Work in Progress: Mflow, a Flow-based Music Programming Platform for Young Children

Victor Hugo Minces
Department of Cognitive Science
University of California, San Diego
La Jolla, CA, US
victorminces@gmail.com

Wanli Xing
College of Education
University of Florida
Gainsville, FL, US
wanli.xing@coe.ufl.edu

Chenglu Li
College of Education
University of Florida
Gainsville, FL, US
li.chenglu@ufl.edu

***Abstract*— Students of all socioeconomic backgrounds love music and express their identity through music. There are strong historical connections between music and computing, and computer-based music has a heavy presence in contemporary popular culture. Thus, programming electronic music can provide the type of authentic learning experience that fosters participation in computer science (CS) by minoritized students. Although important efforts have been made in that direction, they have not reached young children in mainstream public classrooms, particularly in schools serving children from low-income and marginalized backgrounds. Developing a computational tool and educational program that reaches this key demographic holds the potential to greatly increase CS knowledge and participation in the future workforce. For this, our team has created M-flow, a flow-based music programming platform that seeks to be engaging for children from the outset, and that makes it extremely easy for non-specialized teachers to learn and implement CS activities in the classroom.**

Keywords—Computer Science, STEAM, Music, Elementary School

I. Introduction

Literature suggests that children need access to authentic CS learning experiences, which are experiences designed with attention to the learners' interests, identities and background— in what is called *personal authenticity* —and that reflect professional practice— in what is called *professional authenticity* (National Academies, 2021). Given the broad interest that children have in music, we suggest that music can provide this type of personal authenticity. Some educational programs have explored this possibility (see *music and computing* below), although their effectiveness has not been thoroughly evaluated. Furthermore, those programs have targeted older students, and have often required highly specialized facilitators, who are not accessible to most children.

Creating an authentic learning experience that can reach elementary students in their general education classrooms and provide early exposure to CS can be transformative at the societal level by engaging a large number of children in the CS pipeline. For this, it is necessary to develop a platform that is motivating and easily learned by teachers without CS experience, and that can be readily used by students to start developing musical and computational ideas. Given its simplicity and its widespread use by naive coders, including musicians, we propose that a flow-based programming paradigm (FBP, see description below) holds the potential to achieve that goal. Because FBP is widely used in the workforce, Mflow (Fig.1) does not only provide personal but also professional authenticity.

This work was supported by NSF Innovative Technology Experiences for Students and Teachers (ITEST) ID 1657366. Connecting STEM to Music and the Physics of Waves.

II. Background

Students' engagement with a topic can be associated with the value assigned to the topic [1]. This value can be extrinsic—e.g., associated with a grade reward—or intrinsic—e.g., associated with a student's pre-existing interests [2]. In general, it might be difficult to cater to the individual pre-existing interests of a heterogeneous student population [3]. However, interest in music is near-universal [4], and the connections between music and CS are vast, positioning CS as an encompassing interest-generating subject. This association between music and CS can lead to an increase in the utility value of CS, which can be seen as useful for creating music [5].

