Learning Kinematics in Elementary Grades Using Agent-based Computational Modeling: A Visual Programming-based Approach

Pratim Sengupta
Mind, Matter & Media Lab
Vanderbilt University, USA
pratim.sengupta@vanderbilt.edu

Amy Voss Farris
Mind, Matter & Media Lab
Vanderbilt University, USA
amy.voss@vanderbilt.edu

ABSTRACT
Integrating computational modeling and programming with learning and teaching physics is a non-trivial challenge for educational designers. In this paper, we attempt to address this challenge by presenting ViMAP, a new visual-programming language and modeling platform for learning kinematics, and its underlying design principles. We then report a study conducted with 3rd and 4th grade students which shows that using ViMAP, they were able to develop a) deep conceptual understandings of kinematics and b) relevant programming and computational modeling practices. We also identify how the design principles supported the development of these understandings and practices as students engaged in learning activities that integrated modeling, programming and physics.

Categories and Subject Descriptors
D.3.3 [Programming Languages]

General Terms
Design, Human Factors, Programming Languages

Keywords
Agent-based Modeling, Science Education, Visual Programming, ViMAP, Kinematics, Physics

1. INTRODUCTION
Novices find mechanics quite challenging to understand. Over the past three decades, researchers [1, 2, 4, 5, 8, 9, 10] have reported that novices (e.g., middle school, high school and undergraduate students) face conceptual difficulties in: a) understanding and explaining the formal (mathematical) relationships between distance, speed, time and acceleration, and b) interpreting and explaining the physical concepts, relationships and phenomena represented by commonly used graphs of speed vs. time or distance vs. time. In particular, researchers have found that understanding continuous change in motion is challenging for novice learners. For example, when provided with a situation that involves objects moving with uniform acceleration (e.g., during a free fall or on an inclined plane), students in elementary and middle grades (4th & 6th grades) find it challenging to differentiate between instantaneous speed and average speed, [4] and tend to describe or explain any speed change(s) in terms of differences or relative size of the change(s), rather than describing speeding up or slowing down as a continuous process [5].

In this paper, we frame the educational problem identified above as a design challenge. Our central claim is that novices’ difficulties in understanding motion as a continuous process of change can be addressed by designing a learning environment that integrates conceptual development of physics understanding with the development of representational practices (e.g., modeling and measurement), using agent-based programming, in a manner that leverages novices’ intuitive understanding of motion. We first review the literature in educational computing and physics education, which provides some support for our claim that agent-based programming can indeed enable novices to understand kinematics through modeling, but also highlights challenges in integrating programming with physics curricula. We then present ViMAP, a multi-agent-based visual programming language [12, 14, 34, 7] and ViMAP-MoMo (Modeling Motion with ViMAP), which is a curricular unit in mechanics we designed for our study. We then present the underlying design principles and relevant design elements, and describe how ViMAP, based on these principles and elements, seeks to address the challenges we identified from the literature. We then report a study conducted with 3rd and 4th grade students in an after-school setting. We represent two types of analyses: First, we report a few illuminative cases in the form of analysis of interviews with students and their models, conducted during the course of their interactions with ViMAP. These interviews reveal how the relevant design principles and elements supported students in developing authentic mathematical representational practices (e.g., graphing), computational thinking (e.g., debugging) and scientific practices (e.g., modeling). We then present an analysis of students’ pre- and post-tests, which show that after using ViMAP-MoMo, they were able to develop rich, qualitative explanations about the relationships among distance, speed, and acceleration.

2. LITERATURE REVIEW
2.1 Pedagogical Advantages of Integrating Agent-based Programming with Curricular Physics
The term “agent” in the context of agent-based programming indicates an individual computational object or actor (e.g., a car), which obeys simple rules assigned or controlled by the user (e.g., moving forward, slowing down, speeding up). Among the earliest
and best-known agent-based programming languages that has been widely used to support children’s learning through constructionist activities (i.e., through the creation of personally meaningful computational artifacts) is LOGO [29]. In LOGO, the user learns by programming the behavior and movement of a protean agent—the LOGO turtle. The LOGO turtle was developed to encourage what Papert termed syntonic learning [29]. Turtles are said to be body syntonic, in that understanding a turtle (i.e., its objectives, behaviors and rules of operation) is related to and compatible with learners’ understandings of their own bodies. This body-syntonicity enables young learners to bootstrap their intuitive knowledge in order to learn canonical science concepts.

Researchers have long argued that learning programming, in particular, learning agent-based programming, is reflective with learning other domains [27]. This means that on the one hand, programming can serve as an effective vehicle for learning science and math concepts that students usually find challenging to understand [23, 24, 25, 26], and on the other hand, studies show that when computational mechanisms are anchored in real (world) problem contexts, programming and computational modeling become easier to learn [27, 25].

We believe that integrating agent-based programming will have the following advantages for our purposes. First, from the conceptual perspective, using ViMAP, students will be able to control the behavior of computational agents by specifying discrete, agent-level mechanisms in a body-syntonic fashion, using LOGO based primitives (e.g., controlling the movement of an agent by specifying its behavior for every step). This in turn will enable novices to access and build upon their intuitive knowledge of motion [1, 15, 23, 35] in order to develop a deep understanding of motion as a continuous process of change as aggregations of discrete actions. Second, from the perspective of development of students’ epistemic practices, designing computational models by creating and iteratively refining programs corresponds to core scientific practices, such as model construction, refinement, validation and deployment, design-based thinking and verification [29, 30, 6, 11]. Hestenes [6] argued that learning to build scientific models of physical phenomena is at the core of developing physics expertise. Papert [29] argued that learning to model using programming fosters the development of debugging, i.e., the ability to iteratively refine one’s own thinking through model building and refinement. Soloway [32] argued that learning to program involves learning how to construct mechanisms and explanations.

