ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 8(S1),89-96, January 2015* 10.17485/ijst/2015/v8iS1/57701

Assessing the Usefulness of Object-based Programming Education using Arduino

YunJae Jang[1*], WonGyu Lee[2] and JaMee Kim[2]

[1]Department of Computer Science Education, Graduate School, Korea University, Seoul, South Korea; yunjae.jang@inc.korea.ac.kr
[2]Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul, South Korea; lee@inc.korea.ac.kr, jamee.kim@inc.korea.ac.kr

Abstract

The purpose of this study is to use Squeak etoys and Arduino to design education activities so that middle school students can learn the object-oriented paradigm, and assess how usable the education activities are at the level of middle school students. As for education activities, learning methods, such as imitating, modifying and creating, were introduced. Usefulness assessment was divided into a cognitive domain and a definitional domain before the assessment tools were developed. An object-based programming class for 11 middle school students was then conducted and the results analysed. Positive results were confirmed in both the cognitive domain and the definitional domain. The results of this study are meaningful in that the possibility of object-oriented education was discovered for middle school students as well. Also, considering the sequence of object-oriented education, it was found that it is necessary to develop object-oriented programming education at the level of both elementary and high schools.

Keywords: Object-oriented Paradigm, Object-based Programming Education, Usefulness Assessment

1. Introduction

In programming languages, which have continuously developed, the paradigm has been shifting from procedure-oriented languages to object-oriented languages. Java, one of object-oriented languages, is the programming language most frequently used around the world[1]. Currently, programming languages, which started from the procedure-oriented paradigm, also support the object-oriented paradigm. Due to the paradigm shift in programming languages, object-oriented programming languages are included in programming-related education courses at universities[2].

Object-oriented programming, which is leading the paradigm shift, is too difficult for novices to learn at universities or high schools despite its numerous advantages[3]. Most education about object-oriented programming languages covers everything from design to implementation of object-oriented programming languages over a one-year period. As students learning programming languages for the first time have difficulties[4], students who are learning object-oriented programming language are experiencing similar difficulties[5]. For instance, object-oriented programming learners must first understand the object-oriented paradigm[6,7].

Researchers who realized the importance of understanding the object-oriented paradigm suggested that object-oriented analysis and design should be taught first before coding[3,6]. In other words, learners need to first understand the characteristics and overall usage of the

* *Author for correspondence*

language before learning a new language. Interestingly and in order to facilitate this, studies on using card-type games to teach the object-oriented paradigm first were conducted[8,9]. In object-oriented programming courses, object-oriented design is a matter of perspective from which to view the surrounding environment. Perspectives can be sufficiently learned before professional programming techniques are learned. As the object-oriented paradigm is similar to the perspective from which people understand the surrounding environment, it is possible in the basic thinking process that learners can learn before acquiring professional knowledge[10]. Accordingly, it is necessary to adopt a learning method to facilitate initial learning of the basic thinking process for elementary school grades and gradually increasing levels. Programming languages and education methods are necessary so that an appropriate object-oriented paradigm can be learned by middle school students. If learners' cognitive burden can be reduced, novices will be able to easily absorb programming education in general. Object-based programming languages in the object-oriented paradigm were affected by the learning-by-example type of learning method that can reduce learners' cognitive burden[11]. Object-based languages copy objects as prototypes. For example, Squeak Etoys, an object-based programming language, provides a tile-based visual coding method as well as a prototype method. The visual method enables novices to easily learn programming[12].

This study used both Squeak Etoys and Arduino to design educational activities so that the object-oriented paradigm can be learned at an early school age. The purpose of this study is to assess the usefulness of educational activities that were conducted in a way suitable for the grade level of students.

1.1 Research Question

The overarching research question of this study is, "Will object-based programming education activities utilizing Arduino be useful to learners?" More specifically, it is whether it will help learners learn the object concept, whether it satisfies learners' expectations about learning, and whether it triggers learning engagement and maintains learning endurability. These research questions were used to check whether the knowledge necessary for learning object-oriented programming can be obtained.

