Examining the Realities and Nuances of 'Low-stakes' Interest-Driven Learning Environments

Marcelo Worsley
Northwestern University
Evanston, IL USA
marcelo.worsley@northwestern.edu

***Abstract*—The push to develop low-stakes and personally meaningful computer science experiences is creating novel opportunities to broaden participation in CS. These opportunities have become increasingly present across contexts and have expanded the possibilities for introducing and sustaining student participation in computing. However, while these experiences tend to be effective ways for engaging new participants and new forms of participation, we must be careful to not overlook how 'high-stakes' these experiences might be for learners. To explore this tension, this paper describes two case studies of students engaging in coding and computational thinking with Minecraft Education Edition. The first case study involves a 7-year-old Black and Latina girl who experiences significant frustration when her computer program destroys significant portions of her project. The second is from a Latino boy who avoids using the coding capabilities in Minecraft EDU out of fear that the code might not work properly. Building on these case studies, this paper suggests that the field take steps to ensure that the language and actions associated with low-stakes and high-stakes are reflective of learner perceptions, and that we design learning experiences that appropriately reflect this nuance.**

Keywords— Game-based learning, Informal, Formal, Pedagogy

I. Introduction

Education researchers have long advocated for interest-based learning experiences [1]. These experiences build on learner goals, identities, and interests to foster increased student motivation [2]–[4]. When initially conceptualized, much of these experiences were implemented in schools [1], [5], [6]. In recent years, interest-based learning has experienced a noticeable resurgence with a noted expansion into informal learning environments [7]–[11]. Researchers and educators have developed initiatives that bridge learning and games [10]–[13], cooking [14], music [15], [16], physical movement [17]–[19], art [20] and the environment [21], for example. These informal spaces can provide an engaging entry point to science, technology, engineering, and mathematics (STEM) disciplines and serve as a counterpoint to the emphasis that schools and school districts may place on standardized tests. However, despite the tendency for interest-based learning to offer a more engaging alternative, we should also recognize the potential high stakes nature that these experiences might embody for youth. This paper explores this idea by examining the following research question: In what ways do youth demonstrate the possible high stakes nature of interest-based learning in a game-based learning environment?

In answering this question, we first turn to describing prior literature from the interest-based learning and computer science education research communities. We then describe the context for this study and move into a presentation of two case studies from different game-based learning experiences. Next, the discussion section offers some key takeaways from the case studies and describes possible implications that this work might have outside of the game-based learning context. Finally, the conclusion highlights future directions for this work, and additional questions that the research community might consider in the context of interest-based learning environments.

II. Prior Literature

A. Frameworks of Interest-Based Learning

While seldom referred to as interest-based learning in contemporary education research, Nasir's [2] model for the interactions among learning, goals, and identity serves as an important grounding as we think about interest-based learning. Nasir describes bi-directional interactions among a learner's identity, learning, and goals as facilitating interest-based learning. Learners may have certain goals, based on their identity, that subsequently push them to embark on new learning experiences. Similarly, as they achieve their goals and succeed at learning, they grow their identities within that specific domain. As we look through the case studies in this paper, we will see some of these interactions among our participants' goals, learning, and identity.

B. Culturally Responsive-Sustaining Computer Science

A growing number of scholars are exploring ways to incorporate and elevate learner identities in the context of computer science. Much of this work falls under the banner of culturally responsive-sustaining computer science education [7], [20], [22]. Importantly, these approaches are about more than simply motivating youth to learn computer science. Instead, these initiatives demonstrate ways that computing might contribute to learner's appreciation, awareness, and engagement with their culture. The learning context that we describe in this paper takes a similar approach by helping students see ways that computation can support their process for designing and building in a game-based learning environment.

