

Supporting Independent Learning in a Programming Environment by Integrating Community-Based Help

Michelle Ichinco
Department of Computer Science, Tufts University
michelle.ichinco@tufts.edu

ABSTRACT

Since students learning programming do not have the support network of a teacher and classroom, help systems require another type of community to fill in for what would normally exist within schools. One type of substitute support network is an online community of users who are all working in the same programming environment and have various different skill levels. I describe a community-based help tool that allows users to request answers to specific questions about their code and receive responses in text and/or code all without having to leave the programming environment. The goal of this help tool is to keep students engaged in programming by providing answers to users' questions.

INTRODUCTION

With the amount of technology present in our everyday lives, as well as the increasing number of jobs which could benefit from employees with computer science experience, it would make sense for students to be exposed to computer science at a young age and begin to study it in middle and high schools. In 2005, Bureau of Labor Statistics estimated that over 90 million Americans will work with computers by 2012 and 13 million Americans will describe themselves as programmers, while there will still be fewer than 3 million professional programmers [3]. Despite the increasing number of Americans who would benefit from having computer science as a part of their formal education, most middle and high schools lack the resources needed to teach computer science. By the time students reach college, many have already been scared away from programming, due to its perceived difficulty. In fact, it is often during the middle school years that students, especially girls, decide whether or not to study math and science [4,6].

Due to the lack of support for computer science education in schools, we need a tool in which middle school students can learn programming independent of teachers and classrooms. Looking Glass, a successor to Storytelling Alice, is being developed to fit this niche. Looking Glass is a programming environment aimed at middle school students, in which users tell stories by programming. Since programming performance relies on the amount of time a user spends programming, it is especially important for Looking Glass to keep students engaged in programming for extended periods of time [2]. One barrier to this extended engagement is the lack of formal education in computer science, which would normally support the learning process by answering students' questions.

Because most middle school students do not have access to formal computer science education, Looking Glass needs a tool that will answer users' questions and keep them engaged in learning. One common question-answer system is the online forum, in which users can ask textual questions and receive responses from other users. While online forums may be successful for certain use-cases, there

are some problems that arise when attempting to apply the help forum to middle school students using Looking Glass:

- Difficulty in finding help: In order to access an online forum, a Looking Glass user must open their web browser and know how to find the forum. Having to leave the environment they are working in could prevent users from seeking help.
- Effort required to ask for help: Once a user accesses an online forum, in order to describe their question, they might want to look at the code they are working on, along with their question, for clarity. In order to look at their current code, a user has to flip back and forth between the internet browser and Looking Glass, making asking a question more complex.
- Difficulty in attaching code: If a user would like to attach their code, they would have to take a screen shot of the code and attach it, since Looking Glass code is not text. In order to reproduce the code from a screen shot, a user answering the question would actually have to rebuild the program, which makes working with the code to find a solution extremely difficult.
- Wrong answers from unskilled users: Forums also lack a system of ensuring that questions are answered by users with relevant knowledge and skills. A mid-level user might look at a few of the questions and think that they do not know any of the answers and give up, while the beginner questions are simply on pages three and four. A beginner user might try to answer an advanced question, but instead give a wrong answer that causes more frustration.
- Inability to understand questions: Forums commonly allow open ended questions with very little direction. This freedom can cause off-topic and unclear questions that are difficult to understand or answer.
- Need to try out the code: Often, when trying to answer a programming question, the user answering the question needs to actually run the code to see how it works and then try out a couple of different changes to see what works. However, in a forum, Looking Glass code would have to be a screen shot, so it would require a lot of effort to recreate the project in order to try to fix it. Then, if they want to reattach the fixed code, they also run into the “Difficulty in attaching code” issue, which might prevent them from sending it, making the answer less complete.
- Inability to understand answers or lack of desire to read: If attaching fixed code is not an option, the question-asker must rely solely on a textual response to their question. If the answerer either has the wrong answer (possibly due to being an unskilled user), or cannot communicate the answer through text, the question asker will not be able to understand the answer.

Solving these problems necessitates a different kind of help system, such that asking questions and receiving answers is an easy and successful process.

The community-based help system in Looking Glass seeks to improve on the idea of the help forum in order to encourage users to ask for help and answer questions, as well as to provide users with helpful responses that allow them to continue programming and extend their engagement in Looking Glass. This help system is integrated into the Looking Glass programming environment, rather than existing on the internet. Users asking a question are prompted to ask their question in a specific manner and to select the piece of code they would like help with. Users answering the question receive the entire context of the question and can edit it and send it back. Only users with a higher skill level than the user who asked the question will be prompted to answer it, in hopes that users with higher skill levels would be better able to answer the question.

