# From Boring to Boarding: Transforming Refactoring Education with Game-Based Learning

Wajdi Aljedaani
Oakland University

Anwar Ghammam
Oakland University

Mohamed Wiem Mkaouer
University of Michigan-Flint

Marouane Kessentini
University of Michigan-Flint

## Abstract

Software Refactoring, a critical skill in software development, involves reorganizing code without altering its functionality. Despite its importance, many developers find refactoring complex and risky, often hesitating to adopt tools designed to assist in this process. To bridge this gap in this paper, we investigate the impact of gamification on the refactoring process and the usability of existing refactoring tools. Our research introduces RefGame, an innovative game-based tool that aims to enhance the learning experience in identifying and applying refactoring techniques. We used RefGame in an exploratory study that involved 322 computer science students. Then, we collected the feedback provided by these students via a survey. Our findings provide insight into the potential of gamification to make software refactoring education more accessible and effective, addressing the current challenges in teaching and learning refactoring techniques.

## CCS Concepts

• **Software and its engineering → Software evolution**; **Maintaining software**; **Software maintenance tools**; **Development frameworks and environments**.

## Keywords

Refactoring, Gamification, Software Engineering Education, Game-based Learning

**ACM Reference Format:**
Wajdi Aljedaani, Anwar Ghammam, Mohamed Wiem Mkaouer, and Marouane Kessentini. 2024. From Boring to Boarding: Transforming Refactoring Education with Game-Based Learning. In *2024 ACM/IEEE 8th International Workshop on Games and Software Engineering (GAS '24 ), April 14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3643658.3643924

## 1 Introduction

Software refactoring, i.e., improving code's internal design without changing its external behavior, is considered an important and useful quality-assurance technique to reduce a system's maintenance costs by improving its internal quality [1], but it is difficult to acquire for software developers. Due to the perceived difficulties and risks of performing it, refactoring is often avoided in practice [2–4]. Even though textbooks with rules and best practices are available, they can barely mediate the skills for actually reviewing larger code bases for refactoring opportunities or performing non-trivial refactorings. Still, it has received little attention from software engineering education and training so far [5].

Despite the availability of powerful tools within integrated development environments (IDEs) such as Eclipse and ReSharper and a wealth of online documentation [6], there is a reluctance to adopt these resources due to their complexity and steep learning curves [7]. This reluctance is compounded by the risk that automated refactoring tools can engender an overreliance that may detract from the foundational understanding of refactoring principles [8].

Traditional educational methods in software refactoring face the challenge of bridging the gap between theoretical knowledge and practical application. We contend that a lack of motivation and hands-on experience hinders the adoption of refactoring practices and tools. To counteract this, we advocate for the incorporation of gamification into refactoring education. Educational games can enhance practical skills and motivation by creating an engaging and interactive learning environment [9].

In recent years, several approaches for serious gaming and gamification in programming and software engineering have been proposed [10–14] enabling a focus on specific conceptual areas while offering lower entry barriers and the advantages of interactive learning experiences. In turn, in the field of software refactoring, so far, only a few gaming approaches can be identified [15–18]. However, the above studies are not ideally suited for novice programmers. These games do not provide a gradual, step-by-step introduction to refactoring concepts. Furthermore, most of them are not open-source or do not provide the essential artifacts necessary for a comprehensive learning experience. This gap highlights a need for more beginner-friendly and accessible educational tools in software refactoring.

To refine the pedagogical strategies for teaching refactoring and code smells, we have introduced RefGame, an innovative game-based educational tool. Comprised of three main interactive games, RefGame guides students through identifying different refactoring techniques. Players engage with code examples or definitions to identify refactoring types, applying their theoretical knowledge to practical scenarios within the game's framework. The tool encourages learning by scoring developers for correctly identifying refactorings and fostering a competitive spirit through a leaderboard.

*GAS '24 , April 14, 2024, Lisbon, Portugal*
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0561-8/24/04.
https://doi.org/10.1145/3643658.3643924

Our study seeks to complement existing research on the adoption of refactoring tools, providing new insights into the potential of gamification in software development education. The driving research question for our investigation is:

**Does game-based tool enhance students' learning experience in software refactoring?**

The contributions of this paper are as follows:

- We develop a creative gamified tool that helps novice developers or students learn more about refactoring in a combination of educational content with enjoyable gaming.
- We report the results of the survey from 322 students.
- We report how students receive a gamified refactoring education tool compared to the conventional educational approaches in terms of different factors (e.g., challenges, focus, social interaction, fun, relevance, and learning perceptions)
- We report the potential improvements suggested by the survey participants.

The rest of this paper is organized as follows. section 2 presents related studies and clarifies the research gaps the current study aims to fill. section 3 provides a detailed description of the tool, its objectives, motivation, learning context, design, and elements. section 4 and section 5 summarize the experimental setup we conducted as well as our survey results. Furthermore, section 6 discusses the threats to our methodology. Finally, the conclusion of this work is in section 7.

## 2 Related Works

This section discusses the fundamental background pertaining to conventional methods employed in the refactoring education process, as outlined in subsection 2.1. Subsequently, an examination of approaches that incorporate games in the context of education software refactoring, along with their inherent limitations, is presented in subsection 2.2.

