2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #33409

A Comparison of Novice Coders' Approaches to Reading Code: An Eye-tracking Study

Dr. Geoffrey L. Herman, University of Illinois at Urbana-Champaign

Dr. Geoffrey L. Herman is a teaching associate professor with the Deprartment of Computer Science at the University of Illinois at Urbana-Champaign. He also has a courtesy appointment as a research assistant professor with the Department of Curriculum & Instruction. He earned his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign as a Mavis Future Faculty Fellow and conducted postdoctoral research with Ruth Streveler in the School of Engineering Education at Purdue University. His research interests include creating systems for sustainable improvement in engineering education, conceptual change and development in engineering students, and change in faculty beliefs about teaching and learning. He is a member of the Computing Research Association - Education steering commitee.

Sofia Meyers, University of Illinois at Urbana-Champaign
Miss Sarah-Elizabeth Deshaies

Introduction and Literature Review

Considerable effort has been spent documenting students' misconceptions and difficulties when learning to program, but much of this research has focused only on students' difficulties and only after some formal instruction [1], [2]. Constructivist theories encourage us to consider how we can help students construct their own knowledge with the experience and knowledge they bring to our classrooms [3]. The overarching research project had many goals, however, this paper will focus on just two main data-driven goals. The first is to understand what productive knowledge *experienced students* (students who had taken at least three programming courses) bring to code reading that *complete novices* (students who have never programmed or studied programming before) may not have yet as measured through code-reading accuracy, total reading times, and fixation heatmaps. The second goal is to explore whether beacons (i.e., information rich portions of code) can be identified in eye-tracking studies when participants are tracing code.

Eye tracking

Eye-tracking (using cameras to track the focal point of an individual's vision) has been used to explore the cognitive processes involved in reading for over 40 years and has recently expanded in scope from “traditional" reading to reading code. Reading can be broken down into two distinct behaviors, fixations (moments where the eyes are relatively still and visual information is obtained) and saccades (moments when the eyes move and no information can be obtained by the reader). There are several types of time measures used within traditional eye-tracking studies. One such measure is total reading time, this can be defined as the sum of the fixation durations, or the total time it takes to read a sentence (or an area of interest). Often this is the simplest measure to obtain and relates to the sum of the amount of time it takes to process information and the attention given to the currently fixated object [4].

As individuals gain knowledge in a domain, their perception of domain problems also change [5]. For example, chess experts see strategic collections of pieces rather than individual pieces and physics experts focus on physics relevant details (i.e., is an object sliding or rolling) rather than geometric shapes (i.e., is an object a ball or a box) [3]. Eye-tracking provides a great moment-to-moment observation of these changes in perception and can reveal where experts direct their attention [4].

For example, Madsen and colleagues illustrated the trajectory of balls by drawing the same ball at different points in time at different points in space [6]. When asked to analyze the diagram to answer a question about the ball's velocity, novices fixated on the absolute position of the ball while experts fixated on the spacing between balls. This difference reveals a deeper conceptual understanding of physics, because the spacing between balls indicates the change in absolute position over a period of time (i.e., velocity) [6]. Similar differences between experts and novices have been found in other domains. The finding is particularly striking because white space normally encodes a lack of information and is easily ignored, but the experts found empty space more interesting and information rich than drawings of the ball.

In computer science, eye-tracking studies have revealed similar differences between experts and novices. Crosby, Scholtz, and Wiedenbeck found that experts were more likely to fixate on beacons—information rich lines of code (e.g., function headers or common coding conventions such as finding the middle element of an array) [7]. Similarly, prior studies have found that expert programmers may take a longer time scanning a program to comprehend it, but then focus on problem relevant portions of the code once they are identified. Crosby and colleagues also found that experts focused on beacons longer than novices even though they needed less time to initially comprehend the meaning and significance of those beacons [7]. They also found that more meaningful variable names helped novices more than experts to identify beacons [7]. In these studies, expert programmers were either senior students or industry programmers while novices were students who had taken an average of two terms of programming. No studies have examined what information *complete novices* fixate on when reading code for the first time.

Code reading

Code reading is held as an important early gateway into learning to program [8], [9], [2], [10-11]. Reading and tracing code to determine its functionality is surprisingly difficult and reveals that novices struggle both to accurately read code and to extract meaning from that code [2]. Xie and colleagues argue that programming should first teach students how to read/trace code and then teach them how to comprehend common programming templates or plans [8]. These templates could be seen as being connected to the idea of beacons, where certain common coding patterns can help an expert more quickly identify the purpose or meaning of a line of code. Based on studies that have suggested that tracing skills may play an important role in helping students read code [9], some have proposed that early code reading should focus on teaching students how to formally trace programs with promising results [10].

Research Questions

We seek to examine what knowledge *complete novices* bring to the programming domain and whether beacons are a relevant construct in early learning of code reading. Our research questions are as follows.

