HOW USE-MODIFY-CREATE BRINGS MIDDLE GRADES STUDENTS TO COMPUTATIONAL THINKING

Jennifer Houchins[1], Danielle Boulden[1], James Lester[1], Bradford Mott[1], Kristy Elizabeth Boyer[2], & Eric Wiebe[1]

[1]*North Carolina State University;* [2]*University of Florida*

This design case chronicles the efforts of an interdisciplinary team of researchers as they collaborated with middle grades science teachers and students to build and refine an epidemic disease curriculum module. The initial five-day design was delivered in five science classrooms at three nearby schools where researcher classroom observations and teacher feedback drove iterative refinements of the module's materials. The final design of this module consisted of four instructional days of modeling and simulation activities that integrate computational thinking practices, computer science concepts, and life sciences content. The paper aims to illustrate the design motivations to address contextual constraints such as tight curricular schedules and varied levels of exposure to programming for both teachers and students. The instructional materials presented in this design case were the result of a three-year long research-practice partnership with science teachers at nearby middle schools.

Jennifer Houchins is Director of Technology Programs at the Friday Institute for Educational Innovation.

Danielle Boulden is a Research Scientist at the Center for Educational Informatics.

James Lester is Director of the Center for Educational Informatics and a Distinguished University Professor.

Bradford Mott is a Senior Research Scientist at the Center for Educational Informatics.

Kristy Elizabeth Boyer is Director of the LearnDialogue Group and an Associate Professor of Computer & Information Science & Engineering.

Eric Wiebe is a Senior Research Fellow at the Friday Institute for Educational Innovation and Professor of STEM Education.

INTRODUCTION

In 2006, Jeannette Wing called for computational thinking (CT) to be seen as a universally applicable attitude and skill set. Since that time, CT has been recognized as an essential component of addressing future STEM workforce needs, particularly for jobs involving computing, because it provides students with critical problem-solving and higher order thinking skills across disciplines (Settle et al., 2013). This has prompted schools to push for the incorporation of CT into classroom instruction via school-wide initiatives. One such initiative was the impetus for designing our epidemic disease curriculum module to integrate CT into the standard science curriculum.

Designing a curriculum module that integrates CT into the standard middle grades science instruction required our design team to consider the needs of two audiences: students and teachers. For this reason, each time the materials of our curriculum were piloted in the classroom, a member of the design team was present to observe the instruction and the interactions of both audiences with the materials. In this way, our design team sought to develop a deep understanding of how our materials were used in the classroom and any failures associated with our design. To accomplish this, we relied on team debriefs at the end of the pilot and conducted member checks with teachers to drive the key design decisions that resulted in the final version of our curriculum module (Smith, 2010).

This paper details the design process for developing an epidemic disease curriculum module of four classroom

instructional days that enables middle grades students to iteratively build a computational model simulating the spread of an epidemic disease. In this design case we outline our efforts to build and refine instructional materials for our epidemic disease curriculum module that integrate computational thinking practices, computer science concepts, and life sciences content through scientific modeling within a programming environment. In the following sections we illustrate the rationale behind our design decisions as they were influenced by the backgrounds of a design team consisting of faculty and researchers with research interests in computer science education. We also describe contextual constraints to assist the reader in understanding the scope of issues that prompted these design decisions. Finally, we try to provide a thick and rich description of our design process that includes two iterative cycles of refinement informed by collaborative reflexive practice through implementing our materials in the classroom. In this way, we seek to provide the reader with a rigorous design case that addresses the key quality markers set forth by Gray (2020).

CONTEXT OF THE DESIGN

The designed instructional activities and materials described in this narrative evolved from a National Science Foundation (NSF) funded project to create a curricular unit to support the development of middle school students' computational thinking (CT) and computer science (CS) practices. Ultimately, the project sought to foster students' development of computational thinking practices through engagement with computationally rich science and problem-solving activities. The work described here was part of a larger project developing both a game-based learning environment and a set of out-of-game activities. This design case focuses on this later phase of work. The goals of this second phase were to develop out-of-game activities that enabled students to model, simulate, and analyze data on scientific phenomena using block-based programming interfaces within their science classrooms.

Our design team was composed of eight people: 1) one faculty from science education, 2) one faculty and three researchers from computer science with a focus on education, and 3) three researchers with backgrounds in the learning sciences. The design team came together to collaborate on an interdisciplinary project with a focus on computational thinking and computer science education, the major area of research interest for each member of the design team. It's important to note that although each member of the team had instructional and curriculum design experience within their respective fields, only one of the team members had extensive expertise designing integrated curricula that furthered the learning of scientific phenomena through computer science practices.

The team was guided by the Next Generation Science Standards' (NGSS) proclamation of CT and modeling as two essential science and engineering practices that should be used to forward disciplinary core understanding in teaching and learning (Lead States, 2013). Additionally, as the majority of our design team have a focus on CT and computer science education, we were inspired by researchers such as Weintrop et al. (2016) and Wilensky et al. (2014), who have done prior work in science education by integrating science disciplinary content with CT practices through modeling and simulation. We felt strongly that the combination of CT and modeling together could be a powerful strategy for the teaching and learning of scientific concepts and thus sought to design curricular materials that would be appropriate for middle grades students and teachers in mainstream science classrooms because of the results shown by these authors. For this reason, we decided our design would be centered around developing students' CT and computer science conceptual knowledge by learning modeling and simulation.

However, there were a number of contextual constraints that we needed to contend with as we began to think about designing and developing our materials. The first constraint that presented a challenge for our design was how to integrate CT into the science curriculum while balancing the CT, CS, and science content effectively. Although the promotion of CT in mainstream K-12 classrooms is on the rise, definitions of learning goals and practices for operationalizing its use in the classroom still vary across disciplines (Lowe & Brophy, 2017). This not only presented a design challenge for our team as we had few existing exemplars to work with, but we also considered it to be a potential instructional challenge for teachers who may very well have not had the necessary background to integrate CT practices that involved programming.

The second major contextual constraint we faced in designing our epidemic disease module was a wide variability of prior student experience with computing. Meeting the needs of students with varying levels of exposure to computing practices continues to be challenging in formal K-12 classrooms with diverse student backgrounds and skillsets. We therefore felt that we needed to be particularly sensitive in designing instructional supports that would ensure that our curricular objectives were accessible by all students.

