Enhancing Future K-8 Teachers' Computational Thinking Skills through Modeling and Simulations

Abstract

It is now highly recommended for teachers to incorporate computational thinking (CT) into their science classes. Our research modifies the existing structure of a science methods course for preservice teachers to include CT via modeling and simulations. In the first study, preservice teachers were introduced to the basics of coding through an Hour of Code tutorial, followed by an exercise where they programmed an animated model of the solar system using Scratch. In the second study, we created a web-based simulation to visualize Newton's second law of motion (F=ma) with a dynamic graph feature. The simulation is a race between two cars with interactive settings that the user can change, such as, changing the mass and force of each car. Results from both studies reveal that after completing the exercises, preservice teachers learned the material effectively, felt that CT exercises would be beneficial in K-8 education, and plan to incorporate CT into their future classrooms.

Keywords: Computational Thinking, Scratch, Simulations, Coding, Computer Model

Introduction

The Science and Engineering Practices within the Next Generation of Science Standards (NGSS) framework includes using math and computational thinking (CT) as one of the eight practices for science educators. Many teachers are unsure what CT is and how teachers can promote this practice into their science teaching. It is important to provide teachers with knowledge of CT so they can use it successfully in their future classes (Yadav et al. 2017; Yadav et al. 2014).

Computational thinking (CT) is no longer only for computer scientists, but for everyone (Wing 2006). The challenge is that this learning is best done in early years of education to ensure a solid foundation (Wing 2008). CT is a fundamental skill like reading, writing, and arithmetic (National Research Council 2010). The Computer Science for All initiative encourages exposing children in K-12 to computational thinking and coding. Computer scientists and educators need to combine their skills to effectively communicate computing to younger generations. Computing education research will be most effective when multiple disciplines work together (Guzdial 2008).

Our research involves faculty from the Departments of Computer Science and Teacher Education who teamed up to embed CT and coding into a Science Methods course for preservice elementary and middle school teachers. We discuss two studies done in the Science Methods course. In the first, preservice teachers coded a model of the solar system using Scratch programming. In the second study, we introduce a custom-developed web-based simulation of a car race to teach Newton's Second Law of Motion. The goal of this work is to incorporate CT into our science training for preservice teachers, in order for them to appreciate how to effectively embed CT into their future classrooms.

Background Literature

Defining Computational Thinking (CT)

Many researchers have tried to define CT, yet there is no clear-cut definition (Voogt et al. 2015). Voogt et al. (2015) state that rather than trying to define CT we should look for similarities in the discussions of CT which can lead us to a description of CT and how to apply it in K – 12. According to Wing (2006), "computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." Aho (2012) defines CT as "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms."

The International Society for Technology in Education (ISTE) and the American Computer Science Teachers Association (CSTA) have defined CT as a problem solving process that includes the following (Barr et al. 2011):

- formulating problems in a way that enables us to use computers and other tools to help solve them;
- logically organizing and analyzing data;
- representing data through abstractions such as models and simulations;
- automating solutions through algorithmic thinking (a series of ordered steps);
- identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and
- generalizing and transferring this problem solving process to a wide variety of problems

These CT characteristics provide a foundation into understanding what CT is and how it can be used. Weintrop et al. (2016) proposes the CT-STEM taxonomy which can be broken down into four major categories: Data and Information skills, Modeling and Simulation skills,

Computational Problem Solving Skills, and Systems Thinking Skills. Model building is important because it promotes inquiry, conceptual changes, and representational literacy (Brennan and Resnick 2012). One type of model building is using computer programming to construct models that enhance students' CT skills and conceptual understanding. As Weintrop et al. (2016) point out, NGSS and Common Core Mathematics now require students to create models in addition to using them.

Our research focuses on modeling and simulations based on Weintrop's CT-STEM taxonomy and having preservice teachers code a model of a solar system. Furthermore, we use a custom-developed interactive simulation where preservice teachers are able to manipulate parameters as they learn about Newton's Second Law of Motion.

Incorporating CT into Science through Modeling and Simulations

Barr and Stephenson (2011) discuss how a good definition of CT should incorporate examples of how it can be used in the classroom. Many science concepts are abstract and require mental or physical modeling to help students visualize abstract concepts, such as atoms, energy, force, solar system, photosynthesis, and more. Modeling enables us to view a real world activity by simplifying certain characteristics of that phenomenon (Weintrop et al. 2016).

