

# Programming Mechanics in Commercial Games: A Multiple Case Study

Leonardo Sorrentino

Tallinn University, Estonia

[sorreleo@tlu.ee](mailto:sorreleo@tlu.ee)

**Abstract:** Game based approaches to programming education have shown themselves to be effective at engaging students with the subject matter. The potential has been recognized since the creation of LOGO in the 1960s, and the field has seen a notable expansion since then. The most common approach these games use to teach is level by level maze navigation puzzles. In these games, players write code or place blocks to control the properties and behaviors of entities in the game. Parallel to the increase in educational games, there has been a growing niche of commercial games using programming as the main mechanic. To stand out in a saturated market, some of these games adopt novel and engaging approaches to incorporating programming into their game mechanics, going beyond simple maze navigation. This points to complementarity between educational and commercial games about programming: the former tend to be better pedagogically grounded and aligned with curricular requirements; while the latter offer more innovation and often provide a more engaging player experience. As such, learning from commercial titles could inspire new approaches in educational games about programming and make them more engaging and appealing. To this end, this paper presents a close reading of commercial programming games, with the goal of identifying potential novel directions that educational game designers could follow. Two game were chosen for the analysis: *GLADIABOTS*, and *The Farmer Was Replaced*. This article explores how they employ novel approaches to integrate programming with other mechanics and aspects of gameplay, and the learning that affords. More specifically, it looks at how the design of the game allows for the incorporation of sound computing education approaches into the core gameplay loop. As a result of the analysis, the paper presents design recommendations that can contribute to future efforts in creating engaging games for programming education.

**Keywords:** Game-based learning, Computing education research, Close reading, Computational thinking

---

## 1. Introduction

Video games have increasingly found their way into educational settings over the past decades, giving rise to a field of study known as game-based learning (GBL) (Hung et al. 2018). GBL has demonstrated broad applicability across numerous disciplines, ranging from language learning (Xu et al. 2019) to mathematics (Kebritchi et al. 2010) and programming (Videnovik et al. 2023). Programming has become a popular field for GBL approaches, as games afford a straightforward way to represent programming concepts in action through their rules, as opposed to describing them like a textbook might (Videnovik et al. 2023).

Despite the popularity of the field, however, most games made with the specific objective of teaching programming concepts rely on similar mechanics, especially when they use a block-based language. The most common approach is to have the player write code to control a character, with the goal to navigate a maze with a grid-like structure and reach a specific tile.

One of the first examples of using maze navigation as a game mechanic to teach programming is in the book *Karel the Robot: A Gentle Introduction to the Art of Programming* (Pattis 1981), where the reader has to write a set of instructions to guide the robot across a grid while avoiding obstacles. This, in turn, drew on the educational programming language LOGO, developed in 1967, which involved writing step-by-step instructions to control the movement of an on-screen cursor (visualized as a turtle). 45 years since *Karel the Robot*, there have been innovations both on the theoretical side (Branthome 2024) and the technical side, with better understanding of user interfaces and progression systems. However, the most popular game-based platforms for programming education, such as *Lightbot* (Lightbot Inc. 2008), *CodeCombat* (CodeCombat Inc. 2024) and *Gidget* (Lee 2014) are still built around the same basic maze navigation mechanics. While effective, maze navigation games do not take full advantage of the affordances offered by the medium, as they push the player towards finding a solution predetermined by the designer, and do not allow the player to freely explore and create with the tools at their disposal.

Parallel to the growth of educational games, there has also been an increase in commercial games that use programming as a core mechanic, meaning that writing code either with text or blocks is the main way the player has to interact with the game. To stand out in a saturated gaming market, these games experiment with their core mechanics to find a unique selling point that will make the game appealing to their target audience, leading to novel and interesting ways of integrating programming into the design of the game. Analyzing these games

can provide learning game designers with ideas on how to make their own artifacts more engaging and appealing.

