

Studying situated learning in a constructionist programming camp: A multimethod microgenetic analysis of one girl's learning pathway

Katarina Pantic

**Utah State University
Logan, USA**

katarina.pantic@aggiemail.usu.
edu

Deborah A. Fields

**Utah State University
Logan, USA**

deborah.fields@usu.edu

Lisa Quirke

**University of Toronto
Toronto, Canada**

lisa.quirke@mail.utoronto.ca

ABSTRACT

Computationally generated data have increasingly been used to provide insights into individual students' learning in constructionist learning environments. However, such studies have either missed examining the influence of local, physical environments, or they have taken students out of the situated scenarios to study them in isolation. In this paper, we explore an expanded methodological approach in order to examine how computationally generated data insights can potentially be informed or expanded with a microgenetic approach. To achieve that, we examine one ten-year-old novice girl's learning of programming in a week-long Scratch camp, applying a microgenetic approach to analysis across multiple forms of data, from traditional observational and artifact documentation to frequent, computationally generated save data. The findings highlight the utility of this approach in identifying Mila's growing engagement with coding, as well as the iterative and social nature of her learning experiences with Scratch.

Author Keywords

Computer science education, do-it-yourself media, learning trajectories, microgenetic methods, big data, Scratch

ACM Classification Keywords

H.5.m. Measurement, Computer Programming, Learning Trajectories.

INTRODUCTION

Constructionist learning environments can be challenging spaces to examine. This is, in part, because students learn by making things that are personally meaningful to them, leading each individual down a potentially unique learning pathway (e.g., [29]). In addition, constructionist learning environments are social, providing an environment where students and mentors learn from each other (e.g., [9]), value

mistakes (e.g., [3, 21]), share their designs (e.g., [10, 19]), and learn the culture of the community [20]. Not only are learners taking potentially diverse pathways to learning as they create original designs, but their activities are situated in a social community [23] where others' ideas, actions, and designs influence learning. How can we understand and analyze learning in these rich, complex environments?

While ethnographic-style observation and repeated artifact documentation have traditionally formed the backbone of research on constructionist communities in the past, computationally generated data have increasingly come into play to provide insight into individual students' learning. These data may consist of clicks, chat, project saves, and other recorded actions that take place on screen. In environments where students are working on different activities and projects, these data may provide some purchase on how students are learning. For instance, [8] used logfile data from MOOSE Crossing, an online programming platform and community, to document the "situated support" for learning of one girl new to the community. She demonstrated the importance of social, personalized help in the relationship two girls (one experienced, one new) developed over one weekend as they navigated projects, tutorials, and each other's expertise. Using a somewhat different approach, Monroy-Hernandez [26] studied the practice of remixing in the online Scratch community, using many means to illuminate the content and social relationships that affected remixing, including the relevance of acknowledging the original creator in the online community [27]. When these and similar studies investigate learning in online communities such as MOOSE Crossing or Scratch, where data are available that illuminate social interactions (i.e., chat or remixing), they can provide a situated lens on learning. What can computationally generated data (or more colloquially "big data") contribute to understanding constructionist environments that are not completely virtual?

Recognizing the importance of physical environments for learning, multimodal analytics have taken steps to understand learning patterns exhibited through computational documentation of gestures (e.g., [36]) and eye movement (e.g., [4]). However, in order to do this, researchers have often taken students out of the more natural

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

IDC '16, June 21-24, 2016, Manchester, United Kingdom

© 2016 ACM. ISBN 978-1-4503-4313-8/16/06 \$15.00

DOI: <http://dx.doi.org/10.1145/2930674.2930725>

situated scenarios to study students engaging in particular tasks in an isolated setting. Although there are calls for using multimodal analytics in more socially authentic scenarios, in large part the field is still developing the methods to do so [36]. One exception to this approach is tracking individuals' movement in constructionist settings to study how long students spend at individual stations (e.g., [35]). Without additional data on what students do or with whom they interact, however, it is difficult to understand their learning from a situated perspective.

In this paper we seek to understand one novice girl's pathway to learning programming in a 30-hour Scratch Camp. We do so by analyzing multiple forms of data, including traditional observation and artifact documentation, as well as frequent, computationally generated save data (i.e., big data or backend data). The former provide insight into the camper's motivation, personal interests, project goals, and the conversations and supports that influenced her learning. The latter provide a microgenetic lens on her learning, enabling us to see minute-by-minute progress on her projects, the introduction and deletion of programming commands, and shifts to new programming strategies. Together these data provide insight into this case study's situated learning of programming. This research is part of a larger study investigating authentic assessments of learning in constructionist programming environments by developing quantified measures of programming concepts in Scratch projects. Here we apply those measures to explore what we can learn about one camper's learning pathway from these different perspectives. Thus this paper is as much an exploration of method as it is an investigation of learning.

BACKGROUND

The promise of big data to illuminate learning pathways of novice programmers has prompted a growing number of recent studies in this area (see [2], 2014 for a review). These studies have gathered and applied big data in a variety of ways, for example to find and track patterns of programming across hundreds or even thousands of participants in free choice environments, participants whose programs may differ substantially from others. To do this, researchers have sorted individual code blocks into distinct categories of programming concepts (i.e., loops, variables, conditionals, and Booleans). For example, in a two-year study of a Computer Clubhouse [24], researchers used frequency counts of programming blocks in Scratch to stand in as measurements for youths' use of these programming concepts. Others have built on frequency counts of blocks to assess the quality of users' programs on the Scratch site, adding on analytical methods such as K means clustering or latent class analyses to find and highlight classes of programmers [14] or types of programming trajectories [37]. Wolz, Taylor and Hallberg [34] have also developed specific tools to assess programming, such as The Scrape tool. This tool allows educators or researchers to alter this frequency count technique by dragging individual or multiple programs into the tool, and categorizing blocks by type (variables,

looks, motion, control, etc.) [34]. Though these studies have pioneered our ability to assess the behavior of a large numbers of users on websites such as Scratch, they all share a similar assumption: namely, that the simple presence of specific blocks in programs equals an understanding of these concepts by users. In their framework for examining computational thinking, Brennan and Resnick [6] emphasize this point by highlighting the potential for serious differences between learners' programs and their actual understanding of how their code functions.

