Building the Bio-CS Bridge: Expanding High School Curriculum that Integrates Biology and Computer Science

An Interactive Qualifying Project Proposal

Submitted to:

The Faculty of Worcester Polytechnic Institute

in Partial Fulfillment of

the Bachelor of Science Degree

Project Advisor: Doctor Elizabeth F. Ryder

Submitted by:

Jacob Bernard

Date Submitted: 28 May 2022

Abstract

Computer science teaching in high school must increase to meet the demand of graduates in the subject and should incorporate material from different subjects. The Bio-CS Bridge curriculum uses agent-based modeling of biological systems to address CS and Biology standards. Existing curriculum using the block-based language Starlogo Nova was expanded and improved, and new activities and simulations were added using the text-based language NetLogo. These resources will allow students with a variety of CS backgrounds to learn from the curriculum.

Table of Contents

1. Introduction

There is an increasing need for cross-disciplinary education at all levels. For computer science, it could be argued that it is even more important since there is rarely a computer science task that does not exist withing the context of one or more additional disciplines (CSforAll). Additionally, there is a need for biologists with competencies in disciplines like CS (Labov, Reid, & Yamamoto, 2017). Education at the high school level in the U.S. in particular can grow in these areas.

The Bio-CS Bridge project consists of a transdisciplinary group of university specialists, graduate and undergraduate students, and high school teachers and students (About the Bio-CS Bridge Project). This team is developing modular curriculum for high school biology and computer science classes. The Bio-CS Bridge project stems from a need to teach computer science concepts with real world contexts. Computing is a tool that can be used in many different disciplines and integration with biology concepts in a great example of this.

The CS unit one Bio-CS Bridge Curriculum uses agent-based modeling to simulate different biological systems. Agent-based modeling can simulate real phenomena by programming the actions of individual agents. By building and using simulations of biological systems, students learn the biology of individual organisms while learning computer science concepts. This project sought to further increase the reach and effectiveness of the current curriculum. The CS curriculum was expanded and improved to better suit a diverse range of students.

2. Literature Review

2.1. Teaching Computer Science

2.1.1. High School Computer Science Standards

Unlike English language and mathematics, there is currently no common CS curriculum among all states in the US. The Digital Literacy and Computer Science framework (DLCS), which is used as a basis for CS curriculum in Massachusetts, is organized into four strands, as seen in figure 1 (Massachusetts Department of Elementary and Secondary Education, 2016). A strand is a discipline within a general learning area (International Bureau of Education UNESCO). Each strand is further divided into topics that each comprise groups of standards. Practices are skills students should be able to use. These standards and practices were developed in conjunction with the Massachusetts Computing Attainment Network beginning in June of 2014. The development of the standards ended and formal adoption of the standards by the Board of Elementary and Secondary Education (BESE) began in June of 2016.

Learning Progression				
Grade Spans	**Strands**			
K-2 3-5 6-8 9-12	**Computing and Society [CAS]** a. Safety and Security b. Ethics and Laws c. Interpersonal and Societal Impact	**Digital Tools and Collaboration [DTC]** a. Digital Tools b. Collaboration and Communication c. Research	**Computing Systems [CS]** a. Computing Devices b. Human and Computer Partnerships c. Networks d. Services	**Computational Thinking [CT]** a. Abstraction b. Algorithms c. Data d. Programming and Development e. Modeling and Simulation
Practices				
Connecting, Creating, Abstracting, Analyzing, Communicating, Collaborating, Research				

Figure 1: The strands and practices of the Massachusetts Digital Literacy Framework (Massachusetts Department of Elementary and Secondary Education, 2016).

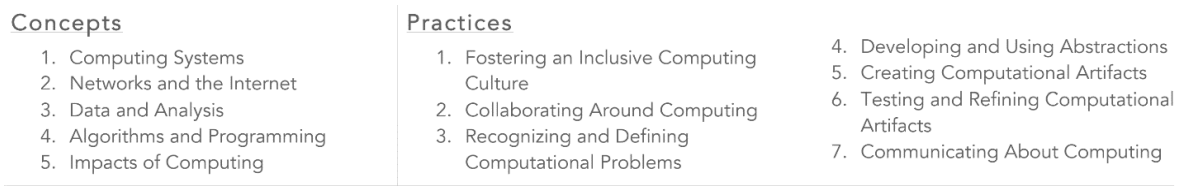


Figure 2: The CSTA concepts and practices (CSTA, CS Standards).

The Computer Science Teachers Association (CSTA) has created standards that many states use to develop their own, and which are slightly different from those of the Digital Literacy and Computer Science Framework. The Computer Science Teachers Association released their K-12 standards in 2017 (CSTA, About CSTA's K-12 Standards). The standards have 3 levels. Level 1 is for students between ages 5 and 11, level 2 for ages 11 through 14, and level 3A for ages 14 through 16. Level 3B covers ages 16 to 18 to finish the coverage of high school standards. The level 3A and 3B standards cover 5 main concepts: Computing Systems, Network and the Internet, Data and Analysis, Algorithms and Programming, and Impact of Computing (CSTA, CS Standards). The biggest difference between these standards is the organization and ordering, one example of which is the Computational Thinking strand in the DLCS framework, which lacks a direct comparison in the CSTA standards. Additionally, there is slightly more importance placed on inclusivity in the CSTA standards than the DLCS standards. Figure 2 shows the CSTA concepts and practices in more detail.

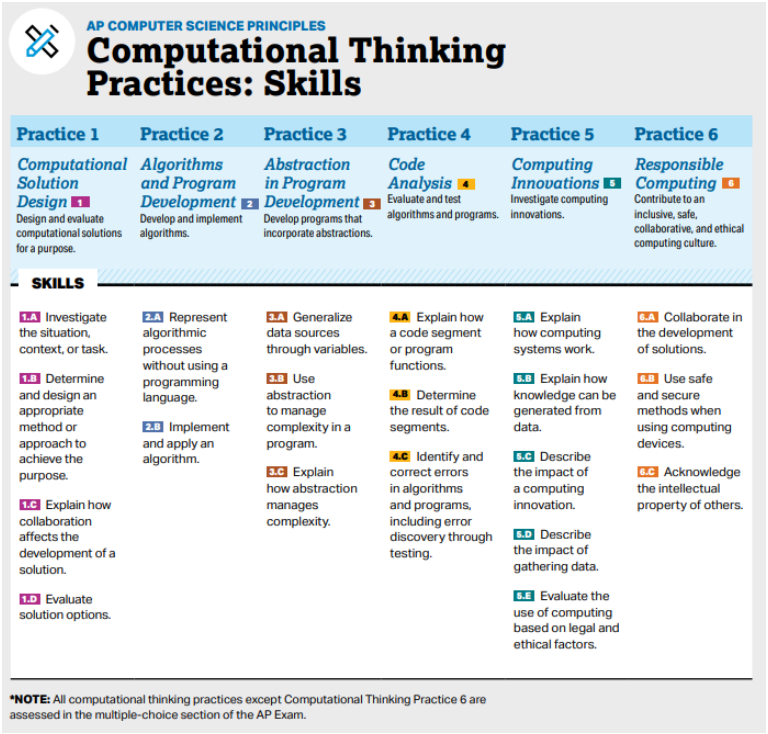


Figure 3: The Computational Thinking Practices for the AP Computer Science Principles exam (College Board, 2020).

