Building Blocks: Kids Designing Scientific, Domain-Specific, Block-Based, Agent-Based Microworlds

Kit Martin, Northwestern University, kitmartin@u.northwestern.edu
Connor Bain, Northwestern University, connorbain@u.northwestern.edu
Hillary Swanson, Utah State University, hillary.swanson@usu.edu
Michael Horn, Northwestern University, michael-horn@northwestern.edu
Uri Wilensky, Northwestern University, uri@northwestern.edu

Abstract: Next Generation Science Standards foreground science practices as important goals of science education. In this paper, we discuss the design of block-based modeling environments for learning experiences that ask students to actively explore complex systems via computer programming. Specifically, we discuss the implications of the design and selection of the types of blocks given to learners in these environments and how they may affect students' thinking about the process of modeling and theorizing. We conclude with a discussion of some preliminary findings in this design based research to inform design principles for block-based programming of science phenomena as a medium for learning to build theory.

Introduction

Research shows it is important for students to engage in authentic science practices in the classroom (Schweingruber et al., 2013). The construction of scientific theory is a key practice of science. Previous research has investigated student engagement in theory building through agent-based computational modeling (Wilensky, 2003; Wilensky & Reisman, 2006). Modeling asks students to formalize their theories in the form of a computer program. It also allows students to implement and test their theory by comparing the output of a model with the outcomes they expect. The creation, refinement, and application of models of phenomena is central to the work of modern-day scientists and mathematicians (NGSS Lead States, 2013; Weintrop et al, 2015). By integrating these practices into curricula, we allow students the opportunity to engage in theorizing, computational thinking, and authentic scientific exploration. We are designing modeling environments that lower the threshold for participation in elements of scientific theory building by integrating the computational power of NetLogo (Wilensky, 1999) with the accessibility of blocks-based modeling languages. Our modeling environments use a block-based interface to NetLogo called NetTango (Horn & Wilensky, 2011; 2012). NetTango blocks are not a full programming language, but domain-specific blocks relevant to the domain that is modeled. While as designers we might choose which blocks to provide learners, we should also understand what learners think about agent actions and how our designs may influence their thinking. We use design-based research to inform our design of domain-specific, agent-based, block-based microworlds. We use data from a pilot implementation of an environment used by 132 high school biology students to inform future designs.

NetTango

NetTango is a block-based programming environment for constructing NetLogo models. NetTango features *semantic blocks (*Wilkerson-Jerde & Wilensky, 2010; Wilkerson-Jerde, Wagh, & Wilensky 2015), which we now call *domain blocks* (Wagh and Wilensky, 2017). Domain blocks are primitive elements of code that represent agents' actions that can be used together to simulate a particular phenomenon. Importantly, NetTango blocks are not domain general – they are specifically designed to be relevant to the domain that is modeled. For example, to simulate population dynamics between predator and prey, blocks might represent agents such as *wolves* and *sheep*, and actions such as *give birth*, *move*, *eat* and *die*. This means that these blocks may not directly correspond to text-based NetLogo primitives. While that is supported (i.e. *forward 10*), they can also be used as abstractions of larger bits of underlying code into a more complex agent action such as *hunt*. Our larger research agenda is focused on designing domain-block libraries for simulating complex systems phenomena and studying how children use the blocks to engage in scientific theory building.

Agent-based ecosystem education

Drawing on a rich history of constructionist microworlds (Papert, 1980; Edwards, 1995), work with agent-based models has been used effectively for classroom teaching (Wilensky and Reisman, 2006; Klopfer et al, 2009; Sengupta and Wilensky, 2009). Examples of classroom use include *GasLab,* a series of agent-based modeling activities exploring statistical physics (Wilensky, 1999b; 2003), and *Beesmart*, a curriculum about the hive-finding behavior of honeybees (Guo and Wilensky, 2014a; Guo and Wilensky, 2014b). These examples highlight

that observing and/or interacting with microworlds allows students to construct their own understandings of science through exploring and adapting models in a classroom setting.

Methods

Three teachers from two large Midwestern US public school districts implemented a two-week biology unit called *Modeling Ecosystems Computationally.* The unit is specifically designed to engage students in computational thinking practices through the use of the technological environments (including NetLogo) while also covering traditional content knowledge in ecosystems (See also Wilensky, 1997; Wilensky & Rand, 2015). The unit is based on a real prey-predator ecosystem at an island called Isle Royale. The unit begins by giving the students an overview of the ecosystem: a closed environment with wolves and moose. Students are asked to think about the population changes of different species on the island and gradually work toward creating a computational model of the ecosystem to simulate and investigate population dynamics. They start with block-based coding using NetTango which allows students to define sets of behaviors for the wolves and moose using semantic "blocks" that encode various actions.

