Game-Driven Pedagogical Framework: Enhancing Student Engagement, Enjoyment, and Motivation in Programming Education

Rumaizah Che Md Nor, Abdul Hadi Abdul Talip*, Zubaidah Bohari
Universiti Teknologi MARA Cawangan Sarawak, Kampus Samarahan 2, Malaysia
**Corresponding author: Abdul Hadi bin Abdul Talip (adie0951@uitm.edu.my)*

Received: 8 April 2026
Received in revised form: 13 July 2026
Accepted: 8 August 2026
Published: 11 November 2026

ABSTRACT

Teaching programming control structures is a core but continually difficult component of computer science education since traditional lecture-based instructional approaches often cannot maintain student engagement or correct current problematic assumptions. However, despite the promising potential that game-based learning presents, the published applications are heavily based on superficial rewards (e.g., points, badges) and rarely address common misconceptions related to programming control structures. This paper proposes a theory-based teaching framework, Game-Driven Pedagogical Framework (GDPF), which is a synthesis of Self-Determination Theory, Cognitive Load Theory and Constructivist Learning Theory in a four-layer architecture. A mixed-method assessment involving sixty undergraduate students showed that GDPF positively influenced students' engagement, enjoyment, and motivation, with engagement emerging as the strongest contributor to positive learning perceptions. Qualitative findings also showed enhanced conceptual understanding and increased confidence in solving programming problems. Such results make GDPF a reproducible, pedagogically founded alternative to conventional lecture-based approaches, which can provide practical recommendations on integrating the idea of a game-based approach into the teaching of programming.

Keywords:
Game-based learning; Pedagogical framework; Programming control structures; Programming education; Gamification

Introduction

Computer science education is essential to the development of computational thinking and programming skills that are required in the digital economy (Li et al., 2023). However, despite the development of instructional technologies, such as interactive coding tools, automated grading systems, and intelligent tutoring systems, learners still cannot get through basic ideas like loops and conditionals (Combéfis, 2022; Prather et al., 2023). These control structures form the keystones of programming logic, but because of being abstract, they create a fixed number of misunderstandings, elevate error rates, and raise the withdrawal and failure rates in introductory classes (Watson and Li, 2014; Altadmri and Brown, 2015; Xu et al., 2025).

Traditional lecture-based approaches often struggle to sustain student engagement when teaching programming control structures. Because programming concepts such as loops and conditionals are abstract and cognitively demanding, many students experience difficulties in understanding, applying, and retaining these concepts, leading to reduced motivation and poorer learning outcomes (Qian & Lehman, 2017; Swacha, 2023; Ishaq & Alvi, 2023). These challenges highlight the need for instructional approaches that both simplify complex concepts and promote active student engagement.

Game-based learning (GBL) has come out as a promising solution to this challenge. It has been shown by previous studies that motivation, fun, and knowledge acquisition can be improved through the use of game technologies, including interactive tasks, feedback, and goal-focused assignments (Zhan et al., 2022; Cuervo-Cely et al., 2022). Meta-analyses support the idea that gamification can have a beneficial effect on engagement and learning in programming education, although the game features should be oriented to the pedagogical goals to achieve a high level

of effectiveness (Ishaq and Alvi, 2023). The current reviews indicate that gamification in the context of programming education is mostly introduced and sustained on the basis of points, badges and leaderboards, which lack sufficient focus on the underlying cognitive processes (Zhan et al., 2022), which limits the success of the practice in the long-term effects on student learning.

Although game-based learning has demonstrated positive effects on student engagement and learning, existing approaches primarily emphasise gamification features such as points, badges, and leaderboards. Many of these approaches lack an explicit integration of established learning theories and provide limited support for addressing students' misconceptions related to programming control structures. To address this gap, this study proposes the Game-Driven Pedagogical Framework (GDPF), which integrates Self-Determination Theory (SDT), Cognitive Load Theory (CLT), and Constructivist Learning Theory (CLeT) within a structured four-layer framework for teaching programming control structures. The framework comprises four interconnected layers: instructional planning, game-driven programming exercise design, interactive learning implementation, and learning assessment, providing a structured pathway for integrating pedagogical theory with game-based programming instruction. In this regard, the study contributes to the growing body of knowledge on game-based learning by proposing a theory-driven pedagogical framework and providing practical guidance for instructors seeking to integrate game elements with established learning theories to enhance students' learning experiences in programming education.

Literature Review

Challenges in Programming Education

The challenge of learning programming, especially in terms of control structures like loops, conditionals, recursion, etc., is one that beginners continue to face. These ideas require abstract thinking and logical reasoning, but students often have difficulty with syntax, semantics, and the ability to apply conceptual knowledge to problem-solving (Chiodini et al., 2025; Hoq et al., 2025). The misunderstandings about the control flow are also the reasons why introductory programming courses are characterised by a large number of logical errors, higher cognitive load, and dropout rates (Watson & Li, 2014; Altadmri & Brown, 2015; Xu et al., 2025). Conventional modes of teaching programming, lecture-based teaching, syntax-based exercises and textbook-based learning have long dominated classrooms. Nevertheless, research shows that such passive methods of learning are not always effective in the way they involve students (Ramírez-Echeverry, Restrepo-Calle, & Torres Jiménez, 2025). A lot of students struggle to associate programming with the real world and find it abstract and unrelatable, so they fail to get motivated and retain it (Swacha, 2023). Additionally, traditional evaluations pay much attention to code correctness rather than problem-solving processes, which restricts a person to deep concept development (Keuning, Jeering, & Heeren, 2018). These weaknesses underscore the necessity of other pedagogical techniques that could eliminate the misconceptions and at the same time improve the motivation and interest of the learners.

Game-Based Learning in Programming Education

To address these issues, teachers have sought alternative approaches to teaching to make programming more approachable and interesting. Game-based learning is one of these strategies that can use the motivational and interactive nature of games to help in learning. GBL has become a successful method of teaching programming. It uses the interactive game aspects, including challenges, rewards, and real-time feedback to improve student motivation and learning results (Ishaq & Alvi, 2023; Swacha, 2023). It has been shown that the use of GBL within the context of programming classes enhances the ability to solve problems, engagement, and knowledge retention (Li et al., 2025). Serious games, gamification, and coding puzzles are some of the implementations of GBL that encourage experiential learning (Giannakoulas & Xinogalos, 2024). However, although GBL has demonstrated to be helpful, most of the implementations lack a framework that is organised to suit the purposes of pedagogy, which restricts its effectiveness in solving fundamental programming challenges. GBL can also encourage students in the short term and fail to guarantee conceptual development in an individualistic way, unless it is supported with an explicit pedagogical framework.

