PoseToCode: Exploring Design Considerations toward a Usable Block-Based Programming and Embodied Learning System

Nisha Chatwani*
Viterbi School of Engineering
Dept. of Computer Science
University of Southern California
Los Angeles, USA
email: nchatwan@usc.edu

Chloe Kuo*
Viterbi School of Engineering
Dept. of Computer Science
University of Southern California
Los Angeles, USA
email: cmkuo@usc.edu

Thomas R. Groechel*
Viterbi School of Engineering
Dept. of Computer Science
University of Southern California
Los Angeles, USA
email: groechel@usc.edu

Maja J Matarić
Viterbi School of Engineering
Dept. of Computer Science
University of Southern California
Los Angeles, USA
email: mataric@usc.edu

***Abstract*—As interest and need for computer science skills continue to grow, educators have been teaching block-based programming languages to young students. Such languages are intuitive and widely available, but have not as yet explored** *embodied (i.e., kinesthetic)* **learning techniques. To explore the use of embodied learning for teaching programming, we created PoseToCode (P2C), a web-based embodied learning programming activity in which students perform physical poses with their arms to create code blocks that control an on-screen robot. We first conducted a pilot study to identify key P2C design considerations. We used those insights to finalize P2C and then deployed it in a local 5th grade class (n=24, 10 students completed all surveys) to measure P2C usability, compare it to a seated block-based programming activity (Code.org), identify new design considerations, and explore if P2C increased student curiosity in learning coding. Our results support P2C as a usable system design, show no difference in student curiosity when compared to a traditional block-based programming language, and identify potential improvements to our design considerations for future activities that integrate embodied learning with block-based programming. This work aims to inform the process of creating usable and effective systems for block-based programming and embodied learning activities.**

***Keywords*—embodied learning; block-based programming; education; human-computer interaction**

I. Introduction

Computer programming is a challenging subject to learn; educators have been exploring ways to make it more accessible to students [1]. With computer programming skills (i.e., coding skills) becoming increasingly more desirable for students interested in pursuing science, technology, engineering, and math (STEM), block-based coding activities are often used to introduce basic coding concepts through a simple interface. Block-based programming exercises have become widespread in recent years as programming is being taught to even younger students [2]. *Block-based programming* is the use of dragging and dropping code blocks that snap together in order to make a program or series of instructions in a natural language [3]. Code blocks can have slots allowing for other blocks to be inserted, and code blocks can also represent higher level coding structures, such as loops, conditional statements, and functions. Past research has compared block-programming to text-programming activities in classrooms [4] and identified properties of block-based coding activities that contribute to the activity's convenience [5], such as the plug-and-play functionality and the ability to clearly see what each code block accomplishes, making the learning activity more intuitive than text-based programming activities.

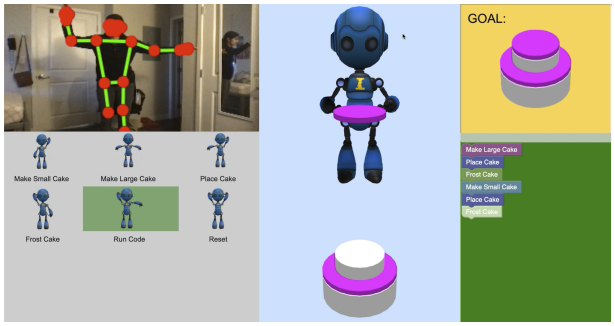


Figure 1: PoseToCode Exercise 3: Build a Cake. The user (top left) must physically perform poses (bottom left) to create a sequence of codeblocks (bottom right), which, when executed, instruct the virtual robot (middle) to construct a cake (top right). The robot is displayed near the top of the screen because the following exercise involves programming objects underneath it.

Block-based programming languages are focused on mentalistic (i.e., non-kinesthetic) learning. As the COVID-19 pandemic has demonstrated, mentalistic learning can become difficult and tiresome for many students [6]. *Embodied learning*, on the other hand, integrates the body with the mind through movement based-learning. Research in embodied learning has shown the general approach to have a positive effect on

 ISBN: 978-1-61208-982-9

children's learning outcomes [7]. Inspired by this literature, we aimed to develop a block-based programming environment that supports embodied learning, since, to the best of our knowledge, no such kinesthetic block-based programming environment exists.

