Programming in the Pond: A Tabletop Computer Programming Exhibit

Michael S. Horn
Computer Science and
Learning Sciences
Northwestern University
2120 Campus Dr.
Evanston, IL 60208 USA
michael-horn@northwestern.edu

David Weintrop
Learning Sciences
Northwestern University
2120 Campus Dr.
Evanston, IL 60208 USA
dweintrop@u.northwestern.edu

Emily Routman
Computer History Museum
1401 N. Shoreline Blvd.
Mountain View, CA 94043 USA
Current Address:
Emily Routman Associates
182 Hillcrest Rd.
San Carlos, CA 94070 USA
eoroutman@sbcglobal.net

CHI 2014, April 26 - May 1, 2014, Toronto, ON, Canada.
ACM 978-1-4503-2474-8/14/04.
http://dx.doi.org/10.1145/2559206.2581237

Abstract
We present the design of an interactive tabletop exhibit intended to engage visitors in free-form computer programming activities at the Computer History Museum in Mountain View, California. We describe our design goals and outline challenges associated with creating this interactive experience for a free-choice learning environment. We review results of testing sessions with users from our target audience across three successive prototypes.

Author Keywords
Computer programming; Computer science education; Museums; Exhibit design; Multi-touch tabletops; Computer supported collaborative learning; Graphical programming languages

ACM Classification Keywords
D.1.7 [Visual Programming]. H.5.2. [Information Interface And Presentation]: User Interfaces – Interaction styles; K.3.2 [Computer and Information Science Education]: Computer science education.

Introduction
Informal learning institutions such as museums and science centers have a core mission to inspire interest in learning by creating unique and memorable experiences. With the increasing role of digital

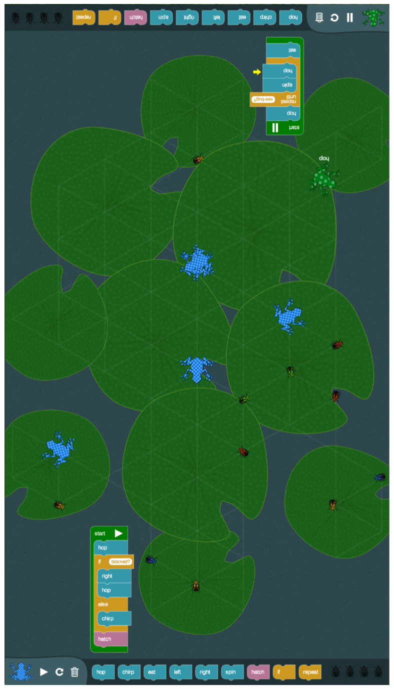
Figure 1. The Frog Pond exhibit.

Figure 2. Two children interacting with Frog Pond.

technology in everyday life, it is not surprising that many exhibits and even entire museums are now dedicated to exploring the role of computers and technology in society. The Computer History Museum (computerhistory.org) in Mountain View, California is one such institution. Working with the Museum, we are designing an interactive tabletop environment called *Frog Pond* (Figure 1, 2) that will be part of a larger exhibit called *Make Software, Change the World.* The goal of Frog Pond is to engage visitors in free-form computer programming activities. In this paper we present our design goals and discuss challenges inherent in creating interactive experiences for free-choice learning environments, especially around potentially confusing topics like computer programming. To date we have developed and tested three iterative prototypes with users. We present results from these formative testing sessions and show how they informed our design.

Design Goals and Challenges

Our design process is guided by several high-level goals. The first is to provide an engaging and fun experience with computer programming for visitors from a variety of backgrounds and experience levels. Central to this goal is that visitors should be able to create simple programs that nonetheless result in interesting and complex outcomes. To accomplish this goal, we were inspired by the NetLogo programming language [8]. NetLogo is an agent-based modeling environment in which simple programs describing the behaviors of computational agents can simulate a wide variety complex and emergent phenomena [9]. From NetLogo we adopt the conventions that programs describe the actions of simple computational agents (turtles in the case of NetLogo); that there can be many such computational agents on the screen at any time; and that the interactions among agents can result in complex patterns.

