HNUE JOURNAL OF SCIENCE
Educational Sciences 2025, Volume 70, Issue 5, pp. 157-167
This paper is available online at http://hnuejs.edu.vn/es
DOI: 10.18173/2354-1075.2025-0100

DEVELOPING A SIMULATION TOOL FOR MAZE-SOLVING ROBOT PROGRAMMING USING PYTHON

Nguyen Trung Khanh and Le Xuan Hien
Faculty of Information Technology, Hanoi National University of Education, Hanoi city, Vietnam
*Corresponding author: Nguyen Trung Khanh, e-mail: khanhnt1312@hnue.edu.vn

Received June 10, 2025. Revised September 17, 2025. Accepted September 30, 2025.

Abstract. Simulation tools play a vital role in robotics programming education by enabling students to visualize and test algorithms in a virtual environment. In the Robotics Programming course within the Bachelor's program in Informatics Teacher Education at the Faculty of Information Technology, Hanoi National University of Education, a C++ based simulation tool has been used, but it showed significant usability and pedagogical limitations. This study aims to develop a Python-based simulation tool to enhance the teaching and learning experience in this course. Student feedback from Cohort 71 (120 students) was collected through surveys based on established frameworks, including the Technology Acceptance Model (TAM), user experience (UX), and computational thinking (CT), which identified key challenges in the previous tool, such as a steep learning curve and limited visualization capabilities. Based on these insights, the new tool was designed with an intuitive graphical interface and improved algorithm visualization to support hands-on learning. Its effectiveness was empirically evaluated with 65 students from Cohort 72, who experienced both the old and new tools, allowing for a direct comparison. Results indicate significant improvements in usability, engagement, and problem-solving efficiency with the Python-based simulator. This comparative quantitative evidence demonstrates not only the pedagogical benefits of Python-based simulations in robotics education but also the contribution of this study in providing reliable empirical data to support the adoption of this tool.

Keywords: robotics simulation, maze-solving algorithms, educational technology.

1. Introduction

In Vietnam, the 2018 national high school curriculum for Informatics [1] has introduced educational robotics and programming into the Grade 10 syllabus. In this specialized module, students learn to assemble educational robots and program them to perform tasks. This content is a critical part of the curriculum, intended to develop students' computational thinking and programming skills, as well as related concepts in engineering and physics, thereby supporting effective STEM education at the high school level. These curriculum reforms create an urgent need for teacher education programs to adapt and prepare future Informatics educators with the appropriate knowledge and skills. Accordingly, Informatics teacher training programs at universities across Vietnam have been updated to include programming and control of educational robots, to keep pace with the curriculum changes, and to improve the quality of teacher preparation aligned with actual classroom practice. Several universities now offer relevant courses under various titles, such as “Visual Programming and Robotics” (Ho Chi Minh City University of Education) and “Robot Control Programming” (Thai Nguyen University of Education), among others.

At Hanoi National University of Education, content on robot programming has been incorporated into a two-credit “Robot Programming” course for Informatics pedagogy majors (including an English-language track) since Cohort 69 (2019) [2], aiming to equip students with relevant knowledge and skills. The intended learning outcomes of this course include (1) understanding the roles and practical applications of robots, as well as robot assembly and programming skills; and (2) preparing future teachers with the necessary knowledge to teach the three educational robotics topics of the Grade 10 curriculum (2018), enabling them to fully grasp the content and performance standards needed to design effective lectures and laboratory sessions on educational robotics. Within these objectives, robot programming and control constitute the core content of the course, contributing to the development of programming and algorithmic thinking, and an understanding of the nature and mechanisms of robots.

