Exploring How Game Genre in Student-Designed Games Influences Computational Thinking Development

Giovanni Maria Troiano[1], Qinyu Chen[1], Ángela Vargas Alba[2], Gregorio Robles[2], Gillian Smith[3], Michael Cassidy[4], Eli Tucker-Raymond[5], Gillian Puttick[4], and Casper Harteveld[1]
[1]Northeastern University | {g.troiano, c.harteveld}@northeastern.edu; chen.qinyu@husky.neu.edu
[2]Universidad Rey Juan Carlos, | grex@gsyc.urjc.es; a.vargasa@alumnos.urjc.es
[3]Worcester Polytechnic Institute | gmsmith@wpi.edu
[4]TERC | {michael_cassidy, gilly_puttick}@terc.edu
[5]Boston University | etuckerr@bu.edu

ABSTRACT
Game design is increasingly used in modern education to foster *Computational Thinking* (CT). Yet, it is unclear how and if the game genre of student-designed games impact CT and programming. We explore how game genre impacts CT development and programming routines in *Scratch* games designed by 8th-grade students using a metrics-based approach (i.e., *Dr.Scratch*). Our findings show that designing particular games (e.g., action, storytelling) impact CT and programming development. We observe, for instance, that CT skills develop and consolidate fast, after which students can focus on aspects more specific to game design. Based on the results, we suggest that researchers and educators in constructionist learning consider the impact of game genre when designing game-based curricula for the learning of programming and CT.

CCS Concepts
•Human-centered computing → **Human computer interaction (HCI); •Applied computing** → *Interactive learning environments;*

Author Keywords
Game-based learning, game design, computational thinking, video games, Scratch, Dr. Scratch, constructionist learning

INTRODUCTION
Game-based learning is emerging in contemporary education and its benefits have been extensively discussed [109, 113, 67, 4, 128, 32, 120, 25, 112, 110, 27, 125]. Previous work, for instance, showed how video games allow for effective learning of the science and engineering in parallel [64]. Others used games to help support the education of underserved students [25]. The benefits of game-based learning are generally exploited through (1) an *instructionist* (or *play-centric*) approach [26], where students play *serious games* [113, 67, 6, 94] to learn (e.g., [80, 25, 55, 102, 109]), or (2) a *constructionist* approach [46, 51, 115, 81, 90, 95, 48, 91], where students learn (as in construct knowledge [85]) through creating artifacts. As modern education moves more towards constructionist learning, *game design* [23] is prominently used in the classroom as a learning tool (e.g., [35, 121, 7, 88, 89, 118, 19, 77, 9]). Nowadays, students design games [79] to learn in many educational contexts and about various topics, including problem-solving [103], computer science [78], and climate science [7, 89, 118, 42].

Recently, game design has emerged in constructionist learning for computational thinking (CT) [74, 75, 71, 96, 42, 2, 76, 133, 58, 20, 118]. In game-based constructionist curricula (e.g., STEM [88, 89, 118]) the focus is *"teaching coding and academic content through game making"* [51]. As such, teachers may ask students to design games on specific topics (e.g., climate change [118]), for learning scientific content while developing CT. While current game-based curricula for CT focus much on content uptake, the impact of what students design on CT and programming is mostly unexplored. Previous work [71], however, showed that what students design (e.g., video games, music applications), does impact CT.

In this paper, we investigate if and how the game genre of student-designed video games impact CT development and programming routines, both as (1) CT proficiency, (2) the development of CT proficiency over time, and (3) programming in Scratch [93] as *block usage* [18]. We examine 404 Scratch video games designed by 8th-grade students using a metrics-based approach (i.e., Dr.Scratch [73]), and explore the impact of game genre on CT development, both overall and for each individual CT dimension (see [118]), and on block usage. To identify the game genres, we perform emergent coding [11] based on the *Triadic Game Design* (TGD) model [37].

Through our analysis, we show how CT develops differently in the early phases of game design based on game genre, and thus provide empirical evidence that game genre impacts both CT development and programming routines; to our knowledge, no previous work assessing CT in game-based constructionist curriculum provided such evidence. We discuss how our results have implications for CT assessment and conclude by offering

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
https://doi.org/10.1145/3313831.3376755

pedagogical recommendations for researchers and educators working with constructionist learning, CT, and game design.

RELATED WORK

In this paper, we focus on how game design practices (in the instance of game genre) impact students' development of programming and CT skills in constructionist STEM curricula. As such, we position our work in (1) game-based constructionist learning (specifically game design) and (2) programming practices and CT assessment (in Scratch via Dr.Scratch). Next, we briefly review related work in those areas.

Game(Design)-Based Learning

Game design is becoming widespread as a creative activity that supports students' learning [87, 53, 60, 92, 127, 89, 88, 118], and it has been used to teach a variety of topics, including science [78, 59], environmental awareness [54], climate science and systems thinking [42, 7, 88, 89, 118], problem-solving [1], and software engineering [15]. Earlier work discussed the benefits that game design brings to education as a learning tool [51, 125, 79, 23]. In that respect, Ke [55] showed how students that personally re-elaborate concepts through game design are highly engaged by the learning process, and in turn increase their chances of becoming proficient in the topic of interest. As a learning activity, game design is grounded in the educational philosophy of constructionism [46, 51], where students are prompted to "construct knowledge" [81, 91], rather than being "instructed to it" (for a review, see [49]). As such, creating digital artifacts (i.e., video games) for learning purposes has a long tradition within constructionism.

Dating back to the early 1980s, Seymour Papert used LOGO [82, 90] to let young students learn creatively by programming and designing games [81]. More recently, Resnick [93, 62] further evolved the concept of LOGO into Scratch, which is now widely used as an online game design platform, and its ever growing community counts thousands among young game designers and programming enthusiasts. Thanks to Scratch, game design has become widespread not only in informal learning [84], but in the modern classroom too (e.g., [71, 74, 66, 41, 124, 31, 30]).

In this paper, we analyze how game genre influences programming routines (i.e., block usage) and CT development of Scratch games designed by 8^{th}-grade students in constructionist curricula. Hence, the way we refer to game(design)-based learning, programming practices, and CT development throughout the rest of this paper is primarily in the context of designing games with Scratch.

Programming Practices and CT Assessment

Programming [86, 50] and computational thinking (CT) [132, 126] are key in modern education for digital literacy [29, 57, 24]. The definition of programming has long been discussed [39], and as McCracken [65] put it down, programming is the *"process of translating from the language convenient to human beings to the language convenient to the computer"*. In the context of Scratch this is exploited through block-programming [101]. As we analyze games designed in Scratch, in this paper we refer to programming practices as the way in which students use Scratch blocks (and which ones they use more frequently), to design their games in a constructionist STEM curriculum focused on climate science. As for CT, its definition is still debated among researchers and understood differently depending on the context in which it is applied (see [107]). Nevertheless, CT is generally intended as the process of expressing problems and solutions that can be executed by computers (for a review of CT definitions, see [126, 131, 111]). CT is also often regarded as a competence associated with problem-solving [61, 123, 52, 108].

Previous work proposed various approaches and techniques for assessing CT (e.g., [130, 21]). Among quantitative approaches, the *computational thinking test* (CT-t) [98, 99, 100], for instance, assesses CT skills through a multiple choice test that allows for one correct answer on various programming and CT topics. Another example of CT assessment is the Progression of Early Computational Thinking (PECT) [106], which assesses CT in Scratch on three main variables: (1) *evidence variables* (i.e., the actual code written in Scratch), (2) *design pattern variables* (i.e., how programming strategy and patterns are implemented), and (3) *CT concepts* (i.e., abstraction). Dagiene et al. [16, 17] proposed the *Bebras* model, to assess CT based on five CT dimensions (e.g., algorithmic thinking, decomposition).

Others assess CT through qualitative approaches. For instance, Brennan and Resnick [14] consider CT as a mixture of computational concepts, practices, and perspectives, and assess CT through portfolio-based and artifact-based analyses, while Thomas et al. [116, 117] used journal annotation for exploring the difficulties that African-American middle-school girls face as they collaborate while engaging in *computational algorithmic thinking* (CAT), as well as the strategies they employ to address such difficulties.

In this paper, we focus on metrics-based approaches to assessing CT (e.g., [63, 18]). In particular, we use Dr.Scratch [73] to assess the CT of student-designed games in constructionist STEM curricula. Dr.Scratch [73] is a web application based on Hairball [13], which automatically assesses CT in Scratch projects based on a 0 to 3 points scoring mechanism (see Table 1), and for seven CT dimensions. Most work assessing CT via Dr.Scratch focused on the final score achieved by Scratch projects (i.e., *CT proficiency*) [71, 74, 73, 76, 75].

However, previous work also used Dr. Scratch to explore how CT develops in Scratch projects, from beginning to end [118]; the work showed how the CT of student-designed games for STEM develop differently in each CT dimension defined by Dr. Scratch, revealing that *parallelism*, *synchronization*, and *logic* develop proficiently throughout, that *user interactivity* and *data representation* develop early but never reach proficiency, while *abstraction* shows little development. Here, we expand on the above mentioned and explore how CT proficiency and development are influenced by game genre in student-designed games. Furthermore, we integrate an analysis of block-usage in Scratch based on game genre.

Table 1: *Dr. Scratch* metrics, showing competence level for each CT dimension and relative *Scratch* practices [69].

