Gamification in Teaching Computational Thinking: Enhancing Engagement and Learning Outcomes in K-12 Education

Dr. Virender Kumar Chandoria, Assistant Professor

Department of Education, University of Allahabad, Uttar Pradesh, India.

Dr. Pooja Singh, Assistant Professor

CTE-Nuh, Maulana Azad National Urdu University, Hyderabad

Mr. Sumit Kumar Singh Chauhan, Scholar,

Department of Education, University of Allahabad,

Uttar Pradesh, India.

Ms. Surbhi Pal, Scholar,

Department of Education, University of Allahabad, Uttar Pradesh, India

Abstract;The integration of gamification in K–12 education has become the essential tool to enhance student engagement and effectiveness of the learning outcomes. This paper discusses the contribution of gamification to the process of teaching Computational Thinking(CT), considering an educational setting for K–12. Generally, it is the application of problem-solving skills and techniques that includes such thinking as algorithmic abstraction and decomposition. This paper explores how GBL strategies can promote the development of computational thinking, enhance student engagement and achieve better academic performance.

To highlight the benefits of gamification for both students and educators, a review of existing literature, case studies and examples from classroom practice is included. The challenges in the effective implementation of gamification in teaching computational thinking and their potential solutions are also discussed.

Keywords: Gamification, K-12, SDT, Algorithm-Thinking.

1. Introduction

With the rapid progress of technology and the interest in digital literacy, it has now become an essential 21st-century skill for students. "Computational thinking" refers to the problem-solving methods inspired by computer science that include decomposition of complex problems into smaller parts, detection of patterns, abstraction of concepts to make problems simpler and development of algorithmic solutions. These skills, though deepening computer science understanding, are broadly applicable across disciplines and so should feature prominently in contemporary education. That being said, integrating CT into K-12 curricula isn't without its challenges, especially as to how one might get younger learners excited about abstract concepts. It has proven intriguing to apply game design elements like challenges, rewards and progression to non-gaming contexts this process commonly referred to as gamification. Gamification is a technique of changing the process of education so that it will be fun, interesting and engaging for students. Gamification improves the learning outcomes, motivation and engagement. This paper shall consider the theoretical frameworks on its application, review the relevant literature on the combination and interaction between the two elements and provide specific implementation recommendations while exploring how gamification may help teachers teach and learn CT in K-12.

2. Theoretical Framework

2.1 Gamification in Education

According to Deterding et al. (2011), Gamification refers to adding elements like points, badges, leaderboards, levels and rewards to non-game contexts that help in increasing the engagement level of the users and increase their motivation levels. Thereby, in this education setup, Gamification is said to be found as the most powerful method for keeping learning exciting, interactive and enjoyable. This system has proven to work best because traditional methods are perceived by people as either dull and boring or repetitive and monotonous (Kapp, 2012). Gamification injects elements of play into the learning process and, therefore, encourages feelings of achievement, competition and momentum toward goals, making it a more engaging learning process. The theoretical basis for gamification in education is based on self-determination theory and flow theory. Self-Determination Theory (SDT), as conceptualised by Ryan and Deci, 2000, defines the importance of intrinsic motivations in learning. People tend to like what they are doing according to SDT, in so far as the underlying psychological needs for autonomy, relatedness and competence are met. These are variously met by gamification: For instance, the features of autonomy would be realised in how learners can take decisions regarding their learning path or goals for the gamified systems. The existence of such features would allow the development of competence: how well progress is tracked, giving the right kind of feedback and reward mechanisms. This makes students capable enough to monitor observable improvements in their abilities. And lastly, the concept of relatedness is taken into consideration when gamified systems feature competitive or cooperative aspects like challenge-related team-based activity or scores that help the learners make connections with other people in this class.