The pedagogical tools for this content are diverse, including hands-on work with physical robots, virtual simulation environments, and instruction using pseudocode and flowcharts. In practice, one teaching approach being implemented is to have students program a robot to solve a maze in a simulation environment. The maze-solving robot problem has been studied since the early days of robotics. For example, Claude Shannon developed a maze-solving robot in 1951 [3] and remains a classic challenge used to test both robot control programming and hardware assembly skills. In this task, a robot is placed in a maze and must autonomously navigate to the maze’s exit without human intervention, and mazes can vary in size, structure, and exit configuration to provide diverse challenges for the student programmers. However, existing robotics simulation tools have limitations. Block-based visual programming environments (for example, those available at vr.vex.com) often do not clearly demonstrate the robot’s underlying operations, because many commands are pre-constructed; thus, they do not help students understand the fundamental workings of the robot. Furthermore, when simulating more complex algorithms (such as shortest-path or backtracking algorithms), block languages have inherent limitations in expressing complex data types. On the other hand, text-based simulation platforms using languages like C++ can be overly complex for introductory instruction: their intricate syntax can cause students to focus on debugging syntax errors rather than on developing control logic, and such tools often have less engaging interfaces and complicated installation requirements.

Among programming languages for simulation, Python offers significant advantages: its syntax is simple and concise, and it can naturally express complex data structures and algorithms. Using Python also reinforces the programming skills of future Informatics teachers, in line with the trends of the 2018 curriculum. Additionally, Python is easy to install on multiple platforms and provides graphical libraries that allow clear and attractive visual representations. Based on these considerations, this study proposes the development of a Python-based simulation environment for the maze-solving robot problem, enhancing the learning experience of students in the course of “Robot programming” at Hanoi National University of Education.

2. Content

2.1. Theoretical and practical foundation

2.1.1. The development of educational robotics

The rapid advancement of science and technology has brought significant transformations to the educational landscape, particularly in teaching content and instructional methods. Among the most prominent technological breakthroughs in education is the integration of programming and robotics into the K-12 curriculum. Educational robotics has been shown to support the development of a range of essential skills, including analysis, design, programming, and interaction with robots, as well as soft skills such as communication, collaboration, critical thinking, creativity, and problem-solving. Moreover, it provides a highly visual and interactive environment for students to grasp abstract concepts more effectively [4], [5].

Educational robotics has been explored through various approaches over several decades. Lau et al. (1999) employed LEGO Mindstorms as a creative learning tool to teach engineering and programming concepts [6]. Kory and Breazeal (2014) investigated the use of robots as storytelling companions to foster language development in preschool children [7]. Other studies have examined robots as teaching assistants to support classroom instruction (Han & Kim, 2009) [8], or as tools for developing computational thinking through robot programming. Amy Eguchi (2014) highlighted how educational robots integrated into STEAM learning environments can foster 21st-century skills while maintaining high student engagement [9]. Similar studies confirmed that programming with robots helps students develop computational and problem-solving skills [10]-[12].

In Vietnam, educational robotics has gained momentum, especially since the rollout of the 2018 General Education Curriculum. Various teaching and learning initiatives incorporating robotics have demonstrated positive outcomes. For instance, Duong Bich Thao et al. (2022) designed a STEM-oriented learning module using VEX IQ robots to enhance high school students' problem-solving abilities [13]. Similarly, Duong Quoc Cong [14] and Dang Dong Phuong [15] developed robotics-based instructional activities for secondary school students that align with engineering design processes.

2.1.2. Maze-solving Robot simulation

Given the resource-intensive nature of robotics education, especially in terms of equipment and space, researchers have also focused on the development of simulation tools. In Vietnamese higher education, efforts such as simulating palletizing robots using the Robotics Toolbox in MATLAB (Nguyen Khac Khiem et al., 2022) [16] or virtual instruction in robot engineering (Nguyen Thi Thanh, 2017) have provided alternative methods for teaching robotics without requiring full physical setups [17].

A classic problem in robotics education is the maze-solving robot, with origins dating back to 1951 and later popularized by the IEEE Micromouse competition in 1977. Since then, the problem has evolved in both hardware and algorithmic complexity, making it an exemplary task for teaching robotics programming, especially for higher education. However, implementing this problem in a real-world setting often requires complex infrastructure, including robots, physical maze boards, and ample space, which limits its accessibility. To address this, several simulation environments have been developed. For instance, Nelson et al. (2004) applied MATLAB to model AI-based maze exploration behaviors [18], while Annaz (2012) created a virtual maze-solving environment using Visual Basic [19].