2.2 Challenges in Integrating Programming and Physics Education

Our review of the literature reveals that a central challenge in integrating programming with K16 classroom physics is the following: in order to program computer simulations of any physical phenomenon, both teachers and students must have some operational fluency with the programming language being used for instruction, in addition to the relevant domain knowledge. Therefore, in classroom-wide research studies of physics curricula that involve programming, the curricular units reported in the literature typically devoted a significant amount of time on programming instruction prior to science instruction. For example, in a study conducted by Redish and Wilson [31], undergraduate students underwent extensive programming instruction in Pascal prior to any physics instruction. In the studies reported in [1] and [23], middle and high school students underwent 15 weeks of instruction, out of which the first five weeks of classroom instruction were devoted solely to learning programming taught by a programming expert.

Researchers have also identified conceptual difficulties faced by learners in learning LOGO. For example, researchers found that even when students had developed some proficiency with a programming language after a few weeks of programming instruction, despite that fairly extensive learning experience, many students still found writing Logo programs in the Boxer environment to be extremely challenging [23]. In terms of classroom instruction, this often led to digressions during the class sessions, as the teachers had to clarify details of programming.

In earlier studies conducted on novice programming with LOGO, Pea and his colleagues noted similar concerns. In their studies, novice learners found the procedurality of the LOGO language, its control structures that allow very brief recursive programs, and the use of conditional tests to be deeply challenging without adequate scaffolding [33].

3. THE VIMAP LEARNING ENVIRONMENT

3.1 An Overview

ViMAP is a multi-agent-based visual programming language [12, 14] based on the NetLogo modeling platform and environment [19]. NetLogo is a modeling environment in which the user can create, interact with or manipulate thousands of agents, whose behaviors are controlled by simple rules that can also be specified and controlled by the users. Each agent in NetLogo is a LOGO turtle, whose behaviors are controlled by simple rules many of which are based on LOGO commands. In ViMAP, the user can command the behavior of NetLogo turtles by composing programs using a visual, drag-and-drop, snap-to-grid interface.

Using ViMAP, students can construct a new NetLogo simulation or modify an existing NetLogo simulation using visual and tactile programming, usually known as visual programming (or visual modeling). The ViMAP world is divided into three parts: construction-world (C-World), where learners construct their own programs; enactment world (E-World), where a protean computational agent (or a set of agents)—the classic LOGO turtle (or a set of turtles)—carries out learners’ commands through movement in representational space; and measurement world (M-World), which supports the Measuring Aggregation functionality.
(or MAGG). In the M-World, learners can dynamically generate inscriptions (e.g., graphs) of aggregate level phenomena by measuring relevant mathematical properties (e.g., distance, speed, etc.) of the enacted agent-level behavior (see Figure 1).

We discuss next three core aspects of the ViMAP environment that are central to this study: the programming primitives, measuring aggregation (MAGG) functionality, and modularity.

3.2 Core Elements of ViMAP

3.2.1 Programming Primitives

Each programming command in the C-World is visually represented as a green square icon, with an abbreviated label that identifies the function of the command. Computationally, command blocks are also represented as a class of NetLogo turtles. Command blocks can be dragged and moved by mouse-clicks. To minimize confusion between enacting turtles and command block turtles, we have termed the enacting agents “wabbits”. In activities that involve learners controlling the behavior of a single wabbit, we refer the wabbit as “Oscar”. The learner can construct their program by selecting commands from a list, choosing initialization values with sliders wherever necessary, and spatially ordering commands using a visual, drag-and-drop, snap-to-grid interface.

3.2.2 The Measuring Aggregation Functionality (MAGG)

An important element of ViMAP-MoMo are programming primitives and interactive functionalities that enable learners to design discrete mathematical measures that can bridge their agent-level understanding (which is generally intuitive in nature) and the aggregate-level understanding (which is generally found to be counter-intuitive), collectively termed the Measuring Aggregation functionality (MAGG). MAGG enables curriculum designers to design units with ViMAP that go beyond programming, in that using MAGG, learning activities can be designed to focus on generating mathematical inscriptions. Learners can access the MAGG utility simply by clicking on the icon labeled “measure” in the ViMAP interface, which clears the C-World by hiding the command blocks for each ball. A long-supported Logo-based representation of motion in introductory physics is the dot-trace representation [23, 29]. ViMAP-MoMo uses a flag (a cross-mark) in lieu of a dot as a representation of instantaneous position. In a simple program, where during every iteration, Oscar the wabbit moves forward by a certain distance and then plants a “flag” (a special class of NetLogo turtles), the distance between two successive flags can provide us with a measure of displacement during that iteration. In this case, the displacement is also the instantaneous speed during that time interval of one step. Learners can determine the total time elapsed by counting the number of flags, or by using the plant numbered flags command.

