IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45SE/2025/4538725/
Vol. 2025 (22), Article ID 4538725

Bridging Computer Programming Education with Quantum Computing and Art: Temporal Rhythms Observed in Space*

Michael ROROS, James BRAMAN, Cody MAYFIELD
And Mel AKHIMIEMONA

Community College of Baltimore County, Baltimore, Maryland, USA

Correspondence should be addressed to: Michael ROROS, ro1016117@ccbcmd.edu

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

This paper introduces *Temporal Rhythms Observed in Space* (Trois), an educational art-based framework designed to teach computer programming concepts through an interactive generative system inspired by quantum computing concepts. Traditional programming courses rarely expose students to emerging topics like quantum computing; instead, they focus on traditional programming concepts. Trois addresses this gap by combining cellular automata with visual patterns inspired by quantum time crystals to create an accessible, creative learning experience. While not a complete simulation of a quantum system, Trois conceptually models behaviors such as temporal symmetry breaking, emergent oscillations, and many-body interactions. The aim is to provide students, especially those without advanced mathematical training, with a hands-on framework to explore programming constructs alongside quantum-inspired thinking. The framework is implemented in Python, using a modified version of Conway's Game of Life, extended with real-time visual feedback and interactive parameters. Students can experiment with rule variations, observe evolving states, and manipulate inputs to better understand core programming ideas such as loops, conditionals, and state transitions. Future work includes broader deployment in programming curricula and additional features to further align with quantum computing concepts. This paper discusses Trois' major components, current implementation details, and the next steps for using it for a programming course.

Keywords: Programming Education, Generative Art, Quantum-Inspired Computing

Introduction

Learning computer programming can be difficult for students due to its complexity and high level of abstraction. Although failure rates are a concern, some institutions require a "C" or higher grade to progress to higher-level programming courses, which adds to grade challenges. Bennedsen and Caspersen (2019) surveyed 161 institutions worldwide and found that introductory programming (CS1) failure rates were, on average, 28 percent. Other studies have also documented high failure rates in introductory programming courses, with nearly one in three students failing (Álvarez et al., 2024). Learning programming languages can be difficult, as these are designed to be used in

Cite this Article as: Michael ROROS, James BRAMAN, Cody MAYFIELD And Mel AKHIMIEMONA, Vol. 2025 (22) "Bridging Computer Programming Education with Quantum Computing and Art: Temporal Rhythms Observed in Space " Communications of International Proceedings, Vol. 2025 (22), Article ID 4538725, https://doi.org/10.5171/2025.4538725

professional settings to build applications and are not designed for educational purposes (Gomes & Mendes). Specific topics that cause difficulty for students include the use of recursion (Lahtinen et al., 2005), object-oriented design (Thomasson et al., 2006), variable scope (Becker et al., 2019) nested control structures (Hao et al., 2023), and program debugging (Whalley et al., 2023), to list a few challenges. Changes in student preferences in consuming instructional materials to learn programming have also evolved over the years (Vincenti et al., 2015). Educators must constantly adjust and adapt course materials for the challenges of changes from students, programming, and the use of instructional materials while at the same time maintaining a level of interest and engagement. with course materials.

There have been innovative approaches to use creative constructs to reimagine teaching computer science concepts in a captivating way to encourage students to program or consider technology-related fields. Using images, music, and other multimedia elements are ways to provide immediate feedback to students, making learning fun, yet challenging, while at the same time exploring complex topics such as loops, arrays, programming libraries, and much more. Work by Terroso and Pinto (2012) report on creative coding for teaching non-programmers using Processing with positive feedback. This can also be seen in other research where creative coding is used in CS1 courses to create a sense of creativity and excitement about programming, encouraging student motivation (Greenberg et al., 2012). Other work uses sound and music to help teach concepts such as loops, lists, and functions with high student motivation (Freeman et al., 2019). Brunvand and Stout (2011) experimented with teaming computer science and art students in a cross-disciplinary course to create kinetic art, connecting strengths from both fields. The intersection of technology and art can inspire many interesting projects and collaborations (Braman et al., 2009).

