Building the Bio-CS Bridge: Expanding High School Computer Science Curriculum Using Agent-Based Modeling

An Interactive Qualifying Project

Submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science

Project Advisor:

Dr. Elizabeth F. Ryder

Submitted by:

Ryan Rabbitt

Date submitted: day May 2023

Abstract

The currently in-progress Bio-CS curriculum is designed to teach students of varying computer science backgrounds the fundamental concepts of computer programming using a variety of languages, including HTML, CSS, Javascript, Netlogo and Starlogo. Using various applications of these languages, the curriculum aims to teach the concept of computational thinking, an essential skill in computer science. This curriculum aims to combine the topics of Biology and Computer Science by using agent-based modeling to test hypotheses and model environments. The curriculum is being expanded to better fit the AP Computer Science Principles curriculum, by addressing major topics covered by the course in new lessons and activities.

Table of Contents

1. Introduction

Agent based modeling is a useful tool in a wide variety of fields, including but not limited to computer science. A curriculum that covers computer science principles for students outside of the computer science field is essential as modern society adapts to fully utilize the technology that has become more widely accessible across a variety of job fields. The Bio-CS Bridge project aims to teach computer science principles in parallel with biology curriculum, by teaching various applications of biological models developed in multiple coding languages.

Unit 1 of the Bio-CS Bridge aims to guide students through creating models in either Netlogo or Starlogo. Included in the curriculum are activities that aim to cover the major principles of computer science, including user input, object instances, and method declarations. Through this curriculum, students from various levels of coding skill are guided through a beginner-friendly, accessible coding language that has real-world biological applications, including ecosystem simulations and population modeling. Teaching Netlogo in parallel with biology is an efficient way to introduce students to fundamental biological and computational concepts, as developing ecosystem models requires both an understanding of the elements being tested, as well as the coding logic needed to model said elements' relationships with each other.

Over the course of this project, the Bio-CS Bridge curriculum has been expanded to better fit the requirements of AP Computer Science Principles, by adding new activities to teach coding fundamentals required to be covered by the AP exam. With these new activities, students may become more confident in the development process, and become familiar with complex concepts that are fundamental in computing.

2. Background & Research

2.1 Bio-CS Bridge Project

The Bio-CS Bridge Project aims to develop a curriculum that teaches Biology and Computer Science in parallel in a high school environment. This curriculum addresses computer science principles and applies them in coding activities that model biology concepts. Using real-life applications to teach computer science benefits students in many ways, keeping them engaged in the material as well as showing the purpose in what they are learning. Students are guided through creating their own models and simulations and collecting their own biological data to apply to real-world problems. The main goal of the Bio-CS Bridge curriculum is to enable students to learn both Biology and Computer Science simultaneously, and allow students from either respective background to understand the importance and interdependence of both subjects. The curriculum also aims to be accessible to students from any level of Computer Science experience, providing extensive explanations to those with no coding experience, and extra challenges for those with an affinity for coding. The Bio-CS curriculum is written and developed by an interdisciplinary team of students and professors, from both computer science and biology backgrounds. (*Bio-CS Bridge,* n.d.).

The Beecology Project is a citizen science project aiming to collect and visualize ecological data for native pollinator species. On the Beecology website are a variety of visualization tools that can be used to compare different sets of data from the extensive database of pollinator species. Also included are multiple simulations built in Starlogo and Netlogo, which were developed by the Bio-CS Bridge Project team, and are used extensively in the Bio-CS Bridge curriculum. (*Beecology Project*, n.d.). The Netlogo/Starlogo unit of the curriculum has multiple activities in which students develop their own version of the models, practicing

biology and computer science concepts as they follow the activities. Over the course of these activities, students begin with a basic version of a Netlogo or Starlogo model, and extend said model with new features.

2.2 High School Computer Science Teaching Standards

The Massachusetts Department of Elementary and Secondary Education (DESE) has developed a framework for Computer Science curriculum, spanning from grades K-12 (*Massachusetts Department of Elementary and Secondary Education*, 2016). This framework covers the areas of computing and society, digital tools and collaboration, computing systems, and computational thinking. These main topics are covered differently for each grade level, split into four spans of grades. These spans are grades K-2, 3-5, 6-8, and 9-12. In following the Massachusetts DESE standards, students should gain an understanding of computational thinking and inspiration for a technical career in the future. Curriculum should integrate technology in a way that effectively supports problem solving in a variety of disciplines, as well as prepares students for the technological world of modern day society. Curriculum should also be designed to support students at a variety of skill levels, including support for those who require guidance through tutorials and providing higher difficulty material to engage those with talent in digital literacy and computer science (*Massachusetts Department of Elementary and Secondary Education*, 2016).

