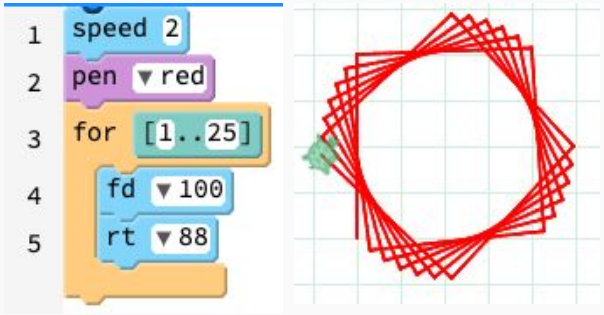
Using Computational Thinking in the Physics Classroom with STEMcoding

James Newland - CSAAPT Fall 2023
University of Houston College of Education & Bellaire HS

Computational Thinking as Constructionism

Turtle LOGO from New Mindstorms VHS MIT 1986



Turtle Blocks Coffeescript (https://pencilcode.net/edit/first)

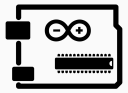
Seymour Papert - Constructivism with a physical artifact is ***constructionism***

> "The goal is to use computational thinking to forge ideas that are at least as "explicative" as the Euclid- like constructions (and hopefully more so) but more accessible and more powerful (Papert, 1996)."

Jeanette Wing on CT (2006) - "It represents a universally applicable attitude, and skill set everyone, not just computer scientists, would be eager to learn and use."

For me, computational thinking is rooted in social constructivist theory and constructionism.

Computational Thinking (CT) In Science Classes



"*Introducing computation ... into high school or college non-major physics courses ... **should not** be done **solely** to **provide** an early start to the kind of **skills students** would **need as a professional physicist or as a software engineer**.*"
(Orban & Teeling-Smith, 2018, p. 248)

Computing is for everyone.

Data Practices	Collecting Manipulating	Analyzing Visualizing
Modeling & Simulation Practices	Find & Test Solutions	Designing Constructing
Computational Problem Solving Practices	Programming Assess Solutions	Abstractions Debugging
Systems Thinking Practices	Relationships in Systems	Thinking in Levels

Components of CT from Weintrop, et al., (2016).

The Euler-Cromer Modeling Paradigm

```
1  x = 375;
2  y = 250;
3
4  vx = 10;
5  vy = 0;
6
7  dt = 0.1;
8
9  function draw(){
10
11     // Update location
12     x = x + vx*dt;
13
14     // Draw axes and other stuff
15     display();
16
17     drawBlob(x,y,vx,vy);
18     // Add more graphics here before the end of draw()
19
20 } // end draw()   DO NOT ADD ANY CODE AFTER THIS LINE!!!
```

With **Euler-Cromer modeling**, iteration and variable accumulation allows for stepwise state changes. Usually the time variable is iterated and there is a simple visualization component. This is a variation of numerical integration. Here we are using p5js so the draw function handles iteration automatically.

How much CS content knowledge is required?

Leveraging CS Pedagogy in Science Coding

Step 2b. Change Fnety to include the weight of the ship

To add gravity to the code, we **CAN'T** just set ay = g. This would make the ship accelerate towards the ground like we want (feel free to try it out!), but this approach will **NOT** let us fire the thrusters to slow our descent. We are going to need to do something a bit more clever than that.

The clever (and right) way of adding gravity to the code is to remember these three facts:

1. gravity is a force

2. The force of gravity is the weight

3. The net force is the sum of all the forces

This means that we need to replace this code `Fnety = Fy;` with something else:

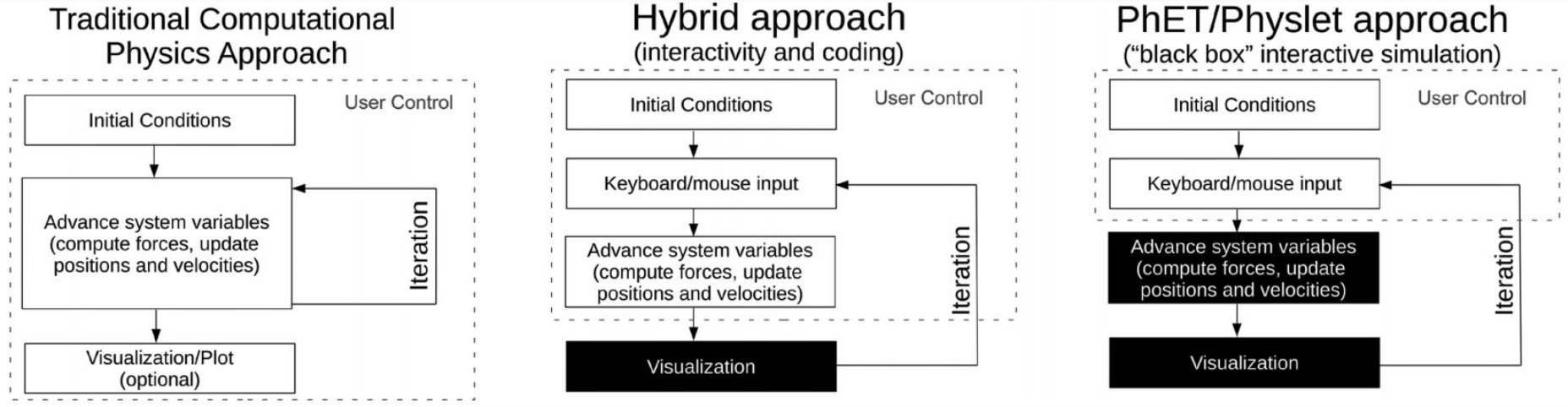
```
W = ?????; // what goes here?
Fnety = Fy + W;
```

```
1  x = 375;
2  y = 250;
3
4  vx = 10;
5  vy = 0;
6
7  dt = 0.1;
8
9  function draw(){
10
11     // Update location
12     x = x + vx*dt;
13
14     // Draw axes and other stuff
15     display();
16
17     drawBlob(x,y,vx,vy);
18     // Add more graphics here before the end of draw()
19
20 } // end draw()   DO NOT ADD ANY CODE AFTER THIS LINE!!!
21
```