Computational models can be simulated on the computer. Many believe that these interactive simulations are at the focal point of CT (National Research Council 2011). Simulations are effective CT tools as they provide a method for students to test predictions, record observations, and view scientific phenomena in a realistic timeframe (National Research Council 2011). For example, with simulations students can visualize chemical interactions or visualize force and motion. "Simulations that allow students to use computational thinking are

not simply animations, they are dynamic computer models that involve students in changing conditions and observing new outcomes" (Sneider et al. 2014). The Physics Education Technology (PhET) project is one example of a web-based simulation environment which many use and modify parameters to demonstrate STEM content (Weintrop et al. 2016; Sneider et al. 2014).

Many researchers have studied the effect of using simulations to provide students with a way to understand complex systems (De Jong and Van Joolingen 1998). As Foley (2012) state, "Computer simulations, particularly animated or depictive simulations, are a particularly powerful way to articulate a scientific model." Bell and Trundle (2008) used computer simulations for preservice teachers to understand moon phases. They found that a well-designed computer simulation within a conceptual change model can be very effective in promoting scientific understandings for preservice teachers.

Students can also design and create models of their own (Weintrop et al. 2016). In science, a core computational thinking concept, abstraction, can be demonstrated through model building (Barr and Stephenson 2011). While interacting with a simulation is a great way for students to visualize and make predictions about what they are learning, having the ability to modify and create code allows students to create their own models to explore other phenomena (Foley 2012). While CT is more than programming, programming is an important tool which can be used to advance CT skills (Voogt et al. 2015). Scratch is an example of one programming environment which can be used effectively for model building.

Scratch Programming

Scratch (scratch.mit.edu) is a visual programming environment that allows K-12 students to create stories, games, animations, and interactive presentations. It uses a drag-and-drop approach (rather than typing code) that allows novice students to build programs by snapping together visual blocks that control the actions of different dynamic actors (objects) on a computer screen (Grover and Pea 2013; Resnick et al. 2009; Sengupta et al. 2013). Most Scratch users are between the ages of 8 and 16, peaking at 12 (Resnick et al. 2009).

Scratch has been very successful in introducing math and science concepts to elementary and middle school students. Foley (2012) introduced Scratch programming to middle school students by having them code a ball drop exercise to use model building while learning about motion, such as position, velocity, acceleration, and force. While they found that their students needed more computational thinking skills before being ready for that exercise, other studies have shown that students have reported having a positive experience in programming when using Scratch (Fesakis and Serafeim 2009; Smith and Neumann 2014).

Maloney et al. (2008) introduced several core programming concepts in Scratch such as conditional logic, iterative and parallel thinking, and data abstraction to urban youth groups ranging in age from 8-18 in an afterschool environment. They found that students were very engaged with Scratch.

Purpose of our Studies

We present two studies done to examine using CT via modeling and a simulation in a science methods class for preservice K-8 teachers. In our first study, we use Scratch to remediate common misconceptions in the solar system. Preservice teachers programmed a

simple computer model of the solar system in order to develop their computational thinking skills while improving their science content knowledge. Our second study used a custom-developed web-based simulation to allow the preservice teachers to modify parameters, test theories, and visualize Newton's Second Law of Motion via a race between two cars.

Research Questions

1. Will using computational thinking (via modeling and simulations) be an effective way for preservice teachers to learn science content?
2. After incorporating computational thinking in their exercises, will preservice teachers want to use this skill in their future classrooms?

Study 1: Solar System via Scratch Programming

Methodology

We designed a project to improve preservice K-8 teachers' reasoning skills and CT using computer programming. The tasks consisted of coding exercises from an Hour of Code tutorial as well as using Scratch programming to model the earth's counter-clockwise rotation around the sun.

Participants

We tested this work on 32 preservice K-8 teachers (26 female and 6 male) at a Midwestern university taking a science methods course: 19 were graduate students and 13 were undergraduate students. We incorporated both a pre- and post-questionnaire with questions pertaining to any prior misconceptions on the solar system as well as their prior experience in

programming and their views on using coding as a teaching technique (see Appendix). Five of the preservice teachers were used initially as a pilot study and received a post-test only.