### 2.1 Software Refactoring Education

In the field of software engineering education, conventional instructional methods to teach software refactoring have traditionally focused on providing a theoretical foundation, as evidenced by previous work [19–24]. Common approaches involve lectures, textbook study, and structured practical exercises, where students manually refactor code snippets to reinforce theoretical knowledge. In particular, Sripada et al. [19] explored the application of refactoring in a second-year course in software engineering, observing positive impacts on code quality. Demeyer et al. [20] introduced basic refactoring principles through an example of LAN simulation, revealing insights into tool limitations in supporting the entire refactoring process. This prior research underscores the need for effective instructional strategies in software refactoring education.

In a separate study, Huchard et al. [25] implemented a Formal Concept Analysis (FCA) based approach with master students, revealing challenges in data selection and students using the approach more as an analytical guide than as a turn-key solution. These studies contribute diverse perspectives to exploring software engineering concepts in education. Bucchiarone et al. [13] present PapyGame, a gamified version of a robust modeling environment (Papyrus) that aims to improve the learner's motivation, make the learning process an enjoyable experience, and boost learning outcomes. Yigitbas et al. [14] present GaMoVR, a VR-based and gamified learning environment that gives students learning about UML modeling an interactive and fun learning experience.

Although their importance in teaching refactoring is significant, this conventional mode of education is not without its limitations. It tends to be theoretical and may fail to fully engage students or bridge the gap between conceptual knowledge and its practical application in real-world scenarios. Traditional methods often do not simulate the dynamic and complex nature of actual software development environments, potentially leaving students ill-prepared for the nuances and challenges of professional software refactoring.

### 2.2 Software Refactoring Education Gamification

The adoption of gamification in refactoring education represents a significant shift in how software development skills are taught and learned [15–18] Agrahari et al., in their paper [15], propose a desktop game named Refactor4Green to teach code smells and refactoring to novice programmers. The core idea of the game is to introduce code smells with refactoring choices through the theme of a green environment. Haendler et al. [16] present REFACTORY, a non-digital multiplayer card game to learn the principles of software refactoring without development-related complexities. In their game, the players simulate a software development team confronted with bad code smells. The players learn to combine refactoring techniques to remove smells and to balance the costs and value of refactoring. Santos et al. [17] created CleanGame, a gamified tool that focuses on one crucial element of the refactoring curriculum: the identification of code smells by conducting an experiment with 18 participants to investigate the efficacy of gamification in the context of strengthening training after completion. Their findings indicate that, on average, participants were able to detect twice the number of code smells when using a gamified technique for learning reinforcement compared to a non-gamified approach.

Baars et al. [18] propose CodeArena, which is an extension to the popular 3D sandbox game called Minecraft. CodeArena converts patterns in a codebase that are considered harmful to monsters in Minecraft, which can then be fought to improve the codebase. In this way, the developer can gradually improve the quality of the code while learning about code quality engagingly.

These games, while innovative in their approach to teaching code smells and refactoring techniques, face significant limitations. First, these games do not provide a gradual, step-by-step introduction to refactoring concepts. Furthermore, most of them are not accessible or do not provide the essential artifacts necessary for a comprehensive learning experience, such as source code and instructional materials, which hinders other educators from adapting or enhancing these games for diverse educational needs. This lack of availability and adaptability significantly undermines the utility and applicability of these games in software refactoring education. This gap highlights a need for more beginner-friendly and accessible educational tools in software refactoring.

## 3 RefGame

To answer our research question, we present RefGame, our gaming tool for refactoring education. Specifically, we outline the goals of the games, provide a comprehensive explanation of the game design in relation to fundamental game concepts and mechanics, and ultimately elucidate the sequence of game play and exemplary variations of the game.

## 3.1 Objectives

This innovative gaming tool is designed to revolutionize the way students learn about code refactoring. It aims to blend the excitement of gaming with the serious task of learning how to refactor functionally sound code. The objective of RefGame is threefold: (1) to enhance the educational experience, incentivize players to acquire further knowledge on software refactoring and develop crucial refactoring skills, (2) to enable performance tracking through a personalized dashboard, (3) and foster a sense of community and competition among learners.

## 3.2 Motivation

The impetus for creating this tool arises from the necessity to make the process of learning about code refactoring more captivating and applicable. Gamification is employed to enhance the level of enjoyment and involvement in response to the potential lack of interest associated with traditional teaching techniques. The aesthetic aspects [26] addressed by the game are narrative (role as software developers/students), challenge (identify types of refactoring and match them with appropriate definitions or examples), and fellowship (competing players in a social framework, for example, students from the same class). From the perspective of gamer's types, [27], the set of games in the tool addresses primarily games that aim to rank and win (score sheets) and achievers that aim to achieve status or goals (e.g., by collecting as many correct answers and getting higher scores). Group-oriented variants also address socializers since players in the same group are competing against each other to get higher scores.

## 3.3 Learning Context

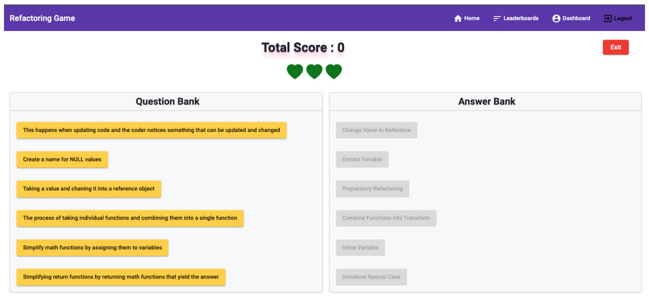
The game can be viewed as a supplementary component to various educational and instructional activities and settings, which are explored in section 2. The game employs a concepts-first approach to introduce the practice of software refactoring. This resource caters to students and practitioners involved in software projects who have limited experience in the area of software refactoring. Specifically, the game caters to individuals who lack programming expertise or have a stronger inclination toward project management. For instance, the tool could be employed within the context of a university course focused on software design, maintenance, or agile development. Given the identified competencies, a learning path focused on developing those competencies may be described. In this path, the games serve as an initial guide. From there, learners can be guided towards higher levels of proficiency.

