Sandwich Robot for Computational Thinking: Reflections From Testing With High School Pupils

Emanuela Marchetti, Andrea Valente and Nina Bonderup Dohn
Department for Media, Design, Learning and Cognition, University of Southern Denmark, Denmark

emanuela@sdu.dk
aval@sdu.dk
nina@sdu.dk

Abstract: Our study discusses results from testing and co-designing *Sandwich Robot*, a game on algorithmic thinking targeting beginner programmers. The player composes code from a minimal set of blocks, to make a robotic character gather ingredients for a sandwich, following a specific order; the gameplay is similar to other Computational Thinking (CT) games like *Karel the Robot* or *LightBot*. The latest prototype of the game has been tested with three classes of 15-20 pupils from a technical high school in Denmark. The test was conducted in collaboration with their teacher in the subject *Informatik*, which focuses on CT and basic programming, and ended with a co-design workshop. Our results shows that pupils were interest in the basic concept behind our game, they generally described our game as "fun". Pupils and their teacher saw learning potential in the game, as a tool for understanding algorithmic thinking, but also for refining their learning at later stages. This expanded the role we envisioned for our game, leading us to rethink its relation to the pupils' learning process. On-going and future work involves addressing pupils' feedback and providing a level editor to allow Kahoot-style challenges. Our main contribution provides an exemplar of how the thinking behind algorithmic problem-solving can be transposed into game mechanics, embodied in a casual game, and how a non-technical narrative can support beginners learning CT.

Keywords: Computational Thinking, Playful learning, Games, Co-design, High school

1. Introduction

This study proposes a User Centred investigation about how beginner programmers face algorithmic thinking in solving computational problems, in CT. We are interested in investigating which kind of solutions they would adopt, while dealing with the complexity of algorithmic thinking, and which factors can challenge beginners in formulating algorithmic solutions. We also want to look at how a CT casual puzzle-like game with a funny narrative, designed to demystify technical and mathematical aspects of CT, could foster algorithmic thinking practices through code manipulation. Sandwich Robot is conceptualized as a casual game, not to feel intimidating for non-hardcore players imposing them a high-paced, challenging gaming style. Casual games are in fact designed for a wide audience, not requiring high-level gaming skills or strong time commitment (Kultima, 2009). The game, *Sandwich Robot*, proposes one puzzle per level, where a player must instruct a robot to collect ingredients and make a sandwich; the game uses a Blockly-based visual language for the instructions, and has an occasional-style gameplay.

Our inquiry was conducted in close collaboration with a teacher of the subject *Informatik* at Oerestad Gymnasium (OEG) in Copenhagen. Informatik in Denmark is the main subject focusing on developing CT competences and basic programming skills at a high school level (see the official definition of the subject, in Danish: https://www.ug.dk/6til10klasse/informatik-paa-de-gymnasiale-uddannelser). Our user group is based in a technical high school, and the study program has great focus on scientific subjects, including coding and mathematics, therefore, we experienced that the pupils were generally motivated to engage in programming, some seemed passionate about it, while for others it was part of their daily learning activities. CT was generally framed within creative interventions, such as designing websites, games or apps for local institutions (Witfelt, 2022). In this sense, proposing a game for simplify algorithmic problem solving for the pupils fitted well the current pedagogical practices of the school, and the teacher saw our game as a meaningful resource to introduce specific aspects of CT.

In the next section we introduce the theoretical grounding of our study and related work, afterwards we present our methodology for data gathering and analysis, thus the design and implementation of Sandwich Robot are presented, and finally we analyze and discuss our gathered data.

