

A RESEARCH-BASED GAMIFICATION SOLUTION FOR TEACHING ARRAYS TO NOVICE APPLIED SCIENCE STUDENTS

by

Tine Vervloet

Student number: 852327742

Course code: IM9703

Thesis committee: Dr. Ebrahim Rahimi (chairman), Open University
Dr. Ir. Clara Maathuis (supervisor), Open University

1. INTRODUCTION

In recent years, the demand for digitally skilled professionals has risen sharply, driven by the growing integration of software systems in nearly all aspects of society and industry [Cha and Kim, 2018; Crowne, 2002]. As a result, programming education has become a crucial component in applied science curricula. However, programming remains a challenging subject for many students. Introductory programming courses consistently show high failure and dropout rates, with difficulties often arising from abstract and technical concepts that are hard to visualize and apply in practice [Mellado and Cubillos, 2024; Yoo et al., 2022].

One of the most persistent obstacles for novice programmers is the concept of arrays [Rosminah et al., 2012]. Although they are among the most fundamental data structures in computer science, arrays pose significant cognitive and instructional challenges. Students frequently struggle with abstract memory models, indexing errors, and integrating arrays with loops and conditions [Bosse and Gerosa, 2017]. These challenges not only hinder learning progression but also lead to demotivation and disengagement, potentially affecting students' overall performance in programming-related courses [Jenkins, 2002].

At the same time, innovative pedagogical approaches are being explored to better support novice learners. For instance, gamification, defined as the integration of game-like elements into non-game contexts, has shown promising results in increasing motivation and enhancing conceptual understanding in educational settings [Mellado and Cubillos, 2024]. Gamification helps maintain high motivation, especially when students face challenges during their initial programming experience or with specific topics. At the same time, it aids in reducing the cognitive load of programming topics, and helps to achieve a better understanding of the required concepts. In programming education, gamification has the potential to provide immediate feedback, reinforce key patterns, and support active experimentation in a low-stakes environment. However, while many gamified tools exist, few are specifically designed to address the difficulties related to arrays in applied science contexts. Moreover, existing solutions are often not well integrated into regular teaching practices, limiting their long-term impact.

This research aims to bridge this gap by investigating how a digital gamification solution can be designed to support array learning for novice programming students and provide a sustainable integration into a teaching environment. Drawing on both literature and practical teaching experience, the study explores the core learning obstacles associated with arrays, reviews existing gamified tools and strategies, and proposes a tailored educational solution aligned with the specific needs of applied science education.

This graduation assignment contributes to the existing literature of programming education in two ways. First, it provides a structured analysis of relevant research across three domains: programming education challenges, available educational tools, and gamified approaches specifically targeting arrays. Secondly, it translates this theoretical foundation into practice by designing a tailored gamified solution that supports novice students in understanding arrays more effectively. This prototype aims to bridge the gap between research insights and classroom application, offering a practical tool grounded in both pedagogy and cognitive learning principles.

The following report is structured as follows. Section 2 presents an overview of related work, including challenges in programming education, difficulties specific to arrays, and the role of gamification and pedagogical strategies. Section 3 provides a problem analysis,

based on both literature and personal teaching experience. Section 4 outlines the research design, including research questions, methodology, and validation strategy. Finally, Section 5 describes the planning and risk analysis for the implementation of the graduation project.

2. RELATED WORK

Before addressing the specific focus of this graduation assignment, i.e. arrays and the related pedagogical challenges, it is important to explore the broader context of programming education. Therefore, this Section 2 first outlines the general difficulties faced by novice programmers and explains the persistent failure rates in introductory programming courses. It then zooms in on the programming challenges related to arrays, reviewing their abstract nature and the common errors students face. Next, gamification is introduced as a promising approach to increase student engagement and understanding, followed by a discussion of current pedagogical strategies such as PRIMM and Parsons problems. The subsequent subsections review the current existing tools and games aimed at array learning, as well as development frameworks for educational game design. Finally, a critical reflection highlights the research gap, questioning why conceptual difficulties persist despite the many tools that are already available.

2.1. GENERAL PROBLEMS WITH PROGRAMMING

With the growth of the IT industry, the increasing number of software-focused startups, and software becoming an integral part of multiple disciplines, the demand for skilled software engineers has risen significantly [Cha and Kim, 2018; Crowne, 2002]. Educational programs should address this need by providing students with a solid foundation to become industry-ready programmers. However, programming courses continue to face high failure rates, with approximately 30% of students failing introductory courses, a percentage significantly higher than in other scientific disciplines [Mellado and Cubillos, 2024; Yoo et al., 2022].

The primary difficulties students face, within introductory programming courses, include syntax errors, understanding variable types, function parameter passing, and memory access using pointers [Bosse et al., 2019; Islam et al., 2019; Kadar et al., 2021; Mhashi and Alakeel, 2013]. Programming is widely considered complex and is included in the seven grand challenges of computing education [Rosminah et al., 2012]. Despite extensive research and the implementation of multiple pedagogical strategies, students continue to struggle with fundamental concepts, raising concerns about the effectiveness of current teaching methods.

2.2. PROGRAMMING CHALLENGES SPECIFIC TO ARRAYS

Among the many challenges in programming education mentioned in the previous section, arrays remain one of the most difficult topics for students to grasp. In this work, one-dimensional arrays are defined as ordered collections of elements of the same data type, stored in contiguous memory locations and accessed via a single index. They are among the most basic data structures in programming and are often used to introduce concepts such as indexing, iteration, and memory allocation. The scope in this work focuses on their use in introductory programming courses, particularly how students understand and manipulate array elements in problem-solving tasks.

Given this definition of arrays, multiple studies consistently rank them among the top

five most challenging programming concepts [Assaf et al., 2022; Bosse et al., 2019; Kadar et al., 2021; Rosminah et al., 2012]. This is further supported by research highlighting that arrays are particularly difficult due to their abstract memory representation and common issues related to indexing, memory allocation, and multidimensional manipulation [Bosse and Gerosa, 2017]. Data indicates that 53% of students perceive arrays as a hard-to-learn concept, creating a significant barrier to succeed in programming education [Assaf et al., 2022].

