Expressions on the Nature and Significance of Programming and Play

Titus Barik
North Carolina State University
Raleigh, North Carolina, USA

Abstract—**Play is all around us, an essential and innate phenomenon that serves as an important mediator in creativity, interest, learning, and drive. Though play is thought to be universal, the way in which it materializes is situationally-dependent and not well-understood, particularly in software engineering. To understand how programmers express the concept of play, we conducted a qualitative study on the online social news website, Hacker News—a venue for software practitioners. From Hacker News, we qualitatively analyzed nearly 1,000 user-submitted comments containing the terms "programming" and "play." The contribution of this work is a contemporary synthesis of how software practitioners interpret programming and play in experiential terms. Our findings suggest how programming and play can be understood through rich metaphors, among them, play as: art, playgrounds, spontaneity, and tinkering. Hacker News authors reflect about childhood experiences as a catalyst for learning programming, and contrast play against work.**

I. Introduction

If light were dark and dark were light
The moon a black hole in the blaze of night
A raven's wing as bright as tin
Then you, my love, would be darker than sin.

Jim Stein
The Invocation

```
if ((light eq dark) && (dark eq light)
   && ($blaze_of_night{moon} == black_hole)
   && $ravens_wing{bright} == $tin{bright})){
 my $love = $you = $sin{darkness} + 1;
};
```

Angie Winterbottom
Best of Show in Perl Poetry Contest [1]

Play is all around us. The activity of play is thought to be an essential and innate phenomenon, found in children, adults, and even animals [2], [3]. Children, for example, need not be taught how to play, yet are able to do so naturally [4], suggesting that play may serve an important evolutionary function, a sort of behavioral phenotype [5], [6]. And animals, even without the capacity for language, are able to signal to others, "this is play," through metacommunicative means [7].

Of course, humans have an expanded range of play activities over animals, most evident in the hallmark of childhood, where opportunities for play are abundant [8]. As Pellegrini notes, childhood play "enables children to learn the skills necessary for successful functioning in adulthood" [9], and its significance has been linked to creativity and imagination [10], self-esteem [11], and cognitive development [12]. Notably, even small children are able to distinguish activities between those that are play and those that are work [13].

Despite the prevalence of play and its apparent innateness, the study of play and its role is curiously underrepresented in adulthood [3]. One possibility to explain this underrepresentation is that play is primarily a childhood function; the necessity for play, that is, the "play impulse" [14], diminishes as we enter our professional careers. Essentially, we do not study play because play does not exist. Yet another possibility is that play continues in adulthood, but materializes in unexpected and ambiguous ways, perhaps because play is perceived as socially unacceptable, frivolous, or irresponsible [15].[1]

Indeed, the nature of play is often ambiguous, in part, because play is both context-dependent and multi-faceted—embodying a diversity of "metaphoric playfulness" that "takes on multiple forms" [15]. As Bekoff argues, "the enormity of the problem of definition is evidenced by the fact that the word 'play' itself often is used in definitions of play" [5]. Consequently, researchers have suggested that, rather than attempting to provide a comprehensive definition of play, play may be most readily understood in experiential terms [5], [16], [17].

In this paper, our interest as software engineering researchers directs our attention to a specific instance of adulthood play in professional contexts: programming and play, and to understand how the activities of programming and play are expressed by software practitioners as they comment on the nature and significance of play as a substructure of their broader daily experiences.

To that end, we conducted a qualitative study in which we obtained nearly one thousand authored comments from Hacker News[2]—a social website for software practitioners focusing on computer science, software development, and entrepreneurship—pertaining to the topics of "programming" and "play." We framed these comments as *small stories* [18] and analyzed them through descriptive and narrative coding [19]. Finally, we employed metaphors as an analytic technique to characterize and present the results of our qualitative study.

The contribution of this work is that it offers researchers a contemporary synthesis on how software practitioners interpret

[1]Of course, still another possibility, as Bekoff notes, is that "if it was as much fun to study play as it is to engage in it, more would be known about the activity" [5].

[2]https://news.ycombinator.com

978-1-5386-0443-4/17/$31.00 ©2017 IEEE

and negotiate the intersection of programming and play as they work, play, and live the incredible human experience.

II. Methodology

We felt that a paper on play, especially in light of the special topic of the conference this year at VL/HCC ("programming and play"), warranted a playful yet pragmatic approach to analytic inquiry on the subject.

Research context. We used Hacker News, a social website for software practitioners, to conduct our investigation. As a community, Hacker News contains over 1.5 million user-submitted comments on a variety of cultural and technical topics of significance to the hacker community (for example, "Steve Jobs has passed away," "Announcing the first SHA-1 collision," and "Be Kind," to convey a sense of the diversity of topics). Wu and colleagues, through a survey with software developers who use GitHub, found that Hacker News serves as an important venue for software developers to exchange ideas as part of a broader cultural ecosystem [20].

Barik and colleagues conducted a formative study using Hacker News to demonstrate that investigations within the online community can yield insights into qualitative research topics, with results comparable to and sometimes surpassing traditional qualitative research techniques, such as interviews or surveys [21]. We adopt their approach in conducting our inquiry of programming and play.

Data collection and bean counting. We used the Algolia[3] search engine API, which indexes all of Hacker News, to retrieve JSON-formatted comments containing both the terms "programming" and "play." The results are sorted by relevance according to the internal Algolia search algorithm.[4] A limitation of Algolia is that it returns a maximum allowable 1,000 relevant results out of the approximately 8,300 possible comments available in the full data.

Retrieved comments from Hacker News spanned the time period from December 18, 2007 through March 28, 2017. Authored comments were extracted across 904 topics. Comments were authored by 818 distinct user handles. Hacker News incorporates a points-based reputation system which allows certain users to upvote or downvote comments. In our collected data, points for comments ranged from 0 to 265 ($u = 13, sd = 17$), with negative points being indicative of "troll" or otherwise unacceptable comments by community standards.

The number of words in a comment averaged 262 words ($sd = 287$), roughly equivalent to the length of abstracts in typical academic papers.

Data cleaning. We conducted an initial pass over the data in which we marked 199 comments in the data as false positives. Representative examples of false positives include: URLs, for example, in play.google.com; the term playlist, when referring to music; the term "plays well with," regarding the compatibility of two software libraries; "plays a role," a "hand to play," "at play," "to play devil's advocate," and similar derivatives, when used colloquially; "playbook," as a reference to Ansible playbooks; "playback," in reference to video; and so on.

