Ecology and Evolution Simulation and Quest Design for an Educational Massive Multiplayer Online Game

by
Mark Zhang
B.S., Massachusetts Institute of Technology (2013)

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
June 2013

Signature of Author .....
Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by .....
Eric Klopfer
Professor of Urban Studies and Planning
Thesis Supervisor

Accepted by .....
Dennis M. Freeman
Professor of Electrical Engineering and Computer Science
Chairman, Masters of Engineering Thesis Committee

Ecology and Evolution Simulation and Quest Design for an Educational Massive Multiplayer Online Game

by
Mark Zhang
B.S., Massachusetts Institute of Technology (2013)

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
June 2013

Abstract

In this design-based research project, I developed two simulations to be used as student tools in a massively multiplayer online game targeted at STEM education, the Radix Endeavor. I designed both the underlying agent-based model as well as the user interface for each simulation, and furthermore designed quests for my simulations for the purposes of playtesting. My final ecological prototype is able to authentically model fairly complex food webs of six or more organisms, and my final evolutionary prototype can handle complex fitness relationships between the individual traits of a single population and various environmental factors. In my thesis, I discuss the design and implementation of these simulations, the feedback we received from students, the overall effectiveness of my prototypes, and recommendations for further work.

Thesis Supervisor: Eric Klopfer
Title: Professor of Urban Studies and Planning

TABLES OF CONTENTS

1. Acknowledgments

I would like to thank the Radix Endeavor team, particularly my supervisors Louisa Rosenheck and Eric Klopfer, and their colleagues Susannah Gordon-Messer and Angie Tung, for giving me the opportunity to work on this project and for their encouragement, expertise, and excellent feedback.

Furthermore, I would like to thank Mr. Mark Knapp of Josiah Quincy High School and Amanda Tsoi of Somerville High School for permitting us to playtest our prototypes with their students.

2. Introduction

There has never been a greater need for improved STEM education in the United States. A strong foundation in math and science is essential for the kinds of skilled jobs present in the modern global workplace, and the fastest growing job markets right now are ones in which science and mathematics play a central role. Meanwhile, the United States' economic edge is slipping, as demonstrated both by a shrinking share of filed patents and also by a diminishing share of high-tech exports worldwide (Achieve, 2013).

Unfortunately, U.S. students are performing far below par in mathematics and science compared to the rest of the world. To cite just one of many statistics, the U.S. ranked 17th in science, and 25th in mathematics on the 2099 PISA assessment, and less than ten percent of U.S. students scored in the top two of six performance levels (Gurria, 2010). There is a pressing need for innovation in STEM education.

We believe that Massive Multiplayer Online (MMO) games are a promising educational medium. MMO games are a popular game genre in which players control avatars in a persistent multiplayer world. MMOs have several elements, like quests and item collection, which lend themselves well to STEM learning. One of the biggest benefits of MMOs is the collaborative nature of gameplay, which we believe offers a great opportunity for collaborative learning. Another great aspect of MMOs is their ability to immerse, and we believe that with proper execution, an educational MMO can greatly increase internal motivation towards STEM learning.

The Radix Endeavor is an exciting current initiative at the MIT Education Arcade to develop an educational MMO Game for STEM learning at the high school level. The initial version of the game is designed to align with the Common Core standards in mathematics and Next Generation Science Standards for high school students, newly designed and up-to-date educational standards that have been adopted by almost every state in the past few years. The core game-play mechanic behind Radix involves player-controlled avatars exploring the MMO world and solving STEM-related problems, or "quests" in the MMO parlance, for its various inhabitants. Our goal is to design quests in such a way that students will discover key STEM concepts for themselves through investigation and inference.

Two related areas that the Radix Endeavor will focus on are ecology and evolution. Designing authentic and easily configurable models and intuitive and engaging user interfaces for ecology and evolution, as well as quests utilizing these tools, with the goal of conveying certain educational concepts will be the focus of my thesis.

3. Goals

In this section, we will first discuss how our project fits into the Radix Endeavor, and touch on goals of Radix that are relevant to our discussion. Then, we will discuss the scope of the simulations, and the goals for the models and the user interface.

3.1 Overall Goals

In the context of Radix, the simulations will serve as a tool that the player can access when he or she is performing ecology and evolution-related quests . We envision that the player will travel to various parts of the Radix game world and meet many different non-player characters in a variety of environments. These different characters will pose different ecological and environmental inquiries and challenges. For example, perhaps over-hunting has reduced a prey population to a critical level. Or perhaps a volcano has recently become active and is erupting at constant intervals, drastically changing the surrounding environment.

To solve these quests, players will use a simulator tool to make predictions and inferences as well as suggestions for how to fix or mitigate the problems. For example, to discover the best course of action to help a struggling ecosystem, the player might run ecosystem simulations with various starting conditions (i.e. more deer or less wolves) on their simulator and observe which starting conditions produce the most balanced results. This highlights a unique aspect of Radix: players essentially role-play as scientists and do the kind of work that real scientists are doing. Thus, our goal with quest design is to make the quests as authentic to real-life science as possible while still making it engaging and interesting for the students, and likewise, our goal with the simulations is not to provide straightforward answers, but to leave room for student interpretation and inference.

It should stressed that the primary goal of this project is not to develop new ways of modeling ecosystems for scientific research, but for pedagogical purposes. The consequence of this is that simulations do not have to be extremely precise as long as they convey enough information for students to be able to draw inferences. However, we were

careful to eliminate all behaviors in our simulation that could potentially mislead students or cause them to draw inaccurate conclusions.

3.2 Ecology Simulation Goals

For the ecology module, we plan to focus on relationships between populations in an ecosystem, particularly predator/prey relationships. Things that we do not plan on focusing on include energy and nutrient transfer, reproduction and death rates, and immigration/emigration. While these may be implicit in our model, in the interest of reducing cognitive load on the student, we did not emphasize these aspects in the simulation. Through our simulation, we want to give students an intuition about predator-prey relationships over time, what kinds of starting conditions are viable, and finally a sense of how food webs in an ecosystem are interconnected and what happens if one part of the web is disrupted.

The simulation should believably model real-life behaviors to give a sense of realism. Populations sizes should not oscillate wildly, and they should thrive and die out when expected. Secondly, the simulation should be dynamic and easy to make inferences from. Populations in the stable state of the system should oscillate between high and low points based on the size of other populations, instead of simply leveling out. This allows students to better see the relationships between different populations. Thirdly, the system should be flexible and easy to extend. This is crucial because we will seek to add new content to our ecology module after our initial launch, and we do not want to overhaul our system. These requirements also apply to the evolution simulation which follows.

3.3 Evolution Simulation Goals

Our main curriculum goals with the evolution module are looking for contemporary evidence (for example, variation of traits in current populations), looking for historical evidence (fossils), and natural selection over time. The areas related to evidence are not well-suited for a simulation, so Radix relies on narrative to convey these concepts. The evolution simulation focuses on the distribution of quantitative, continuous traits in a population over time in the fact of changing environmental factors. For example, we might look at the distribution of fur length as the climate changes. We will not focus on the genetics underlying these traits, including the dominant and recessive nature of certain traits. In particular, we do not account for spontaneous genetic mutations which produce novel, advantageous phenotypes in our simulation. We only model organism traits such fur length that range along a certain value.

Through our simulation, we obviously aim for the student to understand how the process of natural selection works over time to select the best fit phenotype, at least with respect to quantitative, continuous traits. We also aim to convey through our simulation's design that populations, rather than individual organisms, evolve over a long time period of time. We will discuss the user interface components that we designed to facilitate this learning in following sections.

3.4 User Interface Goals

So far, our discussion has been primarily on the models backing the simulations, but the user interface that is presented to the student is also very crucial to the simulation's success. The user interface consists of the sliders and buttons which allow the student to

control the simulation, as well as the graphs, lists, and visuals which display the simulation's state. The design of the user interface will determine how accessible and engaging our simulation is to the student. The goals for the user interface that it be comprehensive, intuitive, and visually appealing.

Obviously, the UI should be comprehensive - that is, it should allow the student to view all relevant data from the model which would be useful towards understanding the educational objectives. Intuitiveness is also critical. We know from experience that the average student has a fairly low attention span, and that if the parts of the user interface are hard to discover or use, they will simply be ignored. Instructions, too, are rarely read so we require an interface that is instantly usable. Finally, it is extremely important for the simulation to be visually exciting, because it captures the student's interest and makes them excited about the subject matter. Colorful graphs that update quickly, and a grid that shows the movements of an ecosystem's organisms are examples of ways to make the simulation more visually engaging. While we did not include them in our prototype, graphics such as animated animals would also go a long way towards making the simulation visually attractive.

4. Background

Several ecological and evolutionary simulations already exist, and some of them are quite excellent in execution. In this section, I will go over four prominent examples of previous work in ecology and evolutionary simulations, discussing their merits as well as

any potential drawbacks. Screenshots of these simulations could not be included without permission, but the simulations are all readily available online through the URLs provided in the bibliography. I will then present two approaches to modeling these simulations, including previous work, and justify our decision to go forward with one of these approaches.

4. 1. Previous Work

4.1.1 Ecology Simulation - Habitable Planet Food Web Simulation

The first ecological simulation was executed by Habitable Planet, an online teacher development course on ecological systems on Earth. The simulation focuses on food webs and allows the user to specify which predators and prey are present and what the relationships between these populations are (Habitable, 2013). A small window in the interface shows the current state of the simulation with various animal-shaped icons, while a large graph which displays the populations over time dominates the interface. Based on the smoothness of the graphs and the fact the numbers play out the same way each time, the simulation appears to used a system-based model, with the state of the model governed solely by various equations. The pros and cons of this will be discussed in the following section.

One ingenious aspect of the simulation is that it is possible to create many different kinds of food webs simply by turning various relationships in the simulation on and off. The interface is easy to use and pleasing to the eye, but does not allow the user to specify the starting numbers of each population, which means it cannot be used to learn about how different starting conditions can determine the ultimate fate of an ecosystem. Additionally,

all the populations are squeezed onto a single graph with a common scale. Since some of the populations are naturally much smaller than others, this makes it difficult to understand the relationships between different populations at a glance, since it is difficult to tell whether a smaller population is increasing or decreasing. I will want to preserve the clean, simple interface of the Habitable Planet simulation, while allowing for greater flexibility in the simulation to maximize educational versatility, as well as making the display more comprehensive.

4.1.2 Ecology Simulation - Explore Learning Food Chain "Gizmo" Simulation

Another prominent ecological simulation is from the online learning website ExploreLearning (ExploreLearning, 2013). The simulation features a fixed food web with grass, rabbits, snakes, and hawks. The user has the ability to change the numbers of the various populations and change each population to "diseased" which appears to reduce their reproduction rate. Interestingly, the user is able to perform these actions while the simulation is running. This ability to make adjustments to the population sizes in real-time is a big strength because it makes the simulation more interactive and engaging. While we ultimately chose not to include this feature because we felt that it could be abused and distract from the purpose of the module (it would mess with the relationships that were displayed in the graphs), we nevertheless note its potential for engagement if placed in a carefully designed interface. Additionally, the simulation is flexible enough for a variety of starting conditions, although it only allows for a single food-chain.

The major drawbacks are in the visuals. The image of the hawks and rabbits does not change depending on the population size, so the only thing that really changes in the

simulation are the numbers, which I believe makes it less engaging for the students. Additionally, the graphs display the population size as a percentage of the starting condition, which makes it difficult to figure out the actual size of the population. Overall, however, this representation does allow the graphs to do a better job of showing relationships between populations than the Habitable Planet simulation, although placing the populations in separate graphs would make it easier to discern actual magnitude. While this simulation suffered from some minor problems, it nevertheless provided a good reference as I built my own simulation.

4.1.3 Evolution Simulation - PhET Simulation on Natural Selection

One noteworthy example of an educational evolution simulation is a PhET simulation on Natural Selection. This simulation centers around a population of rabbits, which automatically reproduce after a certain amount of time (Adams, 2011). The user can choose to add a mutation (like brown fur and long tail) at will, and can also choose selection factors like wolves and the location of the ecosystem (equator or arctic). Half of the screen is taken up with a visual display of the ecosystem, a large graph dominates the other half.

The simulation is fun to play with and fairly easy to use. However, the graphs are very difficult to interpret. Every single variation of rabbit is superimposed on the same graph. Additionally, there is no sense of variation. A rabbit either has brown fur or white fur. There is no in-between. Finally, there are a small number of inaccuracies. Allowing users to specify mutations at will is interactive, but has the danger of conveying a false impression of how mutations work. Being able to change environments (from the equator

to the arctic) is also unrealistic. There is also no upper limit to the population. The PhET simulation is probably the most fun out of all the simulations here, but we would like to preserve as much fun as possible while making our simulation more scientifically accurate. We think it's important for teachers to be able to use our simulations without requiring them to explain to students each time that so-and-so doesn't actually occur in real life.

4.1.4 Evolution Simulation - EvoBeaker Simulation on Darwinian Snails

Of all the simulations I surveyed, the EvoBeaker's natural selection simulation on Darwinian snails most closely matches we are aiming for (Simbio, 2013). The simulation is intuitive, has a beautiful interface, and is very engaging. I particularly like that one of the primary ways that the student can interact with the simulation is by dropping predators (crabs) into the population of slugs. It is a very immersive action which I'm sure students loved. The visual representation of the ecosystem is also very evocative. The simulation allows you to have multiple ecosystems side-by-side which have different environmental attributes, which you can use for comparisons. The simulation also uses multiple graphs instead of trying to fit everything on one graph. While we have slightly differing educational objectives, the EvoBeaker simulation proved to be an excellent reference for my evolution simulation.

4.2 Agent-based versus System-based Modeling

When it came to determining our simulation's underlying model, there were two main approaches. The first was a system-based approach: we would model the relationships in the system with equations. Each population was a variable, and each

equations stated the rate of change of a population with respect to the values of the other populations. At each time-step, the state of the system was determined by the state at the previous time-step and the relationship equations. The second approach was an agent-based approach. Each individual animal in a population was modeled separately and given certain behaviors and properties. At each time-step, each animal made an action, and the state of the system was the aggregate result of the behavior of all the animals in the system. Agent-based systems relied on proper regulation of the emergent behaviors of the system.

Educational simulations have been developed using both of these approaches, so they are both viable. One prominent example of an agent-based education system is the successful StarLogo initiative, where students are able to program their own agent-based simulations using a simplified and highly visual programming language. A paper on StarLogo discusses several case studies where simulations programmed in StarLogo were tested on their educational value to students (Klopfer, 2009). The paper noted that the dynamicness and unpredictability of some of the simulations, for example one in which students played around with simulated forest fires, made it more interesting for students, a aspect we will touch upon again. The paper also documents some successes with teaching students through games. The agent-based made it easy for students to program small pieces, which would then come together to form more complex behavior.