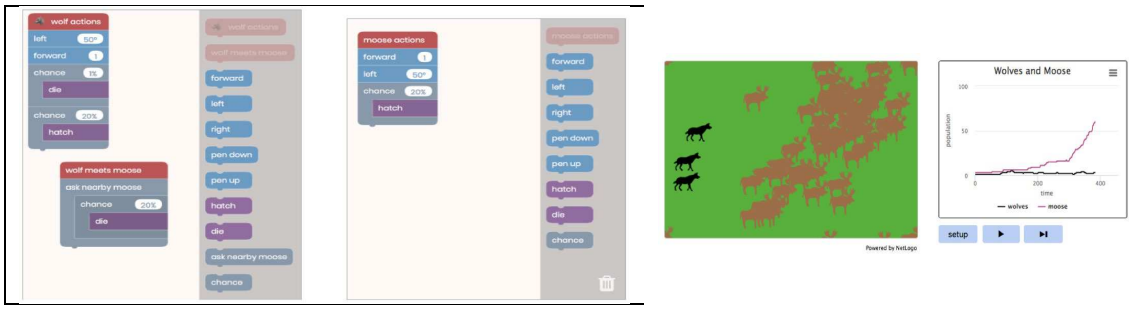

Figure 1. A screenshot of one student's recreation of Isle Royale in NetTango. On the left is their block-based code, on the right is their model.

As can be seen in Figure 1, each block in the model mirrors the text-based NetLogo code with the exception of wolf meets moose. The designers deliberately chose these "low-level" blocks to scaffold the transition to text-based coding in NetLogo later in the unit. As such, these blocks offer only a limited amount of domain specificity. After each student constructs her or his model, students are asked "If you could, what other types of behavior blocks would you add to make a more realistic wolf and moose predator/prey model?" It is data from 132 students on this question that we focus on in this proposal to answer our research question: *what level of abstraction should designers create blocks to engage students in scientific modeling*? We wanted to see what sorts of blocks learners would suggest to be added to the environment given their experiences with the above model, specifically along two dimensions: 1. System Level (*see* Table 1) and 2. Complexity Level (*see* Table 2*)*.

Analysis

In this paper, when we talk about theory building, we describe the process of theory building with domain blocks. In this process users follow an iterative cycle of model design, construction, test, and refinement. Users are given a target phenomenon, such as "the spread of disease" or, in this case, "the Isle Royale ecosystem" and asked to assemble the given blocks in a way that reproduces the phenomenon. Users snap blocks together and then run their models to test their theory. If there is not a match between observed and expected outcomes, users modify their models. Through this process, users construct a theory about the relationship between agent actions and the aggregate-level phenomena represented in the graph. However sometimes the blocks we provide are insufficient to fully represent their theory. We queried 132 students about what blocks they would add to their model in order to make it "more realistic." At the highest view, these suggestions represent the parts of their theory the students believe should be part of the model, that they did not think are possible to recreate using the existing blocks. We wanted to know what sorts of blocks students were suggesting. We had two hypotheses: **H1**: As is supported by previous work in complex systems education, students are able to suggest additional micro-level blocks (actions with a single agent) but would do less work defining actions across multiple agents (meso system level). **H2**: Students will suggest blocks that could already be implemented by re-combining the existing blocks.

Results

Following the below codebook, two independent coders coded 20% of the 273 suggestions of new blocks for the Isle Royale model activity. Interrater reliability for the System Level codes (see Table 1) was sufficient to proceed

and a single coder proceeded to code the remainder of the dataset for completeness. To code for computational complexity, the coders originally used a heuristic of "how much text code would be necessary to implement a block." However, this resulted in unacceptably low Kappa values. After discussing the conflicts, the coders agreed to instead focus on which blocks could already be implemented using the existing blocks and recoded the 20% slice using these new definitions (see Table 1).

Table 1: Codebook for System and Complexity Level

Code	Definition	Example
Micro	A block that pertains to one agent in the model	"wolf run"
Meso	A block that pertains to the actions/being of multiple (but not all) agents in the model.	"hunt as pack"
Macro	A block that pertains to global aspects of the model or would require omniscient knowledge of the state of the world to be implemented.	"count wolves"
Primitive	A block that can be implemented in the NetTango model by simply modifying a parameter of a single existing block.	"wait"
Recombination	A block that could be recreated using a combination of existing blocks in the model.	"die chance"
Model Extension	A block that would require an extension of the model beyond what would be possible with the existing agents and blocks.	"follow moose"

Table 2. Code counts and Cohen's Kappa for System and Complexity Level codes

	Micro	Meso	Macro	Primitive	Recombination	Extension
Frequency	166	64	25	64	77	114
Cohen's Kappa	0.817	1.000	0.777	0.871	0.700	0.955

While students seem to be able to suggest new microlevel actions not currently supported by the model, 77 of the 273 (28%) were blocks that could be modeled by using existing blocks. For instance, "defend" could be indirectly modeled by lowering the chance of a moose dying when a wolf attacks. "Stay" could be modeled with the combination of a chance block and forward block. That close to a quarter of suggestions could be modeled using existing blocks could indicate that learners may not see these blocks as "abstract" computational building blocks with which to create larger functions but rather as "puzzle pieces" that just need to be assembled in the correct order (i.e. "I want my animals to run so I need to have a run block."). This is reinforced by the fact that 64 (23%) of blocks could be implemented via modifying a parameter of an existing block.

