Comparing 'Programming-Physics' and 'Algebra-Physics'

Based on *A Comparison of Programming Languages and Algebraic Notation as Expressive Languages for Physics* by Bruce L. Sherin

Paul Tranquilli[1]

[1]Computational Science Laboratory (CSL)
Department of Computer Science
Virginia Tech

CS6604: Computational Thinking
November 5, 2013

Before Virginia Tech

BS Applied Mathematics, University of California, Merced

MS Applied Mathematics, University of California, Merced

At Virginia Tech

Working in the Computational Science Laboratory with Dr. Adrian Sandu

Developing efficient numerical methods to solve partial differential equations.

- Specifically on new time integration methods for large, stiff systems of ordinary differential equations.

"Algebra-Physics" versus "Programming-Physics"

"Algebra-Physics"

- Solutions represented as algebraic equations.
- Classical form taught in universities.

"Programming-Physics"

- Solutions represented as computer programs.
- Sometimes used to illustrate points, but not generally used as primary teaching tool.
  - Feynman used a form of Euler's method to approximate the motion of a planet around the sun.

So, as we said, we began this chapter not knowing how to calculate even the motion of a mass on a spring. Now, armed with the tremendous power of Newton's laws, we can not only calculate such simple motions but also, given only a machine to handle the arithmetic, even the tremendously complex motions of the planets to as high a degree of precision as we wish!

–Richard Feynman in *Lectures*

"The burden of the lecture is just to emphasize the fact that it is impossible to explain honestly the beauties of the laws of nature in a way that people can feel, without their having some deep understanding in mathematics. I am sorry, but this seems to be the case."

–Richard Feynman in *The Character of Physical Law*

If Feynman is right then, in replacing equations with a programming language, we are not simply substituting one tool for another, we are changing the very nature of physics understanding.

–Bruce L. Sherin

I want to elevate programming languages to the status of bona fide representational systems for expressing physical laws and relations, and I want to study the properties of these systems.

–Bruce L. Sherin

Questions the paper seeks to answer

1. How does the understanding associated with "programming-physics" differ from "algebra-physics"?
2. Is "programming-physics" a respectable form of physics?

Questions for us to answer

1. How does the choice of notation or representational forms affect our thought process?
2. Can a computational approach to problem solving lead to insights or understanding not associated with other techniques?
3. Is there a value to developing, or making explicit, a notational framework for Computational Thinking ideas?

Hypothesis

Programs are easier to interpret than equations.

- They can be run mentally.
- Students often revert to rote manipulation of equations without understanding their meaning.

"Programming-physics" priveleges a different intuitive vocabulary

- Certain problems, or ideas, might be easier or more intuitive.
- May be better suited to expressing causal intuitions than algebraic expressions.

Structure of the study

- UC Berkeley students enrolled in Physics 7C.
  - Physics 7C is a third semester introductory physics course for engineering majors.
  - Students have taken both Physics 7A and 7B.
- Subjects divided into two pools each containing five pairs of students
  - The algebra pool worked with their partner to solve a set of problems at a whiteboard.
  - The programming pool worked with a partner at a computer, and were given some training on the programming aspects.

Analysis

Interpretive devices

- Structures used to interpret and understand the meaning of things.
- For example the idea that programs can be interpreted by mentally 'running' them.

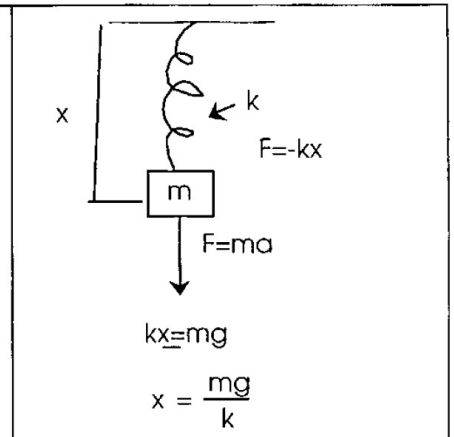
Symbolic forms

- Structures that students learn to recognize in symbolic expressions.
- Associated with elements in a conceptual vocabulary.