2. Background

2.1 Object-based Programming and Physical Etoys

Object-oriented programming languages can be classified into class-based languages and object-based languages. They originally started with a programming language named Self, and is supported in languages such as Javascript and Lua. Object-based programming refers to a programming environment that does not support succession or polymorphism in object-oriented programming, and can use limited types of objects[13]. Also, in a similar sense, it is also known as prototype-based programming, and refers to programming languages that can use the operating mode of objects through the duplication process as they do not have class and use objects as prototypes[14].

Physical Etoys is an object-based programming environment that novices can easily use [15]. Physical Etoys can put together programming codes in a tile script mode by adding objects, which can be connected to Arduino or Etoys, which are educational programming environments. It is possible to express the relationship between objects necessary for solving problems by using various component objects.

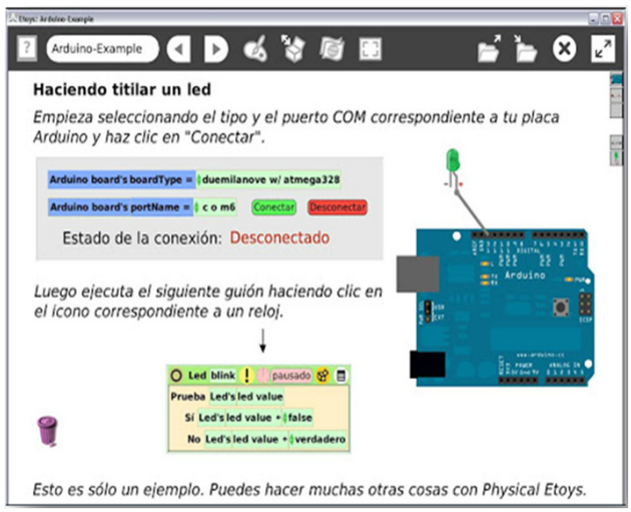

Figure 1. Physical etoys interface.

If Physical Etoys is used for learning, it is possible to reduce errors in connecting components by connecting components that can be connected in Physical Etoys first

before actually connecting Arduino and components. It is also advantageous for adding multimedia objects, provided by Etoys, to create rich projects.

2.2 Arduino

Arduino is a single board micro controller invented by the Interaction Design Institute Ivrea in Ivrea, Italy[16]. Arduino can be used to control devices like LEDs and motors by receiving data from input devices like illumination and switches sensors, and processing it through its micro controller.

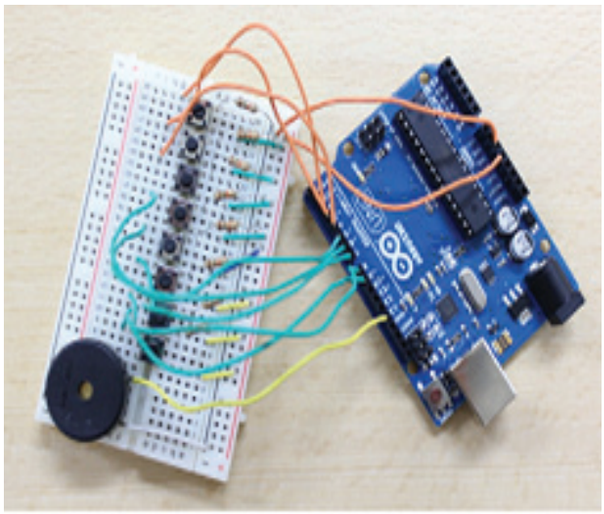

Figure 2. Arduino board.

Arduino can be used to create various types of physical computing products that can interact with the surrounding environment. Arduino's advantages are low price, small size, open source-based scalability, and the ability to quickly make prototypes. Arduino quickly became popular among media artists as a tool for creating interactive works of art at a low cost, but as it was developed and became widespread among professional developers or geeks, its popularity also spread among the DIY (Do it yourself) and maker movement communities. As a result, Arduino has recently been influencing industries like 3D printers and IoT (Internet of Things), and affecting programming education and robot education as well. The programming environment for Arduino control is Arduino IDE. One of the popular features of Arduino is that it is possible to utilise the Arduino control library in various programming languages like C, Java, Python and Processing[17].