One additional part of computer science education in high schools is AP curriculum. 39% of high school seniors in the US took at least one AP exam in 2018 (College Board). The AP Computer Science Principles course has a multiple-choice exam at the end of the course and a project-like task that must be completed during the course. There is significant overlap with the AP CSP standards and many of the standards found at the high school level. The Computational Thinking Practices that are found in the AP CSP class also correspond to and mirror the practices found in both the CSTA and Digital Literacy Framework standards.

2.1.2. Problems With CS Education

Computer science teaching first started in 1946 at Columbia University (IBM). Although that was over 70 years ago, computer science teaching in high schools continues to lag the overall surge in importance of computer science-related fields. For instance, in 2010, only 14 states had adopted computer science instructional standards that included more than 50% of the CSTA standards (Wilson, Sudol, Stephenson, & Stehlik, 2010). In contrast, 95% of the 2010 Carnegie Melon University computer science graduates had jobs waiting after graduation (Hoffmann, 2010). This problem still persists today with more jobs being created than students graduating with degrees in computer science (Bauer-Wolf, 2017).

In 2018 there were 16,001 bachelor's degrees in computer science awarded to women. During the same year, there were 64,270 awarded to men (National Center for Science and Engineering Statistics). This significant gap in representation is also observed in minority populations, with only 6,558 of the total 80,271 bachelor's degrees in computer science awarded to Black or African American students. Without specific guidelines of inclusivity as are included with the CSTA standards, it will be harder for schools to close these gaps. CSTA standard 2-IC-21 *Discuss issues of bias and accessibility in the design of existing technologies* is an example of an inclusivity standard that discusses how bias can be introduced into new technologies like facial recognition. There is not an equivalent standard in the DLCS framework; instead, bias is only discussed in the context of how information is presented in the context of research. Weaving concepts of equity into standards in this way allows teachers to include them in lessons more easily.

In addition to the already damaging problems with inclusivity, inequality with teaching computer science is further exacerbated by funding issues of high schools. High schools, especially those with high minority populations, can be forced to go into debt to purchase computers. For example, in 2008 the San Diego Unified School District sold bonds to support construction, maintenance, and technology needed to support teaching of computer science. The 21,000 iPads that were bought for around $400 each may end up costing the district more than $4,000 over the course of paying them off (Diallo, 2019).

Although there are problems with the current state of computer science education in the US, there are also strategies that have been implemented that are very effective in teaching computer science concepts. One particular idea that is increasingly important to teach computer science is how to collaborate and work effectively in a team (Barkataki, 2011). This practice is found in both the Digital Literacy Framework and the CSTA standards. Additionally, an increased focus on teaching computer science in the context of other ideas is becoming more relevant and needed (Cooper & Cunningham, 2010). Computer science and programming aren't practiced in a vacuum and therefore shouldn't be taught in one.

2.2. Teaching Biology

2.2.1. High School Biology Standards

The Next Generation Science Standards (NGSS) have been adopted by 20 states and 24 additional states have adopted their own standards based on the NGSS (National Science Teaching Association). The NGSS include Life Science, Earth and Space Science, and Physical Science standards. These standards cover a wide range of topics for grades K-12. Massachusetts has implemented its own version of the NGSS that has changes to better tailor the standards for Massachusetts students (Massachusetts Department of Elementary and Secondary Education, 2014).

2.2.2. Past and Present Biology Teaching

The NGSS were created based on the idea that science needs to be more inquiry-based with more outside concepts integrated rather than memorization of facts (National Science Teaching Association). At the college level, a report from the American Association for the Advancement of Science describes how biology education must show students how to work with other disciplines and should reflect the scientific practices used in biologists' careers, such as quantitative reasoning, interdisciplinarity, modeling, and simulation (AAAS, 2009), and these ideas are also present in the NGSS.

An additional way that biology learning is expanding is with the introduction of cooperative learning. Cooperative learning is contrasted with competitive learning which can be seen in high schools and colleges where students work individually, have classroom-wide goals and tasks, and are graded on a curve. Competitive learning pits students against each other instead of viewing other students as a valuable resource in their learning experience. Cooperative learning allows students to work with fewer classmates with more similar goals and to use each other's strengths to complete a project or task (Tanner, Chatman, & Allen, 2017). Cooperative learning is a valuable idea to incorporate into biology in particular because of the interaction between biologists and other specialists like computer scientists and physicists in solving problems in the real world. Without showing students how to work collaboratively in school environments, collaborating in the real world can prove challenging.

2.3. Combined Biology-CS Teaching

2.3.1. Pedagogy

Computational thinking is a core part of the integration of biology and computer science. Computational thinking is the process of solving problems in a way that lends itself to computing solutions. With computers being more and more commonly used as tools to solve problems that would otherwise be too complex, being able to effectively use a computer to process data and output valuable information about a problem is extremely relevant (Jona, Wilensky, Trouille, Horn, & Orton, 2014). Focusing on computational thinking when teaching biology and computer science concepts can help bridge the gap between computer science ideas and their real-world implementations (Weintrop, et al., 2016).

In addition to focusing on computational thinking to better ready students for post-secondary study or work with computer science, there are specific strategies that help underrepresented students learn computer science concepts. Pedagogy that includes 3 specific ideas was found to support engagement of underrepresented students. Pedagogy should demystify computer science by

connecting computer science to real life, demonstrate how computer science can address social issues impacting students' social lives and communities, and welcome student's perspectives and input in the problem-solving process (Ryoo, 2019). To make effective and inclusive curriculum, these three ideas should be followed.

2.3.2. Benefits

The first major benefit of integrating computer science concepts into biology curriculum is that biology is a required topic in most U.S. high schools. Computer science, as discussed above, is still being integrated into high school curriculum and is not likely to be required for graduation for most students in the near future. By integrating it into biology courses that are currently being taught, students can have begin having CS instruction before independent CS classes are set up. High schools with large minority populations have reduced access to computer science courses (Goode, 2007). By integrating CS standards into biology classes, high schools will be able to expand learning opportunities to minority students when they would otherwise be unable to. Additionally, allowing computer science classes to be elective-only may exacerbate the low participation women and minorities (Wilensky, Brady, & Horn, 2014).

Teaching computer science isn't limited to writing code. Research into a problem and analysis of the best solution are needed to use programming tools effectively. The process of integrating computing concepts into a biology classroom helps motivate students to take a more active role in learning (Garraway-Lashley, 2014). Computer science topics have been found to be enriched when taught within the context of other connected ideas (Cooper & Cunningham, 2010). In addition to the benefits of increasing the variety of CS topics that are available to students, teaching biology and CS together can be mutually reinforcing. For instance, building and using simulations to test hypotheses helps students to understand both biological systems and the CS used to model them. (Wilensky & Reisman, 2006). The Bio-CS Bridge Curriculum combines the following two standards: HS-LS2-1 Ecosystems: Interactions, Energy, and Dynamics: *Use mathematical and/or computational representations to support explanations of factors that affect carrying capacity of ecosystems at different scales* from the Next Generation Science Standards and 9-12.CT.e.1 *Create models and simulations to help formulate, test, and refine hypotheses* from the Massachusetts DLCS. Creating a simulation of an ecosystem that allows hypothesis testing satisfies both the biology and the CS standards. By including these additional topics, both biology and CS concepts are expanded upon and made more relevant.

2.3.3. Challenges

Finding computer science teachers who have the relevant background to discuss biological processes and biology teachers who are able to instruct in programming concepts may prove to be challenging. High schools in the U.S. are historically underfunded and if teachers with additional qualifications must be hired for new courses, this curriculum may not be realistic (Leachman, Masterson, & Figueroa, 2017). High schools may not be able to integrate the curriculum or may attempt to integrate it without sufficient teacher training, both of which would negatively impact students.