At the high school level, as well as all other grade levels, the seven main practices established by the Massachusetts DESE (Connecting, Creating, Abstracting, Analyzing, Communicating, Collaborating, Research) are integrated. Students should understand the impact of technology in society, as well as how to best apply technology to problem solving. Students should gain a strong understanding of computational thinking, such as writing and debugging

algorithms, creating and modifying data structures, and creating computational models to test a hypothesis. Students should gain the knowledge required to build their own models from the initial planning phase to fully functioning code.

2.3 AP Computer Science Principles Standards

The AP Computer Science Principles course covers the core concepts of Computer Science, and is designed to be accessible to students with little to no programming experience. It is an introductory-level course that introduces students to the basics of the computer science field. The course instructs students on how to develop their own algorithms and programs to solve complex problems, as well as explain the real-world applications of computing (*College Board*, 2020).

The AP Exam has two components: a multiple choice section of 70 questions that evaluate students' understanding of the core concepts of the course, and a performance task, in which a student completes a computer program over the course of at least 12 hours that meets the AP exam requirements. Students must submit a video demonstrating their program, as well as a written response providing code segments and explanations of key components in said code. The performance task requirements are open-ended, but require certain criteria, including input and output, list creation and manipulation, student-developed procedures, and function calling. The code can be written in any language, as long as said language includes all of the necessary syntax to fulfill these requirements.

2.4 Agent-Based Modeling

Agent-Based Modeling is a computer simulation technique in which individual agents are programmed to behave and interact with other agents and the environment. Models are built to simulate systems with individual elements that interact in certain ways, and are intended to emulate real-world behaviors

(*Agent-Based Modeling*, 2022). Agent-based models can be designed for a wide variety of systems, such as social networks, ecosystems, or disease spreading. The flexibility of agent-based modeling allows it to be applied to a variety of fields, especially those that study dynamic systems such as ecology and sociology. Multiple pieces of software have been developed to make agent-based modeling more accessible, such as Netlogo, Starlogo, Anylogic, and AgentScript. These softwares include data visualization tools, such as graphs, to better display and analyze the data collected by running a model.

Netlogo is an Agent-Based Modeling software designed to be accessible in both research and education environments. The user interface is simple, and extensive documentation is available from the Netlogo website. Netlogo enables users to create models with large numbers of agents, all of which operate concurrently (Tisue & Wilensky, 2004). These agents can be classified into different types, called breeds by the syntax. With this distinction, Netlogo can be used to model complex systems with different types of individual agents interacting with each other in a variety of ways. Moving agents are defined as "turtles" in Netlogo syntax, while the grid the turtles move on is made up of agents called "patches". Netlogo also provides many primitives for location, adjacent agents, and movement, all of which can be called by agents to define their behavior. With Netlogo's dictionary of predefined functions and primitives, many systems can be designed with little to no programming experience.

The Netlogo language also includes standard programing constructs, including lists, loops, conditionals, and user-defined procedures. Agentsets, a primary feature of Netlogo, act as a collection of agents, and behave similarly to lists. Lists can also be defined in Netlogo, although certain functions only operate on agentsets or lists, and the two constructs are not interchangeable. This is

because while lists are executed in the user-defined order, agentsets are unordered, and thus are executed in a random order each time.

2.5 Planning Models

One essential skill often overlooked in teaching computer science is the ability to plan a coding project. Designing a model first requires planning what to make, an expected end result, and functions required for the expected behavior of the model (*Code Conquest,* n.d.). To do this, developers often use different types of standardized diagrams to map out their final product. Of the many types of diagrams used in this planning stage, UML diagrams are one of the more popular types, as they provide a standard notation for visualizing the design of a system and its internal interactions (*Booch et. al.,* 2005). UML diagrams can be used to model different aspects of a system, such as its structure, components, behavior, and interactions between different components. This allows developers to outline potential functions to be used to model certain behaviors, which is essential in developing an agent-based model, which relies heavily on system interactions.

