Exploring Persistence in Computing Education: An Ordered Network Analysis of Student Engagement and Persistence in a Minecraft Learning Environment

Authors: Bess Hagan

Abstract

This research investigates persistence in the development of computational thinking (CT) skills within a virtual game-based learning environment. Data was collected from adult and student participants engaged in an introductory computing course using the WHIMC BarrelBot learning environment. Event log data was analyzed using Exploratory Data Analysis (EDA) and Ordered Network Analysis (ONA) to identify markers of struggle and persistence as users engaged in problem-solving tasks. Our findings offer valuable information for future studies on persistence in computing education and may assist in developing persistence learner models that enhance targeted interventions for struggling learners. This research contributes to a growing body of work on persistence in education and ongoing efforts to create inclusive and effective educational experiences tailored to the diverse needs of all students.

Keywords: Persistence, computational thinking (CT) skills, virtual game-based learning environment, WHIMC BarrelBot, computing education, event log data, Exploratory Data Analysis (EDA), Ordered Network Analysis (ONA), engagement, struggle, problem-solving strategies, educational interventions, student behavior, targeted interventions, inclusive educational experiences

Introduction

Persistence in education is critical to student success, particularly in challenging subjects like computing, which requires computational thinking (CT) skills such as planning and problem-solving. Teaching CT to young students not only prepares them for future academic and career opportunities but also provides an opportunity to practice logical reasoning and creative problem-solving. Fortunately, virtual learning environments can foster CT skill development while simultaneously collecting information-dense data on how young learners approach computational problem-solving. This data can provide vital information on how students engage, struggle, and persist in learning, allowing for the development of more inclusive virtual learning environments that can anticipate student needs and help them persist when they face challenges.

Accordingly, we investigate how students persist while learning by utilizing a Minecraft-based virtual learning environment built by WHIMC[1] (What-If Hypothetical Implementations in Minecraft) called BarrelBot. This innovative learning environment introduces computing concepts such as loops and functions to students by engaging them with the virtual character BarrelBot, a programmable entity they use to navigate obstacle courses of increasing complexity in a setting that is fun, familiar, and interactive.

Our research focuses on the analysis of log data generated by junior high school-aged participants as they attempt to solve puzzles in the BarrelBot learning environment. We seek to identify markers of engagement, struggle, and persistence during these activities. However, prior to our student data collection, adult-generated mock data was used in the development of

[1] https://whimcproject.web.illinois.edu/

Python scripts to extract, process, and calculate key metrics, and this information was used to create operational definitions for in-game events that could be mathematically visualized using Ordered Epistemic Network Analysis (ONA). ONA is a powerful tool for understanding sequential co-occurrences, and it provided valuable insights into student behavior as we endeavored to identify markers of struggle, persistence, and disengagement.

Ultimately, the findings from this study have the potential to inform the design of learner models and educational interventions that can help a diverse student population persist when they face learning challenges, especially in computing education.

Related Works

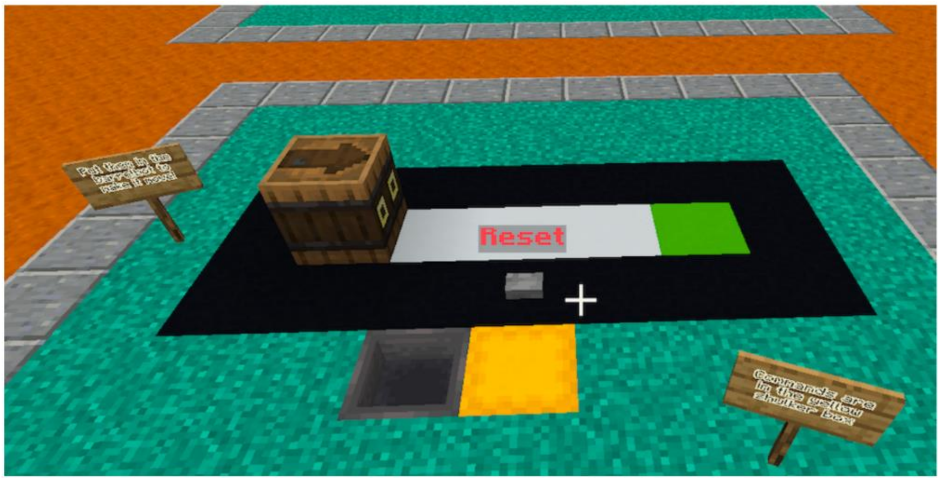
Persistence is often described as the continued effort toward a goal despite obstacles and setbacks (Israel-Fishelson, & Herskovitz, 2020; Owen et al., 2019). Research on persistence in education highlights its importance in student success, particularly in challenging subjects like computing, which require sustained effort and CT skills such as problem-solving and logical reasoning. Over time, game-based learning environments have emerged as an effective platform for teaching these skills because they offer hands-on learning experiences that engage students. Moreover, log data generated on these platforms can be utilized to study persistence and inform the development of interventions that support students in overcoming challenges to improve learning outcomes.

Israel-Fishelson and Hershkovitz (2020) explored how CodeMonkey, a game-based learning environment, fostered persistence and the acquisition of CT skills among elementary students. They delved deeper into this subject by researching micro-persistence to examine how students persist at the task level (Israel-Fishelson, & Herskovitz, 2021), and their works collectively underscore the importance of tailoring educational interventions to support persistence at different stages of learning.