In addition to the general problems with expanding the curriculum in schools, there are many different ways in which schools may have trouble integrating computer science classes into the present school environment. One potential problem is the software required to run certain programs. Schools in the US have a wide range of devices, from Chromebooks and iPads to Mac and Windows

laptops, and software is not always available for these different operating systems. Chromebooks, which are best used with web-based software, have been increasingly found in high schools, with over 40 million Chromebooks used in education worldwide (Thurrott, 2020). To ensure students have equal access to computer science education, online materials, programs, and programming environments should be used whenever possible instead of software that must be installed on a machine. Since web-based programming environments can be used on any operating system with a web browser, they may be easier to implement in schools than downloaded programming environments with specific operating system requirements. The main concern that arises from moving to web-based programs is the need for high-speed and high-bandwidth internet. Although this could be a problem, higher quality internet is being found in more and more schools across the nation with 99% of schools in the US on scalable fiber optic connections (Education Super Highway, 2019)

2.4. Agent-based modeling

2.4.1. Overview

Computational modeling is a tool used by teachers, researchers, and other professionals to test theories on a variety of topics (The Center for Connected Learning and Computer-Based Modeling). Modeling can be seen as a quick and inexpensive way to validate theories before funding more expensive research on a subject. There are a variety of different ways that modeling can be done including mathematical and agent-based modeling (Bodine, Panoff, Voit, & Weisstein, 2020). Mathematical modeling can be done with differential equations to simulate changes in populations over time and space, such as the transmission of disease in an epidemic, or diffusion of a substance in a solvent. Agent-based modeling, in contrast, is done by programming individual agents to follow behavioral rules that allow them to act like their real-life counterparts in a simplified way. In addition to this being simpler to understand for a student with little mathematical or computer science background, individual variability, as is found in real biological systems, can be easily implemented. Students learning agent-based modeling find themselves thinking as the animal or organism they are attempting to model, which can be beneficial from an educational standpoint (Wilensky, Brady, & Horn, 2014).

Agent-based modeling allows students to ask questions like, “What would a wolf do when it needs to eat food?” and others that allow the student to put themselves in the agent’s footsteps. This thinking in particular is one reason why agent-based modeling has been shown to be an effective way to teach biology concepts within a computer science curriculum. This questioning thinking challenges students to research specific phenomena and abstract the biological rules necessary to create computational models (Wilensky & Reisman, 2006). Thus, students not only think deeply about a biological system, but they learn computer science practices such as abstraction and decomposition.

2.4.2. StarLogo Nova

StarLogo Nova is a block-based programming language that originated from Logo which was developed in the mid-1960s in the MIT Artificial Intelligence Laboratory. Logo introduced many concepts; the most notable was the “turtle” that can be controlled to move in an environment. StarLogo further expanded on this and introduced the idea of independent turtles that can interact with each other (LOGO FOUNDATION).

Starlogo Nova is one of the two agent-based platforms utilized in the Bio-CS Bridge curriculum. Block-based languages substitute the textual commands in traditional languages with "blocks" that can be combined in pre-determined ways to represent traditional structures. Figure 4 shows two examples of "blocks" of code in block-based languages.

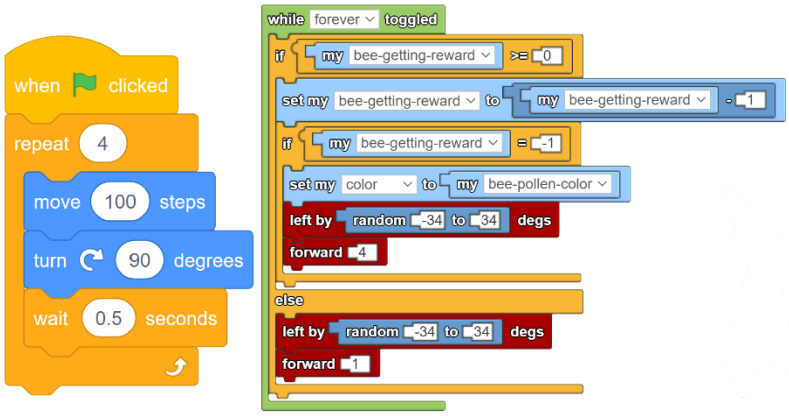


Figure 4: A "block" of code in scratch (left) and StarLogo Nova (right).

Block-based programming languages have been used as introductory programming languages in the past but have recently gained popularity in introductory high school courses. Block-based languages are perceived as "easier" than text-based ones by students and have additional features that students find helpful (Weintrop & Wilensky, 2015). One feature that is not as easily found in text-based programming languages is the library of components that users can scroll through to find what tools they need. Additionally, the limited nature of block-based languages stops many different syntax errors from being made. For example, figure 4 shows the StarLogo Nova code block with an if-statement on the right. A student can only put a conditional block in the correct spot of the if-block. This structuring can dramatically reduce the number of syntax errors and increase the time spent on learning concepts. While the format is thus easier for beginners, the language is complex enough to allow for the introduction of important computer science concepts, such as logic statements, variables, and structured code using procedures. Another advantage of StarLogo Nova is that the platform is web-based. This allows students who may not have home computers to access programs and to learn from school computers that have access to the internet. Additionally, since the platform is web-based, it is not necessary for the computers running StarLogo Nova to be as powerful.

2.4.3. NetLogo Language

The NetLogo language was created in 1999 by Uri Wilensky (Wilensky, NetLogo Frequently Asked Questions, 2021). One of the main differences between NetLogo and StarLogo Nova is the fact that NetLogo is a text-based programming language. NetLogo's text-based nature allows for more powerful simulations that could otherwise be limited by StarLogo Nova's block library (Wilensky, NetLogo). NetLogo has a downloaded platform and a web-based platform which provides flexibility to teachers who wish to use the language. Schools that have computers that can install Windows-based programs can use the more advanced downloadable version while schools with Chromebooks or computers that cannot have additional software installed can use the web version.

NetLogo has 3 main parts, the observer, turtles, and patches. The observer oversees the whole environment and can ask patches or turtles to perform commands. Turtles are individual agents that can perform commands or ask other turtles or patches to perform commands. Patches are squares that make up the background of the simulation environment and can ask other patches or turtles to perform commands as well (Wilensky, NetLogo). With all of these available interactions, models can be

created with the interactions between agents in mind that can help validate hypotheses in a controlled environment.

One often cited NetLogo model is the Wolf Sheep Predation model found in the biology section of the NetLogo model library (Wilensky, Wolf Sheep Predation). Seen in figure 5, this model simulates the interactions between sheep, wolves, and grass and can show how the populations fluctuate over time based on changes to the interactions. The model includes easy-to-use sliders for adjusting the amount of energy gained from eating for both sheep and wolves, the frequency at which they reproduce, and the time each patch of grass takes to regrow. By adjusting any of these properties, the model will behave differently, and different population outcomes will occur.