2. Related Work – Theory and Existing Research

Our study is mainly grounded on CT and Hermeneutics (Gadamer, 1989), as a pedagogical framework to foster understanding of CT through digital games. Hermeneutics describes learning as a path along a *circle*, starting from learners' previous knowledge and acquiring new knowledge while proceeding across this circle and

encountering new texts to interpret (Gadamer, 1989; Sotirou, 1993). During this path, conflicts arise as these new texts might contradict the learners' previous knowledge; however, previous knowledge still provides a resource to deal with new directions for meaning making (Tomkins and Etough, 2018). In this context, we see casual games as hermeneutic resources to foster learning of algorithmic thinking within a familiar framing. Casual games are designed to address a wide audience, displaying different levels of skills and commitment to gaming, building on principles such as: *Acceptability,* as they do not include controversial topics and are often free to play; *Accessibility,* these games do not require extensive training to be played, being typically seen as a secondary activity, and are easy to find and install; *Simplicity,* game mechanics and functional elements are kept to a minimum, or automated, like saving (Kultima, 2009). Casual games are perceived as non-intimidating, which can easily fit other purposes like learning, physical or mental exercise or even providing relieve from stress and anxiety (Pine et al., 2020). In our game, the casual-style narrative of *instructing a robot to make a sandwich*, was chosen to frame programming within cooking, a simple daily activity and recurrent theme in casual games (Kultima, 2009), and the robot as a main character suggests a playful automation scenario.

CT has become a highly popular learning topic, following its popularization by Wing (2006, 2011) who stressed the importance for all citizens to develop a mindset corresponding to that of computer scientists and engineers. We follow a recent adaptation of Wing's definition, according to which CT *"consists in the cognitive processes involved in the development of digital artifacts and programs to live in today's world."* (Dohn et al., 2022). This definition on the one hand is useful in formulating as a main goal in the global society that more children and young adults become proficient makers of technological artefacts. On the other hand, it allows querying how appropriate any given digital artifact is for living in today's world, ethically, socially, environmentally (etc.) speaking. Thus, the definition speaks to other understandings of CT, which focus on fostering young people's awareness of the risks and rights of our information technology (Grønning, 2017). More generally, over the past decades, CT has developed into a complex umbrella concept for an interdisciplinary knowledge area, including hardcore technological skills, like coding or making digital artifacts with microcontrollers, as well as soft skills rooted on design and social sciences, related to conducting iterative design processes and understanding how technology use can affect individuals and groups (Tedre and Denning, 2016). Dohn et al. (2022) present a framework that explicates this umbrella concept, highlighting three basic approaches: a cognitive (focused on arithmetic thinking processes), a situated (focusing on pupils' participation in creative coding practices) and critical (focusing on critiquing ethical and political implications of algorithms in society).

In the Danish school system, in which we conduct our inquiry, many schools have adopted a hands-on learning approach to CT within existing subjects. Here, teachers and pupils engage in making high-fidelity prototypes, tinkering with sketches and digital technologies, developing both soft skills and some hardcore competences of CT. Still, according to our contact teacher from OEG, most pupils find challenging to understand algorithms and craft code in programming languages and, furthermore, they present different proficiency levels, making it harder for the teachers to address all the pupils' needs. Therefore, in our CT studies we address how to expand the available palette of learning tools and resources, to support teachers in framing these aspects of CT. In order to identify key areas for learning CT, we refer to Chongtay's (2022) taxonomy of CT skills:

- Decomposition – logical analysis and data organization.
- Pattern matching – representation of data through model and simulations.
- Design of algorithms – construction of automatic, sequence-based solutions.
- Pattern generalization and abstraction – generalization of algorithmic solutions from one specific problem to a class of similar problems.

Interestingly Chongtay's first two skills deal with analytical competences, such as being able to logically analyze data structures and representations in computational models. While the more advanced principles deal with active design of algorithms, creating an automatic solution to address a computational problem, and finally abstraction and generalization in designing algorithmic solutions that could address not only one but an entire class of computational problems. The last skill deals with structured use of programming languages, reusing code, and being aware of how control flows through their code depending on inputs.

