Improving Communication Between Pair Programmers Using Shared Gaze Awareness

Sarah D'Angelo
Northwestern University
Evanston, Illinois USA
sdangelo@u.northwestern.edu

Andrew Begel
Microsoft Research
Redmond, Washington USA
andrew.begel@microsoft.com

ABSTRACT

Remote collaboration can be more difficult than collocated collaboration for a number of reasons, including the inability to easily determine what your collaborator is looking at. This impedes a pair's ability to efficiently communicate about on-screen locations and makes synchronous coordination difficult. We designed a novel gaze visualization for remote pair programmers which shows where in the code their partner is currently looking, and changes color when they are looking at the same thing. Our design is unobtrusive, and transparently depicts the imprecision inherent in eye tracking technology. We evaluated our design with an experiment in which pair programmers worked remotely on code refactoring tasks. Our results show that with the visualization, pairs spent a greater proportion of their time concurrently looking at the same code locations. Pairs communicated using a larger ratio of implicit to explicit references, and were faster and more successful at responding to those references.

ACM Classification Keywords

H.5.3. Information Interfaces and Presentation (e.g. HCI): Group and organizational interfaces - collaborative computing, computer-supported collaborative work

Author Keywords

Eye-tracking; collaboration; pair programming

INTRODUCTION

Remote work is becoming increasingly popular because it supports a flexible lifestyle, and helps people avoid a long commute. Technological advances have improved remote work, however we still lose some important physical affordances of collocated work, such as non-verbal cues and gestures. This especially impacts the ability to understand where your partner is looking in a shared workspace. However, recent work using dual eye tracking technology has addressed this problem in the context of remote collaborative work [27]. The dual eye tracking solution enables tool builders to share eye gaze

information with each worker and incorporate non-verbal cues into remote work.

In our work, we apply this method to pair programming. Pair programming is a commonly practiced method of programming that allows two programmers to work together on a single project [49]. This method is popular at many technology companies, but often occurs only in teams that make it their standard practice. It requires scheduled co-presence during the work week along with special computer setups [4]. Previous work has demonstrated the effectiveness of this method and the perceived value by the participants [6, 12, 16, 23, 39]. For example, pairing demands a high level of attention and focus from the partners because they are continuously working with someone else and avoiding distractions [10, 46].

Pair programming is an appropriate task for studies of shared gaze awareness because it is tightly coupled, synchronous, and collaborative. These factors discourage people from working remotely when they are part of a pair programming team. Extending the reach of pair programming to support remote collaboration can improve flexibility of work location, accessibility, enable geographically distributed work, and accommodate the needs and preferences of a diverse workforce. A survey we conducted (reported in Section 3) found that developers would like this flexibility, but current tools for remote pair programming fail to recreate an effective pairing experience.

General tools for remote work, such as the use of Skype with screen sharing, lack the ability to share non-verbal cues that facilitate communication between collocated pairs [15]. Special purpose tools, such as Saros, a distributed IDE [43], support limited communication of mouse-based gestures. However, pairs still cannot make eye contact or see gestures that are not drawn on screen. The absence of these features makes disambiguating references difficult [22]. In a remote setup, pairs sometimes have to use many explicit references to coordinate with their partner on a specific location in code, often resorting to speaking line numbers. In this work, we aim to support synchronous collaboration in remote settings with a shared workspace. We designed a novel, shared gaze awareness visualization that reduces the effort required to communicate about specific locations in code by showing a collaborator where in the source code his/her partner is looking.

We evaluate our design for sharing gaze awareness in a study of pair programmers working in a simulated remote setting. Our design is subtle and accounts for noise in the eye tracking

signal to avoid the negative effective of distracting displays that can mislead partners [13]. We conducted a within-subjects experiment to evaluate the visualization and understand how sharing gaze information affects the way pair programmers communicate about specific locations in code.

Our results show that when programmers can see where their partner is looking, they concurrently look at the same code more often. With the visualization, they are able to communicate about locations in the code more successfully, efficiently, and effectively. Thus, the design serves as a helpful aid for coordinating remote work. And, the same design could easily be integrated into many other forms of collaborative document editing.

We discuss the role of shared gaze awareness in remote pair programming and offer implications for the design of shared gaze awareness in computer-supported collaborative work.

RELATED WORK

Eye movements provide valuable information about how pairs communicate. For example, before speakers refer to an object, they look at it [20]. Listeners look at objects soon after they are referred to [1]. This relationship is important for coordination and developing common ground between partners [41, 42].

Remote work is challenging because it lacks important features of collocated work, such as knowing where your partner is looking. *Sharing gaze awareness*, or displaying where your partner is looking on your screen, is one potential intervention to help facilitate conversation and coordination. For example, listeners can use the speaker's gaze to help disambiguate similar objects [22].

Dual eye tracking studies have begun to explore how sharing gaze in real time between two remote partners influences coordination [27]. For example, remote collaborators can find a target in a image faster when shown their partner's gaze on screen [7]. Additionally, in a collaborative learning task, shared gaze information improved gains in learning and increased the partners' joint attention [44]. Sharing gaze also affects how partners communicate by allowing them to make use of more *deictic* references (i.e. this or here) using a gaze representation as a referential pointer [3, 13].

Pair programming is an interesting application space for sharing gaze awareness because it requires a high level of shared understanding [48]. Knowing what someone is looking at in code can provide insights for problem solving. For example, when novices were shown an expert programmer's gaze pattern in code review, it helped them identify bugs faster [47]. Additionally, in real-time gaze sharing between expert and novice, watching an expert's gaze as they explained an algorithm reduced the variability of gaze patterns in novices [38, 5].

Shared gaze is typically represented as a single point on screen that reflects the gaze coordinate stream interpreted from the eye tracker [3, 7, 13, 44]. However, alternative designs have explored depicting gaze in the periphery to subtly direct attention to specific areas in an image [2, 32]. Additionally, Stotts et al. displayed a head overlay on the screen to share gaze

information and facial expressions [21]. Related work in collaborative document editing has also investigated unobstrusive indicators to signal where partners are working [36, 29]. We contribute to this space by evaluating a simple and transparent design for displaying gaze awareness in documents.