C. Guidelines and Best-Practices for Game-based Learning

Finally, this paper is informed by prior research on game-based learning. One area of game-based learning research uses youth interest in games as a primary motivation for exploring computational thinking. Some game-based learning environments require students to program an avatar's actions, or solve different types of puzzles, while other platforms provide more open-ended virtual sandboxes. In this project, youth are invited, though not required, to learn different computational thinking concepts through Minecraft Education Edition (EDU). Many aspects of the platform align with Gee's guidelines for game design [23]: co-design, customizations, identity, manipulation and distributed knowledge, well ordered problems, pleasantly frustrating, cycles of expertise, skills as strategies, information "on demand" and just in time. These principles provide a useful backdrop for thinking about youth experiences within game-based learning environments. In particular, we see how the principles of just in time, pleasant frustration, and skills as strategies effect identity and learning within our two case studies.

III. METHODS

A. *Program Participants and Contexts*

The author has worked with a suburban school district to provide in-school and out-of-school game-based learning experiences using Minecraft EDU for the past few years. This partnership has included creating and implementing after-school clubs, training teachers how to use Minecraft in their classes, supporting summer programs for youth, and facilitating district-wide Minecraft challenges. In total, the research team has worked with more than one thousand students and 15 educators. Within this paper we take a close look at two student experiences in these programs. The participants were selected because they offer prime examples of how youth may experience aspects of an interest-based activity as high stakes. At the same time, the difference in age and context also provides some indication that their experience is not strictly tied to one type of learning context or the other. While we are not suggesting that these students' experiences are the norm, we have noted similar interactions among other individuals in their respective peer groups. Data for each case study is based on direct interactions between the research team and the students. During the different programs, the research team generated field notes and memos that inform this analysis. We also conducted interviews with some participants and parents after the conclusion of the programs.

B. *Background on Minecraft Education Edition*

Minecraft is a virtual sandbox game that remains immensely popular among elementary and middle school students. In the game, players utilize blocks of different materials to construct buildings, powered machines, and entire worlds. Over the last decade, educators and researchers have utilized the game for many educational purposes [13], [24]–[28]. As interest in Minecraft grew, Microsoft decided to create a version of the game that was specifically designed to support learning and teacher management. This version, Minecraft EDU, includes the ability to conduct chemistry experiments and explore professional worlds that replicate historic locations and align to various curricular standards. Minecraft EDU also includes an embedded programming environment where students can programmatically interact with or modify their virtual world. Using the coding interface, students can gain exposure to and develop expertise with variables, conditionals, functions, iteration (loops), decomposition, debugging, problem solving, collaboration, and many other computational thinking related constructs. Many of these are natively available through Minecraft but are simplified through the embedded programming environment. Youth participating in our programs are introduced to many of these capabilities but are not required to utilize them. Our programs also teach participants how to use scripting in Minecraft EDU. This capability gives participants familiarity with a set of commands that they can use within the in-game chat terminal, or within command blocks. The specifics of these coding interfaces is not essential for understanding the case studies. Instead readers should understand that Minecraft EDU involves several components that can be used to support computational thinking and computer programming.

IV. CASE STUDIES

A. *Amelia*

Amelia is a gregarious seven-year-old Black and Latina girl who attends a title one school in the Midwestern region of the United States of America. Amelia does not identify as a gamer, but occasionally enjoys playing games like Animal Crossing New Horizons, Mario Kart, and Super Smash Brothers on the Nintendo Switch. Amelia is quite social and enjoys playing with other girls her age. Amelia signed up to participate in a 6-week Minecraft after-school club that met once per week during the winter term. The club included 15 students from Kindergarten through 2nd grade. Each grade had roughly the same proportion of participation, and the participants met with three members of the research team on a weekly basis in an open classroom. The club was majority Black and Brown, and had boy-girl gender parity. The original six-week program was disrupted by the COVID-19 pandemic, but a four-week summer session was offered during the summer of 2020.