R1: How accurately does each group of novice programmers trace code snippets?
R2: In what ways do the code reading behavior of complete novices and experienced students differ?
R3: To what degree do these differences indicate the importance of beacons in code reading and the contexts in which they matter?

Methods

Because we wanted to better understand what knowledge students bring into code reading/tracing tasks, we chose to interview *complete novices* and *experienced students*. We defined complete novices as students who had never read computer code before, informally or

formally. We defined experienced students as students who had taken at least three programming courses.

We chose to use C/Java for our interviews, because they are commonly taught as a first programming language and were the languages that most experienced students were taught on our campus. Pilot interviews revealed that complete novices struggled significantly with variable declaration keyword abbreviations such as `int`, and specific function names such as `printf` or `system.out` without interviewer intervention. We swapped the keyword `int` for `integer` and replaced `printf` or `system.out` with `display` so that these keywords would not need to be explained to the complete novices by the interviewer. Analysis of verbal protocols revealed no confusion for either group about the meaning of these changes.

Interview Protocols

We used two separate but related interview protocols for this study. All protocols used the same eight code snippets for eye tracking. These code reading tasks sought to examine participants' understanding of the concepts of assignment, conditionals, and loops. Two tasks focused on assignment, two tasks focused on only if-else conditionals, two tasks used a `while` loop (one with a nested conditional, one without a nested conditional), and two tasks contained `for` loops.

All assignment tasks focused on using multiple assignments where one variable's value was overwritten and that newly overwritten value was used in a subsequent assignment (e.g., Fig. 1, Problem 1). Conditional tasks assigned only an initial value to a variable and then asked students to assess whether one numerical value was greater than or less than another (e.g., Fig. 1, Problem 2). `while` loops were constructed so that the termination condition was achieved without incrementing by 1 each iteration (e.g., Fig. 1, Problem 7). `for` loops always incremented an iterator by 1 from the initial value until reaching the terminating value (e.g., Fig. 1, Problem 4).

Problem 1

```
1  integer cansInFridge;
2  integer cansInCase;
3  cansInFridge = 5;
4  cansInCase = 24;
5  cansInFridge = cansInFridge – 1;
6  cansInFridge = cansInFridge + cansInCase;
7  display(cansInFridge);
```

Problem 2

```
1  integer numEggs= 5;
2  if (numEggs< 2) {
3     display(“Buy more eggs!”);
4  }
5  else {
6     display(“Enjoy your omelet.”);
7  }
```

Problem 7

```
1  integer pounds = 250;
2  while (pounds > 190) {
3     display(pounds);
4     pounds = pounds - 10;
5  }
6
7  display("You reached your goal!");
```

Problem 4

```
1  integer factorial = 1;
   integer i;
2
3  for (i = 1; i < 5; i = i + 1) {
4     factorial = factorial * i;
5  }
6
7  display(factorial);
```

Fig. 1. Example code snippets from interview protocol. Problem 1 focuses on multiple assignments to the same variable. Problem 2 focuses on simple numerical conditional comparisons. Problem 7 focuses on `while` loops that do not increment by 1. Problem 4 focuses on `for` loops that do increment by 1.

Experienced Student (ES) Protocol

The participants' eye tracking display presented slides that resembled the image in Fig. 2, except without the answer in the Display Box. While their eyes were being tracked, *experienced students* read all eight code snippets silently, announcing only what they thought would be displayed in the Display box after the code snippet finished executing. Participants were instructed to read the code snippets silently so that participants would not feel the need to explain their reasoning, creating variances in reading times based on speaking speed or verbosity of explanations but only on comprehension speed. After providing their answers to all code-reading tasks, we retrospectively interviewed participants to ask how they determined their answers. These retrospective interviews are analyzed in other publications. This protocol took less than an hour.