RELATED WORK

Past study on help-seeking behavior has identified a process students perform when asking for help. They first become aware that they have a problem they cannot solve. The user then decides that they would like to request help in some manner and that they will use the help system provided. Next, the user must successfully perform the help request and receive a response. Then, the user can evaluate whether the answer they received was useful [1]. Research shows that often, a user who becomes aware that they do need help will often choose not to ask their question, or do so inefficiently so that they never receive a response to their problem.

Some students choose not to seek help due to a fear of their peers viewing them as inferior. Another barrier to students asking for help is the amount of effort required to request assistance. Studies show that when a student is trying to work on a task and ask help at the same time, the cognitive effort required may impact learning [1]. In order to encourage users to ask questions, the system should be anonymous and be simple and intuitive.

Another issue in being able to ask useful questions and receive correct answers involves the ability to include context in the questions and answers. If a question or answer is not tied to the context of the problem, users may struggle to understand a textual response [1]. An unclear question is almost impossible to answer. Even if a response is submitted, it is highly unlikely that it would actually answer the question. Overcoming ineffective questions and answers requires a system that allows the easy inclusion of the context of the question and a system that guides users to clearly communicate their questions. A system that can help users to ask clear questions, as well as include the code along with the question would give other users much more information about the problem, making a question easier, and thus more likely, to be solved. If a system could help users to ask meaningful and specific questions and allow users to include their code, hopefully those asking for help would receive more useful and direct responses to their questions.

A study on Naver Knowledge-iN, a popular South Korean question and answer community, found that a forum for answering questions was extremely effective for questions about history, politics, technology and other general questions. Research found that using a point system to encourage users to provide good answers to questions worked effectively [5]. Though the point system does motivate users to answer more questions, there are still problems in how users ask questions. Researchers found that often, questions are not related to the topic they are posted under, are too complicated, or are in some other way unanswerable by the user base that would be reading and attempting to answer the question.

Current community-based help systems, such as forums, lack the features needed to overcome barriers to effective help seeking behavior. They are not created so that they are necessarily easy to find or use. Existing help systems also do not provide a simple method for including the context of a programming question, nor do they in any way prevent or discourage unclear or off-topic questions. Forums have no system to ensure that users with the appropriate knowledge are answering the questions, allowing any user to send a response. The integrated help system I will describe, designed for Looking Glass, seeks to address these issues in order to answer users' questions and increase learning of programming.

INTEGRATED COMMUNITY-BASED HELP TOOL

Looking Glass

Looking Glass is a programming environment designed for middle school students in which they can create 3D animations. Users drag and drop lines of code in order to create their story, which prevents users from having syntax errors. Thus, all questions users have about how to accomplish a task in Looking Glass will be semantic, rather than syntactic. While students might be able to use static information to assist with syntactic errors, semantic questions often cannot be answered as easily and require a support network such as a teacher to answer complex questions.

The Community

The online community (currently under development) is a website where users can share their animations and view other users' work. It provides a network of users who can answer other users' questions. Once there are users involved in the community, some users will inevitably understand concepts that other users do not understand. A hierarchy of skills will develop throughout the community, so that each higher skill level can answer the questions of the levels below.

The help system integrated into Looking Glass will provide the communication necessary between users in the programming environment and the community. When a user asks a question, members of the community will be prompted, from within Looking Glass, to respond with a solution.

Help System

When designing a help system within a programming environment in which users have no other human support, it is crucial for the design to minimize difficulty of use and to maximize the number of successful help requests.



