British Journal of Educational Technology *Vol 0 No 0 2020* 1–16
doi:10.1111/bjet.12898

Modifying games with ChoiCo: Integrated affordances and engineered bugs for computational thinking

Chronis Kynigos and Marianthi Grizioti

Chronis Kynigos is Professor, Director at the Educational Technology Lab, http://etl.ppp.uoa.gr, leads design research on constructionist innovations in educational and out of school settings. Known for the term "half-baked artefacts" both for didactical engineering and for learning though modding and de-bugging. Responsible for the design of two authoring systems, MALT2, ChoiCo and around 1800 micro-experiments accessible through the Greek Ministry of Education. Marianthi Grizioti is a computer scientist and a PhD student at the National and Kapodistrian University of Athens. She is a member of the Educational Technology Lab and her research interests include computational thinking, ICT education, game-based learning and HCI. Address for correspondence: Chronis Kynigos, Educational Technology Lab, Department of Educational Studies, School of Philosophy, National and Kapodistrian University of Athens, Athens 15784, Greece. Email: kynigos@ppp.uoa.gr

Abstract
Although there is wide rhetoric that programming should be learnt by all as an element of computational thinking (CT), in practice, it is mostly implemented narrowly as an end in itself consisting of routine practice and traditional testing of the ability to code. This paper discusses a way in which programming could be seen through a wider integrated pedagogical approach as jointly cultivating meaning making of computational concepts in conjunction with the adoption of practices and strategies in a relevant meaningful context for learners. We elaborate on a case of learning to code through digital game modding where programming and other computational concepts coexist. Our design frame includes the principles of black and white box designs, of students as de-buggers of engineered half-baked games and of games embedding both concepts and values in simulations involving wider contested complex issues. We use our analysis of the meaning making of students as they debug a choice-driven simulation game specially designed to bring this integrated kind of learning to the fore. We show elements of context-aware integrated CT connecting otherwise fragmented areas such as databases, block-based programming, Geographical Information Systems design.

Introduction

Computational thinking (CT) refers to the mental process of understanding and solving problems by using efficiently a set of concepts, practices, strategies and behaviours fundamental to computer science, such as problem decomposition, algorithm building, analysis, generalisation and abstraction (Brennan & Resnick, 2012; Grover & Pea, 2018; Lee *et al.*, 2011; Wing, 2008). Researchers, scientists and educational stakeholders seem to agree that CT is a competency that all students should develop as digital citizens in the 21st century society (Grover & Pea, 2018; Wing, 2008).

Programming, according to researchers, can be a strong tool to foster CT. However, there is still lack of pedagogical approaches that would enable students to explore and express meanings integrating both computational concepts and computational practices through programming. Traditional coding activities focus strictly on teaching specific concepts (eg, conditionals, loops,

Practitioner Notes

What is already known about this topic

- Computational thinking is considered an essential element of digital citizenship. In this context, it has been portrayed as an educationally strategic 21st century competency that all students should acquire. According to several studies it involves the development of both concepts and practices that are necessary for computational problem solving. Concepts such as variables, conditionals, data structures and practices such as decomposition, abstraction, generalisation.
- However, most of the pedagogical approaches for CT focus rather narrowly on computational concepts and engage students with closed quizzes and puzzles, leaving aside practices.
- Not surprisingly, students have difficulties in describing and using computational practices (eg, abstraction, decomposition, pattern recognition) even if they have sufficient knowledge on programming concepts.
- Computational design activities, eg, game or simulation design, require high-order skills and knowledge which are complex and inaccessible by the young and inexperienced students.
- There is need for approaches that would foster students to express and develop meanings on both practices and concepts through an integrated context.

What this paper adds

- It suggests the design of tools with affordances aiming at students' integrated engagement with concepts and practices.
- It provides an example of a tool integrating affordances connected to diverse computational concepts such as those pertaining to programming, data handling and Geographical Information Systems (GIS).
- It discusses design principles connected to the embedding of integrating affordances, such as making use of the context of designing and playing digital games.
- It refers to the embedding of domain concepts in realistic contexts of wider complex real life issues such as sustainability or dietary habits.
- It provides in-depth analysis of student engagement with such tools and uses this data to elaborate the arguments and issues related to this design approach.

Implications for practice and/or policy

- There is value in disconnecting CT from positivist diagnostic approaches related to respective concepts.
- There is need for more resources into the design and development of digital media embedding affordances for concepts and practices while maintaining relevance and interest for their users.
- More examples are needed and analyses of student learning processes and how they may affect their digital citizenship more widely.
- These approaches need to combine individualistic and collective approaches to human learning.

variables) through closed tasks and quizzes, leaving aside computational practices like abstraction, generalisation, pattern recognition (Brennan & Resnick, 2012, Kynigos & Grizioti, 2018, Lee *et al.*, 2011). As a result, students tend to develop a concept-specific, fragmented knowledge and perception of CT rather than an integrated computational literacy (Robins, Rountree, &

Rountree, 2003). Very often, students who understand programming concepts struggle to apply them or combine them with practices to solve a computational problem (Armoni, 2013, Robins *et al.*, 2003).