This contextual constraint was not only evident for students but for teachers as well. Another design challenge that we faced was the need for materials that integrate CT practices into science content but offload the need for students or their teachers to have a background in programming. We found that many of the teachers we worked with also had low exposure to and self-efficacy around CT and CS concepts and practices.
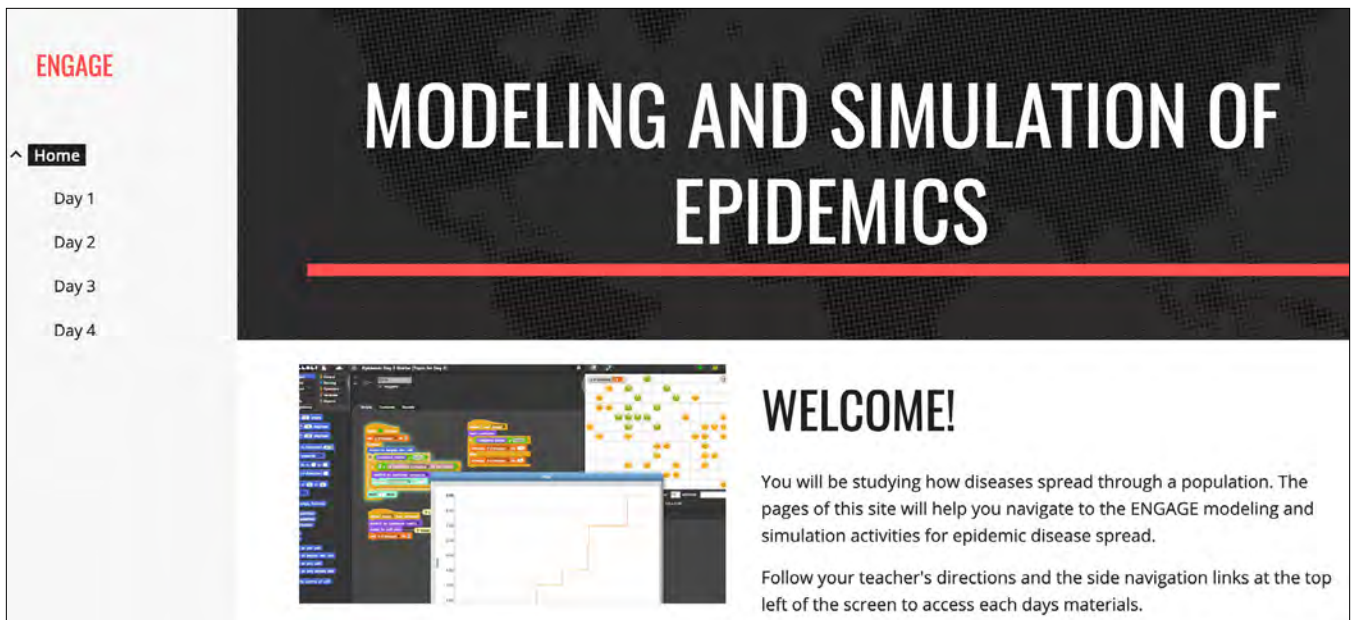
FIGURE 1. Google site designed to deliver the instructional materials of our disease spread module. The site provided teachers with a single repository and easy access to the online materials for each day's activities

THE DESIGN PROCESS

The following sections describe the initial design of our epidemic disease curriculum module, issues with the design observed during classroom delivery of the module's materials, and the design choices driven by those observations that resulted in our final curriculum design. Our design team engaged in an iterative process, guided by both external feedback and internal review, for refining our materials.

Description of the Materials

We delivered our curriculum to students through an online portal (shown in Figure 1) created using Google Sites™. This website was created to help teachers facilitate their students' navigation throughout each instructional day of the unit. Each day was indicated by a tabbed menu on the left side of the portal and included a hotlink to the Snap!-based programming environment, as well as an introductory statement that outlined to students what they would be expected to accomplish during the class period. The teacher guides and student worksheets were created in Google docs so that they could be easily arranged in a folder to share with teachers as they implemented the curriculum. If requested by the teacher, the students were given printed versions of the worksheets for each day of the unit.

Components of the Initial Design

Our research and development team initially designed a five-day unit that enabled students to model the spread of epidemics using an agent-based paradigm. Motivation for the chosen science content for the design of this unit was driven in part by the inclusion of epidemic disease in

our state learning standards and this topic's amenability to computational modeling. Moreover, empirical research studies (e.g., Wilensky & Rand, 2015) supported our intuition this content had great potential to be represented computationally, lending further support for our design decision. The five-day unit we developed for the classroom consisted of both "unplugged" (e.g., without a computer) and "plugged" (e.g., with a computer and engagement in programming) instructional time.

Each activity within the unit was designed to add a layer of complexity—both conceptually and computationally—as students learn to build block-based code that models the spread of disease. Our design team sought to strike a balance between developing CT skills and science conceptual knowledge with the progression of the instructional activities that introduced the scientific content through modeling and block-based programming. Each activity in this progression was structured to fit within one 45-min block of instructional time, or one day of instruction in the typical middle school science classroom. The details of each instructional block are provided below.

Day 1

The curriculum module begins with teachers introducing (or reviewing) the concepts associated with the spread of disease. The teacher guides provided with our instructional materials include relevant vocabulary that students are expected to know at the end of the unit (an example is shown in Appendix A). Once the introduction to the vocabulary and concepts is complete, teachers lead students in an embodied cognition activity (see Figure 2) designed to help students translate and abstract conceptual ideas about

epidemics. The design team's decision to include an embodied cognition activity was influenced by empirical research studies that suggest such an approach enhances student learning of abstract concepts by allowing them to make mental connections by physically modeling the spread of a disease as they adopt characteristics and behavior models of people and viruses while moving about the classroom (e.g., de Jong et al., 2013; Shapiro, 2019; Wilensky & Reisman, 2006).

Figure 2 shows the embodied cognition activity instructional steps provided in the teacher guide. The aim is that students move around the classroom mimicking the movements of individuals in the disease spread simulation. The teacher guide recommends teachers use the tiles of their classroom floor as a grid-like guide for students' movements. The instructional materials include laminated emojis that represent

Part 3: Model this in class
- Have students space out
- Each student should have a healthy or infected emoji picture to indicate what their current state is
- To begin this activity, everyone is healthy
- Teacher calls out one day so students move one step in any direction. After each day (one step), students assess who is healthy or infected. Those who are infected raise their hands and teacher tallies the number on the board
- After a number of moves, students should note that no one will ever be infected.
- Now 'seed' one student as infected in the class.
- Replay the simulation and observe that the tallies of infected on the whiteboard begin to grow.