Before this work, Owen et al. (2019) used log data to differentiate wheel-spinning (persistence without meaningful progress) from productive persistence (where students make steady progress toward their learning goals). Their findings also emphasized the need for more timely and adaptive interventions to support students in ways that allow them to persist productively in virtual learning environments.

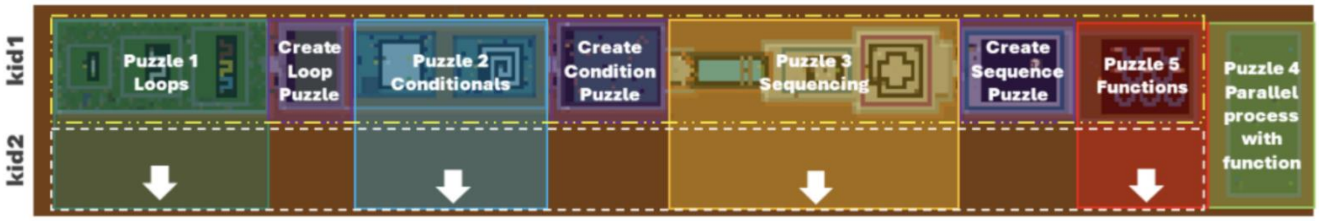
Research conducted by Israel-Fishelson and Hershkovitz (2020, 2021) and Owen et al. (2019) demonstrate how game-based learning environments are ideal platforms for studying persistence in computer science education, and they provide a comprehensive view of why further research in this area is needed. Our research builds on these works by applying ONA (Tan et al., 2022) to data collected from the WHIMC BarrelBot learning environment. Using this analysis tool, we can mathematically visualize in-game student behavior as networks of sequential co-occurrences to gain insight into how students approach problem-solving activities. Generalizations from this analysis to identify markers of struggle and persistence may help to inform the design of educational models and interventions that better support students in developing CT and computing skills that are tailored to the specific needs of a diverse student population.

The Learning Environment: WHIMC BarrelBot


Put them in the barrelbot to make it move!
Reset
Commands are in the yellow shulker box

BarrelBot's first obstacle course in the Puzzle 1 Loops (P1L) section

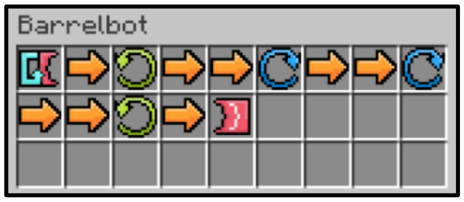
Introduced in 2023, WHIMC BarrelBot is an innovative system designed within a Minecraft server to present an introductory computing course to primary school-aged students. This virtual environment features BarrelBot, a crate-shaped character with an emotive face that adds an element of engagement to this interactive learning experience. Using a block-based approach, students program BarrelBot to navigate obstacle courses, learning fundamental programming concepts such as loops, conditional logic, algorithmic sequencing, and functions.

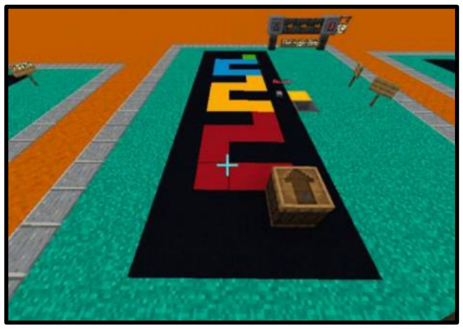


Lane sections are divided into computing concepts. The creative areas between each section allow students to apply what they have learned by creating puzzles of their own for the user in the next lane to solve.

During data collection, each student in this learning environment is assigned a lane. Each lane is subdivided into sections of increasing difficulty, structured to introduce computing concepts that progressively build throughout gameplay. Although there are creative areas between sections where students pair up to build and solve puzzles for one another, our research omits these due to their lack of fixed solutions. Instead, our focus is on the Puzzle 1 Loops (P1L) section of each lane, which contains three puzzles. The first puzzle is straightforward, requiring students to program BarrelBot to move forward four times, serving as an introduction to basic programming mechanics. The second puzzle is more complex, with a serpentine obstacle course. The third puzzle introduces loops, challenging students to use a loop to navigate a course similar to the second puzzle.

When ready, students begin puzzles by obtaining code blocks from a yellow Minecraft Shulker Box (a generic container) embedded in the ground. After placing these code blocks in their inventory, they are free to program BarrelBot in that puzzle region. Any number of directions can be added, removed, or rearranged inside the bot, with the only limitation being the number and type of directions obtained from the Shulker Box for that puzzle. Students can test their solutions by pressing a grey stone button near the Shulker Box, and they can adjust, run, and reset their program as many times as they like. Correct solutions allow BarrelBot to navigate through the obstacle course successfully, while incorrect solutions result in BarrelBot crashing off course, displaying a dizzy expression on its face.





Students can explore their environment and solve puzzles in any order, or they may choose not to engage with BarrelBot at all. However, access to the creative area beyond P1L, where students build their own puzzles and interact with their lane partners, requires the completion of the third puzzle. Knowledge of this may provide the necessary motivation students require to work on puzzles to completion in this section.

