Programming and its affordances for physics education: A social semiotic and variation theory approach to learning physics

Svensson, Kim; Eriksson, Urban; Pendrill, Ann-Marie

Published in:
Physical Review Physics Education Research

DOI:
10.1103/PhysRevPhysEducRes.16.010127

2020

Document Version:
Publisher's PDF, also known as Version of record

Programming and its affordances for physics education: A social semiotic and variation theory approach to learning physics

Kim Svensson,* Urban Eriksson, and Ann-Marie Pendrill

National Resource Centre for Physics Education, Department of Physics, Lund University, SE 221 00 Lund, Sweden

(Received 28 September 2019; accepted 28 April 2020; published 13 May 2020)

A small group of interested upper secondary education students participated in a workshop where they created a particle-based physics engine and used the engine to implement a hanging cloth simulation and a two-dimensional heat diffusion model of their own creation. During the implementation of their models, learning opportunities present themselves in the form of opening up and exploring different dimensions of variation for the students. By varying aspects and discerning how these changes affect the program, students can construct meaning about the system. The students were video and audio recorded during the workshop and interviewed afterwards. Based on the transcripts, students use of programming was analyzed using social semiotics and variation theory of learning with a focus on the three aspects: coding, visualization, and interaction. The analysis identifies usages of programming such as a transductive link between semiotic systems, the ease of varying and iterating aspects, and the ability to enter into a loop of discovery and understanding.

DOI: 10.1103/PhysRevPhysEducRes.16.010127

I. INTRODUCTION

This paper aims to highlight why programming could be a useful tool for meaning making in physics education and focuses on the interplay between coding, visualization, and interaction. By describing programming as a semiotic system (described below) to be used in communication and meaning making in physics education, a theoretical framework is provided that allows us to study programming through the lens of variation theory by focusing on how programming's affordances change (also described below). A small group of upper secondary education students, who knew each other well, participated in a study designed to take them through the process of creating a physics simulation using Python [1] in the Processing IDE [2]. Our study investigates to what extent the students were capable of using programming to extract meaning from simulations, how they modified the resulting representation and how they interacted with the simulation.

By using logical operators and algorithmic thinking, programs can be constructed that perform a wide range of different tasks, such as simulating different physical phenomena. In physics, a student may create a model of a physical concept, run the simulation, and ask "What happens in this specific scenario?" The student may then analyze the output from the program to get an answer to the question. Whatever answer produced by the simulation, the student has an opportunity to learn something; if the answer is an "error" the student has been informed where their code may be wrong and can change the code and in the process they explore and learn different aspects of the concept they are implementing. If the simulation conforms to the expected behavior, questions about the content of the simulation can be asked. There is a "communication" between the program and the student, where the student tries to extract relevant information, through the visualization, from the program about different aspects of the model they implemented.

In a simple simulation, where a planet orbits a star, the student may ask "What happens if I change the mass of the planet?" If the student has implemented a correct version of Newton's law of gravitation, $F = GmM/r^2$, there should be no change in the behavior. However, if there is an error in the implementation, the planet will behave differently and the result will differ compared to real world experiments and expectations. The student may then create a plethora of different simulations to observe if any conforms to experiments done in the real world. Programming allows for an iterative and exploration-based approach to learning physics and making models.

The paper begins with an overview of the theoretical framework used to analyze programming in physics education. A small study, with the aim to investigate the claims

*Kim.Svensson@fysik.lu.se

of the theoretical framework, and its results are presented. The discussion focuses on explaining the results using the theoretical framework and concludes that the framework offers a new and innovative way of describing the learning experience provided by programming in physics education.

II. BACKGROUND

Programming is an important aspect of current physics research and is not new to physics education. In the 1980s it was used by Seymour Papert [3] with the Mindstorms system, which was later adopted by LEGO in their Mindstorms [4] production line. Programming was also used with the MUPPET program [5,6] in physics education as a way to explore different concepts found within the physics discipline. With the introduction of higher level programming languages, more research into the use of programming in physics education has been performed, for example, Refs. [7–10], which focused on fostering computational thinking. Using programming, many different animations and simulations have been created and their usefulness for conceptual change and physics education have been investigated [11–13]; with the findings that animations and simulations help students understand concepts better. This is true not only in physics education, but in other science education settings as well [13–16]. Kuo-en Chang *et al.* [11] found that allowing the students to formulate their own hypothesis about a physical concept and then test the hypothesis allowed for more conceptual change than a step-by-step instruction when using a simulation.

Several programs have been created with the sole purpose of being used in education, in various disciplines, such as MuPPET and NETLogo [5,17], and the American Association of Physics Teachers [18] argues that programming and the knowledge of creating and using simulations to investigate and explore models of physical phenomena, should be a crucial part of modern physics education. However, programming and simulations must be implemented into the courses and into the curricula into a meaningful and useful manner [19–22]. The focus of the implementations, performed or analyzed in Refs. [19–22], have been to foster computational thinking in the student, where programming can play an important role, but is not required. Programming has been recognized to have the potential as a great tool for physics education [6]. However, the implementation, usage, and goals when using programming differ from location to location and from teacher to teacher.

The theoretical aim of this paper is to look at programming as a phenomenon and its specific usefulness in physics education through the lens of *social semiotics* (see below). It is the combination of coding, visualization, and interactive activities that gives programming the versatility to be used in many different fields of physics research such as cosmology, fluid dynamics, and atomic physics. By studying not only the code, but the visualization and how to interact with the program, it is believed that a much richer understanding of the potential use of programming can be gained. This larger view of programming is also the view taken by The Swedish National Agency for Education (Skolverket, Sec. 1. 3 [23]).

III. THEORETICAL FRAMEWORK

Below is a description of the theoretical frameworks used in the analysis of programming. The analysis combines social semiotics (Sec. III A) and the variation theory of learning (Sec. III C) and finds their ideas useful for describing programming as a means for meaning making in the physics classroom.

A. Social semiotics and programming

Social semiotics [24–27] is a theoretical framework built around understanding and investigating group meaning making and the resources that are used to create meaning through communication. The resources are called *semiotic resources* and encompass "representations, tools, and activities used to create or derive meaning in specialized groups" [25]. Using this definition, we may look at programming as a means for communication between student and program. A student may ask questions of a program to get an answer, or as a means to construct new representations or tools. However, programming is not a semiotic resource; instead it should be seen as a *semiotic system* [28] because programming can be used to describe many different scenarios and be used to extract many different answers to many different questions. A specific semiotic resource is used in a specific scenario to convey a specific meaning, such as a time-velocity graph or a specific circuit diagram. A semiotic system is a system of communication that is qualitatively different from other means of communication. The communication system "image" is a semiotic system that is qualitatively different from "text" which is another semiotic system. However, text can be used to convey different meanings in different situations: When a semiotic system is applied in a specific scenario, a semiotic resource is created or extracted from it. An author uses text to write a book, the book is the semiotic resource and the text is the semiotic system used to produce the semiotic resource. Many different semiotic systems may be used in tandem to create a single semiotic resource such as this paper which uses the semiotic systems text and image to convey meaning in a disciplinary relevant manner.

Programming can be described as a semiotic system used to create or investigate other semiotic resources. By using programming it is possible to move between different semiotic resources and between different semiotic systems, such as taking a long list of data points as the input and produce an animation as the output. The programmer has transformed a semiotic resource (a specific list of data

points) in a semiotic system (list of numbers or data) to another semiotic resource (a specific animation) in another semiotic system (animations). This kind of transformation is called *transduction* within the *multimodality* framework [29], or a "re-representation." Transductions are important in physics education [30–32] because they force students to discern the relevant aspects represented in different ways. Transductions can be complicated or hard to grasp and students should be given the time to explore and understand them [33]. Programming is well suited for student-controlled transductions because they perform the transduction at every step of the implementation, from the initial mathematical model to the visualization on the screen. The importance of using multiple representations for enhancing learning has been explored by, e.g., Refs. [34,35], who found that when and how students use multiple representations plays an important role in student learning. Often, but not always, the use of multiple representations has been found to be beneficial for student learning. This is also confirmed by the social semiotic framework, where Refs. [25,28] model this in terms of "critical constellations" of semiotic resources.