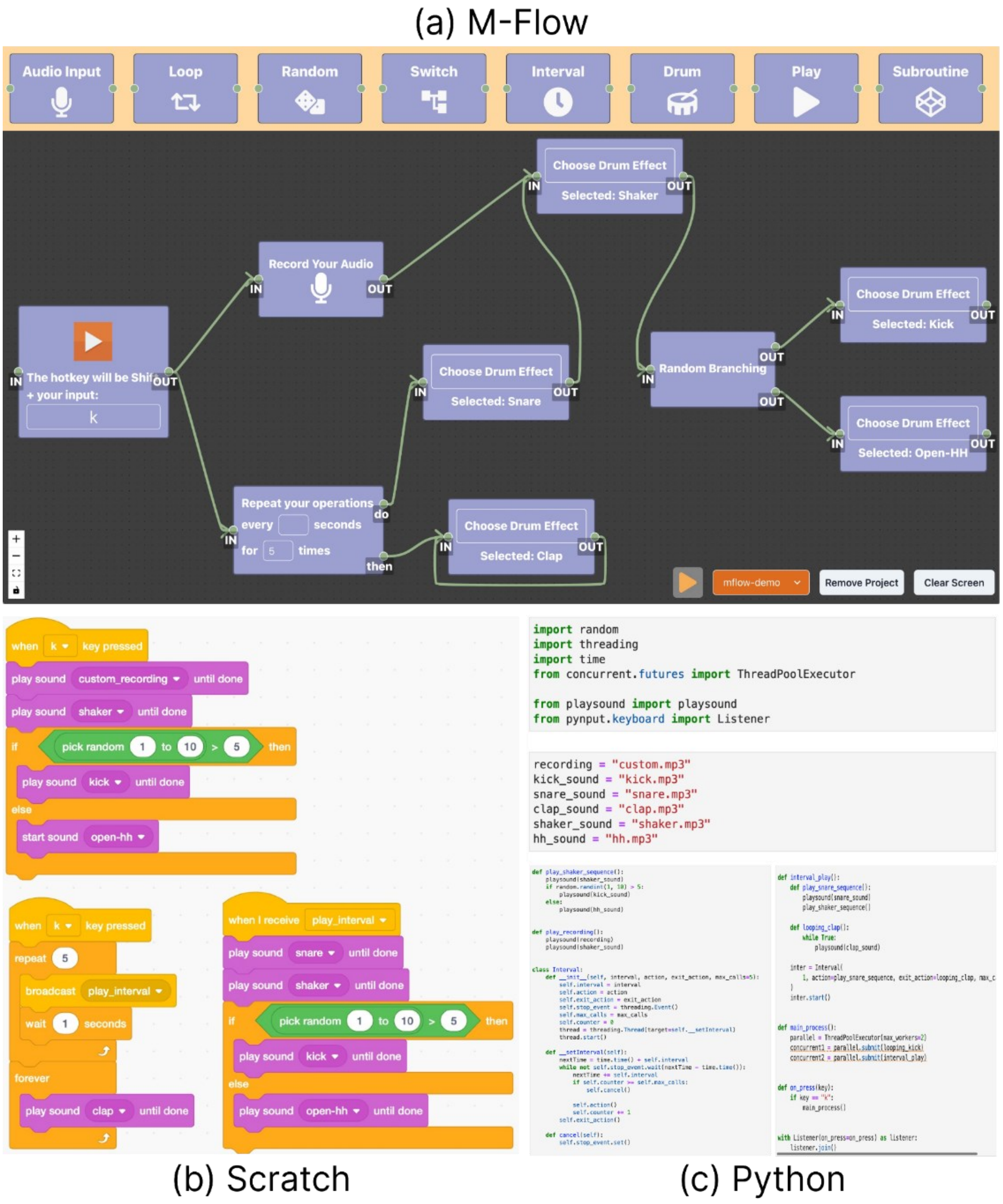
A. Music and STEM.

The idea of using music to engage students in STEM, particularly underrepresented students, is supported by theoretical considerations, and by empirical evidence by this PI and others. For example, ScienceGenius has been shown to engage underrepresented students in science classes by supporting them in creating science-related hip hop lyrics [6], which also improves the students' emotional state [7]. Math is another point of connection between music and STEM—for example, using statistics to describe musical compositions—and has been demonstrated to improve math disposition in elementary students [8] and teacher self-efficacy in pre-service teachers [9]. Some programs have connected music to STEM through the physics of waves, signal processing, and music technology, using tools such as signal generators, spectrograms, oscilloscopes, and simulations of propagating waves. One such program is *Listening to Waves* [10]–[13] was shown to be effective for engaging low-income and underrepresented students in science.

B. Music and Computing.

Electronic music structure largely replicates common structures in computer code: loops that make nested rhythms and phrases, sequences connecting those rhythms, sequences of notes making melodies, randomness, and conditional structures that allow musicians to trigger sounds using hardware [14]. This parallelism positions electronic music as a means for students to create and understand computational structures. Although there is an emerging field of work that leverages the connections between music and CS, such efforts have largely focused on high school students or students in out-of-school programs, which generally have a pre-existing interest in CS. Efforts to attract younger children, and children without a pre-existing interest in CS, have not gained much traction.