Procedures

Part 1: Hour of Code

In order to teach basic programming techniques, we used 20 exercises from the Hour of Code (Code.org http://studio.code.org/hoc/1). The 'Hour of Code' is a nationwide initiative by CSEdWeek and code.org to introduce computer programming to students. We used this tool to teach preservice teachers the basic logic behind programming concepts, such as breaking a problem into steps and using loops and conditionals. We required them to complete at least the first 12 exercises in this tutorial and many completed all 20. Commands such as move forward, turn left or right, loops, and conditionals are utilized. Figure 1 shows the first exercise in the Hour of Code tutorial where the goal is for the angry bird to catch the pig.

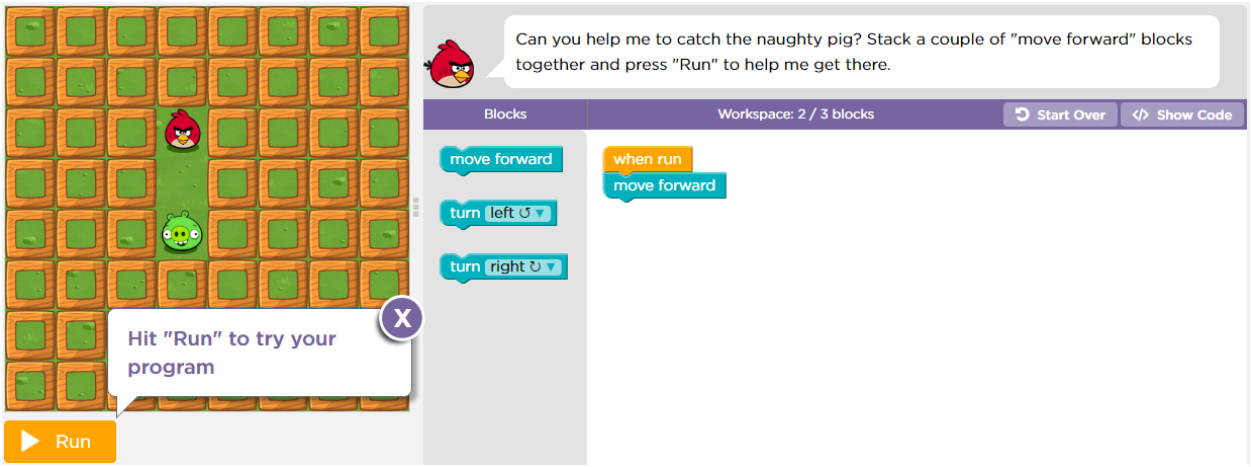

Figure 1. Hour of Code Tutorial to Introduce Basic Programming

Part 2: Solar System Model using Scratch

After completing the code.org exercises, the preservice teachers learned that the earth revolves around the sun counterclockwise by showing them a Scratch programming project of a solar system using programming with loops and animation. The preservice teachers saw the motion of the earth around the sun, were shown the code, and were then asked to code the solar system themselves. By creating these programs, they were able to learn science concepts by programming and visualizing the solar system in motion (see Figure 2). At the completion of the project, they were given time to add to the project, such as adding more planets or other objects, as well as try out other features or modify angles to enhance their model and make it unique.