Furthermore, a lack of knowledge and difficult conceptualization about zero-based indexing and out-of-bounds errors frequently lead to runtime and logic errors for novice programming students. These errors further complicate students' learning experiences [Rigby et al., 2020]. These listed challenges highlight the importance of pedagogical strategies to reduce the complexity of the fundamental understanding of an array to guide students through the beginning of their programming journey.

2.3. GAMIFICATION AND ITS ROLE IN PROGRAMMING EDUCATION

Gamification is the use of game-like elements in non-game contexts to motivate and influence behavior [Deterding et al., 2011]. It involves incorporating features such as rewards, progress tracking, challenges, competition, and storytelling to make tasks more engaging and enjoyable. Its core goal is to tap into the psychological drivers that make games compelling, such as achievement, curiosity, and social interaction, to improve participation and outcomes.

Gamification has emerged as an innovative approach to improving student engagement and comprehension in programming education. Studies indicate that incorporating game elements into learning environments can significantly boost motivation and retention rates [Mellado and Cubillos, 2024]. These solutions can vary from developing an immersive 3D game to implementing a leaderboard system where students earn badges. By providing immediate feedback, giving step-by-step guidance to the solution, and reducing the cognitive load by structuring the materials, these games help students quickly identify and correct errors, reinforcing their problem-solving skills. Meanwhile, the ongoing challenges keep students engaged, leading to a high level of participation and retention. This means that their extrinsic motivation can be addressed with elements like points and badges to encourage students participation, whereas a better understanding of the problem can give a boost to the students' intrinsic motivation [Jenkins, 2002].

Examples already exist for different programming concepts, e.g. Gamified Parsons Problems for databases¹, and Parsons LogEx for logical equivalence², and their usage shows promising results based on participants' responses [van den Berg et al., 2024]. These projects demonstrate how puzzle-based approaches such as Parsons problems, discussed further in Section 2.4, help reduce cognitive load by shifting attention away from syntax toward program logic. By reordering provided code fragments, students are encouraged to reason about the program's semantics, aiding in a deeper conceptual understanding of the underlying programming constructs.

Gamification can be a highly effective approach for teaching arrays to novice programmers. By integrating real-world elements into a game environment and visualizing array structures, it lowers the barrier to understanding fundamental programming concepts. The

¹Gamified Parsons Problems

²Parsons LogEx

use of clear, interactive representations makes abstract ideas more concrete and relatable. In addition, guiding students through algorithms step by step, much like a visual debugger, helps address common challenges such as indexing and iteration [Rosminah et al., 2012]. Immediate feedback from the game environment highlights mistakes as they happen, eliminating the need to wait until the entire algorithm runs to identify errors. This step-by-step learning process supports a deeper and more structured understanding of array programming.

While gamification shows promise, its long-term effectiveness remains uncertain. Many serious games fail to scale effectively within academic curricula due to their niche focus and lack of adaptability to different teaching environments. Furthermore, implementing game-based learning requires additional resources, which may not always be available to educators [Pears et al., 2007].

2.4. PEDAGOGICAL STRATEGIES

Numerous pedagogical strategies have been developed to address the challenges of learning programming. A well-known framework for teaching complex concepts is the 4C/ID (Four-Component Instructional Design) model, which is widely used for designing instruction around complex skills and task integration [Merrienboer, 1997]. However, because of its focus on advanced skill development, it is less suited to the needs of novice programmers, who benefit more from simpler approaches. Another commonly used framework for classifying educational objectives is Bloom's Taxonomy, which organizes learning goals into distinct cognitive levels [Krathwohl, 2002]. In the context of computer science education, these objectives can be tailored based on the work of [Kadar et al., 2021].

- **LO1 (Remembering):** Students recall basic programming structures such as array declarations.
- **LO2 (Applying & Analyzing):** Students implement and manipulate arrays within program segments.
- **LO3 (Creating):** Students develop solutions that integrate arrays with functions and control structures.

Bloom's taxonomy is a general framework that can be applied across various subjects. In the context of programming education, however, more specialized frameworks have been developed. One such framework is PRIMM³, which outlines five stages to support novice programmers: Predict, Run, Investigate, Modify, and Make. The first two stages correspond to Bloom's lower-order learning objective (LO1 – remembering and understanding), the third and fourth align with LO2 (applying and analyzing), and the final stage maps to the higher-order objective (LO3 – creating) [Sentance and Waite, 2017]. Putting PRIMM into practice, students begin by predicting what a given piece of code will do, then run it to test their prediction. They investigate the code through tracing, annotation, and debugging, followed by modifying it to alter its functionality. Finally, they create their own program. This structured, step-by-step approach promotes active learning and deeper conceptual understanding. Studies have shown that PRIMM yields promising results in programming

³PRIMM - Support for teaching programming in school

education, not only for learning arrays but for foundational programming skills in general [Sentance et al., 2019].

In addition to PRIMM, several other pedagogical strategies support novice programmers. As discussed in Section 2.3, Parsons problems, which are a form of puzzle-based learning, require students to reorder shuffled code fragments [Du et al., 2020; van den Berg et al., 2024]. This reduces cognitive load by removing syntax concerns, allowing learners to focus on logic and structure. This strategy primarily aligns with Bloom’s LO2. Moreover, gap-filled programming tasks similarly present partially completed code, encouraging students to reason about missing components without being hindered by syntax, also supporting LO2 [Lo et al., 2014]. Pair programming promotes collaboration, thinking-out-loud, and peer learning, making it a strategy that is able to span all levels of Bloom’s taxonomy [Sobral, 2020]. Lastly, the concept of semantic waves, which alternates between abstract theory and concrete examples, promotes deeper conceptual understanding and supports LO1 [Waite and Sentance, 2021]. Together, these strategies offer a solid foundation for designing instructional tools that reduce cognitive barriers and improve student engagement with array-based programming challenges.

2.5. EXISTING SOLUTIONS AND VISUALIZATION TOOLS

To help students master various learning objectives, gamification and visualization tools have been developed to improve their understanding of complex programming concepts. These are not limited to issues related to array programming, but include a wide variety of programming challenges as pointed out in [Vahldick et al., 2014]. These tools cover a wide range of fundamental topics, such as PointerViz for visualizing pointers [Venigalla et al., 2020], VPL (Virtual Programming Lab), a Moodle plug-in with automated assessment [del Pino et al., 2012], PythonTutor for displaying data structures [Guo, 2013], and Examplelore for assisting with API usage [Glassman et al., 2018].