Methods

Our initial efforts focused on generating data from the virtual learning environment and conducting exploratory data analysis (EDA) to understand the dataset. The data included unique identifiers such as usernames and puzzle names, as well as the puzzle boundary coordinates to identify puzzle regions across all user lanes. Data on button presses, code blocks used, and successful outcomes were also used to analyze user behavior in the learning environment. Amongst the log data, player movement was the most robust, logging the x, y, and z coordinates of all logged-in users every two to three seconds. This data was necessary to determine which puzzle regions users were in across multiple tables in the database. To simplify data processing, rows from this table where players' x, y, and z-coordinates remained stable were removed from the analysis. Therefore, it is important to note that time within the event log generated for ONA is ordered but relative to the activity of the learner.

Button presses that allow the user to run or reset each puzzle are grouped by usernames and puzzle names, and a count was recorded for each grouping to determine whether the user is running or resetting each puzzle. When the count is odd, the player is running the puzzle and when it is even, the user is resetting the puzzle. Combining this knowledge with known solutions to puzzles can establish whether the user ran the puzzle successfully or unsuccessfully. However, successful outcomes in this analysis are taken directly from the database as they are logged, regardless of the z-coordinate associated with that data.

Logged code block data from each BarrelBot interaction was converted into strings of text to calculate the edit distance (using Levenshtein distance) between interactions with the bot and between students' proposed solution at each interaction with the puzzle's solution. This method allowed us to numerically calculate the amount of change occurring between attempts and to see if the user was getting closer or farther from the solution. Thus, when the distance between the user's attempted solution string and the solution string for a puzzle reached zero, the user successfully ran the puzzle. These calculations and comparisons between the lengths of user solution strings were used to identify whether users were adding, removing, or rearranging directions between BarrelBot interactions as they attempted to solve puzzles.

Data was collected from approximately 100 junior high-aged students over two days. On the first day, these students were divided into five groups, and each group took turns playing the P1L section of BarrelBot using the same eighteen user accounts; some students worked side-by-side on the same computer while others worked alone. Data from two of the five groups was excluded from this analysis due to data loss and other difficulties that occurred during on-site data collection. With this in mind, our analysis of student behavior will treat each user account as though it were an individual student participant, but we will refer to students as users to avoid misrepresenting our results.

The following day, seventeen students from the initial five groups returned to play through the entire virtual learning environment a second time, beginning with the first three puzzles. These students worked in the same room on separate computers using the same user accounts as the day before but may or may not have been assigned the same username. Students were paired together by lane number so that creative areas beyond the third puzzle could be worked on cooperatively. Some student pairs actively worked together while others did not, and students were allowed to move around the room to assist one another regardless of whether they had completed any puzzles themselves.

Methods Subsection 1 – Processing Data & Operational Definitions

EDA was followed by developing various Python scripts to extract, process, and calculate key metrics, which were then used to create operational definitions for in-game events. These operational definitions were used to develop a Python script to calculate and format event log data for mathematical visualization using ONA.

Event	Operational Definition
Enter the puzzle region	The user's z-coordinates are within the minimum and maximum z-coordinate for a given puzzle.
Exit the puzzle region	The user entered the region of a puzzle, but their z-coordinates are now outside the minimum and maximum z-coordinate for that puzzle.
Interact with BarrelBot	The user approaches BarrelBot in its starting position and opens the menu where they may choose to make changes to the program using code blocks from their inventory.
Press stone button	The user presses the grey stone button for a given puzzle.
Run the program	Beginning at 1, count the number of times the user has pressed the stone button on a given puzzle. If the count is odd, the program stored in BarrelBot executes.
Reset the puzzle	Beginning at 1, count the number of times the user has pressed the stone button for a given puzzle. If the count is even, the user has run their program and BarrelBot has either successfully navigated the obstacle course or crashed. Resetting the puzzle places BarrelBot in the starting position but does not alter the existing program. Users must reset the puzzle before they can interact with BarrelBot or run their program again.
BarrelBot succeeds	The user has programmed BarrelBot to successfully navigate the obstacle course for a given puzzle by running the correct solution in BarrelBot.
BarrelBot fails	The user has pressed the run button, but they have not programmed BarrelBot to successfully navigate the obstacle course for a given puzzle.

To accomplish this, we used timestamps, usernames, and puzzle names to group in-game events into a binary event log. Events included entering and exiting puzzle regions, interactions with BarrelBot, button presses for running or resetting the puzzle, and whether the student's solution ran the bot successfully or not. Calculations for some events are dependent on the knowledge of other events. For example, a user cannot exit a puzzle region without first entering that region. Other events, such as BarrelBot fails, require the conversion of student solutions to strings and knowledge of static puzzle solutions for each puzzle.

To gain further insight into the students' problem-solving processes, the 'Interact with BarrelBot' event was further divided into subcategories. Each event subcategory calculation required both ONA event log data, student solution string lengths, and edit distance calculations comparing the student solution strings between each interaction with BarrelBot for a given puzzle. Dividing interaction events into these subcategories allowed us to capture whether the learner added or removed code blocks, looked in the bot without making any changes, rearranged the same number of code blocks previously stored in the bot, or decided to remove all code blocks from the bot.

Subcategories for BarrelBot Interactions	**Operational Definition**
Interact: Make no changes	While interacting with BarrelBot, the user neither adds, removes, nor rearranges any code blocks in their program.
Interact: Add code blocks	While interacting with BarrelBot, the user increases the number of code blocks in the bot.
Interact: Remove code blocks	While interacting with BarrelBot, the user decreases the number of code blocks but leaves at least one code block in the bot.
Interact: Remove all code blocks	While interacting with BarrelBot, the user removes all of the code blocks from the bot.
Interact: Rearrange code blocks	While interacting with BarrelBot, the user alters the program in the bot, but the number of code blocks in the bot does not change.

