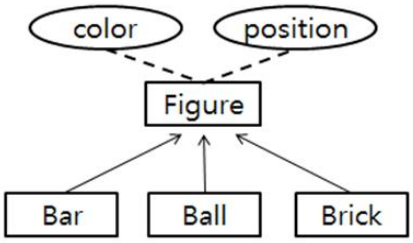

Figure 2. Class Hierarchy

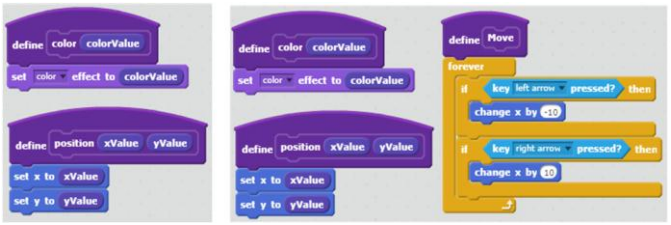

Figure 3. Implemented Class Figure (left) and Class Bar (right)

PROPOSED APPROACH AND EVALUATION

Analysis for the Previous Approach

Because Scratch is not a programming language that supports object-oriented concepts, authors of previous research have focused on a learning process centered on the concept of inheritance of common properties and functions included in the superclass, while adopting existing sprites in the role of classes or objects, without clearly distinguishing between objects and classes. In order to express the inheritance relationship, we have to duplicate the same script blocks for each sprite implemented for a class. For example, two More Blocks are created as the member properties of the class Figure on the right side of <Fig. 3>. However, because these More Blocks are for the Figure sprite, we must explicitly rewrite the same More Blocks each time we implement a child class, that is, the Bar, Ball, or Brick sprite.

In the Scratch 2.0 offline editor, it is possible to save specific frequently used script blocks in a personal repository (Backpack), and to apply them after dragging them from the repository when the same script blocks are written for new sprites. However, there are obvious limitations in inheritance learning using only More Blocks and the private repository. First, the purpose of using More Blocks is to simply copy and use script blocks to avoid the inconvenience of rewriting a script block, rather than taking an inheritance viewpoint. The inconsistency that occurs when expressing the concept of inheritance learned through unplugged activity using Scratch is likely to make it difficult for learners to understand object-

oriented concepts naturally and clearly. Second, if the definition of the superclass changes in the class hierarchy, then the contents stored in the private repository do not change, and so all script blocks that use the definition must be rewritten. Third, the scope of reuse of More Blocks is limited to one time (which may be a development error). That is, we can define a new More Blocks B using the previous More Blocks A, but if the additional block C is further defined using B, then A cannot be referred to. This means that we cannot define a class hierarchy of depth greater than two.

Designing Additional Functions

We aim to provide additional functions to naturally learn the relationship between class and object and the inheritance concept, which constitute the most fundamental aspects of object-oriented technology through Scratch. In addition, by applying the Scratch programming style, we reduce the learning-overhead on the newly added functions for the learner as much as possible. To this end, we design additional functional elements of scratch as follows.

First, we provide a template definition function. In this paper, we use the term 'template' instead of 'class.' For the convenience of users who are familiar with Scratch, we support user creation of user-defined templates, in a similar manner to how More Blocks are handled. When creating a template, various default script blocks, including More Blocks, can be combined to create properties or methods.

Second, we maintain template storage to store the created templates. They can be stored in the existing Backpack storage or in a separate space. A generated template can be shared or reused in multiple projects.

Third, a created template can be reused when another template is created, so that a template hierarchy can be generated. That is, when a new template is created, it inherits the entire contents of the parent template.

Fourth, in order to achieve the effect of creating an object instantiated by a specific template, an existing sprite or user-generated sprite can be instantiated as an object of a specific template. That is, an instantiated sprite of a specific template can be programmed by utilizing a member block defined in the template.

