VPython: Motion of a charged particle in a magnetic field

Start with the program shell (on course website: Files page; also printed on the back of this handout). The program shell draws a "floor" and displays a uniform magnetic field, which is initially set to < 0, 0.2, 0 > T.

A: First, get a proton moving across the screen

A1. Create a sphere representing a proton, at location < 0.3, 0.15, 0 >. Give it an initial speed of 3e6 m/s (this is only 1% of the speed of light), moving in the +z direction, by defining an appropriate vector quanitity:

```
velocity = vector(## you fill this in)
```

After creating the proton, add the following line of code to prepare for leaving a trail later:

```
trail = curve(color=particle.color) ## use your own variable names
```

A2. Write a loop to move the proton in the direction of its velocity, by following the instructions below. (If this is a very familiar task to you, you can do it now, without reading the rest of this section A.)

A computer animation of the motion of an object uses the same principle as a movie or a flip book does. You display the object at successive locations, each at a slightly later time; to your eye, the motion can look continuous. In a program, you calculate the position of the object, and it is displayed in that position. You calculate where the object will be a very short time `deltat` later, and change its position to the new location, so it is displayed in the new location. You will use the relation between velocity (a vector) and position (a vector) to calculate the position of a moving particle at successive times, separated by a short time step . Remember that, in vector terms:

$$\vec{r}_f = \vec{r}_i + \vec{v}\Delta t \left(\text{because } \vec{v} = \frac{\Delta \vec{r}}{\Delta t} \right)$$

In VPython, this translates to a statement like:

```
particle.pos = particle.pos + velocity*deltat
```

where `velocity` is the vector quantity which you have initialized earlier in the program.

This statement may look odd to you if you have not had much programming experience. The equals sign here has the meaning of "assignment" rather than equality. This instruction tells VPython to read up the old position of the particle, add to it the quantity, and store the new position as the current position of the particle. This will automatically move the particle to the new position.

Read the program shell: What is the value of `deltat` that is set in the program shell?

Inside the loop, ignoring steps 1-3 for now, update the position of the proton by using its velocity (step 4):

```
particle.pos = particle.pos + velocity*deltat ##use own variable names
```

Inside the loop, add this line after updating the position:

```
trail.append(pos=particle.pos) ## use your own variable names
```

A3. Run your program. The proton should move in a straight line and leave a trail.

Check your work so far with a neighboring group, then both groups check with instructor

B: Now you are ready to add a magnetic force, and observe its effect

B1. Once you have the proton moving correctly in a straight line, add a magnetic force.

INSIDE the loop, **BEFORE** the line updating the proton's position, you need to:

a. Calculate the magnetic force on the proton (remember, this is a vector).

b. Use the magnetic force to modify the velocity of the proton, using the momentum principle. Reminder:

$$\Delta\vec{p} = \vec{F}\Delta t, \text{ where } \vec{p} \approx m\vec{v} \text{ if speed is small compared to speed of light}$$

To calculate the new momentum, we add the impulse to the old momentum:

$$\vec{p}_{\text{new}} = \vec{p}_{\text{old}} + \vec{F}\Delta t$$

Translating this into VPython, your momentum update statement inside the loop might look like this:

```
particle.p = particle.p + Fmagnetic * deltat ## or whatever variable
names you used
```

Before the start of the loop, initialize the particle's momentum, based on $\vec{p} \approx m\vec{v}$ (since $v << c$).

Inside the loop, after updating the momentum but before updating the position, you need to calculate

```
velocity = particle.p/particle.m ## or whatever variable names you used
```

Do the following experiments and be prepared to answer the related questions.

B2. Describe the path of the proton's motion. Explain qualitatively in terms of force and momentum.

B3. What is the approximate radius of the path?

B4. What happens to the radius of the path if you double the initial speed? Halve the initial speed?

B5. Inside the loop, update the time (`t = t + deltat`). Change the while statement to

```
while t<3.34e-7:
```

and run the program. The proton should make just one complete orbit. If not, review your calculations. If you double the initial speed, the radius of the path will also double, so how far around will the proton get in the time 3.34e-7 seconds? Try it. What do you find? This striking behavior is the basis for the cyclotron (see textbook).

B6. Save the current starting position and velocity, and start the proton with velocity in the direction of the magnetic field. What kind of path does the proton follow? Why?

B7. Restore the original starting position and velocity, and change the while statement to do 10 complete circles. Add a significant +y component to the proton's velocity. Now what kind of a path does the proton follow? Why? What's the connection with question B6?

B8. Change the proton to an antiproton, a particle with the same mass as the proton, but a charge of -e. What changes?

Check your work with a neighboring group, then both groups check with instructor, who will ask about your answers to the questions above.
Then Reporter turns in program to WebAssign.

Program Shell (available as handout on course website: see Resources/Lab Handouts):

```
from __future__ import division
from visual import *

## INITIAL VALUE OF B
B0 = vector(0,0.2,0)

## THIS CODE DRAWS A FLOOR AND DISPLAYS THE MAGNETIC FIELD *************
xmax = .6
dx = .1
yg = -.1
for x in arange(-xmax, xmax+dx, dx):
   curve(pos=[(x,yg,-xmax),(x,yg,xmax)], color=(.7,.7,.7))
for z in arange(-xmax, xmax+dx, dx):
   curve(pos=[(-xmax,yg,z),(xmax,yg,z)],color=(.7,.7,.7))
bscale = 1
for x in arange(-xmax, xmax+dx, 2*dx):
   for z in arange(-xmax, xmax+dx, 2*dx):
         arrow(pos=(x,yg,z), axis=B0*bscale, color=(0,.8,.8))

## YOUR PROGRAM BEGINS HERE ##*****************************************
deltat = 1e-11
t = 0.
while 1: ## Implement steps 1-3 after making the proton move in a straight line.
   ## 1) Add code to calculate the magnetic force on moving proton.
   ## The function cross(a,b) calculates the cross product axb.

   ## 2) Calculate the new momentum of the proton.

   ## 3) Calculate the new velocity of the proton (p/m).

   ## 4) Update the position of the proton (moves in a straight line initially).
```