Moreover, there are many digital applications to learn coding in a narrow way but surprisingly few tools designed through an integrated approach to CT. Typical genres of such tools are digital games for learning and simulations with some degree of structural access for the user (diSessa, 2001), even to the point of programmability. These tools are designed to invite learners to change game rules or simulation behaviours by means of programming. For such learners, however, the design from scratch of a computational simulation or game is a highly complicated and sophisticated process that requires deep conceptual knowledge and strategies and too much time making the production of something interesting inaccessible by young and inexperienced students (Salen & Zimmerman, 2005). Games designed to afford programming, for instance, may confuse students with difficult details and arduous semantics demanding too much time and effort to familiarise with the tool (eg, the functionalities, the programming language, technical issues) and then to programme a functional algorithm, rather than getting involved with the actual design and production processes (Salen & Zimmerman, 2005). This complex and broad approach does not usually allow students to focus on computational practices.

The challenge therefore discussed here is to enable students to explore, express and combine computational ideas avoiding the complexity of developing a computational system from scratch. This challenge entails the task to better integrate pedagogy with affordance design in order to allow a balanced engagement with computational practices and concepts. As affordances, we considered the opportunities that the digital environment offers to the learning process by enabling actions between the artefact and the student (Calder, 2012). The didactical integration of diverse, interconnected affordances in a learning environment could give students access to otherwise complex scientific concepts and support the meaning generation process (Noss & Hoyles, 1996). Attempting to address this challenge, we discuss game modification, or modding, with diverse affordances.

To address and discuss this task, we designed and developed a game authoring system which we call ChoiCo—"Choices with Consequences"[1] (see also Kynigos & Yiannoutsou, 2018). We consider a digital game as a complex system of interrelated and overlapping computational ideas bound together to form the game mechanics. Instead of either expecting from students to build this complex system from scratch or completely hiding the system structure from them, we adopted the process of "modding" in order to give them progressive access to these ideas and thus built ChoiCo as a tool for creating and modifying games. We combined three computational affordances which, so far, have been used as individual learning tools. The first is a map-based (GIS) editor which allows for designing the game interface, including the game areas and the available game choices. The second is a database for accessing and setting the game parameters and the consequences of the game choices. The third is a block-based programming language for constructing the game rules such as events, ending conditions, feedback, etc.

In order to investigate student meaning-making process of computational concepts and practices through modding with integrated affordances, we engaged in design research, implementing an empirical study. The participants were 12 middle school students who played and modified the simulation game "Eating Out" in ChoiCo. During the data analysis, we addressed the following research questions:

1. What meanings students express and construct for the computational concepts and practices when they collaboratively modify a simulation game.

2. Whether and how students transform or develop these meanings through the process of game modding.
3. How students utilise the three affordances of ChoiCo to modify the game and whether the integration of the affordances supports student meaning-making process

The rest of the paper is structured as follows. First, we discuss the theoretical background that framed our research, then we describe the methodology and tools of an empirical study, and finally we discuss the findings from the qualitative data analysis.

Theoretical background

Rethinking programming designs for CT

The notion of CT is not new. From the early appearance of LOGO programming, Seymour Papert used the term *algorithmic thinking* to describe the way students think when they construct their own artefacts in LOGO to solve problems (Papert, 1980). Some decades later diSessa talked about an upcoming literacy, similar to the already known literacies, like mathematical or linguistic, which he referred to as *computational literacy* (diSessa, 2001). DiSessa however made a clear distinction between the three pillars of this new literacy: digital technology (material pillar) which is the medium for exploration, mental processing and interpretation of what is explored to personal knowledge (cognitive pillar) and social communication of knowledge (social pillar). In 2006, Wing with her analysis of *CT* expanded the ideas of Papert and diSesssa beyond computer science or mathematical problem solving. Wing approached CT as "a set of skills, strategies and behaviours, that draws on fundamental concepts of computer science," but it can be implemented for solving problems across all disciplines as well as in everyday life. The academic approaches to the value of learning to programme have thus progressively perceived it as an expression of algorithmic thought, a new kind of literacy and a set of skills, strategies and behaviours connected to a wider value of cultivating CT for the 21st century citizen.

However, despite these academic elaborations on the potential value of learning to programme, programming in practice has been considered as an end in itself without much thought on how it can be put to use by students for expressing algorithms, creating digital objects and behaviours and solving problems. Students have mostly been taught programming through closed and concept-specific exercises which allow them to develop narrow and fragmented knowledge (Brennan & Resnick, 2012; Gomes & Mendes, 2007). One of the most common and most important difficulties that students face in programming as a result, is that they do not have the strategic knowledge to efficiently apply the concepts they learn in order to solve computational problems (Armoni, 2013; Robins *et al.*, 2003). So, in this paper, we consider it a challenge to elaborate approaches and display programming tools and activities that balance the focus on both concepts and practices of the computational process allowing students to use programming in relevant and interesting contexts.

So, as an example, we choose to discuss game modding as an approach to achieve a balanced engagement with and development of CT. In our design, we combine constructionist pedagogy (Papert, 1980, Kynigos, 2015) with integrated computational affordances that enable students to access otherwise complex computational ideas of the game structure. In constructionist learning environments, the affordances are formed by the available tools and functionalities for construction, but also by the behaviours and interpretations that students may assign to the tools as they use them. Thus, the type and the connections of available affordances can play an important role in the strategies and concepts that students apply and communicate during constructions. The affordances that we employ for game modding, ie, block-based programming, editable database

and map design, have been used so far only as individual learning tools for teaching clusters of computational knowledge. Our ChoiCo tool was designed to integrate all three affordances in a unified context and enable students to utilise them as part of game modding. We expected that this design would allow us, as researchers and educators, to focus on how students construct, transfer and connect meanings of computational ideas, instead of focusing only on how they understand isolated programming concepts.