We can also analyze the relationship between two gaze streams during a collaborative task to evaluate effective coordination in pairs. The relationship between reference forms and gaze overlap can be a useful indicator of understanding between pairs [18]. Sharing cues, such as highlighting text, increases the amount of time pairs spend looking in the same place together [28, 45]. Additionally, gaze location can be predicted based on conversational content and actions in the shared workspace [34]. We investigate gaze overlap and the relationship between reference form and acknowledgments (both verbally and based on gaze).

Recreating the ability to see what your partner is looking at in remote collaborative work has many possible applications, ranging from pair programming [5] to remote physical tasks [17, 24]. We contribute to this space by evaluating a novel design for sharing gaze awareness to support effective communication in tightly-coupled tasks.

SURVEY

To motivate this work, we randomly selected 500 software developers at a large, technology company and asked them to respond anonymously to a survey about pair programming. We received 159 complete responses for a 32% response rate. 52.7% reported that they had pair programmed before. Consistent with previous work [6, 12, 16, 39], a majority of the respondents who have pair programmed say that they enjoy pairing and believe that it is effective.

In our work, we are particularly interested in expanding the capabilities for pair programming to effectively support remote pair programming. So asked survey respondents whether they might take advantage of working remotely if they could. Our survey revealed that if they had no constraints, our respondents would like to spend over 11 hours per week (out of a 40 hour work week) working remotely. Respondents explained that working remotely would allow them to avoid a long commute, work in a place with fewer distractions, and adjust their schedule to accommodate personal needs. For example, one respondent said that working remotely would allow him/her to *"spend time with my 2-year-old son during the day, and then work in the evening more after he has gone to sleep."* Remote work can accommodate a wide range of lifestyle preferences and allow for new opportunities that may not have happened otherwise. At large companies with offices in many countries, effective remote work could improve and prompt new international collaborations.

Despite this desire to work remotely, those who pair program (usually on teams that practice pair programming religiously) rarely engage in remote pair programming. Our survey revealed that only 18.9% of the 87 respondents who said they had ever pair programmed had pair programmed remotely. Those that did pair remotely said they liked its convenience and flexibility. It allowed developers to work with people who

are not in the same location, and was "necessary if the subject matter expert was not local."

Of the remaining 81.1% who had never tried remote pair programming, 40.3% considered it, but some worried that the technology and management would present barriers to success. One respondent said "I don't think our technology would support it enough to be effective." The most common tool used by our respondents is Skype with desktop sharing, which has no specific features designed for collaborative editing/programming.

We believe that the benefits of pair programming should not come at the expense of the ability to work remotely. Guided by our survey responses, we undertake a study to test the effectiveness of a novel, remote pair programming support tool.

PILOT OBSERVATIONS

We observed 3 teams at the same large technology company to gain an understanding of how they practice pair programming. Each video-recorded observation lasted one hour in the team's primary office space. Two researchers were present during the observations and taking notes.

Each of the three teams practiced a different form of pair programming. The first pair employed a special hardware setup using a single computer with two mirrored sets of dual monitors. They spent the hour planning a refactoring of the code that one of them had written with a different partner the week before. The second pair operated in a traditional driver/navigator role structure in front of one of the pair's dual monitor computer. The driver spent the hour explaining to the navigator the functionality of a particular code abstraction he had written by himself an hour before we observed them, when his partner was out at a meeting. The third team practiced "mobbing," in which they had four people in front of one computer with a large 50-inch monitor. One person, who was facile with source control commands, worked the keyboard, while the others sat around him. They spent their hour debugging a problem that had shown up in the nightly build of their web service.

As we observed each team, we were on the lookout for aspects of collocated pair programming that would not be well-supported in remote work. We found three: first, remote pairs would not be able to make eye contact, something that was frequently practiced in all three teams. From an outside observer's perspective, pair programming looks more like a social, conversational activity than an activity purely about the program on the monitor.

Second, the members of the teams made numerous hand gestures both at the screen and at one another, which would simply not be visible in a remote setting. Sometimes these gestures merely supported the on-going conversation (i.e. body language), but sometimes the person making the gesture was pointing at something on the screen to get the other person to notice it. The remote tools our survey respondents talked about having used would definitely not support resolving this kind of gesture.

Third, the teams made constant verbal and gestural references to specific locations on the screen, something that would be difficult to impossible to disambiguate while working remotely. The mobbing team demonstrated this acutely; even in their collocated setting, the "driver" experienced numerous difficulties interpreting the code locations at which his teammates were pointing. One pair was successful at using mouse-based selection to indicate the part of the code the other was supposed to pay attention to, but the other pairs experienced difficulty distinguishing between verbally called out functions with similar names, or where there were multiple calls to the same function in a single document.

We decided to focus primarily on the issue of referring to locations on screen by transforming the partners' gaze locations into on-screen visualizations.

GAZE VISUALIZATION

In this work, we investigate the effect of displaying gaze information to remote pair programmers in an unobtrusive and transparent way. We designed a novel, shared gaze visualization to indicate where in the code your partner is looking when programming together on a shared display. We conducted a within-subjects experiment in which pair programmers are asked to complete two refactoring tasks together, with and without our shared gaze visualization. We evaluate the effectiveness of our design by investigating where pairs are looking and how they communicate about specific locations on the screen. Additionally, we conducted post-task surveys and semi-structured interviews with our participants to understand the perceived utility of shared gaze awareness. Our results show that with shared gaze awareness, pairs spend more time looking at the same locations, are faster to acknowledge references, and use more implicit language when making references.

Design Process

Much of the previous literature on displaying gaze information uses a small, rapidly-moving point to illustrate where the eye is looking at the screen. This type of visualization moves too fast and can obscure the information on screen, both of which are distracting [13]. The single point representation is also inaccurate because the imprecision caused by noise and microsaccades that is inherent in current eye tracking technology results in biases in the returned screen coordinates [8, 35].

