Transforming Computing Education Through Integrated Learning: A 3D Programming Course For Undergraduate Students

Bowu Zhang[1], **and Mira Yun**[2]
[1]Department of Mathematics and Computer Systems, Mercyhurst University, Erie, PA, USA
[2]Department of Computer Science & Networking, Wentworth Institute of Technology, Boston, MA, USA

Abstract— *Computing is increasingly used in a wide variety of fields, i.e., to compose music, to analyze literature and to detect the century-old remains in archeology research. Those interests in computing have led to a growing global trend in the college education where students and faculty are engaged in computer-based study of human-computer interaction, digitization, advanced visualization and other research techniques. This paper explores the integrated learning in computing and other non-computing disciplines by looking at an undergraduate course of 3D programming. Students in this course can experience programming through multiple lenses so that they can have a comprehensive understanding of not only programming concepts, but also the knowledge of applying the computing tools to issues in their own fields. In particular, we demonstrate a number of student projects from this course to show that this integrated learning equips students with valuable tools and knowledge that will build their skill sets, enhance their careers and prepare them for complicated issues facing the world today.*

Keywords: Computing Education, 3D Programming, Integrated Learning

1. Introduction

Computing has played a significant role in a wide variety of fields including mathematics, biology, chemistry, art, music and history [1]–[7]. Nowadays almost every field incorporates substantial work and research involving scientific computation [8], data processing and analysis [9], and software design [10], etc. For example, biological scientists may need computing to analyze the protein structure in a human cell [11], while archeology professors may use computing to estimate the age of historic remains [12]. Today, computing enables and inspires new ways to create and interact in every aspect of our lives, while those non-computing fields inform how we might use new technology to enrich our human experience. Therefore, the computing courses constantly attracts many students who are not CS majors but have passion for learning useful computing tools to support interests in their primary fields. However, traditional computing courses have been referred by many students as abstract, difficult, and boring [13], thus many gave up in the middle or just stayed away from computing courses.

In this paper, we discuss the integrated learning in computing and other disciplines. We keep track of the progresses in an undergraduate computing course to see how we are able to make the learning of computing courses enjoyable by integrating interesting topics from a variety of fields. The integration will allow students to gain knowledge from distinct fields. Integrated learning is the core component of today's college education [14]–[16]. In this constantly changing world, high-end education should prepare students for a life where everything is increasingly interconnected. In other words, students should have a comprehensive understanding of not only their own field, but also a broad knowledge of the changing world. We will look at one undergraduate computing course to display that how the integration complements a variety of other majors, including Anthropology, Biochemistry, Biology, Chemistry, Earth Sciences, Physics, Psychology, and Sociology. The class of "Introduction to 3D programming" is a 3-credit course for non-cs majors at the Mercyhurst University. This course is designed to provide a broad exposure to the field, increasing employability in a variety of settings related to computing. Similar to most of the computing courses, "Introduction to 3D programming" includes a final project where students make contacts with practitioners in their field, complete a research project, and present their work. We will show that courses such as the 3D programming equip students with valuable tools and knowledge that will enhance their careers or prepare them for continued study in graduate and post-baccalaureate professional programs. These give our students the opportunity to cross the technology divisions, so that students will get ready for the transdisciplinary challenges they will face in the world ahead.

The rest of the paper is organized as follows. Section 2 presents the most related work. Sections 3 discusses how the integrated learning is implemented in a 3D programming course for undergraduate programs. A number of course projects are detailed in section 4, and evaluated in terms of integration with non-computing disciplines. Finally, we conclude our work in section 5.

2. Related Work

The computing education has attracted attentions from educational professionals of all stages including the early childhood development, the elementary education, the secondary education and the high education, as computing is increasingly used in every division of our lives. Various methods have been discussed to stimulate student interest in learning computing courses. Based on the course topics, previous research on computing education can be divided into