B. Affordances and programming

Affordances is a term used to describe what different objects offer a student that interacts with the object [36]. If a student interacts with a bottle, they may get the urge to "drink" or to "pour" or, if the bottle is empty, to "throw away" or "recycle" or to "fill" the bottle. These are all examples of affordances of the bottle. However, if another student interacts with the bottle, they may extract other meaning or urges from the bottle. What the bottle affords the second student differs when compared to what it affords the first student. This difference can be explained by how the two students discern different affordances. Which affordances they will discern depend on their prior knowledge, profession and many other factors such as their mood and the setting they are in. The bottle has a multitude of different affordances, but which affordances are discerned depends on the student interacting with it. For example, a professor in particle physics will discern some disciplinary relevant meaning from the formula

$$A_{\mathrm{PV}} = -m_e E \frac{G_F}{\sqrt{2}\pi\alpha} \frac{16\sin^2\Theta_{\mathrm{cm}}}{(3+\cos^2\Theta_{\mathrm{cm}})^2}\left(\frac{1}{4} - \sin^2\theta_{\mathrm{W}}\right).$$

The professor's discerned meaning probably differs from what a novice in the physics field may discern from the same formula.[1]

Programming offers the student the opportunity to modify the code with the intent to increase the discernibility of different aspects. What a student discerns is based on their ability to extract meaningful information from the resulting representation; by modifying the representation, students may discern relevant aspects more easily. Programming also requires that each part of the implementation is made explicit, and therefore requires discernment of its different parts, and opens up the possibility for learning [38]. The theory of affordances has been put to use in physics education by Refs. [25,39,40] and has morphed into disciplinary [41] and pedagogical affordances [42] which describe how well a semiotic resource can be used, or is used, in the discipline or as a pedagogical resource.

This paper does not use the term affordance that Norman [43] introduced, which states that affordances are only related to the physical interaction between the actor and the object. This paper uses the term affordance as describing anything that an object allows an agent to discern from it. Thus, it is possible to add and remove affordances as well as change the existing affordances by modifying an object. This use of affordances is much closer to how the social semiotics and the multimodality [29,44] communities use the term and can be read as the "meaning potential" of an object.

1. Semiotic resources and affordances

Semiotic resources used in teaching and learning offer certain meaning for the student to discern, or intended meaning. We may look at what a semiotic resource offers and what a student can discern from that semiotic resource to ascertain how well they understand a specific concept [40]. If a semiotic resource does not offer a specific meaning-making affordance, no student may discern that meaning from it. However, if the semiotic resource was modified, it could gain the specific affordance needed to convey the intended meaning and be used in communicating and understanding the intended meaning. A modification of a semiotic resource could be as simple as a person saying "This pen is a spaceship," which allows a student to discern spaceship-relevant aspects from the pen. A change in the semiotic resource is accompanied by a change in its affordances. See Fig. 1 for a visual demonstration of how a change in the semiotic resource also changes how well specific meaning can be extracted from the semiotic resource. In Fig. 1, no new information was added in the transformation, only how the information was represented. The affordances are separate but related to the information and meaning contained within a semiotic resource. However, changing the affordances of a semiotic resource is no precise art and is mostly guided by conjecture and educated guesses.

2. Modifying the semiotic resources

Whenever a semiotic resource is modified, by adding color, gestures, description, or any other modification, the affordances of the semiotic resource are modified or adjusted. The change may increase, decrease, or remove the discernibility of an affordance. Programming allows the

[1]The formula describes the probability of two electrons scattering off each other through Möller scattering [37].

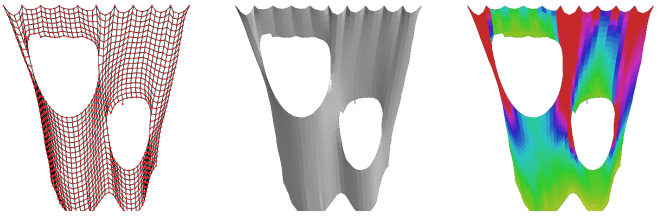


FIG. 1. A simple simulation of hanging cloth is visualized in three different ways. The left visualization shows the structure of the simulation, how the particles and springs are connected. The middle visualization shows the overall structure of the cloth and highlights larger deformations using the shading. The right visualization shows the magnitude of the forces each particle experiences.

student to modify the semiotic resource they create in any way they see fit (given the appropriate knowledge and ability) and, as a secondary effect, modify the affordances in any way they desire. This allows the student to create semiotic resources that allow the student to discern the specific affordances they aspire to discern. If a semiotic resource is not clear enough in its meaning, the student may modify it to create a new semiotic resource so the discernibility of a specific meaning-making affordance is increased, thus making it discernible to the student.

Within the multimodality framework [29,44], it is possible to change a representation by modifying it, or by re-representing it. If the change occurs within the same mode, such as rewriting a text, it is called a *transformation* [29]. If the change takes the representation from one mode to another, such as moving between formula and graph, it is called a transduction [29,32]. Social semiotics have adopted these terms and are using them to refer to different types of changes to semiotic resources. The importance of these changes or modifications can be understood from the variation theory of learning discussed below.

C. Variation theory of learning and programming

The variation theory of learning [38,45] states that to learn something, that something must first be discerned as its own aspect and to discern it, the student must experience variation about said aspect with respect to a static background. Marton [45] presented a good example about learning colors that highlights this. It is through the variation, compared to the static background, that the specific color stands out and can be discerned. Only by comparing to what it is not, can the color be identified as something it is. By varying the aspect a student should learn, that aspect becomes discernible and becomes possible to learn.

Programming allows for quick and easy variation among different variables and structures. By changing the mass of particles in a simulation, a direct effect can be discerned in the simulation. Perhaps the particle sinks, perhaps it floats, the change in mass will be discerned, experienced, and, potentially, understood. New questions may arise when old ones have been answered. Not just the variables can be changed, but also how the learner interacts with the simulation and how the simulation is represented. Programming provides ample opportunity, and quantifiable ways, to open up new dimensions of variation for the student and it also allows for the exploration of said dimensions of variation.

D. Programming as a tool for meaning making

Programming may be used as a tool for meaning-making in physics education in the same spirit as mathematics is used as a tool to investigate and understand physics. Through the act of implementation, the ideas and models of the students are made explicit and necessarily dissected into smaller understandable pieces that can later be joined together to form the whole model or idea. The pieces can also be modified, both internally and externally. Internal modification changes how a piece functions, for example, changing the interaction between particles. External modification means how the different pieces fit together, in what order they are placed and called.

The statement "Energy of the system is conserved" is an external piece, it gives information about how the system interacts with the outside world, but it does not give any information about the nature of the energy within it. The internal piece would describe the energy in the system itself, its potential, kinetic, thermal, or chemical energy and how they transform into each other. Programming provides the student with ways to explore both the external and internal parts of the concept they are implementing. They can explore which phenomena emerges and which interactions need to be explicitly inserted.

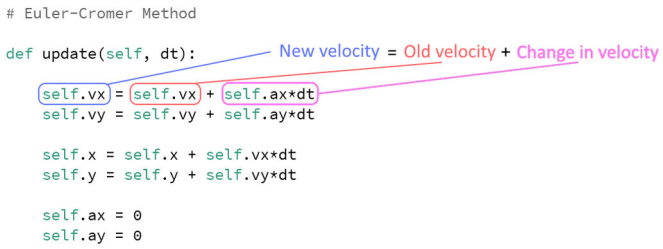
1. Example: The update() function

Within the particle class, created during session 2 of the workshop, is a function that updates the particles position and velocity based on the acceleration of the particle during each timestep.

