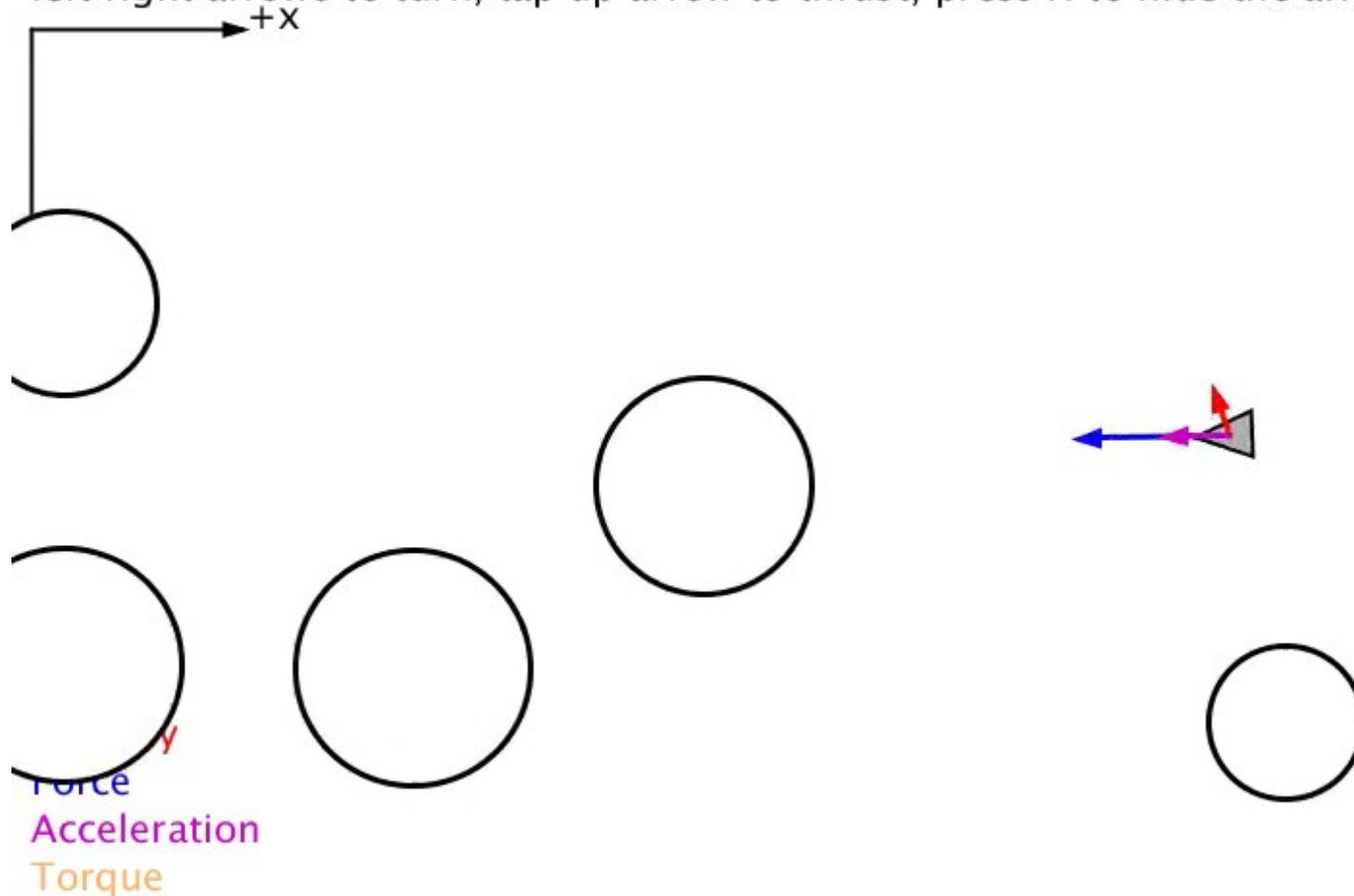


Enriching Introductory Physics Courses with Programming Exercises using Classic Video Games

Prof. Chris Orban (OSU)

Gregory Ngirmang (OSU), and Mark Schillaci (OSU)

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



Interactive Animations

- Physics e-textbooks now routinely include videos and interactive animations (ex. java applets)
- **Interactive animations can illustrate the ideal behavior of a physics problem in a way that live demonstrations cannot**



[PHET Colorado: Energy Skate Park]

Interactive Animations vs. Programming Exercises

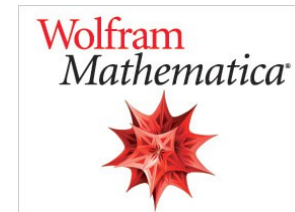
- Interactive animations are a must for high school physics and algebra-based physics in college
- But what can be done for the next course level – introductory calculus-based physics?
- **Some institutions incorporate programming in introductory calculus-based physics** (VPython, java...)
- **However, most calculus-based physics courses at OSU and elsewhere do *not* involve programming exercises**
- OSU's Marion campus is an exception to this, as will be discussed

Which programming framework to use?

- The preferred programming language for physics pedagogy ***used*** to be java
- For a variety of reasons, java is no longer the dominant language for physics instruction
- There is no clear successor


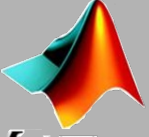
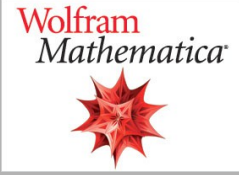




VPython



processing



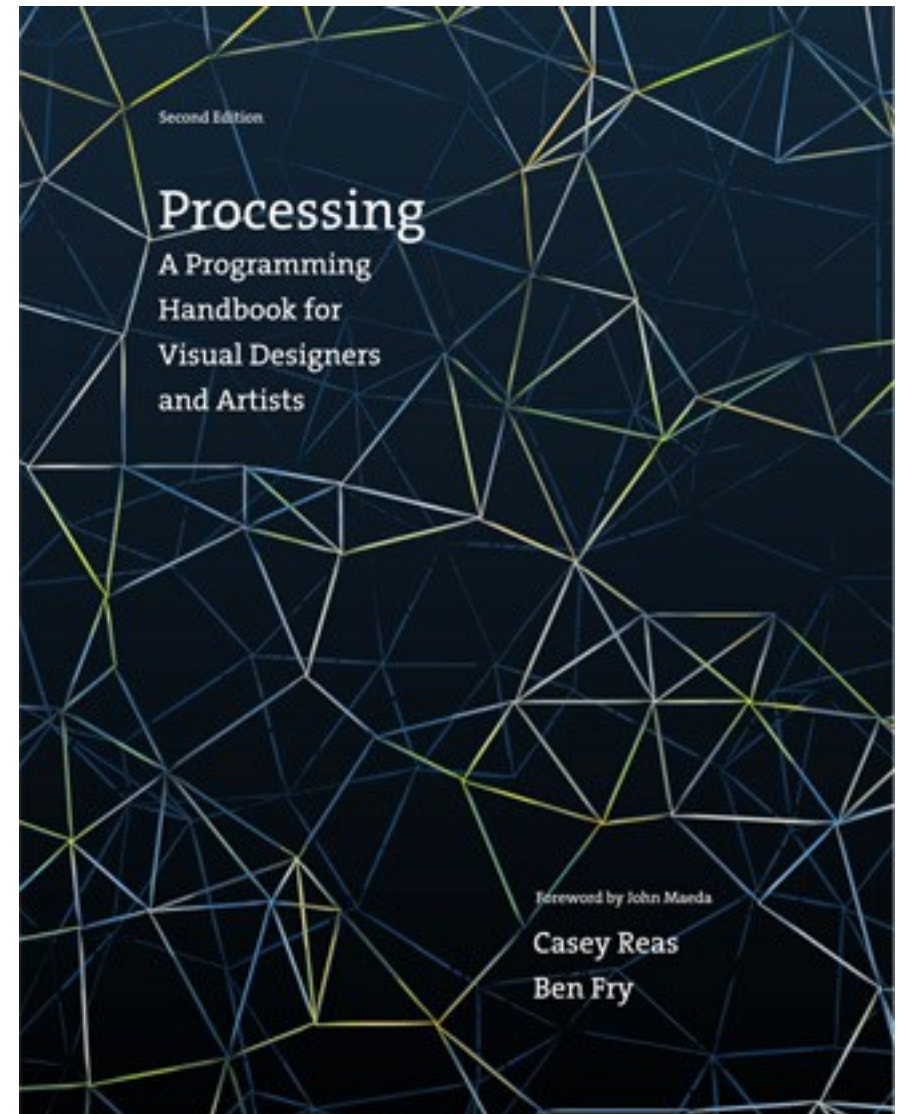
		 		 Processing
Availability	Free	Not Free	Free	Free
Installation	-	-	+	+
Simplicity	-	+	-	+
Community	Big	Big	Large and small	Big
2D Graphics	+/-	+	+/-	+
3D Graphics	+/-	+	+	Not Yet
Browser Interface	Requires Install	Requires Install	No Install (HTML5)	No Install (HTML5)