3. Our study: Design and Data Collection

Sandwich Robot is a game on algorithmic thinking, targeting beginner programmers. The player composes code from a minimal set of blocks, to make a robotic character gather ingredients for a sandwich, following a

specific order. The gameplay is similar to other Computational Thinking (CT) games like **Karel the Robot** (https://compedu.stanford.edu/karel-reader/docs/python/en/intro.html) or **LightBot** (https://lightbot.com/).

Figure 1: Level 1 is visible on the left, with the GUI of the game; on the Right, a detail of the blocks

Figure 1 shows the whole GUI for level 1 of the game. The page has the typical layout of a block coding application: on the left the area where the blocks can be moved to compose code, a "run" button to execute the code, and on the right the world in which the execution will take place, in our case it is *map of the shop* with the robot and the ingredients. The "run" button is near a slider that controls the speed of the execution, and an "help" button. On top of the shop map there is the sandwich assembling area, where the ingredients collected by the robot will pile up. Finally, at the bottom of the shop map area, the goal shows which ingredients should go in the sandwich: here tomato, salad and ham, in that order. The far-right part of figure 1 shows all commands available in the minimal Blockly language that the robot understands: blocks for **moving** and **rotating**, **conditional** and two kinds of **loops**. We designed this language not to be Turing complete, and apart from the *untilStepOn* loop, all other commands always terminate in one or a small number of steps; this fact has a pedagogical consequence: it makes the **Notional Machine** (Sorva, 2013) very simple, while keeping the language in line with the core of any imperative programming language. The problem with the *untilStepOn* loop, is that it is fundamentally a *while loop*, i.e. if the condition is never met, the loop keeps running forever; we implemented infinite loop detection via step counting.

The latest version of the game is available at: http://www.andval.net/robot4/, it includes a practice level and ten levels, organized by growing complexity. Each of the ten levels can be solved by hard-coding all the steps and turns needed to build the correct sandwich; however, the game rewards solutions with 1, 2 or 3 *stars*, and a shorter, *smarter* code gives more stars. Moreover, we designed the level progression to help learners reflect on the various commands in our Blockly language: e.g. some levels have larger shop maps than others and few ingredients, and that is designed to make the *untilStepOn* instruction more relevant.

To better foster understanding of pattern generalization, our level design focused on grounding conditionals and showing that an algorithm is a solution to a class of similar problems. Hence, we devised *twin-worlds levels*, where the player writes a single solution to control two robots, simultaneously but each in its own separate shop map (Figure 4).

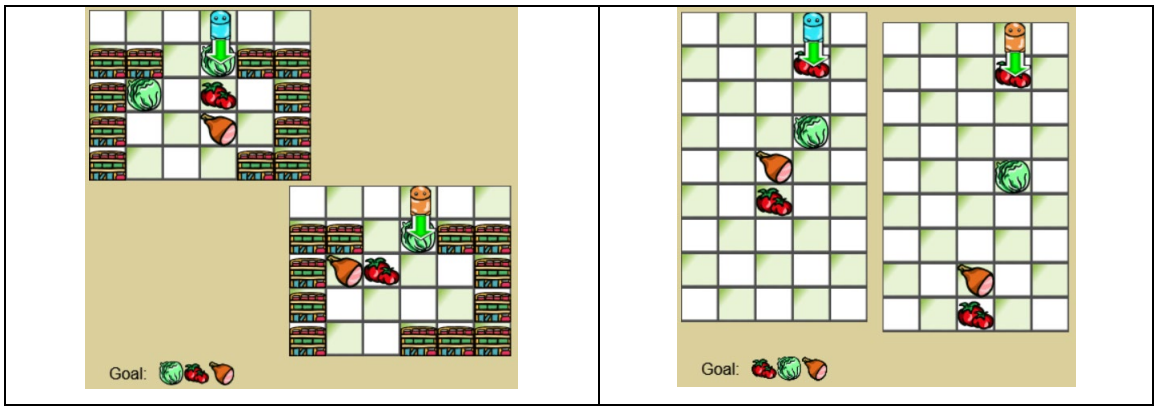

Figure 2: Twin levels 3 and 8 (left and right respectively)

Each level of our game can also be played offline, as a tangible version: the learners can print an A4 page with the shop map, cut the game pieces and play the game in class or at home. Supporting tangible interaction and offline continuation is central to our game concept (Marchetti et al, 2022).