Apart from the aforementioned tools that cover a broad range of topics, multiple aids have been developed to help students understand the concept of arrays better. An overview is generated and the gamification approaches and addressed problems are listed in Table 1. Two examples are discussed below in more detail. A first one is *Wu’s Castle* where students can learn arrays by allowing them to visually manipulate array elements through interactive gameplay, where they program magical creatures to modify arrays representing snowmen formations [Eagle and Barnes, 2008]. A second example is the game *Space Geeks* that provides an interactive, science-fiction-themed environment where students must write Java code to progress through different levels, reinforcing array concepts through hands-on problem-solving [Assaf et al., 2022].

Research shows that students using animation-based learning tools perform better than those who rely solely on textbooks [Ibrahim Osman and M. Elmusharaf, 2014]. However, despite the availability of these tools, widespread adoption and integration into standard curricula remain a challenge [Linden and Lederman, 2011].

2.6. OPEN-SOURCE DEVELOPMENT TOOLS

Various frameworks for gamified educational tools are described in the literature and supported by open-source libraries. These range from simple web-based applications to complex 3D role-playing games, as shown in Table 1. Among the more advanced options,

Reference	Name	Gamification	Problem(*)	Language
[Eagle and Barnes, 2008]	Wu's Castle	2D RPG	LC, OOB, IDX, NL, DBG, AVT, LV	C++
[Assaf et al., 2022]	Space Geeks	3D serious game	DEC, INIT, IDX, LC, OBO, LVX	Java
[Elvina et al., 2018]	PythonTutor	2D platformer	DEC, INIT, IDX, EAM, LIT, DBG	Python
[Galles]	USFCA QueueArray Visualizer	Web-based visualizer	IDX, EIR, CBB	None required for users (purely visual tool)
[Karram, 2021]	CodeCombat	2D browser game	DEC, INIT, IDX, EAM, LIT, DBG	Python and JavaScript
[Chang et al., 2012]	DreamCoders	2D RPG	DEC, INIT, IDX, EAM, LIT, DBG	C#
[Jagadeesha et al., 2023]	DSAScratch	Programming environment	DEC, INIT, IDX, EAM, LIT, DBG	Block-based visual programming
[Debabi and Bensebaa, 2016]	AlgoGame	2D RPG	DEC, INIT, IDX, EAM, LIT, DBG	Not specified; focuses on algorithmic concepts
[Mourato and Santos, 2024]	Paddle	Web-based visualizer	DEC, INIT, IDX, EAM, LIT, DBG	Java

Table 1: Overview of array-focused serious games and tools. (*) Legend: LC: Loop Control; OOB: Out-of-Bounds Errors; IDX: Indexing (including off-by-one errors); NL: Nested Loops; DBG: Debugging Logic; AVT: Array Value Tracking; LV: Loop Execution Visualization; DEC: Declaration of Arrays; INIT: Initialization of Arrays; EAM: Element Access and Modification; LIT: Loop-Based Iteration; OBO: Off-By-One Errors (explicit); LVX: Lack of Visualization; EIR: Element Insertion and Removal; CBB: Circular Buffer Behavior).

Pygame⁴ (for 2D games) and Unity⁵ with C# and WebGL export (for 3D or visually rich environments) are well-established platforms. They offer a high degree of flexibility and can provide immersive user experiences while addressing learning objectives. However, it is important to consider that these environments often require significant effort in areas such as graphics, animation, and interface design. This may divert time and resources away from the core educational goals of the project.

An alternative approach is to focus on a puzzle-based game structure, which can be implemented more efficiently using web-based development environments. These platforms are typically more straightforward to set up and allow for rapid development aligned with pedagogical principles. Frameworks such as React⁶, Elm⁷, and Vue.js⁸ provide lightweight, component-based architectures ideal for creating interactive, logic-driven educational tools like Parsons problems or card-based exercises. Their open-source nature, extensive UI libraries, and existing templates support fast prototyping and easy customization. In addition, browser-based deployment ensures broad accessibility without requiring students to install additional software, making these tools particularly suitable for use in educational settings.

2.7. RESEARCH GAP AND CRITICAL REFLECTION: ARE WE REINVENTING THE WHEEL?

Despite the numerous existing strategies and tools aimed at improving programming education, students continue to struggle with fundamental concepts such as arrays. This persistent issue raises an important question: *If so many solutions exist, yet the problem remains, is developing another tool truly the best approach?* Studies indicate that many educational tools and gamification methods are not widely adopted due to limitations such

⁴Pygame

⁵Unity

⁶React

⁷Elm

⁸Vue.js

as lack of flexibility, high workload for educators, and institutional resistance [Pears et al., 2007]. Additionally, many tools developed within research projects are not maintained long-term, leading to fragmentation and limited real-world impact [Naps et al., 2002].

This research aims to explore existing solutions for teaching arrays and evaluate their effectiveness in supporting novice programming students at an applied science university. By analyzing the strengths and limitations of these tools, the work will identify the most promising features and build upon them to design our own gamified solution. Rather than reinventing the wheel, the goal is to learn from what already works and develop a practical, engaging, and scalable tool that enhances array comprehension in real-world educational settings.

3. PROBLEM ANALYSIS

Arrays are a fundamental concept in programming, yet they pose significant challenges for novice programmers, especially for those without prior programming experience. Understanding arrays is crucial as they form the basis for more complex data structures and algorithms. From personal experience of teaching programming to electromechanical students who will learn to automate industrial processes, it is important that they fully understand the basic concepts to later program various automated systems such as gantry cranes and pick-and-place robots. However, these students often struggle with grasping the concept, applying correct syntax, and troubleshooting errors related to arrays. These difficulties have been widely discussed in literature as discussed in Section 2 and are further confirmed by personal teaching experience.

3.1. LITERATURE

The observations outlined above are not new. The academic community widely recognizes and acknowledges the challenges associated with teaching programming to novice students. In particular, the use of primitive data structures, such as arrays, has proven to be a fundamental difficulty within computer science education. Section 2 already provided a comprehensive summary of the related work and highlights several key areas where students commonly encounter problems, including understanding the concept of an array, misconceptions about zero-based indexing and out-of-bounds errors, array manipulation, allocating memory, interpreting compiler feedback, and function parameter passing with arrays.