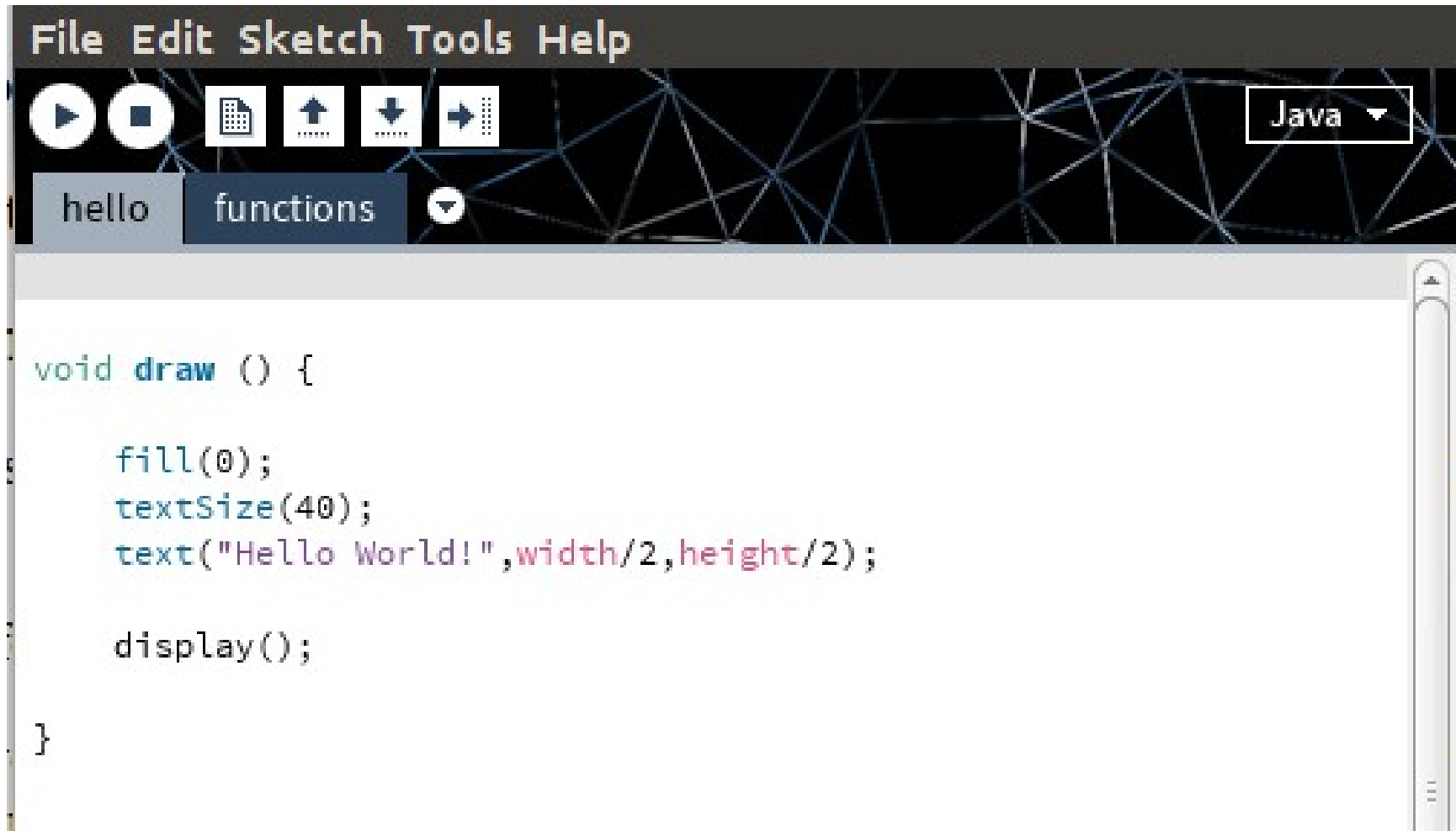
Processing



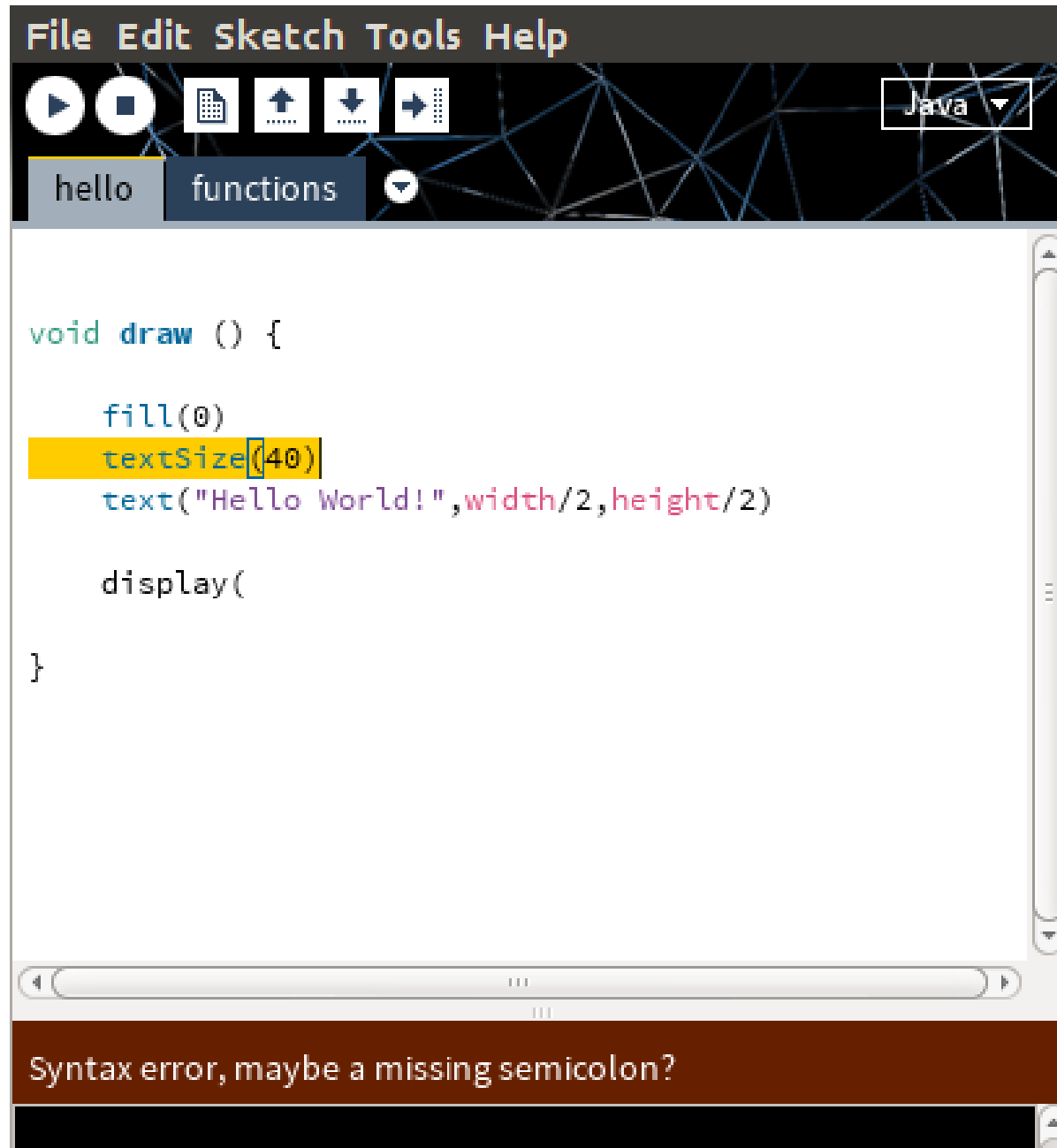
- Developed by Engineers from MIT
- Intended for non-programmers / artists
- Designed for “instant visual feedback” and 2D animations
- Native 60 fps operation
- Fully browser & tablet & smartphone compatible