Second, the exhibit should facilitate *immediate apprehendablity* [1], meaning that it should be intuitive and easy for people to learn how to use even if they have never programmed before. Here we build on our prior experiences creating a robotics and programming exhibit called *Robot Park* for the Museum of Science, Boston [4]. While this exhibit was easy to learn, visitor programs could only result in simple, uncomplicated behaviors (*low-floor, low ceiling*). With Frog Pond we hope to improve this aspect of the experience.

Third, the exhibit should support *active prolonged engagement* (APE) [5]. This implies a number of important sub-goals. It should be *socially scalable* [7], meaning that it should accommodate simultaneous interaction from as many people as can gather around the table and that the experience should become richer as a result. It should also allow for meaningful experiences through different social configurations such as cooperative (side-by-side programming), competitive (head-to-head programming), and parallel but independent engagement. And, even though the

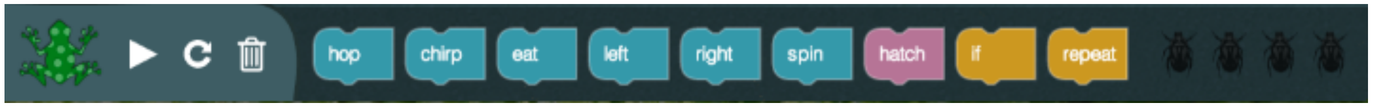

Figure 3. Frog Pond's programming panel. This panel controls the green, polka dot frogs.

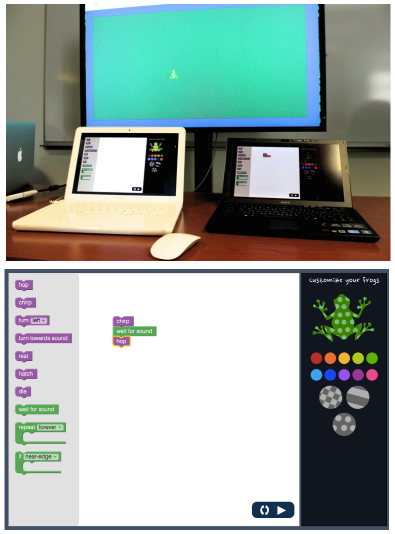

Figure 4. Iteration 1: the laptop setup and programming interface

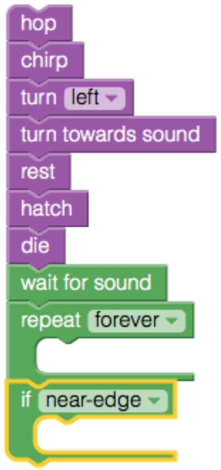

Figure 5. A common program.

exhibit does not enforce a specific activity or sequence of activities, it should nevertheless provide cues that subtly guide users towards interesting experiences.

A final design goal for our exhibit is that it should be consistent with the overall mission of the Computer History Museum and the goals of the *Make Software, Change the World* exhibit. This means that it should be clear to visitors using Frog Pond that what they are doing is computer programming, and the experience should encourage exploration of programming in other contexts. The target audience is the general public (with no technical expertise), ages 11 to adult. The interactives are intended to be usable by younger visitors as well with adult facilitation. This is a departure from the Museum's current exhibitions, which are aimed at visitors of high school age and above.

Frog Pond

Frog Pond presents visitors with a large pond filled with lily pads and brightly colored frogs. Insects buzz around the pond and eventually settle on the lily pads. A faint grid shows paths along which frogs can move without falling in the water. Visitors construct programs using a graphical language to specify the behaviors of their frogs (Figure 9, 10, 11). The interface design is inspired by programming environments such as Scratch [6] and Alice [2]. Frog Pond has two programming panels (Figure 3), one at either end of the table, each of which controls one color of frog. The language includes movement commands (such as hop and spin) as well as conventional control structures (conditional statements and loops). Visitors can also hatch new frogs, making it possible for a program consisting of only a few blocks to fill the table with hopping and chirping frogs. We developed the exhibit using the Dart language running in a web browser on an Ideum Platform 46 tabletop.

Designing Frog Pond

Over the past year, we developed three major iterations of Frog Pond. Each version was tested with users from our target audience.