The game is implemented as a serverless web application, using JavaScript and Google Blockly. When a new player starts the game, she is assigned a progressive number, and an entry is made in a mongoDB cloud database; all further attempts at solving a level (failed or successful) are sent to the database automatically, via AJAX, and stored in JSON format.

3.1 Ethnographic Data Collection

The empirical study builds on principles from User Centred Design (Preece et al., 2019), within a mixed method framework (Shannon-Baker, 2022), combining ethnographic observations, co-design and automatically saved data. Each of the pupils' gaming session was automatically saved, so that we could analyze how the pupils played our game, e.g. how many attempts to solve a level, if they played the levels in sequence or moved freely across, and which levels were most challenging.

We engaged in a close collaboration with a high school CT teacher and his pupils, so that we observed his learning activities (Marchetti et al., 2022), and then we involved his pupils in testing and improving our game. We are currently on the 2nd test iteration of our game and this time we tested with three different classes:

- 1st group – 14 pupils 9 boys and 5 girls in 3rd grade
- 2nd group – 17 pupils 13 boys and 4 girls 2nd grade
- 3rd group – 14 girls and 12 boys in 2nd grade

The length of high school in Denmark is three years, so the first group was at the final year of their education. The third group constituted the largest class we observed at the school. In general, all the classes were composed of males and females, from different ethnical background.

Our test was structured in 3 phases:

- Introduction and survey circa 30 minutes – the teacher presented us, our project and our game to his pupils.
- Testing Sandwich Robot 1 hour– the pupils play the 10 levels of our game under our supervision.
- Co-design session circa 2 hours – the pupils are invited to comment on post-its 3 positive and 3 negative aspects of our prototype, and to propose improvements to our game through collaborative sketching.

The teacher acted as our mediator, he was involved in planning the testing session, and provided a supportive presence, also securing correct communication given that two of us are not Danish native speakers. During our test, we gathered diversified kinds of data which enabled us to analyze the players' interaction with our game, both in terms of their performance and gaming experience. Table 1 shows an overview of the kinds of data that we gathered during the testing sessions:

Table 1: Type of data gathered during the test

Introduction	• Written Notes – response of pupils to our presence and test setup • Drawings – mood and bodily interactions among the pupils
Gaming Session	• Written Notes – from observations and informal situated interviews with the pupils • Drawings – Pupils' concentrated mood and bodily interaction while playing • Auto-saved data – automatic saving of each game session showing how the pupils played the different levels
Co-design Session	• Written Notes – from observations among the pupils as they were engaged in co-design • Drawings – showing how the pupils engaged with sketching during co-design • Sketches and notes – made by the pupils during co-design

To conduct an ethical data gathering, we agreed on our methodology with the teacher, who presented the whole procedure to the students and gave them the possibility to opt out of the study. To protect the privacy of the pupils participating in our study, we decided to avoid video recordings and instead we combined ethnographic observations with note taking and drawing. We see ethnographic drawing as a visual ethnographic tool, allowing to manually gather information, which might be complicated to described in words as ethnographers would do with a camera (Kuschnir, 2014; Marchetti, 2022), such as: facial expressions, bodily postures, gestures, and social interactions. A difficulty with ethnographic drawing is to capture dynamic actions, however, during our study the pupils were quiet and concentrated most of the time, except when they experienced the urge to communicate with each other, or with us: in such occurrences they would display changes in posture and facial expressions.