The Processing Interactive Development Environment (IDE)



The Processing Interactive Development Environment (IDE)



Our Philosophy

- Students are given code and are asked to modify it
- At least initially, these modifications are given to the student in a step-by-step process
- Only by the 3rd lab does the student need to modify the program without full guidance on what to do

```
float x;  
float vx;  
float deltaVx;
```

This is the Entire Code that the student sees!

```
float theta = 0;  
float Fthrust = 30.0;  
float mass = 3.0;  
float dt = 0.1;
```

```
void draw() {  
  
    // Update velocities  
    vx += deltaVx;  
  
    // Update location  
    x += vx*dt;  
  
    // Set deltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
  
    // Turn on thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        } else if (key == CODED && keyCode == DOWN ) {  
            // Do nothing  
        }  
    }  
  
    // Draw ship and other stuff  
    display();  
  
} // end draw()
```

Exercise Sample

```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    // Update location  
    x += vx*dt;  
    // Set eltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
    // Turn or thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```



React to keyboard

Exercise Sample

```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    // Update location  
    x += vx*dt;  
    // Set eltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
    // Turn or thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```

React to keyboard

Pressing UP adds
rocket thrust!

Exercise Sample

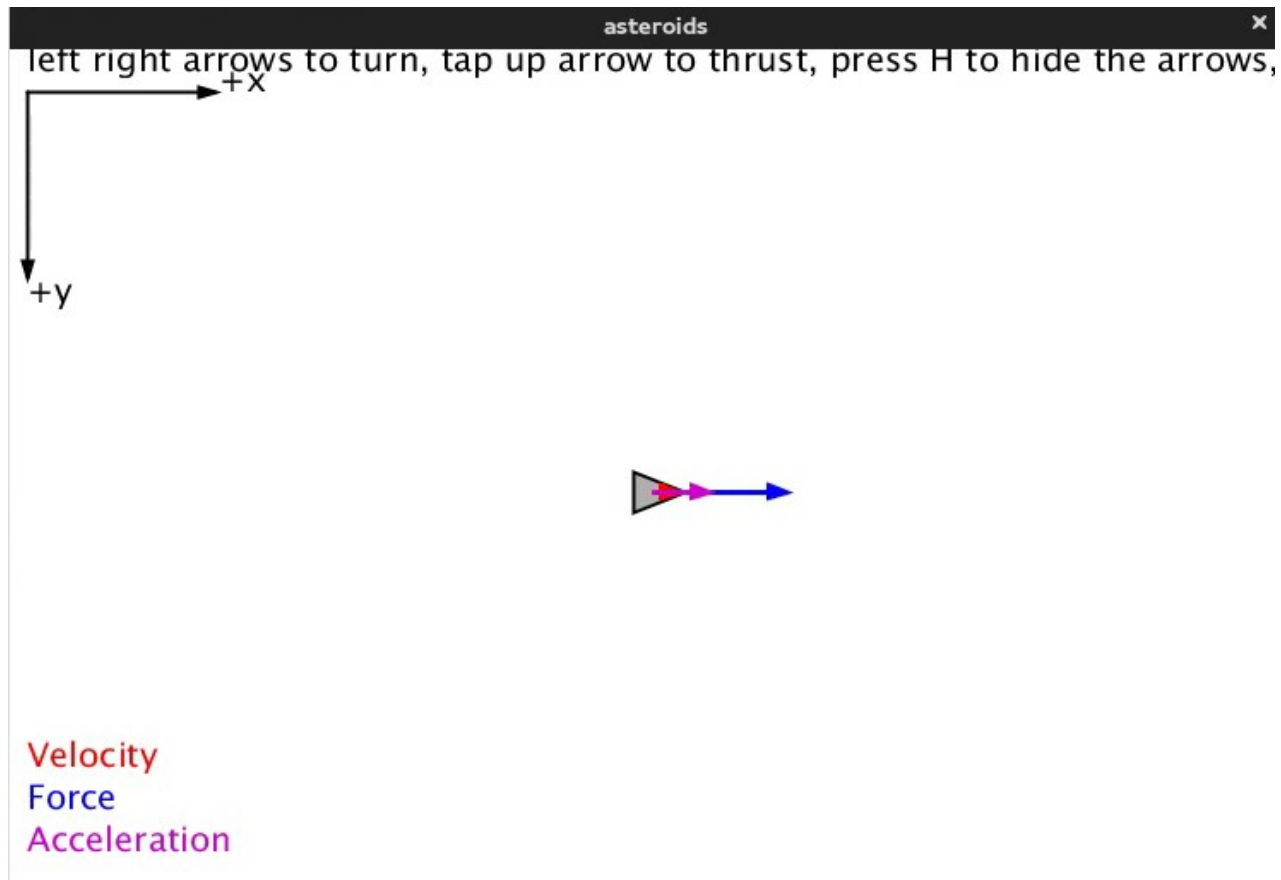
```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    // Update location  
    x += vx*dt;  
    // Set eltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
    // Turn or thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```

$$\Delta x = v \Delta t$$

React to keyboard

Pressing UP adds
rocket thrust!

Exercise Sample



Exercise Sample

```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    // Update location  
    x += vx*dt;  
    // Set eltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
    // Turn or thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```

$$\Delta x = v \Delta t$$

React to keyboard

Pressing UP adds
rocket thrust!