The objective of this paper is to analyze popular commercial games with unique programming mechanics, focusing in particular on how their design facilitates a constructionist approach to learning, and the effect that has on the integration of learning elements in the design. To achieve this, the paper uses hermeneutic inquiry and close reading techniques to analyze two commercial games that display contrasting approaches to integrating programming concepts in the gameplay: *GLADIABOTS* (GFX47 2018), an arena auto-battler where the player can program the behavior of their robots with a visual scripting language, and *The Farmer Was Replaced* (Herzog 2025), an incremental game where the player can program a drone with a Python-like language to harvest crops automatically. Finally, the results of the analysis are used to propose strategies to integrate novel design ideas in the creation of games to teach programming.

## 2. Background

The theoretical background for this analysis is based on three fields: game design theory, game-based learning (GBL) and computing education research (CER).

Game design theory is a field of study dedicated to looking at how games are created, structured and experienced (Zubek 2020). The sections of game design theory relevant to this analysis are the concepts of core loop (Guardiola 2016) and uncertainty (Costikyan 2013): the core loop can be used to analyse the way the learning content is integrated with the mechanics, while uncertainty gives insight into how the game mechanically challenges the player. The core loop is a tool used to represent the experience of playing the game as a flowchart of basic actions that the player takes over and over again (Guardiola 2016). Uncertainty refers to how games make outcomes unpredictable, keeping the experience challenging and interesting for the player. Games, both educational and otherwise, can be viewed as problem-solving activities in which players navigate different types of uncertainty (Costikyan 2013). Costikyan (2013) identified 10 possible sources of uncertainty in games, that can be either intrinsic to the game, such as randomness and analytical complexity, or come from external factors such as player unpredictability. Outside of games, uncertainty has been found to be a key component for the learning process, as it creates desirable difficulty and enhances long term retention (Overoye and Storm, 2015).

The second field this analysis is based on is GBL. GBL has been defined as “the use of games, whether digital or non-digital, as educational tools to facilitate learning and skill development” (Wanglang 2024 p. 138). GBL focuses on using games to teach, either by themselves or as a support to other modes of instruction. Games have been found to have many characteristics that can contribute to effective learning, such as the ability to be intrinsically motivating and immerse the learner in a context of use (Gee 2003), create a safe environment for experimentation and failure (Plass et al. 2020), create an adaptive learning environment (Plass and Pawar 2020), and to engender self-regulation and reflection (Taub et al. 2020)

While many games have been specifically designed to teach, there is a growing interest in the potential of repurposing commercial games for education. Blanco and Engström (2020) pointed out that determining whether a game is engaging or interesting for the player can be a significant challenge, and that looking at successful commercial games can offer insights into designs that have proven to be successful.

The third field that this analysis draws from is CER. CER is an interdisciplinary field of research dedicated to the study of learning and teaching in the discipline of computing, using research approaches and theories from education research (Daniels et al. 2023).

An area of CER particularly relevant to this analysis is the study of teaching computational thinking (CT). CT has been popularized by Wing (2006), and is generally defined as a problem-solving approach that aims at formulating problems and their solutions in a way that could be carried out by an information processing agent (Wing 2006). Brennan and Resnick (2012) proposed a framework for understanding computational thinking based on three dimensions: concepts, such as conditionals and iterations, practices, such as debugging, and perspectives, meaning the perspectives the designers form about the world and themselves.

Another relevant area of CER is the study of different educational approaches to teaching programming. An approach that has been found to be effective in creating deep engagement with the material is inquiry-based learning, where exploration and curiosity are prioritized over direct instruction (Dagienè et al. 2023). A subset of inquiry-based learning specific to computing education is code production, a version of project-based learning. In this approach, rather than small tasks built around specific language constructs, students learn to

create complex systems to accomplish a task or project (Apiola et al. 2023). This approach is closely related to constructionism, a theory developed by Papert (1993) that centers around the computer as a medium for expression and learning, and letting the student explore the tools and create artifacts relevant for them.

There has also been research at the intersection of GBL and CER: meta analyses have shown that GBL is one of the most popular approaches to computing education (Apiola et al. 2023, Dagienė et al. 2023), and can be effective both for teaching specific programming languages (Branthome, 2024) and CT, with particularly strong effects for CT concepts and perspectives (Ma et al., 2023). Weintrop et al. (2016) looked specifically at using video games as constructionist environments for teaching CT, finding that giving the player a high-level goal and then leaving them freedom to explore how to reach that goal leads to high engagement with the material, as the players iterate through possible solutions and explore the full range of language features.