	CT Dimension	Competence Level		
Term	Definition	Null (0)	Basic (1)	Developing (2)	Proficient (3)
Abstraction	The ability to conceptualize and then represent an idea or a process in more general terms [126]	—	More than one script and more than one sprite	Definition of blocks	Use of clones
Data representation	Representing data through abstractions, such as models and simulations [8]	—	Modifiers of sprite properties	Operations on variables	Operations on lists
Flow control	A high-level way of programming a computer to make decisions, simple or complicated, executed once or multiple times [105]	—	Sequence of blocks	Repeat, forever	Repeat until
Logic	Conditionals and rules that allow to build up and represent complex ideas [105]	—	If	If else	Logic operations
Parallelism	Handling multiple scripts or sequences of code that run simultaneously [83]	—	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is >%s, two scripts on when backdrop change to
Synchronization	The coordination of simultaneous threads or processes	—-	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
User interactivity	Designing and programming for user input	—	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio

METHOD

In this section, we describe the curriculum in which the games were designed, the data collection and analysis, and how we labeled game genres in student-designed games.

Contructionist STEM Curriculum

The serious games we asses and analyze in this paper were designed by 8th-grade students (13 to 14 years old), as part of a constructionist STEM curriculum focused on climate science and systems thinking. The constructionist curriculum was based on participatory pedagogy approach [5], to allow students to shape content and solve problems in a collaborative fashion, use others' games for inspiration and ideas [10], and develop self-efficacy [45, 119]. The curriculum was implemented in 35 science classes over three years (approximately 19 students per class), at four different schools by nine teachers. Prior to the curriculum implementation, all teachers participated in a professional-development program, which facilitated their understanding of programming in Scratch and CT. While the program aimed to address teachers' knowledge, gaps, and comfort level [68], it is important to note that most teachers had no prior experience with either Scratch or game design. However, the teachers were given the freedom to adjust the curriculum according to their available class time and objectives. The implementation took between four to six weeks, and while all teachers followed the curriculum closely, differences were noticeable among their implementations.

The curriculum entailed students and teachers exploring together climate change phenomena (e.g., ice-albedo feedback, global warming, energy consumption), and later picking a climate change topic to explore via game design. As such, as part of the curriculum, students were tasked to program and design serious games in Scratch, which represent gamified versions of climate change topics of their choice (e.g., CO2 emission). Before designing their games, the students were tasked to play a variety of existing video games on climate change, for instance NASA's *Offset*[1], and *Power Up*[2] and critically think about how those games were designed to convey educational messages; this exercise was necessary to let student understand the different design characteristics between a "regular" video game and an educational one. As students were asked to design video games using Scratch, they were initially introduced to block-programming through a 10-Block challenge; the scope of the exercise was to have students get familiar with the platform by programming a simple application in Scratch made with only 10 blocks. After the 10-Block challenge, teachers and students refined initial ideas on the climate science games, and students started working on their Scratch projects in pairs.

Data (Scratch Games) Collection

The students created their games on the Scratch website and were instructed by teachers to give their projects a tag-name, to make their games easily identifiable. The games were then stored in an external repository, from which we could retrieve the data for later analysis. To track and collect the data (e.g., block usage, time-stamps) contained in the compressed .sb2 files, we used the API provided by Scratch developers. Following previous work [118], we tracked progress in Scratch projects using a Python script that takes a snapshot at one minute intervals, and only stored those snapshots where changes in code occurred.

CT Proficiency and CT Development Assessment

Dr.Scratch [73] evaluates the CT proficiency of student-designed games in Scratch, overall (i.e., on a scale from 0 to 21), and on each of the seven CT dimension provided by Dr.Scratch (i.e., on a scale from 0 to 3, see Table 1). For CT

[1] https://climatekids.nasa.gov/offset/
[2] https://climatekids.nasa.gov/power-up/

development, we follow the temporal progress of CT scores in student-designed games using a snapshot-based analysis. As the numbers of snapshots varied greatly, ranging from 5 to 375, we normalize those numbers to represent their CT development coherently across projects. Previous work [118] normalized snapshots distribution using quartiles. Instead, we use deciles (i.e., a total of ten intervals between snapshots), to provide further details in the CT development analysis. For instance, if a project has 21 snapshots, D_0 would be snapshot 1, D_{10} would be snapshot 20, while $D_1 - Q_9$ would be evenly distributed between those snapshots (i.e., snapshots 2, 4, and 6, etc.). As 79 among the 404 Scratch projects had less than 11 snapshots, the number of available projects for the CT development analysis was reduced to 325.

Scratch Block-Usage Analysis

We analyze how design practices (i.e., game genre) impact CT. However, as CT practices (and development) in Scratch are deployed through block-programming [101], we also analyze how students made use of Scratch blocks to design their games, and observe how block usage is influenced by game genres. Scratch features 10 categories of programming blocks (see Table 2). In our results, we will not report the use of *more blocks* (i.e., customize), as upon a preliminary count of block usage those blocks were considerably lower than other blocks. We analyze the other nine block categories and retrieved the counts for each project using Hairball [13]. We consider how those nine categories of Scratch blocks were used across projects at large, as well as how they were used across different game genres in student-designed games. We analyze only the main block categories and do not consider their sub-categories (i.e., hat, stack, reporter, Boolean, and cap blocks). As Dr.Scratch scores CT when a particular block is used, but only on its first instance, a block-usage analysis will allow us to capture iterative programming and design practices that may not be captured by the Dr.Scratch metrics analysis.

Labeling the Game Genre of Student-Designed Games

We labeled the student-designed games through emergent coding [11]. We initially used the TGD model [37] as frame of reference for our coding, as the model provides a comprehensive description of the most basic and common game genres (i.e., action, puzzle). Furthermore, we considered the work of Heintz and Law [40] on game genre classification in HCI.

Table 2: Scratch blocks (version 2.0)

Blocks	Explanation	Example
Motion	Control a sprite's movement	Go to X: ()
Looks	Control a sprite's look	Say ()
Sound	Control sound and MIDI	Play Sound ()
Pen	Control the pen	Stamp
Data	Hold values and strings	Set () to ()
Event	Control events	Broadcast ()
Control	Control scripts	Wait Until ()
Sensing	Detect things	Reset Timer
Operators	Perform math functions	() < ()
More Blocks	Custom blocks	Define ()

Based on both the TGD and the work of Heintz and Law, we started discussing potential labels for student-designed games based on their game content and dynamics (e.g., providing fast-paced action, simulating real-life situations, containing riddles), and considered the following genres: (1) action, (2) simulation, (3) puzzle, (3) strategy, (4) quiz, (5) adventure, and (6) storytelling.

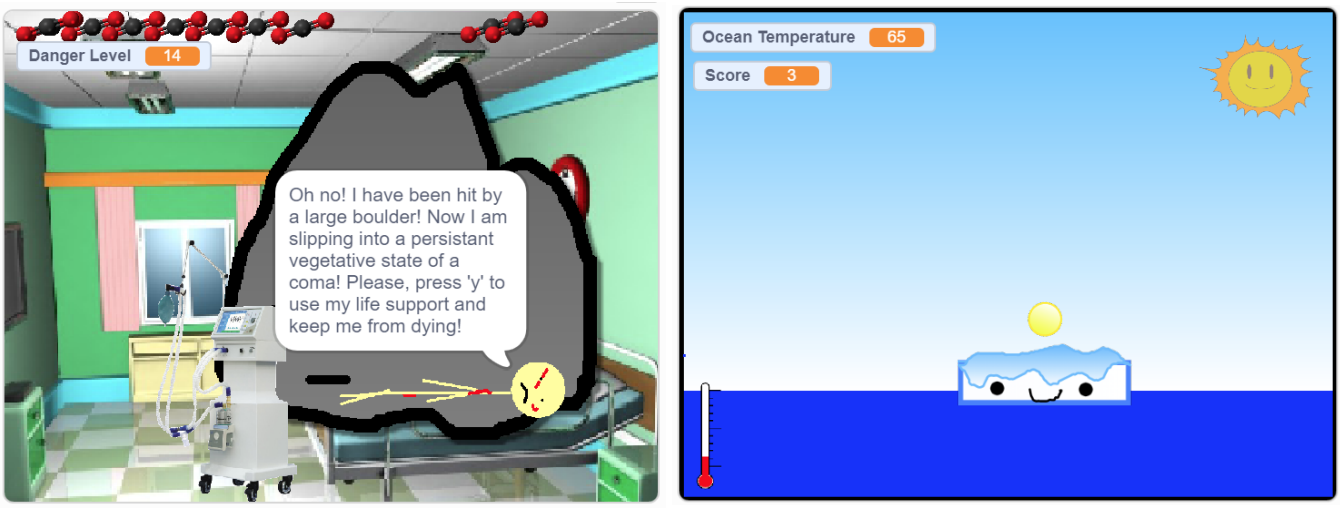
We coded the game genres for student-designed games as follows. Two coders independently coded an initial set of games and cross-validated their coding to achieve consensus on labels for game genres. We coded the genre in student-designed games using a *main-genre* and an optional *sub-genre* label. For instance, *pong* or *shooter* are sub-genres of *action*. The process was iterated over the entire data set until a mature coding scheme was established; coding saturation and consolidation of labels for game genres happened after approximately 100 games were coded in pair by both coders. While we could easily identify a main- and sub-genre for the majority of games, there were few games that clearly encompass two or more genres in the same game; we coded those as *multiple-genre* games (e.g., action-adventure). In the scope of our work, we focused on the identified genres with a sufficient number of games (see 3), and thus excluded these multiple-genre games. Furthermore, we removed game genres which labeled games in clusters of five or fewer (e.g., adventure), as this would have led to unfair comparison with other much bigger clusters. This exclusion reduced the number of analyzable games for CT proficiency from 404 to 391.