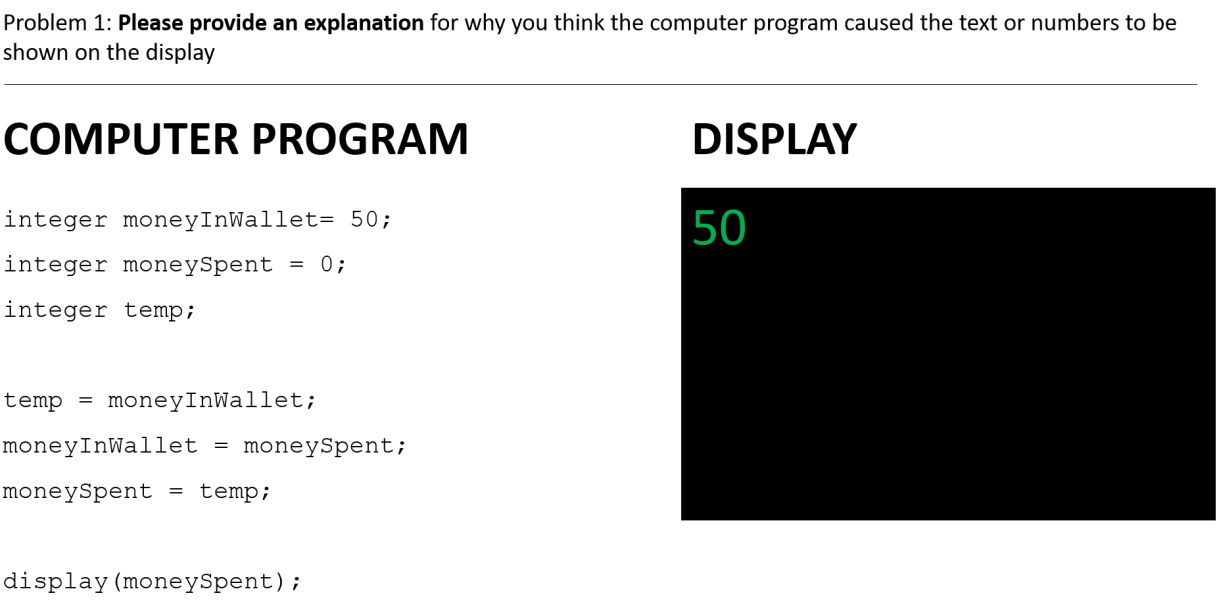

Fig. 2. Participants were given a code snippet (left) and asked to explain why they thought the code snippet produced the value shown on the Display box or predict what they thought would be displayed on the Display box (right).

Complete Novice (CN) Protocol

This protocol was broken up into two sessions, each less than an hour long.

First Session: While their eye movements were being tracked, complete novices read four code snippets (one from each category of task) silently before announcing what they thought would be displayed. This data was used as a baseline measurement of code reading behavior. After these initial code-reading tasks were completed, we performed a retrospective interview, asking the participants how they determined their answers.

After this baseline measurement, we gave participants a brief, self-guided training exercise. Participants were given eight new code snippets that were different from the snippets used during the eye-tracking portion of the study but had the same distribution of code snippets. The participants were shown the correct answer after the code executed (Fig. 2). Participants were asked to explain why they thought the code produced the answers it did.

Second Session: After the training exercise, we performed the same eye-tracking study and retrospective interview with the complete novices as with the experienced students. Four of the code snippets were repeated from the baseline measurement to assess learning. Four of the code snippets were new to the complete novices and assessed near transfer of learning.

Participants

A total of 102 participants (55 female, 43 male, 4 not disclosed) participated in the study. Due to eye-tracker calibration issues, only 90 participants' data were analyzed: 34 Experienced Students (ES) (6 female, 26 male, 2 not disclosed), and 56 Complete Novices (CN) (43 female, 12 male, 1 not disclosed). ES participants were primarily computer science majors. CN participants were primarily non-STEM majors. All participants were traditionally aged students at a large, Midwestern Research University.

During the consent process, participants confirmed that they met the prerequisite conditions for assignment to each group. Participants were compensated $30/hour for participating.

Apparatus

Eye-tracking data and audio was recorded with Tobii T120 eye-tracker running Tobii studio 3.4.7 software using standard I-VT filter. The eye-tracker was calibrated to each participant individually. All participants were assigned a randomly generated ID to link eye-tracking data with audio recordings. In accordance with IRB regulations, the recordings and .csv data files were exported to a secure Box folder with each subfolder labeled with participant randomly generated 5-digit ID and contained video, complete .csv data files, and audio recordings.

Results

Research Question 1

R1: How accurately does each group of novice programmers trace code snippets?

For research question 1, we tabulated participants' correct/incorrect answers by major concept (assignment, conditional, iteration). For iteration questions, we also marked whether the participants' answer revealed any indication that iteration had occurred. A qualitative analysis of students' responses is explored in a different manuscript.

Table I

Percentage of correct responses on tracing tasks for complete novices and experienced students. Percentage of correct responses on Pre-/Post- tasks were compared using Mann-Whitney U tests.

	Assignment	Conditional	while loop	for loop
Complete Novices (PRE)	30%	78%	0%	0%
Complete Novices (POST)	64%	92%	2%	1%
p-value (PRE-/POST-)	<0.001	<0.001	0.16	0.32
Cohen's d	0.60	0.39	0.18	0.13
Experienced Students	100%	100%	100%	92%

The experienced students correctly traced all the code snippets, except for off-by-one errors on the iteration “for loop” snippets (i.e., one too many iterations). For the Problem 4 code snippet (Figure 1, Problem 4), 6 of 38 participants reported the answer as 5 factorial instead of 4 factorial.

During the baseline activity, roughly a third of the complete novices correctly traced the assignment code snippets. This percentage doubled, rising significantly ($p < 0.001$, d = 0.60) after the training exercise. These results reveal that many complete novices (at the university level) bring useful intuitions to their first programming experience and that many just need a small nudge to begin understanding how multiple assignments might work.