To address this gap, we created PoseToCode (P2C), which combines embodied learning with block-based programming to create a usable system with the end goal of increasing student curiosity and understanding of programming. P2C (Figure 1) is an embodied learning block-based coding activity where students perform poses to create code blocks that guide a virtual robot through a series of exercises. We created a set of design considerations for P2C discovered in informal pilot testing. To validate P2C design, we deployed it in a local 5th grade classroom of 24 students. We measured usability and performed a comparison to a traditional block-based programming activity, and analyzed the study results to develop future design considerations for making P2C more intuitive and easier to use. The results of the study support P2C as a usable design and identify improvements for future coding languages that integrate embodied learning and block-based programming. We have made P2C open-source and have posted a publicly accessible repository [8] and demonstration [9].

II. Background

This research builds on past work relating to block-based programming and embodied (i.e., kinesthetic) learning.

A. Block-Based Programming

Block-based programming exercises have become very popular because coding blocks are usually easier to interpret than text-based code [5]. With interest in STEM education dropping off in middle and high school [10], block-based programming provides a way to introduce coding to students at a young age in order to nurture curiosity about computer science. Block-based programming exercises allow novice programmers to develop quickly because they use *component-based programming*, wherein code instructions are partitioned into distinct functionalities [11]. This approach promotes reusability of software components, as well as flexibility in replacing components with code that has the same functionality [12], making it easier for students without extensive programming experience to put components together and build more complex programs.

The most popular block programming environments like Scratch [13] (Figure 2) and Code.org [14] provide users with the opportunity to explore different parts of programming such as loops and conditional statements, allowing users to make games and create artwork with ease [5]. The plug-and-play nature of block-based programming makes it more appealing to students compared to text-based programming.

Recent work has compared student perceptions of block-based programming and traditional programming. A 5-week study in a high school computer science class revealed that the students using a block-based coding environment retained more programming knowledge and showed greater interest in learning coding than the students using a text-based programming environment [4]. Studies have also been conducted to explore what students find beneficial and effective in block-based programming environments [5]. The results showed that students recognized the individual block colors, natural language block labels, and drag-and-drop functionality of block programming environments made them visually more clear [5]. Our work aims to utilize these known positive aspects of block-based programming and integrate them with embodied learning.

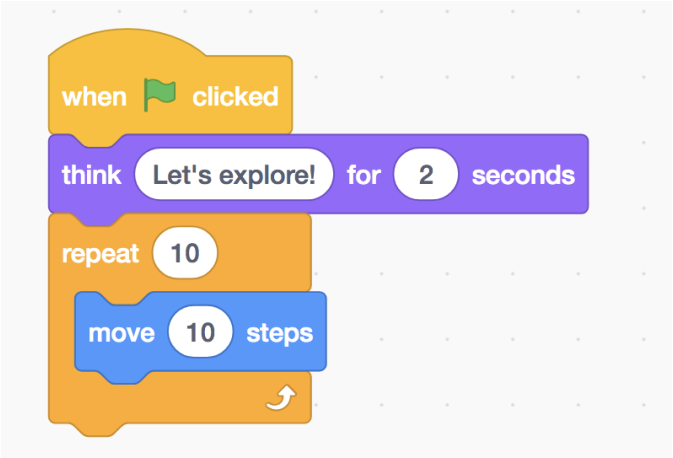


Figure 2: Scratch: A popular block-based programming environment that allows users to program virtual characters to execute sequences of actions [13].

B. Embodied Learning

Embodied learning has been recognized as an alternative to traditional mentalistic (i.e., non-kinesthetic) learning [7]. Embodied learning (i.e., kinesthetic learning) is the set of "pedagogical approaches that focus on the non-mental factors involved in learning, and that signal the importance of the body and feelings [15]."

Embodied learning has been shown to promote engagement by increasing student curiosity about topics they are learning. Past work has explored teaching methods that promote engagement through embodied learning in the classroom [16]. A study on gesture-based learning used a Microsoft Kinect sensor and application to create an interactive learning environment in which students moved their body to learn about the solar system [17]. The study results showed that gesture-based learning resulted in students wanting to learn more, and they felt more engaged in the learning process. The positive outcomes of gesture-based learning–increased student engagement and curiosity about the topic–inspired our work.