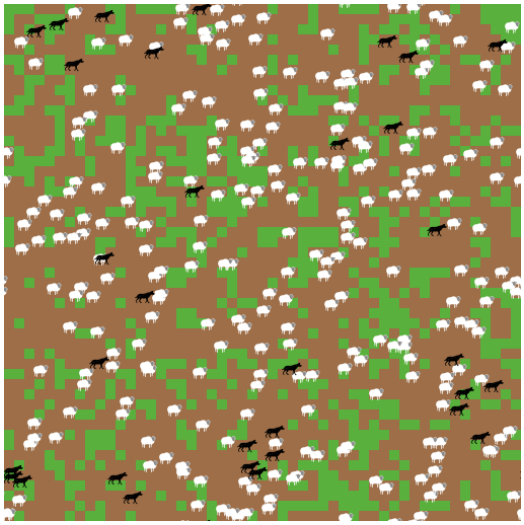


Figure 5: A screenshot of the interface of The Wolf Sheep Predation model from the NetLogo model library. This simulation includes the grass regrow element which helps to stabilize the populations of sheep and wolves. The green is grass, and the brown is where the grass was previously eaten. The black figures represent wolves, and the white are sheep.

2.4.4. Block vs. Text-Based Modeling

For the purposes of teaching agent-based modeling, there is no one language or system that is perfect in every way. Although advanced students may find block-based languages limit creativity, less advanced students may find text-based programming languages confusing to start with due to syntax errors that are common when learning (Plaza, et al., 2020). To combat this issue, the use of multiple languages could be used, following the same curriculum. Both NetLogo and StarLogo Nova have agents, some area that the agents are placed in, interactions between agents and a similar programming setup with buttons that can be set to run procedures created in the program. This can be seen in figure 6 with the two interfaces shown together. Because of the significant similarities, they are ideal candidates for the parallel teaching strategy that will be discussed further in the methodology section.

2.5. Bio-CS Bridge and Beecology Projects

The Bio-CS Bridge Project team develops, tests, and implements modular computer science and biology curricula that each incorporate aspects of the other to enrich learning (About the Bio-CS Bridge Project). As discussed previously, combining scientific practices with computational thinking ideas while teaching can more effectively teach students both ideas. The Bio-CS Bridge Curriculum is motivated by the Beecology Project, which is a citizen science project that aims to increase the ecological information available to researchers looking to address recent pollinator decline (Beecology Project). Part of the project are the simulations that are used to test new hypotheses that would otherwise take years to address. The simulations and data that the Beecology Project uses are used extensively in the Bio-CS Bridge project as a real-world context to the biology and computer science lessons.

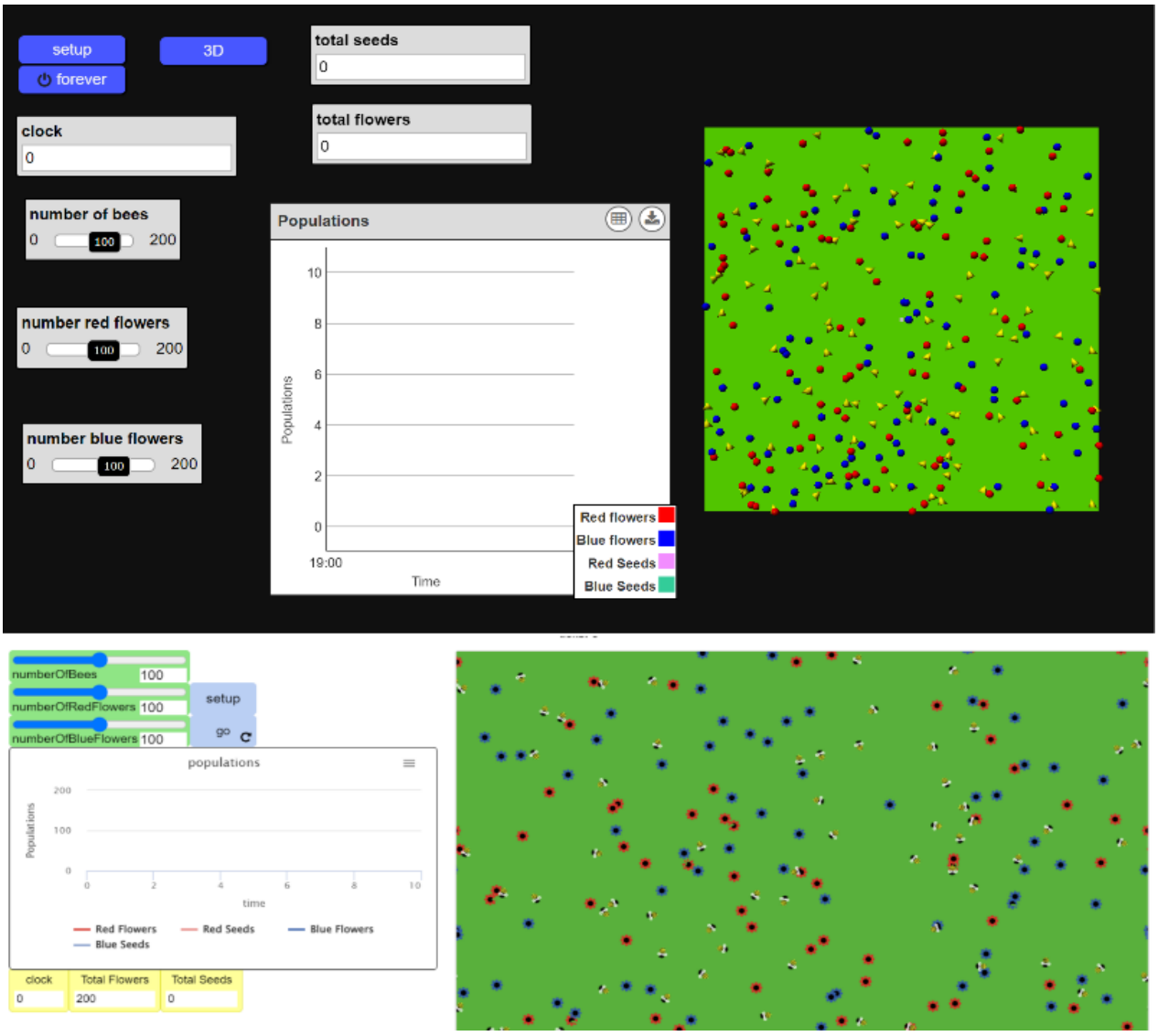


Figure 6: The interface for StarLogo Nova (top) and NetLogo (bottom). Both feature sliders that can control the simulation, displays that can show numbers while the simulation runs, and graphs.

3. Methodology

The three research questions that guided this project were:

1. **Which biology and computer science standards would align well with combined teaching?**
2. **How can existing curricula be improved to better integrate the two separate subjects?**
3. **How can existing curricula be expanded to better serve the diverse needs of students?**

These research questions guided the research of the project and the creation of resources for the Bio-CS Bridge CS curriculum. The first and second research question helped focus the project on the integration between the two subject areas. The third research question helped guide the expansion of the curriculum to better fit the diverse needs of students. Cumulatively, the questions helped focus the project on adapting the curriculum for use teaching many different students at different knowledge levels. The following sections illustrate the methodology followed to complete this.

