

Boosting Student Motivation through Game-based Learning in Programming Education with Gamify-IT

Niklas Meißner 

University of Stuttgart
Stuttgart, Germany

niklas.meissner@iste.uni-stuttgart.de

Niklas Krieger 

University of Stuttgart
Stuttgart, Germany

niklas.krieger@iste.uni-stuttgart.de

Sandro Speth 

University of Stuttgart
Stuttgart, Germany

sandro.speth@iste.uni-stuttgart.de

Steffen Becker 

University of Stuttgart
Stuttgart, Germany

steffen.becker@informatik.uni-stuttgart.de

Abstract

Game-based learning (GBL) has emerged as a powerful instrument for enhancing student engagement and motivation in education. This paper evaluates the impact of Gamify-IT, an educational game designed for programming education, on student motivation and learning experiences. Building on the original Gamify-IT concept, we extended it with novel features, including diverse minigames, leaderboards, and a redesigned achievement system, to align with Marczewski's HEXAD player types. The platform was used in a first-semester programming course by about 100 students, and its impact was evaluated through comparative analyses of two cohorts in consecutive CS1 programming courses: one limited to traditional teaching methods and the other incorporating Gamify-IT. Our results demonstrate that Gamify-IT has a positive impact on student motivation, as evidenced by higher initial and sustained motivation levels throughout the semester. Feedback from a survey regarding the usefulness and experiences of the platform highlights its strengths, including an engaging and innovative learning approach, game design, and visuals, while also identifying areas for improvement, such as addressing technical issues and expanding game variety. Despite these challenges, Gamify-IT demonstrates the potential of GBL in programming education, offering valuable insights for designing inclusive and effective gamified educational tools for programming education.





CCS Concepts

• **Applied computing** → **Interactive learning environments**; • **Social and professional topics** → **CS1**; **Software engineering education**.

Keywords

Game-based Learning, Student Motivation, Interactive Learning, Programming Education, Software Engineering Education

ACM Reference Format:

Niklas Meißner , Sandro Speth , Niklas Krieger , and Steffen Becker . 2026. Boosting Student Motivation through Game-based Learning in Programming Education with Gamify-IT. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE TS 2026)*, February 18–21, 2026, St. Louis, MO, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3770762.3772508>

1 Introduction

Programming education forms the cornerstone of computer science curricula, equipping students with critical skills required for computational problem-solving and software development [7]. However, educators consistently face challenges in sustaining student engagement and motivation [5, 14], particularly in large introductory programming courses such as CS1. Despite efforts to integrate innovative teaching methods, the abstract nature of programming concepts and the steep learning curve often discourage students, leading to suboptimal learning outcomes [5, 6]. Game-based learning (GBL) and gamification have emerged as promising approaches to tackle this challenge [1]. By incorporating game mechanics into educational contexts, lecturers seek to foster motivation, engagement, sustained learning, and improved teamwork [2, 11, 13].

In previous work, we introduced *Gamify-IT* [20], a Unity- and web-based educational game that offers an immersive role-playing experience for programming education. The initial prototype demonstrated potential, showing students were motivated to engage with course content through the platform's minigames and interactions like NPCs, books, and dungeons. However, the evaluation was limited to a small user study, leaving questions about scalability and broader applicability unanswered. Moreover, no insights were obtained on whether Gamify-IT boosts student motivation in programming education through prolonged use.

Therefore, building on this foundation, we extend the capabilities of Gamify-IT to adapt it for the CS1 programming education course at the University of Stuttgart and evaluate its impact on a larger cohort in a field study. Our enhancements incorporate Marczewski's HEXAD player types [10], refining the platform to accommodate diverse motivational drivers. The original Gamify-IT platform focused on Bartle's player types [3], which are now rather outdated. Furthermore, we address technical and usability limitations identified in the original implementation and introduce additional features to increase student interaction and satisfaction.



