Proceedings of the International Conference on Education, Vol. 11, Issue 1, 2025, pp.86-97

ISSN 2424-6700 online
DOI: https://doi.org/10.17501/24246700.2025.11107

IMPACT OF GAMIFIED PLATFORMS AND SIMULATION WEBSITES ON TEACHING PROGRAMMING CONCEPTS IN THE IBDP COMPUTER SCIENCE

Sakthipreya V*
Computer Science Department, Faculty, Indus International School, India

Abstract: The International Baccalaureate Diploma Programme (IBDP) Computer Science curriculum poses various challenges in teaching computational thinking (programming) concepts. As learners are from a variety of backgrounds other than computer science, they fail to understand the abstract concepts. This research examines how the gamified platforms and simulation-based tools help learners gauge the programming concepts. Even though there are studies that show the usage of gamification in STEM education, it lacks application in the IBDP Computer Science. This study incorporates a single-group quasi-experimental design with a wide range of pre- and post-assessments. Various assessment methods were incorporated to balance by including quantitative and qualitative data. Based on the results, it is evident that there is a 46% increase in learners' comprehension of the IBDP Computer Science programming concepts and a significant improvement in their engagement level. With the qualitative data gathered, it is confirmed that the gamified platforms and simulation-based tools are fun and engaging for learners throughout their learning journey. It also enhances 21st-century skills like problem-solving and critical thinking amongst learners. Teachers gain actionable recommendations for integrating gamified and simulation-based tools to improve engagement and conceptual understanding in programming.

Keywords: international baccalaureate diploma programme (IBDP), programming pedagogy, gamified platforms, student engagement, stem education, computer science pedagogy

Introduction

Learners enrolled in the IBDP Computer Science curriculum do not always have prior knowledge in programming. The traditional way of teaching often leads learners with less attention to problem-solving and difficulties in grasping programming concepts like data structures, recursion, and algorithmic thinking. This leads to reduced attention, poor problem-solving skills, and in many cases, loss of interest in the subject—sometimes resulting in students switching from Higher Level to Standard Level or opting out of the subject altogether. In order to overcome this challenge, the use of gamified and simulation-based tools are used to bridge this learning gap. While gamification and simulation-based tools have shown promise in enhancing engagement and understanding in STEM education broadly, their targeted use within the IBDP Computer Science curriculum has not been adequately studied. There is limited evidence on how such tools align with the IB's specific learning goals, such as developing inquiry skills, promoting deep conceptual understanding, and enhancing learner autonomy. The primary focus is on IBDP Computer Science teachers and learners, particularly those in Grades 11 and 12. The study is relevant for educators aiming to improve teaching strategies and for learners who need engaging, supportive approaches to master programming concepts. This study explores the integration of gamified platforms (e.g., Code.org, Scratch, Repl.it, Code Combat) and simulation-based tools (e.g., P5.js, Visual go) to improve learners' interaction with programming content. These tools are designed to make abstract computational processes more tangible and visually accessible. The research evaluates the effectiveness of these tools in bridging conceptual gaps and increasing learner

*Corresponding Author's Email:*sakthipreya.ib@gmail.com

engagement, directly addressing the shortcomings of traditional instruction in the IBDP Computer Science context.

Literature Survey

According to past researches, it is evident that there is a significant impact on the students' learning outcome through gamification and simulations in STEM education:

Lee and Shute (2020), along with Santos et al. (2021), explored the integration of gamification elements such as point systems, leader boards, and coding challenges in programming education. Their studies demonstrated a clear increase in learner motivation and participation, with students reporting higher engagement and enjoyment. However, while these strategies are effective in boosting short-term motivation, they rely heavily on extrinsic rewards. This poses a challenge when aligning with the International Baccalaureate Diploma Programme (IBDP), which emphasizes intrinsic motivation and inquiry-driven learning. Using a mixed-method approach involving pre-/post-motivation surveys and classroom observations across 12 weeks, the researchers concluded that although gamification can enhance engagement, it must be used cautiously within the IBDP to avoid undermining deeper learning goals.