Studies suggest that these challenges stem from cognitive overload, a lack of visualization, and ineffective instructional methods. Research also indicates that structured pedagogical techniques and interactive learning environments could enhance student comprehension.

3.2. PERSONAL EXPERIENCE

Teaching the introductory programming course "Programmeren"⁹ at Thomas More University College has provided direct insight into the challenges students face. This three-credit course, taught in Dutch, covers fundamental programming principles using Visual Basic for Applications (VBA). With a strong emphasis on hands-on learning, students are

⁹Programmeren (B-TM-YT0633)

encouraged to apply theoretical concepts through practical exercises, an approach essential for real-world problem solving.

The curriculum introduces fundamental programming structures, including variable types, decision structures, loops, arrays, subroutines, and functions. Debugging techniques and structured programming methodologies are also covered to develop problem solving skills. However, arrays consistently present a challenge, as they require students to integrate multiple programming concepts. While most students progress smoothly through the early part of the course, arrays often become their first major stumbling block. Those who do not master this topic tend to struggle with programming for the remainder of the course.

A structured test at week 7 of the 12-week course provides further insight into these difficulties. The assessment evaluates students' ability to define arrays of fixed length, use loops to fill them, integrate conditional logic, and compare variable values within arrays. Previous analyses indicate that students often struggle to apply these concepts in practical scenarios. An in-depth examination of the exam results will help identify specific patterns of difficulty.

3.3. THE ARRAY LEARNING GAP

By comparing literature findings with real-world classroom observations, it becomes clear that student difficulties with arrays are consistent across different teaching environments and programming languages. The overlap between documented research and personal teaching insights highlights the need for improved instructional strategies for novice programmers. Despite existing educational tools, misconceptions about indexing, memory management, and iteration structures continue, leading to low success rates in programming courses. Understanding the root causes of these challenges is essential for designing effective teaching methods. Insights from both literature and classroom experience will inform the development of strategies aimed at downsizing these difficulties.

This research seeks to identify the fundamental obstacles students face when learning arrays and to explore the potential of gamification as innovative solutions to enhance comprehension, engagement, and strengthen knowledge retention.

4. RESEARCH

Given the problem analysis presented in Section 3, this research confirms, based on both the author's personal teaching experience and literature review, that novice programming students face significant challenges. While these difficulties arise in all areas of programming, this study specifically focuses on the concept of arrays, which are one of the most fundamental data structures. Existing literature shows that gamified approaches can reduce cognitive load and increase students' intrinsic motivation. Therefore, this research aims to design a gamified learning solution that builds on pedagogical strategies and is useful for first-year students following a programming course in a STEM-focused program at an applied science university. The next section outlines the research design developed to achieve this goal.

4.1. RESEARCH QUESTIONS

To address the core challenge of enhancing students' understanding of arrays, this research is guided by one main research question and four supporting sub-questions. These questions are structured to systematically identify the problems students face, review existing solutions, develop a gamified approach, and assess its educational impact. Together, they provide a coherent framework for conducting a design-oriented study within the context of applied science education. Given the design and development focus of this research, the graduation project adopts the Design Science Research framework [Hevner, 2007] as its overarching research strategy.

How to design a gamified solution to aid novice programming students in understanding arrays?

To provide a well-founded answer to this main question, the following four sub-questions have been formulated:

RQ1: What are the main cognitive and learning challenges that novice applied science students encounter when learning arrays? To clarify the specific learning obstacles associated with arrays, this sub-question draws on both existing literature and practical teaching experience from the main author. It aims to identify the primary challenges students encounter, such as misconceptions about indexing, difficulties with abstract representations, and struggles with syntax and logic. The insights gathered from RQ1 will form the basis for defining the educational and learning requirements of the gamified solution. These requirements will serve as the functional inputs for the design and development of the tool in this software engineering project.

RQ2: What types of tools or strategies, particularly those using gamification, have been developed and are the most promising ones to support array learning in programming education? This question aims to explore the current landscape of educational tools and gamified solutions. It examines how these approaches seek to enhance conceptual understanding and evaluates their strengths, limitations, and relevance to the target group. Building on the comparative Table 1, related categories and instructional architectures are grouped together. Additionally, the question investigates which specific game elements, such as immediate feedback, progression systems, or storytelling, align well with established pedagogical models. This research question thus provides an overview of techniques and features from the current knowledge base that can be used to address the functional requirements, as identified through RQ1, for this software engineering project.

RQ3: How to design and develop a gamification solution for learning arrays using insights from RQ2 to address students' needs and challenges resulted from RQ1? Building on the requirements identified in RQ1 and the techniques and features gathered in RQ2, this question focuses on designing an appropriate gamification solution for teaching arrays to novice programmers. The goal is to move beyond theoretical design by developing a functional application that incorporates the proposed gamification elements. This prototype will serve both as a proof of concept and as a practical

tool for testing with learners.

RQ4: In what ways does the implemented game support novice applied science students in overcoming their difficulties with learning arrays? This fourth sub-question focuses on designing an appropriate evaluation protocol. It includes strategies for measuring learning gains and user experience to assess the effectiveness of the developed solution. The application created in RQ3 will be evaluated using this developed protocol. Key considerations include technological constraints, curricular alignment, teacher and student acceptance, and the long-term sustainability of the solution.

The DSR framework is used to define the listed set of research questions, and mapped onto the different DSR phases as depicted in Figure 1.