[3]https://hn.algolia.com/

[4]https://www.algolia.com/doc/guides/relevance/ranking/

Qualitative analysis. We ported the Algolia JSON results to a format compatible with the ATLAS.ti data analysis software, and used the software to qualitatively code the data.[5] We conducted coding over multiple iterations. In the first cycle, we used descriptive coding, and assigned short codes and labels to capture and summarize the salience of the comments [19] by framing them as small stories [18]. In the second iteration, we conducted a systematic metaphor analysis to organize the comments into clearly structured patterns, and it is through the identification of these metaphors by which we describe the results [22], [23]. Together, the first and second iterations provide both the metaphor for the classified comments and the necessary contextual details to describe the nature of the respective metaphor.

Analysis rationale. We briefly provide justification for our choice of two analytic machineries: 1) comments as small stories, and 2) metaphor analysis. The first, small stories research, impacts the way in which we choose to interpret the comments. The second, metaphor analysis, influences the organization and presentation of our qualitative findings.

Small stories analysis is both small in the literary sense, in that it methodologically applies to small vignettes or "messy" episodes [24], and as an analysis paradigm that contrasts with "big stories," or traditional, prototypical narratives—such as novels or formal autobiographies and participant interviews. As Bamberg and Georgakopoulou argue, small stories analysis takes the perspective that there is "worth" in the everyday stories and experiences, even when such incidents may be "seemingly uninteresting tidbits," and even when they fail formal criteria of narrative, such as temporal ordering of events [18]. For us, small stories research is less a prescription for how to conduct analysis, but rather, a perspective that places value on small stories. We argue that the shared expressions from authors on Hacker News, though casual, offers such a contribution to the research community.

The second analytic machinery is that of metaphor analysis, which can be conducted in a variety of ways, among them, as a rhetorical instrument, to describe the research process itself, and to describe the results of qualitative research [22]. We have opted to use metaphor analysis as a means to describe the results of our research, for two reasons. First, metaphor analysis is particularly appropriate when there is a need to organize and abstract multiple, potentially divergent expressions, and when no expression is particularly privileged or more important than any other [22]. Second, metaphor analysis can aid in the reliability of the finding: even if the credibility of the narrative itself is suspect, Moser argues that the tacit metaphor present within the narrative remains a reliable and authentic belief of the explicit expression [25].

Verifiability. To support verification of our findings, we have placed both the original Hacker News dataset, in JSON

[5]http://atlasti.com/

TABLE I
THE METAPHORS OF PLAY

Play as...	**Section**	**Summary**
artistry	Section III-A	Comparisons to programming as related to art or playing music, acquisition of mastery, and performance.
catalyst	Section III-B	Play as a catalyst activity which leads to programming, typically expressions and nostalgic experiences from childhood.
fun	Section III-C	Play as enjoyment or fun and the characteristics to elicit these positive affects.
playgrounds	Section III-D	Metaphors and expressions of programming environments as virtual playgrounds; the sandboxes and toys of programming.
spontaneity	Section III-E	Programming and play as a spontaneous and undirected activity.
tinkering	Section III-F	Expressions of programming and play as dabbling or casual tinkering.
anti-work	Section III-G	Tensions in negotiating play and work as dichotomous experiences.

format, as well and our ATLAS.ti analysis file, on our research site.[6] We encourage researchers to use this data to present complementary interpretations on the nature and significance of programming and play, or use it as a basis for triangulation with other forms of inquiry.

III. THE NATURE AND SIGNIFICANCE OF PROGRAMMING AND PLAY

In this section, we present expressions on the nature and significance of programming and play, organized through metaphor. The complete list of metaphors of play, situated through programming, is found in Table I.[7]

A. *Play as artistry*

"Everyone programs differently," says $\text{HN}_{2450199}$. "Every artist paints differently, plays differently, or sculpts differently. Programming is an art. Instead of brushes, we have abstract data types, instead of paint we have user interfaces. If we use the right strokes in the right places, we end up with a masterpiece" ($\text{HN}_{2450199}$). Indeed, "programming and music, especially Jazz improvisation, seem to involve very similar kinds of problem solving," says $\text{HN}_{4180791}$. They add, "both of them require thinking about a problem at many different layers of abstraction simultaneously, and both are fundamentally about recognizing and manipulating abstract patterns. Both are passions that, from the outside, might seem like they require a lot more work than they give back in payoff. But if you enjoy the process, it doesn't feel like work at all."

These expressions by $\text{HN}_{2450199}$ and $\text{HN}_{4180791}$ capture the core experience of the metaphor of play as artistry: how practitioners interpret the act of programming as an act of artistry, passion, and performance. $\text{HN}_{4622301}$ elaborates, "you know how rare it is to have a passion and talent for something, and at the same time have that something be valuable and useful. Think of all the brilliant and passionate people that are into disciplines where only a few thousand people in the world get a decent income. Think of all dancers, screenwriters, directors, painters, [and] poets."

A noteworthy undercurrent we found within play as artistry is that expressions of programming are often situated about mastery within the craft. "While learning piano, no one wants to sit in a room playing scales over and over," says $\text{HN}_{3662494}$. They continue, "they want to play Clair de Lune. Programming is similar in that it can be an art but it is first and foremost a skill. You need to struggle with boring parts until you figure out that you can actually make very cool things. But first you need to learn about loops and variables." And $\text{HN}_{9290112}$ similarly makes the case through Jimi Hendrix, in reference to music and skill and watching others perform the activity instead being an active participant: "my take on programming has become that watching a programming video is like watching Jimi Hendrix play the guitar. A beginner learns almost nothing applicable because they don't have the mechanical technique as a basis for informing their seeing. A virtuoso will see new techniques and ways they could do them better." $\text{HN}_{7465770}$ adds, "it requires some extended time of immersion to really make significant progress."