The learner can toggle back and forth between the code window (C-World) and the measure window (M-World) by clicking on the “rules” and “measure” tab respectively. Students can use the measure utility to develop graphs of motion. With the measure tab selected, if the learner then clicks on any two flags in the E-World, then the flags are immediately highlighted and a link (straight line) is generated connecting the two flags (see Figure 2a, right side). The learner then clicks the “Move Link” button to generate a vertical line segment of the same color as the flags being investigated, and with the same height as the distance between the clicked flags. These vertical bars appear in the M-World with labels denoting the height of the bar, which in turn represents the distance between the parent flags. Figures 2a and 3 illustrate typical screens showing the measure window during this activity. Notice that the left side of the interface is now occupied by series of links (vertical bars) that are labeled by their lengths. Once the speed and distance bars are visible in the M-World, the learner can click, drag, and move the links in order rearrange them and identify trends in the change in speeds and distances of the wabbit or wabbits.

3.2.3 Modularity

In software programming, modularity is defined as the extent to which a program is composed of separate, interchangeable components called modules or procedures, by breaking down program functions into modules, each of which accomplishes one central function and contains everything necessary to accomplish this. In ViMAP, students can create procedures comprising of ViMAP commands. In our study, they create procedures to denote different phases of motion.

3.3 Design Principles for ViMAP

In this section, we present the design principles that guided our design of the ViMAP platform. Pertaining to each principle, we discuss the specific challenge(s) in integrating programming and physics learning (see Section 2.2) that it targets to address.

3.3.1 Domain Specific and Domain General Primitives

Several researchers pointed out that the challenge in using programming to learn other domains is that programming as an activity often requires more skills and knowledge which is disconnected from the domain-specific learning goals [28, 24]. Therefore, rather than designing domain general LOGO primitives, the programming primitives in ViMAP were carefully designed to be relevant to modeling kinematic phenomena. ViMAP primitives include primitives for controlling turtle movement (e.g., stepping forward, speeding up, slowing down), aesthetics (e.g., drawing and color) and control flow (e.g., loop).

3.3.2 Learning Activities That Integrate Both Programming and Physics Learning Goals

The learning activities (described in Section 3.4) were designed to integrate learning programming and physics, without necessitating learning programming separately and then applying it to learn physics. For example, in Phase I students learned to use ViMAP primitives by engaging in drawing and painting, but by using the physics-based primitives. Both these activities leverage aesthetics and visualization as robust points of entry for learning mechanics. Phase I was designed to be a conceptual stepping stone to the Phases II and III, in which students used these physics-based commands in order to model the motion of the moving objects. Furthermore, learning activities in Phases II and III also emphasize modeling real world scenarios using ViMAP. In Phase II, the real world scenario involves wooden ramps and three balls, and in Phase III the real world scenario was presented in the form of a YouTube video of a car moving on a roller coaster. Thus, the focus of the activities shift from learning programming using both physics-based and domain-general primitives in Phase I, to modeling physical phenomena in Phases II and III. That is, the programming in Phases II and III develops iteratively as students learn to make meaningful observations. The latter—i.e., learning to problematize, or, figuring out what’s relevant or worth observing in a phenomenon—as Lehrer & Schau[30] pointed out, is central to the development of the practice of modeling, as well as a modeling based epistemology.
3.3.3 Multiple “Liveness” Scaffolds
Finally, in order to support debugging, ViMAP offers learners a range of “liveness” factors [16]. For example, learners can choose to edit their code while the simulation is running, or they can choose to stop the simulation, make appropriate changes to the code (model), and then re-run the simulation. Learners can also choose to overlay the graphical results from multiple runs, and the results of each run are automatically saved in a different color. As commands are executed, the currently active agent’s tagged page of code blocks is brought to the foreground, and the currently running code block is highlighted in red. The user can also set a time delay between command blocks as a further aid to debugging and code visualization.

3.4 SEQUENCE OF LEARNING ACTIVITIES

3.4.1 Phase I: Drawing “Constant Speed” and “Constant Acceleration” (Sessions 1 and 2)
During the first activity, we introduced the children to ViMAP by showing them how to manipulate different elements in the user interface. We then asked the students to generate their first ViMAP algorithms by selecting and arranging blocks of code to make Oscar “draw” simple shapes – squares, triangles and circles. In doing so, our goal was to familiarize students with the use of available programming commands (“forward”, “right turn”, “left turn”, “pen down”, “set heading”, “pen up” and “repeat”), and debugging.

In the second session, we asked students to modify their programs for generating squares, triangles, and circles in order to generate spiraling shapes in which each line is longer (or shorter) than the previous one. Our goal was to introduce students to the “speed-up” and “slow-down” commands. Using these commands along with the forward command results in changing the value of Oscar’s step-size. That is, if a learner’s code contains “forward 20”, followed by a “speed-up 2,” then Oscar’s step-size will increase by two units every iteration (i.e., 20, 22, 24, and so on). From the perspective of learning physics, by engaging in this activity, we expected students to develop representations of continuous change in motion.

