Autonomy-Supportive Game Benefits Both Inexperienced and Experienced Programmers*

Michael J. Lee and Ruiqi Shen
Department of Informatics
New Jersey Institute of Technology
Newark, NJ 07102
{mjlee, rs858}@njit.edu

Abstract

As more people turn to discretionary online tools to learn new skills such as computer programming, exploring how to better support a wide range of learners is becoming increasingly essential to train the next generation of highly skilled technology workers. In our prior work, users with high learner autonomy complained that most online resources they used to learn more programming did not provide them with the flexibility they preferred to navigate through learning materials, locking them into a set sequence of topics/concepts. To explore this, we implemented a *level-jumping* feature into an online educational programming game. We tested it with 350 new users, tracking their progress through the game for 7 days each. We found that those with high learner autonomy did use the level jumping feature more than those with low learner autonomy. We also found that males were more likely to use this new feature, regardless of learner autonomy level, compared to their female counterparts. Finally, we found that those with low learner autonomy ultimately completed more levels than their high autonomy counterparts, and that this was particularly true of female learners (who completed the most levels overall). Based on these findings, we believe that autonomous-supportive features such as flexible navigation may be beneficial to all users of online educational tools, and that encouraging its use by a wider group of users (particularly females), may increase positive effects.

1 Introduction and Related Work

With the continued influx of individuals turning to discretionary online learning learning resources such as Massive Open Online Courses (MOOCs), tutorial sites, and educational games, learning more about how to support a wide range of people with different preferences, experience, and learner autonomy, is essential to train the next generation of highly skilled technology workers. According to educators and researchers, learner autonomy—the ability for one to control their own learning [6]—plays a vital role in developing lifelong learners [3]. In order to gain a better understanding of their autonomy, our past work explored the experience of Computer Science (CS) learners', particularly about learning CS both online and in the classroom [14, 17]. We found that learners who showed a *high level* of autonomy felt that they were not supported by the educational system(s) or teachers. For example, they complained that the online systems they used did not give them enough freedom to explore on their own, and that their teachers often failed to help them achieve their learning goals. On the other hand, we found that learners who showed a *low level* of autonomy felt that they needed extra guidance from their teachers and curriculum. Interview results also indicated some patterns, for example, that learners with more subject-area experience showed higher levels of autonomy than those with less experience, and learners with a higher level of autonomy preferred to study using an autonomy-supportive system while those with lower levels of autonomy preferred to study using a non-autonomy supportive system [13, 17].

Considering these past results and the role learner autonomy plays in developing lifelong learning, we believe that it is important to address the needs of CS learners with different levels of learner autonomy. In particular, we found in prior work [13, 16] that autonomy-supportive features (i.e., the freedom to freely navigate to any portion of a course's curriculum) in learning systems were consistently requested by those with high learner autonomy. This was in contrast to those with low learner autonomy, who preferred much more structured and linear pathways through a curriculum. In order to test how an autonomy-supportive feature might affect learners (based primarily their level of learning autonomy), we implemented a *level-jumping* feature into an online educational programming game (see Figure 1). This is in contrast to most online learning curriculums and MOOCs that we have encountered, which are often locked to a specific sequence where later parts of the course are inaccessible until earlier parts are completed. We tested the game with this new level jumping feature with 350 new users, tracking their progress through the game for one week (7 days) each, spanning a total of 1.5 months.