Embodied learning has also been shown to be beneficial in improving youth cognition and academic performance. A study conducted in elementary school classrooms used motion-based educational games to implement embodied learning, and demonstrated positive impacts on the students' cognitive skills, including their short-term memory, and their linguistic academic performance [16].

 ISBN: 978-1-61208-982-9

Given its potential, embodied learning has been explored in computer science education. One study examined the effects of teaching middle and high school students introductory programming concepts through a gesture-based interface. While the results were not significant, the results validated the promise of engaging students in programming through embodied learning activities [18]. Our work further expands on the potential benefits of embodied learning in early computer science education since the act of programming is kinesthetic in P2C.

III. PoseToCode Design

This section outlines the technical design of PoseToCode and pilot study insights leading to design considerations.

A. Technical Design

PoseToCode (P2C) enables users to create and execute code by posing with their body. The P2C interface (Figure 3) has the user's **video feed (A)** in the top left corner where the Mediapipe pose detection library [19] draws lines showing landmarks on the user's body. Directly below the video feed, a grid shows a set of **progress bars (B)**, with images depicting the virtual robot performing a pose; each pose image is labelled to indicate what code block it will create. The **virtual robot (C)** is in the middle of the screen, and in the top right corner, there is an image of the **goal state (D)** indicating what the code blocks should produce to complete the exercise. Lastly, the Blockly workspace with the **code blocks (E)** created so far is on the right of the visual interface.

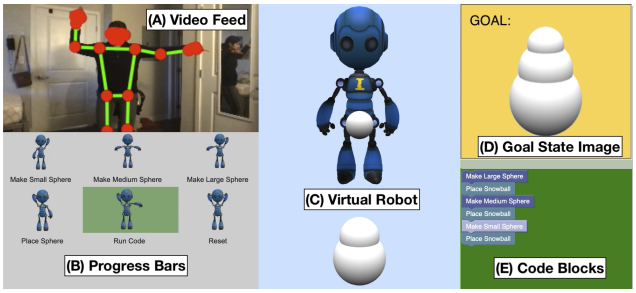


Figure 3: PoseToCode Exercise (Level 2): Student (top left) posing to create code blocks (bottom right) that guide the virtual robot to build a snowman. A) Flipped video feed with the MediaPipe detected pose drawn. B) Grid of pose images and progress bars for each pose. C) Virtual robot that performs instructions from code blocks. D) Goal state image. E) Blockly workspace with crated code blocks.

In P2C, the user's poses are recognized with Google Mediapipe pose detection software [19] executing at 60Hz. At time t, the pose key points are run through a deep neural network to map each arm to {HIGH, MED, LOW, NONE}. Based on the arm mappings, the corresponding pose progress bar (e.g., {Left: HIGH, Right: MED} $\rightarrow$ "Run Code") fills at a rate of $1.2 * \Delta(t, t-1)$ while all other progress bars decay at the rate of $0.8 * \Delta(t, t-1)$. When the user holds a pose for 4 seconds, a custom Blockly [20] code block that corresponds to that pose is created. Code blocks are instructions that control a virtual robot on the screen; they can be created, erased, and executed. If the user's executing code reaches the goal state, they are moved to the next exercise. Alternatively, if the code fails to reach the goal state, the user must continue attempting the same challenge until they succeed or their time spent on the activity reaches the 10-minute limit. We chose that time limit based on pilot testing the activity and ensuring that it fit into the available classroom time.

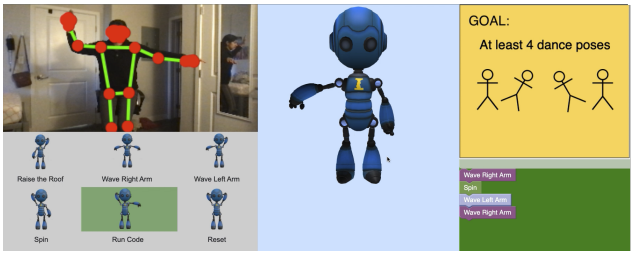


Figure 4: PoseToCode Exercise 1: Choreographing a dance routine for the virtual robot