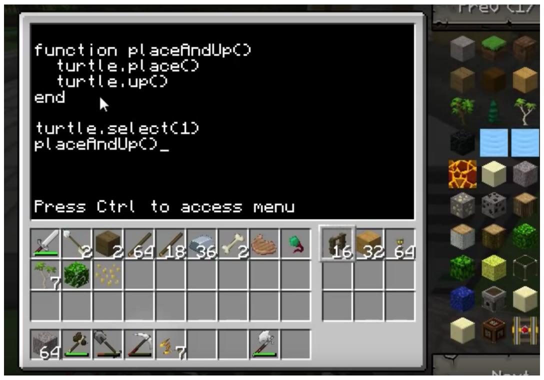
If code is art, then GitHub is the exhibition halls through which the art is displayed. For some authors like $\text{HN}_{9290112}$, their mastery translates to performance for others, either as publicly-accessible repositories for employment or more informally to just "show off" for friends ($\text{HN}_{3310273}$, $\text{HN}_{4133140}$). However, these expressions are not universally shared. $\text{HN}_{7993835}$, for example, says, "not everyone is putting all their achievements out for the world to see." As $\text{HN}_{2764004}$ explains, "a big problem with 'GitHub-as-a-Resume' for the experienced developer is that it destroys the idea of side-projects as play. I can no longer just fool around with something for the sake of fooling around with it."

B. *Play as catalyst*

"I started off just playing games, but pretty soon I wanted to write my own games," says $\text{HN}_{8983884}$. They continue, "thanks to all the resources already available, I picked up BASIC quite easily." Many authors contributed experiences of how play, particularly through computer and video games, led to programming and "sparked their interest" ($\text{HN}_{6293058}$) or "pushed [them] to take a programming course" ($\text{HN}_{6304166}$). From "playing around with gorilla in `QBASIC` to pretending to know C and trying to make Pac-Man" ($\text{HN}_{2142519}$), "inserting four letter words into Tank Battle using a hex editor" ($\text{HN}_{8685495}$), "making a shoddy Zelda clone" ($\text{HN}_{6741327}$), or otherwise "reverse engineering games" ($\text{HN}_{6924501}$), these authors describe play as a catalyst for intermingling programming and play

[6]http://go.barik.net/hnplay

[7]For traceability, quotations are cited as $\text{HN}_{:\text{id}}$, where `:id` is the unique identifier of the comment. These comments can be accessed as a JSON document at `http://hn.algolia.com/api/v1/items/:id`. Despite the additional cognitive processing cost, we also use the gender-inclusive "they" as a singular pronoun when referring to author expressions [26].



```
function placeAndUp()
  turtle.place()
  turtle.up()
end

turtle.select(1)
placeAndUp()_

Press Ctrl to access menu
```

Fig. 1. ComputerCraft is a mod created for Minecraft that adds LOGO-like capabilities, such as *turtles*, to the game. Turtles are programmed in the Lua programming language.

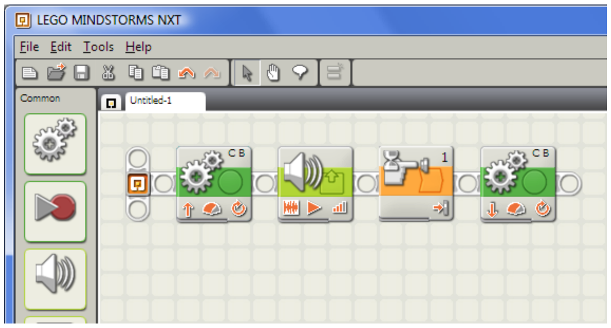


Fig. 2. HN6218320 describes how intuitive, building-block coding interfaces made programming an enjoyable experience. Shown in this figure is the Lego Mindstorms NXT interface for programming Lego robots.

activities, sometimes leading to professional programming. As HN3078213 experienced when they first discovered they could break into the game and modify it: "the excitment of that moment stuck with me and was the enabler of the amazing life I've had since." Similarly, HN4891359 felt "lucky enough to be born at the right in the right environment." They describe their nostalgic experience as "tak[ing] things we didn't like and turning them into things we did like" (HN4891359).

In some cases, we found experiences where playing games were eventually replaced with programming. As HN3991129 tells us, "for some reason I pretty much stopped playing computer games when I started programming. I think Quake II and Diablo II are the last games I played seriously. After that I got into indy game development as a hobby and I started making games instead of playing them."

As advice to providing a catalyst to others, HN97114444 says this: "speaking from my own experience, give him not a programming language, but either a game that includes a programming language, or a tool for making games that includes programming." We heard several expressions of how "in-game programming" (HN5151905), or *mods* (Figure 1), led from play to programming for play. As HN5151905 describes, "modders have had an endless field day with this. It's now a game that contains other games." HN8318882 adds, "as this young, enthusiastic population of users grows up, they will have a lot of buying power and interest in things like the customizations and programming aspects of the game." HN3501677 reflects, "I think we underestimate the imagination of today's kids."

C. Play as fun

Within this metaphor emerges a sense of how practitioners experience positive affect, such as fun and enjoyment, through the activity of programming and play.

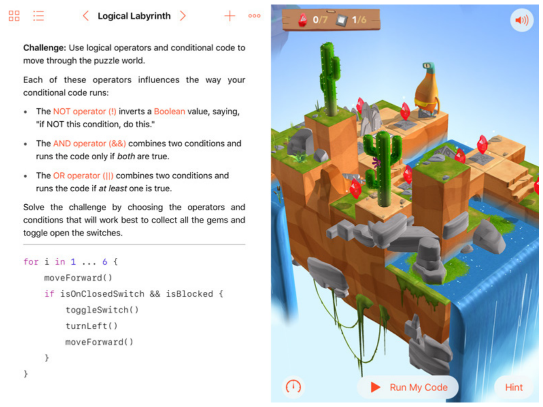
A recurring theme in many author stories were related to simplicity as a driver of these positive feelings, especially when this simplicity arises as unexpected surprises. For instance, HN4670326 observes, "I never thought I'd be playing with APIs, but it turns out programming is easier than I *thought it would be*. Ruby is a lot more fun than the C++ I learned in engineering school. Hours after everyone else in the house went to sleep last night, I was here in the office with ten tabs open in Chrome." And HN3285341 says, "I learned to program with ActionScript. I studied art and had no background in programming, but I found it fun to play with Flash and ActionScript. For example, getting a webcam involved is a couple of lines of code." Likewise, HN6218320 orients his introduction to programming in terms of expressions of enjoyment: "I got into programming by playing with Lego Mindstorms when I was a kid (Figure 2). The building block coding interface was really fun and intuitive. 'Oh no,' a purist would sneer, 'you can't code like that, it's horrible. Here, read this tome on C and microcontroller programming before you start.' No thanks. I got to enjoy the end result immediately (making a fun robot), and then from there I could look further into making something more complicated, which ignited my programming journey."

