To appear in the Proceedings of the 8th IEEE Integrated STEM Education Conference, March 10, 2018

Integrating Algebra, Geometry, Music, 3D Art, and Technology using Emoticoding

Angelos Barmpoutis, Member, IEEE
University of Florida, angelos@digitalworlds.ufl.edu

***Abstract* – Emoticoding is a technique for learning computer programming that has been shown to improve student learning outcomes and reduce blank page trauma during the students' first encounter with text editing interfaces. In this paper, a generalized method is presented for integrating computer education with other learning topics, such as algebra, geometry, music, and 3D art, using emoticoding. The proposed method is based on the theoretical framework of brain-activating text replacements, which assists students to make connections between the tokens of a typed language (such as computer code) and a set of replacing graphemes (such as interpretative visual or textual replacements). When the computer code is instantly being replaced with graphemes from another learning topic, for example geometric shapes or music notation, the students can build associations between the underlying concepts, which in turn reinforces learning of the associated topics. A work-in-progress user interface with four sets of visual replacements is presented in this paper for substituting the discrete tokens of a computer program (JavaScript) with symbols from algebra, geometry, music notation, and solid shapes. The proposed replacements are demonstrated with computer scripts through the emoticoding framework using learning objectives from K-12 common core standards.**

Index Terms - Computer Education, Emoticoding, Source Code Editors, STEM Education, Computer Programming

INTRODUCTION

In the past decade, information and communication technology curricula have been designed and made compulsory in K-12 education in several countries [1, 2]. Research on early computer education, however, has been a well studied subject since the 1960's [3], and until today several techniques have been developed for addressing learning difficulties related to computer coding [4,5,6].

These learning techniques can be categorized based on: a) the type of user interaction they use; b) the different age group or stage of learning they target; and c) the degrees of freedom given to the users. Table I provides a synoptic overview of these categorizations. Overall, there are three different types of educational interfaces:

- Tangible User Interfaces (TUI) consist of limited sets of physical blocks, which represent various programming functions that the user must arrange in a logical order. It has been shown that TUIs can be very effective in early computer education (K-3) [7,8].
- Graphical User Interfaces (GUI) are the virtual equivalent of TUIs [9, 10, 11, 12]. They offer a block-based programming environment in which the users can drag and drop coding blocks to create simple programs. GUIs have been effective in middle stages of learning.
- Text Editing Interfaces (TEI) are closer to the professional programming languages, as they offer a text editor within an integrated development environment, in which the users can type scripts that adhere to the syntactic rules of a programming language [13,14]. These are more appropriate for higher stages of learning (9th grade-college), and may be limited to a focus area, such as graphic design, music remixing, game design, etc.

TABLE I
CATEGORIES OF LEARNING INTERFACES FOR COMPUTER CODING

Type	Stage	Degrees of Freedom	Examples
TUI	1	Sequence of command blocks, Limited set of tangible blocks.	RoboBlocks [7], Turn [8], and others.
GUI	2	Sequence of command and input argument blocks, Limited set of visual blocks.	Scratch [9], Tynker [10], Alice [11], Greenfoot [12], and others.
TEI	3	Sequence of typed characters, Easy to use but limited API.	Processing [13], EarSketch [14], and others.
TEI	4	Sequence of typed characters, Easy to use, full-control API.	Python, JavaScript, Swift, Java, etc.

One key problem in this sequence of educational tools is the large disconnect between GUIs and TEIs, as the former underestimate the complexity of the latter [9] in terms of degrees of freedom as well as overall capabilities that may lead to "blank page trauma" during the students' first encounter with coding interfaces [6,15]. A hybrid solution between GUIs and TEIs is the technique of Emoticoding [16,17], which offers a visual overlay that appears instantly in the text editor on top of each programming token using an interaction similar to the typing of emoticons in social media [16].