Similarly, roughly three quarters of complete novices demonstrated an understanding of conditionals. This number significantly rose ($p < 0.001$, d = 0.39) to roughly 9 in 10 after the training exercise. These results reveal that most novices bring an intuitive understanding of conditionals, at least when comparing the relative magnitude of two numbers.

Complete novices did not demonstrate an intuitive understanding of iteration *initially* and the training exercise did not significantly increase the number of students who understand them. Two students revealed an accurate understanding of iteration on multiple code tracing tasks only after the intervention, suggesting that some students may bring useful prior experiences that prepare them to understand the idea of iteration with even the most modest of instruction.

Research Question 2

R2: In what ways does the reading behavior of complete novices and experienced students differ?

The main dependent variables for this study are total reading time and fixation count. Total reading time was chosen because it is used as an indicator of attention spent on text or code. Fixation count was chosen because increased number of fixations alludes to increased reiteration and/or re-fixation within the snippets of code. The predictors are Novice Type with two levels (Experienced Student (ES) and Complete Novice (CN)) and line of code read (this depended upon the problem analyzed, which ranged from 7 lines in Problem 1 to 15 lines for Problem 3). Our main hypothesis is that there will be differences in the reading times between the experienced students and the complete novices. We also suspect that reading times will be more evenly spread across the lines of code for the complete novices, whereas the experienced students will focus their attention on lines of code that are particularly informative to the output to be displayed (i.e., indicating possible beacons) and will have increased number of fixations on lines that require iteration. In other words, we expect the experienced students to take advantage of the beacons within the code and spend more time in those areas as indicated through total time reading and increased number of fixations.

For brevity, we focus our results on only the part of the protocol that was the same for all participants: the entire eye-tracking protocol for the experienced student group and session 2 for the two complete novice groups. Within that, we further focus on two of the eight coding problems that were presented: the shortest and longest code snippets. Problem 1 was chosen because it is a representative of most problems, in that at least one line of code contained a significant difference in total reading time between the complete novice group and the

experienced student group. Problem 3 was chosen because it was the only problem in contrast to this, whereby the experienced student group spent significantly longer time reading line 5.

One fixation above 3000 milliseconds was removed from the analysis due to participant inattention. All eye-tracking data was processed through R [12] and ANOVA models were made with base R. Interactions were explored with emmeans [13] and figures were made using ggpubr [14] and ggplot2 in the tidyverse package [15]. The results are constrained to total gaze duration times and fixation counts.

Analysis of Problem 1

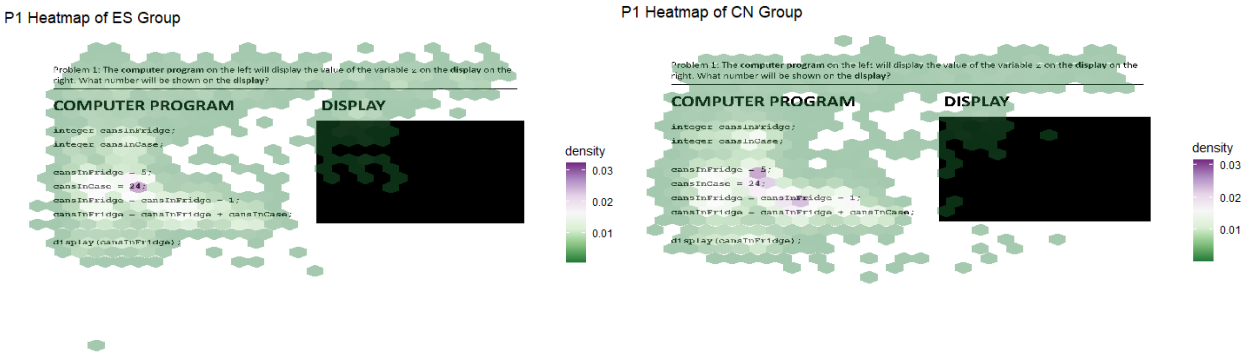

Fig. 3. Problem 1 Heatmaps for each novice type, showing the relative number of times each population fixated on each portion of the display. Experienced students spent significantly less time fixating on lines 5 and 6 (lines that re-used prior assignments) than the complete novices.

The fixations for both groups are qualitatively illustrated using heatmaps (Fig. 3). Heatmaps for each population were created using ggoplot2 binhex. The heatmaps show that all participants' attention were focused on lines 3 through 6. However, the complete novices typically needed multiple fixations to comprehend the assignment operations on lines 5 and 6 as illustrated by the purple and white hexagons on those lines. In contrast, the ES group did not generally need multiple fixations to comprehend these lines as illustrated by the lack of purple hexagons.