3.4.2 Phase II: Modeling & Graphing Real-World Motion (Session 3)
In Phase II, we asked students to use the same primitives they used in Phase I in order to model “real-world” motion. The physical setup involved two balls rolling down two ramps of different elevations and one ball being pushed on a horizontal surface (the floor). We gave the students a more case-specific version of ViMAP. In the E-World of this environment, there were three waddles, each represented as a different colored ball. Each of these balls was on a ramp, and each ramp had a different slope. The students were asked to write a program for each ball that would make the simultaneous movements of the two balls appear to happen similarly to that of the balls in the physical setup with the ramps. Students iteratively refined their programs for the different balls by running the resultant E-World simulation, and observing whether the simulated target outcome (i.e., which ball reaches the end first), matched that of the real world. Once students generated their programs for the three balls, they were asked to represent how speed and distance of each of the balls changed over time by using the MAGG functionality presented earlier. Figure 2a shows a sample set of inscriptions (speed and distance graphs) generated using the speed and distance bars, as described in section 3.2.2, during this activity. While both the red and yellow balls start from rest, they experience different amounts of acceleration—the red ball travels a steeper slope and therefore accelerates faster. This becomes evident in the relative distances between two successive flags for each ball: the flags of the red ball are farther apart than those of the yellow ball.

3.4.3 Phase III: Modeling Motion on a Roller Coaster (Sessions 4, 5, and 6)
In this activity, students first watched a YouTube video that was shot from the perspective of a passenger on a roller coaster car. We led a whole-class discussion, in which we segmented the motion of a car on a roller coaster into sections by identifying changes in acceleration (resulting changes in the slope of the roller coaster).

Students then worked on creating a ViMAP model to represent the motion of the car by representing each phase of its motion in the form of a mathematically meaningful shape. For example, a period of constant acceleration for five seconds can be represented as a rectangular spiral with five sides, where each side is longer than the previous side by a fixed magnitude. This magnitude represents the value of acceleration as estimated by the students, while the number of sides represents the time (5 seconds). Each distinct phase is represented in ViMAP as a procedure block in the C-World. Students chose the name of the procedures. For each procedure, students also were asked to generate speed vs. time graphs by generating and temporarily arranging distance and speed bars, using the MAGG functionality.

4. METHOD

4.1 Participants & Setting
The study took place in a large metropolitan city in a classroom on the campus of large private university in the mid-southern USA. Fourteen children from 3rd and 4th grades of local schools were recruited by email and web solicitation for a six-session course. Classes met once a week and took place from 9:00 am to 11:30 am on six consecutive Saturday mornings. In order to recruit participants for this study, the email solicitation was sent out to a listserv maintained by the metropolitan school district in the city, and an advertisement for the course was posted on the school district’s website and the university website. Besides the specified grade levels (3rd and 4th grades), there were no other selection criteria for the course. In the email and online solicitations, we specifically mentioned that this course was intended for students with no programming experience or background, and that applicants were admitted on a first-come-first-serve basis.

None of the students in this course had any prior programming experience. The student population consisted of two female and twelve male students. The ethnic-racial composition of the student population was as follows: African American: n = 1; Biracial (Latino-White): n = 1; Asian: n = 1; White: n = 11. Two students were also recent European immigrants whose families migrated to the U.S. four years earlier.

Each student worked individually on a Macintosh desktop and continued to work individually for most of the course. On a few occasions, the facilitators led classroom discussions where all students participated.
4.2 Analysis

4.2.1 Illuminative Case Studies
In order to investigate the process of conceptual development during learners’ interactions with the model, we adopted an explanatory or illustrative case study approach for our analysis [17]. As Yin pointed out, while case studies may be of several types, explanatory case studies are well suited as a methodology to answer how and why questions. Because our central goal is to characterize the process of knowledge construction in the learners’ minds as they interact with the simulation – i.e., a how question – we believe that an explanatory case study based approach is well suited for our research goals.

The data for the cases come from in-depth interviews with the participants, the ViMAP models generated by the students, and field notes. The interviews were conducted while the learners were engaged in the modeling and programming activities, and were videotaped. In some cases, the interviews ensued when the learner called upon the teacher in order to help him or her with a difficulty. In other cases, we conducted interviews in order to ask learners to explain their programs or models. Each case is presented in the form of thematic analysis [21, 22] of student-generated artifacts, along with associated interview data and/or field notes. A theme captures something important about the data in relation to the research question, and represents some level of patterned response or meaning within the data set [21, 22]. In the context of our study, each theme represents a significant aspect of the co-development of learners’ knowledge and representational practices. The three cases we present in the next section illuminate the following themes of learners’ epistemic development: i) debugging (Case 1), ii) role of real-world validation in modeling (Case 2), and iii) coordinating multiple forms of computational representations to model and explain real world phenomena (Case 3).

4.2.2 Pre-Post comparisons
To assess the growth in students’ conceptual understanding of physics, we administered students a written pre- and post-assessment. Students were assigned up to 45 minutes for each assessment. In these assessments, students were provided with four types of problems: constant acceleration problems; interpretation of speed-time graphs; the generation of a speed-time graph, and two problems that required students to differentiate between objects’ positions, speeds, and accelerations. Students’ responses to each question were coded as correct or incorrect. We provide detailed descriptions of each question along with the associated coding rubric, and present sample students’ responses in Section 5.4.