Game modding: a "black-and-white" box approach

A digital game can be considered as a complex interactive system (Salen & Zimmerman, 2005) that consists of different components such as the interface, the characters, the objects and the story. All these aspects are bound together with rules, patterns and relations to create the final gameplay which is mediated to the player who then translates it to a personalised gameplay experience (Salen & Zimmerman, 2005).

This system offers an organiser; a Web of interrelated computational ideas such as abstractions and patterns, classes and functions, variables and conditionals, data structures and objects. The question, however, is how a young student could be engaged with that system in a meaningful way. On the one side, we have the approach that put the student to the role of the player. For the player, the system behind the gameplay is a "black box" and most computational ideas are hidden by design. Therefore, a student who plays a game can only explore just the tip of the iceberg. In addition, the student has a passive role since (s)he can only discover parts of the system but not intervene to change them. On the other side, we have the student in the role of the designer who creates a game system from scratch in the digital environment. We call this a "white box" approach, because the student can hypothetically access all the computational ideas of game design. However, in practice, students have neither the computational knowledge nor the experience to build such a complex system from scratch. Lowering the threshold and having students begin building things with generic building blocks often ends up with them stuck on "plateaus" (Noss & Hoyles, 1996), never having a sense of constructing something complex and meaningful, their constructions thus only reach the level of simple games.

To address the above challenge, we employed "black-and-white box" design approach as elaborated by Kynigos (2004), in which the designer decides what elements of the game are useful to be open and modifiable by the player and makes them accessible through high-level computational components. This design enables even non-technical users to modify, connect and build upon these elements without changing the core structure of the game. DiSessa (2001) coined the term "deep structural access" as a user affordance allowing for the use of generic building blocks with no limits to the complexity of things which can be created with them DiSessa, just as Papert's LOGO primitives for example (Papert, 1980). The design idea we use here is based on a twist to the deep structural access approach. It has been called "Principled deep structural access" accordingly by Kynigos (2004) who discussed such a design in reference to "E-Slate," an environment for constructing educational software with higher order building blocks. The design principle behind a building block thus no longer prioritizes it to be generic but rather to provide a concise tool to build with which in cases maybe quite complex from a technical point of view.

ChoiCo accordingly affords a database, a GIS and a programming language as three building blocks for games. In our design, students can utilise the integrated affordances of the ChoiCo design tool to modify some of the game elements and access the powerful ideas with which these elements were constructed and embedded into the game system. In that sense, game modding is a pedagogical approach for accessing computational ideas that balances game playing and game design.

On modding for learning CT

We borrow the term modding from the gaming community where it is used to describe the process of modifying the elements of a digital game to create a different version of it, which is called "mod" (Sotamaa, 2010). The process of modding usually involves analysis and decomposition of the game structure, iterative testing, coding, graphics design and so on. The "modder" therefore has both roles of player and designer. Several studies have shown the benefits of modding as learning activity in enhancing student meaning expression and skill development (El-Nasr & Smith, 2006; Kynigos & Yiannoutsou, 2018; Sotamaa, 2010).

In order to support student meaning making on CT concepts and practices, we adapt modding to a constructionist learning activity. It differs from other pedagogical practices like remixing, in the part of maintaining the original game idea, instead of creating a completely new game from an existing one or combining different games (remixing). The modifications are not strictly guided but students are free to decide what they want to change as they play the game.

To provoke modifications, we designed the game as a half-baked artefact, ie, artefacts with didactically engineered "bugs" aiming to challenge the users to modify or to extend them (Kynigos & Yiannoutsou, 2018, Grizioti & Kynigos, 2018). In our design, students play with a half-baked simulation game that has faulty behaviours engineered in the rules, the content and the values, inherent in gameplay. These "bugs" aim to work as trigger mechanisms for students to discuss computational solutions for improving the game. Before we describe the game in detail, we briefly present ChoiCo, the digital environment that students used in the study to play and modify the game.

Design tool and game

ChoiCo: a game designer of simulation games

ChoiCo (Choices with Consequences)[2] is an open source, online authoring tool for, playing, designing and modding choice driven simulation games. ChoiCo design is based on a microworld for learning about sustainability through playing and modifying elements of city-based games, called SuS-x (Kynigos & Yiannoutsou, 2018). With our focus on game modding for CT, we extended SuS-x idea to an online environment that integrates new affordances and supports the design of a wider range of games.

ChoiCo games simulate complex real-life issues that depend on various parameters and choices and have no clear solution. The player revolves around map-based settings making choices (these can be items, buildings, actions, etc) that have positive or negative consequences to the game attributes (eg, Money, Health, Fun). The aim is to keep these attributes balanced for as long as possible. The gameplay is based on decision making, prediction and balance.

In the "Design Mode," the user can design or modify the game elements with three integrated affordances: (1) a map-based (GIS) editor that gives access to the game interface, the available choices and the game areas (Figure 1), (2) an interactive database for setting the game attributes and the consequences of each choice. (Figure 1) and (3) a block-based programming language for programming the game rules (Figure 2).