In Fig. 2, the connection between position, velocity, and acceleration is made explicit by reading the code: The new velocity is equal to the old velocity plus a change in the velocity (acceleration) during the timestep. And the new position is equal to the old position plus a change in the position (velocity) during the timestep. The relationship between the concepts of position, velocity, and acceleration is made explicit and understandable through the use of programming. See Appendix B for an overview of the particle engine and the particle class used in the study.

E. Kolb's learning cycle

Programming's introduction of an iterative approach to physics modeling and understanding is well matched by Kolb's learning cycle [46]. Kolb describes the act of learning as a process where the student moves between




```
# Euler-Cromer Method

def update(self, dt):
    self.vx = self.vx + self.ax*dt
    self.vy = self.vy + self.ay*dt

    self.x = self.x + self.vx*dt
    self.y = self.y + self.vy*dt

    self.ax = 0
    self.ay = 0
```

FIG. 2. The update() function of the particle class uses the Euler-Cromer method for integration. The connection between position, velocity, and acceleration becomes explicit when implemented into code. Velocity is used as the "changer of position during a timestep" and acceleration is used as the "changer of velocity during a timestep." The acceleration is calculated in a separate function, applyForce(), which extracts the acceleration from all the forces a particle experiences. The acceleration is reset between each timestep to avoid an "impetus-like" force.

different phases of the cycle. The different phases, in order, are as follows:

- Concrete experience and observation: Performing an experiment or having a realization.
- Reflection: Reflecting on the concept or observation and its connection to theory.
- Abstraction: Formation of abstract concepts and generalizations.
- Hypothesis: Testing implications of concepts in new situations.

The cycle moves from concrete experience to reflection to abstraction to hypothesis and back to concrete experience. As a student learns, they may enter this cycle at any point and move through the different phases as they learn about different concepts. Programming fits well into this cycle because the implementation of simulations often takes on this cycle, or iterative, approach. The act of observing the simulation provides opportunity for reflection: "Does it do as I want?", which in turn leads to abstraction: "If I change the constant to a linear term that depends on the distance…", which can then be tested using the program. The cycle can describe very large concepts that takes months or years to learn, or very small aspects such as learning the meaning of a for loop.

However, Kolb's learning cycle also provides a checklist of learning opportunities that should be provided to the students in order to facilitate learning. If the students do not have a moment to reflect on their observation, they will not progress to abstraction or hypothesis. Programming, in its very structure of implementation and testing, provides the opportunity for the student to move through Kolb's learning cycle at their own pace.

F. Summary of theoretical frameworks

Multimodality and social semiotics provide a language for talking about semiotic resources and how to modify them through different transformations or transductions. As code is implemented and simulations are visualized, the student moves between many different modes and performs many transformations and transductions with an obvious one being the transduction from formula into code and code into visualization. As the students construct their own simulations they create their own visualizations, or representations of the phenomena. The student-created representations then form a basis for explorations of the simulation through discernment and by interacting with using the mouse or keyboard. The interactivity and the discernment process provide the student with an environment where new questions may be asked and modifications to the code can be done to explore these new questions. Kolb's learning cycle describes this process well and it is through the use of variation (in the code, visualization, or by interaction) that new scenarios emerge that afford the student new meaning to understand and discover.

IV. RESEARCH QUESTIONS

The theoretical framework, described in Sec. II, and programming's potential synergy with said theoretical frameworks, provided us with some aspects that we have looked for in this study:

(1) How does programming help students to make predictions about their model or system?
(2) In what ways do the students create variation in the visualization to increase the discernibility of different aspects?
(3) How much programming knowledge do the students think is needed to use programming to explore and implement different physical concepts?

It is these learning predictions that make programming a potential tool for meaning making for physics education. However, the predictions are based on a proficiency in programming because a certain knowledge is required to perform the modifications. The study also aims to see how much programming knowledge is needed to extract meaning (about physics) from the simulation and to make changes to the simulation.

V. METHOD AND ANALYSIS

The study focused on qualitative observation of six upper secondary education students' actions and interactions with a workshop designed around creating physics simulations using the programming language Python [1] and using the Processing IDE [2]. The participants volunteered for the workshop after a quick visit to their class where the research and the workshop were described. The participants all came from the same physics-focused class and were familiar with each other. Students S1, S3, and S5 all had prior knowledge of basic programming for the workshops. S2, S4, and S6 knew about programming but had no practical experience. All the participants were between

17 and 18 years of age, half of them were female and half of them were male.

A. The workshop

The workshop consisted of five different sessions, each session was two hours long with a short break in the middle and the four first sessions were designed to introduce a specific part which was needed to implement a physics simulation and the last session was reserved for interviews and discussions. The code and details for the workshop are provided in the Zenodo database [47]. The structure of the workshop and descriptions of each session can be seen in Table I.

B. Data acquisition method

To obtain useful data from the students, several different activation methods [48] were used, such as peer discussions, code along, projects, and interviews. Each of these activation methods were designed either as a way to provide a new take on programming in physics, or as a way to extract information from the students by making them explain or discuss their ideas, problems, or thoughts. The whole workshop was video and audio recorded using high definition GoPro Hero 6 cameras and several Olympus WS-852 digital voice recorder devices. The students were also interviewed during the last session. The interviews were divided into single and group interviews and focused on open-ended questions. The questions can be found in Appendices A 1 and A 2 and are designed to provide the participants an opportunity to speak freely about programming, physics education, and the workshop as a whole. The group interview also included a problem, related to programming and physics, for them to discuss.

TABLE I. A list of the five different sessions of the workshop with the content of each session. During each session a different aspect is investigated or explored with the students. Each session builds on what was learned in the previous session.

Session 1	Introduce the notion of animation by incremental changes between frames and how to display different shapes on different locations in the window. Updating attributes between frames to introduce velocity and acceleration and ending the session with a ball bouncing in the window.
Session 2	Create a particle-class that is based on the code written in Session 1. The particle can show(), applyForce(), interact() and update(). The session ends with hundreds of balls bouncing in the window.
Session 3	Start with a group problem-solving session. The problem was: *"Create a model that replicates the simulation shown in here"* (Fig. 3). A model of the hanging cloth problem was then implemented which was based on the student's ideas.
Session 4	The students were asked to come up with a model for two-dimensional heat-diffusion and to implement it by themselves. The lecturer's task was to guide the students around potential pitfalls and help them with specific programming questions.
Session 5	Solo interviews as well as group interviews with the participating students. Questions for the interviews can be found in Appendices A 1 and A 2.

1. Peer discussions

The students were asked to come up with a model that would mimic the behavior of a simulation shown on the projector, see Fig. 3. The simulation was a model of a piece of cloth, hanging at the top of the display window and pulled down by a simulated gravitational force. The students could interact with the simulation using the mouse by pressing either the left mouse button or the right mouse button and dragging the mouse across the cloth, see Fig. 3. The students were divided into groups of three and asked to come up with a model, based on the particle simulation created in session 2, with the aim to model the behavior of the simulation. They had thirty minutes to come up with a model that would reproduce the phenomena discerned in the finished simulation. The students had papers and a whiteboard to discuss their ideas and the discussions were aimed to activate and increase their learning, but also to make them explain their thoughts to each other. The peer discussion exposed their problem solving process which was documented and analyzed. The discussions were audio and video recorded using one stationary camera per group and a mobile camera that could capture unexpected events not contained within the field of view of the stationary cameras.