To avoid these problems, we initially focused on line-based text designs that could make the gaze visualization move more slowly. Our first design highlighted the background of an entire row of text, but initial pilot tests revealed that this to still be too distracting; too many pixels had to change color as the eye moved rapidly up and down the screen. First, we shrank the highlight rectangle horizontally. We started with a 10 pixel wide rectangle, but we found it to be too subtle in our pilot tests. So we increased it to 20 pixels, which was perceived as noticeable, but not distracting. Next, we tried illustrating the user's focus by increasing the saturation of the highlight color the longer the user looked at a particular line, but the subtle color change went unnoticed by our study participants. Finally, we tried changing the number of lines in
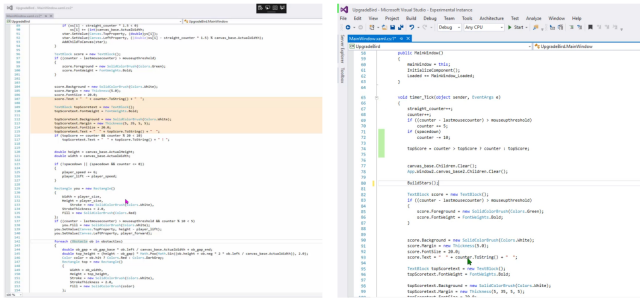
Figure 1. Screen Captures with First Iteration of Gaze Visualization (left) and Final Design (Right)

Figure 2. Experiment Setup

the highlighted rectangle to show the current range of lines at which the user was looking. However, when the eye moved rapidly around the screen, the rectangle would change size a lot. Also, with only 20-30 lines typically visible in the editor, the rectangle too often covered most of the screen, making the visualization useless.

In the end, we fixed the vertical span of the rectangle to 5 lines, colored it with a single pair of semi-saturated colors (chosen for maximal contrast with the default font colors used in Visual Studio), and moved it to the left margin of the text where the participants were already looking, so as to not attract their attention to irrelevant screen locations (e.g. blank spaces on the right side of the editor).

Gaze Visualization Design

Our final design is a 5-line high, 20 pixel wide, filled, colored rectangle displayed on the left margin of the other programmer's code editor (and vice versa), centered on the Y coordinate returned by the eye tracker (see Figure 1). The X coordinate is ignored, since in C#, there is rarely more than one code concept per line. Reducing the shared space to a 5-line span of code circumscribes the referential domain, allowing participants to use less specific language when reference to locations in code [30].

Similar to the hysteresis effect employed by a thermostat, if the eye moves within the lines circumscribed by the rectangle, the rectangle stays still to minimize the quick jumping movements typical to many visual gaze visualizations. If the eye moves outside the box a line or two up or down, the rectangle smoothly scrolls with it. When the eye jumps to a completely disjoint line, the box is re-centered on the new location. In addition, the rectangle drawn in our visualization changes colors. Normally, it is yellow, but becomes green when participants are looking at the same thing (defined as when their rectangles vertically overlap). This signal is designed to support shared intentionality (and shared attention) [48] by letting both programmers know that they are looking at the same part of the code at the same time.

STUDY METHOD

Participants

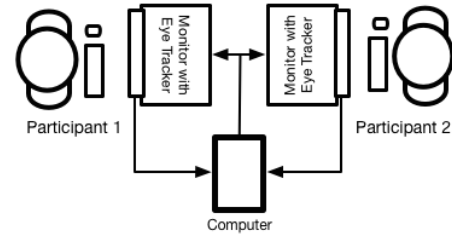Twenty-four software developers from a technology company who were comfortable programming in C# participated in this study. Participants were recruited through emailed announcements on Agile development mailing lists at the company. All participants were consented and received USD$50 compensation for participating. For the study, pairs of participants were asked to work together in remote pair programming teams.

Setup

From our survey, we found that the most common pairing setup was a single computer with a single, shared display. When survey respondents paired remotely, they created this environment using a Skype call with simultaneous screen sharing. Since our study intended to evaluate a visualization that looked different on each person's computer, we could not use Skype. Instead, we simulated a Skype call using a single computer with two displays placed back-to-back, so that the participants could each work at their own station without seeing one another, yet still talk aloud to communicate (see Figure 2).

Each of the two participants had their own display (Monitor 1 (TX300): 23" 1920x1080, and Monitor 2 (EyeX): 24" 1920x1200, both at 96 DPI), mouse, keyboard, and an attached remote eye tracker (a Tobii TX300 (around USD$45,000) and a Tobii EyeX (USD$139)) running at 30Hz.[1] We used Team-Player4 [14] software to enable each participant to see and manipulate their own independent mouse cursor on the single experimental computer. A GoPro Hero 3 camera was set up to the left of each participant to record their facial expressions as they worked. Both displays were screen-captured and merged with their respective GoPro video feed into a single 2x2 MP4 format movie (see Figure 3).

To simulate screen sharing and support our gaze visualization, we created a Visual Studio 2015 extension. The extension displays the project code simultaneously in two code editor windows whose text (Consolas, 13 pt) and user interface are slaved to one another, but are displayed on the two separate monitors. When one participant scrolls his/her window, the other window scrolls to the same place. If a participant highlights text on their window it is also highlighted on their partner's window. If the participant switches editor windows, an identical window would open on the other screen. The extension captures eye tracking information from two eye trackers simultaneously. It renders the gaze visualization in both code

---

[1]We would have liked to use two of the much cheaper Tobii EyeX trackers, but it is not possible to put more than one EyeX on a computer.
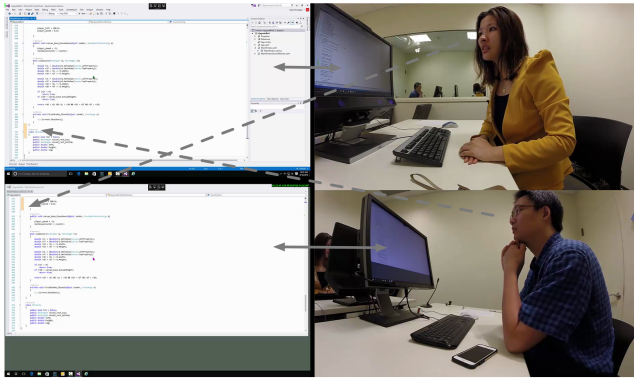
Figure 3. Combined Videos: Screen capture paired with monitor (indicted by horizontal arrows) and gaze of participant represented on the screen of their partner (indicated by diagonal arrows.)