RESULTS

First, we provide an overview of the game genres that we identified in student-designed games, which give context for the CT assessment and Scratch block-usage results. Then, grouped by game genre, we show results from the Dr.Scratch assessment for the CT proficiency of 391 games, and for the CT development of 325 games (i.e., the score progression from D_0 to D_{10}). We then present the results for block-usage. Finally, we combine the results on genres, CT proficiency, and block-usage to get an overview of the role of genres from both CT and programming perspective.

Game Genres of Student-Designed Games

We initially identified five main game genres, namely (1) *action*, (2) *puzzle*, (3) *quiz*, (4) *simulation*, and (5) *storytelling*. However, as action games resulted in a considerably bigger cluster (n = 285) compared to others (e.g., storytelling, n = 25), we broke down action games into further sub-genres, to have a more fair comparison among different game genre groups. As such, we performed an additional round of coding on action games, to identify and label fitting sub-genres. From the emergent coding, it became evident—and surprisingly so—that a large number of games could be labeled after specific types of video games, for instance like the popular Atari™ *Pong* game, or after games that recently gained popularity through mobile gaming like incremental (or idle) games [3], whose game-play simply consists of repeatedly clicking on virtual objects that appear on screen; as such, we refer to those games as *clicker*. Pong and clicker games make together almost half of the student-designed games (n = 190, 47.0%). Table 3 shows



(a) Carbon Clicker. **(b) Albedo Pong.**

Figure 1: Examples of the two dominant game genres: (a) a clicker game; (b) a pong game.

the identified 10 game genres and their descriptive statistics. Below, we explain each game genre as understood in our emergent coding. Note that all student-designed games in our analysis were gamified versions of climate science topics (e.g., CO_2 emission). As such, we exemplify game genres based on their climate science content and game-play challenges.

Clicker Games: The players' task is to click on the screen for various purposes, including advancing between game scenes, or "pop" virtual objects. An example of a student-designed clicker game is *Carbon Clicker* (see Figure 1), a video game that has the player clicking through scenes of a story, which ends with planet Earth destroyed by the increasing pollution generated by human technology.

Maze Games: Players are tasked to find a way out of maze by navigating a space constrained by walls and obstacles; often, the student-designed games would reset players' position to default if they collide with virtual obstacles (e.g., walls). *The Duck Game* is one example of a maze game from our data set, where the player (a duck with an axe) must avoid trees on their way out of the maze, to not pollute the atmosphere by producing CO_2.

Platform Games: They have similar game-play to popular games like Nintendo®'s Mario Bros (i.e., avoid enemies, jump on platforms, grab objects). An example of a platform game designed by students is *Jump Into Climate Change!*, a game where the scope is to ride a bike and jump to avoid objects that can be harmful for the environment (e.g., plastic products).

Pong Games: They are inspired by the classic Atari™ video game, where players control a paddle to deflect a ball, in a player vs player (PvP), or player vs environment (PvE) configurations. An example of pong game from our data set is *Albedo Pong* (see Figure 1), where the paddle is made of ice and it will shrink more and more every time sun rays hit the ocean surface.

Puzzle Games: They contain riddles (e.g., matching objects in the right configuration) that must be solved by players to complete the game. An example of a puzzle game designed by students is *Comfortable Climate*, where the player is tasked to move between different rooms of an apartment, find appliances that increase CO_2 emission, and turn them off.

Quiz Games: They are based on question-and-answer game-play, much like *Trivial Pursuit*® [22]. An example of a student-designed quiz game is *Yum Project*, a game where the player has to respond correctly to what food is eco-friendly compared to other food options (e.g., oatmeal vs steak and cheese).

Shooter Games: They are based on spatial awareness and reflexes, and task players with shooting at moving targets (e.g., TAITO® Space Invaders). An example of a shooter game in our data set is *Hit The Target*, a game where the player has to shoot at targets through a sight; the score will increase if the player shoots at eco-friendly objects only (e.g., bikes).

Simulation Games: They aim to simulate real-life situations for players to experience in virtual settings, for instance like driving an airplane (see [37], p.75-76). One of the most interesting examples of simulation games designed by students is *Government Simulator*, a game where players are asked to act like politicians, and balance their decision-making between producing wealth in the economy and sustainability.

Storytelling Games: They are primarily based on a plot or a narrative, through which the player progresses to learn about a particular topic or story; such games may not be interactive or include little interactivity, essentially tasking players to click through a story. *Extreme Climate* is an example of student-designed storytelling game, where players learn about CO_2 emission and pollution through an interactive story.

Swipe Elimination (or Swipe Action) Games: They task players to swipe on the screen to delete digital objects (e.g., Halfbrick Fruit Ninja). An example of such a game from our

Table 3: The identified game genres and descriptive statistics.

Genre	Count *n* (%)	CT Score *M* (*SD*)	Block Count *M* (*SD*)
Clicker	94 (23.3)	15.1 (1.78)	314 (238)
Maze	32 (7.9)	14.7 (1.81)	349 (260)
Platform	23 (5.7)	15.7 (2.16)	359 (321)
Pong	96 (23.8)	14.3 (2.25)	239 (224)
Puzzle	17 (4.2)	15.5 (3.08)	374 (308)
Quiz	50 (12.4)	13.4 (2.74)	315 (382)
Shooter	11 (2.7)	16.3 (2.28)	482 (454)
Simulation	22 (5.5)	15.8 (2.48)	408 (278)
Storytelling	25 (6.2)	13.8 (2.18)	242 (122)
Swipe Elimination	21 (5.2)	14.8 (2.19)	372 (233)

data set is *Extreme Weather*, a game where players swipe-away pollution from the atmosphere to keep the sky "clean".

Total CT Proficiency Across Game Genres

Table 3 shows the final CT score across the 10 game genres as assessed by Dr.Scratch. On a quick observation, we see that quiz produced the lowest CT score (M = 13.4, SD = 2.74), while shooter produced the highest (respectively M = 16.3, SD = 2.48). Puzzle and quiz show large variations in CT score (respectively SD = 3.08; SD = 2.74), while clicker and maze show low variations (respectively SD = 1.78; SD = 1.81). We further investigated the data set by conducting a one-way ANOVA to compare the effect of game genre on the final CT score. We found a significant difference between game genres, $F(9, 381) = 4.90$, $p < .001$. Post hoc tests using Tukey HSD revealed that the significance can be mainly attributed to the quiz games (compared to in particularly clicker, platform, and simulation game genres), with shooter games close to significance, and storytelling games close to significance too compared to platform, simulation, and shooter games. These results suggest that designing quiz games may not be beneficial to students from a CT learning perspective.

CT Dimensions Proficiency Across Game Genres

Table 4 shows the score for each CT dimension defined by Dr.Scratch across game genres. We see how abstraction scored low in most genres, with the exception of shooter games (M = 2.18, SD = 0.98); these results are consistent with previous work [118, 71]. Synchronization and parallelism instead scored higher than other CT dimensions, with storytelling scoring highest in parallelism (M = 3.00, SD = 0.00), and maze scoring highest in synchronization (M = 2.97, SD = 0.18); these results are also consistent with previous work [118, 71].

To compare the effect of game genre on the score of individual CT dimensions we conducted multivariate ANOVAs (i.e., MANOVA). We found that designing different game genres had a significant effect on the scoring of individual CT dimensions, $F(9, 381) = 2.60$, $p < .001$; Pillai's Trace = 0.40. Post hoc tests using Tukey HSD (based on one-way ANOVAs for each CT dimension) reveal here why quiz and storytelling differ from other genres. First, quiz and storytelling score lower on flow control (respectively M = 1.82, SD = 0.66; M = 1.72, SD = 0.61) compared to other game genres. Second, we see that quiz and storytelling also score the lowest on logic among game genres (respectively M = 1.36, SD = 1.10; M = 1.00, SD = 0.96). Among action-based games (i.e., clicker, maze, pong), we see that platform scored higher in logic, and highest overall (M = 2.52, SD = 0.95). We also see that maze scored lowest on data representation (M = 1.72, SD = 0.52). For user interactivity, we notice again differences among action-based games, where clicker (M = 2.03, SD = 0.18) scored higher compared to pong, which also scored lowest overall (M = 1.91, SD = 0.33). All genres, however, score (on average) ~2 in user interactivity, which is consistent with previous work [118, 71].