2. Code along

The workshop used the new concept called *code along* commonly used in online lectures, see, for example, Ref. [49], where a small piece of code was coded live,

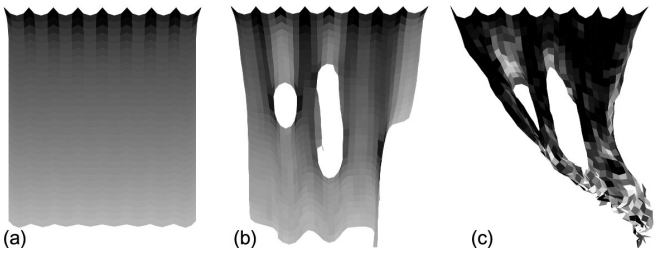


FIG. 3. A piece of cloth is simulated using particles that interact with the nearest neighbor with a force based on Hooke's law. The color shading represents how much a spring is elongated with light color representing small elongation and dark color representing long elongation. (a) A piece of cloth is hanging, only influenced by gravity. (b) The piece of cloth is cut using the left mouse button, the cloth reacts in real time. (c) The torn cloth is pushed around by the mouse using the right mouse button.

with the students, explained and explored. The code-along structure was designed to keep the students active and ensure an "I can do this myself" atmosphere by making the students write the code themselves, and by making sure that they got help when they made errors. During the code-along sessions, moments were devoted to explore the code, both in a guided scenario and by allowing the students to freely modify the code. In the instructed situation, the students were asked to vary a specific aspect and asked to observe how that variable affects the simulation. In the free situation, the students could change whatever they wanted, with the aim that the students would investigate some interesting aspects of the simulation. The students were video and audio recorded during the lectures, as was the lecture itself.

3. Project

During the fourth session the students were asked to come up with a model for heat diffusion, implement the model, and study the results. The students were encouraged to work in small groups, the same groups as in the peer discussion, to ensure verbal discussions and explanations. The project aimed to see how well the students could adapt the other parts of the workshop. The whole project session was video and audio recorded using two stationary cameras and one mobile camera, and several microphones.

C. Analysis

The analysis of the study was based on the recordings, visual and auditory, from the workshop, but also from field notes taken by the lecturer or researcher during the workshop. The aim of the analysis was to identify and analyze the student's problem solving processes around the code, how they interacted with it, what they discussed, and how they approached problems related to the code and to the physics. Special care was taken when observing how they represented their models and their simulations and what changes they made to the code to create new representations.

The analysis uses a qualitative research approach, inspired by grounded theory and the constant comparative approach [50,51], that are currently being drawn on for educational interpretive studies (e.g., Refs. [52–54]), and aims to discover a theoretical structure related to the learning process of using programming in physics education. We use a cyclic approach by relating larger observed structures to smaller details and vice versa, which ensures a coherence of the underlying theoretical ideas that emerge. The videos were cut into smaller clips with the intent to extract interesting interactions or events that pertain to programming and/or physics learning or exploration. The clips were transcribed multimodally [44,55] and relevant learning structures were identified. This is an interpretive grounded theory approach, as discussed in Ref. [51], were the observed structures are interpreted using existing theories, such as social semiotics and variation theory. The extracted data was discussed and interpreted with experts within the physics education research field at Lund University. By iterating this process of identification and description, we eventually obtained a saturated description where all the interesting phenomena have been categorized, described, and organized by using a grounded theory framework in combination with social semiotics and variation theory; see Secs. III A and III C.

1. Representations

By studying how the students represented their simulations and how they choose to interact with them, it is possible to get a glimpse of what the student may or may not discern from the program. If a student changed how to visualize a simulation, it was because of some reason. That reason could be that the students wanted to highlight a specific aspect of the simulation, or that the first representation contained too much information and it was hard to discern anything because of all the clutter. It should be noted that the default representation in the physics engine displays a colored circle for each particle. This is often adequate in most situations, but if used to construct a gridlike structure, like in Fig. 5, it would quickly become cluttered and a new representation is better suited, such as using a wire-frame structure or filled parallelograms as seen in Fig. 3. Because of the instant feedback nature of programming, students may enter into an instant feedback loop, were they study their representation, change something they wish to highlight, study the new representation, change it again based on the new information, and so on. By observing this feedback loop, the students' focus could be determined and how this focus changed during the loop, indicating that the students have learned something that made them shift focus.

2. Affordances

Affordances describe what a student discerns from a specific semiotic resource, or in this case, representation. By studying what a student discerns from a representation it is possible to obtain knowledge about their knowledge about the object from a certain discipline's perspective. However, affordances will be used in a different manner for the analysis in this paper. The qualitative affordance analysis will look at what a student aims to discern and what changes the student performs to a representation to be able to discern that affordance. That is,

> *How does the student modify the semiotic resource so that the discernibility of specific affordances are changed?*

By looking at how students manipulate representations, a connection between what they perceive to be important, their relevance structure [38], and what the representation affords can be seen.

3. Extracting relevant student interactions

From the transcripts and from observing the videos, interesting discussions and interactions were identified and extracted from the mass of data by the first author, following the analysis method described above. Through discussions among the authors, interesting passages were chosen in such a way as to reflect the students' actions that pertain to both programming and physics. The chosen excerpts show different learning sequences by the student interactions among each other and with the code, such as figuring out a solution, asking investigative questions, discussions, the problem solving process of implementing the code, or creating a model. Data that are unrelated to these aspects were weeded out in the process, for example, when the students discuss what they plan on doing in their spare time. From the theoretical frameworks of social semiotics we know the importance of the interactions and discussions, but also how very small modifications may play a significant role in the learning process. We aimed to extract data that capture both situations.

It was also important to gauge the students' overall ideas about the workshop, programming, and physics, because their expectation, prior knowledge, and perception of the environment where the workshop and data collection took place will inform how they react to the content of the workshop. To extract this information we asked what they thought about the workshop and what programming proficiency would be required to participate in the workshop. This was done in the interview during session 5 using the questions found in Appendix A.

VI. RESULTS

The results presented here are seen through the lens of social semiotics and the parts that make up social semiotics, such as transductions, affordances, and semiotic resources. During the workshop, the participants performed a series of different activities and experienced different methods of activation. It is through these different activation methods and the qualitative analysis of the recordings and notes that the results have been constructed. The qualitative analysis identified the following aspect represented by the actions of the participants: transduction, variation, unpacking formulas, predicting, and iterating.

From the interview with the students, we found that the students were happy with the pace of the workshop and they thought the level of the programming and physics was good. Some commented on the need for basic programming knowledge to fully make use of the workshop, but that it was easy to follow the instructions. See Sec. VI E for more thoughts on the programming proficiency of the students. It thus appear that the setting, pace, and content of the workshop itself did not pose a hindrance to the student's ability to program or express themselves. Student comments are discussed in more detail in Sec. VI E.

A. Hooke's law

The students moved between different semiotic resources with relative ease when guided by a teacher. When writing the applyForce() the students implemented $\bar{F} = m\bar{a}$ and extracted the acceleration from an external force, $\bar{a} = \bar{F}/m$. During the implementation, they unpacked the formula and realized some information hidden in the notations and its structure; its two-dimensional nature and that the mass cannot be zero. The students followed along with the transduction from a formula to the implementation of said formula. In the third session of the workshop, the students implemented Hooke's law into code: $\bar{F} = -k\bar{x}$, where $\bar{F}$ is the force resulting from the displacement $\bar{x}$. The students had no discernible problems following the transduction presented in Fig. 4.

The transduction also highlights that there are two parts to $\bar{F}$, a magnitude and a direction. `dx` and `dy` from the code provides the direction and `F=-k* (le*—rest_le)` is the magnitude of the force.

B. Forces and $\bar{F} = m\bar{a}$

In the applyForce() method the applied force is converted into an acceleration and added to the current acceleration. The transduction highlights how to move from a force to a resulting acceleration. S1 commented the following on implementing forces on a simulation of a Frisbee, translated from Swedish to English:

S1: *For example, I just did a project with a Frisbee… and there I could go in and check… to see if what I had written by hand and implemented was correct. I had to think extra on the forces when I added them to the Frisbee.*

This implied that having to implement the formula into code, or performing the transduction, provided the student with a learning opportunity that had not been apparent before. Thus, the transduction of the force required the student to unpack the formula and identify its different parts to be able to implement it correctly.