Nonetheless, students suggested a total of 114 new blocks, or pieces of theory, that could not be modeled using the current NetTango primitives alone. These additions include 67 (60%) micro level blocks, 31 (28%) meso level blocks, and 14 (13%) macro level blocks. This seems to reinforce our first hypothesis that students naturally uptake thinking about agent-level, micro model interactions more readily, while larger interactions between sets of agents and macro-level ecosystem level thinking are more rare. It is possible this is due to a lack of priming of these sorts of meso-level interactions. In future work, we plan to make available to learners more meso-level blocks. Additionally, 209 (79%) of the block suggestions, regardless of System or Complexity Level, were substantially domain specific–that is, students described their blocks in terms of specific biological elements in this particular ecosystem NetTango model. Consequently, the fact that we provided blocks that were of low domain specificity could have limited students' ability to fully realize their own modeling goals.

Discussion and conclusion

In this paper we point towards two main ideas: (1) students may have difficulty imagining combinations of blocks to create larger actions; (2) students are adept at suggesting single agent actions when primed with other single agent actions. While we see students actively experimenting with building their own models, we also see evidence of learners treating the system more like a "puzzle" and less like a computational model where abstractions are considered par for the course. In this sense, they may feel limited in the ability to express their theories partially because they do not see a clear connection between a programming primitive and more contextualized actions and behaviors. Students seem to have little trouble suggesting actions from the point of view of a single agent,

but it is possible that students simply take up whatever "level" of thinking is presented in the given blocks. In future work, we plan to design an experiment with many different block variations in the same model to test this possibility.

While block-based environments provide an opportunity for students to actively engage in theory building, like any educational scaffolding, there are possible unintended consequences that result from the design. In future work, we plan to survey students' thinking about the modeling environment itself to better understand their perceptions of computational modeling after engaging with such models. If our goal is to provide low-floor high-ceiling environments for theory building in science classrooms, we need to better understand students' perceptions of this floor and ceiling.

References

Edwards, L.D. (1995). Microworlds as representations. *Computers and Exploratory Learning*

Guo, Y., & Wilensky, U. (2014a). Beesmart: a microworld for swarming behavior and for learning complex *systems concepts*. Paper presented at the Constructionism 2014 Conference, Vienna, Austria.

Guo, Y., & Wilensky, U. (2014b). *NetLogo BeeSmart – Hive Finding Model*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. http://ccl.northwestern.edu/netlogo/models/BeeSmart-HiveFinding

Horn, M., & Wilensky, U. (2011). NetTango [Computer Software]. Evanston, IL: Center for Connected Learning and Computer Based Modeling, Northwestern University. Retrieved from http://tidal.sesp.northwestern.edu/nettango/

Horn, M.S. & Wilensky, U. (2012). *NetTango: A mash-up of NetLogo and Tern*. In Moher, T. (chair) and Pinkard, N. (discussant), When systems collide: Challenges and opportunities in learning technology mashups. Symposium presented at the annual meeting of the American Education Research Association, Vancouver, British Columbia.

Klopfer, E., Roque, R., Huang, W., Wendel, D., & Scheintaub, H. (2009). The Simulation Cycle: combining games, simulations, engineering and science using StarLogo TNG. E-Learning, 6(1), 71. doi:10.2304/elea.2009.6.1.71

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc..

Sengupta, P., & Wilensky, U. (2009). Learning electricity with NIELS: Thinking with electrons and thinking in levels. *International Journal of Computing and Mathematical Learning*, 14, 21–50.

Schweingruber, H. A., Quinn, H., Keller, T. E., & Pearson, G. (2013). A framework for K-12 science education: Looking toward the future of science education. *Bridge*, *43*(1), 43-50.

Wagh, A., & Wilensky, U. (2012). Breeding birds to learn about artificial selection: Two birds with one stone? In: *Proceedings of ICLS 2012, Sydney, Australia*.

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*, *54*(5), 615-641.

Wilensky, U. (1999a). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (1999b). GasLab—an Extensible Modeling Toolkit for Exploring Micro- and Macro- Views of Gases. In Roberts, N. , Feurzeig, W. & Hunter, B. (Eds.) *Computer Modeling and Simulation in Science Education*. Berlin: Springer Verlag.

Wilensky, U. (2003). Statistical mechanics for secondary school: The GasLab modeling toolkit [Special Issue]. *International Journal of Computers for Mathematical Learning, 8*(1), 1-4.

Wilensky, U., & Rand, W. (2015). An introduction to agent-based modeling. *Modeling Natural, Social, and Engineered Complex Systems with NetLogo*, 504.

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.

Wilkerson-Jerde, M., & Wilensky, U. (2010). *Restructuring Change, Interpreting Changes:The DeltaTick Modeling and Analysis Toolkit*. Paper presented at the Constructionism 2010 Conference, Paris.

Wilkerson-Jerde, M., Wagh, A. & Wilensky, U. (2015). Balancing curricular and pedagogical needs in computational construction kits: Lessons from the DeltaTick project. *Science Education, 99*(3), 465-499.

Acknowledgments

We would like to thank the National Science Foundation (NSF Stem+C: Awards # 1842375 and 1640201) for funding this work.