The specific episode that we will focus on with Amelia took place in the summer of 2020. Amelia, and many of the other after-school club participants, were excited to continue working with Minecraft after their after-school program abruptly ended because of COVID-19. Over the course of 4 weeks in July, the participants collaboratively created a treehouse amusement park. In the process of creating the amusement park, the research team showed the students different coding capabilities in Minecraft. Of particular interest to Amelia's example is a piece of code called Super Digger. This code can be created within the Minecraft EDU block-based programming environment and allows the user to quickly destroy anything within a specified distance from the player's location. It is referred to as Super Digger because the code can greatly simplify the process of excavation. We had introduced the Super Digger code to assist us in destroying structures that we had created with errant or otherwise faulty code. Amelia used the Super Digger code with assistance on a few different occasions. On one particular day, Amelia was individually working in a world that she was enhancing when she came across a pre-built structure that was blocking the area where she wanted to build. She wanted to clear the space and turned to the Super Digger code. The first time she tried it did not work. She tried again, but with the same result. With the help of a research team member, Amelia realized that she was not using the correct event handler. She had programmed the Super Digger to work while walking, instead of while flying. They helped her update her code and left her to try running the code. Moments later she resurfaced visibly upset. The Super Digger code had worked, but she had destroyed more of the space than intended. After some discussion we decided to reload the original world. She remade her changes to the original world, reloaded the code from the previous world, and proceeded to clear out the space as desired. Unfortunately, after a minute or two, she was once again distraught. She had started the code, but was unable to make the code stop running. As a result, she had destroyed the very structure that she was trying to extend. Even as she talked with the researchers, her player was continuing to fly around, inadvertently destroy everything. As this continued, Amelia remarked that she was giving up on Minecraft. She refused to try building the structure again. She was convinced that she could not do it properly using the code but was equally convinced that doing it manually was a poor use of time given her knowledge of the code builder. Amelia held to this refusal to play Minecraft for the next several months. While her friends would occasionally get together to play in a shared world, Amelia opted out. Roughly one year later she decided to give Minecraft a try once again and has not returned to using the code builder ever since. She will still use different text chat commands for teleporting and changing the time of day, but the code builder remains an object of scorn.

One interpretation of Amelia's experience is that it failed to satisfy Gee's pleasant frustration guideline. She made multiple attempts to utilize the code builder, but to no avail. Instead of being an experience where she could leverage computer science to improve her Minecraft world, she became detached from both coding and Minecraft. One might also interpret Amelia's experience as an instance where the challenges she encountered put her identity as somewhat Minecraft proficient into question. In the same way that Nasir talks about identity, learning, and goals working in concert with one another, one can equally consider that negative experiences in one dimension, might disrupt the others. In this case, the code builder was actively detracting from Amelia's goals of creating her structure. The ongoing challenges reinforced that she was not effectively learning how to use the code builder, and this was negatively impacting her self-perception.

B. Alejandro

Alejandro is a 7th-grade Latino boy. He enjoys playing console video games and computer games, but does not really identify as a gamer. While happy to talk, Alejandro tends to keep to himself. In class, he sat at a computer where there was no one to his left, and typically an open seat to his right. This relative aloneness did not seem to bother him.

Alejandro's episode occurred during an in-school unit where students were asked to design a game using Scratch, Minecraft, or digital fabrication tools. The students had approximately two weeks to conceptualize and build their games. As someone who had prior experience with Minecraft, Alejandro immediately gravitated to creating his game in Minecraft. The first two days of our interaction with Alejandro involved students completing a coding tutorial in Minecraft EDU's block-based programming environment. Students followed the Chicken Rain tutorial which shows them how to make chickens automatically spawn over the players head as they walk. After completing this first task, students were challenged to turn this into a mini-game where they kept track of the number of chickens they could shoot with their bow and arrow. Alejandro completed the first task with no assistance, but needed some support to complete the second portion, which required creating a new variable and including a new event handler. Nonetheless, Alejandro was noticeably excited when he got his code to work. At the conclusion of the two days of getting acquainted with block-based coding in Minecraft EDU, students were free to work on their games in teams or individually. The next day Alejandro jumped right into creating his world. He elected to create a rodeo challenge where players would ride horses and protect the horses from angry non-player characters that attack the horses. The rodeo world included a fenced area for the horses, a seating area for spectators, and another fenced area for the attacking non-player characters.

