A Computational Thinking Approach to Learning Science by Building Simulation Models

Satabdi Basu, Gautam Biswas, John Kinnebrew, Amanda Dikes, Pratim Sengupta, & TAG-V
ISIS & Peabody, Vanderbilt University
Supported by funds from NSF Cyber Learning Program

Research Goals and Motivation
The goal of this research is to develop a computer based learning environment for synergistic teaching of Computational Thinking skills and science concepts which can be seamlessly integrated into middle school classrooms.

What motivates this research?
• Computational Thinking (CT) defines an approach for formulating and solving problems by drawing on core computer science concepts like abstraction, decomposition, recursion, simulation
• CT is being emphasized as a fundamental skill for everyone like reading and writing
• CT is learnt best when learnt in conjunction with other domains
• Commonalities exist between CT skills and science learning; computational modeling shown to be effective for science learning

The state of the art
Synergistic CT-based environments for science education have not yet been integrated into classrooms in any significant way. Current approach - Sequestered approach:
• Learn a simple programming language, then use it to model scientific phenomena
• Requires time to learn language use and syntax
• Difficult to learn for younger children
• Requires science instructor with programming expertise

Problems to address
• Exploit the synergy between CT and science education by selecting a pedagogical approach that supports learning of both
• Design an activity sequence progressing from conceptualization of a phenomena to problem-solving using the acquired knowledge
• Choose a programming paradigm that makes the CT principles explicit without the challenges of learning a programming language syntax
• Make explicit the computational commonalities across different science domains
• Diagnose problems faced by students at varying levels of understanding while working in a CT-based environment for science learning
• Develop supporting tools and scaffolds to help students overcome their problems and regulate their cognitive and metacognitive processes
• Design assessments for measuring computational thinking competence

Proposed Solutions
We adopt a learning-by-design pedagogy based on a model construction-verification-refinement progression.

The learning activity sequence includes:
• Conceptualizing a given phenomena by selecting entities to model, their properties and behaviors, and property-behavior associations
• Building a computational model for the phenomena using a given visual programming paradigm
• Executing the computational model to visualize agent based simulations and graphical plots, and comparing the generated simulations and graphs against expert simulations and graphs
• Refining the conceptual and computational models based on differences identified between the user and expert simulations

Results from initial studies
Results from initial studies show significant learning gains for both units for both domain content and computational modeling skills.

<table>
<thead>
<tr>
<th></th>
<th>Pre</th>
<th>Post</th>
<th>P-value</th>
<th>Effect size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematics Domain content (max = 38)</td>
<td>15.28(4.37)</td>
<td>19.12(4.37)</td>
<td>&lt;0.0001</td>
<td>0.34</td>
</tr>
<tr>
<td>Kinematics Computational modeling (max=17)</td>
<td>3.78(2.53)</td>
<td>6.74(3.76)</td>
<td>0.0002</td>
<td>0.42</td>
</tr>
<tr>
<td>Ecology Domain content (max=49.5)</td>
<td>6.19(3.36)</td>
<td>11.98(5.33)</td>
<td>&lt;0.0001</td>
<td>0.70</td>
</tr>
<tr>
<td>Ecology Computational Modeling (max = 27.5)</td>
<td>9.15(3.07)</td>
<td>19.65(7.01)</td>
<td>&lt;0.0001</td>
<td>0.70</td>
</tr>
</tbody>
</table>

However, we also noticed that students faced several kinds of challenges related to modeling, programming, agent-based thinking and understanding domain content. The average number of challenges did go down with time in general (unless the learning activity itself introduced some new complexity) and the verbal scaffolds provided in our study proved helpful, reinforcing the need to incorporate them into the system as supporting tools or automated scaffolds.

Designing supporting tools and scaffolds
For supporting students’ tasks in CTSiM, various tools have been and will continue to be developed. The tools recently developed include:
• A set of searchable hypermedia resources with text, diagrams, videos, and simulations acting as a source of domain knowledge
• A model-tracing tool to trace the code command-by-command with the simulation to help correlate the models with the simulations
• A set of searchable hypermedia resources with text, diagrams, videos, and simulations acting as a source of domain knowledge
• A code commenting-out tool that enables students to test their code in parts

Other proposed forms of support include:
• Designing a guided dynamic workflow to help design structured experiments, explicitly specify goals, and use the comparison results effectively to verify and refine models
• Designing adaptive scaffolding provided by an intelligent agent to help students with using the tools efficiently and completing the learning task

Acknowledgements
This work was supported by the NSF (NSF Cyber-learning grant #1237350).