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Existing Frameworks and Gaps

A number of platforms and frameworks have tried to incorporate game-based learning in programming education. Learners can use platforms like Scratch and Code.org, which offer interactive, game-like environments to begin learning (Meerbaum-Salant et al., 2013; Kalelioğlu, 2015), and learners can be motivated to learn programming by playing games that are designed to teach (Giannakoulas & Xinogalos, 2024). In spite of this, there are still four enduring flaws: (1) the use of superficial reward systems that have no pedagogical basis (Coelho et al., 2025); (2) a focus on short-term pleasure and neglect of misconceptions or cognitive load (Baah et al., 2024); (3) the lack of a theory-based framework that can be applied consistently regardless of the context (Li et al., 2023); and (4) inadequate attention to control structures that is a well-documented Such loopholes highlight the need to have an instructional design that goes beyond gamification and incorporates game-led, pedagogically sound principles. The Game-Driven Pedagogical Framework (GDPF) is the solution to overcome such drawbacks, combining ideas of SDT, CLT and CLeT in a logical four-layered framework.

Table 1 demonstrates that Scratch, Code.org, and serious games offer excellent points of entry into programming education but lack a systematic pedagogical foundation, focus on superficial incentives, and are rarely aimed at correcting the misconceptions of control structures. GDPF addresses these deficiencies by incorporating motivational (SDT), cognitive (CLT), and constructivist (CLeT) theories within a systematic, replicable framework, thus making GDPF a theory-based step forward in comparison to current platforms and a way to provide a medium of both interaction and profound intellectual insight.

Table 1
Comparison of Existing Game-Based Frameworks and GDPF

	Scratch	Code.org	Serious Games for Programming	GDPF (Proposed)	Evidence Source
Pedagogical Grounding	Weak (primarily exploratory play)	Weak (mostly task-based gamification)	Moderate (designed around challenges)	Strong (grounded in SDT, CLT and CleT)	Meerbaum-Salant et al. (2013); Kalelioğlu (2015); Giannakoulas & Xinogalos (2024)
Focus on Motivation	High (intrinsic play & creativity)	High (short-term badges & rewards)	Moderate (engagement via gameplay)	High (supports intrinsic motivation via autonomy, challenge, and structured progression)	Li et al. (2024); Gupta & Goyal (2022); Zhan et al. (2022)
Focus on Cognitive Load & Misconceptions	Low (misconceptions often persist)	Low (little attention to reducing cognitive overload)	Partial (some scaffolding, but not systematic)	High (explicitly designed to manage cognitive load and reduce misconceptions)	Sweller (1988); Prather et al. (2023); Quintero-Manes et al. (2024)
Target on Control Structures	Limited (general coding, not explicit focus)	Limited (focuses on introductory coding)	Weak (not directly focused on control structures)	Strong (directly targets control structures)	Prather et al. (2023); Xu et al. (2025)
Replicability for Instructors	Moderate (widely adopted, but unstructured)	High (global adoption, lesson packs)	Moderate (custom designs, not generalizable)	High (structured, replicable 4-layer framework)	Li et al. (2023); Giannakoulas & Xinogalos (2024)
Limitation	Lacks structured pedagogy; misconceptions remain	Rewards are often surface-level not theory-driven	Case-specific; lacks a unified pedagogical model	Still requires comparative validation across contexts	Coelho et al. (2025); Baah et al. (2024)

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Positioning the GDPF

GDPF addresses the given gaps by grounding its design on the three prominent theories: SDT to provide motivation sustainability, CLT to address cognitive demands, and CLeT to encourage active knowledge building. GDPF is structured into four layers, and it goes further than the surface-based gamification by mapping the game elements to the learning goals. In particular, it (1) ensures that the mechanics of games help in learning as opposed to being distracting, (2) uses progressive tasks to challenge general misconceptions in control structures, and (3) provides a structured framework that can be customised by instructors to different contexts. Therefore, GDPF improves pleasure and conceptual knowledge- an outcome that had hardly been attained methodically/systematically on a replicable scale using the previous methods.

Conventional gamification adds rewards on top of existing lessons. GDPF reverses this sequence: the learning goal is defined first (Layer 1), and a game mechanic is chosen afterward specifically to serve that goal (Layer 2). This is not required by platforms like Scratch or Code.org, which support open-ended or task-based play without linking each game element back to a specific learning goal in advance.

Research Objectives/Questions

The specific objectives of the study are:

1. To investigate the level of students' engagement, enjoyment, motivation, and perception of their learning experience when using GDPF in programming control structures.
2. To analyse the interrelationships among engagement, enjoyment, and motivation, and their influence on students' overall perceptions of learning with GDPF.
3. To contribute to the development of innovative pedagogical strategies by integrating GDPF to foster deeper learning in programming control structures.

Theoretical Framework

The Game-Driven Pedagogical Framework (GDPF) is developed based on three known learning theories: SDT, CLT and CLeT. All these theories together offer the motivation, cognitive, and experience-based basis that will be useful in developing effective game-based programming teaching.

Self-Determination Theory (SDT)

SDT focuses on the psychological needs of autonomy, competence, and relatedness as motivational drivers of intrinsic motivation (Deci & Ryan, 2000). In GDPF, these needs are nurtured by small challenges which increase programming competence, collaboration among peers which promotes relatedness and a chance to enable learners to choose problem-solving strategies that increase autonomy. Recent reports establish that relatedness and autonomy are likely to be promoted through gamified and game-based methods based on SDT, but intentional scaffolding is required to support competence completely (Li et al., 2024; Gupta & Goyal, 2022).

Cognitive Load Theory (CLT)

CLT describes that learning has been found to be optimised when an instructional design avoids irrelevant cognitive load but does not underestimate more complex tasks (Sweller, 1988). The use of interactive games as tasks in GDPF minimises the unwarranted distractions, scaffolding activities allow the simple, more complicated tasks to build up and immediate feedback allows confusion to be eliminated before it builds up. Recent studies show that CLT-based designs can facilitate programming education by decreasing the number of misconceptions and facilitating problem-solving (Sandoval-Medina et al., 2024; Quintero-Manes et al., 2024).

Constructivist Learning Theory (CLeT)

CLeT states that learning occurs most effectively through active interaction and experience and not passive learning (Piaget, 1970). This principle is achieved in GDPF by way of practical coding exercises, game-based activities that provide a simulation of real-world programs, and group discussions that foster groups in the process of constructing knowledge. Recent findings have shown that constructivist learning classrooms enhance student engagement and comprehension when tasks are active and collaborative (Kim et al., 2025).

Game-Driven Pedagogical Framework (GDPF)

Through the combination of these theories, GDPF presents a layered model of design, which integrates motivational principles (SDT), cognitive efficiency (CLT), and active learning (CLeT) in a logical manner. Such a theoretical basis preconditions that the framework not only makes the programme work interesting but also contributes to further conceptual insight and prevents long-standing misunderstandings in control structures. Specific design elements are integrated in the four layers of GDPF as the theoretical foundations, as shown in Figure 1. SDT provides details about motivational features like learner autonomy, progressive challenges, and collaborative activities. CLT informs the sequencing of tasks for programming, the mode of teaching and the feedback to the learner, ensuring that the cognitive load is not unnecessarily high. CLeT is manifested in hands-on coding, real-world problem-solving in a game, discussion with peers, and reflective learning that allows students to actively develop knowledge about programming.