Exercise Sample

```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    // Update location  
    x += vx*dt;  
    // Set eltaV to 0 if user has it on  
    deltaVx = 0;  
    // Turn or thrust the ship depending on what is pres.  
    if (keyPressed)  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```

$$\Delta x = v \Delta t$$

ONLY 1D

Student's goal is to make it 2D!

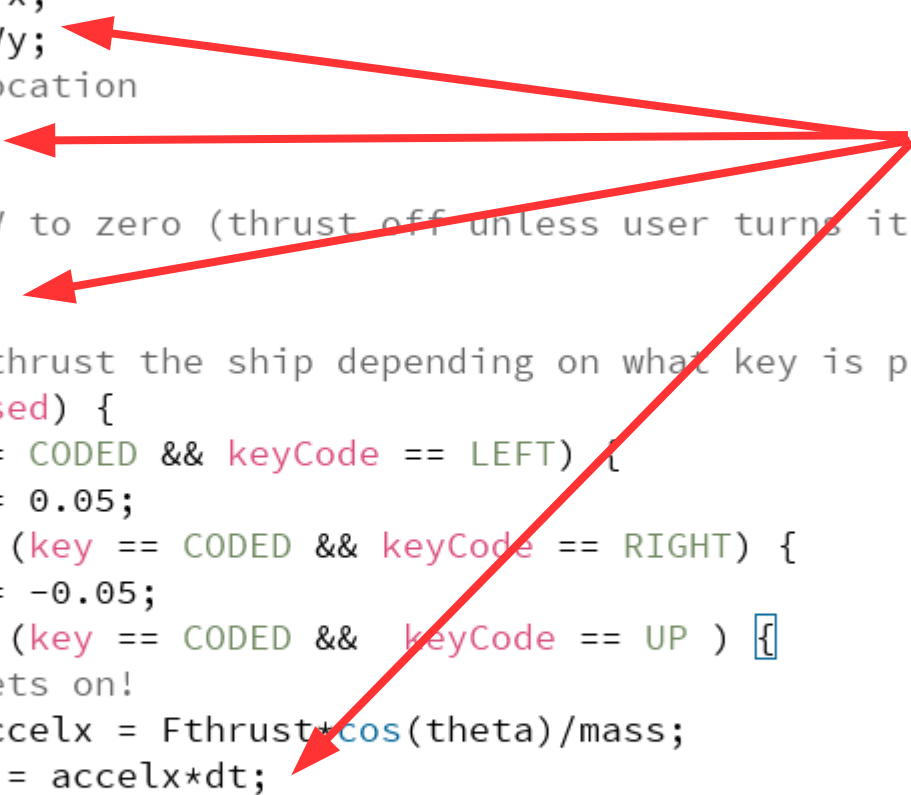
React to keyboard

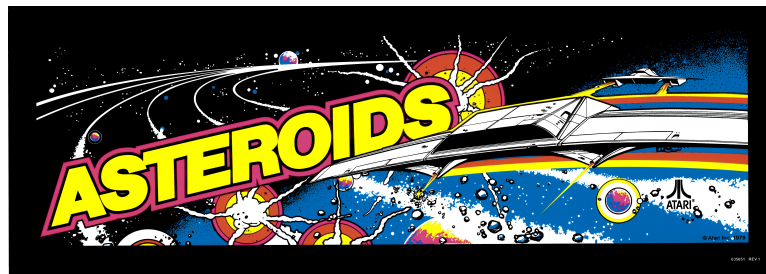
Pressing UP adds
rocket thrust!

Exercise Sample

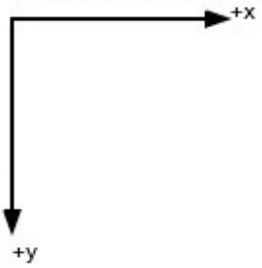
```
void draw() {  
    // Update velocities  
    vx += deltaVx;  
    vy += deltaVy;  
    // Update location  
    x += vx*dt;  
    y += vy*dt;  
    // Set eltaV to zero (thrust off unless user turns it on)  
    deltaVx = 0;  
    deltaVy = 0;  
    // Turn or thrust the ship depending on what key is pressed  
    if (keyPressed) {  
        if (key == CODED && keyCode == LEFT) {  
            theta += 0.05;  
        } else if (key == CODED && keyCode == RIGHT) {  
            theta += -0.05;  
        } else if (key == CODED && keyCode == UP ) {  
            // Rockets on!  
            float accelx = Fthrust*cos(theta)/mass;  
            deltaVx = accelx*dt;  
            float accely = -Fthrust*sin(theta)/mass;  
            deltaVy = accely*dt;  
        }  
    }  
    // Draw ship and other stuff  
    display();  
} // end draw()
```

Simply add other
dimensions
Independently!





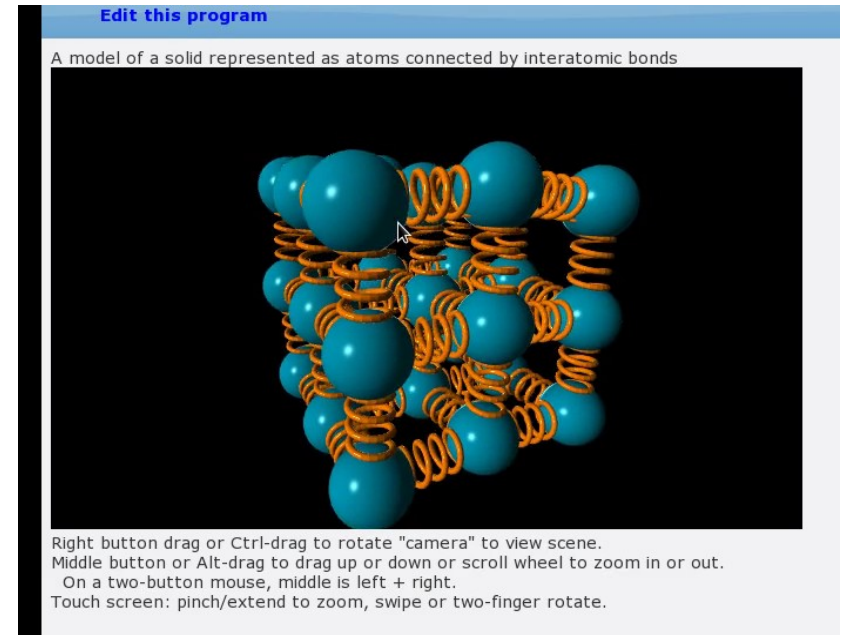