According to Morrison, et al., (2014) cognitive load theory suggests that learning is impaired when the processing requirement exceeds the capacity of working memory.

Morrison & Margulieux (2015) show that subgoal labeling and worked examples can scaffold learning for new coders struggling with cognitive load.

CT as Modeling in Physics - Simulation with Euler-Cromer



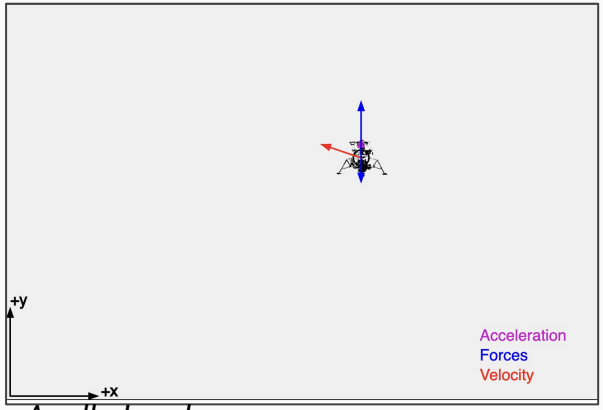
Illustrations of different approaches to computationally enriched physics content (Orban & Teeling-Smith, 2018)

A lot of CS pedagogy needed. Same cognitive load as a CS course. Science knowledge should be secure also. Meant for college physics/astronomy majors.

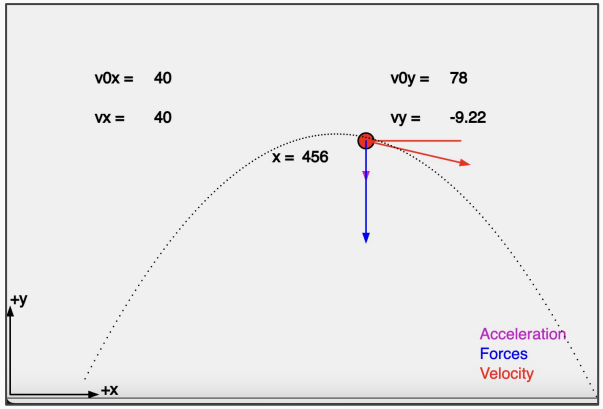
Much less cognitive load and minimal CS pedagogy needed. Students can create science knowledge and some CS knowledge. Meant for intro science courses 8-16.

Data analysis and visualization are emphasized. Science knowledge can be created with nearly no CS pedagogy needed. Works for K-16.

STEMcoding - Modeling physics in simple video games



Apollo Lander



Bird Launcher

```
x = 0;
y = 0;

vx = 0;
vy = 0;

deltaVx = 0;
deltaVy = 0;

theta = 0;

Fthrust = 30.0;
mass = 3.0;
dt = 0.1;

function draw(){
  // Update velocities
  vx += deltaVx;

  // Update location
  x += vx*dt;

  // velocity is unchanged if there are no forces
  deltaVx = 0;

  // Turn or thrust the ship depending on what key is pressed
  if (keyIsDown(LEFT_ARROW)) {
      theta += 0.0;
  }
  if (keyIsDown(RIGHT_ARROW)) {
     theta += -0.0;
  }
  if (keyIsDown(UP_ARROW)) {
     // Rockets on!
     accelx = Fthrust*cos(theta)/mass;
     deltaVx = accelx*dt;
   }
```

left right arrows to turn, tap up arrow to thrust, press H to hide the arrows, press U to un-hide