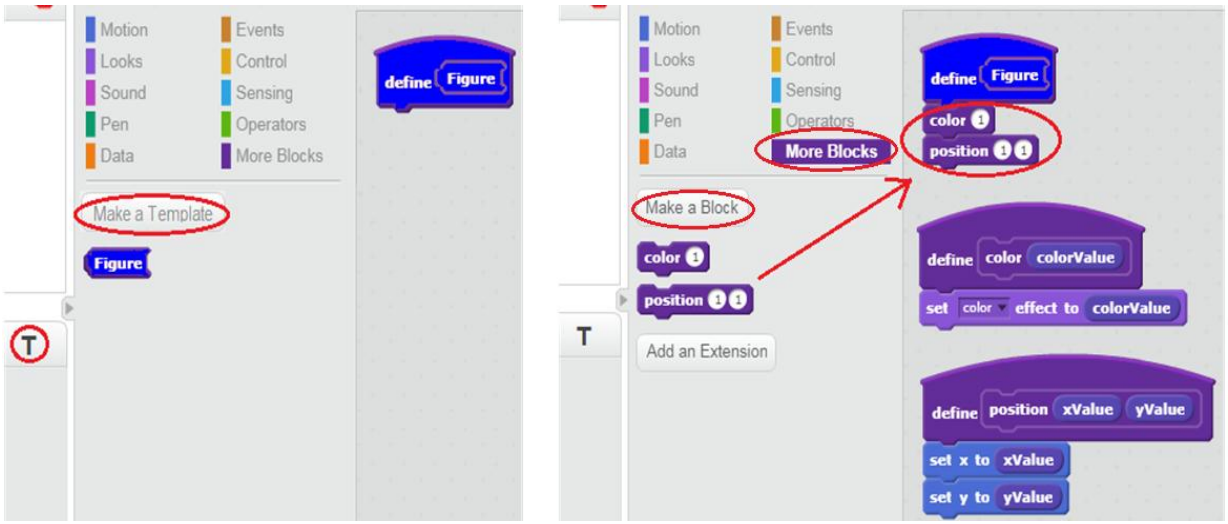

Figure 4. Creating a New Template

Figure 5. Template Inheritance

Application Example

The following are the series of assumptions in Scratch to support the functions proposed in the previous section, where we program the results analyzed in <Fig. 2>. <Fig. 4> shows the process of creating the template Figure using the template definition function. If the 'T' icon and 'Make a Template' button are selected, a separate window will pop up where a new template can be created. If the name of the template is entered, then a template with a blue block is created and placed in the scripts area. The generated template will also appear as a block icon at the bottom of the 'Make a Template' button. The right side of <Fig. 4> shows the process of adding property members to the template Figure. First, using More Blocks we create user-defined blocks for color and position properties, respectively. Then, we drag the block icons for the two added blocks and attach them as members of the template Figure, thus completing the creation of the template Figure.

<Fig. 5> shows the process of defining the template Bar, which is a child of the template Figure. To create the template Bar as a child of the template Figure, drag the block icon for the template Figure to the right of the template Bar placed at the scripts area. This means that the template Bar will inherit the members of the template Figure. As shown on the right side of <Fig. 5>, the template Bar automatically includes blocks for the color and position properties, and the shape of the block icon for the template Bar is combined with the block icon for the template Figure.

<Fig. 6> shows the final result for the template Bar after adding the method Move using More Blocks. Templates and user-defined blocks created in this manner may be stored in Backpack for future use. By repeating this process, we can create templates and define a template hierarchy to naturally express the object-oriented concept.