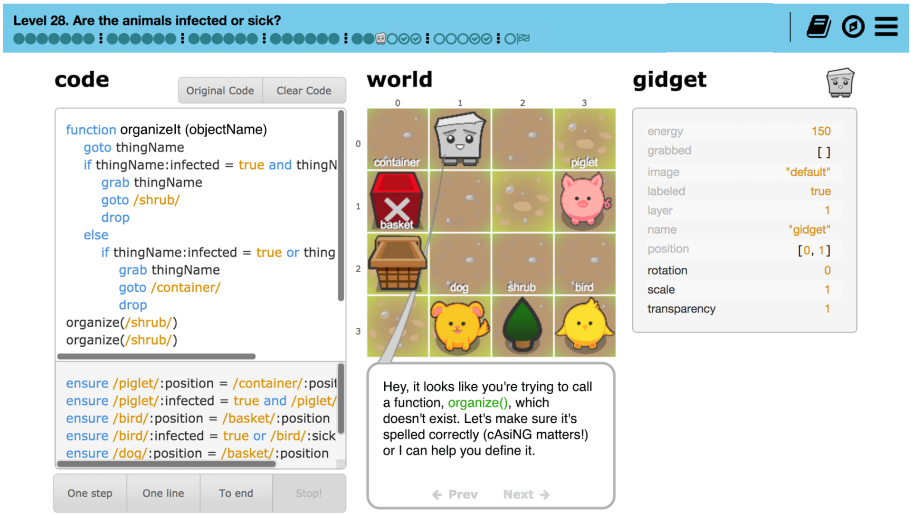

Figure 1: *A screenshot of the Gidget introductory programming game.*

2 Method

2.1 The Gidget Educational Game

We modified our free introductory coding game, Gidget *(www.helpgidget.org)*, for this study. Gidget has a total of 37 core levels in its curriculum, where each level teaches a new programming concept (e.g., variable assignment, conditionals, loops, functions, objects) using a Python-like, imperative language [8, 10]. The objective of each level is to fix existing code to help the game's protagonist pass 1–4 test cases (i.e., statements that evaluate to `true`) after running the code. Each level introduces at least one new programming concept, becoming progressively difficult in subsequent levels. Therefore, users are exposed to more programming concepts the farther they progress through the game. Finally, the game also includes a set of help features to help players overcome obstacles while coding on their own [7, 10]. This includes a frustration detector that provides encouraging hints/messages to those that are struggling with a level [9], and also auto-generates additional levels covering the same concept(s) to provide additional practice [8].

Normally, the game follows a specific order of levels (i.e., curriculum), building on content from previous levels. While the user interface shows the sequence/map of all core levels in the game (and indicating the players' current level; see top of Figure 1), it only allows the player to jump back to any previously completed level (at any time during game play). Players can also jump

Figure 2: *Closeup of the level selection map. The current level shows Gidget, completed levels are solid circles, uncompleted levels are hollow circles, and uncompleted exam levels are hollow circles with a check mark.*

forward to the last level they have reached sequentially, but no further. All levels are visualized on the map as circles, with completed levels shown as solid circles, incomplete levels shown as hollow circles, and incomplete exam levels (explained in [10]) shown as hollow circles with a check mark. Finally, the currently loaded level is indicated with the Gidget character (see Figure 2). Hovering the mouse cursor over an incomplete level does not show any visual change, and clicking on an incomplete level does not do anything.

For this study, we modified the level-selection interface to allow players to jump to any level in the game, regardless of level completion status. Placing the mouse cursor on any other level grays out the current level's Gidget character and places a solid Gidget character that slowly rocks back-and-forth on that level marker. Clicking on the rocking character immediately jumps the player to that level. In addition, to keep the overall experience consistent across all users of this study, we disabled the game's auto-generated extra levels (as described in [8]). This was to prevent cases where someone might jump to a difficult level, trigger the frustration detector, then offered multiple additional practice levels covering the same concept(s). Finally, we specifically pointed out this level-jumping feature in the game's introductory on-boarding tutorial (which all players see the first time they load the game), explaining the user interface, interaction method, and the level-jumping feature.

2.2 Participant Recruitment

Our goal was to observe if and how players would use the level-jumping features within the game. We evaluated our system with a group of 350 new users of the game. The sign-up screen asked users for their age, gender, e-mail address, a checkbox indicating whether they have prior programming experience, and a checkbox (with link to consent form) asking if they were willing to participate in a research experiment. We intentionally did not define "programming" or "programming experience" as we determined in past studies [14, 15] that using a specific definition could potentially confuse or discourage participants who might consequently miscategorize themselves or self-select out of participation even though they meet our eligibility criteria. Mirroring a previous study [14], we asked those who indicated that they had prior programming experience two additional questions: how many years of programming experience they had (rounded up to the nearest .5 or integer), and how they would rate their

programming experience level on a four-level scale (beginner, intermediate, advanced, and professional). We used these two measures to assign each player a learner autonomy score from 1 (low learner autonomy) to 3 (high learner autonomy) based on our prior work [14], which showed that these two measures were significantly correlated with learner autonomy. This prior study combined subsets of the Learner Autonomy Scale created by Macaskill and Taylor [11] and the E-learning Autonomy Scale developed by Firat [5], and demonstrated that more years of experience and higher self-rating in programming experience has a positive relationship with autonomy level.

For this study, we only selected users that indicated they were 18+ years and willing to participate in a research experiment. Adapting the methodology from our prior studies [8, 9], we set the observation time to 7 days (168 hours) per user to have a consistent evaluation window for all users. To promote quick account creation, we did not collect other demographic information such as ethnicity, geographical location, or education level. Participants were required to read and digitally sign an online consent form that briefly described the study. We were intentionally vague in our description of the level-jumping feature, stating that we were "testing new navigational features" to minimize potential leading or biasing of participants to pay attention more to that specific part of the interface. However, we debriefed all participants of the study procedures 7 days after the end of their individual observation window, by e-mail.

3 Results & Discussion

We report on our quantitative results comparing our participants' outcomes—split by demographic and experience features—using nonparametric Wilcoxon rank sums tests, Chi-Squared tests, or simple linear regression, with a confidence of $\alpha = 0.05$, as our our data were not normally distributed. For all post-hoc analyses regarding gender data, we use the Bonferroni correction for three comparisons: ($\alpha = .05/3 = 0.0167$).