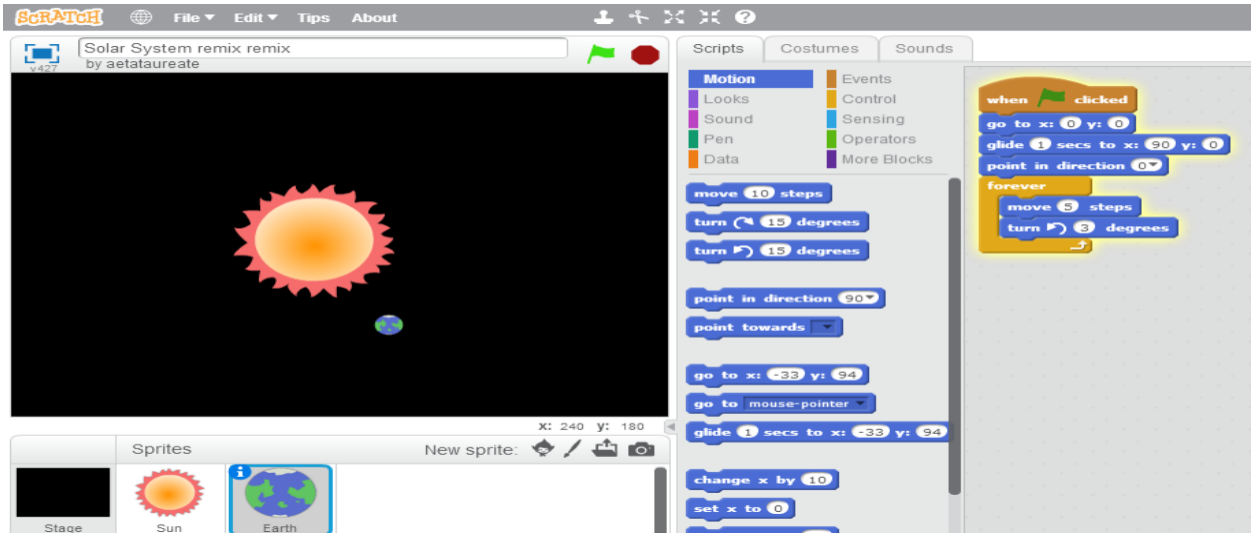

Figure 2. Using Scratch to Model the Solar System

Results on the Solar System Programming

According to the pre-questionnaire, only six out of the 32 preservice teachers (19%) had previously tried programming and seven (22%) said they had previously thought about using programming in their future classes. One wrote "I have never considered using programming

because I don't really know what it is or how to use it" and another response regarding whether she had previously tried programming, "no. I am scared."

After completing the experiment, 29 preservice teachers (91%) felt that programming can be a good educational tool. Below are two examples:

-"I think it would be a very engaging resource for students and could be incorporated into many different subjects. It wouldn't have to be just science or math. You could create stories for Reading/LA or animate different historical events in social studies. While some kids might not catch on as quickly as others, I still think that all the students would enjoy using the program."

-"Programming can teach students to think critically and carefully about the steps they need to reach their goal. Programming can enable students to be creative and direct their learning. This is a great interactive way to engage students in a science lesson."

Thirty preservice teachers (94%) felt that programming can help in terms of critical thinking and modifying misconceptions. The other two were not sure. When asked about their previous misconceptions on the solar systems rotation, 18 (56.25%) said this experiment helped clear any misconceptions they had compared to 12 (37.5%) who did not have any previous misconceptions. Two responses were unclear (6.25%).

We also asked them what challenges they foresaw when incorporating programming into their teaching. The four main challenges include:

- *Challenging* - It can be challenging for the teachers and/or students to learn.
- *Frustration* – Students may become frustrated with details of code implementation.
- *Time* – Teachers would need to dedicate more preparation time.
- *Access to computers* – Some schools do not have access to computers that can be used.

Study 2 Newton's Laws of Motion with Simulations

Methodology

Together with Computer Science undergraduate students, we created a web-based simulation to help teach Newton's Second Law of Motion in the preservice science methods course. We interviewed middle school teachers and chose the topic of a car race as one that would be very engaging to middle school students. As shown in Figures 3 and 4, users have the option to choose different values for force (F) and mass (M) and the acceleration (A) is computed accordingly. Depending on the mass selected the image of the car changes into larger/smaller size vehicles accordingly. A graph depicting the distance and time it takes for the car to travel is dynamically drawn to the screen as the cars are in motion. There is also an option to switch the graph window into a quiz which guides them with questions on Newton's Second Law of Motion and they can interact with the simulation to help answer the questions (see Figure 5).