The second mainstream theoretical foundation of gamification is flow theory, as proposed by MihalyCsikszentmihalyi in 1990. As for the flow theory, learning and engagement are maximised among the students when they are in their flow state—a cognitive state that is characterised by a comprehensive immersion, concentrated attention to the task and at the same time, a balance of one's skills and challenges at hand. Gamification achieves the flow in learning because tasks for learning are so designed and at proper levels of difficulty. The learners neither have to fight too complicated challenges nor get bored with too simplistic tasks. These gamified systems also give prompt feedback and incremental rewards to the students. This further bolsters motivation and engagement by the learners. The researchers demonstrated in various experiments that it is possible to facilitate some learning outcomes using gamification. For instance, Hamari, Koivisto and Sarsa (2014) showed that gamification augments the motivation, participation and performance of students in several learning situations. However, they pointed out further that gamification effects are context-dependent, strongly depending on design quality as well as the relevance of gamified elements for the needs of the learners. Therefore, those teaching must take into great consideration these variables when their students experience gamified learning.

2.2 Computational Thinking in Education

Computational thinking (CT) is a basic 21st-century skill that is solving problems in a systematic and logical way, much like how computer scientists do. As Wing (2006) has pointed out, CT goes beyond computer science, acting as a universal framework to address problems across disciplines. The four core components of C.T. are decomposition, pattern recognition, abstraction and algorithmic thinking, which allow learners to break down complicated challenges, identify trends and come up with efficient solutions. For example, decomposition

breaks down problems into easier, more manageable parts, as shown in mathematics by step-by-step solving of equations or in engineering by designing complicated systems through the solution of sub-components (Shute et al., 2017). Pattern recognition helps determine patterns and trends, an essential function in data science, biology and literature as these fields analyse recurring motifs or themes to find insights (Brennan & Resnick, 2012).

Both abstraction and algorithmic thinking, equally critical in simplification and solution finding, involve ignoring some of the unimportant details while drawing on what matters most so as to design models or algorithms addressing large classes of problems. In the real sense, how the navigation apps have implemented abstracted algorithms for deriving routes between two points within various locations would give one a better indication (Wing, 2008). Algorithmic thinking, on the other hand, involves writing detailed step-by-step instructions. It is therefore fundamental not only in programming but also in everyday tasks such as cooking or planning (Grover & Pea, 2013). These skills are the backbone of computational thinking, which enhances critical thinking and problem-solving skills necessary in a digital and data-driven world. The importance of CT to educators has made them integrate CT into K-12 curricula, where students should be equipped with the knowledge to adapt to emerging technologies and succeed in diverse professional fields (Barr & Stephenson, 2011).

Despite the challenges in the form of teacher training and requirements of resources, the integration of CT in education outweighs such hurdles. Studies have shown that exposing children to CT early is very effective and helps to increase

cognitive skills, creativity and teamwork (Bers et al., 2014). Moreover, it provides a pathway to getting programming and coding skills in high demand among the workforce in the present times. However, if such curricular designs are to be fully integrated, they have to harmonise, with much more convention, computational thinking with other typical subjects of mathematics, science and the humanities, as pointed out by Yadav et al. (2014). By encouraging decomposition, pattern recognition, abstraction and algorithmic thinking, CT endows learners with problem-solving capabilities, adaptation toward speedy technological change and innovations across disciplines. Ultimately, computational thinking empowers people to solve problems in a systematic way and succeed in an ever-changing, technology-driven world.

3. Gamification and Computational Thinking: A Synergistic Approach

The integration of gamification in teaching computational thinking gives a lot of synergy in terms of enhancing pupil engagement and critical thinking skills simultaneously. Computational thinking is regarded as the decomposition of challenging problems into manageable parts through recognition of patterns, then abstraction of salient points and construction of solution steps that are the algorithms developed for solving problems (Wing, 2006). Since gamification techniques can provide dynamic learning environments, they can be very widely applied to the teaching of abstract ideas such as these. This section goes a bit deeper into how particular elements of gamification align with the core components of CT and how game mechanics can further develop CT skills in K–12 education. The following subthemes illustrate aspects of computational thinking as applied in gamification—the enhancement each can offer by giving learners an interactive fun learning process.