Mujtaba and Reiss (2020), as well as Williams and Ahmed (2022), examined the use of virtual coding labs and programming simulators to improve conceptual understanding. These tools provided immersive, hands-on experiences that significantly improved learners' comprehension and debugging abilities. Their quasi-experimental study compared traditional learning methods with simulator-based learning, using post-tests and student interviews. The simulator group consistently outperformed the control group. However, the high-tech requirements of such tools can limit accessibility. Moreover, the studies did not align their frameworks with the IBDP's emphasis on global contexts, Approaches to Learning (ATL) skills, and constructivist pedagogy, highlighting a gap in contextual relevance.

Li and Gao (2023) introduced a hybrid instructional model that combines gamification and simulation techniques. Their experimental study, involving three cohorts (traditional, gamified, and hybrid), used pre- and post-tests alongside coding project evaluations. The hybrid group showed the highest gains in knowledge retention, coding fluency, and student motivation. Despite these promising outcomes, the approach has yet to be tested within the IBDP context. Its complexity and the need for significant teacher training present challenges for classroom implementation, especially in schools with limited exposure to IB pedagogy.

Zhang et al. (2019) designed a game-based coding curriculum aimed at developing computational thinking skills such as problem decomposition and logical reasoning. Conducted as a longitudinal study in middle-school settings, the research used computational thinking rubrics and student reflections to assess progress. While the study confirmed growth in algorithmic reasoning and sequencing abilities, it also revealed uneven results among lower-performing students. The lack of structured scaffolding—an essential feature of differentiated instruction in the IBDP—was a notable limitation.

Lee and Hammer (2021) investigated the influence of gamification on student retention and engagement in university-level programming courses. Based on a large-scale survey of over 500 students, the study found a positive correlation between the use of gamified platforms and increased course retention and completion rates. However, since the research was conducted in higher education settings, it did not address the flexible, criterion-based assessment approaches characteristic of the IBDP, such as Internal Assessments or Paper 1 and Paper 2 examinations. As such, its direct applicability to the IBDP remains limited.

Salen and Zimmerman (2022) explored constructivist learning through interactive simulations in high school coding environments. Using qualitative data from student reflections, classroom observations, and teacher interviews, the case study revealed that students developed deeper conceptual understanding and engagement when learning through inquiry and real-world scenarios. The approach closely aligns with IBDP philosophies. Nevertheless, the success of such simulation tools depends heavily on teacher facilitation and access to adequate technological infrastructure—factors that can vary widely between institutions.

Fernandes et al. (2023) focused on the role of gamified environments in enhancing students' problem-solving abilities. Their controlled experiment, guided by Bloom's taxonomy, showed that the treatment group exhibited significant gains in analytical and logical reasoning. These outcomes support the development of critical thinking skills central to the IBDP. However, the study did not explicitly map its activities to the programming-specific criteria or internal assessments required in the IB curriculum, suggesting a need for greater alignment.

Finally, Johnson and Smith (2024) conducted a descriptive study on the use of simulation-based programming tasks to promote industry readiness. Students involved in scenario-based simulations reported improved confidence in applying coding skills to real-world contexts, while teachers observed enhancements in employability skills. Despite its relevance to vocational education, the study lacks alignment with the academic and inquiry-based objectives of the IBDP. It also does not account for the structured, theory-oriented assessments required by the program, limiting its applicability in IB classrooms.

Research Objectives

This study focuses on:

- Investigating how gamified platforms and simulations influence student engagement and motivation in
- Evaluating the extent to which these tools improve conceptual understanding, critical-thinking, and problem-solving skills.
- Assessing students before and after the intervention of gamified and simulation learning techniques

- Offering practical recommendations for educators on effectively implementing these strategies in the classroom.

Methodology

This study was held in an international school offering the IBDP computer science curriculum. The IBDP Computer Science students (Grades 11 and 12) from the selected international school are the participants in the study. These students had varying levels of prior programming experience. A single-group quasi-experimental method was incorporated to research the gamification and simulations efficacy in teaching the IBDP programming concepts.

Role of Stakeholders

Learners actively participated in the intervention by using gamified and simulation-based tools during classroom instruction. They were the primary focus of the study, with their engagement, motivation, and comprehension being assessed.