## 3.4 Tool Design

The scope of our tool design is defined by the question: How to design games for teaching software refactoring concepts and enhance the learning experience in identifying and applying refactoring techniques? For this purpose, our tool design was divided into three main components: Game Design (as the interaction environment), Software Refactoring (as the technical domain), as well as competition Environment Management (since the games aim at mediating certain competitive environments to users).

**3.4.1 Games Design:** Our tool design is diverse, featuring different types of games such as matching, racing, and snake games, each focusing on different aspects of refactoring. Our tool design is diverse, featuring different types of games such as matching, racing, and snake games, each focusing on different aspects of refactoring. Each game comprises the basic game components, such as actions performed by the player and the rules, the run-time behavior of the game, including feedback mechanisms and user interaction. The gameplays are designed to be both challenging and educational, encouraging repeated play and continuous improvement.

**Matching Game:** The game interface is depicted in Figure 1. The matching game facilitates students' acquisition of the ability to discern various refactoring types and associate them with suitable definitions or code examples. When the player successfully aligns three items in a row, their score grows, allowing them to ascend the rankings on the leaderboards. If the player successfully matches all of the questions without depleting all of their lives, they will successfully finish the game. Answering all of the questions will result in an additional bonus being added to your total score.



This happens when updating code and the coder notices something that can be updated and changed

Create a name for NULL values

Taking a value and chaning it into a reference object

The process of taking individual functions and combining them into a single function

Simplify math functions by assigning them to variables

Simplifying return functions by returning math functions that yield the answer

Change Value to Reference

Extract Variable

Preparatory Refactoring

Combine Functions into Transform

Inline Variable

Introduce Special Case

**Figure 1: Matching Game Interface**

**Racing Game:** The racing game combines the excitement of a race with the challenge of answering refactoring questions. The game interface is depicted in Figure 2. The game is controlled using arrow keys, allowing players to skillfully dodge and weave through dense, ever-changing traffic. As the game progresses, the challenge intensifies, with the car's speed incrementally increasing, demanding greater focus and faster reactions. Interspersed throughout this high-octane environment are critical refactoring questions. Answering the refactoring-related questions correctly provides a strategic advantage by temporarily slowing down the oncoming traffic, offering a brief respite in the midst of the thrilling race. Additionally, the game is peppered with various power-ups: the Health Power-up grants an extra hit, allowing players to withstand an additional collision. The Multiplier Power-up doubles the score for ten seconds, adding a layer of strategy to score maximization, and the Lightning Power-up permanently increases the car's speed, upping the ante of the gameplay. These power-ups not only aid in navigating the game's challenges but also serve as opportunities to engage with and answer refactoring questions, reinforcing the player's understanding of code refactoring. This racing game is more than just a test of driving prowess; it is a dynamic and interactive platform that merges the thrill of a racing game with the educational pursuit of learning code refactoring, making it an innovative tool in the realm of educational gaming.

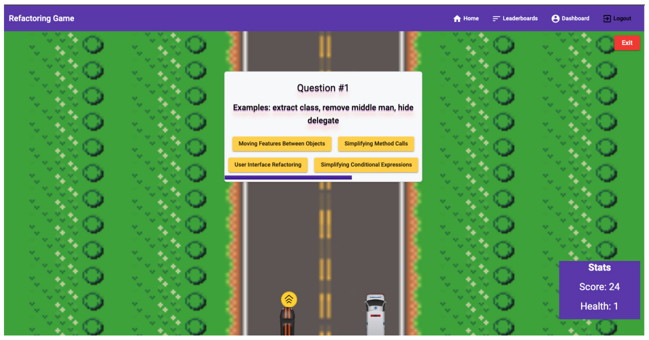

Refactoring Game
Home
Leaderboards
Dashboard
Logout
Exit
Question #1
Examples: extract class, remove middle man, hide delegate
Moving Features Between Objects
Simplifying Method Calls
User Interface Refactoring
Simplifying Conditional Expressions
Stats
Score: 24
Health: 1

Figure 2: Racing Game Interface

**Snake Game:** The Snake game interface is shown in Figure 3. Not only do players have to guide a growing snake to eat things on the screen in this new version of Snake, but they are also taking an educational trip through the world of code rewriting. The arrow keys are used to control the game and direct the snake to "eat" code egg fragments. Each egg is an idea or example of reworking. When a player eats an egg, they are asked to match the refactoring description or example with the right type of refactoring. The player can choose from the choices on the left and right sides of the screen as they see this matching game. Picking the right match not only raises the player's score but also helps them learn about different refactoring methods and principles. But if the user makes the wrong choice, you lose scores. The educational challenge is boosted by the way the classic game Snake is played: the game stops if the player runs out of score points, gets three questions wrong, or the snake hits itself. This game cleverly mixes the fun and addicting gameplay of Snake with an educational twist. It makes learning about code refactoring fun and interactive. It is like real life, where developers have to find and use the right refactoring techniques to keep and improve code quality. Players are encouraged to think quickly and correctly.