Supporting the notion that play is in some way an innate phenomenon, HN6697246 says that "as a kid I, perhaps naively, didn't think playing around with computers was even a real thing you could do as a job. I assumed when I grew up I would have to get a real job, doing proper engineering, or carpentry, or accounting, or something." This sort of playing around is perhaps "open-ended and entirely informal" (HN6335181) and it is unsurprising that authors like HN4612137 "love playing around with widgets and technology just to see what happens."

Still others authors describe the nature of programming and play through terms such as "empowerment" and "excitement" (HN3662188). HN3662188 tells us: "when I was eight, we got a Commodore VIC-20. It plugged into the TV and booted into BASIC. Just running `PRINT` statements and simple loops was unbelievably cool. Getting to play with Logo on the Apple][e was awesome, too. Shapes and angles and horribly flickering animations were exciting." In the end, HN6133077 suggests that it is not play itself that garners this excitement, but rather, that "much of the 'pleasure of programming' is attributable to the *possibility* of play with such a system."

D. Play as playgrounds

When authors elaborate on the metaphor of playgrounds, they often describe the environments and the enabling of



Challenge: Use logical operators and conditional code to move through the puzzle world.

Each of these operators influences the way your conditional code runs:

- The NOT operator (!) inverts a Boolean value, saying, "if NOT this condition, do this."
- The AND operator (&&) combines two conditions and runs the code only if *both* are true.
- The OR operator (||) combines two conditions and runs the code if *at least* one is true.

Solve the challenge by choosing the operators and conditions that will work best to collect all the gems and toggle open the switches.

```
for i in 1 ... 6 {
    moveForward()
    if isOnClosedSwitch && isBlocked {
        toggleSwitch()
        turnLeft()
        moveForward()
    }
}
```

Fig. 3. The Apple Swift playground both captures the nature of physical playgrounds and transforms them into a virtual programming environment.

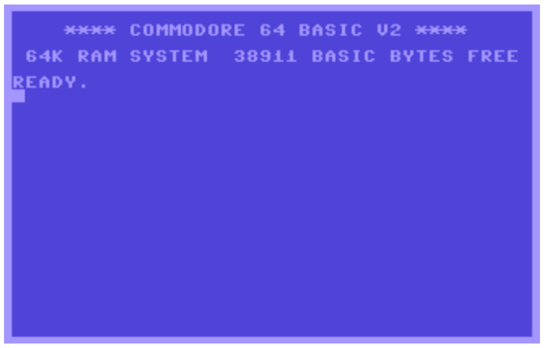
programming and play within them through the vocabulary of physical playspaces [27]. For example, authors describe the environments as "fun but messy" (HN$_{3128849}$), "generally safe" (HN$_{8922465}$), and as "sandboxes" (HN$_{4707117}$, and Apple Playgrounds in Figure 3), environments where one can "muck it up as much as you like" (HN$_{5607939}$). Simultaneously, authors recognize that play need not be "useful for productive programming" (HN$_{6710130}$) and that the environment should "get out the way [of play]" (HN$_{3848668}$, HN$_{3837895}$).

Like many playgrounds—from the traditional ones with slides and seesaws to the "junk" playgrounds of old tires and packing crates [28]—the landscape of programming and play is littered with expressions of toys and playthings, equally divergent. Within these playgrounds, "programming computers [is] like coloring with crayons and playing with Duplo blocks" (HN$_{7596048}$), using environments like PicoList where they can "tackle learning everything from the ground up" (HN$_{9959701}$), or "cod[e] up the occasional fun toy or two until [they] got bored and ditched it" (HN$_{7509951}$).

HN$_{4335734}$ offers a description to contextualize the nature of play as one in which there are no rules (at least in perception). HN$_{4335734}$ says, "I thought games were amazing because I could manipulate these little worlds, and I could do anything imaginable within their rulesets. But, seeing 'Hello!' scroll forever and ever made me realize that, with this coding thing, *there were no rules*. The fact that I could make this computer do whatever I wanted, if only I could speak its language, was irresistible."

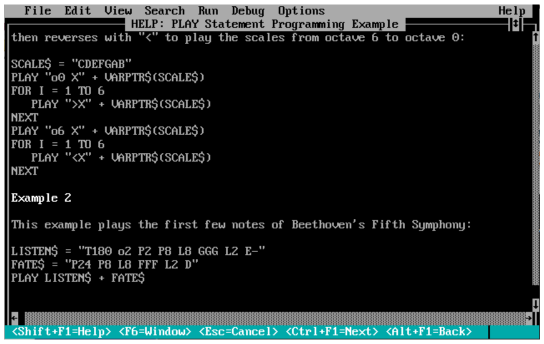
E. Play as spontaneity

Authors express this metaphor of play as one of "stumbling into programming" (HN$_{6304166}$, HN$_{5421795}$), described by HN$_{8029370}$ as one in which the boundaries of play to programming and play is "thin and easily crossed." A key idea within this metaphor is that programming and play is in some sense experienced when it is always-on, unintentional, or a default mode of the system—as HN$_{5421795}$ says, "people used to have to choose not to program when they turned their computers on, in order to play games." The author continues their recollection,



```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
```

Fig. 4. In the Commodore 64, the system boots up in a programming environment by default. As HN$_{5421795}$ remarks, people had to explicitly choose not to program when they turned their computers on.


File Edit View Search Run Debug Options Help
HELP: PLAY Statement Programming Example
<Shift+F1=Help> <F6=Window> <Esc=Cancel> <Ctrl+F1=Next> <Alt+F1=Back>

```
then reverses with "<" to play the scales from octave 6 to octave 0:

SCALE$ = "CDEFGAB"
PLAY "o0 X" + VARPTR$(SCALE$)
FOR I = 1 TO 6
   PLAY ">X" + VARPTR$(SCALE$)
NEXT
PLAY "o6 X" + VARPTR$(SCALE$)
FOR I = 1 TO 6
   PLAY "<X" + VARPTR$(SCALE$)
NEXT

Example 2

This example plays the first few notes of Beethoven's Fifth Symphony:

LISTEN$ = "T180 o2 P2 P8 L8 GGG L2 E-"
FATE$ = "P24 P8 L8 FFF L2 D"
PLAY LISTEN$ + FATE$
```

Fig. 5. Interpreters like Microsoft `QBASIC` made accessible I/O features of the underlying hardware, making it simple for tinkerers to display graphics and play audio. Shown here is an example of the `PLAY` statement, through which the programmer can play audio through the PC speaker by providing a string representation of musical notes.