Figure 1
Game-Driven Pedagogical Framework (GDPF)

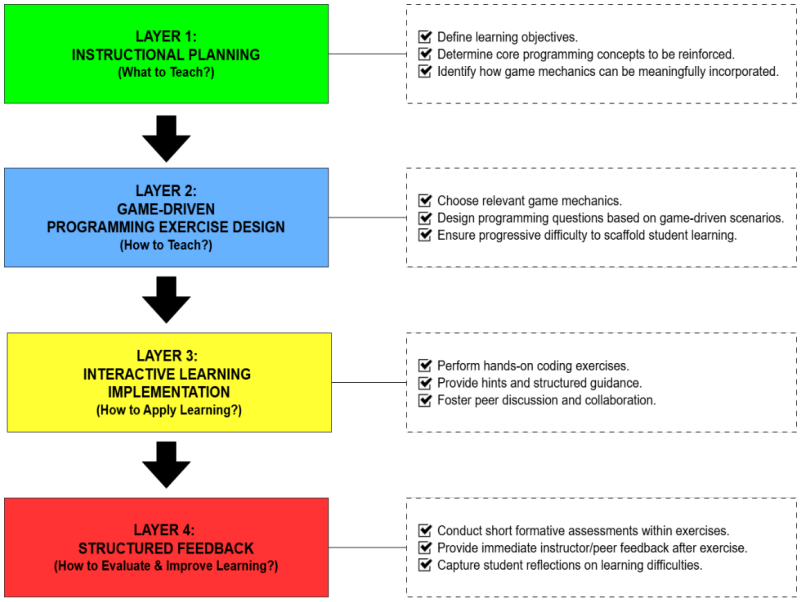


Table 2 shows the details of the GDPF, highlighting the purpose and the key steps involved in each layer of the framework.

Table 2
Game-Driven Pedagogical Framework (GDPF)

Layer 1: Instructional Planning (What to Teach?)	
Purpose	This layer sets the foundation by identifying what students need to learn.
Key Steps	● Define clear learning objectives related to control structures. ● Determine the core programming concepts that students need to practice. ● Identify how and where the game mechanics can be incorporated into the lesson.

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Layer 2: Game-Driven Programming Exercises Design (How to Teach?)	
Purpose	This layer focuses on crafting meaningful programming exercises that integrate game-driven elements.
Key Steps	● Choose a relevant game mechanic that aligns with the programming concept. ● Incorporate game mechanic exercises that align with learning objectives. ● Provide incremental challenges by gradually increasing the exercise's difficulty.
Layer 3: Interactive Learning Implementation (How to Engage Students?)	
Purpose	This layer ensures students actively engage with the exercises and practice problem-solving.
Key Steps	● Perform hands-on coding where students experiment and debug their programs. ● Provide hints and feedback to guide students as they code. ● Foster peer discussion & collaboration to enhance learning.
Layer 4: Learning Assessment (How to Measure Success?)	
Purpose	This layer focuses on evaluating students' understanding and refining the exercises for better learning outcomes.
Key Steps	● Conduct formative assessments to monitor progress. ● Conduct summative assessments to evaluate overall learning. ● Gather student feedback through surveys to measure engagement and motivation.

Methods

Research Design

This study employed a mixed-methods design to examine students' experiences and perceptions of the Game-Driven Pedagogical Framework (GDPF) in learning programming control structures. The quantitative strand involved a survey using a five-point Likert scale, ranging from 1 = strongly disagree to 5 = strongly agree, to measure students' engagement, enjoyment, motivation, and perceptions of learning. Descriptive statistics were used to determine the general tendency, while inferential procedures such as correlations and regressions were used to examine correlations and predictive relationships among the measured variables. The qualitative strand was used to supplement the survey with open-ended questions that enabled the respondents to give reflective descriptions of their GDPF experience. Findings from both strands were integrated through triangulation, ensuring that quantitative trends were supported and enriched by qualitative insights. This combined approach provided a comprehensive understanding of GDPF's role in enhancing active student engagement and fostering deeper learning in programming.

Participants

This study involved 47 undergraduate students aged between 18 and 22 who were enrolled in the Diploma in Computer Science program at Universiti Teknologi MARA (UiTM) Cawangan Sarawak, Kampus Samarahan 2. The sample consisted of a convenience sample of students who were already enrolled in the course. Subjects volunteered to participate and informed consent was obtained.

Research Instruments

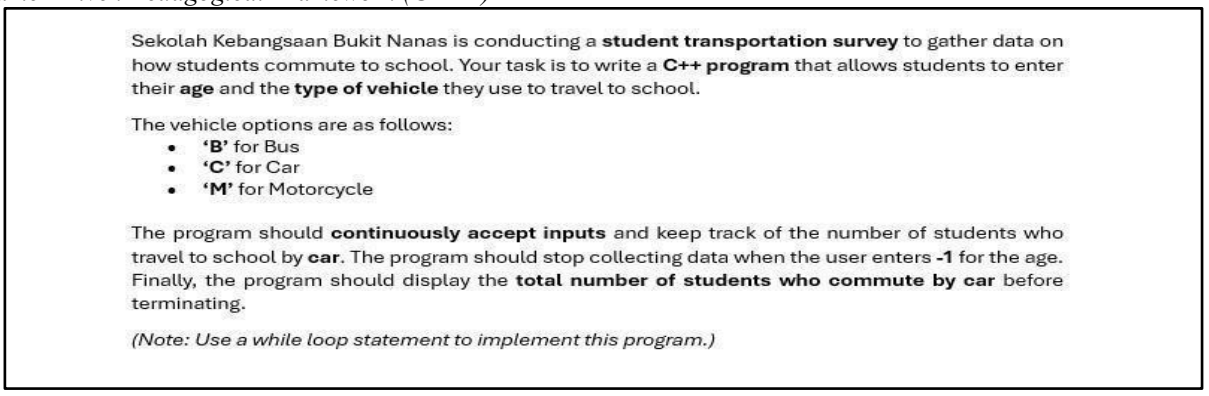
The GDPF intervention was conducted during a programming lesson on control structures and consisted of four sequential stages aligned with the proposed framework. First, the instructor introduced the learning objectives and fundamental concepts of loops and conditional statements (Layer 1: Instructional Planning). Next, students completed game-driven programming exercises designed with progressively increasing levels of difficulty and game mechanics that reinforced the targeted programming concepts (Layer 2: Game-Driven Programming Exercise Design). During the implementation stage, students solved the programming tasks individually while receiving instructor guidance, immediate feedback, and opportunities for peer discussion when needed (Layer 3: Interactive Learning Implementation). Finally, students completed the learning assessment, including formative evaluation, a post-intervention questionnaire, and reflective feedback on their learning experience (Layer 4: Learning Assessment).

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Students worked on two programming tasks created according to GDPF principles, assisted by the instructor's feedback and discussion with peers. The initial task, as shown in Figure 2, was a typical problem with a C++ transportation survey that focused on syntax, loops and conditionals but provided very little interaction. The second one, as shown in Figure 3, was the game-based activity "Defeat the Mighty Pig", which used the same concepts in an interactive game with player lives, attacking options, and losing/winning conditions. The design is based on the principles of GDPF because it is motivated by the specific purpose of setting clear goals based on SDT, managing complexity with progressive challenges based on CLT, and facilitating active learning based on meaningful contexts based on CLeT. After the activity, students were asked to provide short qualitative reflections on the programming difficulties they faced and subsequently completed a survey designed to capture engagement, enjoyment, motivation, and perceptions of learning. To ensure construct and contextual validity, the survey instrument was adapted from established baseline scales and systematically tailored to the programming education context and the instructional components of the Game-Driven Programming Framework (GDPF). Specifically, items measuring *behavioural and emotional engagement* were derived from Fredricks et al. (2004) and modified by shifting the referent from general classroom participation to active coding within a gamified environment with adaptation of general attention prompts to focus on active immersion and peer collaboration during game-driven programming tasks.