This work is licensed under a Creative Commons Attribution 4.0 International License. *SIGCSE TS 2026, St. Louis, MO, USA*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2256-1/26/02

<https://doi.org/10.1145/3770762.3772508>

Our enhancements include new single- and multiplayer minigames (e.g., *Tower Defense*, *Memory*, and *Towercrush*) and enriched game mechanics (e.g., *Achievements*, *Leaderboards*, and *Customizables*). To assess Gamify-IT and the efficacy of the enhancements, we deployed the revised version of Gamify-IT in a CS1 programming course, where 101 students tested and played the game over the four weeks we offered it. Students used it to explore key programming topics such as object-oriented programming, control flow structures, and code error analysis. Our research questions are as follows:

RQ1: “Does the Gamify-IT platform influence student motivation to engage with programming concepts?”

RQ2: “What are students’ perceptions of the usability of Gamify-IT as a learning tool in programming education?”

This paper presents the results of our study and highlights insights into the potential of GBL environments in programming education. Section 2 provides an overview of the CS1 course, in which we applied and investigated the use of Gamify-IT. Section 3 explores related work in GBL, gamification, and programming education, incorporating the foundational Gamify-IT approach. Section 4 describes the enhancements made to Gamify-IT. Section 5 discusses our evaluation process, results, and implications. Then, Section 6 addresses threats to validity, Section 7 presents our lessons learned, and Section 8 summarizes our conclusions, respectively.

2 CS1 Programming Course Background

The CS1 programming course at the University of Stuttgart is a mandatory first-semester course introducing computer science students and related study majors to foundational programming concepts and principles. This semester (winter term 2024/25), the course served over 500 students, making it one of the most populous courses in the curriculum. The course employs a blended learning approach, combining traditional lectures with interactive lab sessions to reinforce theoretical knowledge through practical application. *Java* is the primary programming language taught in this course as it is widely used in industry and academia. The curriculum covers essential topics, including variables and data types, control flow structures, methods, object-oriented programming, and fundamental algorithms. Emphasis is placed on developing problem-solving skills and understanding programming paradigms, equipping students with a robust foundation for more advanced computer science courses. Students must complete weekly programming assignments to progressively improve their skills and understanding of the material.

3 Related Work

GBL and gamification in education have recently gained considerable attention as a method to improve student motivation and engagement [1, 4, 13]. In the field of programming education, several studies have shown the potential of educational games to overcome the challenges associated with teaching and learning programming:

Ahmad et al. [1] conducted a study showing that integrating game elements such as badges, experience points, and challenges significantly improved students’ motivation and performance in



Figure 1: Gamify-IT overworld.

computer science courses. Similarly, Alhammad and Moreno [2] systematically mapped gamification in software engineering education, identifying leaderboards, achievements, and narratives as key factors in driving student engagement.

Gamify-IT combines role-playing game elements with educational content delivery [20]. The platform’s immersive environment, consisting of worlds, dungeons, and minigames, allowing students to explore and interact with learning materials playfully. Figure 1 shows the Gamify-IT overworld, where students can wander around, explore the world, gain and repeat knowledge through NPCs and books, and test their knowledge in minigames. Despite its innovative design, Gamify-IT faces challenges that limit its broader application. The game focuses on Bartle’s player types [3], which are outdated nowadays. While the evaluation demonstrated promising results in terms of engagement, the study was limited to a small cohort, and the platform’s scalability and effectiveness in larger classrooms remain unexplored.

LightBot is a puzzle-based educational game by Yaroslavski [22] that introduces programming concepts such as loops, conditionals, and recursion through visual commands. Players control a robot to complete tasks, reinforcing logical thinking without requiring coding [22]. Mathrani et al. [11] found that *LightBot* enhanced students’ understanding of programming constructs, particularly for beginners, by fostering engagement and reducing anxiety.

CodinGame [8] and *CodeCombat* [9] are two other widely recognized platforms for learning programming. *CodinGame* immerses students in real-world coding challenges across multiple programming languages, appealing to advanced learners who enjoy solving complex problems in competitive environments. *CodeCombat* offers a fantasy-themed setting where students navigate characters through coding challenges, helping them learn syntax and basic programming constructs. However, *CodinGame* emphasizes advanced problem-solving tasks, while *CodeCombat*’s limited focus is on syntax learning and minimal integration of programming concepts.

Based on the related work and the constraints of our CS1 programming course, we aimed to provide students with an immersive game that allowed them to explore and unlock learning materials on their own, while also facilitating the integration of more abstract programming concepts with non-programming tasks. Therefore, chose and enhanced *Gamify-IT* to meet our new requirements.