The study included 350 participants (aged 18–58; median 20). As a whole, our participants were composed of 180 females (51.4%), 161 males (46%), and 9 'not listed' or 'decline to state' (2.6%). In addition, 255 (72.9%) indicated that they did not have any prior programming experience, and 95 (27.1%) indicated that they had at least .5 years of prior programming experience (latter's range .5–33; median 2). We operationalized our key dependent variables, *engagement* and *jumping*, as the number of levels completed and the number of times the jumping feature was used, respectively.

3.1 High Learner Autonomy Players Use the Jumping Feature More

We found that all learners used the level jumping feature at least 2 times, regardless of having low learner autonomy (range 2-17; median 3) or high learner autonomy (range 2-37; median 8). Looking at the data more closely, we found that there was a significant difference in the number of levels participants completed by autonomy level ($W = 26703, Z = 12.370, p < .05$), with the high autonomy learners using the feature more than their counterparts.

We believe that all learners jumped at least two times because this feature was specifically mentioned in the on-boarding game tutorial, and at the minimum, someone using the feature to jump forward (first jump), would need to jump back to their original level (second jump). Next, our finding that high autonomy learners use the jumping feature more often than their low autonomy counterparts verifies our hypothesis (based on our previous work in [14]) that those with more experience (and therefore higher learner autonomy) would use and benefit from this jumping feature. Unlike low autonomy (inexperienced) learners, who do not necessarily know much about the topic and therefore would be better served learning programming concepts in a sequenced curriculum, the goal of high autonomy (experienced) learners may be to review or improve on their existing programming skills, and/or to look for programming resources. Therefore, they may be more likely to use the jumping feature to browse through the different parts of the curriculum quickly, being more in control of their learning.

3.2 Males Use the Jumping Feature More

We found a significant difference in usage of the jumping feature by gender (χ^2(2, N=350)=17.226, p<.05). Doing post-hoc analysis with the Bonferroni correction, we found that males used the jumping feature significantly more overall than their female counterparts (W=42.307,Z=4.109,p<.05/3). This result was independent of low learner autonomy (χ^2(2,N=255)=6.1464, p<.05) or high learner autonomy (χ^2(2,N=95)=6.1583, p<.05) in programming.

This result was not too surprising, as prior research [2] has shown that compared to females, males are statistically more likely to use selective information styles (following the first promising information, then potentially backtracking) [12], have lower risk aversion (be less wary of consequences) [4], and more willing to tinker (playfully experiment) [1]. Based on this, we believe that our male players were more likely to use the jumping feature simply because it was available in the interface (and also mentioned in the tutorial).

3.3 Low Autonomy (Female) Learners Complete More Levels

Next, we explored if there was a difference in the number of levels participants completed. This is not a completely fair comparison, as everyone may have encountered levels in a different sequence (with later levels being considerably more difficulty than earlier levels) because of the jumping feature.

We found that low autonomy learners completed significantly more levels compared to their high autonomy counterparts ($W = 15002.5, Z = -1.987, p < .05$). Further analysis revealed that there was a significant different in the number of levels completed by gender within the low autonomy group ($\chi^2(2, N=255)=43.3806$, $p<.05$). A post-hoc analysis with the Bonferroni correction showed that the low autonomy group females completed significantly more levels compared to their male counterparts (W=-61.579,Z=-6.655,p<.05/3). We calculated a simple linear regression to predict level completion based on jumping behavior. Within the low autonomy group, we found a positive relationship between these variables ($F(1, 253) = 255.290, p < .05$), $R^2 = .502$). Examining this more closely, we found that that this affect was strongest with females, where females in the low autonomy group who jumped more often completed more levels ($F(1, 144) = 206.433, p < .05$), $R^2 = .589$).

This result supports our hypothesis discussed in Section 3.1. The goal of high autonomy (experienced) learners may be to review or improve on their existing programming skills, and/or to look for programming resources. If high autonomy learners were using the level jumping feature primarily to explore what programming concepts the game curriculum covered, it would explain why they did not necessarily stay to solve/complete those levels. On the other hand, a low autonomy (inexperienced) learner's aim in playing a programming game is more likely to learn new things, and most or all of the programming concepts would be new to them. Therefore, whether or not they jump through the curriculum, less experienced learners have more incentive to complete levels. Perhaps those low autonomy learners that jump around the levels have a better idea of what is coming next (and also gain additional insights from the broken, starting code each level provides), and therefore more successful in completing levels. Most surprisingly, although our female participants were most likely not to use the jumping feature, those that did went on to be the individuals that completed most (or all) of the game levels. Females who did decide to use the jumping feature may have jumped back and forth between levels as a comprehensive information processing problem-solving strategy [2, 12], where they used the jumping feature to preview what was coming up, thereby gathering fairly complete information about the entire system before proceeding.