Teacher played a dual role: implementing the gamified and simulation-based instruction, and observing/measuring student progress and behaviour during the intervention.

Data Collection Methods

Pre-Assessment: Conducted to establish baseline comprehension and engagement levels.

Intervention tools: Gamified platforms (Code.org, Scratch, Repl.it) and simulations (P5.js, visualgo) were integrated into classroom instruction.

Post-Assessment: Evaluated the impact of the intervention.

Assessment Methods Used

1. Likert scale engagement Surveys: Measured engagement and motivation levels using a Likert-scale engagement survey from students.

2. Observational Checklist: Assessed students' participation, problem-solving abilities, and interaction with gamified tools.

3. Multiple choice knowledge test: This test was designed to evaluate students' foundational understanding of key programming concepts such as variables, data types, control structures (loops and conditionals), and syntax.

4. Code debugging task: This helped learners to identify, explain, and correct the issues/ bugs in the given code.

5. Baseline coding challenge: This task is to solve a problem using programming constructs such as loops, conditionals, data structures, and variables.

6. Findings & Analysis

The implementation of gamified platforms and simulation-based tools led to a substantial improvement in both cognitive and affective domains of learning in the IBDP Computer Science curriculum. Quantitative data such as multiple choice knowledge test, code debugging task, and baseline coding challenge demonstrated a 46% increase in programming comprehension and problem-solving skills, indicating that students were not merely engaging superficially but internalizing complex computational concepts such as debugging, algorithm design, and logical reasoning.

Simultaneously, qualitative data such as likert-scale engagement survey and the observation revealed a marked rise in learner engagement and motivation, with over 85% of students expressing preference for gamified learning over traditional methods. The increase in active participation and collaborative behaviours suggests that the intervention positively influenced learners' attitudes toward programming, possibly reducing anxiety and fostering a growth mind-set.

Quantitative Analysis

Multiple choice knowledge test

To evaluate the impact of gamified and simulation-based instruction on students' understanding of fundamental programming concepts, a multiple-choice knowledge test was administered both before and after the intervention. The test included questions targeting core programming topics such as variables, data types, control structures (loops and conditionals), and syntax (refer Appendix 3). The results revealed a substantial improvement in students' understanding of programming concepts (refer Figure 1).

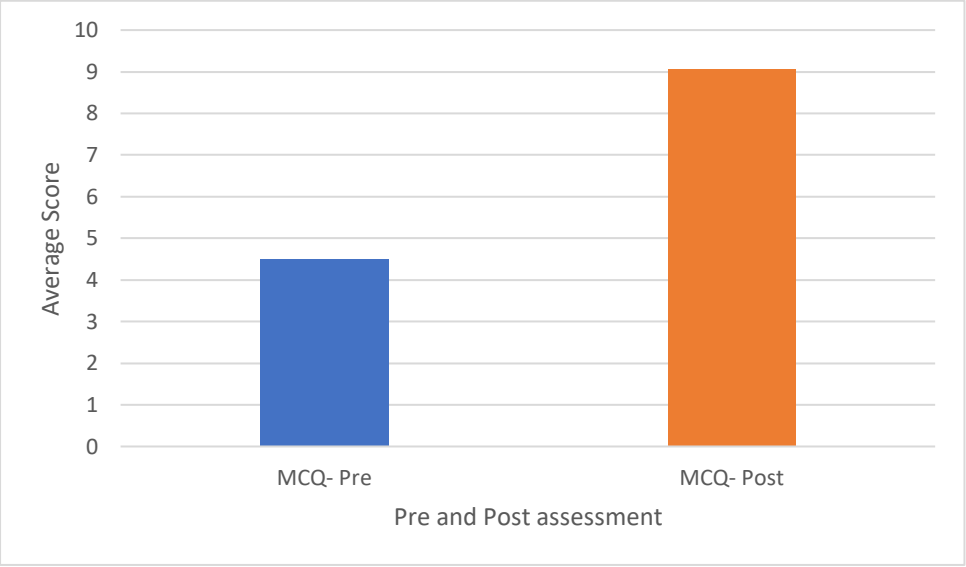

Figure 1: Comparison between pre- and post- assessments of multiple choice knowledge test

The average pre-assessment score was 4.5, indicating a moderate grasp of foundational programming knowledge. After the intervention, the average post-assessment score rose to 9.06, demonstrating a 46% increase in performance (refer Appendix 3a). This significant gain in scores reflects a strong positive impact of using gamification and simulation-based learning tools in the IBDP Computer Science classroom. Students not only showed improved factual recall but also demonstrated enhanced conceptual understanding and application skills.

