Solving Simple Problems in C

Lab 5

Section M

Submitted by:

Haadi-Mohammad Majeed

Submission Date

19/10/2018

Date

15/10/18

Lab Problem

The objective of lab 5 was to create a program that would work with the Dual Shock 4 controllers to determine when it is falling, and landing, whilst calculating the overall displacement with drag taken into consideration. Subsequently, to calculate it, the time was determined for air time, and a percent error was also outputted.

Analysis

Looking at part 1, the lab wants code that checks if the controller is stationary or not, and wants a specific output based on that until it detects that it has begun falling then it wants a set type of output with that every few inputs/milliseconds until it lands with a sudden stop. Both would have to be within some sort of loop as to check/run systematically. At the end of the fall, the programme should output the distance it fell and how long it took to fall that distance.

Part 2 if a continuation of part 1, using the structure previously built, it wants a better calculation of how much it has fallen, thus it considers of drag here. It calculates it from within the loop this time while it cycles, instead of doing everything outside the loops.

Design

The initial plan for the lab consisted of the creation of 2 core body loops for sensing when the remote is at rest, and then the instant it begins to fall it changes to the second loop that analyses that data to determine how far it has fallen and the velocity along with the time it took for the controller to fall said distance calculated. The second part of the lab had a modification to the second loop as to calculate the distance regarding time and drag force instead of just time, as such the programme outputted a more precise distance it had fallen and outputs a percentage for error based on the calculated vs the actual. I decided to roll with a do-while loop for both loops so that the code runs a post-check and not a pre-check, this way I can ensure that the body runs at least once and then checks after data has been imported into the system. The algorithms for calculating said outputs were given, so the logic was the only component absent.

Testing

The lab supplied 3 samples of data, and the programme was tested extensively against each of them to ensure accuracy and precision. The first few tests had outputs that were completely wrong as it was caused by a logistical error where I had initialized variable incorrectly or just forgetting to re-assign them properly. The next few runs had numbers that were slightly off compared to my peers, and this too was caused by a small logic error, as this time I had not properly assigned the initial time correctly and the first instance was calculating

with an initial time of 0. After some more hotfixes the programme was outputting the correct calculated values and was compared to peer's programme's outputs.

Comments

My code had a few errors near the end, all such being small common mistakes such as using %d for a double when I should be using %lf in a scanf or printf statement. Another issue I had was not properly assigning velocity in the beginning before it started falling, causing my values to all be off slightly. These logic errors didn't prevent the code from compiling however caused incorrect outputs from the programme. Making sure I go back to check datatype configuration to variables will be a key thing I need to work on.

Implementation

```c
/*-------------------------------------------------------------------------
-                           SE/CprE 185 Lab 04
-             Developed for 185-Rursch by T.Tran and K.Wang
-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------
-                               Includes
-------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>


/*-------------------------------------------------------------------------
-                               Defines
-------------------------------------------------------------------------*/
const int lineSkip = 10;
const double GRAVITY = 9.8;


/*-------------------------------------------------------------------------
-                               Prototypes
-------------------------------------------------------------------------*/
int close_to(double tolerance, double point, double value) {
    if (value > (point + tolerance) || value < (point - tolerance)) {
        return 0;
    }
    else {
        return 1;
```

```
    }
}

double mag(double x, double y, double z) {
    return sqrt((pow(x, 2.0)) + (pow(y, 2.0)) + (pow(z, 2.0)));
}

/*--------------------------------------------------------------------------
-                            Implementation
--------------------------------------------------------------------------*/
int main(void) {
    int lineCount;
    double t, runOnce, startTime, endTime, preTime;
    double gx, gy, gz, distFallen, timeTotal, v, x;
    t=0;
    v = 0;
    runOnce= 0;
    lineCount = 0;
    printf("Haadi-Mohammad Majeed\nhmajeed\n"); //initial obligatory statement
    do
    {
        scanf("%lf, %lf, %lf, %lf", &t, &gx, &gy, &gz);

        if(runOnce == 0)
        {
            printf("Okay, I'm now recieveing data.\nI'm waiting ");
            runOnce++;
        }
        if(lineCount%lineSkip == 0)
        {
            printf(". ");
        }
        lineCount++;
        fflush(stdout);
    } while((close_to(0.2, mag(gx, gy, gz), 0.0) == 0));
    preTime = t * .001;
    startTime = t;
    runOnce = 0;
    do
    {

        scanf("%lf, %lf, %lf, %lf", &t, &gx, &gy, &gz);
        v = v + 9.8 * (1 - mag(gx, gy, gz)) * (t * .001 - preTime);
        x = x + v * (t * .001 - preTime);
```

```
        preTime = t * .001;

        if(runOnce == 0)
        {
            printf("I'm falling");
            runOnce++;
        }
        if(lineCount%lineSkip == 0)
        {
            printf("!");
        }

        lineCount++;
        fflush(stdout);
    } while(mag(gx, gy, gz) < 2);

    endTime = t;
    timeTotal = (endTime - startTime)/1000.0;
    distFallen = .5 * GRAVITY * timeTotal * timeTotal;

    printf("\nOuch! I fell %lf meters in %lf seconds\n", distFallen , timeTotal);

    int percentage = 100-((x / distFallen) * 100);
    printf("\nCompensating for air resistance, the fall was %.3lf meters\n", x);
    printf("This is %d%% less than computed before.", percentage);

}
```

Pre-lab

1. Do the same drop 5 times in the classroom and record the distances. How consistent are your results? What could cause any variation?

There was small variations between each one but that was due to inconsistencies from the height each one was dropped at, the errors were caused via human inputs, not computational.

2. Run your program with sample data from a previously recorded drop by using the following command:

./DS4Drop < lab5_sampledata2013_1.csv

How far is it from the third-floor railing to the bottom floor according to your code,

using the sample data?

Sample data 1 says it fell about 8 metres which seems a bit shorter than the actual height, if I had to guess from the comfort of my own room, it was about the distance from the second floor railing to the ground floor, so about 5 more metres to get to the third floor railing would be my estimate.

3. In your report, include a graph of the magnitude of the acceleration as a function of time from the sample data. Label where the freefall is happening, where it hits the ground, and your tolerances. Explain and justify any tolerances you are using.

The tolerance I used was 0.2 as it was roughly the difference between each segment to the next, it was enough to not false detect, but not too much to not go off accurately.