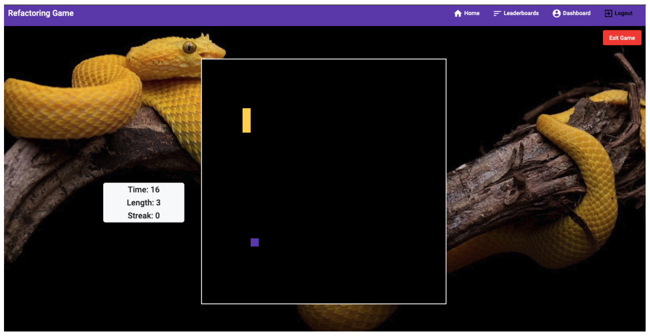


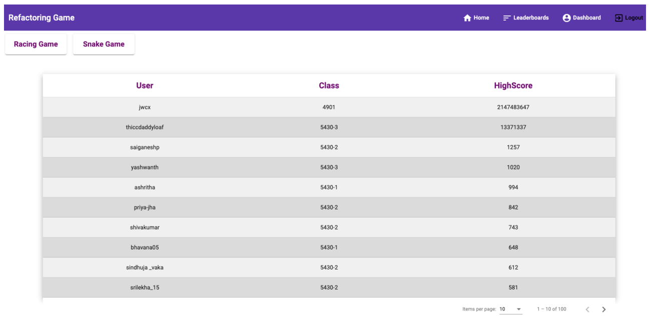
Figure 3: Snake Game Interface

## 3.5 Software Refactoring Activities

Our refactoring game is designed with a dual-activity structure. In each round, players are presented with a series of refactoring definitions and code snippets. The primary challenge is to accurately match each of these elements with their corresponding refactoring type. This setup ensures a comprehensive engagement with the concepts of refactoring, enhancing both understanding and skill in identifying various refactoring types in different coding scenarios.

## 3.6 competition Environment Management

**Leaderboard & Dashboard:** The leaderboard fosters a competitive environment, ranking students based on their performance in various games. This ranking system not only motivates students to improve their skills but also fosters a sense of community as they see how they fare against their peers. Recognition on the leaderboard can be a powerful motivator, encouraging continued engagement and effort. On the other hand, the personalized dashboard offers a more introspective view, allowing students to track their individual progress. It displays detailed analytics, such as total correct answers, high scores, and games played, along with a unique feature that tracks 'technical debts' or areas where the student frequently struggles. This tailored feedback enables students to identify specific areas for improvement and guides them to relevant resources for further learning. Together, the leaderboard and dashboard create a holistic learning environment, blending competitive spirit with personalized educational guidance, thereby enhancing both the engagement and effectiveness of the learning process.



| User | Class | HighScore |
|---|---|---|
| jwcx | 4901 | 2147483647 |
| thiccdaddyloaf | 5430-3 | 13371337 |
| saiganeshp | 5430-2 | 1257 |
| yashwanth | 5430-3 | 1020 |
| ashritha | 5430-1 | 994 |
| priya-jha | 5430-2 | 842 |
| shivakumar | 5430-2 | 743 |
| bhavana05 | 5430-1 | 648 |
| sindhuja _vaka | 5430-2 | 612 |
| srilekha_15 | 5430-2 | 581 |

Items per page: 10 1 – 10 of 100

Figure 4: Leaderboard Interface

## 4 Experimental Setup

In this section, we conduct a survey to address our research question, aiming to assess students' perspectives, level of satisfaction, and general disposition toward our gamified refactoring education strategy. Before broadly deploying the survey, a pilot test was conducted with a cohort of 13 computer science students. These students were selected based on their enrollment in software engineering courses and familiarity with basic refactoring concepts. The pilot study served a dual purpose: it not only evaluated the questionnaire's validity in terms of clarity, objectivity, and accuracy but also examined the participants' interactions with the gamified refactoring tool itself. Feedback was collected on both the tool and the questionnaire to ensure a comprehensive understanding of the student's experiences. Most participants (9 out of 13) confirmed the clarity of the questions, with suggestions for rephrasing three specific questions for better clarity. Some questions were reworded to eliminate potential bias, enhancing objectivity. The consistency of the responses, aligned with the participants' educational backgrounds, suggested the accuracy of the questionnaire in capturing their perspectives. Insights from the pilot study were instrumental in refining both the questionnaire and the tool. The final survey, enriched by this feedback, was then developed into an online form using Google Forms to efficiently gather and distribute information [28].

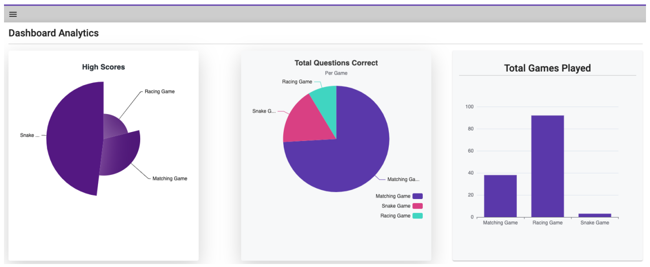


**Figure 5: Dashboard Interface**

The studied survey question is publicly available [1]. The questionnaire comprises 17 questions, 12 multiple-choice questions, and 5 open-ended questions. The questionnaire was semi-structured, with open and closed questions, allowing the respondents to express their views and explain them without restriction. We also made some questions optional so that participants were not required to answer them. The survey is divided into five parts: Q1 to Q6 are aimed at gathering background information from participants; Q7 to Q20 are related to the participants' experience while using RefGame, the relevance of the game contents, and their satisfaction with it. It is worth mentioning that questions Q17 to Q20 are about the participant's feedback and potential improvements for our tool. Once the tool was deployed online and the survey was ready, we conveniently sampled 322 participants interested in participating in the study.