5. FINDINGS

5.1 Case 1: Learning to Debug
Students’ debugging activities involved two components: first, identification of how elements of their programs corresponded to aspects of the output, and second, iterative refinement of their program so that it generates the desired or target output. This teaching episode highlights how the liveness scaffolds (as discussed in Section 3.3.3) can support students’ identification of the relationship between the ViMAP program in the C-World that controls the actions of a wabbit, and the wabbit’s enacted actions in the E-World. The interview is quoted below:

Line 1. Amy: So, what needs to change about this square?
Line 2. Nathan: The lines need to get smaller, but when I
Line 3. make them smaller, it makes four rectangles that

At the beginning of this interaction, Nathan had already figured out the mathematical mechanism that would generate a spiral but was struggling with computationally representing the mechanism. In lines 2–4, Nathan said, “The lines need to get smaller, but when I make them smaller, it makes four rectangles that touch each other.” Nathan had written a program for a square, and after each run of his program, he manually reduced the forward value. The resultant outcome in the E-World was a set of “four rectangles that touch each other,” where each rectangle was smaller than the previous one. Amy (the second author), then suggested adding a new rule type, slow down, at the end of his existing program (line 5). Upon implementing this change, Amy and Nathan ran the program and this produced a square-shaped spiral (Figure 2).

Figure 2. Screenshot of Nathan’s Spiral: Each command block changes its color to red while being executed

Amy then asked Nathan to explain why that change generated a spiral (line 12). Nathan, while looking at the code being enacted, suggested that it had something to do with the forward parameter, i.e., Oscar’s step-size (line 20), but his intonation suggested that was not quite exactly what happened during the execution. Amy suggested that they increase the value of the delay between the executions of successive commands, so that the program would run very slowly (lines 23–28). Figure 2 shows a screenshot from
the video of the interview, selected during the execution of this program.

As shown in Figure 2, each command changes its color to red as it is being enacted. Upon watching the new program being enacted by Oscar, Nathan observed the forward parameters change each time the slow-down command was executed. Without further prompting, Nathan continued to watch the commands execute slowly, and as a result of doing so, in Lines 33-35, Nathan’s statements show that he identified the relationship of Oscar’s step sizes to the speed-up command.

This episode, therefore, shows the following: at the beginning of the episode, Nathan already understood the mechanism for generating the rectangular spiral, although he had not yet discovered the functionality of the “slow-down” command. However, it is in attempting to represent the mechanism formally, that he was able to debug his earlier program with assistance. The assistance that Amy provided was in the form of showing Nathan how to use two liveness scaffolds—the delay functionality, and the code step-through highlighter. It was by using these software scaffolds that Nathan was also able to identify the mathematical relationship between the commands “slow-down” and “forward”.

5.2 Case 2: Role of Real-world Validation in Learning Modeling

While modeling the motion of the balls rolling down ramps using ViMAP-MoMo, students were frequently prompted to compare the actions of the blue, red, and yellow balls in the real-world and about how these differences would show up in their programmed simulations. In trying to model a real-world scenario through such frequent comparisons between the modeled and the real worlds, our analysis of students’ models show that students learned to recognize important constraints in the scenario to be modeled. For example, in Nathan’s initial attempt to model the behavior of the three balls, none of his balls were accelerating; the ball on the steepest slope had the largest speed, while the ball on the ground had the lowest speed. When the first author noticed this, he asked Nathan to revisit the physical setup, and asked him to observe carefully whether the balls were all starting out with non-zero speed in the physical setup. It was through this iterative comparison between his computer model and the physical setup that Nathan realized that initial conditions play an important role in kinematics. In this particular case, Nathan realized that in order to accurately model the scenario in the physical setup, the yellow and the red balls should start from rest in his model.

Figure 2a: Graphing motion of balls on inclined planes using the MAGG functionality

We also found that like Nathan, the majority of the students initially modeled the blue ball to move at a constant speed; it was only after several rounds of validation by re-examining the real-world scenario a few times, that they realized that the blue ball, which in the physical setup was rolling on a frictional surface with only an initial push, should also slow down every step in their simulation. Furthermore, this process of model refinement and verification helped Nathan and several other students realize that a ViMAP-MoMo model where the red ball reaches the end of the ramp first is not necessarily a complete or correct model, and that even when the red ball may initially appear to be behind the yellow ball, it will still “win” in the end. That is, rather than focusing solely on simulating the outcome, validation of the students’ models with the physical setup made them focus on the mechanistic aspects of the phenomena. This shows that validation of the simulated world with the real world played an important role in the iterative refinement of students’ models.

5.3 Case 3: Coordinating Multiple Models

Here we report an interview that highlights how Olivia, a 3rd-grade student, constructed her conceptual understanding of the multiple representations generated during Phase III. In the interview, Olivia is actively coordinating the representations of speed bars, her ViMAP program in the form of a series of command blocks, and the composite image of multiple spirals. In this interview, Olivia was asked by one of the authors to explain the three different representations she designed to model the motion of the boy traveling on the roller coaster, as shown in the YouTube video. The following is a partial transcription of the interview:

Line 1. Amy: Can you tell me what all this means? Pretend
Line 2. that I don’t know anything about what you are doing.
Line 3. Olivia: Okay, so this is a roller coaster, the model of the roller
Line 4. coaster [gestures to the shapes window], and in this model,
Line 5. what I did was, it started at rest, so um, the pentagon, the one
Line 6. speeding up, speeding up slowly, and then the green square
Line 7. there is also speeding up a little bit more.
Line 8. Amy: Just based on what it looks, on how it looks, how do you
Line 9. know that the green square shows speeding up?
Line 10. Olivia: Um, because it’s going inside of the square, not, like,
Line 11. outside.
Line 12. Amy: Because it’s going inside? [asks for clarification]
Line 13. Olivia: [looks at the corresponding speed bars] Oh, No,
Line 14. because it’s going outside, not inside, because
Line 15. if it were going inside, it would be slowing down.
Line 16. Amy: Oh, okay, that’s the way I see it, too... So it’s speeding
Line 17. up here [pointing to pentagon], and then what’s happening in
Line 18. the green then, tell me one more time.
Line 19. Olivia: Um, it’s speeding up, but not by very much, because
Line 20. it’s not spreading out as much as in the lines.