Relying on proxy measures for learning, such as frequency counts of blocks, is not the only challenge faced by researchers attempting to study learners' experiences with programming. The very nature of constructionist learning environments makes it difficult to authentically assess the learning of computer science concepts using big data alone. Some research has relied on comparing kids' programs to set solutions such as Repenning et al [30], who assessed hundreds of kids' game code in this way. Similarly, Blikstein et al [5] examined novice programmers' thinking as they solved a targeted checkerboard problem. In this way, they were able to identify different pathways to the most efficient solutions and document processes such as "sink states" (times when students were stuck on the problem and not progressing). Though these studies add to our understanding of kids' learning, still others have begun to develop big data measures that are more compatible with constructionist learning strategies, such as work by Werner, McDowell and Denner [33] and Berland et al [2]. The former study investigated the high level actions of programmers using the Alice programming environment with the goal of developing an analysis of problem-solving strategies novice programmers use. Along a similar vein, Berland et al [2] tracked students' progress across two hours of programming in iPro. During this playful introduction to programming, the researchers used learning analytics to look at the types of programming strategies kids employed (including tinkering, exploring, and refinement). A limitation of these studies was that though they applied big data techniques to assess novice programmers within complex constructionist learning environments, they focused exclusively on programming measures, and therefore were unable capture the social or contextual factors that shape learning.

How, then, can researchers harness the potential of automated or big data measures to track the progress of learners, while still capturing the complex nuances of deeply social and authentic constructionist learning environments? In this paper, we outline the utility of an exploratory mixed-methods, microgenetic approach for this task by highlighting the trajectory of learning in a case study of a novice Scratch user. Microgenetics involves a variety of techniques used to study the process of learning on a moment-by-moment basis [11], and can include high density observations, or think aloud interviews, accompanied by competitive argumentation analysis (e.g., [32]). Though this approach has been applied in broader studies of education (i.e., [12])

to highlight factors that shape learning [7], it has not yet been widely used in studies of programming; one exception is Murphy et al's [28] research on strategy development by novices. By combining backend data from the Scratch programming environment with detailed observational data highlighting learners' experiences in a 30-hour programming camp, we were able to document not only what changes kids were making to their programs on a moment-by-moment basis, but also the types of challenges, motivations, and peer interactions that shaped their learning. This study combines big data with richly contextual qualitative sources to focus on one novice's trajectory of learning, presenting a case study of Mila (pseudonym), age 10. By developing this exploratory, mixed method, microgenetic approach, we hope to use this snapshot from our larger study of kids' programming to eventually develop an efficient and useful way to apply big data to assessment of programming in constructionist environments. The development of such an approach would answer a need that currently exists for both researchers and educators in computer science education [2].

METHODS

Data Context

Scratch 2.0 is an online programming environment (<https://scratch.mit.edu/>) that engages children in making creative media while learning the basics of coding. Developed by MIT and launched in 2007, Scratch uses an intuitive drag and drop structure of "snappable" coding blocks designed to diminish the number of syntactic errors. Scratch's design supports a wide range of participation from different users, from a low entry threshold for beginners to vast programming opportunities for more advanced coders [22]. This takes place in the context of the Scratch community, which allows users to learn through both sharing and receiving feedback on their projects [25].

The data in this paper come from a larger study on kids learning to code in Scratch collected at a set of three, week-long "Scratch Camps" at a university in the U. S. Rocky Mountain Region in summer 2014, held in partnership with the local 4-H club. Each camp took place across 30 hours in five days, with approximately 25 hours devoted to programming. Campers (n=67) aged 10-13 were largely novice programmers: only a few had more than 2 hours of experience coding, and most had none. Campers were taught and mentored by one professor and four graduate students, with several 4-H club members also offering assistance. Using Scratch 2.0, kids created a series of open-ended projects within constraints designed to encourage them to learn specific programming concepts and techniques (see Table 1; see [18] for a fuller description of the pedagogy). A variety of projects were included: Scribble time & Name project (Day 1), Story project (Day 2), Music Video project (Day 3), Video Game project (Day 4), and a Free Choice project (Day 5). Each project promoted specific programming strategies (e.g., broadcast, timer), which were introduced at the beginning of each day. Campers received

feedback on their projects throughout the camp from both peers and mentors. Scratch Camp concluded with a graduation ceremony and a special gallery walk where interested parents and other guests could browse campers' completed work.


Day #	Project Name	Fixed Project Requirements
Day 1:	Scribble & Name Project	Introduction to Scratch interface Name Project: each letter is an animated sprite
Day 2:	Story	2+ sprites that interact; 3+ scenes; 3+ magic effects; obvious beginning, middle, and end; 1+ broadcasts
Day 3:	Music Video	Get two sprites to interact; use 
Day 4:	Game	Introduction screen with instructions; user controls (mouse or keyboard); 2+ levels; variables used; end screen
Day 5:	Free Day	Finish unfinished projects or start a new one

Table 1. Scratch Workshop Layout

Data Collection

Though the larger study collected data at all three Scratch Camps, those presented in this paper highlight the learning trajectory of one participant, Mila (pseudonym), age 10. We chose Mila as a focus for in-depth analysis for several different reasons. First, she was one of the younger children in our camps and was completely new to Scratch and to coding. As we believed that some of the programming concepts taught in Scratch Camp may have been more challenging for younger learners to acquire, we determined that a case-study of a younger learner might illustrate some of these challenges. Second, Mila was one of the 15 participants (out of 65) on whom our participant-observer kept detailed observational notes. Finally, unlike most other children who enrolled in our camp, Mila was unique in the fact that she began the week-long camp completely disinterested and unimpressed with computer programming. The shift she experienced after encountering Scratch, one that led to the development of her interest in programming, made her an interesting case-study subject.

The data analyzed for this paper came from several sources. Using the online version of Scratch 2.0, we were able to gather "snapshots" of campers' code with the assistance of the Scratch Team at MIT. These took the form of JSON (JavaScript Object Notation) files collected online every time the kids shifted from editing backgrounds, costumes, or sound back into coding, and by default every two minutes. This provided 32,465 code snapshots in text-based format, 553 from Mila's programs; this source of data represents the big data component of our research. One limitation was that

though the JSON files contain all of the text included in a program, these files could not be run in Scratch to see if the project worked or what it actually looked like while running. To complement these data, we manually collected campers' projects, which were saved approximately once every 75 minutes, at the end of each of the four programming sessions held each day. This resulted in 17 individual saves of Mila's work.