This paper utilizes the theoretical framework of brain-activating text replacements, in order to assist students to build associations between a typed computer script and a set of replacing graphemes and symbols from algebra, geometry, music notation, and solid shapes through the method of emoticoding. A work-in-progress user interface is presented that integrates various STEM areas, such as math,

and technology, with other learning topics, such as 3D art and music. The proposed method is hypothesized to simultaneously reinforce learning on both computer programming and other individual topics that will be used as interactive code replacements.

The contributions of this paper are threefold: a) a novel work-in-progress framework for integrating algebra, geometry, music, 3D art, and technology is presented; b) the brain-activating replacement method [16] is extended to account for couplings of tokens between two different learning subjects; c) new sets of proposed replacements are demonstrated using coding exercises with learning objectives from K-12 common core standards [18].

METHOD

Let Ω be the set of all acceptable tokens in a given programming language. A computer script s can be expressed as an element of a space S that contains all acceptable ordered sets of tokens from Ω as follows:

$$s = \{\omega_1, \quad \omega_2, \quad \cdots \quad \omega_n\} \in S, \text{ where } \omega_i \in \Omega. \tag{1}$$

A mapping between the set Ω and another set of tokens Ψ can be defined as:

$$M: (\Omega, S) \to \Psi \tag{2}$$

that maps every token $\omega_i \in \Omega$ in a given context $s \in S$ to a token ψ_i, thus forming a new script:

$$s' = \{M(\omega_1, s), \quad \cdots \quad M(\omega_n, s)\} = \{\psi_1, \quad \cdots \quad \psi_n\}. \tag{3}$$

The mapping M in (2) and (3) has the following properties:

- $s = t \Rightarrow M(\omega, s) = M(\omega, t)$, where $s, t \in S$. This implies that a token $\omega \in \Omega$ may be mapped to different elements of Ψ within different contexts s and t.
- The cardinality of s in (1) and s' in (3) is the same, i.e. the mapping M does not affect the number of tokens.
- From the above properties it is derived that the scripts s and s' have parallel syntactical alignment.
- The identity mapping is defined as: $M(\omega, s) = \omega, \forall \omega$.

The above framework can be applied to the process of typing text in a text editor. Typically the typed tokens and the displayed tokens are identical; hence an identity mapping is used. If a non-identity mapping is utilized instead, the typed tokens ω are instantly being replaced by ψ from a different set of tokens Ψ. The following observations can be made for this form of text typing:

- The graphemes of Ψ can be different from those in Ω. For example, they can be visual interpretations of Ω.
- The association between the tokens of Ω and Ψ is a new form of knowledge that the user acquires during typing.
- The tokens of Ω can be seen as a form of action (typing) and the tokens of Ψ as a form of reaction, the alternation of which triggers the cognitive process.

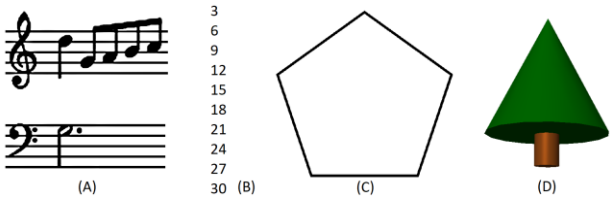

FIGURE I
THE RESULTS OF THE FOUR SAMPLE SCRIPTS DISCUSSED IN THIS PAPER

- In this model, code editing becomes an active experimentation process that involves action, reflection, and abstract conceptualization that reinforces learning.
- If the graphemes of Ψ are carefully chosen from a different educational topic, learning can be reinforced on both associated subjects, i.e. computer programming (Ω) and the other chosen subject (Ψ).
- This process is employed in several other functions of computer typing, such as the use of keyboard shortcuts, typing of emoticons, typing of Chinese logograms, etc.

IMPLEMENTATION

An implementation of the proposed method has been developed through the emoticoding framework [17]. The term *emoticoding* does not refer to the use of emoticons, but to the use of the same human-computer interaction as in emoticon typing. A set of metaphors has been employed to provide visual interpretation of the tokens in JavaScript using name tags for variable names and pipes for functions [15]. Through this set of metaphors, the individual tokens involved in the syntactic pattern "X(…)" obtain distinct visual interpretation. For example, the role of the parenthesis is to construct the opening of the pipe. Furthermore, jigsaw puzzle metaphors can be used to visually separate the individual tokens from each other.