Constructs assessing *intrinsic motivation and enjoyment* were adapted from Deci and Ryan's (1985). Self-Determination Theory frameworks, wherein standard activity prompts were contextualised to reflect the game-based pedagogy by modifying general enjoyment statements to evaluate how game mechanics and creative control over algorithm development enhanced intrinsic interest in programming. Finally, items evaluating *perceptions of learning and self-efficacy* were drawn from Schunk and Pajares (2005) and adapted by replacing broad academic competence statements with domain-specific programming benchmarks, such as the perceived ability to master control structures, debug syntax errors, and apply logical algorithms to real-world scenarios within the GDPF. Through these targeted lexical and thematic adaptations, the established scales were explicitly aligned with the unique cognitive and interactive demands of our game-driven programming curriculum. Open-ended questions were also added to help obtain individual experiences and to receive more qualitative data on how students view GDPF.

Figure 2
Game-Driven Pedagogical Framework (GDPF)


Sekolah Kebangsaan Bukit Nanas is conducting a **student transportation survey** to gather data on how students commute to school. Your task is to write a **C++ program** that allows students to enter their **age** and the **type of vehicle** they use to travel to school.

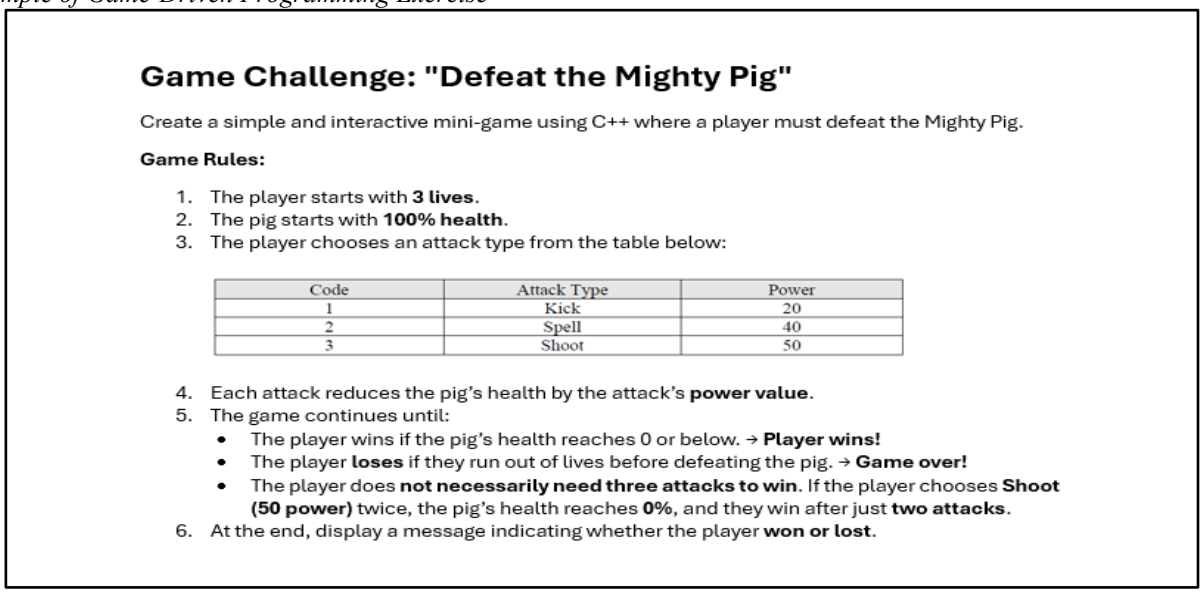
The vehicle options are as follows:

- **'B'** for Bus
- **'C'** for Car
- **'M'** for Motorcycle

The program should **continuously accept inputs** and keep track of the number of students who travel to school by **car**. The program should stop collecting data when the user enters **-1** for the age. Finally, the program should display the **total number of students who commute by car** before terminating.

(Note: Use a while loop statement to implement this program.)

Figure 3
Sample of Game-Driven Programming Exercise



Game Challenge: "Defeat the Mighty Pig"

Create a simple and interactive mini-game using C++ where a player must defeat the Mighty Pig.

Game Rules:

1. The player starts with **3 lives**.
2. The pig starts with **100% health**.
3. The player chooses an attack type from the table below:

Code	Attack Type	Power
1	Kick	20
2	Spell	40
3	Shoot	50

4. Each attack reduces the pig's health by the attack's **power value**.
5. The game continues until:
   - The player wins if the pig's health reaches 0 or below. → **Player wins!**
   - The player **loses** if they run out of lives before defeating the pig. → **Game over!**
   - The player does **not necessarily need three attacks to win**. If the player chooses **Shoot (50 power)** twice, the pig's health reaches **0%**, and they win after just **two attacks**.
6. At the end, display a message indicating whether the player **won or lost**.

Data Analysis

SPSS version 25 was used to analyse quantitative data to measure the experience and correlation between engagement, enjoyment, motivation, and perceptions of learning, and apply descriptive statistics, correlation analysis, and regression analysis. Open-ended questions gave qualitative data, which were analysed descriptively. The reviews of student reflections were analysed and sorted out to determine common ideas and points of view. Through this process, such common trends as heightened enthusiasm, understanding of control structures, and motivation were pointed out in learning a game-based task.

Results

Demographic Profiles

Table 3 presents a summary of the respondents' demographic profiles. The study included 23 male students (48.9%) and 24 female students (51.1%). Regarding prior programming experience, 29 respondents (61.7%) reported having a basic understanding as their highest level, while 3 respondents (6.4%) had no programming experience at all. Moreover, the majority had not participated in any formal programming courses before enrolling at UiTM. Most participants engage in computer or mobile gaming, with the majority playing games on a daily basis.

Table 3
Demographic Profiles of Respondents

Demographic	Label	Frequency	Percentage (%)
Gender	Male	23	48.9
	Female	24	51.1
Highest level of prior programming experience	No prior experience	3	6.4
	Basic understanding	29	61.7
	Intermediate knowledge	15	31.9
Formal programming course before entering UiTM	Yes	14	29.8

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Demographic	Label	Frequency	Percentage (%)
Playing computer or mobile games	No	33	70.2
	Yes	45	95.7
Frequency of playing games in a week	No	2	4.3
	Rarely (Less than once per week)	6	12.8
	Occasionally (1 - 2 times per week)	10	21.3
	Regularly (3 - 5 times per week)	7	14.9
	Frequently (6 - 7 times per week)	4	8.5
	Daily (Multiple times a day)	20	42.6

Reliability Test

A reliability test was used to determine internal consistency of the research instruments: Cronbach's Alpha was used since the coefficient measures internal consistency of item sets and establishes reliability of multi-item scales by estimating the inter-item correlation (Cronbach, 1951). In basic research, a Cronbach's alpha of 0.70 or more is considered normal but 0.80 and above is desirable in order to ensure higher reliability especially in more advanced research contexts (Nunnally, 1978). The Cronbach's alpha of all items in Table 4 is above 0.80, which shows that the tools used in the present study are very reliable and consistent.