In addition to the JSON files and manually saved projects, we analyzed two other sources of data. The first consisted of observational notes taken by a researcher focusing on five campers per camp, including Mila. These notes provide additional context on campers' goals, interactions, struggles, and learning moments throughout the camp. The participant-observer interacted with Mila (and four other campers) in a reserved fashion, asking her to think aloud about the difficulties she was facing, decisions she was making, and waiting for her to come to conclusions on her own. Mila was prompted gently only when she asked for help or looked frustrated. The final data source consisted of video-taped debrief sessions held among the research team at the end of each day of camp. In these sessions, the counselors discussed their observations, the campers' learning, and any interactions they recalled.

Developing Measures of Programming

Our study of Mila's learning is set against a backdrop of prior work creating and testing measures of programming relevant to the context of Scratch camp (for more details see [18]). With backend data from all of the camps in hand, we began a process of cleaning and reducing it, creating a Scratch JSON parser to capture and quantify details of each code snapshot (cumulative measures). We then developed a layer of analyses that quantified changes in Scratch programs over time. These became a series of quantitative measures in several categories of programming concepts, inspired by Brennan and Resnick's [6] outline, but including other concepts pertinent to evaluating youths' learning in the Scratch Camps: Booleans, conditionals, event driven & parallelism, initialization, loops, randomization, and sensing (for more details see: www.workingexamples.org/example/show/727). Each measure was tested and developed iteratively, carefully comparing the quantitative measures with campers' actual programming by looking at the series of projects created by about a third of campers attending Scratch Camp. Through this process we realized that several measures provided duplicative information. For instance, initialization of backgrounds versus initialization of costumes provided the same information about campers' learning as they happened at similar times. We combined some measures (e.g., several measures of initialization) and deleted others, narrowing our list of measures to 26 that were useful for tracing campers' learning. This list was specific to the context that we studied. Researchers in other contexts (clubs, classes, museums, online environments) that have different learning goals may

find it helpful to use some of our broader list of 46 measures or to develop their own.

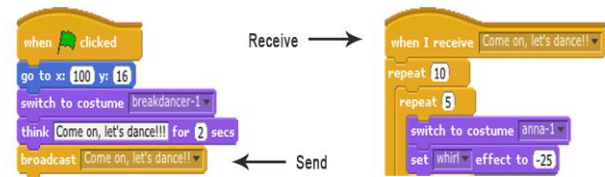


Figure 1. Functional broadcast with matching receive.

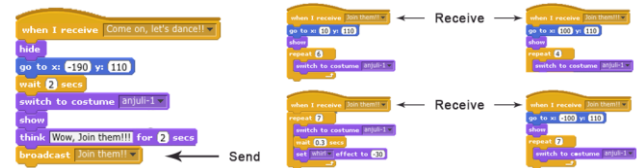


Figure 2. Broadcast with multiple receives in Scratch.

Although we included frequency counts of blocks (as done previously in other studies [14, 24, 34]), our measures of programming concepts include several features that take a next step in evaluating whether youth are using a concept correctly. For instance, each measure only included blocks attached to "event hat" blocks; "event hats" are those blocks that, when connected, integrate test blocks into the program in Scratch. Since students can drag blocks into the floating spaces in Scratch, they can test blocks individually without connecting them to an "event hat". After some analysis we decided not to include these "floating blocks" in our study. In addition, we looked for uses of sets of blocks. In the case of measuring campers' use of broadcasts with a type of event driven programming, we looked for whether broadcasts were used functionally. In this case, if a broadcast was sent, was it also "received" (see Fig. 1)? At a more challenging level, did campers use multiple receives for a single broadcast, showing some understanding of parallel programming (see Fig. 2)? With these measures we were able to approach analyzing the trajectory of one camper's programming.

Case Study Analysis

The analysis of Mila's trajectory of learning began in two simultaneous ways. First, we looked at her progress by graphing each of the 26 measures over time across her backend saved projects (i.e., JSON files), sometimes looking at multiple measures at once to see whether increases or decreases occurred at similar time points. We also analyzed other changes, such as the number of programming blocks or art changes over time. At the same time, we hand-analyzed Mila's Scratch projects, looking at four aspects: i) project functionality, ii) requirement satisfaction, iii) changes introduced from one save to the other, and iv) identification of what she spent her time doing. We noted which concepts Mila struggled with and when she made breakthroughs in learning. Comparing her progress across the frequent

backend saves as well as the near-hourly project saves, we identified specific time points that suggested changes in Mila’s programming and learning. Then we turned to the observational data to investigate these moments further.

We studied debrief meeting transcripts and observational data in several iterative cycles. First, we isolated all points at which Mila was mentioned and performed open coding, consistent with a grounded theory approach to field notes [13]. Second, we looked at her progress from the beginning to the end of Scratch Camp with a focused coding approach [13], highlighting moments of learning, excitement, frustration, changes she introduced to her programs (and why she introduced them), and interactions with others. In a repeat pass through these data, we also compared them to the moments identified in the earlier analysis of backend and project save data. Wherever possible we noted the contexts in which key learning moments, looking at what led up to and followed these moments, to tell a fuller story of Mila’s learning experiences. Our combined analyses provide a rich understanding of Mila’s entry into programming during the Scratch Camp with insights that one form of analysis could not provide.

RESULTS

By applying an exploratory mixed methods approach, a method which combined frequent backend saves with observations and traditional project saves for a close moment-by-moment analysis, we documented and interpreted several dimensions of one girl’s (Mila’s) learning in a week-long constructionist programming camp. First, we noted how her aversion to Scratch shifted to engagement as she was able to incorporate her personal interests into her Scratch projects. Second, we found that her understanding of programming concepts developed in iterative cycles; it took many applications and explanations of a new programming concept in her projects before she understood it deeply. Finally, we saw a strong social dimension of her learning through interaction, advice, and feedback she received from her peers and camp leaders. The following sections describe these three aspects of Mila’s learning pathway in more detail, highlighting the role of different forms of data in uncovering them.

A Shift in Engagement

Unlike the vast majority of participants at the Scratch Camps, Mila did not begin the week-long camp with an enthusiasm for programming: in fact, she expressed disappointment at having to attend. She had been interested in attending an art camp instead, and was unimpressed with the small number of girls enrolled at Scratch camp that week (only three girls attended the first Scratch Camp, including Mila). However, by the end of the first day, Mila’s attitude had dramatically shifted and she told her mother that she liked Scratch, an interest that continued to develop throughout her time at camp.

