Project B

Two for the price of one

Official release: March 9th at 1pm
Due March 25th at 6pm.

In this project, you will be actually solving two different problems. The first is a physics-based problem; the second involves reading and writing encoded messages. In both cases, the problems are much more open-ended then you have had in the past. Neither is very long. For example, my solution to the first one is 65 lines long and has 2 functions and a main.

Part I – Cannons for Junkyard Wars (70% -- 60% for code, 10% for questions/graphs)

You are working on a project for *Junkyard Wars* (http://en.wikipedia.org/wiki/Scrapheap_Challenge) and have been asked to figure how to shoot a ball as far as possible from a cannon. The ball is always fired with an initial velocity of 500 m/s (about 1.5 times the speed of sound). Further, as the initial velocity can't be changed, they'd like you to figure out how to shoot targets at different ranges by only changing the angle of fire from the cannon. You are firing a ball about the size of a baseball but it weighs 1 kg (about 8 times that of a baseball). This is being done at sea-level and the land is very, very flat. It turns out that solving the problem by hand is very difficult (due to air resistance varying with the current speed of the ball). So, as the most expert programmer on the team, you've been asked to make a "targeting plan". Thankfully, other people on the team can explain the physics to you (found below).

Figure 1: Cannon firing at an angle

Background

The forces acting on the ball in the "y" direction (up and down) are: $9.8kg*m/s^2$ in the downward direction (due to gravity) and $D*v_y^2$ in the opposite direction of travel air resistance:. In the "x" direction the only force acting on the ball is air resistance (so $D*v_x^2$) in the opposite direction of travel. That is:

$$F_y = -9.8kg*m/s^2 \pm D*v_y^2 \quad \text{and}$$
$$F_x = \pm D*v_x^2$$

Where the ± is determined by the direction of the flight and D is a constant. Graphically, the forces look something like this:

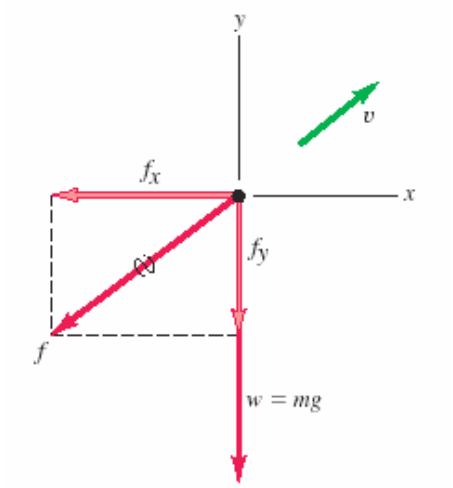

Figure 2: Direction of forces

So the secret to doing this is to model the position and velocity of the ball. You will need keep this information for both the x and y directions. Then what you will do is model a short period of time passing (called a timestep). You change the position based upon the current velocity and the size of the timestep. And you change the velocity based on the forces acting on the ball (drag and gravity). Let's do a quick example.

Assume the angle of fire is 45 degrees above the x axis. This provides us with an initial x velocity of 500m/s*cos(45 degrees) and an initial y velocity of 500m/s * sin(45 degrees). [Don't forget that in C++ the angles passed into sin() and cos() must be in radians.]. The initial position is x=0 and y=0.
So we get

v_x=353.55 m/s; v_y=353.55 m/s; p_x=0; p_y=0 (all at time 0)

Say our time step is 0.01 seconds. The p_x will change by 0.01s*v_x or 3.53 meters. The value of p_y will change by the same amount. The velocity in the x direction will drop by D*v^2 times the time step, where D for this problem is 0.00126 kg/m (see below for where that number came from, but for the entire problem you can just use that value of D). So the change in v_x is (0.00126 kg/m*(353.5 m/s)^2 * 0.01s)/1kg or 1.575m/s. For the y velocity there is also a gravitational component that produces an acceleration of 9.8m/s^2. Recall that v=a*t+v_0. In our case that means the y velocity will be further reduced by 9.8m/s^2*0.01s or .098 m/s. So we get: (you may get *slightly* different answers due to rounding!)

v_x=351.98 m/s; v_y=351.88 m/s; p_x=3.53; p_y=3.53 (all at time 0.01)

Your program then needs to do the same type of computation until the ball's y position is again at (or slightly below) 0. That is, it has hit ground.

Specification

Your job is to write a program in C++ that determines where the ball will hit the ground for all angles for all angles from 90 degrees (straight up) to 1 degree (almost purely horizontal) in 1-degree increments. Your program should print its output in two columns. The first should be the angle of fire (in degrees) the second should be the distance traveled before hitting the ground when fired at that angle. Thus the output might look something like this:

```
90  0.00001
89  260.901
88  450.000
```

etc.

