Modularizing Scratch Code to Develop Interactive Media Content

Looi Qin En

***Abstract*—Scratch has widely been recognized for its ability to teach children programming skills and helps to develop self-confidence. This study proposes another target group of audience who can capitalize on Scratch to reap multifold benefits—educators. Educators have been reluctant towards adopting new pedagogies to teach because of the difficulties faced when learning the advanced technologies the pedagogies are built on. By using modularized Scratch programming code templates, educators can develop a wide variety of interactive media content, ranging from games to interactive art, in order to better engage their students and increase learning effectiveness. This study demonstrates how modularized Scratch programming is easy to learn and produces interactive media reflecting the use of new education pedagogies. We created a set of Scratch code modules for the development of computer games to show how the modules can be used to develop a variety of educational games, including but not limited to business simulation games and adventure games to meet various learning objectives.**

***Index Terms*—Modularization, Interactive Media Content, Scratch Programming, Media-rich Graphic User Interface, Learning Effectiveness**

I. Introduction

SCRATCH is a media-rich programming environment developed by the Massachusetts Institute of Technology (MIT) Media Lab developed in 2007 [1] to encourage programming regardless of background and prior programming experience [2].

Scratch was developed with the main objectives of enhancing programming literacy amongst youths, especially those in less economically affluent communities. Through a simple-to-learn graphic user interface (GUI), Scratch prides itself as an effective platform to introduce programming to the young. Beyond the technical skills learnt, Scratch hopes to teach the young problem solving skills and boost their self-confidence [3].

Whilst Scratch's target audience is the young, this paper suggests another target audience who can benefit from Scratch's intuitive and easy-to-learn features—educators. This is achieved using a programming technique known as code modularization. Code modularization involves a variety of code blocks divided according to the programming objectives. Earlier studies have popularized this technique, applying it to thousands of lines of codes in legacy systems [4]. We will demonstrate how code modularization can be applied to comparatively simpler programming using Scratch.

The purpose of this study is to show how educators can develop *interactive media content* using Scratch to engage the students in the classroom using modularized Scratch codes. *Interactive media content* is defined as media developed from new ICT technologies, including but not limited to: computer games, animation, videos and interactive art. In this study, all interactive media content are products developed from Scratch.

Earlier works have re-affirmed the value of meaningful and motivating interactive learning tools in classroom teaching [5]. Hence, by using the interactive media content as teaching aids to complement traditional teaching methods, teachers can engage their students more in the learning process and thus, increase learning effectiveness.

There have been many suggested approaches to implement such new pedagogies in the teaching of specific subjects such as physics [6] and mathematics [7], but this study aims to provide a more generic approach towards introducing new pedagogies.

By incorporating the code modularization technique used in software engineering, this new pedagogical approach can be easily developed by educators to customize to their own needs. Through the explanation of how modularization of Scratch programming code is executed as well as a case study of how the modularization was implemented to develop interactive media content at an educational enterprise, we hope to convince educators that development of interactive media content using Scratch is effective and easy to accomplish.

II. Scratch In Education

The application of Scratch in education has been widely discussed, with previous studies attempting to extend its influence beyond young children. In one study [8], Scratch was deployed in higher education, where it was used as an introduction to programming in Harvard summer school's Computer Science S-1: Great Ideas in Computer Science.

Manuscript received November 24, 2010; revised December 31, 2010. This work was supported in part by Hwa Chong Institution, Singapore, the Ministry of Education, Singapore, the Massachusetts Institute of Technology, USA and the Center for Excellence in Education, USA under the Research Science Institute 2010 summer program.

Looi Qin En is a student researcher at Hwa Chong Institution (Singapore), 661 Bukit Timah Road, Singapore 269734 (phone: 65-97565638; e-mail: stuqel@i2r.a-star.edu.sg).

Results from this study were promising as 76% of the students felt that they had benefitted from Scratch, especially those with no prior programming experience.

This study aims to further extend Scratch's influence beyond the student to the educator. If Scratch is an effective learning tool, then it is important for educators to know how to use it so that they can harness and maximize the benefits associated with the programming language.

