Frogs to Think with—Improving Students' Computational Thinking and Understanding of Evolution in A Code-First Learning Environment

Yu Guo[1], Aditi Wagh[2], Corey Brady[1], Sharona T. Levy[3], Michael S. Horn[1], and Uri Wilensky[1]

[1]Computer Science and Learning Sciences
Northwestern University
Evanston, IL USA
yuguo2012@u.northwestern.edu

[2]Department of Education
Tufts University
Medford, MA USA
aditi.wagh@tufts.edu

[3]Faculty of Education
University of Haifa
Haifa, Israel
stlevy@edu.haifa.ac.il

ABSTRACT
This paper presents Frog Pond, an interactive *code-first* learning environment about biological evolution. We deployed Frog Pond as part of a six-day curricular unit on natural selection implemented in six 7th grade science classes. Here we describe a case study of two students, Charlie and Aaron who participated in the unit. Comparing pre- and post- interviews in which they were asked to design a program for a hypothetical simulation of evolution, we found that both students shifted from an event-based programming approach to a rule-based approach. Both students also drew upon their experience with Frog Pond to explain an evolutionary phenomenon. However, the level of sophistication of the two students' explanations varied along with the aspects of Frog Pond they drew upon. These findings have implications for design improvement to better support students' understanding of evolution.

AUTHOR KEYWORDS
code-first learning environment; computational thinking; evolution; agent-based modeling

ACM CLASSIFICATION KEYWORDS
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous

INTRODUCTION
Evolutionary change and its underlying mechanisms are core disciplinary ideas in the life sciences [3, 11]. Understanding mechanisms of biological evolution has been shown to be notoriously difficult for students [10, 18]. This is in part because changes that emerge in a population of organisms over time unfold without a plan and without the knowledge or intentionality of individuals in that population. This can be counter-intuitive to learners inclined to think in terms of centralized and deterministic processes [16, 17]. In other words, it is tempting to think of individual organisms as *wanting* to change (e.g., cheetahs want to become faster) and to think of evolution as following a master plan in which organisms become progressively come to approximate a "more evolved" goal state over time. This is in stark contrast to a world in which random events (e.g., genetic mutations), occurring without a master plan, generate trait variations that result in differential survival and that, over time, produce remarkable species-level adaptations.

Computational modeling environments are increasingly used to facilitate learning about complex scientific phenomena such as evolutionary change [12, 13, 18]. Enabling students to program such models has shown promise for improving understanding of both the scientific phenomena and of programming [19, 20]. However, in many real-world classrooms, there is not enough time for students to delve into programming activities at a sufficient level of sophistication. Thus, students often explore pre-built models by adjusting parameters and observing outcomes in simulations' behavior, without an opportunity to crack open the black box.

In this paper, we investigate how student programming in a code-first modeling environment [5] supports learning about evolutionary processes. Our data was collected in the context of a six-day curricular unit implemented in 7th grade science classrooms. We draw on pre- and post-interviews in which students were asked to design a program for a hypothetical simulation of evolution.

BACKGROUND

Object-to-think-with
Whether something is easy or difficult to learn depends, in part, on the ability of learners to assimilate new knowledge into a collection of existing mental models [9]. Papert referred to seminal models, both external and internal, that learners draw upon when encountering new situations as "object[s]-to-think-with" [9, pg. 11].

Much work has been done on using computational models to facilitate learning about evolutionary change (e.g. [13, 20]). Evolutionary changes can be seen as a type of emergent phenomena, in which the overall trends at the population level emerge from interactions among individual organisms. One of the sources of students' difficulties with such phenomena lies in a "level slippage"—the tendency to attribute population level changes to individual level behavior or vice versa [17]. For instance, students might attribute a population's changing over time to its constituent *individuals* changing over their lifetime. In fact, several interactions and mechanisms at the micro level are at play, including sexual reproduction, inheritance, death, and random variation in inherited traits as well as the changing environment.

Across both in research and education contexts, emergent phenomena are increasingly simulated using computational *agent-based models* (ABMs). In such models, individual computational agents have properties and can enact behavioral rules as the simulation runs. For example, the NetLogo modeling environment [14] is widely used in both schools and research labs. By emphasizing the behavior of individual actors in a system, ABMs can help students draw on their own bodily and sensory experiences in the world [16-19].

Due to the many constraints faced by classroom teachers in schools, work with agent-based modeling in schools often involves students manipulating pre-built simulations of evolutionary processes without a chance to tinker with the underlying computer code. However, research has found that having students build models facilitates learning about deeper mechanisms of change [12, 19]. Prior work has found that programming toolkits for evolutionary change such as EvoBuild [12] can facilitate more rule-based explanations both *in-situ* and in post-tests. As such, we designed a blocks-based programming environment called NetTango [8] that provides easy entry to NetLogo modeling.

Frog Pond expands on work with EvoBuild by providing a code-first environment to foreground sense making of evolutionary change as a computational phenomenon. In addition, the flexibility of the environment enables students to rapidly and effortlessly move back and forth between the code and enacted behaviors, across multiple time scales, and across varying environmental pressures.