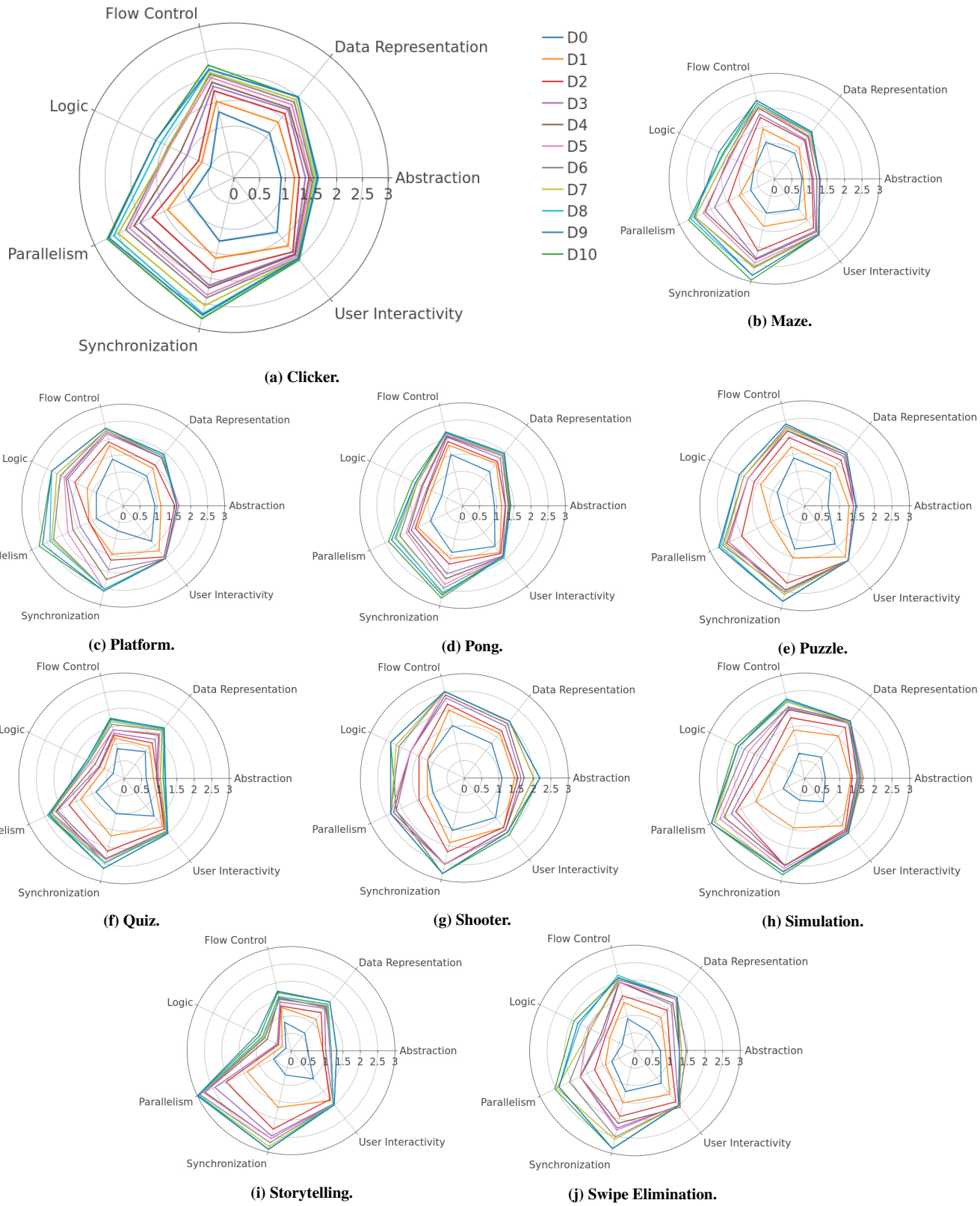
CT Development Across Game Genres

Figure 2 shows the results from the CT development analysis on two-dimensional radar charts, for each game genre, based on deciles (i.e., from D_0 to D_{10}). A conclusion we can quickly draw upon a first observation of the radar charts is that, remarkably, most CT development happened within D_0 to D_3. This suggests that this learning phase may be a time-critical period for CT development. After this period, CT skills incrementally improve over time, but by and large the CT "profile" has been established. The time between D_0 and D_{10} is approximately two weeks, which is about the time students spent on programming and designing their games within the STEM curriculum. The time between D_0 and D_3 is ~3.8 days, which means that the time-critical period for CT development took place within the first week. As students were mostly developing their games in the classroom, and most curricula offered four classes per week of ~50-60 minutes each, we approximate the time-critical period of actual development to be ~4 hours.

However, there are discernible differences on how scores progress in each CT dimension based on game genre. For instance, pong has a CT score of ~1 in all CT dimensions already at D_1, while puzzle shows a score of ~0.5 on the same decile. This trend extends in general to all action-based versus other genres, with the exception of puzzle, which has a CT score of ~1 in all CT dimensions aside from abstraction and parallelism. Additionally, we can see how, although simulation and storytelling score around ~0.5 in D_1 in almost all dimensions, they increase rapidly in D_2 towards a CT score of ~1.5, except for logic in quiz which does not reach ~1 until D_{10}. These results suggest that CT development progresses differently in the first learning phases for different game genres. However, regardless of game genre, user interactivity and data representation are never going beyond a CT score of ~2 on D_{10}. Again, this is consistent with results from previous work [118, 71], suggesting that the score progression of these two particular CT dimensions is not affected by what games genre students design.

Scratch Block-Usage Across Game Genres

The last step in our analysis involves the consideration of block-usage in Scratch across genres. Table 3 shows the descriptive statistics, indicating that shooter used by far most blocks (n = 482), followed by simulation (n = 408), and puzzle (n = 374). The least amount of blocks is used by pong (n = 239). While a one-way ANOVA did not reveal a significant difference in block-usage among game genres, we did find a



(a) Clicker. (b) Maze. (c) Platform. (d) Pong. (e) Puzzle. (f) Quiz. (g) Shooter. (h) Simulation. (i) Storytelling. (j) Swipe Elimination.

Figure 2: Dr.Scratch scores per dimension, decile, and per genre. The innermost decile is D_0 and the outermost is D_{10}.

Table 4: Descriptive statistics for each of the CT dimensions across game genres, in *M* (*SD*).

Genre	Abstraction	Parallelism	Logic	Synchronization	Flow Control	User Interactivity	Data Representation
Clicker	1.66 (0.90)	2.72 (0.65)	1.68 (1.00)	2.81 (0.45)	2.23 (0.50)	2.03 (0.18)	2.00 (0.21)
Maze	1.25 (0.62)	2.75 (0.62)	1.53 (1.06)	2.97 (0.18)	2.22 (0.49)	2.03 (0.18)	1.72 (0.52)
Platform	1.70 (0.93)	2.57 (0.84)	2.52 (0.95)	2.65 (0.78)	2.30 (0.47)	2.00 (0.00)	1.96 (0.21)
Pong	1.48 (0.85)	2.39 (0.90)	1.63 (0.87)	2.73 (0.53)	2.21 (0.45)	1.91 (0.33)	2.00 (0.41)
Puzzle	1.53 (0.80)	2.76 (0.66)	2.00 (1.12)	2.82 (0.73)	2.35 (0.61)	2.00 (0.00)	2.00 (0.61)
Quiz	1.16 (0.55)	2.54 (0.95)	1.36 (1.10)	2.68 (0.79)	1.82 (0.66)	2.00 (0.20)	1.88 (0.56)
Shooter	2.18 (0.98)	2.18 (0.98)	2.36 (1.03)	2.82 (0.40)	2.54 (0.52)	2.09 (0.30)	2.09 (0.30)
Simulation	1.64 (0.90)	2.91 (0.29)	2.00 (1.07)	2.82 (0.66)	2.32 (0.57)	2.00 (0.30)	2.09 (0.43)
Storytelling	1.32 (0.69)	3.00 (0.00)	1.00 (0.96)	2.92 (0.28)	1.72 (0.61)	2.00 (0.00)	1.80 (0.50)
Swipe elimination	1.38 (0.80)	2.42 (0.87)	1.86 (1.15)	2.90 (0.30)	2.24 (0.44)	2.04 (0.00)	1.90 (0.30)

significant, albeit small correlation between the final CT score and the number of blocks used in Scratch, $r_p = .22$, $p < .001$.

Figure 3 shows a comparison of Scratch of the percentages of block-usage across game genres. We see, for instance, how quiz and storytelling did not use control blocks frequently (i.e., less than 15%), which explains their low CT scores on flow control. For storytelling, the use of logic blocks is almost 0%, evidencing how choosing to design this type of game prevents students from learning the use of Scratch blocks that contain logical statements (i.e., if, if-else, AND, OR), and thus developing in logic. It is noteworthy that quiz and storytelling used more than 25% of their total blocks in looks blocks, showing that such genres lead students to develop proficiently in synchronization, but limit the learning of abstraction and logic. Furthermore, we see that simulation and shooter use more data blocks than other genres, and thus lead to higher CT proficiency in data representation. Regarding the use of other blocks, they seem to be less reflective of the CT dimensions, and do not seem to differ so much among genres.

As with the Dr.Scratch scores, we also looked at how the block usage (both count and type) develop over time (using deciles). Here, we found three patterns that deserve further inquiry (see Figure 4). When we observed the block usage development for all games, most genres seem to develop linearly over time (i.e., the growth in count steadily increases along the same

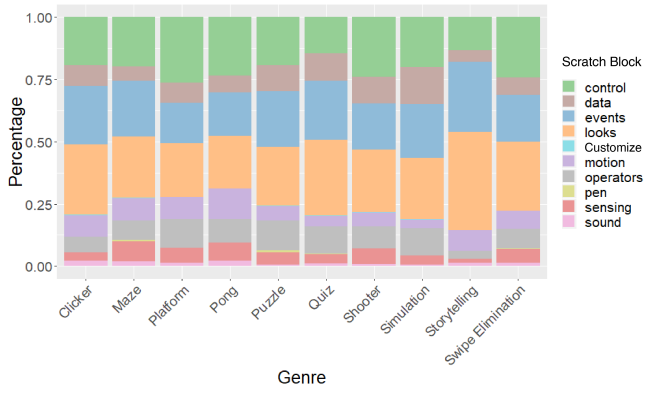


Figure 3: Percentages of block usage across game genres.

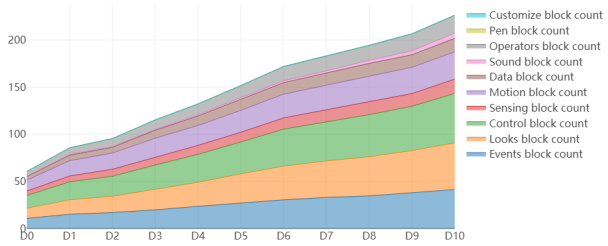
curve and with the same proportions of types of blocks used). Therefore, what students build in the beginning of their project is a "microcosm" of what they end up making; they just iterate and increase the use of same blocks until the end. Pong games, for instance, are a clear example of such linear patterns (see Figure 4a). Then, there are genres like platform, which show a more exponential growth in block usage, especially between D_6 and D_{10} (see Figure 4b). Finally, in storytelling games we see that, compared to pong and platform, the type of block-usage is already dominated by looks and event blocks between D_0 and D_1, and continues developing in that "focused growth" fashion until the end (i.e., D_{10}), with just little increase observable in the use of control blocks (see Figure 4c). In sum, the three patterns exemplified above show that game genres may impact the development of block-usage in Scratch.

