Computational Activity 3: Simulating motion of a fan cart

Derived from activity VP03 in the instructor resources that accompany Ruth Chabay and Bruce Sherwood's Matter and Interactions *textbook.*

There is a typical pattern for the kinds of computational models that you will create in this course. The programs will generally follow something like this template to track the motion in small time steps Δt:

- Define the object's initial position $\vec{r}$ and momentum $\vec{p}$.
- Loop (forever, or until some amount of time has passed, or some condition has been fulfilled):
  - Calculate the net force $\vec{F}_{net}$ based on the physical interactions that affect the object.
  - Use the net force to update momentum: $\vec{p} \leftarrow \vec{p} + \vec{F}_{net}\Delta t$.
  - Calculate velocity from momentum: $\vec{v} \leftarrow \vec{p}/m$.
  - Use velocity to update position: $\vec{r} \leftarrow \vec{r} + \vec{v}\Delta t$.

Visit `glowscript.org` to run VPython code.

In this activity, you will simulate the simplest nonzero net force: a constant. For example, it might represent the force that is generated by a fan mounted on a cart that moves on a level track. Here is the beginning of a VPython program that would implement that situation:

```
scene.width = 800 # pixels
scene.y = 400 # pixels

vgraph = gcurve(color=color.green)

# All dimensions are in m
track = box(pos=vector(0,-0.05,0), length=2.0, height=0.03, width=0.10, color=color.white)
cart = box(pos=vector(0,0,0), length=0.1, height=0.06, width=0.06, color=color.cyan)

m_cart = 0.8 # kg
p_cart = m_cart * vector(0.2,0,0) # kg * m/s
delta_t = 0.01 # s
t = 0 # s
```

1. Before you run this program, please make a prediction of what will happen: will the cart move? Will a graph appear? Then, please check your prediction.

2. Modify the program to position the cart so that its left end is initially aligned with the left end of the track.

3. Write a `while` loop to move the cart from one end of the track to the other with constant momentum. (You did something like this in a previous activity.) If you encounter unexpected behavior, it may help to Google how the `rate` function works in VPython.

4. Inside the loop, instruct the computer to calculate the velocity of the cart from its mass and momentum. This will be important when we add a nonzero net force.

5. After the loop completes, print the value of the elapsed time. (Note that because of roundoff errors inherent in floating-point calculations, this value may actually be slightly larger than the time you expect.)

6. Make a prediction: what should a graph of p_x versus t look like if momentum is constant?

7. To fill in data points on your graph, insert the following line of code inside your loop, at the end of the loop: `vgraph.plot(pos=(t, p_cart.x))`. Was your prediction correct?

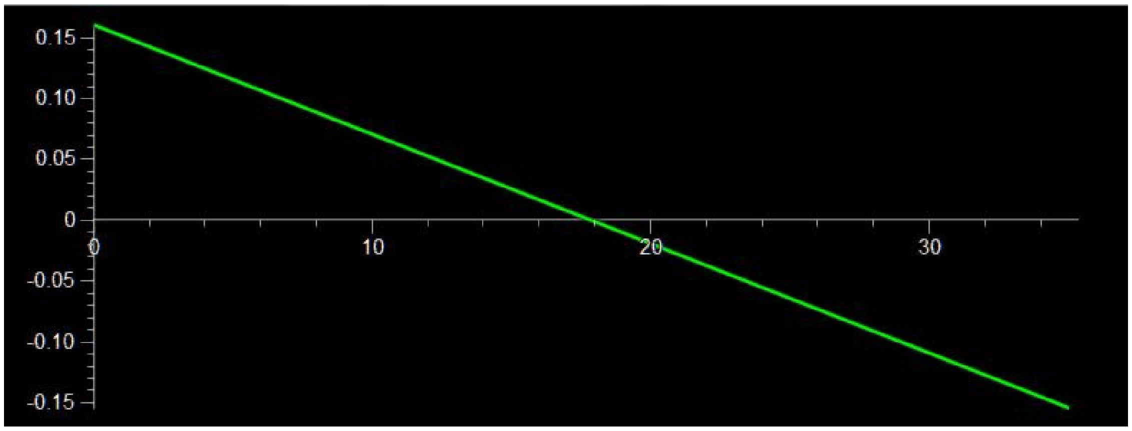
8. Add an instruction to define a constant force named `Fnet` (a vector) in the $+x$ direction.

9. Add statements to the `while` loop in your program to update the cart's momentum to reflect the net force applied. (See the general template above.)

10. Adjust the magnitude of the force so that it takes about half as long for the cart to reach the end of the track with the fan turned on as it did with the fan off.

11. Explain what feature of the graph of p_x versus t corresponds to the constant net force applied to the system.

12. Find values for initial position, initial momentum, and net force that produce a graph of p_x versus t like the one below. (You may need to let your loop run longer than before.)



13. Describe the motion of the cart that corresponds to this graph.

14. You're done – each person should upload the final version of their program to WorldClass.