3.1. Standards and Practices

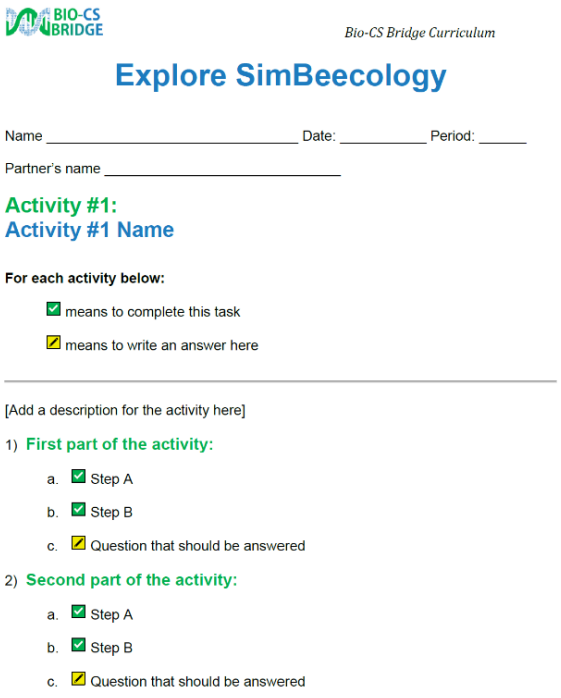
Following the research questions, biology and computer science standards were found that could be incorporated more heavily into the current curriculum. The new tutorials that were created followed the original standards and practices of the curriculum with one major exception. A new tutorial was created that guides the students through the creation of procedures to abstract the code and remove duplicate lines from the simulation. This tutorial covers standards 9-12.CT.a.1: *Discuss and give an example of the value of generalizing and decomposing aspects of a problem in order to solve it more effectively* and 9-12.CT.d.2: *Decompose a problem by defining functions, which accept parameters and produce return values*. Students were guided through how to make a singular function to make both the hawks and mice move in the SimBeecology simulation.

In addition to the creation of the procedure tutorial, the existing StarLogo Nova tutorials were recreated in NetLogo. The simulations had to be created in NetLogo, which will be explained below. After the simulations were created, the original style of the tutorials was followed. Students were given instructions on how to complete a task with pictures to aid when necessary. Whenever possible, students were guided to find the answer to problems on their own instead of providing the answers to copy into their own programs.

3.2. Improving Existing Curriculum

The existing SimBeecology simulation, written with the Starlogo Nova language, uses a food web to illustrate the importance of bees as a keystone species in an ecosystem. The bees pollinate the flowers in the simulation which, in turn, produce seeds. The seeds that are not eaten by mice grow into new flowers that further sustain the mouse population. Lastly, hawks are included in the simulation that eat the mice. This simulation is used as the basis of three lessons that form the core of the CS Unit 1 Curriculum.

The original StarLogo Nova tutorials were reordered and expanded to better fit with the lesson plans and suit the new material that was created. An introduction tutorial, separate from any lesson plan, was originally part of the overall curriculum. This introduction tutorial was modified and added to the lesson 1 tutorial to ensure that students would have the necessary background to complete the rest of the tutorial. Additionally, a code tracing activity, where students read through sections of code and were asked to figure out the meaning of them, was moved to the lesson 1 tutorial instead of the lesson 3 tutorial where it was originally found. Finally, part of the lesson was modified and repurposed as an extension rather than required material. The lesson 2 tutorial stayed mostly intact with an extension added that guided students through the addition of a new species to the simulation. The lesson 3 tutorial was only modified cosmetically to match the new tutorial template.


BIO-CS BRIDGE

Bio-CS Bridge Curriculum

Explore SimBeecology

Name _____ Date: _____ Period: _____

Partner's name _____

Activity #1:
Activity #1 Name

For each activity below:

☑ means to complete this task

☑ means to write an answer here

[Add a description for the activity here]

1) **First part of the activity:**
   a. ☑ Step A
   b. ☑ Step B
   c. ☑ Question that should be answered
2) **Second part of the activity:**
   a. ☑ Step A
   b. ☑ Step B
   c. ☑ Question that should be answered

Figure 7: A template for tutorials. Included are activity titles, section title, and markers for students to either complete an action or write an answer.

The curriculum was also expanded to include a tutorial that addressed procedures. This tutorial was created with feedback from the teachers that more abstraction could be included in the curriculum. Students were first guided through how to identify areas of a program that could be abstracted into a procedure, then guided through the creation of a procedure. The tutorial also included parameters in the procedure to ensure that future examples of procedures with parameters wouldn't need to be re-explained significantly.

A tutorial template was created in order to reconcile the multiple different formats used in the original tutorials. The new template was based on the colors in the Bio-CS Bridge logo and the activity and extension structure that was created (Figure 7). Also added to the template was the copyright statement explaining the share alike license that is present on all Bio-CS Bridge curriculum. Lastly, icons to signal that students must answer a question or complete a task were added to decrease the likelihood that a student misses a section. According to the Bio-CS Bridge teachers, these icons are especially helpful to students with learning differences.

3.3. Creation of the NetLogo Simulations and Tutorials

The NetLogo simulations that needed to be created were based on the StarLogo Nova simulations that are in the original curriculum. For the beginning of Unit 1, this was the SimBeecology simulation with various stages of completeness.

The NetLogo simulations were not only created to provide a simulation, but also to give examples of proper programming practices. Comments were used frequently in the code to explain procedures or specific lines. Additionally, the code was organized to make the addition of more species to the simulation easier for students by grouping the similar statements into blocks that could be used for reference.

To create the initial simulations with only the bees and flowers, the StarLogo Nova code was studied and used as a baseline for implementation. Most commands in StarLogo Nova had a direct NetLogo equivalent. The NetLogo simulation was started with the creation of the flower and bee breeds. Next, the setup procedure was created that spawned the bees and flowers, and the go procedure was created with helper functions that created the movement of the bees and the pollination and seed generation interactions. Figure 8 shows the beginning of the NetLogo code. The complete code and all tutorials are uploaded as part of this report. They will be published on the Bio-CS Bridge Curriculum webpage in Summer 2022.

```
breed [ flower flowers ]
flower-own [
  flower-occupied
  successful-pollination
]

breed [ bee bees ]
bee-own [
  bee-getting-reward
  bee-pollen-color
]

breed [ mouse mice ]
breed [ hawk hawks ]
breed [ Seed Seeds ]
Turtles-own [
  energy
  age
]

to setup
  clear-all
  create-bees
  create-flowers
  ask patches [
      set pcolor green
  ]
  make-plots
  reset-ticks
end
```

Figure 8: The beginning of the SimBeecology simulation code that defines the agent breeds and the setup procedure.

After the initial simulation was created for the first tutorial, the subsequent levels of the simulation were made. The next major addition was to limit the growth of flowers so that the population did not grow exponentially. After that, procedures were created to allow mice to eat the seeds, and then hawks to eat the mice. Lastly, the simulation was modified to incorporate a procedure with parameters. These simulations built off the first simulation and followed their corresponding StarLogo Nova equivalents except for the procedure simulation.

The simulation for the procedure activity was created from the SimBeecology tutorial with the motivation of being able to use the NetLogo curriculum for the AP Computer Science Principles class. With that in mind, the procedure simulation incorporated parameters to generalize the movement of the mice and hawks. The procedure activity was one tutorial that included significant input from teachers involved with the Bio-CS Bridge project. After an initial meeting that created the idea for a tutorial to cover this subject matter, a follow-up meeting was used to improve the tutorial's wording and organization.

4. Results

Following the research questions, biology and computer science standards were found that could be incorporated more heavily into the current curriculum. For example, the Massachusetts Digital Literacy Framework standard 9-12.CT.d.2 *Decompose a problem by defining functions, which accept parameters and produce return values* was identified as one standard that could be incorporated into the unit 1 curriculum. Additionally, entirely new tutorials and simulations were developed using the NetLogo language to give teachers and students more flexibility in their choice of language. The existing StarLogo Nova curriculum was also improved to better communicate learning objectives.