FIGURE 2. Teacher guide directions for leading embodied cognition activity. As teachers lead students through the activity on day 1, their movements about the classroom "embody" the interactions of individuals in the disease spread model used in subsequent activities.

whether a student is "healthy" or "infected." As students move about the classroom, they carry the emojis to represent their current state. At the end of each step, the students who have the "healthy" emoji assess whether they have come into contact with another carrying the "infected" emoji (i.e., an "infected" student in a neighboring tile). If so, the teacher instructs them to change state by flipping their emoji over to "infected." Thus, the design of this day's activity is such that students' movement about the classroom and engagement with one another "embodies" the modeled behaviors they will observe in the epidemic disease module's subsequent instructional activities. Whereas it is important to note that some classrooms may not have the advantage of a tile floor that resembles the modeling environment, this activity's design aims to mimic the computational model.

Immediately after the embodied cognition activity, students complete the worksheet shown in Appendix B, formalizing the notion of defining agent characteristics and behavior through rule building observed during the activity. This worksheet also incorporates elements of pseudocode, which offers students scaffolding as they are exposed to how full sentences can be transformed into programming code (a design choice based on the work of Grover, Pea, & Cooper, 2015).

Day 2

On day 2 of the module, students begin building a computational model by programming a person agent and assigning its characteristics and behaviors within the block-based learning environment. To begin the day's instructional time,
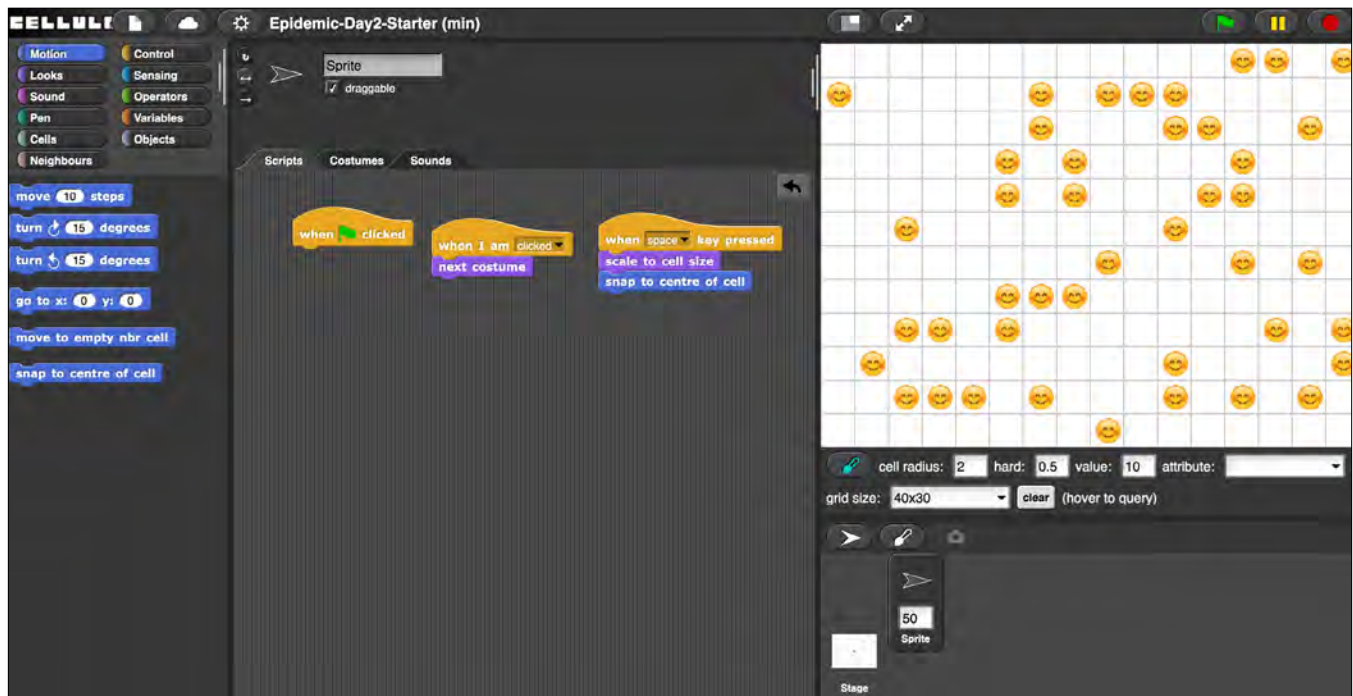
FIGURE 3. Cellular environment set up for students to start building an epidemic model on day 2 of initial design.

teachers introduce students to the learning environment, Cellular (Monash Cellular, n.d.), by pointing out that the code blocks are color-coded based on the type of functionality and how to execute their models.

Cellular is a block-based programming environment developed at Monash University. Cellular builds on Berkeley's Snap! environment by adding a grid-based structure to the stage and special code blocks to control agents' movements on the grid. This environment is specifically designed for building agent-based models like the disease spread simulation, making it a great fit as a learning environment for our epidemic disease curriculum module.

Figure 3 shows the Cellular environment pre-loaded with our designed starting point for the computational model students will build for this day's activity. The starting point includes 50 emojis on the stage that represent the person sprite, a block-based programming term used to refer to a computational agent, and three code-block starters. This is what students see when they first log in to Cellular. The remainder of this day's activity consists of students using the programming environment to build a model that includes a person agent with two attributes, healthy or infected, and follows the same rules of behavior they outlined in the previous day's worksheet. The overall goal of this day's instructional time is that at the end of the day's activity, students should have a working block-based model where person agents move around in the simulation and healthy individuals who encounter infected individuals become infected themselves.

Day 3

The third day adds complexity to computational artifacts students develop over the course of our instructional unit by focusing on concepts like iterations, variables, and initial conditions. Students extend the model they began building on day 2 with functionality that simulates how populations are infected over time. Instructional time includes reviewing the embodied cognition activity from day 1 where the number of infected individuals is tracked at each step of the activity. Students then learn how this information is translated into variables and how to produce plots to visualize infection rates over time within the block-based learning environment.