On the particular day that we focus on in this paper, Alejandro had used the Chicken Rain code to quickly spawn horses for his stable and had manually constructed the seating area for one side of the arena. After hearing about other students who had successfully used programming in their projects, he asked about ways that code might expedite the process for making the other side of the arena. The opposite side of the arena would be a mirror image of the side that he already created. Minecraft EDU has a copy function that allows players to clone an existing structure, and place it in a different location. However, that function does not allow for rotating the structure, which means that making a mirror image of the existing seating area would not be possible. Instead, we worked on programmatically creating stairs in the code builder. We went to a remote part of the world to get acquainted with how to create the stairs after first verifying that the clone feature noted above, would not work for this task. Once in that remote space, it took several tries, but we eventually were able to create a set of stairs that was 10 blocks long and went up 20 blocks in the air. Alejandro used the coding agent so we could watch the actions unfold. The code instructed the agent to put down blocks while moving from one end to the next. Upon reaching the end of any row, the agent would turn to the right, move up one, walk forward one step, turn the right, and then proceed back to the other end. Once at the other end, they would turn left, move up one space, walk forward one step, and turn left again, before heading back down to the end again. To make the program run, the user simply needed to provide the length and height of the desired stairs. Alejandro was impressed with himself when we got it to work and showed his peers that he had successfully coded a large staircase. I congratulated Alejandro on his persistence and went to work with some of the other students in the class. However, when I passed by Alejandro a few minutes later, he was manually constructing the other side of the arena. When I asked him about why he decided not to use the code, he remarked that "it was too risky." In short, while he was able to eventually figure out how to make the stairs, the various roadblocks and challenges that we had to overcome instilled a general sense of distrust in the code. Moreover, he was concerned that if he tried to use the code, something might go wrong, and portions of his arena would need to be reconstructed. Even though I reassured him that we could make a copy of the entire world before testing the code, he was insistent on his preference to simply do it manually. He had already invested too much time and energy into his project to see it potentially get destroyed by running his computer program.

Alejandro's case surfaces a fascinating phenomenon where, in the language of Nasir [2], his goals propelled him to learn about using the code builder, but where the learning ultimately caused him to realize that he would rather not utilize the code builder to complete the arena. Moreover, Alejandro realized that much of the just in time information that comes from testing one's code in Minecraft EDU is insufficient for supporting programmatically creating the stairs. Put differently, the code testing process provided useful information in the moment that he could process and adapt but dealing with that type of iteration in the designated arena space was out of the question. At the same time, Alejandro's example also relates to Gee's 'skills as strategy' principle. He learned how to programmatically make stairs to create the arena seating. However, he still found the approach to be too risky.

V. Discussion

Looking across these case studies, we see apparent examples where youth engaging in interest-based computer science activities experience noticeable frustration and discontent. Moreover, there are ways that the processes of coding, or using computer programming led to significant aggravation, and, at times, withdrawal from an activity that the youth typically enjoyed. The cases shared in this document are a few of the many instances that we have observed across in-school and out-of-school Minecraft-related learning experiences. The cases also highlight possible ways for navigating these moment of youth distress and offer some suggestions for how designers might mitigate such situations. At the most basic level, some of the learner apprehension to use code could be reduced by making it easier for youth to undo the changes to the Minecraft world that resulted from using coding. In a similar vein, the platform could provide a transparent overlay, for example, of the expected outcome of the code. This way using coding becomes less costly. These technological changes could have addressed the core frustrations that arose within each of the case studies. Absent this technological component, adults tried to promote instructional supports that accounted for possible coding errors. Namely, students were encouraged to test their code in areas of the virtual world that the student did not plan on using. In other instances, the facilitators suggested that youth make a copy of their virtual worlds before testing their computer programs. These suggestions, however, may not always assuage youth concerns, nor will a single approach work for all youth. On the contrary, part of what we learn through Nasir's framework is that youth experiences are heavily mediated by their identities and goals. Hence, it is important for adults to be sufficiently trained with a collection of strategies to address both the content and the emotional nature of potential moments of youth distress.