"Eating out" game

The game that students played and modified belongs to the genre of simulation games (Amory, Naicker, Vincent, & Adams, 1999; Gee, 2008). In contrast to other genres like action or adventure games, there is no specific strategy that the players should follow to win. The flow of the game is formed by the player's actions upon the simulation and it continuously changes in a non-linear way. Many simulation games, eg, the Sim City series, deal with socio-scientific situations that

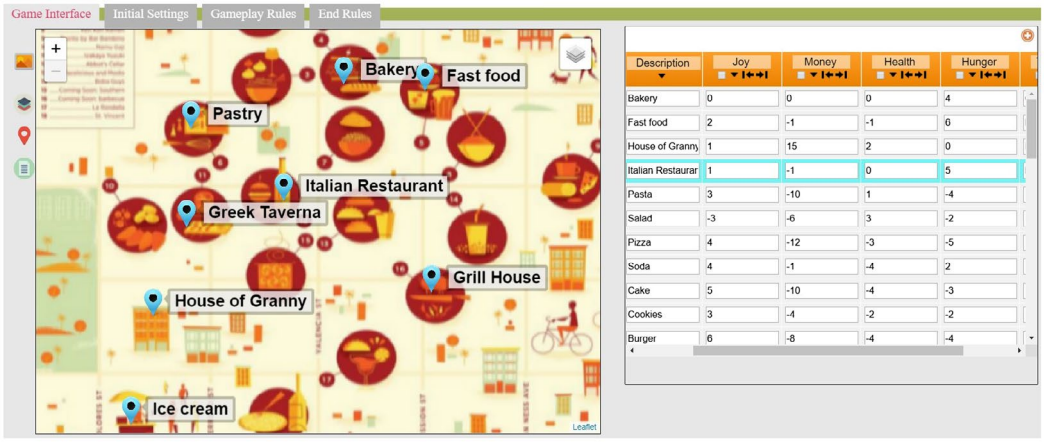

Figure 1: Map editor and interactive database in ChoiCo

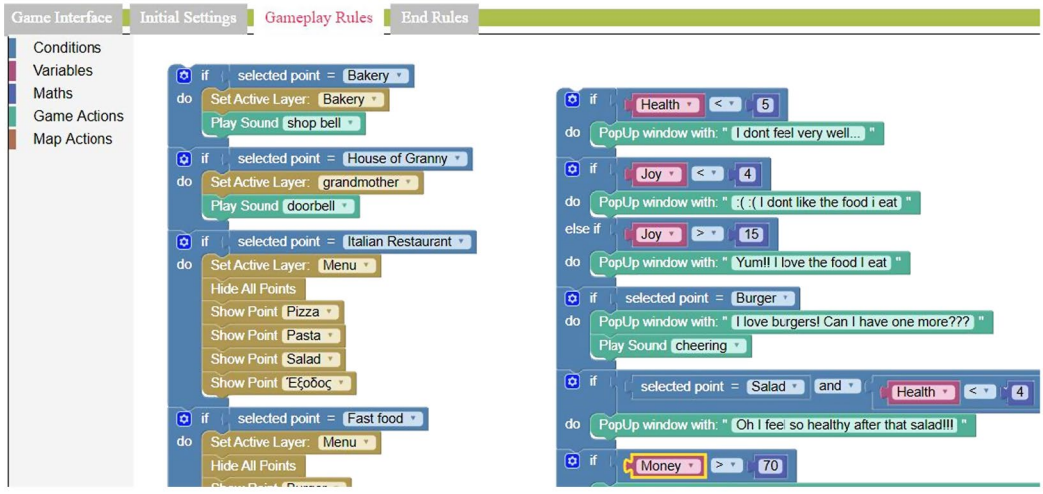

Figure 2: Block-based programming editor in ChoiCo

are inevitably based on the personal views and assumptions of the designer (Friedman, 1999). For example, "SimCity" assumes that low taxes will lead to city growth and in "The Sims" games watching TV increases your Sim's mood, but the player may have a different opinion. We see a potential to that games to raise discussions about the subjective principles and increase the motivation of players to change them.

The "Eating out" game simulates the eating habits of a citizen who visits places in a city centre to eat and in each of them makes food choices (Figures 3 and 4). Every choice affects the four game attributes: "Hunger," "Health," "Money" and "Joy." The game is built on our assumptions on what is healthy, delicious or filling and we expect from the students to question these values and modify the game to express their own views on the issue.

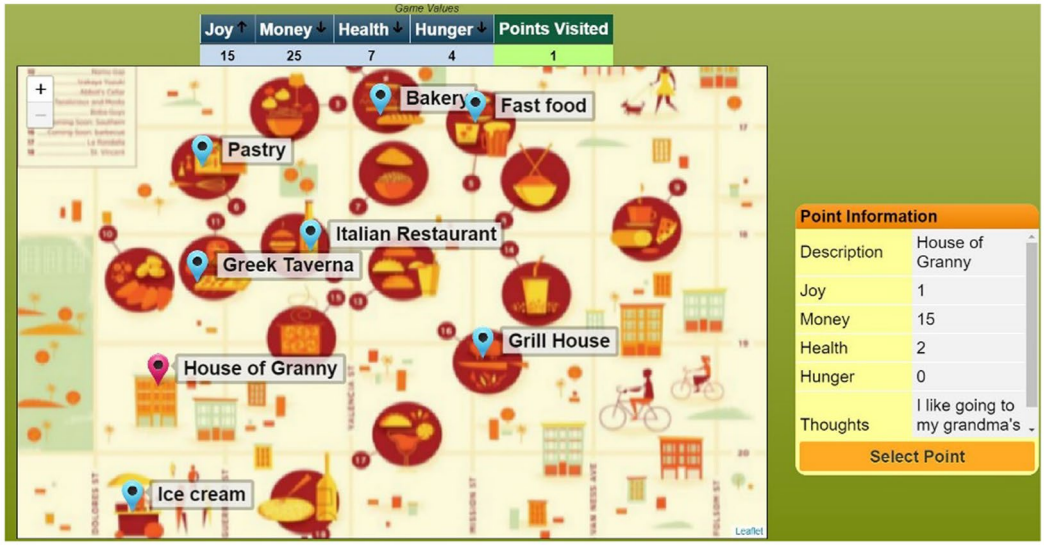

Figure 3: "Eating Out" game—City area

Figure 4: "Eating Out" game—House of granny area

The engineered bugs

To make the game a half-baked artefact (Kynigos, 2007), we integrated two "bugs" in the game system. They are both logic errors which do not affect the gameplay, but they make the game system unbalanced.