Finally, we collected sketches and notes created by the pupils during the co-design sessions. To facilitate the pupils to creatively tinker about our game, we provided sketching materials, such as: drawing paper, markers, scissors, tape, and printed handouts of the *twin levels* 3 and 8 of our games, which we see as most challenging (as discussed in the previous section). By providing templates of those levels, we hoped that the pupils could give us specific feedback on how to simplify them. This design material was acquired in the tool Nvivo and subject to a thematic cluster analysis, combining deductive and inductive coding (Shannon-Baker, 2022). Since we asked the pupils to list on sticky notes three aspects they liked and three they did not like about the game, we introduced a general deductive coding. On the other hand, the pupils' design proposals were analyzed inductively, identifying emergent themes in their sketches and notes.

4. Analysis and Discussion

In this section we discuss the data gathered through our study, starting with the automatically saved data from the pupils' game sessions (4.1), to end with ethnographic and co-design data (4.2). Finally, we propose a critical perspective on our data, regarding a hermeneutic perspective on CT for beginner programmers and future works regarding improvements of our prototype (4.3).

4.1 Quantitative Analysis of Auto-Saved Data

Each player starts by registering, entering a few non-sensitive data, and receiving a unique identifier number; the data requested in the registration form is limited to only first name or a nickname, self-reported programming experience (ranging from 1 to 5) and age, to ensure adherence to Danish GDPR rules. No sensitive or private data is stored on our database, we just use the unique identifier to "trace" the history of each player. When playing a level, every time the player runs the Blockly code, an AJAX call is sent to the cloud database and records: the player's id, the timestamp of the attempt, and the code, information about success and how many stars were awarded, and finally, information about the number and timing of clicks the player used to compose the code in each attempt. The automatic saving of game sessions allows us to perform offline data processing and analysis for each player, the most interesting of which are:

- number of attempts to win a level, and total time spent playing a level
- **play interval** of a level: i.e. first and last time a level was played
- how **continuous** were the attempts, calculated as

$$\frac{number\ of\ attempts\ at\ solving\ level\ N}{number\ of\ attempts\ at\ any\ levels,\ during\ the\ play\ interval\ of\ level\ N}$$

- number of times the mouse was clicked, in a single attempt, for a specific level
- the **click density** for an attempt, measured in clicks per attempt, calculated as:

$$\frac{number\ of\ clicks\ in\ all\ attempts}{number\ of\ attempts\ at\ solving\ that\ level}$$

For every level, we also calculated: the minimum, maximum, and average length of the solution, the average number of wins, and the average time spent by players in that level. Table 2 shows the overall results of all tests we performed so far, from August 2022 to March 2023; in Table 2, the time is measured in seconds.

Table 2: Overall results from all our tests

level	avg num attempts	avg num wins	avg time spent	avg continuous strategy	avg click density	min length	max length	avg length
0	2	1,26	41	67	2	5	8	6,25
1	3	1,26	636	62	5	5	7	5,75
2	6	1,6	897	65	10	7	17	13,07
3	11	2	1.187	55	7	4	15	8
4	4,2	1,53	923,53	53	12,44	4	16	10
5	2,4	1,26	772,26	50	7,15	6	11	6,66
6	3,73	1,06	246,93	67	9,63	8	12	10,7
7	2,33	0,86	183,2	74,03	10,96	9	12	9,5
8	2,73	0,8	209,13	67,36	10,75	7	13	9
9	2,6	1,2	273	62,42	8,7	4	16	9,09
10	8,6	0,8	611	71	16,95	9	20	15,37

We designed some of our indices to look at the *strategies* players adopted in their attempts. The *average continuous strategy* column in table 2 refers to how much a level was played without jumping to other levels, averaged over players. The *average click density* column attempts at estimating where players working on a level adopted a **think strategy** or more of a **tinker strategy** instead. A player can stop and think out a solution, before coding it with the blocks, while another might keep trying variations, relying more on the game to provide feedback: these two strategies define a spectrum that goes from **think** to **tinker**. The last 3 columns in table 2 refer to the length of a *correct submitted solution*, measured in number of blocks; levels with large span between min and max length should be considered more difficult to solve optimally.

