Collisions

Objectives

Before doing this activity you should have studied Chapter 10 through Section 10.9 in *Matter & Interactions 4e.*

After completing this activity you should be able to:

- Qualitatively explain the effect of impact parameter on scattering angle.
- Demonstrate conservation of momentum in a computational model of a collision.

1 Explain and Predict

Study the following VPython program carefully. Make sure you understand the whole program, but don't run the program yet. Reading and explaining program code is an important part of learning to create and modify computational models.

```
from __future__ import division, print_function
from visual import *
from visual.graph import *
scene.width = 800

##Constants
massAu = (79+118)*1.7e-27
massAlpha = (2+2)*1.7e-27
qAu = 2*1.6e-19
qAlpha = 79*1.6e-19
oofpez = 9e9     ## one-over-four-pi-epsilon-zero
deltat = 1e-23

##Objects
Au = sphere(pos=vector(0,0,0), radius=8e-15, color=color.yellow, make_trail=True)
Alpha = sphere(pos=vector(-1e-13,0,0), radius=2e-15, color=color.red, make_trail=True)

#Initial Values
pAu = massAu*vector(0,0,0)
pAlpha = vector(1.043e-19,0,0)
t = 0

##Calculation Loop
while t < 1e-20:
    rate(1000)
    Alpha.pos = Alpha.pos + (pAlpha/massAlpha)*deltat
    t = t + deltat
```

Without running the program, answer the following questions:

⇒ What is the physical system being modeled?

⇒ In the real world, how should this system behave? On the left side of your whiteboard or paper, draw a sketch showing how you think the objects should move in the real world.

⇒ Will the program as it is now written accurately model the real system?

⇒ On the right side of the whiteboard or paper, draw a sketch of how the objects created in the program will actually move on the screen, based on your interpretation of the code.

Check your work before continuing.

2 Check Your Predictions

⇒ Run the program. Does it behave as you predicted?
⇒ If the computational model is not adequate or complete, modify the program to create a physically correct model of the system.

3 Graphs of Momentum Components

On a whiteboard or a piece of paper, sketch your predictions for the following graphs:

- $p_{Au,x}$ versus time
- $p_{Alpha,x}$ versus time
- $(p_{Au,x} + p_{Alpha,x})$ versus time

On a separate set of axes, sketch your predictions for the following graphs:

- $p_{Au,y}$ versus time
- $p_{Alpha,y}$ versus time
- $(p_{Au,y} + p_{Alpha,y})$ versus time

Now, add code to your program to produce these graphs. To produce two separate graph windows, you must explicitly create each window (called a `gdisplay`). By default a `gcurve` will belong to the most recently created `gdisplay`. The code to set this up (before the loop) might look like this:

```
X_momentum = gdisplay(x=0,y=400,height=200)
px_Alpha = gcurve(color=color.red)
px_Au = gcurve(color=color.blue)
px_total = gcurve(color=color.cyan)

Y_momentum = gdisplay(x=0,y=600, height=200)
py_Alpha = gcurve(color=color.red)
py_Au = gcurve(color=color.blue)
py_total = gcurve(color=color.cyan)
```

⇒ Do the graphs look like your predictions?
⇒ Is momentum conserved in your model?
⇒ If momentum is not conserved, modify your program to fix your model.

4 Impact Parameter

The impact parameter b is what the closest distance between centers of the alpha particle and the gold nucleus would be if there were no deflection, as discussed in Section 10.6 of *Matter & Interactions 4e.*
⇒ Set $b = 5 \times 10^{-15}$ m in the initial values section of the program.
⇒ Use b to change the initial position of the alpha particle in your code.
⇒ How do you think your p_y graphs will change due to the new impact parameter?
⇒ Run your program.

5 Scattering Angle

The scattering angle is the angle measured between the initial momentum vector and the final momentum vector of the scattered alpha particle. The components of the unit vector corresponding to the final momentum are its direction cosines, so the x component of the unit vector $\hat{p}$ is the cosine of the scattering angle. Therefore the scattering angle, in radians, is the arccosine of the x component of $\hat{p}$.

⇒ Use the `acos()` function to get the scattering angle (in radians). For example, if your unit vector is called `phat`, you could write:

```
theta = acos(phat.x)
```

Calculate and print the scattering angle, converted to degrees, at the end of your program, outside of your calculation loop.

⇒ Determine the scattering angle of the alpha particle for the following impact parameters, and record these values:

$$b_1 = 5 \times 10^{-15} \text{ m}$$
$$b_2 = 10 \times 10^{-15} \text{ m}$$
$$b_3 = 100 \times 10^{-15} \text{ m}$$

⇒ Find and record values of the impact parameter b that result in these scattering angles:

$$\theta = 45^\circ$$
$$\theta = 90^\circ$$
$$\theta = 135^\circ$$