left right arrows to turn, tap up arrow to thrust, press H to hide the arrows, press U to un-hide



Velocity
Force
Acceleration

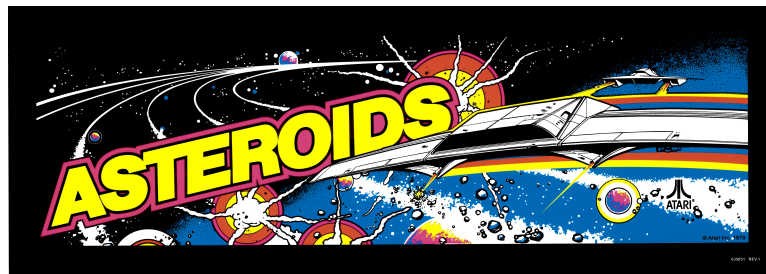
VPython = Visual Python

- Developed by Bruce Sherwood & collaborators at North Carolina State University
- Described as “3D Programming for Ordinary Mortals”
- **Not really intended for 2D graphics**
- Integrated into *Matter & Interactions* textbook curriculum, which is used at “several dozen” institutions
- Nice web interface (glowscript.org)

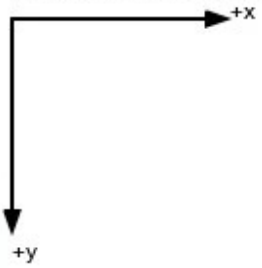


	VPython / Matter & Interactions Approach	Our Approach (processing)
Emphasis	Illustrate physics in 3D early to ease transition to electro- magnetism later	Illustrate 2D physics simply & in a fun way
Interactivity	Centered on exploring 3D world & perspectives	Game-like
Other goals	Show that all interactions can be understood as changes in momentum	Replicate “canned” interactives (No emphasis on momentum)

Tour of Programming Labs



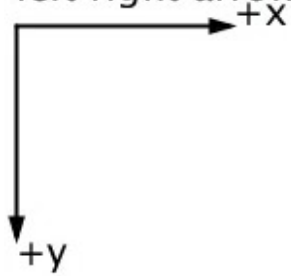
left right arrows to turn, tap up arrow to thrust, press H to hide the arrows, press U to un-hide



Velocity
Force
Acceleration

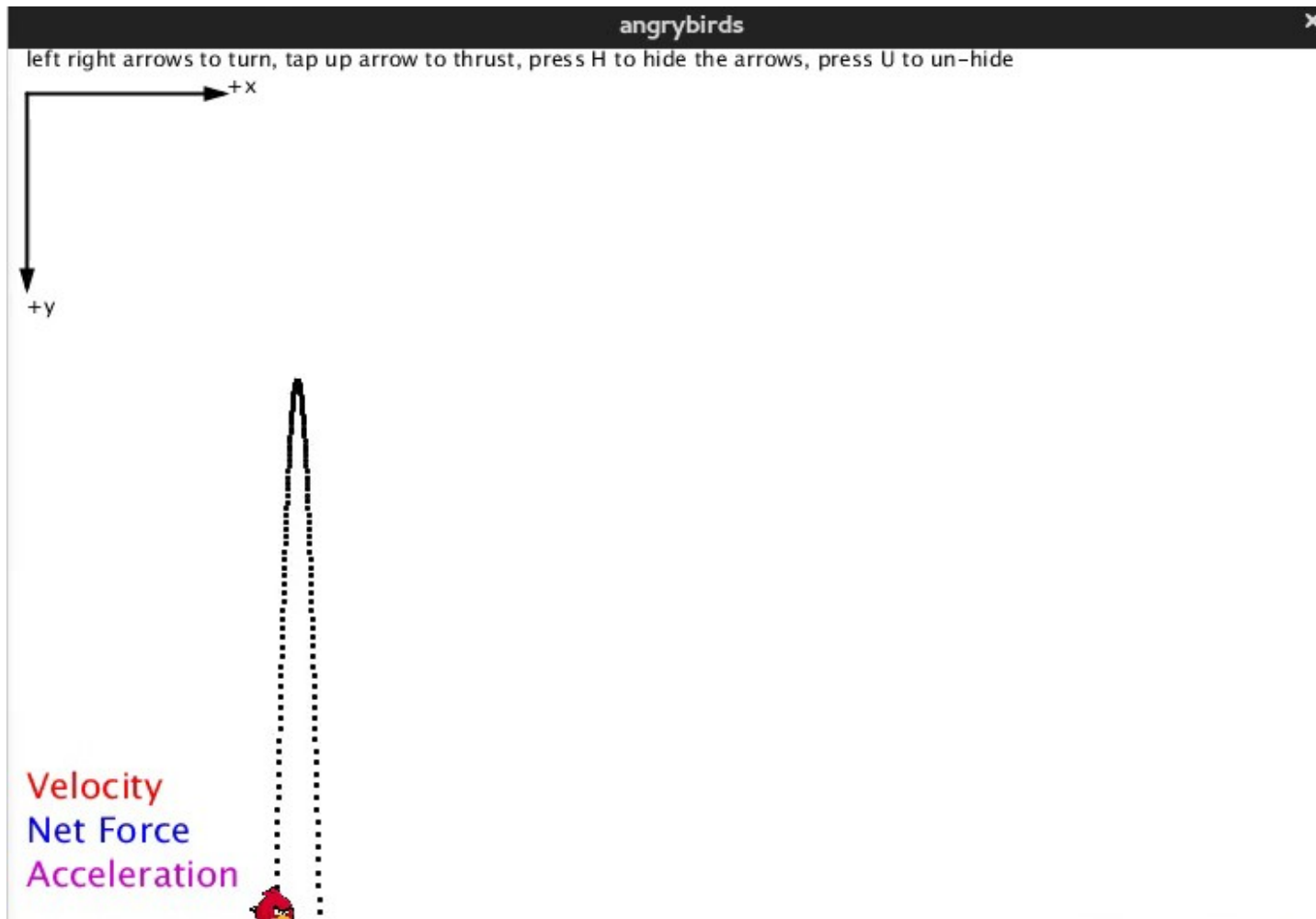


lunarlander x
left right arrows to turn, tap up arrow to thrust, press H to hide the arrows,



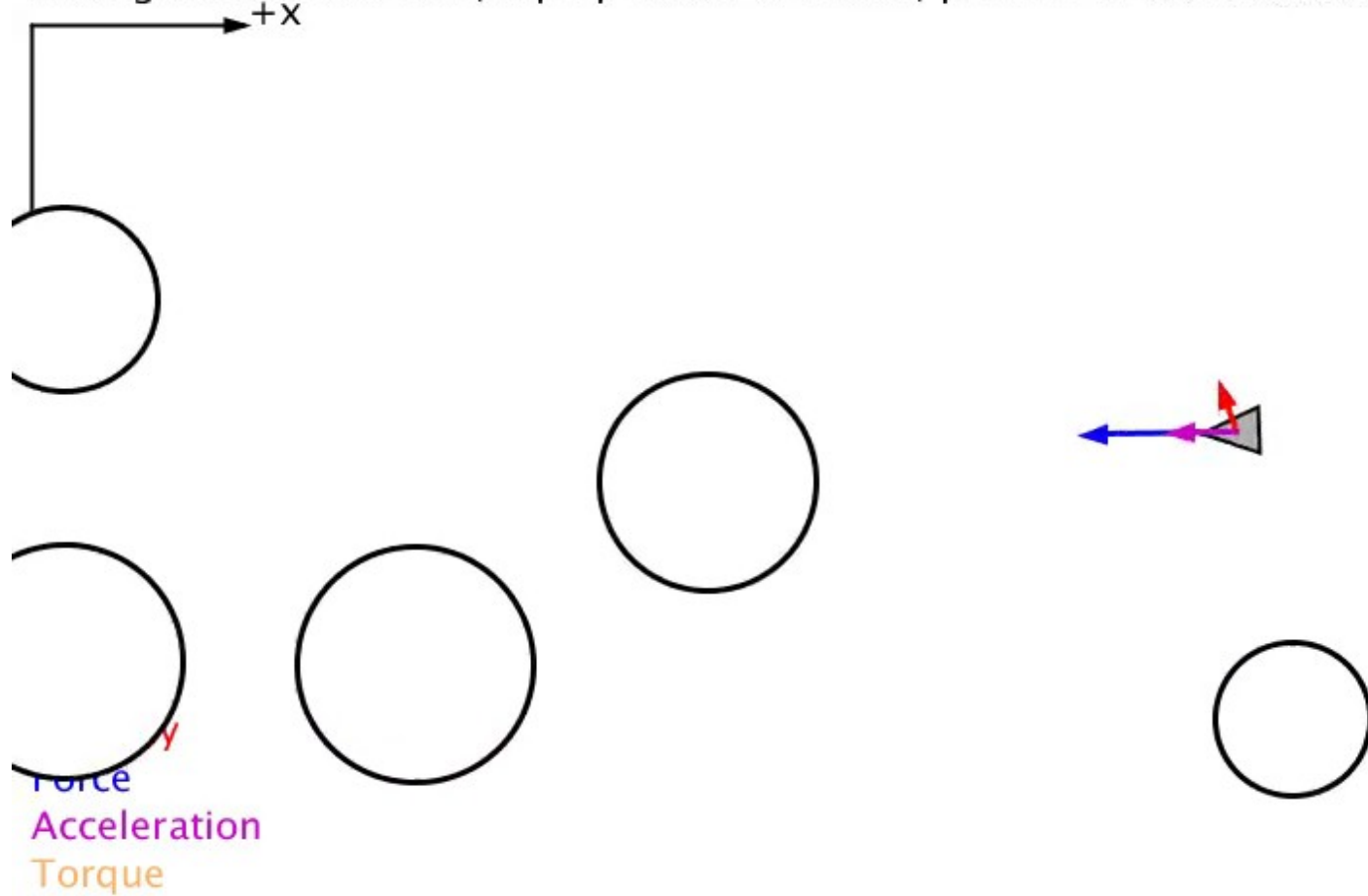
Velocity
Net Force
Acceleration





Asteroids with Torque (title slide)

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



Assessment

Assessment