3.1. Encouraging Problem Decomposition

Decomposition of complex problems is a significant aspect of computational thinking; the segmentation of tasks into smaller, more manageable subtasks allows for easier handling (Wing, 2006). By default, the gamified learning environment shows this as it has levels and structured challenges. Contrary to educational environments, which often seem abstract and difficult to decompose, the gamified environment enables decomposition by forcing the students to break down tasks into logical steps to move forward. For instance, in CodeCombat, users must write codes to solve puzzles, which continually grow in complexity as they learn to break large problems into smaller tasks in coding. For example, Minecraft Education Edition challenges can be as simple as constructing buildings or solving puzzles, for which a plan and breaking it down into smaller steps must be applied (Holmes, 2016). These iterative processes teach not only decomposition strategies but also experience in problem-solving. More importantly, gamification rewards and progression systems reflect the importance of decomposition; they provide feedback that would have otherwise focused on mastery of skills and a sense of achievement (Anderson &Rainie, 2012).

3.2. Promoting Pattern Recognition

The main objectives of pattern recognition, which is one of the parts of computational thinking, include finding similarities, repeating patterns and structures that could make solving a problem easier (Wing, 2006). This could be anything from identifying repeating patterns in games to figuring out the logic behind mechanisms in a game. This naturally encourages pattern recognition through the challenges in the form of puzzles, chores, or opponents that display recurring patterns across levels on a gamified learning platform. The challenges are more complex, allowing students to look for and use patterns. Students might use Scratch to build games in which characters repeatedly do specific behaviours. This

calls for the identification and use of patterns in the code that carries out those activities (Resnick et al., 2009).Like a puzzle, Tynker also uses coding puzzles as tools to help learners distinguish and optimize repetitive patterns used within loops. Finding and using such patterns will lead to designing improved algorithms; therefore, the problem will be addressed even faster intuitively. Loops and conditions, too, emphasise it more strongly in integration skill-boptimiseuilding of what gamers go through (Lorenzo et al., 2017). Pattern recognition therefore is more of a creative exercise and independent thinking in the minds of students when they have to identify different relationships between game or coding elements. The skills also transcend gaming since they now enable the student to make sense of trends in data or to analyse connections in mathematics as a problem-solving tool for scientific problems. Here, pattern recognition goes beyond serving as a boost to computational thinking and instead prepares it for broader applications in life.

3.3. Fostering Abstraction

Abstraction is the process of removing complex problems by focusing on the important details and, at the same time, ignoring unnecessary information (Wing, 2006). In computational thinking, abstraction enables students to establish general solutions that can apply to many situations. In gamified learning environments, abstraction is best practiced due to the opportunities given to students to engage with a complex problem in a reduced but challenging manner. For instance, in Minecraft Education Edition, the student is encouraged to solve problems within the game world by abstracting away extraneous details. If a student is tasked with designing a redstone-powered machine, they must focus on the core functions of the machine (e.g., power source, activation, mechanisms) while ignoring unrelated environmental elements (such as scenery or terrain). By this, students acquire the

ability of abstraction, which will define what elements one needs to attain a required outcome while filtering out non-relevant noise (McGonigal, 2011).

Other examples of gamified contexts in teaching abstraction include the games of RoboZZle and LightBot, in which the students must give step-by-step instructions on how to solve certain problems, but only the core steps are required to successfully complete them. This forces students to abstract away unnecessary details and focus on the core logic of the problem, thus reinforcing computational thinking strategies (Lee et al., 2014). Abstraction is eased through gamification as problems are made simpler in their presentation. For example, the visual aspect of a game allows a student to notice the essential parts of a problem without the noise that comes with extra information. Tools such as CodeCombat can enable a student to learn the core structures of loops and conditionals without fear of syntax errors, freeing the student to focus on the abstract logic of the program rather than technicalities (Gee, 2003).