How Mila’s engagement changed was not immediately visible. Looking first at the manually saved Scratch projects (where we could see and play her entire project), it appeared that she was not actively working with the code of Scratch. There were very few changes between her initial projects, which at first seemed surprising since she spent quite a bit of time on them (1-1.5 hours). In addition, the measures of programming concepts revealed little about Mila’s first day other than that she began to initialize her sprites’ location (using x, y coordinates) as well as the backgrounds (by setting the initial background). Comments recorded by counselors during the daily debrief and observational notes suggested that Mila enjoyed the art and aesthetics of working with Scratch. For that reason, we reimagined the measures we generated in order to capture her interest development and interest-driven learning by comparing blocks she worked on with the number of artistic changes she made.

Analyzing the time Mila spent making changes to the scenes, costumes and other visual effects of her programs, Table 2 shows the proportion of saves where Mila changed the costumes (each programmable sprite can have multiple “costumes” or looks) or backgrounds of her project, illustrating her keen interest in art and drawing. In 59% of the automated saves on Day One, she edited the drawings, compared with 49% of saves that included changes to the code. Over the five days of Scratch Camp, Mila appeared to spend less time editing the drawings, and was increasingly engaged with coding. Comparing the rows of Table 2 shows the proportion of saves where artwork changed versus the proportion of saves where code changed. These categories are not mutually exclusive, as both art and code can be changed in a single save. After the first day, the proportions shifted from being art-dominant to coding-dominant as more saves involved changes in code than changes in art. Days 3 and 5 are particularly interesting as the proportion of saves related to code is double or triple (respectively) to those of art. This points to the intense work that Mila put into her Music Video, which she began on Day 3 and continued as her final choice activity on Day 5.

	Day 1	Day 2	Day 3	Day 4	Day 5
Proportion of saves with costume or background changes	59%	45%	31%	45%	24%
Proportion of saves with coding changes	49%	52%	61%	48%	71%

Table 2. Proportion of saves relating to changes in art or coding.

Another way to look at Mila’s engagement with programming is with the number of blocks she altered in a given day (see Table 3). On Day 1, the backend (JSON) files reveal that she actually experimented with more than 150 code blocks. However, as was apparent from the Scratch saves collected at the end of every session, she did not keep

most of the changes, demonstrating a form of experimentation and exploration of different coding blocks on her first day with Scratch. This explains why we could not see evidence of her programming from running the manual project saves of Mila's work. The backend data provide a much fuller picture of the minute-by-minute changes she made. Looking only at the number of blocks changed, Days 2 (Story Project) and 3 (Music Video Project) appear to be days of high engagement for Mila in terms of code. As we explore in the next two sections, these were the days when she learned a great deal about some new programming concepts (events, parallelism, and conditionals). In addition, these projects were also much longer (day-long projects versus hour-long projects) than those of Day 1.

	Day 1	Day 2	Day 3	Day 4	Day 5
Total coding blocks added or removed	152	269	289	149	99

Table 3. Total number of coding blocks added or removed each day.

Analysis of observational notes showed how motivated Mila was to work on the Story and Music Video projects. For example, on Day 2, Mila “sounded really excited about” her story with princesses and ninja unicorns, wiggling her fingers enthusiastically as she talked. Similarly, on Day 3, Mila was immersed in an all-girl collaborative Star Wars Music Video project in which she included elaborately hand-drawn sprites (e.g., R2D2) and accessories (e.g. light sabers), making Princess Leia a warrior who fights Darth Vader and saves Luke (see Figure 3). She and her friend chose to pursue this as their Free Choice project on Day 5, integrating each girl's section of the music video together and creating additional scenes. From initial group planning about how the project would look, to designing and coding, observational notes describe Mila as very engaged and motivated to work on this project. Although the final day did not involve as many additions or deletions of code blocks (see Table 3), it was a time when Mila and her two friends worked carefully to debug the code of their music video, taking turns smoothing out the final product.

While Mila began the Scratch camp with a spirit of antagonism, the ability to integrate art into her projects seemed to provide an entry into programming. Throughout her projects she customized nearly all of the sprites and backgrounds, and these changes were a key part of what made audiences respond to them. In particular, the feminist rendition of the Star Wars music video she created with the other two girls drew enthusiastic responses from the other campers and praise from Mila's mother. Other campers followed the girls' lead by giving a light saber to Princess Leia (who never wields a light saber in the films) in their own videos. This practice of taking existing interests and integrating them into new activities follows Azevedo's [1] “lines of practice” theory where he demonstrates that interest

is not a single entity (such as an interest in programming) but is made up of lines of practice that carry across spaces and help define the ways individuals engage in a hobby. Integrating handmade art, relationships with girls, and an interest in popular media (like Star Wars) show the way that Mila's practice of programming took shape over the camp. Evidence from the backend data provided a high level view on Mila's engagement with code and drawings, showing increased use of code on specific days, and hinting at a shift from adding code to refining code at the end of the Camp. It also filled in the blanks of



Figure 3 A snapshot from Mila's Star Wars Music Video.

what Mila did on the first day, when few of the many changes she made showed up in the manually saved Scratch projects.

In the next sections, we look more closely at Mila's learning of specific programming concepts and how these were situated within the social environment of the camp. The combination of data sources, in particular, the qualitative investigations of her projects, were key in uncovering this finding, something that was not apparent in our original big data measures.

Iterative Learning

While Mila quickly learned simpler programming commands, the process of learning more complex concepts, such as broadcast, was not as straightforward. In Scratch, broadcast is a command that triggers all scripts in all sprites that begin with a matching “when I receive <msg>” trigger block [25], and functions as a form of inter-sprite/background communication and synchronization (see Figures 1 and 2). The microgenetic method of analysis applied in this study revealed the complexity of that trajectory, together with the factors that framed or enhanced this learning experience. This interplay of factors, however, was not immediately transparent. Even though the measures we developed automatically captured the number of blocks she worked with, when she worked with them, and how frequently, it was only by cross-referencing all available data that we gained a fuller picture of when and how Mila's understanding of broadcast developed and expanded over time.

Our observational notes reveal that it took Mila at least four interactions with various camp leaders over several hours on Day 2 before she finally mastered the use of broadcasts (see Fig. 4). We identified four key instances that influenced Mila's learning of broadcasts: listening to instruction on the concept (interaction {1}, 9:00am), talking to a counselor one-on-one about the function and usage of broadcasts (interaction {2} at 10:10am), facing a dysfunctional broadcast where she learned more about "debugging" (interaction {3} at 11:24am), and finally, learning about multiple receives with the same broadcast (interaction {4} at 12:46am). Figure 4 visualizes these instances: a cumulative representation of Mila's work with broadcast (x-axis represents time, y-axis represents the number of blocks used). Numbers {1–4} (also marked with a red dot), as noted above, represent moments when Mila learned about broadcast from different interactions with various camp leaders. The interactions were mapped after careful comparison of observational notes with trends from our measures applied to backend JSON saves. In this section, we describe and explain those four cycles of exposure to the concept, as well as how these influenced Mila's understanding of broadcast.