- A online survey was conducted in a class of 24 physics students at OSU Marion
- This took place after the asteroids lab (without torque)
- In addition, we have pre and post-test scores of 4 rocket questions from Force Concept Inventory (FCI)
- Short summary of the results:
 - **The survey shows that the students enjoy the lab and that it is not too hard**
 - **FCI scores indicate larger gains for these 24 students on these questions than on Columbus campus**

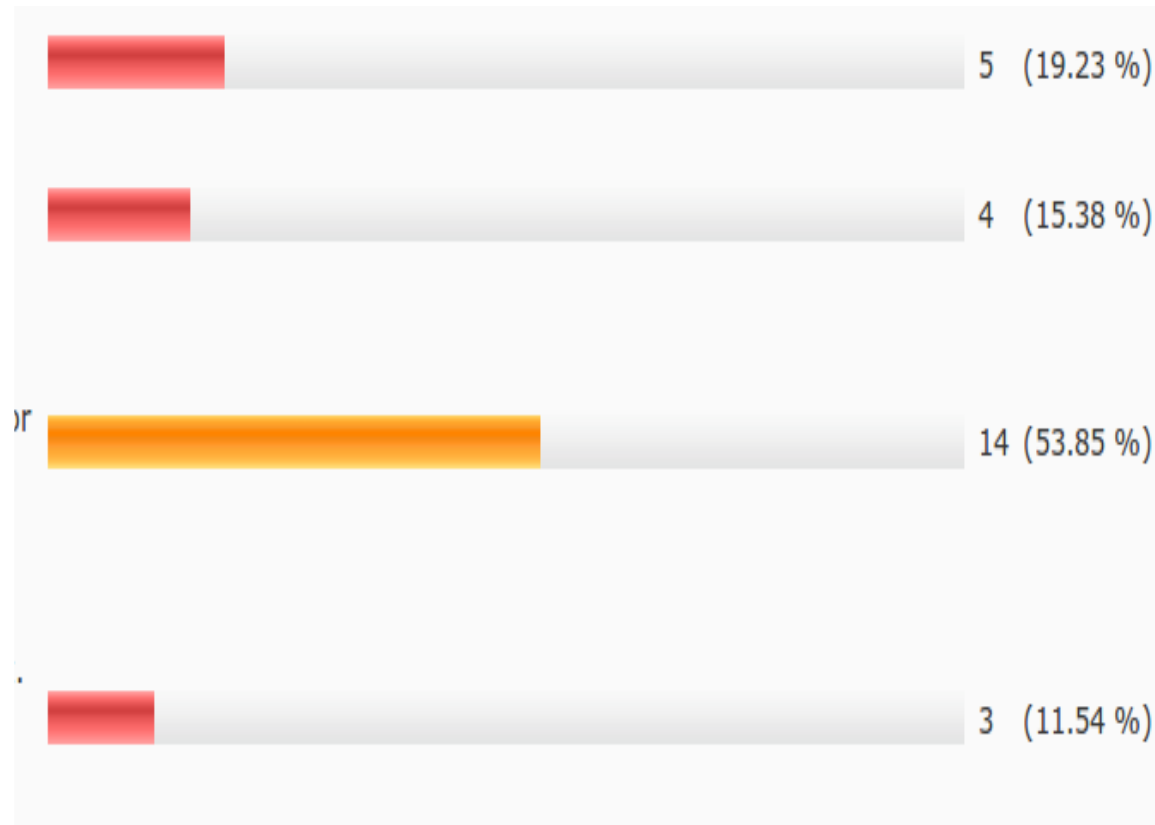
Have you ever written or modified a computer program before?

No, I have never done any programming before.

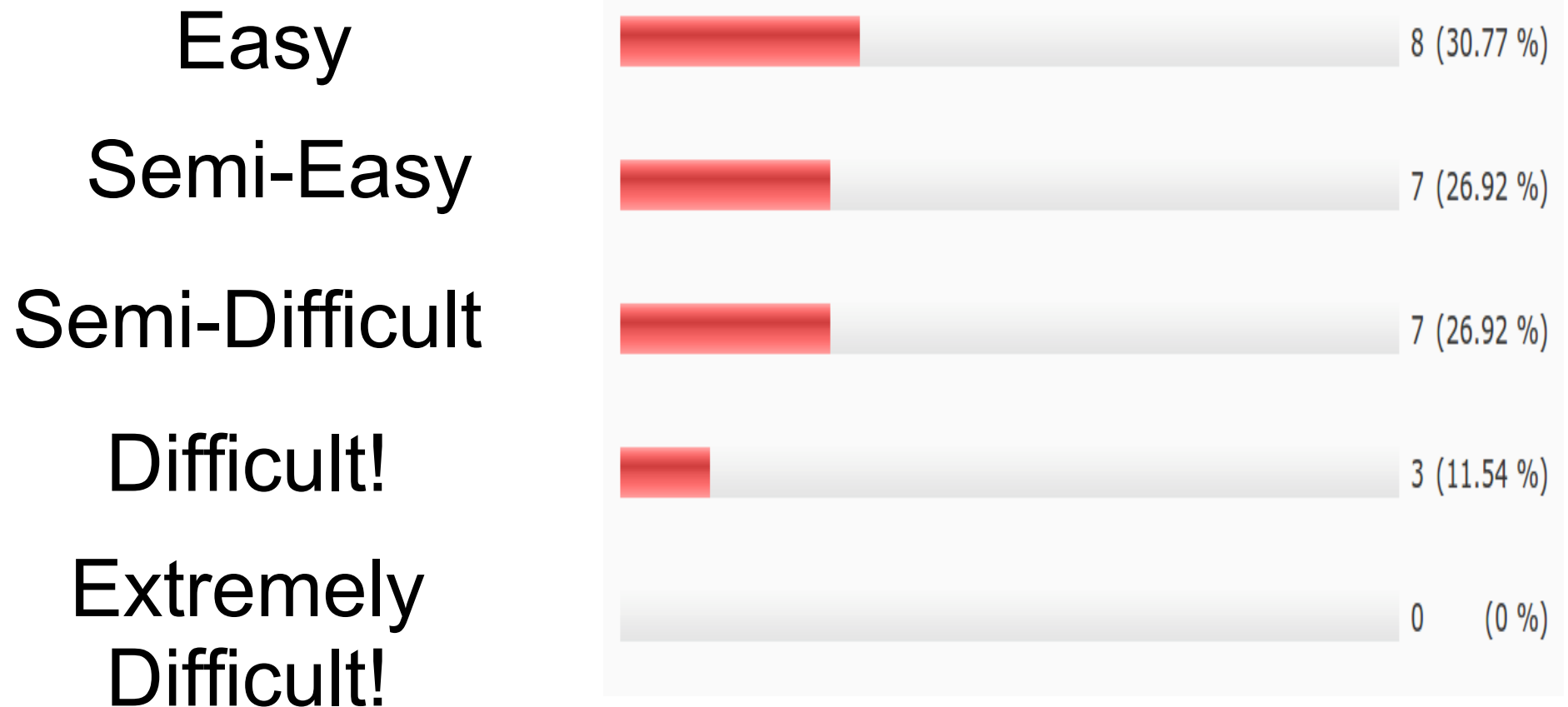
Yes, I have done a little bit of programming before.

Yes, I but I still feel like I have a lot to learn.

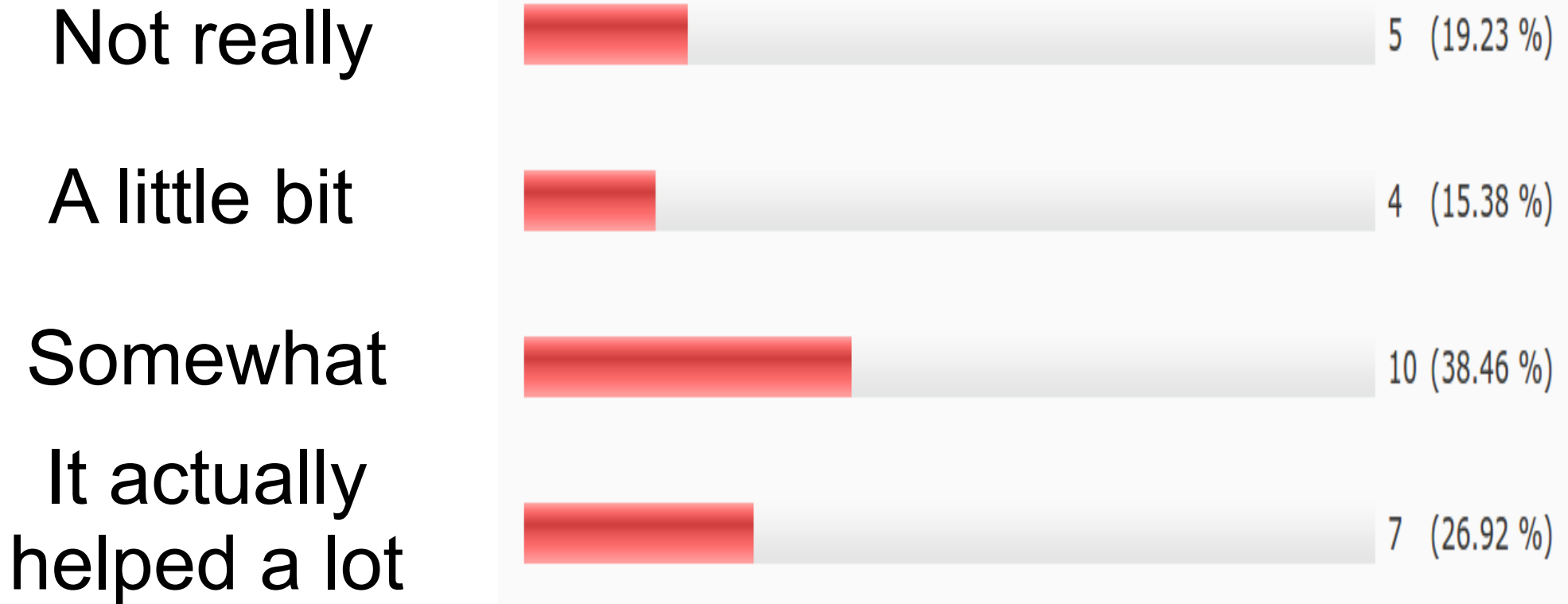
Yes, I have a lot of programming experience



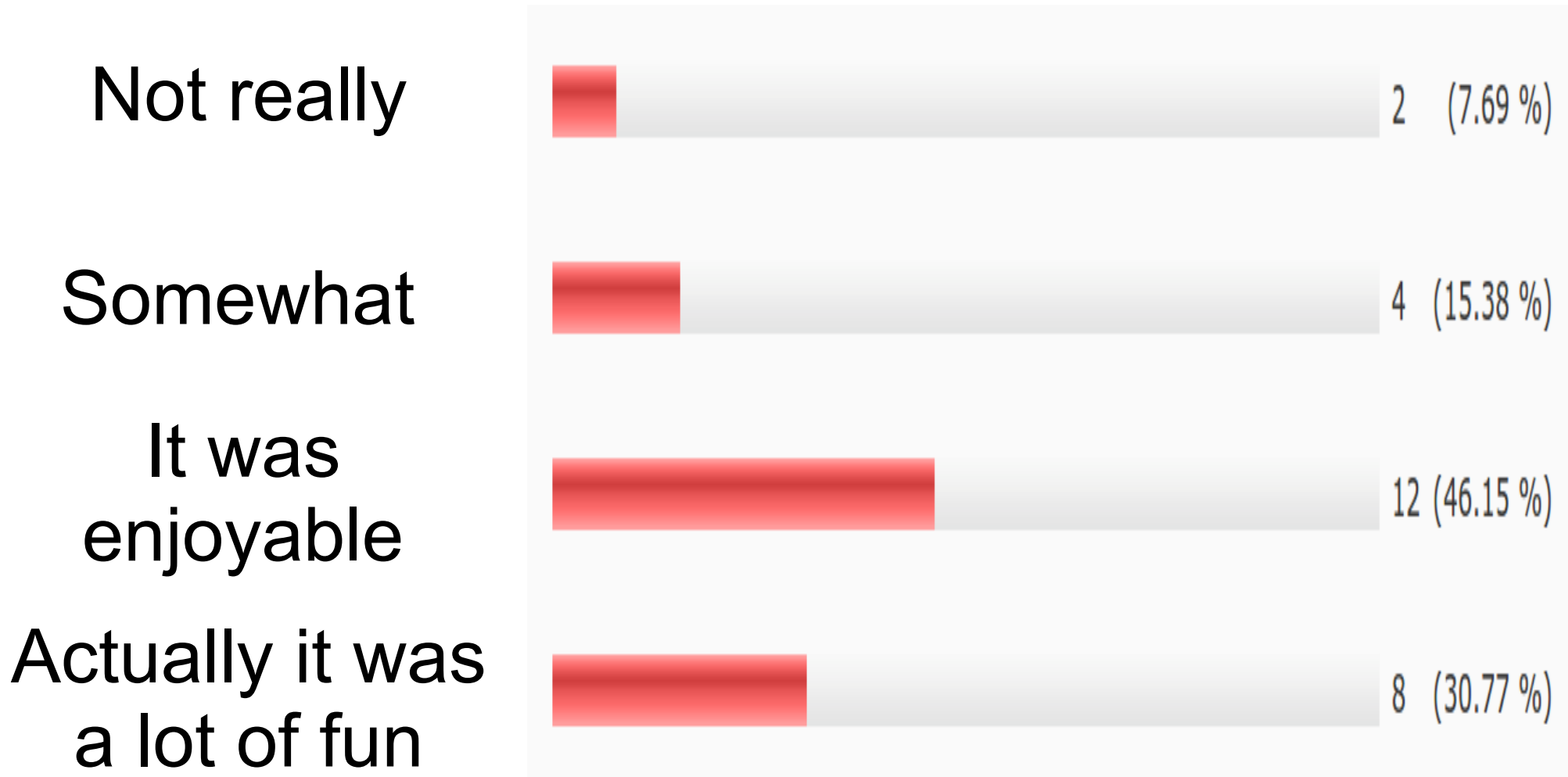
Compared to the difficulty of completing other (non-programming) labs, **how difficult was it to complete the asteroids lab?**



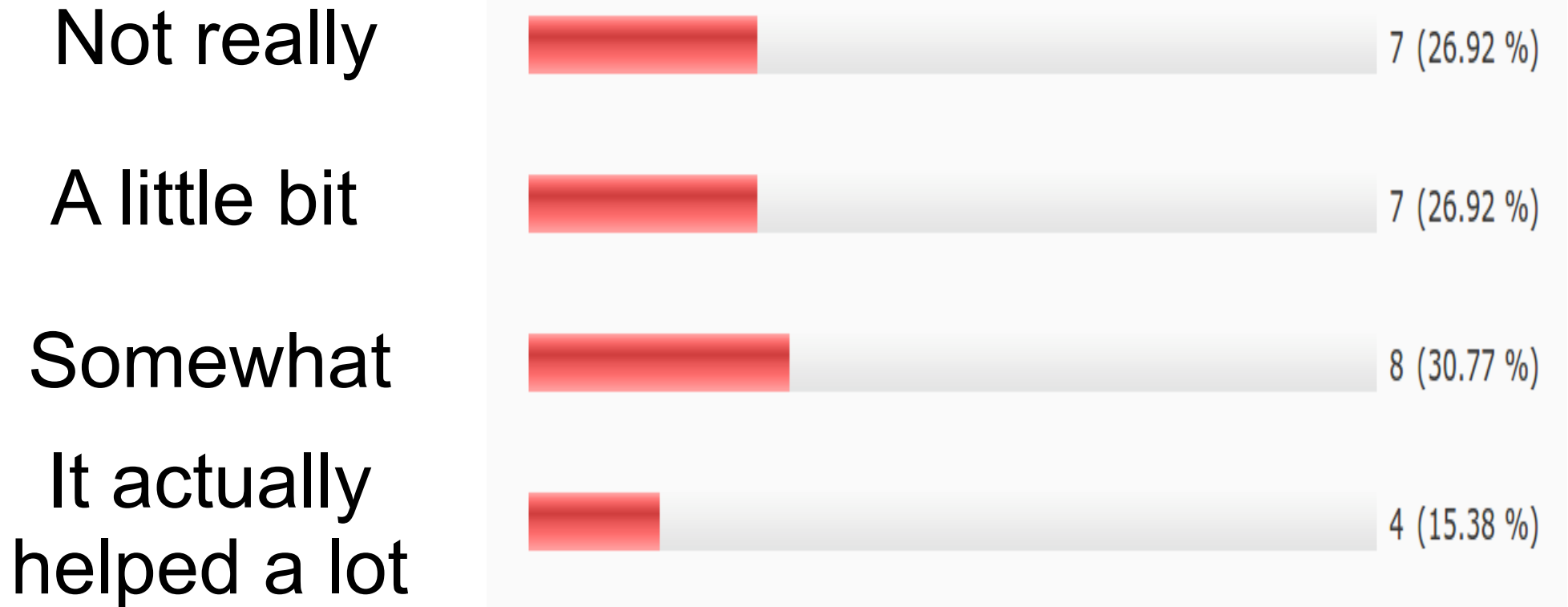
Did the asteroids lab help you understand programming better?



Was the programming lab fun?



Did the asteroids lab help you understand physics better?



Force Concept Inventory (FCI)

- There are four rocket questions on the FCI, which is the most widely used assessment tool in physics education research
- Students took the full FCI at the beginning of the course
- After the asteroids lab, students were tested on these four rocket questions again