Prior studies have shown how educators have been unwilling to use new pedagogies to teach in the classroom [9]. The technologies suggested in the pedagogical approaches are frequently studied by researchers and there has been consensus on their 'effectiveness'. However, the technologies are rarely implemented in actual classrooms to enrich the educational process.

Two reasons are cited for the educators' unwillingness towards using the technologies in the classroom: they are not familiar with the available information-communication technologies (ICT), and even if they are, it is difficult to keep up with the rapid pace of development. As asserted by the study, effective integration of the technologies require the learners (students) to be the focus and educators should not spend significant amounts of time and effort to learn how to use the technologies to teach.

Scratch has the potential to pervasively introduce ICT technology as teaching aids in the classroom. As a media-rich environment which empowers users to create various media including animations, games and interactive art, Scratch is highly versatile and customizable, yet easy to learn and implement.

Scratch was developed to encourage self-exploration of programming and individual learning [10]. However, the programming process proposed in this study is one where educators have more control over and the results can be monitored. We believe this to be more beneficial and appealing to educators because they are able to meet the learning objectives for their students, and if these objectives are not met, the necessary improvements can be implemented. Also, the educators will be able to monitor the progress of the students, thereby enhancing the appeal of this process for educators compared to letting students freely explore Scratch on their own; where the outcomes are uncontrollable and educators are unable to judge whether the learning objectives have been achieved [11].

III. Modularization of Code

Even with its intuitive GUI and easy to learn functions, programming in Scratch still involves programming codes, although the language is presented in a more visual and graphical style. Scratch breaks down traditional code languages into code fragments, otherwise known as "blocks". Using these blocks of code, Scratch programmers drag and drop the blocks into scripts to develop programs and content, as seen in Figure 1.

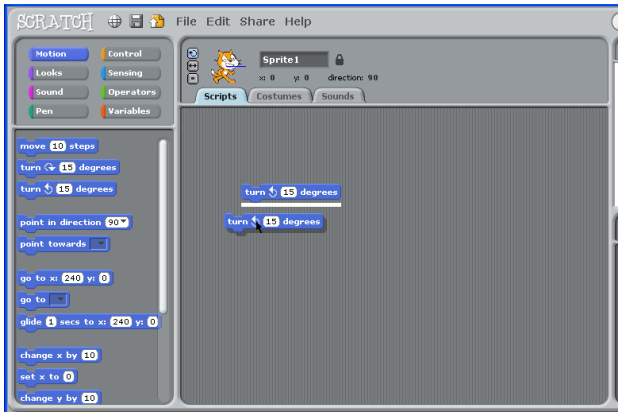

Fig. 1. Drag and drop blocks of code to provide 'instructions'

Traditionally, new Scratch programmers would develop large blocks of continuous code in order to achieve specific content. These large blocks of code are analogous to a storyline, where programmers insert blocks to command the program to run from the start to the end continuously. An example of programming using large blocks of code is shown in Figure 2.

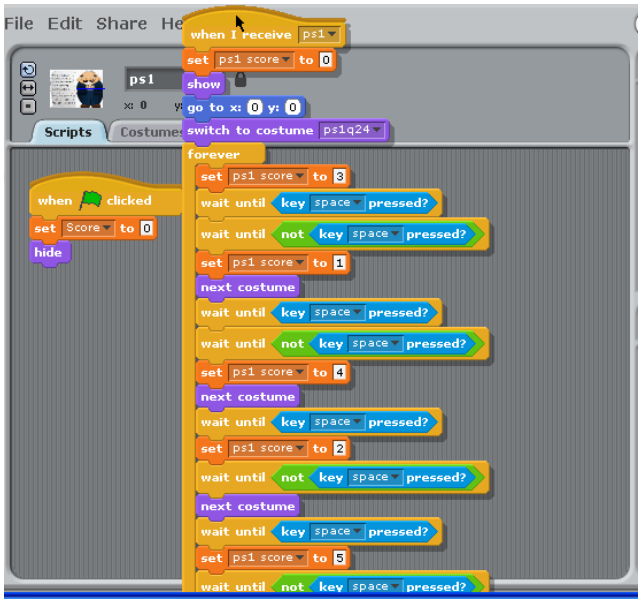

Fig. 2. A large block of programming code with instructions from start to end.