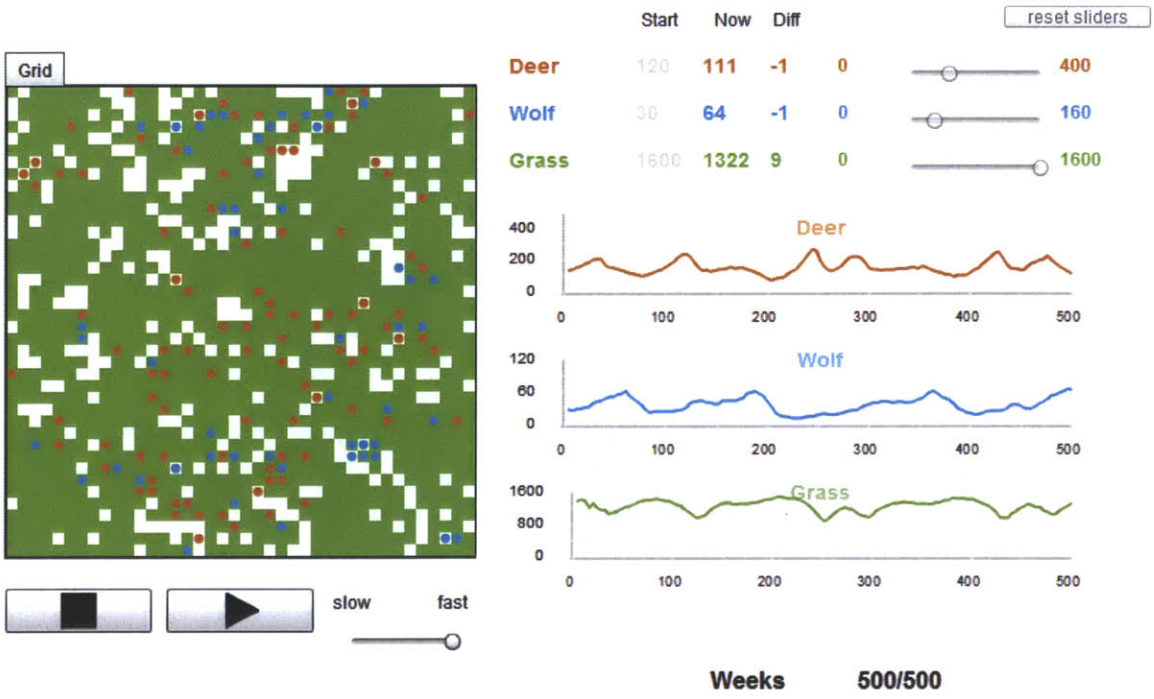
Stella is an example of a system-based simulation system built for educational purposes. Based on the online material, it appears that Stella specializes in allowing students to find patterns and answer questions about data (Isee, 2013). This is certainly an advantage of system-based models. They are more well-behaved and when well-executed, are easier for students to reason about and draw inferences from.

Both of these approaches have their strengths and weaknesses. Using initial prototypes, we found the agent-based approach to be more realistic and easier to extend but less predictable and somewhat more resource-intensive. On the other hand, the system-based approach was very predictable and light-weight, but also harder to control. The equations were a less natural way to specify the constraints of the ecosystem, and tweaking the system was more difficult and time-consuming, requiring lengthy calculations to convert the variables in the equations to more intuitive variables like population size and reproduction rate. Additionally, extending the system to handle evolution would be easy from an agent-based standpoint, but modeling evolution with equations would be more difficult and convoluted. Based on those observations, we decided to move forward with an agent-based approach for our simulation.

5. Ecological Simulation

The Ecosystem Simulation simulates several populations of animals in a food web over a series of time steps. The populations interact with each other in predator-prey relationships. The simulation is agent-based and involves Organism agents, which include plants and animals. These Organisms interact within an Ecosystem.

I will first describe the structure of the underlying model and how the simulation runs in each timestep. I will then go over the various components of the UI. Throughout the discussion, I will explain the rationale behind various design decisions.



The Ecological Simulation

5.1 The Model

In the ecological simulation, the ecosystem is modeled by a collection of agents which consist of plants and animals. Each agent occupies a space in the ecosystem, which is represented by a two dimensional grid.

5.1.1 Animal Parameters

Animals are described by the following attributes:

- **Name** - The name of the animal.
- **Location** - The animal's location in the ecosystem grid.

- **Energy** - Energy determines whether predators live or die. Every timestep, the energy of animals decreases by 1. Animals gain energy by eating prey.
- **Reproduce period** - This is the official number of timesteps in an organism's reproduction cycle.
- **Reproduce counter** - This is the number of timesteps before an organism can reproduce again. Every time it reaches 0, the organism makes an offspring, which is placed in a location adjacent to the parent. The reproduce_counter is then reset to reproduce_period. It is necessary to randomize the initial values of reproduce_counter when first creating the ecosystem (that is, assigning every organism a random reproduce_counter value between 0 and reproduce_period), as opposed to setting everything to reproduce_period, because we want the population size to increase smoothly and not suddenly double in one timestep.
- **Brood size** - The number of offspring that are born each time the organism reproduces. For animals, this is always one, but for plants we tried values of two to four. Even if the brood_size is four, if there is only one adjacent square to an organism, it will only make one offspring. Thus, in many cases, the brood_size for plants is irrelevant. Brood_size matters most when there are fewer plants in the ecosystem and relatively high perimeter-to-animal ratio. Higher brood-size means plants recover faster from disaster. We generally kept a brood_size of 2 for plants.
- **Hunting limit** - The hunting limit system is central to our ecosystem simulation and will be discussed further later. The hunting limit for an animal represents the population density under which this animal's predators begin to have difficulty catching the organism (their chance of catching this animal as prey decreases). For

example, if the hunting_limit of an animal is 0.1, then once the animal occupies less than 0.1 of the squares in the ecosystem, the predators of this organism will have a less than 100% chance of catching the animal. This is an example of an ecological modeling concept known as density-dependent predation.

- **Minimum hunt rate** - the minimum possible chance that this animal has to catch one of its prey. This number gives us additional flexibility in our ecosystem model. If we find that the predator population is dropping too quickly even though the prey population is starting to recover but hasn't yet overcome its hunting_limit, we could raise this number, which would help slow the predator population's decrease at the prey population's expense. Ideally, this would save the predator population while still allowing the prey population to recover at a slower rate. Obviously, a minimum_hunt_rate that is too high would cause an animal to kill off its prey populations completely.
- **Prey type** - This specifies the names of the types of organisms that this organism can consume, and how much energy the organism recovers when consuming each prey. In our ecosystem, organisms have only one opportunity to catch prey, and this is when their energy is 0. A more intuitive approach would be to allow the organism to catch prey whenever their energy goes below a threshold and they are "hungry". However, our approach of only allowing one chance makes it much easier to reason about what percentage of a given population should be dying at any given period, and it is important for our simulation to be extensible and easy to tweak.

5.1.2 Plant Parameters

Plants are similar to animals but slightly simplified. Specifically, the following attributes are nonexistent:

- **Prey type** - plants do not have prey.
- **Minimum hunt rate** - plants do not catch prey
- **Energy** - plants do not die from lack of energy. The only way a plant can die is by being consumed.

5.2 A Timestep in the Simulation

Every timestep, each animal performs the following actions:

- Generates offspring according to the reproduce_counter
- Move around the ecosystem (the only purpose of this is to make the visualization look interesting. It does not affect the simulation's behavior)
- If their energy is at 0, attempt to catch and kill a prey. (see Catching a Prey)
- If the animal failed to catch a prey (due to its prey's hunting_limit) the animal dies.

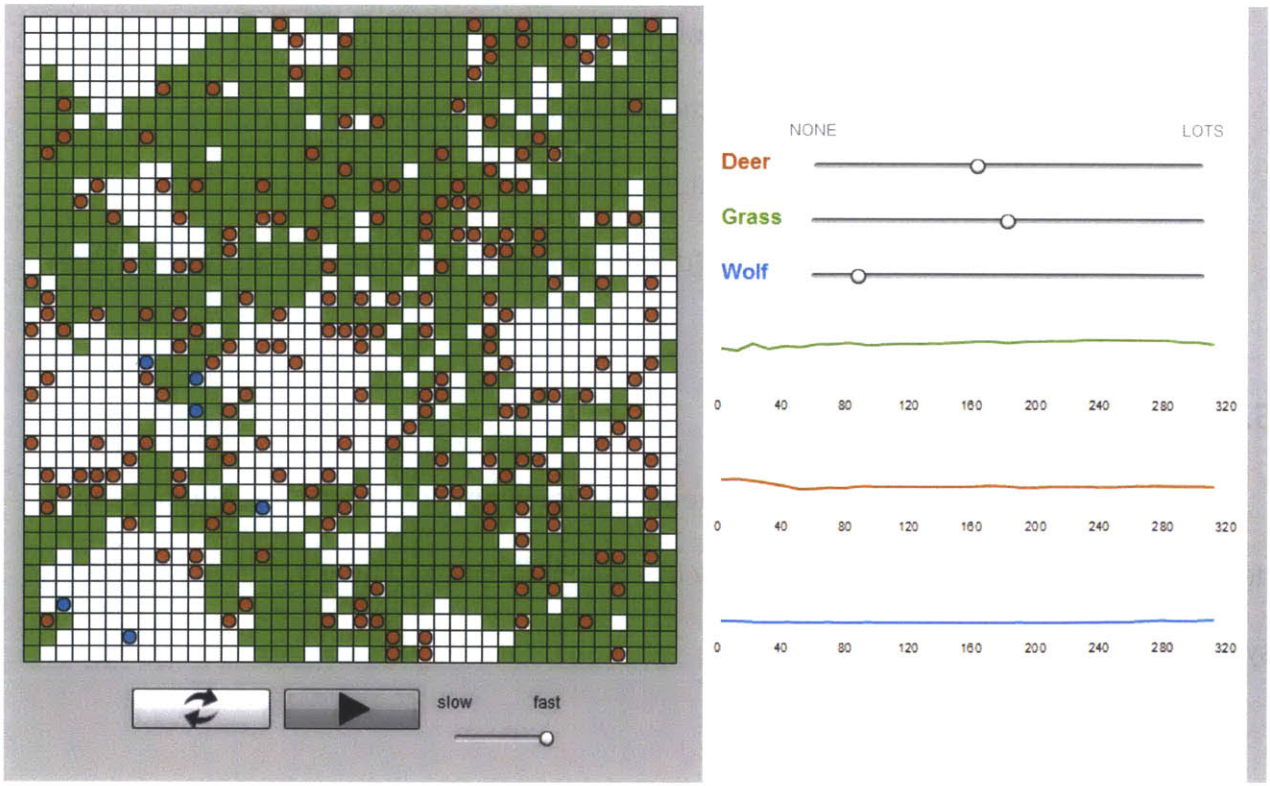
5.3 Predator-Prey Modeling

Here we will discuss the way we model predator-prey relationships, and our major design decisions. We will also explain why our model works, and how it mirrors real-world systems.

5.3.1 Location-independent Predation

In our simulation, predation does not depend on location. That is, a predator can catch a prey from anywhere on the map. The only factor that affects whether catching a prey is likely is the total number of prey in the ecosystem. Thus, our model is individual (agent-based), but non-spatial.

We experimented extensively with only allowing predators to catch prey close to them. This includes allowing first-level predators to only eat plants close to them, as well as allowing second-level-predators to only eat animals close to them. Under this scheme, there was no hunting limit as described above. Predators and prey populations would naturally fluctuate due to predation and lack of prey. A screenshot of an early prototype with this scheme is shown below.



Location-dependent predation with inhomogeneity in ecosystem

When predation is location-dependent, the biggest difference is lack of homogeneity in the ecosystem. Some parts of the ecosystem will have no organisms, while other parts will have many. In general, the plant distribution determines the distribution of other organisms. This lack of homogeneity has a very attractive aspect - it makes the simulation captivating to watch. The ecosystem is very dynamic and different parts of the grid will teem with organisms and then be completely dead fifty timesteps later. The student can watch as consumers migrate to follow their food source. In short, lack of homogeneity produces a lot of interesting emergent behavior.

The biggest problem we faced, which caused us to reject location-dependent predation, was that it was much harder to keep the ecosystem balanced. Take the case of a second-level predator. It turns out that it is much harder to keep second-level predator populations alive when the prey population is in a downswing. This is because in order for the predator to feed, not only must there be sufficient prey, but the predator must be able to catch the prey. If we gave the predator an excellent movement algorithm, so that it would always be able to find the nearest prey, then the predators would kill off the prey because they were overly effective hunters. However, if we gave the predators a poor movement algorithm, they tended to die off even though there were prey elsewhere on the grid. Basically, we wanted the effectiveness of the predator to be dependent on the number of prey in the ecosystem, and location-dependent muddled this connection. For simplicity, we chose to have the underlying model be location-independent.

However, even though the model is fundamentally location-independent, there are still steps that could be taken to give the appearance of location-dependent predation. For

example, whenever a predator successfully feeds, we could have it kill a prey nearby if such a prey exists, and kill a prey randomly on the board otherwise. While I did not implement any of these in my prototypes, maintaining the appearance of location-independent predation could allow for emergent behavior that has high potential for engaging students.

5.3.2 The Hunting Limit

The hunting limit in my simulation is the mechanism that provides the balancing force for predator-prey relationships. As discussed above, the hunting limit is described as a threshold proportion of grid squares in the ecosystem that the prey population must occupy. The hunting limit is used as follows to determine predation success:

For a given predator, the chance of catching a prey is determined as follows:

- if the prey population is above the hunting_limit, the predator always catches a prey
- if the prey population is below the hunting_limit, the chance of catching a prey is *(minimum_hunting_rate) + (1 - minimum_hunting_rate) * (current_prey_population) / (total_spaces_in_ecosystem * hunting_limit)*. In other words, once the prey population goes under the hunting limit, the chance of catching a prey decreases linearly to the minimum hunting rate. To determine whether a prey is actually caught, we use the built-in random() function.

As an example, if the size of the ecosystem is 40x40 = 1600, the hunting limit is 0.1, the number of prey is 80, and the predator's minimum hunting rate is 0.2, then the chance that the predator catches a prey is 0.2 + 0.8 * 80 / (1600 * 0.1) = 0.6. Of course, in our

system, since the predator dies if it does not catch a prey, this also represents the chance that this predator will die this turn.

We chose in our simulation to have the diminishing force on the predators depend only on the prey density. This is different from other models like the popular Lotka-Volterra equations (Weisstein, 2013), which has the diminishing force on the predators depend on the product between predators and prey. One advantage for this is simplicity: it is easier to tweak the model if we need to worry about fewer variables and our simulation appears to exhibit realistic behavior even with this simplification. Here is our motivation:

There are several factors in real-world systems that would prevent predator populations from growing without bound. Many of these factors, like for example, competition for limited habitat space, tend to have a stabilizing effect, rather than a diminishing effect. In some ecosystems, these stabilizing factors may cause the predator population to level out before it grows large enough to overpower the prey population. However, since our simulation aims to show clear relationships between predators and prey, we do not want an ecosystem in complete stasis. Thus, we do not have any "upper limit" to the predator populations.