The maze-solving problem encompasses a wide variety of maze configurations, each requiring different algorithmic approaches for optimal performance. Simple mazes with no loops or branches may be effectively solved using basic wall-following strategies, while more complex mazes with multiple decision points, loops, or varying goal positions demand advanced algorithms such as breadth-first search (BFS), depth-first search (DFS), Dijkstra's algorithm, or A* search. In educational contexts, exposing students to this diversity is essential not only for developing their algorithmic thinking but also for helping them understand the trade-offs between time complexity, memory usage, and optimality of solutions. While many simulation tools exist, few are optimized for undergraduate Informatics teacher training. Some are overly complex and geared toward advanced algorithm research, while others are too simplistic to support the application of sophisticated algorithms.

In the Programming Robotics course at Hanoi National University of Education, students have traditionally used a C++-based maze-solving simulator as part of the laboratory activities. The simulator was developed in C++ and adapted from the materials originally provided in Phil C. Müller's

CS118 Courseware. The tool allows students to program a virtual robot to navigate mazes using classical algorithms such as wall-following and depth-first search. Its interface is relatively simple, displaying the maze as a static grid with stepwise robot movements. Although it provides a useful baseline for introducing algorithmic thinking, the simulator requires manual compilation and offers limited visualization options, which can hinder accessibility and user engagement.

This highlights the need for an appropriately designed simulation tool that enables students to engage with the maze-solving robot problem in an intuitive yet technically rich environment, aligned with their educational level and learning objectives.

2.1.3. Empirical Survey

**** Overview and results of the empirical survey***

To establish a baseline for tool improvement, an empirical survey was conducted with 120 students in Cohort 71 who had completed the Robot Programming course (Semester 2, 2023-2024) using the C++-based maze-solving simulator. The survey instrument was organized into five conceptually distinct dimensions, reflecting both theoretical and practical considerations. These dimensions were derived from established frameworks – specifically, the Technology Acceptance Model (TAM) [20], user experience (UX) theory, and computational thinking (CT) literature [21] to capture students' perceptions of the simulation tool. The TAM informed measures of perceived ease of use and usefulness, which underlie students' acceptance of educational software. The CT framework, which emphasizes problem-solving skills such as algorithmic thinking and debugging, guided the Algorithmic Thinking Support dimension. Together with UX constructs (e.g., usability, engagement, satisfaction) and learning-theory perspectives on instructional effectiveness, these foundations shaped the survey's content. Each of the five dimensions was operationalized through multiple Likert-scale items (1 = strongly disagree, 5 = strongly agree) targeting specific sub-criteria of student interaction and learning. Descriptive statistics for each item are presented in Table 1.

Table 1. Descriptive statistics for the empirical survey's results

Criteria	Means	Standard Deviation
1. Ease of use		
1.1. The interface of the simulation tool is clear, easy to view, and user-friendly.	2.058	0.82
1.2. The robot control commands (e.g., 'turn left', 'turn right', 'go') are easy to understand and use.	2.025	0.851
1.3. I can quickly learn how to use the tool without difficulties.	1.908	0.785
1.4. The installation and start-up process is simple and time-efficient.	1.908	0.719
2. Academic Effectiveness (Learning Outcomes & Instructional Impact)		
2.1. The tool helps me grasp key concepts and navigation algorithms for robots.	3.133	1.183
2.2. The simulation enhances my ability to apply programming knowledge.	3.608	0.906
2.3. Using the tool has improved my ability to solve programming problems.	1.958	0.789
2.4. After using the tool, I observed improvement in my academic outcomes (e.g., grades, confidence).	1.833	0.723
3. User Experience (UX – Engagement, Satisfaction, Motivation)		