An example of a class diagram in the standard UML format is shown in Figure 1. Each class is divided into three sections: the name of the class, the attributes and their accessibility, and the functions included in each class. The classes are also connected via relationship arrows, with different implications based on the arrowhead and number.

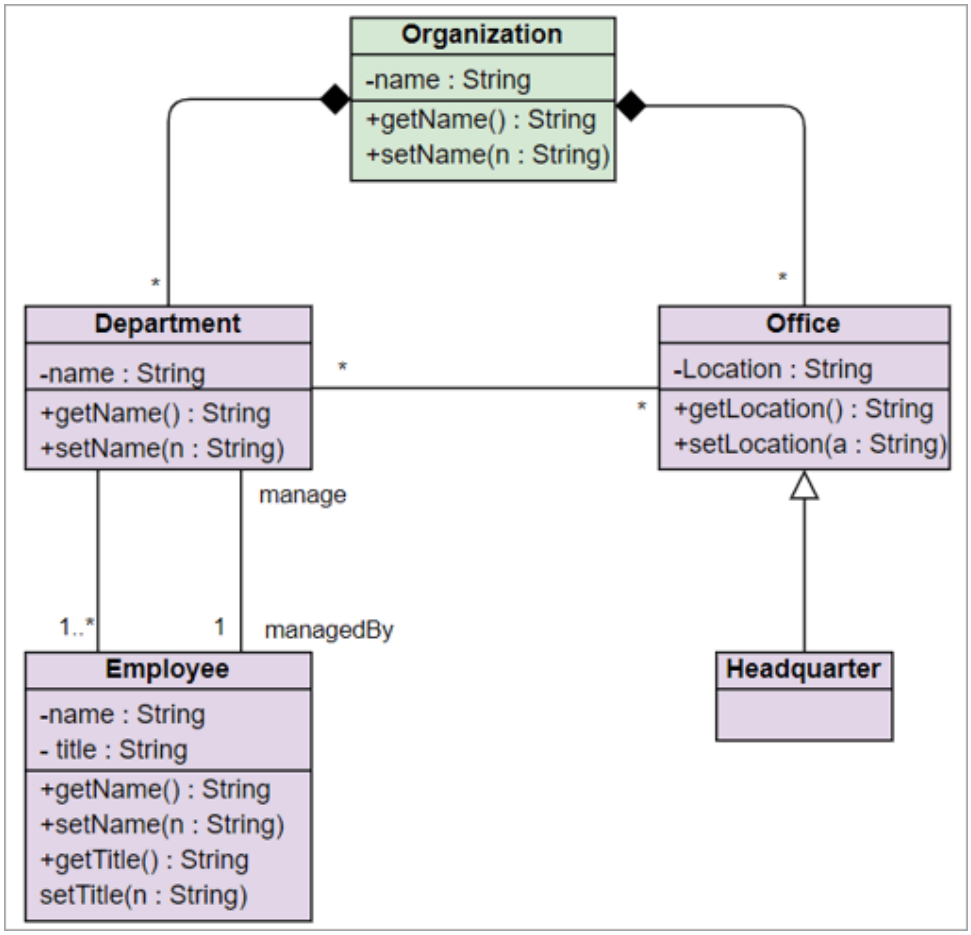

Figure 1. UML Class Diagram Organization Structure (UML diagram tutorial: A complete guide to UML diagrams, 2023).

Another essential tool in planning a model is the Event Diagram (Fig. 2). An event diagram allows developers to visually represent events caused by certain behavior, and the behavior said events trigger in a system. There are a wide variety of formats for event diagrams, although the UML standard contains multiple ways to model this behavior. Sequence diagrams are used to show the order in which functions are called, and which objects call and receive those events. Below is an example of a sequence diagram that models the interactions that occur in an online shopping system.