If there was no upper limit to a predator population, the prey population would eventually become over-strained and begin to decrease, until prey is so scarce that the predators begin to starve. Lack of prey appears to be a primary diminishing force on a large predator population and the one we are interested in. Furthermore, a larger predator population is naturally factored into the diminishing force on predators because the larger predator population will naturally decrease the prey size. Thus, it does not seem necessary

to include the larger predator population in the equation as a separate variable. Thus, for simplicity, the predator chance depends only on the prey population size.

When there are too many prey, the predator population will continue to rise since the prey population is over the hunting limit. When there are too many predators, the prey population will drop until it goes below the hunting limit and the predator population begins to reduce. The hunting limit needs to be chosen with some care. If the hunting limit is too high, then the predators obviously will die out because the prey population is always so low that a portion of the predator population is dying off. However, if the hunting limit is too low, then the predator population will increase to an unsustainably large number. Then, even when the prey population does finally dip below the hunting limit, the predator population will still be large enough to consume the prey population entirely. A delicate balance needs to be held.

We have observed that a higher hunting limit generally causes smaller oscillations in the population sizes over time, whereas a lower hunting limit generally causes greater oscillations in population sizes. The fact that a low enough hunting limit causes the prey populations to die out is simply an extension of this observation, where the oscillations are so great that the troughs hit zero, from which there is no recovery. One of the goals of our model is for relationships between predator-prey populations to be easy to see. Thus, we have chosen a fairly low hunting limit for our prey populations. A list of parameters that produce good simulations can be found in Appendix A.

5.3.3 The Hunting Limit with Multiple Prey

In order to handle more complex ecosystems, I adapted the hunting limit to handle multiple prey types.

As a reminder, for a single prey population, if the prey population is below the hunting_limit, the chance of catching a prey is *(minimum_hunting_rate) + (1 - minimum_hunting_rate) * (current_prey_population) / (total_spaces_in_ecosystem * hunting_limit)*.

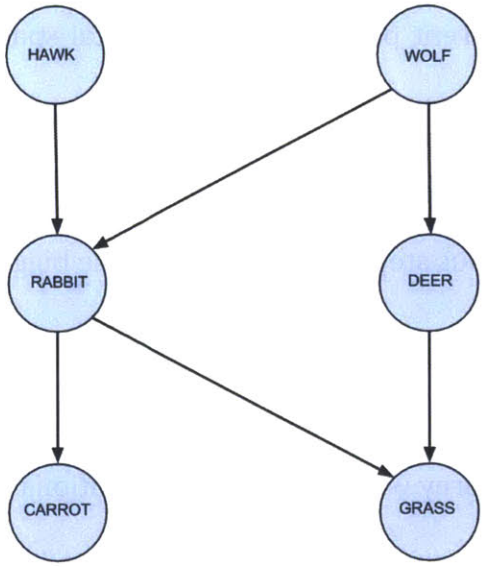
When there are multiple prey, the chance of catching some prey is determined as follows:

- When selecting which prey to use in the above equation, take the prey with the maximum value of current_prey_population / (total_spaces_in_ecosystem * hunting_limit).
- If it turns out that a prey was caught, the prey that is caught is the same prey which was used in the previous step, i.e. the prey with the biggest population relative to its hunting limit.

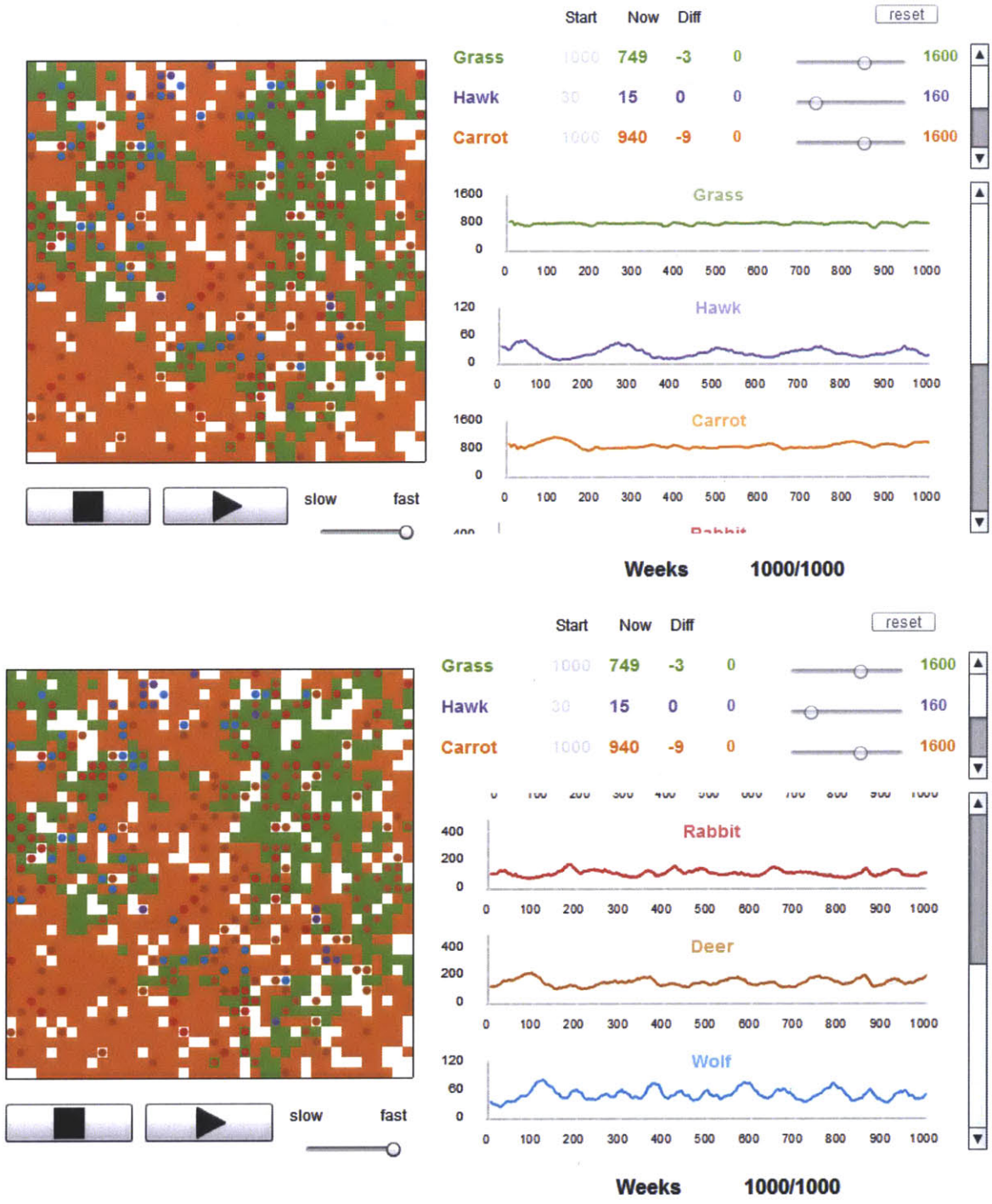
Extending the hunting limit mechanism in this way makes it easy to create complex food webs. The rule that the prey with the biggest population is always caught creates an additional balancing force. We observed that without this rule, one of the prey populations tended to be consumed more than the other. Even if this happened for a short period of time, since the only augmenting force on prey populations is reproduction, the impact of the imbalanced predation would be felt for many generations. Adding the biggest population rule alleviated this problem.

5.4 More Complex Ecosystems

For the most part, the system described above could handle complex ecosystems of up to six organisms, while preserving the fluctuations in population size that allowed students to easily identify predator-prey relationships. A period of trial and error is necessary to pick good values for the hunting limit, but we have found that this process is relatively painless. Below we have included a screenshot of a simulation run with 6 organisms, and the food web that describes their relationships. The parameters that describe these simulations are included in Appendix A.



The Food Web for a Complex Ecosystem



Screenshots of a single run of the Complex Ecosystem

As long as none of the populations died out, everything ran smoothly. However, the following occurred when we removed grass:

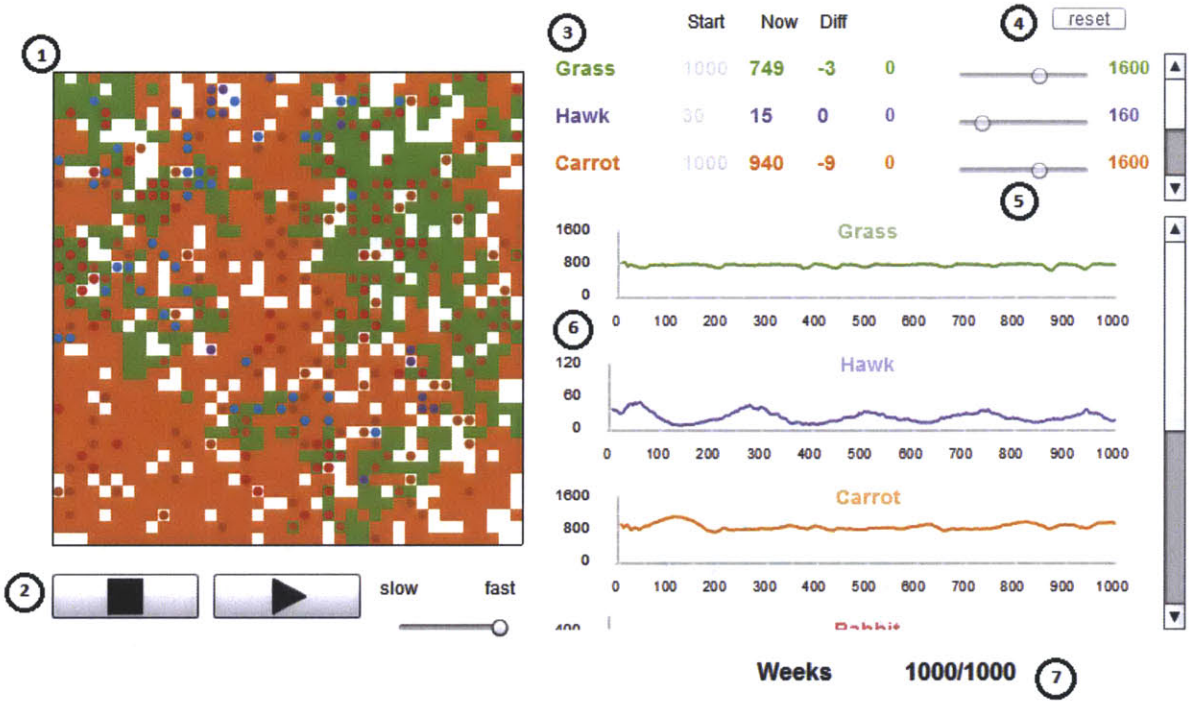
- The equilibrium point for the rabbit population was much lower than when both grass and carrots existed, even though the deer were no longer competing with the rabbits.
- As a result, the hunting limit for the rabbits was too high, and the hawks and wolves both died out.

This is an unfortunate consequence of our hunting-limit mechanism. A hunting limit that worked well when the rabbit population was sustained by two plant populations was too high when there was only one plant population to sustain the rabbits. The way we chose to deal with this was to introduce "ecological modes." Essentially, we would change the hunting limits of various prey populations once the ecosystem had entered into a fundamentally different mode, i.e. one with fewer animals. This worked well for the case where grass died out, but has the disadvantage that in order for the simulation to work in all cases, we would have to thoroughly test and discover all the "modes" and make separate parameters for each case.

Alternative approaches may be possible. In real-world systems, it seems that when prey is scarce for a period of time, predators undergo behavioral modifications that make them more adept at catching prey (or the prey's defense mechanisms become less effective due to fewer numbers). It may be possible to add more parameters to our model to account for this, and remove the need for ecological modes. This approach appears promising as a route for further research.

5.5 User Interface Highlights

We will now briefly highlight some of the elements of the user interface design for my ecological simulation prototype.



1) Prominently displayed is a colorful view of the current state of the ecosystem. Students can watch as various populations sizes increase, decrease, and move about the board.

2) The intuitive controls for the simulation are here. In additional to play and pause, students have the option to run the simulation quickly when they simply want to see the results, and slowly when they want to take a closer look.

3) We provide three numbers for each population: the original size, the current size, and the change in the last timestep. These three numbers provide the student comprehensive information without being overwhelming. The original size is provided to remind the student what the initial starting conditions were. The change in the last

timestep lets the student know whether the population is currently increasing or decreasing and by how much. By using the current sizes and changes in each population, the student can infer how different population sizes affect each other. Note that the color of each organism is consistent throughout, to make it easier for students to relate the various pieces of data to the same organism.

4) The reset button allows the student to quickly reset the starting conditions back to the original starting conditions that were provided, for ease of use.

5) The sliders allow the student to change the starting conditions and are only movable when the simulation has been reset.

6) Each population is displayed on a separate graph, and each graph has its own axis, which is clearly labeled. The range of the y-axis on each graph is configurable. Unfortunately, with multiple graphs, the interface becomes clunky with scrollbars, which are not the optimal solution. A basic tabbed solution (which we used in the Evolution Simulation) wouldn't work here because it is essential that students be able to compare multiple graphs side-by-side to see relationships between them. However, having multiple graph slots, and the ability to select which animal to view for each might work. Squeezing all six graphs into a 2x3 arrangement could also conceivably work.

The advantage to having all populations overlayed on one graph is that it is much easier to see relationships between the populations. One disadvantage is that in a complex ecosystem, the number of overlayed graphs can be overwhelming, so the UI would need to allow the user to make certain graphs invisible. The main obstacle to having a single graph is that it is difficult to have the producers and second-level consumers on the same graph because their population sizes are orders of magnitude apart, and we did not want to

downplay this fact. While multiple graphs is a temporary solution, this is one area of the interface that could use additional work.

The scale for the y-axis of each graph is currently manually set in the prototype. I picked the numbers by running the simulation a few times, and picking a max y value for each population to be about 25% larger than the largest value seen for that population. I decided not to have the graphs dynamically resize as the simulation ran because the resizing had a disorienting effect.

7) Each simulation runs for a set number of weeks. It was important to pick a specific amount of time, i.e. weeks, instead of simply saying "timesteps", to give the student a concrete sense of time.

While the UI is not without its flaws, particularly for complex ecosystems, on the whole, I believe that it is intuitive, comprehensive, and visually appealing.