Table 4
Reliability Test

Variable	Number of Items	Cronbach's Alpha (α)
Engagement	8	0.874
Enjoyment	9	0.855
Motivation	8	0.895
Perception	11	0.937

Normality Test

In order to conduct parametric statistical tests such as correlation and ANOVA, it is necessary to meet the criteria of normality in the data. As a result, a normality test was conducted by looking at skewness values. The skewness values of all variables presented in Table 5 were within the regular acceptable range of -1.0 and +1.0, hence supporting the approximate normal distribution (Hair et al., 2019; Pallant, 2020). The implication of these findings is that the parametric tests of normality, Pearson's correlation analysis, and regression modelling were met reasonably.

Table 5
Normality Test

Variable	Skewness Value
Engagement	-0.231
Enjoyment	-0.425
Motivation	-0.364
Perception	-0.525

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Examine the extent of students' engagement, enjoyment, motivation, and perception of their learning experience when utilising GDPF in programming control structures.

In order to measure the engagement, enjoyment, motivation, and perception of their experience with GDPF, the respondents were requested to answer several questions pertaining to those variables and derive the mean scores of each variable based on their responses and classify them as presented in Table 6.

Table 6
Level of Agreement According to Engagement, Enjoyment, Motivation and Perception on Students' Experience with GDPF

Mean Score	Mean score interpretation table
1.00 – 1.80	Strongly disagree
1.81 – 2.60	Disagree
2.61 – 3.20	Moderately agree
3.21 – 4.20	Agree
4.21 – 5.00	Strongly agree

Source: Moidunny (2009)

Table 7 provides information on the mean scores of student experiences using the GDPF on four dimensions, including engagement, enjoyment, motivation, and perception. The results demonstrate great concurrence in all variables with the means ranging between 4.21 and 5.00. These findings indicate that the GDPF provided participants with an overwhelmingly positive experience in the assessed areas.

Table 7
Mean Score According to Engagement, Enjoyment, Motivation and Perception on Students' Experience with GDPF

Variable	Mean Score	Level
Engagement	4.2367	Strongly agree
Enjoyment	4.3428	Strongly agree
Motivation	4.3910	Strongly agree
Perception	4.5029	Strongly agree

The descriptive statistics of individual items are represented in Table 8. The highest mean (M = 4.38, SD = 0.739) was associated with the statement 'I actively participate in hands-on coding exercises during lessons', which implies that the students actively engage in the classroom processes. On the other hand, the lowest mean (M=3.72, SD=0.902) was found in the statement 'I actively seek out additional programming challenges beyond class assignments', suggesting relatively low involvement in independent learning outside the classroom.

Table 8
Students' Engagement in Learning Programming Control Structure

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
I find learning programming control structures to be engaging and stimulating	0 (0.0%)	0 (0.0%)	3 (6.4%)	24 (51.1%)	20 (42.6%)	4.36	0.605
I actively participate in hands-on coding exercises during lessons	0 (0.0%)	0 (0.0%)	7 (14.9%)	15 (31.9%)	25 (53.2%)	4.38	0.739
I feel engaged with the topic in the programming exercises.	0 (0.0%)	0 (0.0%)	7 (14.9%)	21 (44.7%)	19 (40.4%)	4.26	0.706

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
I remain focused and committed to completing programming assignments.	0 (0.0%)	0 (0.0%)	5 (10.6%)	20 (42.6%)	22 (46.8%)	4.36	0.673
I try to solve programming exercises even when they are difficult or frustrating	0 (0.0%)	1 (2.1%)	6 (12.8%)	19 (40.4%)	21 (44.7%)	4.28	0.772
I am eager to explore new programming concepts beyond class assignments.	0 (0.0%)	4 (8.5%)	4 (8.5%)	18 (38.3%)	21 (44.7%)	4.19	0.924
I feel more confident applying programming skills after completing exercises.	0 (0.0%)	1 (2.1%)	3 (6.4%)	22 (46.8%)	21 (44.7%)	4.34	0.700
I actively seek out additional programming challenges beyond class assignments.	0 (0.0%)	5 (10.6%)	12 (25.5%)	21 (44.7%)	9 (19.1%)	3.72	0.902
Total	0	11 (2.92%)	47 (12.5%)	163 (43.35%)	158 (42.02%)	4.24	0.554

Table 9 lists the descriptive statistics relating to the enjoyment of learning programming control structures among students. The results show that the highest mean score was obtained in the item of 'I feel excited when I successfully solve a challenging programming problem' (M = 4.64, SD = 0.605), which may mean that students find a lot of enjoyment and satisfaction in solving a challenging problem in programming, but they are not as willing to spend more time and refine their codes beyond what the task demands, perhaps because of time constraints, workload, or less interest in perfection more than functionality. Conversely, the item with the lowest mean score was the item, 'I spend extra time working and refining my code because I enjoy the challenge' (M = 4.00, SD = 1.063); this could be attributed to the fact that although students are excited by the problem-solving process, they are less willing to put in more work than necessary to complete the task; this could be the result of time limitations, workload, or a desire to achieve functionality rather than perfection.

Table 9
Students' Enjoyment in Learning Programming Control Structure

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
I enjoy solving programming exercises related to control structures.	0 (0.0%)	0 (0.0%)	7 (14.9%)	18 (38.3%)	22 (46.8%)	4.32	0.726
I feel a sense of accomplishment when completing programming tasks.	0 (0.0%)	0 (0.0%)	3 (6.4%)	15 (31.9%)	29 (61.7%)	4.55	0.619
I find debugging and troubleshooting my code to be an enjoyable challenge.	1 (2.1%)	0 (0.0%)	13 (27.7%)	16 (34.0%)	17 (36.2%)	4.02	0.921

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
I feel excited when I successfully solve a challenging programming problem.	0 (0.0%)	0 (0.0%)	3 (6.4%)	11 (23.4%)	33 (70.2%)	4.64	0.605
I find solving programming problems rewarding.	0 (0.0%)	1 (2.1%)	2 (4.3%)	20 (42.6%)	24 (51.1%)	4.43	0.683
The topic covered in the exercises provides a good balance between learning and fun	0 (0.0%)	0 (0.0%)	4 (8.5%)	22 (46.8%)	21 (44.7%)	4.36	0.640
The structure of the lessons makes learning programming enjoyable.	0 (0.0%)	0 (0.0%)	5 (10.6%)	19 (40.4%)	23 (48.9%)	4.38	0.677
I look forward to coding exercises and assignments in class.	0 (0.0%)	0 (0.0%)	4 (8.5%)	21 (44.7%)	22 (46.8%)	4.38	0.644
I spend extra time working and refining my code because I enjoy the challenge.	2 (4.3%)	1 (2.1%)	11 (23.4%)	14 (29.8%)	19 (40.4%)	4.00	1.063
Total	3 (0.71%)	2 (0.47%)	52 (12.29%)	156 (36.88%)	210 (49.65%)	4.34	0.508

Table 10 summarises the mean and the standard deviation of the motivation of students in learning programming control structures. The scores show that the most significant mean was achieved on the item 'I believe that learning programming is valuable for my future career' (M = 4.62, SD = 0.534), which demonstrates the significance of constructive feedback when maintaining motivation among the students. In the meantime, the lowest mean score was achieved on the item 'I am eager to learn more advanced programming concepts' (M = 4.23, SD = 0.729) indicating that, even though students are interested in developing their more advanced programming knowledge, they might be restrained by the perceived difficulties or the necessity to get better acquainted with the foundational ideas before proceeding with the further development of their knowledge.