4.1. Unit 1 Organization

The Computer Science Unit 1 curriculum includes a tutorial for each lesson plan. This tutorial is what students are given to complete to help enforce the lesson they are currently working on. The existing Lesson 1 explored the basic SimBeecology simulation in StarLogo Nova (described in the Methodology section) and assumed a quick orientation had been done to familiarize the students with the interface for StarLogo Nova. Lesson 2a explored adding new trophic levels like adding mice to eat the seeds and hawks to eat the mice. Lastly, lesson 2b showed how to make hypotheses and test them with the simulation. The overall ordering of the lessons stayed the same with the new versions completed during this project, with only slight differences. The orientation was included in the lesson 1 tutorial, the code tracing activity was moved to the first tutorial, and some extra material was added. In addition, an entirely new activity was created to introduce the use of Procedures to make code more modular and less prone to errors.

One of the significant changes to the initial tutorials was done through the addition of extensions. Extensions are additions onto the tutorials that can be completed by students who finish the required work early. These extensions were added to provide additional learning to students who can understand the concepts presented quickly. The extensions expanded on learning that was done in the tutorial, often giving the student more freedom to make additions to the code. The presence of extensions serves to differentiate the curriculum, allowing teachers to engage the varied learners they may have in a single classroom. The last, less technical, major change to the curriculum was the formatting of the tutorials shown in figure 7. The color scheme of headings and drawings was changed to match the Bio-CS Bridge colors, the fonts were standardized to a more easily readable font, and the tutorials were broken up into smaller sections with headings to allow for easier use between class periods. Another addition was the use of icons that show the students tasks that must be completed and questions that must be answered. Although some of these features were present in some of the original tutorials, there was not a consistent formatting among all the tutorials.

4.2. NetLogo Simulations

The main goal of the project was to introduce NetLogo as a language for the Bio-CS Bridge CS Unit 1 curriculum. The was done for multiple reasons, the first being to make the unit more advanced for classrooms that may be able to learn using a text-based language. The next reason was to make the language more suited for the AP Computer Science Principles class, which requires some capabilities not currently implemented in Starlogo Nova, including the use of data structured as lists. To be able to use NetLogo for the curriculum, the existing StarLogo Nova Simulations had to be re-created in NetLogo. Figure 9 shows the NetLogo SimBeecology simulation.

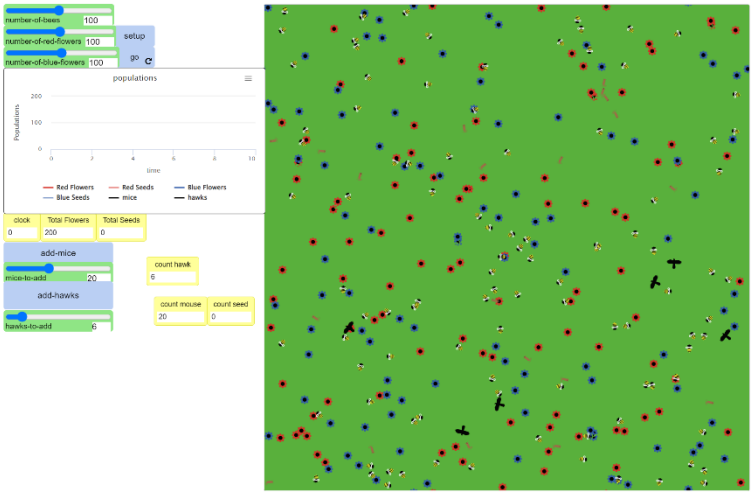


Figure 9: The NetLogo SimBeecology simulation with hawks and mice added. The buttons, sliders, graph, and count boxes can be found on the left while the simulated environment can be found on the right.

The NetLogo simulations that were made corresponded to the different versions of the Starlogo Nova SimBeecology simulation that are required by the tutorials, including a baseline version that students modify as they follow the tutorials, and a completed version that will be made available to teachers. Although the Netlogo simulations were designed to function in the same way as the Starlogo Nova ones overall, there were some improvements made, as well as some modifications necessitated by the NetLogo platform. One improvement was the tuning of the simulation to more accurately represent a balanced ecosystem. To keep the populations of various species in the simulation realistic, , values like how quickly agents move or reproduce or how much energy is gained by eating prey must be balanced appropriately. Additionally, they must be balanced realistically where the population of a higher trophic level must be lower than the ones below it. For example, there must be more mice than hawks in a simulation, and there shouldn't be spikes where the population of hawks rises above the mouse population at any time. With these ideas in mind, balancing was done to ensure that the simulation was representing the biology accurately.

The specific modifications that had to be made between the StarLogo Nova and NetLogo simulations mostly consisted of strategies to complete the same task. For example, NetLogo allows one agent to ask another agent to run a command while StarLogo Nova does not. Because of this feature, the bee pollination code and killing of prey for hawks was more complicated in StarLogo Nova. In NetLogo, the hawk eating the mouse simply told the mouse to delete itself when it was eaten.

4.3. Lesson 1 Tutorial

After consultation with the teachers, the first lesson was reorganized to include an Orientation to the Starlogo or NetLogo platforms. This was done after receiving feedback from the teachers explaining that without the orientation included in the curriculum itself, as opposed to as a separate activity, it might be skipped due to time constraints or on accident since it may not be clearly shown as necessary.

Additionally, the code tracing activity that was initially found in the lesson 2a tutorial was moved to the lesson 1 tutorial. This change was done to add more interaction with the code in the first tutorial and to give the students more time to familiarize themselves with the SimBeecology code before they were asked to modify it in the next tutorial. The NetLogo version of the code tracing activity can be seen in figure 10.

The last major change that was done to the Lesson 1 tutorial was changing the flower explosion work to an optional extension. In the initial tutorial, the students were walked through how to limit the population growth of the flowers so that they would not expand exponentially. This was changed to an optional extension to save the time added with the addition of the orientation and code tracing activities. The students were provided with new code for the next tutorial that implemented this change so that future learning was not dependent on this optional activity.

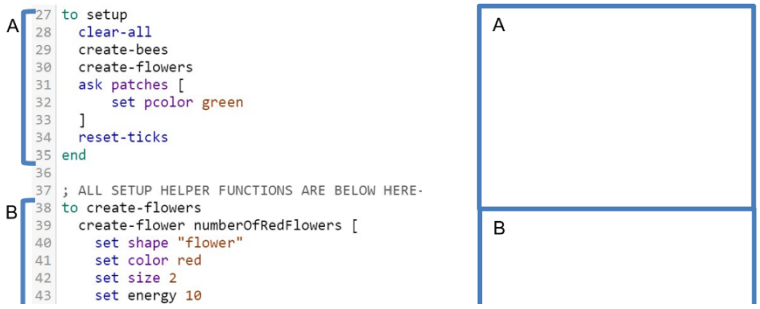
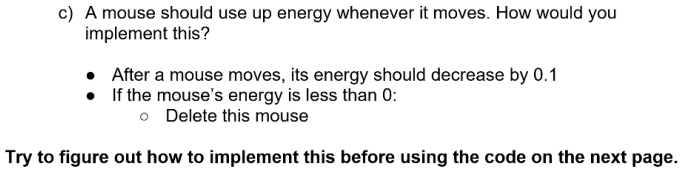


Figure 10: Part of the code tracing activity found in the lesson 1 tutorial. Students are shown the code on the left and asked to describe the use of the code on the right.

