Physics Text Game

# Goal

To create a simple projectile game involving integration of Newton’s laws of motion, without worrying about graphics. You will use the Vec2 class that I am providing, and you will create a file for a Particle class and another file to hold your main program.

## Particle class

This class will be the base one for all our physics objects going forward.

The constructor will require the mass (scalar), position (Vec2), and velocity (Vec2). Give these all default values, mass = infinity, and position and velocity the zero vector. The constructor should store these as object variables with self. It should also create a new object variable that is the net force accumulated on the particle and initialize this as the zero vector.

The particle class should contain the following other functions:

* clear\_force, which resets the force to zero vector.
* add\_force, which accumulates a new force into the net force vector.
* update, which projects forward the motion of the particle in time by a provided timestep
  + Specifically, using the Euler-Cromer method:

## Main program

In another file, write your main program. The scenario is that you are manning a mortar, and you need to target an enemy. The enemy will be between 100 m and 2.0 km away and on flat ground between 100 m lower and 100 m higher than your mortar.

The game will report the enemy’s location and wind velocity. The game is to see how many shots it takes to hit the enemy. A hit occurs if your shell lands within 10 m of the target. For each shot, you choose the angle of launch from the vertical (between 0° and 45°; 0° is straight up), and the initial speed is set by how many augmentation charges you add to the mortar shell. See the following table (taken from [Wikipedia](https://en.wikipedia.org/wiki/Mortar_(weapon)) for the 81 mm L16 mortar):

|  |  |
| --- | --- |
| Number of Augmentation Charges | Muzzle Speed (m/s) |
| 0 | 73 |
| 1 | 110 |
| 2 | 137 |
| 3 | 162 |
| 4 | 195 |
| 5 | 224 |
| 6 | 250 |

Once the initial velocity is set, the mortar (4.3 kg) is launched. Its position and velocity will be updated with a timestep of 0.01 seconds. On each timestep, the force on the mortar is cleared, then the following forces are added:

* gravity, .
* air resistance, , where , where **v** is the shell’s velocity and **w** is the wind velocity and c = 0.000126. Wind should be set randomly at the beginning of the game from a distribution that includes both positive and negative x coordinates and realistic wind speeds.

Don’t print the position at each timestep. Just print where it impacts the ground.

If a hit does not occur, tally the try and give another chance with the same enemy location and wind.

# Objectives

Objectives in \***bold** are **primary**. The others are secondary. Recall that primary objectives will be crucial for later projects, but secondary ones are mostly important for the present project. After the due date, only primary objectives can be fixed to increase your score.

1. **\*(4) Particle class with its constructor includes all variables with defaults and assigns them to object variables.**
2. **\*(4) Force variable implemented with add\_force and clear\_force functions.**
3. **\*(4) Particle class update function updates velocity and position appropriately.**
4. (2) Game sets up random enemy position and reports it.
5. (2) Game sets up random wind velocity and reports it.
6. (4) Ask for number of charges and launch angle and set initial velocity of the mortar shell.
7. **\*(3) Game loop clears force, adds forces, and updates.**
8. **\*(1) Gravitational force implemented.**
9. **\*(2) Air resistance implemented.**
10. (2) Check if mortar shell hits the ground.
11. (2) Check if the enemy was hit. Report the result and go back to launch again or report how many shots were needed to get a hit.