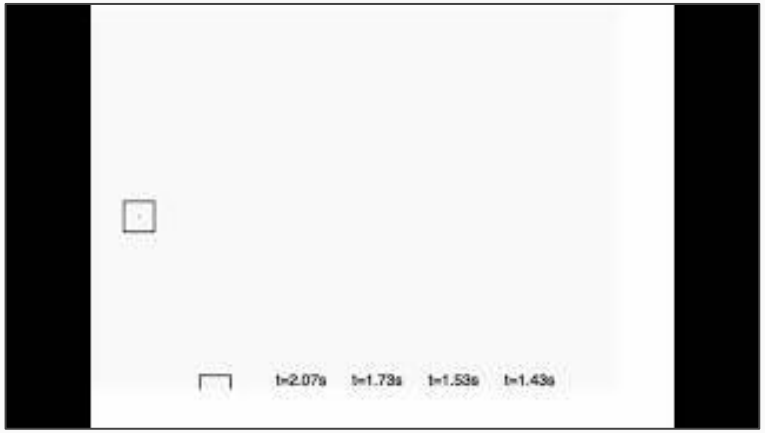
+y
+x
Velocity
Force
Acceleration

Planetoids

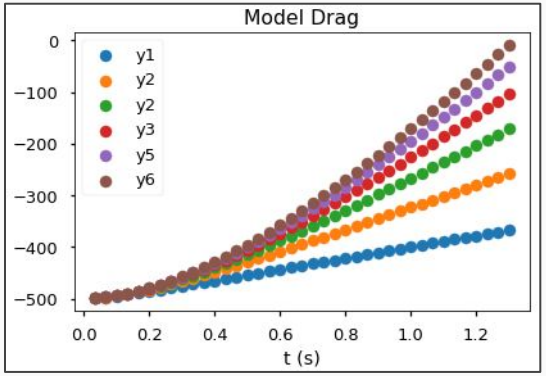
STEMcoding - web-based physics curriculum and LMS covering intro physics using p5js, designed for teaching coding & simple event handling.

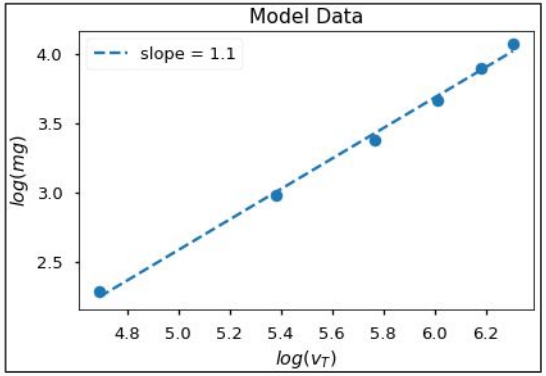
STEMcoding Airdrag Example

Students create air drag models which create datasets to be analyzed.




```
28  // Return the acceleration due to drag.
29  // b is the drag coeffecient for the object
30  // vel is the current velocity of the object
31  // m is the mass of the object
32  function a_drag(v, b, m){
33      // Put the acceleration due to air drag here.
34      // Don't assign a direction here. (no negative)
35      a = 0; // fix this line
36
37      // Do not exceed acceleration due to gravity!
38      if (a > abs(g)) a = abs(g);
39
40      // Return the acceleration due to drag
41      return a;
42  }
```





t	y1	y2	y3
0.1	5	5.1	6
0.2	10.1	13	16
0.3	17	23	27.5
0.4	24	34.5	42.5
0.5	31	45	55
0.6	39	58	72
0.7	47	72	87
0.8	56.5	85	103

References

Cromer, A. (1981). Stable solutions using the Euler approximation. *American Journal of Physics*, *49*(5), 455–459. https://doi.org/10.1119/1.12478

Morrison, B. B., Dorn, B., & Guzdial, M. (2014). Measuring cognitive load in introductory CS. *Proceedings of the Tenth Annual Conference on International Computing Education Research - ICER '14*, 131–138. https://doi.org/10.1145/2632320.2632348

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, *39*(1), 21–29. https://doi.org/10.1145/2787622.2787733

Odden, T. O. B., & Burk, J. (2020). Computational Essays in the Physics Classroom. *The Physics Teacher*, *58*(4), 252–255. https://doi.org/10.1119/1.5145471

Orban, C. M., & Teeling-Smith, R. M. (2020). Computational Thinking in Introductory Physics. *The Physics Teacher*, *58*(4), 247–251. https://doi.org/10.1119/1.5145470

Orban, C., Teeling-Smith, R. M., Smith, J. R. H., & Porter, C. D. (2018). A hybrid approach for using programming exercises in introductory physics. *ArXiv*, *831*. https://doi.org/10.1119/1.5058449

Skudder, B., & Luxton-Reilly, A. (2014). Worked examples in computer science. *Conferences in Research and Practice in Information Technology Series*, *148*, 59–64.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wolfram, S. (2017). *What Is a Computational Essay?* https://writings.stephenwolfram.com/2017/11/what-is-a-computational-essay/