Figure 2: Some parts of the Gamify-IT enhancements and improvements.

4 Enhancements of Gamify-IT

To address the challenges and limitations identified in the original Gamify-IT platform and to frame the educational game to the CS1 programming course, we implemented several enhancements to improve functionality, scalability, and user engagement, as we identified these as critical factors for the use of an educational game [16]. We wanted to encourage students to play Gamify-IT in our course and motivate them to continue playing and learning with additional minigames and game mechanics. Thereby, we followed the guidelines for gamification in education from Raymer [18]. As Gamify-IT was based on Bartle’s player types [3], which is somewhat outdated nowadays and there are more recent studies on player types, we have changed the approach to address the different motivational drivers of individual students. Therefore, the enhancements were guided by Marczewski’s HEXAD player types [10], which provide a framework for understanding diverse student motivations. A demo video of all Gamify-IT enhancements is available on YouTube¹.

4.1 Minigames

(1) *Tower Defense*: Tower Defense integrates multiple-choice quizzes into a strategic gameplay experience. The left part of Figure 2 shows the minigame. The students progressively solve tasks (top right corner of the minigame) to improve their towers, which they use to defend their castle against waves of enemies. This combination of task-solving and strategy is aimed at students who enjoy tactical thinking and reinforces their learning through frequent tasks. In future, these tasks should also include small programming exercises.

(2) *Memory*: This game builds on the classic card-matching concept. Students are presented with pairs of cards and must flip them to match. Cards can be text, images, or code in Markdown format and allow mapping of programming concepts to code or images (e.g., UML). Memory encourages active recall and reinforces programming fundamentals in an engaging and low-pressure format.

(3) *Towercrush*: This game enables students or teams to answer programming-related questions as quickly as possible, fostering engagement and competition among students. Correct answers allow players to attack their opponent’s tower, creating a fast-paced environment that rewards speed, accuracy, and collaboration.

4.2 Game Mechanics

New game mechanics in Gamify-IT introduce established features from non-educational games, aiming to improve student engagement, motivation, and interaction. By addressing diverse motivational drivers and creating immersive experiences, these mechanics aim to make learning more engaging and rewarding:

(1) *Achievements*: A revamped achievement system allows instructors to define and create tailored milestones for students and enhance them with distinctive icons for visual appeal. The right part of Figure 2 shows the achievements gathered by a student. This system encourages students to achieve specific learning objectives and game-related milestones, fostering a sense of accomplishment and engaging to continue playing and learning. For example, completing a minigame with a perfect score might unlock a “Programming Perfectionist” badge, motivating students to strive for excellence.

(2) *Leaderboards*: The leaderboard system introduces a competitive element to the platform but with a league-based structure. Students are grouped into smaller leagues of similar levels, ensuring fair competition and reducing intimidation. Promotions within leagues serve as additional incentives for continuous engagement, creating a dynamic and encouraging environment. By default, students are displayed anonymously with a pseudonym on the leaderboard, but have the option of publishing their name. This should ensure that students are not exposed to others or becoming demotivated.

(3) *Music and Sounds*: Unique background music and sound effects have been integrated into the platform to enhance immersion. Each minigame and overworld activity features custom audio tracks and sound effects, aiming to create an atmosphere that aligns with the gameplay and maintains student engagement.

(4) *Currency and Shop*: The inclusion of a currency system introduces virtual coins that students can earn by completing minigames with their respective tasks. These coins can be spent in the in-game shop to purchase customizations, making progress more tangible and adding a layer of gamified rewards to educational activities. The middle part of Figure 2 shows the Shop with customizables.

(5) *Customizables*: Students can personalize their in-game avatars with items purchased from the shop. Hats, clothing, and accessories (i.e., glasses) provide a fun and creative outlet. They are displayed in the overworld, encouraging students to take pride in their avatars and interact with peers in a visually stimulating environment.