Two rounds of Frog Pond studies

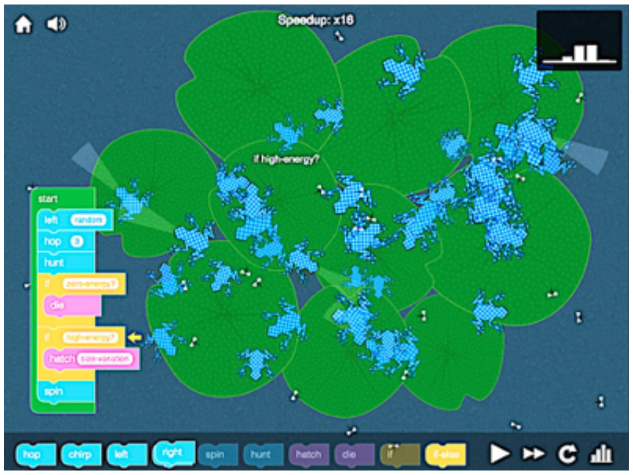
Prior work with Frog Pond in a museum showed that it provided a low threshold for users to create simulations by assembling code blocks. Although interactions typically lasted for only 15-20 minutes, users were able to understand that they could program the behaviors of simulated frogs in a lily pond using blocks such as "hop", "hunt", and "hatch". Most middle-school aged users were also able to reason with their encoded blocks about at least one target evolutionary trend, such as the advantages a small-sized frog might have over a larger-sized frog in a particular scenario [5]. Next, to promote more extended modeling sequences than were possible in a museum setting, we developed a unit for classroom implementations based on our findings about interaction design from these museum studies. We deployed this Frog Pond curriculum as a six-day unit on natural selection and adaptation in middle school science classes.

The goal of this paper is to examine whether, and how, a code-first environment supported learning about evolutionary mechanisms. Specifically, we investigate two issues: First, does programming simulations of adaptation in a code-first environment impact how students represent a different evolutionary scenario in the form of code? Second, does programming in a code-first environment impact how students account for and explain a different evolutionary scenario?

To do this, we present a case study using pre- and post-interviews of two students, Charlie and Aaron (pseudonyms), who worked as a pair throughout the unit. We examine their responses to a question about co-evolutionary trends in the speeds of cheetahs and gazelles. We analyze their explanations for this co-evolutionary trend as well as the computer "pseudocode" they constructed on paper to represent the scenario in the form of a simulation. We found changes between their pre- and post-interview responses. In representing the scenario in the form of code, both students shifted from an event-based programming approach to a rule-based approach. In addition, both students drew upon their experience with Frog Pond to explain the evolutionary scenario, albeit in different ways. However, depending on which elements of Frog Pond they drew upon, the level of sophistication of their explanations varied. We describe these shifts in the students' approach and how they used Frog Pond as an object-to-think-with. These findings have implications for design improvement to better support the design of code-first environments as well as students' understanding of evolution.

THE DESIGN OF FROG POND

We designed Frog Pond as a code-first learning environment [5] (Figure 1).



```
start
  left random
  hop 3
  hunt
  if zero-energy?
    die
  if high-energy?
    hatch size-variation
  spin
```

Figure 1. Frog Pond simulation.

A code-first environment has three important features. First, the primary mode of interaction is through programming. For example, in Frog Pond, nothing interesting will happen until students program their frogs to do something like hop forward, eat flies, change direction, or reproduce. Second, it should extremely easy for novice users to create working programs within the first few minutes or even seconds of playing around with the system. This doesn't mean that the programming language has to be visual, but blocks-based languages such as Scratch [7], Blockly [4], DeltaTick [20, 21] and StarLogo TNG [6] are based on principles of direct manipulation, which can make them less intimidating to new users and facilitate a faster learning curve. Finally, code-first environments should be expressive. Even though learner programs are relatively simple, they should nonetheless result in diverse and complex outcomes.

In the Frog Pond environment, learners program instructions for a group of frogs in an ecosystem using domain-specific, blocks-based primitives. On running the program, each frog repeatedly enacts the encoded instructions to interact with other frogs and the simulated environment. These enactments can produce many different outcomes that highlight concepts of stabilizing, directional, and disruptive selection pressures. In other words, the simulation can result in changes in the frog population: 1) growing bigger or smaller (directional pressure), 2) staying around the same size (stabilizing pressure), or 3) separating into two distinct sub-populations, consisting of larger and smaller individuals (disruptive pressure).

To create Frog Pond, we designed a blocks-based programming environment called NetTango [8] that provides easy entry to NetLogo [14] modeling. We added several features to NetTango that optimize it for use on touchscreen devices and that make it extremely easy to construct and run working programs even in their early exploration of the environment. The goal was to simplify the language as much as possible to get students up to speed quickly without sacrificing too much of the language's expressivity of NetLogo and ABMs.