A two-way ANOVA was run on the total gaze duration with Novice Type and Area Of Interest (AOI) as predictors, where the AOIs were defined as each line of code, in the model. The results indicated a significant main effect of Novice type ($F(1, 590) = 15.32$, $p < 0.001$) and a main effect of AOI ($F(7, 590) = 23.52$, $p < 0.001$). There was also a significant interaction between the two effects ($F(7, 590) = 2.83$, $p = 0.007$). Indicating that the ES and CN spent different amounts of time in total and that different amounts of time were required on different parts of the display. A pairwise comparison revealed significant differences between groups on lines 5 and 6. On Line 5, there was a significant difference between CN and ES ($\beta = 1956$, SE = 406, $p < 0.001$), such that there was an estimated almost two seconds longer total gaze duration made by the CN group on line 5 as compared to the ES group's total gaze time. On Line 6, there was a significant difference between CN and ES ($\beta = 1361$, SE = 408, $p = 0.001$), such that there was

an estimated second longer total gaze time made by the CN group on line 6 as compared to the ES group's total gaze time.

A separate two-way ANOVA was run on the number of fixations made with each line of interest as one predictor variable and with novice type as a second. The results indicated a significant main effect of Novice Type ($F(1, 590) = 15.16$, $p < 0.001$) and a main effect of AOI ($F(7, 590) = 28.95$, $p < 0.001$). The interaction was also significant ($F(1, 590) = 2.99$, $p = 0.004$). This indicates that each group had a different number of fixations, some interest areas required more fixations than others, and that each group had a different number of fixation in different areas of interest.

Table II

Pairwise comparison between complete novices and experienced students for total gaze duration (ms) and fixation count for Problem 1. The results from the Tukey Estimated Marginal means post-hoc comparison is shown below.

	Total Gaze Duration (ms)		Fixation Count	
Comparison	Complete vs. Experienced Students		Complete vs. Experienced Students	
AOI	Estimate(SE)	*p-value*	Estimate(SE)	*p-value*
L1	175 (471)	0.710	0.29 (1.97)	0.882
L2	184 (445)	0.679	0.80 (1.86)	0.668
L3	233 (448)	0.603	1.71 (1.87)	0.363
L4	249 (413)	0.547	0.68 (1.73)	0.694
L5	**1956 (406)***	<0.001	**8.73 (1.70)***	<0.001
L6	**1361 (408)***	0.001	**5.01 (1.70) ***	0.003
L7	383 (471)	0.416	1.81 (1.97)	0.359

Table III

Problem 1 Mean and Standard Deviation of Total Gaze Duration and Fixation Counts. A two-way ANOVA revealed a significant difference between the groups for the main effect ($p < 0.001$).

	Total Gaze Durations (ms)		Fixations	
AOI	CN	ES	CN	ES
L1	853.00 (634.79)	677.75 (619.45)	4.00 (2.04)	3.71 (3.14)
L2	1006.22 (699.82)	821.85 (637.06)	4.76 (3.29)	3.96 (2.92)
L3	1605.52 (1220.75)	1372.07 (975.08)	7.85 (5.80)	6.15 (4.14)
L4	2143.62 (1940.19)	1894 (1366.89)	9.68 (7.52)	9.00 (5.71)
L5	**3972.25 (4070.23)***	**2015.94 (1668.34)***	**18.44 (16.47) ***	**9.71 (7.55) ***
L6	**3371.81 (3160.02)***	**2010.65 (1197.14)***	**15.69 (13.30)***	**10.68 (6.07)***
L7	1099.31 (1074.72)	715.83 (697.43)	5.56 (4.52)	3.75 (3.11)

Table 2 outlines the pairwise comparison which revealed significant differences between groups on lines 5 and 6. On line 5, the complete novices made significantly more fixations than the ES group (β =8.73, SE = 1.70, $p < 0.001$). On line 6, the same pattern occurred with the complete novices making significantly more fixations than the ES group (β = 5.01, SE = 1.70, p = 0.003). This finding is similarly replicated in the total gaze duration analysis, such that the increased number of fixations resulted in longer total reading times on line 5.

Analysis of Problem 3

Problem 3 (Fig. 4) was the longest code snippet with the most control structures.

```
1    integer running = 1;
2    integer waterBreaks= 0;
3    integer minutes = 0;

4    while (minutes < 20) {
5       if(running == 1) {
6          minutes = minutes + 4;
7          running = 0;
8       }
9       else {
10         minutes = minutes + 2;
11         waterBreaks = waterBreaks + 1;
12         running = 1;
13      }
14   }

15   display(waterBreaks);
```

Fig. 4. Problem 3 from the eye-tracking study. This problem was the most complicated problem in the study and assessed students' understanding of iteration, conditionals, and assignment.

