Engaging Programming Students Through Simpler User Interfaces

Blessing Leonard and Giovanni Vincenti(✉)

University of Baltimore, Baltimore, MD, USA
{blessing.leonard,gvincenti}@ubalt.edu

Abstract. We used line_explorer, a novel programming instructional tool to administer tests, which involved assigning a math problem based on a programming concept (loops) and the accompanying instructions for solving it to students. There were two display options for the instructions: the first provided all instructional information at once, while the other displayed the instructional information in phases. Findings from the study confirmed that an interface can affect the comfort level of the user, and thus can influence the effectiveness of the tool. Results from the research will aid in identifying parameters that need to be improved or adapted to line_explorer and other programming instructional tools, to encourage better comprehension of programming concepts, as well as improve the willingness of students to use these instructional tools.

Keywords: Human-centered design · Cognitive load · CS education

1 Introduction

Background. Human-centered design can be defined as the process of involving basic human functionalities in the design process, rather than waiting until the product is completed to force users to act according to its will. This approach ensures the users have a pleasant first experience with the program and thus are willing to keep using it. line_explorer aims to provide a comfortable user interface that will help reduce the discomfort associated with programming by allowing "the user to act as the code compiler and step through each line of code seeing the logic flow and what values get assigned to variables in the program." [1].

Findings from a line_explorer pre-design survey [2] suggests that although the average student is not excited about programming, they are able to get the work done; and while they were fine with accessing instructional videos on their mobile devices, students preferred computer-based platforms to mobile-based ones for interactive instructional tools. line_explorer is expected to function as a support tool for instructors of beginner to intermediate courses, and for students to review the material on their own before coming to class [1], therefore effectively creating an interactive support system that can be used to implement the flipped classroom model [3] in computer programming courses.

Project Goals. The goal of this research was to identify the impact (if any), of different amounts of information displayed to students as they work through a programming concept. Given that this is the second phase of testing line_explorer, some features were

W. Karwowski et al. (Eds.): AHFE 2019, AISC 963, pp. 113–121, 2020.
https://doi.org/10.1007/978-3-030-20135-7_11

changed and tested against the initial version to look for any differences (positive or negative) in the reception of the programming tool. The hypothesis was that a reduction in the amount of information displayed to the student at the launch of the program would positively affect their reception of its usefulness. It should be noted that this research focused solely on line_explorer's evaluation mode. To help decide the kind of changes necessary to test the hypothesis, some research was done on the effect of cognitive load on interfaces and instructional tools. A demonstration mode—used for instructional purposes—also exists. This mode is discussed towards the end of the manuscript.

2 Cognitive Load

Oviatt defines cognitive load as the "mental resources a person has available for solving problems or completing tasks at a given time." [4]. Including cognitive load theory in the designing phase ensures the creation of interfaces that will decrease the amount of resources used on peripheral processing, so that the bulk of the mental resources used will be directed towards the main task at hand.

The first concept that was at the core of the redesign process was the need to provide only the number of features required to get the work done, as unnecessary features often act as a form of distraction [4]. For instance, the presence of a multi-colored pen on an interface where the color of the pen does not affect the result of the task at hand, can lead to the mind being occupied with figuring out: how many colors there are, which colors it likes best, what each color means, if and how the functionality of the cursor changes depending on its color, etc. before even trying to focus on the task itself. In this case, there is a high cognitive load as the working memory is fully engaged, but not because of the work at hand.

The second element that drove this project's redesign is the need to keep the interface of the new tool as close as possible to that which the students (users) are already familiar with [4]. The importance of not jarring the students with a completely different interface is to prevent the distraction of processing the difference(s) between what they are used to, and the new design before them. Hence, making it harder to concentrate on the main task, and resulting in a negative toll on the time and quality of the task completion.

The article also suggested that users' ability to communicate multimodally be considered [4]. For instance, an audio-visual or verbal-visual demanding tool will be easier to use compared to an audio-verbal demanding tool. This is because the same processor manages the audio and verbal mental resources, while a different processor in the working memory manages the visual demands. Thus, cross modality was advocated for over intra-modality.

3 Methodology

To test the hypothesis that less information leads to more productivity, we created a modified version of line_explorer, and tested it by comparing the users' performance and reactions with the preexisting version. Figure 1 shows the original interface