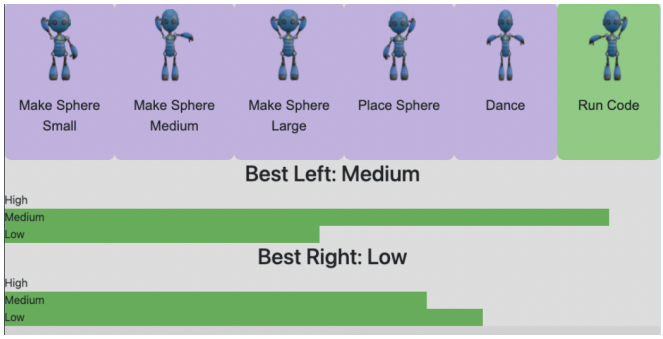
A full P2C activity is composed of a series of three challenges for the user to complete within 10 minutes. The process consists of creating code blocks by posing and then executing all of the created code blocks. To complete the first challenge (Figure 4), the user must instruct the virtual robot to perform a dance routine of four or more dance moves. To complete the second challenge (Figure 3), the user must instruct the robot to build a snowman. To complete the third challenge (Figure 1), the user must instruct the robot to construct a frosted three-tiered cake. After all three challenges are completed, the user moves on to a freeplay mode challenge until the time limit is reached.

B. Pilot Study Insights About Design Considerations

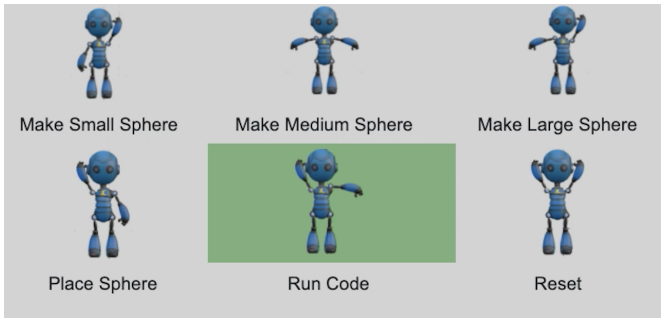
We performed an informal pilot test of P2C with two engineering students at our university, using the snowman building exercise. We gained the following insights about P2C design considerations:

- *Accessibility across low-end computers*: Schools use a variety of laptops with a wide range of processing power and operating systems. Therefore, we created PoseToCode to be operating-system agnostic, executing via the web at 60Hz on the currently most popular Chromebook with a webcam.
- *Accessibility at a distance*: Multiple interface elements are needed to accommodate students who stood away from the computer while completing the programming exercises.
  - *Visibility*: Code blocks must have clear visibility, requiring them to be large, and therefore also enforcing having fewer on-screen components.
  - *Webcam as the only input source*: We piloted different interfaces (e.g., space bar pressing) but participants did not wish to step to and away from the computer, preferring to only use their body pose as input.

 ISBN: 978-1-61208-982-9



(a) Original pose key with individual bars for each arm.



(b) Updated pose key where pose bars fill up directly.

Figure 5: Original (a) and updated (b) pose key designs. The original design showed each individual arm state and a pose map. Participants found this difficult to map the arm states to each pose. The updated design directly filled up each respective pose.

- *Real time input and feedback*: The webcam continually collects input from the user at 60Hz, resulting in the following considerations:
  - *Direct pose key* The original pose key (Figure 5a) consisted of left and right arm states ({HIGH,MED,LOW,NONE}) with only the accumulated states shown to the user, not the state of the best current pose. When the progress bar corresponding to the state from each arm reached 100% completion, the mapped pose produced the corresponding code block. Many pilot users cited this as complicated as they needed to read the arm states and then the respective texted-based pose map. To address this, we developed an updated direct pose key (Figure 5b) that combines the state of the arms into one pose state that corresponds directly to a progress bar.
  - *Reactive and persistent pose bars*: To increase reactivity, the best pose progress bar at time t increases at the rate of $1.2 * \Delta(t, t-1)\%$ while all other bars decay at the rate of $0.8 * \Delta(t, t-1)\%$. The slower decay rate (0.8) compared to the growth rate (1.2) allows for a semi-persistent pose meter, because the progress bars grow faster than they shrink. For example, when a user is doing run code pose, if their left arm goes out of frame, the run code meter decays more slowly than it grows. Thus when the user brings their arm back into the frame, the progress of the run code pose is easily recovered.
  - *Both arms down not used as input*: Users need to take time to think about their input. The most common pose while thinking was leaving both arms at their sides, leading to a {L=LOW, R=LOW} classification.