Genres, CT Proficiency and Block-Usage

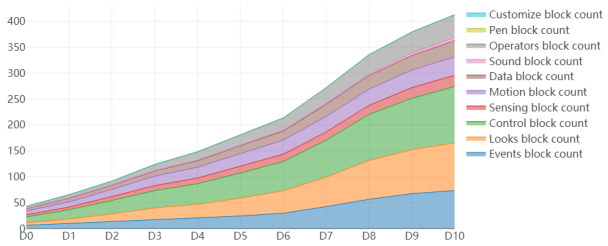
As shown before, the CT final score given by Dr.Scratch and the number of blocks used among game genres is small (r_p = .2). Figure 5 offers a different point of view relating block-usage and CT development scores. For each genre, the average final Dr.Scratch score is given in the horizontal and the average number of blocks in the vertical axis, respectively (see also Table 1). This allows to identify characteristics of the different genres. Specifically, we can observe from Figure 5 that quiz uses as many blocks as clicker; however, its mean CT score is almost two points lower. Then, storytelling and pong make use of fewer blocks while the CT development score is below average. We see some genres (puzzle, platform and simulation) that outperform in CT score with similar number of blocks (maze, swipe and clicker). Finally, shooter shows to be the one for which more blocks are used and a higher CT development is achieved. In Figure 5 we grouped genres based on these combined results (through the rectangles).

DISCUSSION

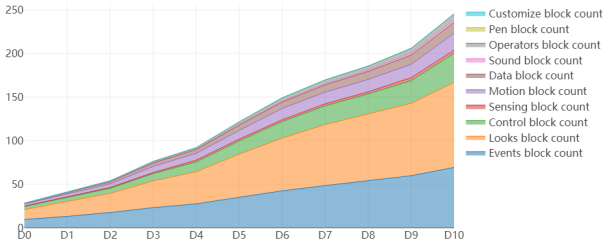
We examined 404 student-designed games for constructionist STEM curricula focused on climate science, systems thinking, and CT. We analyzed 391 student-designed games for CT proficiency and 325 for CT development, and explored how programming and CT were influenced by game genres. Results showed that designing different game genres impact CT proficiency and development, as well as block-usage in



(a) Pong, example of linear growth



(b) Platform, example of exponential growth.



(c) Storytelling, example of focused growth.

Figure 4: Evolution of the block counts for three genres.

Scratch. Next, we discuss how our results have implications for CT and constructionist learning, and conclude by outlining limitations.

Impact of Game Genre on Programming and CT

We explored the impact of game genre on programming and CT practices, and yielded results that integrate earlier work, which asked for further exploring CT proficiency and development in contextualized design practices [118]. Considering the CT proficiency (i.e., the final CT score as assessed by Dr.Scratch), we can draw the following conclusions. First, quiz games (but potentially also storytelling) result in less CT proficiency overall, at least compared to platform, simulation, and shooter games. Second, puzzle seems to vary greatly in CT ($SD = 3.08$), which explains why, despite its relatively high score ($M = 15.5$), it does not contrast sharply with any of the other game genres. Third, regarding the individual CT dimensions, there are noticeable differences among game genres. For instance, platform and simulation are most proficient in logic, while maze games show basic (i.e., 1) in data representation.

Results on CT development (i.e., Dr.Scratch score across the deciles) show game genre impacts CT score progression, especially in the early phases. For instance, action games lead to higher CT development within D_0 and D_1. These results also confirm that quiz and storytelling prevent students from developing in logic, and that along with simulation, those genres show the least progress observable between D_0 and D_1. Hence, our work shows that game genres may play a key role in shaping programming practices and CT development. Hence, our results should be considered by researchers and educators who work with constructionist curricula that incorporate the learning of programming and CT via game design [46].

CT Develops First and Consolidates

Our results suggest that CT develops fast. There is a time critical period in which CT is developed. After that, CT skills only increase marginally (i.e., the gain in CT development in the last deciles decreases steadily). The exception to this "rule" can be found in some of those genres that achieve higher levels of CT development scores, such as shooter, platform or simulation, where some dimensions show a steady improvement (e.g., parallelism in simulation or platform games). Besides the aforementioned exception, we observe this *CT-develops-first-and-consolidates* "behavior" for all other game genres, independently of initial CT development (i.e., their CT "profile"). In other words, no matter how CT develops within the first three deciles, we discover the same patterns for all genres. These results show that students acquire the "basic" CT skills needed to design their games in the early phases of learning. Once these skills have been consolidated (i.e., by D_3), students can focus on those aspects that are more related to "actual" game design practices, and be more refined in their programming practices (e.g., deploying *finesse* [43, 104]).

Influence of Game Familiarity and Preferences

We observed that students designed their games prominently as action-based. Specifically, clicker and pong games dominated, comprising 47% of all games. Although, students were not asked to consider particular game genres by the curriculum, this choice may have stemmed from popularity and familiarity with existing games. As such, we notice that clicker games are currently very popular, and may have influenced students in their design; the same could apply to the great popularity of pong games. Furthermore, both games are based on simple game mechanics, which might have been a further reason for students to choose those genres. Kafai [47] already showed how popular games impact students' game design. Our results

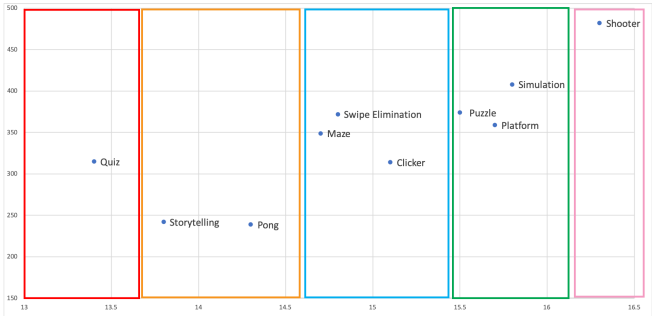


Figure 5: Mapping of game genres on the average final Dr.Scratch score (x-axis) and the average number of blocks used (y-axis). Genres are grouped in rectangles.

confirm such trend, and we encourage future research to further investigate those trends and what the implications might be to CT learning.

As indicated earlier, storytelling games limit the development of logic. Such results are especially relevant to differences in gender for game design. Previous work showed how girls enjoy storytelling games and generally the story aspects of games [38]. As such, some efforts sought to broaden CS education and attract female students through the use of storytelling (games) [28, 129, 122]. However, while increasing motivation to learn CS is important, our findings suggest that focusing on such genre has a clear trade-off between game design and the likelihood of becoming proficient in specific CT dimensions (e.g., logic, abstraction). Future research should carefully consider the impact of demographics (see also [33]) on game design, and explore alternatives for resolving such trade-offs.

Implications for Constructionist Learning

Our results challenge the notion that a free-form constructionist curriculum leads to a wide variety of artifacts. Although students were given the freedom to design any game (provided it tackled climate science), they chose to design games mostly after modern mobile-gaming (i.e., clicker), or after classical video games (e.g., pong), and action-based games predominated (e.g., maze, platform). As such, we observed how programming routines and CT learning were in turn influenced by students' game design, and possibly the framing of educational content too. This raises questions to whether constructionist approaches should consider better calibrating learning outcomes against students' personal interests. In that respect, we outline a series of pedagogical recommendations:

- As proposed by Han and Bhattacharya [36], we suggest that students interests and skills be *balanced through guided tasks*, to mitigate the effect of "openness" in constructionist learning observed in our results.
- Educators and designers of constructionist curricula may be inspired by the work of Tangworakitthaworn et al. [114], by creating a mix and match of *constructionist/ instructionist* approaches, to provide learners with guided tasks that maximize the learning of both game design and CT.

We have also noticed implications to the learning of CT transversal to the learning of other skills in constructionist curricula (e.g., [12]). The lack of scientific evidence on the development of CT through school-age programming, and its transfer possibilities for the acquisition of other competencies, was discussed by previous reviews [34, 44]. Here, we offer insight into the possibilities of using CT to leverage science concepts by means of a constructionist approach. We note that a fear that is commonly reported by educators and researchers [51, 56] is that game design and CT would "jeopardize" students' learning of content. However, our results show that the CT skills needed are mainly developed in the first phases of learning (i.e., from D_0 to D_3), and thus students can devote their efforts to game design and content learning in the remaining time of their learning activities.

Limitations

Compared to [118], we included contextual game design information (i.e., game genre) to CT proficiency and development analyses. There is, however, other contextual information that may be of influence, which we did not consider in this work. For instance, the different teaching styles, students' prior knowledge of programming in Scratch, and teamwork dynamics of student pairs, may have influenced the results. Future work should determine the extent to which these factors also impact programming routines and CT development. In terms of validity, our results might be limited to our data collection, the use of Dr.Scratch, and curriculum implementation. For data collection, some students may have uploaded their Scratch projects when almost completed, and thus showing high CT scores already within the first deciles. We should compensate for that shortcoming by better monitoring students' upload of their Scratch projects. Our results are based on Dr.Scratch assessment of CT. Whilst a validated tool for measuring CT [70, 72] and successfully applied in several contexts [97], there is room for improvement in how it measures CT in its current form. The results showing how CT develops fast may be a reflection that key CT competencies do indeed develop in the beginning of CT learning, but may also be limited to Dr.Scratch not being able to capture iterative and more complex CT practices. In that respect, we included an analysis of block-usage to provide more insights. However, future work should be geared towards exploring how (and if) the Dr.Scratch metrics need to be expanded or re-calibrated [118] for better capturing CT development beyond basic skills. Finally, the curriculum structure and the topic of climate change may have influenced the game design of students. We did not consider such implications, as we focused primarily on the influence of game genre on programming and CT learning. Future work should assess the CT development of student-designed games produced in diverse contexts. This would reveal if the trends in our results are consistent (or differ) across different disciplines and learning goals, and hopefully produce a broader overview of CT development via game design.