¹<https://youtu.be/j4zkwCWUHUI>

Category	Feature	HEXAD Player Type
Game Mechanics	<i>Overworld*</i>	Free Spirits
	Achievements	Achievers, Players
	Leaderboards	Achievers, Disruptors, Players
	Currency and Shop	Achievers, Free Spirits, Players
	Customizables	Free Spirits, Socializers
Minigames	Tower Defense	Achievers, Free Spirits, Players
	Memory	Achievers, Philanthropists
	Towercrush	Disruptors, Players, Socializers
	<i>BugFinder*</i>	Philanthropists, Socializers
	<i>Chickenshock*</i>	Achievers, Players
	<i>Crossword Puzzle*</i>	Achievers
	<i>Finitequiz*</i>	Achievers, Players

Table 1: Marczewski’s HEXAD player types in Gamify-IT. Features that are denoted with a * were already existing in the original Gamify-IT implementation [20].

4.3 Considered HEXAD Player Types

The new features in Gamify-IT were carefully designed to address the six HEXAD player types described by Marczewski [10], ensuring a wide range of motivational drivers are considered, motivating all students with different elements. Below is an introduction to each player type and how the platform’s enhancements engage them:

(1) *Achievers*: Motivated by success and accomplishment, Achievers thrive on completing goals and earning recognition. The revamped achievement system provides numerous milestones to reach, with visually distinct badges that celebrate progress. The leaderboard system also appeals to Achievers by allowing them to measure their performance against others and aim for top rankings.

(2) *Socializers*: Socializers value interaction and collaboration. We implemented an overworld multiplayer mode with avatar customizations. Students can express themselves visually in the overworld, fostering connection and engagement. Games like Towercrush create opportunities for cooperation and shared success.

(3) *Free Spirits*: This player type seeks autonomy and exploration. The overworld and dungeon designs cater to Free Spirits by providing large areas to discover and explore. The introduction of non-linear gameplay paths, and the ability to customize avatars align with their desire for creative freedom and independence.

(4) *Philanthropists*: Altruistic by nature, Philanthropists find joy in helping others and creating positive experiences. The team play aspect of Towercrush enables these players to support their peers during competitive matches. Future iterations of Gamify-IT could enhance features with in-game mentoring or resource sharing.

(5) *Players*: Players are motivated by rewards and extrinsic incentives. The achievement and currency system is specifically designed for this group, as it rewards progress. The ability to unlock items serves as an ongoing motivator to engage with the platform content.

(6) *Disruptors*: Disruptors enjoy challenging systems and introducing change. Competitive features like Towercrush offer a dynamic platform to outwit opponents and leaderboards also allow them to dominate the competitive scene.

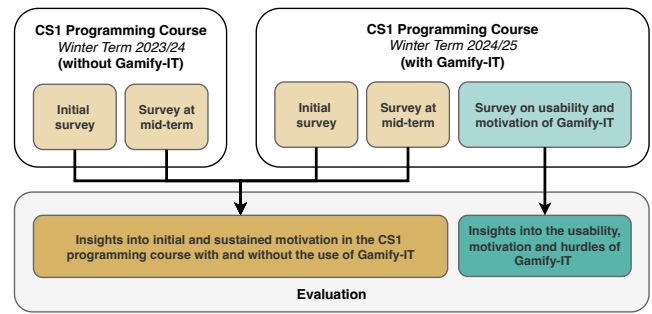


Figure 3: Design of our evaluation process.

5 Evaluation

This chapter outlines the evaluation of the *Gamify-IT* platform, incorporating the developed enhancements in this paper. For this purpose, the revised Gamify-IT platform was used in a real programming course for first-semester students at the University of Stuttgart. Gamify-IT was offered voluntarily to provide additional learning support for students who preferred to learn the course material independently and in a playful manner. However, Gamify-IT was not mandatory, so not all CS1 participants interacted with it.

5.1 Evaluation Process

The evaluation aimed to measure the impact of Gamify-IT on student motivation in our CS1 programming course while gathering detailed feedback about its usability and effectiveness. Figure 3 illustrates the design of our evaluation process to answer the posed research questions (see Section 1). The first method involved a comparative analysis of motivation levels in two cohorts of programming students: last year’s course (Winter Term 2023/24 - “Course 1”), where Gamify-IT was not used, and this year’s course (Winter Term 2024/25 - “Course 2”), which integrated and used Gamify-IT for four consecutive weeks after the beginning of the course. Surveys were conducted at the beginning and middle of the semester for both cohorts, using the same set of questions to get self-assessed motivation levels and preparedness for the course. The data from these surveys was analyzed using an independent t-test to compare motivation levels between the cohorts.