2.3 Usefulness Assessment

Janet Read and MacFarlane (2000) developed the Fun Toolkit as a tool for collecting opinions on technology utilization education and usefulness data from young learners[18]. Read and MacFarlane introduced the 1st version in 2000, the 2nd version in 2002, and the 3rd version in 2008[18-20].

The Fun Toolkit measures learners' expectations, engagement and endurability to measure usefulness of education by measuring how fun the education was to young learners[18]. The tool for measuring expectations is the "Smileyometer" which is a 5-point Likert scale that uses five smileys especially designed for children. This tool asks, "How high are your expectations?" about educational activities to which learners choose one of the illustrations corresponding to "awful," "not very good," "good," "really good" or "brilliant." The Fun Sorter is the tool for measuring the relative fun of learning activities. This tool lists the learning activities of learners from best to worst in response to questions like "worked the best" and "liked the most." Lastly, the tool for measuring endurability is the Again-Again table. The Again-Again table asks "Would you like to do this again?" with regard to educational activities (activity or product) to which learners choose "Yes," "Maybe" or "No." This tool's advantage is that learners can check the endurability of their learning based on quick answers without any cognitive burden. The advantage of the Fun Toolkit is that young learners can easily understand questions and quickly answer surveys.

3. Methodology

3.1 Research Design

The design of this study is the one-group pretest-posttest design.

3.2 Participants

Twelve freshmen and juniors in middle school were recruited in advance for participation in classes. Eleven of them were male and one was a female. The classes were given from May 19, 2014 to May 30, 2014.

3.3 Programming Class Design

The object-based programming class model is as follows:

Table 1. Learning type

Learning type	Explain
Learning by Imitating	Learners imitate examples of object-based programming to learn about objects and how to use them.
Learning by Modifying	Based on the basic examples of object-based programming, learners modify the attributes and functions of objects, then add similar objects for programming.
Learning by Creating	Learners utilise the programming examples they have imitated and modified so far to design and develop new programs.

The object-based programming class was given for 2 hours a day for a total of 12 hours. The contents of each class are shown below:

3.4 Instruments

The tool for assessing the usefulness of the object-based programming class consists of the cognitive domain

Table 2. Activity plan

Day	Learning type	Contents	Hour
1~3	Learning by Imitating	Outputting (LED) Basics of electronic circuits (output)	2
		Concept of objects and relationship Outputting (LED) Making traffic lights	2
		Using a button to control traffic lights Basics of electronic circuits (input)	2
4	Learning by modifying	Utilising sensors (light, temperature) Etoys multimedia object	2
5,6	Learning by Creating	Project Design	2
		Project Development	2

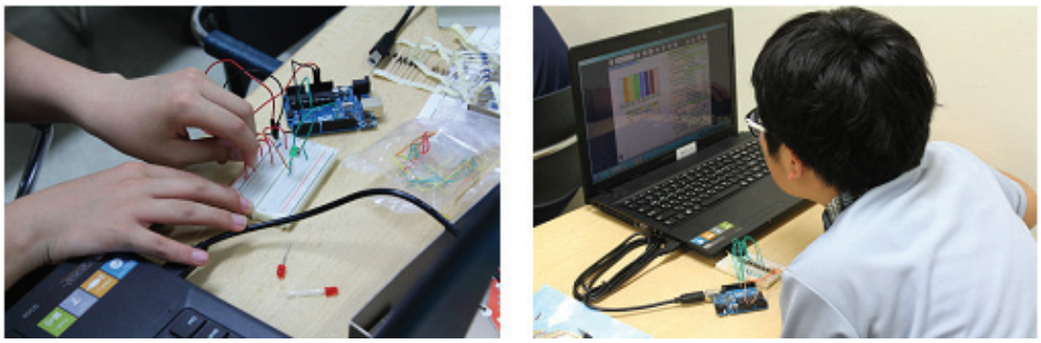

Figure 3. Activity of object-based programming class.

and the affective domain. For the cognitive domain, researchers developed preliminary and follow-up questions in a way fit for the objective of the object-based programming class. For the affective domain, the fun toolkit for measuring the usefulness of the class based on the element of fun was reconfigured to fit the level of middle school students and the contents of the class.