Four learning domains have been identified in order to demonstrate the proposed method, namely algebra, geometry, music, and visual arts. In addition, technology (computer programming) will be employed as the fifth educational area that will be integrated with the others using the proposed framework. Each case is demonstrated with a sample coding exercise with learning objectives from K-12 core standards [18, 19, 20]. Figure I presents the corresponding output obtained from each script.

I. Music and Computer Programming

An example of the proposed method is shown in Fig. II, which conforms to the music standard: "When analyzing selected music, read and perform using standard notation." (5th grade core standard MU:Pr4.2.5b) [19]. Figure II shows a 7-line script (in JavaScript) that reproduces a music score in the form of a computer program using a set of provided commands, such as "quarterNote", "eightNote", etc. Each command is being replaced with a symbol from music notation thus exposing the student to both programming and music principles through the proposed method. The typed script (Ω) is provided on top, and the token replacements (Ψ) that the students instantly see are shown on the bottom.

```
1. quarterNote("D6");
2. eighthNote("G5");
3. eighthNote("A5");
4. eighthNote("B5");
5. eighthNote("C6");
6. track(2);
7. halfNoteDot("G4");
```

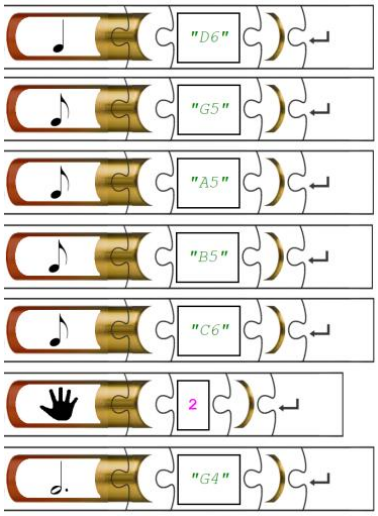

FIGURE II
COMPUTER CODE FOR PLAYING "MINUET IN G MAJOR, BWV ANH. 114

Note that the students do not see what they type in its original form Ω; they see Ψ instead.

II. Algebra and Computer Programming

Mathematical operations and algebraic thinking are intrinsically associated with computer programming, as they are fundamental elements of computational processes. For example, consider the following learning objective: "Gain familiarity with factors and multiples." (4th grade standard CCSS.MATH.CONTENT.4.OA.B.4) [18]. Several different programming exercises can be defined for this objective such as "print out all multiples of 3 up to 30". This can be solved using a 6-line script (in JavaScript) using a function "print", which is provided to the students. The names of the variables in this script can be replaced by nametags and the names of the functions by pipes using the emoticoding framework as shown in [15].

III. Geometry and Computer Programming

Learning objectives from geometry such as: "Graph points on the coordinate plane to solve real-world and mathematical problems." (5th grade standard CCSS.MATH.CONTENT.5.G.A.1&2) [18] can be practiced using a set of provided commands for 2D drawing. Figure I C shows the output of a script that draws a pentagon. Visual representations of geometric concepts can be used as token replacements of the function names "move" and "turn", which are provided to the students in order to solve this problem. For example see the visual replacement of the function "moveUp" in Fig. III.

```
1. color('brown');
2. cylinder();
3. moveUp(1);
4. color('green');
5. cone();
```