We are aware of the small number of samples and our purpose is not to draw statistically significant conclusions, but instead to highlight how these findings relate to the analysis of the qualitative data discussed in the next subsections.

4.2 Observations of Gaming Sessions and Co-Design

During the gaming session, we saw that the students were very focused, so that it was hard to understand if they were having fun or not. In general, they played on their own, occasionally helping each other or calling us and their teacher for help. During the informal interviews conducted at the end of the gaming sessions, the pupils commented positively on the game. Most pupils said that the game was “fun” and three students even said “addictive” with a smirk, as they felt compelled to reach a 3 stars award on their solutions. This is confirmed by the auto-saved data, showing distinctively how the pupils played multiple times the same level to achieve higher scores. Moreover, most pupils said that they liked how the game was challenging and forced them “to think”, a girl said that she liked how: *“it (the game) is challenging my mind, I can see myself playing on my mobile phone on the bus.”*

During the co-design session, the mood changed dramatically, and the pupils were very active in sketching and cutting out paper, while socializing with each other in small groups. The design material they created was subject to a thematic analysis on Nvivo so to identify the most common experiences with our games and the most wished-for design requirements. In Table 3 we show the results of the coding of the material.

Table 3: Categories of Coding from Nvivo analysis of the pupils’ design materials

Name	Files	References
Negative comments	14	18
Usability	19	46
turn	3	4
Speed	4	4
Commands	11	16
feedback to players	9	10
New features	3	3
Personalisation	6	6
Sound	6	7
more ingredients	7	9
layout-navigation	17	23
Characters	4	5
Time	7	8
multiplayer	3	3
live-game	2	2
Positive comments	14	20

The most typical positive comments included the graphics and the concept behind the game, which was defined as: “fun”, “self-explanatory” or “simple”. On the other hand, negative comments dealt with suggestions to improve the graphics, the mechanic and instruction code for making the robot turn, and the lack of space in the coding area.

Looking more in details into the usability category, different input was gained and divided into subcategories, the most represented being how to control of the movements and speed of the robot. In the subcategory *Commands* we have gathered comments about the size of the buttons, colors, and placement. Most pupils appreciated the star system, they found it motivating to retake the same level and improve their results; a few of them suggested to have more explicit information on how they can improve their score.

Under the category *New Features*, we have gathered all the suggestions regarding new ideas for new game releases, such as: personalization of the robot-character, creation of new levels, addition of sound and juice effects at level completion, like a chewing sound and/or animation of the sandwich being eaten. All the players wanted a broader selection of ingredients, more obstacles to avoid in the levels. Most pupils asked for having timed game sessions to measure how quickly they could solve the level, and in this respect, we discussed the possibility of having multiple difficulty levels so players could start playing freely, but when feeling more confident they could play the timed game sessions. Finally, all the pupils suggested to have a multiplayer modality, in which they could challenge each other or collaborate online in gathering the ingredients. An original proposal was that of transposing the game into a live-action role play game, where players compose the code and enact the robot’s actions live on a playground. One group also suggested the intriguing new name *Sandwich Maker*, where the word “maker” can suggest an automatic appliance or a cooking robot.

Pupils and the teacher saw learning potential in the game, as a tool for scaffolding algorithmic thinking, but also for training and refining learning at later stages.