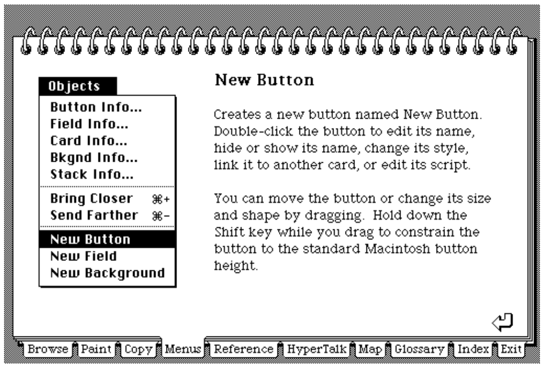
noting that systems like the Amstrad Microcomputer and the C64 booted the machine into programming by default (Figure 4). HN$_{5421795}$ continues, "the very next thing that appeared was the text cursor, the assumption being that you would now begin to type code. You could choose not to program by putting a game cassette in the drive, holding `Shift+Enter`, and pressing play. *But by default, you were programming*."

HN$_{8029370}$ echoes a similar sentiment as they played within MUDs—text-based, multi-player, real-time, virtual worlds—during their childhood. HN$_{8029370}$ says, "playing the game led almost inevitably to understanding the mechanics and becoming a coder." However, HN$_{8029370}$ continues, "I feel sorry for today's generation of gamers, since there is a much larger wall between playing and creating these days."

Other authors appear to express similar laments with respect to the lack of spontaneity in modern programming environments. HN$_{7278026}$ describes today's experiences and attempts to play with programming as "frustrating" and "requiring an amount of boilerplate" that was "mindboggling." HN$_{7278026}$ observes, "it's a far cry from `PRINT "HELLO WORLD"`."

F. Play as tinkering

Play as tinkering is "never anything too serious" (HN$_{2415083}$). Understanding during this metaphor is considered to be at the



New Button

Creates a new button named New Button. Double-click the button to edit its name, hide or show its name, change its style, link it to another card, or edit its script.

You can move the button or change its size and shape by dragging. Hold down the Shift key while you drag to constrain the button to the standard Macintosh button height.

Fig. 6. Hypercard, a programming tool developed by Apple for what we today would call end-user programmers. In the metaphor of anti-work, authors highlighted tools such as Hypercard as being in opposition to "real programming."

surface (HN$_{4924837}$, HN$_{2560474}$) and only to "understand the basics" (HN$_{2457334}$). It is, as HN$_{5786832}$ states, "people who are mostly messing around playfully." As HN$_{4327009}$ notes, "they read a bunch of blogs maybe, but they don't think deeply about what the underlying systems are doing."

HN$_{6979112}$ talks about their childhood experiences of watching their father tinker, or dabble with programming: "I was lucky enough to learn the BASICs from my father. He, being a mathematician and having dabbled with programming at university, would play around with his own things on the computer in the evening, and I was rather interested in watching him create simple graphical things on the screen" (Figure 5).

Like HN$_{6979112}$'s father, tinkering is something that a practitioner does "on the side" (HN$_{4848921}$, HN$_{2764004}$) as tangential to their primary development activities. Though the duration varies, many authors describe tinkering as a time-bounded activity, as something they do "a bit" (HN$_{5687821}$, HN$_{6043994}$), for "two minutes" (HN$_{4856885}$) or even "for a few months" (HN$_{3071883}$).

We found that not all authors considered tinkering to be a positive metaphor, contrasting the shallow understanding that tinkering entails against "real programmers," or "hackers." HN$_{2560474}$ says, "a little bit of knowledge is maybe worse than not knowing at all. I'd rather have someone just say 'I don't know' than try to make a decision based on a few weeks playing with Python tutorials. Dabbling a bit in a couple of languages is about as far from programming, or understanding software development, as playing with toy cars is from Formula 1 racing or designing engines." HN$_{2560474}$ adds, "I tinker, I play, I build, but I have mostly surface knowledge. 'Hacker' is a term that should be reserved for real programmers who actually know what they are doing."

G. Play as anti-work

Authors expressed strong beliefs towards the dichotomy of play and work, and we encountered strong cynicism in the expressions for this metaphor regarding both and work activities. As HN$_{6445685}$ says, "a big distinction for me is 'programming as work' and 'programming as play'." They continue, "I can spend eight hours at work and be absolutely drained when I get home, but still leap at the chance to say, take a bite out of my functional programming course" (HN$_{6445685}$). HN$_{6468915}$ echoes the sentiment: "there's a reason it is called work and not play. If work was always enjoyable, then they wouldn't need to pay you to do it."

Other authors contrasted programming for work as "real programming" (HN$_{3610936}$, HN$_{7761708}$), suggesting that non-work programming is in some sense "pretend programming" (HN$_{7761708}$ offers the example of Hypercard as pretend programming, Figure 6). For example, HN$_{8396063}$ comments that "pretend" programming applications are essentially "programming for the masses" and that they "inadvertently throw up road blocks to experienced programmers" (HN$_{8396063}$). "Real programming," as HN$_{3309984}$ suggests, is the difference between deploying a *product* versus deploying code. HN$_{3309984}$ argues: "the joy of being a professional, well paid developer is in creating the product, not the code. The joy of programming for programming's sake is something you do in your own time."

The concept of this "spare time" pressure between work and play is shared by other authors as well. HN$_{9000211}$ contrasts differences in programming between their professional adult career and that of childhood: "for a child it is easier—he has the time available and that desert of despair mentioned can be actually fun. Because to a child, making the computer do stuff can feel like magic, whereas to an adult it feels like a chore to get somewhere, a chore that eats all of the available time." HN$_{7426608}$ concedes that programming for work cancels the ability to engage in programming as play: "one needs hobbies that are not programming. That's no different really than a chef coming home making himself a sandwich."

But not all authors expressed the same cynicism. HN$_{9120744}$ talks about the importance of programming and play, and its application to work and creativity: "as much as I love programming sometimes it's play that's most important. You need to gather *fresh ideas* from the world around you. It can be isolating to be stuck in the rut of progress and innovation that is so endemic to our culture."

