FlowPilot: A Generative AI-Driven Visual Language for Computational Thinking Education

Tommaso Turchi
Department of Computer Science
University of Pisa
Pisa, Italy
tommaso.turchi@unipi.it

***Abstract*—This paper introduces FlowPilot, a novel flow-based visual programming language designed to enhance Computational Thinking (CT) education. FlowPilot leverages generative AI to create a dynamic, browser-based environment where users can construct programs using natural language descriptions. By integrating AI-driven block generation with a flow-based visual interface, FlowPilot supports key CT pillars such as abstraction and decomposition. This approach offers a unique platform for learners to explore programming concepts at various levels of complexity, fostering a deeper understanding of computational processes.**

***Index Terms*—Visual Programming, Computational Thinking, Generative AI**

I. Content and Claims

In our increasingly software-driven world, computational literacy and coding have become essential skills. Initiatives like the Hour of Code[1] have introduced these concepts to diverse groups, focusing on fostering Computational Thinking (CT) skills – problem-solving abilities central to Computer Science, such as abstraction and pattern recognition. Wing [1] defined CT as skills needed to solve complex problems using computers, applicable across various domains.

Programming has proven effective in developing CT skills [2], and many techniques have been employed to lower its barriers and foster the spread of computational literacy, often with mixed results [3], [4]. Visual Programming (VP) emerged as a paradigm to help reduce the syntactic burden by representing instructions graphically. For instance, puzzle-like blocks represent components, with shapes indicating input/output compatibility [5]. However, Visual Programming Languages (VPLs) often face a limitation: they're typically fixed at a specific abstraction level, providing predefined blocks that may not align with a user's mental model. For example, in programming a calculator, a VPL might offer individual blocks for each arithmetic operation, while a user might prefer a single block for all calculations. This mismatch between the VPL's tools and the user's preferred abstraction level can potentially hinder the development of CT skills.

A. FlowPilot: Enhancing CT through Generative AI

To address these limitations, we introduce FlowPilot[2], an innovative flow-based visual programming language primarily designed for novice programmers and CT education. FlowPilot leverages generative AI to enhance Computational Thinking (CT) education and is implemented in JavaScript, running in the browser using the ReactFlow library[3] for diagramming to provide an intuitive interface for users to construct programs visually.

The integration with OpenAI's GPT API[4] allows users to specify the behavior, inputs, and outputs of new blocks (or nodes) using natural language (see Figure 1). The system then uses OpenAI models to generate corresponding JavaScript code, creating a function for each block. FlowPilot supports a range of data types for inputs and outputs, including Numbers, Strings, Booleans, Arrays, and Objects, providing flexibility for various programming tasks.

Once the code is generated, FlowPilot automatically names the block by summarizing the function code. Users can then connect these blocks to construct their program. When connections are made, a play button appears on output blocks, allowing users to execute that portion of the program and see the results immediately.

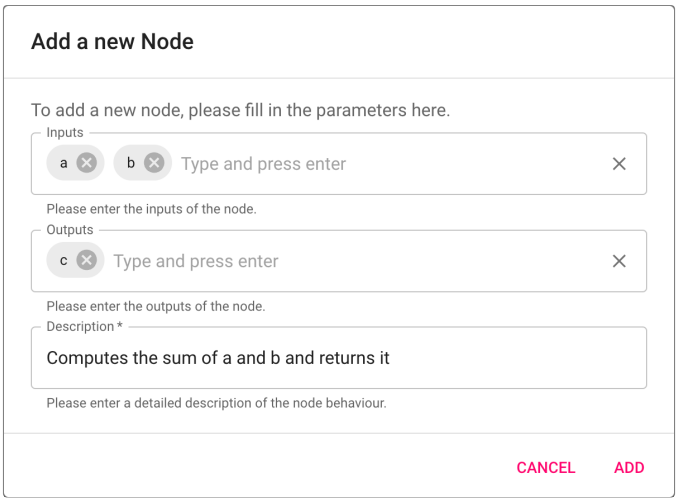

Fig. 1. FlowPilot's block (called node) creation dialog, demonstrating the natural language interface for specifying custom block behavior.

[1]https://hourofcode.com
[2]https://flowpilot.trx.li
[3]https://reactflow.dev/
[4]https://platform.openai.com/docs/overview

This approach enables rapid prototyping and experimentation, allowing users to focus on problem-solving and CT rather than syntax. By generating code based on natural language descriptions, FlowPilot effectively lowers the barrier to programming while still exposing users to real JavaScript code. Furthermore, FlowPilot can support multiple levels of abstraction within the same environment. Using our calculator example, FlowPilot can accommodate both the lower-level approach with individual arithmetic operation blocks and the higher-level approach with a single calculation block (see Figure 2). This flexibility allows users to work at the abstraction level that best suits their current understanding and the problem at hand.