Information formatted for ONA was explored to categorize students by their problem-solving approaches in the hope that doing so might alert us to specific differences in how students' behavior reflects struggle or persistence. For instance, we hypothesized that removing all code blocks might indicate that something significant is happening to the learner in the environment. They may be frustrated and ready to give up on the puzzle or they may be deliberately resetting the puzzle for themselves to apply what they have learned without being influenced by previous failed attempts, which may signal persistence. Knowledge of how the student approached this puzzle before deleting their program may help us confirm this hypothesis and distinguish between the two outcomes to build better persistence learning models.

Preliminary ONA analysis on mock data highlighted the necessity of a status column as well, to determine whether sequential co-occurrences were part of the problem-solving process. This status column was first used in combination with usernames and puzzle names to conduct ONA analysis on data collected from student participants and later used to simplify ONA input files for analysis by removing behavior occurring before each student's first interaction with BarrelBot and after their first success for each puzzle.

Methods Subsection 2 – EDA and ONA

Epistemic Network Analysis (ENA) identifies and quantifies connections and represents these connections as dynamic visualizations of networks that can be compared visually and statistically[2]. ONA, a form of ENA, processes ordered coded data, such as an event log. Our event log was formatted for ONA to create mathematical visualizations of networks representing sequential co-occurrences of in-game user behavior. We used a stanza window of 3 and increased edge weights as needed to improve the visibility of directed edges and nodes which

[2] https://www.epistemicnetwork.org/

co-occurred with themselves. Finally, the placement of nodes about the mean, standard deviation, and other nodes was taken into consideration during our visual analysis.

We began with a cyclical process of EDA and ONA on mock data generated by adults. EDA findings on adult interactions informed the design of operational definitions of in-game events where users actively engage in solving BarrelBot puzzles. These definitions were used to process data into a binary event log for ONA, and ONA allowed us to further refine this event log. For example, we lacked definitions for meaningful movement beyond entering and exiting each puzzle region, so we excluded event rows where users were moving but not engaging in other meaningfully defined actions.

Next, we collected student-generated data over a two-day period. This data was cleaned, processed, and formatted into a binary event log, excluding behavior both before users began each puzzle and after each puzzle's completion. EDA provided context on user behavior from junior high school-aged students, revealing that Puzzle 1 (P1) offered little insight into solution building due to its simplicity. Consequently, we removed users from the data set who had not made a significant attempt to solve Puzzle 2 (P2) or Puzzle 3 (P3) in the P1L section of the learning environment.

The first round of ONA established a baseline for behavior across the remaining users. These visualizations represented average user behavior for each puzzle, informing the next round of EDA where we categorized students for a comparative analysis of problem-solving and solution-building strategies on P2 and P3 based on grouped user data, resulting in the following seven categories:

- Completed P1 and attempted P2
- Completed P1 and P2 without attempting P3
- Completed P2 without attempting P3
- Completed P2 and attempted P3
- Completed P3 and attempted P2
- Completed P3 without attempting P2
- Completed P1, P2, and P3

ONA was used again to compare grouped users by puzzle, forming generalizations about user behavior. These insights informed the final round of EDA through each user's detailed event log, which included intermediate calculations used to process and format the ONA event log, such as student solutions and edit distance calculations.

Results

Data was collected, cleaned, and processed from 62 user accounts interacting in the platform over two days. Completion rates for each puzzle were calculated by dividing the number of users that completed the puzzle by the number that attempted it. Initial completion rates were 87.9% for P1, 69.6% for P2, and 55.3% for P3. After calculating these rates, 10 users were excluded from further analysis. For the remaining 52 users, data related to activities before they began each puzzle and after they successfully programmed and ran BarrelBot were also excluded from the analysis. ONA was performed to compare the P1, P2, and P3 networks for all users. Codes were created for entering puzzle regions, exiting puzzle regions, BarrelBot interactions, running the program in BarrelBot, resetting the puzzle, failed attempts, and successful completions. Statistical analysis of these networks showed a Pearson and Spearman Goodness of Fit of 0.99 across the x-axis with a variance of 0.41 and 0.92 across the y-axis with a variance of 0.26. When code for general BarrelBot interactions was replaced with codes for the five interaction subcategories (adding, removing, removing all, rearranging, and making no change to code blocks) variance was reduced to 0.36 on the x-axis and 0.14 on the y-axis.