4.4. Lesson 2a Tutorial

The tutorial for Lesson 2a centered around the addition of trophic levels to the simulation. Mice were added to the simulation and ate seeds dropped by the flowers. Hawks were then added which ate mice. Students were guided through the process of creating the mice in the simulation and slowly given fewer answers and more direction to come up with their own code. For the hawks, the students were told to refer mostly to the code they implemented for the mice breed and fall back on the instructions, if needed. Then, for the extension for this tutorial, students were asked to find another organism that could be added to the simulation by using a food web. The students were given open-ended questions to guide their creation of the new animal breed in the simulation as shown in figure 11.


c) A mouse should use up energy whenever it moves. How would you implement this?

- After a mouse moves, its energy should decrease by 0.1
- If the mouse's energy is less than 0:
  - Delete this mouse

Try to figure out how to implement this before using the code on the next page.

Figure 11: The guide for students that explains the problem and asks for their own solution. The layout of the lesson was planned such that the code for the function is found on the next page so that the students can attempt to solve the problem on their own before they are given the answer.

4.5. Lesson 2b Tutorial

The lesson 2b tutorial guided the students through testing of hypotheses with the SimBeecology simulation. Students were asked to generate hypotheses relating to bees as a keystone species and test their hypotheses with multiple runs of the simulation, recording the data from each one. Figures 12 and 13 show a table that students are asked to fill out and the data that students use to fill out the tables.

This tutorial serves as one of the major sources of biology education for the beginning of the unit. After creating biological rules when programming the mice and hawks in the first tutorials, students are then asked to use the simulation to draw conclusions about the interactions between these organisms.

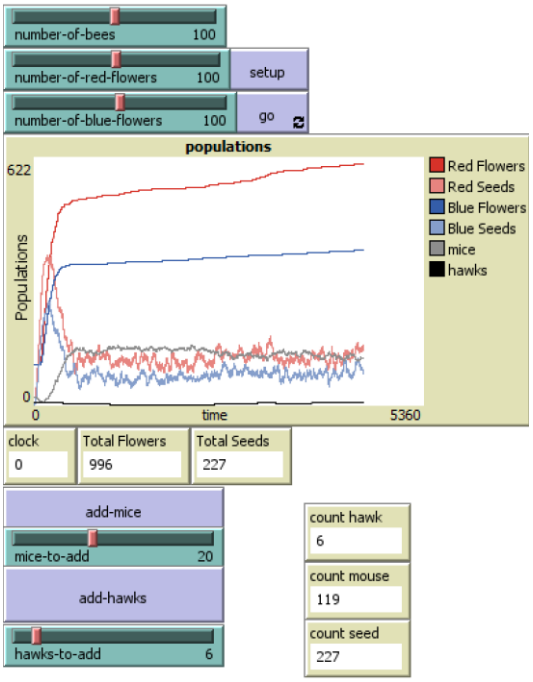


Figure 12: The data from NetLogo that students are asked to use to fill out the tables within the tutorial.

Run #1 - Baseline

Starting Numbers	Bees	Red Flowers	Blue Flowers	Red Seeds	Blue Seeds	Mice	Hawks
Run 1							
Run 2							
Run 3							
Ending Numbers	**Bees**	**Red Flowers**	**Blue Flowers**	**Red Seeds**	**Blue Seeds**	**Mice**	**Hawks**
Run 1							
Run 2							
Run 3							
Average							

Figure 13: The first table that students are asked to fill when testing their hypotheses in the lesson 2b tutorial.

4.6. Procedure Tutorial

An additional tutorial was created to show the students how to use procedures in NetLogo and StarLogo Nova. This tutorial was created to begin to adapt the curriculum for use in an Advanced Placement Computer Science Principles course. For curriculum to be used in an AP CSP course, it must be approved by the College Board and must include specific topics. The topics necessary include procedures, lists, loops, conditionals, and more. The procedure tutorial was created with the future goal in mind of making the curriculum usable for this class.

Figure 14 shows the procedures that the procedure tutorial is based off. The starting code had the two procedures on the top for moving the mice and hawks. The bottom procedure was made in the tutorial and is used in place of the two above. The tutorial walked students through the abstraction of the movement of the mice and hawks in the SimBeecology simulation. After identifying the similarities and differences between the different sections of code, the students were given the opportunity to create their own procedure with some direction and then shown one example of a procedure that could be used. The fact that multiple solutions are possible for a given problem was explained and students were encouraged to test their own function in the code before using the one given to them.

```
to mouse-move
  rt random 30 - 15
  fd 0.4
  set energy energy - 0.1
end
to hawk-move
  rt random 30 - 15
  fd 1
  set energy energy - 0.15
end
```

```
to move [move-distance energy-loss]
  rt random 30 - 15
  fd move-distance
  set energy energy - energy-loss
end
```

Figure 14: The original movement code for the mouse and hawk (top) and the new procedure for moving both the hawks and mice (bottom).

5. Conclusion

The goal of the Bio-CS Bridge project is to develop, test, and implement curriculum for high school biology and computer science classrooms. The following three research questions guided the work on this project: 1) Which biology and computer science standards would align well with combined teaching? 2) How can existing curricula be improved to better integrate the two separate subjects? 3) How can existing curricula be expanded to better serve the diverse needs of students? Standards that could be more heavily emphasized in the curriculum were identified and a specific procedure tutorial was made to fill that gap. The unit 1 curriculum was modified heavily to include both NetLogo and StarLogo Nova as languages that could be taught, and to make the overall organization easier for students and teachers. Lastly, between the addition of the new tutorial, the creation of the NetLogo tutorials and simulations, and the addition of optional extensions to the tutorials, the curriculum was expanded to fit the needs of a wide range of students.

The new tutorials that were made for the unit 1 curriculum were created with constant feedback from teachers that are currently using the original curriculum in classes. The feedback was collected through virtual meetings with teachers that use the current Bio-CS Bridge material. This feedback was critical in identifying the necessary changes that had to be made to the original curriculum and identifying new areas that the curriculum could be expanded. The best example of this is the procedure tutorial that was created to introduce abstraction in the Unit 1 CS curriculum.

5.1. Future Work

Through the creation of the new tutorials and the answering of the research questions, more opportunities for growth in the current curriculum were naturally found. One of the major areas for potential future work is the further expansion of the NetLogo curriculum to allow for its use in the AP Computer Science Principles class. The AP CSP class has strict standards on what must be taught for the curriculum to be approved for use including but not limited to procedures, loops, lists, and conditionals. The procedure tutorial was an important first step in this direction. In addition to other requirements that must be met, tutorials should be added that cover lists and looping. This could be implemented by adding memory to bees in the form of a list of the previously visited flowers. Bees could then loop through the list to find ones that it has visited in the past and either avoid or seek them out. This behavior is consistent with known bee behavior from field studies.

Another potential area for future work is in the expansion of biology topics in which the computer science curriculum is taught. In an early meeting with some of the teachers that use the Bio-CS Bridge curriculum the idea of each lesson having a different biology topic was presented and discussed. It was decided that presenting a new biology topic with a new computer science topic could easily confuse students and that a single biology topic should be presented for as much of the unit as possible. Although switching topics within a unit could be confusing, further work could be done to create multiple versions of the same computer science unit, each with a different biology context in it. This could be done to help teach a biology concept that students are learning concurrently or to follow up learning that has been done in the past.

6. References

AAAS. (2009). *Vision and Change in Undergraduate Biology Education*. Washington: American Association for the Advancement of Science.