This paper is positioned at the intersection of these three bodies of literature. It treats game design decisions as pedagogical choices, examining how mechanics that afford open-ended exploration create constructionist learning environments, and looks at how those environments facilitate computing education through inquiry-based learning.

### 3. Methodology

The methodological foundation of this analysis is hermeneutic inquiry, applied specifically through the technique of close reading. Hermeneutic inquiry is an epistemological tradition “grounded in the expert interpretation of texts from a given point of view” (Tanenbaum 2015). In this tradition knowledge, rather than deriving from an objective measurement of reality, emerges from the balance between the demands of the text and the reader's own situated perspective (Føllesdal 2001).

Close reading is a hermeneutic technique with roots in literary theory, defined as a thorough examination, deconstruction, and interpretation of a media text (Bizzocchi and Tanenbaum 2011). Among its applications is the analysis of a text's poetics, the mechanisms through which a work produces its effects (Tanenbaum 2015). Employing close reading as the primary analytical tool makes it possible to examine how specific mechanics are implemented, situating them within the broader context of the game and exploring in detail how they are constructed and used to support learning.

Close reading is a broadly applicable approach that takes different forms depending on the medium and context to which it is applied. Video games present their own particular challenges and considerations, and dedicated frameworks have been developed to support hermeneutic analysis in this domain (Kłosiński 2022).

Tanenbaum (2015) outlines four key challenges that arise when close reading video games rather than other types of texts: indeterminacy, scope, difficulty, and random access. Given that this analysis centers on game mechanics, difficulty is the most pertinent of these, as players at different levels of skill engage with it in fundamentally different ways. One approach to managing this challenge is to adopt the perspective of a naive interactor: an imagined player encountering the game for the first time, without prior knowledge or experience (Tanenbaum 2015).

A further tool for managing the interpretive demands of close reading games is the use of analytical lenses, defined as deliberate frames a researcher adopts to direct attention toward specific phenomena while setting aside less relevant aspects of the experience (Tanenbaum 2015). Because gameplay generates more information than can be fully analyzed, these lenses serve as filters that bring particular dimensions into focus. Consistent with the aims of this paper, the two lenses guiding the analysis are “computational concepts and practices” and “integration of programming in the core loop”

The two case studies were selected from the 50 games with the highest number of positive reviews with the “programming” tag on the platform *Steam*, and were chosen to demonstrate contrasting design approaches, to capture some of the diversity that exists in the genre. The data collection process combined multiple approaches, with the aim of capturing both immediate responses to the game and more considered reflections, supported by screenshots and recordings to anchor observations in concrete material. The approach draws on the methods employed by Tanenbaum (2015) and Kłosiński (2022). While playing, observations were recorded aloud, and screenshots were taken at regular intervals, as well as whenever a relevant element appeared. After completing each challenge or section of the game, a pause was taken to reflect and record initial impressions, followed by a written summary at the end of each play session. The following day, the notes were revisited for a second round of analysis.

## 4. Analysis

### 4.1 GLADIABOTS

*GLADIABOTS* (2018) is a strategy game developed by GFX47, where the player controls a team of combat robots that face off against another team. Unlike most games in the genre, where players control their units directly during battle, in *GLADIABOTS* the player can use a visual scripting language to program the AI of their robots, who will follow those commands during the match. The game offers three main game modes: training, where the basics of the game are introduced, single-player challenges, where the player fights pre-programmed teams of increasing difficulty, and multiplayer, where the player's team faces off against teams from other players.

The main difference in the play experience this method of using programming in the game creates compared to a maze navigation game is in how the sources of uncertainty shape the experience. In maze navigation games such as *Gidget* or *Lightbot*, the main form of uncertainty is solver's uncertainty (Costikyan 2013), meaning that each level has an optimal solution, and once that is found there is no reason to keep playing. *GLADIABOTS* adds hidden information and player unpredictability as sources of complexity, as the player does not have access to the opponent's programming, neither in single nor multiplayer mode. As a result, rather than having a series of problems each with a distinct solution, playing the game is an experience closer to project-based learning, where the player iterates on a few designs based on their performance without ever reaching a perfect solution.