In lines 3 – 7, Olivia describes what each of the different shapes shown in Figure 3 represents in terms of the phases of motion. Her explanation indicates that she understood that the pentagon and the square spirals represented two phases with different amounts of acceleration, and this was evident in the relative sizes of their sides. When Amy asked her to explain why she thought that the green square spiral represents speeding up, she explained that the square is growing outward – so the lines are getting longer—and this indicates a process of speeding-up (Line 13).

However, immediately before this statement, Olivia mentioned that the square was “going inside” (Line 10). Computationally, this meant that Oscar was taking shorter steps every turn, while going forward and turning right by 90 degrees. When Amy asked her to clarify her statement, Olivia looked to the speed bars and changed her mind, “It’s going outside, not inside, because if it were going inside [i.e., if the steps were getting shorter], it would be slowing down.” In Line 18, Amy again asked Olivia about the speed bars. This time, she provided a warrant for her claim that the spiral represents a phase of speeding up, based on her observation of the “spread” of the negative space in the square spiral: “Um, it’s speeding up, but not by very much, because it’s not spreading out as much in the lines.” This shows that Olivia interpreted the increasing or decreasing negative space in the
shapes she generated as representations of acceleration or deceleration.

The interview then continued as follows:

Line 20. Amy: Good. Tell me about the next one.
Line 21. Olivia: Um, the next one is the little triangle-ish thing.
Line 22. Amy: star, and it was hard to measure that one too—
Line 24. Olivia: But I did. And that was um, it was kind of staying
Line 25. at a constant speed, and so it didn’t like [interrupt] get
Line 26. thicker or anything.
Line 27. Amy: The red ones, [pointing to red speed bars], these, is there any mathematical pattern that you see there?
Line 28. Olivia: Yes. There, it's speeding up by two.
Line 29. Amy: Okay, so does that tell us anything about the roller coaster?
Line 30. Olivia: That it's speeding up.
Line 31. Amy: Oh, I thought you were telling me...
Line 32. Olivia: ...it was constant. No. I was [incomprehensible]
Line 33. Amy: Oh; is there anywhere in your model where the
Line 34. roller coaster is going at a constant speed?
Line 35. Olivia: No.

This excerpt shows that Olivia resolved another ambiguity resulting from the overlapping shapes as shown in Figures 3 and 4. In Lines 24 – 26, she reported that the procedure represented by the “little triangle-ish thing” represents a phase of “kind of” constant speed. However, by then she had already measured all the distances between flags in the model, by creating speed bars using the MAGG utility. The speed bars, in their arrangement, showed a clear pattern of Oscar’s increasing step sizes as time progressed. Note that in the previous excerpt, she also referred to the increase in Oscar’s step-sizes as “spread” between the lines (line 20), as evident in the negative space of the shapes generated. In her explanation, less negative space within a shape indicates “less” speeding up. However, in this excerpt, she referred to the shape as getting “thicker”. We believe that the overlapping lines of the red shape (see Fig 4) made it difficult for her to identify the spread. At this point, Amy pointed to the relevant speed bars, and asked Olivia if there was a pattern that she noticed in their arrangement (Lines 27 - 28). Olivia identified that speed bars indicated that the red shape is speeding up twice as fast (line 29). The implicit comparison is with the procedure represented by the green square. This episode therefore shows that creating and having access to complementary representations – speed bars and shapes – was a critical resource for Olivia’s learning.

![Figure 3. A screenshot of Olivia’s model: The speed bar representation is on the left, and shapes representing different phases of motion are on the right.](image)

Overall, this interview shows that Olivia used three types of computational representations to support her description and explanation of the motion of the roller coaster. She refers to the speed bars, which she arranged sequentially (in time), as indicators of whether the boy on the roller coaster was speeding up or slowing down. As evident from both the interview excerpts in this case, it was in comparing the interpretations of these representations that Olivia constructed her meanings of the models she generated.

This case also shows that Olivia was able to use her prior experience with constructing speed bar graphs during the previous activities she conducted during in Phase II. Olivia had measured each of them and arranged them sequentially to construct the graph. The speed bars were easier to interpret, and at the end of the interaction, Olivia was satisfied that what was difficult to interpret in the spiral shapes represented the same information as is depicted by the speed bars. This episode therefore suggests that in addition to the multiple, complementary representations of the speed bars and the shapes, the sequence of activities we designed was also played an important role in Olivia’s epistemic growth.

5.4 Pre- and post assessments of conceptual understanding in physics

5.4.1 Constant Acceleration Problems

The first two items in the pre- and post-assessment ask students to reason about scenarios of constant acceleration that are familiar to them: the motion of a ball in free fall (Question 1) and the motion of a ball immediately after it is tossed up into the air (Question 2). The problems are quoted below:

**Question 1**: A ball is dropped from a building and takes 10 seconds to reach the ground. Predict where the ball would be after 1, 2, 3, and 4 seconds. Please use pictures or diagrams to explain your answer.

