The Power of Student Coding in Physics

...

Joshua Gates / @DeltaGPhys / joshgates@tatnall.org / Newton's Minions
The Tatnall School, Wilmington DE

2016 AAPT National Meeting - Monday 18 July, 3-3:30 pm CC-306

Goals

- ➔ Motivation to integrate computation into (introductory/9th grade/AP/intermediate level, etc.) physics courses
- ➔ What can teachers do with VPython? *What can students do with it?*
- ➔ Supporting students who have never coded, to a degree appropriate to their level
- ➔ Can students learn coding via project-based learning?

Computation as a Tool for Qualitatively New Classroom Experiences

- ➔ Integral to most physics work encountered *after* graduating school
- ➔ Computational thinking develops:
  - ◆ Logical (linear) reasoning
  - ◆ Decomposition of larger tasks into sub-problems and incremental development
  - ◆ Abstraction
  - ◆ Creation and application of algorithms
  - ◆ "Soft skills" - tinkering, perseverance, and collaboration
- ➔ Computational thinking is a core scientific practice, as designated by the National Research Council in the Next Generation Science Standards
- ➔ It enables 'experimentation' in situations both physically inconvenient and mathematically intractable
- ➔ Computation is the "fifth representation" (verbal, graphical, diagrammatic, algebraic)
- ➔ *It has been nearly totally absent from the physics curriculum, especially below the intermediate undergraduate level*

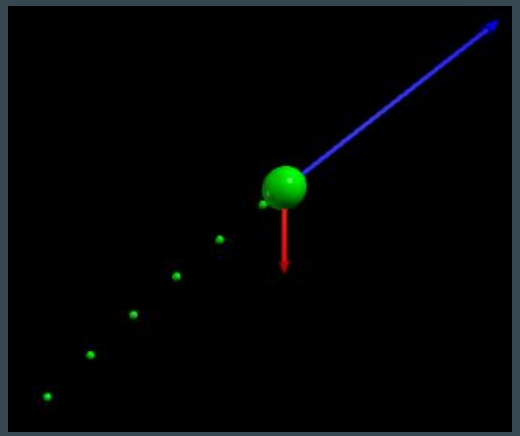
Using VPython in Introductory Courses

- ➔ Developed by David Scherer, Ruth Chabay, and Bruce Sherwood, implemented at NCSU for introductory calculus-based physics
- ➔ Adds to Python seamless and easy 3D rendering of objects and graphing; includes perspective, light-sourcing, rotation, and zooming natively
- ➔ Friendly language for new programmers
- ➔ Available for local installation and in an online form (Glowscript); Glowscript also runs in mobile device environments (phones and tablets)
- ➔ In addition to the physics-friendly nature of the Visual module, Python is the most popular introductory language in highly-ranked colleges' beginning CS courses, giving a good foundation for their coding future

Traditional Coding Instruction Strategies

Introductory level (9th grade physics, 11th/12th grade physics)

- ➔ Skills: mathematical operators, vectors, loops, 3D objects, variable manipulation
- ➔ Physical concepts: constant-velocity and constant-acceleration motion, balanced and unbalanced force situations



- ➔ Example assignment progression
  - ◆ Vector practice/relative positions (M/I 1.P.121)
  - ◆ Constant-velocity cart (also try PhysUtil) (M/I 1.P.122)
  - ◆ Constant-acceleration cart
  - ◆ Motion under unbalanced forces
  - ◆ Projectile motion (modify program above so that velocity and acceleration are no longer parallel)

The Big Physical Idea: Updates and Iteration

Given knowledge of the system's initial position and velocity, as well as the net force on the system, the algorithm for predicting motion can be described as a set of rules applied locally in space and time:

1. At a given instant in time t, compute the net force acting on the system
2. For a short time Δt later, compute the new velocity of the system using Newton's 2nd law ("velocity update formula")
3. At the same new time ($t + \Delta t$), compute the new position of the object using this updated velocity ("position update formula")
4. Repeat Steps (1)-(3) starting at the updated time $t + \Delta t$

You can structure an entire introductory mechanics curriculum around just three updates!