This form of large-block coding in Scratch is simple and gives the programmer a sense of continuous flow during the development of content, hence is commonly used. Programmers can easily identify logic or semantic errors by analyzing the flow of the whole block of code.

However, this method of programming is only suited for personal self-exploration of coding in Scratch, like how it is used in current educational contexts where the young are encouraged to develop their own programs at their own individual free-will. Such large-block coding is not suitable for educators' use, because it is too complex and inefficient.

Large-block coding is complex because programmers have to understand the logic of the entire programming code block. Despite its easy-to-learn GUI, it still involves learning of the programming code blocks and requires practice in order for interactive media content to be built. The hassle educators

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

have to go through in learning and practicing Scratch programming in order to develop their own customized content is the core reason why educators are unwilling to adopt ICT technologies in the curriculum. Thus Scratch programming should not be complex and educators should not need to devote significant time and effort to learn or practice how to implement the programming code blocks.

The solution we propose in this study to address the aforementioned problem was used to solve a similar problem in an earlier study. The concept of clean modularization and reusing existing code to enhance the appeal of programming was demonstrated in the introduction of AspectJ, a new extension to the Java programming language, to developers [12].

Also, large-block coding is highly inefficient, because every time an educator wants to develop new content to suit their students' needs or the curriculum needs, they would have to create a whole new block of code. Previous blocks of code created by either themselves or others cannot be used because they are created for a highly focused purpose and modifying existing blocks of code could be even more inefficient and problematic than programming new code.

In view of the problems associated with large-block coding, this study proposes modularization of the coding blocks in Scratch. *Modularization of code* is defined as the development of customized interactive media content using modularized code templates. Earlier studies have demonstrated how self-contained, side-effect free code modules can be developed in order to maintain the structural integrity of legacy systems [13], indicating how code modularization can be adapted to solve a similar problem, as in the case of this study.

Modularized code templates are smaller blocks of code; each modularized code template serves a unique purpose, and when all the templates are linked to each other, the similar outcome is achieved as compared to large-block coding. These templates would be used to develop the customized interactive media content each educator requires.

It is important to recognize that the modularization of code is not breaking down the large blocks of code per se; the modules of code template are developed based on purpose, as described in [14]. For example, the most fundamental code template which any animation or game media content developed in Scratch would have is the "Start Script-Stop Script" code module template. This short block of code provides the simple instructions of when to start running the scripts in the animation/game and when to stop. Figure 3 shows the code module.

Hence, this demonstrates how modularization of code is different from simply breaking the large code down into smaller blocks; the "Start Script" code and "Stop Script" code would be in different code modules if the latter method is used. Also breaking down large code serves little use, for the code modules are but continuations of the previous modules and the same problems of complexity and inefficiency would still not be solved.

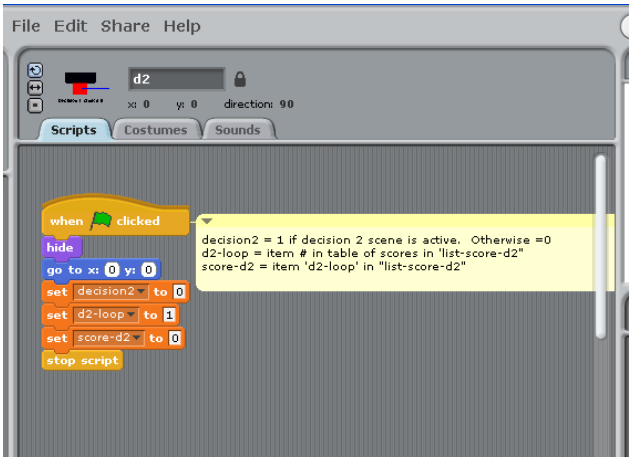

Fig. 3. The fundamental modular code template—"Start Script-Stop Script"

Through the modularization of codes based on each code block's purpose, both problems of complexity and inefficiency would be addressed effectively. With each code block serving a unique purpose, educators can easily change the details to suit their purposes. For example, if the educator wants to change the point scoring system in the game s/he is developing, s/he only needs to change the score item on the code block as seen in Figure 4 and the scoring system would be changed according to preferences.