The first is the point "House of granny" that has only positive consequences in contrast to all other game choices that cause both positive and negative consequences to keep the game

balanced. If a player discovers this inconsistency, (s)he can select this choice unlimited times without losing.

The second "bug" was to programme a conditional statement that ends the game when *Money* is lower or equal to 15. Even though this is correct in terms of coding, it is not logical in the game context, considering that the initial value of *Money* is 25. Our aim is to trigger discussion about the role of conditional statements and variables in the game rules and the gameplay and urge students to modify them.

Methodology

Task design

For the design of the modding activities, we adapted the three stages model of progressive engagement with CT "Use-Modify-Create" (Lee *et al.*, 2011). The model suggests an increasingly deep engagement of students with the affordances of a computational environment that would transform them from users to creators. In our case, students are engaged with the three stages of game modding "Play-Fix-Create Mod," with the last two stages including a repeating cycle of game modifications (Figure 5). Students start by playing the half-baked game, then, triggered by the engineered bugs or the simulation, access the game structure to improve it with small modifications and finally they collaboratively create a new version of the game. As they proceed with the activity their role changes from player to evaluator and finally to designer of their own game.

Based on the above model the empirical study was divided into three parts. For the first part, each group of students plays the half-baked game "Eating out" several times aiming to make as many choices as possible. They also answer brainstorming questions in a worksheet such as "How did you achieve your high score?", "Discuss and decide with your team three things you don't like about the game." For the second part, students load the game in the "Design Mode" of ChoiCo and they are asked to fix the problems they had already detected. The improved game is given to another group to test it and provide feedback. For the third part, each group is asked to create a new version of the game that expresses their personal ideas. The final mod is given to another group to play and evaluate it. Then students could make improvements according to the feedback they received from the other group.

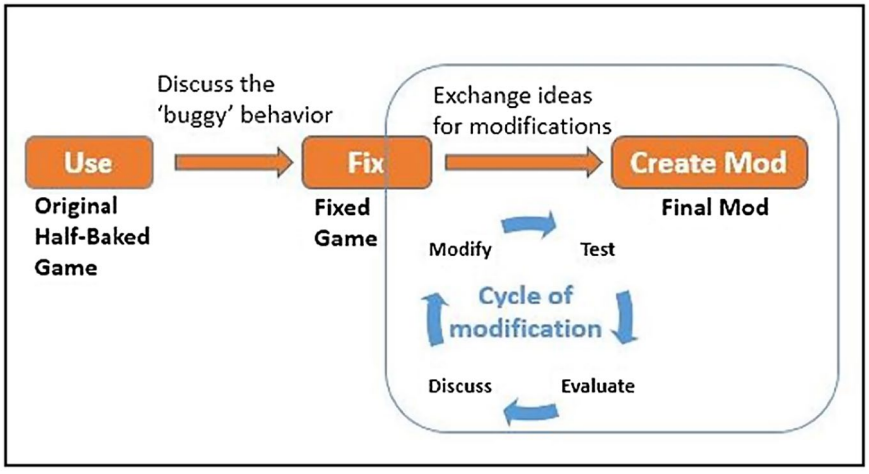

Figure 5: Three stages of progressive engagement with game modding

Context

To investigate the research questions of this paper, we organised a 9-hour study in a Greek middle school as an after-school activity. The participants were 11 students (7 boys and 4 girls) from the third grade, aged 14–15 years old who participated voluntarily. They had little or no experience with block-based programming in Scratch and no previous experience with game design or modding. The students worked collaboratively in mixed gendered groups of 2–3 persons in the computer laboratory using one computer per group.

The research method we used is that of Design-Based Research (Edelson, 2002). The study was carried out through three repeated cycles of design and implementation, utilising every implementation as an opportunity for data collection, evaluation and review in two levels of design: (1) regarding the modding activities and (2) regarding the ChoiCo environment in terms of usability and added value.

Data collection and analysis

During the study, we collected a set of data that included screen and audio recordings of each group with a total duration of 45 hours, semi-structured, artefact-based interviews (Brennan & Resnick, 2012 with each group at the end of each implementation, student games in different stages and student worksheets. Before the analysis, we removed any personal information from the transcripts and we replaced student names with aliases.

We did a qualitative analysis of the dataset using the "critical incident" as the unit of analysis (Tripp, 2011). As critical incident, we consider a representative moment of student meaning-making process about CT ideas in the context of game modding. For the identification of critical incidents, we correlated student discourses, their modding actions as occurred from screen capturing and their relevant replies to the interviews and worksheets. After the first analysis, we had a set of approximately 250 critical incidents which were further analysed with a set of codes in order to identify patterns of critical incidents emergence. The coding schema was reviewed and evaluated by an external researcher from the field of computer science and mathematics education.