After hearing the lecture (interaction {1}) and seeing examples of broadcasts in some existing projects at the beginning of Day 2, Mila started working on her story. While the initial lecture on broadcast exposed Mila to the concept, she did not immediately apply it. The observer-participant noted: "[Mila] had used 'when the space bar clicked' and then a variety of wait commands. I asked her if she thought about using a broadcast to do what she wanted. She was unsure on how to do that, so I explained..." As this excerpt shows, hearing a lecture on broadcast was not enough for Mila to start using the code in her own project. When asked why she had not yet used it, she did not know how it could be useful for her project, nor did she remember how it worked. At this point, the observer helped her set up her first broadcast, explaining its role and functionality one more time (interaction {2}). Soon after, shown in the incremental steps of the green line in Figure 4, Mila added a few broadcasts on her own. This development of understanding is aligned to one of the major facets of constructionism, which is the important role of "objects-to-think-with", such as Scratch building blocks, play in effective appropriation of knowledge [20]. That is, even though she initially could not connect the newly gained information on broadcast to her project idea, by using concrete broadcast examples in her projects, she constructed her own knowledge about broadcast and learned to use it properly and independently.

Despite being successful in setting up several fully functional broadcasts on her own, Mila did not appear to completely understand how they worked. When interaction {3} occurred, Mila was frustrated that the new scene she had just programmed was not starting and asked for help. The counselor immediately realized that while Mila had used a <when I receive message 2> block, she had not put in the

necessary <broadcast message 2> block to initiate the new scene. The counselor asked Mila to show her "where 'message 2' was being broadcast" from. Not being able to find that particular block of code, Mila realized it was missing and added in the appropriate block where she wanted it in her program. Debugging her code in this way helped Mila to make her understanding of broadcasts more explicit. To our knowledge she did not make this error again.

Later in the day, Mila learned the importance of setting up the different types of code in choosing when to use broadcast and when other strategies would work better. Mila was trying to set different backgrounds to different scenes, using broadcast to switch them. While this worked in most instances, it did not work for the first scene because broadcasting messages must be under another "event hat" in Scratch such as the green flag (<when green flag pressed>). A counselor assisted her in realizing that she could not use broadcasts to set the first background and that the green flag was most useful in this instance. After this interaction, Mila made a connection between two challenging concepts for novice programmers, initialization and events (with broadcasts). Additionally, on this same occasion, one of the camp leaders suggested that Mila might be interested in using multiple "receives" with a single broadcast as an efficient way to trigger multiple actions to start at the same time (interaction {4} in Figure 4). Once introduced to this concept, a form of parallelism in Scratch, Mila continued to apply broadcasts (some with a single receive and some with multiple receives) in her Story project, continuing these strategies fluently in the Music Video project she worked on during Day 3. While broadcasts were not generally useful to Mila (or many other campers) during the Video Game project on Day 4, she continued to use them when she picked up her Story and Music Videos again at the end of Day 4 and throughout Day 5.

Mila's experiences call into question the assumption that the presence of specific blocks in kids' programs means understanding their use (e.g., [14, 25, 34]). Similarly, our initial analytical measures in this study—which identified whether or not concepts were being used correctly (e.g., a broadcast is used with a matching receive, and is part of a functional piece of code)—were also insufficient in showing whether Mila truly understood a concept. Her learning was an iterative process, and correct application of broadcasts did not mean that she fully understood how or when to use them. Our microgenetic approach applied across different forms of data helped us identify key moments in this learning process.

Analyzing measures of broadcast use through the backend data enabled us to see minute-by-minute when Mila introduced new broadcasts, when this happened in quick succession, and when she began to use broadcasts in a new way (i.e., using multiple receives).

Observational data provided further insights into when requests for help and help received supported understanding. Integrating these insights from different angles, we see how

over the course of several hours Mila was introduced to the concept, applied it, debugged it, utilized it in new ways, and used it in new projects, giving a much fuller and detailed picture of her learning process.

Social Aspects of Learning

Mila's learning of programming concepts was situated not only within the project goals, formal Camp teaching, and her

interactions with the camp leaders, but also among her two camp friends and the broader audience of her camp peers.

Investigating her learning of another programming concept, conditionals, provided deeper insight into the peer-to-peer influences that shaped Mila's learning trajectory with programming. Once again, comparing analysis of frequent

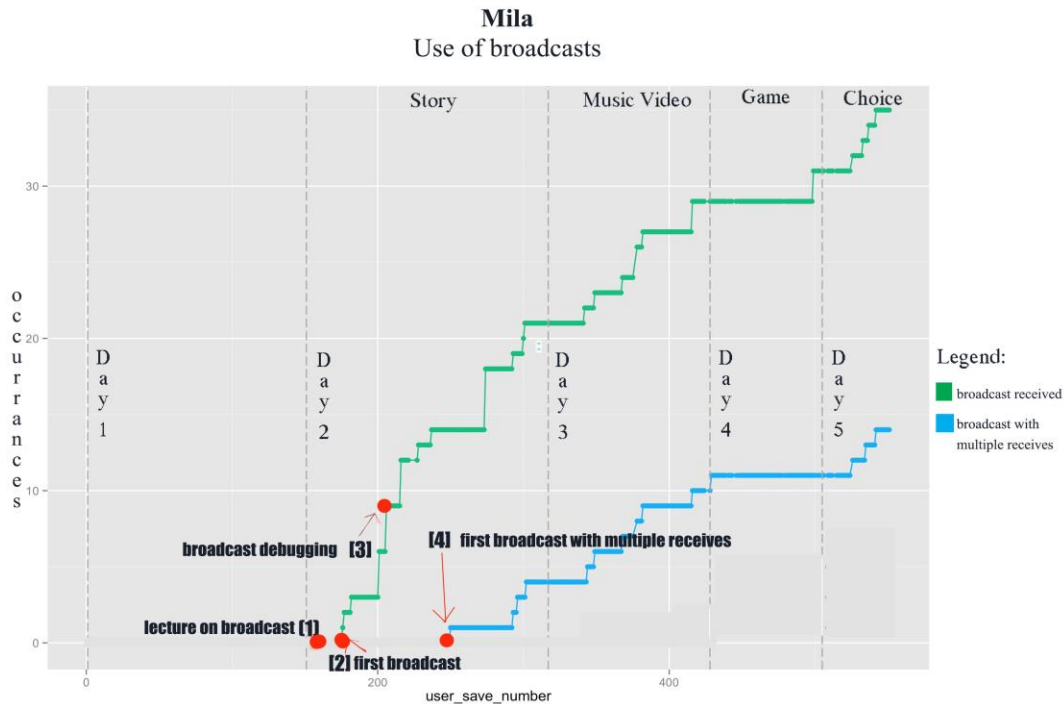


Figure 4. Cumulative measures of Mila's use of broadcast over time.