Figure 1: Looking Glass Action Editor

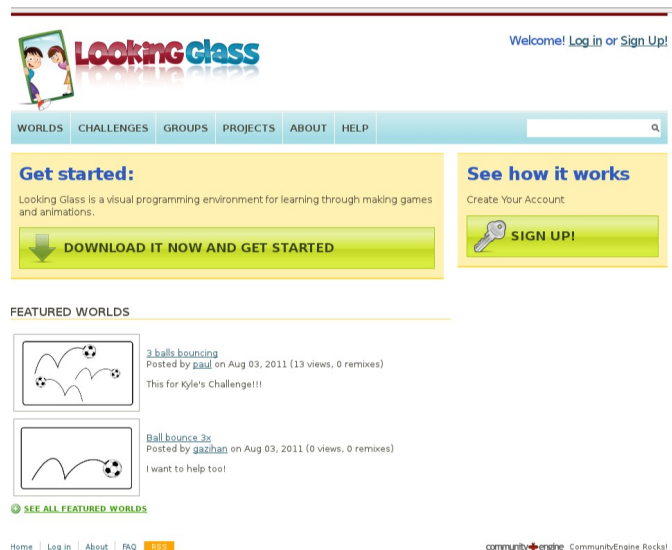


Figure 2: Looking Glass Community Website

The integration of the help system into Looking Glass is key in improving usability and encouraging students to ask questions. Having the help system available in the action editor, where users will first discover that they have a question, makes it highly visible and easy to find. Placing the system right next to the code editor allows users to look at their code at the same time they are typing their question. Making the help tool easier to find when users need assistance would hopefully encourage users to utilize the feature more than if they had to locate it on the internet. Essentially, integrating the help system in Looking Glass seeks to decrease the amount of effort required for users to ask for help. Making it easier to ask for help should increase the number of users who choose to do so.

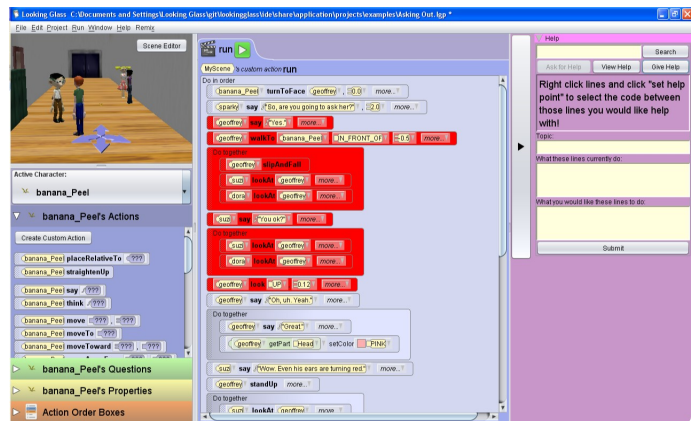


Figure 3: Action Editor with help tool and code selection

This help system seeks to make questions easier for users to answer by enabling the easy attaching of code, as well as by putting constraints on how questions should be asked. Hopefully, questions and answers will be easier to understand if the code for both can be attached with little effort. Furthermore, if the system prompts users to phrase their questions in a specific way, the questions should be clearer and more direct.

Users should receive more helpful answers if their questions are clearer and both sides of the system can view the code. Another method used to improve answers in this help system is to ensure that only users with more knowledge than asker of the question can answer it. This should increase the quality of answers, as users will only be shown questions below their skill level.

By connecting with other community members, a user of this help system can access four help features: searching the database of answered questions, requesting help from another community member, answering another user's question, and viewing answers to their questions. In this case, it is important to attempt to overcome the issues in other help systems, in which users either don't ask questions, ask unanswerable questions, or do not receive helpful responses. Thus, each feature of the help system was designed in order to motivate users to ask question, motivate users to answer questions, and help users to ask clear questions.

Searching

Imagine a Looking Glass user named Julie. Julie has just started using Looking Glass and wonders how to make a pig turn. Julie can type "turn" into the search box. The results returned to her will be other questions related to the term "turn," which have useful responses. Julie can then browse the various questions and answers, looking at both the text and the code in order to find a solution to her question.

The ability to search for previous questions makes seeking help accessible. A user does not need to commit to typing out a question and waiting for a response. With an active community of question askers and answerers, the search function should give a user immediate access to various questions and

answers related to their search term. Since users asking for help can view both the question and answer code, the user who searches for help may be able to gain a better understanding of the context and answer than a user who only has text. Having the search box integrated into Looking Glass also means that opening these files is simple and direct.

This search feature is likely similar to many help systems, but its location in the programming environment makes it unique and more effective. Since a user searching for assistance can view the contexts of the question and answer, they receive more information than a user searching an online forum.

Asking a Question

Later, perhaps Julie has figured out how to make the piggy turn. She has created a small scene in which the piggy completes a series of actions. However, Julie would like the piggy to turn, Look Thoughtful, and say “this way” all at the same time. This question relates specifically to Julie's scene, so she would choose to submit a request for help in the “Ask for Help” section.