5.6 Summary

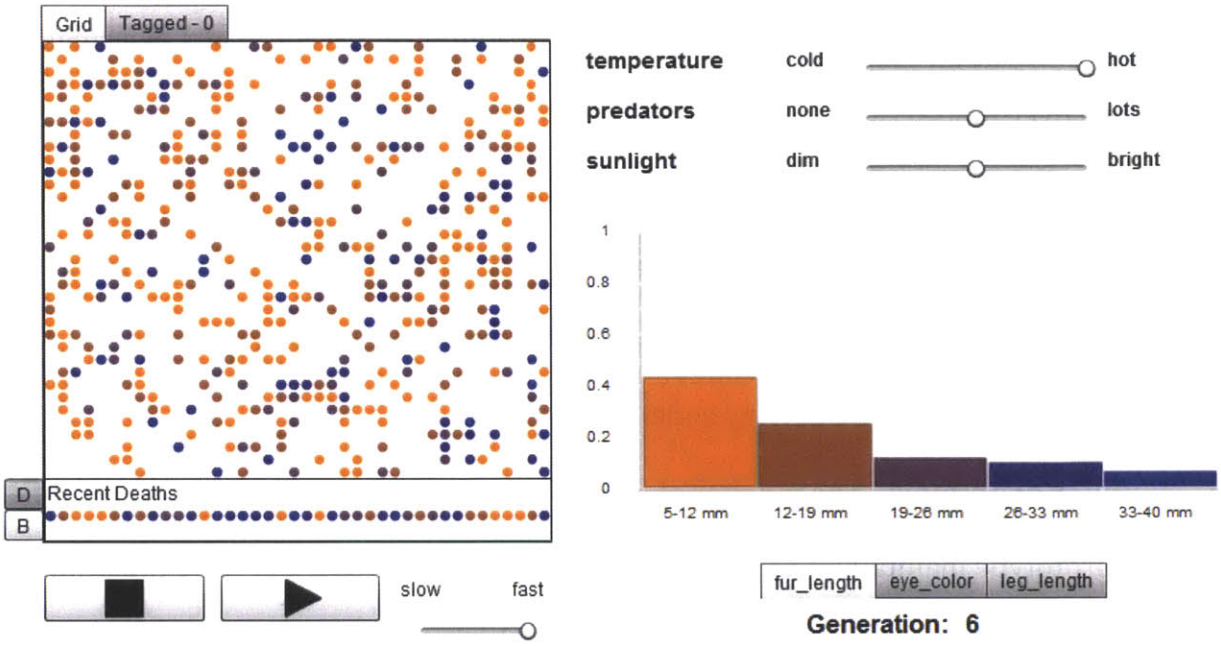
In this section, I described the underlying model and highlighted the user interface for my ecological simulation. The ecological simulation features a location-independent predation mechanism where the chance of catching a prey depends solely on the size of the prey population. The simulation works well even for complex ecosystems as long as none of the populations die out (which shouldn't happen in balanced ecosystems). The user interface features individual graphs and a carefully chosen set of statistics for each population, resulting in a user experience that is intuitive, comprehensive, and visually appealing. In the following section, I will describe my evolutionary simulation.

6. Evolutionary Simulation

The Evolution Simulation simulates the distribution of traits in a population of animals over many generations in response to certain environmental attributes. For example, the simulation might simulate the distribution of leg length in a population of rabbits over many generations in response to the number of predators in the environment. The user can change the environmental attributes in real time and watch the animal's trait distribution respond to the change. The simulation is agent-based and involves a single type of animal agent. Each animal in the simulation has some calculated chance of making offspring which is based on their evolutionary fitness level in the current environment.

The goal of the simulation is not only to demonstrate that the distribution of traits changes over time, but to convey that individual animals do not undergo any change in phenotype. Rather, it is the reproductive biases that cause the changes in trait distribution over time. To reinforce this educational goal, I also included a tagging feature in the simulation that allows students to "tag" individual animals in a given timestep of the simulation, and observe the cause of death of these individual and whether they reproduced.

I will first describe the structure of the underlying model and how the simulation runs in each timestep. I will then go over the various components of the UI, including the tagging feature. Throughout the discussion, I will explain the rationale behind various design decisions.



The Evolution Simulation

6.1 The Model

In the evolutionary simulation, the ecosystem is modeled by a collection of agents representing a single population of animals. Each agent occupies a space in the ecosystem and possesses several attributes with variable values. The simulation itself also has a number of parameters such as environmental attributes.

6.1.1 Simulation Parameters

There are a number of parameters which pertain to the simulation as a whole. They are:

- **Animal type** - The name of the animal that is being simulated in a particular run. The evolutionary simulation focuses on the distribution of various traits in a single animal over time.
- **Environmental attributes** - The various environmental attributes of the simulation. These are attributes like temperature and rainfall. Each attribute is modeled as a number between 0 and 1. The user can adjust these values in real-time as the simulation is running by moving sliders in the UI.
- **Number of animals** - The number of animal agents that the environment simulates. Our prototype maintains a constant number of animals throughout the simulation. It wouldn't be difficult to extend the simulation to keep track of the populations size (for example, if the environment changes and many rabbits are suddenly unfit and die, the population goes down until evolution allows the majority of rabbits to deal with the new environment), but I chose to exclude this factor for simplicity, as the fact that the population size will fluctuate does not directly align with our primary educational goals. Thus, throughout the generations, the number of animal agents is constant but the distribution of various traits changes. In our prototype, the number of animals is 500.
- **Diversity per generation** - This is the proportion of newly generated animal objects in each generation which have completely random attributes. This allows us to maintain the genetic diversity of the animals in the simulation and ensure that the simulation does not break down after a few hundred generations. (see the Reproduction section below). Our current value for this is 0.1 (so one-tenth of all newly generated animal objects have completely random attributes). Animal objects

are only newly generated to replace animals in the previous generation that failed to reproduce, since animal objects that were able to reproduce silently carry over and represent their children in the following generation. Therefore, the actual proportion of animal objects with completely random attributes in each generation is much lower, since the animals that carry over obviously do not have random attributes. This is discussed further below.

- **Maximum fitness level** - This is the maximum fitness level that any animal can have. This value should be less than 1. The reason for this is that in our simulation, we don't actually kill off animals that are able to reproduce, but use them in place of their offspring in following generations for efficiency reasons (see Reproduction). If every animal had a fitness level of 1, then no animals would die, and the distribution of traits would remain completely fixed over generations, which is unrealistic. To ensure that there is at least some small fluctuation of traits across generations even when there is no selecting force, we use a maximum_fitness_value of 0.99 in our prototype.
- **Starting distributions** - For each animal attribute, we store the initial distribution of the each animal trait as a string (i.e. "uniform", "right-skew") When we initialize the environment and populate it with animals, we generate each animal's traits randomly according to the specified distribution (for example, we have a random function that generates right-skewed values).

6.1.2 Animal Parameters

There are also some parameters that are particular to animals.

- **Attributes** - Similar to environmental attributes, each animal attribute is modeled as a number between 0 and 1.

For each animal attribute, we also maintain some additional parameters:

- **Maximum and minimum displayed values** - while I keep track of animal attributes as a number between 0 and 1, the user expects a more sensible value for animal attributes, for example, "4.5 inches" for leg length. I transform the raw attribute value (0 - 1) using this additional data before displaying it. For example, for leg length, the max and min displayed values might be 3 inches and 6 inches. A raw attribute value of 0.5 would be translated to 4.5 inches.
- **Biases** - for each animal attribute, this stores type of relationship the attribute has with various environmental attributes. For example, one entry might be ("temperature" => forwardBias"). "forwardBias" corresponds to a fitness function that I use when calculating fitness. (see Calculating Fitness) Just as in the real-world, an attribute can have zero, one, or multiple relationships with different environmental attributes, allowing for very complex evolutionary interactions.

6.2 A Timestep in the Simulation

Every timestep (also known as a generation), the simulation performs the following actions:

- Calculates the fitness for each animal currently in the simulation. The fitness is a number between 0 and 1. (see Calculating Fitness)
- (Natural selection) For each animal, we remove the animal object from the simulation with probability equal to 1 - fitness. (this represents removing animals that aren't able to reproduce from the genetic pool)

- (Reproduction) The remaining animals then randomly "reproduce" (see Reproduction), creating new animals until we have again reached the environment's number_animals value.
- As part of the above process, we also add some organisms with random attributes according to the value of the diversity_per_generation parameter (see Reproduction).
- The animals then move around in the simulation.

6.3 Calculating Fitness

We will now describe how we calculate the fitness of an organism based on its individual attributes and the current environmental attributes in our simulation.
Each animal attribute can have relationships with several of the environmental attributes. (for example, the fitness of leg_length might depend on both predators and availability of food)

For each pair of animal attribute and environmental attribute, we specify a *fitness function F*, which takes as input an animal attribute and an environmental attribute and computes a *fitness value* for that pair. This fitness value represents the contribution towards the animal's fitness that this particular animal attribute makes (given the current environmental attributes). A fitness value of "1" means perfect fitness, and a fitness value of "0" means no fitness (meaning the animal is completely unviable)

Here is an example fitness function reverseBias:

```
func reverseBias(envAttr, animalAttr)
```

```
        return Math.abs(envAttr - animalAttr)
    // When envAttr is 0, higher values of animalAttr are more fit.
    // When envAttr is 1, lower values of animalAttr are more fit.
    // When envAttr is 0.5, the maximum possible fitness is 0.5.
```

I provide the fitness functions I used for my prototype in Appendix B.

To calculate the fitness for an animal, we compute the fitness value for all possible pairs of envAttr and animalAttr and then multiply all the fitness values together with the maximum_fitness_value. For example, if the maximum fitness value is 0.9 and the fitness values for the animal attributes leg_length and ear_length and the environmental attributes num_predators and temperature taken pairwise are 1, 1, 0.5, and 0.8, then the final fitness value is 0.9 * 1 * 1 * 0.5 * 0.8 = 0.36.

The fitness value is the chance that the animal will be removed from the simulation. If the fitness values for the population are *lower* in general when environmental changes occur, then more animals will die each generation and "evolution" will occur faster in the simulation (and vice versa). Thus, to change the speed of evolution in the simulation, one can raise or lower the range of values that are returned from the fitness functions. For example, by dividing all fitness values by 2 in the fitness function, one can substantially increase the speed of evolution.

Using this system, any evolutionary relationship between animal attributes and environmental attributes can be modeled, including how strong the relationship is and thus how fast evolution takes place. The fitness functions are a little complicated, but since the aim of the simulation is not to be perfectly precise but rather to convey certain concepts to

students, it should be sufficient to simply use the noBias, forwardBias, and reverseBias functions provided in Appendix B for the majority of purposes.

6.4 The Reproduction Phase

In the simulation, every animal is supposed to live only one generation, with their offspring comprising the following generation. However, the simulation runs significantly slower if we have to destroy and re-make 500 animals in every timestep. For efficiency, we only remove animals that were unable to reproduce from the simulation (which we determine using their calculated fitness level), and recycle the other animal agents in the next generation. In practice, this works without a hitch.

Thus, during the reproduction phase, we simply need to generate animals to replace all the animals that were removed due to lack of fitness.

The reproduction phase is comprised of the following steps:

- First, we take the number of animals we need to generate (which is equal to the number of animals that were removed)
- We multiply this number by diversity_per_generation to get the number of "mutated" animals we need to generate. These are animals with completely random attributes. This part of the process simulates both mutations and the dominant-recessive nature of many traits, which causes some recessive traits to be buried within an animal's genes but re-surface generations later. It also ensures that, for example, even when 99% of the animals have very short fur, there will still be a couple animals generated every generation with long fur (so that in case the

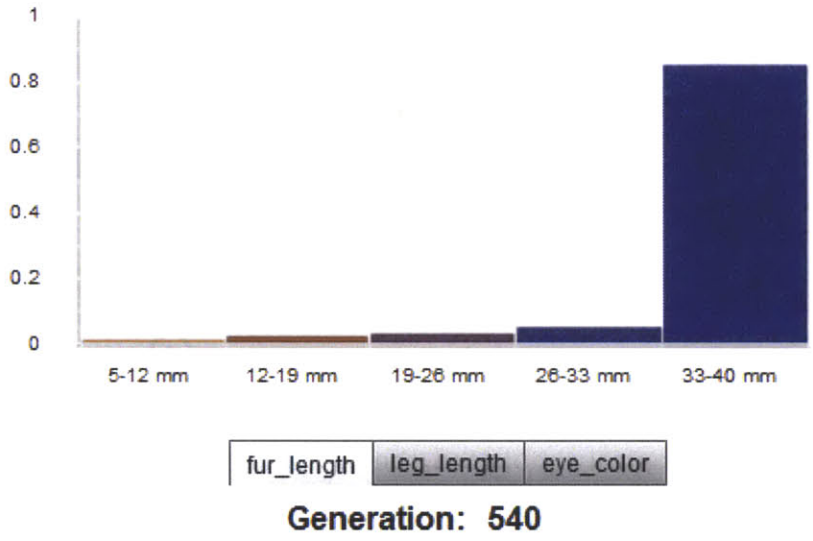
environment changes to favor longer fur, there will still be some long furred animals around).

Currently, 10% of the newly generated animals have randomized attributes, which initially appears like a very large number of mutations. As stated above, the newly generated animals only comprise a fraction of each generation, so the actual mutation rate is much lower. Furthermore, the vast majority of these animals actually represent animals with recessive phenotypes, rather than animals with spontaneous mutations.

- After generating the random animals, we generate the rest of the new animals by repeatedly picking two animals from the previous generation (not including the newly generated ones), and combining their attributes to form a new animal. For each attribute of the new animal, we randomly pick the corresponding attribute of one of its two parents, so that the new animal's attributes are a mix of its parents. We randomly combine the attributes of the trait parents instead of simply cloning one of the two parents because if we had simply cloned one of the two parents, then the number of distinct organisms in our simulation is reduced. Despite the addition of mutated organisms, this reduction in distinct organisms over many generations causes different animal attributes to be linked in undesirable ways. This is because the animal pool will eventually consist of only a few distinct organisms, and if these organisms all happen to have long legs and short fur, then long legs and short fur will be linked. The consequence of this is that when short fur is selected for, long legs will also increase inexplicably, which is undesirable. We also couldn't average the attribute values of the two parents because this averaging effect creates a

normal distribution for traits. Normal distribution curves make it less obvious to students which traits are being selected for since they all look pretty similar, which goes against our goals.