- Results can be compared to FCI pre and post-testing from OSU Columbus campus (A. Heckler)

Rocket Questions	Pre-test % Correct OSU Marion (N=24)	Post-test % Correct OSU Marion (N=24)	Pre-test % Correct OSU Columbus (N=166)	Post-test % Correct OSU Columbus (N=166)
Q21	45	→ 68	43	→ 40
Q22	45	→ 73	46	→ 59
Q23	32	→ 86	48	→ 54
Q24	64	→ 86	75	→ 75

Student Comments

“I was confused at first as to how to get my programs to run and display properly... Once I discovered that the lab was very step-by-step and I found that even with my basic knowledge it was easy and fun.”

“I think this is a great way for students to apply and visualize the functions that we learn in this course. I really enjoyed it!”

“I am a computer science major so programming comes easily to me. The lab was fun because i got to make a program with an actual use (unlike some programming class projects).”

“I spent more time trying to figure out how programming languages work than actually understanding the physics. In the end I was able to complete the tasks simply by mimicking code elsewhere in the program...so it worked, but I didn't have a very thorough grasp of how it worked.”

“Something that could be improved is the instructions to turn in the program just to make it a little easier. All in all a very fun lab.”

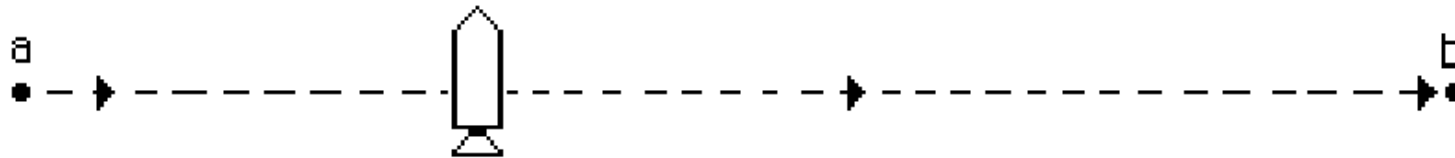
“I enjoyed this lab, although I don't really get coding since there is so much to it. I do think that there could be a little clearer in the instructions about the files, but other than that I thought it was great. Thanks!”