Caballero, et al. "Integrating Numerical Computation Into the Modeling Instruction Curriculum"

Helpful Activities for First-time Coders

Code Modification

- ➔ Provide code for students, explaining the sections and modeling good commenting
- ➔ Students modify the code to alter the behavior - changing initial conditions or adding new functionality to the existing shell

"The overall goal for designing a piece of software (a computer program) is to solve a given problem. The design process should be based on a thorough understanding of the problem, and it should include the goal of creating an understandable, adaptable solution. Novice programmers should begin learning about the design process by studying well-designed programs and modifying them. Later in the course, students should be able to work from a specification to develop a design for a program or part of a program." AP CS Guide

Pair Programming

- ➔ Students work in partners, with one typing and the other reviewing each line of code; this takes a bit longer, but generally produces superior work
- ➔ Discussion is an integral part, and students will need to be trained to do this effectively
- ➔ Done effectively, it provides students experience in analysis, communication, planning, and collaboration, while strengthening their understanding of the CS and physics concepts, due to their need to explain their approaches to each other and to critique each other's work
- ➔ Setting a short time limit (5 minutes or so) is one of several strategies to help make this work

“Hands-on” Programming Activities

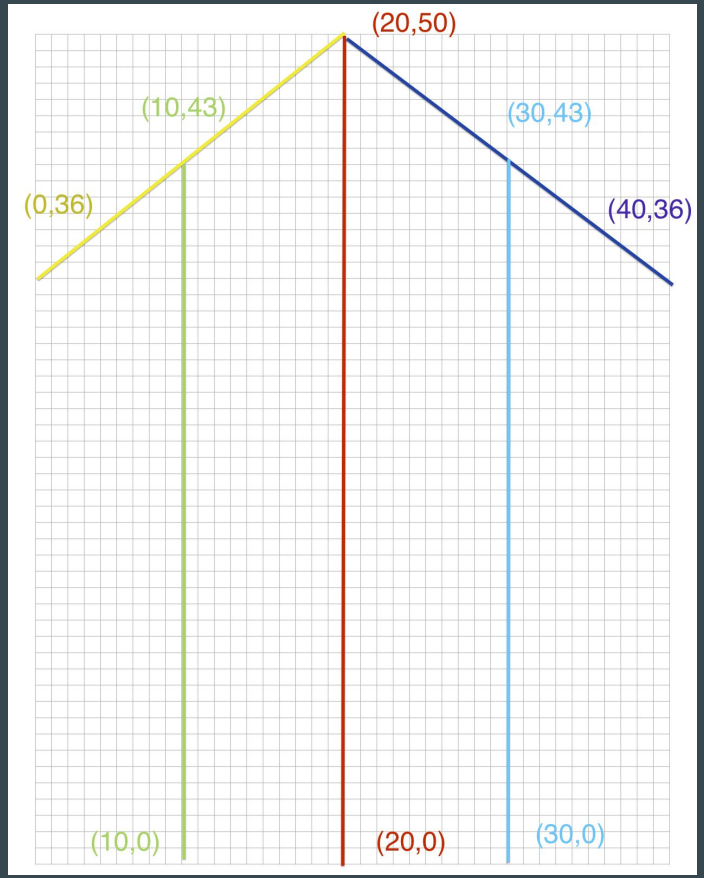
These activities help teach computer science principles without using the computer. They can be helpful for anyone, but are especially useful for 9th grade physics students.

LEGO Instructions

- ➔ Have three members of a four-member group build a simple structure, writing instructions for the fourth person to build it (without having seen it)
- ➔ Teaches value of specificity of instructions (somewhat less helpful with inerrancy of compilers)
- ➔ Develops the idea for the need of a strict syntax, which isn’t apparent to many young students