Results

By the end of the study all five groups had created two mods of the original game. The first was a slightly different version with improvements on the half-baked game while the second had more sophisticated modifications. Table S1 of Appendix 1 presents the 5 final mods created by the 5 groups and the modified elements of each. All groups modified the code of the ending rules and the consequences values in the database records. Four of them changed the game parameters or added new ones and three of them added created an alternative game scenario (story, graphics and game areas).

Exploring the game system in levels

We employ the concept of "levels," introduced by Wilensky and Resnick (1999), to interpret how students explore and realise computational ideas by shifting between levels of the simulation game system. It seems that they approached and modified the game in two levels: (1) in a micro-level, that involves any individual modification to a specific game element (eg, change an ending rule, modify the data in the database for some objects, change the graphics of an area, etc) without considering the rest of the game and (2) in a macro-level that refers to the transformation of the game as a system in which all these individual modifications from the first level connect and obtain meaning. As the activity unfolded students progressively realised that making changes to one level lead to different behaviours and patterns at another level. The iterative

transition between the two levels helped them to express and develop meanings on computational concepts and practices as well as combine computational ideas to achieve a meaningful modification. Table S2 of Appendix 1 shows the most common computational ideas expressed by the students and the modding actions they performed at the time of meaning generation.

It seems that ChoiCo affordances enabled restructurations of complex computational ideas. With this term, we refer to *reformulations of knowledge disciplines through new representational and communicational forms* (Wilensky & Papert, 2010). For example, with the database affordance, students were able to experiment with data structures (records) and observe the results in the gameplay. Data structures and classes are considered advanced programming concepts and students are usually introduced to them in last classes of high school or university. Additionally, in many cases, students combined different affordances to access and understand computational ideas like the concept of variable, which was represented as a game parameter, as a database field and as a code variable.

In the following sections, we discuss representative student meaning-making strategies that emerged from the analysis of the critical incidents.

Exploring patterns in the game consequences

Pattern recognition refers to the ability to identify and match similarities (patterns) between computational elements as a way of gaining extra information for them (Grover & Pea, 2018). Recognising patterns is the first step for generalisation and abstraction. In our study, students explored patterns as a computational practice to balance the game consequences. The critical incident analysis suggests that they were engaged in three steps of meaning making on patterns: (1) pattern recognition, (2) pattern analysis and (3) pattern generalisation. Below, we elaborate on the three stages with examples of representative critical incidents.

During the "play" stage students discovered and verbally described patterns of the simulation consequences as part of the gameplay. Then they used these patterns to improve their play strategy like the students in critical incident 1.

Mary: "All the tasteless foods increase your health but also reduce the joy, while the tasty ones do the opposite"

John: "So we can achieve a high score if we keep selecting a healthy food 2,3 times then a tasty but unhealthy food and go on like this"

Critical incident: 1

Furthermore, they used patterns to detect and describe the bug of the choice "house of granny." For instance, Mary mentions as she plays the game: "Look! None of the other buildings gives you money except from the house of granny. That's a cheat!" Mary categorises the game choices according to their consequences and then she refers to the category "Buildings" where the "house of granny" belongs. She describes a pattern of consequences that all the "buildings" follow and the house of granny is not. George, a student from group 1 says to his teammates: "We can keep selecting the house of granny forever and we will never lose! Unlike all the other choices of the game, this one has only positive effects on your score!" George describes a different pattern from Mary. He approaches the problem in a macro-level, identifying a pattern that applies to all choices. The problem, according to George, is that all choices must have both negative and positive effects (consequences) but the house of granny is not. Both students, describe the "bug" as inconsistency to a pattern that prevents a meaningful gameplay.

In the second stage, the database affordance allowed students to further analyse the patterns in the data cells values. During the analysis of the game data, they found patterns of consequences between different choices and categorise them to certain types. The two students in critical incident 2, refer to the game choices as records in the database with properties represented by the fields. They modify the value of the cell "Hunger" of the record "House of Granny" to match the pattern of other records of the same "type" (all buildings). They also seem to realise the game as a system in which a small change in one value (micro-level) effects the final gameplay experience (macro-level).

John:: Look at the column of the parameter Hunger. All buildings have positive numbers which means they will increase it apart from the house of granny
Mary:: Ok. I will make the value of Hunger for the house of granny to be five
John:: Or even greater to be more balanced. Because, this choice also increases Money

Critical incident: 2
In the "Create Mode" stage, students designed abstract rules for the data records in order to balance the game mod. Group 2 created the "Travelling in Europe" mod in which the player choses between different European cities to visit and things to do in them. In order to achieve a balanced game, the students created five general categories of game choices that followed specific rules for their consequences (Figure 6). For instance, the "cities" category had the following rules: "Joy: 0, Money: −10 to −200, Tiredness = -(Money/10)." Then they added game choices as instances of the categories (eg, Rome, Istanbul, London). As they claimed in the interview "It was much easier to create a balanced game that way. We were sure that all choices of the same category follow the same pattern regarding their consequences." In this strategy, we can see an expression of generalisation practice since students generalise the patterns to abstract categories that resemble the concept of classes or structures in programming.

