GAME DESIGN, PROBLEM SOLVING AND PROBABILISTIC THINKING: AN EXPLORATORY STUDY

Uwe Schürmann[1], Georg Bruckmaier[1,2], Alexander Repenning[1,3]

[1]University of Applied Sciences and Arts Northwestern Switzerland,
[2]University of Basel, [3]University of Colorado Boulder

This exploratory study investigates learning processes related to problem solving and probabilistic thinking through the programming of computer games with random elements – based on Seymour Papert's constructionist approach. The data consist of video recordings of programming phases and video-stimulated recall interviews with four student pairs (Year 6) from two Swiss primary schools. The results show cyclical design processes reflecting key components of computational thinking. Probabilistic preconceptions emerge and partly develop during the game design process.

INTRODUCTION

Since Wing's (2006) call to establish *Computational Thinking* (CT) in general education, the concept has gained relevance in curriculum development. CT is, in short, the ability to solve problems systematically using computer science principles, supporting cognition across domains, including mathematics. Integrating CT into mathematics education opens new avenues for conceptual learning through creative activity (Kallia et al., 2021). The concept has roots in Papert's (1986) constructionist theory, which emphasises learning through creating digital artefacts. Programming becomes a means of epistemic exploration and is not an end in itself, but a means to help students engage meaningfully with mathematical ideas (Papert, 1996).

Game Design (GD) provides a setting where CT and mathematical thinking converge. It involves CT practices such as decomposition, abstraction, iteration, and debugging (Shute et al., 2017), which align with mathematical problem solving processes—namely, recognising a problem, devising a strategy, executing the solution, and monitoring and evaluating the outcome (Schoenfeld, 1992).

Furthermore, GD can foster probabilistic reasoning when learners define or adjust chance-based parameters. Designing and testing games with randomness exposes students to probabilistic variation. A self-programmed game thus offers a meaningful context for exploring chance and likelihood, supporting Papert's idea of *learning by making*. Building on these ideas, the present study examines how learners engage with problem solving and probabilistic thinking during GD in a constructionist digital environment.

2026. In H. Krzywacki, M. Hähkiöniemi, E. Haataja, M.S. Hannula & P. Hästö (Eds.). *Proceedings of the 49th Conference of the International Group for the Psychology of Mathematics Education: Research Reports*, pp. 1315-1322. PME.

THEORETICAL BACKGROUND

Computational Thinking as Problem Solving

Given diverse definitions of CT, this study adopts a functional one: CT is conceptual thinking required to solve problems effectively and efficiently—algorithmically, with or without computers—to develop solutions reusable across contexts (Shute et al., 2017). CT and programming are not synonymous; CT refers to cognitive strategies like reformulating problems, recursive solution building, decomposition, abstraction, and error correction (Wing, 2006). This can lead to systematic procedures that support automation. CT is thus applicable even without computer use, while programming implements such thinking.

Various structural models describe CT in programming. Classical approaches follow a top-down path: decomposition → abstraction → pattern recognition → algorithm design. Others view programming as iterative and cyclical—planning, implementing, testing, evaluating—with creative, exploratory aspects akin to tinkering (Resnick & Rosenbaum, 2013; Woo & Falloon, 2022).

Computational Math Thinking Using Probability as an Example

Integrating (computer) technology aims to make mathematics education more creative, exploratory, and constructionist (Papert, 1996). We refer to Computational Math Thinking (CMT) as combining CT with mathematical content and reasoning. CMT involves analysing, modelling, and solving mathematical problems using computational or algorithmic methods (Weintrop et al., 2016).

The topic *Data and Chance* is well suited for CT in primary mathematics education. Random processes are hard to grasp but can be made tangible via digital simulation. Automated data generation enables learners to identify patterns, test hypotheses, and develop preconcepts. This is crucial in Swiss curricula, where formal probability calculations are introduced only after Year 6. Instead, students learn to describe chance qualitatively (e.g., likely, uncertain) and compare probabilities (EDK, 2011). Programming games with randomness supports probabilistic understanding and CT engagement.