IV. Methods

We used the insights from the pilot study to design the following full user study.

A. Hypotheses

Hypothesis 1 (H1): *Users will evaluate PoseToCode to be a usable system.*

B. Procedure

A single-session within-subject study was conducted virtually over Zoom with a local 5th grade class in the context of a K-12 STEM outreach event. The study was approved by the University's Institutional Review Board (IRB #UP-20-01171).

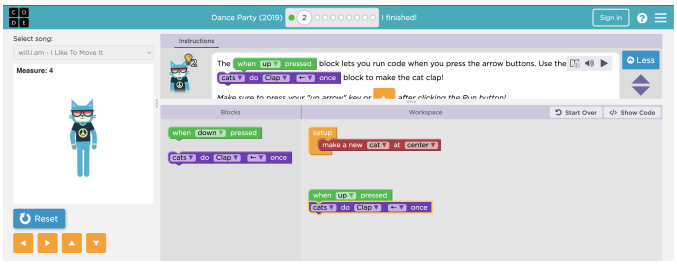


Figure 6: Code.org: Dance Party 2019 block programming activity that aims to teach basic coding concepts by guiding users to code a dance routine for a virtual character [14].

The student participants were randomly assigned to one of two conditions: 1) PoseToCode programming activity first; and 2) Code.org [14] Dance Party 2019 activity (Figure 6) first, a drag-and-drop block programming exercise giving students instructions to create a dance routine for a virtual character. We chose Code.org's Dance Party 2019 Activity because, similarly to PoseToCode, it teaches sequential code logic, and code blocks are used as instructions for a virtual character. Each student participant used a Chromebook to complete the activities.

Student participants first completed a demographic survey. They then engaged in their first coding activity for up to 10 minutes, followed by a post-activity survey. The student participants then moved to their second activity, followed by a second post-activity survey. Finally, the student participants were given a final survey comparing the two activities. The student participants were allowed to end an activity early at any point, in which case they were automatically directed to the next step in the study procedure. A diagram of the study procedure is shown in Figure 7.

This study was conducted virtually, given the COVID-19 pandemic. The virtual format of the study constrained the ability to provide clarifications and assistance to the student participants to only the Zoom chat function, where the students

 ISBN: 978-1-61208-982-9

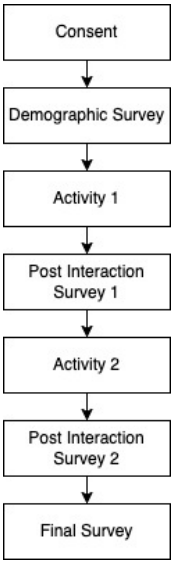


Figure 7: Diagram of the study procedure: pre-study survey, two coding activities, post-activity surveys after each activity, and an activity preference survey.

asked questions via this function. A single teacher in the classroom had to physically move to each individual student participant to answer their questions. Conducting the study in person has the potential to make student participants' experience more enjoyable.

C. Participants

A local Los Angeles 5th grade class of 24 students (7 male, 16 female) was recruited. All student participants were volunteers and were given no form of compensation. Prior to the study, formal consent was obtained from each students' legal guardian and child assent was obtained from each student participant. A total of 10 participants (5 male, 5 female) completed all surveys and both programming activities. This paper reports on the data and analyses from those 10 participants. We recognize this could lead to survivorship bias, but this was chosen to reduce possible ordering effects and to use all pairwise data.

In the pre-survey, we asked student participants what programming education platforms, if any, they had used in the past. All 10 indicated that they had previous technical experience with Scratch and two indicated that they also had past experience with Code.org. Additionally, we asked student participants to indicate their level of agreement with the statement "I want to learn more about computer programming." Of the 10 participants, 5 strongly agreed with the statement, and the other 5 agreed with the statement.

D. Data Collection

The pre-study surveys collected data on the students' prior exposure to programming. The post-activity surveys obtained system usability scores (SUS) [21], aiming to assess the perceived activity difficulty for PoseToCode and Code.org, and to capture the student participants' attitudes towards programming after each activity. The final post-study survey obtained each student's preferred activity between PoseToCode and Code.org, as well as qualitative data via a write-in form on why they preferred one activity over the other.