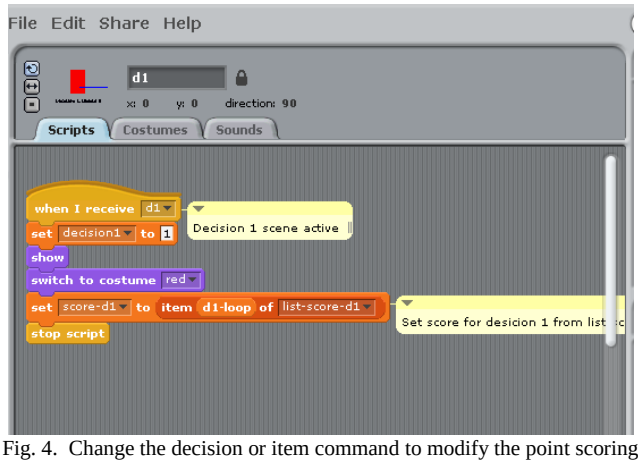

Fig. 4. Change the decision or item command to modify the point scoring system

In order to decide on which code module templates are to be made available for each type of interactive media, education policy makers would need to discuss and collaborate with programmers. For the purpose of discussion in this study, we propose four fundamental code module templates to develop a specific type of interactive media—computer games.

Start-Stop Code: As discussed earlier, the start-stop code module is the most basic code module which instructs when to start and stop running the scripts (scripts are sequences of instructions given by the codes), causing the game to begin and end accordingly.

Point Scoring Code: Educators who want to track the progress of students would use this code module to allocate points for the decisions made. When students play the game, they make certain decisions, and based on their choices, points are awarded and/or deducted.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Character Animation: Most computer games would involve at least one character; this code module offers the basic functions of animations (e.g. walking) and dialogue (e.g. in speech dialog boxes) for each character. For multiple characters, this code module can be duplicated.

Storyline: A game needs to have a storyline and a problem for gamers to solve; this code module helps to present the problem(s) faced in the game through text and graphic display.

Samples of the start-stop code and point scoring code modules have been shown in the above figures (Figures 3 and 4 respectively). Demonstrations of the character animation code and storyline code modules will be presented in the case study (Figures 5 & 7 and 6 & 8 respectively).

IV. Benefits of Modularization

Beyond the increased simplicity and enhanced efficiency of modularizing Scratch codes, this process also offers multiple benefits.

Firstly, through these smaller blocks of code which are classified according to purpose (e.g. point scoring), it is easier to spot logic and semantic errors and resolve the problems. With large code blocks, when an error is identified when previewing the output program, the programmer has to go through the whole block of code to identify the problem or semantic error. Upon identification of the problem, any change to the code could possibly affect other codes, resulting in more semantic errors and problems. For example, if the point scoring system for one game scenario is changed within the large block of code, the other point scoring systems for other game scenarios need to be changed similarly.

However, with these smaller code blocks, any problems can be easily directed to the corresponding modular code block (e.g. if the point scoring system is not functioning as intended, the programmer works on the point system modular code block directly). Furthermore, the changes made in each code block are contained and only affect the relevant codes within the block itself. The independence of code blocks resulting from code modularization prevents a chain reaction from occurring after a change has been made to one line of code (i.e. codes which serve different purposes would not be affected by a change made in a specific code block).

In addition, the development of the interactive media content would be more efficient because the code templates are developed for almost any variation of the specific type of interactive media. The modularized codes used in this study were developed to function as a game engine. By using these modular code templates, a wide variety of games can be developed to suit multiple purposes.

Earlier studies conducted on code modularizations have shown how this technique provides a high degree of flexibility and code reusability [15], although a different programming language was used—Java.

Hence, educators do not have to develop new blocks of code to cater to different students or different curriculum objectives; all that is required is to modify pre-existing code templates. Students can then look forward to a diversified and non-static learning experience.