Iteration 1

Our first prototype (Figure 4) used a graphical programming environment called Blockly [3]. It separated the programming interface (displayed on laptops) from the pond itself (displayed on a larger monitor). This iteration included two challenges. One was to keep the frogs from falling into the water at the edges of the screen. The second was to have frogs communicate with one another by chirping, thus allowing for games like follow the leader. We also included the ability for users to customize the appearance of their frogs. Our initial round of user testing involved 12 children (9 boys, 3 girls) ages 8-14 as part of Take your Daughters and Sons to Work Day.

While the participants responded positively, there were a number of design issues identified. First, there was confusion about the goals of the environment. In post-interviews, children suggested adding challenges, such as the ability to eat the flies or to make an obstacle course. Second, users had difficulty understanding the link between their programs and the movement of the frogs. This round of testing also revealed a number of limitations of the programming language to support short-term interaction with inexperienced users. For example, a common first program was to include every primitive once in a list (Figure 5). Second, despite the interlocking shape of the blocks, it was not immediately

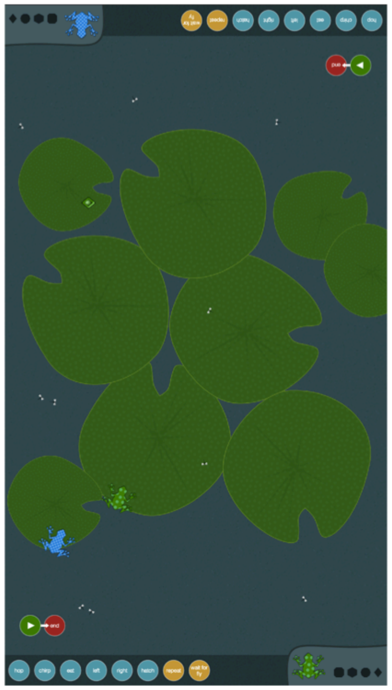

Figure 6. Iteration 2.

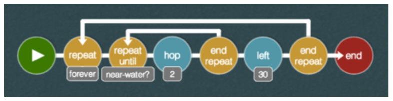

Figure 7. A sample program from Iteration 2.

apparent to the children how to construct programs. Some children placed the blocks left-to-right, like composing a sentence.

Iteration 2

Our second iteration of Frog Pond (Figure 6) included a number of improvements. In this version, programs were constructed in the pond itself, as opposed to on a separate screen. Composing a program involved dragging commands from a programming panel into a chain of commands bookended by start and stop commands. We also displayed the name of the commands in text below the frogs as they moved around the screen (Figure 11). These two changes were designed to help users link their composed programs with the observed frog behaviors. We replaced the Blockly language with a new, left-to-right arrangement (Figure 7), leveraging the sentence building we observed in our first series of tests. To suggest possible activities we included four brightly colored *gems* in the pond. If a frog landed on a gem, it was added to the user's task bar (Figure 8). We included more water on the screen with lily pad shapes to make navigation more difficult. Finally, we decided to remove the ability to customize frogs' appearances as it distracted from the central programming activity.

Our second round of user testing took place during a single day at a museum in Chicago. Approximately 30 visitors interacted with the exhibit while we observed from a distance. About half of these visitors were in the age range of 9-15 years old; the rest were younger children and adults. Two major design issues emerged. The first stemmed from a combination of the touch interface and the new format of the programming blocks. Many visitors seemed to think that the round blocks were buttons that should respond to simple taps. Second, visitors seemed to expect that you could directly interact with the frogs and the gems by touching them. Several visitor groups gave up on the exhibit before discovering the programming interface because they assumed the table was not responsive. This is especially problematic given the short interaction time typical of museum exhibits. There were also some design successes with the second iteration of Frog Pond. The introduction of the gems achieved its intended goal as a few of the visitor groups engaged in races to collect gems. In one case, there were six people gathered around the table in a very intense race to collect all four gems before the other team. It was clear that the race was motivating, but the competitive, nature of the activity meant that visitors weren't really reflecting much on their programs. One group simply hit the play button over and over again, while the other group had a loop that kept hatching more and more frogs that eventually covered much of the screen.