P2C behavioral data were automatically collected in order to 1) find behavioral data correlations for usability; and 2) create a dataset to compare future iterations of P2C to. As the student participants performed the PoseToCode activity, behavioral data were collected consisting of the time each student took to complete the activity, the number of exercises each student successfully completed within the time limit, and the number of code blocks created throughout the activity.

V. Results

This work evaluates the usability of P2C through participant surveys outlined in Section IV-D. Participant interviews were also analyzed for details about their experience.

A. Quantitative Results

To explore design considerations about PoseToCode, we adapted the System Usability Scale (SUS) [21] questionnaire for young students in order to compare usability compared to the Code.org activity. PoseToCode yielded a median SUS score of 63.75, slightly below the SUS average of 68, and Code.org yielded a median of 75, above average. A SUS score between 68 and 89 indicates that the system has "good" or above average usability, and PoseToCode's SUS score is within the 35-40th percentile [22]. The statistical power of the SUS scores generated are low since there are only 10 responses. Thus, this supports hypothesis 1 but leaves room for improvement. Figure 8 shows the difference in SUS scores between Code.org and P2C. Mann-Whitney tests indicated that PoseToCode (*Mdn* = 4) is more difficult than Code.org (*Mdn* = 5) with conditions ($U = 20$, $p = .020$).

A Pearson's r correlation indicated no significant relationship between total time spent on the PoseToCode activity and SUS score (rs(10) = -0.367). Similarly, Pearson's r correlation

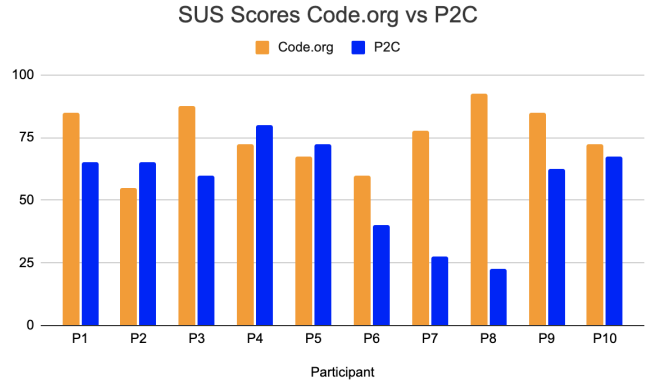


Figure 8: Bar graph comparing SUS scores for Code.org and P2C for all 10 student participants (Ok = 25-59, Good = 60-89, Excellent = 90-100 [22]).

 ISBN: 978-1-61208-982-9

did not yield statistical significance between the number of code blocks created by a student participant during the PoseToCode activity and SUS score (rs(10) = -0.252). Both post-activity surveys asked participants how much they agree with the statement "I want to learn more about computer programming." Mann-Whitney tests revealed an insignificant difference between the responses for PoseToCode (*Mdn* = 4 (Agree)) and Code.org (*Mdn* = 5 (Strongly Agree)) conditions (U = 41, p = .480).

B. Qualitative Results

The qualitative results from the post-study survey showed that 5 of 10 student participants preferred PoseToCode over Code.org. For the participants who started with PoseToCode, 4 of 5 preferred PoseToCode, and for the participants who started with Code.org, 4 out of 5 preferred Code.org. Two themes emerged from analyzing the free responses, as follows.

1) PoseToCode is more active than than Code.org: Participants who preferred PoseToCode found PoseToCode to be a more active and engaging activity than Code.org. Three of five participants who preferred PoseToCode over Code.org used the word *fun* to describe PoseToCode in their reasoning for choosing their preference. Only one student out of five who preferred the Code.org activity over PoseToCode described Code.org as *fun.* P3 indicated that they preferred PoseToCode because they *get to be active and move around,* and similarly, another participant noted that with PoseToCode, *you do more activity and more exercise* than Code.org.