Figure 4. Example of Flappy Bird Game

Code	Description
Deictic	When a participant uses a deictic reference such as: this, that, or here. For example, *"I think it's this one over here."*
Abstract	When a participant uses an abstract concept or broader category to refer to an object. For example, *"We need to change the variable."*
Gaze	When a participant directly refers to the gaze visualization to refer to an object. For example, *"Right where I am looking."*
Specific	When a participant uses a specific word or name to describe an object. For example, *"I think it is called in Update Score."*
Typing	When a participant is referring to the text that they are currently typing in the shared window. For example, *"You just made a typo."*
Line Num	When a participant uses a line number to refer to a location. For example, *"Fix the error on line 139."*
Select	When a participant highlights a word or section of text to direct their partner to it, usually this reference is paired with a deictic reference. For example, *"This section right here (highlighting code)."*

Table 1. Reference Codes and Descriptions

editor windows (the only thing that is not duplicated between the code windows) and logs eye tracking data to disk.

Tasks

We employed a 2x2 counterbalanced (gaze awareness and task), within-subjects experimental design. Participants worked on two refactoring tasks intended to "clean up" the C# code for a Flappy Bird-like game [40] (see Figure 4), with our gaze visualization turned on and off. The two tasks were equivalent in difficulty (pilot tests helped us fine-tune this) and took about 15 minutes each.

To introduce the participants to the code, they were asked to collaborate with one another to make small changes to the game, e.g. changing the sizes of the bird or obstacles. This task was unmeasured. Then, the participants worked together to accomplish two refactoring tasks while being recorded and logged.

After completing each task, each participant filled out a 13-question Likert-scale questionnaire about their feelings about their performance on the task. The survey questions were adapted from McCroskey and McCain [31] on task attractiveness and from Begel and Nachiappan [6] on the qualities of good pair programming partners. Each question asked the participant to state the extent to which they disagreed or agreed with statements about how they worked with their partner. For example, "I felt like my partner and I were on the same page most of the time," or "I felt that I did more work than my partner." At the end of the study, the pair was interviewed informally about their impressions of the gaze visualization to subjectively evaluate its design and utility.

Video Analysis

To understand how the gaze awareness visualization influenced communication between the pair, all videos were coded for references and acknowledgments. Every time a participant made a reference or verbally acknowledged a reference, the time was logged, the type was recorded (reference or acknowledgement), and it was given a description code. The description code indicated the form of the reference, and ranged from implicit (deictic) to explicit (text selection). See Figure 5 for a
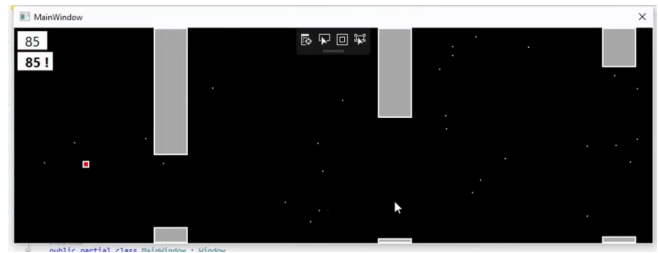
complete list and Table 1 for code descriptions. The acknowledgment description indicated a "yes" if the reference was verbally understood, or "confused" if recipient did not follow the reference.

Gaze Analysis

The main signal we analyzed from our log data was a *gaze overlap* signal. This is true if the two participants were looking at the same thing at the same time. Each participants' view area is defined as a sequence of five visible lines in the code editor, vertically centered on the participant's last eye gaze location. In each recorded sample, if any of the first partic-
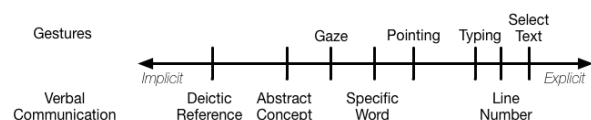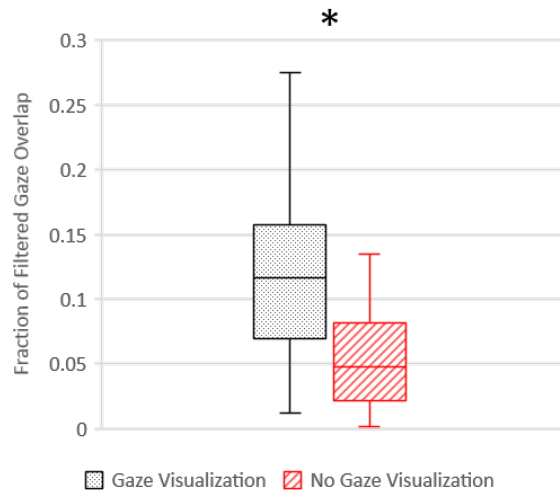

Figure 5. Reference Categories

Figure 6. Fraction of gaze overlap in each pairs' tasks in the two experimental conditions. The ∗ indicates that the two conditions are statistically significantly different from one another.

Figure 7. Fraction of references of each type uttered by the pair in the two experimental conditions. An ∗ is shown above reference types that are statistically significantly different from one another.

ipant's view area's lines overlapped the second participant's view area, we said that the gaze overlap was true. However, we filtered out gaze overlaps that lasted less than 100 ms to avoid spurious indications of gaze overlaps when the eye jumped around a lot. All of the results in Section 7 use this filtered gaze overlap measure.

Next, we measured the time from a person's utterance of a reference to a location on the screen to when that reference was verbally acknowledged by the other person. However, since the goal of the visualization is to help the participants look something when the other person referred to it, this analysis also considers shared gaze as an form of acknowledgment. When there was no explicit verbal acknowledgement, we defined the first filtered gaze overlap within 10 seconds as an implicit acknowledgement of the reference.