Table 10
Students' Motivation in Learning Programming Control Structure

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
The topic used in the programming exercises motivates me to learn programming.	0 (0.0%)	0 (0.0%)	9 (19.1%)	16 (34.0%)	22 (46.8%)	4.28	0.772
I feel motivated to improve my programming	0 (0.0%)	0 (0.0%)	5 (10.6%)	21 (44.7%)	21 (44.7%)	4.34	0.668

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
skills through regular practice.							
I am eager to learn more advanced programming concepts.	0 (0.0%)	0 (0.0%)	8 (17.0%)	20 (42.6%)	19 (40.4%)	4.23	0.729
I am willing to dedicate extra time and effort to improve my programming skills.	0 (0.0%)	0 (0.0%)	8 (17.0%)	17 (36.2%)	22 (46.8%)	4.30	0.749
I feel motivated when I receive feedback on my programming work.	0 (0.0%)	0 (0.0%)	1 (2.1%)	18 (38.3%)	28 (59.6%)	4.57	0.542
I feel motivated to learn different approaches to coding when collaborating with others.	0 (0.0%)	0 (0.0%)	4 (8.5%)	22 (46.8%)	21 (44.7%)	4.36	0.640
I believe that learning programming is valuable for my future career.	0 (0.0%)	0 (0.0%)	1 (2.1%)	16 (34.0%)	30 (63.8%)	4.62	0.534
I am excited to apply what I learn in real-world programming tasks.	0 (0.0%)	1 (2.1%)	3 (6.4%)	18 (38.3%)	25 (53.2%)	4.43	0.715
Total	0 (0.0%)	1 (0.27%)	39 (10.37%)	148 (39.36%)	188 (50.0%)	4.39	0.512

Table 11 illustrates the descriptive statistics for the construct of perception. The results show that the item 'Overall, I am satisfied with the game-driven pedagogical approach' achieved the highest mean score (M = 4.60, SD = 0.496), reflecting students' strong overall approval of the teaching method. Conversely, the lowest mean score was recorded for the item 'The structured learning activities supported my learning needs effectively' (M = 4.38, SD = 0.610). Although still relatively high, this suggests that some students may feel the structured activities could be further refined to better align with individual learning preferences and needs.

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

Table 11
Perception of the Learning Experience using Game-Driven Pedagogical Framework

Items	5-Point Likert Scale					Mean	SD
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree		
The game-driven exercises improved my understanding of programming control structures.	0 (0.0%)	0 (0.0%)	2 (4.3%)	16 (34.0%)	29 (61.7%)	4.57	0.580
I find game-driven exercises more engaging and effective than traditional teaching methods for learning programming.	0 (0.0%)	0 (0.0%)	3 (6.4%)	16 (34.0%)	28 (59.6%)	4.53	0.620
The exercises helped me retain programming concepts better.	0 (0.0%)	0 (0.0%)	2 (4.3%)	19 (40.4%)	26 (55.3%)	4.51	0.585
The challenges in the exercises were appropriate for my skill level.	0 (0.0%)	0 (0.0%)	2 (4.3%)	22 (46.8%)	23 (48.9%)	4.45	0.583
The programming exercises helped me develop better problem-solving strategies.	0 (0.0%)	0 (0.0%)	3 (6.4%)	19 (40.4%)	25 (53.2%)	4.47	0.620
I enjoyed collaborating with peers during programming tasks.	0 (0.0%)	1 (2.1%)	2 (4.3%)	19 (40.4%)	25 (53.2%)	4.45	0.686
The structured learning activities supported my learning needs effectively.	0 (0.0%)	0 (0.0%)	3 (6.4%)	23 (48.9%)	21 (44.7%)	4.38	0.610
The approach used in this course made programming topics more interesting.	0 (0.0%)	0 (0.0%)	2 (4.3%)	16 (34.0%)	29 (61.7%)	4.57	0.580
The programming exercises challenge me at an appropriate level.	0 (0.0%)	0 (0.0%)	2 (4.3%)	21 (44.7%)	24 (51.1%)	4.47	0.584
I would recommend game-driven learning for future programming courses.	0 (0.0%)	0 (0.0%)	3 (6.4%)	16 (34.0%)	28 (59.6%)	4.53	0.620
Overall, I am satisfied with the game-driven pedagogical approach.	0 (0.0%)	0 (0.0%)	0 (0.0%)	19 (40.4%)	28 (59.6%)	4.60	0.496
Total	0 (0.0%)	1 (0.19%)	24 (4.64%)	206 (39.85%)	286 (55.32%)	4.50	0.469

Correlation Analysis

Pearson's correlation analysis was employed to assess the strength and direction of relationships among engagement, enjoyment, motivation, and perception in learning programming control structures, as it is appropriate for continuous variables (Pallant, 2020). Table 12 summarises the correlation results, which revealed positive and statistically significant associations among all constructs. The strongest correlation emerged between enjoyment and motivation ($r = 0.778$, $p < 0.05$), indicating that students who derived greater enjoyment from the learning process were also more

motivated with the game-driven pedagogical approach and demonstrated higher persistence in learning. In comparison, a moderate correlation was identified between motivation and perception ($r = 0.650$, $p < 0.05$), suggesting that students' motivation contributed positively to their overall perception of the approach, although to a lesser extent than enjoyment.

Table 12
Pearson's Correlation Analysis

	Engagement	Enjoyment	Motivation	Perception
Engagement	1.000			
Enjoyment	0.813 (0.000*)	1.000		
Motivation	0.674 (0.000*)	0.778 (0.000*)	1.000	
Perception	0.766 (0.000*)	0.766 (0.000*)	0.650 (0.000*)	1.000

*Significant at $\alpha = 0.05$

Regression Analysis

A multiple regression analysis was conducted to examine the extent to which engagement, enjoyment, and motivation predicted students' perception of the game-driven pedagogical approach. Results of regression analysis in Table 13 reveal that the model was statistically significant ($F = 26.675$, $p < 0.05$) and the construct of engagement, enjoyment and motivation explains about 80.7% of the variance in perception. Among the predictors, engagement emerged as the strongest predictor ($\beta = 0.348$, $p < 0.05$), suggesting that students' active participation during lessons directly translates into higher overall perceptions of the method. In contrast, enjoyment and motivation were not significant predictors ($p > 0.05$).