This iterative, project-based approach is further supported by the way *GLADIABOTS* structures its progression across game modes. Tutorials, single player challenges, and the multiplayer mode, form a natural scaffold for self-regulated learning, allowing players to choose when they are ready to move from guided instruction to open-ended competition. The single-player challenges, with their incrementally harder pre-programmed opponents, give players a relatively controlled environment in which to test and refine their designs before exposing them to less predictable conditions. Multiplayer then introduces an ELO ranking system, which means the difficulty of opposition the player faces adjusts continuously to their current skill level. The result is an adaptive experience that is still perceived as fair and not patronizing, which has been shown to improve learning outcomes (Plass and Pawar 2020).

The ELO system also removes a skill ceiling from the game, as improvement in the player's programming is met with stronger opponents, prompting further iteration. This creates a feedback loop characteristic of adaptive learning environments: performance data (the match result and ranking change) becomes the primary signal driving the next iteration of the code.



**Figure 1: A Match in *GLADIABOTS***

From an educational perspective, the game requires the player to engage with all the concepts and practices of CT individuated by Brennan and Resnick (2012). Some of the CT concepts, such as sequences, events, conditionals and operators are at the core of the game: the block based language used to program the bots is

built around a series of actions the robot can perform, that are evaluated (and executed if true) in a set order. For example, the AI in Figure 2 will flee from the closest enemy if the current shield value is under 50%, attack the closest enemy at short or medium range if not fleeing, or move towards the closest enemy if not attacking. The concepts of parallelism and data are more advanced, and instead emerge from having multiple bots communicate with each other and coordinate to act in parallel.

CT practices, on the other hand, are baked into the design of the game modes: the tutorials often have the player lose a scenario and analyze what went wrong with the code before introducing the concept that will solve the problem, introducing testing and iteration. This practice is further encouraged as, at the end of the tutorial, the player is given starting example AIs that they can use as a baseline to iterate and improve upon. The practices of reusing and remixing existing code and abstracting emerge naturally from the structure of the game: the amount of robots combined with the amount of possible actions each robot can take makes it impossible to solve the game, so challenges do not have a fixed solution the player needs to discover. Rather, the player creates one or more AIs (each bot in the team can have their own custom AI, and it is common in advanced play to have a separated AI for each role) and, using trial and error, remixes the same code they started from each time they attempt a new challenge. The game further supports this with the language feature of sub-AIs, essentially sections of code the player can copy and paste across different AIs. Abstracting as a practice also emerges from the complexity of the game, as it forces the player to adopt a more heuristic approach to problem-solving, especially when it comes to team strategies such as avoiding encirclement or flanking that cannot be coded directly, and instead need to emerge naturally from the interaction between robots.