In addition to determining whether or not a reference was acknowledged, we recorded the time it took for that implicit or explicit acknowledgement to occur. If neither implicit or explicit acknowledgement was found, we considered that reference to be unacknowledged and unsuccessful.

RESULTS
In this section, we present the results of our data analysis. We looked at three questions.

1. How much time did the participants spend concurrently looking at the same thing?

2. What kinds of references were used by the participants?

3. How successful were references to locations in the code?

Timing
The basic premise of this work is that pairs that spend more time looking at the same thing work better together. For each pair of par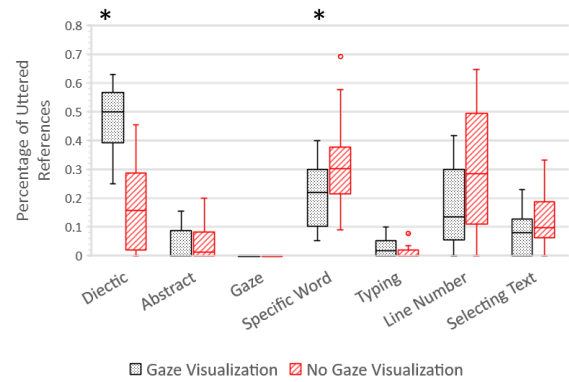ticipants, we counted the total amount of time spent looking at the same thing (i.e. sum of filtered gaze overlap time) and divided it by the total time spent on each task to produce the fraction of time spent looking at the same thing. This is shown in Figure 6. Using JMP, we computed a one-way ANOVA to compare the effect of the gaze visualization on the fraction of filtered gaze overlap time, both with and without the gaze visualization. The mean score for the gaze visualization (M = 11.9%, SD = 0.6%) was significantly greater than without the gaze visualization (M = 5.9%, SD = 0.6%) [F(1,35) = 58.243, p < 0.0001]. This result suggests that in each task, participants spent 6% more time looking at the same things when using the gaze visualization.

Prior work has shown a similar effect in a primarily visual collaborative learning task [44]. Related studies suggest that pairs perform collaborative tasks better the more that they attend to the same things at the same time [9, 18, 33]. To further understand how this affect communication and coordination, we check that the increased gaze overlap is correlated with a *functional* improvement in collaboration.

References
As each person worked together on the refactoring tasks, each had to repeatedly make references to locations in the code to signal to their partner where to find or place some relevant code. These 470 total references span a continuum of the explicitness of the grounding inherent in the utterance (as shown in Figure 5). We counted the number of utterances of each reference type made by participants as they completed their tasks. Since each pair performed the same set of tasks, we normalized the values to produce the fraction of each utterance type made by participants over their tasks. The distribution of these ratios, divided by experimental condition, can be seen in Figure 7.

A oneway ANOVA indicates that the fraction of *deictic* references is significantly greater with the gaze visualization turned on (M = 46% SD = 4%) than when turned off (M = 18% SD = 4%) [F(1,11) = 24.9663, p = 0.0004]. The fraction of more explicit, *specific word* references is lower with the gaze visualization (M = 21% SD = 3%) than when it was not avail-
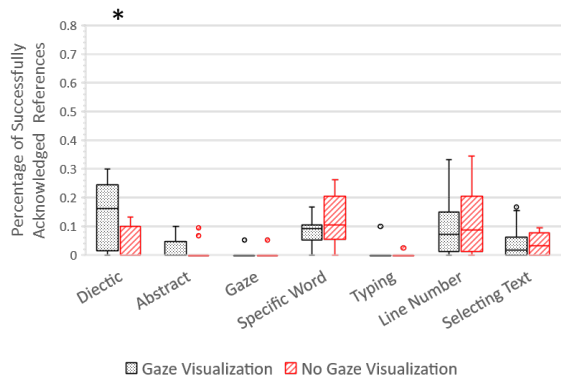
Figure 8. Fraction of *successfully* acknowledged references of each type uttered by the pair in the two experimental conditions. An ∗ is shown above reference types that are statistically significantly different from one another.

able (M = 34% SD = 3%) [F(1,11) = 7.1602, p = 0.0216]. This means the pairs were more comfortable using the more implicit, deictic references when they knew that they were able to see where the other was looking.

Acknowledged References

The reference fractions shown in Figure 7 show *all* the references that were attempted by the pairs, regardless of whether they were successfully understood and acknowledged. Over all reference types, there are many unsuccessful references. Only 37% of the attempted 249 references without the gaze visualization were acknowledged successfully. With the gaze visualization, 7% more (44%) of the 221 attempted references were. We now take a closer look at the overall 190 *successful* references to see if the gaze visualization helped the participants to communicate more effectively or efficiently.

In Figure 8, we show the distribution of successful references of each type, divided by condition. A one-way ANOVA shows that the proportion of successful deictic references is significantly higher when the participants saw the gaze visualization (M = 14.3% SD = 2.1%) than without it (M = 3.6% SD = 2.1%) [F(1,11) = 12.3214, p = 0.0049]. The other reference types show no significant differences between conditions. This means that people not only made more deictic references with the gaze visualization, but they were also 19% better at communicating them successfully.

Finally, we look at the time it took to communicate a reference to see whether the gaze visualization had an impact on making references quicker to acknowledge. Across all references, a one-way ANOVA shows that successful references were acknowledged more quickly with the gaze visualization (M = 1.7 sec SD = 0.11 sec) than without it (M = 2.1 sec SD = 0.1 sec) [F(1, 417) = 7.6624, p = 0.0058]. Surprisingly, when we split the data by reference type, deictic references are not significantly faster, but line number and specific name references are. A one-way ANOVA shows that line number references are faster with the gaze visualization (M = 1.9 sec SD = 0.3 sec) than without it (M = 2.5 sec SD = 0.2 sec) F(1,186) =

p = 0.0496]. It also shows that specific name references are faster with the gaze visualization (M = 1.4 sec SD = 0.2 sec) than without it (M = 2.3 sec SD = 0.2 sec) [F(1,212) = 7.7144, p = 0.0060]. The other reference types are not significantly faster or slower between the two conditions.