Iteration 3

The third iteration of Frog Pond included further revisions to the interface based both on our observations as well as requests from other stakeholders in the project. This included another redesign of the programming interface, reintroducing some of the characteristics of Iteration 1 while retaining successful features of Iteration 2. First, we reintroduced a Blocky-like design with interlocking blocks that stack vertically, with some modifications for ease of use. This design had the advantages of being consistent with related experiences being planned for the Museum and a stronger relationship to text-based programming. We also added several features to improve the learnability of the programming language for novice users. First,

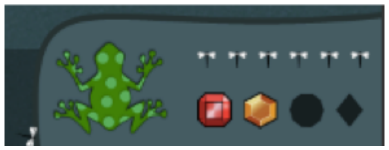

Figure 8. The fly and gem collecting portion of the programming panel.

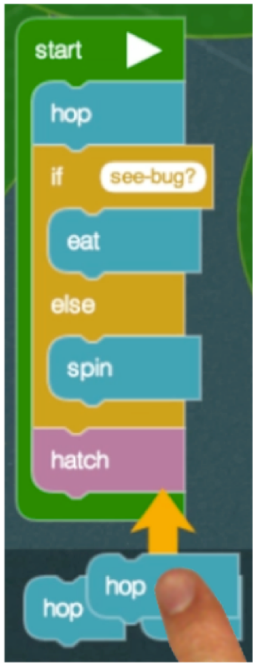

Figure 9. The visual feedback when a user puts her finger on a block

when a user touches a block in the panel, an arrow appears to indicate that the block should be dragged onto the screen (Figure 9). The program bracket also rearranges itself to make room for the new block showing where it could be placed (Figure 10). If the user only taps on a block as if it were a button, it automatically flies into the existing program.

We also introduced a hexagonal grid over the lily pads to show exactly where the frogs can move, providing an easier way to gauge distances and to encourage more thoughtful programming. In an effort to be more consistent with our pond theme, we replaced the gem-collecting task with the challenge of eating brightly colored beetles that settle onto intersections on the grid. Other minor tweaks were made to address earlier problems. For example, to help visitors link the commands with the frog behaviors we added an arrow tracing the flow of control of the program as it runs, (Figure 11). We tested the latest version of Frog Pond with 28 children ages 11 to 14 as part of a weekend activity at the Computer History Museum.

We divided the kids into groups of three or four and gave each group roughly ten minutes to interact with Frog Pond. No instructions were provided. At the conclusion of their time, each group was asked a series of follow-up questions. All of the groups were engaged for the full 10 minutes without losing interest. Within the first minute of play, every group was able to create and run a program with more than one command. It took most groups at least a minute to understand the goal was to catch bugs and to figure out how to do so, but all groups were able to catch multiple bugs within the 10-minute period. The least successful group worked for almost five minutes before catching their first bug. Players raised a few questions over the course of play, including “Are there any more commands?” and “How can you get rid of the extra frogs?” Both of these questions are encouraging as they suggest that the children were engaged with the programming activity. In follow-up interviews, all groups enthusiastically responded that the game was fun. Some players did offer qualifiers, saying it was fun but confusing, or that they wished the game was a little easier. Two students commented that the environment was educational because you learned how to put together commands and that it really showed how programming worked. Two suggestions were made by at least one member of every group: provide instructions on how to play, and add a scoreboard, awarding points for eating bugs. Several students suggested having a time limit because otherwise “no one will want to stop.” Following this advice, we are currently in the process of designing and implementing a help system and scoring function, and will evaluate the need for a time limitation.

We conducted an additional round of testing with general visitors at the Computer History Museum. Approximately 20 groups of 1 to 4 visitors used Frog Pond during 1.5 hour period of observation. As visitors began to interact with the program we offered the option of a tutorial (as a means to simulate a *help* feature that we are in the process of designing). Visitors who requested assistance were provided with three instructions: the goal of the program, how to add or remove commands, and how to reset their frog to the starting position. Visitors who used Frog Pond included: adults alone or in pairs, children alone or in pairs, and one to two adults with one or two children. Duration of play ranged from 20 seconds to 12 minutes. One child, approximately 10 years old,

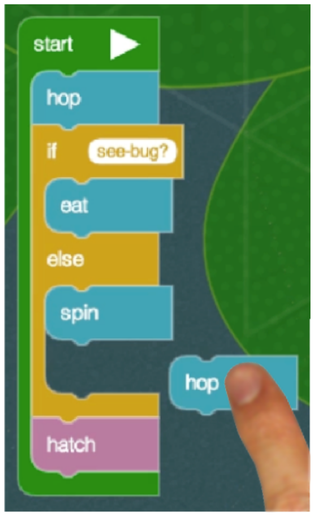

Figure 10. Visual cues to help visitors construct programs.