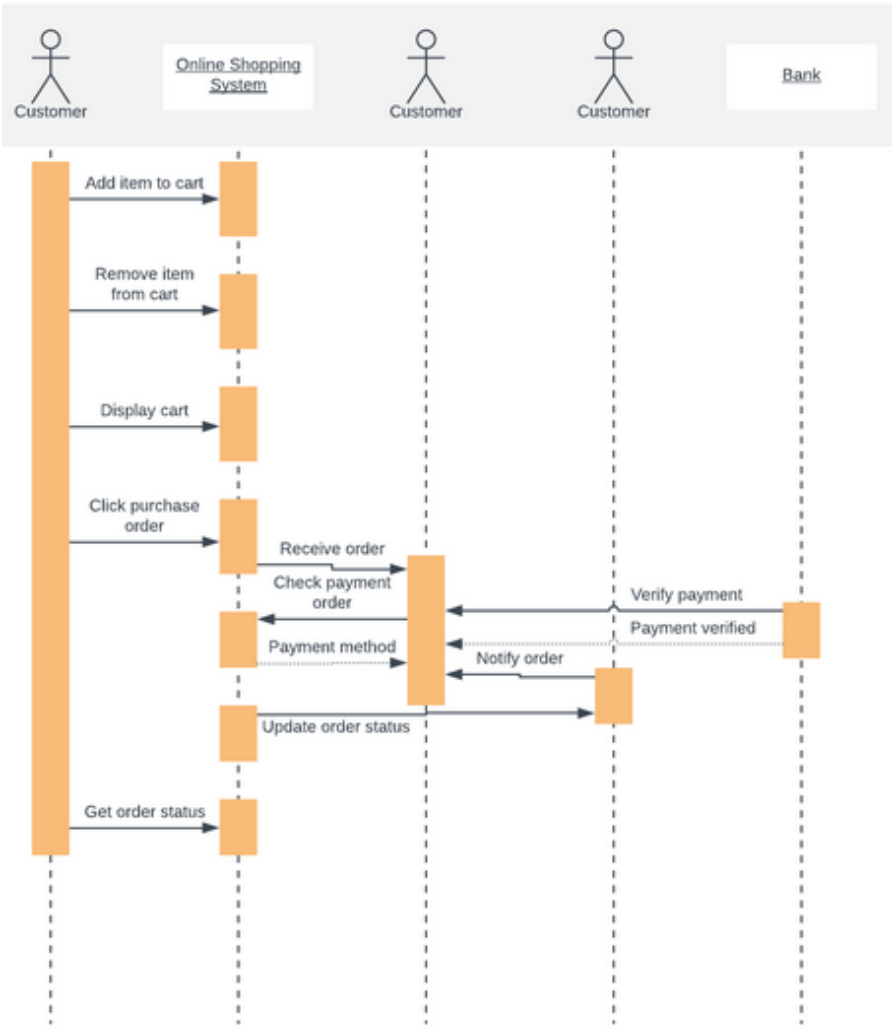

Figure 2. UML Sequence Diagram for an Online Shopping System (Lucidchart, 2020).

While all coding tasks may require different formats for diagrams used in the initial planning stage, the UML standards provide a universal format to be applied in a variety of ways, which can be adapted as developers see fit. Teaching the UML diagram format allows students to think about how their code should flow, and better plan their code.

2.6 Teaching Computer Science

Finding the right methodology to properly cover the essential topics required in a Computer Science course is a challenge that has many different approaches, all

with their own pros and cons. Computer science curriculum should focus on key concepts of the field, and should cover conceptual and experimental issues throughout the course. Two different programs are needed to run in parallel, one for students with a general interest in computer science, and one for those with a deeper interest and understanding of computer science. Both programs should also have both mandatory and optional material, to fully engage students from all backgrounds (Hazzan et. al. 2008).

Using project-based learning is essential in teaching computer science, as the field relies heavily on the essential components of a PBL curriculum. Creative problem solving, critical thinking, and engineering skills are practiced when students engage in real-world problems, keeping students engaged in the material (PBS Education, 2022).

2.6.1 Agent-Based Modeling as a Learning Tool

Using an agent-based modeling software such as Netlogo to teach computer science principles has many benefits, as it is a beginner-friendly tool that generates results that are easy to observe. With the visual feedback presented in Netlogo, students can easily see the results of their code execution, and can decide whether they have achieved the expected results through visual means. Most other types of programming tend to introduce students to code by printing their results to a console, which may be difficult to grasp when students have no prior coding experience.