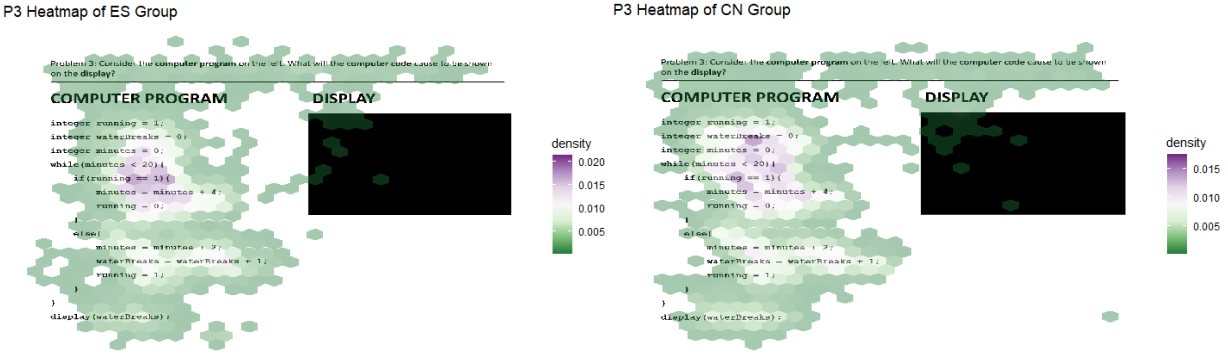

Fig. 5. Problem 3 Heatmaps for each novice type. Experienced students spent significantly more time fixating on line 5 (the nested conditional) than the complete novices.

The heatmaps (Fig. 5) for Problem 3 show that both groups focused on lines 2 through 7. However, the ES group gave the highest amount of attention to lines 4-6 (with significantly longer gaze duration and fixation counts on line 5—a conditional that played a major role in terminating the loop).

A two-way ANOVA for total reading times was created for Problem 3, the results of which found a significant main effect of AOI ($F(14, 963) = 15.58$, $p < 0.001$) and a significant interaction between AOI and Novice Type ($F(15, 963) = 2.49$, $p = 0.003$). This means that there were several areas of interest that had longer fixation durations and that these areas of interest had longer or shorter fixation durations depending upon the group. The pairwise comparison (Table 4) revealed significant differences between groups on lines 1, 2, and 5. On line 1 there was a significant difference between CN and ES ($\beta = 782.8$, SE = 405, $p = 0.053$) such that the CN group had significantly longer total gaze durations on line 2 as compared to the ES group. We found a similar group difference on line 2 ($\beta = 1049.20$, SE = 373, $p = 0.005$). However, on line 5 there was the opposite effect, whereby the ES group was estimated to have longer total gaze times as compared to the CN group ($\beta = -912.3$, SE = 962, $p = 0.015$).

A separate two-way ANOVA was created to explore the number of fixations made to each line of code by the Novice Type for Problem 3. The results indicated a significant main effect of AOI ($F(14, 963) = 18.44$, $p < 0.001$), and a significant interaction between AOI and Novice Type ($F(15, 950) = 2.42$, $p = 0.004$). This main effect result pattern is similar to the total reading time ANOVA, such that there were several areas of interest requiring more fixations, but that this depended upon which group the participant was in. A pairwise comparison revealed significant differences between groups on lines 1, 2, 3 and 5. On lines 1, 2 and 3, the CN group made significantly more fixations than the ES group (**Line 1**: $\beta = 3.59$, SE = 1.78, $p = 0.043$; **Line 2**: $\beta = 5.88$, SE = 1.64, $p < 0.001$; **Line 3**: $\beta = 3.25$, SE = 1.98, $p = 0.048$). On line 5, the opposite pattern occurred whereby the ES group made significantly more fixations than the CN group ($\beta = -3.42$, SE = 1.64, $p = 0.037$). The comparatively less time spent by the ES group on code reading in general but comparatively more time reading the nested conditional supports the idea that the ES group was obtaining information in a meaningful way.

Table IV
Pairwise comparison between complete novices and experienced students for Total Gaze Duration and Fixation Count for Problem 3.

	Total Gaze Duration (ms)		Fixation Count	
Comparison	Complete Novice vs. Experienced Students		Complete Novices vs. Experienced Students	
AOI	Estimate(SE)	*p-value*	Estimate(SE)	*p-value*
L1	**782.80 (405)***	**0.054**	**3.59 (1.78)***	**0.043**
L2	**1049.20 (373)***	**0.005**	**5.88 (1.64)***	**< 0.001**
L3	691.50 (373)	0.064	**3.25 (1.64)***	**0.048**
L4	465.4 (374)	0.214	1.186 (1.64)	0.471
L5	**-912.30 (373)***	**0.015**	**-3.42 (1.64)***	**0.037**
L6	-657.00 (373)	0.079	-1.86 (1.64)	0.255

L7	-308.20 (395)	0.436	-1.66 (1.73)	0.338
L8	NA	NA	NA	NA
L9	-25.80 (457)	0.955	-0.310 (2.01)	0.879
L10	171.60 (382)	0.653	1.46 (1.67)	0.384
L11	-581.30 (390)	0.136	-0.950 (1.71)	0.579
L12	-39.0 (432)	0.928	-0.125 (1.90)	0.948
L13	NA	NA	NA	NA
L14	NA	NA	NA	NA
L15	527.90 (409)	0.198	3.00 (1.80)	0.095

Table V

Problem 3 Mean and Standard Deviation of Total Gaze Durations and Fixation Counts. The ANOVA revealed a significant difference between the populations for the main effect (p = 0.008). We exclude lines that include only closing curly brackets since most participants never fixated on them.