In summary, our quantitative results showed that participants used more deictic references with the gaze visualization, suggesting that they are using the visualization as a referential pointer. This is consistent with prior work [3, 13]. Our study adds to this body of knowledge by showing that deictic references are successfully acknowledged more often with the gaze visualization, and that references are acknowledged faster, suggesting that it is useful aid for coordination.

Completion Time

There was no significant different in completion time between the gaze visualization and no gaze visualization conditions. Each pair was allotted 15 minutes to complete the coding task and all pairs used the entire time and were stopped by the researcher to change tasks. One explanation for the lack of difference is our sample population. We believe that the pairs worked for the entire allotted time because of their own personal quality standards as professional developers. Additionally, our informal instructions were intended to encourage natural collaboration and may have contributed to the continued effort to "clean up" the code after the basic refactoring task was completed.

Post-Task Questionnaire

A oneway ANOVA on the results from our post task survey indicate that when the gaze awareness visualization was displayed, participants said that their partner was able to focus more on the task (M = 4.57 SD = 0.05) compared to without the visualization (M = 4.42 SD = 0.05) [F(1,25) = 4.54, p = 0.04]. This suggests that participants were looking at the gaze visualization to gain insight about what their partner was attending to. Similarly, each partner knew when the other was looking at the document since the visualization was only displayed at that time. Participants might have been using this information to infer that their partner was on task, compared to without the visualization in which that added information is not available.

None of responses to the other questions were significantly different between conditions, though they may be subject to experimental bias. This suggests that the addition of the gaze awareness visualization might not influence the perceived quality of interaction between pair programmers. The benefits of shared gaze awareness are subtle, and might need to be used for extended periods of time before the perceived value is apparent.

Interview Results

At the end of the study, our interviews with the participants revealed that when they were aware that they were making use of the visualization, they used it to help communicate about locations in the and establish a shared understanding of what the other person is looking at.

"I can get a context of where he is looking at the time, so it's a replacement for pointing." (Pair 3 — Participant 2)

Participants told us that they typically resort to using line numbers when talking about locations in code, but that they preferred to use our gaze visualization because it was easier. This is reflected in our quantitative analysis of references, which showed that participants used more deictic references with the gaze visualization, but more specific word references without it. This suggests that the visualization assisted coordination by allowing pairs to use deictic references and communicate without grounding on items first [11, 13, 19].

Participants also found the color changing to be a useful indicator that both partners were on the same page.

"It was nice when it was green because we know we are looking at the same line...talking about the same line of code." (Pair 4 —Participant 1)

Participants spent more time looking at the same areas of code with the gaze visualization, suggesting that they were tightly coordinated and attended to the same information at the same time.

Participants said that the gaze visualization helped them to understand what their partner was talking about and attending to. None of the pairs expressed experiencing any difficulty interpreting the visualization, suggesting that a 5-line range of code is a sufficient range for assisting communication through successfully circumscribing the referential domain.

Interestingly, half of the participants expressed that they did not notice the visualization, stating that, *"I think I just forgot about it!"* (P7S2). Although they might not have been explicitly aware of the gaze visualization or how it was affecting their work, we did see changes in behavior. This result is not surprising when we realize that in collocated collaboration, gaze is a subtle signal, and we designed our gaze visualization to reflect that.

DISCUSSION

The addition of gaze awareness to remote collaborative work is a useful aid for coordination. The novel gaze visualization described in this work helped remote pair programmers communicate more effectively. These pairs used more deictic references which were more successfully acknowledged with the gaze visualization, suggesting that pairs used gaze awareness to effectively ground on items in the code. Pairs were faster to acknowledge references with the gaze visualization, suggesting that they understood what their partner was referring to. This may be due to faster grounding, or they could have already been in sync with their partner and were already looking at the same location when their partner communicated with them.

While we applied our tool to the pair programming domain, it could easily be extended to work with other types of documents. Our left-aligned visualization would work well with other types of text based documents. Additionally, the type of visualization evaluated here could be re-imagined to fit into a scroll bar, similar to visualizations explored by Hill and colleagues [25]. This would enable the visualization to work for documents whose contents are mirrored, but whose UI is not.

The effectiveness of the novel representation of gaze that we evaluate in this work demonstrates that gaze visualization design should take task structure into account. In the the context of pair programming, a non-obtrusive and non-distracting representation was appropriate for the task and well received by our participants. After the experimental session, when participants told us their subjective thoughts about the gaze visualization, they recognized that sometimes they wanted the visualization to show up only when they were intentionally trying to communicate a location to their partner.

"I think the bigger thing for me is not where is he looking all the time but where does he want to have me look." (Pair 6 — Participant 2)

Future designs may consider a representation whose visibility is tied to the conversation between the partners. Using deictic references to inform the display of gaze awareness could provide more intentional cues and filter out non-signalling eye movements.

While eye trackers have become very inexpensive, they are not yet ubiquitous. However, the pair programming teams we observed were not shy about asking their employers for other specialized equipment to support pairing that cost many thousands of dollars. We can easily imagine remote pair programmers intent on making their programming experience more effective and efficient spending a few hundred dollars to do so.

Shared gaze awareness is a way to enhance the remote collaboration experience by providing people with an on-screen representation of where their partner is looking. This work fine-tunes gaze awareness with a simple and unobtrusive design that effectively supports coordination and communication. Our work aims to integrate novel and practical features into remote work in ways that have the potential to make it even better than collocated work [26].

Threats to Validity

All experiments contain tradeoffs in the design and implementation that affect their internal and external validity. The eye trackers we used are inherently inaccurate devices. In ideal situations, they can be accurate to 0.5 degrees, but as participants move around, the accuracy degrades. In this experiment, we had to rely on the participants to tell us when they believed the accuracy had degraded enough to require recalibrating the eye trackers. This happened to only 3 out of 24 participants in the study, and was done in between tasks. Based on our past experiences with eye trackers, we designed our gaze visualization to be five lines tall to account for the vertical inaccuracy we have seen in "perfectly" calibrated trackers.