3.1. The simulation environment is lively and appealing.	1.958	0.757
3.2. The tool responds to commands quickly, without lag or frequent errors.	3.667	0.969
3.3. The tool's interface is user-friendly and makes it easy to locate necessary functions.	1.992	0.736
3.4. I feel satisfied and confident when programming with this tool.	1.7	0.702
4. Algorithmic Thinking Support (Computational Thinking Framework)		
4.1. The tool encourages me to analyze problems and plan algorithms before programming.	3.517	1.176
4.2. Programming robots in mazes enhances my logical thinking and problem-solving skills.	3.483	1.225
4.3. The tool helps me apply programming concepts (e.g., loops, conditionals) in practice.	3.633	1.008
4.4. Experimenting with different maze types deepens my understanding of how algorithms work.	2.342	0.769
5. Technology Acceptance (TAM – Perceived Usefulness & Future Use Intention)		
5.1. I believe this tool is very useful for learning programming and robotics.	2.992	0.769
5.2. I intend to use this tool regularly in related classes.	2.175	0.853
5.3. Compared to traditional learning methods, the tool saves time and effort.	3.183	0.836
5.4. I am willing to recommend this tool to peers or fellow students.	2.567	0.834

**** Reliability assessment of the survey instrument***

To assess the internal consistency and reliability of the survey instrument, Cronbach's Alpha was calculated for each of the five evaluation criteria as shown in Table 2.

Table 2. Cronbach's alpha value for each evaluation criterion of the empirical survey

Criteria	**Cronbach's alpha**
1. Ease of use	0.78
2. Academic Effectiveness (Learning Outcomes & Instructional Impact)	0.893
3. User Experience (UX – Engagement, Satisfaction, Motivation)	0.739
4. Algorithmic Thinking Support (Computational Thinking Framework)	0.710
5. Technology Acceptance (TAM – Perceived Usefulness & Future Use Intention)	0.847

All subscales yielded Cronbach's Alpha values above the acceptable threshold of 0.70, indicating satisfactory internal reliability for research. These results demonstrate that the survey instrument is reliable and appropriate for subsequent analyses.

**** Descriptive analysis of survey results***

The descriptive statistics indicate that students generally rated the C++-based simulator unfavorably in terms of usability. In the ease of use dimension, mean scores across items were low (M ≈ 1.9–2.1), showing disagreement with positive statements such as "the interface is user-friendly" and "the installation is simple". These results suggest that the old tool was perceived as difficult to operate. With respect to Academic Effectiveness, responses were inconsistent. While students agreed that the simulator helped them apply programming knowledge (item 2.2, M = 3.61), they disagreed that it improved their ability to solve problems or academic outcomes (items 2.3–2.4, M < 2.0). This pattern reveals that the tool had a limited impact on deeper learning. For User Experience, the results were also mixed. The responsiveness of the tool was appreciated (item 3.2, M = 3.67), but ratings for appeal, ease of locating functions, and satisfaction/confidence were low (items 3.1, 3.3, 3.4, M ≈ 1.7–2.0). Thus, although technically stable, the simulator did not engage students effectively. In the Algorithmic Thinking dimension, three items (4.1–4.3, M ≈ 3.5) indicate moderate support for analyzing problems, applying loops and conditionals, and enhancing logical thinking. However, experimenting with diverse maze types (item 4.4, M = 2.34) was rated less favorably. Finally, Technology Acceptance showed neutral to negative evaluations. While some students considered the tool useful (item 5.3, M = 3.18), intention to reuse or recommend it was low (items 5.2 and 5.4, M = 2.18 and 2.57).

While the tool demonstrates some effectiveness in enhancing algorithmic understanding, its lack of usability, inconsistent user engagement, and moderate technology acceptance point to important shortcomings. These findings justify the need to develop a new simulation environment as follows:

- Prioritizes intuitive interaction and clearer feedback mechanisms;
- Enhances visual design and student engagement;
- Offers customizable maze challenges aligned with different learning levels;
- Integrates stronger scaffolding and support for systematic computational thinking.

In addition to the data presented above, the survey also collected students' preferences regarding programming languages. Approximately 95% of respondents preferred Python over other text-based programming languages such as C++, JavaScript, and Java.

These findings highlight the need to redesign the tool with both usability and educational value in mind. A Python-based simulator with a graphical interface, simplified installation, and enhanced algorithm visualization is necessary to meet the learning needs of pre-service Informatics teachers and better align with modern teaching practices.