	Total Gaze Durations (ms)		Fixations	
AOI	CN	ES	CN	ES
L1	**1659.89 (1319.87)***	**877.14 (842.54)***	**8.17 (5.67)**	**4.57 (4.13)**
L2	**2575.02 (1582.39)***	**1525.82 (765.86)***	**13.41 (8.10)**	**7.53 (3.43)**
L3	2492.26 (1730.06)	1800.74 (789.74)	**12.63 (8.15)**	**9.38 (4.69)**
L4	2829.70 (2063.19)	2364.26 (1327.57)	13.51 (9.56)	12.32 (6.49)
L5	**2613.69 (2155.59)***	**3526.00 (2687.14)***	**12.52 (9.20)**	**15.94 (9.58)**
L6	2581.02 (2562.23)	3238.00 (2140.45)	13.17 (11.65)	15.03 (8.23)
L7	1483.67 (1087.61)	1791.87 (1218.32)	7.20 (5.49)	8.87 (5.49)
L8	NA	NA	NA	NA
L9	434.80 (338.62)	460.57 (439.26)	2.17 (1.63)	2.48 (1.63)
L10	2287.11 (2244.21)	2115.50 (2254.41)	11.49 (10.00)	10.03 (7.73)
L11	2181.80 (2052.63)	2763.13 (2704.34)	11.76 (10.41)	12.71 (8.82)
L12	759.88 (757.19)	798.92 (618.83)	4.00 (3.86)	4.12 (2.64)
L13	NA	NA	NA	NA
L14	NA	NA	NA	NA
L15	1323.85 (1535.73)	795.97 (727.14)	7.37 (7.67)	4.37 (3.50)

Research Question 3

R3: To what degree do these differences indicate the importance of beacons in code reading and the contexts in which they matter?

Our data is inconclusive about the importance of beacons in code reading and more research is needed. Our data does suggest some hints about when beacons are important, particularly the differences in problem 3 from all other problems in the protocol. Problem 3 was the longest, and most complicated, problem of the eight that the participants had to solve and contained a loop with a nested conditional. Experienced students demonstrated an

understanding of the importance of this beacon by fixating on this line of code significantly more frequently and for longer durations than the complete novice group. The increased number and duration of fixations indicate that these fixations were not just simple checks on information, rather they were deliberate moments of information processing. In all other problems (and even those not reported in this paper), experienced students spent less time reading the code than did the complete novices. We believe that this is due to the length and complexity of Problem 3 and that future studies should include longer snippets of code to further examine the potential importance of beacons.

Limitations

The study relied almost exclusively on very simple code snippets, which helped us understand how novices and beginner programmers might understand specific programming constructs but limited our ability to investigate when/how beacons may be important.

While the whole population of the study had representation from male and female participants, the ES and CN groups had dramatically different demographics, limiting our ability to investigate any gender-based differences. Our study recruited primarily non-STEM-major participants for the CN group, limiting our observations about what STEM-oriented complete novices may have behaved but perhaps providing potential future insights for broadening participation in computing.

Discussion and Conclusions

Programming has become a necessary skill in almost all domains and is particularly necessary for Engineering students. It is imperative, then, that we investigate what skills and knowledge students who have no prior formal coding experience bring to the learning process. Our study provided evidence that college-aged, complete-novice programmers bring some intuitive understandings about assignment and conditional operations. After a brief training exercise (< 30 minutes with no formal instruction), a majority of the CN group were able to correctly reason about multiple assignment calculations and nearly all were able to correctly reason about basic conditionals. Assignment was notably trickier than conditional operators, suggesting that colloquial knowledge of conditionals translates well into programming contexts, but that the difference in meaning between assignment and equivalence for the ‘=’ sign is troublesome. However, the behavior of assignment appears to be something that most complete novices can discover inductively. Even though the complete novices, could reason about these constructs correctly, they took longer to reason about both assignment and conditional operations than more experienced students. The mechanisms for this increased speed are not clear, but future studies into the role of beacons in code reading will certainly need to consider the relative speed of code reading for different lines of code and not just absolute speed of code reading.

In contrast with assignment and conditionals, few novices demonstrated any understanding of iteration. Additionally, with the more complex code from Problem 3 that included conditionals and iterations, the experienced students spent more time and needed more fixations to trace the code. This difference suggests that the experienced students were actually iterating through

Problem 3 to trace it, while the complete novices were not. These findings further suggest that for most of our code snippets, the concept of beacons is not particularly helpful—the code is just too simple. In contrast, with the more complicated Problem 3, we begin to see that the more experienced students spent more time fixating on the nested conditional, suggesting that the conditional may have played a central beacon role in understanding and tracing the code.