3.4. Encouraging Algorithmic Thinking

Algorithmic thinking is the ability to develop systematic procedures or algorithms to solve problems. This is possibly the area of computational thinking that is most closely related to gamification, as many games need players to develop or follow algorithms in order to complete tasks, advance through levels, or optimise their performance (Wing, 2006). Students are pushed to create algorithms in gamified settings, sometimes without even recognising that they are using algorithmic thinking. Students must develop algorithms to tackle more challenging issues in embedded coding challenges seen in games like CodeCombat and Tynker. Every task builds on the one before it, emphasising the necessity for students to create

effective algorithms and debug them as needed (Lorenzo et al., 2017). For example, in Tynker, a student might begin by coding a simple movement control, but then he needs to advance to more complex algorithms to better understand logic and flow control as he progresses.

In Scratch, students design interactive animations and games, which involves writing algorithms to control characters, manage game logic and determine outcomes. Creating these algorithms teaches the student not only how to think step by step but also how to optimise those steps to ensure efficiency and effectiveness (Resnick et al., 2009). Another enhancement to algorithmic thinking in gamified systems is rewards and progression. As the students move from one level to another or collect achievements, they are always implementing and perfecting their algorithms. Games are iterative processes, which means the players keep on repeating the actions and improving their strategies, similar to how an algorithm is refined in programming (Anderson & Rainie, 2012).

In some of the games, algorithmic thinking extends to strategic decision-making too. In Minecraft Education Edition, for example, the students could design systems or solve puzzles that require the implementation of algorithms to reach a specific goal, say automating a process using redstonecircuitry. It makes students think logically, breaking tasks into manageable steps and hence it encourages the practical application of designing an algorithm in the simplest of ways (Holmes, 2016).

3.5. Enhancing Engagement and Motivation

The secondary benefit is a boost in engagement and motivation, as the primary intent of gamification in teaching computational thinking is skill development. The gamification touches upon students' natural instincts of play, achievement and

recognition. A gamified environment motivates students to spend time and effort solving problems and learning their skills in computational thinking through rewards, levels, badges and leaderboards (Gee, 2003). Therefore, students cannot concentrate well in a traditional school environment because of the abstraction of ideas in computational thinking. The gamified system, however, transforms learning into an engaging process and success is manifested in concrete rewards and achievements. Instant feedback, achievements and progression systems are provided in order to motivate students in the pursuit, for these give a feeling of accomplishment and further propel them in learning (Anderson & Rainie, 2012).

In addition, gamification encourages teamwork or competition, which can be beneficial for social learning. For example, in Minecraft Education Edition, students work together to solve difficult problems, share ideas and discuss ways, thus developing communication and teamwork skills (McGonigal, 2011). Students who are not interested in traditional approaches may become more interested in the curriculum due to the collaborative nature of gamified settings.

3.6. Fostering a Growth Mindset

One of the largest benefits that gamification may provide in teaching computational thinking is a growth mentality. A growth mentality is the belief that intelligence and skill may be enhanced through diligence and persistence (Dweck, 2006). Ga mification helps students view failure as an essential part of learning by providing frequent feedback loops, rewards and opportunities for several tries. Failure sometimes happened to be success, as in the world of games. Failure can mean retrying; not understanding something leads to solving, completion or even getting beyond one level. That problem-solving cycle is part and parcel of

computational thinking. The benefit is discovered here since refining involves assessing the strategy and solution through reflection, which leads to self-reflection (Dweck, 2006). Gamified learning environments foster a culture of trying and skill development because they provide a safe space for experimenting and failure. Given that students learn not only technical skills but also resilience and tenacity in the face of difficulty, this approach to teaching computational thinking may prove to be quite effective (Gee, 2003).

4. Case Studies and Examples

Examples for noteworthy applications and case studies, where gamification had proven successful in the field of teaching computational thinking among K-12 education communities are further elaborated. They represent how gamification helps engage students in different settings while improving their awareness regarding computational thinking in interesting interactive game-based environments. Examples include the following, all explored below.