Game Design, Game Based Learning, and Gamification

Gamification has been used to address negative attitudes towards mathematics, often using reward elements from games (e.g., badges, leaderboards) to boost motivation—at least in the short term (Buckley & Doyle, 2016). In contrast, GD fosters engagement through the meaningful act of creating. To describe this, Kafai (2018) coined the term *hard fun*: learners enjoy the game (fun), though the underlying challenge may be difficult (hard).

GD shares features with Game-Based Learning (GBL), as both leverage the cognitive and motivational potential of games (Kafai et al., 1998; Plass et al., 2015). Just as CT can be fostered through unplugged activities (Chen et al., 2023), GD and GBL include

digital and analogue games. Educational intent may be explicit (serious games) or implicit (e.g., card games with underlying mathematics concepts). Both often include gamification but aim for deep engagement—not just 'sugar-coating' (Bruckman, 1999, p. 75) mathematics content. In GBL, students typically play pre-designed games. GD, by contrast, positions learners—following Papert—not as players but as designers.

RESEARCH QUESTIONS

Based on this theoretical background, the study is guided by the following two research questions (RQ):

- RQ1: (How) Do GD processes unfold as problem-solving processes when learners design games with probabilistic elements?
- RQ2: (How) Does probabilistic thinking manifest and develop during these design processes?

METHODOLOGY

Participants and Sampling

The study involved *N*=18 sixth-grade students (9 female) from two Year 6 Swiss primary school classes, selected via convenience sampling. From nine student pairs, four were selected for in-depth analysis. The average age of the selected participants was 12.1 years (SD = 0.5). One student per pair was interviewed in a video-stimulated recall session. All students took part voluntarily. Written informed consent was obtained from parents or guardians. All names used in this paper are pseudonyms to protect the participants' identities.

To obtain information-rich cases, pair formation followed a purposeful sampling strategy (Nyimbili & Nyimbili, 2024). After piloting the setting with a different student pair, the sampling combined two principles: *homogeneity* (Year 6 students from the same Swiss canton using the same mathematics textbook, and socially compatible pairings based on teachers' recommendations) and *maximum variation* (differences in mathematical attainment level and varying levels of prior programming experience, both in and out of school). Table 1 shows the student pairs.

Table 1: Student Pairs

Student Pair	Programming Experience	Attainment Level	Student for Recall
Lina & Noel	yes	high	Noel
Mira & Loris	yes	low	Mira
Sami & Tiago	no	high	Sami
Sophie & Alina	no	low	Sophie

Study Design and Instruments

An exploratory-descriptive qualitative design combined observations of GD processes (on-screen actions and student dialogues) and video-stimulated recall interviews. Prior probabilistic thinking was assessed using a translated multiple-choice test on probabilistic thinking (Stacey et al., 2013). Programming experience was inferred from school context and interview questions regarding informal exposure to programming.

The main task involved creating a version of *Whac-A-Mole* in *RULER.game*, a visual programming environment. Students controlled mole appearance on a 7x4 grid via appearance frequency (deterministic) and percentage chance (random), so that the game was playable but not predictable. Prior classroom activities excluded randomness, allowing students' understanding to emerge during the task.

All students watched a gameplay video of an arcade version of Whac-A-Mole beforehand to highlight randomness. Post-task, one student per pair discussed key programming sequences in a semi-structured video-stimulated recall interview.

Data Collection and Analysis

The study took place over two or three mornings at the participating schools in October 2025, with each pair spending approximately 30 minutes on the design task and 15 minutes in a follow-up interview. The full procedure included four stages: (1) a short, standardised test assessing probabilistic thinking; (2) a guided introduction to the programming tool Ruler.game; (3) the actual task to design a Whac-A-Mole game; and (4) a video-stimulated recall interview.