The second method focused on gathering detailed feedback from students in Course 2 through a comprehensive survey. The game statistics showed that 101 students played the game. This survey captured quantitative data on Gamify-IT’s usefulness and motivational impact, along with qualitative feedback on what students enjoyed most and how the platform could be improved. The responses were analyzed using descriptive statistics to summarize the quantitative ratings and thematic analysis for the open questions to identify insights. All survey data are publicly available on Zenodo².

The content of the CS1 course was identical in both semesters, to ensure that changes in content and presentation formats had as little impact as possible on the results. To create the content in Gamify-IT, we used the content from the lecture slides for *NPCs* and *books*, and we used *EvalQuiz* [17] as well as our expertise in creating programming [15] and UML exercises [21] using LLMs.

²<https://doi.org/10.5281/zenodo.14636201>

5.2 Survey Design

We constructed a total of three surveys with questions aimed at collecting information suitable for the analysis. The questions were created collaboratively by the authors and tested on test candidates. The surveys were then distributed to students by instructors in class and online in a discussion forum for our CS1 programming course. All surveys and collected data are available on Zenodo².

The surveys comprised: (1) An initial survey to record students' self-assessed motivation, preparation, and prior knowledge at the course start, as well as some demographic questions about their degree program and their current semester. All questions were asked on a 10-point scale, with "1" being the lowest level of agreement and "10" being the highest. (2) A mid-term survey to record again the self-assessed motivation of the students and which factors influence students' motivation in the course, such as feedback from tutors and exercise results. All questions here were also asked on a 10-point scale. Questions about Gamify-IT were not asked in this survey. (3) A Gamify-IT experience survey addressing students' perceptions and opinions about the usefulness and motivational impact of Gamify-IT in learning programming concepts. This survey focused on understanding students' perceived motivational aspects as well as their experiences and impressions of specific features of the game. The survey comprised questions on a 10-point scale, questions on a Likert scale (from "strongly disagree" to "strongly agree"), single and multiple choice questions, and open-ended questions.

All surveys were conducted using Microsoft Forms. The initial and mid-term surveys were kept as short as possible and required only a minute to complete. We aimed to get as many students as possible. The third survey was estimated to take 3-4 minutes, as we wanted to gain deeper insights, including open-ended questions.

5.3 Results

5.3.1 Changes in Motivation Levels Within and Between Cohorts.

The analysis of motivation levels revealed significant differences both within and between the two cohorts. As shown in Figure 4, in Course 1, where Gamify-IT was not used, 345 students participated in the initial survey with an average self-assessed motivation level of 7.21 on a 10-point scale. By the middle of the semester, 293 students participated, and their motivation level decreased to 5.56.

For Course 2, which utilized Gamify-IT, 352 students participated in the initial survey with an average self-assessed motivation level of 7.10 on a 10-point scale at the start of the semester. By the middle of the semester, it decreased to 6.03 with 185 participating students.

When comparing mid-semester motivation levels between the two cohorts, Course 2 showed a clear advantage. The mean motivation level for Course 1 was 5.56, while for Course 2, it was 6.03. An independent t-test yielded a t-statistic of -2.08 and a p-value of 0.038, confirming that this difference was statistically significant.

5.3.2 Feedback from the Gamify-IT Experience Survey. The additional survey on Gamify-IT in Course 2 provided further insights into how students perceived and experienced the game. 42 students participated in the survey. Around 62% of the participants reported testing and playing the game. Among those who did not, the most common reasons were a lack of time (70%), technical issues (10%), and disinterest in games (10%). Some students noted challenges with account setup or unclear instructions as barriers to participation.

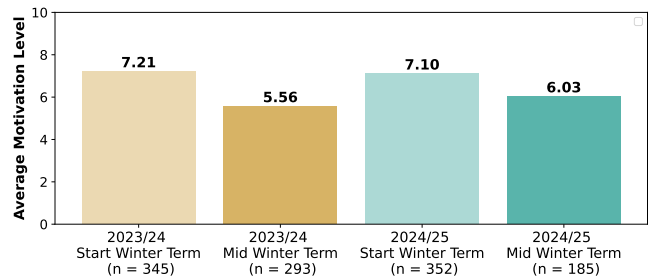


Figure 4: Comparison of motivation in CS1 with two cohorts.