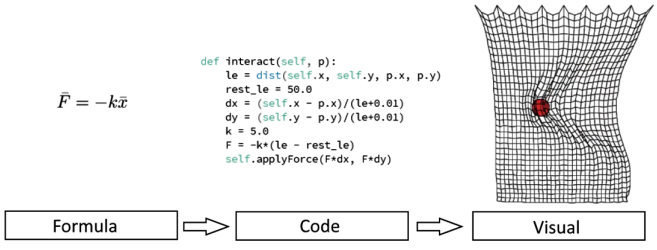


FIG. 4. Programming simulations requires transductions between different types of semiotic systems that each provide different meaning potential. To move between different semiotic systems requires the student to define the transduction explicitly and highlights each aspect of the system to the student.

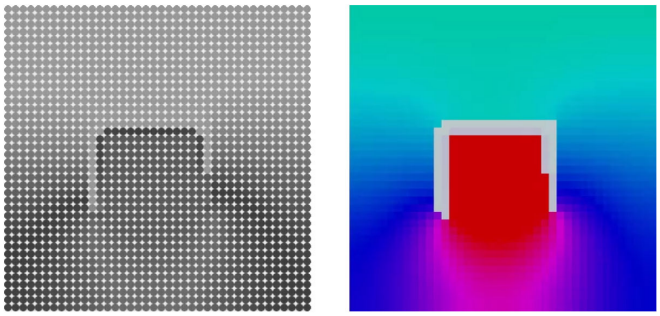


FIG. 5. S3 student started with the left visualization and modified it, in several steps, to end up with the right visualization. This is a recreation of the student's code and may differ in the precise final result, but the transition is the same—going from a black and white representation using circles to a colorful representation using squares.

C. Modifying the visualization

During the fourth programming session, S3 had managed to create a correct implementation of a heat diffusion simulation. However, S3 was not happy with the visual representation of the simulation and aimed to change it. See Fig. 5 for screenshots of how the visualization was modified. Programming provided S3 with the ability and the opportunity to modify a representation, something that books and static images do not provide.

Another student commentated on the ability to connect the visualization to attributes of the particles:

S4: *I made the temperature depend on… no, the color depends on the temperature. I placed self.t* [the temperature of the particle] *as the red color.*

As the student explained the idea behind the modification, they had to reflect on their implementation and understand how it works. The reflection is triggered by the student's requirement to match their explanation to their implemented model.

D. Internal modeling and predictions

Several of the students were able to make predictions of their yet-to-be implemented models and compare their predictions with the real world and/or other simulations. This shows an understanding and an ability to internally model the computer program in their heads, run it and compare the expected result with a reference as seen in sections VI. D. 1 and VI. D. 2.

The students' internal modeling and their ability to compare it with their models allow the student to iterate on their model. This was seen in the students' approach to implementing their models. By discerning how changes affected their model, an iterative process began, which allowed the student to get feedback from the model and adapt accordingly. The feedback loop also allowed the students to adjust their expectations or to understand parts of their model.

1. Heat diffusion

During the last programming session, the students were asked to implement a 2D heat diffusion simulation using the programming structure that had been produced during the previous sessions. As a student managed to implement a working model of the heat diffusion they exclaimed (translated to Swedish from English):

S5: *Yes!* [S5 puts their hands in the air] *It does what I want!* [S5 stands up to celebrate.]

The other students joined in with the celebration and could see that S5's implementation was correct. The realization that the simulation was correct came from a visual inspection of the representation S4 had coded. From the visual representation, S5 and the other students could discern that the implementation was correct, or at least reasonable. They did not need to see the code or how it was implemented, but only the visual representation of the program itself. The students were able to distinguish between an incorrect implementation and a correct implementation through the visualization itself and they could predict what a correct visualization would look like based on the expected behavior of their model.

The students then continued by examining the simulation closer:

S5: *It does what I want. So, theoretically, it will spread out.*
S3: *Ok, I'm coming to check… what have you done?*
S4: *Does it bounce against the wall?*

By observing the working simulation, new questions sprung up in their minds as they saw the thermal energy spread out among the particle: Would the thermal energy diffusion bounce at the wall? Questions that they, nor the lecturer, had not thought about before emerged and programming provided a way to answer them. The students entered into the positive feedback loop as soon as one step of the implementation was completed and began to investigate new aspects of the simulation.

2. Hanging cloth

During the group discussions in session 3, where the students were tasked with coming up with a model for a hanging-cloth simulation, see Fig. 3, the students discussed the forces in the cloth (translated from Swedish to English):

Gesture [S5 makes a gesture where two particles come together and pushes them apart]
S3: *No, it should not be a force outwards there.*
S5: *Yes, because they don't want to be together.*
S3: *It's not a slime, it is cloth. I can do this.*
Gesture [S3 takes part of their shirt and pushes it together into a ball shape.]
S3: *There is no force outward now.*
Gesture [S5 does the same with their shirt.]
S5: *Yes, if I release…*
Gesture [S5 releases the shirt and the cloth spreads out.]
Gesture [S3 releases the shirt and the cloth spreads out.]
S3: *Oh, it does spread.*

The students S3 and S5 were exploring the nature of the forces in a piece of cloth and how they would model it when the question became *"Does it have a force pushing out or is it only pulling in?* The question was resolved when S5 took a piece of their shirt, pressed it together into a small clump, released it, and saw that the shirt expanded. Through their arguments they identified a question in their model and used experiments in the real world to get an answer. The group had come up with a prediction of their model and tested it using the real world.

Another group did the same procedure, but instead of asking the real world for answers, they asked a simulation. From their own model, they made a prediction and tested the prediction using the simulation. In this case, the prediction was that the hanging cloth could not be pulled below its lowest point, since that would imply a springyness in the forces describing the cloth, something the group did not have in their model at this time.

S4: *Can you throw the curtain above?*
S1: *What?*
S4: *… is it possible to throw the curtain up completely?*
S1: *But it can go down, then it [the simulation] does not work if we can pull it down.*
S4: *Can you pull it down?*
S1: *It can be pulled down more than it is.*

Gesture [S1 uses the mouse to drag the curtain downwards.]

S1: *It can move down.*
S4: *That is a bit strange, maybe.*
S2: *Yes, but that would work here too if all* [the particles] *move.*
S1: *But, then it does not work…*
S2: *It can still work, they still move freely so that means that if they are on the sides from the start, they can move downwards.*
S1: *Eeh.*

The conversation continued about the model, but the main point is that the students interacted with the simulation, compared it to their yet-to-be implemented model and found that they differed in their function. It was also through the interaction with the simulation that they observed a phenomenon, the cloth being pulled and elongated downwards, that clashed with their own model. In this scenario, S2 realized that their model would be able to accommodate the new phenomenon, but S1 was not so sure.

Both groups made predictions about their models, found ways of testing their predictions, and updated their models based on the observation. The only difference was that one group used the real world and one group used a simulation to answer their questions. The students used both the virtual model and a real life model to make experiments from which they extracted information and drew conclusions.

As one group began discussing their ideas for implementing the cloth simulation, S2 dismissed the idea to use springs at first because it was not something they had experienced in the workshop. This mindset, to not use ideas or material from "outside," is seen in many educational settings and this workshop was no exception.

S2: *Like, if we define these particles to have two times the radius…*
S4: *Yes.*
S1: *What was it we did with the spring constant?*
S2: *I don't think it's relevant, I think we should use what we have worked with* [in the workshop].

However, S2 soon realized that using a springlike force between the particles may allow them to model the cloth and changed their perspective to include information from outside the workshop, such as their prior physics knowledge.

S2: *Can we not have a lot of balls sticking together.*
S4: *Yes, that is what I thought, we did something where we had a force that pulled two balls together.* [S4 is referring to a gravitational simulation as a test of the interact() method, implemented during session 2.]
S2: *Yes, and using the spring constant.*
S4: *Yes, maybe.*

E. Answers to interview questions

Here follows some selected answers to the questions about programming and physics and how they can or want to use it in their physics education. The questions can be found in Appendix A.