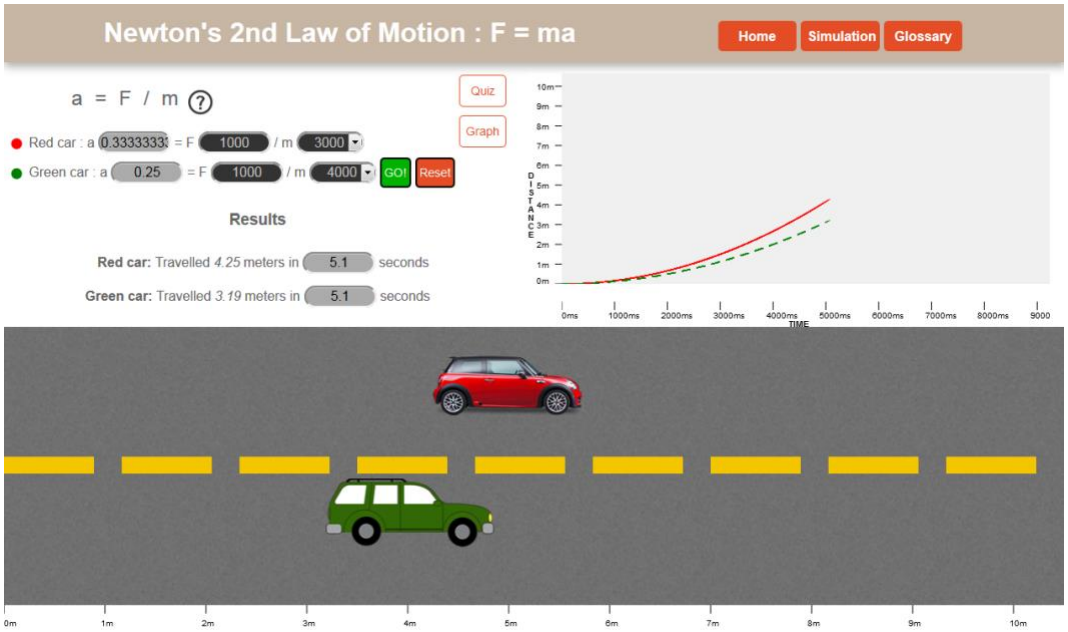

Figure 3. Simulation with Mass Manipulated

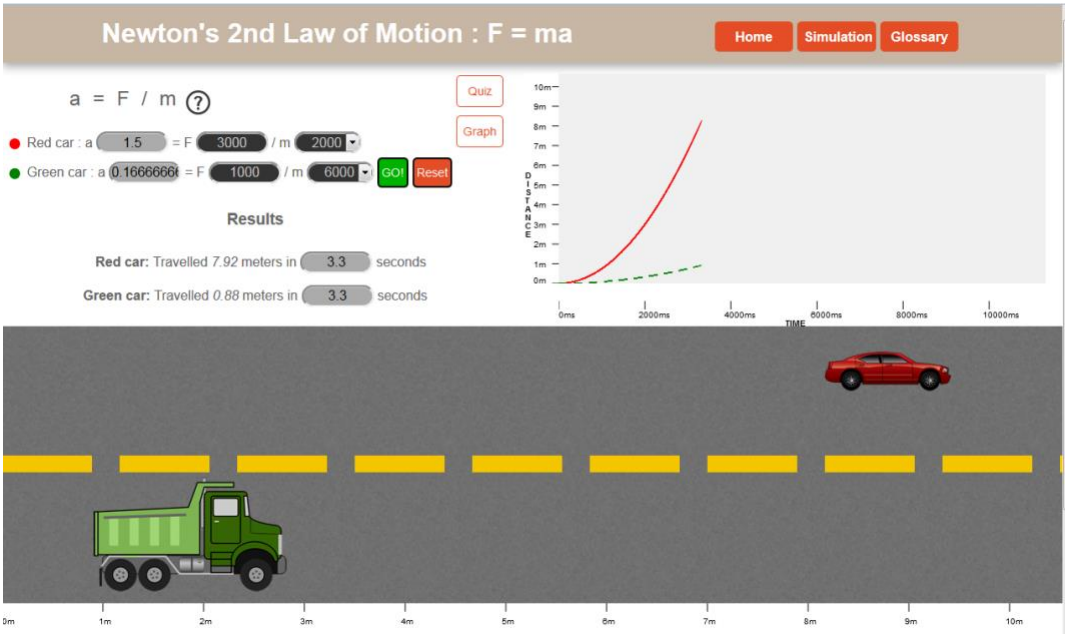

Figure 4. Simulation with Mass and Force Manipulated

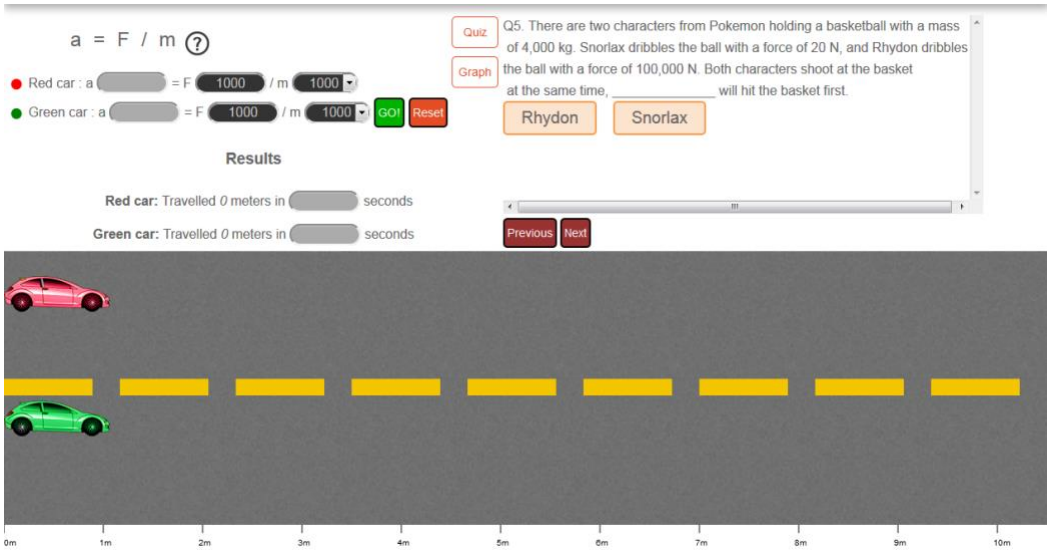

Figure 5. Simulation with Quiz

We tested the website on 20 preservice K-8 teachers (17 female and 3 male) in a Science Methods course. A pre-questionnaire was provided, which contained questions regarding their previous computational thinking and simulation experience as well as questions on the content such as calculating force and acceleration.

After completing the questionnaire, the preservice teachers were told to play around with the simulation and partake in the 12 question quiz on the website. Then they were administered the post-questionnaire, which contained questions relating to their experience as well as the science content. See Appendix for the pre- and post-questionnaire.

Results on the Newton's Laws of Motion with Simulations

After completing the simulation, 18/20 preservice teachers (90%) felt that computational thinking can be a good educational tool. Below are some examples of student responses:

- *"I really like it because it can be a different way for some students to learn. Specifically if they are very pattern focused."*

- *"I think it's an important tool to use to approach teaching in a different way"*
- *"It is very interesting and helpful for visual learners"*
- *"It is a great way to engage students in using different tools to solve problems (science and math)."*
- *"I think it will be a great resource. Allowing students the opportunity to break apart the problem to look at individual features is a great way to help them understand step by step."*

When asked whether they would incorporate this kind of simulation into their future teaching, 18/20 (90%) said they would (one student wrote maybe and one wrote probably not since she was focusing on younger children who would not be learning physics). Below are some examples of student responses:

- *"Yes, It is one thing to teach the 2nd law of motion and another to actually see it or create it yourself"*
- *"i definitely would. The idea of using something that's hands on is always more beneficial for the students' learning"*