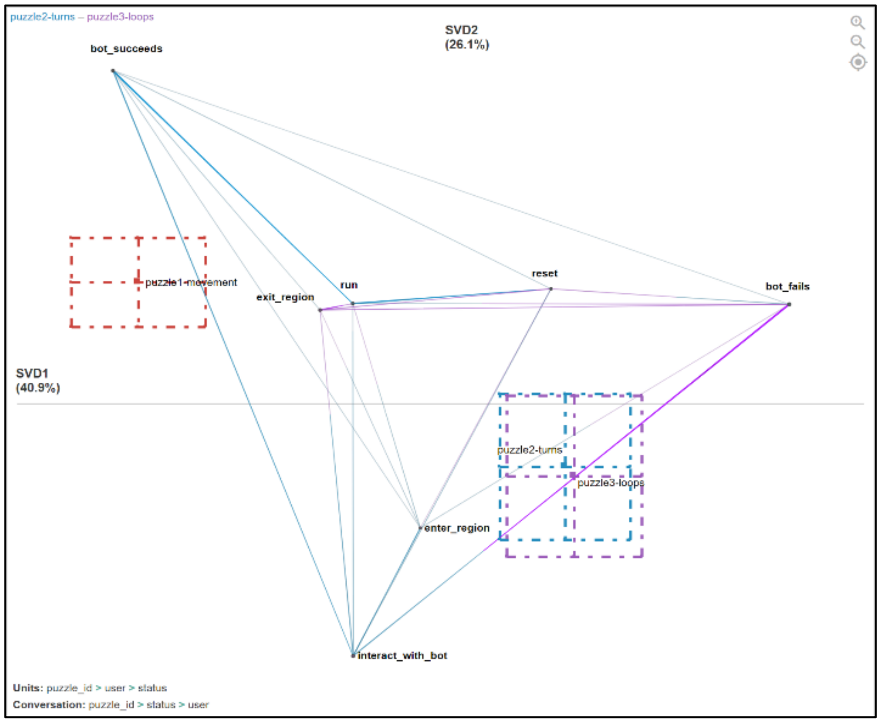


Figure 1 - ONA comparison of P2 and P3 networks

Differences between the sequential co-occurrences between the networks of P2 and P3 were minimal, as can be seen in Figure 1, with blue edges representing P2 and purple edges representing P3. Accordingly, we grouped data from P2 and P3 to form their average network for comparison with P1.

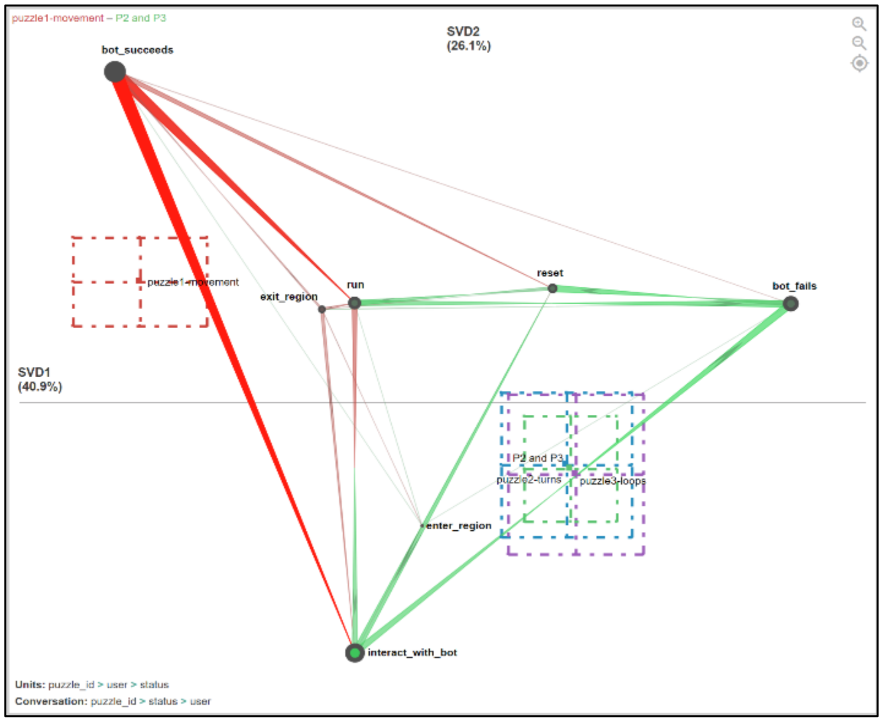


Figure 2 - ONA comparison of P1 with P2 and P3's average network

In Figure 2, the difference between P2 and P3's average green network and P1's red network revealed stronger co-occurrences of running BarrelBot successfully for P1. Conversely, co-occurrences related to bot interactions, resetting the puzzle, and running the program unsuccessfully were more heavily weighted in the average network of P2 and P3.

P1 was subsequently removed from further ONA and EDA analysis, and users were grouped by the puzzles they attempted and completed. The distribution was as follows: 19 users completed all puzzles (15 of whom were revisiting the P1L section), 5 users completed P2 but did not attempt P3, 12 users completed P2 and attempted P3, 11 users completed P1 and attempted P2, 2 users solved P3 but did not attempt P2, 2 users completed P2 but did not attempt P3, and 1 user completed P3 and attempted P2.