2.2. System development

**** Tool objectives***

Based on theoretical and practical insights, the proposed simulation tool should aim to fulfill the following core objectives:

- Enhance usability and accessibility: Provide a lightweight, easy-to-install environment compatible across platforms, minimizing setup complexity.
- Offer an intuitive graphical interface: Replace or supplement the command-line interface with real-time visual simulation of the robot navigating the maze.
- Support multiple maze-solving algorithms: Allow users to implement, test, and compare various strategies (e.g., DFS, BFS, A*, wall-following) with ease.
- Improve algorithm transparency: Visualize decision-making steps, sensor input, and control flow to help students understand the logic behind algorithm execution.
- Encourage experimentation and customization: Enable learners to modify mazes, tune parameters, or extend functionality, fostering deeper engagement and active learning.

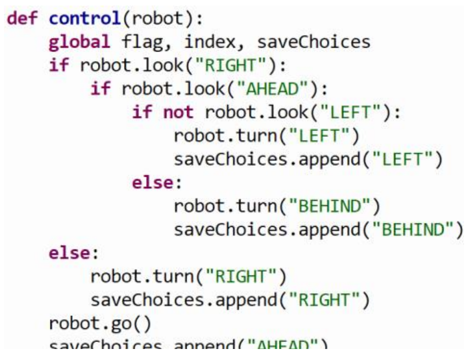
**** Technical architecture***

The tool was developed using Python 3.11, Pygame for GUI rendering, and A modular architecture separating maze generation, robot movement logic, and algorithm execution.

Students write code in a structured Python script that interfaces with the simulator, allowing full control over decision-making logic.

**** Overview of proposed simulator***

This study introduces a lightweight (about 17MB), desktop-based robot maze simulation tool designed to support the teaching and learning of maze-solving algorithms through Python programming. One of the tool's key advantages is its platform independence; it can be executed on any Python IDE (Most students installed in the previous courses) without the need for installing additional applications or external libraries. This approach significantly reduces technical barriers and promotes ease of access for students and instructors alike.



```
def control(robot):
    global flag, index, saveChoices
    if robot.look("RIGHT"):
        if robot.look("AHEAD"):
            if not robot.look("LEFT"):
                robot.turn("LEFT")
                saveChoices.append("LEFT")
            else:
                robot.turn("BEHIND")
                saveChoices.append("BEHIND")
    else:
        robot.turn("RIGHT")
        saveChoices.append("RIGHT")
    robot.go()
    saveChoices.append("AHEAD")
```

Figure 1. An example of maze-solving program using wall-following algorithm in the proposed simulator

The tool offers a visually intuitive interface that enhances engagement and helps students better visualize the robot's behavior in real-time. Users interact with a virtual robot using a set of predefined commands such as turn_left(), move_forward(), and wall detection functions (as shown in Figure 1). These commands align with fundamental programming constructs and facilitate a practical, hands-on learning experience.

Importantly, the tool is highly configurable, allowing educators to adjust various parameters to suit different learning objectives (Figure 2). These include:

- Maze size, to control the complexity of navigation;
- Simulation speed, enabling step-by-step execution or faster visualization;
- Destination placement, to vary problem-solving paths;
- Maze design complexity, ranging from simple grid paths to intricate networks.

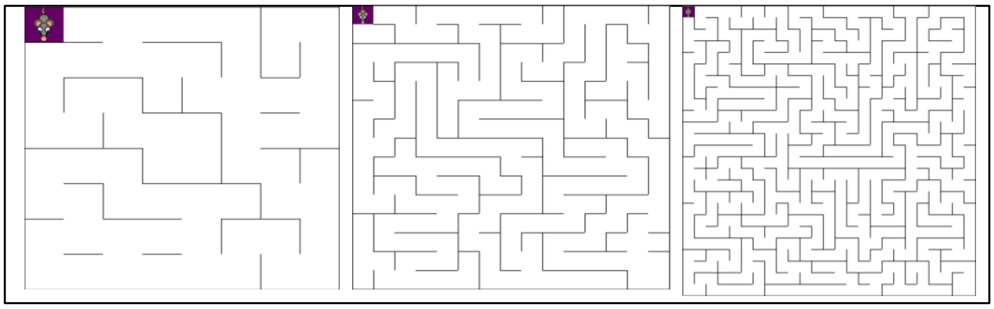


Figure 2. Maze variation in the proposed simulator

These adjustable features support the implementation and comparison of various maze-solving strategies (e.g., wall-following, backtracking, DFS/BFS) within one consistent environment. The tool's flexibility and accessibility make it a valuable resource for Informatics education, especially in courses focusing on algorithmic thinking and programming foundations.