More recently, there has been growing research in efforts to teach quantum computing in several technology courses. As quantum computing (QC) continues to emerge as an important technology that needs attention, it represents a shift in how we think about classical computer science. QC, however, is even more complex than learning basic computer programming alone, requiring an understanding of physics and strong mathematical skills. Work by Adams et al. (2015) aims to create a CS1 course focused on introducing writing code for qubits before seeing classical syntax and data types, with the idea of providing an early on-ramp to these complex concepts. Other projects have introduced these concepts as hands-on exercises where students build small quantum circuits and algorithms (Galetto, 2024). Traditional computer programming education for CS0/CS1/CS2 rarely connects coding concepts to those of the arts and the field of quantum computing simultaneously.

The TROIS Project

The Trois project, or **T**emporal **R**hythms **O**bserved **i**n **S**pace, is both an educational and artistic project that aims to encourage students to learn programming concepts by constructing an interactive art piece that also uses quantum computing concepts. Although not a proper quantum system, Trois conceptually models key behaviors typically found in quantum time crystals, such as temporal translation symmetry breaking, emergent oscillations, and many body interactions. The goal is to allow students to experience a unique hands-on project where they can apply their newly learned programming skills to an interactive project that is quite complex but presented in an intriguing way. The art component itself pulls inspiration from Conway's Game of Life. However, instead of evolving shapes within a 2D array, it is based on a 3D cube where each slice or separate copy of the cube can represent different moments in time. This allows for temporal oscillations and emergent patterns that conceptually mirror the properties of quantum time crystals, such as time-based symmetry breaking and periodicity without energy input. The cube becomes a living artwork: each voxel (3D pixel) changes color based on past and future states, generating a mesmerizing visual experience. The simulation is currently being built using Python with libraries like NumPy for matrix manipulation and Matplotlib for visualization. Layers of rules can be adjusted to create dynamic "quantum-like" oscillations, giving students a playful yet powerful entry point into computation, mathematics, and systems thinking. Our current implementation also uses the Qiskit Python library, an open-source quantum computing framework.

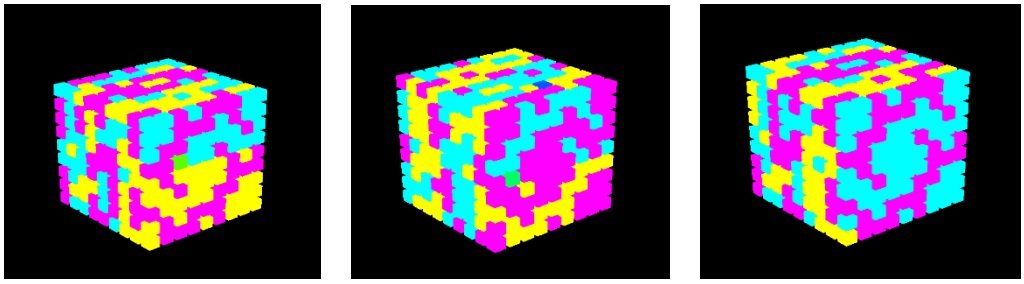

Fig 1: Cube Example

The artistic component utilizes the general rules of Conway's Game of Life, where we use cellular automata, as it has emergent behaviors that can create interesting patterns, requires no additional input from the user, and has a simple implementation. The Game of Life mimics some entanglement properties, and as the changes in the cube can be seen as deterministic time slices, each slice can be evaluated. In future project iterations, we will use an entanglement simulation of the cube with multiple other cubes, applying the same rules where one cube's states interact with the central cube's states.

Course Module

Trois' educational impact is aimed at bridging programming and the arts. It provides a hands-on platform for students to explore foundational programming concepts, logic, and abstract QC principles using the cube in an expressive, visual medium. By embedding an artistic output into the programming logic, students can see their code come to life as interactive, evolving digital sculpture. This makes complex ideas and behaviors, time symmetry, and multidimensional data more tangible and less intimidating. Trois invites students to become programmers and artists, encouraging curiosity, experimentation, and appreciation for the cross-disciplinary nature of science and art.

As part of a pilot study, TROIS will be used in a 200-level Python programming course similar in topic coverage to a typical CS1 course. Specifically, the project will be divided into three-course modules. One module will be embedded in basic quantum computing principles with basic information, application, and theory. Students will be provided with simple Python code and related exercises that they will implement and practice. Next, students will be provided with code snippets on creating simple 3D models (basic shapes) and working with color changing for these shapes. They can also download code to experiment with their own 3D cube in Python. In the third related module, students will be introduced to creating some simple color change experiments with their cubes related to quantum foundations as a prelude to a small project. Aspects of the course module materials would be similar to a quantum computing curriculum using a software-driven approach (Mykhailova & Svore, 2020).