V. Case Study of Modularization

In order to demonstrate how modularization of codes in Scratch can help to develop interactive media content more effectively, we implemented the technique to develop computer games in Build-It-Yourself (BIY) [16].

BIY is an enterprise which conducts workshops to encourage children (aged 10-12) to solve problems creatively using technology. The skills taught to the children are ICT-enhanced skills such as team work and documentation skills. In order to understand and assess how effective BIY's workshops have been in teaching the children these ICT-enhanced skills, two computer games were created using the same modularized Scratch game code: The Computer Tycoon and Fix the Robot.

The purpose of developing these games—to measure soft skills, could possibly be one of the main reasons why educators would use interactive media content as an alternative/complementary teaching tool, in recognition of the ineffectiveness of traditional assessment methods involving the pen-and-paper tests in measuring skills [17].

The Computer Tycoon is a business simulation computer game where the gamer runs a computer trouble-shooting business and has to make decisions in order to overcome the problems s/he faces. Based on the decisions made, points are awarded for sound and logical decisions, and the total score of the game is the reflection of the level of competency of the child's ICT-enhanced skills. Figure 5 shows the instructions of the game (developed from storyline code) whilst Figure 6 shows one of the decisions the gamer has to make—choosing a business partner (developed from character animation code).

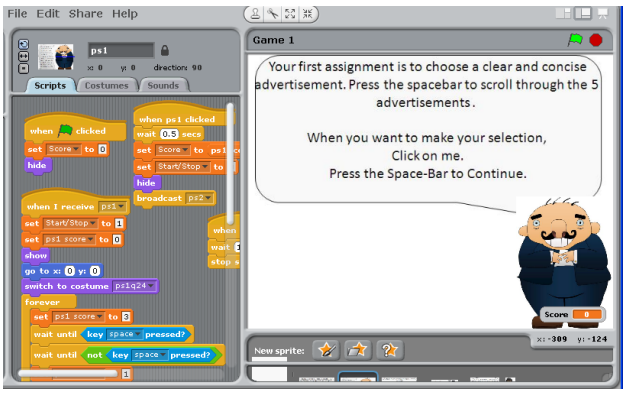

Fig. 5. The 'Instructions' screen on the preview pane on the right, the modified modularized code template on the scripts pane on the left.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

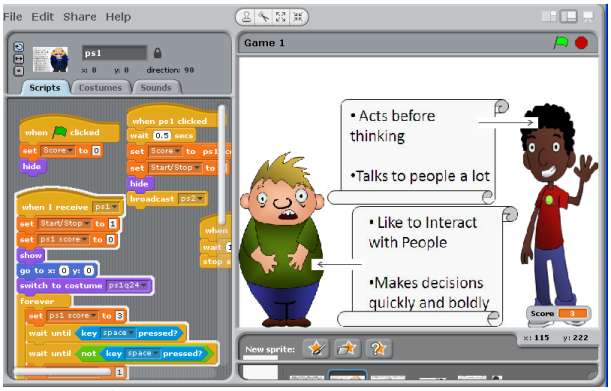

Fig. 6. One of the decisions the gamer has to make—choosing a business partner

Fix the Robot is an adventure game developed from the same modularized code templates. Utilizing the same point scoring system and coding instructions, the score from the adventure game also aims to reflect the level of competency of the children's ICT-enhanced skills, albeit using a different genre of game. The gamer is presented with a situation: their robot is broken and they have to find out how to repair the robot as well as search for resources to rebuild it. Figure 7 shows the introduction of the problem to the gamer—that the robot is damaged (developed from storyline code). Figure 8 shows one of the decisions the gamer has to make—understanding the problem faced (developed from character animation code).