CONCLUSION

We explored the impact of game genre on CT development and programming in Scratch video games designed by 8th-grade students. We showed how different game genres (e.g., clicker, puzzle) may lead to developing CT differently and has impact on programming routines (here block-usage in Scratch). Furthermore, we observed how CT develops fast in the first phases of game design, and that more refined design practices seem to engage later on to consolidate CT profiles. Finally, we discussed how our results have implications to the design of future constructionist curricula that foster the development of CT via game design. We wish future work to extend our results and further explore the role of game design in constructionist learning for CT and programming.

ACKNOWLEDGEMENTS

This project was supported by the NSF under Grant No. 1542954 (Puttick PI). We also want to thank our collaborating teachers, school districts, and students.

REFERENCES

[1] Mete Akcaoglu. 2014. Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development* 62, 5 (2014), 583–600.

[2] Jennifer Albert, Barry Peddycord III, and Tiffany Barnes. 2015. Evaluating Scratch Programs to Assess Computational Thinking in a Science Lesson (Abstract Only). In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 679–679. DOI: http://dx.doi.org/10.1145/2676723.2691928

[3] Sultan A Alharthi, Olaa Alsaedi, Zachary O Toups, Joshua Tanenbaum, and Jessica Hammer. 2018. Playing to wait: A taxonomy of idle games. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 621.

[4] Alan Amory, Kevin Naicker, Jacky Vincent, and Claudia Adams. 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 (Oct. 1999), 311–321. DOI:http://dx.doi.org/10.1111/1467-8535.00121

[5] Renate Andersen and Marisa Ponti. 2014. Participatory pedagogy in an open educational course: challenges and opportunities. *Distance education* 35, 2 (2014), 234–249.

[6] Leonard A. Annetta, Marshall R. Murray, Shelby Gull Laird, Stephanie C. Bohr, and John C. Park. 2006. Serious Games: Incorporating Video Games in the Classroom. (2006). https://er.educause.edu/articles/2006/1/serious-games-incorporating-video-games-in-the-classroom

[7] Jackie Barnes, Amy K. Hoover, Borna Fatehi, Jesus Moreno-Leon, Gillian Smith, and Casper Harteveld. 2017. Exploring Emerging Design Patterns in Student-made Climate Change Games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG '17)*. ACM, New York, NY, USA, Article 64, 6 pages. DOI: http://dx.doi.org/10.1145/3102071.3116224

[8] Valerie Barr and Chris Stephenson. 2011. Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads* 2, 1 (Feb. 2011), 48–54. DOI:http://dx.doi.org/10.1145/1929887.1929905

[9] Ashok R. Basawapatna, Kyu Han Koh, and Alexander Repenning. 2010. Using Scalable Game Design to Teach Computer Science from Middle School to Graduate School. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '10)*. ACM, New York, NY, USA, 224–228. DOI: http://dx.doi.org/10.1145/1822090.1822154

[10] Ahmet Baytak and Susan M. Land. 2010. A case study of educational game design by kids and for kids. *Procedia - Social and Behavioral Sciences* 2, 2 (Jan. 2010), 5242–5246. DOI: http://dx.doi.org/10.1016/j.sbspro.2010.03.853

[11] Erik Blair. 2015. A reflexive exploration of two qualitative data coding techniques. *Journal of Methods and Measurement in the Social Sciences* 6, 1 (2015), 14–29.

[12] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, and others. 2016. *Developing computational thinking in compulsory education-Implications for policy and practice*. Technical Report. Joint Research Centre (Seville site).

[13] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. 2013. Hairball: Lint-inspired Static Analysis of Scratch Projects. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 215–220. DOI: http://dx.doi.org/10.1145/2445196.2445265

[14] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *In AERA 2012*.

[15] Kajal Claypool and Mark Claypool. 2005. Teaching Software Engineering Through Game Design. *SIGCSE Bull.* 37, 3 (June 2005), 123–127. DOI: http://dx.doi.org/10.1145/1151954.1067482

[16] Valentina Dagiene and Gabriele Stupuriene. 2016. Bebras–A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education* 15, 1 (2016). https://eric.ed.gov/?id=EJ1097494

[17] Valentina Dagienė, Gabrielė Stupurienė, and Lina Vinikienė. 2016. Promoting Inclusive Informatics Education Through the Bebras Challenge to All K-12 Students. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016 (CompSysTech '16)*. ACM, New York, NY, USA, 407–414. DOI: http://dx.doi.org/10.1145/2983468.2983517

[18] Alexandra A De Souza, Thiago S Barcelos, Roberto Munoz, Rodolfo Villarroel, and Leandro A Silva. 2019. Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops. *IEEE Access* 7 (2019), 82848–82866.

[19] Jill Denner, Shannon Campe, and Linda Werner. 2019. Does Computer Game Design and Programming Benefit Children? A Meta-Synthesis of Research. *ACM Trans. Comput. Educ.* 19, 3, Article 19 (Jan. 2019), 35 pages. DOI:http://dx.doi.org/10.1145/3277565

[20] Jill Denner, Linda Werner, and Eloy Ortiz. 2012a. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education* 58, 1 (Jan. 2012), 240–249. DOI: http://dx.doi.org/10.1016/j.compedu.2011.08.006

[21] Jill Denner, Linda Werner, and Eloy Ortiz. 2012b. Computer Games Created by Middle School Girls: Can They Be Used to Measure Understanding of Computer Science Concepts? *Comput. Educ.* 58, 1 (Jan. 2012), 240–249. DOI: http://dx.doi.org/10.1016/j.compedu.2011.08.006

[22] Ap Dijksterhuis and Ad Van Knippenberg. 1998. The relation between perception and behavior, or how to win a game of trivial pursuit. *Journal of personality and social psychology* 74, 4 (1998), 865.

[23] Jeffrey Earp. 2015. Game making for learning: A systematic review of the research literature. In *Proceedings of 8th international conference of education, research and innovation (ICERI2015)*. 6426–6435.

[24] Yoram Eshet. 2004. Digital Literacy: A Conceptual Framework for Survival Skills in the Digital era. *Journal of Educational Multimedia and Hypermedia* 13, 1 (January 2004), 93–106. https://www.learntechlib.org/p/4793

[25] Sara I. de Freitas. 2006. Using games and simulations for supporting learning. *Learning, Media and Technology* 31, 4 (Dec. 2006), 343–358. DOI: http://dx.doi.org/10.1080/17439880601021967

[26] T. Fullerton. 2006. Play-centric games education. *Computer* 39, 6 (June 2006), 36–42. DOI: http://dx.doi.org/10.1109/MC.2006.205

[27] Alex Games and Luke Kane. 2011. Exploring Adolescent's STEM Learning Through Scaffolded Game Design. In *Proceedings of the 6th International Conference on Foundations of Digital Games (FDG '11)*. ACM, New York, NY, USA, 1–8. DOI: http://dx.doi.org/10.1145/2159365.2159366

[28] Elisabeth Gee, Earl Aguilera, Kelly Tran, Dani Kachorsky, Priyanka Parekh, and others. 2017. The Design and Outcomes of Story-Enhanced Games to Teach Computer Science Concepts. American Educational Research Association Conference, San Antonio, TX.

[29] Paul Gilster. 1997. *Digital Literacy*. John Wiley & Sons, Inc., New York, NY, USA.

[30] Katerina Glezou. 2014. Student Engagement in digital storytelling with Scratch in classroom settings. *Pridobljeno* 11, 10 (2014), 2015.

[31] David Goldschmidt, Ian MacDonald, Judith O'Rourke, and Brandon Milonovich. 2011. An interdisciplinary approach to injecting computer science into the K-12 classroom. *Journal of Computing Sciences in Colleges* 26, 6 (2011), 78–85.

[32] Begoña Gros. 2007. Digital Games in Education. *Journal of Research on Technology in Education* 40, 1 (Sept. 2007), 23–38. DOI: http://dx.doi.org/10.1080/15391523.2007.10782494

[33] Shuchi Grover, Satabdi Basu, and Patricia Schank. 2018. What we can learn about student learning from open-ended programming projects in middle school computer science. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 999–1004.

[34] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[35] Raija Hamäläinen. 2011. Using a game environment to foster collaborative learning: a design-based study. *Technology, Pedagogy and Education* 20, 1 (March 2011), 61–78. DOI: http://dx.doi.org/10.1080/1475939X.2011.554010

[36] Seungyeon Han and Kakali Bhattacharya. 2001. Constructionism, learning by design, and project based learning. *Emerging perspectives on learning, teaching, and technology. Retrieved April* 29 (2001), 2007.

[37] Casper Harteveld. 2011. *Triadic Game Design: Balancing Reality, Meaning and Play* (1st ed.). Springer Publishing Company, Incorporated.

[38] Casper Harteveld, Gillian Smith, Gail Carmichael, Elisabeth Gee, and Carolee Stewart-Gardiner. 2014. A design-focused analysis of games teaching computer science. *Proceedings of Games+ Learning+ Society* 10 (2014).