Students who engaged with Gamify-IT rated the game's usefulness and motivational impact moderately, with an average usefulness rating of 5.50 out of 10 and an average motivational perception of 5.27. While these ratings indicate room for improvement, many students appreciated the platform's innovative approach to learning. 53.8% found Gamify-IT particularly effective in making programming concepts more enjoyable, while 23% disagreed. 65.3% of students reported feeling a sense of accomplishment when completing minigames. However, 53.8% stated that Gamify-IT did not encourage them to work on programming topics beyond the course.

The design of Gamify-IT was another area where students offered feedback. Here, 69.2% of the students indicated that they found the game design and visuals appealing, while 11.5% disagreed. Many respondents felt that the challenges and minigames struck an appropriate balance between fun and education (42.3% (strongly) agreed, 30.8% (strongly) disagreed). However, some students noted technical issues, particularly with the crossword minigame, which detracted from their experience. Suggestions for improvement included adding more diverse minigames, such as puzzles or coding challenges, and more diverse and simpler overworld and dungeon maps. In addition, students expressed interest in localizing the platform to their native languages to improve accessibility.

5.4 Discussion

The results indicate that Gamify-IT had a positive impact on student motivation in our CS1 programming course, but it also left room for further improvement. The statistically significant difference in self-assessed mid-term motivation levels between the cohorts of Course 1 and Course 2 reinforces the value of integrating gamified tools like Gamify-IT into educational settings. However, since student motivation is influenced by many factors and not just one tool, it is important to consider the explicit feedback on Gamify-IT itself.

The feedback from the survey provides valuable insights into both the strengths and areas for improvement of Gamify-IT. Students appreciated the engaging and innovative learning approach, the game design and visuals, and the way the minigames gave them a sense of accomplishment. However, the feedback also highlights challenges such as technical issues and a need for more varied game content. With an average rating of 5.50 on a 10-point scale, students rated the usefulness of Gamify-IT as rather moderate and indicated with an average of 5.27 that the game did not motivate them quite more to learn programming concepts. Overall, Gamify-IT has had a great impact on the CS1 programming course, but it still needs improvement to sustain student motivation.

6 Threats to Validity

In this section, we discuss potential threats to the validity of our study. First, we discuss internal validity, then external validity, and finally construct validity according to Runeson and Höst [19].

Internal validity: One potential threat to internal validity arises from differences in the teaching approaches between Course 1 (without Gamify-IT) and Course 2 (with Gamify-IT). While the curriculum and learning objectives were identical, variations in instructor delivery, classroom dynamics, or external influences, such as the availability of additional learning resources, may have affected the results. Nevertheless, in our opinion, the course and curriculum are held in both semesters the same way, making the results comparable. In addition, students in both courses may differ in their initial motivation and willingness to learn, and students in Course 2 may be more enthusiastic players of general, non-educational games. Nevertheless, considering that the course has around 500 students this semester, we think the average student characteristics are quite balanced. Another threat arises from the timing of the surveys that might not capture fluctuations in motivation during critical periods, such as submission deadlines. However, we conducted the surveys in Course 1 and Course 2 at the same time of the course, thus ensuring comparability with the course conditions. It is also possible that students who are already more motivated are also more willing to take part in the surveys, which results in less coverage of unmotivated students in the surveys. However, we have tried encouraging all students to participate by distributing the surveys in class and online in the discussion forum.

External validity: While the study included many participants across two cohorts, all students were enrolled in the same CS1 programming course (in two different semesters). This limits the applicability of the results to other contexts, such as advanced programming courses, different educational institutions, or diverse cultural settings. Nevertheless, we consider that different contexts in programming education face similar motivational challenges. Therefore, Gamify-IT could be used as an educational game to boost student motivation and yield similar results.