two categories: 1) research on traditional computing courses including programming, software design and data analysis [17]–[19]; and 2) research on integrated computing courses which integrates computing with other disciplines such as art, history and biology [20]–[22]. Work on the first category mainly aim to simplify and visualize the abstract concepts in hard core computing courses. Authors in [23] suggested to add more user-friendly interfaces in programming environment to make users/students comfortable with programming. In [24], authors evaluated the popular programming languages used in education, and they concluded that the object-oriented language shows superiority over other languages in computing education, as students can easily relate the concepts to their lives. A variety of new courses and new education tools have been proposed in the research of the second category. [25] introduced a new major named the art in computing where music, visual art, and theatre are involved in the computing education, and they reported that this new major significantly increased participation in computing, especially attracted students who are not necessarily strong in mathematics. A 3D mapping course in Geology was introduced in [26]. The authors demonstrated a 3D virtual geology field trip and discussed the development of the 3D educational environments. Similar courses have been brought in by [27], [28] where cutting-edge techniques are employed to motivate students learning in computing courses. In this paper, we focus on the integrated learning in computing education. A 3D programming course is analyzed to demonstrate how different disciplines can be integrated into computing. Student learning outcomes support that this integrated learning equips students with valuable tools and knowledge that will build their skill sets, enhance their careers and prepare them for complicated issues facing the world today.

3. Teaching 3D Programming in Alice

The course of "Introduction to 3D Programming" is a 3-credit class at the Mercyhurst University. The majority of student population come from mathematics, business intelligence, music and dance. A substantial group of students do not have any prior programming experiences. In the course, we study the 3D programming in Alice. Alice is a open-source programming environment to create animated movies and videos. Different from the traditional programming languages such as c, c++, and java, that emphasize the semantics heavily, Alice adopts a virtual world where users can simply "drag-and-drop" instructions to create programs without worrying about syntax errors. In Alice, every item is modeled as a 3D object which can be placed into the virtual world, and every object has built-in functions and properties that can be used in programs to let them perform operations that users want. Users can view how the program is executed through the animated movie, resulting in a straightforward understanding of how the programming concepts are applied to objects in the virtual world. Users also can access sounds, images, and texts, enabling them to learn programming from varied perspectives. In addition, users can create their own objects and functions and share with the entire Alice community. In a word, Alice makes the programming easy and interactive, and therefore is widely used in the introductory level programming courses [29], [30].

The programming topics discussed in this course include objects, variables, functions, selection structure, loop structure, and mouse and keyboard events. Typically, students have opportunities to practice every programming concept taught by creating animated movies and simple video games. Exercises about objects include Alice and Rabbit [31]. Exercises about variables include simple mathematic operations. Exercise about selection structure include the grading problem [31]. Exercises about loop structure include the repetitive song of crazy scientist [31]. Exercises about mouse and keyboard events include the helicopter simulation problem [31]. A final project is required at the end of semester that integrates aspects of all programming concepts discussed and demonstrates understanding of the synthesis among them.

Many efforts are made to set up the integrated learning in 3D programming and other seemingly unrelated disciplines. The integration begins with the spirit of object oriented programming, which constitutes the foundation of the Alice software. Since everything is made visualized in the Alice virtual world, it is easy for students to relate the programming to their personal lives. The examples and exercises used in the class are also based on the real world scenarios such as daily talk, gaming, and shopping, which show students the possibility to model every aspect of our life in 3D programming. The final project is designed to frame opportunities for students, after exploring course offerings, to begin a work they may never consider before. Topics ranging from dance choreography to biology DNA pattern recognition allow students to tailor the programming to fit their interests. Compared with that of other programming courses (Java, C++ or Python), student satisfaction with the course of 3D programming is around $\%10$ higher. This course gains the general popularity among non-CS major students.

4. Course Work

The course of "Introduction to 3D Programming" emphasizes hands-on exercises heavily. Programming exercises under all kinds of scenarios are assigned after learning every new concept. Those exercises help students build the skills and allow students to explore the increasing connections between the programming and the life/field they are familiar with. More importantly, students will need to complete a final project that integrates aspects of all discussed concepts and demonstrates understanding of the synthesis among them. Various topics ranging from dance choreography to biology DNA pattern recognition have been chosen in the past years. An interesting observation is that students tend to tailor the programming to fit their interests in their primary fields. Music majors often prefer to include complicated sounds effects in their projects, while mathematic majors like to

(a) (b) (c)

Fig. 1
(A) INDIVIDUAL DANCE; (B) PAIR DANCE; (C) GROUP DANCE.

play with statistics (e.g., random number question/quick math problem). Students are motivated to apply what they learn about programming to topics they deal with most of the time in college. Those projects significantly enhance their understanding in both programming and their own fields. A number of examples projects are discussed as shown in the following sections.