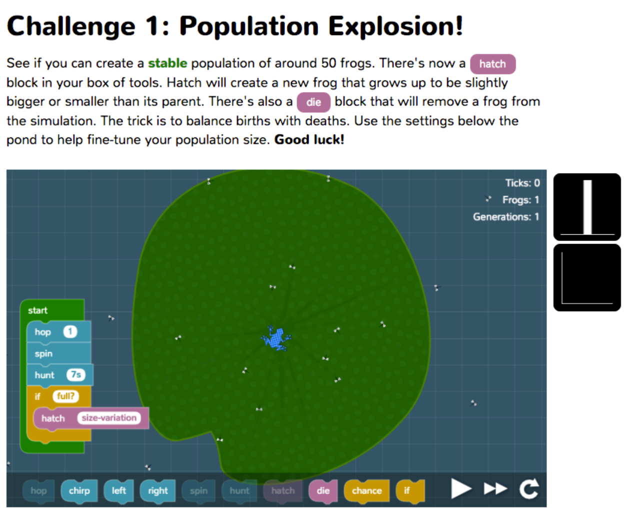
We invested substantial effort making the link between the programming blocks and the effects on individual frog behaviors as clear as possible. This included overlaying the programming blocks directly on top of the simulation window so that both are visible at the same time. A new round of simulation starts with a very small number of frogs that slowly act out the student-authored programs in a step-by-step fashion. At each step, the corresponding code blocks are highlighted to provide visual cues for students to know which command has just been executed. Students can speed up the simulation to 16 times faster than the normal speed to observe changes across multiple generations. Even when the screen fills up with potentially hundreds of frogs, it is still possible for students to zoom in and track the movements of an individual frog by clicking on it. When a frog is highlighted, students can see its energy level, its size, and the command it is currently executing. Finally, we added a shareboard feature that allows students to easily share their programs and the current state of the simulation environment with other students in the class. On the interface, we also display population-level graphs that are dynamically linked to the simulation state. These include a histogram of frog sizes and a plot of frog count to show two important macro level measures: the distribution of different sized frogs in the population and the variation in population size over time.

The goal of this paper is to describe how interacting with a code-first environment like Frog Pond can support students' learning about evolutionary mechanisms. Below we describe our Frog Pond curricular unit and focus on how two students explained and accounted for an evolutionary phenomenon after interacting with Frog Pond. We describe the study and data analysis and present findings related to learning outcomes.

THE FROG POND CURRICULAR UNIT

Our curricular unit design is driven by an overarching question: Why are there so many different kinds of living things on earth? To answer this question, we ask students to explore the mechanisms of natural selection by programming virtual frogs in a simulated ecosystem—a frog pond. Students can drag and drop ten different types of programming blocks to control the frogs' behavior. When dropped on a certain area of the screen, the blocks snap together to form a program. There are eight behavioral blocks ("hop", "chirp", "left", "right", "spin", "hunt", "hatch" and "die"), and two logic blocks ("if" and "if-else"). A simple program a student may create is "left (random), hop (1), hatch (size-variation), chirp". When the simulation starts, this simple program will be executed line by line and repeatedly. As a result, the single initial frog at the center of the lily pads turns left with random degrees, hops forward 1 body-length unit, hatches another frog, which may be a little bigger or smaller than itself, and then chirps. Then, the two frogs on the screen execute the program over again, move about and hatch their offspring, so on and so forth. The variations in frog size have multiple implications: big frogs hop farther than little frogs, which means they are more likely to fall into the water (and die). Big frogs also have longer tongues than little frogs, which means they are more likely to catch flies when they hunt. However, little frogs use less energy than frogs, which means they do not need to eat as much to survive. When a simulation is running, the environment (for example, the arrangement of lily pads, the number of flies over the pond, and the energy each fly can provide) interacts with frogs' behaviors on one hand and traits on the other, leading to advantages or disadvantages for different sized frogs.

In the unit, students explore natural selection by engaging with five increasingly sophisticated challenges to guide students to explore natural selection.



Challenge 1: Population Explosion!

See if you can create a **stable** population of around 50 frogs. There's now a hatch block in your box of tools. Hatch will create a new frog that grows up to be slightly bigger or smaller than its parent. There's also a die block that will remove a frog from the simulation. The trick is to balance births with deaths. Use the settings below the pond to help fine-tune your population size. **Good luck!**

Ticks: 0
Frogs: 1
Generations: 1

Figure 2. Challenge 1 in the Frog Pond curricular unit.

In Challenge 1 (Figure 2, above), students are asked to create a stable population of around 50 frogs. This challenge foregrounds the concepts of reproduction and death, guiding students to an understanding that a stable population is in fact the result of multiple dynamic processes across generations: a dynamic equilibrium as opposed to a static state. In Challenge 2, students are asked to create a stable population consisting almost entirely of little frogs. The goal of this challenge is to experiment with directional selection pressure—one that drives organisms' traits in one direction over successive generations. In Challenge 3, students are asked to create a population that has almost all big frogs. The target concepts in this challenge are similar to those in Challenge 2, though it is more difficult to program behaviors that favor larger frogs because there are multiple competing selection factors that must be balanced. In Challenge 4, students are asked to create a stable population consisting of medium-sized frogs. To do this, students have to take into account selection pressures from both directions and program their frogs accordingly. In Challenge 5—the hardest challenge—students are asked to simulate disruptive selection pressure, in which it is advantageous to be either really small or really big, but not to be medium-sized. This challenge helps students understand diversification and eventually speciation—how animals with a common ancestor can become vastly different over time. For this challenge, beetles are added into the ecosystem as a new food source, which is extremely nutritious but hard for little frogs to catch. We also introduced a few very small lily pads. Students can remove one small lily pad that connects to the bigger pads to create two segregated environments, allowing them to see that isolation can expedite the diversification of frogs' traits.