Construct validity: A key construct in this study is the self-assessed level of motivation, which was assessed using a 10-point scale. Their reliance on subjective self-assessments introduces potential measurement bias. Students' interpretations of motivation may differ, leading to response variability. However, by using identical surveys in both cohorts and given the relatively large cohort sizes, we ensured consistency in measuring motivation levels. We assumed that the subjective motivation assessments were overall balanced. Another threat is that not all the study findings could be ascertained. Of the about 500 course participants in Course 2, only 101 actually played Gamify-IT, and only 42 students in total completed the survey on their experiences. For this, we tried to gather additional data from multiple sources, such as the initial and mid-term surveys, to validate findings through cross-referencing.

7 Lessons Learned

We have gathered some lessons learned that provide insights into the use of GBL in programming education. They are valuable for advancing research on GBL in programming education and for lecturers who want to incorporate an educational game.

The key insights comprise: (1) An educational game highly depends on its design and technical stability. The user experience is extremely important as students are supposed to learn with the game and will do less or not at all if the game is not appealing. We had some reported technical issues with Gamify-IT, which led to increased student frustration and decreased student motivation. (2) An educational game must explicitly promote the learning objectives of a programming course. If the students see no benefit in playing the game because it does not improve or appeal to them, they will not play it. (3) An educational game must appeal to all students to enable greater motivation. Gamify-IT so far has a limited number of minigames and game mechanics for the player types "Socializers", "Philanthropists", and "Disruptors", which must be considered to find broader acceptance. This emphasizes the need for an iterative design that relies on student feedback on minigames and game mechanics to promote GBL in programming education. Including features that address different motivational drivers of students, such as collaborative games or coding competitions, can therefore increase the appeal and effectiveness of the game.

8 Conclusion and Future Work

This paper presents an approach for integrating game-based learning (GBL) into programming education. The objective was to increase students' motivation to learn programming concepts. We enhanced Gamify-IT with novel features, such as various minigames and game mechanics, designed to adapt to Marczewski's HEXAD player types. We then used the revised Gamify-IT platform in the CS1 programming course at the University of Stuttgart and made it available to students for four consecutive weeks to learn through playing. Of about 500 students in the course, 101 students tried out the game on a voluntary basis. By comparing two cohorts of the same course in different semesters (one with Gamify-IT and one without Gamify-IT), we investigated the platform's impact on student motivation. We therefore carried out initial and mid-term surveys to determine changes of students' self-assessed motivation. Afterward, we surveyed the cohort that used Gamify-IT about their experiences, such as its usefulness and motivational impact.

The answer to the first research question (*RQ1*) can be derived from the study results. Despite the moderate motivational impact reported in the survey capturing the Gamify-IT experience, the mid-term surveys showed a statistically significant higher level of motivation for the cohort using Gamify-IT. Therefore, *RQ1* can be answered positively. However, further studies are needed to validate the effect in other programming courses. The second research question (*RQ2*) can be answered by examining the survey results on experiences with Gamify-IT. Students appreciated the engaging and innovative learning approach, game design, visuals, and how the minigames gave them a sense of accomplishment. Nevertheless, technical issues need to be addressed, and expanding the variety of minigames could appeal to more students.

Future work will evaluate Gamify-IT in different contexts involving CS1 courses at other universities. Further iterations will refine the minigames and game mechanics to sustain student motivation. We also want to establish a more adaptive approach of gamification and personalized feedback considering the diverse backgrounds and individual competence levels of the students [12].