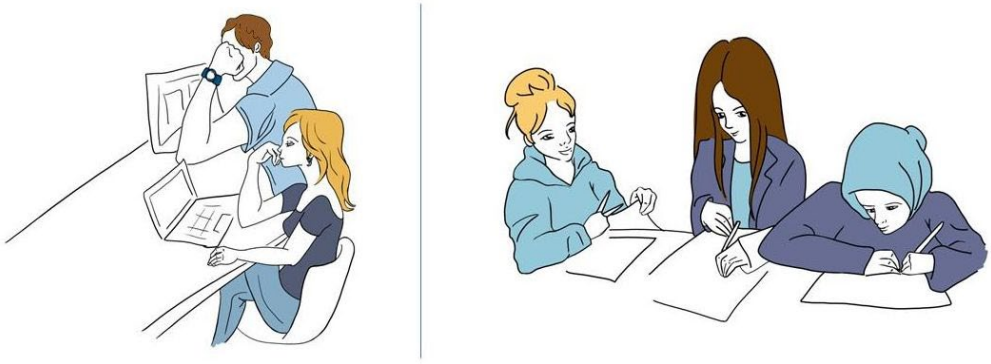

Figure 3: Drawings from gaming and co-design sessions

4.3 Critical Discussion

A suggestion made by several of the pupils was the wish for a larger coding area. On the face of it, this is a simple suggestion, highlighting an apparent lack of user-friendliness in the prototype design. However, the limited area for coding was intentional on our part, as we wanted the pupils to experiment with getting their code short, because a shorter code requires a deeper understanding of abstraction and generalization (Chongtay, 2022). For instance, a solution using *untilStepOn* loops would work even if the shop map is scaled up, while an hard-coded solution would not; in our game shorter code is usually better quality code. Therefore, we had hoped that a limited area for coding would naturally suggest that the code should not be too long. This hope is contradicted by our observations and the automatically saved data which show that the pupils tended to assemble long sequences of instructions, rather than create code with loops. These considerations point to a tension often present in programming classes: between pursuing programming learning objectives on the one hand and providing pupils with freedom to choose projects and solutions, supporting their feeling of self-determination and ownership, on the other hand (Bjerre and Dohn, 2018). Taking a step back, the issue here is also one of different values: the programming-internal values of abstraction and generality, according to which the best code is the one with the fewest steps, or the one covering the most cases. The values of user-friendliness and understandability, where longer code is often more intuitive and comprehensible. And potentially also values of sustainability: our analysis of the auto-saved code suggests that sometimes the players proceeded to “compress” their code, using loops as much as possible, in order to collect 3 stars. However, in so doing they hyper-regularized their code, making the robot perform inefficient moves, just to

make the code easily expressible with the available loops. This phenomenon is actually part of the gameplay of the more advanced levels in other games, like LightBot. The extra, unnecessary moves or rotations might have a negligible cost in our simple program, but this would not be the case if the program was instead run by a physical robot that would consume energy to perform useless operations.

One future use of our prototype that has become apparent to us in this study, is thus to form the outset for an in-class discussion of values in designing digital artifacts. This brings us back to our initial discussion of conceptualizations of CT: it is important that young people learn, not only to create digital artifacts, but to create ones that are appropriate for living in today's world, from an ethical, social, and environmental point of view. Moreover, from the analysis of the auto-saved data, we know that our players favor different problem-solving styles, such as various levels of continuity in solving a level, and the spectrum of think/thinker strategies. We could have easily enforced the idea of a best/optimal strategy, by demoting solutions that required many failed attempts, or forbidding to play the next level until the current is solved. Instead, we designed our game to support these personal problem-solving strategies, to communicate to the students that they can find their own place in the programming practice, and in keeping with the principles behind User Centred design. Our little coding example may be a concrete way to foster understanding of the larger concerns that need to be taken into consideration when developing technology, and the risks of focusing too narrowly on programming-internal values. In this way, our prototype can be used to support both cognitive and critical aspects of CT, in the framework suggested by Dohn et al. (2022).

5. Conclusion

The main contribution of our paper proposes an example for how to transpose the thinking behind algorithmic problem solving into game mechanics, embodied in a casual game. It also shows how the choice of a non-technical context can better support beginners in learning CT. Ongoing work focuses on improving the game's graphics, creating new sets of levels, and experiment with suggested re-designs. Our analysis of the play strategies and the feedback from the players, suggests that future work involves the creation of a level editor and enabling group-to-group asynchronous challenges, as in tools like Kahoot; this would make the game more open to user customization and extend the playability of our game.