ID	Description	Joy	Money	Tiredness
42	London	0	-100	10
51	Barcelona	0	-105	11
55	Instabul	0	-20	2
59	Hotel 1	40	-100	-5
63	Hotel 2	15	-80	1
67	Hostel	-20	-15	15
71	Motel	-45	-5	20
75	Colosseum	15	-10	5
79	Batican	20	-15	10

Figure 6: Implementation of patterns in the database

Programing a meaningful gameplay

Rules is a fundamental element in game design. The rules bind together the game components, define the game interactions and feedback, control the story and form the gameplay experience (Salen & Zimmerman, 2005). Similar to the game patterns described before, students expressed meanings for computational ideas as they modified the rules in micro and macro level. At first, they made micro-level modifications to fix individual game rules, like increasing the number 15 of the conditional statement "if Money < 15 then gameover()." As the process of modding progressed, they conceived each algorithm element as part of the whole game system (macro-level) and they considered how a modification would affect the final gameplay. Critical incident 3 is a representative moment of students meaning expression while they use all three affordances to shift between levels and to create a meaningful gameplay.

Nick: I will set the initial value of Danger to 0
George: But…if the initial value is 0 and your first choice is "Leave school" you will immediately lose. Because, look here (opens the database) "Leave school" increases Danger by five points.
Nick: And why will you lose?
George: Because, here (opens the code) we have programmed the game to end if Danger is greater or equal to 5
Nick: Oh! You are right. Ok change the limit of the statement to 100
James: But then the game would be too easy! Look (opens the database). All the choices increase or decrease danger by 5–10 units. It is almost impossible for a player to reach 100. 50 makes more sense.

Critical incident: 3

In this episode, Nick makes a micro-level modification (setting the initial value of Danger variable to 0). This modification is correct in terms of programming, but it is not meaningful in the game macro level context. George explains this, by verbally "running" the algorithm in a hypothetical gameplay scenario. Nick corrects the code but only for this specific scenario (*change the limit to 100*). Then James generalises further the solution analyses all possible choices and their consequences. He suggests a limit that would create a meaningful gameplay, ie, neither too hard but nor too easy. The diverse affordances seem help students generate meanings about the sequence of commands and game events in micro and macro levels.

Discussion

The presented results of the empirical study cannot be generalised due to the small number of participants and to the short time of implementation. However, they provide some first insights on how students explore and express meanings on computational concepts and practices when they modify a half-baked simulation game with diverse affordances.

We studied student activities in the context of employing tools and engaging in practices characterised by an integrated approach to programming. Our method consisted of combining a number of diverse didactical techniques, tools and approaches to learning. We elaborated the idea of questionable socio-systemic games integrating concepts and values (Kynigos & Yiannoutsou, 2018) and gave students didactically engineered half-baked games to make changes to by modding them. We used black and white box techniques (Kynigos, 2004) so that the students would handle a mix of generic and higher order building boxes allowing them to grapple with technically, structurally and semantically complex digital artefacts. We perceived programming to be a process (Papert, 1980), a way of thinking which is learnable and can be cultivated and which to the students looks relevant with respect to real-life issues and their own realities, yet embedding important computational ideas to be used and given meaning to. So, we tried to analyse student activities in this integrated context. Aspects of these activities emerged through our analysis

which reveal how hard it is to understand the value of programming as a literacy, an expressive act of a digital citizen. An example of three of these aspects follows.

First, students got engaged in a high-level process of testing and debugging by questioning the rules of the half-baked game and then modify the parts related to the engineered bugs. Debugging is considered a core practice of CT (Brennan & Resnick, 2012; Kafai & Bruke, 2014) but also one that novice programmes face severe difficulties to implement, since they tend to perceive each command as an individual part of the programme (Robins *et al.*, 2003). The fact that, in our study, students approached the computational system as a fallible artefact (in the sense of Ernest *et al.*, 1991) allowed them to construct, express and test personal meanings about it. During gameplay, they discovered and discussed ideas about the game faulty behaviours which then they "translated" into meanings for computational concepts and debugging practices while they fixed and improved the game eg, the game choices to database records, the game parameters to variables, etc. It seems that half-baked game design combined with the subjective issues of a simulation game can offer a familiar context for students that is also open to questioning. In addition, "black and white" box design, allowed them to focus on the computational concepts and practices that caused the faulty behaviours, avoiding the noise of searching and analysing the whole game system from the beginning.

Second, students utilised the three affordances as a means to analyse the game structure in micro and macro levels and to access otherwise complex ideas as data structures, interface layers, variables and UI events. The affordances provided students with restructurations of complex computational concepts and practices allowing them to generate meanings in the same way that agent-based simulations allowed students to investigate complex scientific concepts in the studies of Wilensky and Papert (2010). In addition, the multiple representations of computational concepts eg, the variable, helped students not only to construct but also to transfer, combine and transform meanings. A characteristic example was student meaning generation process about the programming variable, a concept about which they usually have significant misconceptions (Armoni, 2013; Robins *et al.*, 2003). In our study, a game variable had three representational forms, as a parameter in the game interface, as a database field in the database and as a block in the programme. When students added a new parameter in the game, they defined it in the database as a new field, they initialised it and controlled it with block-based programming and they tested it by playing the game with the map editor. Thus, they expressed meanings for the concept of programming variable in different contexts (database, programming, game) and they had to combine their understanding of variable in these three representations in order to use it correctly and meaningful in the game.

Third, students generated meanings about computational practices and concepts through a progressive process of understanding that sometimes involved the transformation and combination of concepts they already knew. It seems that the integration of the "Use-Modify-Create" model for CT (Lee *et al.*, 2011), with the deep structural access to the functionalities of the digital environment (diSessa, 2001), allowed for student activity around the computational ideas of the game system. With this design, game modding acted as a scaffolding for students allowing them to gradually build mental models. For instance, students developed understanding about patterns through three progressive steps of meaning-making: pattern recognition, pattern analysis and pattern abstraction. In every step, they conceived and approached patterns in a different context and in correlation with other computational ideas.