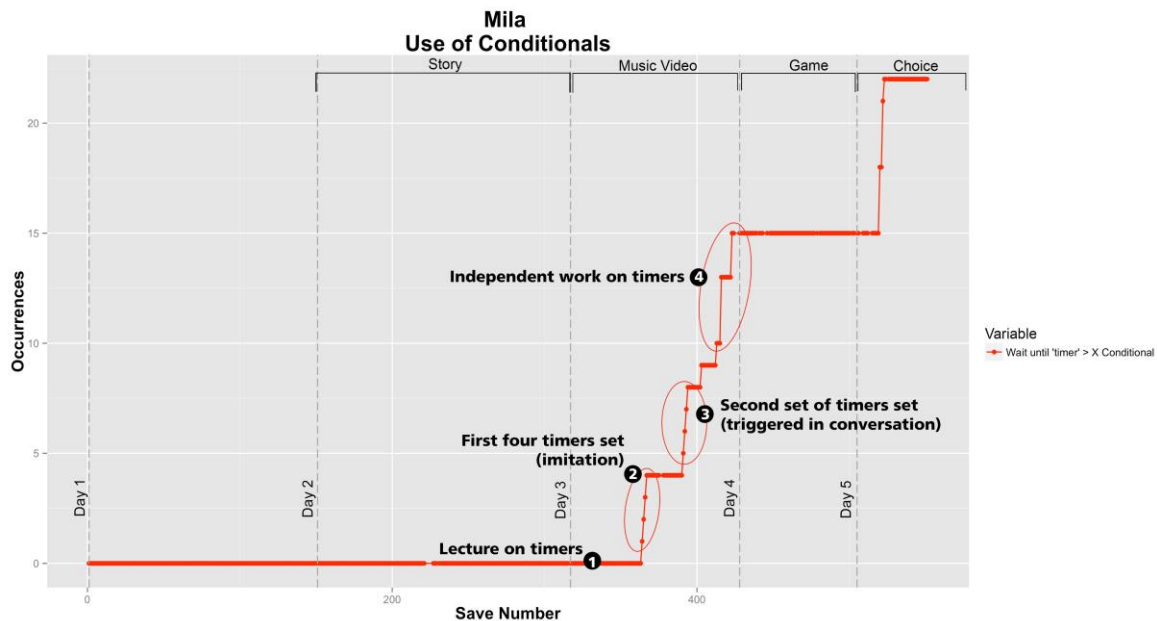


Figure 5. Cumulative measure of Mila's use of conditionals (wait until)

backend saves with qualitative observations in a microgenetic fashion, provided a rich picture of how peer interactions influenced Mila's learning process.

Some key examples of peer interaction that influenced Mila's understanding of conditionals occurred on Days 3 and 5 in the context of making a Star Wars-themed music video. On these days, campers created original Music Videos set to 30-second-long segments of music. This project was designed to support a shift from programming long sequences with `<wait X seconds>` to using conditionals and variables for the first time. Counselors encouraged campers to use a new programming concept for this project: a Scratch-defined variable in the form of a code block called `<timer>` inside a conditional: `<wait until (timer >, X)>`. The goal was to introduce campers to conditionals and variables in this way. The `<timer>` was particularly useful in the music video as kids could time events on the Scratch stage with particular lyrics or sounds in their chosen song. However, learning to use conditionals within the music video was challenging for many of the kids.

Our data reveal that overhearing conversations between camp leaders and her peers sometimes helped Mila learn. Similar to social interactions described in Section 4.2, this type of behavior was not visible in the quantitative measures we developed, but was something we extracted from a comparative analysis of both her project saves and our observational notes. During the first session of the day, Mila was actively engaged in creating her Music Video, and added over 50 blocks of code. She did not, however, use the required `<wait until (timer>X)>` in the first version of her project, even after being introduced to the concept by camp leaders (see {1} in Fig. 5). Unlike her first use of broadcast, which was influenced by an interaction with one of the camp leaders, this time, it was a conversation she overheard between her peers and one of the counselors that prompted her to begin using the timer. As the observer described:

"I checked on Jane, because she was using waits all over in her code. I asked her why she needed the waits, instead of the timer variable. She said because she already had the code done and it was easier. I told her I would help her build one of the time variables, and then she could duplicate it and add it in."

Following this interaction, Mila suddenly exclaimed that she also had "too many waits [in her project], and needed some help" to fix that. Although Mila was not the one receiving feedback, overhearing the conversation made her inspect her own code and realize she also was not meeting the requirements. Interestingly, it was her friend who took over the role of instructor and shared the new approach, at which point Mila added several of the appropriate new scripts to her project (see {2} in Figure 5) and continued working on her music video. An hour later, when she encountered another

dysfunctional piece of code, it was again her peer who provided counseling and helped Mila solve the issue. On both occasions, the role of camp leaders/instructors was shouldered by a peer with their own expertise. This finding aligns with another important facet of constructionism: the importance of learning the culture of the community simultaneously to learning the content [20]. Since social support was a visible and encouraged characteristic of the camp, Mila and her peers learned and also taught each other, mimicking the culture of the community in which they participated.

Later, upon some prompting from the counselor, Mila added more `<timer>` blocks (see {3} in Figure 5), but this time she added them without needing assistance. The prompting only served as a reminder, but it was obvious that she "understood how to make the code again and how it worked." Though we learn about these interactions from our qualitative observations, by comparing the time stamps on the notes to the time stamps on the quantitative measures we are able to see if, and how much code she added upon each interaction. In other words, not only do our observational data allow us insight into social interactions, but we also got to see the outcome (if any) of such interaction through the detailed and frequent saves of our backend data. This rich insight into the learning process together with the factors influencing it is one of the biggest advantages of our combined methodological approach.