About the Bio-CS Bridge Project. (n.d.). Retrieved from Bio-CS Bridge: https://biocsbridge.wpi.edu/website/home

Barkataki, R. L. (2011). Teaching teamwork in engineering and computer science. *2011 Frontiers in Education Conference (FIE)*, pp. F1C-1-F1C-5.

Bauer-Wolf, J. (2017, October 27). *Falling Behind*. Retrieved from Inside Higher Ed: https://www.insidehighered.com/news/2017/10/27/even-booms-student-enrollment-not-enough-degrees-keep-jobs-computer-science

Beecology Project. (n.d.). *About the Beecology Project*. Retrieved from Beecology Project: https://beecology.wpi.edu/website/home#page-title

Bodine, E., Panoff, R., Voit, E., & Weisstein, A. (2020). Agent-Based Modeling and Simulation in Mathematics and Biology Education. *Mathematical Biology Education*.

Boytchev, P. (2009, January). *Logo Tree Project*. Retrieved from Elica: https://web.archive.org/web/20090306084150/http://elica.net/download/papers/LogoTreeProject.pdf

Code.org. (n.d.). *Advocate for computer science education*. Retrieved from Code.org: https://advocacy.code.org/

College Board. (2020). *AP Computer Science Principles Course and Exam Description*.

College Board. (n.d.). *AP Program Results: Class of 2018*. Retrieved from College Board: https://reports.collegeboard.org/archive/2018/ap-program-results/class-2018-data#:~:text=ap%2Dcohort%2Dresults%2DAP%2DParticipation.&text=The%20class%20of%202018%20took,in%20the%20class%20of%202008.

CollegeBoard. (n.d.). *AP COMPUTER SCIENCE PRINCIPLES: The Course*. Retrieved from CollegeBoard: https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course

Cooper, S., & Cunningham, S. (2010). Teaching Computer Science in Context. *Association for Computing Machinery*, 5–8.

CSforAll. (n.d.). *About CSforAll*. Retrieved from CSforAll: https://www.csforall.org/about/

CSTA. (n.d.). *About CSTA's K-12 Standards*. Retrieved from Computer Science Teachers Association: https://csteachers.org/page/about-csta-s-k-12-nbsp-standards

CSTA. (n.d.). *CS Standards*. Retrieved from Computer Science Teachers Association: https://csteachers.org/Page/standards

Diallo, A. (2019, April 22). *School districts are going into debt to keep up with technology*. Retrieved from The Hechinger Report: https://hechingerreport.org/school-districts-are-going-into-debt-to-keep-up-with-technology/

Education Super Highway. (2019). *2019 State of the States.* Retrieved from Education Super Highway: https://stateofthestates.educationsuperhighway.org/?utm_source=release&utm_medium=newsroom&utm_campaign=SotS18#national

Egger, D., Elsenbaumer, S., & Hubwieser, P. (2012). Comparing CSTA K-12 Computer Science Standards. *WiPSCE '12: Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, 125–132.

Garraway-Lashley, Y. (2014). Integrating computer technology in the. *International Journal of Biology Education*, 13-30.

Goode, J. (2007). If You Build Teachers, Will Students Come? The Role of Teachers in Broadening Computer Science Learning for Urban Youth. *J. Educational Computing Research*, 65-88.

Hoffmann, L. (2010). Career Opportunities. *Communications of the ACM*, 19-21.

IBM. (n.d.). *The Origins of Computer Science.* Retrieved from IBM: https://www.ibm.com/ibm/history/ibm100/us/en/icons/compsci/

International Bureau of Education UNESCO. (n.d.). *Curriculum strands.* Retrieved from International Bureau of Education UNESCO: http://www.ibe.unesco.org/en/glossary-curriculum-terminology/c/curriculum-strands

Jona, K., Wilensky, U., Trouille, L., Horn, M., & Orton, K. (2014). *Embedding Computational Thinking in Science, Technology,.*

Labov, J. B., Reid, A. H., & Yamamoto, K. R. (2017, October13). Integrated Biology and Undergraduate Science Education: A New Biology Education for the Twenty-First Century? *CBE—Life Sciences Education, 9*(1).

Leachman, M., Masterson, K., & Figueroa, E. (2017). *A Punishing Decade for School Funding.* Center on Budger and Policy Priorities.

LOGO FOUNDATION. (n.d.). *Logo History*. Retrieved from Logo Foundation: https://el.media.mit.edu/logo-foundation/what_is_logo/history.html

Massachusetts Department of Elementary and Secondary Education. (2014, January 9). *Massachusetts' Adaptation of Next Generation Science Standards.* Retrieved from Massachusetts Department of Elementary and Secondary Education: https://www.doe.mass.edu/stem/standards/ngss-maacomparison.html

Massachusetts Department of Elementary and Secondary Education. (2016, October). *Massachusetts Digital Literacy and Computer Science Standards Panel.* Retrieved from Massachusetts Department of Elementary and Secondary Education: https://www.doe.mass.edu/stem/standards.html

National Center for Science and Engineering Statistics. (n.d.). *Women, Minorities, and Persons with Disabilities in Science and Engineering*. Retrieved from National Center for Science and Engineering Statistics: https://ncses.nsf.gov/pubs/nsf21321/data-tables

National Science Teaching Association. (n.d.). *About the Next Generation Science Standards*. Retrieved from National Science Teaching Association: https://ngss.nsta.org/about.aspx

Odadžić, V., Miljanović, T., Mandić, D., Pribićević, T., & Županec, V. (2017). Effectiveness of the Use of Educational Software in Teaching Biology. *Croatian Journal of Education*, 11-43.

Plaza, P., Peixoto, A., Sancristobal, E., Castro, M., Blazquez, M., Menacho, A., . . . Lopez-Rey, A. (2020). Visual block programming languages and their use in educational robotics. *2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 457-464. doi:10.1109/EDUCON45650.2020.9125219

Ryoo, J. (2019). Pedagogy that Supports Computer Science for All. *ACM Transactions on Computing Education*, 1-23.

Tanner, K., Chatman, L., & Allen, D. (2017). Approaches to Cell Biology Teaching: Cooperative Learning in the Science Classroom—Beyond Students Working in Groups. *Cell Biology Education*, 1-72.

The Center for Connected Learning and Computer-Based Modeling. (n.d.). *Software Tools*. Retrieved from The Center for Connected Learning and Computer-Based Modeling: http://ccl.northwestern.edu/tools.shtml

Thurrott, P. (2020, January 21). *Google: 40 Million Chromebooks in Use in Education*. Retrieved from Thurrott: https://www.thurrott.com/mobile/chrome-os/chromebook/228534/google-40-million-chromebooks-in-use-in-education

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, 199-208.

Weintrop, D., Beheshti, E., Horn, M. O., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, 127-147.

Wilensky, U. (2021, April 20). *NetLogo Frequently Asked Questions*. Retrieved from Netlogo: https://ccl.northwestern.edu/netlogo/faq.html

Wilensky, U. (n.d.). *NetLogo*. Retrieved from NetLogo: https://ccl.northwestern.edu/netlogo/

Wilensky, U. (n.d.). *Wolf Sheep Predation*. Retrieved from NetLogo: http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation

Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories—An Embodied Modeling Approach. *Cognition and Instruction*, 171-209.

Wilensky, U., Brady, C., & Horn, M. (2014). Fostering Computational Literacy in Science Classrooms. *Commun. ACM*, 24-28.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). *Running On Empty: The Failure to Teach K–12 Computer Science in the Digital Age*. Association for Computing Machinery.