Contents

Version 0.2

Simplified off the table.

V0.2: What main owns is filled out.

Lab 2 Simulation Steps

This section give details on the process of running the simulation. The lowest level detail will be in the other sections. The intent here is to stitch it all together.

High-Level Sequence

Initializations

Main owns the job of timing how long the simulation takes to run. Main has to own this detail to make sure that timing starts as soon as possible and completes at the last possible instant. Main also is responsible for kicking off any other initializations (notably graphics) and making sure that the simulation can be run. If all is well it kicks off the input loop but lab2.c shouldn't own the input loop.

The input loop

The input loop reads and runs the simulation on each ball. Input doesn't own the code that runs the simulation.

Simulating one ball

This code lives in lab2, not in physics.

The clock begins at 0.0 with each new ball. The code starts out by launching the ball by changing its status to be on the table.

And then, so long as the ball is on the table:

1. Output the ball (see output)
2. Update the clock (add the time step to the value)
3. Update the ball floating point data (see physics)

Once the ball leaves the table, ~~change the status to off the table and~~ do final output on the ball.

What main owns

The main function will do performance timing. See the example main below. The very first line of code calls now() to get the current time of day with high precision. The very last thing done is another now call and some math, followed by a print. Between them is all the rest of the code for main.

```
int main()
{
        double start, runtime;

        start = now();  // this is the very first executable statement

        /* your code goes here */

        /* at this point we are done, graphics has been torn down*/
        runtime = now() - start;
        /* after graphics has been torn down, we can freely print */
        printf("Total runtime is %.9lf seconds\n", runtime);

        return EXIT_SUCCESS;
}
```

The now function is in n2.o, available on piazza. The header file is n2.h, also available on piazza. If you do not have graphics support, your makefile might resemble the following:

```
lab2: lab2.o bits.o input.o output.o physics.o n2.o
        gcc -g -o $@ $^  -lm
```

If your lab includes graphics, you will need to reference the pinball library, which include the code in n2.o in it.

```
lab2: lab2.o bits.o input.o output.o physics.o
        gcc -g -o $@ $^  -L. -lpb -lncurses -lm
```