[39] Douglas R Hartree. 2012. *Calculating instruments and machines*. Cambridge University Press.

[40] Stephanie Heintz and Effie Lai-Chong Law. 2015. The Game Genre Map: A Revised Game Classification. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '15)*. ACM, New York, NY, USA, 175–184. DOI: http://dx.doi.org/10.1145/2793107.2793123

[41] Felienne Hermans and Efthimia Aivaloglou. 2017. Teaching software engineering principles to k-12 students: a mooc on scratch. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE, 13–22.

[42] Amy K. Hoover, Jackie Barnes, Borna Fatehi, Jesús Moreno-León, Gillian Puttick, Eli Tucker-Raymond, and Casper Harteveld. 2016. Assessing Computational Thinking in Students' Game Designs. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (CHI PLAY Companion '16)*. ACM, New York, NY, USA, 173–179. DOI: http://dx.doi.org/10.1145/2968120.2987750

[43] Sid L Huff, Malcolm C Munro, and Barbara Marcolin. 1992. Modelling and measuring end user sophistication. In *Proceedings of the 1992 ACM SIGCPR conference on Computer personnel research*. ACM, 1–10.

[44] Computing in Schools Special Interest Network. 2015. *Computing and digital literacy: Call for a holistic approach*. Technical Report. Council of European Professional Informatics Societies.

[45] Henry Jenkins. 2009. *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press.

[46] Yasmin B. Kafai. 1996. *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Routledge.

[47] Yasmin B Kafai. 1998. Video game designs by girls and boys: Variability and consistency of gender differences. *From Barbie to Mortal Kombat: gender and computer games* (1998), 90–114.

[48] Yasmin B. Kafai. 2005. The Classroom as "Living Laboratory": Design-Based Research for Understanding, Comparing, and Evaluating Learning Science Through Design. *Educational Technology* 45, 1 (2005), 28–34. http://www.jstor.org/stable/44429186

[49] Yasmin B. Kafai. 2006. Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. *Games and Culture* 1, 1 (Jan. 2006), 36–40. DOI: http://dx.doi.org/10.1177/1555412005281767

[50] Yasmin B Kafai and Quinn Burke. 2013. Computer programming goes back to school. *Phi Delta Kappan* 95, 1 (2013), 61–65.

[51] Yasmin B. Kafai and Quinn Burke. 2015. Constructionist Gaming: Understanding the Benefits of Making Games for Learning. *Educational Psychologist* 50, 4 (Oct. 2015), 313–334. DOI: http://dx.doi.org/10.1080/00461520.2015.1124022

[52] Filiz Kalelioglu, Yasemin Gilbahar, and Volkan Kukul. 2016. A Framework for Computational Thinking Based on a Systematic Research Review. In *Baltic Journal of Modern Computing*. 583–596.

[53] Marjaana Kangas. 2010. Creative and playful learning: Learning through game co-creation and games in a playful learning environment. *Thinking skills and Creativity* 5, 1 (2010), 1–15.

[54] Engin Karahan and Gillian Roehrig. 2015. Constructing Media Artifacts in a Social Constructivist Environment to Enhance Students' Environmental Awareness and Activism. *Journal of Science Education and Technology* 24, 1 (Feb. 2015), 103–118. DOI: http://dx.doi.org/10.1007/s10956-014-9525-5

[55] Fengfeng Ke. 2008. A case study of computer gaming for math: Engaged learning from gameplay? *Computers & Education* 51, 4 (Dec. 2008), 1609–1620. DOI: http://dx.doi.org/10.1016/j.compedu.2008.03.003

[56] Fengfeng Ke. 2014. An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education* 73 (April 2014), 26–39. DOI: http://dx.doi.org/10.1016/j.compedu.2013.12.010

[57] Aaron D. Knochel and Ryan M. Patton. 2015. If Art Education Then Critical Digital Making: Computational Thinking and Creative Code. *Studies in Art Education* 57, 1 (2015), 21–38. DOI: http://dx.doi.org/10.1080/00393541.2015.11666280

[58] Praveen Kuruvada, Daniel Asamoah, Nikunj Dalal, and Subhash Kak. 2010. The Use of Rapid Digital Game Creation to Learn Computational Thinking. *arXiv:1011.4093 [cs]* (Nov. 2010). http://arxiv.org/abs/1011.4093 arXiv: 1011.4093.

[59] James C. Lester, Hiller A. Spires, John L. Nietfeld, James Minogue, Bradford W. Mott, and Eleni V. Lobene. 2014. Designing game-based learning environments for elementary science education: A narrative-centered learning perspective. *Information Sciences* 264 (April 2014), 4–18. DOI: http://dx.doi.org/10.1016/j.ins.2013.09.005

[60] Angeline S Lillard. 2013. Playful learning and Montessori education. *NAMTA Journal* 38, 2 (2013), 137–174.

[61] Sze Yee Lye and Joyce Hwee Ling Koh. 2014. Review on Teaching and Learning of Computational Thinking Through Programming. *Comput. Hum. Behav.* 41, C (Dec. 2014), 51–61. DOI: http://dx.doi.org/10.1016/j.chb.2014.09.012

[62] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 16.

[63] Sven Manske, Sören Werneburg, and H Ulrich Hoppe. 2019. Learner Modeling and Learning Analytics in Computational Thinking Games for Education. In *Data Analytics Approaches in Educational Games and Gamification Systems*. Springer, 187–212.

[64] Merrilea J. Mayo. 2007. Games for Science and Engineering Education. *Commun. ACM* 50, 7 (July 2007), 30–35. DOI: http://dx.doi.org/10.1145/1272516.1272536

[65] Daniel D McCracken. 1957. *Digital computer programming*. John Wiley & Sons, Inc.

[66] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2010. Learning Computer Science Concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 69–76. DOI: http://dx.doi.org/10.1145/1839594.1839607

[67] David R. Michael and Sandra L. Chen. 2005. *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade.

[68] Punya Mishra and Matthew J Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record* 108, 6 (2006), 1017.

[69] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[70] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE global engineering education conference (EDUCON)*. IEEE, 1040–1045.

[71] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2017a. Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing* (2017).

[72] Jesús Moreno-León, Marcos Román-González, Casper Harteveld, and Gregorio Robles. 2017b. On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2788–2795.

[73] Jesús Moreno-León and Gregorio Robles. 2015. Dr. Scratch: A Web Tool to Automatically Evaluate Scratch Projects. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15)*. ACM, New York, NY, USA, 132–133. DOI:http://dx.doi.org/10.1145/2818314.2818338

[74] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016a. Code to Learn: Where Does It Belong in the K-12 Curriculum? *Journal of Information Technology Education: Research* 15 (2016), 283–303.

[75] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016b. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE Global Engineering Education Conference (EDUCON)*. 1040–1045. DOI: http://dx.doi.org/10.1109/EDUCON.2016.7474681

[76] Jesús Moreno-León, Marcos Román-González, Casper Harteveld, and Gregorio Robles. 2017. On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 2788–2795. DOI:http://dx.doi.org/10.1145/3027063.3053216

[77] Chrystalla Mouza, Yi-Cheng Pan, Lori L. Pollock, James Atlas, and Terry Harvey. 2014. Partner4CS: Bringing Computational Thinking to Middle School through Game Design. FabLearn. (2014).

[78] M. Overmars. 2004. Teaching computer science through game design. *Computer* 37, 4 (April 2004), 81–83. DOI: http://dx.doi.org/10.1109/MC.2004.1297314

[79] Kristine Oygardslia. 2015. Students as Game Designers: Learning by Creating Game Narratives in the Classroom. In *Interactive Storytelling (Lecture Notes in Computer Science)*. Springer, Cham, 341–344. DOI:http://dx.doi.org/10.1007/978-3-319-27036-4_32

[80] Marina Papastergiou. 2009. Exploring the potential of computer and video games for health and physical education: A literature review. *Computers & Education* 53, 3 (Nov. 2009), 603–622. DOI: http://dx.doi.org/10.1016/j.compedu.2009.04.001

[81] Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

[82] Seymour Papert. 1993. *The Children's Machine: Rethinking School in the Age of the Computer*. BasicBooks, 10 East 53rd St.

[83] Youngki Park and Youhyun Shin. 2019. Comparing the Effectiveness of Scratch and App Inventor with Regard to Learning Computational Thinking Concepts. *Electronics* 8, 11 (2019), 1269.

[84] Kylie A Peppler and Yasmin B Kafai. 2007. From SuperGoo to Scratch: Exploring creative digital media production in informal learning. *Learning, media and technology* 32, 2 (2007), 149–166.

[85] Jean Piaget. 1972. *Psychology and Epistemology: Towards a Theory of Knowledge*. Allen Lane. Google-Books-ID: 7DkdAQAAMAAJ.

[86] Marc Prensky. 2008a. Programming is the new literacy. *Edutopia magazine* (2008).

[87] Marc Prensky. 2008b. Students as designers and creators of educational computer games: Who else? *British Journal of Educational Technology* 39, 6 (Nov. 2008), 1004–1019. DOI:http://dx.doi.org/10.1111/j.1467-8535.2008.00823_2.x

[88] Gillian Puttick, Jackie Barnes, Giovanni Maria Troiano, Casper Harteveld, Eli Tucker-Raymond, Mike Cassidy, and Gillian Smith. 2018. Exploring how student designers model climate system complexity in computer games. In *Proceedings of the Summit on Connected Learning, CLS'18*. ETC Press, 196–204.