Mass on a spring, an "algebra-physics" problem

Problem 4

A mass hangs from a spring attached to the ceiling. How does the equilibrium position of the mass depend upon the spring constant, k, and the mass, m?



Jim: Okay, and this makes sort of sense because you figure that, as you have a more massive block hanging from the spring, then your position x is gonna increase, which is what this is showing. [Points to m then k] And that if you have a stiffer spring, then your position x is gonna decrease. [Uses fingers to indicate the gap between the mass and the ceiling] That's why it's in the denominator. So, the answer makes sense.

An example interpretive device

Changing Parameters – A narrative interpretation in which one quantity is varied while others are held fixed.

- Jim validates his answer through construction of a narrative that confirms his physical intuition.
- This narrative is constructed by varying the mass or spring stifness while holding other parameters constant.

List of interpretive devices

TABLE III

Interpretive devices by class

Narrative	Static
CHANGING PARAMETERS	SPECIFIC MOMENT
PHYSICAL CHANGE	GENERIC MOMENT
CHANGING SITUATION	STEADY STATE
	STATIC FORCES
Special case	CONSERVATION
RESTRICTED VALUE	ACCOUNTING
SPECIFIC VALUE	
LIMITING CASE	
RELATIVE VALUES	

An example symbolic form

prop+ : $\left[\frac{\ldots x \ldots}{\ldots}\right]$ – A symbolic form in which one quantity varies directly with a second quantity.

- Jim notices that the mass variable, m, is in the numerator of his expression.
- In this way he recognizes that the position varies directly with the mass.

prop- : $\left[\frac{\ldots}{\ldots x \ldots}\right]$ – A symbolic form in which one quantity varies inversely with a second quantity.

- Similar to before, Jim notices that the stiffness variable, k, is in the denominator of his expression.
- In this way he recognizes that the position varies inversely with the spring stiffness.

List of symbolic forms

TABLE IV

Symbolic forms by cluster

Competing Terms cluster		**Terms are Amounts cluster**	
COMPETING TERMS	$\square \pm \square \pm \square \ldots$	PARTS-OF-A-WHOLE	$[\square + \square + \square \ldots]$
OPPOSITION	$\square - \square$	BASE ± CHANGE	$[\square \pm \Delta]$
BALANCING	$\square = \square$	WHOLE − PART	$[\square - \square]$
CANCELING	$0 = \square - \square$	SAME AMOUNT	$\square = \square$
Dependence cluster		**Coefficient cluster**	
DEPENDENCE	$[\ldots x \ldots]$	COEFFICIENT	$[x\square]$
NO DEPENDENCE	$[\ldots]$	SCALING	$[n\square]$
SOLE DEPENDENCE	$[\ldots x \ldots]$	**Other**	
Multiplication cluster		IDENTITY	$x = \ldots$
INTENSIVE·EXTENSIVE	$x \times y$	DYING AWAY	$[e^{-x\ldots}]$
EXTENSIVE·EXTENSIVE	$x \times y$		
Proportionality cluster			
PROP+	$\left[\frac{\ldots x \ldots}{\ldots}\right]$	RATIO	$\left[\frac{x}{y}\right]$
PROP−	$\left[\frac{\ldots}{\ldots x \ldots}\right]$	CANCELING(B)	$\left[\frac{\ldots x \ldots}{\ldots x \ldots}\right]$

Dropped ball, a "programming-physics" problem

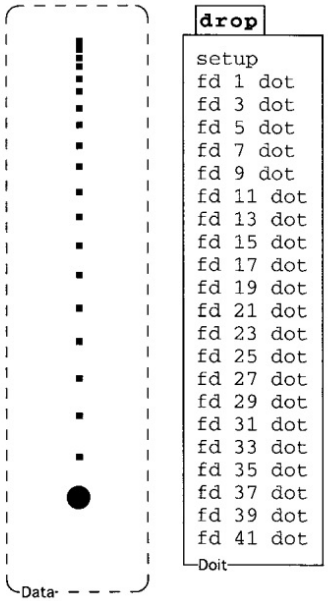


Figure 3. Anne and Clem's simulation of a dropped ball without variables.