Given the connections between music and computing, the number of reported experiences using music to engage students in CS is surprisingly few. The most successful educational program is probably Earsketch [15]. Earsketch is a Python-based programming platform and educational program used mostly in high schools and elective CS classes, and has been found to improve minority students' dispositions towards CS. Other studies [14], [16], [17] relied on very heavy support from the team leading the intervention. For example, Heines and Walzer (2018) implemented coding activities with a group of middle school choir students, who presumably did



```
when k key pressed
play sound custom_recording until done
play sound shaker until done
if pick random 1 to 10 > 5 then
    play sound kick until done
else
    start sound open-hh

when k key pressed
repeat 5
    broadcast play_interval
    wait 1 seconds
forever
    play sound clap until done

when I receive play_interval
play sound snare until done
play sound shaker until done
if pick random 1 to 10 > 5 then
    play sound kick until done
else
    play sound open-hh until done
```

```python
import random
import threading
import time
from concurrent.futures import ThreadPoolExecutor

from playsound import playsound
from pynput.keyboard import Listener
```

```python
recording = "custom.mp3"
kick_sound = "kick.mp3"
snare_sound = "snare.mp3"
clap_sound = "clap.mp3"
shaker_sound = "shaker.mp3"
hh_sound = "hh.mp3"
```

```python
def play_shaker_sequence():
    playsound(shaker_sound)
    if random.randint(1, 10) > 5:
        playsound(kick_sound)
    else:
        playsound(hh_sound)


def play_recording():
    playsound(recording)
    playsound(shaker_sound)


class Interval:
    def __init__(self, interval, action, exit_action, max_calls=5):
        self.interval = interval
        self.action = action
        self.exit_action = exit_action
        self.stop_event = threading.Event()
        self.max_calls = max_calls
        self.counter = 0
        thread = threading.Thread(target=self.__setInterval)
        thread.start()

    def __setInterval(self):
        nextTime = time.time() + self.interval
        while not self.stop_event.wait(nextTime - time.time()):
            nextTime += self.interval
            if self.counter >= self.max_calls:
                self.cancel()

            self.action()
            self.counter += 1
        self.exit_action()

    def cancel(self):
        self.stop_event.set()
```

```python
def interval_play():
    def play_snare_sequence():
        playsound(snare_sound)
        play_shaker_sequence()

    def looping_clap():
        while True:
            playsound(clap_sound)

    inter = Interval(
        1, action=play_snare_sequence, exit_action=looping_clap, max_c
    )
    inter.start()


def main_process():
    parallel = ThreadPoolExecutor(max_workers=2)
    concurrent1 = parallel.submit(looping_kick)
    concurrent2 = parallel.submit(interval_play)


def on_press(key):
    if key == "k":
        main_process()


with Listener(on_press=on_press) as listener:
    listener.join()
```

Fig. 1. Comparing Mflow and other popular platforms. Flow-Based Programming makes programming more intuitive and simplifies the creation of musical structures**.** There are several parallels between programming and musical structures. M-Flow makes those structures visible, thus simplifying the creative process. This image compares similar musical structures written in (a) M-Flow, (b) Scratch, and (c) Python. Not only does M-flow look more intuitive, the skill level necessary to create a comparable Scratch program is higher, requiring knowledge of events and broadcasting.

not have a pre-existing interest in CS. The authors found that both Scratch and Blockly were too complex for these adolescents, and that students needed one-on-one adult support to be able to engage meaningfully with the program, a level of support that is not scalable. Regardless of their rigor and scalability, there is a paucity of reports investigating the efficacy of the approach in elementary schools. The need for heavy support even in older students, as shown by Heines and Walzer (2018) suggests that the programming paradigms utilized are not accessible and scalable enough to motivate this population.

C. Flow-Based Programming.

Most programming languages, including Scratch and Python-based Earsketch (used in the experiences described above), use an *imperative programming paradigm*, in which the commands are executed sequentially along the command line. In contrast to imperative programming, FBP is a programming paradigm in which the code is a direct visualization of the program's structure, boxes representing processes connected by arrows representing the program's flow. This intuitive approach allows people with little CS experience to rapidly develop functional applications. Because of its intuitiveness, a music-oriented FBP platform is well suited to be adopted in *fifth-grade general education* classrooms, thereby engaging students who might have never been interested in CS or exposed to it.

If music and computing share several features, the parallels are made visible in FBP, which is particularly well suited to represent musical structures. FBP platforms are very popular with electronic musicians, the most popular being Pure Data [18], and Max/DSP. Although accessible for adult naive coders, our experience indicates that Pure Data and Max/DSP are too complex for children, and the installation process is a significant barrier to implementation [10]. M-Flow seeks to create a simpler interface that allows children to start creating sounds immediately, and to deepen their engagement through the creative process, thus developing a learning experience with a low-floor and a high-ceiling [19].

D. Flow Based Programming in K-12 Computer Science Education.

When teaching coding, it is suggested that students produce flowcharts before they start working on their code on the computer [20]. Since the programming structure of FBP is similar to a flowchart, FBP allows students to more easily transform their thinking into code. This was the approach of RoboLab and NXT-G, which until recently were the standard platforms for controlling LEGO Mindstorms, and that were very popular among educators using robotics as a learning tool. In spite of their popularity, the efficacy of FBP for children was not evaluated, although anecdotal reports indicate that they were useful for teaching children as young as kindergarten [21].