- *"For sure! I think that teaching Newton's law can be tricky just with plain theory, but with a simulation it gets way easier."*
- *"I would certainly like to incorporate this kind of simulation into my future teaching. This would be a great activity to incorporate on a lesson on Newton's second law of motion (especially the car race!)."*

Results of the post-questionnaire showed that after completing the simulation preservice teachers understood Newton's Second Law, learned how CT can be beneficial in education, and wanted to incorporate it into their own future classrooms. We also asked about any challenges they foresaw in implementing this in their future classrooms and a significant concern is funding and availability of computers and other technology in the schools.

Discussion

With the goal of Computer Science for All, there is a big push to incorporate computational thinking in k-12 classrooms. There are many concerns among teachers and researchers on understanding what CT is and how to effectively integrate it into a school curriculum, which is already packed with content teachers need to cover (Grover and Pea 2013).

Our research discusses the integration of CT into a science methods course for preservice K-8 teachers through modeling and simulations. We focus on two studies, one having preservice teachers model a solar system with Scratch, a visual programming environment, and the second using a web-based simulation to evaluate different scenarios when learning Newton's Second Law of Motion. We chose these methods as simulations can foster CT by allowing users to modify parameters and test predictions (Sneider et al. 2014; National Research Council 2011), while incorporating coding goes beyond a simulation by creating models of different scientific phenomena (Foley 2012).

Our research questions were whether using CT (via modeling and simulations) would be an effective method for learning science content and whether after incorporating CT in their exercises, preservice teachers would want to use these skill in their future classrooms. The results of both our studies reveal that preservice teachers were very engaged in our classroom

exercises, learned the material effectively, and plan to implement similar exercises in their own future teaching.