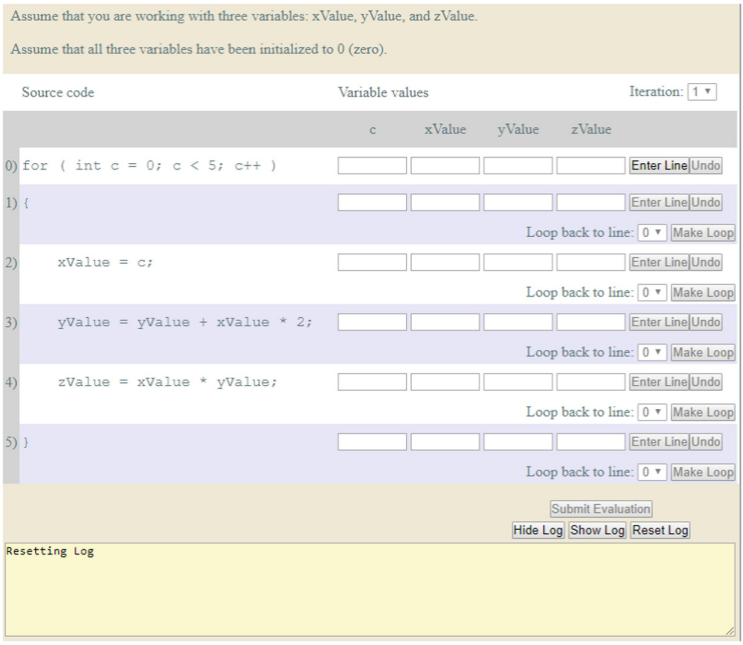

Fig. 1. Original interface used for the control group (Interface A).

(Interface A) while Fig. 2 shows the modified version (Interface B). The new interface focused on increasing the amount of information displayed based on the user's progress in solving the given problem. Both interfaces were preloaded with the same concept: a for loop requiring five iterations and whose statements were a set of instructions for solving the math problem contained in the loop. The test required basic programming and math skills—such as addition and multiplication—to work through the example.

The revised model removed some features completely, starting from the progress log and its controls (at the bottom of the screen). This feature's original purpose was to give students a way to trace their work, but users did not utilize it at all in previous tests [5]. Then, we moved the Make Loop function from every line to the last line of each iteration. The original purpose of this feature was to test whether the students would know when a new iteration of the loop would begin. In previous tests users did not use this feature as well, other than when appropriate [5].

The 59 test subjects were students enrolled in three courses within the Applied Information Technology program at the University of Baltimore: COSC 151 (Computer Programming I), COSC 251 (Computer Programming II), and COSC 351 (Object-Oriented Programming). The first two courses are modeled after typical CS1/CS2 courses, and the last course covers up to linear data structures. All classes include a lecture and in-class coding practice, and students participated in this experiment after loops were introduced. Participants were enrolled in the following academic programs: Applied Information Technology (AIT): 24 students, Digital Communication (DCOM): one student, Psychology (PSYC): two students, Forensic Science (FSCS): two students,

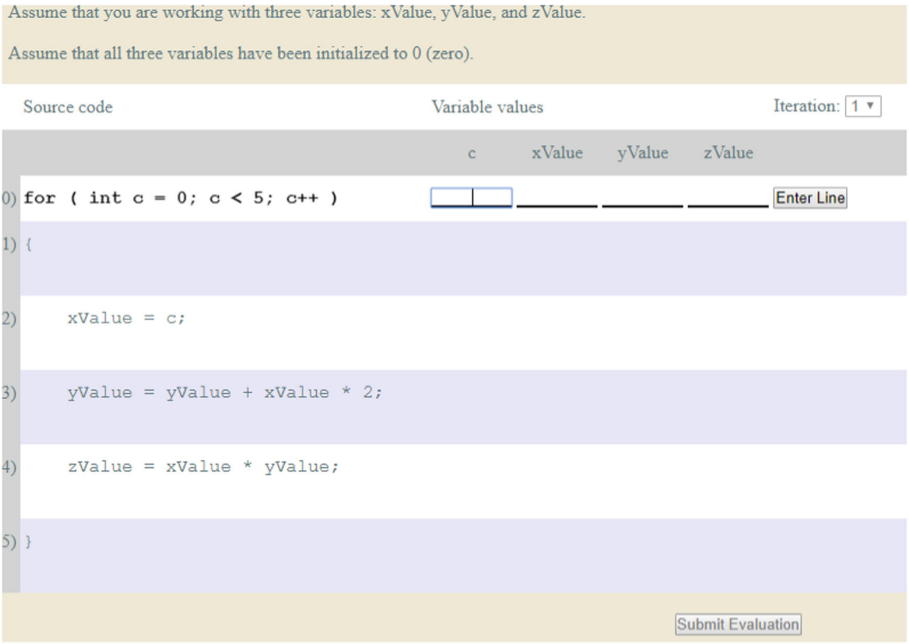

Fig. 2. Modified interface used for the experimental group (Interface B).

Information Science/Systems (INSS): six students, Business Administration (BUAD): one student, Simulation and Game Design (SGD): 19 students, Information and Interaction Design (IID): one student, and three unknown majors.