4.1 Dance Choreograph Project

Music majors and dance majors tend to dance choreography projects. Dance movements in the projects are often designed based on the real dance programs and implemented by manipulating the body of human figures in Alice. Also movements need to be adapted and timed to fit the music. In addition, to make the dance ”flow”, transitions are often included. The dance project often involves a number of figures including human and animals dancing on a stage. Control structures such as the selection and the loop statements are often used. The basic program flow is described as follows: There is a group of dancers on the stage. The dancers begin to dance when the music start, and they will stop once the music ends. We classify the dance projects into two categories based on the number of dancers: 1) individual dance where each dancer dances independently; 2) group dance where dancers are implemented as a collection, and perform similar dance movements simultaneously or in turns. Detailed implementations are displayed below.

First, let’s focus on the individual dance program. The program begins with adding a dancer to the virtual world from the local gallery. We may need to adjust the camera so that the dancer is right in front of the camera. Then in the object tree, select the dancer and create an object method named ”dancing”. Next, add instructions to the dancing method to let the dancer move. We may pose the legs, arms and other parts of body to create different movements. For example, we can let the arm swing by dragging and dropping the statement ”dancer-upper-body-left arm.turn right 1/2 revolution”, or we can let the dancer spin by the statement ”dancer.turn right 1 revolution”. Between movements, we let the camera zoom in and zoom out to make a special visual effect using the dummy objects: We drop invisible dummy objects at the position where we want the camera to be; Then let the camera move by specifying the target as the created the dummy object. Once we’ve made a step or two that we are pleased with, we can make our dance method repeat itself by using a loop. Suppose we have a dance method that works nicely for a step or two as in figure 2

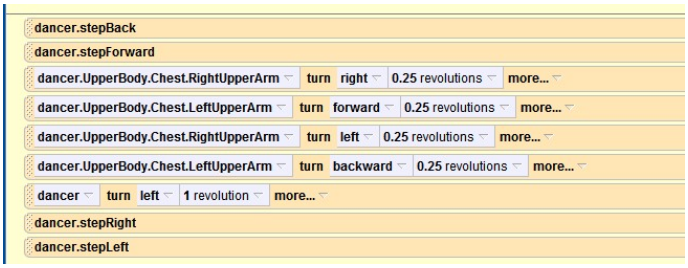

Fig. 2
INDIVIDUAL DANCING METHOD

To make it look like a real dance, we need to use some randomness. The world has random functions that can produce random numbers within a fixed range. We can let the dancer move a random distance forward or backward or let him/her spin a random number of revolutions. Test the method by dragging it to the world.my first method. Go back to the dancing method and let the dancer repeat the routine movements by using the loop. Here we can specify a fixed number of iterations or have the dance method keep repeating over and over while the music plays. One way to make the dance repeat indefinitely is to call the dancing method within a loop in the my first method. All we need do is to add two lines in the world. my first method 3

To insert music into the program, we can use the object.play() method to import existing songs or you can use recorder to make your own dance music!

Group dancing requires synchronized movements. For example, dancers need to turn right simultaneously, or as some

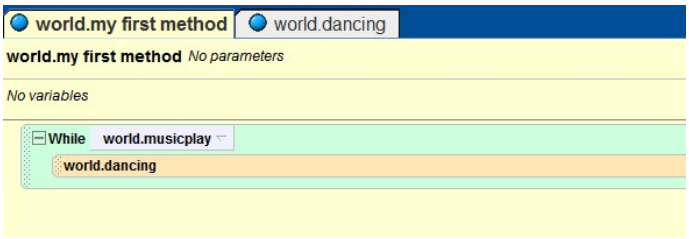

Fig. 3
REPETITION IN WORLD.MY FIRST METHOD

dancers move a leg forward, the others should move the other leg back, and so on. An easy way to implement the group dancing is to put all dancers in a special data structure called list. Lists are data structures used to group and to organize objects. Since list items can be processed one-by-one or all at once (The control structures for all in order and for all together correspond to these), by using lists, we can save on writing code if we want to process similar actions on a group of objects. What we need is to write a single dancing method and apply it to each list item in turn rather than to write a method for each item or having a separate call for each item. We can access a particular object by its index in the list. To create a list of dancers, we need to add a number dancers to the virtual world (Fig. 4).

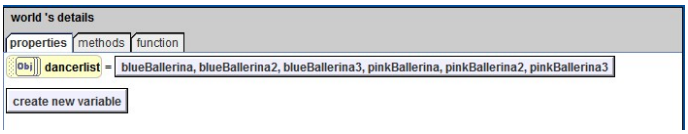

Fig. 4
A LIST OF DANCERS

Similar to the individual dancing method, in the group dancing method, we just need to create a few steps. We can do with as few as Fig. 5 and 6(a)

To apply the above the dancing method to the list, we need to either call "for every item in the list all together" or "for every item in the list all in order" as shown in the figure 6(a). Check the dancing method to make sure there are no errors. Then we can call the dancing method in the my first method repeatedly to fit with the music.