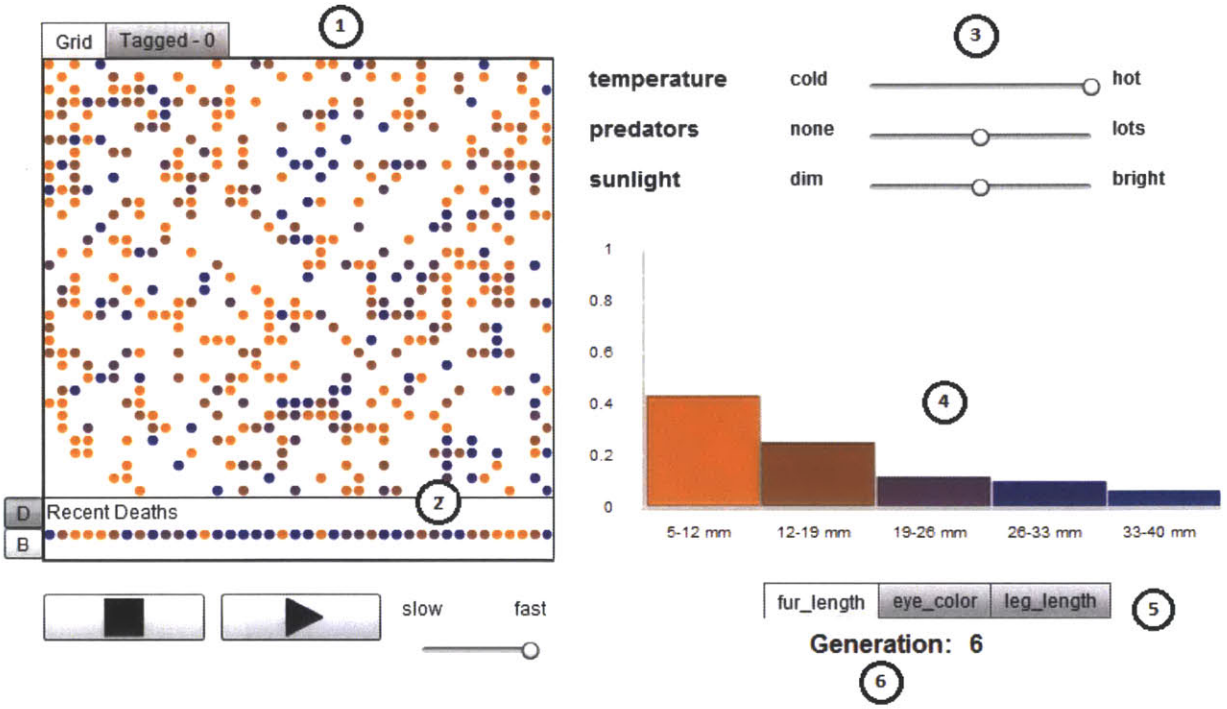
- At this point, the number of animals is again equal to the environment's number_animals value, and the reproduction phase is complete.



Even after hundreds of generations under a very cold environment, there are still a tiny number of organisms with short fur every generation due to the diversity_per_generation mechanism.

6.5 User Interface Highlights

We will now briefly highlight some of the elements of the user interface design for my evolution simulation prototype.



1) Similar to the ecological simulation, the evolutionary simulation features a colorful display of current state of the ecosystem. The dots are colored based on the value of the currently selected attribute of each particular animal, where the attribute is selected in element 5. As before the animals move around on the board.

2) The interfaces features a "birth feed" and a "death feed", which display the most recent animals born and deceased in the simulation. These feeds were intended to allow students to draw richer inferences by revealing which animals were recently born and died. While this can also be inferred from the graph, the graph does not make explicit the fact that certain animals are dying more frequently than others. So students could, for example, lower the temperature of the environment and see that a lot of shorter fur bunnies were dying. However, the feeds were ultimately ineffective because deaths did not always reflect an evolutionary bias. For example, if temperatures fell and longer fur bunnies became

dominant, then there would eventually more deaths in the feed from longer fur bunnies simply because there were more of them, not due to any evolutionary pressure. We needed another way to convey that shifts in trait distributions were caused by individual bunnies with low fitness dying prematurely, which led to the Tagging System (see below)

3) Unlike in the ecological simulation, the student can use these sliders to change environmental attributes in real-time. This reflects the fact that environmental attributes change over time and allows the student to have a more active part in the simulation's evolution, something that was missing from the ecological simulation.

4) In the evolutionary simulation, we only display the graphs for one animal attribute at a time, because the animal attributes are independent and comparisons are not necessary. We only show the distribution at a single instant, rather than over time. You can click any bar to tag five animals from that bar's attribute range (see Tagging System)

5) Students can use these buttons to change the currently selected animal attribute.

6) The generation number is displayed here. The magnitude of the generation number is not particularly important. The main purpose of the generation number is to give a sense of how long various evolutionary shifts take place.

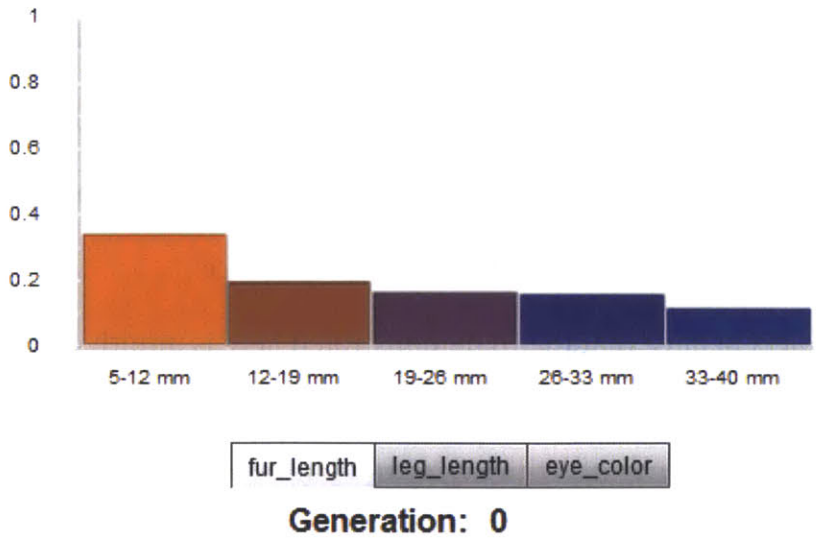
On the whole, I believe that the UI demonstrated here is intuitive, comprehensive, and visually appealing.

6.6 Designing Against Misconceptions

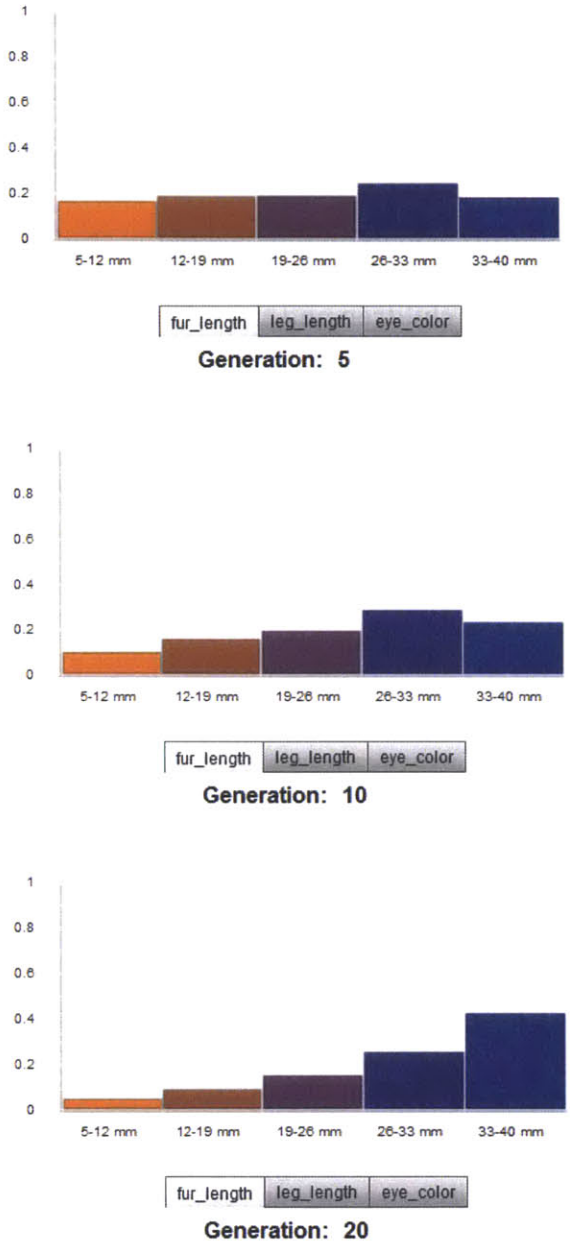
Whereas the overall concept of food chains is pretty straightforward, there are some common misconceptions about evolution that we wanted to ensure were not perpetuated

through our simulations. Two majors ones are: 1) evolution does not occur instantly but over many generations, and 2) individual animals do not evolve.

We wanted to keep the rate of evolution in the system relatively slow (at least ten generations for a major shift in the distribution) to show that evolution does not occur instantaneously. Our simulation roughly follows the evolutionary model of punctuated equilibrium, where the population of animals is in stasis for a majority of the time, with rapid evolution following a period of environmental disruption (represented by the student moving the slider to change the environmental attributes). However, this "rapid evolution" still takes many generations to occur and we were careful to reduce the rate of evolution to clear this misconception. Reducing the rate of the evolution was done simply by raising the values that were returned from the fitness functions, as discussed above.



Initial distribution for the simulation. At Generation 0, the environment suddenly becomes "cold", strongly favoring longer fur length.



After twenty generations, the longest fur animals dominate.

To make it clear that individual animals were not evolving, and that the shift in trait distribution was caused by unfit animals being less likely to reproduce over many generations, we first prototyped the birth and death feeds, which were intended to show that when distributions shifted, the individuals comprising the shrinking bars were more

likely to die than other individuals. However, as discussed above, this approach ran into problems. One way to improve the death feed was to limit the feed to only show animals that died of "unnatural" causes. However, we also realized that merely having a death feed still required a rather large intuitive leap on the part of the student, to infer from the deaths that these animals weren't having children and thus their traits were becoming less common within the population. We instead opted for the Tagging System, which is described in the following section.

6.7 The Tagging System

The tagging system allows the user to "tag" animals by clicking on them individually in the grid, or by clicking on the bars to tag organisms within a range. The user can then view the animals in a separate tab (which replaces the grid display). After stepping the simulation to the next timestep, the user can see each animal's cause of death, and how many children they had. The tagging system makes it appear as though all animals die in each generation, although we know that in fact, many animals are recycled. The information in the tagging system is fake, and generated with a few simple rules:

- If an animal was not removed due to the natural selection process, it "died of natural causes."
- Otherwise, we randomly pick which environmental attribute it died to, weighting attributes based on how badly they impacted fitness. For example, the animal might "die of overheating." These are collectively referred to as premature deaths.
- If a animal died of natural causes, it randomly has between 0 and 3 children (this number is made up on the spot and has no relation to anything in the simulation)

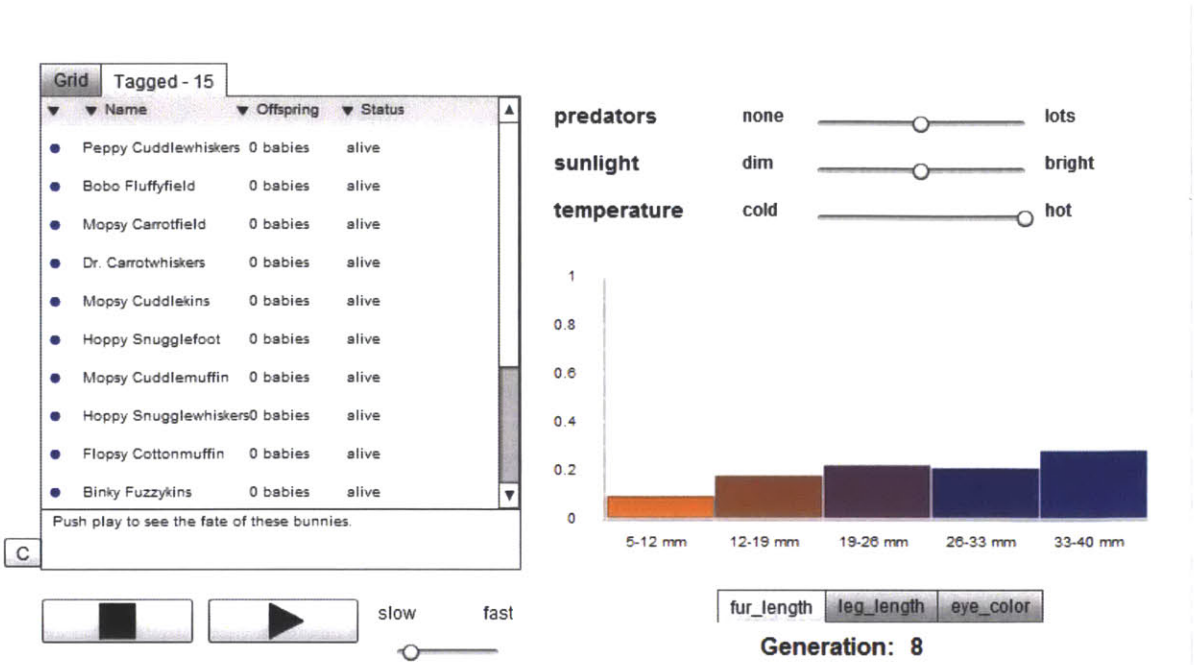
- If a animal died prematurely, it randomly has between 0 and 1 children.

The tagging system reinforces the concept that individual animals do not evolve, and that evolutionary shifts are caused by a disparity in reproduction patterns across animals of different attribute values. For example, when the environment is very hot, the student can tag ten bunnies with long fur and see that many of them died from overheating and had very few, if any, children. The student may then conclude that fewer long-furred bunnies are being born because their parents are dying before they can reproduce.