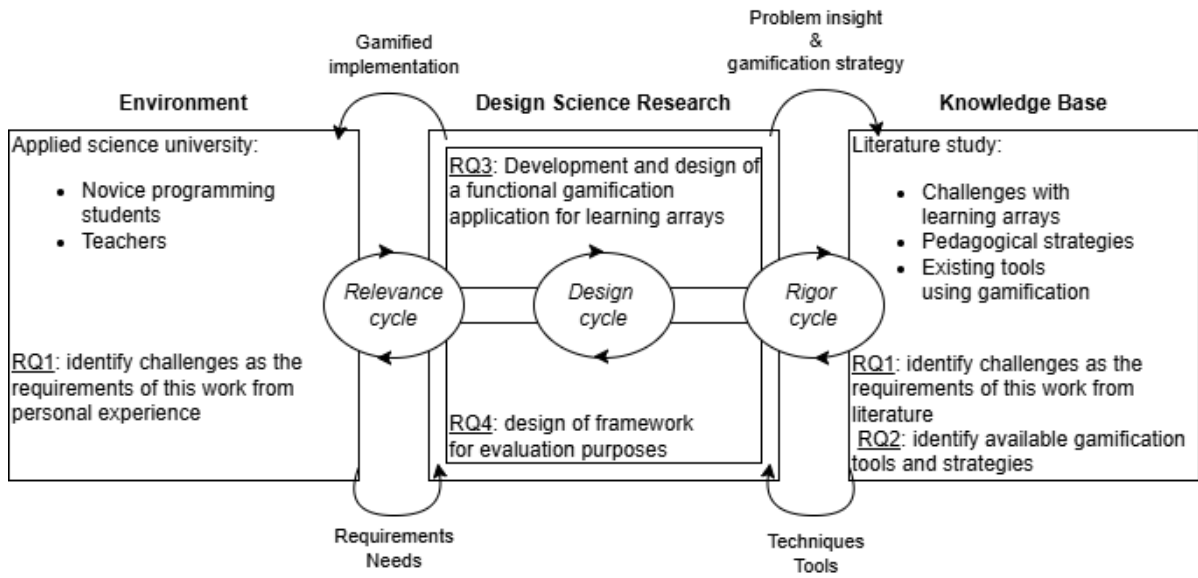


Figure 1: Research Science Diagram mapping the research questions on the chosen research framework (figure inspired by [Hevner, 2007]).

4.2. RESEARCH METHOD

To answer RQ1, two main sources of information will be used. Firstly, a preliminary study has been presented in Section 2. In addition to these initial findings, a more comprehensive and in-depth literature review will be conducted to better understand the needs of novice programming students. Secondly, leveraging the main author's personal teaching experience, targeted interviews and questionnaires will be developed and presented to both students and teachers at an applied science university. Based on these two sources, the primary cognitive and learning challenges will be identified and categorized. The most prominent of these challenges will be addressed throughout this research.

To gather insights during this phase, students (who failed the course during the first exam period) and teachers will be recruited through class announcements and email. Participation will be voluntary, with clear communication on the purpose of the study and the expected involvement. Their input will help identify key learning challenges and refine the gamified solution accordingly. Since this RQ1 and the following RQ4 involve interaction

with students, this graduation assignment will be submitted to the Research Ethics Committee (cETO)¹⁰ to verify whether this scientific research is in accordance with the ethical guidelines.

To address RQ2, a deeper exploration of the available literature will be carried out. Section 2 and Table 1 already provide a preliminary overview of existing tools and gamification solutions. In this phase, the literature study will be expanded to broaden the existing landscape and map these tools to the challenges identified in RQ1. This analysis will offer a clearer understanding of current solutions and allow for a comparison of results, ultimately highlighting the most promising and comprehensive tools and strategies. As shown in previous research [van den Berg et al., 2024], systematically mapping the existing educational challenges to corresponding tools is an important result of RQ2. This provides useful insights for selecting relevant strategies to address the identified challenges from RQ1.

RQ3 focuses on the design and development of a functional application. Based on the findings and interview results from RQ1, the requirements for this application will be outlined first. Then, insights from the knowledge base developed in RQ1 will inform the selection of the most suitable strategy for the application. These findings will guide both the design and development phases. Special attention will be given to ensuring that the application effectively incorporates gamification elements to increase engagement among novice programming students. As outlined in the related work in Section 2.6, suitable open-source development environments are readily available and will be leveraged in this project. To maintain focus on the core educational challenges, particularly those related to learning arrays, the use of complex 2D or 3D visual designs will be avoided. Instead, a web-based environment is preferred for its simplicity and accessibility. The primary development environment for this software engineering project will be Python, while the learning activities will be designed around VBA, the programming language currently used by the main author in classroom instruction. This ensures alignment between the developed tool and the real-world teaching context in which it will be applied.

To answer RQ4, an evaluation protocol will be developed. However, to keep the main focus of this work on the development of the gamified solution, a lightweight evaluation protocol will be designed that primarily focuses on user experience, supported by in-depth interviews with students and teachers. Therefore, the evaluation will use existing validated instruments, including the System Usability Scale (SUS) [Brooke, 1996] to assess usability and the Intrinsic Motivation Inventory (IMI) [Ryan and Deci, 2000] to measure user engagement and motivation. These tools will provide a structured and efficient way to gather feedback without requiring extensive experimental setups. The application developed in RQ3 will be deployed among students, and their results and feedback will be used to assess the effectiveness of the solution using the proposed validation protocol. For consistency, the same group of students and teachers will be approached to participate using the same communication channels as before. This approach ensures continuity and relevant insights into the effectiveness of the solution in a real-world context.

4.3. VALIDATION

To validate the research questions using the chosen methodology, three complementary perspectives will be considered. First, the educational environment will be examined in detail. This includes insights drawn from personal teaching experience and interviews

¹⁰Research Ethics Committee (cETO)

conducted with novice programming students. These findings will be supplemented by interviews with teachers, which will help to summarize and categorize the main challenges faced by learners. This analysis serves to validate the identified problem and to validate the needs from a practical and experiential point of view.

Second, the existing literature will be explored as a foundational source of knowledge. This review will complement the observed challenges and help to confirm the relevance and accuracy of the personal perspective on the problem. Additionally, the literature will provide insights into pedagogical frameworks and existing tools, offering a broader view of the current state of the art. In this way, the problem is validated from a theoretical and academic standpoint.

Finally, the gamified solution developed in this study will be deployed among students. Feedback and data will be gathered through a focus group to evaluate the effectiveness and practicality of the proposed solution, using the SUS and IMI protocols as validation methods. In addition, one-on-one interviews with teachers based on the observed results will provide further insights into how well the solution addresses the key challenges faced by novice programming students to validate the developed application in this work.

5. PLANNING

The graduation assignment will begin in May 2025 and will run over a period of approximately 22 weeks, ending in late September. The first weeks will focus on identifying the cognitive and learning challenges novice applied science students face when learning arrays, addressing RQ1. This involves analyzing exam data, preparing questionnaires and interview protocols, and conducting student interviews before the end of the lecture period (May 22nd). These steps are supported by a targeted literature review and complemented by expert interviews. The outcomes will contribute to an extensive understanding of the key difficulties students experience. Findings from this phase will be finalized by mid-June. Simultaneously, work on the graduation report itself will begin from week 3 onward, ensuring a continuous documentation process. At the same time, the graduation assignment will be presented to the cETO as discussed in Section 4.2 to ensure that all research activities comply with the applicable ethical guidelines. The required documents will be submitted as one of the first subtasks of RQ1.