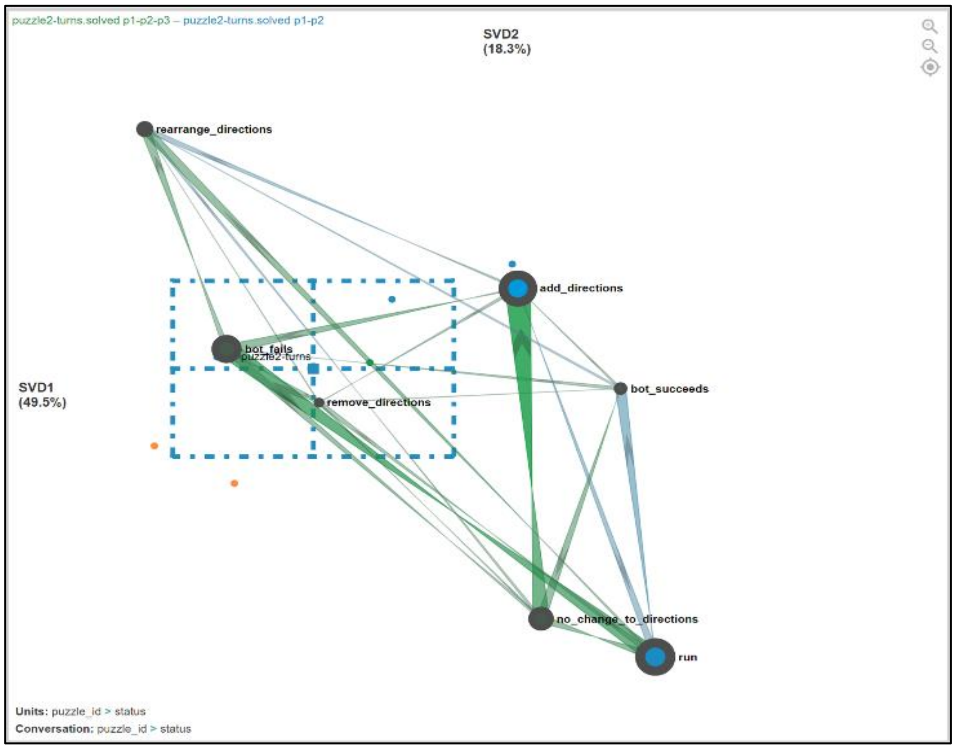


Figure 3 - ONA comparison of user groups for P2

In a comparison of two user groups that both completed P2 (Figure 3), the network of users who completed fewer puzzles (blue edges) had more events co-occurring with themselves, represented by the larger nodes with bright blue circles. For these users, adding code blocks to BarrelBot and running the program in BarrelBot often co-occurred with themselves. While the group that completed all three puzzles (green edges) had more bidirectional co-occurrences between interacting with BarrelBot without making changes and adding code blocks to BarrelBot, with slightly more weight on the directed edge from interactions without making changes occurring before adding new code blocks.

In Figure 4, the networks for users who completed all puzzles (blue edges) were compared to those who completed P2 but attempted P3 unsuccessfully (orange edges) on P3. Similar to the more successful group from P2 (Figure 3), unsuccessful users on P3 had more bidirectional co-occurrences of interactions where they made no changes to their program by adding code blocks to BarrelBot and adding code blocks to BarrelBot co-occurring with itself. Additionally, they had more co-occurrences of running BarrelBot unsuccessfully leading to interactions without making changes, and interactions without changes leading to itself. In contrast to this, the group that completed P3 had more co-occurrences of running their program unsuccessfully and rearranging code blocks before and after failed runs.

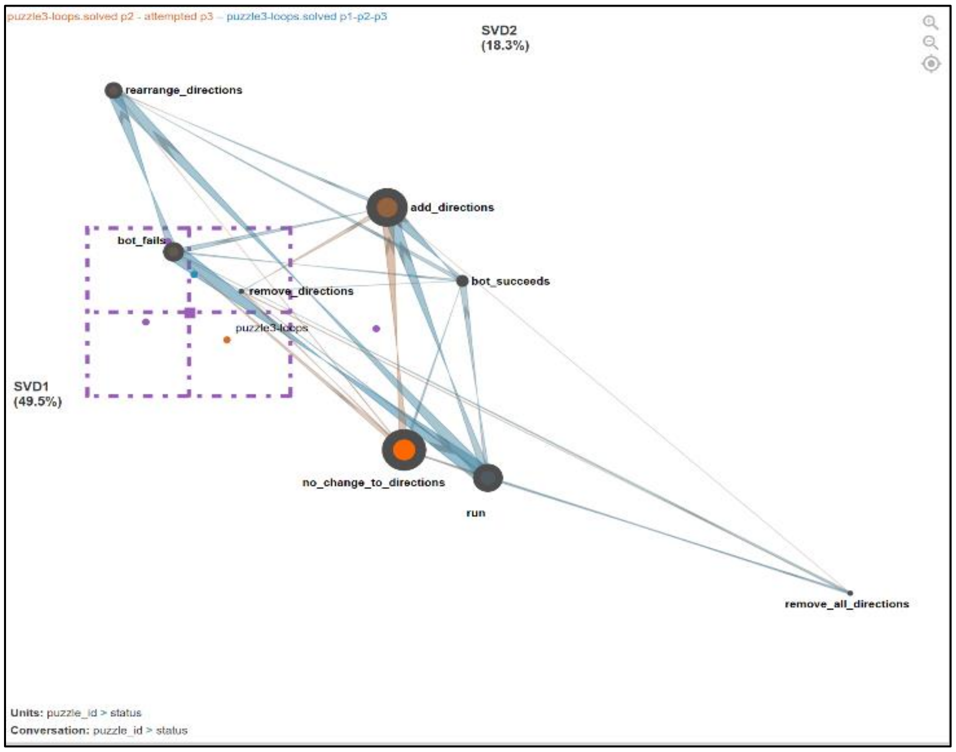


Figure 4 - ONA comparison of user groups for P3

These results led to another round of EDA in each user's detailed event log, focusing on (1) BarrelBot interactions where no changes to code blocks had been made and (2) edit distance calculations, with regard to whether the user successfully completed the puzzle or not. Edit distances of zero between interactions with BarrelBot indicated no changes to the program, which was common across users regardless of puzzle completion. It was also found that most users ran their program between BarrelBot interactions, but the number of times users ran their program between changes varied and multiple test runs were not an unusual co-occurrence.