All videos were transcribed and coded using MAXQDA. Coding followed a structured qualitative content analysis (Kuckartz & Radiker, 2023), with deductive categories based on programming/problem-solving phases (planning, implementation, testing, evaluation) and subcomponents of probabilistic reasoning (recognising randomness, using probabilistic terms, comparing chances). Additionally, inductive coding captured unanticipated themes using the method of subsumption (Schreier, 2012). Where applicable, codes were quantitatively evaluated (e.g., frequency and duration of code applications) in the sense of evaluative content analysis (Mayring & Fenzl, 2022). Recall data helped validate observations.

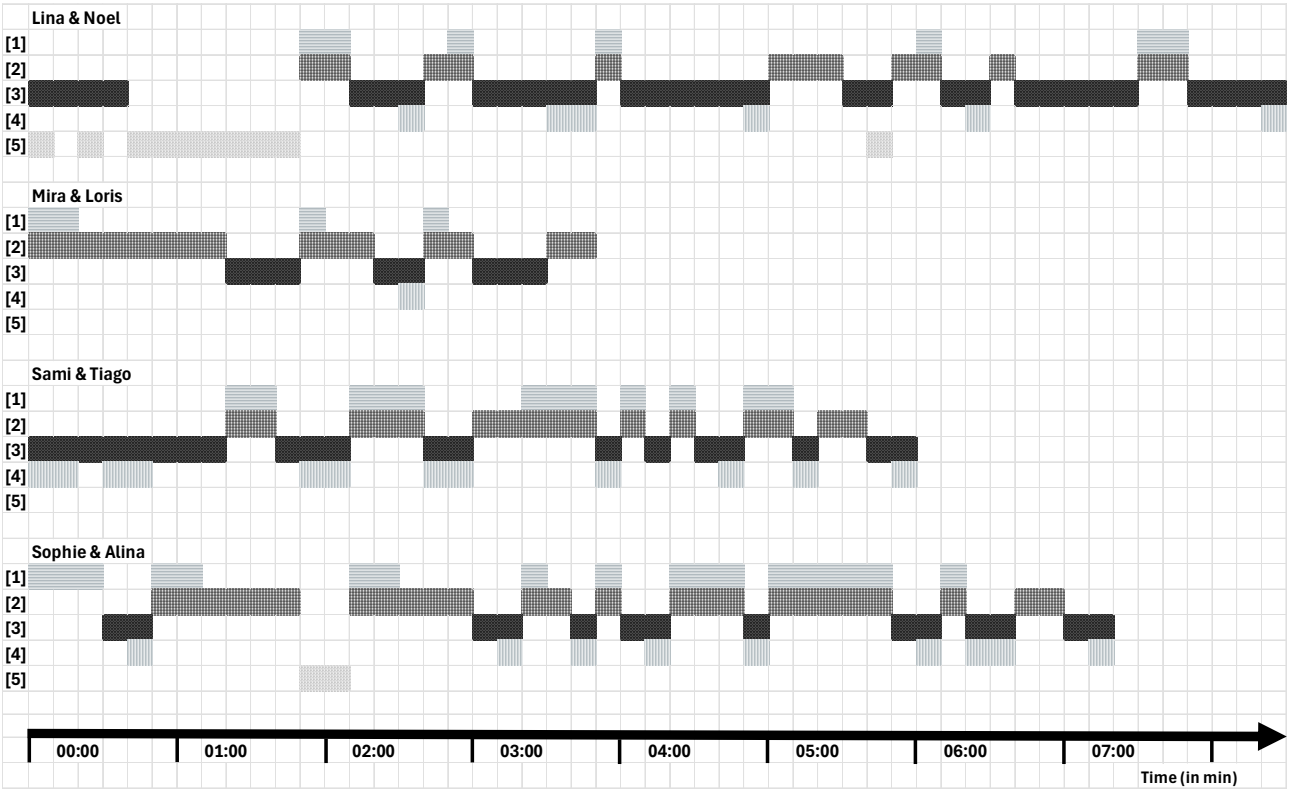
RESULTS

RQ1: Game Design as Problem-Solving

All four student pairs engaged in problem-solving cycles involving planning, implementation, testing, and evaluation. Rather than following a linear sequence, their work was characterised by iterative tinkering, with planning and evaluation closely interwoven with coding and testing (see Figure 1).