IV. Limitations

Our inquiry into the expressions of programming and play were conducted through an analysis of a single source of practitioner experiences, Hacker News. Consequently, our findings are biased towards the types of participants who self-select to participate within this community. A second bias is introduced as a result of arbitrarily bootstrapping our search through the use of the narrow terms "programming" and "play." In doing so, we have lost potential expressions of play in which the author did not explicitly use both terms; for example, they may have used "coding" rather than "programming." For instance, although the poetic epigraph in our introduction highlights a performatory mode of programming and play, such poetry contests did not appear in our stories of this metaphor.

There are several potential effects of using data from online communities in general, specifically with regard to credibility and authenticity of presented experiences. For example, previous research has found that individuals sometimes "identity shift" in computer-mediated environments, in which public self-presentations differs from private self-presentation, either from denial or accountability [29], or for impression management, in which an individual wishes to present themselves to others in socially desirable ways [30]. However, other studies have suggested that in online communities, some individuals experience a form of "online disinhibition," in which they are more likely to self-disclose more frequently or intensely than they would in person, in part due to perceived anonymity and invisibility [31]. In short, there is likely some sort of perturbation of the experiences we've presented in this paper, but we don't know exactly what that is and to what extent such effects influence our findings.

Another effect from online communities may arise as a result of the moderation and points-system used within Hacker News to rank and display comments, in which individuals in the community internalize their true opinions and instead converge to a form of *groupthink*. Fearing reprisal from other members of the community, individuals may be compelled to only share experiences that they believe would be positively scored by their peers [32]. A quick search for the phrase "unpopular opinion" on Algolia partially validates that some members are aware of groupthink, and explicitly preface their comments with an indication that it may be unpopular to the community, likely to mitigate reprisal from non-conforming expressions.

Finally, we acknowledge that qualitative research, however rigorously conducted, involves not only the qualitative data under investigation but also a level of subjectivity and interpretation on the part of the researcher as they frame and synthesize the results of their inquiry. In particular, though many authors express notions of play, authors whose thoughts are better articulated tend to be given greater representation in the results. Thus, we emphasize that our own findings should be examined as only one of many possible presentations of the nature of programming and play.

V. Related Work

Our work is positioned within the interdisciplinary field of software studies, which aims to understand software systems through the ways in which it shapes society and culture. That is, it is not the software system itself that is the central artifact of investigation, but rather, the emphasis is placed on the human activities surrounding the use of software. In other words, software studies as a means of inquiry asks us to devise and incorporate new ways of thinking about software and the role it plays in our lives [33].

For instance, a series of essays collected by Fuller explore how computer interfaces influence our everyday lives and reshape how we interact with the world [33]. Kitchin and Dodge examine software from a spatial perspective, examining how the nature of space is dependent on the product of code. They discuss how software, for example, automated check-in kiosks at airports, transforms the design of the physical layout of the airport [34]. Cox and McClean emphasize the aesthetic and political implications of code, framing code through different sociological forms: reading code as a cultural artifact, similar to art or film, interpreting code as expressions of speech, and exploring how human speech is translated in a form that computers can interpret [35].

As an idiographic inquiry, Montford and colleagues undertake a close study on a single-line of code: `10 PRINT` and its variations, and use it as a lens to trace historically how professionals and hobbyists read and write code [36]. Our own work also applies a singular lens to understand a particular social phenomenon: programming and play, as expressed by authors on Hacker News.

Finally, Bergström and Blackwell inquire into the diverse ways in which programming is done, presented as practices of programming [37]. They use analytic methods to understand, through accounts of embedded and lived experience, the different accounts of programming practices. In a similar spirit, we aim to understand the space of metaphors through which practitioners express programming and play.

VI. Discussion

In this section, we discuss three aspects of our narrative inquiry of Hacker News. First, we discuss the expressions of *sparse metaphors*, or metaphors that occur within our analysis yet provide insufficient depth to support full inquiry. Second, we retrospect on our decision as researchers to use as Hacker News source of inquiry. Third and finally, our analysis of expressions by Hacker News authors suggests that expressions of programming and play predominantly originate from childhood, and we briefly discuss the implications of this.

Sparse metaphors for programming and play. During our analysis, we encountered a number of metaphors we have termed *sparse metaphors*. Although these metaphors provide additional expressions on programming and play, we found an insufficient number of authors who shared their experiences. Thus, we want to emphasize that the *availability* of the expressions we found and characterized should not be conflated with *importance*. Rather, our work attempts to convey the space of expressions regarding the nature and significance of programming and play as found in Hacker News.

For example, we identified the metaphor of "play as artistry," but along the way, also identified "play as chess" as a sparse metaphor. Although we did not find sufficient context to promote this metaphor to a first-class citizen, it's easy to see envision ways in which programming and play are at times like playing a puzzling game of chess. Other metaphors that we were unable to promote are "play as love," "play as imagination," and "play as construction"; instead, these metaphors "fill-in" the context for metaphors such as "play as catalyst" (Section III-B), "play as playgrounds" (Section III-D), and "play as anti-work" (Section III-G).

Our identification of sparse metaphors brings to attention other contemplations that embody programming and play, yet do not appear within our analysis of Hacker News. As one example, within the *live coding* movement, composers seek new forms of expression by coding music or coding visuals for projection on the fly using laptops [38], [39], [40]. McLean reflects on his experiences of art and programming to produce music; he argues, "a musical score is a kind of source code, and a musical performance is a kind of running program. When you play from a musical score or run a program you are bringing instructions to life" [41]. *Hackathons* are brief, intensive events through which people who are not normally collocated converge to write code together—outside of their day-to-day routine—to explore, learn, socialize, and form stronger connections with others in the organization [42]. And socialization in physical space—both good-natured and unpleasant—sometimes finds its way as remnants within source code repositories—as jokes, puns, and "gallows humor" found in comments within the code [43], [44]. Finally, toolsmiths have applied *gamification*—the application of game-design elements to non-game environments—to make programming experiences more enjoyable for software practitioners [45], [46].

Interpreting Hacker News. Although the primary contribution of this research has been to provide a synthesis of interpretations of programming and play, a secondary (and tacit) aim of this work is to investigate the ways in which researchers can use online, everyday, self-reported and volunteered experiences of practitioners to understand, shape, and frame software engineering. From our perspective, communities like Hacker News are somewhat like a needle in a haystack. But unlike an actual haystack, we also recognized that these digital haystacks could be mined to effectively extract relevant needles.