Day 4

On this day of "unplugged" activity, instruction is centered around a worksheet we designed where teachers ask students to use a flowchart (i.e., a Finite State Machine; see Appendix C) to help model the interactions and variable changes occurring between four health states (healthy, infected/contagious, infected/not contagious, and dead). Throughout this activity, students complete the worksheet we designed where they determine which states a person in the disease model can transition between and in which direction that transition occurs. Once they have described the disease state transitions, teachers instruct students to research and incorporate other information about a particular disease into their flowchart worksheet. For example, the teacher guide is designed to include some flexibility in
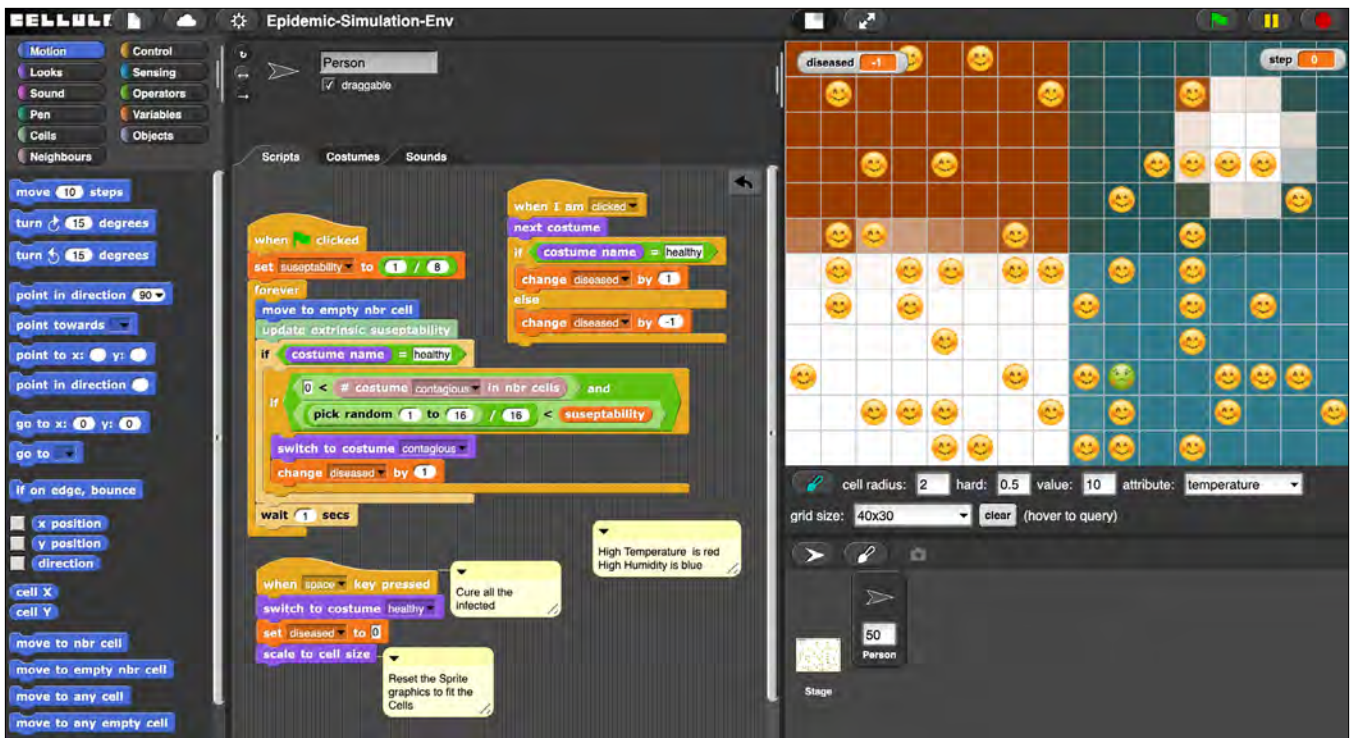

FIGURE 4. Pre-built model for student scientific exploration of disease spread on day 5.

Blocks and Scripts

In Snap, all code is made out of **blocks**. When you "snap" blocks together, they form a **script**, which runs from top to bottom. You create a block by dragging it in from the list on the left. There are 10 categories of blocks (e.g. Motion, Sensing), and you can change the category to find more blocks. Each category's blocks have a different color, so all "Motion" blocks are blue.

← script

FIGURE 5. In this figure, readers can see how basic elements of the block-based programming are presented and explained in the supplemental materials of the curriculum module to provide scaffolding for students with less familiarity with programming.

that teachers may have students focus on seasonal flu and research model parameters such as the average period of time an individual is infected and the amount of that time that they can transmit the disease to others.

Day 5

The final day of the unit is designed to provide students with an example of how computational models are useful for exploring scientific concepts like the spread of disease. Learning scientists and computer science researchers on our team designed a pre-built simulation in the Cellular modeling environment (Figure 4) that employs all the concepts they have learned over the course of the module. This simulation is pre-loaded in the environment when students log in for the day's activities and allows students to explore the effects of humidity, temperature, and population density on disease transmission by carrying out experiments with different values of these variables and making observations about how these environmental factors influence disease spread through the population. The design team also developed a worksheet (Appendix D) to accompany the simulation exploration activity that asks students to predict the impact of these environmental factors and then directs them on how to carry out the experiments to explore their predictions.

Supplemental Materials

In addition to both plugged and unplugged activities, the activity materials included optional tutorials for basic block-based programming concepts (sample shown in Figure 5) and the Cellular programming environment to address the experience gap between those students who had been exposed to some form of programming in agent-based modeling environments (e.g., Scratch) before and those who had not. We designed these supplemental materials as additional scaffolding for students with little to no prior exposure to block-based programming.

DESIGN SHORTCOMINGS OBSERVED FROM THE INITIAL CLASSROOM IMPLEMENTATIONS

Our initial five-day epidemic disease curriculum module was implemented in the science classrooms of five different teachers at three nearby schools during the spring semester of 2018. These implementations offered our team invaluable opportunities to solicit feedback from teachers and students, as well as observe students and teachers experiencing the curriculum in authentic classrooms. After our team collected and analyzed the feedback and observational notes, we met to reflect on and discuss how our curriculum could be improved to meet our initial project goals. After peer debriefing and triangulation of these data, we concluded that the shortcomings of our epidemic disease curriculum module fell into three categories of design failures (Gray, 2020): 1) the design was not implemented as intended, 2) the design failed to address the complexity of the learning context, and 3) the design fell short of producing the learning outcomes desired.

Elaborating on the first design shortcoming, we noted that not all the materials were implemented in the classroom. Given an already tight curricular schedule, science teachers were reluctant to include any of the programming tutorials along with the 5-day curricular modules. As a result, many of the students who lacked prior experience with programming struggled to build the scientific model for day two without intense instructional support from others.

In addition to the design not being implemented as intended, we also failed to account for the complexity of the learning context in that we needed to meet the needs of both the students and the teachers. Our participating teachers had little to no background in programming (we now believe to be the general case for middle grades science teachers), and therefore lacked both the confidence and ability to not only provide the necessary support to struggling students, but also to allow more experienced students the freedom to explore and test out their own programming solutions. Instead, teachers were observed regressing to a direct instructional approach that they perceived didn't need the same depth of computational modeling PCK (pedagogical content knowledge) (Lytle et al., 2019), as they would require students to follow along in lock-step as they watched their teacher develop a computational modeling solution step-by-step. This approach also perpetuated the shortened coverage of curricular materials.