Conclusions and Future Work

Our next significant step in this work for Trois is solidifying the framework and specific content for the course modules. After the course modules are developed and deployed, we will collect student data based on a survey to gain insights into student interest, difficulty, perception, and more. We will also compare grade changes and student engagement with the new course design compared to historical performance. Improvements and changes to the project design and course materials will be made based on student feedback and grade metrics.

Trois' current implementation details and the next steps for using it for a programming course have been discussed. As an educational and artistic project aimed at teaching computer programming concepts by constructing an interactive art program using cellular automaton and aspects of quantum computing, we intend to make engaging course materials. The framework aims to create an innovative and engaging method to teach specific programming course topics and an accessible platform for students to explore concepts of quantum computing principles. We are excited to see the possibility of teaching these complex topics while inspiring students to work on challenging topics in a new and exciting way.

References

- Adams, A. J., Borela, R., Young, J. S., & Conte, T. M. (2025). A blueprint for Q-CS1, an introductory quantum programming course. In Proceedings of the 56th ACM Technical Symposium on Computer Science Education, Volume 2 (SIGCSE TS 2025) (pp. 1351–1352). Association for Computing Machinery.
- Álvarez, C., Samary, M. M., & Wise, A. F. (2024). Modularization for mastery learning in CS1: A four-year action-research study. Journal of Computing in Higher Education, 36, 546-589.
- Becker, W., James, K., & Minier, M. (2019). Four scope-related misconceptions held by computer science students. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 1278-1284). ACM.
- Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming -12 years later. ACM Inroads, 10(2), 30-36.
- Braman, J. Vincenti, G., & Trajkovski, G. (2009). The Handbook of Research on Computational Arts and Creative Informatics. Information Science Reference. Hershey, PA.
- Brunvand, E., & Stout, P. (2011). Kinetic art and embedded systems: A natural collaboration. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE' 11) (pp. 323-328). https://doi.org/10.1145/1953163.1953263
- Freeman, J., Magerko, B., Edwards, D., McKlin, T., Lee, T., & Moore, R. (2019). EarSketch: Engaging broad populations in computing through music. Communications of the ACM, 62(9), 38-45.
- Galetto, F., López, H. H., Rahmati, M., Buonocore, A., Liu, C., Wang, Y., & Dou, W. (2024). Experience in teaching quantum computing with hands-on programming labs. The Journal of Supercomputing, 80, 14029–14056.
- Gomes, A., & Mendes, A. J. (2007) Learning to program – difficulties and solutions. Proceedings of the 2007 international convergence on Engineering Education, September 3-7, Coimbra, Portugal.
- Greenberg, I., Kumar, D., & Xu, D. (2012). Creative coding and visual portfolios for CS1. In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (pp. 247-252).
- Hao, X., Xu, Z., Guo, M., Hu, Y., & Geng, F. (2023). The effect of embedded structures on cognitive load for novice learners during block-based code comprehension. International Journal of STEM Education, 10, Article 42.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. SIGCSE Bulletin, 37(3), 14-18.
- Mykhailova, M., & Svore, K. M. (2020). Teaching quantum computing through a practical software-driven approach: Experience report. In Proceedings of the 51st ACM Technical Symposium on computer science education (pp. 1019-1025).
- Terroso, T., & Pinto, M. (2022). Programming for non-programmers: An approach using creative coding in higher education. In Proceedings of ICPEC 2022 (OASIcs Vol. 102, Art. 13, pp. 13:1–13:8).
- Thomasson, B., Ratcliffe, M., & Thomas, L. (2006). Identifying novice difficulties in object-oriented design. SIGCSE Bulletin, 38(3), 123-127.
- Vincenti, G., Hilberg, S. & Braman, J. (2015). Student preferences for consuming supplemental instructional material in CS0/CS1/CS2 courses. In D. Slykhuis & G. Marks (Eds.), Proceedings of Society for Information Technology & Teacher Education International Conference 2015 (pp. 558-566).
- Whalley, J., Settle, A., & Luxton-Reilly, A. (2023). A think-aloud study of novice debugging. ACM Transactions on Computing Education, 23(2), Article 28.