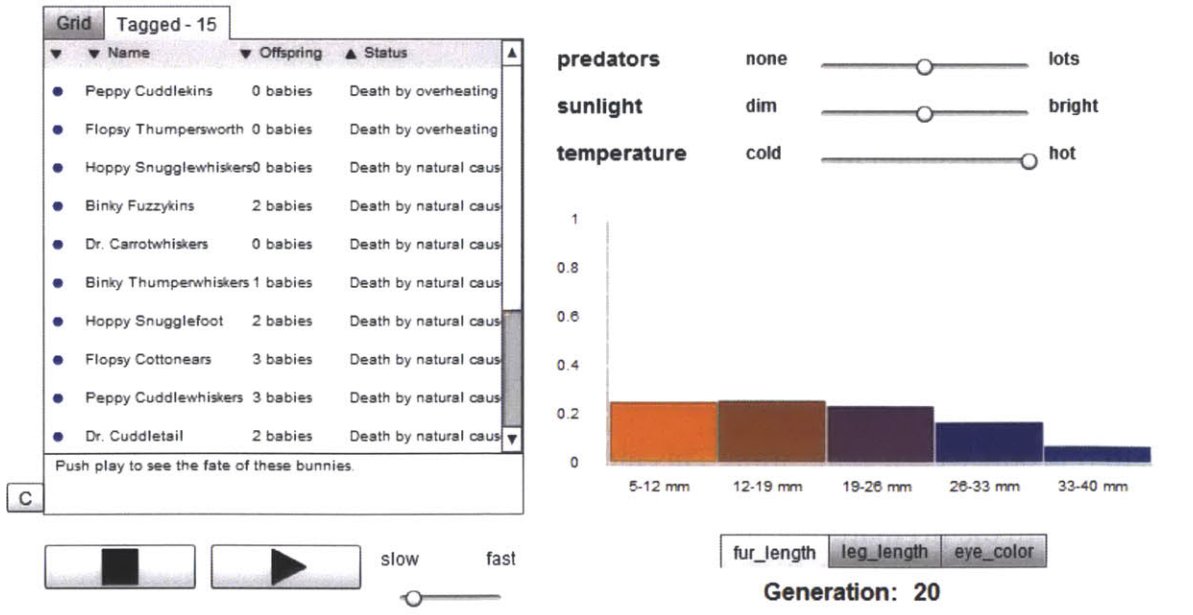
Note that even if an animal died prematurely, it can still have a child. We didn't want to perpetuate the misconception that it is impossible for animals with bad fitness to have children. However, we also limited the number the number of children that unfit bunnies could have to one to make sure it was clear that dying prematurely negatively impacted the number of offspring.

Furthermore, note that not all unfit animals die prematurely in the tagging system. Only the ones that failed to reproduce in the simulation die prematurely in the tagging system. This was designed to make it clear that that even though a portion of the population may be unfit in a new environment, that portion does not die out completely.

Finally, an effective tagging system turned out to be at odds with our desire to slow the rate of evolution. If we slow down the rate of evolution, then fewer unfit animals will be unable to reproduce each generation. This means that fewer animals in the tagging system will be shown to die prematurely, making the trends in the tagging system less obvious. We mitigated this somewhat by artificially inflating the number of premature deaths in the tagging system. We were careful not to inflate the number by too much (right now the inflation is 50%) as it would introduce a noticeable discrepancy in the simulation.



Tagging system right after tagging the animals



Tagging system after the simulation has been stepped. Notice that some animals have died of overheating and had no children.

6.8 Summary

In this section, I described the underlying model and highlighted the user interface for an evolutionary simulation that allows for a variety of complex evolutionary relationships between the attributes of a single animal population and the attributes of the environment. The model includes mechanisms that maintain genetic diversity and allow the simulation to run over a long period of time, giving the student ample opportunity to make inferences. The user interface allows the student to focus on only one attribute at a time, making for a cleaner experience that is also intuitive, comprehensive, and \visually appealing. To further our educational goals, I also implemented some experimental UI elements, including birth and death feeds, and a tagging system.

7. Evaluation

7.1 Overview

I had the privilege of playtesting my prototypes with students in Mr. Mark Knapp's class at Josiah Quincy High School and Amanda Tsoi's class at Somerville High School in Boston. My simulations were intended to help students solve quests, so in preparation for each playtesting session, I integrated my prototypes into an isometric game world that the Radix group was using for prototyping and play-testing, where players could travel around the map and accept quests from non-player-characters. I also designed and implemented some quests for the students to perform, with the help of my supervisor Louisa Rosenheck and her colleagues. The quests I used for playtesting will be discussed in the following section.

In each round of playtesting, we asked students to work through the quests for about an hour. During this session, we would walk around the room, asking the students questions about their progress and their experience. Common questions that we asked included:

- Can you explain what this particular component is displaying?
- How did you figure out the solution to this quest? Can you show me what you did?
- Can you explain why this phenomenon occurs? (examples: predators dying out, distributions for a particular trait changing in response to environmental changes)

I will now describe the quests that we presented to the students for both the ecology and evolution playtesting sessions, along with the feedback we received from each session as a result of observations and answers to the questions above. Unfortunately, I was not able to incorporate much of the feedback into my project during my thesis period, but I also give recommendations for future improvements to the simulations that are motivated by the feedback.

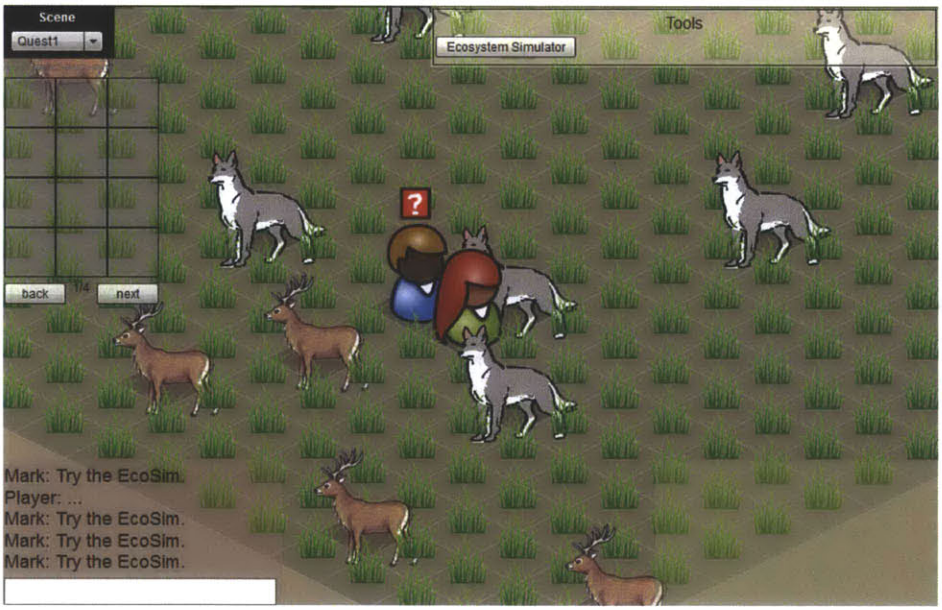
7.2 Ecology Quests

We designed three ecology quests in total for playtesting. The first quest gently introduces the student to the simulator, and the second and third quests require the student to use the simulator more extensively and make deeper inferences. We were able to integrate the game-world tightly into the quests, so that the game-world reflects the current quest and the responses given by the player.

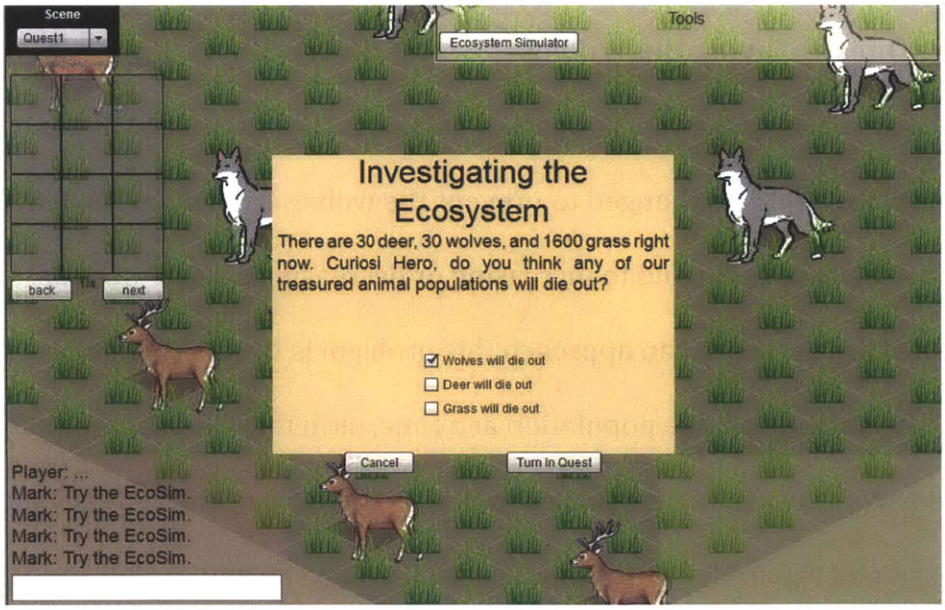
7.2.1 Ecology Quest - Investigating the Ecosystem

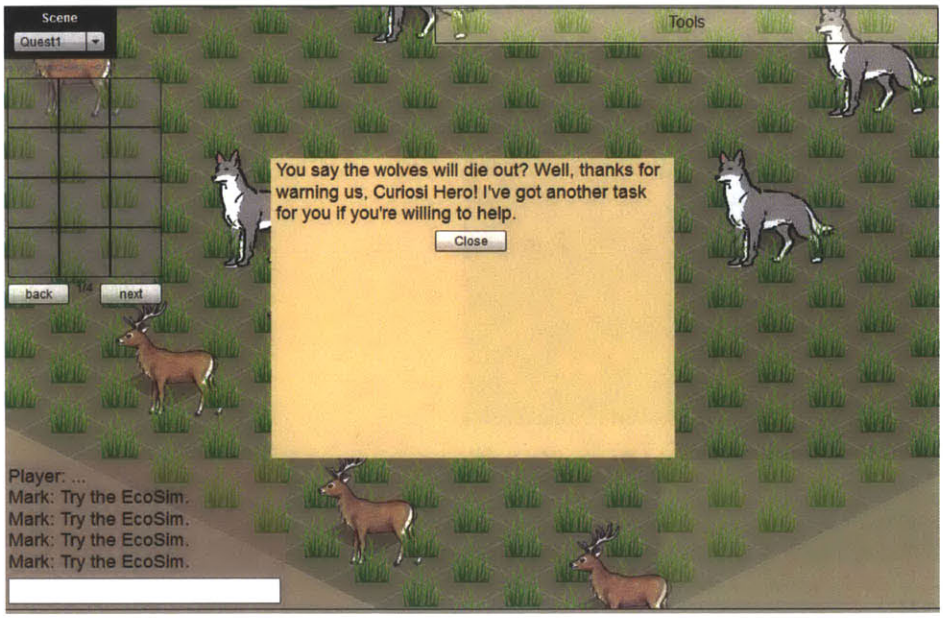
The first ecology quest, *Investigating the Ecosystem*, asked the student to investigate the effects of over-hunting on the local deer population. The quest-giver asks the student to select which populations (grass, deer, and wolf) are likely to die off based on the current population levels. Surprisingly, the wolves die out before the deer, because with reduced numbers of deer in the environment, the wolves are not able to catch enough deer to sustain themselves and die out faster than the deer. Without any predators, the deer are then able to recover. To discover this, the student merely needs to run the simulation with the appropriate starting conditions. As shown in the following screenshots, beating the quest merely requires correctly answering a multiple choice question.

Talking to the quest-giver initiates the Quest



After the quest has begun, the simulator tool can be opened in the top right corner.



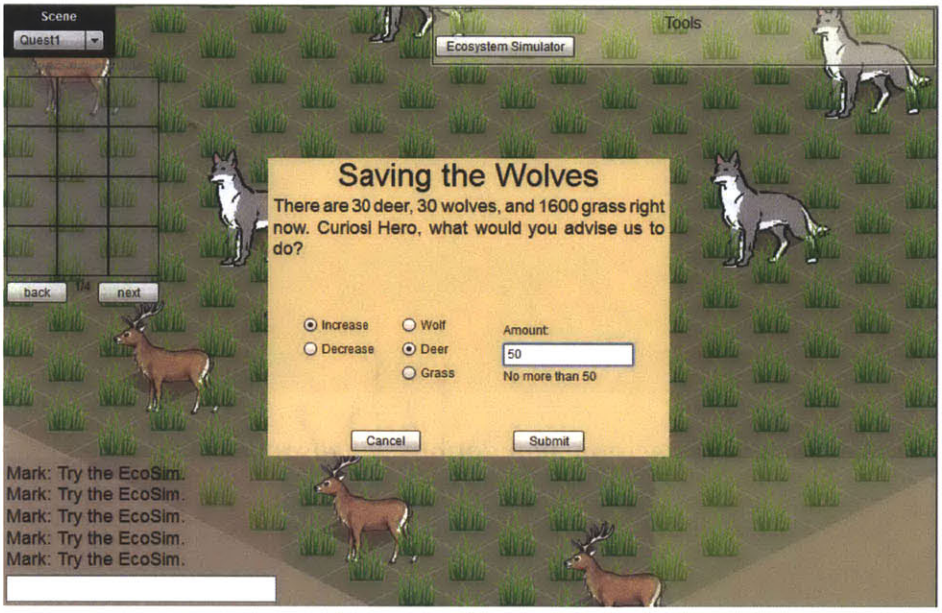


Talk to the quest-giver again to answer his question and complete the quest.

7.2.2 Ecology Quest - Saving the Wolves

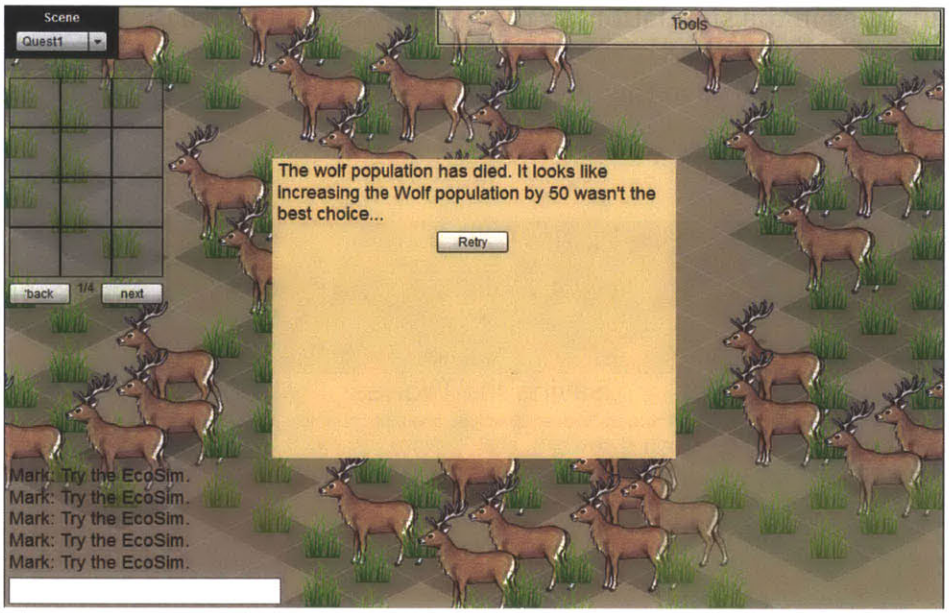
The second ecology quest, *Saving the Wolves*, asks the student to figure out how the current populations could be changed to prevent the wolves from dying. When answering the quest, the student is restricted to increasing or decreasing a single population of organisms. One way a student can approach this problem is by modifying the starting conditions of the simulation one population at a time, until the ecosystem is stable. After the student answers, a headless instance of our ecosystem (one with only the model and no UI) is initialized and run in the background with modified starting conditions based on the student's answer. If all populations are still alive after 500 timesteps, the student passes the quest. While we could have simply hard-coded the correct range of answers in this instance instead of running the simulation, this kind of approach would be useful for more complex ecosystems. Furthermore, the animals in the game-world are modified to reflect

the final populations in the simulation, so that if the wolf population dies, there will be no wolves after the student turns in the quest. This kind of tangible feedback makes the experience more immersive.