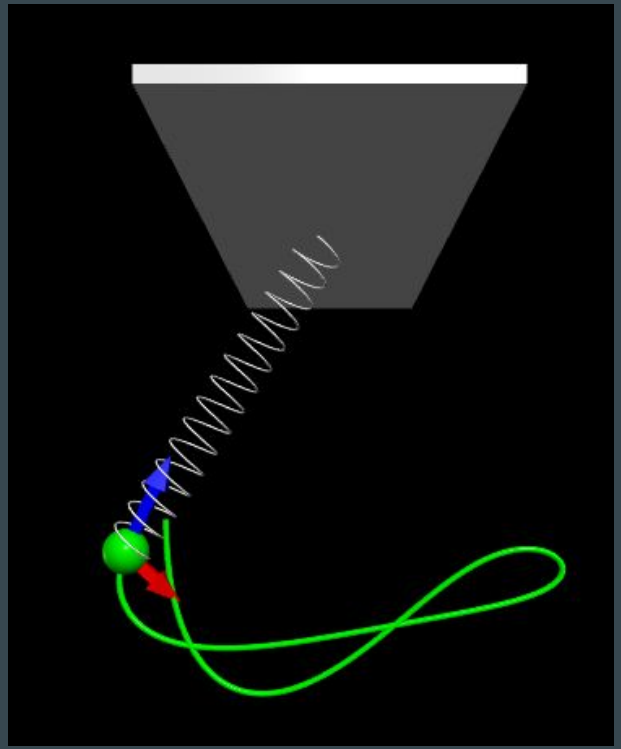
Folding Functions

- Teaches the principle of functions, needing inputs and performing a task based upon those inputs
- Starting with a piece of graph paper, students create a set of function calls to a “folding function” in order to create a paper airplane
- Function takes two points (defining the fold axis) and a fold direction (graph grid together or apart) to create the airplane

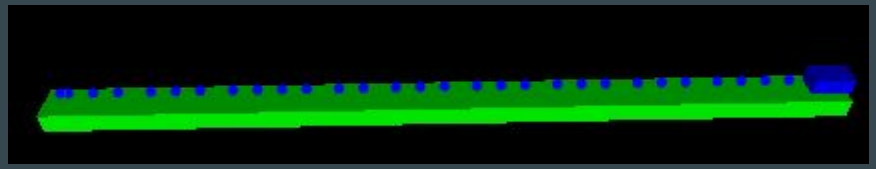


Benefits of VPython for the Introductory Course

- This progression shows the unity of constant-*v* and constant-*a* motion, of 1D- and 2D-motion, and of situations with balanced and unbalanced forces
- Allows for a more robust understanding of vectors and the laws of physics
- Can *easily* be extended for non-constant forces, such as springs and drag
- Relativistic models can also easily be included; this is a feature of the Matter and Interactions course



VPython at the AP Level



- ➔ At the AP level (or for calculus-based undergraduate courses), students can generally self-start learning to code with VPython
- ➔ A set of introductory videos at vpython.org can take them from zero to a great starting point from which to explore further:

1. 3D Objects (YouTube)
2. Variable Assignment (YouTube)
3. Practice exercise (M/I 3.e. 1.P.121)
4. Beginning Loops (YouTube)
5. Loops and Animation (YouTube)
6. Practice exercise (M/I 3.e. 1.P.122)

Additional videos can be helpful later as well:

- Scale Factors (YouTube)
- Debugging Syntax Errors (YouTube)
- Lists, Parts 1 (YouTube) and 2 (YouTube)

Additional Programming Tasks for the AP Level

Once the momentum principle (forces and motion) has been completed, the programming can become a bit more abstract, but it is still valuable

Example Tasks

- ➔ Gravitation - modifying a program created during the first term, students graph kinetic, potential, and total energy of a satellite system to explore bound orbits
- ➔ Tipping Box - students use VPython to create a graph of the potential energy of a box/Earth system as it is tipped, identifying equilibrium points and types
- ➔ Physical pendulum - students apply their understanding of equilibrium states and energy to physical pendula

VPython and Novel Questions

...as a tool for teachers

- ➔ Aerobraking - an exploration of the maneuver used by the Mars Reconnaissance Orbiter, using the Mars Atmosphere Model: Glowscript Animation
- ➔ Dot Physics - Rhett Allain's blog on WIRED, where he often uses VPython to solve problems from movies or video games. His Angry Birds series of problems has generated a book
- ➔ Counterfactual animations - you can use VPython to create animations of situations with correct or incorrect physics, challenging students to puzzle out which physics is accurate

… as a tool for students: independent projects

The assignment for the project was simply to use VPython to answer a question using physics. The diversity of situations and topics was considerable

- ➔ A simulation of the effects of the sun disappearing for a period of time. The animation begins at the moment the sun vanishes, then a click brings the sun back; eccentricities are calculated for each planet.
- ➔ A poster outlining the results of two students' attempts to use publicly-available data to recreate the Apollo 11 launch, simulating each stage and comparing the results to the actual results
- ➔ A program simulating a falling tank, firing its main gun in an attempt to change its landing position (a recreation of a scene from the film *The A-Team*)
- ➔ A poster presenting the results of both an experimental and simulated verification of the theoretical solution of a problem from this year's AP Physics C exam