Conclusions

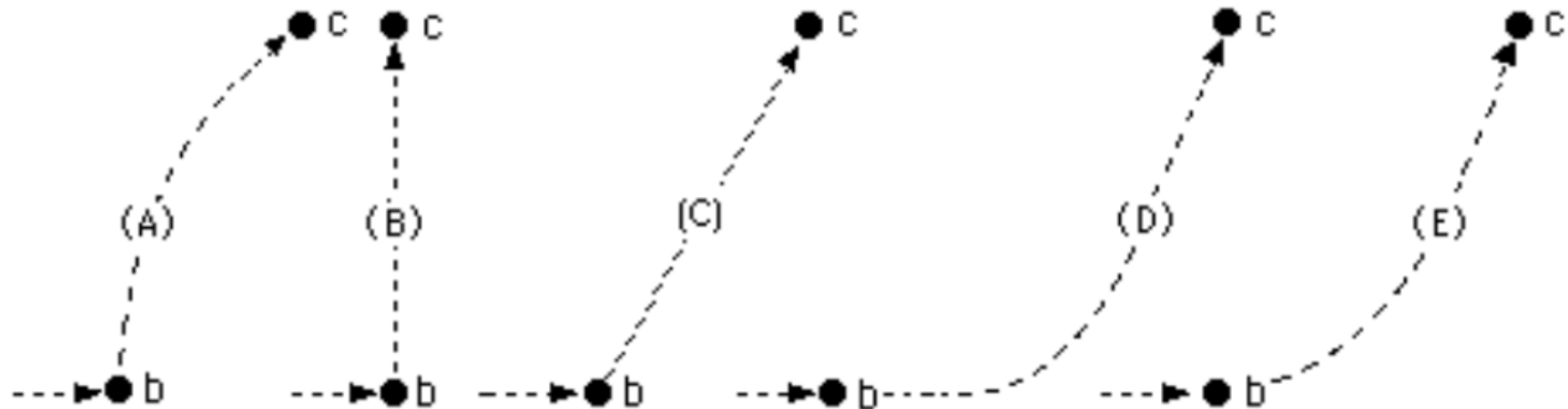
- Processing is a full-featured, free platform for adding programming into STEM curricula
- This goes one level deeper than interactive examples
- Students enjoy a game-making & modifying approach
- Preliminary assessment is very positive. Further assessment/studies planned for fall & spring.

Extra Slides

A rocket drifts sideways in outer space from point "a" to point "b" as shown below. The rocket is subject to no outside forces. Starting at position "b", the rocket's engine is turned on and produces a constant thrust (force on the rocket) at right angles to the line "ab". The constant thrust is maintained until the rocket reaches a point "c" in space.



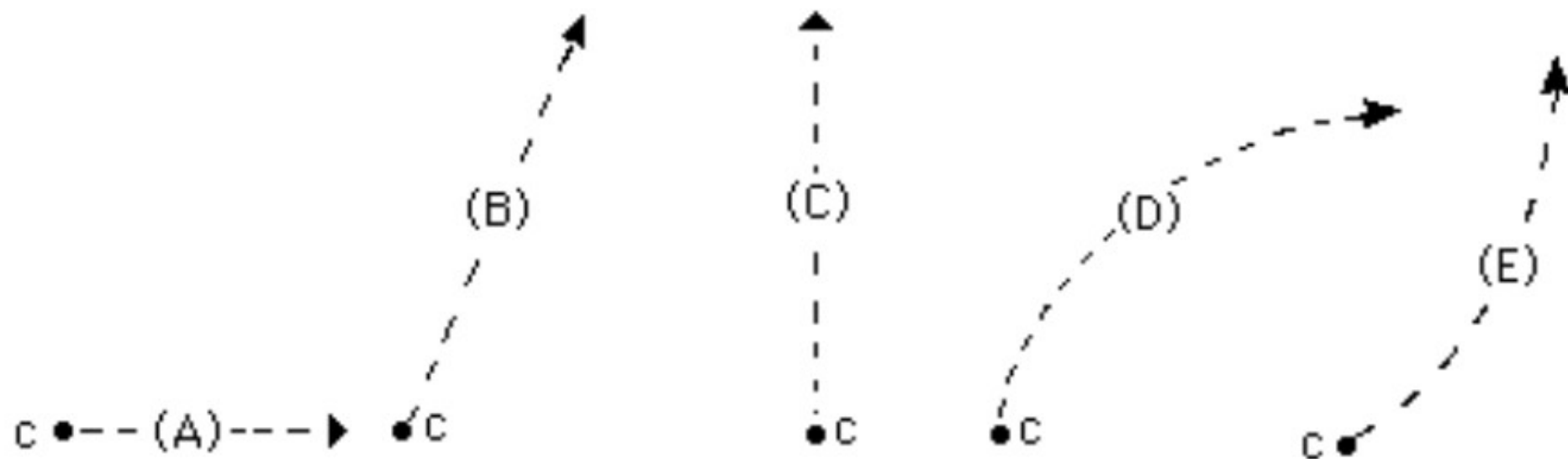
21. Which of the paths below best represents the path of the rocket between points "b" and "c"?



22. As the rocket moves from position "b" to position "c" its speed is:

- (A) constant.
- (B) continuously increasing.
- (C) continuously decreasing.
- (D) increasing for a while and constant thereafter.
- (E) constant for a while and decreasing thereafter.

23. At point "c" the rocket's engine is turned off and the thrust immediately drops to zero. Which of the paths below will the rocket follow beyond point "c"?



24. Beyond position "c" the speed of the rocket is:
- (A) constant.
 - (B) continuously increasing.
 - (C) continuously decreasing.
 - (D) increasing for a while and constant thereafter.
 - (E) constant for a while and decreasing thereafter.