Agent-based modeling was developed as a tool for easy simulation of interacting objects. In a model, individual objects, or agents, are instantiated, and are allowed to interact. The resulting evolution of the system can be studied from the perspective of the whole population and the individual agent's behavior (Rutgers University, 2003). Because of this easy-to-observe environment, agent

based modeling is a great tool for teaching students, and is thus used heavily in the Bio-CS Bridge curriculum.

3. Methods & Results

3.1 Netlogo Programming Essentials

In order to fulfill the AP CS Principles curriculum requirements, Netlogo's syntax must be considered. As Netlogo is an agent-based modeling language, some of the language's structure differs from procedural and object oriented programming languages. Many of the language's primitives are designed to make certain functionality easy to implement, such as agents and agentsets acting as its equivalent to a class/object type in Java or other object oriented languages. Agents in Netlogo have built-in attributes, as well as custom attributes which can be easily defined.

```
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
]
```

Figure 3. Defining breeds and breed attributes in Netlogo's syntax.

Another key feature in Netlogo is the definition of procedures, or functions. As user-defined procedures are one of the requirements on the AP CS Principles exam, explaining how to define and call a procedure in Netlogo is necessary to include in the Bio-CS Bridge curriculum materials. Included in the existing curriculum is a lesson on how to create custom procedures, although more complex procedures are also introduced in additional lessons.

Netlogo has two different data structures for containing multiple items. Agentsets and lists behave somewhat similarly, although with a few key differences as discussed in section 2.4. Netlogo also contains multiple types of

loops. These two concepts are required to be covered in the AP CS Principles curriculum, although were previously not included in the Bio-CS Bridge curriculum. Included in the new lessons developed over the course of this project are materials on lists and loops in Netlogo, which are introduced with gradually increasing complexity. These lessons will be discussed further in the following sections.

3.2 AP CS Requirements

The AP CS Principles curriculum requires coverage of a few key concepts in programming. These concepts are tested on the AP exam in both a multiple-choice section, and a student-developed computer program. This program requires implementation of all of the key concepts, as well as a written response to show the student's understanding. These concepts include input/output, list initialization and modification, custom procedures, and algorithms (College Board, 2020). The Bio-CS Bridge curriculum aims to cover all of these topics, as well as prepare students to plan and develop their own program.

The current public version of the Bio-CS Bridge curriculum is composed of four computer science units, each using a different coding language. The goal of this project was to cover all of the AP CS Principles exam requirements in Unit 1 of this curriculum, by developing additional activities in Netlogo to cover concepts that were not yet covered. The public curriculum includes activities that cover the basic coding principles of Netlogo, including basic syntax, adding breeds, and basic procedures. With this curriculum, only a few concepts required in the AP CS Principles exam are covered, failing to cover list operations and basic algorithm concepts.

3.3 Additions to Curriculum

The new curriculum developed includes two lessons that cover both lists and loops in parallel, as well as a few lessons that aim to prepare students to develop their own simulation from scratch. These new lessons aim to fill the gaps in the current curriculum, by covering list initialization and modification, list operations, and loops, which are essential in algorithm development. The new lessons also include concepts essential to the development process, which were lacking in the current curriculum.

3.3.1 Lists and Loops Activities

Lists and loops are covered over the course of two activities. The first is a pre-activity that guides students through developing a simple program that utilizes three of the four Netlogo loop types: while, foreach, and repeat loops, and initializes and modifies a list using most of the Netlogo list primitives. The goal of this activity is to introduce students to the concept of lists and loops, and how the two concepts work together. The activity provides simple use cases for each type of loop, as well as applications of the list primitives included in Netlogo, which allow students to become more comfortable with working with more complex data structures and algorithms.

```
; global variables
;  store the color list here so we can work with it
; current-color keeps track of what color we are on
; num-colors keeps track of how many colors are in the color list
globals[
  colors-list
  current-color
  num-colors
]

to setup
  clear-all
  ; create flowers
  create-flowers number-of-flowers [
    setxy random-xcor random-xcor
    set shape "flower"
    set color red
    set size 2
  ]
  ; initialize global variables
  set colors-list [red]
  set current-color red
  set num-colors 1
  reset-ticks
end
```

Figure 4. Part of the starter code for the Lists + Loops Pre-Activity.