Code debugging task

To assess students' ability to identify, explain, and correct errors in programming, a code debugging task was administered before and after the intervention (refer Appendix 4). The task involved analysing and fixing short segments of faulty code, which evaluated students' logical reasoning, understanding of syntax, and familiarity with programming constructs such as loops, conditionals, and variables (refer Appendix 4a).

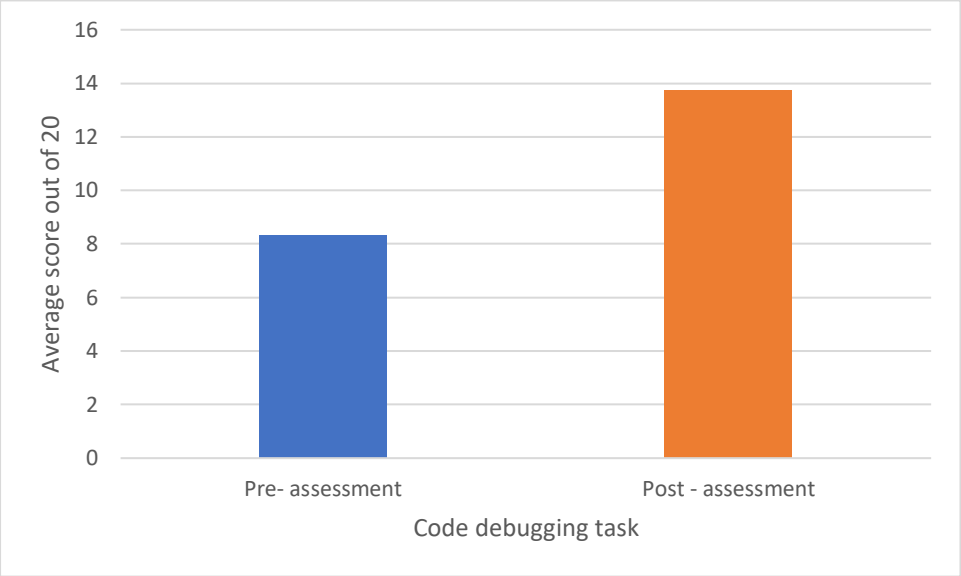

Figure 2: Comparison between pre- and post- assessments of Code debugging task

The task was scored on a 20-point scale, with students evaluated on the following criteria: Identification of bugs, Explanation of the error, Corrected solution, and Code clarity and structure (refer Appendix 4). The average pre-assessment score was 8.31, reflecting emerging but inconsistent debugging skills. The post-assessment score improved to 13.75, indicating a 65.4% increase in performance (refer Figure 2) (refer Appendix 4b). This notable increase in average scores suggests that the use of gamified platforms (e.g., Code.org, Scratch) and visual simulations (e.g., P5.js, Visual go) significantly enhanced students' ability to: analyse code logically, identify and correct syntactical and logical errors, and improve the structure and readability of code.

Baseline coding challenge

To evaluate students' practical application of programming concepts, a baseline coding challenge was administered both before and after the intervention. The challenge required students to design and implement a solution using constructs such as loops, conditionals, variables, and data structures (e.g., arrays, lists, stacks, queues). This assessment aimed to measure their problem-solving skills, code logic, and structural understanding of programming (refer Appendix 5a). The challenge was graded on a 100-point scale using a structured rubric (refer Appendix 5).