This study showed that students may acquire and develop CT when they explore and combine concepts and practices in an integrated context. Maybe, therefore, it is time to seek for pedagogical designs that would support this integration of CT ideas, instead of studying and teaching CT concepts or practices as individual clusters of knowledge. To achieve this integration, there is a

need for deeper analysis and elaboration of the learning processes that emerged in this study. Further research, including, for instance, the study of student activities in diverse game designs and settings, could reveal new aspects of CT development in integrated contexts that were not scientifically present in this study such as abstraction and decomposition. Based on the discussed results, we are currently redesigning the activities and the half-baked game in order to organise a second, larger study that will focus on answering these questions.

Acknowledgements

The research of Grizioti Marianthi has been financially supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 531).

Statements on open data, ethics, conflict of interest

The half-baked game "Eating Out" as well as the final game mods created by the students can be found online in ChoiCo website "etl.ppp.uoa.gr/choico." The games have been translated in English language and any identifications of students have been removed. The transcribed data of audio recordings and interviews have been decoded so that there are no data of student identity and they are stored in the servers of NKUA. Parts of them can be provided after a request and a justification of use to the authors. The data are currently available in Greek language but parts of them can be translated if necessary.

The study was organised as an after-school event entitled "Play and modify video games" and participants signed up voluntarily in an online Google form. Parents were informed for the event and the study from the school math teacher and from the researcher. Before the study students and their parents provided written consent for staying extra hours after school, for audio recordings and interviews.

There is no conflict of interest in the current research. The software (ChoiCo) that is used is open source and has been developed by Educational Technology Lab of National and NKUA.

Notes

[1]ChoiCo is freely available at: http://etl.ppp.uoa.gr/choico

[2]http://etl.ppp.uoa.gr/choico

References

Amory, A., Naicker, K., Vincent, J., & Adams, C. (1999). The use of computer games as an educational tool: Identification of appropriate game types and game elements. *British Journal of Educational Technology*, *30*(4), 311–321.

Armoni, M. (2013). On teaching abstraction in CS to novices. *Journal of Computers in Mathematics and Science Teaching*, *32*(3), 265–284.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada. 1–25.

Calder, N. (2012). *Processing mathematics through digital technologies*. Dordrecht: Springer Science & Business Media.

diSessa, A. (2001). *Changing minds: Computers, learning and literacy*. Cambridge, MA: MIT Press.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *Journal of the Learning Sciences*, *11*(1), 105–121.

El-Nasr, M. S., & Smith, B. K. (2006). Learning through game modding. *Computers in Entertainment*, *4*(1), 7–27.

Ernest, P., Skovsmose, O., Van Bendegem, J. P., Bicudo, M., Miarka, R., Kvasz, L., & Moeller, R. (1991). The Philosophy of Mathematics Education. In: *The Philosophy of Mathematics Education. ICME-13 Topical Surveys*. Cham: Springer.

Friedman, T. (1999). The semiotics of SimCity. *First Monday*, *4*(4). Retrieved from: https://journals.uic.edu/ojs/index.php/fm/article/view/660/575

Gee, J. P. (2008). Learning and games. *The Ecology of Games: Connecting Youth, Games, and Learning*, *3*, 21–40.

Grizioti, M., & Kynigos, C. (2018). Game modding for computational thinking: An integrated design approach. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (pp. 687–692). Trondheim, Norway: ACM.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE*, Coimbra, Portugal.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In *Computer Science Education: Perspectives on Teaching and Learning in School*, (pp. 19–37). London: Bloomsbury Academic.

Kafai, B. Y., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kynigos, C. (2004). A black and white box approach to user empowerment with component computing. *Interactive Learning Environments, Carfax Pubs, Taylor and Francis Group.*, *12*(1–2), 27–71.

Kynigos, C. (2007). Half-baked Logo microworlds as boundary objects in integrated design. *Informatics in Education*, *6*(2), 1–24.

Kynigos, C. (2015). Constructionism: Theory of learning or theory of design? In S. J. Cho (Ed.), *Selected Regular Lectures from the 12th International Congress on Mathematical Education* (pp. 417–438). Cham, Switzerland: Springer International Publishing.

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating Turtle geometry, dynamic manipulation and 3D Space. *Informatics in Education*, *17*(2), 321–340.

Kynigos, C., & Yiannoutsou, N. (2018). Children challenging the design of half-baked games: Expressing values through the process of game modding. *International Journal of Child-Computer Interaction*, *17*, 16–27.

Lee, I., Fred, M., Jill, D., Bob, C., Allan, W., Jeri, E., … Linda, W. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32–37.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books Inc.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172.

Salen, K., & Zimmerman, E. (2005). Game design and meaningful play. In *Handbook of computer game studies* (pp. 59–79). Cambridge, MA: The MIT Press.

Sotamaa, O. (2010). When the game is not enough: Motivations and practices among computer game modding culture. *Games and Culture*, *5*(3), 239–255.

Tripp, D. (2011). *Critical incidents in teaching (classic edition): Developing professional judgement*. New York: Routledge.

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulations of knowledge disciplines through new representational forms. In *Proceedings of the Constructionism 2010 Conference* (p. 97). Paris, France.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and technology*, *8*(1), 3–19.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–372.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.