2.3. Evaluation

2.3.1. Participants in the tool implementation

The participants in the implementation phase of the proposed tool were 65 undergraduate students from Cohort 72, enrolled in the Robot Programming course during the second semester of the 2024–2025 academic year at Hanoi National University of Education. These students were required to use both old C++ and the newly developed Python-based maze-solving simulation tool as part of their practical learning activities.

Over the period, the students engaged with the tool to learn, implement, and test various maze-solving algorithms, including the random mouse algorithm, wall-following, Trémaux algorithm, flood-fill algorithm, breadth-first search (BFS), and depth-first search (DFS). The simulation environment was integrated into weekly lab sessions and was used both for in-class activities and self-directed assignments. The purpose of this deployment was to evaluate and identify the tool's effectiveness in supporting students' understanding of robot behavior and learning experience in a simulated environment.

2.3.2. Survey tool

To ensure consistency and allow for direct comparison, the same survey instrument used in Section 2.1.3 was administered to a new cohort of students. These students had the opportunity to experience both the original and the newly developed simulation tools. The goal was to obtain more accurate and informed feedback regarding the improvements made and to evaluate the effectiveness of the new tool in addressing the limitations identified in the earlier version.

2.3.3. Result

The students' responses are summarized in Table 3. The criteria listed correspond to the components outlined in the survey described in Section 2.1.3.

Table 3. Comparison of students' mean scores and standard deviations between the old and new tools

Criteria	**Old C++ Simulator**		**New Python Simulator**	
	Mean	***Standard Deviation***	***Mean***	***Standard Deviation***
1.1	1.906	0.701	4.344	0.956
1.2	2.859	0.864	4.484	0.866
1.3	1.828	0.801	4.094	0.931
1.4	2.063	0.704	4.359	0.855
2.1	2.141	1.171	4.219	0.856
2.2	3.391	1.22	4.219	0.856
2.3	2.172	0.876	4.438	0.808
2.4	3.438	0.845	3.172	0.961
3.1	2.313	0.882	4.219	0.976
3.2	3.531	0.901	3.547	0.846
3.3	1.969	0.901	4.234	0.964
3.4	2.078	0.872	4.297	0.878
4.1	2.875	0.74	4.203	0.904
4.2	2.609	0.978	4.313	0.808
4.3	2.5	0.901	4.313	0.788
4.4	1.578	0.68	4.422	0.725
5.1	2.281	0.717	4.453	0.846
5.2	1.953	0.738	4.234	0.897
5.3	2.781	0.819	4.375	0.927
5.4	1.656	0.667	4.406	0.824

2.3.4. Data analysis

The comparative analysis of student feedback between the previous C++-based simulator and the newly developed Python-based simulator reveals a clear and consistent trend in favor of the new tool. In terms of Ease of Use, the Python version demonstrated a substantial improvement. For instance, item 1.1 ("The interface of the simulation tool is clear, easy to view, and user-friendly") increased from a mean score of 1.906 to 4.344, while 1.2 ("The robot control commands (e.g., 'turn left', 'turn right', 'go') are easy to understand and use.") rose from 2.859 to 4.484. These results indicate a more intuitive and accessible interface that better supports student interaction and command comprehension. The Academic Effectiveness scores also improved markedly. Students reported better understanding of key concepts and algorithms (2.1: 2.141 to 4.219) and a stronger belief that the tool improved their ability to solve programming problems (2.3: 2.172 to 4.438). However, perceptions of overall improvement in learning outcomes (item 2.4) were slightly lower with the new tool (from 3.438 down to 3.172). This discrepancy suggests that while the Python-based simulator was viewed as more effective for specific skills and applications, students did not yet perceive a corresponding improvement in overall academic results. As the measure is based on self-reports, future studies with larger samples should include objective performance data (e.g., test scores or course grades) to clarify whether the new tool translates into measurable learning gains.