RESEARCH QUESTIONS

With the above design and deployment, we asked two research questions:

1. In what ways, if any, does programming in a code-first environment support learning about evolutionary mechanisms?

2. Does learning about evolutionary shifts by programming in a code-first environment impact how students reason about evolutionary scenarios outside the code-first environment?

METHOD

Implementation

The Frog Pond environment was used to create the five challenges described above, each of which involved programming a simulation. Each challenge formed an activity. The five activities were implemented in a middle school in an ethnically diverse suburb of a large Midwestern city. One science teacher led the Frog Pond activities in six 7th grade science classes over a period of 6 days. Students worked in pairs in each class. Students were encouraged to talk to their partners, participate in whole class discussions, and share successful code in a gallery, which the teacher and other students could see.

Participants

A total of 124 students from six classrooms took part in the study with informed consent. Those who did not consent to participate in the study still used Frog Pond in their regular class activities, but the research team did not collect any data from these students. Of these, 71 students took a written pre- and post-assessment on topics of evolution and computational thinking. This included 41 girls and 30 boys between 12 and 13 years old. The students were asked to self-report their race / ethnicity at the end of the pre-assessment and were allowed to select multiple categories. The students reported that they were 38% White, 29.6% Asian American / Pacific Islander, 28.2% "Other", and 8.5% Latino. One student selected African American and one student Native American. Close to 60% (42 students) reported speaking language(s) other than English at home. The most common languages were Urdu (8 students), Spanish (6), Vietnamese (4), Romanian (3), and Arabic (3). Other languages included Albanian, Assyrian, Bengali, Greek, German, Gujarati, Hindi, Korean, Malayalam, Malaysian, Mandarin, Montenegrin, and Serbian. We also asked students if they had ever coded a computer program before. Only 18 students (25.4%) indicated yes in response to this question.

During the unit, students worked in pairs on the activities. In consultation with the teacher, we selected 12 pairs as focal groups to represent a range of interest and prior academic performance in science class. We interviewed these focal group students before and after the unit to elicit their understandings of evolutionary mechanisms. We also video recorded their interactions with Frog Pond software during class. In this paper, we present an analysis of pre- and post- interviews of one focal pair, Charlie and Aaron. We selected these two students for analysis for three reasons: 1) Data integrity: Charlie and Aaron participated in all 5 Frog Pond activities as a pair, which was not always the case with our other focal dyads. 2) Contrasting cases: Although they worked together, their interview responses indicated stark differences in the sophistication of their reasoning. 3) Representativeness. Taken together, their responses were representative of other focal students.

Written Pre- and Post-Assessments

In addition to our focal interviews, 71 participants took a written assessment before and after the unit that included questions on evolution and computational thinking. The two assessments were identical except for a demographic questionnaire at the end of the pre-assessment. The assessments included nine Likert scale questions on a five-point scale about students' evolution attitudes and beliefs (e.g. *I know a lot about evolution. Evolution explains the origin of insects. Evolution is not happening today*). We constructed a measure, *Evolution Attitude*, as the average of seven of these items (Cronbach's alpha = 0.70). Students' average pre-assessment score was 3.49 (SD=0.71) compared to a post-assessment score of 3.75 (SD=0.76). This difference was significant: $t(70) = -3.97$, $p < 0.001$.

The assessments also included a scenario describing the predator—prey relationship of cheetahs and gazelles as a way to elicit student reasoning around concepts of natural selection and co-evolution. The short scenario was followed by a series of six open-ended questions. To evaluate student responses on these questions, we developed a coding scheme based on prior research on evolution concepts (e.g. inheritance, differential survival, and variation). We blinded students' written assessments so that the researchers could not see age, sex, race/ethnicity, or other background information. The assessments were then randomized so that researchers could not tell whether they were evaluating a pre- or post-assessment. Two researchers independently coded a set of 40 random student assessments. For this set, they achieved a 94.25% agreement rate (Cohen's kappa = 0.68). After establishing inter-rater reliability, the researchers divided up and coded the remaining assessments. Individual codes were assigned only once across all six open-ended prompts, even if a student mentioned the idea more than once. From this, we constructed a composite score. Students scored an average of 2.62 (SD=2.02) on the pre-assessment questions compared to an average of 3.56 (SD=2.53) on the post-assessment questions. This difference was significant $t(70) = -3.00$, $p < 0.002$. The coding suggests that student answers were more elaborate in the post-assessment.

With these results as a backdrop, we turn to the main focus of this paper, which is a qualitative analysis of our clinical interview data.

