DS4 Drop

LAB 6

SECTION 2

Gavin Monroe

SUBMISSION DATE:

10/4/17

Problem

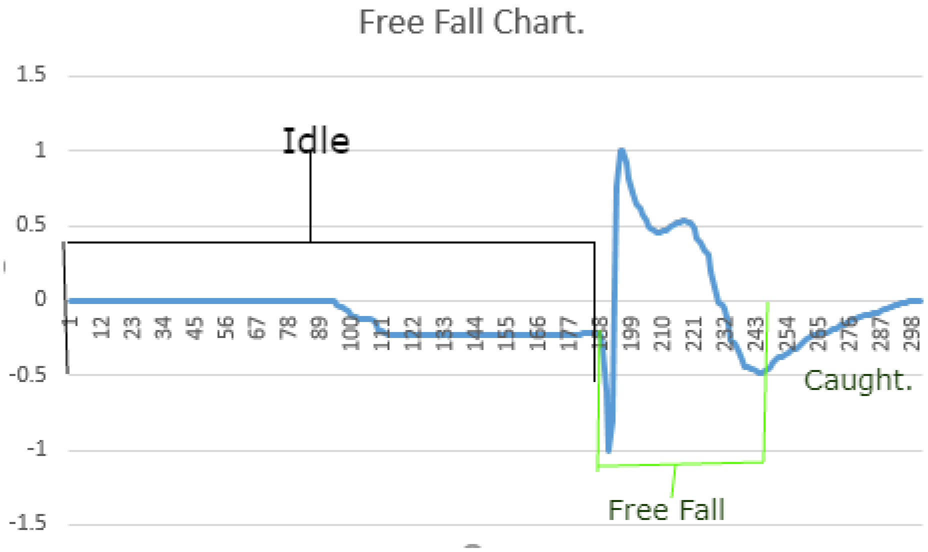
Problem 1: Do the same drop 5 times in the classroom and record the distances. How consistent are your results? What could cause any variation?

We were pretty consistent with our drops, except for maybe 2 of the 5. Each output was around 800 to 900 column entries. The variation would have been caused by me who probably started the program too early or too late and stopping with the same problem. I fixed this issue by re-recording the output while my partner dropped the controller when I said go.

Problem 2: How far is it from the third floor railing to the bottom floor according to your code, using the sample data? The sample data is from a previous drop from the third floor.

```
        $ ./lab6 < sF.csv
Ok, I'm now receiving data.
I'm Waiting

Help me! I'm falling!


Ouch! I fell 39.514 meters in 4.032 seconds.

39.5 meters
```

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



Time is in Milliseconds. And Acceleration is in gravity. It never hit the ground because we caught it. Our tolerance was simple since all we needed to check if it was greater than 0.03.

Problem 4: Demo your source code to an undergraduate TA using our 3 story drop data set and have them enter your demo score into the “Lab 06 Demo” column in Blackboard. Be sure your comment your code and functions that you write.

Completed and Also did extra credit.

Analysis

Physics was a big piece of the puzzle so knowing gravity’s acceleration along with the distance equation, I could find the distance. Of course I needed to do a little more work with detecting the falling motion but after using the mag function I was fine. Then I worked that into my tolerance to detect the falling motion.

Design

I tried to make it easy for myself by using some old but good code that I made last lab. I had to re-work some of it but it worked nice. I used in total around 4 functions, with each having their own purpose of course. I also only used one loop.

Testing

When testing the code I really had to go step by step to keep everything in order. I ran into some problems with time keeping but I fixed that by having some global vars.

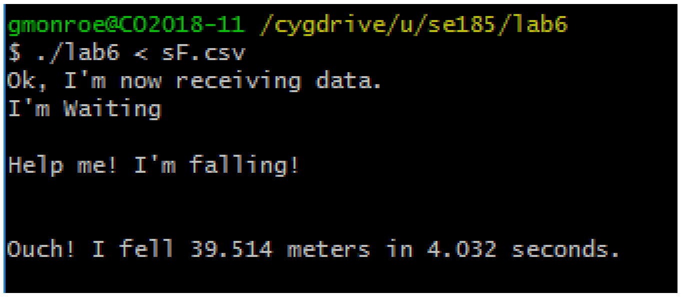
Comments

I had so much fun doing this!