Figure 4: Code Selection

In order to do this, Julie can select the lines of code she wants help with. By choosing the beginning of the question, or the “turn” line, and the end of the question, or the “say” line, all of the lines in between are selected.

Julie can then type in the text of her question. The three text boxes ask her to input a topic, a short description of what the lines of code currently do, and what she would like them to do.

There are two main differences between this help system and forums: the addition of suggested question inputs and code selection. The titles for the text boxes in the “Ask for Help” section were chosen in order to give users purpose in their question. Without the constraints, a middle school student may not know how to phrase their question. Adding constraints, in theory, will give middle school students direction in their questions. The goal is to encourage users to ask clear, concise questions that describe their problem in a way that another middle school student can answer. This is crucial in a community-based system where both those asking questions and those answering questions are in middle school and are often inexperienced with programming and teaching others.

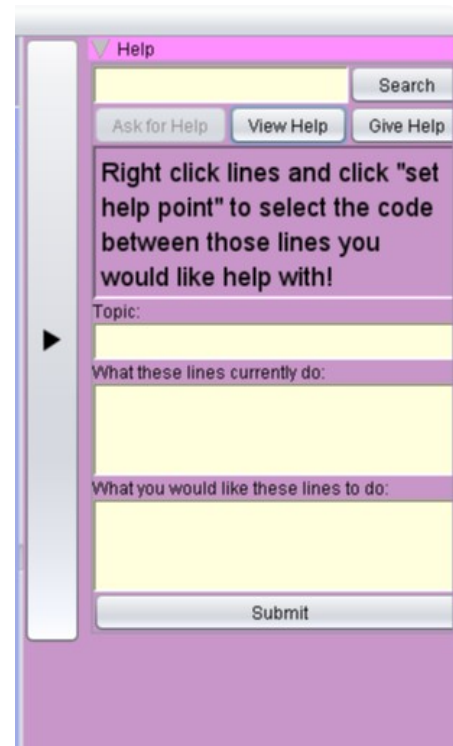


Figure 5: Ask for help

Another key feature of this help system is the user's ability to highlight the code the user is asking about. Making code selection easy and available should increase the number of users who attach their code. Having the code attached gives the question valuable context, making it easier for another user to understand the question and answer it. The differences between this method for asking questions and the open-ended nature of forums should improve the clarity of questions being asked. For community-based help, clarity of questions is crucial in order for other users to be able to answer the questions. The questions submitted here are also anonymous, so users have no need to fear the embarrassment of asking a stupid question.

Answering a Question

While Julie is struggling with how to make various events take place concurrently, another user, Ann, has been working in Looking Glass for a while and has used this skill many times. Ann has just received help on another question and decides to look at other questions to see if she knows how to answer any. When she sees the topic, "make things happen at the same time," she decides to open the world, since she's pretty sure she knows how to fix it.

If Ann did not want to answer this question, she could have clicked "View Another Question." All of the questions that appear in her "Give Help" section are ones that she should be able to answer, based on her skill level. Only showing Ann questions that are around or just below her skill level may increase Ann's interest and ability to answer a question. If Ann only recently learned how to use a "do-together" block, she might be excited to show off her knowledge. Furthermore, if all of the questions are based on topics she already knows, it is more likely that she will give clear and correct answers.

When Ann opens Julie's project, Ann can see the code selection and make the changes necessary, so that the pig turns, looks thoughtful, and speaks at the same time. She can then type a response such as "use a do-together!" and then submit. Submitting also attaches Ann's edited code to the answer.

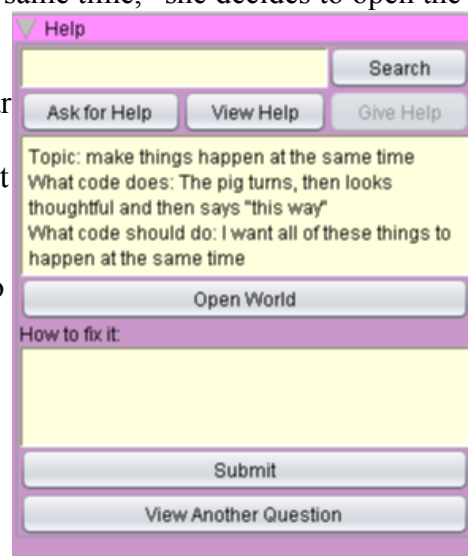


Figure 6: Give help

Beyond the integration of this aspect into the programming environment, users may be more likely to answer questions because they are able to view the code and edit it in order to answer the question. The system encourages the responder to type a textual solution, but if the user does not know how to describe their answer, they can choose to only send back the code. Giving the user both of these options might increase the number of users responding to questions over a system where they can only provide a written answer. This may also increase the effectiveness and clarity of the answer, as many middle school students may not be able to accurately describe programming concepts that they have mastered within the code.