E. CS in Mainstream Elementary Classrooms and Non-Specialized Teachers.

Elementary schools have been slow to take on the challenge of introducing CS in their classrooms. Google and Gallup (2016) found that the top two reasons they do not implement CS in elementary school is (a) lack of funds to hire a CS teacher and (b) too few teachers available with the skills to teach CS. One recommendation then is to develop and deploy strategies for existing teachers, non-specialized in CS, to become effective CS teachers through professional development. However, studies found that teachers and students felt difficult to absorb the computing concepts used even in the most accessible platforms such as Scratch, Blockly, or Alice [22], [23], and their implementation in the classroom requires intense training [24].

III. Mflow

Mflow is a flow-based programming platform that allows to create sound compositions through concatenating sound structures and control structures, or blocks. The sound blocks are either pre-made sounds, like different types of drums, or audio recordings created by the user. They can be dragged into the canvas from the upper bar, and manipulated once they are in the canvas. For example, the user can drag an "audio input" block into the canvas, and then click on the block to record sounds through the computer microphone. Once a sound-block finishes executing (the sound ends) its output triggers the next block down the flow line. Control blocks can trigger sound blocks (or other control blocks), for example, a loop block will trigger a chain of sound blocks, when that chain of sound blocks finishes executing, it will play again for the number of repetitions specified in the loop block. Another control structure is the *interval* block, an interval block can trigger a sound repeatedly for a number of times and with a given period, as specified by the user. For example, it can play a snare drum every 0.5 seconds, for 10 times. Importantly, with the interval block the sounds overlap (when a sound starts the other sound still persists). For better understanding, the reader is advised to experiment with Mflow at *author's-website.com*. Mflow allows to easily create common musical structures that would be much harder to implement in standard children platforms like Scratch. For example, Mflow allows to easily create convergent structures, in which a block is triggered by either of two other blocks. Creating a similar structure in Scratch would require the relatively advanced knowledge of *broadcasting*. Another example is the interval structure, which is common in music production, and would be much harder or perhaps even impossible to program in Scratch. Figure 1 allows to compare similar sound compositions implemented in Mflow, Scratch, and Python.

IV. Methods

Mflow is at a stage of development in which it can start to be used by children and teachers to create simple compositions. Our activities focus on reaching underrepresented children. We have given Mflow workshops at a college course for students pursuing teaching careers, at a STEAM teacher-training for elementary school teachers, and at a STEAM summer out-of-school program for 4th to 6th grade children. Figure 2 shows children at the STEAM summer camp (informed consent was obtained from the parents of the participants for media release). We have also trained teachers to implement Mflow and related activities in 4th grade classrooms. In all cases participants were enthusiastic about the tool. We have started collecting survey data to understand children-engagement. The data has not been fully analyzed yet. Therefore, we present this experience as a work in progress.

V. Future directions

Future directions involve embarking on a rigorous study of how using Mflow changes the students and teachers' attitudes

Fig. 2. Mflow has been started to be implemented in workshops. Front and back view of children creating sound compositions at a STEAM summer camp catering to mostly low-income children.

towards CS, and how Mflow can be used to teach basic CS concepts. This type of study will also allow to iteratively improve Mflow to maximize learning and engagement.

VI. CONCLUSIONS

Mflow is a tool built with the objective of being immediately engaging and intuitive, in such a way that non-specialized teachers can readily adopt it in the classroom, and that children can readily use for programming sound and making music. If, as expected, it is effective, it holds the potential of reaching a large number of children through the school system, exposing them to an authentic CS experience at an early age. Mflow is free and browser-based, it can be found at www.listeningtowaves.com/mflow.

AKNOWLEDGEMENTS

We thank Akshay Nagarajan for his guidance and comments.

REFERENCES

[1] J. M. Harackiewicz, J. L. Smith, and S. J. Priniski, “Interest matters: The importance of promoting interest in education,” *Policy Insights Behav. Brain Sci.*, vol. 3, no. 2, pp. 220–227, 2016.

[2] A. Wigfield and J. S. Eccles, “Expectancy–Value Theory of Achievement Motivation,” *Contemp. Educ. Psychol.*, vol. 25, no. 1, pp. 68–81, Jan. 2000, doi: 10.1006/ceps.1999.1015.

[3] C. Walkington and M. L. Bernacki, “Motivating Students by ‘Personalizing’ Learning around Individual Interests: A Consideration of Theory, Design, and Implementation Issues,” in *Motivational Interventions*, vol. 18, Emerald Group Publishing Limited, 2014, pp. 139–176. doi: 10.1108/S0749-742320140000018004.