From week 7, attention will shift to RQ2, which explores tools, strategies, and game features that can support students in the understanding of learning arrays, particularly through gamification. Literature studies on both pedagogical strategies and existing tools will provide insights into promising approaches. These literature studies are followed by an analysis of applicable strategies and game features.

In July, the focus turns to RQ3, centered on the design of a gamified learning solution based on the challenges and opportunities identified in RQ1. This includes defining user needs, selecting a design framework, and making fundamental design choices. Iterative game development begins simultaneously, ensuring the game evolves in response to the pedagogical and technical knowledge base. In parallel, preparation for validation starts by developing review criteria and defining how effectiveness will be assessed.

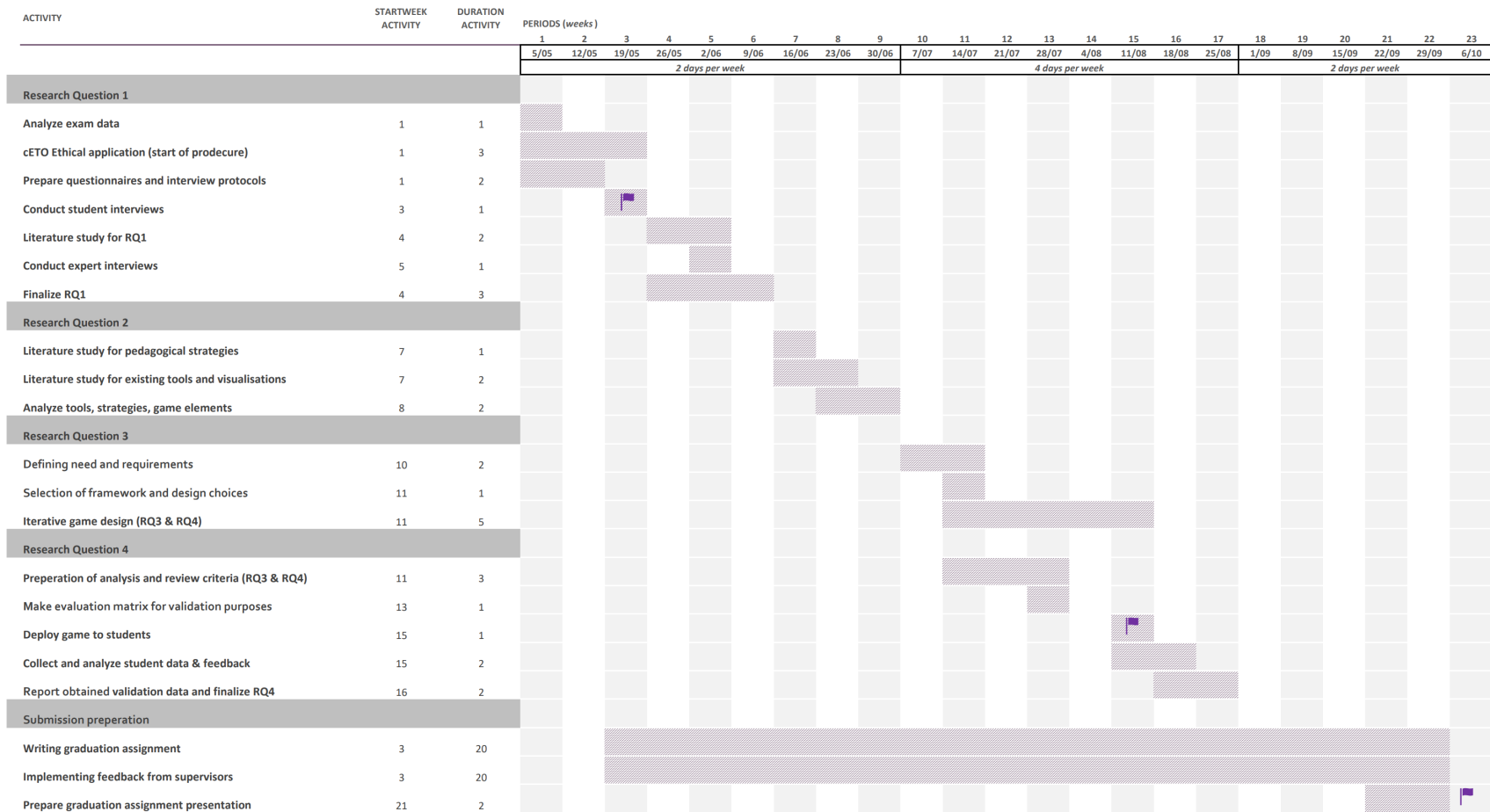


Figure 2: Gantt chart of the planning of the graduation assignment.

By mid-August, validation activities for RQ4 will take place. An evaluation matrix will be constructed as an integral part of RQ4, and the game will be deployed to students preparing for the August 27th re-exam. Their interactions and feedback will be collected and analyzed. These results will be used to assess how effectively the game helps students overcome difficulties with arrays, leading to the final reporting and conclusion of RQ4 in early September.

Throughout September, time will be devoted to finalizing the graduation report, implementing feedback from supervisors, and preparing the presentation. The entire assignment will be completed by the end of September. A complete overview of the listed activities and their corresponding planning is shown graphically in the Gantt chart in Figure 2. Based on the available working time, approximately two days per week in May, June, and September, and three to four days per week in July and August, this plan allows for roughly 416 working hours to be dedicated to the project.

5.1. RISK ANALYSIS

Although this graduation project is supported by a clear structure and a detailed timeline, several risks may arise during its execution. These risks are related to different phases of the research process and are discussed with respect to the research questions and the overall DSR framework. The following overview presents the most prominent risks in order of relevance.

A first risk will be discovered early on in the project. Since scientific research will be conducted on students, the Research Ethics Committee (cETO) will be consulted to verify whether the work is in accordance with the ethical guidelines. The required documents will be submitted as soon as possible to adapt fast if the required approval is not granted. As an alternative, this work will deduct data completely from literature. Regarding the validation part of the gamified solution, professionals and teachers in the field can be a backup validation strategy.