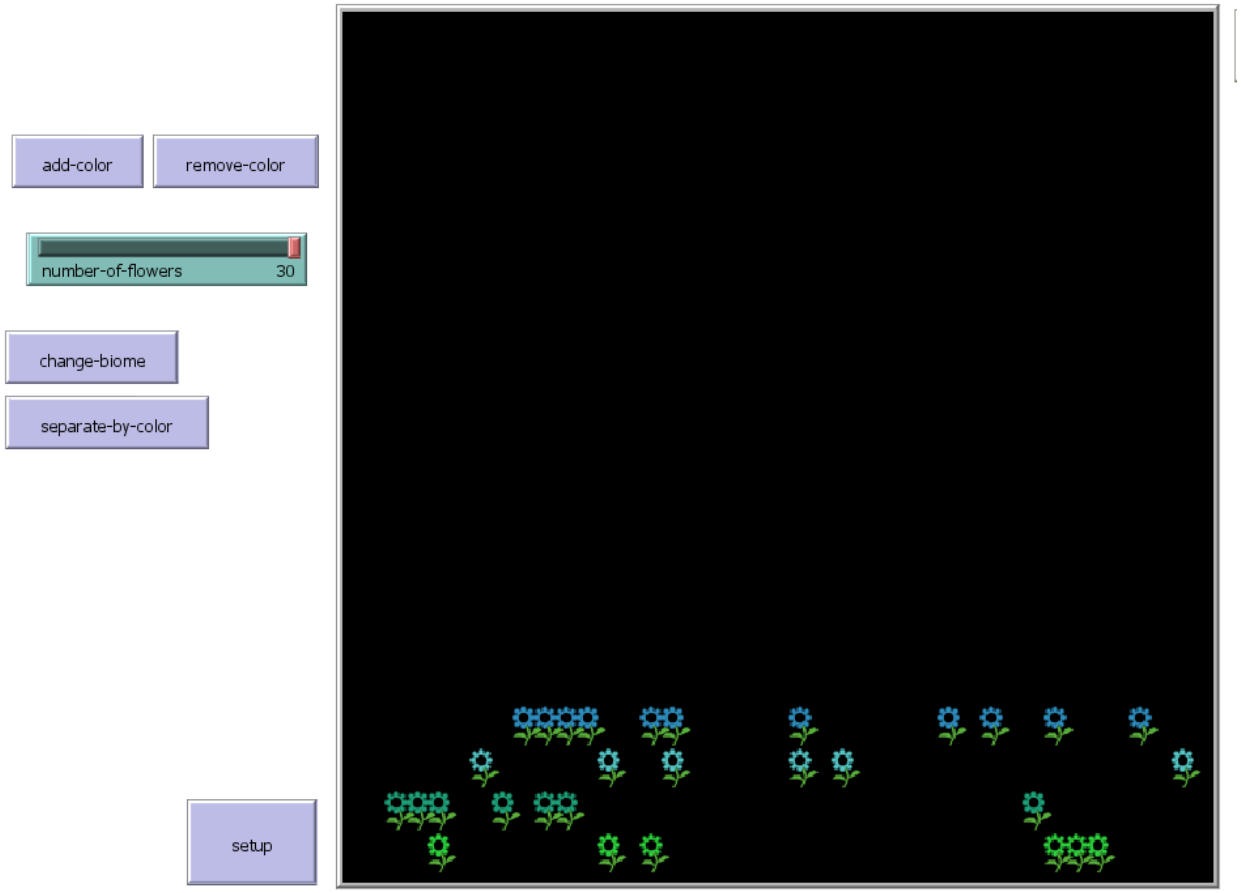

Figure 5. Lists and loops pre-activity resulting program.

The resulting program built in this activity is a simple environment setup that generates flowers in a random location, whose colors can then be changed through multiple user inputs, as well as separated into rows by their color. This functionality is achieved by combining all of the use cases presented throughout the activity, allowing students to demonstrate their understanding of the concepts.

After this activity, students should have enough experience to work with more complex functions using lists and loops, which are covered in the second lists and loops activity. In this activity, students are asked to expand on an existing simulation used in previous activities, called “simbeecology”. The activity guides the students through writing additional functionality in two sections: a bee memory simulation and a modified flower generation method. The first section covers a

more complex use case for lists, while the second section covers a new concept related to loops. The students are given prompts to complete certain functionality, and are asked to identify which list primitives to use, as well as how to create loop structures. A major concept covered in this activity is nested loops, in which one loop calls a second loop inside itself. This is used to generate a grid pattern of flowers.