Source Code

```
/* SE 185 Lab 5 Wrapper Program
Gavin Monroe
Fletcher Smith
*/

#include <stdio.h>
#include <math.h>

#define TRUE 1

/* Put your lab 4 functions prototypes here, as well as the prototype for lab 5 */
double mag(double x2, double y2, double z2);
void findFall(double ayInput);
void outputInfo(int oldd, int neww, double tm, double aTime);
void calcTime(int tm, int on);
void calcDistance(int tm, int on);

//Detect Change Vars
int d;
int d2;

//Distance & Time
double dist;
int start = 0;
int beforeTime;
double afterTime;

int main(void) {
        int t, b1, b2, b3, b4;
        double ax, ay, az, gx, gy, gz;
        while (TRUE) {
                fflush(stdout);//Flush Code
                scanf("%d, %lf, %lf, %lf, %lf, %lf, %lf, %d, %d, %d, %d", &t, &ax, &ay, &az, &gx, &gy, &gz, &b1, &b2, &b3, &b4 );//Grab Input


                d2=d;//set old orientation.


  findFall(mag(ax, ay, az));//Find Direction which the controller is facing.
                outputInfo(d2, d, t, afterTime);//Output Result.
  calcTime(t, d);//Output dots
  calcDistance(t, d);//Get Distance


        }
  return 0;
}


double mag(double x2, double y2, double z2){
        /*      CODE SECTION 1 */
        double sqrtX = pow(x2, 2);
        double sqrtY = pow(y2, 2);
        double sqrtZ = pow(z2, 2);
        return sqrt(sqrtX + sqrtY + sqrtZ);
}
void calcTime(int tm, int on){
        int varTime = tm % 100;
 if (varTime == 0 && on == 1){
   printf(".");
 }
 if (varTime == 0 && on == 2){
   printf("!");
 }
}
void calcDistance(int tm, int on){
 if (on == 2){
        double sec = tm - beforeTime;
        sec = sec / 1000;
        afterTime = sec;
        double v = 9.8;
    dist = v * sec;
  }
}
void findFall(double ayInput){
  if (ayInput != 0 && start == 0){
    start = 1;
    d = 1;
  }
  if (ayInput  >= 0.5 && start == 1 && d==1){
    d = 2;
  }
  if (ayInput  < 0.5 && start == 1 && d == 2){
    d = 3;
  }
}
void outputInfo(int oldd, int neww, double tm, double aTime){
        if (oldd!=neww){
                if (neww==1){
                        printf("Ok, I'm now receiving data.\nI'm Waiting");
                }
                if(neww==2){
      beforeTime = tm;
                        printf("\n\nHelp me! I'm falling!\n");
                }
                if(neww==3){
                        printf("\n\nOuch! I fell %3.3lf meters in %3.3lf seconds.\n", dist, aTime);
                }
                if(neww==4){
                        printf("RIGHT\n");
                }
        }
}
```

Screen Shots



```
gmonroe@CO2018-11 /cygdrive/u/se185/lab6
$ ./lab6 < sF.csv
Ok, I'm now receiving data.
I'm Waiting

Help me! I'm falling!


Ouch! I fell 39.514 meters in 4.032 seconds.
```

DS4 Drop Part II

LAB 5

SECTION 2

Gavin Monroe

SUBMISSION DATE:

10/14/17

Problem

Problem 1: How much difference tends to occur in drops in the lab area? From the 2nd floor?

There isn’t a huge difference but I would say there is definitely a difference regarding maybe the accuracy of measurement. The Lab area is more controlled since we are literally the ones who are dropping it, but with the second floor its already set, all the data is there. I would say with the lab area, the distance was a constant small distance, while for the 2nd floor it was a distance set with the csv file.

Problem 2: How far is it from the second floor railing to the bottom floor according to your code?

New Equation:

```
Compensating for air resistance, the fall was 39.514 meters.
```

Old Equation: 0.5 * (v * pow(sec, 2))

```
Compensating for air resistance, the fall was 79.659 meters.
```

With this new equation you can see how much more accurate it is.

Problem 3: What issues arose in implementing Part 2?

I used the “real” equation to find distance for a falling object last lab, so this ;ab I had to go back to the previous lab and used the old equation(0.5 * (v * pow(sec, 2))) to find the difference for each fall difference. I made sure to do some research on finding the difference for two numbers just too make sure I got the right difference. There was two equations I had to use for this new part two lab, and 4 equations altogether. I also ran into numbers not lining up so I went back and rewrite some equations so the compiler could understand it better.

Analysis

I needed to overall look at the old placement to output the new one, I needed to find the overall difference along with calculating the distance with a new equation.

Design

Used 4 simple equations in a total of 4 different functions to help input and display information accordingly.

Testing

In testing I had to find the small mistakes I made and go back and fix them. These mistakes were important to fix due to the accuracy of my numbers. Needed to use the correct numbers with the correct equations.