Students specify which population should be changed, and by how much

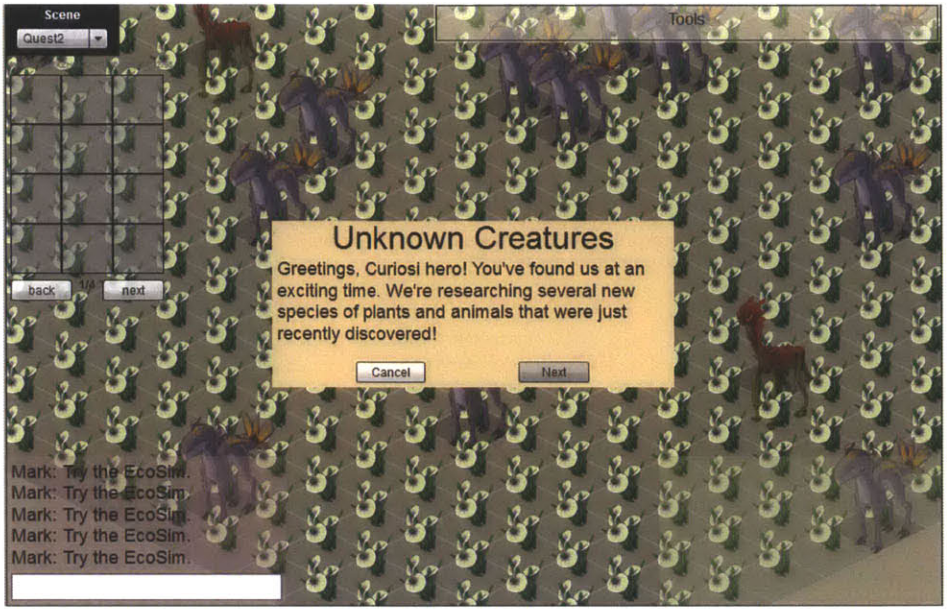
After the student makes their selection, a simulation is run in the background, and the game-world changes to reflect the result.

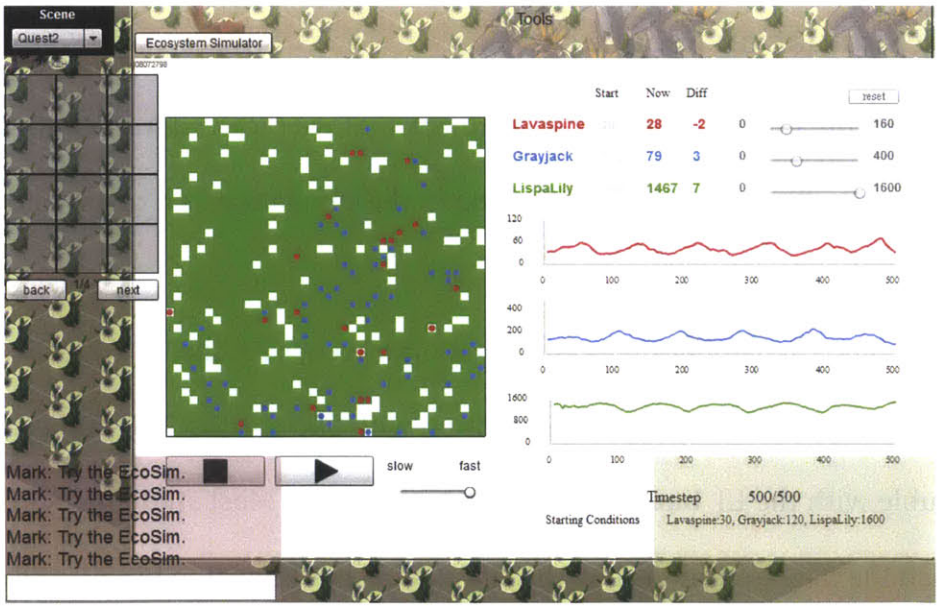


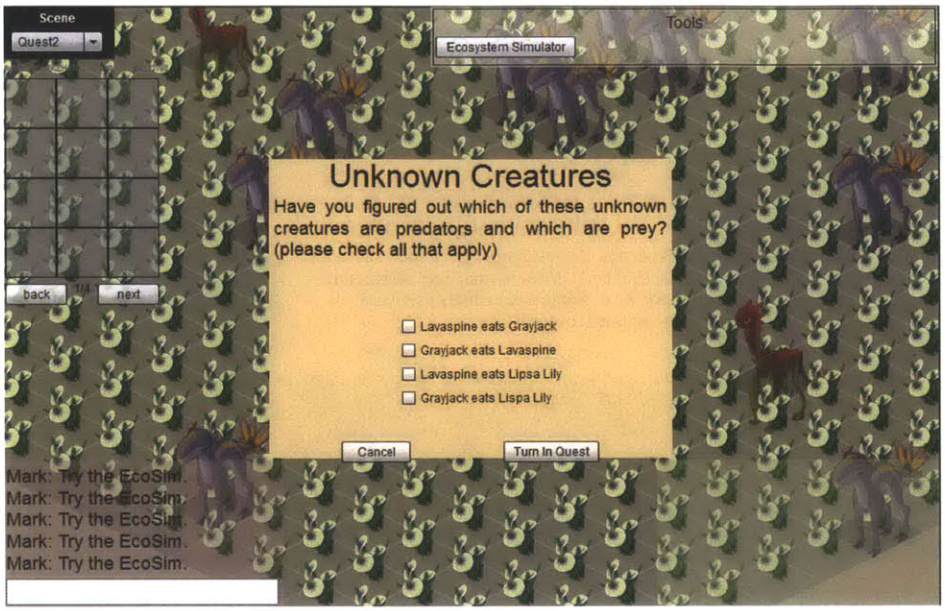
If the student makes the wrong decision, populations will disappear from the game-world. The student can reset the world by pressing "Retry"

7.2.3 Ecology Quest - Unknown Creatures

In the third ecology quest, *Unknown* Creatures, we provide the student with three populations of imaginary animals, and ask them to identify the predator-prey relationships between these populations. One way to approach this quest is to observe which animal populations go down when others go up. Some students were also able to solve the quest by using the relative sizes of the populations after realizing that prey were more plentiful than predators. We found this quest to be more challenging than the previous two.


Scene
Quest2
Tools
back
1/4
next
Unknown Creatures
Greetings, Curiosi hero! You've found us at an exciting time. We're researching several new species of plants and animals that were just recently discovered!
Cancel
Next
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.


Scene
Quest2
Tools
Ecosystem Simulator
back
1/4
next
Start
Now
Diff
reset
Lavaspine
28
-2
0
160
Grayjack
79
3
0
400
LispaLily
1467
7
0
1600
120
60
0
0
100
200
300
400
500
400
200
0
0
100
200
300
400
500
1600
800
0
0
100
200
300
400
500
slow
fast
Timestep
500/500
Starting Conditions
Lavaspine:30, Grayjack:120, LispaLily:1600
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.
Mark: Try the EcoSim.



Screenshots of the Unknown Creatures quest

7.3 Ecology Simulation Feedback

In this section, I will go over major feedback that we received from students during play-testing, and the changes that I recommend to the simulation as a result of this feedback.

The first piece of feedback was about the intuitiveness of the UI, which was one of our major goals. We tested the ecology simulation with two classes. The first class didn't have any trouble with the UI, but the second class struggled. Most students figured out that you could push the play button to start the simulation, but they didn't realize that you could move the sliders to change the starting conditions of the simulation. Emphasizing the sliders using colors or placement would alleviate this, and some written instructions would help as well. Additionally, providing a title or label to the graph would help to make their purpose more clear.

Secondly, we realized that many students were reluctant or unwilling to use the ecology simulator, and would instead repeatedly guess at the multiple choice questions until they got it right. This presents a big problem because students that "beat" the quest in this manner have learned nothing. At the very least, we need to include a flag that checks whether the student opened and ran the simulator before even allowing them to attempt the question. Detecting whether the student actually performed the actions in the simulation needed to reasonably solve the quest would be more ideal but also difficult and tedious. Another way of preventing the students from repeatedly guessing is to prevent the student from submitting more than one guess every minute.

A third thing we noticed was that some students would make multiple changes to the simulation at once. The quest *Saving the Wolves* only allows the player to affect one population, yet the simulation does not impose this restriction. Making changes to multiple populations at once also makes it harder to discover patterns. One option might be to automatically reset all the populations whenever the student moves a different slider in certain quests.

A final, more positive behavior that we noticed was that some students became very engaged with the simulation. A few students experimented with the simulation by their own volition after completing the quests, while others enthusiastically cheered on the animals in the simulation as it ran, saying things like "oh my god, the wolves are dying out." It is very promising to see the students engaged with the simulation even in this relatively unpolished form, and I think that additional visual polish, such as making the dots into actual animal icons, would make the simulation even more riveting. One particular visual embellish I think would appealing to students is to make the appropriate graph flash red

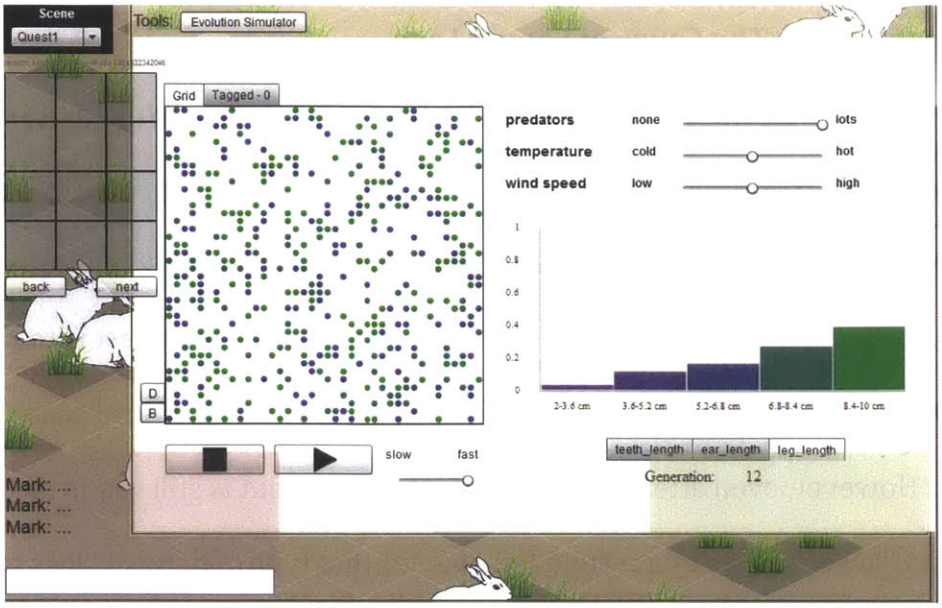
whenever a particular populations dips to a very dangerous point, which would give the simulation more intensity.

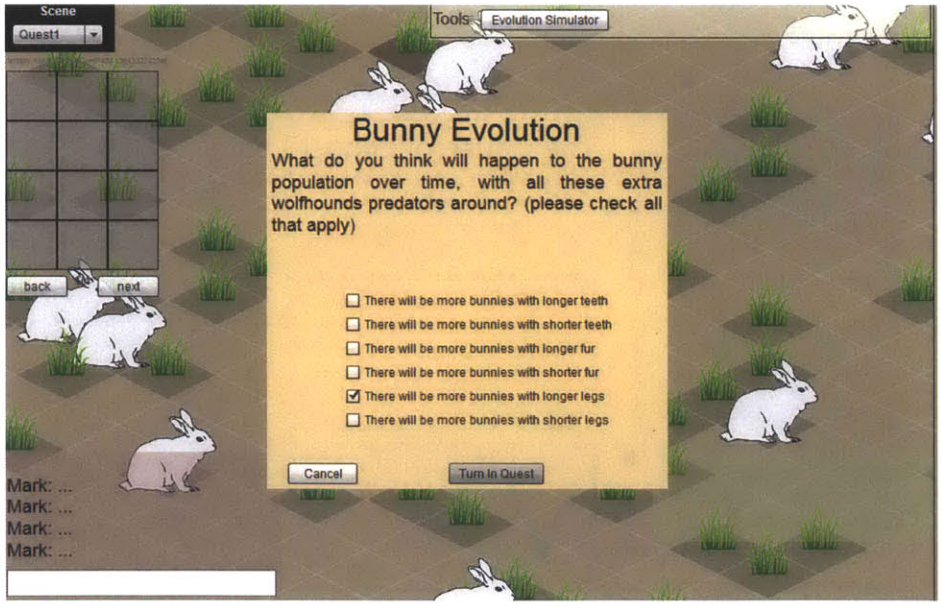
7.4 Evolution Quests

For playtesting, we designed four evolution quests in total. As with the ecology quests, the first quest introduces the player to the simulation, and subsequent quests require students to use the simulator more extensively to make deeper inferences. Unlike the ecology quests, the quests were not very integrated into the game-world during play-testing and were more abstract. For example, the *Bunny Cave Paintings* quest asks players to answer questions about the ancient past, based on cave paintings which are conveyed through the narrative. The four quests are very similar in format and all essentially ask the student to play around with the simulator and answer a multiple-choice question, which is somewhat repetitive. Overall, the evolution quests are more rudimentary than the ecology quests. Both the simulation and the quests will need further revisions in order to better complement each other, as the feedback shows.

7.4.1 Evolution Quest - Bunny Evolution

The first evolution quest that we designed for playtesting, *Bunny Evolution*, asked the player to figure out what will happen to a rabbit population after the number of predators suddenly increases. To solve the quest, the student simply needs to switch the predator slider in the evolution simulation to "lots" and observe that the distribution of leg length in the bunny population shifts towards longer legs.