Furthermore, we are using two different eye trackers in this experiment. Ideally, we would have used two inexpensive eye trackers to model an affordable system, however given the technical constraints imposed by tracker manufacturer, we had to use two different models from the same company. Both eye trackers are advertised to be accurate to within 0.5 degree,

and we used them at the same sampling rate so they should be comparable. However, this more complex experimental setup is a limitation [37].

In this experiment, we simulated a remote work environment with a single computer and multiple monitors instead of the commonly used Skype with screen sharing to test out our theories behind the gaze awareness visualization. Our simulation performs better than the alternative because there is no latency in the audio (participants could just speak to one another in the same room) or the display (since they use the same computer). We did this because we could not make Skype's screen sharing show different displays to each participant. Latencies do cause some communication problems that we did not observe in our experiment. A future implementation of our tool would have to take into account these latencies to accurately assess when the participants are looking at the same thing at the same time.

Finally, we studied *remote* pair programming, not *distributed* pair programming. In remote pair programming, both partners use mirrored displays and look at same things together. Distributed pair programming loosens the mirroring requirement, supporting programmers working on the same codebase, but each having their own unique view of the system. Our visualization is based on a mirrored display, so it would not easily generalize to a distributed pair programming setup.

CONCLUSION
In this paper, we describe a study of remote pair programming. Through observations, we identified that remote pairs would have significant difficulties communicating about on-screen code locations. We took advantage of cheap eye tracking technology to develop a novel, dual eye tracking, gaze awareness visualization that helps pairs to see what the other person is looking at in a code document. We evaluated the visualization in an experiment in which pair programmers refactored source code while working in a simulated remote work environment. When the programmers used the visualization, they were more likely to look at the same thing at the same time. They were both faster and more successful at communicating using implicit, deictic references, and used them more frequently compared with references to explicit line number or specific words on the screen. Most participants recognized the value of the visualization, though some were not even consciously aware that the visualization had changed their behavior and helped them communicate efficiently.

Remote collaboration remains difficult for many people, especially when adapting a working style that originated and is typically practiced in collocated settings. When two people work together to create and edit code, the perceived and real challenges of communicating effectively and efficiently can inhibit them from taking advantage of the flexibility and accessibility affording to them by remote work. Our gaze awareness visualization takes us a step closer to removing those barriers.

ACKNOWLEDGEMENTS

REFERENCES
1. Paul D Allopenna, James S Magnuson, and Michael K Tanenhaus. 1998. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of memory and language* 38, 4 (1998), 419–439.

2. Reynold Bailey, Ann McNamara, Nisha Sudarsanam, and Cindy Grimm. 2009. Subtle gaze direction. *ACM Transactions on Graphics (TOG)* 28, 4 (2009), 100.

3. Ellen Gurman Bard, Robin L Hill, Mary Ellen Foster, and Manabu Arai. 2014. Tuning accessibility of referring expressions in situated dialogue. *Language, Cognition and Neuroscience* 29, 8 (2014), 928–949.

4. Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, and others. 2001. Principles behind the Agile Manifesto. (2001). `http://agilemanifesto.org/principles.html`

5. Roman Bednarik, Andrey Shipilov, and Sami Pietinen. 2011. Bidirectional gaze in remote computer mediated collaboration: Setup and initial results from pair-programming. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. ACM, 597–600.

6. Andrew Begel and Nachiappan Nagappan. 2008. Pair Programming: What's in It for Me?. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08)*. ACM, New York, NY, USA, 120–128.

7. Susan E Brennan, Xin Chen, Christopher A Dickinson, Mark B Neider, and Gregory J Zelinsky. 2008. Coordinating cognition: The costs and benefits of shared gaze during collaborative search. *Cognition* 106, 3 (2008), 1465–1477.

8. Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye Movements in Code Reading: Relaxing the Linear Order. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension (ICPC '15)*. IEEE Computer Society, Washington, DC, USA, 255–265.

9. Mauro Cherubini, Marc-Antoine Nüssli, and Pierre Dillenbourg. 2008. Deixis and gaze in collaborative work at a distance (over a shared map): a computational model to detect misunderstandings. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. ACM, 173–180.

10. Jan Chong and Rosanne Siino. 2006. Interruptions on Software Teams: A Comparison of Paired and Solo Programmers. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06)*. ACM, New York, NY, USA, 29–38.

11. Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. *Perspectives on socially shared cognition* 13, 1991 (1991), 127–149.

12. Alistair Cockburn and Laurie Williams. 2001. Extreme Programming Examined. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Chapter The Costs and Benefits of Pair Programming, 223–243.

13. Sarah D'Angelo and Darren Gergle. 2016. Gazed and Confused: Understanding and Designing Shared Gaze for Remote Collaboration. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2492–2496.

14. Dicolab. 2016. Dicolab - multi-user, multi-cursor collaboration. (2016). `https://www.dicolab.com/products_teamplayer.html`

15. Jörg Edelmann, Philipp Mock, Andreas Schilling, and Peter Gerjets. 2013. Preserving Non-verbal Features of Face-to-Face Communication for Remote Collaboration. In *International Conference on Cooperative Design, Visualization and Engineering*. Springer, 27–34.

16. Ilenia Fronza, Alberto Sillitti, and Giancarlo Succi. 2009. An Interpretation of the Results of the Analysis of Pair Programming During Novices Integration in a Team. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09)*. IEEE Computer Society, Washington, DC, USA, 225–235.

17. Susan R Fussell, Leslie D Setlock, and Elizabeth M Parker. 2003. Where do helpers look? Gaze targets during collaborative physical tasks. In *CHI'03 Extended Abstracts on Human Factors in Computing Systems*. ACM, 768–769.

18. Darren Gergle and Alan T. Clark. 2011. See What I'M Saying?: Using Dyadic Mobile Eye Tracking to Study Collaborative Reference. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11)*. ACM, New York, NY, USA, 435–444.