4.1. CodeCombat: A Game-Based Learning Platform for Coding

CodeCombat is an innovative learning platform that teaches students programming and computational thinking in an immersive, gamified environment. It combines coding lessons with the RPG format, where students use real programming languages, such as Python, JavaScript and others, to control a character in the game. Each level of the game introduces new programming concepts, such as loops, conditionals, functions and arrays, which are the main constituents of computational thinking.

The platform uses a set of increasingly difficult challenges and requires students to write code to solve problems and advance through the game. As students progress, they are given immediate feedback about their code and rewards and badges can be

earned; this encourages them to learn more. Code Combat also has strong collaboration elements, allowing teams to work together to solve coding problems and advance through quests together, which helps create a sense of community and shared learning.

The gamified structure of Code Combat enables the student to learn programming concepts but also helps him or her develop computational thinking skills such as algorithmic thinking and decomposition of problems. This kind of skill, where complex coding challenges are broken down into smaller parts, is basically fundamental to computational thinking. Moreover, the level progression system of the platform will be encouraging students to continually improve, which provides a sense of achievement as they advance through the game.

4.2. Scratch: A Visual Programming Language for Creative Coding

Developed at MIT Media Lab, Scratch probably is the most widely used among all platforms teaching K–12 students about computational thinking? Scratch is a visual language for programming through which students are able to create interactive stories, games and animations using blocks of code. The platform was especially designed to benefit the young learners so that they may express creativity, collaborate with others and solve problems. With Scratch, students can play around with computational concepts like loops, conditional statements and variables in a non-threatening environment that is more gamelike. This reduces the syntax for students, so they will be more concerned with logic and the structure of their programs than with the syntax. Thus, through a trial and error process, students can learn to formulate their algorithms and solve the problems through this design. Because their creative work will be divided into smaller manageable

jobs with each having their own instructions, Scratch fits well in abstraction and decomposition. For example, game design is a process of decomposition into the following parts: character movement, user interaction, scoring and abstractions of rules and mechanics. In addition, the process of sharing work with other people, getting feedback from them and collaboration of students among themselves in their projects within the Scratch community further enhance the learning process. Social aspects of Scratch, that is, aspects of collaboration and communication, are necessary for computational thinking.

4.3. Minecraft Education Edition: Building Computational Thinking through Exploration and Problem-Solving

K-12 classrooms employ Minecraft Education Edition, a popular instructional version of the popular sandbox game, to teach STEM subjects, including computational thinking. Minecraft encourages creativity, exploration and problem-solving. In the Education Edition, students may work on a range of projects, such as building structures or coding in-game features using block-based programming languages like JavaScript or Make Code. One of the main ways Minecraft develops computational thinking skills is by encouraging problem-solving and critical thinking. Often, they are tasked with designing very complex structures, solving a variety of puzzles, or fulfilling certain challenges that force the students to think more strategically and divide the problem into smaller tasks. Such exercises teach skills like algorithmic thinking because students need to establish step-by-step solutions that would enable them to obtain their objectives in the game.

Minecraft Education Edition offers lessons and activities that focus on teaching computational thinking. For instance, students can make use of the game's coding features to create automated systems such as red stone circuits (functioning like an electrical circuit) or programming of NPCs to perform some specific action. These activities need the use of abstraction and deconstruction in order to build a working system from the ground up. Additionally, being an open-ended, sandbox game, Minecraft lets youngsters explore their creativity by letting them design and test their own ideas. This enables people to use computational thinking for a variety of activities, such as creating a historical monument or creating a natural environment simulation.