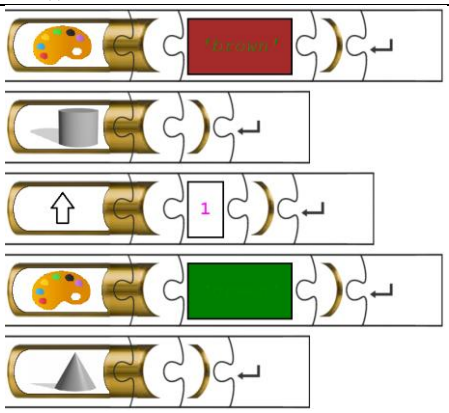

FIGURE III
COMPUTER CODE FOR DRAWING A 3D MODEL OF A PINE TREE

IV. 3D Art and Computer Programming

Finally, Figure III shows how multiple learning objectives from various subjects can be integrated with learning topics on technology and computer programming. More specifically, in this task the students are requested to compose a 3D model of a given shape. This exercise conforms to the Visual Arts standard: "Explore and invent art-making techniques and approaches." (Standards Studio for fresh ideas - upper elementary) [20] and the mathematical standard: "Represent three-dimensional figures using nets made up of rectangles and triangles…" (6th grade standard CCSS.MATH.CONTENT.6.G.A.4) [18]. This example demonstrates a rich set of visual replacements for the commands "color", "cylinder", "cone", which are provided to the students in this exercise in order to compose the 3D model of a pine tree.

PRELIMINARY RESULTS AND DISCUSSION

By observing the examples provided in this paper, it is evident that a student who observes a script in the emoticoding framework (i.e. the visually replaced code) is prompted to memorize the associated computer code by following the given visual hints. For example, in the example of Fig. II, the student who observes the symbol of a quarter note should first recognize this musical symbol and then, using this as a hint, should recall the underlying function name "quarterNote". In case of error, i.e. if the student incorrectly identifies this symbol as an eighth note and subsequently types "eighthNote", a different visual token replacement will unexpectedly appear. This active experimentation process will lead the student to the following two realizations: a) The symbol for an eighth note is not the one that the student had originally in mind; b) the new symbol that appeared corresponds to an eighth note.

This demonstrates that the proposed integrated framework is a try-and-error environment that reinforces learning on both directions, i.e. from Ω to Ψ and vice versa. This is a significant benefit over conventional text editors, and may increase the probability of a student following a technology-oriented career. One of the future directions of this project is to assess this hypothesis on a long-term study.

Furthermore, contrary to TUIs and GUIs, such as block programming, the proposed method does not require transitioning to a new environment or a subsequent additional learning curve. As the proposed framework is actually a TEI method, it trains the students to use a real programming language, a skill that could be later used in the professional world.

Preliminary studies on the use of emoticoding as a computer education framework reported improved student learning outcomes in terms of syntax recall and logic comprehension in K-12 [15] and college-level students [16]. Furthermore, these studies indicate that this method reduces blank page trauma during the students' first encounter with text editing interfaces in comparison to the use of conventional text editors after a 10-week block-based coding training [15].

These results indicate that the integrated framework presented in this paper could benefit several other learning topics beyond computer programming. A future study will be designed to assess this hypothesis using cross-disciplinary learning objectives. To facilitate the design and implementation of this study, a call for collaborators has been published in the website of the emoticoding project in order to establish partnerships with K-12 teachers from various disciplines who would like to experiment with the integration of the proposed method into K-12 curricula.

Following the design of the study in [15], the integration of the proposed method will follow a TUI or GUI-based early exposure to computer programming. Ideally, the participating schools will have already an established computer programming training as part of their STEM curriculum. The proposed method assists the students on their first encounter with TEI coding interfaces, which typically does not happen before middle school. Hence, high-school and late middle-school ages are the most appropriate target ages for the integration of the proposed method into K-12.

The implementation will address different pairs of subjects depending on the focus of each partnering school and the subject of the participating teacher. For example, integration of music with technology (Fig. II) can be implemented as part of the curriculum in a music class in a STEM-oriented school, or as part of a technology class in an Arts-oriented school. A series of training modules and coding exercises will be defined accordingly.

Finally, a dual evaluation mechanism will be established in order to assess the effect of the proposed method on the learning outcomes in the two corresponding subjects (e.g. music & technology) and will be compared with the outcomes without the proposed method.

ACKNOWLEDGMENT

The author would like to express his appreciation to the University of Florida College of the Arts for honoring him with the "Best Teacher of the Year" award in 2017 for inventing and applying the method discussed in this paper.