4 Conclusion

Our findings show that both high and low autonomy learners (particularly males), used the level-jumping feature, with the former using this feature significantly more than the latter. We also found that high autonomy learners tend not to complete the levels they jump to, and that they complete significantly fewer levels overall compared to their low autonomy counterparts. We also found that the few low autonomy female learners who used the jumping feature readily, also ended up completing more levels than any other group. Designers for online resources teaching programming may benefit from allowing all users to skip around and explore the curriculum, instead of locking them into a specific sequence. They may also do well in encouraging more of their learners (especially females) to use these types of jumping features to have them preview and better prepare for what is coming later in the curriculum.

We have several limitations to our study. We recruited participants who opted into a research study while signing up for an educational game. These participants may already have high motivation, and therefore may not be completely representative of the larger population. Next, we asked participants to self-report their years of programming experience and also to rate their own programming expertise. Participants may have different criterion for these selections and therefore may have led to inconsistencies in our user groupings. The groupings themselves may not account for all the different nuances of experience and/or learner autonomy. For future work, we could use more objective measures such as quizzes to test the skill level of participants as an alternative measure to experience. In addition, we could use pre-post tests to measure how this new jumping feature affects players' learning outcomes, and collect qualitative data from participants through questionnaire or interviews.

Our study results show that both high autonomy and low autonomy learners use the level-jumping feature (presumably to preview levels), and that although low autonomy users are less likely to utilize this feature, those that do are especially successful in completing more levels (particularly females). Our future work will examine these outcomes in more detail, and gather complementary qualitative data, to isolate the features that are causing these effects.

5 Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under grants DRL-1837489 and IIS-1657160. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the NSF or other parties.

References

[1] Laura Beckwith, Cory Kissinger, Margaret Burnett, Susan Wiedenbeck, Joseph Lawrance, Alan Blackwell, and Curtis Cook. Tinkering and gender in end-user programmers' debugging. In *ACM CHI*, pages 231–240, 2006.

[2] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Anicia Peters, and William Jernigan. Gendermag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.

[3] Philip C Candy. *Self-Direction for Lifelong Learning. A Comprehensive Guide to Theory and Practice.* Eric, 1991.

[4] Thomas Dohmen, Armin Falk, David Huffman, Uwe Sunde, Jürgen Schupp, and Gert G Wagner. Individual risk attitudes: Measurement, determinants, and behavioral consequences. *J. of the European Econ. Assoc.*, 9(3):522–550, 2011.

[5] Mehmet Firat. Measuring the e-learning autonomy of distance education students. *Open Praxis*, 8(3):191–201, 2016.

[6] Henri Holec. *Autonomy and foreign language learning.* Eric, 1979.

[7] Michael J Lee. How can a social debugging game effectively teach computer programming concepts? In *ACM ICER*, pages 181–182, 2013.

[8] Michael J Lee. Auto-generated game levels increase novice programmers' engagement. *The Journal of Computing Sciences in Colleges*, 36(3), 2020.

[9] Michael J Lee. (re)engaging novice online learners in an educational programming game. *The Journal of Computing Sciences in Colleges*, 35(8), 2020.

[10] Michael J Lee, Amy J Ko, and Irwin Kwan. In-game assessments increase novice programmers' engagement and level completion speed. In *ACM ICER*, 2013.

[11] Ann Macaskill and Elissa Taylor. The development of a brief measure of learner autonomy in univ. students. *Studies in Higher Education*, 35(3):351–359, 2010.

[12] Joan Meyers-Levy and Barbara Loken. Revisiting gender differences: What we know and what lies ahead. *J. of Consumer Psychology*, 25(1):129–149, 2015.

[13] Ruiqi Shen. Interactive computer tutors as a programming educator: Improving learners' experiences. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–2. IEEE, 2020.

[14] Ruiqi Shen, Joseph Chiou, and Michael J Lee. Becoming lifelong learners: Cs learners' autonomy. *J. of Computing Sciences in Colleges*, 35(8):267–267, 2020.

[15] Ruiqi Shen and Michael J Lee. Learners' perspectives on learning programming from interactive computer tutors in a mooc. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–5. IEEE, 2020.

[16] Ruiqi Shen, Donghee Wohn, and Michael Lee. Programming learners' perceptions of interactive computer tutors and human teachers. *International Journal of Emerging Technologies in Learning (iJET)*, 15(9):123–142, 2020.

[17] Ruiqi Shen, Donghee Yvette Wohn, and Michael J Lee. Comparison of learning programming between interactive computer tutors and human teachers. In *ACM Conference on Global Computing Education (CompEd)*, pages 2–8, 2019.