One major risk concerns the validation phase of the developed gamified solution (RQ4). The effectiveness of the application will be assessed through student participation, both in the form of participation and feedback. However, given the timing, close to the re-exam period in late August, student availability may be limited, which could affect the amount of collected data. To mitigate this, the validation phase will focus on short and accessible test sessions and may be supplemented by interviews with instructors or teaching assistants, providing additional perspectives on the educational impact.

A second risk is the limited development time for the game prototype (RQ3). Designing and building a pedagogically sound and technically functional application within the available timeframe poses a challenge. To reduce this risk, the project will prioritize the implementation of core learning components (such as declaration, indexing, and loop control). A tool with limited functionality will be used to ensure that the essential educational value is preserved, even if the application's visual or technical complexity is limited.

Another challenge comes from the possibly limited available literature required for RQ1 and RQ2. It could happen that given the available tools and educational games, only a few focus specifically on array learning. This could limit the direct applicability of existing solutions. To address this, the review will include visualization tools and programming games that target related concepts, such as loops, debugging, or problem decomposition to obtain more gamification features. Additionally, pedagogical frameworks like PRIMM

will be used to generalize and adapt relevant design principles.

The impact of the developed solution on measurable learning outcomes may also be limited (RQ4). Given the relatively short exposure time and small sample size, improvements in test scores could be statistically insignificant. To compensate for this risk, the evaluation strategy will also include qualitative measures such as perceived usefulness and general student feedback. These insights will provide a broader understanding of the tool's potential impact and areas for future work.

In case of unexpected delays, the research questions have been prioritized as follows. Research questions RQ1 and RQ2, targeting student challenges and current solutions, are considered essential. Therefore, they will be completed regardless of external constraints. RQ3 aims to at least obtain a functional prototype with limited functionality. RQ4, which involves the full evaluation of the application, is important but can be addressed through limited pilot testing.

Risk	Description	Counter-measure	Possible alternative
Ethical approval not granted	Ethical approval may be denied	Submit documents as a first step	Use literature data if denied
Limited student availability	Validation coincides with re-exam period	Short test sessions	Interview teachers as backup
Time constraints for development	Limited time for prototype development	Focus on core features build a basic functional version	Deploy game during first semester courses
Limited literature availability for array learning	Few gamified tools for array learning	Broaden search to related concepts and visualization	Focus on gamified elements and strategies from a broad range of other gamified solutions
Impact on learning outcomes	Short exposure may limit measurable impact	Include qualitative feedback for broader insights	Have in-depth discussions with teachers
Unexpected delays	External factors may cause delays	Prioritize RQ1 & RQ2, minimal prototype for RQ3, pilot for RQ4	Graduation deadline can be postponed till middle of first semester

Table 2: Overview of the risks and corresponding mitigations.

BIBLIOGRAPHY

- Abdelbaset Assaf, Mohammed Eshtay, and Lana Issa. Space geeks: A proposed serious game to teach array concept for novice programming students. *Proceedings of the 18th International Conference on Web Information Systems and Technologies*, page 431–438, 2022. doi: 10.5220/0011589200003318. 3, 5, 6
- Yorah Bosse and Marco Aurélio Gerosa. Why is programming so difficult to learn? *ACM SIGSOFT Software Engineering Notes*, 41(6):1–6, Jan 2017. doi: 10.1145/3011286.3011301. 1, 3
- Yorah Bosse, David Redmiles, and Marco A. Gerosa. Pedagogical content for professors of introductory programming courses. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, page 429–435, Jul 2019. doi: 10.1145/3304221.3319776. 2, 3
- John Brooke. *SUS – a quick and dirty usability scale*, pages 189–194. 01 1996. 11
- Kyung-Jin Cha and Yang Sok Kim. Critical success factors for mutual collaboration with suppliers in it outsourcing industry: a case study of a top it outsourcing company in korea. *Enterprise Information Systems*, 12(1):76–95, 2018. doi: 10.1080/17517575.2016.1196734. 1, 2
- Jack Chang, Long Dang, Jebediah Pavleas, Joseph Mccarthy, Kelvin Sung, and Jason Bay. Experience with dream coders: developing a 2d rpg for teaching introductory programming concepts. *Journal of Computing Sciences in Colleges*, 28:227–236, 10 2012. 6
- M. Crowne. Why software product startups fail and what to do about it. evolution of software product development in startup companies. In *IEEE International Engineering Management Conference*, volume 1, pages 338–343 vol.1, 2002. doi: 10.1109/IEMC.2002.1038454. 1, 2
- Wassila Debabi and Tahar Bensebaa. Using serious game to enhance algorithmic learning and teaching. *Je-LKS : Journal of e-Learning and Knowledge Society*, 12:127–140, 01 2016. doi: 10.20368/1971-8829/1125. 6
- Juan Carlos Rodríguez del Pino, Enrique Rubio Royo, and Zenón Hernández Figueroa. A virtual programming lab for moodle with automatic assessment and anti-plagiarism features, 2012. 5
- Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining "gamification". New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308168. 3
- Yuemeng Du, Andrew Luxton-Reilly, and Paul Denny. A review of research on parsons problems. In *Proceedings of the Twenty-Second Australasian Computing Education Conference, ACE'20*, page 195–202, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450376860. doi: 10.1145/3373165.3373187. 5
- Michael Eagle and Tiffany Barnes. Wu's castle: Teaching arrays and loops in a game. *ACM SIGCSE Bulletin*, 40(3):245–249, Jun 2008. doi: 10.1145/1597849.1384337. 5, 6