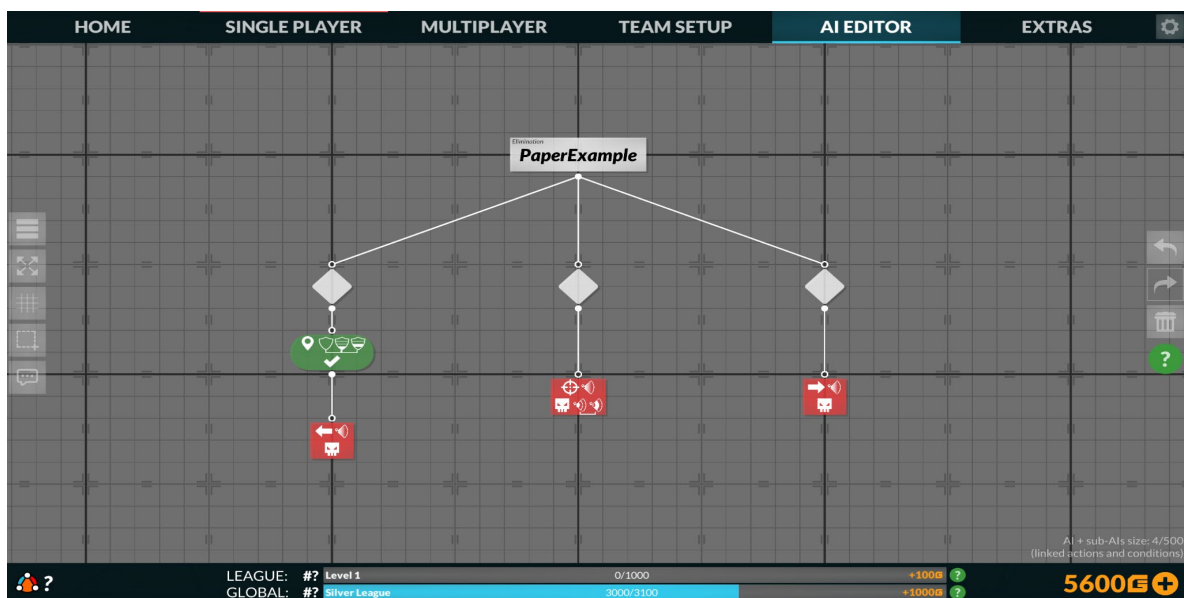


Figure 2: A simple *GLADIABOTS* AI.

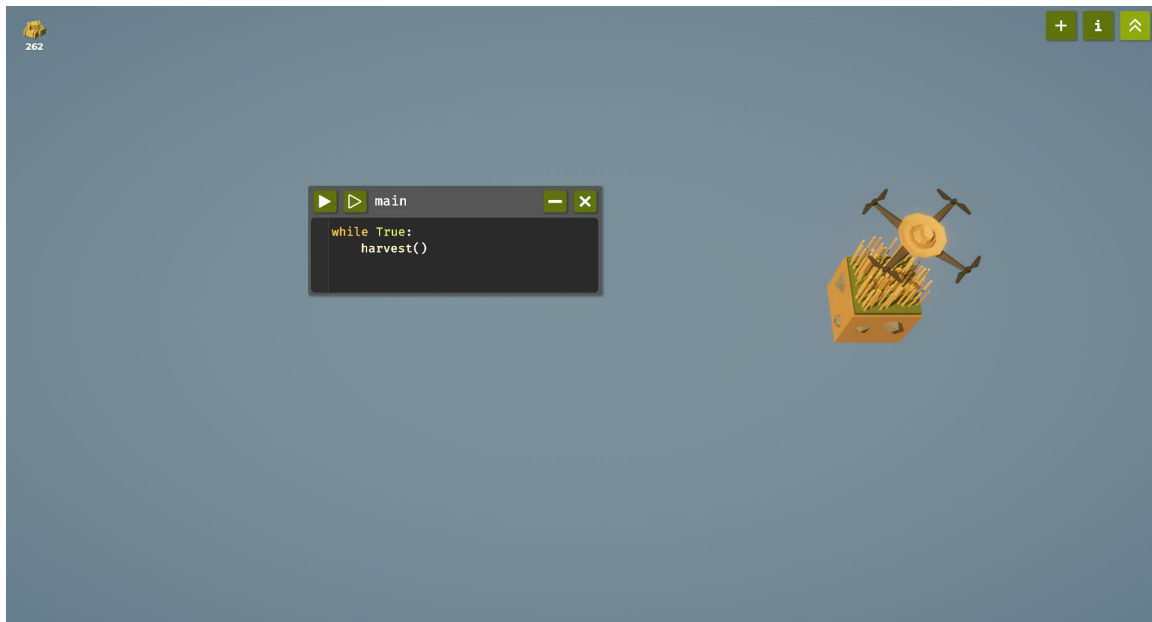
In sum, *GLADIABOTS*' method of incorporating programming as a primary mechanic results in more uncertain and open-ended challenges. The game leverages these characteristics to create an adaptive and self-regulated learning environment where the player can set their own objectives and incrementally move towards them in a mode similar to the one seen in project-based learning, while engaging with foundational CT concepts and practices to test and iterate on their designs.