2) Code.org is easier to use: Student participants who preferred Code.org found Code.org easier to use than PoseToCode because it had fewer software bugs when compared to PoseToCode. For example, P1 wrote about Code.org, *It's much more easy to use. (I like coding this way)* while P2 wrote, *in Code.org it's simple and fun and it's a good way to pass the time but PoseToCode is super frustrating and got me annoyed because of glitched like when it highlights your screen green and then kicks you out.* Three participants wrote in the final survey that PoseToCode was *frustrating* because it *glitched* often, making Code.org more desirable. Additionally, two participants wrote that Code.org gave *more instruction* than PoseToCode and provided more guidance on how to be successful in the activity.

VI. Discussion

This work explored design considerations and validated the usability of PoseToCode (P2C), an embodied learning block-based programming language. Half of the 5th grade student participants who completed all of the surveys preferred P2C over Code.org but this may be linked to an ordering effect. Students who preferred P2C referred to it as being more *fun* and *engaging* while those who preferred Code.org found it easier to use and *less buggy.* While some students reported a *buggy* P2C, we still see moderate to high usability scores of many of the students (Figure 8). We believe this indicates the high potential of P2C as an engaging activity with straightforward room for technical improvement.

We observed some student participants helping one another with PoseToCode. This is worth exploring toward developing collaborative programming activities. The trend of student participants who preferred PoseToCode stating that it was more engaging and active than Code.org indicates a positive feature about PoseToCode and the way its embodied learning focus distinguishes it from standard mentalistic programming activities like Code.org. On the other hand, a potential reason why more participants preferred Code.org is that students are generally more familiar with Code.org's teaching style of programming, therefore making it more intuitive and easier to use than PoseToCode.

A. Limitations and Future Work

The most cited issues during the study were the technical glitches that occurred as students interacted with PoseToCode. One participant described being "kicked out" of the activity after performing the run code pose stopped recognizing their poses from their video feed. A set of unit tests and integration tests need to be designed to interact with the system. Another flawed design consideration noted by a student was that the PoseToCode instructions were unclear. This problem should be addressed with a video tutorial before the main activity that shows students how to create code blocks, execute the code, reset the code, and indicate that they have completed a task.

Some students stayed seated during PoseToCode and some performed the activity while standing. Because PoseToCode was designed to be a standing activity, those who stayed seated may not have been able to have the same range of motion and movement experience as those who interacted while standing, as intended. Study instructions should make it clear that the interaction should be done while standing.

The surveys we used were potentially too long and tedious for the student participants; while 24 students participated in the exercises, only 10 completed all the surveys and are included in the analysis. A future study should take the student age into account when selecting the survey tools. Additionally, when data are collected in person, individual interviews with students could yield additional insights not easily gathered via Zoom.

VII. Conclusion

This work explored introducing embodied learning capabilities into block-based programming toward encouraging student curiosity for coding. We explored how to design a usable system. Through a pilot study and a full study with 5th graders, we learned about key design considerations and needed improvements forming a basis for designing web-accessible embodied (kinesthetic) activities like PoseToCode. P2C is open-sourced and shown in a publicly accessible repository [8] and demonstration [9].

Acknowledgments

Authors denoted with * provided equal contribution.

 ISBN: 978-1-61208-982-9

This work was supported by the National Science Foundation (NSF) under award IIS-1925083 and University of Southern California's Viterbi School of Engineering under the Merit Research Award and Viterbi Fellowship Award.

Citation Diversity Statement – Recent work in several fields of science has identified a bias in citation practices such that papers from women and other minority scholars are undercited relative to the number of papers in the field [23]–[27]. We recognize this bias and have worked diligently to ensure that we are referencing appropriate papers with fair gender and racial author inclusion. First, we predicted the gender of the first and last author of each citation using images of authors. By this measure, our references contain 27% woman(first)/woman(last), 15% man/woman, 12% woman/man, and 46% man/man. This method has limitations since images and names used to predict genders may not be entirely accurate and this method does not account for non-binary, transgender, and intersex people.

REFERENCES

[1] J. Schilling and R. Klamma, "The difficult bridge between university and industry: a case study in computer science teaching," *Assessment & Evaluation in Higher Education*, vol. 35, no. 4, pp. 367–380, 2010.

[2] N. Zamin *et al.*, "Learning block programming using scratch among school children in malaysia and australia: An exploratory study," in *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1–6, 2018.

[3] T. W. Price and T. Barnes, "Comparing textual and block interfaces in a novice programming environment," in *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, (New York, NY, USA), pp. 91—-99, Association for Computing Machinery, 2015.