Difficult Visualization and VPython

In very difficult-to-visualize subject areas, VPython can help greatly help students understand concepts better

- ➔ Students can write programs to perform the integration to determine moment of inertia: line, slab, ring, disc (for ring, disc, use annulus.py by Matt Greenwolfe)

Electromagnetism demonstrations (first four programs all M/I example programs):

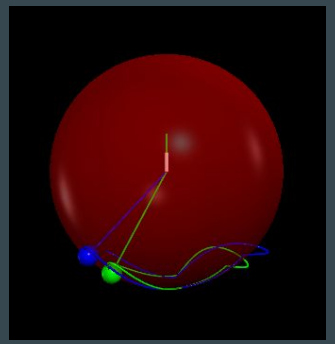
- ➔ Summing electric field components along the axis of a ring
- ➔ A program to visualize fields, Gauss's Law, and Ampere's Law
- ➔ Illustration of finding the magnetic field around a wire
- ➔ Visualization of magnetic field of a moving magnet
- ➔ Students can use point masses to analyze E field of finite line of charge (click/drag to see E field anywhere)

Beyond the Introductory Level

Hamiltonian and Lagrangian Systems

Obtaining the equations of motion can be relatively straightforward, but what then? Historically, not much. Now, students can iterate the systems and *see* the behavior

- ➔ Motion in absolute-value potential well - compares iterative result for period with result gained from using action-angle variables
- ➔ Sliding mass and hanging mass on table
- ➔ Periodically-kicked pendulum



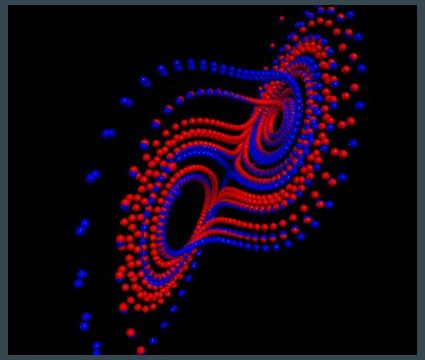
Materials Science

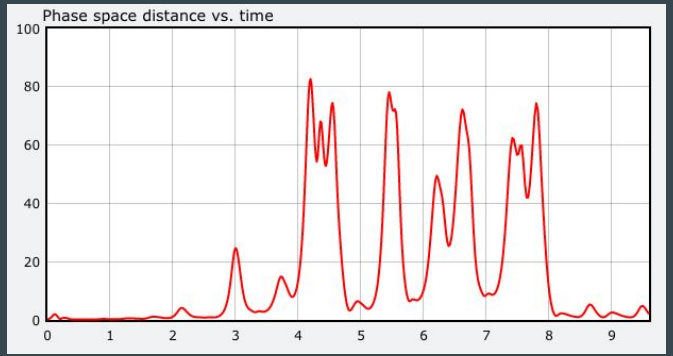
- ➔ Exploring lattice structures (native zoom and rotate are very helpful here)

Chaos and Dynamical Systems

While phase plane visualization is readily available through MATLAB (and even through a standalone Java version of pplane), constructing and viewing animations of evolving systems brings a new level of awareness to students

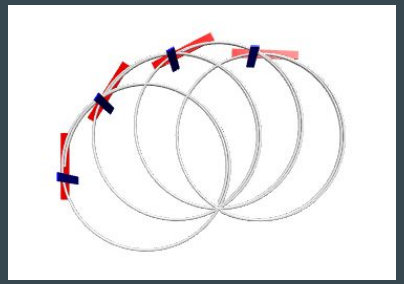
➔ Lorentz system - students can animate two slightly different initial conditions and see their divergence, as well as the two chaotic attractors of the system

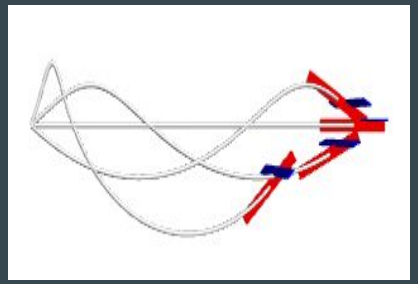


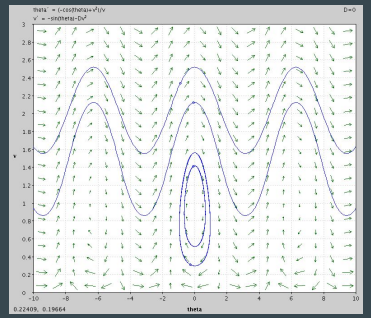


➔ Model airplanes - the system can easily be solved and modeled statically, but simple iteration via VPython gives context to the different possible motions of the model plane