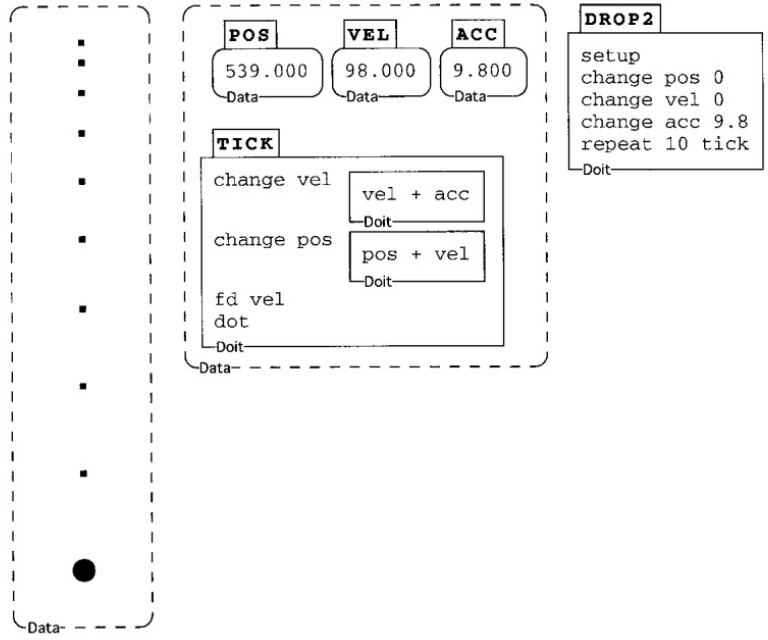


Figure 4. Tim and Steve's simulation of a dropped ball using the Tick Model.

Dropped ball– The idea here is to make a realistic simulation of the motion of a ball that is dropped from someone's hand. Using the tick model, redo your simulation of a dropped ball.