Prior to the commencement of the test, each participant received an access code after a verbal explanation of the purpose of line_explorer, the reason for the test, and the need to go through five iterations of the loop. The test experience included a questionnaire, an instructional video that went through the first iteration of the tested interface (based on the access code entered), the test itself, an instructional video of the untested interface, and finally a poll to vote on the preferred interface and provide feedback on the experience and line_explorer. The tests looked to measure the amount of time taken to complete the test, the accuracy of the final answer, the interface with the highest votes, and the impact of factors such as major, experience, and comfort level on these measurements.

4 Results and Discussion

For COSC151, three voted for Interface A, and out of the three, two rated themselves as having mid-level programming experience (rated at 3 out of 5) and low comfort level (rated at 2 out of 5) while the third had low experience (rated at 1 out of 5) with a slightly higher comfort level (rated at 2 out of 5). Major did not influence decisions, as

the majors were diverse. However, two INSS majors preferred Interface A—one of them was the participant with the lower experience. There were 20 total entries, five of them did not report a preference, and the remaining 12 were for Interface B, as reported in Table 1.

Table 1. UI preference by major for the course COSC 151.

Major	Interface A	Interface B	No answer
AIT	1	5	1
DCOM	0	1	0
PSYC	0	1	1
FSCS	0	1	1
Unknown	0	1	0
INSS	2	1	1
BUAD	0	1	0
SGD	0	1	1

The comfort and experience rating range for those who voted for Interface B was between low and mid-level, but for the two exceptions of higher comfort ratings (averaging around 4). Table 2 provides a visual representation of the average experience and comfort level ratings per major for COSC 151.

Table 2. Average experience and comfort level ratings by major for the course COSC 151 on a scale of 1 (low) to 5 (high).

Major	Experience	Comfort
AIT	2.29	2.86
DCOM	3	2
PSYC	1.5	1.5
FSCS	1	1
Unknown	1	3
INSS	2.2	2.2
BUAD	1	3
SGD	1	1

For COSC251, 6 out of 17 voted for Interface A, one did not vote, and the remaining 10 voted for Interface B (Table 3). This group consisted mainly of Applied Information Technology (AIT) majors (12 students), three Simulation and Game Design (SGD), one Information and Interaction Design, and one unspecified major.

Those that voted for Interface A had mid to high experience and comfort levels (3-5 for experience, and up to 4 for comfort level) except one with a rating of 2 for both. All SGD majors voted for Interface B. The highest experience and comfort level (5 for both) was an Information and Interaction Design major and the participant voted for

Table 3. UI preference by major for the course COSC 251.

Major	Interface A	Interface B	No answer
AIT	5	6	1
Unknown	1	0	0
IID	0	1	0
SGD	0	3	0

Table 4. Average experience and comfort level ratings by major for the course COSC 251 on a scale of 1 (low) to 5 (high).

Major	Experience	Comfort
AIT	2.75	2.75
Unknown	4	4
IID	5	5
SGD	2.33	3.33

Interface B. All but the unknown major that voted for interface A were AIT majors. Table 4 provides a visual representation of the average experience and comfort level ratings per major for COSC 251.

Out of the 22 participants from COSC 351, four voted for Interface A, 15 for Interface B, and three did not vote, as reported in Table 5. All four that voted for Interface A tested Interface B, two of the nonvoters tested Interface B, and 11 of those who voted for Interface B tested Interface A. SGD made up most of the class and had the highest acceptance rate (11 votes) for Interface B (save for INSS' 100% acceptance rate from their two votes). Three participants successfully completed the task and all tested Interface A but voted for Interface B. The recurring reasons for selecting Interface B over A were the fact that it was "clean", less intimidating, and less distracting.

Table 5. UI preference by major for course COSC 351.

Major	Interface A	Interface B	No answer
AIT	2	2	1
INSS	0	2	0
SGD	2	11	1
Unknown	0	0	1

Only one of the eight (overall) successful candidates voted for Interface A, thereby supporting the finding that students tend to deliver expected products regardless of circumstances but are open to new and more relatable processes of learning programming concepts. Five out of these eight participants gave themselves higher ratings

for comfort while their experience ratings fluctuated between low and high, and participants from COSC 251 and 351 seemed to spend substantially more time trying to solve the problem than those from COSC 151. The average, minimum, and maximum time spent—in seconds—on problem solving for each of the three courses are reported in Table 6.

Table 6. Time spent (in seconds) on problem-solving by course.

Course	Average	Minimum	Maximum
COSC 151	141.19	41	389
COSC 251	379.82	103	737
COSC 351	256.58	29	622

Although the results slightly derailed from the initial hypothesis (the impact of interface layout on the effectiveness of line_explorer due to cognitive load), it was able to confirm that the interface affects the comfort level of the user, and thus can influence the effectiveness of the tool. These findings, as shown in Table 7 also suggest the provision of more information (as seen in Interface A) for intermediate users, and less information (as seen in Interface B) for novices and users that are more proficient with basic programming knowledge.