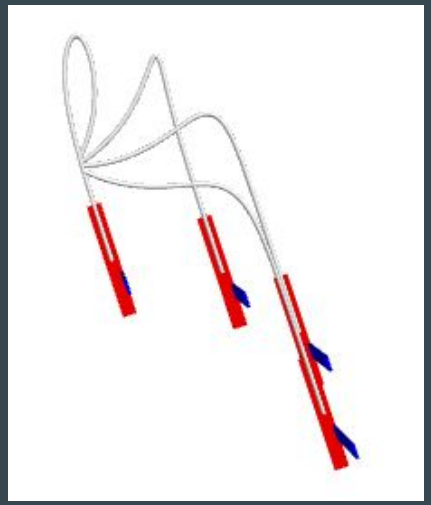
Without drag: loops and undulation

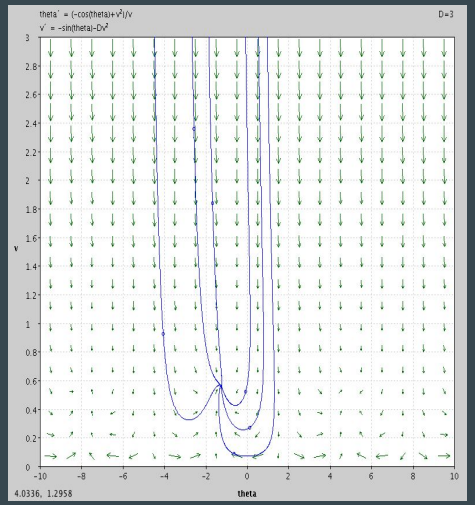






With drag: loss of energy and stalling





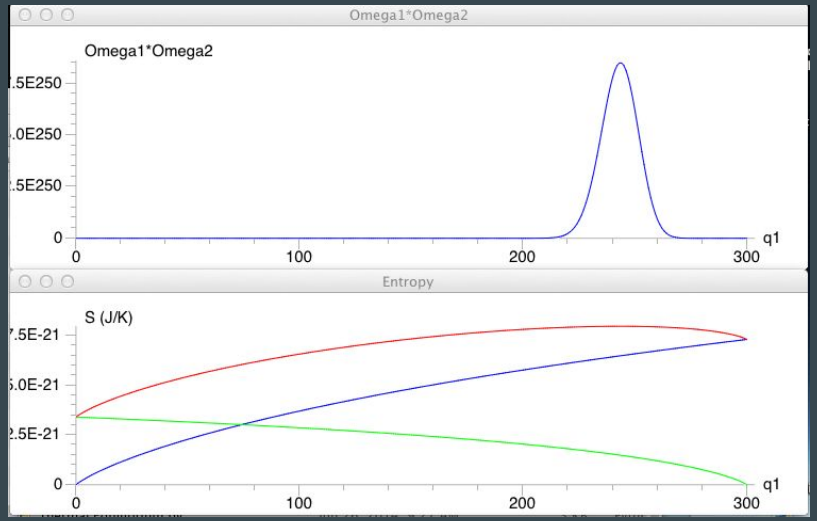
Statistical Mechanics

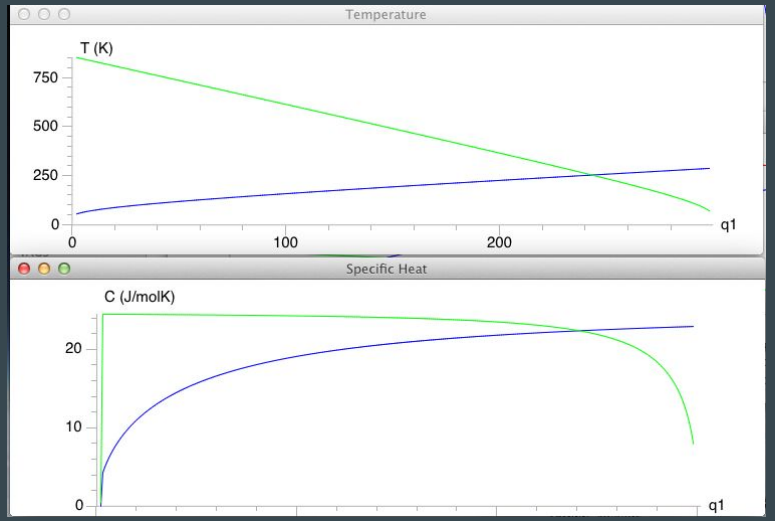
Basic thermodynamic concepts can be explored through simulation with VPython; note that Matter and Interactions integrates VPython and mechanical concepts in this fashion. Students code the problem described below to illustrate their understanding