**Question 2**: A ball is tossed up in the air and takes 10 seconds to reach the top of the building. Predict where the ball would be after 1, 2, 3, and 4 seconds. Please use pictures or diagrams to explain your answer.

In the first question, we characterized correct responses as drawings (and/or written explanations) that explicitly indicated either the ball travelling a larger distance every second or moving faster. In the second question, we characterized correct responses as drawings (and/or written explanations) that explicitly indicated the ball travelling a smaller distance every second or moving slower. Table 1 provides examples of sample correct and incorrect responses for both the questions.

**Table 1. Examples of responses scored correct and incorrect for constant acceleration problems**

<table>
<thead>
<tr>
<th>Question 1 (correct): Ball dropped from a building. The student indicates that the ball is speeding up by showing that the distances increase.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1 (incorrect): Ball dropped from a building. The student does not show indication of acceleration.</td>
</tr>
<tr>
<td>Question 2 (correct): The ball goes down. The student indicates that the ball is speeding down.</td>
</tr>
<tr>
<td>Question 2 (incorrect): The ball goes up. The student indicates that the ball is speeding up.</td>
</tr>
</tbody>
</table>

We used matched pairs t-test for comparing pre- and post-assessment data. Student responses for both of the one-dimensional acceleration items show significant gains. Correct responses for Question 1 increased from one (7%) to fourteen (100%), and matched pair t-test show that the increase is statistically highly significant \( t = 13.00; df = 13; p < 0.0001 \). For Question 2, correct responses increased from three (21%) to
twelve (93%), and matched pair t-test show that the increase is statistically highly significant \([t = 5.701; df = 13; p < 0.0001]\).

5.4.2 Interpretation of Speed-Time Graphs

We designed two assessment items to examine students’ explanations of speed-time graphs (see Fig 4), in which students were asked to write a description of the motion of the object represented by each graph.

![Graphs A and B](image)

**Figure 4.** Graphs A and B, used for the interpretation of speed vs. time graph assessment items

We characterized correct responses as those that made appropriate descriptions of relative speed of the object at different times. Table 2 lists an example of a typical incorrect and a correct response for each graph.

<table>
<thead>
<tr>
<th>Graph A</th>
<th>(incorrect): It would speed up and then slow down.</th>
<th>(correct): It starts fast but slowing down and the time is long.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph B</td>
<td>(incorrect): Going up makes it slow, when going down it goes very fast.</td>
<td>(correct): The car starts by speeding up. Then the car slows down. Finally the car stays at a constant pace.</td>
</tr>
</tbody>
</table>

Number of correct responses increased from six in the pre-test to nine in the post-test for Graph A \([t = 1.15; df = 13; p = .27]\), and five in the pre-test to nine in the post test for Graph B, \([t = 1.74; df = 13; p = .10]\). We found that most students who answered these items incorrectly identified the shape of the graph as the trajectory of the car (e.g., in Graph B, the car is slowing down because it is going up a hill).

5.4.3 Generating a Speed-time graph from a description of motion

Following Dykstra and Sweet [2], we included a question designed specifically to assess the student’s abilities to represent speed graphically. We modified a question described by Dyksta and Sweet and we provided a “skeleton” of a speed-time graph, consisting of a horizontal axis labeled “time” and a vertical axis labeled “speed”. The question read as follows: “Someone made motion in front of a motion detector. The motion consisted of standing at rest for one second, then, walking away from the detector while speeding up for about one and one half seconds, and next, walking away while slowing down for about one and one half seconds, and last, stopping and standing at rest for one second. If the motion detector makes a graph of speed versus time, what would it look like?”. All the students experienced difficulty in understanding this question in the pre-test and asked for assistance from the instructors. To address this, we requested a student volunteer to “act out” the motion, and we talked her through acting out the motion, and then asked if the students understood the question. During the post-test, none of the students experienced difficulty with interpreting the question, so no physical enactment was necessary. We scored student responses on a Likert scale (0, 1, or 2). Responses that did not show the periods of uniform acceleration by including both the periods of speeding up and slowing down were scored as 0; responses that correctly depicted uniform acceleration, but did not correctly show the instances of rest were scored as 1; and responses that showed all components of the motion were scored as 2. In the pre-test, eight students responded with 0-point answers, four students gave 1-point responses, and two students gave 2-point responses. At post-test, three students gave 0-point responses, three students gave 1-point responses, and eight students gave 2-point responses. Paired sample T-test showed that the gain in the average scores in the post-test compared to the pre-test was statistically highly significant \([t = 3.29, df = 13; p = 0.0058]\).

5.4.4 Distinguishing Distance, Speed and Acceleration

The final two items (see Figure 5) were adapted from the Force Concept Inventory [20]. We chose these items because they require the student to differentiate between speed and acceleration. The first is reprinted exactly as appears in the FCI, Item 20. We adapted the second item for this study in order to assess students’ abilities for distinguishing speed, distance, and acceleration.

In the first question, students showed significant gains in their ability to identify speeds when represented as distance traveled during an interval of time. None of the students were able to provide a correct answer in the pre-assessment, whereas nine students identified the correct response in the post-assessment \([t = 4.84; df = 13; p = 0.0003]\).

For the second question, students’ responses were scored as follows: if the student recognized that Block B is moving faster than Block A, the response was assigned a score of 1. The response was assigned a score of 0 if it indicated that one block was accelerating differently than the other. One student answered correctly during pre-test, and six answered correctly during post-test. Paired sample t-test shows that the gain in student performance is statistically insignificant \([t = 2.69, df = 13, p = 2.69]\).