On one hand, we did drudge our way through our share of false positives, troll comments, and difficult-to-parse comments. In some cases, although we could classify the comment and assign it a particular metaphor, we otherwise were unable to easily incorporate or quote the comment as part of the metaphor details. On the other hand, and much to our surprise, we also encountered on several occasions expressions that were essentially "diamonds in the rough"—vivid, clear, and at times imaginative and even poetic (for example, "everyone programs differently" from $\text{HN}_{2450199}$ and "Jazz improvisation" from $\text{HN}_{4180791}$ in Section III-A, were particularly thought-provoking).

Our results suggest that the seemingly mundane is sometimes anything but; as Georgakopoulou argues, perhaps we need to think big with small stories, and the potential they have to help us understand our research, our society, and even ourselves [47]. We encourage other researchers to investigate Hacker News, and other online communities, as a source of qualitative inquiry.

Programming and play is a privileged position of childhood. Finally, the word play arouses memories, from childhood to adulthood [17]. Unfortunately, our analysis of Hacker News suggests that programming and play appears to predominantly originate as a childhood activity, and occasionally in young adults (most noticeable, perhaps, in "play as catalyst" in Section III-B).

One possible interpretation, and supported by Guo's investigation on adults learning computer programming [48] and Costabile and colleagues' work on end-users as unwitting software developers [49], is that expressions on programming and play arise from childhood because the tools that we design as software engineers to inspire programming are *targeted primarily to children* [50], [51], [52], [53]. If so, an important avenue for research is democratizing and making accessible the experiences of programming and play; perhaps by imagining software development tools or platforms that inspire adults to learn computer programming, adapted to forms of entertainment they enjoy.

Another possible interpretation is that expressions on programming and play often originate and relate to childhood because adult practitioners of software mostly perceive play nostalgically; it is something that they remember and recall doing as a child, but for whatever reason, no longer engage in as adults or only engage in as a luxury. To quote the Irish playwright George Bernard Shaw, "we don't stop playing because we grow old; we grow old because we stop playing." We think that's a shame.

VII. Conclusion

In this paper, we conducted a qualitative study within Hacker News to understand how practitioners expressed the nature and significance of programming and play in experiential terms. Through our inquiry, we discovered that programming and play is expressed and reflected through a multiplicity of metaphors, among them: play as artistry, play as tinkering, play as playgrounds, and play as anti-work. These contemporary metaphors provide a telescope through which we are able to reflect and relate to programming and play within our own lives—as practitioners, as researchers, as hobbyists, as parents and grandparents, and as role models.

As tool designers, the stories shared by authors of Hacker News reveal the need for playful experiences within software development environments—experiences that allow programmers to spontaneously and creatively express their ideas to code, to provide safe playgrounds for experimentation, and to support tinkering as a purposeless, ludic activity. Perhaps most importantly, these stories remind us that despite our busy and often hurried day-to-day lives, it's important for all of us to make time for play.

References

[1] J. Orwant, *Games, Diversions, and Perl Culture: Best of the Perl Journal*. O'Reilly Media, Inc., 2010.

[2] E. Fink, U. Saine, and T. Saine, "The oasis of happiness: Toward an ontology of play," *Yale French Studies*, no. 41, pp. 19–30, 1968.

[3] M. Van Vleet and B. C. Feeney, "Play behavior and playfulness in adulthood," *Social and Personality Psychology Compass*, vol. 9, no. 11, pp. 630–643, Nov. 2015.

[4] J. Huizinga, "Nature and significance of play as a cultural phenomenon," in *The Game Design Reader: A Rules of Play Anthology*. MIT Press, 2006.

[5] M. Bekoff, "Social play behavior," *BioScience*, vol. 34, no. 4, pp. 228–233, Apr. 1984.

[6] G. M. Burghardt, "The evolutionary origins of play revisited: Lessons from turtles," in *Animal Play: Evolutionary, Comparative, and Ecological Perspectives*. Cambridge University Press, 1998, ch. 1, pp. 1–26.

[7] G. Bateson, "A theory of play and fantasy," in *The Game Design Reader: A Rules of Play Anthology*. MIT Press, 2006, pp. 314–328.

[8] M. Van Vleet and B. C. Feeney, "Young at heart: A perspective for advancing research on play in adulthood," *Perspectives on Psychological Science*, vol. 10, no. 5, pp. 639–645, Sep. 2015.

[9] A. D. Pellegrini and P. K. Smith, "The development of play during childhood: Forms and possible functions," *Child and Adolescent Mental Health*, vol. 3, no. 2, pp. 51–57, May 1998.

[10] M. Moore and S. W. Russ, "Follow-up of a pretend play intervention: Effects on play, creativity, and emotional processes in children," *Creativity Research Journal*, vol. 20, no. 4, pp. 427–436, Nov. 2008.

[11] L. K. Bunker, "The role of play and motor skill development in building children's self-confidence and self-esteem," *The Elementary School Journal*, vol. 91, no. 5, pp. 467–471, May 1991.

[12] A. Nicolopoulou, "Play, cognitive development, and the social world: Piaget, Vygotsky, and beyond," *Human Development*, vol. 36, no. 1, pp. 1–23, 1993.

[13] L. A. Wing, "Play is not the work of the child: Young children's perceptions of work and play," *Early Childhood Research Quarterly*, vol. 10, no. 2, pp. 223–247, Jan. 1995.

[14] H. Hein, "Play as an aesthetic concept," *The Journal of Aesthetics and Art Criticism*, vol. 27, no. 1, pp. 67–71, 1968.

[15] B. Sutton-Smith, *The Ambiguity of Play*. Harvard University Press, 2001.

[16] S. Brown and C. Vaughan, *Play: How it Shapes the Brain, Opens the Imagination, and Invigorates the Soul*. Avery, 2010.

[17] A. Sandberg, "Play memories from childhood to adulthood," *Early Child Development and Care*, vol. 167, no. 1, pp. 13–25, Jan. 2001.

[18] M. Bamberg and A. Georgakopoulou, "Small stories as a new perspective in narrative and identity analysis," *Text & Talk*, vol. 28, no. 3, pp. 377–396, Jan. 2008.