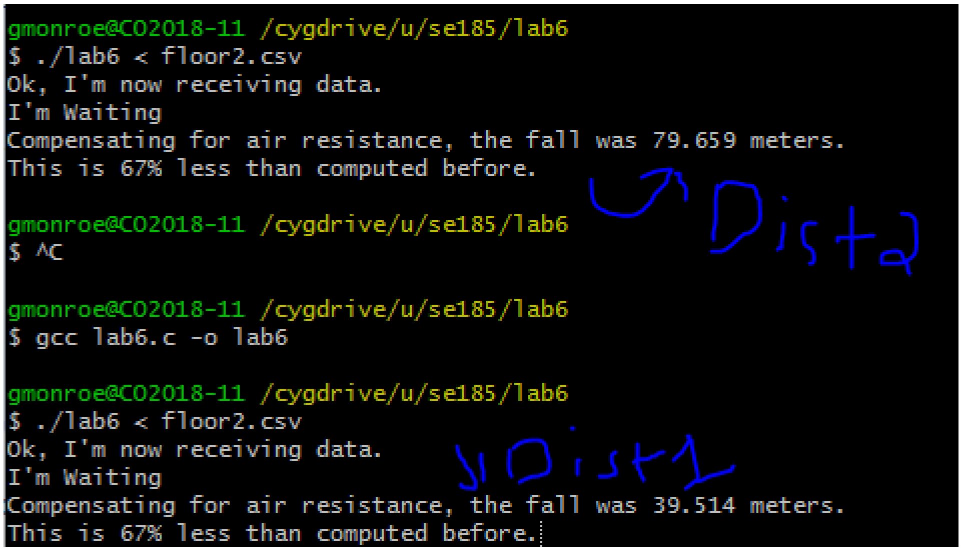
Comments

I loved this challenge just because of the new angle we took on it.

Source Code

```
/* SE 185 Lab 5 Wrapper Program

Gavin Monroe

Fletcher Smith

*/


#include <stdio.h>

#include <math.h>


#define TRUE 1


/* Put your lab 4 functions prototypes here, as well as the prototype for lab 5 */

double mag(double x2, double y2, double z2);

void findFall(double ayInput);

void outputInfo(int oldd, int neww, double tm, double aTime);

void calcTime(int tm, int on);

void calcDistance(int tm, int on);


//Detect Change Vars

int d;

int d2;


//Distance & Time

double dist;

double dist2;

int start = 0;

int beforeTime;

double afterTime;


int main(void) {

        int t;

        double ax, ay, az;

        while (TRUE) {

                fflush(stdout);//Flush Code
                scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az );//Grab Input

                d2=d;//set old orientation.

    findFall(mag(ax, ay, az));//Find Direction which the controller is facing.
                outputInfo(d2, d, t, afterTime);//Output Result.
    calcTime(t, d);//Output dots
    calcDistance(t, d);//Get Distance

        }
    return 0;
}

double mag(double x2, double y2, double z2){
        /*      CODE SECTION 1 */
        double sqrtX = pow(x2, 2);
        double sqrtY = pow(y2, 2);
        double sqrtZ = pow(z2, 2);
        return sqrt(sqrtX + sqrtY + sqrtZ);
}
void calcTime(int tm, int on){
        int varTime = tm % 100;
  if (varTime == 0 && on == 1){
      printf(".");
  }
  if (varTime == 0 && on == 2){
      printf("!");
  }
}
void calcDistance(int tm, int on){
  if (on == 2){
        double sec = tm - beforeTime;
        sec = sec / 1000;
        afterTime = sec;
        double v = 9.8;
    dist = v * sec;
        dist2 = 0.5 * (v * pow(sec, 2));
  }
}
void findFall(double ayInput){
  if (ayInput != 0 && start == 0){
    start = 1;
    d = 1;
  }
  if (ayInput  >= 0.5 && start == 1 && d==1){
    d = 2;
  }
  if (ayInput  < 0.5 && start == 1 && d == 2){
    d = 3;
  }
}
void outputInfo(int oldd, int neww, double tm, double aTime){
        if (oldd!=neww){
                if (neww==1){
                        printf("Ok, I'm now receiving data.\nI'm Waiting");
                }
                if(neww==2){
                        beforeTime = tm;
                        printf("");
                }
                if(neww==3){
                        double dist3 = (dist + dist2) / 2;
                        int perc = ((dist2 - dist) / dist3) * 100;
                        printf("\nCompensating for air resistance, the fall was %3.3lf meters.\nThis is %d%% less than computed before.", dist, perc);
                }
                if(neww==4){
                        printf("RIGHT\n");
                }
        }
}
```

Screen Shots



```
gmonroe@CO2018-11 /cygdrive/u/se185/lab6
$ ./lab6 < floor2.csv
Ok, I'm now receiving data.
I'm Waiting
Compensating for air resistance, the fall was 79.659 meters.
This is 67% less than computed before.

gmonroe@CO2018-11 /cygdrive/u/se185/lab6
$ ^C

gmonroe@CO2018-11 /cygdrive/u/se185/lab6
$ gcc lab6.c -o lab6

gmonroe@CO2018-11 /cygdrive/u/se185/lab6
$ ./lab6 < floor2.csv
Ok, I'm now receiving data.
I'm Waiting
Compensating for air resistance, the fall was 39.514 meters.
This is 67% less than computed before.
```