[4] A. C. North, D. J. Hargreaves, and S. A. O’Neill, “The importance of music to adolescents,” *Br. J. Educ. Psychol.*, vol. 70, no. 2, pp. 255–272, 2000, doi: 10.1348/000709900158083.

[5] M. Bathgate and C. Schunn, “The psychological characteristics of experiences that influence science motivation and content knowledge,” *Int. J. Sci. Educ.*, vol. 39, no. 17, pp. 2402–2432, Nov. 2017, doi: 10.1080/09500693.2017.1386807.

[6] E. S. Adjapong and C. Emdin, “Rethinking Pedagogy in Urban Spaces: Implementing Hip-Hop Pedagogy in the Urban Science Classroom,” *J. Urban Learn. Teach. Res.*, vol. 11, pp. 66–77, 2015.

[7] C. Emdin, E. Adjapong, and I. Levy, “Hip-hop based interventions as pedagogy/therapy in STEM: A model from urban science education,” *J. Multicult. Educ.*, 2016.

[8] S. A. An, D. A. Tillman, R. Boren, and J. Wang, “Fostering Elementary Students’ Mathematics Disposition through Music-Mathematics Integrated Lessons.,” *Int. J. Math. Teach. Learn.*, 2014.

[9] S. A. An, T. Ma, and M. M. Capraro, “Preservice teachers’ beliefs and attitude about teaching and learning mathematics through music: An intervention study,” *Sch. Sci. Math.*, vol. 111, no. 5, pp. 236–248, 2011.

[10] V. Minces, “Developing and Popularizing STEM Online Tools: The Case of ‘Listening to Waves’ Tools for the Science of Music,” in *2021 ASEE Virtual Annual Conference Content Access Proceedings*, Virtual Conference, Jul. 2021, p. 36938. doi: 10.18260/1-2--36938.

[11] V. Minces, A. Booker, and A. Khalil, “Listening to Waves: Engaging Underrepresented Students Through the Science of Sound and Music,” *Connect. Sci. Learn.*, vol. 3, no. 4, Aug. 2021, [Online]. Available: https://www.nsta.org/connected-science-learning/connected-science-learning-july-august-2021/listening-waves-engaging

[12] A. Booker and V. Minces, “Harmonized Mutual Development through Exploring and Creating Sound,” in *International Conference of the Learning Sciences*, Hiroshima, Japan, 2022.

[13] A. Nagarajan, V. Minces, V. Anu, V. Gopalasamy, and R. R. Bhavani, “There’s data all around you: Improving data literacy in high schools through STEAM based activities,” *Fablearn Asia 2020*, 2020.

[14] A. Repenning, J. Zurmühle, A. Lamprou, and D. Hug, “Computational Music Thinking Patterns: Connecting Music Education with Computer Science Education through the Design of Interactive Notations.,” 2020, pp. 641–652.

[15] J. Freeman *et al.*, “Engaging underrepresented groups in high school introductory computing through computational remixing with EarSketch,” 2014, pp. 85–90.

[16] A. Jagiela, J. Laleman, P. Huschka, D. Besser, and A. Thomas, “Developing and Assessing a Music Technology and Coding Workshop for Young Women,” 2018.

[17] J. M. Heines and D. A. Walzer, “Teaching A Computer To Sing (TACTS): Integrating Computing and Music in a Middle School, After-School Program,” *J. Comput. Sci. Coll.*, vol. 33, no. 6, 2018.

[18] M. Puckette, “Max at seventeen,” *Comput. Music J.*, vol. 26, no. 4, pp. 31–43, 2002.

[19] M. Resnick and K. Robinson, *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. MIT press, 2017.

[20] A. Threekunprapa and P. Yasri, “Unplugged Coding Using Flowblocks for Promoting Computational Thinking and Programming among Secondary School Students.,” *Int. J. Instr.*, vol. 13, no. 3, pp. 207–222, 2020.

[21] B. Erwin, M. Cyr, and C. Rogers, “Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school,” *Int. J. Eng. Educ.*, vol. 16, no. 3, pp. 181–192, 2000.

[22] P. Blikstein and S. H. Moghadam, “Pre-college computer science education: A survey of the field,” 2018.

[23] S. Papadakis and M. Kalogiannakis, “Evaluating a course for teaching introductory programming with Scratch to pre-service kindergarten teachers,” *Int. J. Technol. Enhanc. Learn.*, vol. 11, no. 3, pp. 231–246, 2019.

[24] U. Wolz, M. Stone, K. Pearson, S. M. Pulimood, and M. Switzer, “Computational thinking and expository writing in the middle school,” *ACM Trans. Comput. Educ. TOCE*, vol. 11, no. 2, pp. 1–22, 2011.