The cognitive domain was divided into Learning Achievement and Learning Endurability.

Learning Achievement assesses the students'

Table 3. Cognitive domain assessment method

Sub domain	Assessment method
Learning Achievement	1) Abstraction concept: whether the definition of abstraction is understood 2) Object relationship (1): whether the inclusion relation between higher objects and lower objects is classified 3) Classification of object attributes and methods: whether the concepts of attributes and methods are classified 4) Object relationship (2): whether the roles of objects are classified in the system
Learning Endurability	1) Recording what learners remember about what they learned as keywords 2) Picking two of these keywords and explaining them

understanding of the attributes, functions and relationships between objects, and the objectives of the object-based programming class by asking questions. There are a total of four questions assessing the object concept. Two sets of questions for assessing the object concept were developed so that they could be used for pre-testing and post-testing.

Learning endurability is the degree to which students could remember the contents they enjoyed. To asses this, this study measured how much of the object-based programming class the course learners remembered one month after it finished, which is long enough for memory to decay.

The subdomains of the affective domain are Learning Expectation, Learning Engagement and Learning Endurability.

Table 4. Affective domain assessment method

Sub domain	Assessment method
Learning Expectation	1) Expectations for today's class 2) Expectations for next class
Learning Engagement	The observation teaching assistant observes learners' positive and negative attitudes in class.
Learning Endurability	1) Do you want to learn it again? 2) Do you want to learn more in-depth contents?

Learning Expectation refers to how much the learners expected from the classes. Before and after each class the learners were surveyed for their expectations on a scale of 1 to 10 with a higher score indicating higher expectations.

Learning Engagement measures how actively the learners participated in the class. The teaching assistant was assigned the task of observing the positive and negative attitudes of learners and measuring them. A positive attitude means that learners asked questions related to the class and/or helped other learners frequently. A negative attitude means that learners chatted off-topic with other learners, and/or did not pay attention to the class, or otherwise did not show any interest in the class. In this study, two teaching assistants observed 6 learners each, and measured their learning engagement.

Learning Endurability measures how much learners want to learn again what the Fun Toolkit suggests. To measure learners' learning endurability, this study measured how much learners want to learn again the contents of the Fun Toolkit's Again-Again Table and in-depth contents on a 5-point Likert scale.

4. Result

In this study, 12 learners participated in a special class, but the results data of only 11 learners who participated in all classes was analysed.

4.1 Cognitive Domain

4.1.1 Learning Achievement

Before and after each of the object-based programming classes, learners answered questions to help assess their learning of the concepts involved. A total of 5 questions were asked to assess the object-based programming concepts. Each question has 5 points and the total score being 20 points. The average score for the pre-test was 8.18 while the average score for the post-test was 14.55 points, an increase of 6.36 points.

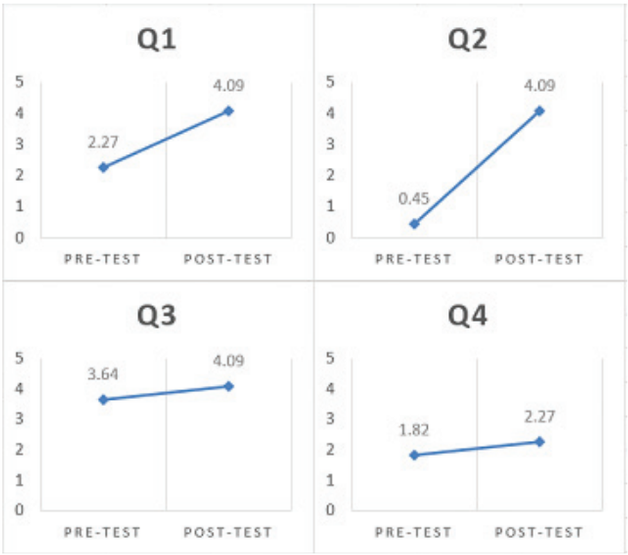

Figure 4. Difference in the average scores of the Pre-test and Post-test by question.