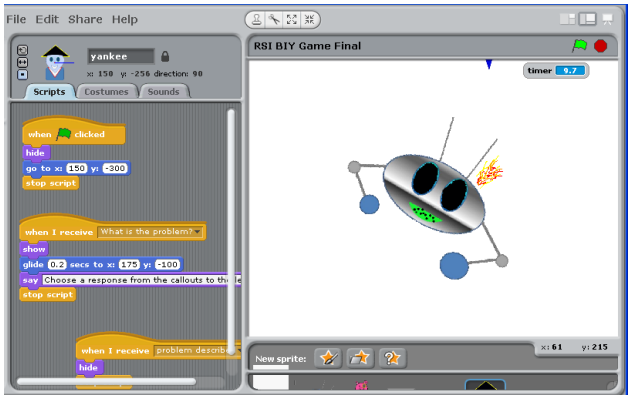

Fig. 7. Introducing the problem to gamers—the robot is damaged and the gamer has to repair it.

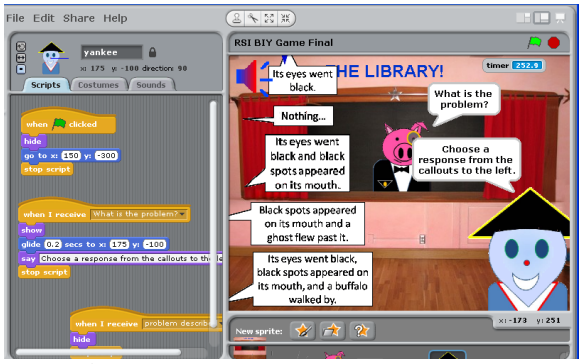

Fig. 8. One of the decisions the gamer has to make—What is the problem?

From this case study, we have seen how a variety of media content, including but not limited to computer games can be developed with modularized code templates. Although both games differ in genre—The Computer Tycoon being a business simulation game whilst Fix the Robot being an adventure game, they stem from the same modularized code templates, hence demonstrating the versatility of modularizing Scratch codes.

VI. Conclusion

By understanding why new ICT technologies are unable to effectively pervade into the curriculum, the Scratch code modularization technique has yielded promising results in the development of customized interactive media content to better engage students.

Furthermore, by leveraging on existing technologies proven to work in educational contexts (Scratch), development of these modularized programming code templates are easy to accomplish and can be efficiently deployed to meet a wide variety of objectives. The case study discussed earlier has shown but a few of the many capabilities and variants these modularized code templates can lead to.

With Scratch widely available worldwide at zero cost, coupled with its intuitive, easy to learn features, we recognize the potential modularized Scratch code can have on classrooms worldwide, especially in less economically affluent communities. All that is needed to capture the attention of many young minds is a computer with Scratch interactive media content, and with the modularized Scratch codes, even the least developed and affluent of communities can provide children a fun, active and engaging learning experience.

VII. Future Work

Although modularization of Scratch code is very promising as discussed in this study, there has to be follow-up work in order to affirm and implement this strategy into curriculum for effective teaching.

Future work would involve educators trying to develop various interactive media content using our modularized code templates to assess the level of intuitiveness and ease of use. This study only involves a theoretical discussion of how the Scratch code is simplified, but actual experimental studies with educators are required in order to effectively gauge how intuitive and efficient programming from these modularized Scratch codes are.

It would be beneficial to conduct comparative studies; educators should attempt to develop their own interactive media content without any use of the modularized codes, then program with the modularized codes, and a more satisfying and efficient programming process for the latter study would augment this study's argument.

The proposed follow-up experimentation would be to establish a hypothetical learning objective for educators to develop interactive media content (e.g. create an animation to demonstrate how an object, regardless of weight, falls at the same velocity at a given height). Given a fixed timeframe for the development (e.g. 3 hours), one group of educators will program without code module templates whilst the other group of educators will be provided code module templates to work with; each educator works individually. Upon

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

completion of the experiment, the end-products of both groups of educators would be compared; a significantly more complete end-product and reduced occurrences of semantic errors for the group of educators provided with code module templates would indicate the effectiveness of this technique and encourage application in the real classroom setting.

In addition, modularized code templates should also be developed for other interactive media content, because educators have varying needs and not all educators would want to implement games in their classroom teaching; they could prefer other interactive media such as videos and interactive art. Hence, by developing a wider variety of modularized code templates, educators would be further encouraged to adopt this strategy to transform the classroom teaching from a passive process to an active and engaging experience.