While we have primarily talked about this work in the context of Minecraft, the ideas raised also have relevance across interest-based learning environments more broadly. Technological, or design, choices of the coding platforms should include some intentional features that recognize the high level of importance that youth might place on the program content within interest-based and/or culturally sustaining computing experiences. Similarly, instructors and facilitators should help normalize making mistakes as a common part of the programming process, and teach students strategies for proactively managing those errors. Finally, educators should be taught core practices for supporting participant socio-emotional well-being. While this is being increasingly emphasized within in-school, K-12 contexts, socio-emotional learning should be equally as integral to educator learning and support in out-of-school contexts. Far from being spaces where youth become relaxed and indifferent, these interest-driven spaces may be closely tied to youth identities, aspirations, and community roles. As a result there can be a high propensity for emotionally charged interaction. Failure to attend to these considerations could result in the cycle of goals, identity, and learning negatively impacting one another, and pushing youth farther away from computing-related experiences.

VI. Limitations and Future Work

The data presented in this paper are a subset of the interactions that we have observed over the past few years. While these case studies do not encapsulate every participant's experience, we have observed several similar episodes among the youth and the programs that we work with. That said, all of the examples are derived from a single school district in a large metropolitan area. Furthermore, this work looked at a single type of game-based learning environment that, despite its popularity, has some apparent shortcomings. Part of what this paper advocates for, though, is for researchers, designers, and practitioners to reflect on these shortcomings and develop tractable strategies to overcome them. We also position this as an area for future research, both in terms of looking at other game-based learning environments and other types of approaches for instantiating interest-based learning. Conducting such work would help delineate additional dimensions that the community might consider and highlight concerns that may be specific to certain types of interest-based programs.

VII. Conclusion

Proliferation of game-based learning and other interest-based learning environments is creating many new avenues for future generations of computing-related professionals to explore and learn about computer science. Moreover, these environments represent a meaningful way to widen participation in computer science. However, utilization of interest- and identity-based learning environments necessitates an additional level of care and attention. Youth proximity and interest to many of the core ideas and activities within these expansive learning environments mean that there may be a higher propensity for them to negatively internalize any shortcomings or perceived failures that they encounter within the space. As researchers, designers, and practitioners, we can help mitigate this by acknowledging that these spaces might feel like very high stakes learning environments for youth. Doing so means that we carefully consider ways that the technology can best support undoing erroneous actions, for example, teaching youth how to plan for likely errors in their coding, and training facilitators with various strategies to appropriately address any socio-emotional needs that may arise.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1822865. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Dewey, *Interest and effort in education*. Forgotten Books, 1913.

[2] N. S. Nasir, "Identity, Goals, and Learning: Mathematics in Cultural Practice," *Math. Think. Learn.*, vol. 4, no. 2–3, pp. 213–247, 2002.

[3] G. Ladson-Billings, "Toward a Theory of Culturally Relevant Pedagogy," *Am. Educ. Res. J.*, 1995.

[4] D. Paris, "Culturally Sustaining Pedagogy," *Educ. Res.*, 2012.

[5] D. C. Edelson and D. M. Joseph, "Motivating active learning: A design framework for interest-driven learning," *DBRC Publ. Retrieved March*, 2001.

[6] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar, "Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning," *Educational Psychologist*, vol. 26, no. 3–4. pp. 369–398, 1991.

[7] K. A. Scott, K. M. Sheridan, and K. Clark, "Culturally responsive computing: a theory revisited," *Learn. Media Technol.*, vol. 40, no. 4, pp. 412–436, 2015.

[8] M. Ito *et al.*, *Hanging out, messing around, and geeking out: kids living and learning with new media*. 2010.