Viewing Answers

Finally, Ann's response to Julie's question will appear in Julie's "view help" section of the help tool. Julie will be reminded of the question she asked and then be able to view Ann's response. She can also open Ann's world and see the code that Ann changed in order to achieve concurrency. If Julie thinks that Ann's response is helpful, she can check the yes box, which will give Ann a reward for her helpful response.

Having these answers directly next to the problematic code should make it easier for users who have asked questions to find their answers. Additionally, if Julie does not want to read the answer, or if Ann's written response is unclear, Julie can see the fixed code, which may be clearer and easier to comprehend. The goal of the attached code is for Julie to be able to understand the solution to her question, or at the least, be able to reuse it so that she can move forward on her project. Having the edited code available should make it much more likely that a user such as Julie can actually use the solution in her project and learn how to accomplish the task.

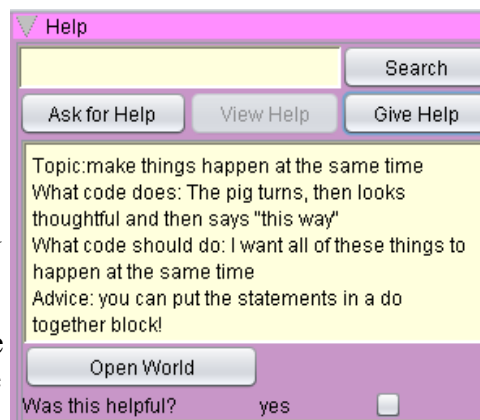


Figure 7: View help

FUTURE WORK: EVALUATION

The evaluation of this work requires the support of Looking Glass and the online community, both of which are still in development phases. User testing of the community help interface is necessary in order to determine whether middle school students understand the features and are able to use them. This would involve giving a user a task in order to determine if they can accomplish it by using the help panel. Each of the features of the help system should be evaluated in this manner. Testing the middle school students' abilities to accomplish these tasks based on the interface would tell which aspects of the designed should be reconsidered and changed in future iterations.

In order to evaluate whether the help system fulfills its overall task, we must consider whether a user decides to ask a question. If a large percentage of users ask questions, then it is successful in overcoming the issue that students often do not ask questions when they have problems. The system also needs to be evaluated on whether users respond to questions and whether those responses are useful to the original asker. These responses are the basis for the help system, as they fill in for the assistance teachers and peers would normally provide. Users' engagement may be based on whether these responses answer their questions and enable them to continue programming and learning.

In general, the help system seeks to keep students engaged in learning programming for extended periods of time in order to increase their programming knowledge. Thus, it is important to evaluate whether the integrated community-based help system successfully accomplishes this task.

ACKNOWLEDGEMENTS

I would like to thank Dr. Caitlin Kelleher for her mentorship this summer. I would also like to thank Paul Gross and Kyle Harms, Ph.D students in the Looking Glass lab, for all of their assistance and wisdom.

REFERENCES

[1] Alevan, V., Stahl, E., Schworm, S., Fischer, F. and Wallace, R. "Help Seeking and Help Design in

Interactive Learning Environments,” *Review of Educational Research*, 2003, 73:277

[2] Bruckman, A., Jenson, C., and DeBonte, A. “Gender and Programming Achievement in a CSCL Environment.” In *Proc. CSCL 2002*. (2002), 119-227.

[3] Scaffidi, C., Shaw, M., and Myers, B. “Estimating the numbers of end users and end user programmers.” In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 207-214, 2005.

[4] Gill, J. “Shedding some new light on old truths: student attitudes to school in terms of year level and gender.” In *Proc. of the American Educational Research Association*. (1994)

[5] Nam, K., Ackerman, M., and Adamic, L. “Questions in, knowledge in? a study of naver’s question answering community” *CHI '09* 779-788

[6] Zimmer, L. and Bennett, S. “Gender Differences on the California Statewide Assessment of Attitudes and Achievement in Science.” *Proceedings of the Annual Meeting of the American Educational Research Association*. (1987).