Figure 1: Structure and overlap of problem-solving components in GD

([1] Planning, [2] Implementation, [3] Testing, [4] Evaluation, [5] Interview Support)



High-performing pairs spent more time testing (up to 60%), whereas low-performing pairs less than 40%. Verbal planning and reflection—key indicators of reflective thinking—were also more frequent among high performers. For example, Sami & Tiago engaged in hypothesis-driven dialogue:

> Sami: Okay. What happens if we set 1%? Let's try it.
>
> Tiago: 1%?
>
> Sami: It's slower now … What if we reduce the time?

They compared their version to the arcade game, showing a design mindset. In contrast, Mira & Loris used brief, directive talk, with little reasoning.

In summary, GD triggered authentic problem solving. However, quality varied substantially across pairs, especially in iteration, reflection, and communication.

RQ2: Emergence of Probabilistic Thinking

Probabilistic thinking emerged in different ways and to varying extents across the student pairs. At the beginning of the design task, most students correctly identified randomness as a core feature of the game, for example:

> Noel: So that you don't know where the next one will come out.

In contrast, Mira & Loris initially assumed deterministic patterns based on prior actions in the game. While they eventually revised this view, they had difficulty articulating it in mathematical language.

When prompted to interpret programming rules involving timing and percentages, high-performing students gave more accurate and abstract explanations. Others struggled. Sophie, for example, assumed that the percentage reflected the player's chance of success, rather than a programming input affecting object generation. And Mira believed that the probability of a mole appearing, and where it would appear, depended on how many moles had previously been hit.

During the interview, students were also asked to manipulate values so that no moles would appear. High performers understood the implications (e.g., 'nothing will appear at 0%'), while lower performers tried deleting rules instead. Additionally, to probe more nuanced understanding, students were asked to compare two game settings: Game A with a frequency of 1 second and a probability of 10%, and Game B with a frequency of 4 seconds and a probability of 40%. High-performing students like Noel and Sami immediately recognised their mathematical equivalence by multiplying both time and probability. Others either treated the percentages as absolute values (e.g., '40 moles') or focused only on one variable at a time, missing the combined effect.

Table 2 summarises which elements of probabilistic reasoning (probability only, frequency only, or both) were considered during design loops (consisting of the phases planning, implementation, testing, and evaluation). While all pairs engaged with individual aspects of probability or frequency, only high-performing pairs consistently integrated both dimensions—indicating a more advanced level of probabilistic thinking.

Table 2: Focus of Design Loops (Probability/Frequency)

Pair	Loops	Prob. Only	Freq. Only	Both
Lina & Noel	8	5	2	1
Mira & Loris	3	3	0	0
Sami & Tiago	7	1	1	5
Sophie & Alina	8	3	2	2

In summary, probabilistic concepts were present in all pairs' GD processes, but to different extents. High-performing students showed deeper and more integrated understanding, particularly when coordinating probability with frequency. Interestingly, prior programming experience appeared to be less influential than overall academic performance in fostering probabilistic reasoning.

DISCUSSION

This study shows how mathematical problem solving and probabilistic thinking can be developed through GD. CT practices like debugging, iteration, and parameter adjustment foster such reasoning, supporting Papert's idea that understanding deepens through creation. This suggests that mathematical concepts like probability can be introduced through CT practices, such as GD. Based on our findings, we propose focusing on three central CT practices to foster CMT:

- *debugging*, which encourages learners to test assumptions systematically,
- *iteration*, which provides opportunities for repeated refinement of probabilistic models, and
- *parameter modulation*, which helps learners explore how small changes affect game behaviour and outcomes—thus promoting CMT.