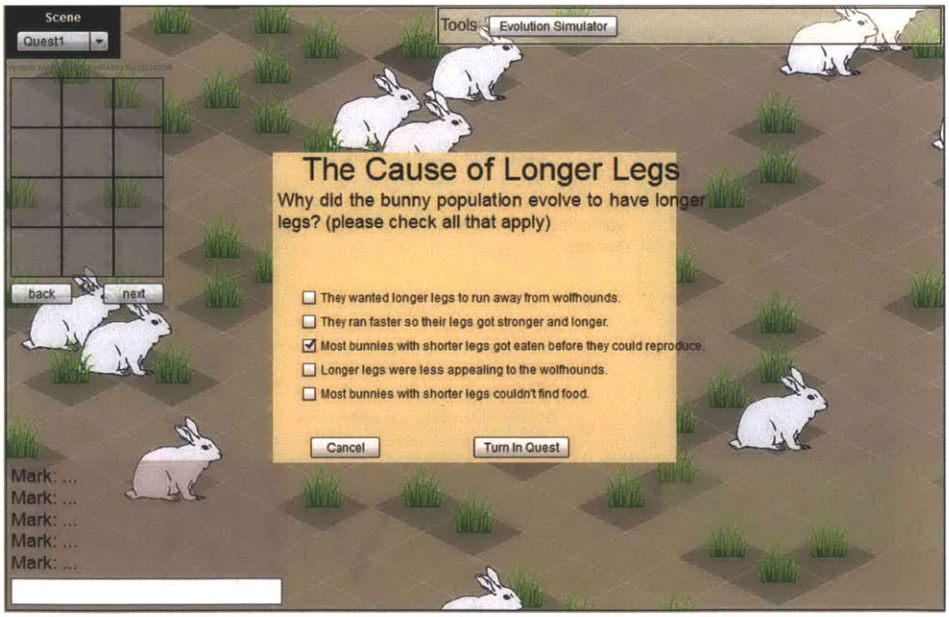




Screenshots of the evolution sim and the turn-in question for the Bunny Evolution quest

7.4.2 Evolution Quest - The Cause of Longer Legs

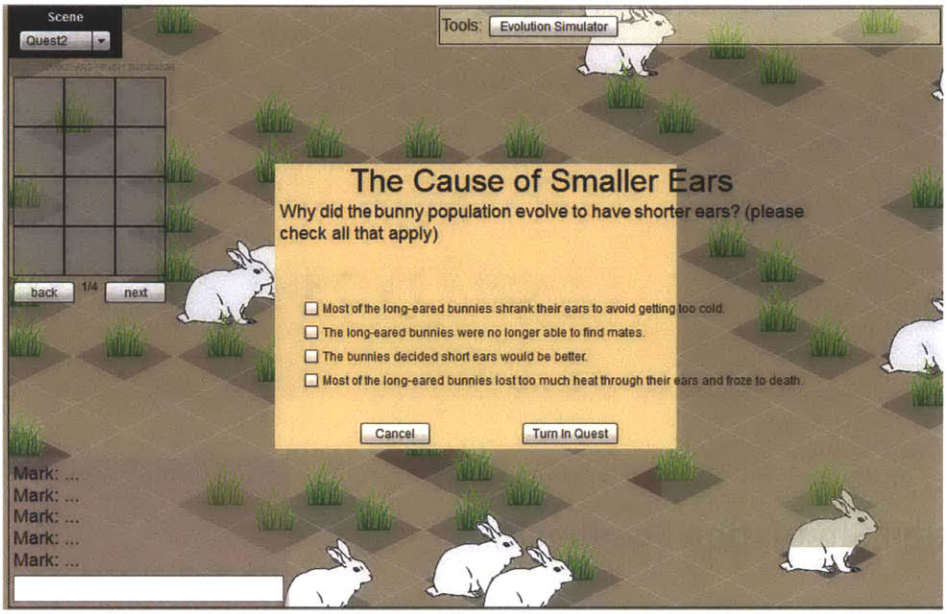
The second quest, *The Cause of Longer Legs*, asks students to explain the cause behind the shifting distribution of longer legs. This quest was added primarily to test whether the simulation was effective at conveying that the shifting distribution was caused by unfit animals being less likely to reproduce. One way to solve the quest is to play with the tagging system and observe that unfit animals that die from unnatural causes have fewer babies. However, even after this observation, the student is still required to make the logical leap that fewer babies of a certain type means that type of bunny is less represented in the population, and that this causes the shifting distribution. We found that students had a lot of trouble with this leap



7.4.3 Evolution Quest - Bunny Cave Paintings

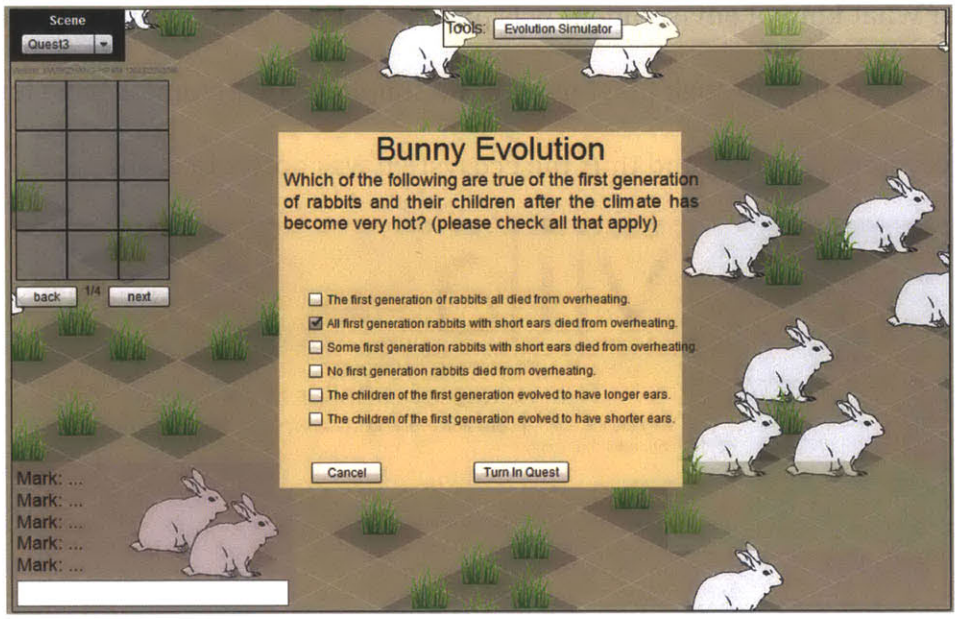
In the third quest, *Bunny Cave Paintings*, the students are informed that cave paintings of ancient bunnies have been found that have much bigger ears. The students are

asked to infer what kind of environment was present during that time. To solve this quest, the student must realize through playing with the simulator that bunnies have longer ears when the ecosystem is hotter, and thus the ecosystem was hotter in earlier ages.



7.4.4 Evolution Quest - Bunny Evolution 2

Finally, in the fourth quest, *Bunny Evolution 2*, we asked the students what immediate effects a hotter ecosystem would have on the bunny population. The purpose of this quest was for the students to realize that the effects of evolution manifest over multiple generations, not within a single generation. Within a single generation, however, many bunnies with short ears, who are not fit for a hot ecosystem, will die from overheating without being able to reproduce.



7.5 Evolution Simulation Feedback

The first piece of feedback that we received was that the UI for the evolution simulation was less intuitive than that of the ecology simulation. One issue was that if a student only moved the slider a tiny bit, there is very little (and sometimes no) difference in the simulation. One solution to this is to give the sliders discrete increments. The students also found the meaning of the histogram of attributes to be unclear. In hindsight, the graph needs to be better labeled. Perhaps a bunny silhouette, or multiple silhouettes with variations on the selected trait (i.e. different lengths ears) would make the simulation clearer.

The tagging system was also a source of confusion. I think the fundamental issue is obscurity of purpose. Students would generally start by tagging rabbits at random. They would notice nothing interesting about the results of the tagging, perhaps try a couple more times, and then lose interest. The problem is that it is very difficult to figure out how

to get significant results from the tagging system, and the UI needs to be reworked to make this easier. The quest might also explicitly ask the student to tag rabbits of a certain type. As with the ecology simulation, we noticed that students often moved sliders in combinations, making it difficult for them to discern patterns. And as before, students often preferred to skip the simulation altogether if possible.

The biggest issue was that the simulation did not break down student misconceptions about evolution, as the feedback from the *Cause of Longer Legs* quest demonstrated. While students were able to make the connection that environmental changes caused certain kinds of rabbits to become more common, they believed that bunnies "wanted" a particular trait, and that they could pass traits that they acquired during a particular generation to their offspring. I think the way forward here is to design a sequence of quests that specifically targets these misconception. We might augment the tagging system to show that at the time of death, the attributes of the bunnies did not change. We might also drill down on each generation and graphically display the number of bunnies that were able to reproduce in each attribute category (i.e. short legs, medium legs, long legs), in order to convey that unfit bunnies had fewer offspring.

The concepts in the evolution module turned out to be much harder to grasp, and there was much more room for confusion and mistaken beliefs. While my evolution prototype is a big step towards dispelling these misconceptions, it was ultimately not enough to bridge the gap, and there are many improvements that we can make to increase its effectiveness.

8. Conclusion

Improved STEM education is critical to our nation's future, and MMO games offer an innovative and promising approach to this problem. My primary contributions in my thesis are the design and implementation of the underlying model and user interface for an ecology simulation prototype and an evolution simulation prototype to be used within the Radix Endeavor, an exciting MMO initiative at the MIT STEP Lab and Education Arcade. The simulations were designed to be flexible, so that they could be used for a variety of purposes and quests, and the user interfaces were designed to be intuitive, comprehensive, and visually appealing to students.

Based on the feedback that I received from our playtesting sessions with students at Josiah Quincy High School and Somerville High School, I was able to recommend several refinements to my simulations. In particular, while the ecology simulation appears solid, the evolution simulation will require careful reworking in order to effectively convey the educational goals that we have set. While I will not be continuing with this work, my source code and several write-ups, including this document, will be handed off to developers and designers at Filament Games production studio, where they will make additional refinements to my simulations and integrate them into the Radix game. It has been a great privilege to work on a piece of this endeavor, and I look forward to the upcoming Radix revolution.

Appendix A - Ecological Simulation Supplements

Here are values for a basic Secondary Consumer => Primary Consumer => Producer ecosystem that worked well in our simulation.

Wolf (Secondary Consumer):

- Reproduce_period: 45
- Brood_size: 1
- Minimum_hunt_rate: 0.2
- Prey_type: {Deer: 6} (this is the number of energy gained from consuming a deer)

Deer (Primary Consumer):

- Reproduce_period: 20
- Brood_size: 1
- Hunting_limit: 0.1
- Minimum_hunt_rate: 0.2
- Prey_type: {Grass: 2}

Grass (Producer)

- Reproduce_period: 12
- Brood_size: 2
- Hunting_limit: 0.75

Here are the values for the more complex ecosystem discussed in the paper:

[Screenshot of that ecosystem]

Hawk:

- Reproduce_period: 40

- Brood_size: 1
- Minimum_hunting_rate: 0.2
- Prey_type: {Rabbit: 9}

Wolf

- Reproduce_period: 45
- Brood_size: 1
- Minimum_hunting_rate: 0.2
- Prey_type: {Deer: 9, Rabbit: 9}

Deer:

- Hunting_limit: 0.08
- Reproduce_period: 20
- Brood_size: 1
- Hunting_limit: 0.08
- Minimum_hunt_rate: 0.2
- Prey_type: {Grass: 4}

Rabbit:

- Hunting_limit: 0.08
- Reproduce_period: 20
- Brood_size: 1
- Hunting_limit: 0.08
- Minimum_hunt_rate: 0.2

- Prey_type: {Grass: 3, Carrot: 3}

Grass:

- Hunting_limit: 0.85
- Reproduce_period: 12
- Brood_size: 1

Carrot:

- Hunting_limit: 0.85
- Reproduce_period: 12
- Brood_size: 1

Appendix B - Evolution Simulation Supplement

Here is the fitness function we used to model a relationship where a high envAttr selects for a high animalAttr and a low envAttr selects for a low animalAttr (for example, number of predators and leg_length)

```
static public function forwardBias(animalAttributeValue: Number, envAttributeValue:
Number) {
    if(envAttributeValue > 0.6) {
        // the bigger the envAttributeValue is, the more that the final fitness depends
    on the bias
        var base:Number = 0.5 + 0.5 * ( 1 - envAttributeValue) / 0.4;
```

```
            var bias:Number= (1 - base) * (Math.pow((animalAttributeValue), .2));
            return bias + base;
        }
        else if (envAttributeValue < 0.4){
            var base:Number = 0.5 + 0.5 * (envAttributeValue) / 0.4;
            var bias:Number = (1 - base) * (Math.pow(1-animalAttributeValue, .2));
            return bias + base;
        }
        // if envAttributeValue is between 0.4 and 0.6, no animals are selected against
        else {
            return 1;
        }
}
```

Here is the fitness function we used to model a relationship where a high envAttr selects for a low animalAttr and a low envAttr selects for a high animalAttr (for example, temperature and fur_length)

```
static public function reverseBias(animalAttributeValue: Number, envAttributeValue:
Number): Number {
        if(envAttributeValue > 0.6) {
            // the bigger the envAttributeValue is, the more that the final fitness depends
        on the bias
```

```
            var base:Number = 0.5 + 0.5 * ( 1 - envAttributeValue) / 0.4;
        var bias:Number= (1 - base) * (Math.pow((1-animalAttributeValue), .2));
            return bias + base;
        }
        else if (envAttributeValue < 0.4){
            var base:Number = 0.5 + 0.5 * (envAttributeValue) / 0.4;
            var bias:Number = (1 - base) * (Math.pow(animalAttributeValue, .2));
            return bias + base;
        }
        // if envAttributeValue is between 0.4 and 0.6, no animals are selected against
        else {
            return 1;
        }
}
```

Here is the fitness function we used to model a relationship where the envAttr and animalAttr do not affect each other:

```
static public function noBias(animalAttributeValue: Number, envAttributeValue:
Number): Number {
        return 1;
}
```

Bibliography

Achieve Inc. (2013). *The Need for New Science Standards.* Retrieved May 2013 from Achieve Inc: http://www.nextgenscience.org/overview-0

Adams, Wendy, Noah Podolefsky and Jonathon Olson. (2011). *Natural Selection.* Retrieved May 2013 from PhET: http://phet.colorado.edu/en/simulation/natural-selection

ExploreLearning. (2013). *Food Chain.* Retrieved May 2013 from ExploreLearning: http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx

Gurria, Angel. (December 2010). *Presentation of the PISA 2010 Results.* Retrieved May 2013 from OECD: http://www.oecd.org/unitedstates/presentationofthepisa2010results.htm

Habitable Planet, The. (2013). *Ecology Lab.* Retrieved May 2013 from The Habitable Planet: http://www.learner.org/courses/envsci/interactives/ecology/

Isee Systems. (2013). *Stella.* Retrieved May 2013 from Isee Systems: http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx

Klopfer, Eric, et al. (2009). *The Simulation Cycle: combining games, simulations, engineering and science using StarLogo TNG.* Oxford, UK: E-Learning and Digital Media, Vol. 6. No. 1.

Simbio. (2013). *EvoBeaker.* Retrieved May 2013 from Simbio: http://simbio.com/products-college/EvoBeaker

Weisstein, Eric W. (2013). *Lotka-Volterra Equations.* Retrieved May 2013 from *MathWorld*--A Wolfram Web Resource: http://mathworld.wolfram.com/Lotka-VolterraEquations.html