19. Darren Gergle, Robert E Kraut, and Susan R Fussell. 2013. Using visual information for grounding and awareness in collaborative tasks. *Human–Computer Interaction* 28, 1 (2013), 1–39.

20. Zenzi M Griffin and Kathryn Bock. 2000. What the eyes say about speaking. *Psychological science* 11, 4 (2000), 274–279.

21. Karl Gyllstrom and David Stotts. 2005. Facetop: Integrated semi-transparent video for enhanced natural pointing in shared screen collaboration. 15 (May 2005), 1–10.

22. Joy E Hanna and Susan E Brennan. 2007. Speakers' eye gaze disambiguates referring expressions early during face-to-face conversation. *Journal of Memory and Language* 57, 4 (2007), 596–615.

23. Jo E Hannay, Tore Dybå, Erik Arisholm, and Dag IK Sjøberg. 2009. The effectiveness of pair programming: A meta-analysis. *Information and Software Technology* 51, 7 (2009), 1110–1122.

24. Keita Higuch, Ryo Yonetani, and Yoichi Sato. 2016. Can Eye Help You?: Effects of Visualizing Eye Fixations on Remote Collaboration Scenarios for Physical Tasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5180–5190.

25. William C. Hill, James D. Hollan, Dave Wroblewski, and Tim McCandless. 1992. Edit Wear and Read Wear. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 3–9.

26. Jim Hollan and Scott Stornetta. 1992. Beyond Being There. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 119–125.

27. Patrick Jermann, Darren Gergle, Roman Bednarik, and Susan Brennan. 2012. Duet 2012: Workshop on dual eye tracking in CSCW. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*. ACM, 23–24.

28. Patrick Jermann and Marc-Antoine Nüssli. 2012. Effects of sharing text selections on gaze cross-recurrence and interaction quality in a pair programming task. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 1125–1134.

29. Sasa Junuzovic, Prasun Dewan, and Yong Rui. 2007. Read, write, and navigation awareness in realistic multi-view collaborations. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 494–503.

30. Robert E Kraut, Darren Gergle, and Susan R Fussell. 2002. The use of visual information in shared visual spaces: Informing the development of virtual co-presence. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM, 31–40.

31. James C. McCroskey and Thomas A. McCain. 1974. The measurement of interpersonal attraction. *Speech Monographs* 41, 3 (1974), 261–266.

32. Ann McNamara, Reynold Bailey, and Cindy Grimm. 2009. Search task performance using subtle gaze direction with the presence of distractions. *ACM Transactions on Applied Perception (TAP)* 6, 3 (2009), 17.

33. Marc-Antoine Nüssli, Patrick Jermann, Mirweis Sangin, and Pierre Dillenbourg. 2009. Collaboration and abstract representations: towards predictive models based on raw speech and eye-tracking data. In *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*. International Society of the Learning Sciences, 78–82.

34. Jiazhi Ou, Lui Min Oh, Susan R Fussell, Tal Blum, and Jie Yang. 2008. Predicting visual focus of attention from intention in remote collaborative tasks. *IEEE Transactions on Multimedia* 10, 6 (2008), 1034–1045.

35. Christopher Palmer and Bonita Sharif. 2016. Towards Automating Fixation Correction for Source Code. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications (ETRA '16)*. ACM, New York, NY, USA, 65–68.

36. Mauro C Pichiliani, Celso M Hirata, Fabricio S Soares, and Carlos HQ Forster. 2008. Teleeye: An awareness widget for providing the focus of attention in collaborative editing systems. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 258–270.

37. Sami Pietinen, Roman Bednarik, Tatiana Glotova, Vesa Tenhunen, and Markku Tukiainen. 2008. A method to study visual attention aspects of collaboration: eye-tracking pair programmers simultaneously. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. ACM, 39–42.

38. Sami Pietinen, Roman Bednarik, and Markku Tukiainen. 2010. Shared visual attention in collaborative programming: a descriptive analysis. In *proceedings of the 2010 ICSE workshop on cooperative and human aspects of software engineering*. ACM, 21–24.

39. Laura Plonka and Janet van der Linden. 2012. Why developers don't pair more often. In *Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE '12)*. IEEE Press, Piscataway, NJ, USA, 123–125.

40. George Powell. 2016. Flappy Clone. (2016). `https://github.com/georgepowell/flappy_clone`

41. Daniel C Richardson and Rick Dale. 2005. Looking to understand: The coupling between speakers' and listeners' eye movements and its relationship to discourse comprehension. *Cognitive science* 29, 6 (2005), 1045–1060.

42. Daniel C Richardson, Rick Dale, and Natasha Z Kirkham. 2007. The art of conversation is coordination common ground and the coupling of eye movements during dialogue. *Psychological science* 18, 5 (2007), 407–413.

43. Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk. 2010. Saros: An Eclipse Plug-in for Distributed Party Programming. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '10)*. ACM, New York, NY, USA, 48–55.

44. Bertrand Schneider and Roy Pea. 2013. Real-time mutual gaze perception enhances collaborative learning and collaboration quality. *International Journal of Computer-Supported Collaborative Learning* 8, 4 (2013), 375–397.

45. Kshitij Sharma, Patrick Jermann, Marc-Antoine Nüssli, and Pierre Dillenbourg. 2013. Understanding collaborative program comprehension: Interlacing gaze and dialogues. In *Computer Supported Collaborative Learning (CSCL 2013)*.

46. Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. 2012. Understanding the Impact of Pair Programming on Developers Attention: A Case Study on a Large Industrial Experimentation. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 1094–1101.

47. Randy Stein and Susan E Brennan. 2004. Another person's eye gaze as a cue in solving programming problems. In *Proceedings of the 6th international conference on Multimodal interfaces*. ACM, 9–15.

48. Josh Tenenberg, Wolff-Michael Roth, and David Socha. 2016. From I-Awareness to We-Awareness in CSCW. *Computer Supported Cooperative Work* 25, 4-5 (Oct. 2016), 235–278.

49. Laurie A Williams and Robert R Kessler. 2000. All I really need to know about pair programming I learned in kindergarten. *Commun. ACM* 43, 5 (2000), 108–114.