4.1.2 Learning Endurability

To assess learning endurability in the cognitive domain, this study measured how many keywords learners remembered one month after the class, and if they can explain certain keywords well. On average learners remembered 7.71 keywords among which they remembered, again on average, 1.14 keywords directly related to the objective of the object-based programming class, such as object and attribute. Learners corectly explained all the keywords they selected. In particular, all of them were able to explain objects and the attributes of the objects correctly.

4.2 Affective Domain

Learning expectation, learning engagement and learning endurability were assessed during and after each of the object-based programming classes.

4.2.1 Learning Expectation

To measure learning expectation, expectations about the class were measured before each class, and expectations about the next class were measured after each class. The degree of expectations was measured on a 10-point scale. The results from a total of 12 learners were averaged as follows:

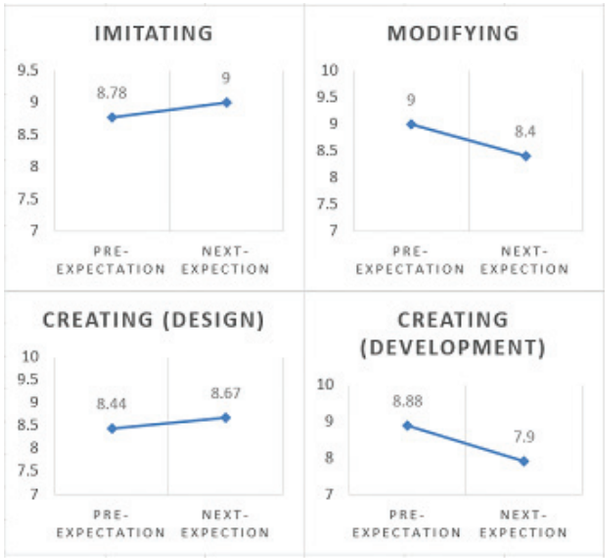

Figure 6. Changes in learning expectation by stage of class.

Timestamp	Q1	Q2
7/7/2014 16:57:37	LED, 버튼, 객체, 속성, 아두이노,	객체: 어떤 물체들이라는말이다 (ex: 건전지 전선, 등) 속성:그물체가 가지는 성질이나 특징(ex 건전지의 속성은 전기를공급해준다는 것이다.)
7/8/2014 11:29:01	아두이노와 피지컬이토이로 기계 만들기, 아두이노, led, 피지컬이토이	아두이노: 로봇의 몸체(하드웨어적) 피지컬이토이: 아두이노 로봇에 프로그래밍을 해 주는 프로그램
7/9/2014 22:03:37	아두이노, 개체,기능, Led, 스위치	Led, 빛을 내는 개체 스위치, 눌러서 신호를 보내는 개체
7/12/2014 19:39:10	아두이노, 브레드 보드(빵판), 푸시 버튼, 빛 센서, 열 센서, 속성, 기능, LED, 전선, 아두이노 인젝션, 입력, 출력, 전류, 아두이노 프로젝트, 저항, 저항값, 트랜지스터, 자신만의 창작 아두이노 프로젝트, 출력 장치가 작동하는 방법 및 과정	아두이노: 오픈소스를 기반으로 한 단일 보드 마이크로컨트롤러이고 보드 안에 CPU가 있어 다른 출력장치를 움직이게 하는 메인 보드 역할을 한다. 아두이노 프로젝트: 아두이노가 브레드 보드의 출력장치를 움직이게 하는 정보등을 만들게 하거나 동시에 컴퓨터까지 같이 제어하는 프로젝트
7/13/2014 16:45:55	아두이노,출력,입력,programing,LED, 등등	입력 : CPU에 어떤 특정한 값을 들여보내는 행위. (ex. 버튼, ultrasonic 센서, 적외선 센서 등등) 출력 : CPU에 입력을 했을 시에 일정한 값이 나오는 것. (ex. LED, 모터 등등)
7/14/2014 14:47:54	객체의 정의, 프로그래밍, 아두이노, 피지컬이토이 등등	객체: 물체나 시스템을 이루는 요소 프로그래밍: 컴퓨터에 명령을을 내리는 언어를 만드는 일
7/14/2014 18:53:11	아두이노, 이토이, LED, 빛 센서, 온도 센서, 입력 장치, 출력 장치, 기능, 속성, 버튼, 프로젝트, 발명품	LED는 빛을 내는 것으로 거의 모든 과정에 LED를 사용했다. 프로젝트로는 우리가 발명품을 만들었는데, 이 수업의 핵심이었다고 생각한다. 앞에서 거쳤던 과정들이 이 프로젝트를 하기 위해 수업 받았던 것 같다.