All of these participants were volunteers. In our experiment, we did not specify a set duration for participants to use the gamified refactoring tool. Instead, they were allowed to spend as much time as they felt necessary to familiarize themselves with it and understand its features. This approach was taken to accommodate individual differences in learning and engagement rates, ensuring that each participant could interact with the tool at their own pace and according to their own comfort level. The procedure was structured as follows: Participants were first introduced to the gamified refactoring tool with basic instructions on its use. They were then free to engage with the game for as long as they deemed appropriate. This flexible approach allowed participants to explore the tool to a degree they felt sufficient to form an opinion or understanding of it.

After their interaction with the tool, participants were invited to complete the questionnaire. We encouraged them to fill out the questionnaire immediately after using the tool to capture their immediate reactions and thoughts. However, we emphasized that the completion of the questionnaire was entirely optional and anonymous. There was no requirement for the participants to complete the game fully before taking the questionnaire, allowing them to provide feedback based on any level of interaction they had with the game, from brief trials to extensive exploration.

[1]https://doi.org/10.5281/zenodo.10534241

### 4.1 Target Users

The survey was completed by a total of 322 participants. The demographic information of the survey participants is illustrated in Figure 6. The majority of participants consist of master's students (77%) and bachelor's students (20.8%). Furthermore, a total of 305 individuals, representing 94.7% of the participants, registered for a computer science major. Regarding the experience of the participants with programming language development, 14 participants (4.4%) reported having more than 8 years of experience, 37 (11.5%) reported having between 4 to 6 years of experience, and the majority of participants (49.1%) reported having at most two years of experience. Participants also reported engaging in gaming activities. Among them, seven individuals (14.9%) claimed to play games consistently, 26.1% played games very frequently, and 41% played games infrequently. Only six individuals (1.9%) claimed that they never played games.

All 322 participants were recruited through a targeted process focusing on computer science students with minimal exposure to software refactoring but familiarity with gaming. We performed the experiment on the software engineering classes as the primary pool for potential participants. The participation was voluntary and incentivized modestly to ensure a representative sample size. This recruitment strategy was integral to the study's aim of evaluating the effectiveness of gamification in teaching complex software refactoring concepts to beginners in the field. The selection of this particular group was intentional, as it was predicted that their previous experience in gaming would make them more open to a gamified learning tool such as RefGame. Our objective was to evaluate the efficacy of gamification in teaching intricate and intimidating software refactoring ideas by targeting students in the initial phases of their software development education. This cohort represents a pivotal section in the educational pipeline, where the implementation of cutting-edge pedagogical approaches can significantly influence their comprehension and proficiency in the field of software engineering. The feedback they provide is highly valuable because it offers insights into how beginners perceive and engage with refactoring concepts in a gamified context. This allows us to assess the potential of such approaches to make software refactoring education more accessible and engaging for newcomers to the field.

## 5 Results

In this section, we provide the results of our survey to answer our research question. We report how students receive our refactoring education tool in terms of different factors (e.g., challenges, focus, social interaction, fun, relevance, and learning perceptions) and the potential improvements suggested by the survey participants.

**Does game-based tool enhance the learning experience of students in software refactoring?**

Figure 8, Figure 7, and Figure 9 show the answers related to the participants' experiences using RefGame. The questions are grouped according to the following factors **usablity**, **Ease**, **challenge**, **satisfaction**, **social interaction**, **focus**, **relevance**, and **learning Perception**. For Q4 and Q5, the results present the percentage of participants' responses ranging from "Extremely Easy"

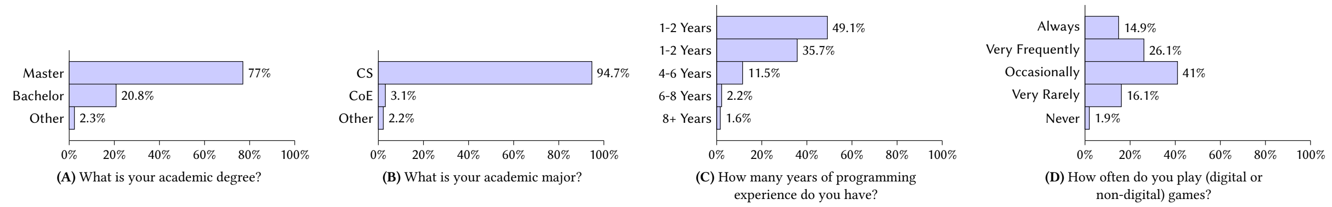


**Figure 6: Demographic information of the Survey participants.**

to "Extremely Difficult". For each of the other questions (Q6-Q15), the results present the percentage of participants' responses ranging from "Strongly Disagree" to "Strongly Agree". In the following, we discuss each aspect of the factoring individually.