DISCUSSION

In this paper we sought to unveil one girl's situated learning in a constructionist programming camp over five days. We did so with an exploratory approach to a relatively unusual set of data that provided both a minute-by-minute look at the changes she made to her programs within the context of observations of her interactions and instruction at the camp. The computationally generated data measures we developed for this research—analyzing programming concepts across frequent project saves and snapshots of kids' coding—allowed us to identify progress Mila made in applying particular computational concepts. We could link applications of concepts to particular projects and times (i.e., broadcasts to the Story project, uses of some conditionals to the Music Video project). Determining the occurrence, time and frequency of concepts used enabled us to identify moments of learning with more precision than in similar studies accounting for learning programming in social context (e.g., [16]). This computational analysis far surpassed what could be done with a similar hand analysis of so many projects (over 500) and fulfilled calls to use big data for analysis of learning in constructionist environments [2, 11].

Importantly, this big data did not stand alone in providing insights into Mila's learning. By combining analysis across

frequent snapshots of code with in-person observations, we illuminated the key importance of repeated conversations between learners and counselors as well as the positive influences of nearby peers. This finding aligns with the view that designing a project is never a clean process but one that is sequential and adaptive, demanding specially developed skills supported in social context (e.g., [6, 11]). Further, hand-analysis of manually saved Scratch projects saved the aesthetic nuances in the project, both in looks, movements, and interactions on screen. Here we could see how Mila's project influenced others as a meme of light sabers spread across the camp, as well as how her personal interests were embedded in her projects.

Our multi-method approach has important implications for other uses of big data. Although there is understandable excitement about what big data can offer to understanding children's interaction with technology [2], it is imperative to remember that it cannot (at least not yet) provide a full view on learning from a situated perspective that acknowledges the role of social interaction, physical location, and personal histories. Multimodal analytics are indeed making strides to develop ways to trace movement, interaction, gesture, and attention (e.g., [35]). One future area of research is to bring these perspectives into conversation with each other in order to provide a fuller lens on learning. Using traditional data such as participant observations, video recordings, and interviews will provide an important check on understanding and interpreting these computational analyses, as they did in our study.

Further, designing constructionist environments to facilitate analysis and assessment of learning may be important to making such learning spaces available to more children. In our study, we accomplished this in part by adding some structure to the pedagogy of the Scratch Camps through four projects with creative constraints designed to target specific programming concepts [18]. While every child made unique projects, in large part they learned similar things that we could trace through the computational measures we developed. Other environments using Scratch or other programming languages may require different methods. As part of the larger project we are developing a "FUN! Tool" (functional understanding navigator), an open source data engine (github.com/ActiveLearningLab) where measures for Scratch or other languages could be input to automatically analyze findings across programmers. The tool is meant to help researchers write and share automated data extraction and manipulation techniques with Scratch and other types of programming language and could be helpful in scaling this type of analysis.

Could this type of analysis be scaled up to a larger set of subjects or across additional contexts? Should it? Elsewhere [17, 18] we describe the application of our measures of programming across all three Scratch Camps, showing trends of learning across students and identifying areas of struggle and difference in learning. We could see trends easily

because each camp shared the same structure, philosophy, and set of projects, even if the participants were different. New contexts might require new measures of programming, which would need testing. However, scaling up the microgenetic approach used in this paper would be challenging since non-virtual (i.e., in person) contextual data is not easily done by computers, though some have developed means for analyzing data such as well-understood interviews [31] through computational means.

CONCLUSION

In this paper, we sought to uncover one girl's (Mila's) learning in an open-ended constructionist programming camp from a situated perspective on learning [23]. To this end we developed a microgenetic approach to studying her learning across frequent, computationally generated project saves, participant observation, and supplemental documentation of her projects that preserved aesthetic features. This allowed us to dig into Mila's shift of engagement and the role of social interactions and peer activity in her learning programming. Each form of data provided something the other did not, together providing deeper insights into her learning than could have been achieved otherwise. This exploratory case study contributes to studies of constructionist learning environments by applying the detailed lens of big data alongside more traditional forms of contextual information.

SELECTION AND PARTICIPATION OF CHILDREN

Participants in this study attended one of a set of three week-long summer programming camps for children aged 10-13 at a public university in the intermountain west of the United States in Summer 2014. Attendees were recruited from the local community via advertisements launched by the local 4-H Club. University IRB approval was obtained in advance of the study. Parents were presented with a consent form upon registration as well as in person on the first day of camp. Children were given an assent form as well as an explanation of the study on day one. Sixty-four campers (34 girls and 31 boys) consented to participate in the study out of 67 campers total.

ACKNOWLEDGEMENT

This material is based upon work supported by a collaborative grant from the National Science Foundation (NSF#1319938) to the second author. The views expressed are those of the authors and do not necessarily represent the views of the National Science Foundation, Utah State University, or the University of Toronto. Special thanks to Sayamindu Dasgupta and Mitchel Resnick of the MIT Media Lab for help collecting data, and to Victor Lee for providing helpful feedback on an earlier version of this paper. Nicole Forsgren, Xavier Velasquez, Tori Horton, Janell Amely, and Jason Maughan assisted in the analysis. Suzanne Fluty and Whitney King aided in leading the Scratch Camp along with several makers' club volunteers.