Table 13
Results of Regression Analysis

	Coefficient	t	p-value	R^2	F-test
Constant	1.212	3.124	0.003	0.807	26.675 (0.000)
Engagement	0.348	2.637	0.012		
Enjoyment	0.330	1.950	0.058		
Motivation	0.087	0.659	0.513		

Discussions

The findings indicate positive student perceptions of GDPF in relation to engagement, enjoyment, motivation, and learning programming control structures. The high engagement rates during hands-on coding activities indicate that learning activities based on games and interaction processes can help to promote active classroom participation. Students reported high-level pleasure in solving tricky programming problems, which points to the importance of entrenching the learning activities in game-based contexts to attract emotional engagement and satisfaction. Nevertheless, the reduced time spent on refining code beyond what is needed by the task indicates that pleasure can be more task-oriented than process-oriented, which can be due to time constraints or workload. The motivational results indicate that students have a high appreciation for programming in their future lives, and this supports the importance of GDPF training students to understand real-life applications. However, a bit of reduced interest in further reading on advanced topics of programming indicates that students might want to have a stronger command of the basics before advancing.

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

The positive relationships, which are strong and positive, between the engagement, enjoyment, motivation, and perception imply that the constructs are closely related to each other. The findings of the regression also point to the engagement factor as the most significant factor determining the general attitude of students towards GDPF, which confirms the fact that active involvement in the innovative teaching methods is an important aspect. Even though enjoyment and motivation showed positive associations with perception, they were not as strong as the predictive ability of engagement. This result is consistent with the previous literature, which has noted the central role of behavioural engagement in defining the learning performance and student perceptions (e.g., Fredricks, Blumenfeld, & Paris, 2004; Skinner & Pitzer, 2012). It further demonstrates that encouraging the active engagement approach can be one of the main ways to increase the acceptance of game-based teaching strategies among students.

These quantitative findings were further supported and enriched by qualitative findings, which reported the students' learning experiences through an inductive thematic analysis of open-ended respondents' feedback. The qualitative coding process was a process of iterative review of student transcripts to identify common themes throughout the data, eventually reducing the data to three overarching themes: Collaborative and hands-on concept mastery, Game-driven immersion and relevance, Novice resilience and self-efficacy. The first theme is "collaborative and hands-on concept mastery," which emphasizes the active engagement and collaboration between students that is important to helping them learn programming concepts. The participants found that hands-on coding exercises, combined with group problem-solving, helped them better understand the logic and deepened their confidence in their skills when solving complex problems.

The students highlighted the importance of hands-on learning by practicing coding challenges and engaging in peer discussions, rather than simply listening to lectures. A number of respondents reported how hands-on activities have assisted their learning to understand this course, as one stated,

> *"Hands-on practices help me the most in understanding this course. It shows me a clearer concept of programming. Other than that, discussing the tasks with my friends also enhanced my knowledge"* (Respondent 12).

This shared learning opportunity established a "psychological safety net" for students to iteratively experiment with solutions without fear of failure, with another student repeating,

> *"I really enjoy doing programming especially with my friends. We helped each other with the programming, we learned from the mistakes that we've made and never repeated it again. Sometimes, programming can be quite difficult but every time our programming executed correctly, we would feel proud of ourselves"* (Respondent 10).

The second is game driven immersion and relevance, highlighting the importance of game-based learning for keeping students engaged and making programming concepts relevant. One common comment was that traditional coding problems were not always relevant to the individual and were less interesting. Programming tasks, on the other hand, through game-oriented learning activities were linked to familiar and meaningful contexts, resulting in more enjoyable and accessible learning. One student explained,

> *"All in all, based on my previous class, the game-driven learning experience is very effective to me because I can relate the logical algorithms into real life situation and it is what all of us are familiar with. I suggest to keep on using that method to encourage students to learn programming especially for control structure. "* (Respondent 14).

The pedagogy employed concepts that are familiar to the students, which encouraged them to actively engage in creative problem-solving and deeper learning, as reported by another participant

> *"integrating something the students are already fond of (games, pop culture, etc.) is effective in making them more engaged in learning programming concepts"* (Respondent 36).

The last theme is novice resilience and self-efficacy, which refers to students' programming experiences as they develop their skills, including those who have limited programming experience or none at all. Many participants started

e-ISSN 2600-8572 | DOI: https://doi.org/10.11113/itlj.v10.258

with a feeling of programming syntax and computational logic being daunting. The interactive and gamified learning process, however, did promote persistence in attempting to solve problems. As students practiced and continued to succeed with coding, they built up confidence in their coding skills over time. One novice learner described this transformation by stating,

> *"All great and perfect. I'm a student with no basic computer science but after learning this code I can do a basic coding* (Respondent 9).

The game-based mechanics served as a crucial bridge for struggling students, with one respondent explicitly highlighting this bridging effect:

> *"Programming was indeed fun, but quite challenging for me. Overcoming these challenges through exciting games and easy-to-understand examples helped me a lot and definitely helped me retain the knowledge. I hope this way of teaching can help more students who struggle with programming, like I did, to learn better"* (Respondent 23).

Ultimately, across all identified themes, the qualitative evidence confirms that combining hands-on peer collaboration with game-driven learning significantly enhances both conceptual retention and coding self-efficacy.

On the whole, the discussion indicates that GDPF is a useful tool to facilitate affective and cognitive elements of learning but indicates the possibility of further improvement of independent learning and structured activities personalisation.

Conclusion

This study introduced the Game-Driven Pedagogical Framework (GDPF) that is a theory-driven framework for teaching programming control structures using game-based learning (GBL) that combines Self-Determination Theory (SDT), Cognitive Load Theory (CLT), and Constructivist Learning Theory (CLeT). The results suggest that students perceived GDPF positively in terms of engagement, enjoyment, motivation, and overall learning experience, with engagement emerging as the strongest predictor of their perceptions of GDPF. The qualitative findings further suggest that the framework enhanced students' conceptual understanding and confidence in solving programming problems.

GDPF theoretically builds on the existing game-based learning methods by applying existing learning theories that are embedded in a structured pedagogical model. In practice, it offers teachers a systematic strategy to create game-based programs activities for active learning, motivation, and conceptual understanding. The results of this study are subject to these limitations and should be interpreted accordingly: the study was conducted in one institution with a relatively small sample of students and is limited to that setting and population. Further studies need to confirm the framework with larger and more diverse student populations, examine the impact of the framework in the long-term in students' conceptual understanding and explore ways to individualise GDPF activities and connect them with new educational technologies like intelligent tutoring systems.

Limitations and Future Studies

While these findings provide encouraging evidence regarding students' perceptions of the Game-Driven Programming Framework (GDPF), several limitations should be acknowledged. First, the study involved a relatively small sample from a single institution, which may limit the generalisability of the findings to other educational contexts and student populations. Second, the relatively short implementation period did not allow for an examination of longer-term learning experiences, knowledge retention, or whether students' positive perceptions might diminish as the novelty of the game-based approach decreases over time. Finally, the evaluation relied largely on self-reported measures, which reflect perceived self-efficacy and enjoyment rather than actual programming competence. Future research should involve larger, multi-institutional samples, longitudinal studies, and objective assessments of coding proficiency, such as standardised coding tests or automated code-quality analyses.