Special stage light and camera movement can be adopted to create magic stage effects. We may also insert images or pictures of friends/families to make the program more fun.

4.2 Soccer Player Project

Mathematic majors and sport team players often dive into sports related projects. In these projects, Alice is used to model a common scene on the field, such as a ice hockey game or base ball game. A typical example comes from the soccer ball game where the user controls a soccer player to shoot the goal. Projects that fall in this category are very

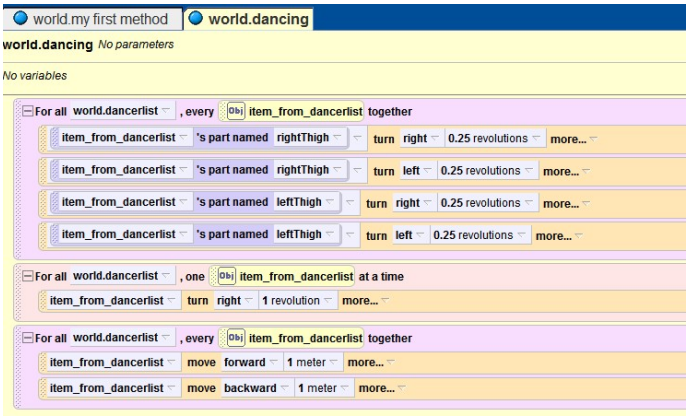

Fig. 5
DANCING METHOD OF TWO DANCERS

interactive users can enjoy playing the game by the keyboard or the mouse while a score object is used to keep track of the progress he or she has made. The basic program flow is described as follows: There is(are) a (or a number of) soccer player(s) and a goal keeper on the soccer field. The soccer player is controlled by a set of particular keys or the mouse to shoot the goal. The user can determine the direction the ball kicked into and the speed that the ball will travel at, while the direction and the distance the goal keeper jumps to defend his goal is controlled by the computer. Player's statistics are often considered in the game, i.e, the chance a soccer player uses his left foot to shoot the goal or the average distance that a ball can travel after kicked by a particular player, which leads to variations to the game. Random functions are often employed to reflect the sport statistics, such as the direction the player may kick the ball or the height the goal keeper to jump. Here we use the random number generator in Alice to implement the randomness in this game. In order to let player kick the ball to the user specified direction, we first need to place a number of invisible circles (Assume there are N circles) which serve as targets within the gate. Then we assign each circle a unique number ranging from $[1, N]$ as its ID so that every circle represents a direction where the player can shoot the gate. What the program does is to generate a random integer on a scale of 1 to N, and let the soccer player shoot the circle whose ID is equal to the random number generated. Or the user can click on one of circles to trigger the mouse click event which make the ball fly to that circle. The random number is also used on the goal keeper to prevent the player from scoring. Normally, the goal keeper can jump in 3 (or 5) directions: up, left and right (upper left, upper right), each of which is represented by an integer from $[1, 3]$. Then the program every time generates a random integer within $[1, 3]$ to indicate the direction the goal keeper to move, and another random number to indicate the height to jump.

Detailed implementations are displayed below. First, add a player and a goal keeper to the virtual world. Adjust their

(a) (b)

Fig. 6
(A) DANCING METHOD OF A LIST OF DANCERS; (B) SOCCER PLAYER.

initial positions to put them right in front of the camera. Then drop a dummy object to label the starting position of the ball so that the ball can be placed back for the next game every time it is kicked off. Create a world level method named "playershoot" to let the user shoot the goal. In the new method, we may let the user use the mouse or the keyboard to choose a circle to kick the ball (invisible circle). Manipulate the body of the player to perform the action of shooting. Let the ball move to the goal(circle) while let the keeper jump simultaneously to stop the ball. The method may look like Fig. 7