Question: What does programming give you, that you could not do in any other way?

S2: *…something else I thought about… that programming gives another angle on the physics. Often, you have exercises you have to solve, and that is the case in programming as well, if we would simulate a pendulum, but its much more… vague. There are different ways to do it. Instead of just solving something, you create.*
S1: *It has given me, that I can take a phenomenon or problem or … anything … from physics. Implement it and visualize it and … figure out answers and see if I've done it correctly.*

This student, S1, has seen that the physics engine created in the workshop can be used to simulate many different physical systems and scenarios. The ease of using the system and the feedback it provides, ensures that the student can discern and interpret the representation accurately.

S4: *… usually you just sit and calculate, there are no moving pictures and you can't interact with your calculations. But in this workshop you got to write your calculations in the form of code but you could also see how it worked in real-time so to say.*
S4: *It is as I said in the beginning. You get a chance to experience physics… I mean, you get the theoretical part but get to perform it… you get to see it in motion.*

The ability to see the equation in motion through an animation provided an extra layer of potential meaning-making compared to their normal physics education.

S1: *What I thought was good was, within physics, is… when we have worked, with forces, you have to think a little extra when implementing them into code… what directions. The good thing is that you get instant feedback if you've… if you've done it correctly or not*.

The instant feedback that programming provides, and the explicit nature of the code itself, gives the student a platform where misunderstandings and errors are easily discerned and corrected.

Question: What knowledge do you think is required to fully use the workshop—More programming?

S2: *I feel that for physics, you only need to understand physics, but here you need programming… at least the basics of programming*.

S1: … *If you haven't programmed before it'll take some time to get into the programming before you can get going with the physics*.

The sentiment, that an introductory course in programming was recommended in order to code and modify the simulations and use them for exploring physics, was mirrored by the other students in the group discussion.

The answers to the question "Can you explain the "Particle" -class?" was mixed but tied to the participants prior knowledge of programming. The students that had done some programming before could explain what the different functions did with greater confidence than the students that had no prior knowledge of programming. However, even students with prior knowledge could not fully recall exactly what the functions did. This lack of knowledge is attributed to the short time the students interacted with the program.

VII. DISCUSSION

The theory of social semiotics combined with the data gathered from the workshop have shown that the theory predicts what affordances programming exhibits and that the students were able to discern and use them to explore the physics phenomena at hand. From the analysis of the programming experiences by the students, indications of a richer use of programming for learning physics can be seen, especially if the students themselves are allowed to create and implement their own models.

- students were capable of creating, implementing, and extracting meaning from physics simulations.
- students with no prior knowledge of programming could implement their own models when guided by a teacher.
- students with some prior knowledge of programming could implement their own models without guidance from a teacher.
- students recognized their own ability to program and suggested that an introductory course in programming, which half of them had taken, is all that would be needed to make use of the programming in the workshop.
- some students highlighted that programming provided another approach to physics education compared to their traditional educational setting.

Students entered into a feedback loop as they tried out different variants of their code or model, discerned the result, and modified their code or model. In every step of the implementation of the model, the students have asked questions of the real world, completed simulations, not-yet-implemented code, and half-implemented code. The answers they received made them change the implementation or model. Either it was an error in the code, a typo, or a thinking error, a "thinko," that made them reconsider and change. Programming forces the student to reconsider their models until a functional model is produced.

Students could and did ask questions about their program, model, or implementation, interpreted the answers, and adapted their model. This process is the learning process as described by Kolb in his Learning Cycle [46]. Students changed the resulting semiotic resources to increase the discernibility of specific meaning-making affordances. The students interacted with the hanging cloth simulation to see if they could increase the discernibility of a certain affordance: "springy-ness." During their investigation they discerned a phenomenon they did not expect: The cloth could be elongated by dragging it, and they had to adapt their model.

Another student changed how the heat diffusion was displayed by changing the shapes of the visualization of the particles from circles to squares to reduce the clutter and thus increase the discernibility of relevant affordances such as the temperature distribution and the temperature gradient.

A. Modifying the affordances

The students interacted or modified the resulting semiotic resources to highlight different aspects, or, to make certain affordances more discernible. When the students interacted with the hanging-cloth simulation, they were unable to discern a specific affordance that would answer their question until they had changed the simulation by pulling on the cloth. The resulting animation then offered the students another set of affordances or made a specific affordance discernible. Specifically, the students used the mouse to interact with the simulation to see if they could pull it downwards, this specific aspect could not be discerned unless they interacted with the simulation. The interaction provided a new scenario where the students could discern that the cloth could be dragged down, but that it could also be pushed up above the attachment points. The new scenario answered one question but created a new one. This use of variation is well described by the variation theory of learning [45].

It is through changes and investigations that answers to questions can be obtained. As the students create new

simulations, new questions will arise that they wish to answer. To answer these questions, they will modify the semiotic resource they are interacting with, creating a new semiotic resource that provides the ability to discern the needed information to answer the new question. Social semiotics [24,25,27] and its semiotic resources are a great way of explaining the ways in which programming can be used to modify and create new semiotic resources that enhance the learning of physics.

B. Programming as a means for meaning making

Thanks to programming's ability to easily and quickly modify different aspects of a simulation, students can open up many different dimensions of variation to explore. The quick and easy exploration that programming provides allows the student to investigate and eventually understand how different aspects relate to each other and how they affect different parts of the simulation.

1. Forces and programming

Forces are an important but hard [56] concept to grasp for a learner. Forces can have many different sources but they all sum up to a net force which will describe the acceleration of an object. A force has two components, a magnitude and a direction, and this may be hard to grasp because both aspects are usually baked into the vector notation used to describe them. In programming, we can choose to make the two components explicit, as can be seen in Fig. 4 and in the answers given in Sec. VI E.

Programming is well suited to take advantage of the variation in the variation theory of learning thanks to its digital and repeatable nature. By changing a single variable in the code and observing the changes, the student is made aware, in an interactive manner, of critical aspects and can modify these to observe changes in the simulation.

Programming offers a wide range of possible transductions. One is transductions that take one semiotic resource from a semiotic system to another, such as going from a formula into an animation. The transductions performed when programming are explicit transductions where each relevant aspect has been considered and taken into account. The explicit transduction is done by the student, and each aspect is laid bare for the student to explore and investigate at their leisure. One explicit transduction can be seen in Fig. 2, where the transduction from mathematical integration is written in code, requiring the student to explicitly write the relationships between position, velocity, and acceleration.

Programming allows for quick and easy changes that affect the affordances of the semiotic resource produced by the code, either by changing the code or by interacting with the semiotic resource itself. New scenarios can easily be created and new aspects can be discerned from the new scenarios. Each student can create semiotic resources that are tailored to their individual questions and ability to discern. Programming thus allows for a wide dynamic range of affordances, some that will greatly enhance the possibility for meaning making and some that may detract from the meaning-making experience of the student.

By using a guide (teacher) when programming, relevant affordances can be made more discernible and students can get a powerful tool to use when investigating and constructing different models of many different physical phenomena.

VIII. CONCLUSIONS

We found that the theory of social semiotics in combination with variation theory can be used as a new way to describe and understand the usefulness of programming as a tool for meaning making in the physics classroom. The students in this study were able to successfully use programming to create simulations and use the process of creating and implementing models as a means for meaning making about different physics concepts.

As students developed their own models, they were able to test it at every step of the implementation. To test their programs, they needed to perform mental modeling to compare with the visualizations; programming helped them test their predictions and modify the system accordingly (RQ1).

To better highlight disciplinary relevant aspects, students modified the shape, color, and location of their visualizations. (See, for example, Fig. 5.) We found that the students could modify the visualizations in ways that enhanced their learning experiences by taking ownership of the visualization process (RQ2).