REFERENCES

1. Flavio S. Azevedo. 2011. Lines of practice: A practice-centered theory of interest relationships. *Cognition and Instruction*, 29, 2, 147-184.
2. Matthew Berland, Ryan S. Baker, Paulo Blikstein. 2014. Educational data mining and learning analytics: applications to constructionist research. *Technol. Knowl. Learn*, 19, 1-2, 205-220.
3. Paulo Blikstein. 2013. Digital fabrication and 'making' in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, 1-21.
4. Paulo Blikstein, Marcelo Worsley.* (in press). Multimodal learning analytics: a methodological framework for research in constructivist learning. *Journal of Learning Analytics*.
5. Paulo Blikstein, Marcelo Worsley, Chris Piech, Mehran Sahami, Steven Cooper, Daphne Koller. 2014. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *J. Learn. Sci.*, 23, 4, 561-599.
6. Karen Brennan, Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. Presented at the AERA (Vancouver, British Columbia, Canada).
7. Sara C. Broaders, Susan W. Cook, Zachary Mitchell, Susan Goldin-Meadow. 2007. Making children gesture brings out implicit knowledge and leads to learning. *Journal of Experimental Psychology: General*, 136, 539-550.
8. Amy Bruckman. 2000. Situated support for learning. Storm's weekend with Rachel. *Journal of the Learning Sciences*, 9, 3, 329-372.
doi:dx.doi.org/10.1207/S15327809JLS0903_4
9. Amy Bruckman. 1998. Community Support for Constructionist Learning. *The Journal of Collaborative Computing*, 7, 47-86.
10. Mihaly Csikszentmihalyi, Rustin Wolfe. 2001. New conceptions and research approaches to creativity: Implications of a systems perspective for creativity in education. In: K. A. Heller, F. J. Manks, R. Subotnik, & R. J. Sterberg (Eds.). *International Handbook of Giftedness and Talent* (Oxford, Elsevier), 81-93.
11. Clark A. Chinn, Bruce L. Sherin. 2014. Microgenetic methods. In *The Cambridge Handbook of The Learning Sciences*, 171-190.
12. Andrea A. diSessa. 2014. A History of Conceptual Change Research. In *The Cambridge Handbook of The Learning Sciences*, 171-190.
13. Robert M. Emerson, Rachel I. Fretz, Linda L. Shaw. 2011. *Writing ethnographic fieldnotes*. University of Chicago Press.
14. Deborah A. Fields, Michael Giang, Yasmin Kafai. 2014. Programming in the wild: Trends in youth computational participation in the online Scratch community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (New York, NY, USA), 2-11.
15. Deborah A. Fields, Yasmin Kafai. 2010. Knowing and throwing mudballs, hearts, pies, and flowers: A connective ethnography of gaming practices. *Games and Culture*, (Special Issue), 5(1), 88-115.
16. Deborah A. Fields, Veena Vasudevan, Yasmin Kafai. 2015. The programmers' collective: Fostering participatory culture by making music videos in a high school Scratch coding workshop. *Special issue of Interactive Learning Environments*, 23(5), 1-21.
17. Deborah A. Fields, Lisa Quirke, Janell Amely, Jason Maughan. 2016. Combining 'big data' and 'thick data' analyses for understanding youth learning trajectory in a summer coding camp. In *Proceedings of the 47th ACM technical Symposium on Computer Science Education* (New York, NY, USA).
18. Deborah Fields, Lisa Quirke, Tori Horton, Jason Maughan, Xavier Velasquez, Janell Amely, and Katarina Pantic. 2016. Working toward equity in a constructionist Scratch camp: lessons learned in applying a studio design model. In *Proceedings of Constructionism* (Bangkok, Thailand).
19. Sara Grimes, Deborah A. Fields. 2015. Children's media making, but not sharing: The potential and limitations of child-specific DIY media websites for a more inclusive media landscape. *Special Issue of Media International Australia*, 154, 112-122.
20. Yasmin Kafai. 2006. Constructionism. In R. K. Sawyer (Ed.) *The Cambridge Handbook of the Learning Sciences*. New York, NY: Cambridge University Press, 35-46.
21. Yasmin B. Kafai, Deborah A. Fields, Kristen Searle. 2014. Electronic textiles as disruptive designs in schools: Supporting and challenging maker activities for learning. *Harvard Educational Review*, 84(4), 532-556.
22. Yasmin B. Kafai, Quinn Burke. 2014. *Connected Code: Why Children Need to Learn Programming*. MIT Press.
23. Jean Lave, Etienne Wenger. 1991. *Situated learning: Legitimate peripheral participation*. New York: Cambridge University Press.
24. John H. Maloney, Kayle Peppler, Yasmin Kafai, Mitchel Resnick, M., and Natalie Rusk. 2008. Programming by choice: Urban youth learning programming with Scratch. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA), 367-371.

25. John H. Maloney, Mitchel Resnick, Natalie Rusk, Bryan Silverman, Evelyn Eastmond. 2010. The Scratch programming language and environment. *Trans Comput Educ*, 10, 4, 16:1–16:15.
26. Andres Monroy-Hernández. 2012. Designing for remixing: Supporting an online community of amateur creators (Doctoral dissertation, Massachusetts Institute of Technology).
27. Andres Monroy-Hernández, Mako Hill, Jazmin Gonzalez-Rivero, D. Boyd. 2011. Computers can't give credit. In the Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems. New York: ACM Press, 3421–3430.
28. Laurie Murphy, Gary Lewandowski, Renee McCauley, Beth Simon, Linda Thomas, Carol Zander. 2008. Debugging: The good, the bad, and the quirky – a qualitative analysis of novices' Strategies. In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (New York, NY, USA), 163–167.
29. Seymour Papert. 1991. Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays*. Norwood, New Jersey: Ablex Publishing, 1-12.
30. Alexander Repenning, David C. Webb, Kyu H. Koh, Hilary Nickerson, Susa B. Miller, Cathrine Brand. Her Many Horses, I., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K. and Repenning, N. 2015. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Trans. Comput. Educ.* 15, 2, Article 11 (April 2015), 31 pages. doi: <http://dx.doi.org/10.1145/2700517>
31. Bruce Sherin. 2013. A computational study of commonsense science: An exploration in the automated analysis of clinical interview data. *Journal of the Learning Sciences* 22, 4, 600-638.
32. Alan H. Shoenfeld, John P. Smith, Abraham Arcavi. 1993. Learning: The microgenetic analysis of one student's evolving understanding of a complex subject matter domain. In *Advances in Instructional Psychology*, 4, 55–175.
33. Linda Werner, Charlie McDowell, Jill Denner. 2013. A first step in learning analytics: pre-processing low-level Alice logging data of middle school students. *Journal of Educational Data Mining*, 5, 2, (2013), 11-37.
34. U. Wolz, B. Taylor, C. Hallberg. 2010. Scrape: A Tool for Visualizing the Code of Scratch Programs. Presented at the ACM SIGCSE (Dallas, Texas, 2010).
35. Marcelo Worsley. 2012, October. Multimodal learning analytics: enabling the future of learning through multimodal data analysis and interfaces. In Proceedings of the 14th ACM international conference on Multimodal interaction, 353-356. ACM.
36. Marcelo Worsley, Paulo Blikstein. 2015. (March). Leveraging multimodal learning analytics to differentiate student learning strategies. In Proceedings of the Fifth International Conference on Learning Analytics and Knowledge (pp. 360-367). ACM.
37. Seungwon Yang, Carlotta Domeniconi, Matt Revelle, Mack Sweeney, Ben U. Gelman, Chris Beckley, Aditya Johri. 2015. Uncovering Trajectories of Informal Learning in Large Online Communities of Creators. Paper presented at L@S 2015 / Learning (Vancouver, BC, Canada, March 14-18, 2015).