Figure 5. Learning endurability survey.

This study measured middle school students' ability to learn object concepts. Among the object-based programming classes, the pre-class expectations were higher than the expectations about the next class in the imitation and project design classes. For the imitation class, expectations about utilizing the examples they learned and expectations about actual development after individual project design grew higher. However, the score for expectations about the next class declined for the example utilization class.

4.2.2 Learning Engagement

To measure learning engagement, two teaching assistants observed the class and recorded the frequency of positive and negative attitudes of the learners during each class. They found that, in the modifying class on day 4, learners' learning engagement was highest, whereas in the project development class on day 6, learning engagement was lowest. In the modifying class, learners use the examples they learned to modify attributes and functions, and in the class for adding new objects, learners' interest and curiosity were stimulated. In this course, it seems that learners's positive attitude was reinforced through various kinds of questions and assistance among fellow students.

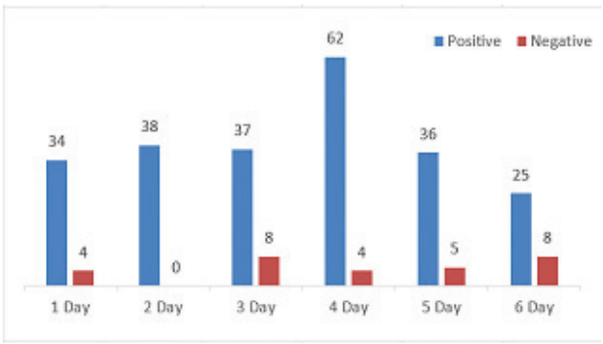

Figure 7. Frequency of learning engagement measurements by class day.

4.2.3 Learning Endurability

To measure learning endurability one month after the course, questions about whether learners want to learn the educational contents again were asked. Out of a possible 5 points, the average score was 4.43. Learners chose items (I want to learn again) above 4 points. The average score on the question asking whether they want to learn more in-depth contents was 4.71 points.

5. Discussion and Conclusion

For this study an object-based programming class for teaching the object concept to middle school students was designed and the usefulness of the class was assessed. To this end, first, the object-based programming class was designed in such a way as to motivate learners and reflect the effective programming class model. Next, the class was given. Then the usefulness of the class was assessed through direct observation and both a pre-test and post-test.

The results of this study can be summarized as follows: First, the object-based programming class was thought to be useful in cognitive terms. To assess usefulness in cognitive terms, this study measured how much improvement learners showed with regard to the assessment questions before and after each class, and how much they remembered what they learned one month after the class. The results showed a 6.36 point average increase from the pre-test to post-test. Also, one month after the course learners were able to remember, on average, 7.71 keywords, and correctly explain 2 of them.