With respect to User Experience, the new simulator provided a more engaging and enjoyable interaction. The largest jumps were observed in perceived enjoyment (3.3: 1.969 to 4.234) and confidence while using the tool (3.4: 2.078 to 4.297), reinforcing the tool's impact on student motivation and satisfaction. In the area of Algorithmic Thinking, improvements were similarly strong. Items such as 4.2 ("Programming robots in mazes enhances my logical thinking and problem-solving skills.") and 4.3 ("The tool helps me apply programming concepts (e.g., loops, conditionals) in practice.") both rose from around 2.5 to above 4.3, highlighting the tool's effectiveness in supporting problem decomposition, testing, and iteration core components of computational thinking.

Finally, under Technology Acceptance, all items showed marked increases. The most notable was item 5.4 ("I am willing to recommend this tool to peers or fellow students."), which increased from 1.656 to 4.406. This indicates strong student support for the broader adoption of the Python simulator in future instructional contexts. Standard deviation values across the new simulator's items remain relatively low (mostly <1), indicating a strong consensus among students regarding the tool's usability and effectiveness. Overall, the data strongly suggest that the new Python-based simulator significantly outperforms the previous C++ version across all measured dimensions. These improvements validate the design decisions behind the new tool, particularly its lightweight deployment, platform flexibility, and visual interactivity, and support its adoption for broader use in Informatics education focused on algorithmic thinking and robotics programming.

2.3.5. Qualitative evaluation

In addition to survey responses, students' qualitative comments reflect generally positive experiences with the new Python-based simulator. Several emphasized clearer and more intuitive visualization: "The tool shows the simulation more clearly, and I can adjust the images to my preference, which makes learning easier" (S1). Others highlighted practical benefits such as faster programming with fewer syntax errors compared to C++: "The Python tool helped me code more quickly and I did not encounter as many syntax errors" (S2). Students also valued being able to experiment with different maze sizes and observe step-by-step execution for easier debugging: "I could try different maze sizes and watch each step, which helped me find mistakes more easily" (S3). A few suggestions for further development were noted, including the ability to display the robot's optimal path for comparison and to simulate motor-level control as in a physical robot. Overall, the comments indicate positive perceptions of usability and learning support, while pointing toward useful directions for enhancement.

2.3.6. Suggestions for tool improvement

Following the implementation and student trial of the proposed Python-based maze-solving simulation tool, several suggestions for further development were collected. A major recommendation was to extend the tool into a web-based version, allowing students to interact with the simulator both online and offline across different platforms without IDE installation. In terms of user interface, students proposed enhancing the visual experience by incorporating 3D visualization and adding animations to make the simulation more engaging. Additionally, students expressed interest in advanced robotic features, such as diagonal movement and distance sensing, to enrich the learning experience and allow exploration of more complex algorithms. These recommendations provide valuable directions for future tool development to better support teaching and learning in algorithmic thinking and robotics programming.

3. Conclusions

The development and deployment of the Python-based robot maze simulation tool have yielded promising results in improving both usability and educational impact compared to the earlier version. Designed to run directly on any Python-supported IDE without requiring additional installations or libraries, the tool provides a lightweight and accessible solution for learning maze-solving algorithms. Its flexible configuration options, such as maze size, complexity, and simulation speed, have allowed for effective integration into various instructional scenarios.

Survey data show clear improvements in student satisfaction, engagement, and perceived learning outcomes, especially in terms of understanding algorithms and applying computational thinking skills. The tool also facilitates experimentation and debugging in an interactive environment, thereby enhancing students' algorithmic reasoning.

Looking ahead, future improvements will focus on expanding the tool's capabilities based on student feedback. These include transitioning to a web-based version, incorporating 3D visualization and dynamic effects, and adding advanced robotic features such as diagonal movement and distance sensing. Such enhancements aim to further align the tool with diverse teaching needs and foster deeper learning experiences in Informatics education.