Interview Question

In both pre- and post- interviews, we gave students a modified version of a scenario that was commonly used in the literature [1]: "Cheetahs are able to run on average about 60 miles per hour when chasing prey. Their main source of food, gazelles, can also run on average about 60 miles per hour. Cheetahs and gazelles typically live for around 10 to 20 years. Many thousands of years ago, the ancestors of both cheetahs and gazelles could run only about 20 miles per hour." Students were then asked to elaborate on "how a scientist would explain how the ability to run fast evolved in cheetahs and the gazelles?"

After students provided initial explanations, we asked them to create a hypothetical computational model of the scenario by writing down instructions that specify the interactions of agents in the scenario. Specifically, the question stated: "Let's say that you are creating a computer model of the cheetahs and gazelles, and how they interact and change over long periods of time. What kinds of instructions would you use in your model?" Students were provided with paper and pen to draw their program, and explain how it would run over time.

ANALYSES AND FINDINGS

We used a bottom-up approach [2] to analyze two complementary sources of data: transcripts of students' pre- and post- interviews, and the code they had produced on paper. We analyzed the code to investigate the extent to which it would account for evolutionary change. This involved examining whether students provided an explanation for differential survival and/or reproduction, whether the code involved an individual changing within its lifetime, and whether they took genetic and trait variations into account. This analysis led to the identification of a salient shift in the nature of students' code that we describe in the Findings section below.

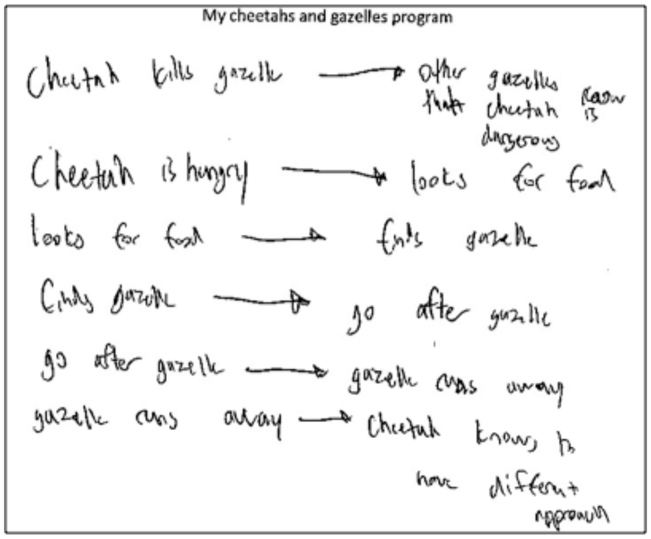
In addition, through iterative rounds of analysis, we also observed that in the course of the post-interview, students repeatedly referred to different aspects of Frog Pond to explain a distinct scenario about cheetah-gazelle co-evolution.. This became another point of analysis, in which we focused on how students drew on their experience with Frog Pond, both explicitly and implicitly, to explain the new scenario.

Finding 1. Shift from event-based to rule-based coding

Analysis of pre- and post-programs revealed that both students shifted from an event-based coding approach to a rule-based approach.

In the pre-interviews, when asked to create a program of the cheetah-gazelle interaction, both students described predation scenarios, in which cheetahs chase gazelles using some hunting strategies. Their descriptions resembled predation scenes that are usually seen in wild life documentaries such as Animal Planet. For example, in the pre-interview, Charlie's approach to a simulation is event-based. It involves events such as gazelles learning that cheetahs are dangerous if they see cheetahs killed other gazelles. It also shows that if cheetahs' attacks are not successful, gazelles run away and cheetahs then know to use a different approach to hunt next time (Figure 3).

Aaron did not write down complete simulation rules in the questionnaire (Figure 4.), but in the interview he explained how the simulation should work. Like Charlie, his explanation was also event-based. For instance, he says, "a hungry cheetah stays on the site and go after a gazelle with a comfortable distance". The cheetah is "being quiet and staying under the grass, try to be less visible". When being chased, the gazelle, "get hide, stand in the grass, try to make it difficult [for the cheetah to catch it]". In this way, both Charlie and Aaron focused on specific events related to predation. However, they did not account for underlying mechanisms that caused changes in speed in both species. Their account did not include rules related to reproduction, death, and chance that result in these changes. Students' program code in the post interview was strikingly different from that in the pre-interview. (See Table 1 and Table 2. Here, to save space, we have typed out their post-programs verbatim instead of providing scans of their hand written code.)


My cheetahs and gazelles program

Chetah kills gazelle → other gazelles know that cheetah is dangerous

Cheetah is hungry → loots for food

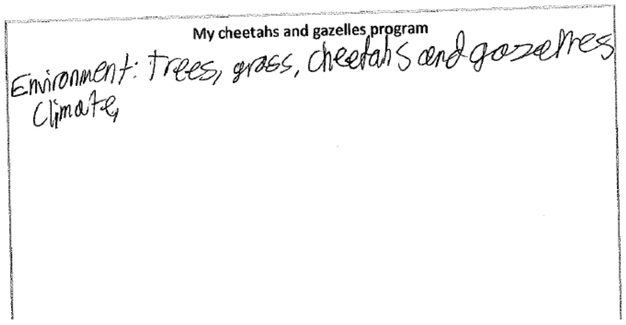
looks for food → fin's gazelle

Finds gazelle → go after gazelle

go after gazelle → gazelle runs away

gazelle runs away → Cheetah knows to have different approach

Figure 3. Charlie's pre-program.


