Contents

Version 0.2

Updated the sequence of events.

Input in Lab 2

Input in lab 2 involves reading in a ball. Lab 4 may involve more input to make the simulation a playable game.

Reading in a ball

At any point in time in lab 2, if the code tries to read in a ball and cannot, the simulation is over and the input loop should return.

The code should use scanf to read in the ball. Ball data is an unsigned char (given in hex) for the bit field followed by 4 doubles in X, Y, theta, force order. All are whitespace separated.

- %hhx is the format specifier needed to read in the bitfield
- %lf is the format specifier needed to read in a double

Put a single space between your specifiers in you format string for easy readability. Put all five items in your scanf call. Be sure to pay attention to the return value provided by scanf – it will tell you how many conversions were made. Any number other than 5 is a failure and the simulation can terminate.

Various input files will be provided that you can look at.

Here is a valid line of input:

```
F8      -5.0    0.0     45      75
```

This ball will start at the bottom of the table, five inches left of the centerline. It is aimed at 45 degrees (up and to the right). It has an initial launch speed of 75 inches per second.

This sequence changed since we no longer have a ball launcher to deal with.

If all five data items are read in, the ball can be ~~placed on the launcher~~ put into play after we make some adjustments to the data.

- Print the loaded message (details in the output section)
- Change the status of the ball to be in play (see bit field discussion in data section)
- Convert the polar coordinate numbers to rectangular coordinates (probably owned by physics)
- Print the launched message (details in output section)

Once those things are done, pass both items (the bit field and the array) to the simulation code to run it.

Separation of concerns

You need a file for most of your input. At a minimum it contains:

- The function that calls scanf to read in a ball
- Any future input-related routines will go here (future labs)

A better set of initial test data

Here are the contents of the test file xperfect.pbd:

```
FF      -0.4    0.0     24.678658542539903      7.043309389123606
```

Recall that the last two items are in polar coordinates and will get converted to rectangular coordinates when it is placed on the launcher. The above numbers should yield:

- VX = 6.4
- VY = 47.052534795 / 16.0 = 2.9407834246875

The second one is “GRAVITY / 16.0”. Those numbers were carefully picked to make important points on the parabola line up perfectly.

Other test data

Look for a file called “x3walls.pbd” that bounces off all three walls.

There will also be data files that are for error checking.