#### 4.2 The Farmer Was Replaced

*The Farmer Was Replaced* (2025) is an incremental game—a subgenre of resource management games where the goal is to accumulate as many resources as possible (Fiadotau 2024)—about programming a drone to automate a farm developed by Timon Herzog. The programming is done with a Python-like language, customized to make the basic farming tasks easier to achieve. The game does not have levels, and progression comes instead from the upgrade tree, that can be used to unlock new resources and new language features. As the game progresses, new types of plants, each with its own requirements, are unlocked, pushing the player to adapt their code to the new requirements. All direct instruction comes from an info window, containing guides on the game interface, programming concepts and the built-in functions. While the drone collects resources in the background the player can try to optimize the collection algorithm to increase how fast resources are gained,

with the game providing plenty of optional documentation on more advanced topics, complete with examples and code snippets that can be used as inspiration. The game is initially an example of solver's uncertainty, where there is a findable optimal solution to gather the most amount of resources; however it quickly transitions to analytical complexity as its main source of uncertainty, as the farm management becomes too complex for a human to find an optimal solution (Costikyan 2013).

From an educational content perspective, the game covers most topics from the fundamental programming concepts and practices listed in the ACM-IEEE curriculum (Kumar et al. 2024). For example, the core unlocks in the game include functions and arguments, variables, data structures, algorithms, flow control, testing, and debugging, and more advanced upgrades go even beyond that, covering topics classified as more advanced, such as multithreading.



**Figure 3: A simple starting program for *The Farmer Was Replaced***

The upgrade tree directs and funnels the player's progress, with the resources required to unlock new upgrades serving as checkpoints that stop the player from accessing certain features before they have understood foundational concepts. However, the game pushes the player to self-regulate in their learning: some mastery of fundamental concepts is required to advance, but it is up to the player to decide the level of optimization they want to achieve.

What makes *The Farmer Was Replaced* unique as a game for computing education is how the learning content is integrated into the incremental mechanics, and how this results in more advanced topics being discovered naturally by the player with the game's help, rather than being taught directly. Each time a feature is unlocked, it comes with a simple function to engage with it at a basic level, but more advanced crops require the player to repurpose older upgrades to take full advantage of them.

An example of this can be seen in how the concept of an if statement is implemented and expanded upon. The if statement is unlocked as a language feature while the player is still working on a single cell farm, as one of the first upgrades. It becomes available after the player buys the first harvesting speed upgrade for the drone, making the drone harvest grass faster than it can grow. As a result, the game introduces both the if statement and the "can\_harvest()" function, a built-in function that returns true if the crop underneath the drone can be harvested. After this introduction the game moves on to introducing other topics, such as coordinates, for loops, and comparison operators. However, later in the game the player unlocks the carrot crops, that, unlike previous crops, requires the ground to be tilled using the "till()" function to be planted. As attempting to till an already tilled soil will revert it to normal ground and cancel any crop growing on it, the player has to find a way to only till the ground if it has not been tilled before, and plant carrots afterward. The most straightforward way to achieve this is to combine three previous upgrades: the "get\_ground\_type()" function, the comparison operator, and the if construct.

The two upgrades that follows carrots, trees and watering, further increase the complexity of flow control structures required to optimize the harvesting and push the player to experiment with different code configurations. Trees in particular are an interesting example of how the game strategically pairs hints and direct instructions with constructionist principles, and how the player is pushed towards applying the concepts organically by the interaction between different game elements. Trees grow slower when next to each other, but the player does not have access to built-in functions that check the content of other cells. When trees are unlocked, the documentation page that comes up has a hint section that the player can reveal that suggests using the remainder operator in combination with the coordinate system to solve the problem, and has an example code snippet.

*The Farmer Was Replaced*, then, shows how games can use a constructionist learning approach to teach programming by integrating the concepts and practices into the requirements for resource gathering. Furthermore, tying language features behind upgrades encourages the player to build on their previous knowledge, and lets the author of the game pace the experience without constricting player freedom with separate levels.