My cheetahs and gazelles program

Environment: trees, grass, cheetahs and gazelles Climate,

Figure 4. Aaron's pre-program.

Cheetah	Gazelle
Cheetah chases gazelle If cheetah is slow, and doesn't get food, it dies [if full] Chance 41% reproduce Offspring speed can vary If cheetah doesn't get food, it finds a different way to attack gazelle If cheetah attacks gazelle, (one dies but) other gazelles learn a different way to run	If gazelles is slow, it dies If gazelle is fast, it survives and .. [if full] Chance 41% reproduce offspring speed vary

Table 1. Charlie's post-program.

Cheetah	Gazelle
Move left 75 degrees Hunt 8 seconds [chance 50% of getting food] If full, hatch If starving, die Right 60 degrees	Right 65 degrees Run away 50% of death Eat grass If full, hatch If starving or hurt, die Left 70 degrees

Table 2. Aaron 's post-program.

First, both students wrote specific rules for individual cheetahs and gazelles. The rules prescribed a set of behaviors that each individual animal would do over and over again throughout the simulation. For example, the rule

```
if full -> hatch [with 41% chance]      (1)
```

means that at each tick of the clock in the model, each animal checks its own internal state. If it is full, it reproduces with a 41% chance. Compared with the event-based approach found in the pre-test, this rule-based approach shows that students focused more on the detailed mechanisms of the animal's behavior than describing specific predation scenes. From a computational modeling perspective, the rules that students came up with in the post-test are plausibly executable by computational agents. These rules are more like lines of code that can actually be found in real scientific modeling programs. Representing evolutionary phenomena with rules is helpful to students in understanding evolutionary processes because it demystifies the changes that take place over long periods of time. Students can see that through each individual's repeated executions of simple rules with some randomness, changes at the population level *emerge* as the result of the interaction between the species and the environment. It is not because of animals' desires to change or due to some magic power that is driving the changes. This first finding may match expectations, given that students had just spent more than a week working with an agent-based modeling environment in which they were developing the code that governed agents actions. This becomes important as we look closer at how this shift also affected students' reasoning about the evolutionary scenarios.

Finding 2. Using Frog Pond as an object-to-think-with

Students' experiences with Frog Pond provided them with a resource to think through and reason about the coevolution of cheetahs and gazelles' speeds, a distinct evolutionary scenario from that experienced with Frog Pond. At first glance, this might not seem particularly surprising. Students had spent 6 class periods using the Frog Pond environment. Hence, one would expect that they would refer to the environment when responding to questions in the post interview. However, the difference in the ways in which the two students referred to code from Frog Pond sheds light on why this is interesting.

At the start of the post interview, neither student seemed to have a readymade explanation for the co-evolutionary scenario. Both students were constructing their explanations on the spot. This was seen in how both students drew upon their different parts of Frog Pond to explain the cheetah-gazelle scenario. Below is an excerpt from Charlie's interview. Charlie's initial explanation was that animals keep running, so they get faster. While elaborating on this explanation, he started to generate another explanation by drawing on his experience with Frog Pond.

Charlie: "Let's say now it's another gazelle or gazelles are together, the cheetah goes at it. All the gazelles will know to run quicker, but because they'll keep getting used to running and running, maybe they'll get faster from that…"

Interviewer: "Okay."

C: "because they'll be able to run, like, right away quicker because what would usually happen was the gazelle's dead, and it's all over, but now the gazelles are running. Now, the cheetah comes, they run, run, run, they're getting faster."

I: "Okay. So, by getting faster, do you mean like they get more practice and so they grow stronger?"

C: "Yeah..."

C: "Or the only, or maybe what happened was, they might get a little faster, but what's more likely to happen is that the slower cheetahs, the slower gazelles, would die off."

C: "And there will only be faster gazelles, and then as time goes by, and the gazelles get a little faster by, you know, chance because when you… ***like with the frogs****. There's a chance of the frog, when they are born, being big or small, but like, let's say the program was fitting for the big frogs to live, then the small ones would die. So, it's not like they're getting any bigger, it's just that the small ones are dying."*

At this point, a new explanation that is qualitatively different from his old idea came to Charlie for the first time. Prompted by his experience with Frog Pond, Charlie explained that it was more likely that slower animals would die off and only the fast ones survive. This explanation became more convincing to him over his old idea of animals getting more practices to become faster. What we would like to highlight here is that Charlie for the first time attributed the change of running speed to the population level, instead of to the individual level. The new explanation includes two important mechanisms of evolution: *differential survival* and *chance of inheritance*, which he had encountered in the Frog Pond environment.