With the combination of the two list and loop activities, students are exposed to the two coding concepts in gradually increasing complexity. The combined goal of these two lessons is for students to be confident in using lists and loops in their own development process, as the AP CS Principles exam expects students to use both lists and algorithms in the coding task.

2.3.2 UML Diagrams Activity

The remaining new activities aim to prepare students for the coding task in the AP CS Principles exam. The goal of these activities are to teach students how to plan and implement their own ideas for a simulation. With this, students would first learn the coding principles required, and then practice developing their own programs. This bridges the gap from guided activities to independent development, a gap that is a difficult jump for students without any prior coding experience.

The activity first explains the different elements of two different types of UML diagrams, both of which are the most relevant to an agent-based modeling development environment similar to Netlogo. The first of these two is a class diagram, which can be easily applied to breeds in Netlogo. Breeds are essentially user-defined object types composed of user-defined attributes, and can thus be used in place of classes, a structure of attributes and functions used in object-oriented programming. While functions are not breed-specific in Netlogo, they are often intended to only be called by a certain breed. Class diagrams are presented in the

activity with the Netlogo language in mind, and an example is given using a simulation presented in a different activity.

After explaining class diagrams, the activity explains sequence diagrams, another UML diagram relevant to agent-based modeling. This diagram type intends to help developers plan the order of events in a complex system, which students may find difficulty with when developing their own simulations. The same simulation is used in the example for this type of diagram.

After walking students through a simple example, they are asked to fill in their own diagrams using a more complex example, the "simbeecology" simulation used in previous activities. After students complete this activity, they can be asked to go through this planning process in further exercises. Upon completion of this activity, students will have familiarity with two different diagram types. Using a combination of the two, both the structure of object types and the order of events can be planned out, giving a more complete understanding of the expected end product to be achieved in the development process.

2.3.3 Developing Simulations From Ideas

The final activity developed was adapted from an existing activity in the current curriculum written in Starlogo, a simpler, block-based version of Netlogo. Due to the drastically different development environments between Starlogo and Netlogo, the activity needed to be partially rewritten to be used for a Netlogo simulation. The reference model for teachers to use as a completed product also had to be translated to Netlogo. A simple starter code was created for use in this lesson, as students are starting from scratch rather than building off of an existing simulation in this lesson. This starter code is intended to be used for any additional lessons that require students to build a simulation on their own, such as in the lists and loops pre-activity previously discussed.

In the activity, students are asked to brainstorm a set of rules for a simulation of bats in a cave, in which a virus infects the population, and a vaccine is spread simultaneously. This activity intends to introduce students to the concept of

independent development, starting from the set of rules brainstormed by the students, and implementing said rules on their own in Netlogo. It first walks the students through the setup of the simulation, then shows how to implement a basic rule. After these instructions, students are asked to implement the remaining rules on their own. In the curriculum, this activity is the students' first exposure to independent development, as they are given the unguided freedom to implement their rules by coding them. With this activity in combination with the UML diagram activity, students should have exposure to enough of the development process to confidently take on the coding task presented in the AP CS Principles exam.

3.4 Additional Features

Each of the newly developed activities include completed Netlogo simulations to be used as teacher references for both helping students with development issues, as well as grading the students' work. Both the second lists and loops activity and the bat vaccine activity include example simulations at different levels of completion. The lists and loops activity includes the expected simulation after part one and part two of the activity, and the bat vaccine model activity includes the simulation after completion of the first rule, as well as an example of other rules implemented.

The lists and loops pre-activity also includes a slideshow presenting an introduction to the concepts of lists and loops. This slideshow covers the definition and implementation of lists, as well as each of the major list primitives included in Netlogo's syntax. It also covers the structure of each of the four loop types in Netlogo: repeat, loop, while, and foreach. This slideshow can be presented as-is, or can be adapted to fit teachers' own style of teaching.