References

- [1] Adnan Ahmad, Furkh Zeshan, Muhammad Salman Khan, Rutab Marriam, Amjad Ali, and Alia Samreen. 2020. The Impact of Gamification on Learning Outcomes of Computer Science Majors. 20, 2, Article 16 (April 2020), 25 pages. doi:10.1145/3383456
- [2] Manal M. Alhammad and Ana M. Moreno. 2018. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software* 141 (2018), 131–150. doi:10.1016/j.jss.2018.03.065
- [3] Richard A Bartle. 2004. Designing virtual worlds. *New Riders* (2004).
- [4] Mauricio R. de A. Souza, Lucas Veado, Renata Teles Moreira, Eduardo Figueiredo, and Heitor Costa. 2018. A systematic mapping study on game-related methods for software engineering education. *Information and Software Technology* 95 (2018), 201–218. doi:10.1016/j.infsof.2017.09.014
- [5] Nejoed Elteгани and Laurie Butgereit. 2015. Attributes of students engagement in fundamental programming learning. In *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*. 101–106. doi:10.1109/ICCNEEE.2015.7381438
- [6] Anabela Gomes and António José Mendes. 2007. An environment to improve programming education. In *Proceedings of the 2007 International Conference on Computer Systems and Technologies (Bulgaria) (CompSysTech '07)*. Association for Computing Machinery, New York, NY, USA, Article 88, 6 pages. doi:10.1145/1330598.1330691
- [7] Orit Hazzan, Tami Lapidot, and Noa Ragonis. 2020. *Guide to teaching computer science*. Springer.
- [8] CoderPad Inc. 2012. CodinGame. (2012). <https://www.codingame.com/> Accessed = 2025-01-18.
- [9] CodeCombat Inc. 2013. CodeCombat. (2013). <https://codecombat.com/> Accessed = 2025-01-18.
- [10] Andrzej Marczewski. 2015. User types. *Even ninja monkeys like to play: Gamification, game thinking and motivational design* 1 (2015), 65–80.
- [11] Anuradha Mathrani, Shelly Christian, and Agate Ponder-Sutton. 2016. PlayIT: Game Based Learning Approach for Teaching Programming Concepts. *Journal of Educational Technology & Society* 19, 2 (2016), 5–17. <http://www.jstor.org/stable/jeductechsoci.19.2.5>
- [12] Niklas Meißner. 2024. MEITREX - Gamified and Adaptive Intelligent Tutoring in Software Engineering Education. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24)*. doi:10.1145/3639478.3639804
- [13] Niklas Meißner, Paul Bredl, Sandro Speth, and Steffen Becker. 2025. Enhancing Motivation in Software Engineering Education through Gamified Agile Project-based Learning. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*. Association for Computing Machinery, 835–846. doi:10.1145/3696630.3727241
- [14] Niklas Meißner, Nadine Nicole Koch, Sandro Speth, Uwe Breitenbücher, and Steffen Becker. 2024. Unveiling Hurdles in Software Engineering Education: The Role of Learning Management Systems. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '24)*. 242–252. doi:10.1145/3639474.3640060
- [15] Niklas Meißner, Sandro Speth, and Steffen Becker. 2024. Automated Programming Exercise Generation in the Era of Large Language Models. In *2024 36th International Conference on Software Engineering Education and Training (CSEET)*. 1–5. doi:10.1109/CSEET62301.2024.10662984
- [16] Niklas Meißner, Sandro Speth, Steffen Becker, and Uwe Breitenbücher. 2025. Empirical Study and Vision for a Holistic Serious Game Platform for Software Engineering Education. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*. Association for Computing Machinery, 800–805. doi:10.1145/3696630.3727237
- [17] Niklas Meißner, Sandro Speth, Julian Kieslinger, and Steffen Becker. 2024. EvalQuiz - LLM-based Automated Generation of Self-Assessment Quizzes in Software Engineering Education. In *Software Engineering im Unterricht der Hochschulen 2024*. GI e.V., Bonn, 53–64. doi:10.18420/seuh2024_04
- [18] Rick Raymer. 2011. Gamification: Using Game Mechanics to Enhance eLearning. *ELearn* 2011, 9, Article 3 (Sept. 2011). doi:10.1145/2025356.2031772
- [19] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14 (2009), 131–164.
- [20] Sandro Speth, Leon Hofmeister, Steffen Becker, and Uwe Breitenbücher. 2023. Gamify-IT - A Web-Based Gaming Platform for Software Engineering Education. In *2023 IEEE/ACM 5th International Workshop on Software Engineering Education for the Next Generation (SEENG)*. doi:10.1109/SEENG59157.2023.00009
- [21] Sandro Speth, Niklas Meißner, and Steffen Becker. 2024. ChatGPT's Aptitude in Utilizing UML Diagrams for Software Engineering Exercise Generation. In *2024 36th International Conference on Software Engineering Education and Training (CSEET)*. 1–5. doi:10.1109/CSEET62301.2024.10663027
- [22] Danny Yaroslavski. 2014. How does Lightbot teach programming. Retrieved January 29 (2014), 2016.