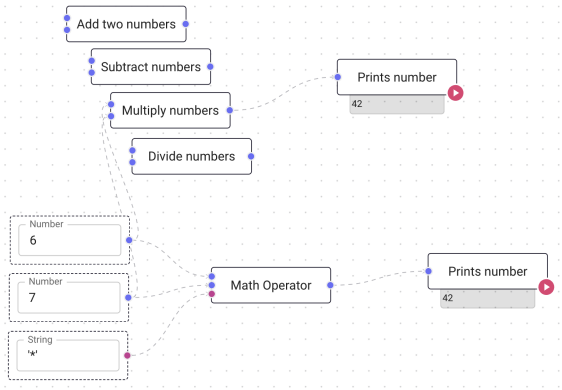

Fig. 2. FlowPilot's flexibility in abstraction levels: Lower-level implementation using individual arithmetic operation blocks (top); Higher-level implementation using a single calculation block (bottom). This demonstrates how users can work at different abstraction levels within the same environment.

FlowPilot is designed to support two key pillars of Computational Thinking:

a) Abstraction: Users can decide the level of abstraction for each block, from individual functions to complex, multi-function components. This flexibility allows learners to grasp the concept of abstraction layers in programming and to move between different levels of abstraction as their understanding grows [6]. We recognize the need to carefully manage potential additional cognitive load and plan to develop adaptive scaffolding techniques in future iterations to guide users in selecting appropriate abstraction levels (e.g., prompt templates).

b) Decomposition: The flow-based interface naturally encourages problem decomposition. Users can break down complex problems into smaller, manageable subproblems represented by individual blocks. The ability to create custom blocks at various abstraction levels further supports this.

Moreover, the process of narrowing and refining block generation prompts may itself foster CT skills. As users iterate on their descriptions to achieve desired functionality, they engage in a form of abstraction and problem decomposition, albeit in a novel context that merges natural language understanding with programming concepts. This process aligns with the concept of "proompting" introduced in [7], potentially representing a new form of CT that merges human intelligence with AI capabilities.

By supporting multiple levels of abstraction within the same environment, FlowPilot provides a more flexible and adaptable learning experience compared to traditional VPLs. This approach allows learners to start at a comfortable level of abstraction and gradually move to more complex representations as their understanding grows.

While initial user feedback has been promising, we acknowledge potential challenges in this approach. The reliance on AI-generated code may lead to a lack of understanding of underlying programming concepts if not properly balanced with traditional learning methods. Additionally, the quality and consistency of AI-generated blocks require careful consideration to ensure they align with educational objectives. To address these concerns, we plan to conduct a comprehensive evaluation study to assess FlowPilot's impact on CT skill development, user experience with flexible abstraction levels, and the effectiveness of AI-generated blocks in supporting learning outcomes. This evaluation will inform future refinements to ensure educational objectives are met.

II. Relevance

FlowPilot is highly relevant to the VL/HCC community for several reasons:

a) Innovative Visual Language Design: FlowPilot extends the traditional Visual Programming paradigm by incorporating AI-generated components, presenting a novel approach to visual language design.

b) Natural Language Integration: The use of natural language for block specification bridges the gap between conceptual thinking and program implementation, a key area of interest in human-centric computing.

c) Generative AI in Education: FlowPilot directly addresses the VL/HCC 2024 special emphasis on Generative AI's impact, offering a concrete example of how these techniques can enhance visual programming environments while raising important questions about balancing AI assistance with human understanding in educational contexts.

d) Ethical Considerations: FlowPilot raises important questions about the role of AI in programming education. It prompts discussions on the balance between AI assistance and human understanding, the potential for AI to democratize programming education, and the implications of relying on AI-generated code in learning environments.

e) Human-AI Collaboration: FlowPilot showcases a new paradigm of human-AI collaboration in programming education, aligning with the conference's interest in critiquing Generative AI approaches from a human perspective.

III. Presentation

We will showcase FlowPilot through an interactive live demonstration during the conference. Attendees will have the opportunity to interact directly with FlowPilot, creating blocks using natural language and constructing simple programs. This hands-on experience will allow participants to explore FlowPilot's innovative features and understand its potential impact on Computational Thinking education.

ACKNOWLEDGMENT

This work was produced with the co-funding of the European Union – Next Generation EU, in the context of The National Recovery and Resilience Plan, Investment 1.5 Ecosystems of Innovation, Project Tuscany Health Ecosystem (THE), ECS00000017. Spoke 3.

REFERENCES

[1] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006.

[2] G. Orr, "Computational thinking through programming and algorithmic art," in *SIGGRAPH 2009: Talks*. New Orleans Louisiana: ACM, Aug. 2009, pp. 1–1.

[3] D. Weintrop, "Blocks, text, and the space between: The role of representations in novice programming environments," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. Atlanta, GA: IEEE, Oct. 2015, pp. 301–302.

[4] A. Malizia, D. Fogli, F. Danesi, T. Turchi, and D. Bell, "Tapasplay: A game-based learning approach to foster computation thinking skills," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2017, pp. 345–346.

[5] "B&B 2019 Foreword," in *2019 IEEE Blocks and Beyond Workshop (B&B)*, 2019, pp. vi–vii.

[6] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*, vol. 1, 2012, p. 25.

[7] A. Repenning and S. Grabowski, "Proompting is computational thinking," *Proceedings http://ceur-ws. org ISSN*, vol. 1613, p. 0073, 2023.