- Elvina Elvina, Oscar Karnalim, Mewati Ayub, and Maresha Caroline Wijanto. Combining program visualization with programming workspace to assist students for completing programming laboratory task. *Journal of Technology and Science Education*, 8(4):268–280, 2018. ISSN 2013-6374. 6
- David Galles. Queue (array implementaion). URL <https://www.cs.usfca.edu/~galles/visualization/QueueArray.html>. 6
- Elena L. Glassman, Tianyi Zhang, Björn Hartmann, and Miryung Kim. Visualizing api usage examples at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3174154. 5
- Philip J. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 579–584, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318686. doi: 10.1145/2445196.2445368. 5
- Alan Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19, 01 2007. 9, 10
- Waleed Ibrahim Osman and Mudawi M. Elmusharaf. Effectiveness of combining algorithm and program animation: A case study with data structure course. *Issues in Informing Science and Information Technology*, 11:155–168, 2014. doi: 10.28945/1986. 5
- Noman Islam, Ghazala Shafi Sheikh, Ridah Fatima, and Farrukh Alvi. A study of difficulties of students in learning programming. *Journal of Education amp; Social Sciences*, 7(2): 38–46, Oct 2019. doi: 10.20547/jess0721907203. 2
- Ashwin Jagadeesha, Pranathi Rayavaram, Justin Marwad, Sashank Narain, and Claire Seungeun Lee. Integrating data structures and algorithms in k-12 education using block-based programming. In *2023 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–10, 2023. doi: 10.1109/EDUCON54358.2023.10125165. 6
- Tony Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer, 2002. 1, 3
- Rozita Kadar, Syarifah Adilah Mohamed Yusoff, Saiful Nizam Warris, and Mohd Saifulnizam Abu Bakar. Students' assessments in learning programming based on bloom's taxonomy. *Journal of Computing Research and Innovation*, 6(3):13–21, Sep 2021. doi: 10.24191/jcrinn.v6i3.223. 2, 3, 4
- Omar Karram. The role of computer games in teaching object-oriented programming in high schools - code combat as a game approach. *WSEAS TRANSACTIONS ON ADVANCES in ENGINEERING EDUCATION*, 18:37–46, Apr 2021. doi: 10.37394/232010.2021.18.4. 6
- David R. Krathwohl. A revision of bloom's taxonomy: An overview. *Theory Into Practice*, 41(4):212–218, 2002. doi: 10.1207/s15430421tip4104_2. 4

- Tanya Linden and Reeva Lederman. Creating visualizations from multimedia building blocks: A simple approach to teaching programming concepts. *SSRN Electronic Journal*, 2011. doi: 10.2139/ssrn.2559026. 5
- Yu-Jen Lo, Chiung-Chen Lin, Lily Hou, Jane-Dong Wu, Yu-Chi Feng, Te-Chin Chu, and Greg Lee. Reducing novice programmers' cognitive load and improving learning efficiency by using gap-filled programming practice system. In Jarmo Viteli and Marianna Leikomaa, editors, *Proceedings of EdMedia + Innovate Learning 2014*, pages 2221–2226, Tampere, Finland, June 2014. Association for the Advancement of Computing in Education (AACE). 5
- Rafael Mellado and Claudio Cubillos. Gamification improves learning: Experience in a training activity of computer programming in higher education. *Journal of Computer Assisted Learning*, 40(4):1959–1973, May 2024. doi: 10.1111/jcal.13000. 1, 2, 3
- Jeroen J. G. van Merriënboer. *Training Complex Cognitive Skills: A four-component instructional design model for technical training*. Educational Technology Publications, 1997. 4
- Mahmoud Mhashi and Ali Alakeel. Difficulties facing students in learning computer programming skills at. In *Recent Advances in Modern Educational Technologies*, 2013. 2
- Rodrigo Mourato and André L. Santos. Educational Program Visualizations Using Synthesized Execution Information. In André L. Santos and Maria Pinto-Albuquerque, editors, *5th International Computer Programming Education Conference (ICPEC 2024)*, volume 122 of *Open Access Series in Informatics (OASIs)*, pages 10:1–10:8, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-347-8. doi: 10.4230/OASIs.ICPEC.2024.10. 6
- Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, page 131–152, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 9781450374491. doi: 10.1145/960568.782998. 7
- Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '07, page 204–223, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781450378420. doi: 10.1145/1345443.1345441. 4, 7
- Liam Rigby, Paul Denny, and Andrew Luxton-Reilly. A miss is as good as a mile: Off-by-one errors and arrays in an introductory programming course. In *Proceedings of the Twenty-Second Australasian Computing Education Conference, ACE'20*, page 31–38, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450376860. doi: 10.1145/3373165.3373169. 3

- Siti Rosminah, Ahmad Zamzuri Mohamad Ali, and Mohamad Ali. Difficulties in learning programming: Views of students. In *1st International Conference on Current Issues in Education*, 09 2012. doi: 10.13140/2.1.1055.7441. 1, 2, 3, 4
- R. M. Ryan and E. L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55:68–78, 1 2000. doi: 10.1037/0003-066X.55.1.68. 11
- Sue Sentence and Jane Waite. Primm: Exploring pedagogical approaches for teaching text-based programming in school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 113–114, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450354288. doi: 10.1145/3137065.3137084. 4
- Sue Sentence, Jane Waite, and Maria Kallia and. Teaching computer programming with primm: a sociocultural perspective. *Computer Science Education*, 29(2-3):136–176, 2019. doi: 10.1080/08993408.2019.1608781. 5
- Sónia Rolland Sobral. Is pair programing in higher education a good strategy? *International Journal of Information and Education Technology*, 10:911–916, 12 2020. doi: 10.18178/ijiet.2020.10.12.1478. 5
- Adilson Vahldick, Antonio José Mendes, and Maria José Marcelino. A review of games designed to improve introductory computer programming competencies. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–7, 2014. doi: 10.1109/FIE.2014.7044114. 5
- Marten van den Berg, Bastiaan Heeren, and Ebrahim Rahimi. Parsons problems for equivalence proofs in logic. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, Koli Calling '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710384. doi: 10.1145/3699538.3699551. 3, 5, 11
- Akhila Sri Venigalla, Chaitanya S. Lakkundi, and Sridhar Chimalakonda. Pointerviz - towards visualizing pointers for novice programmers. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2020. doi: 10.24251/hicss.2020.015. 5
- Jane Waite and Sue Sentence. *Teaching programming in schools: A review of approaches and strategies*. Raspberry Pi Foundation, 2021. 5
- Sujin Yoo, Changbae Mun, Minjong Cheon, Ook Lee, Yoonki Rhee, and Hyodong Ha. A study on the factors affecting academic achievement in the non-face-to-face class environment due to covid-19: Focusing on computer liberal arts education class. *Sustainability*, 14(11):6547, May 2022. doi: 10.3390/su14116547. 1, 2