References

Bjerre, A., & Dohn, N. B. (2018) "Guided tinkering as a design for learning programming." Dohn, N. B. (Ed), *Designing for learning in a networked world*. Routledge, pp. 177-196.

Chongtay, R. (2022) "Computational Thinking som redskab til problemløsning på tværs af fagområder." Dohn, N. B., Mitchell, R., and Chongtay, R. (Eds) *Computational Thinking, Teoretiske, empiriske og didaktiske perspektiver*. Samfundslitteratur, Frederiksberg.

Dohn, N. B., Kafai, Y., Mørch, A., and Ragni, M. (2022) "Survey: Artificial Intelligence, Computational Thinking and Learning." *KI - Künstliche Intelligenz,* Vol. 36, No. 1, pp 5-16. https://doi.org/10.1007/s13218-021-00751-5

Gadamer H.G. (1989) *Truth and method.* Continuum, New York.

Grønning, A. (2017) "Sensitivt digitalt medieindhold: at undersøge fostre på Facebook." In: Drotner, K. and Iversen, S.M. (Eds) *Digitale Metoder, at skabe, analysere og dele data*. pp. Samfundlitteratur, Frederiksberg.

Kultima, A. (2009) "Causal Game Design Values." *The 13th MindTrek Conference*, ACM, Tampere, pp 58-65.

Kuschnir, K. (2014) "Teaching anthropologists to draw: a didactic research experience." *Cadernos de Arte e Antropologia*, Vol. 3, No. 2, p 23-46.

Marchetti, E. (2022) "At observere gennem dine hænder: Tegning som etnografisk praksis." Grønning, A. Lundtofte, T. E. and Walther, B. K. (Eds), *Medievidenskab - Metoder og Teorier.* University of Southern Denmark Press, pp 83-97.

Marchetti, E., Valente, A., and Dohn, N. B. (2022) "Sandwich robot: how non-programmers solve computational problems." Matsuo, T., Takamatsu, T. and Ono, Y. (Eds), *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI).* IEEE, pp 234-239, Kanazawa.

Pine, R., Sutcliffe, K., McCallum, S., and Fleming, T. (2020) "Young adolescents' interest in a mental health casual video game." *DIGITAL HEALTH* 2020, pp 1-7.

Preece, J., Rogers, Y., and Sharp, H. (2019) *Interaction Design, Beyond Human Computer Interaction*, Wyley and Sons.

Shannon-Baker, P. (2022) "Mixed Methods Designs, Integration, and Visual Practices in Educational Research". *Journal of Mixed Methods Research 2022*, Vol. 16, No. 2, pp 159–164.

Sorva, J. (2013) "Notional Machines and Introductory Programming Education." *ACM Transactions on Computing Education*. Vol. 13, 8:1-8:31. 10.1145/2483710.2483713.

Sotirou P. (1993) "Articulating a hermeneutic pedagogy: The philosophy of interpretation." *Journal of Advanced Composition,* pp 365-380.

Tedre M., and Denning P.J. (2016) "The long quest for computational thinking." *The 16th Koli Calling International Conference on Computing Education Research*, ACM, pp 120-129.

Tomkins L. and Eatough V. (2018) "Hermeneutics: Interpretation, understanding and sense-making." Cassell, C., Cunliffe, A. L., and Grandy, G. (Eds) *SAGE handbook of qualitative business and management research methods,* Sage, pp 185-200.

Wing, J. (2006) "Computational thinking." *Communications of the ACM,* Vol. 49, No. 3, pp 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. (2011) "Research Notebook: Computational Thinking - What and Why?" *The Link Magazine*, N/A. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Witfelt, C. (2021) *Informatik for Alle*. Forlaget Columbus.