[9] B. Barron, K. Gomez, N. Pinkard, and C. K. Martin, "The Digital Youth Network Learning Model," in *The Digital Youth Network*, 2020.

[10] J. P. Gee, "Learning by Design: Good Video Games as Learning Machines," *E-Learning Digit. Media*, vol. 2, no. 1, pp. 5–16, 2005.

[11] J. L. Plass, B. D. Homer, and C. K. Kinzer, "Foundations of game-based learning," *Educ. Psychol.*, vol. 50, no. 4, pp. 258–283, 2015.

[12] Y. B. Kafai, *Minds in play: Computer game design as a context for children's learning*. Routledge, 1995.

[13] C. Lane and S. Yi, "Playing With Virtual Blocks: Minecraft as a Learning Environment for Practice and Research," in *Cognitive Development in Digital Contexts*, 2017.

[14] J. C. Yip, T. Clegg, E. Bonsignore, B. Lewittes, M. L. Guha, and A. Druin, "Kitchen Chemistry: Supporting Learners' Decisions in Science," 2012.

[15] J. Freeman, B. Magerko, D. Edwards, T. Mcklin, T. Lee, and R. Moore, "EarSketch," *Commun. ACM*, 2019.

[16] M. S. Horn *et al.*, "TunePad: Engaging learners at the intersection of music and code," in *Computer-Supported Collaborative Learning Conference, CSCL*, 2020.

[17] V. Lee, J. R. Drake, R. Cain, and J. Thayne, "Opportunistic Uses of the Traditional School Day Through Student Examination of Fitbit Activity Tracker Data," pp. 209–218, 2015.

[18] A. Zimmermann-Niefield, M. Turner, B. Murphy, S. K. Kane, and R. B. Shapiro, "Youth Learning Machine Learning Through Building Models of Athletic Moves," in *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, 2019, pp. 121–132.

[19] S. Jones, J. Thompson, M. Smith, and M. Worsley, "Data in Motion: Sports as a site for expansive learning," *Comput. Sci. Educ.*, pp. 1–34, 2020.

[20] Y. Kafai, K. Searle, C. Martinez, and B. Brayboy, "Ethnocomputing with electronic textiles: Culturally responsive open design to broaden participation in computing in American Indian youth and communities," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 241–246.

[21] A. Marin and M. Bang, "'Look it, this is how you know:' Family forest walks as a context for knowledge-building about the natural world," *Cogn. Instr.*, vol. 36, no. 2, pp. 89–118, 2018.

[22] K. Davis, S. White, D. Becton-Consuegra, and A. Scott, "Culturally Responsive-Sustaining CS Education: A framework," 2021.

[23] J. P. Gee, "Good Video Games and Good Learning," *Good Video Games Good Learn.*, pp. 33–37, 2016.

[24] B. Andrus, D. Bar-el, C. Msall, D. Uttal, and M. Worsley, "Minecraft as a Generative Platform for Analyzing and Practicing Spatial Reasoning," *Spat. Cogn. XII*, 2020.

[25] Y. J. Jung, D. Toprani, S. Yan, and M. Borge, "Children's Participation in Rulemaking to Mitigate Process Problems in CSCL," *Proceeding 2017 Comput. Support. Collab. Learn. Conf.*, pp. 652–655, 2017.

[26] S. Nebel, S. Schneider, and G. D. Rey, "Mining Learning and Crafting Scientific Experiments: A Literature Review on the Use of Minecraft in Education and Research," *J. Educ. Technol. Soc.*, vol. 19, no. 2, pp. 355–366, 2016.

[27] [27] S. C. Duncan, "Minecraft, beyond construction and survival," *Well Play. a J. video games, value Mean.*, vol. 1, no. 1, pp. 1–22, 2011.

[28] [28] B. Bos, L. Wilder, M. Cook, and R. O'Donnell, "Learning mathematics through Minecraft," *Teach. Child. Math.*, vol. 21, no. 1, pp. 56–59, 2014.