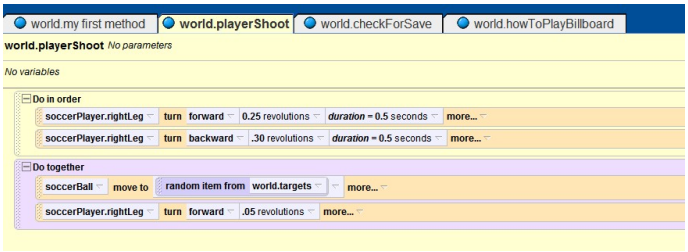

Fig. 7
THE PLAYERSHOOT METHOD

We want the user to play this game a number of times. Therefore, we add statements in the world.my first method to call the "playershoot" method repetitively (Fig. 8). On the other hand, we create an object to keep track of scores the user has got and display the value in the virtual world. Thus another world level method is created to detect if the goal keeper catches the ball. The value of the score object will be

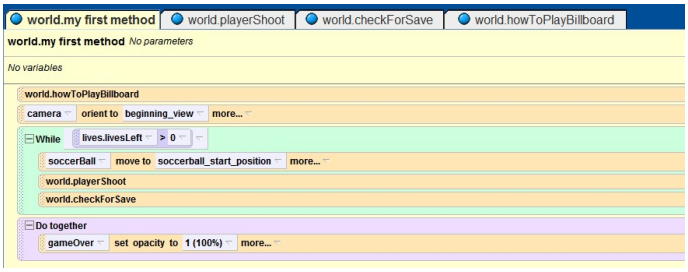

Fig. 8
THE WORLD. MY FIRST METHOD OF SOCCER PLAYER PROGRAM

incremented by one if the ball is not catched. An example method is shown in the figure below:

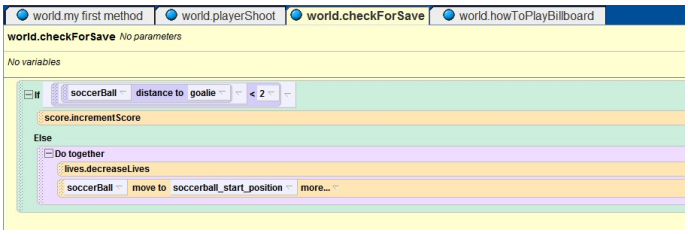

Fig. 9
KEEPING SCORE OF THE SOCCER PLAYER

This project can be easily modified to a multiple-player game by assigning different players different keyboard events. More statistical patterns can be applied to the multiple-

player game since the there might be many variations to the collaborations among players. Professional soccer games such FIFA World Game are based on the same set of design principles. Note that the more players there are in the field, the more random the result will be. This is consistent with the probability theory that the number of possible cases increases as the input size increases. On the other hand, multiple story threads can be implemented in this project by adding the selection control structures. However, this needs careful designs of logic to differentiate the conditions for each possible result.

5. Conclusion

In this paper, we explore the integrated learning in computing courses in undergraduate programs. Integrative courses encourage students to explore important topics through multiple lenses so that they understand how to approach complicated issues facing the world today. In particular, we looked at one programming course for non-cs majors. Course projects were analyzed to show that integrated learning enable students to create their own work in a thoughtful way so they can explore their interests, build skill sets, engage their intellects and learn from a variety of fields.

References

[1] D. Blank, J. S. Kay, J. B. Marshall, K. O'Hara, and M. Russo, "Calico: a multi-programming-language, multi-context framework designed for computer science education," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 2012, pp. 63–68.

[2] V. Ananthanarayanan and W. Thies, "Biocoder: A programming language for standardizing and automating biology protocols," *Journal of biological engineering*, vol. 4, no. 1, pp. 1–13, 2010.

[3] R. Libeskind-Hadas and E. Bush, "A first course in computing with applications to biology," *Briefings in bioinformatics*, vol. 14, no. 5, pp. 610–617, 2013.

[4] J. C. Rubinstein, "Perspectives on an education in computational biology and medicine," *The Yale journal of biology and medicine*, vol. 85, no. 3, p. 331, 2012.

[5] J. Bresson, C. Agon, and G. Assayag, "Openmusic: visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 743–746.

[6] W. Scacchi, "The future of research in computer games and virtual worlds," *Institute for Software Research, University of California, Irvine, Irvine*, 2012.

[7] R. Wille, S. Offermann, and R. Drechsler, "Syrec: A programming language for synthesis of reversible circuits," in *System Specification and Design Languages*. Springer, 2012, pp. 207–222.

[8] P. Wallisch, M. E. Lusignan, M. D. Benayoun, T. I. Baker, A. S. Dickey, and N. G. Hatsopoulos, *MATLAB for neuroscientists: an introduction to scientific computing in MATLAB*. Academic Press, 2013.