You are to then use Matlab to generate a graph of this data (see below under "Questions").

(Note: these numbers are not necessarily correct!)

Your code must be well-written, broken into reasonable functions, and have reasonable comments. See below for directions on turning in your code.

Additional Background:

The value D is a constant that comes from the size of the ball, the density of air, and properties of the surface of the ball. For our ball, D was calculated to be approximately 0.00126 kg/m (assuming we are at sea-level). An ideally smooth ball would have about a fifth that drag. You can learn more about the topic (and get some hints about coding the problem!) from: http://wps.aw.com/wps/media/objects/877/898586/topics/topic01.pdf. I also took one of the diagrams from that site.

Questions

You are to turn in a paper copy answering the following questions. The work should be turned in to the 101 box or in class on Friday March 25th. *Your answers should be stapled and have your lab section number at the top!*. You will need to make modifications to your program to be able to answer these questions, but we want you to turn in a program which does the task described above (so we suggest you make a copy and modify it.)

- Use Matlab to graph the data generated from your program and hand in a printout of that graph. If your executable is named a.out, you would do the following to get your output sent to a file named "data". From there you need to figure out how to read the data into Matlab from a text file.

  ```
  a.out > data
  ```

  From there you need to figure out how to read the data into Matlab from a text file.
- At what angle (in degrees, to 2 significant digits past the decimal place) should the cannon be fired to send the ball 50 meters in the x direction?
- At what angle (in degrees, to 2 significant digits past the decimal place), should the cannon be fired to get the ball to go as far as possible?
- If there were no air resistance how far would the ball go if fired at a 45 degree angle? How far does it go with air resistance?
- For a 45 degree angle shot, how much does your total x distance change if you use a timestep of 0.01 seconds vs. 0.001 seconds (with air resistance)?

T*he following questions are* ***just for fun.*** *In other words,* ***you don't need to do them****:*

- At what angle would you need to fire the cannon to get the ball to land *on top of* an object 200 meters high and 200 meters away (in the x direction)?
- At what angle would you need to fire the cannon to get it to *hit the bottom of* a target 200 meters high and 200 meters away (in the x direction)?
- What is the air speed velocity of an unladen swallow? (try Google)

Part II—I come to bury Caesar (30%)

This problem involves a simple decryption algorithm to enable you to decode messages and make them readable to people who are supposed to see what they contain. You will be supplied with a file that has encrypted message. You will need to write a C++ program to decode the messages and print them out in plain English. This will require skills in manipulating characters, the use of functions, and file I/O.

Background

Your algorithm will be based on a variation of a Caesar cipher, which consists of rotating each letter of the alphabet by a certain shift amount. For example, if the shift amount is 3, A becomes D, B becomes E, and so on:

HELLOYOU becomes KHOORBRX

Notice that the letter Z would wrap around to C using this process. Your task will be just the opposite for decryption (or decoding). So, if you see the letter D in the file, and the current shift amount is "down 3", you would display the letter A as the output.

Specification

The user is to provide a file name as an argument to the program. That file is to be opened and the translation is to be printed to the screen.

Special Characters: % @
If you see a number in the file, immediately preceded by the special symbol % or @, it means you are to decrypt letters from this point on using a shift equal to this number, but don't display the numbers.

%2 means shift upwards by 2 for decryption, but don't display the number 2.

@25 means shift downwards by 25 for decryption, but don't display the number 25.

Examples:
The input %2ZWC is displayed as BYE.
The input @25GH is displayed as HI.

The input file will always start with a % or a @. You will use that shift until the end of the file. All symbols other than uppercase letters should be printed as they appear in the file. The file will not contain the symbols % and @ anyplace other than as the first character in the file.

More examples

@22KLAJ PDA LKZ ZKKN DWH. OKNNU, ZWRA. 2001.

should be decoded as

OPEN THE POD DOOR HAL. SORRY, DAVE. 2001.

and

@9FN WNNM J KRPPNA KXJC

should be decoded as

WE NEED A BIGGER BOAT

Hints

You should consider how ASCII codes work. Note that:

```
#include<iostream>
using namespace std;

main()
{
        char bob='A'+3;
        cout << bob << endl;
}
```

will result in "D" being printed out. Also, keep the mod operator (%) in mind.

Turning in your project

You will use a directory named "PB" for this project, in the same place your other project directories have been placed.

For part I your C++ program should be named "part1.cc"
For part II your C++ program should be named "part2.cc"
Directions for turning in the paper portion are found above.

Credits

I took the first image off the web, but I can't figure out from where I got it. ☹
As noted, figure 2 is from http://wps.aw.com/wps/media/objects/877/898586/topics/topic01.pdf
Part II is a modified version of an assignment written by Dr. Mary Lou Dorf.