4.4. Tynker: Teaching Coding through Gamification and Storytelling

One popular tool to teach K-12-aged students computer programming and thinking about algorithms is Tynker. Tynker is more or less identical to the Scratch application described in which students create games, animations and stories in a graphic programming environment through the mechanism of dragging and dropping coding blocks. In a final twist, however, the curriculum of Tynker becomes more interactive using components based on gamification techniques. One of the advantages of Tynker is that it uses the narrative method as a basis for teaching coding. Students are encouraged to create their own interactive stories and games that require the use of computational thinking techniques such as algorithmic reasoning, pattern identification and deconstruction. For example, when designing a game, students have to break it down into its mechanics: planning character behavior and developing logically correct sequences of events that will flow to drive the gameplay of a game.

Tynker also provides many coding challenges and puzzles in different levels of difficulty. The problems are designed to encourage problem-solving skills since the student needs to be creative and use computational thinking techniques to find their way through each challenge. Reward and level-based gamification elements provide instant feedback, so students experience a feeling of accomplishment when they master a new coding skill.

4.5. RoboZZle: A Puzzle Game to Teach Algorithmic Thinking

RoboZZle is an online puzzle game where, by solving more complex and complex puzzles with the aid of a robot, a child learns the very basics of programming and how to think algorithmically. The game is structured as a series of missions in which the player has to produce a list of instructions or an algorithm telling the robot how to get something or where to get, for instance. The most educational value in RoboZZle lies in its algorithmic thinking aspect. Players have to create efficient algorithms to solve the puzzles that require loops, conditionals and recursion. At a later stage of the game, problems are more complex and demand decomposition, pattern recognition and abstraction. RoboZZle is really effective in teaching recursive thinking, which is an important ability in both computational thinking and computer science. The puzzles of the game encourage students to think step by step in solutions that may be repeated to solve similar problems, making it an excellent tool for promoting the development of algorithmic thinking.

The game also has the social part: players can publish their solutions with others and help in working through puzzles and solving the challenges. In this feature, there can be some elements of community, which requires communication, an important set of skills, as children work to understand computational thinking challenges.

4.6. LightBot: A Game for Teaching Logic and Programming

LightBot is a type of puzzle game designed in such a way that introducing students to basic programming logics and computational thinking comes into play. The entire game is all about following a robot in different progressively challenging levels with the command of simple moves through an interface. Basic programming principles like sequencing, loops and conditionals need to be understood by players while gaming.

LightBot inspires students to become algorithmic thinkers by allowing them to break down big problems into smaller steps; they see patterns in a robot's movements. Using this application is very smooth and not complicated, nor do its challenges increase steeply; the difficulties are built cumulatively. This is because LightBot has an immediate feedback and reward structure that keeps the students interested. The visual design and interactive nature of the game also make it fun for the students, so they are motivated to work their way through the levels.

5. Challenges and Considerations in Implementing Gamification for Computational Thinking

Although gamification comes with great benefits for learning computational thinking, its usage is associated with some known challenges. These include: resource intensiveness of education content when gamified; the balance between game and educational objectives; and maintaining inclusivity and accessibility of the tools.

Resource Intensity

The creation and upholding of gamified educational products require substantial resources, most of which are in technology, expertise and time. Schools may struggle to access the necessary tools and training due to budgetary and logistical constraints (Dichev&Dicheva, 2017). Strong software platforms are required in gamification, but an interesting and practical learning environment also requires regular upgrades and upkeep. In addition to this, teachers also have to receive extensive training that will enable them to introduce gamified techniques into their teaching. Without adequate tools and support, any gamification efforts are liable to fail, which limits their ability to foster computational thinking in the long term.

Keeping Learning Objectives and Game Mechanics in Balance

It's crucial to strike a balance between the primary learning objectives and captivating game elements. Students may become more focused on playing to win rather than internalising the necessary computational thinking when game elements like incentives, levels, or contests seem to take centre stage (Hamari et al., 2014). For example, this may make a student score more badges or get the most scores but not engage enough with the principles of an algorithm or problem solving; therefore, educators should know that game mechanics aligns with intended learning outcomes. This challenge will be overcome by the design of games that harmonise with the integration of computational thinking into the game, as it allows the game to be more educational than just a pastime.