Acknowledgement. This research was completed with the support of the university-level research project funded by Hanoi National University of Education, code SPHN24-22.

REFERENCES

[1] Ministry of Education and Training of Vietnam, (2018). Informatics Curriculum, General Education Program 2018.

[2] Hanoi National University of Education, (2019). Undergraduate Curriculum in Informatics Teacher Education.

[3] Claude AS, (1951). *Presentation of the Maze-solving machine*. Cybernetics: Transactions of the Eighth Conference, p. 173-180.

[4] Amy E, (2012). E*ducational robotics theories and practice: Tips for how to do it right.* In "Robots in K-12 education: A new technology for learning". Information Resources Management Association, p. 1-30.

[5] Bradley SB, Gwen N, Neal G, Viacheslav I. A, (2012). *Robots in K-12 Education: A New Technology for Learning*. IGI Global.

[6] Kin WL, Heng KT, Bẹnamin TE, Pavel P, (1999). Creative learning in school with LEGO (R) programmable robotics products. *29th Annual Frontiers in Education Conference, Designing the Future of Science and Engineering Education. Conference Proceedings.* FIE'99 Frontiers in Education (Vol. 2), p. 12D4-26.

[7] Jacqueline K, Cynthia B, (2014). Storytelling with robots: Learning companions for preschool children’s language development. *The 23rd IEEE international symposium on robot and human interactive communication*. IEEE, 643-648.

[8] Jeonghye H, Dongho K, (2014). R-Learning services for elementary school students with a teaching assistant robot. *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*. ACM/IEEE, 255-256.

[9] Amy E, (2014). Educational robotics for promoting 21st-century skills. *Journal of Automation Mobile Robotics and Intelligent Systems*, 8(1), 5-11.

[10] Marjo V, Erkki S, Eija K, (2008). How children's individual needs challenge the design of educational robotics. *Proceedings of the 7th international conference on Interaction design and children.* ACM, p. 274–281.

[11] Matthew B, Taylor M, Tom B, Carmen PS, Don D, (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, 22(4), 564-599.

[12] Matthew B, Uri W, (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, 24(5), 628–647.

[13] Duong BT, Pham MK, Le TY , Tran TK, (2022). Developing problem-solving competence of high school students oriented to STEM education through the topic of programming with the VEX IQ Robot. *Can Tho University Journal of Science*, 58, 36-45 (in Vietnamese).

[14] Duong QC, (2022). Designing experiential activities in teaching the section “Python programming to control Robots” for lower secondary school students. *Vietnam Journal of Education*, 22(18), 26-31 (in Vietnamese).

[15] Dang DP, Vu QR, Nguyen DA, Le HMN, (2021). Designing and organizing STEM teaching on the topic of a simple vacuum-cleaning robot, following the engineering design process for lower secondary school students. *Ho Chi Minh City University of Education Journal of Science*, 18(8), 1495-1508 (in Vietnamese).

[16] Nguyen KK, Dao MQ, Dao QK, (2022). Research on simulating a Palletizing Robot using Robotics Toolbox in Matlab. *Vietnam Journals Online*, 3, 106-109 (in Vietnamese).

[17] Nguyen TT, (2017). Teaching virtual interaction in Robotics Engineering classes. *VNU Journal of Science: Education Research*, 33(2), 75-80 (in Vietnamese).

[18] A L Nelson, E Grant, J M Galeotti, S Rhody, (2004). Maze exploration behaviors using an integrated evolutionary robotics environment. *Robotics and Autonomous Systems*, 46, 159-173.

[19] Fawaz YA, (2012). A Mobile Robot Solving a Virtual Maze Environment. *International Journal of Electronics, Computer and Communications Technologies*, 2(2), 1-7.

[20] Vernell D, Michael B, (2018). The Technology Acceptance Model (TAM): Exploring School Counselors’ Acceptance and Use of Naviance. *Professional Counselor*, 8(4), 369-382.

[21] Morgane C, Laila EH, Christian G, Barbara B, Francesco M, (2021). Teachers’ perspective on fostering computational thinking through educational robotics. International Conference on *Robotics in Education (RiE)*, 177-185.