ACKNOWLEDGEMENT

The author of this paper would like to express his heartfelt appreciation towards Mr. John Galinato & Dr. Jenny Sendova from Build-It-Yourself and the Bulgarian Academy of Science for the invaluable guidance and advice rendered during this study. The author would also like to thank the members of the Build-It-Yourself team as well as Research Science Institute staff and scholars have provided support and guidance, leading to the successful completion of the study.

REFERENCES

[1] Scratch: Imagine, Program, Share, by the Massachusetts Institute of Technology Media Lab Lifelong Kindergarten Group. Available at: http://www.scratch.mit.edu

[2] J. Fildes, "Free tool offers 'easy' coding", BBC News, 14 May 2007. Available at: http://news.bbc.co.uk/2/hi/technology/6647011.stm. Accessed on 20 November 2010

[3] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick, "Scratch: a sneak preview", in *Proc. of Second International Conference on Creating, Connecting and Collaborating through Computing*, 2004, pp. 104-109.

[4] L. Markosian, P. Newcomb, R. Brand, S. Burson, T. Kitzmiller, "Using an enabling technology to reengineer legacy systems", in *Communications of the ACM Vol. 37 Issue 5*. New York, USA: ACM, May 1994, pp. 58-70

[5] H.-H C., Chen, H. F. O'Neil, "A formative evaluation of the training effectiveness of a computer game", in *Computer games and team and individual learning*. Oxford, UK: Elsevier, 2008, pp.39-54.

[6] D. Toback, A. Mershin, I. Novikova, "New Pedagogy for Using Internet –Based Teaching Tools in Physics Course", CERN Document Server, Physics 0408034, 2004

[7] M. Lampert, "Using Technology to Support a New Pedagogy of Mathematics Teacher Education", in *Journal of Special Education Technology v12"*. Virginia, USA: Technology and Media Division of the Council for Exceptional Children, 1994, pp. 276-289

[8] D. J. Malan, H. H. Leitner, "Scratch for budding computer scientists", in *Proc. of the 38th SIGCSE technical symposium on Computer science education*. New York, USA: ACM, 2007, pp. 223-227.

[9] E. Stefanova, P. Boytchev, N. Nikolova, E. Kovatcheva, E. Sendova, "Embracing and enhancing ideas as a strategy for ICT education", in *Research, Reflections and Innovations in Integrating ICT in Education Vol. 1*. Badajoz, Spain: Formatex, 2009, pp. 206-211. Available at: http://www.formatex.org/micte2009/book/206-211.pdf

[10] M. Resnick, J. Maloney, A. M-. Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, "Scratch: Programming for All", in *Communications of the ACM Vol. 52 No. 1*. New York, USA: ACM, November 2009, pp.60-67

[11] R. B. Kozma, "Technology and Classroom Practices: An International Study", in *Journal of Research on Technology in Education*, 36 (1). Washington DC, USA: International Society for Technology in Education, 2003, 36 (1), pp. 1-14

[12] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. Griswold, "Getting started with ASPECTJ", in *Communications of the ACM Vol. 44 Issue 10*. New York, USA: ACM, Oct 2001, pp.59-65

[13] R. Al-Ekram, K. Kontogiannis, "Source code modularization using lattice of concept slices", in *Proc. of the Eighth European Conference on Software Maintenance and Reengineering 2004*. Tampere, Finland: IEEE Computer Society, 2004, pp. 195-203

[14] K. Schneider, J. Brandt, E. Vecchie, "Modular Compilation of Synchronous Programs", in *From Model-Driven Design to Resource Management for Distributed Embedded Systems, IFIP TC 10 Working Conference on Distributed and Parallel Embedded Systems 2006 Vol. 225*. Braga, Portugal: Springer Link, 2006, pp. 75-84

[15] J. Hannemann, G. Kiczales, "Design pattern implementation in Java and AspectJ", in *Proc. of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications 2002*. New York, USA: ACM, 2002, pp. 161-173

[16] Build-It-Yourself, Cambridge, MA 02139. http://www.build-it-yourself.com

[17] P. Sacks, "Standardized Minds", Da Capo Press, 2001, pg 3.