A little later, as Charlie was constructing code, he included several rules that reflected Frog Pond code (Table 1). For instance, one of the rules he included was:

```
offspring speed can vary
```
(2)

This rule was intended to account for variation at birth—an offspring's traits can be different from partners in random ways, which is the source of difference in a population. Not only did Charlie include a micro-level rule that produces variations, but he also described macro level consequences of agents enacting such rules—differential survival. In other words, he connected the levels.

Similarly, Aaron's constructed code also included elements of rules in Frog Pond (Table 2). In addition, he explicitly referred to his experience with Frog Pond when asked to explain this code.

Aaron: "It's a lot slower [in the real world] than it is in the program, and then as soon… like, I'm going to actually add like, a ***chance*** *of it [the cheetah] eating him [the gazelle]. Like, 50 percent because they're both the same speed. It's even. And* ***if full*** *then they* ***hatch****, but* ***if they're starving****, and they haven't eat, like real life, they'll* ***die****. And then I just put like, moving* ***right****, because they don't normally move left. And then for gazelles…*

I: "So, why did you need to write there if they were moving left? Why was that… a problem?"

*A: "****In the one that we did in the class****, if they don't move, they couldn't get the flies. But you had to move them to get closer to the flies and beetles to eat them. And in real life, cheetahs have to… they have to move left and right to find gazelles and hunt, and that's how they… that's why they have to move.*

A little later, when asked why he included instructions to move "left" and "right", he answered:

A: "Because if they only go left, like, ***I'll use the frog again, the model****... like, if the frogs only went left, they'd eventually fall off the pad. But if they went left and then like,* ***spun****... or, if they went left and then went right they might avoid the water and avoid death and live longer."*

I: "So, how are you thinking it's important here, with the cheetahs and the gazelles?"

A: "Since they don't have water around them, like the water in the model, the gazelle, if it goes only... if goes only left, it could end up... it could end up without any like, longer grass, it could end up in another... like a desert or something and not have the food, like, enough food."

This excerpt revealed that Aaron drew on his programming experience in Frog Pond to explain his code for the cheetah-gazelle scenario. Like Charlie, Aaron was also able to draw on Frog Pond code to reason through some parts of the gazelle-cheetahs scenario. However, Aaron only drew on Frog Pond when asked to directly program the cheetah-gazelle scenario. He also used his Frog Pond experience in a more literal way: He thought that because in Frog Pond, if frogs move too far away from the center of the screen, they would fall off the lily pads into the water and die, so it would also be dangerous for cheetahs and gazelles to move too far off the center, because they would fall into some death zone, which is analogous to the water in the Frog Pond scenario.

Unlike Charlie, who fiddled with changes at both individual and population levels, Aaron singularly drew on the frogs' behavioral rules at the individual level and their immediate consequences. For instance, in the interview, he reasoned that he included the rule "move" because he had noticed that "if they [frogs] don't move, they couldn't get the flies". However he did not explicitly trace how these rules result in population level changes when applied over multiple generations. Moreover, he did not include some rules important for differential survival such as the birth variation rule that Charlie had included. In this way, both Charlie and Aaron drew on Frog Pond to reason through a co-evolutionary scenario in their post interview, albeit in different ways. While Charlie made connections between the micro and the macro levels, Aaron did not.

The finding that students using Frog Pond to think through this co-evolutionary phenomenon also found support in the existence of salient "traces" from Frog Pond that were seen in students' code. For instance, Charlie specified "41%" chance of reproduction as a rule in his hypothetical model. He said:

C: I'll put has 41% reproduce, on the offspring speed can vary, like be any speed, and same thing with the gazelle. And then there's the chance, 41%, that it'd reproduce.

I: Why 41? That's just a... [random percentage?]

C: I put that because on the frog pond simulator thingy, me and my friend who had a really stable [population].

Charlie used the exact percentage from Frog Pond, which is not the intended takeaway from the unit, but this trace shows that he knew that a stable population is an important factor of an ecosystem, and the stability can be achieved by manipulating randomness in the system. Because the paper-based code was not executable, he could not "tune" this number as he did with Frog Pond, but we interpret his use of this rule with the same probability value in the cheetah-gazelle scenario as marking an analogous approach and perspective. Similarly, Aaron also used code blocks from Frog Pond, such as "left", "right", and "hatch" (Table 1). He thought being able to move around and reproduce are important for the animals to survive. However, he did not attempt to align these rules with the specific scenario of cheetahs and gazelles. Further, he had not discovered the underlying connections between the animals' behavior and the distribution of various traits at the population level.

The above examples reveal that Frog Pond became a resource for students to draw upon when reasoning through evolutionary phenomena. Though there was variation in the sophistication of their explanations, both students transposed elements from Frog Pond into their explanations and accompanying code in the cheetah-gazelle scenario.