In the first study, over 90% of the participants felt that incorporating programming was a great education tool and can help students modify any misconceptions. The preservice teachers had some ideas on how they would incorporate it into their own classrooms. Some examples include incorporating Scratch to teach a lesson on velocity and energy. Some students even mentioned examples of applying Scratch outside a science classroom. Examples include:

"It can definitely be used to create a story in language arts. Students would enjoy using the coding system for this."

"I could have students create a math game that involves the sprite asking math questions, and reacting in a certain way if the student is wrong."

In the second study, 90% of the preservice teachers felt that computational thinking and simulations are great additions in the classroom. They also mentioned other concepts that could be effective with simulations in the classroom. For example:

"You may simulate many of Newton's ideas. For example, you may show that a body in motion will remain in motion unless it is stopped by an outside force. You may also show the opposite. That a body at rest will remain at rest unless it is moved by an object. An idea for this simulation would be to have students play billiards at a pool table."

"I think many concepts in physical science would be great ideas for projects like this one, as I think students sometimes find many of these concepts to be abstract or difficult to understand. For instance, chemical reactions and other topics related to motion and force may be good topics for such a project. Also, cellular process such as photosynthesis, cellular respiration, meiosis, mitosis, and diffusion are also potential topics for an interactive website, as

these are also topics that are fascinating, but which students may have difficulty understanding due to their complexity."

Computational thinking is now a requirement for science teachers and incorporating it into our Science Methods course was a valuable way for preservice teachers to see how they could seamlessly integrate CT when they become teachers. Similar to Fesakis and Serafeim (2009), our research reported having a very positive experience programming with Scratch. There are several benefits to incorporating programming and simulations in the classroom. Bell and Trundle (2008) found that simulations are very effective in promoting scientific understandings. Many students can be visual learners, and allowing them to visualize what they are learning can help them remember it. Furthermore, using simulations allow students to be actively engaged in what they are learning, as they are able to get involved, modify parameters, and make predictions while learning the content. Programming can be useful to allow students to create their own models of various phenomena. Learning how to break a problem down into simple steps is a useful skill when figuring out solutions to problems.

One major challenge we foresee are the resources necessary for implementing more technology in the classroom. While some schools may not currently have the budget to allow for all the resources, using CT is a way of the future and will benefit all students. Our studies focus solely on Internet-based computational thinking resources (simulation and Scratch) as more schools now have computers, laptops, iPads, or Chromebooks. Future research can explore tools, such as CS Unplugged, where students do not even need a computer to use CT in the classroom.

Limitations and Future Research

While the results of our studies show that preservice teachers had a positive attitude towards using computational thinking and plan on integrating computational thinking in their future classrooms, a longer-term study can determine whether they actually use computational thinking, simulations, and coding in their future classrooms. Another limitation of this study is that there was no control group. While we saw very positive results from our observations in the classroom and their survey and post-test results, a control group would enable us to have a quantifiable comparison group to examine any performance differences in learning the material. Furthermore, a larger sample size would allow us to report other differences, such as with gender, on using CT in the classroom. In our studies we had few males in the classroom as a result of the education course having more females. Finally, rather than only modifying the science methods course, future work can integrate CT into multiple science and math courses for educators to provide preservice teachers with a richer CT experience that they can use in their future classrooms.

Conclusion

The goal of this research is to incorporate computational thinking into the curriculum for preservice elementary and middle school science teachers, through model building and simulations. We modified a science methods course for preservice teachers to demonstrate how this can be done using Scratch programming and a custom-developed web-based simulation. In this way, these future teachers will identify ways to incorporate CT into their own future K-8 classrooms. Results from both studies show that CT can be incorporated into a science class effectively and be very useful in engaging students in the classroom.

References

Aho, A. V. (2012). Computation and Computational Thinking. *Computer Journal, 55*(7), 832-835, doi:10.1093/comjnl/bxs074.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54, doi:10.1145/1929887.1929905.

Bell, R. L., & Trundle, K. C. (2008). The use of a computer simulation to promote scientific conceptions of moon phases. *Journal of Research in Science Teaching, 45*(3), 346-372, doi:10.1002/tea.20227.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper presented at the American Education Researcher Association, Vancouver, Canada.

De Jong, T., & Van Joolingen, W. R. (1998). Scientific Discovery Learning with Computer Simulations of Conceptual Domains. *Review of Educational Research, 68*(2), 179-201, doi:doi:10.3102/00346543068002179.