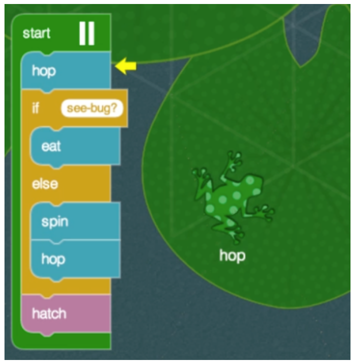

Figure 11. A frog programming running.

returned four times for a total of 9 minutes of game play, and one adult played for 12 minutes before being asked to free up the table for others.

Verbal instructions resulted in players being able to immediately begin programming and to target their efforts toward the goal of catching bugs. Visitors who stayed longer created increasingly complicated programs, showing evidence of growing skills and confidence. Two children estimated to be 7 and 8 years of age respectively tried the program without adult assistance. They created programs successfully but were unable to catch bugs. All other visitors created successful programs.

Conclusion and Future Work

Our overarching goal for Frog Pond is to create an engaging, interactive programming exhibit for museum visitors. Through iterations of design and testing we are gaining confidence that we have met our stated goals. We have seen young children write programs with only a few commands that fill the screen with hopping, chirping, and spinning frogs. Frog Pond appears to be immediately apprehendable as visitors were quickly able to engage with the activity and create programs. Visitors who did not immediately understand the goal of the exhibit only required three brief verbal instructions as guidance. We also saw groups play with Frog Pond for 10 full minutes, which is well beyond the average exhibit visit [5]. In our most recent round of observations, we saw Frog Pond support a variety of visitor arrangements, suggesting the exhibit is socially scalable. Finally, in post interviews, students said they found the exhibit both fun and informative. While these findings are encouraging, they remain preliminary, a fact we will address in future work. The next step is to finalize the design of Frog Pond. This includes adding help and scoring features as well as a "show code" feature to further bridge Frog Pond to other text-based programming visitors will encounter throughout the exhibit. We will collect additional naturalistic user observations on-site at the Museum. In the summer of 2014, we will conduct a formal study to document interaction patterns and evaluate learning outcomes. Frog Pond will open to the public in the winter of 2015.

References

[1] Allen, S. Designs for learning: Studying science museum exhibits that do more than entertain, *Sci. Educ.*, 88(S1), S17–S33, 2004.
[2] Conway, M., Audia, S., Burnette, T., et al., "Alice: lessons learned from building a 3D system for novices," in *ProcCHI '00*, 2000, pp. 486–493.
[3] Fraser, N. *Blockly*. https://code.google.com/p/blockly: Google, 2013.
[4] Horn, M.S., Solovey, E.T., Crouser, R.J., and Jacob, R.J. Comparing the use of tangible and graphical programming languages for informal science education. In *CHI '09*, 975–984.
[5] Humphrey, T. and Gutwill, J. *Fostering Active Prolonged Engagement*. San Francisco, CA: The Exploratorium, 2005.
[6] Resnick, M., Silverman, B., Kafai, Y., et al., Scratch: Programming for all, *Commun. ACM*, 52, 11, 2009.
[7] Snibbe, S.S. and Raffle, H.S. Social immersive media: pursuing best practices for multi-user interactive camera/projector exhibits. In *CHI'09*, 2009, 1447–1456.
[8] Wilensky, U. *NetLogo*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/netlogo, 1999.
[9] Wilensky, U. and Resnick, M. Thinking in levels: A dynamic systems approach to making sense of the world. *Science Education and Technology, 8*(1), 1999.