Program goals:

- ➔ Simulate the distribution of 300 quanta of energy in two aluminum blocks, consisting of 170 and 40 atoms (each atom modeled as three QHO)
- ➔ Calculate and graph the total number of microstates of the two-block system as a function of the number of quanta in block 1
- ➔ Calculate and graph the entropy of each block and of the system as a function of the number of quanta in block 1

- ➔ Graph the temperatures (derivative of energy w.r.t. entropy) of each block as a function of the number of quanta in block 1
- ➔ Graph the specific heat (derivative of temperature w.r.t. entropy) of each block as a function of the number of quanta in block 1





A Different Paradigm for Learning Coding

- ➔ This was explored in the context of a term-long Electrical Engineering elective
- ➔ The course was fully implemented project-based learning (PBL)
- ➔ Students had previously taken an electronics elective, but had no prior coding instruction in the Arduino environment; many had no coding experience at all

The Driving Question

Robots overcome obstacles in many different ways; some are similar to the methods that humans would use and some are markedly different. The goal here is to design an "obstacle" and a robot to navigate the obstacle. Teams will build both the "arena" in which the robot operates and its goal location, and their robot must overcome the obstacle to reach the goal.

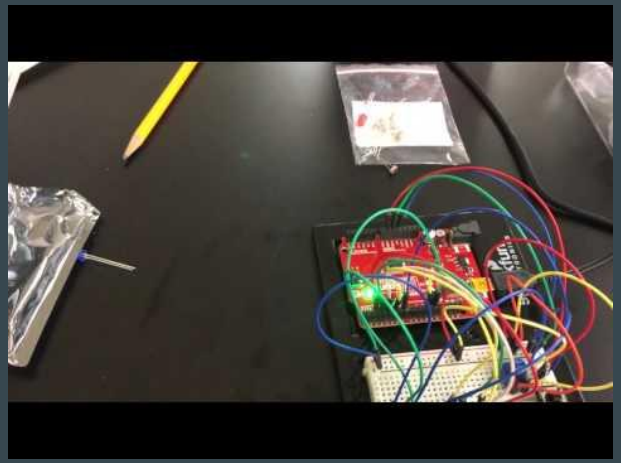
Course Structure

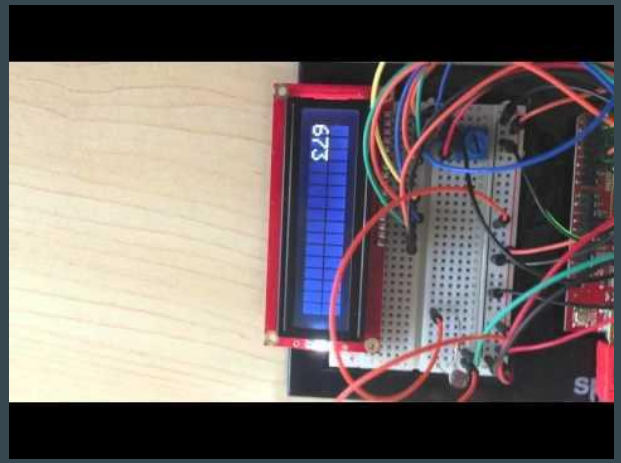
Students use the SparkFun Inventor's Kit and Parallax Robotics Shield for starting hardware, with the SIK guide and the Make: Getting Started with Arduino book for ideas and examples

Deliverables

- "Obstacle" description, along with ideas (plural!) about how the robot might overcome it
- Specifications sheet: details robot's inputs (information from sensors), outputs (expected behaviors, actions, etc.), and expected exceptions (problems that can occur)
- Contribution to the WCGW? (What Could Go Wrong?) meeting: brainstorming unexpected exceptions for each other's projects

- Three Arduino projects from the texts that could pertain to your problem (individual evaluation; three projects different from your partner's three). For each, summarize how you think it might pertain to your project and show how you modified the project/sketch to change how it functions in some way. Present these as a YouTube video, with commented code (showing especially the modifications) linked to the description.