REFERENCES

[1] Hu, C. F., Lin, Y. T., Chuang, H. C. and Wu, C. C. April 2014. "A Recommended ICT Curriculum for K-12 Education." *Proceedings* of the International Conference on Teaching and Learning in Computing and Engineering (LaTiCE), Kuching, pp. 33-36.

[2] Al-Karaki, J. *et al.* April 2016. "Towards an innovative computer science & technology curriculum in UAE public schools system." *Proceedings* of the IEEE Global Engineering Education Conference (EDUCON), Abu Dhabi, pp. 883-891.

[3] Buckingham, R. A., 1965. "The computer in the university." *The Computer Journal* 8(1), pp.1-7.

[4] Lahtinen, E., Ala-Mutka, K., Järvinen, H.M. September 2005. "A study of the difficulties of novice programmers." *ACM SIGSCE Bulletin* 37(3), pp. 14-18.

[5] Jenkins, T. August 2002. "On the difficulty of learning to program." *Proceedings* of the 3rd Conference of the LTSN Centre for Information and Computer Sciences, vol.4, Loughborough, pp. 53-58.

[6] Morgado C. and Barbosa, F. July 2012. "A structured approach to problem solving in CS1." *Proceedings* of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE), Haifa, pp. 399–399.

[7] Sipitakiat, A., Nusen, N. June 2012. "Robo-Blocks: Designing debugging abilities in a tangible programming system for early primary school children." *Proceedings* of the 11th International Conference on Interaction Design and Children, Bremen, pp. 98–105.

[8] Horn, M. S., Jacob, R. J. May 2007 "Tangible programming in the classroom with Tern." *Proceedings* of CHI Extended Abstracts on Human Factors in Computing Systems, San Jose, pp. 1965–1970.

[9] Malan, D. J. and Leitner, H. H. March 2007. Scratch for budding computer scientists. *ACM SIGCSE Bulletin*. 39 (1), pp. 223-227.

[10] García-Peñalvo, F. J., Rees, A. M., Hughes, J. *et al*. November 2016. "A survey of resources for introducing coding into schools." *Proceedings* of the 4th International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, pp. 19-26.

[11] Daly, T. May 2011. "Minimizing to maximize: an initial attempt at teaching introductory programming using Alice." *Journal of Computing Sciences in Colleges* 26(5), pp. 23–30.

[12] I. Utting, S. Cooper, M. Kölling, J. Maloney, and M. Resnick. November 2010. "Alice, greenfoot, and scratch–a discussion." *ACM Transactions on Computing Education (TOCE)* 10(4), pp. 17.

[13] Reas, C. and Fry, B. December 2014. *Processing: A Programming Handbook for Visual Designers and Artists*, Cambridge: MIT Press.

[14] Freeman, J., Magerko, B. and Verdin, R. March 2015. "EarSketch: A web-based environment for teaching introductory computer science through music remixing." *Proceedings* of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, pp. 5-5.

[15] Huynh, K. and Barmpoutis, A. July 2018. "Name Tags and Pipes: Assessing the Role of Metaphors in Students' Early Exposure to Computer Programming using Emoticoding." *Proceedings* of the 9th International Conference on Applied Human Factors and Ergonomics, Orlando, *in press*.

[16] Barmpoutis, A., Huynh, K., Ariet, P. and Saunders, N. July 2017. "Assessing the Effectiveness of Emoticon-Like Scripting in Computer Programming." *Advances in Intelligent Systems and Computing (AISC)* 598, pp. 63–75.

[17] "Emoticoding" 2017. emoticoding.org. Web. Accessed: Dec. 1, 2017.

[18] "Common Core State Standards Initiative". corestandards.org. Web. Accessed: Dec. 1, 2017.

[19] "Core Music Standards" nafme.org. Web. Accessed: Dec. 1, 2017.

[20] "National Visual Arts Standards" www.arteducators.org. Web. Accessed: Dec. 1, 2017.