Second, the object-based programming class turned out to be useful in definitional terms. To assess usefulness in definitional terms, learning expectation, learning engagement and learning endurability were measured. On a 10-point scale, pre-class expectations were 8.77 points on average, and expectations about the next class were 8.49 points on average. In the imitating class and the project design class, expectations about the next class were higher than the pre-class expectations. To measure learning engagement, two teaching assistants observed learners in class who found that a positive attitude was observed 38.7 times on average, while a negative attitude was observed 4.8 times on average. To measure learning endurability, learners were asked one month after the class whether (1) they wanted to learn the class again and (2) whether they wanted to learn more in-depth contents. The follow up session was conducted using a 5-point scale. The average score for the first question about whether they wanted to learn again what they had learned was 4.43 whereas the score for the second question concerning learning more in-depth contents was 4.71 on average. In other words, students were very positive about this class.

In programming education, object-oriented programming education is given only to college students or high school students. Looking at previous studies, most

studies suggested effective methods of object-oriented education for college students or developed programming languages fit for education[8,9,21]. However, this study found that object-oriented programming education requires an understanding of the basic concept of objects and learning at a high level of abstraction. Accordingly, this study is meaningful in that it found that it is possible for students to sufficiently learn object-oriented programming at an early age.

Based on the results of this study, it will be necessary to ensure that more middle school students can receive object-based programming education. Considering the sequence of progression required it will be necessary to develop object-oriented programming education courses for different levels ranging from elementary schools to high schools.

6. Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2012R1A1A2043389).

7. References

1. TIOBE Index. 2014 Aug. Available from: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.
2. ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer Science Curricula. 2013
3. Wei F, Moritz SH, Parvez SM, Blank GD. A student model for object-oriented design and programming. J of Comp Sci in Coll. 2005; 20(5):260–73.
4. Kelleher C, Pausch R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. Comput Surv (CSUR). 2005; 37(2):83–137.
5. Zschaler S, Demuth B, Schmitz L. Salespoint: A Java framework for teaching object-oriented software development. Sci Comput Program. 2014; 79:189–203.
6. Cooper S, Dann W, Pausch R. Teaching objects-first in introductory computer science. ACM SIGCSE Bulletin. 2003; 35(1):191–5.
7. Thota N, Whitfield R. Holistic approach to learning and teaching introductory object-oriented programming. Comput Sci Educ. 2010; 20(2):103–27.
8. Beck K, Cunningham W. A laboratory for teaching object oriented thinking. ACM Sigplan Not. 1989; 24(10):1–6.
9. Kim S, Choi S, Jang H, Kwon D, Yeum Y, Lee W. Smalltalk card game for learning object-oriented thinking in an evolutionary way. OOPSLA '06 Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications; 2006. p. 683–4.
10. Kay AC. History of Programming languages, 2. The Early History of Smalltalk, T. J. Bergin Jr and R. G. Gibson Jr, Eds. New York, NY, USA: ACM; 1996. p. 511–98.
11. Kay A. Children Learning by Doing Squeak Etoys on the OLPC XO. VPRI Research Note. 2007; 1–51.
12. Kay A. Squeak Etoys, Children and Learning. VPRI Research Note. 2005. p. 1–8.
13. Object-based language. 2014 Aug. Available from: http://en.wikipedia.org/wiki/Object-based_language.
14. Prototype-based programming. 2014 Aug. Available from: http://en.wikipedia.org/wiki/Prototype-based_programming.
15. Physical Etoys. 2014 Aug. Available from: http://tecnodacta.com.ar/gira/projects/physical-etoys/.
16. Arduino. 2014 Aug . Available from: http://en.wikipedia.org/wiki/Arduino.
17. Interfacing with software. 2014 Aug. Available from: http://playground.arduino.cc/Main/InterfacingWithSoftware.
18. Read JC, MacFarlane SJ. Measuring fun. Computers and Fun. York, England: 2000.
19. Read J, MacFarlane S, Casey C. Endurability, engagement and expectations: Measuring children's fun. Proceeding of the international workshop Interaction Design and Children. 2002.
20. Read JC. Validating the Fun Toolkit: an instrument for measuring children's opinions of technology. Cognit Tech Work. 2008; 10(2):119–28.
21. Kölling M. The Greenfoot Programming Environment. ACM Trans on Comp Educ. 2010; 10(4):1–21.