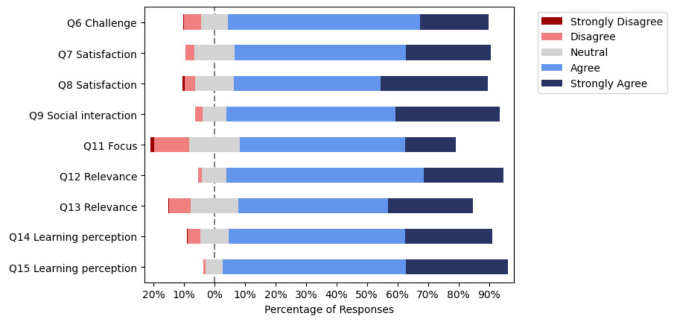


**Figure 7: Participants' experience with RefGame**

**User Interface (*Usability*):** Figure 8 highlights the survey results regarding question Q4, which asks participants about the usability of our tool interfaces. The analysis reveals insightful trends. A negligible 0.3% of participants found the game extremely difficult, indicating that the interface and mechanics are highly accessible to the vast majority. However, 6.8% who found it difficult suggested that there are specific elements that might need refinement for a better user experience. The 30.1% with a neutral stance could imply that the game is neither challenging nor straightforward for them, possibly indicating a need for more engaging or intuitive aspects for users less familiar with gaming or refactoring concepts. Significantly, 49.7% found the game easy, and 13% extremely easy, demonstrating that for a substantial majority, the game's design is effective, user-friendly, and the learning curve well-adjusted. This is illustrated by one of the participant's responses:

> *"The game is very informative, and at the same time, it is fun playing the game and answering the questions at the same time. It is very easy to access and also easy to understand. It is a very simple interface that will help everyone gain knowledge about refactoring. In general, it is an excellent tool."*

Overall, these findings suggest that RefGame is generally well-received. However, the criticism of individuals who found it less accessible is extremely helpful in directing future improvements with the goal of making the game more universally captivating and instructional.

**Ease of learning (Difficulty):**: Figure 9 highlights the Survey results regarding question *Q5*, which asks participants about the difficulty of performing refactoring concepts learning activities

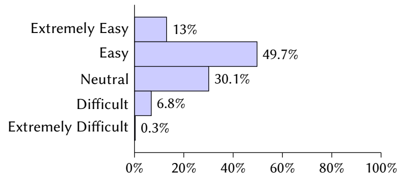


**Figure 8: Game interface usability (Q4).**

with the support of RefGame. The results would seem to suggest that the participants found the activity more challenging to perform without RefGame. A small fraction, 0.6%, found it extremely difficult, suggesting that for a very few, the game might be too challenging or not intuitive enough.

Those who found it difficult, representing 9%, indicate areas where the game could be improved to smooth the learning process. The largest group, 37.1%, felt neutral, which could imply that while they did not find the game particularly hard, it was not exceptionally easy either, pointing towards a moderate learning curve. A significant 45.8% found it easy, suggesting that for nearly half of the participants, the game successfully facilitated learning refactoring concepts. However, only 7.5% found it extremely easy, indicating that while the game is accessible, it still presents a reasonable challenge for most students. Overall, these results suggest that RefGame is effective for a majority of students, but there is room for improvement, especially in making complex concepts more approachable for those who find it difficult or neutral.

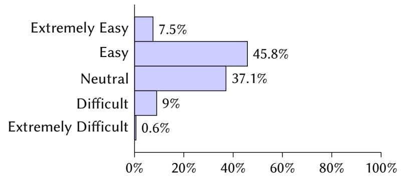


**Figure 9: Ease of refactoring learning (Q5).**

**Challenges:** 63% of the participants agree, and an additional 22.4% strongly agree. The result reflects a predominantly positive reception, suggesting that the game adeptly maintains an engaging level of challenge. However, it should be noted that a small fraction, comprising 5.3%, expressed some level of dissatisfaction, signaling potential areas for further refinement of the game. One participant noted that:

> *"The game was so interesting and challenging. I had fun and learned all the refactoring concepts interactively."*

This response aligns with the majority view and suggests that the game effectively integrates education with engagement, a core aspect of its design. Another participant noted that:

*"The refactoring game is a fun and competitive tool designed to teach students how to improve their code through gamified learning."*

These reflections from the students provide a narrative context to the statistical evidence, painting a picture of a learning tool that is both effective and enjoyable. The figure Figure 7 and the testimonials together suggest that the game is well-received for its educational value and its ability to captivate and motivate students. The positive responses dominate the feedback, indicating the game's success in fulfilling its educational intent while also being engaging enough to maintain student interest.

**Satisfaction:** Out of 322 responses, a significant majority of 56.2% reported feeling 'Very Satisfied,' while 27.6% felt 'Satisfied' with their sense of achievement. The combined total of 83.8% indicates a strong positive response, suggesting that the game is successful in providing a rewarding experience. Meanwhile, a small minority of 13.4% remained 'Neutral,' and an even smaller segment expressed dissatisfaction, with 2.2% feeling 'Unsatisfied' and 0.6% 'Very Unsatisfied.' These figures may point to areas where the game could potentially be improved to enhance student satisfaction. One of the participants said that:

*"This game is interesting. I could game and learn at the same time. Made us productive even while playing."*

This response, presumably from a contented student, correlates with the large proportion of the participants who expressed a high level of satisfaction. It underscores the dual value of the game as both an engaging activity and a productive educational tool, highlighting the effectiveness of gamified learning in imparting knowledge while also delivering a sense of enjoyment and achievement. The positive reception suggests that the game successfully leverages the intrinsic motivation of play to facilitate learning and productivity.