[19] J. Saldaña, *The Coding Manual for Qualitative Researchers*. SAGE Publications, 2009.

[20] Y. Wu, J. Kropczynski, P. C. Shih, and J. M. Carroll, "Exploring the ecosystem of software developers on GitHub and other platforms," in *CSCW Companion*, 2014, pp. 265–268.

[21] T. Barik, B. Johnson, and E. Murphy-Hill, "I heart Hacker News: Expanding qualitative research findings by analyzing social news websites," in *ESEC/FSE*, 2015, pp. 882–885.

[22] R. Schmitt, "Systematic metaphor analysis as a method of qualitative research," *The Qualitative Report*, vol. 10, no. 2, pp. 358–394, 1990.

[23] R. C. Smith and E. M. Eisenberg, "Conflict at Disneyland: A root-metaphor analysis," *Communication Monographs*, vol. 54, no. 4, pp. 367–380, Dec. 1987.

[24] A. De Fina and A. Georgakopolou, *The Handbook of Narrative Analysis*. John Wiley & Sons, 2013, vol. 53, no. 9.

[25] K. S. Moser, "Metaphor analysis in psychology—Method, theory, and fields of application," *Forum: Qualitative Social Research*, vol. 1, no. 2, 2000.

[26] A. J. Sanford and R. Filik, ""They" as a gender-unspecified singular pronoun: Eye tracking reveals a processing cost," *The Quarterly Journal of Experimental Psychology*, vol. 60, no. 2, pp. 171–178, Feb. 2007.

[27] J. Veitch, S. Bagley, K. Ball, and J. Salmon, "Where do children usually play? A qualitative study of parents' perceptions of influences on children's active free-play," *Health & Place*, vol. 12, no. 4, pp. 383–393, 2006.

[28] A. M. Susa and J. O. Benedict, "The effects of playground design on pretend play and divergent thinking," *Environment and Behavior*, vol. 26, no. 4, pp. 560–579, Jul. 1994.

[29] A. L. Gonzales and J. T. Hancock, "Identity shift in computer-mediated environments," *Media Psychology*, vol. 11, no. 2, pp. 167–185, Jun. 2008.

[30] D. C. DeAndrea, S. Tom Tong, Y. J. Liang, T. R. Levine, and J. B. Walther, "When do people misrepresent themselves to others? The effects of social desirability, ground truth, and accountability on deceptive self-presentations," *Journal of Communication*, vol. 62, no. 3, pp. 400–417, Jun. 2012.

[31] J. Suler, "The online disinhibition effect," *CyberPsychology & Behavior*, vol. 7, no. 3, pp. 321–326, Jun. 2004.

[32] C. McCauley and Clark, "The nature of social influence in groupthink: Compliance and internalization." *Journal of Personality and Social Psychology*, vol. 57, no. 2, pp. 250–260, 1989.

[33] M. Fuller, *Behind the Blip: Essays on the Culture of Software*. Autonomedia, 2003.

[34] R. Kitchin and M. Dodge, *Code/space: Software and Everyday Life*. MIT Press, 2011.

[35] G. Cox and A. McLean, *Speaking Code: Coding as Aesthetic and Political Expression*. MIT Press, 2013.

[36] N. Montfort, P. Baudoin, J. Bell, I. Bogost, J. Douglass, M. C. Marino, M. Mateas, C. Reas, M. Sample, and N. Vawter, *10 PRINT CHR $(205.5+ RND (1));: GOTO 10*. MIT Press, 2012.

[37] I. Bergstrom and A. F. Blackwell, "The practices of programming," in *VL/HCC*, Sep. 2016, pp. 190–198.

[38] I. Bergstrom and R. B. Lotto, "Code bending: A new creative coding practice," *Leonardo*, vol. 48, no. 1, pp. 13–25, 2015.

[39] N. Collins, A. McLean, J. Rohrhuber, and A. Ward, "Live coding in laptop performance," *Organised Sound*, vol. 8, no. 03, pp. 321–330, Dec. 2003.

[40] T. Magnusson, "Herding cats: Observing live coding in the wild," *Computer Music Journal*, vol. 38, no. 1, pp. 8–16, 2014.

[41] A. McLean, "Hacking Perl in nightclubs," *perl.com*, 2004.

[42] E. H. Trainer, A. Kalyanasundaram, C. Chaihirunkarn, and J. D. Herbsleb, "How to hackathon: Socio-technical tradeoffs in brief, intensive collocation," in *CSCW*, 2016, pp. 1118–1130.

[43] G. Moody, *Rebel Code: The Inside Story of Linux and the Open Source Revolution*. Basic Books, 2002.

[44] S. Rosenberg, *Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and One Question for Transcendent Software*. Crown Publishing Group, 2007.

[45] T. Barik, E. Murphy-Hill, and T. Zimmermann, "A perspective on blending programming environments and games: Beyond points, badges, and leaderboards," in *VL/HCC*, Sep. 2016, pp. 134–142.

[46] T. Dal Sasso, A. Mocci, M. Lanza, and E. Mastrodicasa, "How to gamify software engineering," in *SANER*, Feb. 2017, pp. 261–271.

[47] A. Georgakopoulou, "Thinking big with small stories in narrative and identity analysis," *Narrative Inquiry*, vol. 16, no. 1, pp. 122–130, Aug. 2006.

[48] P. J. Guo, "Older adults learning computer programming: Motivations, frustrations, and design opportunities," in *CHI*, 2017, pp. 7070–7083.

[49] M. F. Costabile, P. Mussio, L. Parasiliti Provenza, and A. Piccinno, "End users as unwitting software developers," in *International Workshop on End-user Software Engineering*, 2008, pp. 6–10.

[50] C. Kelleher, R. Pausch, and S. Kiesler, "Storytelling Alice motivates middle school girls to learn computer programming," in *CHI*, 2007, pp. 1455–1464.

[51] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–15, Nov. 2010.

[52] M. J. Lee, F. Bahmani, I. Kwan, J. LaFerte, P. Charters, A. Horvath, F. Luor, J. Cao, C. Law, M. Beswetherick, S. Long, M. Burnett, and A. J. Ko, "Principles of a debugging-first puzzle game for computing education," in *VL/HCC*, Jul. 2014, pp. 57–64.

[53] K. T. Stolee and T. Fristoe, "Expressing computer science concepts through Kodu game lab," in *SIGCSE*, 2011, pp. 99–104.