The final shortcoming we observed for our materials was that they did not produce the desired learning outcomes. For example, teachers reported that day 4 of the curriculum added little pedagogical value to the overarching goals and objectives of the unit. In particular, students seemed

to be unable to directly translate the concepts depicted on the worksheet to the computer science concepts. Finally, the pedagogical approach that we presented to students with our current materials had limited effectiveness across a diverse body of students. Essentially, days two and three of the curriculum where students were tasked with building programming algorithms from scratch proved to be too much of a cognitive burden for the majority of the students. In contrast, many of the students did not find the final day of the materials where they use a pre-built model to run a simulation and analyze data challenging enough.

Reflecting upon these design shortcomings, our team concluded that both the students and teachers needed a revised curricular strategy that provided sufficient scaffolding for CT concepts. We decided that we also needed to more faithfully implement our initial curricular strategy and design the materials so that they progressively became more challenging each day, fading instructional scaffolds throughout the unit as students became more familiar with both the science and computer science content. Thus, these factors were the impetus for a substantive redesign of our curricular activities to include additional scaffolded support within a Use-Modify-Create (UMC) progression.

THE FINAL DESIGN: USE-MODIFY-CREATE

In order to address the shortcomings of our prior design, the team decided to adopt the Use-Modify-Create (UMC) curricular progression strategy. The UMC progression, originally developed by Lee and colleagues (2011), is a three-stage learning progression strategy designed to intentionally engage learners in CT through rich computational environments. The rationale for revising our curriculum to follow

this progression is that, according to Lee et al. (2011), it both supports and deepens learners' acquisition of CT based on scaffolding increasingly deep interactions with programming code and CT concepts. Each stage of the UMC progression builds on the previous stage, allowing learners to gain comfort with CT concepts while engaging in investigations. The general progression has students "Use" existing code to understand how it functions within an existing program. Next, students would be guided through a series of activities where they "Modify" the code—meant to build efficacy while deepening their understanding through exploration. Finally, students "Create" new code that embodies the key CS/CT concepts they've been exploring and learning. This progression is expected to not only support students' development of CT skills, but to facilitate increasing ownership of the computational artifacts with which they are engaging.

In the context of our curriculum, the UMC progression was employed as a design decision to provide additional scaffolding for students with less programming experience. Such a design simultaneously scaffolded teachers, such that they were able to build their knowledge and comfort level in supporting the unit at a manageable pace. It also served to shorten the instructional time to four days rather than five, thus alleviating some of the instructional time constraints that teachers expressed experiencing. Despite the shortened instructional time, the revised materials did not cut any substantive content and the previously separate tutorial materials were integrated into the core content. Finally, the first day of instruction did not change, remaining an "unplugged" day where students receive an introduction to the scientific concept they will be exploring computationally for the remainder of the module and engage in the embodied cognition activity.
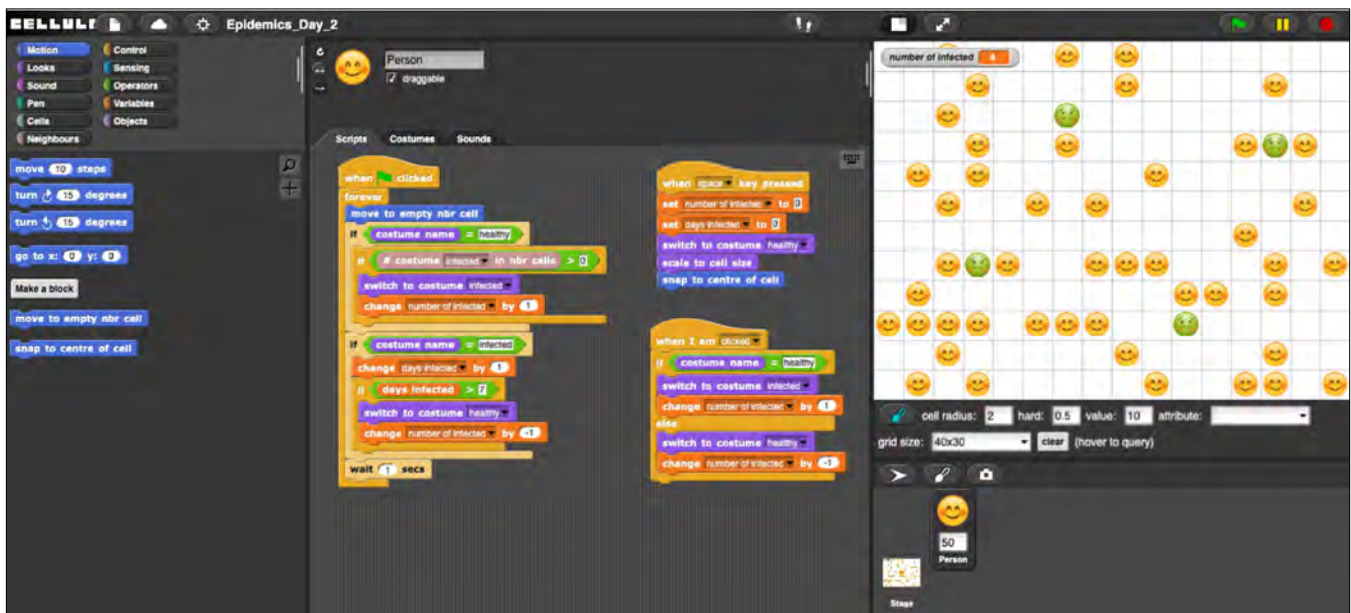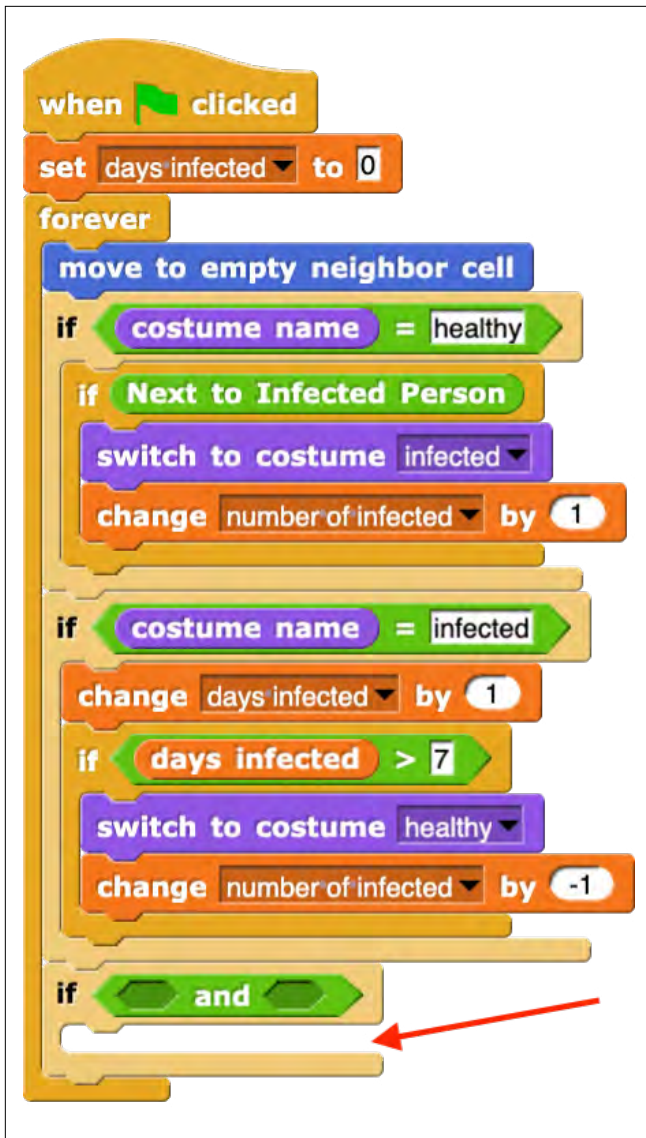