Limitations of this study include the small sample size, short duration of the intervention, and context-specific task design. Nevertheless, the results of this exploratory study indicate the potential inherent in combining CT with GD through mathematical content. Future research should extend time frames, explore scaffolding, test transferability to other topics, and combine qualitative and quantitative insights.

References

Bruckman, A. (1999). Can educational be fun. *Game Developers Conference*, *99*, 75–79.

Buckley, P., & Doyle, E. (2016). Gamification and student motivation. *Interactive Learning Environments*, *24*(6), 1162–1175. https://doi.org/10.1080/10494820.2014.964263

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. *International Journal of STEM Education*, *10*(1), 47. https://doi.org/10.1186/s40594-023-00434-7

EDK. (2011). *Grundkompetenzen für die Mathematik. Nationale Bildungsstandards.* https://edudoc.ch/record/96784/files/grundkomp_math_d.pdf

Kafai, Y. B. (2018). Constructionist visions: Hard fun with serious games. *International Journal of Child-Computer Interaction*, 18, 19–21. https://doi.org/10.1016/j.ijcci.2018.04.002

Kafai, Y. B., Franke, M. L., Ching, C. C., & Shih, J. C. (1998). Game Design as an Interactive Learning Environment for Fostering Students' and Teachers' Mathematical Inquiry. *International Journal of Computers for Mathematical Learning*, *3*(2), 149–184. https://doi.org/10.1023/A:1009777905226

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi study. *Research in Mathematics Education*, *23*(2), 159–187. https://doi.org/10.1080/14794802.2020.1852104

Kuckartz, U., & Radiker, S. (2023). Qualitative Content Analysis: Methods, Practice and Software (2nd ed.). SAGE.

Mayring, P., & Fenzl, T. (2022). Qualitative Inhaltsanalyse. In N. Baur & J. Blasius (Eds.), *Handbuch Methoden der empirischen Sozialforschung* (pp. 691–706). Springer. https://doi.org/10.1007/978-3-658-37985-8_43

Nyimbili, F., & Nyimbili, L. (2024). Types of Purposive Sampling Techniques with Their Examples and Application in Qualitative Research Studies. *British Journal of Multidisciplinary and Advanced Studies*, *5*(1), 90–99. https://doi.org/10.37745/bjmas.2022.0419

Papert, S. (1986). *Constructionism: A New Opportunity for Elementary Science Education.* Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95–123. https://doi.org/10.1007/BF00191473

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of Game-Based Learning. *Educational Psychologist*, *50*(4), 258–283. https://doi.org/10.1080/00461520.2015.1122533

Resnick, M., & Rosenbaum, E. (2013). Designing for Tinkerability. In M. Honey & D. Kanter (Eds.), *Design, Make, Play: Growing the Next Generation of STEM Innovators* (pp. 163–181). Routledge.

Schoenfeld, A. H. (1992). Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics. In D. Grouws (Ed.), *Handbook of research on mathematics teaching and learning: A project of the National Council of Teachers of Mathematics* (pp. 334–370). Macmillan.

Schreier, M. (2012). *Qualitative Content Analysis in Practice.* SAGE. https://doi.org/10.4135/9781529682571

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

Stacey, K., Price, B., Gvozdenko, E., & Steinle, V. (2013). *Specific Mathematics Assessments that Reveal Thinking.* http://www.smartvic.com/teacher/

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. Commun. *ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Woo, K., & Falloon, G. (2022). Problem solved, but how? An exploratory study into students' problem solving processes in creative coding tasks. *Thinking Skills and Creativity*, *46*, 101193. https://doi.org/10.1016/j.tsc.2022.101193