Fesakis, G., & Serafeim, K. (2009). *Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education.* Paper presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France.

Foley, B. (2012). *Students' construction of science simulations: "Is that real enough?".* Paper presented at the American Education Research Association (AERA 2012), Vancouver, British Columbia, Canada, April 2012.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12. *Educational Researcher, 42*(1), 38-43, doi:doi:10.3102/0013189X12463051.

Guzdial, M. (2008). Education: Paving the way for computational thinking. *Commun. ACM, 51*(8), 25-27, doi:10.1145/1378704.1378713.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. Programming by choice: urban youth learning programming with scratch. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education, Portland, OR, USA, 2008* (pp. 367-371). 1352260: ACM. doi:10.1145/1352135.1352260.

National Research Council (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press.

National Research Council (2011). *Report of a workshop of pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Commun. ACM, 52*(11), 60-67, doi:10.1145/1592761.1592779.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380, doi:10.1007/s10639-012-9240-x.

Smith, C. P., & Neumann, M. D. (2014). Scratch it out! Enhancing Geometrical Understanding. *Teaching Children Mathematics, 21*(3), 185-188, doi:10.5951/teacchilmath.21.3.0185.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the Science Framework and NGSS: Computational Thinking in the Science Classroom. *Science Scope. National Science Teachers Association.*, 10-15.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715-728, doi:10.1007/s10639-015-9412-6.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147, doi:10.1007/s10956-015-9581-5.

Wing, J. M. (2006). Computational thinking. *Commun. ACM, 49*(3), 33-35, doi:10.1145/1118178.1118215.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725, doi:10.1098/rsta.2008.0118.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *Trans. Comput. Educ., 14*(1), 1-16, doi:10.1145/2576872.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational Thinking for Teacher Education. *Communications of the ACM, 60*(4), 55-62.

Appendix

Pre-Questionnaire for Study 1 - with Scratch Programming

1) Have you ever tried programming in your personal life? Tell us your experiences!

2) Have you ever thought about using programming to create science projects?

3) What is your conception about incorporating programming in your science classes?

4) Have you ever heard of or used Scratch?

5) On a scale from 1 to 5, how likely are you to incorporate programming when teaching :

6) Describe the Earth's rotation around the sun. Including the direction.

Post-Questionnaire for Study 1 - with Scratch Programming

1) You created a mini science project using the scratch today. Now, how do you perceive Programming as an educational tool?

2) What challenges do you foresee in incorporating programming into your teaching?

3) Do you think programming can improve students' critical thinking or modify any misconceptions?

4) Do you have any future project ideas using Scratch programming for your teaching?

5) On a scale from 1 to 5, how likely are you to incorporate programming when teaching:

6) Did you have any misconception about the Earth's rotation around the sun before this activity? What about now?

Pre-Questionnaire for Study 2 – Simulation of Newton's Second Law of Motion

1) Have you ever heard the term "computational thinking" in education? If yes, explain your experience.

2) Have you ever thought about using computational thinking to create science projects?

3) Do you think you will incorporate computational thinking in your science class? In what way?

4) Have you ever tried web-based simulations or animations to learn science concepts? Please explain.

5) Have you ever tried graphing tools (software, web-based, etc.)? Please explain.

6) How likely are you to incorporate computational thinking when teaching:

7) Have you learned about Newton's Second Law of Motion? Please explain.

8) If Force is 20N and Mass is 5kg, what is the Acceleration?

9) If Acceleration is 5 and Mass is 5kg, what is the Force?

Post-Questionnaire for Study 2 - Simulation of Newton's Second Law of Motion

1) After today's class, what do you think of using computational thinking as an educational tool?

2) What challenges do you foresee in incorporating computational thinking into your future teaching?

3) Do you think using simulations can improve students' critical thinking or modify any misconceptions?

4) Would you incorporate this kind of simulation into your future teaching? Please explain.

5) How likely are you to incorporate computational thinking when teaching:

6) Did you have any misconception about Newton's Second Law of Motion before this activity? What about now?

7) If Force is 20N and Mass is 5kg, what is the Acceleration?

8) If the Acceleration is 5 and Mass is 5kg, what is the Force?

9) Do you have any future project ideas where a simulation/animation/graph/quizzes can help students understand abstract and/or difficult science concepts?