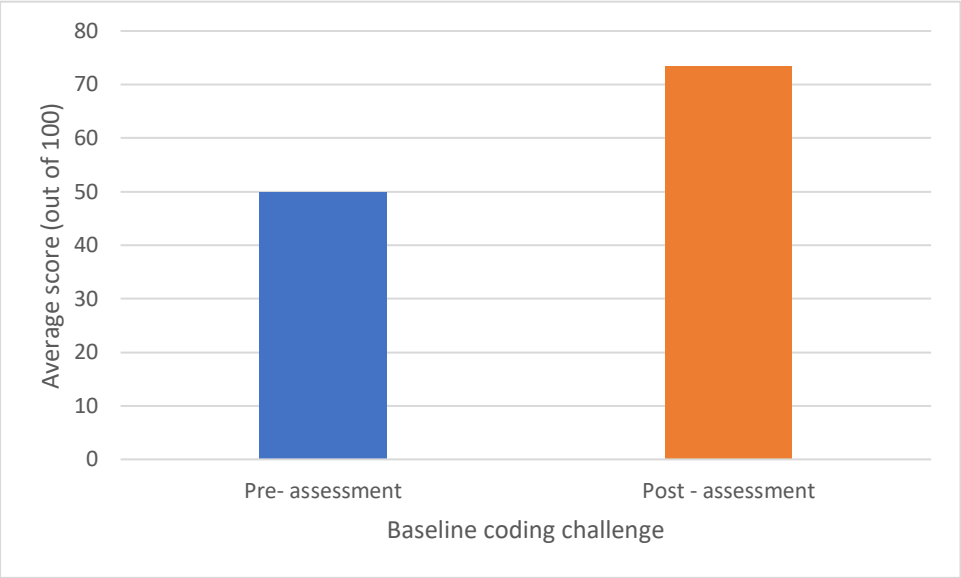

Figure 3: Comparison between pre- and post- assessments of Baseline coding challenge

The average score improved from 50.00 in the pre-assessment to 73.46 in the post-assessment. This reflects a 46.92% increase in performance (refer Figure 3 & Appendix 5b) after the gamified and simulation-based learning intervention. The significant gain in scores indicates that students were better equipped to design algorithmic solutions, implement clean, structured, and efficient code, and apply programming logic to real-world problems.

Qualitative Analysis

Likert scale engagement Surveys

The data presents average scores from five questions (refer Appendix 1), measured before (pre) and after (post) an intervention or activity. The average scores across all questions show a significant increase from pre- to post-assessment, indicating a positive change in participants' responses. Question 1 shows an increase from 2.86 (pre) to 4.43 (post), suggesting that participants improved in the area assessed by this question, possibly reflecting enhanced understanding, confidence, or satisfaction. Question 2 was recorded twice (likely a typo, but assuming two similar questions or repeated

measurement), with average scores increasing from 2.61 to 4.60 and from 2.66 to 4.45 respectively. Both instances demonstrate a marked improvement, reinforcing the consistency of the positive outcome. Question 4 shows a rise from 2.5 to 4.6, reflecting a notable shift in participant responses after the intervention. Question 5 also indicates significant growth, with scores increasing from 2.4 to 4.61 (refer Figure 4).

Overall, the qualitative analysis of these scores suggests that the intervention effectively enhanced participants' performance across all measured dimensions. The substantial increase in average scores implies improved competency, engagement, or satisfaction levels, pointing toward the intervention's success in achieving its objectives (refer Appendix 1a).