FIGURE 6. Pre-built disease model on day 2 of revised instructional unit.

FIGURE 7. Code with skeleton to be modified by students on day 3 of the revised instruction.

FIGURE 8. Example option of functionality students' can choose to add to their model. In this example, the behavior to be added to the disease model is for hospitals to have a limited number of available treatments.

Day 2 Revised Instruction

The second day of instruction was revised to provide students with a completed computational model to carry out scientific exploration in the "Use" stage of the progression. Learners begin their exploration in the "Use" stage through interaction with a pre-existing model (Figure 6) used to carry out scientific simulations of the disease spread phenomena within the block-based programming environment. This revised instructional activity scaffolds CT learning by demonstrating to students how block-based programming is used to model a scientific phenomenon, growing their confidence in using a computational model to carry out simulated experiments. Students open the Cellular environment to find the pre-built code rather than an empty environment where they must start from scratch (e.g., Figure 3). Moreover, teachers offer an explanation of the blocks contained in the model to help students build an understanding of how scientific concepts are represented in code. Finally, a worksheet was added to the day's activity designed to facilitate data gathering and provide an opportunity for structured reflection about the observable science concepts depicted in the model (shown in Appendix E).

Day 3 Revised Instruction

In the next stage, the "Modify" phase, students begin modifying the computational model with increasing scientific complexity. Following the strategy of Lee et al (2011), students engage in modifications and iterative refinements to the pre-built model from day 2. This day's activities are designed to increase students' ownership of the model. In this case, they may modify the parameters of key variables or modify the underlying logic driving the components of the model. The learning environment is pre-loaded with the existing model with an additional scaffolded skeleton, highlighted in Figure 7 with a red arrow, where students will add functionality. The scaffolded skeleton aids students in understanding where modifications can be made to the model (e.g., code blocks added). Once modifications are completed, students carry out more scientific explorations with the new model. As on day 2 of the revised module, worksheet-based activities were added to the instructional time on day 3. The new worksheet-based activities on this day were designed to gauge students' understanding of the code by asking them to articulate the science shown in blocks of code or choose a set of code blocks that correctly represent a particular scientific concept (see Appendix F).

Day 4 Revised Instruction

In the "Create" stage of the UMC learning progression, students apply their scientific knowledge and their CT knowledge of how these scientific concepts are instantiated in the computational model by creating new code. Depending on the ability and interests of the student, this may be a more extensive modification of the existing model or a brand-new

program that expresses their scientific and CT knowledge in novel ways. Thus, on the final day of our revised module, we presented students with a worksheet (Appendix G) containing possible scientific concepts or behaviors that could be added to the model they used on day 2 and modified on day 3. Students were instructed to choose among these concepts and add their own original code to the model to create a more complex functional program. An example choice is shown in Figure 8. The choice shown is of medium (denoted with orange text) and suggests that students update their epidemic disease model so that hospitals have a limited supply of treatments for infected individuals. It should be noted that these design decisions meant that our curriculum differed somewhat from the learning progression model set forth by Lee et al (2011). The decisions provided scaffolding for those students who needed more support while still offering student choice to maintain learner agency. To facilitate students' code development on this day, worksheets were designed prompting students to both articulate their choice of which science concept they wish to add to the model and to either draw the needed algorithm of blocks or write in pseudocode how they intended to implement this concept in their model (Appendix H).

REFLECTIONS FROM USER EXPERIENCE OF THE USE-MODIFY-CREATE APPROACH

To assess the results of our revised curriculum we worked with teachers at two schools to implement the newly revised activities and materials in twelve different science classrooms. This resulted in 241 diverse students and three of their teachers experiencing the new design. It should be noted one of the teachers implemented the previous version of the curriculum during the prior school year and thus was able to give us comparative feedback from her perspective. As with the implementations of our previous version of the curriculum, our design team took measures to capture data (e.g., classroom observations, student and teacher feedback) that we believed would help us to reflect upon the results of our design changes and inform future design iterations.

All of the teachers commented that the "Use" day was an improved entry point for coding as all students regardless of their prior programming experience were able to successfully engage and meet the day's objectives. In particular, one teacher remarked that she felt it was effective at enabling students to orient themselves with block-based code, "Because it was prebuilt, and they were just changing the variables they were able to see this is how code is put together."

Observations of students engaging with the materials over the course of the four-day curriculum, demonstrated that more students were able to independently build the desired final computational models during the modify and create days, whereas in the previous versions of the materials

students often struggled to finish building a complete model. Feedback from the students also suggests that students felt higher agency and more ownership over the artifacts they produced.

Anecdotal evidence collected from teachers also suggests that the UMC scaffolding sequence has benefits for those that are new to implementing computational modeling in their science classrooms. Just as the curriculum provides incremental scaffolding to students, the progression of difficulty also allows teachers to get used to the programming concepts in a gradual manner, hopefully building their confidence to teach it from lesson to lesson. One teacher's comment is particularly reflective of this sentiment, "Day 1, on the computer, you really understand the beginning part. And then day 2 it builds a little more and you're building the code; you're playing with it. And day 3...at that point, it's just really easy to go through." We also witnessed less direct instruction from the teachers, as they were more apt to let students learn through experimentation. It allowed students to develop unique and varied modeling solutions in comparison to the prior year's implementations.