Adding air resistance I

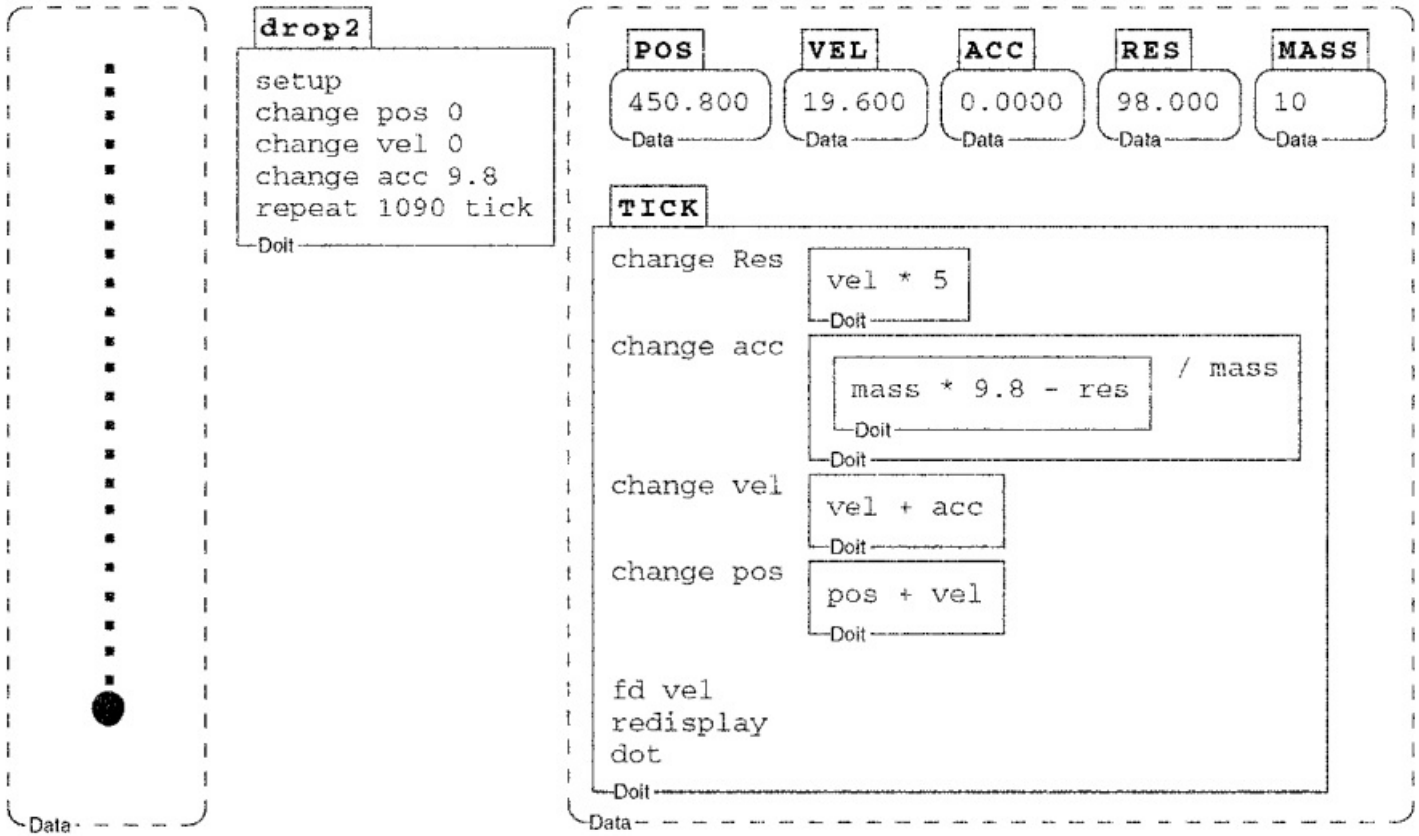


Figure 7. Tim and Steve's final air resistance program.

Adding air resistance II

Tim: No we have to (then account) in the acceleration. So we have to change the acceleration. Do we change the acceleration before we change the velocity? [Pointing into **tick** *box here.] Or do we change acceleration after we change velocity? Which also changes the resistance. Oh, I'm getting confused. What's the order in which you change things?*

. . .

Tim: Well you have to change resistance because you change the velocity, because the velocity – the new velocity is dependent on the acceleration which is dependent upon the resistance. So we should change resistance, then acceleration, then velocity. Right? Does that make sense, or am I just talking nonsense.

An example symbolic form and interpretive device

Serial Dependence – A sequence of linked dependence relations. **d** depends on **c**, **c** depends on **b**, **b** depends on **a**.

Tracing – Constructing a narrative through mentally "running" a program.

How do they compare?

- The narrative devices are very similar between programming- and algebra-physics.
- Tracing is the only interpretive device found unique to programming-physics.
- Programming-physics leads to more physical narratives.
- Programming-physics lends itself to particular examples, and the easier identification of special circumstances.
- Programming-physics may give better understanding of causal relationships.
- Algebra-physics allows for more general interpretations and circumstances.
- Programming-physics has the advantage of more context, when interpreting already completed programs.

- The construction of a computational solution to the air resistance problem led to the confrontation of ideas which may not have arisen from a simple algebraic solution.
  - Can we exploit this benefit to make Computational Thinking attractive to instructors in other disciplines?
  - Does this benefit arise only from programming a solution, or from the construction of a computational approach?
- Is there a benefit to the identification or development of symoblic-forms for Computational Thinking?