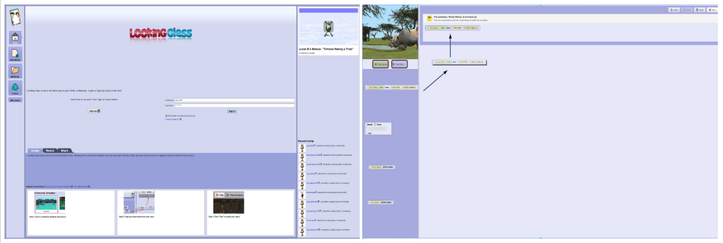
Developing Looking Glass Minigames to Reinforce Programming Subskills

Azeeza Eagal[3], Sophia Lanier[1], Akilah Murphy[2], Jeremy Yu[1,4], Wint Hnin[4], Dr. Caitlin Kelleher[4]

[1] SEAS REU Program, Washington University in St. Louis, St. Louis, MO, USA
[2] Summer Engineering Fellowship, Washington University in St. Louis, St. Louis, MO, USA
[3] NSF Missouri Louis Stokes Alliance for Minority Participation in STEM, Truman State University, Kirksville, MO, USA
[4] Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA

Introduction

Looking Glass is a block-based programming environment that helps introduce programming concepts such as parallel actions, iteration, and object-oriented programming using 3D animations and code puzzles.



- We conducted an observational study using this system in an effort to explore ways of augmenting the puzzle pathways and to gain insight into subskills.
- Subskills are specific skills which are beneficial to users in solving code puzzles.
- We constructed a list of subskills and sought to reinforce select subskills by developing minigames using notes from our study and from working through the pathways ourselves.

Girls Inc. Study

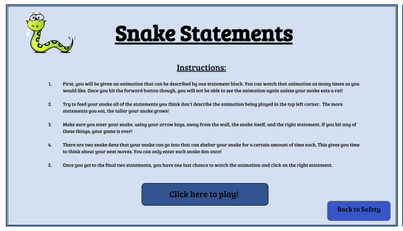
We assisted our lab members in conducting a study consisting of twenty-five high school-aged girls from the Girls Inc. summer camp. Twice a week for five weeks, the students spent an hour playing the pathways version of Looking Glass while we observed and took notes. These notes consisted of a student's codename, the pathway they were working on, the specific puzzle, any problem they might be having, and observations about their behaviors in response to the problem. The goal of this study was to use our observations of the study participants, along with our own observations as we played through Looking Glass, to identify subskills for the code puzzles.

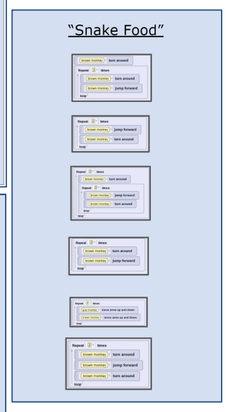
<u>Some examples of subskills are:</u>

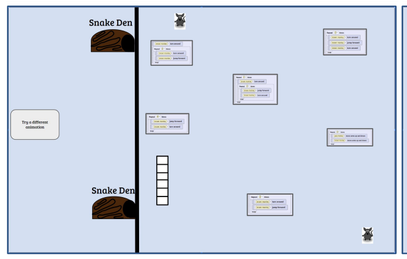
Remembering the order of events in an animation
Matching textual statements to their visual representations
Understanding nested code structures
Identifying the difference between similar code

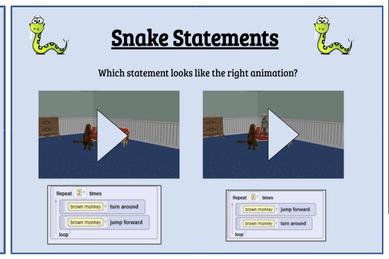
Minigame Prototype: Snake Statements

- Each undergraduate began to develop a minigame targeted to enhance the user's ability in a specified subskill.
- I chose to focus on the subskill matching textual statements to visual representations.
- The Snake Statement minigame was designed to help users break down the animations and to think about each part of the given animation critically.
- For this subskill, I prototyped a version of the famous computer game Snake. In this game, the snake "eats" all of the code blocks that do not correspond to the animation that they had been given. Once the snake has eaten all of the code block but two, the user will have to pick the right statement. If the user picks the right code block, they win!









- We worked on developing the minigame using the Looking Glass IDE. We worked with Java and JavaFX to create these minigames.
- This summer I did not get to integrate the Snake Statements computer prototype into Looking Glass. I would have liked to have three stages of the snake game based on code puzzles in Looking Glass where users have trouble matching textual statement to their visual representations.

Future Work

Going forward with the project, I would like to test our hypothesis -- that our targeted minigames would reinforce the subskills that the users are learning.

To test this hypothesis, I would have two groups: one that used Looking Glass without the minigame and one that did.

I would first have the users do four code puzzles to gauge their initial ability with the code puzzles and the specified subskill.

Then, I would gather data on both groups.

Once the study was over, I would compare the two groups as a whole based on how they approached a code puzzle: how many moves did it take them to complete the code puzzle, which code blocks did they move, and how did the code constructs affect their understanding of the animation.

If the group that had Snake Statements implemented into Looking Glass did better with the subskill and puzzles, then the game was successful.

If they did worse or the same, then the game was not successful in developing the user's ability in the specified subskill.

It would also be interesting to pair users in the two groups based on their initial ability and explore how individually the minigame group altered their approach to the code puzzles.

References

Caitlin Kelleher and Wint Hnin. 2018. Predicting Cognitive Load Based on Learner's History in Solving Code Puzzles. In *Proceedings of the Fourteenth Annual International Conference on International Computing Education Research (ICER '18)*.

David J. Gilmore, "Models of Debugging," Acta Psychologica, ISSN: 0001-6918, Vol: 78, Issue: 1, Page: 151-172

"Looking Glass Community." [Online]. Available: https://lookingglass.wustl.edu/. [24-July-2018].

M. Ichinco, A. Zemach and C. Kelleher, "Towards generalizing expert programmers' suggestions for novice programmers," *2013 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)(VLHCC)*, San Jose, CA, USA, 2013, pp. 143-150.

Acknowledgements

We would like to thank Dr. Kelleher for her guidance this summer, as well as Dr. Carter and Wint Hnin. We would also like to thank the NSF, Washington University, and Missouri Louis Stokes Alliance for Minority Participation Award 1619639 for their support.