6. CONCLUSION & DISCUSSION

6.1 Development of Students’ Conceptual Understanding of Kinematics

Comparisons of pre- and post-test responses of the participants show that in the post-test, students showed significant learning gains in terms of being able to differentiate between speed, distance and acceleration, as well as being able to identify motion due to gravity as a process of continuous change. These results show that students were able to develop a deeper conceptual understanding of the relevant physics involved. This shows that even though students engaged in programming activities throughout the course, it did not distract them from learning physics. Given that integrating programming and physics learning goals in the ViMAP learning environment and learning activities was one of our central design principles and objectives, this result is therefore an important measure of the effectiveness of our design. This is particularly important because it has been shown by previous scholars that the emphasis on programming can distract from the focus on learning physics [1].
Our results suggest that in order to develop an understanding of motion as a process of continuous change, students in our study did not have to abandon their intuitive, “snapshot” views of motion. As discussed earlier, a snapshot view is a discrete representation of motion and researchers have identified such views as naive and incorrect descriptions of motion [9]. Our analysis shows that students in our study indeed entered the instructional setting with snapshot views of motion, as evident in their pre-test explanations of free-fall and free-toss. However, ViMAP primitives and activities were specifically designed to bootstrap their intuitive snapshot views in order to enable them to develop mathematical representations of motion as a process of continuous change, by aggregating multiple snapshots. For example, the measure utility using speed bars was designed to help students view the process of changing speeds of the different balls as a continuous phenomenon. We believe that it was through engaging in the activities in Phases II and III that students learned to piece together multiple "snapshots" - where each snapshot corresponds to movement of the rabbit(s) during a single time-interval (or step)—and thus develop a conceptualization of the changes in speed and position as continuous processes by piecing together the individual snapshots. This is also evident in their post-test explanations of free-fall and free-toss, i.e., scenarios involving constant acceleration, in Questions 1 and 2.

6.2 Role of Software Scaffolds and Teaching
The cases we presented in this paper also reveal that teacher-led scaffolding is essential for learning with ViMAP. In our study, the role of the teacher involved prompting learners for agent-based thinking (i.e., think like the turtle), as well as introducing students to the relevant software-embedded scaffolds for debugging reported earlier in this paper. It is also important to note that while ViMAP primitives are intuitive to understand, we believe that the requisite teacher preparation for using ViMAP is significantly less than earlier studies with LOGO and Boxer [1, 23]. Designing teacher professional development modules for using ViMAP is one of our central foci in our current and future research.

6.3 Designing & Sequencing Learning Activities To Integrate Programming & Modeling with Physics
Unlike previous attempts to integrate LOGO with learning physics, our unit does not require separate programming instruction. Activities in the first phase involved students generating geometric shapes and patterns using domain-specific (i.e., physics-based) primitives for controlling the movement of an agent, as well as domain-general primitives such as those for control flow (e.g., loops) and aesthetics (e.g., color and drawing). We designed these activities to leverage aesthetics and visualization as robust points of entry for learning mechanics and to develop students’ familiarity with programming and modeling using ViMAP primitives. Creative expression can be a productive and personally meaningful route towards computational literacy [3]. Instead of using a physical paintbrush, the students used ViMAP commands, and in this coupling of art and computation in Phase 1, students harnessed the computational power of the medium while they were engaged in learning physics. Our analysis shows that as students engaged in the modeling activities that leverage aesthetics, besides learning physics, they became increasingly proficient in agent-based thinking (i.e., thinking like an agent), programming using ViMAP commands, and debugging.

The rationale behind the overall sequencing of the activities and the phases was to foster necessary representational competencies in the earlier phases (e.g., In Phase 1: agent-based thinking; debugging; learning ViMAP programming including syntax, commands and control flow; and, In Phase II: debugging; iterative modeling; and graphing) which in turn were central to designing and developing more complex models of physical phenomena in Phase III. This tight coupling between learning programming and learning physics is one of the central design principles of ViMAP [12], and we also consider this to be a major contributions of this paper.

6.4 Implications for Educational Design
Historians and philosophers of science have shown that in the development of scientific knowledge, the concepts, tools and presentational practices of science are deeply intertwined [30, 11]. From this perspective, developing scientific understandings and explanations is not simply a matter of stringing together the logical entailments of information; rather, it involves struggling with posing questions, arranging conditions for seeing, developing measures, structuring data, and understanding the entailments of that data. Therefore, in order to integrate programming in K12 science classrooms, we need to rethink the design of programming languages and modeling platforms such that the essential components of the learning environment – i.e., the programming primitives, software scaffolds, learning activities and teacher-led scaffolding – are designed in order to support not just the learning of programming or scientific concepts, but also the learning of representational practices that are central to the science domain. To this end, we believe that the design elements and principles of ViMAP (Section 3.2 and 3.3), as well as the design and sequence of the learning activities (Section 3.4) can provide designers of educational programming languages with a template of issues to consider carefully, for achieving a tightly coupled integration of programming, modeling and science learning in K12 classrooms.

7. ACKNOWLEDGMENTS
This work was partially funded by NSF IIS # 1124175. Thanks to W. Hubbell, G.C. Krishnan and M. Wright for their contributions.

8. REFERENCES