The experienced students spent significantly less time on lines 2 and 3 (variable initializations) than the complete novices in Problem 3 but spent significantly more time on the loop conditional, suggesting that the difference in importance for the conditionals may have been more stark if we had analyzed the relative amount of time participants spent on each line of code rather than the absolute amount of time. Similarly, the number of fixations is inadequate to assess whether the experienced students actually iteratively through or traced the code and the complete novices did not. Future work can explore using scanpath analyses and proportional fixation durations to more deeply interrogate these questions.

This study suggests that the use of beacons for studying novice code-reading behavior requires a certain threshold of complexity before being useful. Future work on the productive knowledge of complete novices may need to focus more on how those novices search for information and the useful analogies or experiences that they bring to their early learning experiences.

References

[1] L. Kaczmarczyk, E. Petrick, J. P. East, and G. L. Herman, “Identifying student misconceptions of programming,” *in Proceedings of the Forty-First ACM Technical Symposium on Computer Science Education*, 2010, Conference Proceedings, pp. 107–111.

[2] R. Lister, B. Simon, E. Thompson, J. L. Whalley, and C. Prasad, “Not seeing the forest for the trees: Novice programmers and the solo taxonomy,” *in Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ser. ITICSE ’06*. New York, NY, USA: Association for Computing Machinery, 2006, p. 118–122. [Online]. Available: https://doi.org/10.1145/1140124.1140157

[3] J. D.Bransford, A. L.Brown, and R. R.Cocking, editors, How People Learn: Brain, Mind, Experience, and School: Expanded Edition. Washington, DC: The National Academies Press, 2000. [Online]. Available: https://www.nap.edu/catalog/9853/how-people-learn-brain-mind-experience-and-school-expanded-edition

[4] K. Rayner, “Eye movements and attention in reading, scene perception, and visual search,” *The Quarterly Journal of Experimental Psychology*, vol. 62, no. 8, pp. 1457–1506, 2009. [Online]. Available: https://doi.org/10.1080/17470210902816461

[5] M. Hegarty, Multimedia learning and the development of mental models. Cam- bridge: *Cambridge University Press*, 2014, p. 673–702.

[6] A. M. Madsen, A. M. Larson, L. C. Loschky, and N. S. Rebello, “Differences in visual attention between those who correctly and incorrectly answer physics problems,” *Phys. Rev. ST Phys. Educ. Res.*, vol. 8, p. 010122, May 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevSTPER.8.010122

[7] M. Crosby, J. Scholtz, and S. Wiedenbeck, “The roles beacons play in comprehension for novice and expert programmers,” 07, 2002.
[8] B. Xie, D. Loksa, G. L. Nelson, M. J. Davidson, D. Dong, H. Kwik, A. H. Tan, L. Hwa, M. Li, and A. J. Ko, “A theory of instruction for introductory programming skills,” *Computer Science Education*, vol. 29, no. 2-3, pp. 205–253, 2019. [Online]. Available: https://doi.org/10.1080/08993408.2019.1565235
[9] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas, “A multi-national study of reading and tracing skills in novice programmers,” *in Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, ser. ITiCSE-WGR ’04*. New York, NY, USA: Association for Computing Machinery, 2004, p. 119–150. [Online]. Available: https://doi.org/10.1145/1044550.1041673
[10] M. Hertz and M. Jump, “Trace-based teaching in early programming courses,” *in Proceeding of the 44th ACM Technical Symposium on Computer Science Education, ser. SIGCSE ’13*. New York, NY, USA: Association for Computing Machinery, 2013, p. 561–566. [Online]. Available: https://doi.org/10.1145/2445196.2445364
[11] K. Cunningham, S. Blanchard, B. Ericson, and M. Guzdial, “Using tracing and sketching to solve programming problems: Replicating and extending an analysis of what students draw,” *in Proceedings of the 2017 ACM Conference on International Computing Education Research, ser. ICER ’17*. New York, NY, USA: Association for Computing Machinery, 2017, p. 164–172. [Online]. Available: https://doi.org/10.1145/3105726.3106190
[12] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/
[13] R. Lenth, emmeans: Estimated Marginal Means, aka Least-Squares Means, 2020, r package version 1.4.7. [Online]. Available: https://CRAN.R-project.org/package= emmeans
[14] A. Kassambara, ggpubr: ’ggplot2’ Based Publication Ready Plots, 2020, r package version 0.4.0. [Online]. Available: https://CRAN.R-project.org/package=ggpubr
[15] H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani, “Welcome to the tidyverse,” Journal of Open Source Software, vol. 4, no. 43, p. 1686, 2019.