Table 7. UI preference by course.

Class	Interface A	Interface B	No answer
COSC 151	3	12	5
COSC 251	6	10	1
COSC 351	4	15	3
Total	13	37	9

In the suggested improvements section of the post-test questionnaire, students provided the following suggestions for improving line-explorer:

- Ability to shuffle between iterations;
- Removing interactivity components from lines that do not contain instructions, such as lines with opening or closing braces;
- Restricting the effects of the undo button to a line—so that correcting an error on a previous line does not erase all progress made after that line;
- Clearer instructions; and
- Creating a function to allow the review of instructional videos while working on the problem.

5 Demonstration Mode

The Demonstration mode, reported in Fig. 3, allows students to practice interpreting code by giving them the option to enter expected values contained in variables for each line. The overall look is very much in line with what we already described in the Evaluation mode, with the instructions panel above the work area, lines of code clearly labeled and easily readable, and a series of input boxes where students can enter expected values for each variable.

Fig. 3. The updated interface of line_explorer in Demonstration mode.

An important element focuses on teaching students about the scoping of variables. On the right side of the panel, users will be able to enter an input only for variables that actually exist. This should allow users to visualize how scoping works, and when variables exist or do not exist. Since the main goal of Demonstration mode is to allow students to explore lines, and people have different learning preferences, we have added two "Help" features, shown in Fig. 4, which the user can toggle at any point.

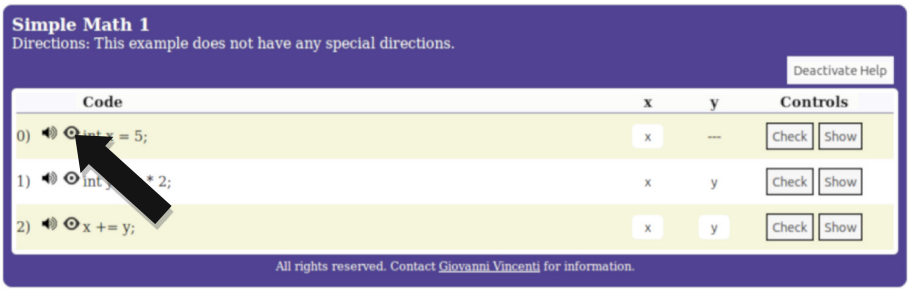

Fig. 4. Help features available in Demonstration mode.

The icons that appear are of a sound speaker and a stylized eye. By pressing the sound speaker, the system will utilize the voice synthesizer available in the browser to vocalize the verbal description of each line. By selecting the eye icon, the system will instead display the same verbal description on a screen overlay, shown in Fig. 5.

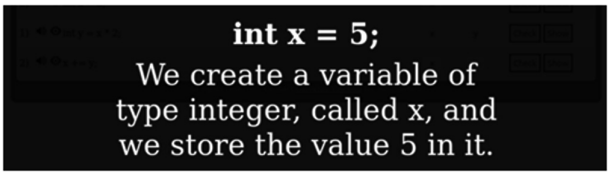

Fig. 5. Visual description of a line of code that the students can access through the Help feature available in Demonstration mode.

As we intend to keep both visual and functional consistencies between the two modes, we will apply the changes reported to the Demonstration mode to the Evaluation one.

6 Conclusions

Since all participants that provided the correct final answer were from the COSC 251 and 351 pools, we can infer that problem-solving progress and final answer accuracy depend more on experience and familiarity than on cognitive load. A sense of comfort on the part of the user is also projected to play a major role in the effectiveness of this tool, perhaps even more than experience; and based on time spent on the assigned task, it can be inferred that experience and comfort level also play a role in the user's desire to work with line_explorer. As testing and data analysis continues, we expect to build future work on findings and feedback from this and previous testing results.

References

1. Vincenti, G., Braman, J., Hilberg, J.S.: Teaching introductory programming through reusable learning objects: a pilot study. J. Comput. Sci. Coll. **28**(3), 38–45 (2013)
2. Vincenti, G., Hilberg, J.S., Braman, J.: Student preferences and concerns about supplemental instructional material in CS0/CS1/CS2 courses. Int. J. E-Learn. **16**(4), 417–441 (2017)
3. Tucker, B.: The flipped classroom. Educ. Next **12**(1), 82–83 (2012)
4. Oviatt, S.: Human-Centered Design Meets Cognitive Load Theory: Designing Interfaces that Help People Think, Journal of the Impossible (2006). https://dl.acm.org/citation.cfm?id=1180831. Accessed 21 Feb 2018
5. Vincenti, G., Hilberg, J.S., Braman, J., Satzinger, M., Cao, L.: Assessing the Usability of a Novel System for Programming Education. https://arxiv.org/abs/1711.05649. Accessed 13 Feb 2018