Discussion

It is important to note that struggle, persistence, engagement, and disengagement exist on a continuum and are related considerably. Users may disengage from an activity for several reasons, but a user cannot persist if they are not first engaged in some form of challenge that they must work through to overcome. Thus, a user must engage and struggle to persist. Conversely, a user who does not persist must disengage after experiencing some form of challenge. If a user disengages from a challenge after struggling to complete it, this may be a signal that some form of intervention is needed to help the user persist. The form of intervention needed for a user with lower detected levels of persistence may or may not resemble the interventions needed for users who persist unproductively. For these reasons, our analysis of log data often refers to markers of struggle with regard to its relation to persistence as we attempt to better understand how young learners persist or fail to persist when facing educational challenges.

Findings from EDA and ONA revealed distinct patterns of behavior among users as they engaged with BarrelBot puzzles. These patterns provided valuable insights into how users approached problem-solving tasks in a virtual learning environment, particularly when considering the complexities introduced by different puzzles. For P1, users faced challenges primarily related to understanding the mechanics of gameplay. This was evident in the event logs, which showed instances of users running and resetting the bot repeatedly before placing code blocks collected from the Shulker Box into BarrelBot. However, due to P1's basic nature, it offered limited insight into deeper problem-solving strategies. In contrast, P2 and P3 revealed more about users' problem-solving processes because they resembled one another as users transitioned from learning to program sequentially to programming iteratively with a loop. Notably, P3 appeared to be the most challenging puzzle in the P1L section, indicating that the use of loops is not the most intuitive computing concept for users to grasp. Detailed log data revealed several users who initially placed the loop code block at the end of their program the first time they used it in BarrelBot, suggesting unfamiliarity with the syntactic organization of code, where loops typically precede the instructions they repeat.

Two basic building strategies emerged from our data collection. Generally, users either built solutions a few code blocks at a time, or they placed several code blocks into BarrelBot and rearranged them as they built toward a solution. Combinations of these strategies were also used, but the frequency of combining these techniques appeared to increase with puzzle complexity. It was never determined if one building strategy was more advantageous than the other, but we observed that several high-performing users, those who appeared to solve P2 and P3 with ease, seemed to favor iteratively adding code blocks over rearranging them.

Most users interacted with BarrelBot without making changes to their program and ran their program between the changes they made as part of their patterned building strategy. Running the program may assist users in making programming choices because it allows users to watch BarrelBot crash. For instance, if the puzzle requires BarrelBot to turn left and move forward but the user programs BarrelBot to turn right before moving forward, the user can see BarrelBot turn in the incorrect direction before crashing. This informs users about what changes to make to the code blocks in BarrelBot. Equally informative, particularly to users who can visualize their program in action without running BarrelBot, are interactions with BarrelBot where users make no changes. Doing so may be an indication that users are peeking inside BarrelBot to look at their existing script. Essentially, this action allows users to toggle back and forth between the existing program in BarrelBot and the puzzle they are working on, which could inform their solution-building choices.

As puzzle complexity increased, peeking inside BarrelBot and running BarrelBot tended to occur more frequently across most users. However, we determined that the number of times a user ran their program unsuccessfully did not clearly indicate struggle or persistence because users who run their program often may be doing so as a part of their solution-building strategy to methodically build and test their program piece by piece. With this strategy, it may take longer to reach a solution, but this does not necessarily mean a user is experiencing struggle. In contrast to this, users who may be struggling may avoid running BarrelBot for testing purposes, which may hinder their progress. Nevertheless, frustration or disengagement may occur when users who were once actively working on a puzzle begin to run BarrelBot excessively between meaningful modifications to their program. Such a change to patterned user behavior in solution building, particularly when a user's progress toward a solution has stagnated, may signify that the user has begun to experience struggle.

Breaks in patterned behavior may also include exiting the puzzle region, but changes in behavior were most evident when users began to remove some or all code blocks from their program, as these events were rare across all users. Code block removal sometimes preceded significant amounts of change resulting in a successful BarrelBot run, but it also led to users failing to complete the puzzles they were attempting. Similar to BarrelBot runs and changeless interactions, instances of code block removal across all users appeared to increase with puzzle complexity. Initial impressions from ONA analysis comparing the behavior of groups of users by the puzzles they attempted or completed suggested that successful users exhibited less variance in their behavior, while less successful users showed more signs of backtracking or stalling co-occurring with unsuccessful runs. These behaviors may have implied that users experiencing difficulty may have been struggling to identify effective problem-solving strategies or were hesitant about making mistakes that might lead to a BarrelBot crash.

EDA following this analysis revealed that users who appeared to solve puzzles with ease behaved differently than the average user in the amount of change they made to their program each time they modified it. The amount of change between solution attempts was quantified using edit distance calculations. Average users tended to make smaller changes between BarrelBot interactions, usually less than four, while high-performing users made much larger changes to their programs each time they interacted with BarrelBot, often much greater than four. Moreover, with nearly every large change they made, the edit distance between their current program and the puzzle's solution decreased by roughly the same amount. Both high-performers and average users were similar in the way they frequently interacted with the BarrelBot without making changes to their script, but high-performing users generally tested their code by running BarrelBot less frequently than the average user, with some only running their program once or twice while building their solution and others running it after each program modification they made.