**Social Interaction & Fun:** A majority, 55.6%, strongly agree, while 34.2% agree, cumulatively indicating that 89.8% of the 322 respondents perceive the game as competitive. A small percentage of respondents are undecided (7.8%), and an even smaller number disagree (2.2%) or strongly disagree (0.2%), suggesting a general consensus about the competitive nature of the game. One of the participants stated that:

*"I found the website interesting and useful as learning was made into a fun activity, and also, the leaderboard is motivating me to get a higher score with a competitive mindset."*

The mention of the leaderboard highlights the game's ability to incentivize learning through competition, resonating with the substantial majority who acknowledge the game's competitive aspect. One of the participants commented that:

*"It is a competitive game which will allow us to sharpen our knowledge on code refactoring. I especially liked the feature where we can see the statistics of our game and track progress too."*

This feedback underscores the game's educational aim, with the competitive elements serving as a catalyst for learning and self-improvement. The ability to track progress through statistics is particularly appreciated, as it provides tangible metrics for growth and motivation. One of the participants noted that:

*"RefGame was fun and educative. It was a bit new to me to be honest and I loved it."*

This sentiment, while not directly addressing competition, emphasizes the game's enjoyable and educational nature, suggesting that even those new to the concept can find it both engaging and informative. In synthesis, the quantitative data from the survey is strongly supported by qualitative testimonials from the students. They collectively affirm the game's success in utilizing competitive elements to enhance the learning experience, making the acquisition of refactoring skills an interactive, motivating, and dynamic process.

**Focus & Relevant:** With a prominent 54.3% strongly agreeing and 16.5% agreeing that the game kept them engaged and focused to the extent that they lost track of time and their surroundings. The combined agreement of 70.8% suggests a high level of immersive engagement offered by the game. Conversely, 11.5% of the students remained neutral, while a minority of 16.5% disagreed (11.5% disagreed and 5% strongly disagreed) with the sentiment of the game being engaging. One of the participants explained that:

*"The RefGame to be an engaging and insightful way to learn and apply refactoring techniques and also it made me more focus on the questions types and next modules of the game."*

This comment reflects the strong agreement seen in the chart, emphasizing the game's role in enhancing concentration and providing depth to the learning process. Another participant said that:

*"My experience with playing this game is quite good. When playing these games, I lost track of time while gaining more knowledge about refactoring."*

This student's experience aligns with the majority view on the chart and highlights the game's educational benefits alongside its engaging nature. Another participant stated that:

*"RefGame was a really fun and interactive way to learn refactoring. Refactoring has always been something that has been boring to try and learn, but this has made it more fun and interesting to be able to learn."*

This feedback addresses a critical aspect of the game's design—its ability to transform a traditionally mundane subject into an interactive and enjoyable activity, which is corroborated by the high percentage of students who reported strong engagement.

**Learning Perception:** a substantial majority of 57.8% 'Strongly Agree' and 28.6% 'Agree' that RefGame significantly contributed to their learning, resulting in an overwhelming 86.4% positive response. Only a small portion of 9.3% remain 'Undecided'. At the same time, the combined 'Disagree' and 'Strongly Disagree' categories make up a minimal 4.3%, suggesting that the majority of users found the game to be an effective learning tool. One of the participants mentioned that:

*"From my perspective, I find the game to be an excellent and engaging tool to learn software refactoring, as it combines fun gameplay with practical coding challenges."*

The student's view highlights the balance of enjoyment and practical learning of the game, which is reflected in the high percentage of positive responses in Figure Figure 7. A participant stated that:

> *"It was learning through gaming. Based on my perspective, I love learning through games, which makes me more of a learner than usual learning. By playing all the games, I loved the Racing game. And the leaderboard makes me fire up to play. Overall, I loved this game so much."*

This feedback not only supports the positive responses to the result but also points to the motivational aspect of the game, particularly the leaderboard, which stimulates a competitive spirit conducive to learning. Lastly, the enthusiasm for the efficiency of gamified learning is echoed by another student:

> *"It is very good efficient to learn things easier. By playing games, we can learn concepts also."*

This perspective emphasizes the efficiency and effectiveness of learning through the medium of a game.

## 6 Threats to Validity

In this section, we present our threat to the validity of our study. RefGame facilitates students in acquiring proficiency in recognizing and categorizing various types of refactoring, with a focus on refining their ability to discern code-restructuring possibilities rather than code smells. To mitigate this limitation, we incorporate a code snippet that enables students to determine the applicable type of refactoring. Another potential threat is that our study sample was exclusively comprised of students. Consequently, the generalizability of RefGame's engagement with practitioners proficient in refactoring remains uncertain. As our tool is designed for students in the novice or introductory stages of refactoring, it is plausible that its impact may be less pronounced among those with a substantial background in refactoring. To explore this further, our next phase involves incorporating code snippets and implementing refactoring types to address code smells or enhance code generated by our tool.

## 7 Conclusion

In summary, the RefGame was viewed as a creative and effective tool for learning refactoring techniques. Its combination of educational content with enjoyable gaming elements made it a popular choice among the participants. While there were some technical issues and areas for improvement, the overall reception of the game was positive, and many students found it a beneficial addition to their learning resources.

## Acknowledgments

We extend our sincere appreciation to Ian Tharp, Sakchham Sangroula, Benjamin Martinez, Cambry King, and Colin Fisher for their dedicated contributions to RefGame. Their efforts have significantly contributed to completing the first version of the tool.