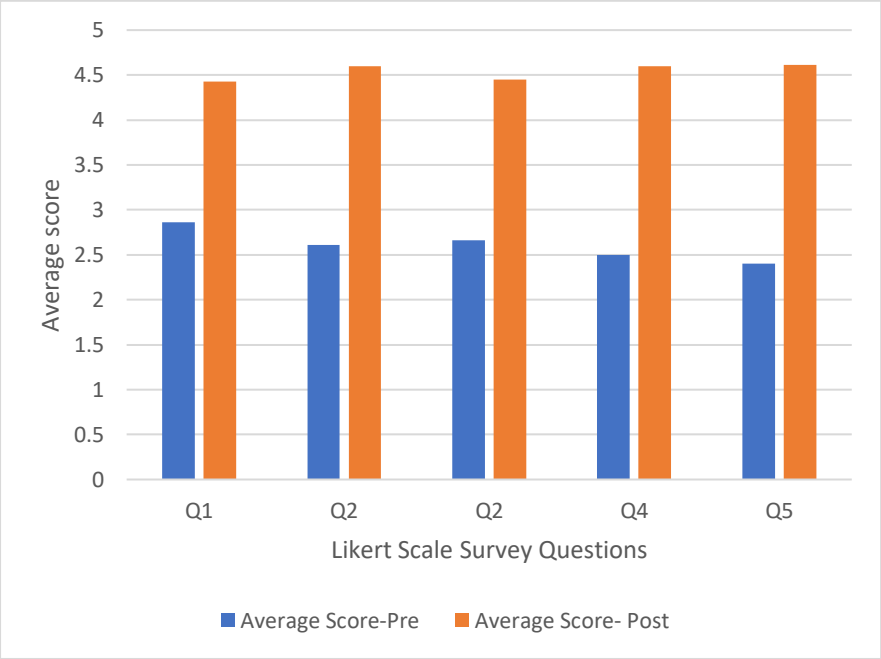

Figure 4: Comparison between pre- and post- assessments of Likert Scale engagement survey

Observational Checklist

To gain deeper insight into student engagement, learning behaviours, and interaction with tools during the intervention, a structured observational checklist was employed. This qualitative assessment focused on the following domains: Participation in Class Activities, Problem-Solving Abilities, and Interaction with Gamified & Simulation Tools. Observers (teachers/facilitators) used checklist (refer Appendix 2) in each domain during both pre- and post-intervention sessions.

Participation in Class Activities showed a remarkable increase from 2.61 to 4.43, suggesting that students were more involved and actively contributing to classroom discussions and exercises during the intervention. Problem-Solving Abilities improved from 2.50 to 4.45, reflecting students' enhanced logical thinking, perseverance, and confidence in tackling programming challenges using gamified methods. Interaction with Gamified & Simulation Tools increased significantly from 2.40 to 4.60, indicating that students found platforms like Scratch, P5.js, Visual go, and Code.org highly engaging and user-friendly, promoting frequent and meaningful interaction (refer Figure 5).

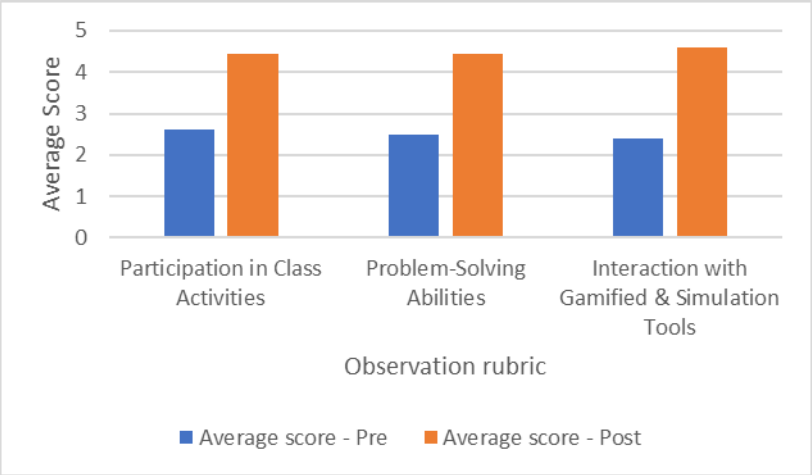

Figure 5: Comparison between pre- and post- assessments of observation

The substantial improvement across all three domains supports the hypothesis that gamified and simulation-based teaching strategies positively impact student motivation, engagement, and learning behaviours (refer Appendix 2a). The qualitative data aligns with the quantitative improvements in test scores, further validating the transformative impact of interactive learning tools on the cognitive and affective domains in the IBDP Computer Science classroom.

Conclusion

This study proves that the simulation-based learning and incorporating gamified platforms can increase the IBDP Computer Science students' motivation, engagement, and understanding. It is evident from the data analysis that there is a remarkable increase in the students' engagement levels and the understanding of the programming concepts. Teachers can inculcate critical thinking skills in students while making programming concepts more understandable and interesting by implementing interactive & game-like strategies. Through this study, students' learning objectives can be achieved and the engagement gaps be closed in teaching the IBDP programming concepts. These learning techniques can also be improved or modified as per students' learning abilities.