Reflecting on this design cycle, the design team felt that the Use-Modify-Create (UMC) approach better supported both teachers' and students' needs while still adhering to the design's contextual constraints. As a result of these modifications, we observed that the UMC-informed curricular materials were implemented by teachers more faithfully to the intended design. Therefore, we believe the approach better addressed the complexity of the learning context, thus alleviating two of the previously mentioned design failures. Additionally, the learning sciences and computer science researchers who observed classrooms during delivery of the epidemic disease module noticed that students were able to start adding code blocks more quickly on the "Modify" day because they had a starting point where it was clear that they should fill in the blanks, unlike the initial design's approach where they started with a blank slate. This left the design team feeling as though students were not only more engaged in the activities, but that the desired learning outcomes were better met by this new version of the epidemic disease curriculum module, addressing the remaining observed design shortcoming from the prior cycle.

IMPLICATIONS

Although these implementations suggest the design of this curriculum is evolving to better meet the learning and teaching challenges associated with integrating CT and science practices, they also illuminated new areas for potential improvement. For example, teacher interviews and classroom observations suggest that some students struggled with moving from the "Modify" stage to the "Create" stage with its more open-ended structure. This transition to "Create" proved to still be a large conceptual

and motivational leap for both teachers and students. At least one teacher noted that although having a cheat sheet with sample code solutions was helpful, they were the least comfortable with the "Create" day. Some teachers suggested that increasing the complexity of code modifications on the "Modify" day and adding more scaffolding on the "Create" day might serve to bridge this gap and therefore meet the desired learning outcomes, while others suggested incorporating more open-ended activities earlier in the curriculum. Therefore, future considerations for our materials will focus on identifying curricular supports to successfully bridge the gap between the "Modify" and "Create" stages of the progression as well as increasing both student and teacher comfort with the open-ended nature of the "Create" stage. Following the recommendations of our piloting teachers, modifications to our materials will be made to increase the complexity of code changes required during the "Modify" stage to aid in a more balanced transition to the "Create" stage.

We also still found that our modified curricular materials did not do enough to address the diverse range of student ability levels in the classroom. Although our new pedagogical design enabled a greater involvement of students who lacked prior programming and CT experience, we found that on the "Use" and "Modify" days more advanced students completed the activities in a much shorter amount of time than their peers. Thus, future iterations of the materials will also include options for extension activities to keep those students from getting bored.

Although the solution to many of these continuing challenges could be addressed, in part, by more instructional days, we recognize that this is a design constraint we should embrace and attempt to meet the challenge of creating even more engaging, adaptive, and efficient materials. Additionally, we will continue to explore ways to increase professional development support for teachers as well as a greater opportunity to involve teachers in the design process so that activities better reflect differentiated teacher and student needs in classrooms.

ACKNOWLEDGMENTS

REFERENCES

de Jong, T., Linn, M. C., & Zacharia, Z. C. (2013). Physical and virtual laboratories in science and engineering education. *Science, 340*(6130), 305-308. https://doi.org/10.1126/science.1230579

Gray, C. M. (2020). Markers of Quality in Design Precedent. *International Journal of Designs for Learning, 11*(3), 1-12. https://doi.org/10.14434/ijdl.v11i3.31193

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Lytle, N., Cateté, V., Boulden, D., Dong, Y., Houchins, J., Milliken, A., Isvik, A., Bounajim, D., Wiebe, E., & Barnes, T. (July, 2019). Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes. [Paper presentation]. 24th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2019). Aberdeen, Scotland UK. ACM. https://doi.org/10.1145/3304221.3319786

Monash Cellular. (n.d.). Monash BlockBooks - Interactive e-Books for Generative Art, Simulation and Robotics. Retrieved November 9, 2020, from http://www.flipt.org/#cellular

NGSS Lead States. (2013). *Next generation science standards*. Washington, DC: National Academies Press.

Settle, A., Goldberg, D. S., & Barr, V. (2013, July 1-3). Beyond computer science: computational thinking across disciplines. [Paper presentation]. 18th ACM Conference on Innovation and Technology in Computer Science Education, Canterbury England, UK.

Shapiro, L. (2019). *Embodied cognition*. London: Routledge.

Smith, K.M. (2010). Producing the Rigorous Design Case. *International Journal of Designs for Learning,1*(1),9-20. https://doi.org/10.14434/ijdl.v1i1.917

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Education and Technology, 25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24-28 https://doi.org/10.1145/2633031

Wilensky, U., & Rand, W. (2015). *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*: MIT Press.

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, 24*(2), 171-209. https://doi.org/10.1207/s1532690xci2402_1

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Small Sample of Teacher Guides Provided in Instructional Materials

Epidemic Disease Activity - Teacher Guide

Day 1

Goals

Understand the logic behind a computational model of the spread of an epidemic disease.

Expected Science Outcomes

Definition of Disease and Epidemics

Definition of Agent

Definition of Host

States that a Host can be in (Healthy, Infected/Sick)

Method of transmission of agent to host (direct contact)

Metamodeling: use of symbols to represent Host states, limitations of model to represent modes of transmission

Expected Computational Thinking Outcomes

Students will use abstraction to model real life factors

Students will understand that objects/agents have shared properties with different values

Instruction Guide

Part 1: Introduction through whole class discussion

What we are studying:

- Epidemiology – the study of disease in populations
- Epidemics – the rapid spread of disease through a population
- Typically it is a viral or bacterial disease in human populations that is of interest

Human

Can be Infected? = _____

Can Move? = _____

Current State = Healthy / Infected

Image:

Rules

If healthy:
and touching healthy Human:
_____

and touching infected Human:
_____

If infected:
and touching healthy Human:
_____

and touching infected Human:
_____

Human Rules Pt 2

If healthy:
and touching Hospital:
_____

If infected:
and touching Hospital:
_____

Hospital

Can be Infected? = _____

Can Move? = _____

Image:

Finite State Machine Activity Worksheet

Name: Healthy

Costume : _____

Duration: _____

Can walk?:_____

Name: Infected/
Contagious

Costume :_____

Duration:_____

Can walk?:_____

Name: Dead

Costume : _____

Duration: _____

Can walk?: _____

Name: Infected/
Not Contagious

Costume : _____

Duration: _____

Can walk?: _____

Day 5 Scientific Predictions Activity Worksheet

Directions: Setup an experiment to answer one of the following research questions.
1. How does the rate of infection change depending on Density?
2. How does the rate of infection change depending on Temperature?
3. How does the rate of infection change depending on Humidity?
4. How does the rate of infection change depending on Temperature and Humidity?