Conflict of Interest

The authors declared that they have no conflicts of interest to disclose.

Acknowledgment

The authors would like to acknowledge the support of first-year Diploma in Computer Science (CDCS110) students enrolled in the Fundamentals of Algorithm and Computer Problem Solving (CSC126) course at Universiti Teknologi MARA (UiTM) Cawangan Sarawak, Kampus Samarahan 2, who participated as respondents during the semester from October 2024 - February 2025.

References

Altadmri, A., & Brown, N. C. C. (2015). 37 million compilations: Investigating novice programming mistakes in large-scale student data. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)* (pp. 522–527). New York, NY: ACM. https://doi.org/10.1145/2676723.2677258

Baah, C., Govender, I., & Subramaniam, P. R. (2024). Enhancing learning engagement: A study on gamification's influence on motivation and cognitive load. *Education Sciences, 14*(10), 1115. https://doi.org/10.3390/educsci14101115

Chiodini, L., Bevilacqua, J., & Hauswirth, M. (2025). Surveying upper-secondary teachers on programming misconceptions. Proceedings of the 2025 ACM Conference on International Computing Education Research, V.1 (ICER 2025), 166–180. https://doi.org/10.1145/3702652.3744227

Coelho, F., Rando, B., Aparício, D. *et al.* The impact of educational gamification on cognition, emotions, and motivation: a randomized controlled trial. *Journal of Computers in Educ.* (2025). https://doi.org/10.1007/s40692-025-00366-x

Combéfis, S. (2022). Automated code assessment for education: Review, classification and perspectives on techniques and tools. *Software, 1*(1), 3–30. https://doi.org/10.3390/software1010002

Cuervo-Cely, K. D., Restrepo-Calle, F., & Ramírez-Echeverry, J. J. (2022). Effect of gamification on the motivation of computer programming students. *Journal of Information Technology Education: Research, 21*, 1–23. https://doi.org/10.28945/4917

Deci, E. L., & Ryan, R. M. (1985). *Intrinsic motivation and self-determination in human behavior*. Springer Science & Business Media.

Deci, E. L., & Ryan, R. M. (2000). *Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist, 55*(1), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

Ramírez-Echeverry, J. J., Restrepo-Calle, F., & Torres Jiménez, S. (2025). Self-regulated learning strategies in computer programming education. European Journal of Education, 60, Article e70052. https://doi.org/10.1111/ejed.70052

Giannakoulas, A., & Xinogalos, S. (2024). Studying the effects of educational games on cultivating computational thinking skills to primary school students: A systematic literature review. *Journal of Computers in Education, 11*, 1283–1325. https://doi.org/10.1007/s40692-023-00300-z

Gupta, P., & Goyal, P. (2022). Is game-based pedagogy just a fad? A self-determination theory approach to gamification in higher education. *International Journal of Educational Management, 36*(3), 341–356. https://doi.org/10.1108/IJEM-04-2021-0126

Hoq, M., Rao, A., Jaishankar, R., Piryani, K., Janapati, N., Vandenberg, J., & Akram, B. (2025). Automated identification of logical errors in programs: Advancing scalable analysis of student misconceptions. In Proceedings of the 18th International Conference on Educational Data Mining (EDM 2025). https://arxiv.org/abs/2505.10913

Ishaq, K., & Alvi, A. (2023). Personalization, cognition, and gamification-based programming language learning: A state-of-the-art systematic literature review. *arXiv preprint* arXiv:2309.12362. https://arxiv.org/abs/2309.12362

Sandoval-Medina, C., Arévalo-Mercado, C. A., Muñoz-Andrade, E. L., & Muñoz-Arteaga, J. (2024). Self-explanation effect of Cognitive Load Theory in teaching basic programming. Journal of Information Systems Education, 35(3), 303–312. https://doi.org/10.62273/GMIV1698

Kalelioğlu, F. (2015). A new way of teaching programming skills to K–12 students: Code.org. *Computers in Human Behavior, 52*, 200–210. https://doi.org/10.1016/j.chb.2015.05.047

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), 1–43. https://doi.org/10.1145/3231711

Kim, S.-K., Kim, T.-Y., & Kim, K. (2025). Development and effectiveness verification of AI education datasets based on constructivist learning principles for enhancing AI literacy. *Scientific Reports, 15*, Article 10725. https://doi.org/10.1038/s41598-025-95802-4

Li, L., Hew, K. F., & Du, J. (2024). Gamification enhances student intrinsic motivation, perceptions of autonomy and relatedness, but minimal impact on competency: A meta-analysis and systematic review. *Educational Technology Research and Development, 72,* 765–796. https://doi.org/10.1007/s11423-023-10337-7

Li, M., Ma, S., & Shi, Y. (2023). Examining the effectiveness of gamification as a tool promoting teaching and learning in educational settings: A meta-analysis. *Frontiers in Psychology, 14*, 1253549. https://doi.org/10.3389/fpsyg.2023.1253549

Li, Q., Jiang, Q., Liang, J.-C., Xiong, W., & Zhao, W. (2025). Roles of programming self-efficacy, cognitive styles, and self-regulated learning strategies on computational thinking in computer programming. Humanities and Social Sciences Communications, 12, Article 1412. https://doi.org/10.1057/s41599-025-05686-y

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239–264. https://doi.org/10.1080/08993408.2013.832022

Moidunny, K. (2009). *The effectiveness of the National Professional Qualification for Educational Leaders (NPQEL)* [Unpublished doctoral dissertation]. The National University of Malaysia.

Piaget, J. (1970). *Science of education and the psychology of the child.* New York, NY: Orion Press.

Prather, J., Pettit, R., Becker, B. A., Denny, P., & Loksa, D. (2023). Explain trace: Misconceptions of control-flow statements. *Computers, 12*(10), 192. https://doi.org/10.3390/computers12100192

Qian, Y., & Lehman, J. D. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education, 18*(1), 1–24. https://doi.org/10.1145/3077618

Quintero-Manes, R., Rodríguez, L., & González, M. (2024). Differentiated measurement of cognitive loads in computer programming learning environments: Scale validation and implications for instructional design. *Journal of Computing in Higher Education.* https://doi.org/10.1007/s12528-024-09411-7

Swacha, J. (2023). Does gamification make a difference in programming? Evaluating FGPE-supported programming education. *Education Sciences, 13*(10), 984. https://doi.org/10.3390/educsci13100984

Sweller, J. (1988). *Cognitive load during problem solving: Effects on learning. Cognitive Science, 12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)* (pp. 39–44). New York, NY: ACM. https://doi.org/10.1145/2591708.2591749

Xu, Z., Zhang, Y., & Lin, X. (2025). Misconceptions about control flow in introductory programming courses. *Proceedings of the 56th ACM Technical Symposium on Computer Science Education (SIGCSE '25)*. New York, NY: ACM. https://doi.org/10.1145/3724389.3731296

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, H., & Lan, Y. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers & Education: Artificial Intelligence, 3*, 100096. https://doi.org/10.1016/j.caeai.2022.100096