## References

[1] Martin Fowler. *Refactoring*. Addison-Wesley Professional, 2018.

[2] Zadia Codabux and Byron Williams. Managing technical debt: An industrial case study. In *2013 4th International Workshop on Managing Technical Debt (MTD)*, pages 8–15. IEEE, 2013.

[3] Erik Ammerlaan, Wim Veninga, and Andy Zaidman. Old habits die hard: Why refactoring for understandability does not give immediate benefits. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 504–507. IEEE, 2015.

[4] Leon Moonen, Arie Van Deursen, Andy Zaidman, Magiel Bruntink, T Mens, and S Demeyer. On the interplay between software testing and evolution and its effect on program comprehension., 2008.

[5] Thorsten Haendler, Gustaf Neumann, and Fiodor Smirnov. An interactive tutoring system for training software refactoring. *Instructor*, 1:4, 2019.

[6] Emerson Murphy-Hill and Andrew P. Black. Refactoring tools: Fitness for purpose. *IEEE Software*, 25(5):38–44, 2008.

[7] Emerson Murphy-Hill and Andrew P Black. Breaking the barriers to successful refactoring: observations and tools for extract method. In *Proceedings of the 30th international conference on Software engineering*, pages 421–430, 2008.

[8] Mauricio A Saca. Refactoring improving the design of existing code. In *2017 IEEE 37th Central America and Panama Convention (CONCAPAN XXXVII)*, pages 1–3. IEEE, 2017.

[9] Michael A Miljanovic and Jeremy S Bradbury. A review of serious games for programming. In *Serious Games: 4th Joint International Conference, JCSG 2018, Darmstadt, Germany, November 7-8, 2018, Proceedings 4*, pages 204–216. Springer, 2018.

[10] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work?–a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*, pages 3025–3034. Ieee, 2014.

[11] Mehmet Kosa, Murat Yilmaz, Rory O'Connor, and Paul Clarke. Software engineering education and games: a systematic literature review. *Journal of Universal Computer Science*, 22(12):1558–1574, 2016.

[12] Oscar Pedreira, Félix García, Nieves Brisaboa, and Mario Piattini. Gamification in software engineering–a systematic mapping. *Information and software technology*, 57:157–168, 2015.

[13] Antonio Bucchiarone, Maxime Savary-Leblanc, Xavier Le Pallec, Jean-Michel Bruel, Antonio Cicchetti, Jordi Cabot, and Sébastien Gérard. Gamifying model-based engineering: the papygame tool. *Science of Computer Programming*, page 102974, 2023.

[14] Enes Yigitbas, Maximilian Schmidt, Antonio Bucchiarone, Sebastian Gottschalk, and Gregor Engels. Gamovr: Gamification-based uml learning environment in virtual reality. *Science of Computer Programming*, 231:103029, 2024.

[15] Vartika Agrahari and Sridhar Chimalakonda. Refactor4green: A game for novice programmers to learn code smells. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, pages 324–325, 2020.

[16] Thorsten Haendler. A card game for learning software-refactoring principles. In *GamiLearn*, 2019.

[17] Hoyama Maria dos Santos, Vinicius HS Durelli, Maurício Souza, Eduardo Figueiredo, Lucas Timoteo da Silva, and Rafael S Durelli. Cleangame: Gamifying the identification of code smells. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 437–446, 2019.

[18] Simon Baars and Sander Meester. Codearena: Inspecting and improving code quality metrics in java using minecraft. In *Proceedings of the 2019 International Conference on Technical Debt (Tool Demos). IEEE*, 2019.

[19] Sai Krishna Sripada and Y Raghu Reddy. Code comprehension activities in undergraduate software engineering course-a case study. In *2015 24th Australasian Software Engineering Conference*, pages 68–77. IEEE, 2015.

[20] Serge Demeyer, Filip Van Rysselberghe, Tudor Girba, Jacek Ratzinger, Radu Marinescu, Tom Mens, Bart Du Bois, Dirk Janssens, Stéphane Ducasse, Michele Lanza, et al. The lan-simulation: a refactoring teaching example. In *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*, pages 123–131. IEEE, 2005.

[21] Jagadeesh Nandigam, Venkat N Gudivada, and Abdelwahab Hamou-Lhadj. Learning software engineering principles using open source software. In *2008 38th Annual Frontiers in Education Conference*, pages S3H–18. IEEE, 2008.

[22] Shamsa Abid, Hamid Abdul Basit, and Naveed Arshad. Reflections on teaching refactoring: A tale of two projects. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 225–230, 2015.

[23] Sara Stoecklin, Suzanne Smith, and Catharina Serino. Teaching students to build well formed object-oriented methods through refactoring. *ACM SIGCSE Bulletin*, 39(1):145–149, 2007.

[24] Tony Clear. Disciplined design practices: a role for refactoring in software engineering? *ACM SIGCSE Bulletin*, 37(4):15–16, 2005.

[25] Marianne Huchard. Full application of the extract interface refactoring: conceptual structures in the hands of master students. In *Proceedings of the 1st International Workshop on Software Refactoring*, pages 33–40, 2016.

[26] Hunicke Robin, L Marc, and Zubek Robert. A formal approach to game design and game research. *GDC. San Jose*, 2004.

[27] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.

[28] Kishor Kumar and Lokesha Naik. How to create an online survey using google forms. *International Journal of Library and Information Studies*, 6(3):118–126, 2016.