[9] F. Erickson, "Qualitative research methods for science education," in *Second international handbook of science education*. Springer, 2012, pp. 1451–1469.

[10] Y. Cai, D. Iannuzzi, and S. Wong, "Leveraging design structure matrices in software design education," in *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*. IEEE, 2011, pp. 179–188.

[11] E. H. Kellogg, A. Leaver-Fay, and D. Baker, "Role of conformational sampling in computing mutation-induced changes in protein structure and stability," *Proteins: Structure, Function, and Bioinformatics*, vol. 79, no. 3, pp. 830–838, 2011.

[12] Y. Kondo, S. Kadowaki, H. Kato, M. Naganuma, A. Ono, K. Sano, and Y. Nishiaki, "Network computing for archaeology: a case study from the replacement of neanderthals by modern humans database project," *Revive the Past*, p. 217, 2011.

[13] R. Pau, W. Hall, and M. Grace, "Its boring: female students experience of studying ict and computing," *School Science Review*, vol. 92, no. 341, pp. 89–94, 2011.

[14] L. D. Fink, *Creating significant learning experiences: An integrated approach to designing college courses*. John Wiley & Sons, 2013.

[15] M. A. Honey, M. Hilton, *et al.*, *Learning science through computer games and simulations*. National Academies Press, 2011.

[16] J. Duffy, L. Barrington, C. West, M. Heredia, and C. Barry, "Service-learning integrated throughout a college of engineering (slice)." *Advances in Engineering Education*, vol. 2, no. 4, 2011.

[17] C. A. Shaffer, T. L. Naps, and E. Fouh, "Truly interactive textbooks for computer science education," in *Proceedings of the 6th Program Visualization Workshop*, 2011, pp. 97–103.

[18] J. B. Fenwick Jr, B. L. Kurtz, and J. Hollingsworth, "Teaching mobile computing and developing software to support computer science education," in *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 2011, pp. 589–594.

[19] M. Sahami, M. Guzdial, A. McGettrick, and S. Roach, "Setting the stage for computing curricula 2013: computer science–report from the acm/ieee-cs joint task force," in *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 2011, pp. 161–162.

[20] A. Schäfer, J. Holz, T. Leonhardt, U. Schroeder, P. Brauner, and M. Ziefle, "From boring to scoring–a collaborative serious game for learning and practicing mathematical logic for computer science education," *Computer Science Education*, vol. 23, no. 2, pp. 87–111, 2013.

[21] L. Ma, J. Ferguson, M. Roper, and M. Wood, "Investigating and improving the models of programming concepts held by novice programmers," *Computer Science Education*, vol. 21, no. 1, pp. 57–80, 2011.

[22] J. B. Labov, A. H. Reid, and K. R. Yamamoto, "Integrated biology and undergraduate science education: a new biology education for the twenty-first century?" *CBE-Life Sciences Education*, vol. 9, no. 1, pp. 10–16, 2010.

[23] O. Shaer and E. Hornecker, "Tangible user interfaces: past, present, and future directions," *Foundations and Trends in Human-Computer Interaction*, vol. 3, no. 1–2, pp. 1–137, 2010.

[24] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. Jacob, "Comparing the use of tangible and graphical programming languages for informal science education," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 975–984.

[25] B. Manaris, R. McCauley, M. Mazzone, and W. Bares, "Computing in the arts: a model curriculum," in *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 2014, pp. 451–456.

[26] S. J. Whitmeyer, "Community mapping in geology education and research: How digital field methods empower student creation of accurate geologic maps," *Geological Society of America Special Papers*, vol. 486, pp. 171–174, 2012.

[27] S. Bogaerts, K. Burke, and E. Stahlberg, "Integrating parallel and distributed computing into undergraduate courses at all levels," in *First NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-11), Anchorage, AK*, 2011.

[28] Y. Khmelevsky and V. Voytenko, "Cloud computing infrastructure prototype for university education and research," in *Proceedings of the 15th Western Canadian Conference on Computing Education*. ACM, 2010, p. 8.

[29] W. P. Dann, S. Cooper, and R. Pausch, *Learning to Program with Alice (w/CD ROM)*. Prentice Hall Press, 2011.

[30] T. Daly, "Minimizing to maximize: an initial attempt at teaching introductory programming using alice," *Journal of Computing Sciences in Colleges*, vol. 26, no. 5, pp. 23–30, 2011.

[31] J. Adams, *Alice in Action: Computing Through Animation*. Cengage Learning, 2006.