Accessibility and Inclusivity

Accessibility and inclusivity constitute another issue of gamification in learning. A game-based learning platform has to adapt to all students of different skills,

different cultural backgrounds and various means of technological access (Gee, 2013). Inclusive game design, then, should incorporate adjustable difficulty levels, multi-language support and accessibility options for students with disabilities. Visually challenged students may require text-to-speech capabilities and students with mobility impairments may require alternative control strategies. Socioeconomic differences may affect some students' access to devices and internet capabilities, thereby preventing them from participating in gamified activities. These challenges require a focused and equitable approach to game design and execution so that all students can participate in gamified computational thinking assignments.

6. Conclusion

A gamified approach presents itself as an effective means of improvement of the teaching of computational thinking in K–12. In the learning setting, reward-giving, challenges that would trigger engagement and making continuous progress can significantly promote motivation, engagement and improvement of students. With effective application of gamification, it shall allow the students to train such important skills as pattern recognition, decomposition of the problem into easier solutions, abstraction and algorithmic thinking. However, educators need to be very careful about the difficulties that come with gamification, such as resource constraint and the balance between game mechanics and educational content. With such thoughtful implementation, gamification will be a powerful tool to foster computational thinking and prepare students for the future.

References

- Anderson, J. Q., &Rainie, L. (2012). The future of gamification. *Pew Research Center*.https://www.pewresearch.org
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.https://doi.org/10.1145/1929887.1929905
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.https://doi.org/10.1016/j.compedu.2013.10.020
- 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*.
- Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. Harper & Row.
- Deterding, S., Dixon, D., Khaled, R., &Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification." *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 9–15.https://doi.org/10.1145/2181037.2181040
- Dichev, C., &Dicheva, D. (2017). Gamifying education: What is known, what is believed, and what remains uncertain. *International Journal of Educational Technology in Higher Education, 14*(1), Article 9.https://doi.org/10.1186/s41239-017-0042-5
- Dweck, C. S. (2006). *Mindset: The new psychology of success*. Random House.

- Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment, 1*(1), Article 20.https://doi.org/10.1145/950566.950595
- Gee, J. P. (2013). *The anti-education era: Creating smarter students through digital learning.* Palgrave Macmillan.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.https://doi.org/10.3102/0013189X12463051
- Hamari, J., Koivisto, J., &Sarsa, H. (2014). Does gamification work? A literature review of empirical studies on gamification. *Proceedings of the 47th Hawaii International Conference on System Sciences,* 3025–3034.https://doi.org/10.1109/HICSS.2014.377
- Holmes, W. (2016). The use of Minecraft in education: A review of the literature. *International Journal of Information and Education Technology, 6*(9), 684–688.https://doi.org/10.7763/IJIET.2016.V6.825
- Kapp, K. M. (2012). *The gamification of learning and instruction: Game-based methods and strategies for training and education.* Pfeiffer.
- Lee, J. S., Zhai, X., & Hong, S. (2014). Gamifying education with RoboZZle: A social puzzle game for teaching algorithmic thinking. *Proceedings of the International Conference on Educational Technologies,* 89–92.https://doi.org/10.1109/ICET.2014.39
- Lorenzo, M., Wilson, G., & Shipman, J. (2017). Tynker: Coding for kids. *Computer Science Education Journal, 28*(2), 103–115.https://doi.org/10.1080/08993408.2017.1349812
- McGonigal, J. (2011). *Reality is broken: Why games make us better and how they can change the world.* Penguin Press.

- Resnick, M., Maloney, J., Monroy-Hernández, A., & Rusk, N. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.https://doi.org/10.1145/1592761.1592779
- Ryan, R. M., &Deci, E. L. (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
- Shute, V. J., Sun, C., &Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.https://doi.org/10.1016/j.edurev.2017.09.003
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.https://doi.org/10.1145/1118178.1118215
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.https://doi.org/10.1098/rsta.2008.0118

☆☆☆