CONCLUSION

This case study provides promising evidence that the code-first Frog Pond environment facilitated learning about evolutionary mechanisms. After using Frog Pond, students shifted from adopting an event-based programming approach to a rule-based one. This shift is important because it suggests that students not only abstracted some regularities from their interactions with the code-first environment, but were able to modify and apply them to a different scenario with distinct evolutionary patterns. It also finds support in the literature that programming facilitates abstraction of rule-governed regularities [12, 16]. Such a shift is likely to pave the way for students to further engage in agent-based modeling and to understand other complex phenomena. In addition, Frog Pond was taken up as an object-to-think-with about evolutionary change as students drew on different aspects of the environment to explain a co- evolutionary scenario. These trends further open directions for future design and research. In addition, students' explanations about evolution have design implications. For example, in future iterations of the Frog Pond software and curriculum, we need to add features that help students make connections between micro-level rules and the macro level consequences in order to better support students' understanding about evolution.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation (grant DRL-1109834). Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect the views of the NSF. We also thank our teacher-collaborators, Sharon Churchwell and Adam Vernola, and our museum collaborators, Matt Matcuk and Jaap Hoogstraten. We are also indebted to several graduate and undergraduate students who also contributed substantially to this research: Amartya Banerjee, Krystal Villanosa, Arthur Hjorth, Randall Harris, and Austin Ryder.

SELECTION AND PARTICIPATION OF CHILDREN

We recruited a teacher who teaches six (6) seventh-grade science classes at a public middle school in an ethnically diverse suburb of a large Midwestern city. The teacher adopted the Frog Pond activities as a six-day curricular unit in her regular teaching, so all her 130 students were invited to participate in the study. The teacher explained the study to the students before the unit began and handed out both parental consent forms and student assent forms. About 100 students consented with the permission of their parents, and participated in the study.

REFERENCES

1. Beth A. Bishop and Charles W. Anderson. 1990. Student conceptions of natural selection and its role in evolution. *Journal of research in science teaching* 27, 5: 415–427.
2. Juliet Corbin and Anselm Strauss. 2008. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage.
3. Theodosius Dobzhansky. 1973. *Nothing in biology makes sense except in the light of evolution*.
4. Neil Fraser and others. 2013. *Blockly: A visual programming editor*. Google. URL: https://blockly-games.appspot.com/
5. Michael S. Horn, Corey Brady, Arthur Hjorth, Aditi Wagh, and Uri Wilensky. 2014. Frog pond: a codefirst learning environment on evolution and natural selection. ACM Press, 357–360. http://doi.org/10.1145/2593968.2610491
6. Eric Klopfer and Hal Scheintaub. 2008. StarLogo TNG: science in student-programmed simulations. *Proceedings of the 8th international conference on International conference for the learning sciences-Volume 3*, International Society of the Learning Sciences, 59–60.
7. John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4: 16.
8. Michael S. Horn and Uri Wilensky. 2011. NetTango 1.0 [Computer Software]. Evanston, IL: Center for Connected Learning and Computer Based Modeling, Northwestern University. Retrieved from http://tidal.northwestern.edu/nettango/
9. Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
10. Karl S. Rosengren. 2012. *Evolution challenges: Integrating research and practice in teaching and learning about evolution*. Oxford University Press.
11. NGSS Lead States. 2013. *Next Generation Science Standards: For States, By States.* National Academies Press Washington, DC.
12. Aditi Wagh. 2016. Building v/s Exploring Models: Comparing Learning of Evolutionary Processes through Agent-based Modeling (A dissertation). Northwestern University, Evanston, IL.
13. Aditi Wagh and Uri Wilensky. 2014. Seeing patterns of change: Supporting student noticing in building models of natural selection. *Proceedings of Constructionism 2014*, Vienna, Aug 19-23.
14. Uri Wilensky. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
15. Uri Wilensky. 2016. Why schools need to introduce computing in all subjects. Retrieved April 3rd, 2016 from http://theconversation.com/why-schools-need-to-introduce-computing-in-all-subjects-53793
16. Uri Wilensky and Kenneth Reisman. 2006. Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach. *Cognition and Instruction* 24, 2: 171–209.
17. Uri Wilensky and Mitchel Resnick. 1999. Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology* 8, 1: 3–19.
18. Uri Wilensky, and Michael Novak. 2010. Understanding evolution as an emergent process: learning with agent-based models of evolutionary dynamics. In Roger.S. Taylor & Michel. Ferrari (eds.), *Epistemology and Science Education: Understanding the Evolution vs. Intelligent Design Controversy*. New York: Routledge.
19. Uri Wilensky and Damon Centola. 2007. Simulated evolution: Facilitating students' understanding of the multiple levels of fitness through multi-agent modeling. Paper presented at the Evolution Challenges Conference. Phoenix, AZ. November 3, 2007.
20. Michelle H. Wilkerson-Jerde and Uri Wilensky. 2010. Restructuring change, interpreting changes: The DeltaTick modeling and analysis toolkit. *Proceedings of Constructionism 2010*. Paris, France, Aug 10-14.
21. Michelle H. Wilkerson-Jerde, Aditi Wagh, and Uri Wilensky. 2015. Balancing Curricular and Pedagogical Needs in Computational Construction Kits: Lessons From the DeltaTick Project. *Science Education* 99, 3: 465–499.