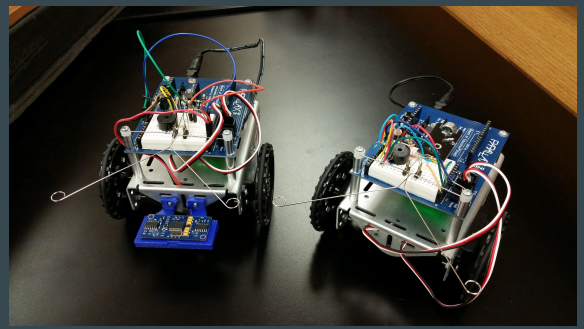


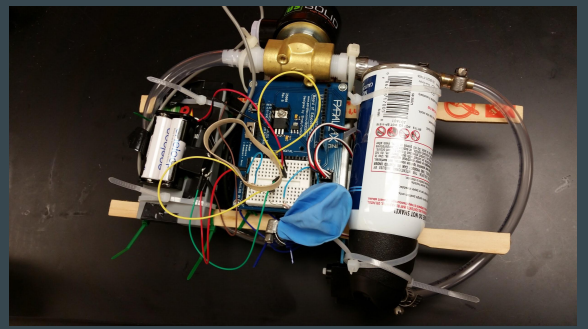


- ➔ Program flow chart (detail the sequence of sensor readings, calculations, and outputs that will take place in the loop, as well as the preliminary variables that need to be set)
- ➔ Schematic: Arduino and all associated electronics (sensors, motors, LEDs, etc.)
- ➔ The fully-functional robot and landscape
- ➔ Reflection on the process and the big question of how robots overcome obstacles and how that is similar to or different from how humans do

Results

- ➔ Successful (all at least to a degree) robots
- ➔ Skills acquired organically during the projects:
  - 3D printing design for Ultimaker 2, using TinkerCAD and/or SketchUp
  - CNC design for Carvey, using Easel
  - Understanding of hardware input and output, looping structure, basic data types in Arduino environment
  - Coding understanding at the end is less topically uniform than with traditional instruction
  - Experience integrating unfamiliar components into their designs by using datasheets and research





Computation in Physics Education: Summary

- ➔ Coding with VPython is easily learnable at a variety of levels and reinforces, rather than undermines, physics and calculus concepts, as well as CS concepts
- ➔ VPython is powerful and flexible enough to be employed from introductory through upper division courses, and can be a unifying tool in a department
- ➔ Learning to code with the Arduino environment's C/C++ variant gives students experience with another common language and the ability to build simple or sophisticated autonomous systems
- ➔ Both more traditional coding instruction and project-based learning can be used successfully to support student learning
- ➔ Giving students latitude in choosing the targets of their efforts increases their investment and their independence in problem definition and framing

Further Resources

- GA Tech VPython Research Group
- Gyroscopic Motion: Show Me the Forces!
  Kaplan, Harvey and Hirsch, Andrew, The Physics Teacher, 52, 30-33 (2014)
- VPython: 3D Interactive Scientific Graphics for Students
  Scherer, David and Dubois, Paul and Sherwood, Bruce, Computing in Science & Engineering, 2, 56-62 (2000)
- Fostering computational thinking in introductory mechanics
  Caballero, Marcos D. and Kohlmyer, Matthew A. and Schatz, Michael F., AIP Conference Proceedings, 1413, 15-18 (2012)
- Understanding student computational thinking with computational modeling
  Aiken, John M. and Caballero, Marcos D. and Douglas, Scott S. and Burk, John B. and Scanlon, Erin M. and Thoms, Brian D. and Schatz, Michael F., AIP Conference Proceedings, 1513, 46-49 (2013)
- Frank Noschese, Action/Reaction
- Andy Rundquist, SuperFly Physics
- John Burk, Quantum Progress
- Aaron Titus, Physics LOGOS
- Rob Salgado, VPython Applications
- Matter and Interactions blog