Recommendations for educators

Based on what we've learned, here are some ways teachers can make gamification a meaningful part of their classrooms:

1. Mix Old and New: Don't throw out what works! Blend tried-and-true teaching methods with fun, game-based activities. This way, your classroom stays organized but also feels fresh and exciting-students stay interested without feeling overwhelmed.

2. Bring Concepts to Life: Programming can be a tough nut to crack, especially when the ideas are abstract. Use simulations and visual tools to make these concepts real and relatable. When students see how things work in the real world, those tricky topics become much easier to understand.

3. Let Students Take the Wheel: Give students the chance to explore, experiment, and solve problems on their own. Interactive, hands-on tasks not only build confidence but also make learning more personal-and a lot more fun!

4. Listen and Learn Together: Keep the conversation going with your students. Use surveys, watch how they interact in class, and track their progress. Their feedback is gold-it helps you fine-tune your approach and make sure you're really meeting their needs.

Future research directions

1. Control Group: Next time, it would be helpful to have a group learning the "old-fashioned" way. This makes it easier to see exactly how much of a difference gamification and simulations make.

2. Long-Term Impact: It's one thing for students to do well right after a lesson, but how much do they remember months later? Future research could follow up with students to see how well they retain what they've learned and how their problem-solving skills grow over time.

3. Explore AI-Powered Tools: There's a lot of potential in using AI tools that adjust to each student's pace and needs. These could make learning even more personalized. Of course, this comes with its own challenges-like making sure teachers are comfortable with the technology and keeping student data safe.

Study limitations

While this study shines a light on how gamified and simulation-based tools can boost learning in IBDP Computer Science, there are a few limitations:

1. No Control Group: Because everyone in the study used the new methods, it's hard to say for sure if the improvements were due to gamification alone.

2. Narrow Focus: All participants were from the IBDP stream, so the results might not apply to other types of students or curricula.

3. Short-Term Gains: The study mostly looked at immediate improvements in understanding and engagement. We don't know yet if these benefits stick around in the long run.

4. Specific Tools and Context: The success of the approach might depend on the particular tools used or how the teacher ran the sessions. This means it might not work the same way in every classroom.

5. Limited Scope: The focus was mainly on abstract programming concepts, so how well this approach would work for other areas of Computer Science is unanswered.

Declaration of Interest Statement

The author declare that they have no conflict of interests.

References

Fernandes, A., Li, K., & Chen, Y. (2023). Gamification and computational thinking: A study on problem-solving in computer science education. International Journal of Educational Technology, 45(3), 112-130.

Johnson, R., & Smith, P. (2024). Enhancing programming pedagogy through simulation-based learning. Journal of Computer Science Education, 32(1), 56-78.

Lee, J., & Hammer, J. (2021). Gamification in education: A systematic review and meta-analysis of learning outcomes. Educational Psychology Review, 39(4), 765-792.

Lee, M., & Shute, V. (2020). The impact of gamified learning environments on student engagement and achievement. Computers & Education, 148, 103789. https://doi.org/10.1016/j.compedu.2019.103789

Li, Y., & Gao, W. (2023). Hybrid learning approaches: Integrating gamification and simulations in STEM education. Interactive Learning Environments, 31(2), 189-207.

Mujtaba, T., & Reiss, M. (2020). Simulation-based learning in computer science: Evaluating the effectiveness of interactive visualizations. International Journal of STEM Education, 7(1), 89-104.

Salen, K., & Zimmerman, E. (2022). Gamification and constructivist learning theories: Aligning game mechanics with pedagogical strategies. Journal of Digital Learning Research, 40(2), 201-223.

Santos, F., Mora, A., & González, C. (2021). Engagement and motivation in gamified learning: A study of digital platforms in computer science education. Journal of Educational Technology, 27(3), 145-167.

Williams, R., & Ahmed, H. (2022). Using interactive simulations to improve conceptual understanding in computer science. Advances in Engineering Education, 15(4), 58-74.

Zhang, L., Patel, N., & Kim, J. (2019). Game-based learning and computational thinking: Evidence from secondary school students. Journal of Emerging Technologies in Learning, 14(6), 99-115.