Regardless of the ease or difficulty users appeared to experience, users across all grouped data occasionally removed code blocks from their programs, particularly on P3. For high-performing users, this behavior change was the most striking due to their highly patterned tendencies in solution building, once again suggesting that changes in patterned problem-solving behavior may be meaningful in the study of persistence. Accordingly, we hypothesize that as the challenge increases for individual users (even high-performing users) established patterns of solution-building may become more variable and less patterned, with the removal of all code blocks, in particular, being a powerful marker of the beginnings of struggle, persistence, or disengagement.

Limitations and Future Work

This analysis has several limitations. First, user data collected on day 1 included instances of both individual and paired students working on the same computer. This could have potentially influenced our results, as collaborative efforts might differ from individual problem-solving approaches. Additionally, onsite interventions that occurred were not recorded in our data set which may have affected our results or complicated the identification of struggle, persistence, or disengagement.

Next, users from the first day of data collection appeared to have run out of time on later puzzles, impacting the reliability of ONA made in our initial groupings. Some of these users appeared to be nearing completion while others may have been exhibiting signs of struggle or disengagement. On day 2, a subset of the first day's participants revisited the P1L section, and they were given more time to successfully complete these puzzles. Consequently, this familiarity may have improved overall user performance and skewed our impressions of solution-building behavior.

In addition to this, time constraints after data collection hindered us from further development of the analysis pipeline to include more puzzles and establish thresholds for significant or minor changes to user solutions between BarrelBot interactions. This hindered our ability to categorize users more appropriately for ONA to more closely examine behavioral nuances. Moreover, ONA limited our ability to detect patterns in solution building. Process Mining might offer a more robust alternative for future studies, as it can better capture the dynamic nature of user engagement or struggle by analyzing patterns that change over time.

Future research should focus on gathering data from child participants to better understand how users struggle and persist in the BarrelBot learning environment. There is also a need to develop clear data collection protocols to prevent data loss and improve the validity of results. Identifying whether students are working alone or receiving on-site assistance or instructions and knowledge of whether student participants are completing puzzles for the first time may also be informative about their behavior.

The analysis pipeline will require further refinement to include all puzzles with static solutions beyond the P1L section and to establish thresholds for significant versus minor changes to users' programs as they build them to identify when users are making progress toward the solution or not. Finally, the time between meaningful events should also be considered, as it may be crucial in identifying patterns indicative of struggle, persistence, or disengagement.

Finally, Process Mining techniques may be better suited than ONA for understanding how different types of students engage with the environment and develop their problem-solving strategies. This approach could help identify patterns in solution building and markers of struggle that occur over time, offering a more comprehensive understanding of student behavior to inform the development of persistence learner models which could lead to the development of support systems that help students stay engaged and persist through challenges in game-based learning environments.

Conclusion

This study employed EDA and ONA to examine student behavior within the WHIMC BarrelBot learning environment, focusing on identifying markers of struggle and persistence in an introductory computing course. Our findings underscore the potential of game-based learning environments to provide valuable insights into student behavior as they engage in problem-solving activities specific to computing education and the development of CT skills.

The preliminary analysis of mock data generated by adult users aided in refining our approach to defining and categorizing engagement in the WHIMC BarrelBot learning environment. This enabled us to clean and process data generated by junior high-aged participants into detailed user logs and binary event logs. Through a cycle of EDA and ONA on these logs, we gained a deeper understanding of their problem-solving behavior and solution-building strategies while attempting to identify general markers of struggle and persistence for young learners. Thus, our analysis serves as an exploratory starting point for future research.

Future research should involve collecting data from a diverse and age-appropriate student population to validate our findings and enhance the understanding of persistence in learning. Additionally, future studies may benefit from employing analytical tools such as Process Mining to gain further insights into the dynamic nature of how students engage, struggle, persist, and disengage in complex problem-solving challenges over time.

Ultimately, this research contributes to a growing body of work on persistence in education aimed to foster more inclusive and supportive learning environments. Research on persistence will aid in the development of adaptive persistence learner models and effective educational interventions that cater to the diverse needs of all students, allowing them to persist in their educational goals and CT skill development.

Acknowledgments

Special thanks to Luc Paquette and Juan Pinto for their input and assistance.

This work was funded in part by the National Science Foundation (DRL-2229612). The opinions, findings, and conclusions do not reflect the views of the funding agencies, cooperating institutions, or other individuals. Any conclusions expressed in this material do not necessarily reflect the views of the NSF.

The author acknowledges the assistance of ChatGPT, a large language model, in the editing and proofreading process of this manuscript.

References

Israel-Fishelson, R., & Hershkovitz, A. (2020). Persistence in a Game-Based Learning Environment: The Case of Elementary School Students Learning Computational Thinking. Journal of Educational Computing Research, 58(5), 891-918. https://doi.org/10.1177/0735633119887187

Israel-Fishelson R., & Hershkovitz A. (2021) Micro-persistence and difficulty in a game-based learning environment for computational thinking acquisition. Journal of Computer Assisted Learning, 37, 839–850. https://doi.org/10.1111/jcal.12527

Owen, V. & Roy, Marie-Helene & Thai, Kp & Burnett, Vesper & Jacobs, Daniel & Keylor, Eric & Baker, Ryan. (2019). Detecting Wheel-spinning and Productive Persistence in Educational Games.

Tan, Y., Ruis, A. R., Marquart, C., Cai, Z., Knowles, M. A., & Shaffer, D. W. (2022, October). Ordered network analysis. In *International Conference on Quantitative Ethnography* (pp. 101-116). Cham: Springer Nature Switzerland.