3.4.1 Netlogo Cheat Sheet

In addition to the lessons developed, reference material was written to better help students learn the Netlogo language. A cheat sheet was written that includes all of the most commonly used functions and syntax, divided by their usage. These sections include functions for turtles, patches, links and the observer, along with each of the UI elements included in the Interface tab. Each element in the cheat

sheet includes a brief description of its functionality and an example code snippet displaying its usage. This cheat sheet is meant to summarize the material learned throughout the unit, to be used as both a quick reference for students and as a study guide.

4. Conclusion

Upon completion of this IQP, the Bio-CS Bridge curriculum has been improved to better fit the AP CS Principles requirements as set by the College Board. These improvements include the addition of activities that cover the topics of lists and loops, essential components of the AP CS Principles curriculum, as well as activities that guide students through the planning and development process of their own simulations in Netlogo. By applying real-world examples of ecosystems and biology concepts, students gain a better understanding of the applications of computer science. Using Netlogo as a teaching tool provides an easy-to-understand user interface, as well as a beginner-friendly language with many built-in primitives that can achieve complex results with little understanding of coding. With this, students can create complex simulations with visual feedback despite having little to no computer science background, allowing them to explore ideas without being discouraged by a difficult-to-understand coding environment. Overall, the new developments in the Bio-CS Bridge curriculum serves as a more complete resource for educators seeking to integrate computer science and biology together, and to promote interdisciplinary learning in the high school environment.

4.1 Future Work

There are several opportunities for future development of this curriculum, to expand the covered topics to provide a more complete coverage of computer science fundamentals. In addition to the new material developed, different versions can be adapted for students with an advanced understanding of coding. This could

include extra credit work, such as additional activities with more room for student interpretation. The curriculum could also expand to cover more advanced computer science topics, such as debugging. This idea was considered for another Netlogo activity, as Netlogo's coding environment provides some of the basic necessities for simple debugging, such as an output console and custom error throwing. Giving students a full understanding of coding fundamentals will prepare them for a future in the increasingly technical world.

References

Agent-Based Modeling. (2022). Columbia University Mailman School of Public Health. https://www.publichealth.columbia.edu/research/population-health-methods/agent-based-modeling

Beecology Project. (n.d.). Beecology Project. Beecology Project. https://beecology.wpi.edu/

Bio-CS Bridge. (n.d.). The Bio-CS Bridge. Bio-CS Bridge. https://biocsbridge.wpi.edu/

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide.* Addison-Wesley Professional.

Code Conquest. (n.d.). *How to Plan a Coding Project.* https://www.codeconquest.com/programming-projects/how-to-plan-a-programming-project/

College Board. (2020). *AP Computer Science Principles Course and Exam Description, Effective Fall 2020*. AP Central. https://apcentral.collegeboard.org/courses/ap-computer-science-principles/exam

General Electric Company. (2019). *Event Diagram.* GE Digital Solutions. https://www.ge.com/digital/documentation/meridium/Help/V43070/IZDZkZmFhNjAtZjEyZS00Y2VkLTg1N2ItN2M1NzJmMjk3OTc3.html

Hazzan, O., Gal-Ezer, J., & Blum, L. (2008). *A Model for High School Computer Science Education: The Four Key Elements that Make It!.*

Lucidchart. (2020). *Types of UML Diagrams*. Introducing Types of UML Diagrams | Lucidchart Blog. https://www.lucidchart.com/blog/types-of-UML-diagrams

Massachusetts Department of Elementary and Secondary Education. (2016). *Digital Literacy and Computer Science Grades Kindergarten to 12*. Digital Literacy and Computer Science (DLCS) Implementation Resources - Science, Technology/Engineering, and Mathematics (STEM). https://www.doe.mass.edu/stem/dlcs/

PBS Education (2022). *Five Steps For Integrating Computer Science in the Classroom*. https://www.pbs.org/education/blog/five-steps-for-integrating-computer-science-in-the-classroom

Rutgers University. (2003). *Agent-Based Models of Industrial Ecosystems.* https://web.archive.org/web/20110720041914/http://policy.rutgers.edu/andrews/projects/abm/abmarticle.htm

Tisue, S., & Wilensky, U. (2004). *NetLogo: A Simple Environment for Modeling Complexity.* https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a65e2af0f4a1b03db4b05357c4cb3b8a6a4d7894

UML diagram tutorial: A complete guide to UML diagrams. Software Testing Help. (2023). https://www.softwaretestinghelp.com/uml-diagram-tutorial/