The students expressed that some prior knowledge of programming was needed to take full advantage of the programming sessions. However, the students without prior knowledge said that they could follow along without difficulty, but they could not as easily implement their own ideas (RQ3).

The theoretical framework illustrates the possible interplay between the semiotic systems: coding, interaction with the simulation, and visualizations. In this study we found that the iterative nature offered by programming facilitates productive transductions between these semiotic systems.

This work gives a few examples of how programming can be used to enhance meaning making in physics education. In a next step, we are offering professional development for teachers to learn the programming method. Their experiences and reflections on opportunities in classroom implementations, as well as difficulties encountered or expected, are captured through follow-up interviews.

ACKNOWLEDGMENTS

The authors thank the participants for their openness and their interest in the workshop and in the project itself. We also wish to express our gratitude to Vattenhallen Science Center at Lund University for providing the locales and computers for each session of the workshop.

APPENDIX A: INTERVIEW QUESTIONS

During the interview session the students were interviewed one by one with the questions in Sec. A 1, but their discussions where also studied during group interviews. The questions used in the group interview can be found in Sec. A 2.

1. Solo interviews

After the workshop, the students participated in solo interviews with the questions, translated from Swedish to English:

1. What do you think about the workshop as a way of learning physics?
2. What do you think you learned during the workshop?
   - What was good, bad, easy, or hard?
   - What knowledge do you think is required to fully use the workshop?
     - More physics?
     - More programming?
     - More tasks?
     - More demonstrations?
3. Can you explain the "Particle"-class?
   - Explain what the different functions do:
     - __init__()
     - show()
     - update()
     - interact()
   - What can they be used for?
4. What is it that programming gives You, that You could (perhaps) not do in any other way?

2. Group interviews

After the solo interviews, all students that were present for the final session participated in a group interview about the workshop. The interview aimed to start discussions among them to see if they could draw upon their programming experience to identify solutions and/or problems. The group interview questions, translated from Swedish to English:

1. What do You think are the pros and cons with programming compared to normal lectures in a classroom?
2. What do You think are the pros and cons with programming?
3. How do You want to or can use programming in physics?
   - What role does programming play in physics research?
4. (For Researcher): How do they use their computational thinking when analyzing the physics problem:
   - If You were to create a simulation of two colliding galaxies, what would you do?

APPENDIX B: THE PARTICLE ENGINE

The physics engine constructed by the participants during the workshop was based on the grid-free method of a particle based physics simulation. The particles are described by their own class with the following methods:

- __init__(x, y, mass, radius)
  - Initializes the particle with some attributes and values. This method is required by Python to initialize any object. This method is used to set initial conditions or default values for attributes of the particle. The particle always have position, velocity, acceleration, mass, and radius to make the other methods work. Other attributes are added by the programmer.
- show()
  - Displays the particle in a window. The default visualization is just a circle with a static color. The user can change how the particle is visualized by modifying this method.
- update(dt)
  - Updates the attributes of the particle using an Euler Cromer [57] integrator. The implementation can be seen in Fig. 6. The update method calculates a change in velocity using the acceleration, which in turn is also calculated by each timestep. The velocity is then used to calculate a change in position. The method is designed to explicitly show how the attributes are updated in each timestep and avoids some simplifications that can be made.
- interact(other)
  - Handles the interaction between particles. It then applies the resulting force on the particle using the applyForce() method. This is the main method that deals with different physical models such as gravitational interaction, Hooke's law or any other interaction between different particles.

```
def update(dt):

    self.vx = self.vx + self.ax*dt
    self.vy = self.vy + self.ay*dt

    self.x = self.x + self.vx*dt
    self.y = self.y + self.vy*dt

    self.ax = 0
    self.ay = 0
```

FIG. 6. The implementation of update(dt) avoids the use of vectors or syntax that could make it simpler. The aim of the function is to explicitly show how the attributes are updated during each timestep. During each timestep, new forces are calculated and a net acceleration is obtained, the old acceleration must be removed, to avoid an impetuslike effect.

- applyForce(fx, fy)
  — Add all the forces together and calculates a net acceleration using $F = ma$ rewritten as $a = F/m$. The method is called from the interact(other) method and is the primary way the user interacts with the particles.

The particle class and its methods are used in the draw loop, built into the processing IDE, to update and show the particles. See Fig. 7 for an example of the simulation loop. During each iteration, each particle interacts with all other particles, it then feels a force downwards (gravity), updates its position, and displays itself in the window.

The loop and the methods and the names of the methods are chosen in such a way that they are easily understood and each part has a well-defined purpose. Using this setup and the methods, it is easy to identify the different parts needed to implement the simulation and what parts are required to have a functioning simulation.

```
def draw():
    background(234)

    for p1 in particles:
        for p2 in particles:
            p1.interact(p2)

    for p in particles:
        p.applyForce(0, 9.81)
        p.update(dt)
        p.show()
```

FIG. 7. The loop that is iterated over each timestep of the simulation. The particles all interact with each other and then they update their position and show themselves in the window. draw() is called as fast as possible, or as fast as the display allows, which is usually around 60 times per second.

[1] G. van Rossum, *Python tutorial*, Tech. Rep. CS-R9526 (Centrum voor Wiskunde en Informatica, Amsterdam, 1995).

[2] B. Fry, C. Reas, and D. Shiffman, The Processing Foundation (2019), https://processingfoundation.org/. Accessed April 3.

[3] S. Papert, *Mindstorms: Children, Computers and Powerful Ideas*, Vol. 1 (Basic Books Inc., New York, NY, 1980), p. 230.

[4] G. Garber, *Instant LEGO MINDSTORMS EV3* (Packt Publishing, Birmingham, UK, 2013).

[5] J. M. Wilson and E. F. Redish, Using Computers in teaching physics, Phys. Today **42**, No. 1, 34 (1986).

[6] E. F. Redish and J. M. Wilson, Student programming in the introductory physics course MUPPET, Am. J. Phys. **61**, 222 (1993).

[7] M. Caballero, M. Kohlmyer, and M. Schatz, Fostering computational thinking in introductory mechanics, Proceedings of the Physics Education Research Conference 2011, Omaha, Nebraska (2011), https://www.compadre.org/per/items/detail.cfm?ID=11799.

[8] A. Gerestrand, Programmering som ett verktyg för lärande, Master's thesis, Gothenburg University, Gothenburg, 2017.

[9] H. A. Dwyer, B. Boe, C. Hill, D. Franklin, and D. Harlow, Computational thinking for physics: Programming models of physics phenomenon in elementary school, in *Proceedings of the 2013 Physics Education Research Conference, Portland, OR*, edited by P. V. Engelhardt, A. D. Churukian, and D. L. Jones (AIP, New York, 2014), p. 133.

[10] O. P. Sand, T. O. B. Odden, C. Lindstrøm, and M. D. Caballero, How computation can facilitate sensemaking about physics: A case study, in *Proceedings of the 2018 Physics Education Research Conference, Washington, DC*, edited by A. Traxler, Y. Cao, and S. Wolf (AIP, New York, 2018).

[11] K.-E. Chang, Y.-L. Chen, H.-Y. Lin, and Y.-T. Sung, Effects of learning support in simulation-based physics learning, Comput. Educ. **51**, 1486 (2008).

[12] K. C. Trundle and R. L. Bell, The use of a computer simulation to promote conceptual change: A quasi-experimental study, Comput. Educ. **54**, 1078 (2010).

[13] R. L. Bell and K. C. Trundle, The use of a computer simulation to promote scientific conceptions of moon phases, J. Res. Sci. Teach. **45**, 346 (2008).

[14] S. Bayraktar, A meta-analysis of the effectiveness of computer-assisted instruction in science education, J. Res. Tech. Educ. **34**, 173 (2001).