[89] Gillian Puttick and Eli Tucker-Raymond. 2018. Building Systems from Scratch: an Exploratory Study of Students Learning About Climate Change. *Journal of Science Education and Technology* 27, 4 (Aug. 2018), 306–321. DOI: http://dx.doi.org/10.1007/s10956-017-9725-x

[90] Mitchel Resnick. 1990. *LEGO/Logo–learning Through and about Design*. Epistemology and Learning Group, MIT Media Laboratory. Google-Books-ID: AEmytgAACAAJ.

[91] Mitchel Resnick. 1996. Distributed Constructionism. In *Proceedings of the 1996 International Conference on Learning Sciences (ICLS '96)*. International Society of the Learning Sciences, Evanston, Illinois, 280–284. http://dl.acm.org/citation.cfm?id=1161135.1161173

[92] Mitchel Resnick. 2003. Playful learning and creative societies. *Education Update* 8, 6 (2003), 1–2.

[93] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and others. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[94] Lloyd P. Rieber. 1996. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research and Development* 44, 2 (June 1996), 43–58. DOI: http://dx.doi.org/10.1007/BF02300540

[95] Judy Robertson and Cathrin Howells. 2008. Computer game design: Opportunities for successful learning. *Computers & Education* 50, 2 (Feb. 2008), 559–578. DOI: http://dx.doi.org/10.1016/j.compedu.2007.09.020

[96] Gregorio Robles, Jesús Moreno-León, Efthimia Aivaloglou, and Felienne Hermans. 2017. Software clones in Scratch projects: On the presence of copy-and-paste in Computational Thinking learning. In *Software Clones (IWSC), 2017 IEEE 11th International Workshop on*. IEEE, 1–7.

[97] Marcos Román-González, Jesús Moreno-León, and Gregorio Robles. 2019. Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In *Computational Thinking Education*. Springer, 79–98.

[98] Marcos Romn-Gonzlez, Juan-Carlos Prez-Gonzlez, and Carmen Jimnez-Fernndez. 2017. Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Comput. Hum. Behav.* 72, C (July 2017), 678–691. DOI: http://dx.doi.org/10.1016/j.chb.2016.08.047

[99] Marcos Román-González. 2015. Computational Thinking Test: Design, Guidelines, and Content Validation. DOI: http://dx.doi.org/10.13140/RG.2.1.4203.4329

[100] Marcos Román-González, Juan-Carlos Pérez-González, Jesús Moreno-León, and Gregorio Robles. 2018. Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction* (07 2018). DOI: http://dx.doi.org/10.1016/j.ijcci.2018.06.004

[101] Ricarose Vallarta Roque. 2007. *OpenBlocks: an extendable framework for graphical block programming systems*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[102] Ricardo Rosas, Miguel Nussbaum, Patricio Cumsille, Vladimir Marianov, Mónica Correa, Patricia Flores, Valeska Grau, Francisca Lagos, Ximena López, Verónica López, Patricio Rodriguez, and Marcela Salinas. 2003. Beyond Nintendo: design and assessment of educational video games for first and second grade students. *Computers & Education* 40, 1 (Jan. 2003), 71–94. DOI: http://dx.doi.org/10.1016/S0360-1315(02)00099-4

[103] Dana Ruggiero and Laura Green. 2017. Problem solving through digital game design: A quantitative content analysis. *Computers in Human Behavior* 73 (2017), 28–37.

[104] Christopher Scaffidi and Christopher Chambers. 2012. Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction* 28, 6 (2012), 383–398.

[105] Anthony Scopatz and Kathryn D Huff. 2015. *Effective computation in physics: Field guide to research with python*. " O'Reilly Media, Inc.".

[106] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 59–66. DOI: http://dx.doi.org/10.1145/2493394.2493403

[107] Cynthia Selby and John Woollard. 2013. Computational thinking: the developing definition. (2013).

[108] Amber Settle, Debra S. Goldberg, and Valerie Barr. 2013. Beyond Computer Science: Computational Thinking Across Disciplines. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '13)*. ACM, New York, NY, USA, 311–312. DOI: http://dx.doi.org/10.1145/2462476.2462511

[109] David Williamson Shaffer. 2006. *How Computer Games Help Children Learn*. Macmillan. Google-Books-ID: ZQqOmEdlSL8C.

[110] David Williamson Shaffer, Kurt R. Squire, Richard Halverson, and James P. Gee. 2005. Video Games and the Future of Learning. *Phi Delta Kappan* 87, 2 (Oct. 2005), 105–111. DOI: http://dx.doi.org/10.1177/003172170508700205

[111] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review* 22 (2017), 142–158.

[112] Kurt Squire. 2011. *Video Games and Learning: Teaching and Participatory Culture in the Digital Age. Technology, Education–Connections (the TEC Series)*. Teachers College Press.

[113] Tarja Susi, Mikael Johannesson, and Per Backlund. 2007. *Serious Games : An Overview*. Institutionen för kommunikation och information. http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-1279

[114] Preecha Tangworakitthaworn, Lester Gilbert, and Gary Wills. 2011. Towards a matching strategy of constructivism and instructionism. (2011).

[115] Katie Salen Tekinbas, Melissa Gresalfi, Kylie Peppler, Rafi Santo, and James Paul Gee. 2014. *Gaming the System: Designing with Gamestar Mechanic*. MIT Press.

[116] Jakita O Thomas. 2018. The Computational Algorithmic Thinking (CAT) Capability Flow: An Approach to Articulating CAT Capabilities over Time in African-American Middle-school Girls. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 149–154.

[117] Jakita O Thomas, Rachelle Minor, and O Carlette Odemwingie. 2017. Exploring African American Middle-School Girls' Perceptions of Themselves as Game Designers. In *Moving Students of Color from Consumers to Producers of Technology*. IGI Global, 49–61.

[118] Giovanni Maria Troiano, Sam Snodgrass, Erinç Arg, Gregorio Robles, Gillian Smith, Michael Cassidy, Eli Tucker-Raymond, Gillian Puttick, and Casper Harteveld. 2019. Is My Game OK Dr. Scratch?: Exploring Programming and Computational Thinking Development via Metrics in Student-Designed Serious Games for STEM. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children (IDC '19)*. ACM, New York, NY, USA, 208–219. DOI: http://dx.doi.org/10.1145/3311927.3323152

[119] E Tucker-Raymond, D Torres-Petrovich, K Dumbleton, and E Damlich. 2012. Reconceptualizing together: Exploring participatory and productive critical media literacies in a collaborative teacher research group. *Reconceptualizing the literacies in adolescent lives: Bridging the everyday/academic divide (3rd ed., pp. 224–243). Mahwah, NJ: Lawrence Erlbaum* (2012).

[120] Selen Turkay, Daniel Hoffman, Charles K. Kinzer, Pantiphar Chantes, and Christopher Vicari. 2014. Toward Understanding the Potential of Games for Learning: Learning Theory, Game Design Characteristics, and Situating Video Games in Classrooms. *Computers in the Schools* 31, 1-2 (April 2014), 2–22. DOI: http://dx.doi.org/10.1080/07380569.2014.890879

[121] Jan-Paul Van Staalduinen and Sara de Freitas. 2011. A game-based learning framework: Linking game design and learning. *Learning to play: exploring the future of education with video games* 53 (2011), 29.

[122] R Vinayakumar, KP Soman, and Pradeep Menon. 2018. Digital Storytelling Using Scratch: Engaging Children Towards Digital Storytelling. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 1–6.

[123] Michael Gr. Voskoglou and Sheryl Buckley. 2012. Problem Solving and Computational Thinking in a Learning Environment. *CoRR* abs/1212.0750 (2012). http://arxiv.org/abs/1212.0750

[124] Julie Warner. 2014. Writing in virtual worlds: Scratch programming as multimodal composing practice in the language arts classroom. In *Bridging Literacies with Videogames*. Brill Sense, 187–207.

[125] Scott Warren, Mary Jo Dondlinger, Richard Stein, and Sasha Barab. 2009. Educational Game as Supplemental Learning Tool: Benefits, Challenges, and Tensions Arising from Use in an Elementary School Classroom. *Journal of Interactive Learning Research; Charlottesville* 20, 4 (2009), 487–505. https://search.proquest.com/docview/211210319/abstract/601290A544A24DD6PQ/1

[126] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016a. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology* 25, 1 (Feb. 2016), 127–147. DOI: http://dx.doi.org/10.1007/s10956-015-9581-5

[127] David Weintrop, Nathan Holbert, Michael S Horn, and Uri Wilensky. 2016b. Computational thinking in constructionist video games. *International Journal of Game-Based Learning (IJGBL)* 6, 1 (2016), 1–17.

[128] Charlotte Lærke Weitze. 2014. Learning, Education and Games. ETC Press, Pittsburgh, PA, USA, 225–249. http://dl.acm.org/citation.cfm?id=2811147.2811160

[129] Linda Werner, Jill Denner, Michelle Bliesner, and Pat Rex. 2009. Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study. In *Proceedings of the 4th International Conference on foundations of digital games*. ACM, 207–214.

[130] Amanda Wilson, Thomas Hainey, and Thomas Connolly. 2012. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *6th European conference on games-based learning (ECGBL)*. 4–5.

[131] Jeannette M. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. DOI: http://dx.doi.org/10.1145/1118178.1118215

[132] Jeannette M. Wing. 2008. Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 366, 1881 (Oct. 2008), 3717–3725. DOI: http://dx.doi.org/10.1098/rsta.2008.0118

[133] Min Lun Wu and Kari Richards. 2011. Facilitating Computational Thinking Through Game Design. In *Proceedings of the 6th International Conference on E-learning and Games, Edutainment Technologies (Edutainment'11)*. Springer-Verlag, Berlin, Heidelberg, 220–227. http://dl.acm.org/citation.cfm?id=2040452.2040499