Question # ____   Hypothesis:_____

Use the table below to answer your research questions and help run your experiment.

Condition	Number Initially Infected	Sprite Density (Sprites/Total # of Cells)	Temperature (High or Low)	Humidity (High of Low)	Time to fully infected (s)	Average Time to fully infected (s)
1: No modifications, Original Setup (Environment is Low Temperature and Low Humidity)						
Trial Run 1	1	50/192	Low	Low		
Trial Run 2	1	50/192	Low	Low		
Trial Run 3		/				
2:						
Trial Run 1		/				
Trial Run 2		/				
Trial Run 3		/				
3:						
Trial Run 1		/				
Trial Run 2		/				
Trial Run 3		/				
4:						
Trial Run 1		/				
Trial Run 2		/				
Trial Run 3		/				

Conclusion: _____

Name: _____ Class: _____

Press the space bar to reset the system.

1. **Fill in the Table**. You will click on the number of sprites indicated in the first column to infect them. Then you will run the epidemic simulation and plot the Number of Infected. Once the disease reaches its peak and sprites start recovering, stop the simulation by clicking the stop sign and record the time it took and the number of sprites infected.

Here's an example plot and how to tell the number of infected and the time. The number of sprites in our example population who got infected was **43** and it took about **5 secs** for the disease to peak.
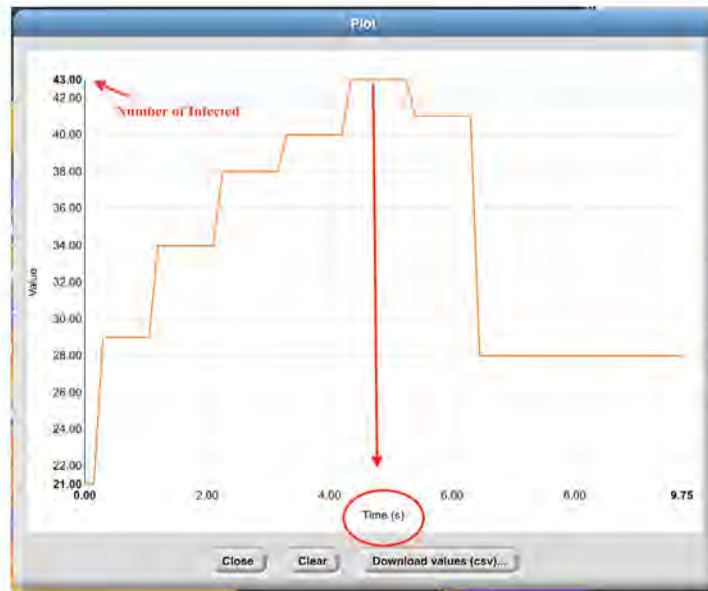
Initial Number of Infected	Time for Disease to Peak	Final Number of Infected
1		
7		
12		

2. Now change the number of days it takes to recover from the disease and run the simulation starting with only one infected sprite each time.

Days to recover	Time for Disease to Peak	Final Number of Infected
7		
3		
1		
You pick a number of days to recover and input it here:		

Questions

1. What patterns do you notice about how quickly a disease spreads based on how many people are initially infected?

2. How does recovery time affect the spread of a disease? Did more or less people become infected? Did it take a shorter or longer period of time for people to become infected?

Sample of "Modify" Revised Instructional Activity Worksheet

Name: _____ Class: _____

Match the Scientific Phenomenon with Code Blocks. Read the scientific behavior listed and choose the set of code blocks that would simulate that behavior in the Cellular Programming Environment. (*Hint*: You can input the set of code blocks and observe the simulation run to confirm they simulate the desired behavior.)

If I'm an infected person and I am near to a hospital, then I can get treatment and recover 3 days faster. Circle the code blocks that would simulate this phenomenon below.

Describe the Code. Read the code blocks below and describe what the code is doing in the space provided beside it. (*Hint*: You can input the set of code blocks and observe the behavior as the simulation runs.)

Sample of Creation Choices for Adding Scientific Concepts

Activity Description

Plugged Activity: Today you will write your own code using the blocks available in the Cellular environment to add your own creations.

Option – Mobile Hospital services (ambulance)
Add functionality to the **hospital** code to switch costume when clicked (hint: see person code). When the green flag is clicked if the hospital looks like an ambulance, the hospital should move to an empty cell.

Option – Introduce Mortality
If left untreated some diseases can lead to death. Update the **person** code such that if 7 days have passed and the person has been treated, they become healthy, otherwise they die.

Option – Game of Chance
Just because someone sneezes next to you doesn't mean you'll automatically catch a cold. Update the **Person** code to include chance. In this example you'll be rolling a dice to see whether or not someone will become infected.

Activity Description

Plugged Activity: Today you will write your own code using the blocks available in the Cellular environment to add your own creations.

Option - Vaccines, illness prevention

Sometimes, healthy people can visit the hospital to receive a vaccine prior to catching an illness. Modify the near hospital portion of the **person** code to allow healthy people to receive treatment and become vaccinated.
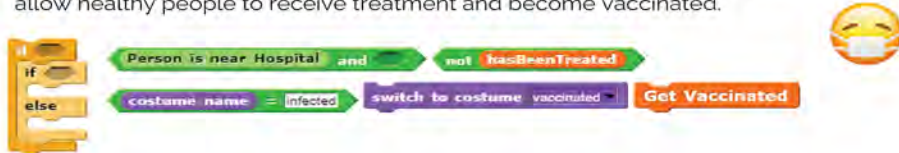
Option - Limited Number of Treatments

Update the **hospital** code to set a limited number of treatments. When the green flag is clicked, the program should always check to see if there are any treatments left. If there are no more treatments left, then set treatments available to false.

To prevent someone from getting too much medicine, set a variable to tell if they have been treated. Now in the main script, update the hospital condition to only treat those who have not been treated.

Name: _____   Class: _____

Activity Description

Plugged Activity: Today you will write your own code using blocks available in the Cellular environment to add in your own extensions to the epidemic simulation

What is the first creation option that you will be trying to code? #_____

Reading the task description, in which sprites (e.g. person, hospital) do you need to add/modify code?

Plan your code here. (You can write out what you will add or draw pictures of the code block(s) you might use.)

After adding your blocks and running your code, how did this affect the model? Why?