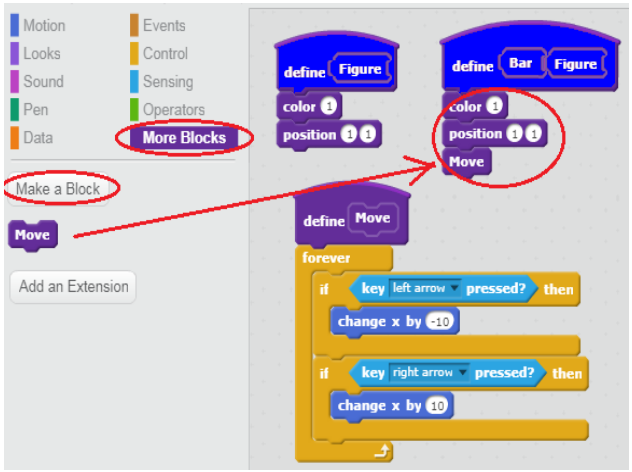

Figure 6. Creating the Bar Template

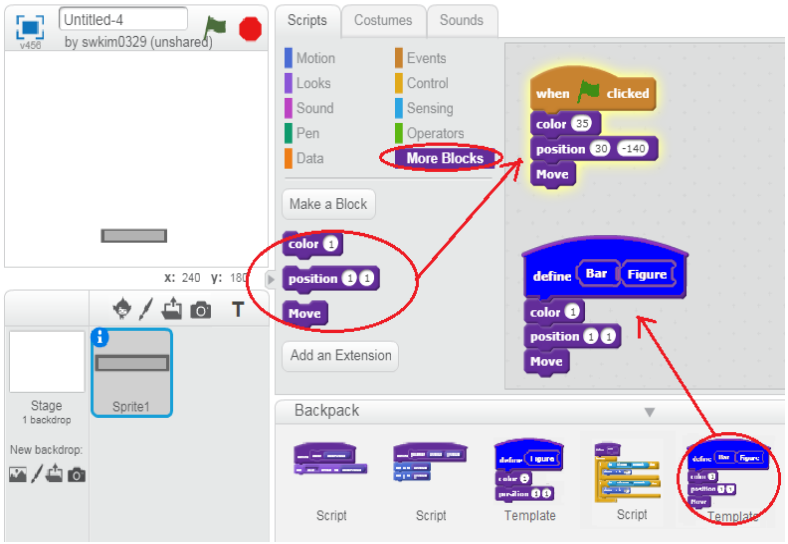

Figure 7. Instantiating a Sprite as an Object of the Template Bar

<Fig. 7> shows the process of instantiating an arbitrary sprite as an object of the template Bar. When we create a new sprite with a bar image and then drag the template Bar from the Backpack to the scripts area, the sprite is now instantiated as an object of the template Bar. As a result, block icons for the color and position properties and the method Move, which are members of the template Bar, will be included automatically in the More Blocks for the Bar sprite. The upper part of <Fig. 7> shows the action programmed using three member blocks when we click the Bar sprite. In this manner, the learner can program each class and class hierarchy defined through the unplugged activities in a visual form using Scratch, so that object-oriented concepts can be understood more naturally.

Expert Evaluation

The applicability and improvement of the designed functions are verified through an expert evaluation. We selected five teachers as experts, who have a high level of understanding of object-oriented technology and experience in computer programming education classes using Scratch for elementary school students. The expert evaluation was conducted by explaining the proposed function and the programming application example according to the scenario, and then obtaining responses to a questionnaire. The questionnaire was composed of eight items for the total of three areas. The experts provided answers according to a five-level Likert scale in one point intervals. The expert evaluation areas and results are shown in <Table 1>.

Table 1. Results of Expert Evaluation

Validity and Applicability of Proposed Functions	Educational effectiveness of the proposed functions	4.5
	Actual utilization of the proposed functions	4.4
Satisfaction of Implemented Functions	Features for creating templates	4.6
	Functions for defining template hierarchy	4.6
	Ability to instantiate sprites	4.4
Intuitiveness and Convenience of Operations	Intuitiveness and convenience for template creation	4.8
	Intuitiveness and convenience for template hierarchy definition	4.6
	Intuitiveness and convenience for instantiating sprites	4.2

The results of the expert evaluation show that the overall satisfaction level with the additional functions proposed in this paper is high. And it can be surmised that this is highly likely to be adopted as a tool for object-oriented concept learning when the proposed functions are implemented actually. In particular, it was evaluated that the proposed functions provide sufficient support for learning the fundamentals of object-oriented concepts consistently. Furthermore, it was evaluated that this can be utilized without any additional burden on learning, because a program can be constructed by using the same programming style in Scratch.

CONCLUSION

Recently, software education has attracted considerable interest worldwide as a tool for generating computational thinking. In particular, block-type coding programming tools, such as Scratch, are widely adopted as educational programming languages. Recently, research has been introduced on learning basic object-oriented concepts using Scratch. However, because Scratch does not reflect object-oriented concepts, it is not sufficient to facilitate consistent learning for beginners learning object-oriented concepts. In this paper, we presented additional functions to enable natural learning of object-oriented concepts in Scratch, and examined an application example. Finally, through an expert evaluation, we confirmed that the additional functions proposed in this paper can be useful for learning fundamentals of object-oriented technology, without significantly increasing the learning overhead.

REFERENCES

[1] Software Policy & Research Institute, Software-oriented Society - Meaning and Corresponding Direction. *SPRi Issue Report* 2014-003 (2014).

[2] Object-oriented Programing, *https://en.wikipedia.org /wiki/Object-oriented_programming*.

[3] T. Kim, Object-oriented Story of K Professor. *Life and Power Press* (2005) 29-35.

[4] Y. Kim et al., Design of Game based Educational Contents for Learning Object-Oriented Concepts. *Proc. of the Summer Conference of the Korean Association of Computer Education*, 17(2) (2013) 35-39.

[5] Y. Kim et al., A Case Study on Course Game Based Elements for Learning Object-Oriented Concept. *J. of Korean Association of Computer Education*, 17(5) (2014) 1-13.

[6] J. Choi, A Study on Object-Oriented Concept Learning Using Storytelling. *Master's Thesis, Korea University Graduate School, Seoul, KOERA*, (2011).

[7] T. Hong, A Study on Object-oriented Programming

Education for Improving Logical Thinking Ability of Elementary School Students. *Master's Thesis, Catholic University Graduate School, Buchoen, Korea*, (2007).

[8] Nelson, H. J., Armstrong, D. J., and Nelson, K. M. Patterns of Transition: The Shift from Traditional to Object-oriented Development. *J. of Management Information Systems*, 25(4) (2009) 271-298.