[15] J. P. Akpan and T. Andre, Using a computer simulation before dissection to help students learn anatomy, J. Comput. Math. Sci. Teach. **19**, 297 (2000).

[16] J. Huppert, S. M. Lomask, and R. Lazarowitz, Computer simulations in the high school: Students' cognitive stages, science process skills and academic achievement in microbiology, Int. J. Sci. Educ. **24**, 803 (2002).

[17] U. Wilensky, *Netlogo, Center for Connected Learning and Computer-Based Modeling* (Northwestern University, Evanston, IL, 1999).

[18] AAPT Undergraduate Curriculum Task Force, *AAPT Recommendations for Computational Physics in the Undergraduate Physics Curriculum* (American Association of Physics Teachers, College Park, MD, 2016).

[19] M. D. Caballero, J. B. Burk, J. M. Aiken, B. D. Thoms, S. S. Douglas, E. M. Scanlon, and M. F. Schatz, Integrating numerical computation into the modeling instruction curriculum, Phys. Teach. **52**, 38 (2014).

[20] J. M. Aiken, M. D. Caballero, S. S. Douglas, J. B. Burk, E. M. Scanlon, B. D. Thoms, and M. F. Schatz,

Understanding student computational thinking with computational modeling, AIP Conf. Proc. **1513**, 46 (2013).
[21] M. D. Caballero, Computation across the curriculum: What skills are needed?, in *Proceedings of the 2015 Physics Education Research Conference, College Park, MD*, edited by A. D. Churukian, D. L. Jones, and L. Ding (AIP, New York, 2015), p. 79.
[22] M. D. Caballero and S. J. Pollock, A model for incorporating computation without changing the course: An example from middle-division classical mechanics, Am. J. Phys. **82**, 231 (2014).
[23] Skolverket, Redovisning av uppdraget om att föreslå nationella it- strategier för skolväsendet—förändringar i läroplaner, kursplaner, ämnesplaner och examensmål [English: Swedish National Agency for Education: Presentation of the assignment to propose national IT strategies for the school system—changes in curricula, syllabuses, subject plans and degree objectives (2016)].
[24] M. A. K. Halliday, Language as social semiotic: Towards a general sociolinguistic theory, Lang. Soc. **10**, 169 (1975).
[25] J. Airey and C. Linder, *Multiple Representations in Physics Education* (Springer, New York, 2017), Vol. 10, pp. 95–122.
[26] U. Eriksson, Social semiotics in physics and astronomy education research—A spiral approach to teaching and learning, in *Proceedings of the CERN PER-Seminar Series* (CERN, Geneva, 2018).
[27] T. Fredlund, *Uppsala: Acta Universitatis Upsaliensis* (Acta Universitatis Upsaliensis, Uppsala, 2015).
[28] J. Airey and C. Linder, A disciplinary discourse perspective on university science learning: Achieving fluency in a critical constellation of modes, J. Res. Sci. Teach. **46**, 27 (2009).
[29] C. Jewitt, J. Bezemer, and K. O'Halloran, *Introducing Multimodality* (Routledge, New York, 2016), Chap. 4, p. 232.
[30] V. Prain and R. Tytler, Representing and learning in science, in *Constructing Representations to Learn in Science*, edited by R. Tytler, V. Prain, P. Hubber, and B. Waldrip (SensePublishers, Rotterdam, 2013).
[31] K. S. Tang, The interplay of representations and patterns of classroom discourse in science teaching sequences, Int. J. Sci. Educ. **38**, 2069 (2016).
[32] T. S. Volkwyn, J. Airey, B. Gregorcic, and F. Heijkenskjöld, Transduction and science learning: Multimodality in the physics laboratory, Designs for Learning **11**, 16 (2019).
[33] T. S. Volkwyn, J. Airey, B. Gregorcic, and F. Heijkenskjöld, *Multimodal Transduction in Secondary School Physics, ISEC 2018* (NIE, Singapore (2018).
[34] S. Ainsworth, The educational value of multiple-representations when learning complex scientific concepts, in *Visualization: Theory and Practice in Science Education*, edited by J. K. Gilbert, M. Reiner, and M. Nakhleh (Springer Netherlands, Dordrecht, 2008), pp. 191–208.
[35] M. A. Rau, Conditions for the effectiveness of multiple visual representations in enhancing stem learning, Educ. Psychol. Rev. **29**, 717 (2017).
[36] J. J. Gibson, *The Ecological Approach to Visual Perception* (Houghton Mifflin, Boston, 1979).
[37] H. Kragh, Relativistic collisions: The work of Christian Møller in the early 1930s, Arch. Hist. Exact Sci. **43**, 299 (1992).
[38] F. Marton and S. Booth, *Learning and Awareness*, Educational Psychology Series (L. Erlbaum Associates, Hillsdale, NJ, 1997).
[39] T. Fredlund, C. Linder, J. Airey, and A. Linder, Unpacking physics representations: Towards an appreciation of disciplinary affordance, Phys. Rev. ST Phys. Educ. Res. **10**, 020129 (2014).
[40] U. Eriksson, C. Linder, J. Airey, and A. Redfors, Introducing the anatomy of disciplinary discernment–An example for astronomy, Eur. J. Sci. Math. Educ. **2**, 167 (2014).
[41] J. Airey, U. Eriksson, T. Fredlund, and C. Linder, The concept of disciplinary affordance, in *Proceedings of the 5th International 360 Conference: Encompassing the Multimodality of Knowledge* (Aarhus University, Denmark, 2014), pp. 20.
[42] J. Airey, L. Cedric, and C. Linder, Social semiotics in university physics education: Leveraging critical constellations of disciplinary representations, in *Proceedings of the 11th Conference of the European Science Education Research Association, Helsinki, Finland*, edited by J. Lavonen, K. Juuti, J. Lampiselkä, A. Uitto, and K. Hahl (2015).
[43] D. A. Norman, Affordance, conventions, and design, Interactions **6**, 38 (1999).
[44] J. Bezemer and G. Kress, Writing in multimodal texts, Written Commun. **25**, 166 (2008).
[45] F. Marton, *Necessary Conditions of Learning* (Taylor & Francis, London, 2015).
[46] D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development* (Prentice Hall, Englewood Cliffs, NJ, 1984), p. 20.
[47] K. Svensson, 2D Physics Simulations using Processing IDE for Physics Education Research—Workshop 2018 (2020), https://doi.org/10.5281/zenodo.3607528.
[48] M. Elmgren and A.-S. Henriksson, *Academic Teaching* (Studentlitteratur AB, Lund, 2015).
[49] D. Shiffman, The Coding Train (2019), https://thecodingtrain.com/. Accessed April 1.
[50] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research, Observations (Chicago, Ill.)* (Aldine, Chicago, Illinois, 1967).
[51] I. Walsh, J. A. Holton, L. Bailyn, W. Fernandez, N. Levina, and B. Glaser, What grounded theory is…A critically reflective conversation among scholars, Organ. Res. Methods **18**, 581 (2015).
[52] J. M. Case, D. Marshall, and C. J. Linder, Being a student again: a narrative study of a teacher's experience, Teach. Higher Educ. **15**, 423 (2010).
[53] J. A. Nielsen, Science in discussions: An analysis of the use of science content in socioscientific discussions, Sci. Educ. **96**, 428 (2012).
[54] U. Eriksson, C. Linder, J. Airey, and A. Redfors, Who needs 3d when the universe is flat?, Sci. Educ. **98**, 412 (2014).
[55] G. Kress, Multimodal discourse analysis, in *The Routledge Handbook of Discourse Analysis, Routledge Handbooks in Applied Linguistics*, edited by J. Gee and M. Handford (Taylor & Francis, London, 2013).
[56] D. Hestenes, M. Wells, and G. Swackhamer, Force Concept Inventory, Phys. Teach. **30**, 141 (1992).
[57] A. Cromer, Stable solutions using the euler approximation, Am. J. Phys. **49**, 455 (1981).