## 5. Discussion

The analysis highlights how changing the way programming is integrated into the core gameplay loop can influence the way players learn and interface with the content. In both games, programming is not the goal itself, but rather the instrument the player uses to interact with the game. This aligns with the concept of intrinsic fantasy (Malone 1981), as the learning content is embedded in a scenario that is motivating on its own terms, rather than being an explicit subject of instruction.

Both games are also built around a shared constructionist core, where the player is given a high-level goal and a programmable system with which to pursue it, without being given a prescribed solution. As a result, most learning moments are not a result of direct instruction, but rather emerge organically from the interaction with the game systems. This encourages the player to form hypotheses, test them, and iterate based on feedback from the game, key practices in computing education (Brennan and Resnick, 2012, Kumar et al. 2024).

The differences between the games make them complementary examples of how different game mechanics can be applied to achieve the same starting principles of free exploration, player-led pacing and intrinsic motivation. In *GLADIABOTS*, the commands the player has access to remain the same, and complexity rises as the player gains a deeper understanding of the systems and what they can do with them, with exploration and pacing coming from the natural progression given by an ELO system. *The Farmer Was Replaced*, by contrast, uses the pacing of an incremental game and an upgrade tree to introduce concepts progressively, relying on the interplay of its systems to push the player to combine the basic language features into more complex constructs.

These findings suggest several actionable design considerations for educational games. First, programming mechanics are most engaging when embedded within a broader system of goals that give them meaning beyond the code itself. Second, adaptive difficulty can support self-regulated learning and encourage exploration. Third, having complexity emerge from interactions between mechanics can make the game into a constructionist learning environment, fostering exploration and iteration. Rather than designing levels around specific constructs, educational game designers could instead create systems in which the interaction between mechanics naturally creates the conditions for those constructs to become useful, letting the player discover their need and value through play.

## 6. Conclusion

This paper has examined how two commercial programming games, *GLADIABOTS* and *The Farmer Was Replaced*, integrate programming as a core mechanic, and the implications this has for the design of educational games. The main theoretical insight that emerges from this analysis is that game design decisions shape how the game communicates with the player, and that the interplay between mechanics can create a learning environment that affords effective computing education by encouraging free experimentation by presenting problems without clear, predetermined solutions.

Future research could build on this analysis by conducting empirical studies with learners to assess the effectiveness of this constructionist approach. Additionally, expanding the case study to include a broader selection of commercial programming games could surface further design strategies, and provide a more comprehensive foundation for translating the strengths of commercial game design into educational contexts.