[4] D. Weintrop and U. Wilensky, "Comparing block-based and text-based programming in high school computer science classrooms," *ACM Trans. Comput. Educ.*, vol. 18, oct 2017.

[5] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: Students' perceptions of blocks-based programming," in *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, (New York, NY, USA), pp. 199—208, Association for Computing Machinery, 2015.

[6] E. Eika, "Learning in higher education under the covid-19 pandemic: Were students more engaged or less?," *International Journal of English Linguistics*, vol. 11, no. 3, pp. 96–97, 2021.

[7] M. Macedonia, "Embodied learning: why at school the mind needs the body," *Frontiers in psychology*, pp. 2098–2099, 2019.

[8] N. Chatwani, C. Kuo, G. Thomas, J. Cordero, and R. Agrawal, "PoseToCode." https://github.com/interaction-lab/PoseToCode, June 14, 2022.

[9] "PoseToCode demonstration." https://posetocode.web.app/tutorial.html, June 14, 2022.

[10] T. L. Pittinsky and N. Diamante, "Going beyond fun in stem," *Phi Delta Kappan*, vol. 97, no. 2, pp. 47–51, 2015.

[11] S. N. H. Mohamad, A. Patel, Y. Tew, R. Latih, and Q. Qassim, "Principles and dynamics of block-based programming approach," in *2011 IEEE Symposium on Computers Informatics*, pp. 340–345, 2011.

[12] A. Seyfi and A. Patel, "Briefly introduced and comparatively analysed: Agile sd, component-based se, aspect-oriented sd and mashups," in *2010 International Symposium on Information Technology*, vol. 2, pp. 977–982, 2010.

[13] M. Resnick *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[14] F. Kalelioğlu, "A new way of teaching programming skills to k-12 students: Code. org," *Computers in Human Behavior*, vol. 52, pp. 200–210, 2015.

[15] A. Paniagua and D. Istance, *Teachers as Designers of Learning Environments: The Importance of Innovative Pedagogies. Educational Research and Innovation.* ERIC, 2018.

[16] P. Kosmas, A. Ioannou, and P. Zaphiris, "Implementing embodied learning in the classroom: effects on children's memory and language skills," *Educational Media International*, vol. 56, no. 1, pp. 59–74, 2019.

[17] M. A. Shakroum, K. W. Wong, and C. Fung, "The effectiveness of the gesture-based learning system (gbls) and its impact on learning experience," *Journal of Information Technology Education: Research*, vol. 15, pp. 191–210, 2016.

[18] L. Streeter and J. Gauch, "Teaching introductory programming concepts through a gesture-based interface," in *International Conference on Human-Computer Interaction*, pp. 116–123, Springer, 2018.

[19] C. Lugaresi *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[20] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with blockly," in *2017 IEEE blocks and beyond workshop (B&B)*, pp. 21–24, IEEE, 2017.

[21] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[22] B. Klug, "An overview of the system usability scale in library website and system usability testing," *Weave: Journal of Library User Experience*, vol. 1, no. 6, 2017.

[23] N. Caplar, S. Tacchella, and S. Birrer, "Quantitative evaluation of gender bias in astronomical publications from citation counts," *Nature Astronomy*, vol. 1, no. 6, pp. 1–5, 2017.

[24] M. L. Dion, J. L. Sumner, and S. M. Mitchell, "Gendered citation patterns across political science and social science methodology fields," *Political analysis*, vol. 26, no. 3, pp. 312–327, 2018.

[25] J. D. Dworkin, K. A. Linn, E. G. Teich, P. Zurn, R. T. Shinohara, and D. S. Bassett, "The extent and drivers of gender imbalance in neuroscience reference lists," *Nature neuroscience*, vol. 23, no. 8, pp. 918–926, 2020.

[26] D. Maliniak, R. Powers, and B. F. Walter, "The gender citation gap in international relations," *International Organization*, vol. 67, no. 4, pp. 889–922, 2013.

[27] S. M. Mitchell, S. Lange, and H. Brus, "Gendered citation patterns in international relations journals," *International Studies Perspectives*, vol. 14, no. 4, pp. 485–492, 2013.

 ISBN: 978-1-61208-982-9