**Ethics Declaration:** Ethics clearance was not required for this research.

**AI Declaration:** No AI tools were used for this paper.

## References:

- Apiola, M. *et al.* (2023) "Exploring the Past, Present and Future of Computing Education Research: An Introduction," in M. Apiola, S. López-Pernas, and M. Saqr (eds.) *Past, Present and Future of Computing Education Research : A Global Perspective*. Cham: Springer International Publishing, pp. 1–7.
- Bizzocchi, J. and Tanenbaum, T.J. (2011) "Well read: applying close reading techniques to gameplay experiences," *Well played 3.0: video games, value and meaning*. Pittsburgh, PA, USA: ETC Press, pp. 262–290.
- Blanco, A.A. and Engström, H. (2020) "Patterns in Mainstream Programming Games," *International Journal of Serious Games*, 7(1), pp. 97–126.
- Branthôme, M., (2024) "Pirates: Design and Evaluation of a Serious Game Aimed at Introducing Python Programming and Easing the Transition from Blocks," *ACM Transactions on Computing Education*, 24(1), pp. 1–24.
- Brennan, K., & Resnick, M. (2012). "New Frameworks for Studying and Assessing the Development of Computational Thinking". Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vol. 1.
- CodeCombat Inc. (2024). *CodeCombat* [video game].
- Costikyan, G. (2013) *Uncertainty in Games*. Cambridge, MA, USA: MIT Press.
- Dagienè, V. *et al.* (2023) "Computing Education Research in Schools," in M. Apiola, S. López-Pernas, and M. Saqr (eds.)
- Daniels, M. *et al.* (2023) "What is Computing Education Research (CER)?," in M. Apiola, S. López-Pernas, and M. Saqr (eds.)
- Fiadotau, M. (2024) "Incremental Games," *Encyclopedia of Computer Graphics and Games*. Springer, Cham, pp. 939–941.
- Føllesdal, D. (2001) "Hermeneutics1," *The International Journal of Psychoanalysis*, 82(2), pp. 375–379.
- Gee, J.P. (2003). What video games have to teach us about learning and literacy. *Comput. Entertain.* 1, 1, 20.
- GFX47 (2018). *GLADIABOTS - AI Combat Arena* [video game]. WhsiperGames.
- Guardiola, E. (2016) "The Gameplay Loop: a Player Activity Model for Game Design and Analysis," *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*. New York, NY, USA: Association for Computing Machinery (ACE '16).
- Herzog, T. (2025). *The Farmer Was Replaced* [video game]. Metaroot.
- Hung, H.-T., Yang, J.C., Hwang, G.-J., Chu, H.-C. and Wang, C.-C. (2018). "A Scoping Review of Research on Digital Game-Based Language Learning". *Computers & Education*, 126, pp.89–104.
- Kebritchi, M., Hirumi, A. and Bai, H. (2010). "The Effects of Modern Mathematics Computer Games on Mathematics Achievement and Class Motivation. *Computers & Education*, 55(2), pp.427–443.
- Kłosiński, M. (2022) "How to Interpret Digital Games? A Hermeneutic Guide in Ten Points, With References and Bibliography," *Game Studies*, 22(2).
- Kumar, A.N. *et al.* (2024) *Computer Science Curricula 2023*. New York, NY, USA: Association for Computing Machinery.
- Lee, M.J. (2014). *Gidget* [video game].
- Lightbot Inc. (2008). *Lightbot* [video game].
- Overono, A., & Storm, B. (2015). "Harnessing the Power of Uncertainty to Enhance Learning". In *Translational Issues in Psychological Science*, 1, 140–148. <https://doi.org/10.1037/tps0000022>
- Papert, S.A. (1993) *Mindstorms*. Available at: <https://www.media.mit.edu/publications/mindstorms/>
- Plass, J.L. and Pawar, S. (2020) "Adaptivity and Personalization in Game-Based Learning". In *Handbook of game-based learning*. Cambridge, MA, US: The MIT Press, pp. 263–281.
- Plass, J. L., Homer, B. D., Mayer, R. E., & Kinzer, C. K. (2020). "Theoretical Foundations of Game-Based and Playful Learning". In J. L. Plass, R. E. Mayer, & B. D. Homer (Eds.), *Handbook of game-based learning*. The MIT Press, pp. 3–24
- Tanenbaum, T.J. (2015) "Hermeneutic Inquiry for Digital Games Research," *The Computer Games Journal*, 4(1), pp. 59–80.
- Taub, M. *et al.* (2020) "Self-regulation and reflection during game-based learning," *Handbook of game-based learning*. Cambridge, MA, US: The MIT Press, pp. 239–262.
- Videnovik, M., Vold, T., Kjønnig, L. *et al.* (2023). "Game-Based Learning in Computer Science Education: a Scoping Literature Review". *IJ STEM Ed* 10, 54.
- Wanglang, C., Sraubon, K. and Piriyaawong, P. (2024) "Combining Game-Based Learning with Design Thinking Using Block-based Programming to Enhance Computational Thinking and Creative Game for Primary Students," *Higher Education Studies*, 14(2).
- Weintrop, D. *et al.* (2016) "Computational Thinking in Constructionist Video Games," *International Journal of Game-Based Learning*, 6(1), pp. 1–17.
- Wing, J.M. (2006) "Computational thinking," *Commun. ACM*, 49(3), pp. 33–35.
- Xu, Z., Chen, Z., Eutsler, L. *et al.* (2020). "A Scoping Review of Digital Game-Based Technology on English Language Learning". *Education Tech Research Dev* 68, 877–904.
- Zubek, R. (2020) *Elements of Game Design*. Cambridge, MA, USA: MIT Press.