How do balls bounce?

Abe Karplus

My experiment is studying the way in which a ball bounces and how much different types of balls bounce.

Table of Contents

Theory..... 2

***Why balls bounce*.....2**

***Discrete time*..... 2**

***Computer simulation*..... 2**

***Continuous time*.....3**

***After the bounce*..... 5**

Hypothesis..... 5

Materials and Methods..... 5

***Equipment*..... 5**

***Measurements*..... 6**

Results..... 8

***Plots*.....9**

***Interpretation of Results*.....10**

Acknowledgments..... 10

Appendix A (Data)..... 11

***Ball 1 Data*.....11**

***Ball 2 Data*.....11**

***Ball 4 Data*.....11**

Appendix B (Program)..... 12

***Variables List*..... 12**

Theory

Why balls bounce

The reason balls bounce is that, when they hit the floor, they are compressed slightly, then expand again, pushing against the floor. Newton's third law says that the floor will then push back on the ball, sending it soaring. If a ball is dropped from a greater height, it will have a greater kinetic energy on impact with the floor, causing it to compress more, thus to expand more, and thus go higher.

A coefficient of restitution is the number that the incoming velocity is multiplied by for a bouncing object to get the outgoing velocity. [http://en.wikipedia.org/wiki/Coefficient_of_restitution]

Our final goal is to be able to find the coefficient of restitution, r, for a ball, given the top of one bounce and then the top of the next bounce. We know that acceleration due to gravity is 980.665 cm/sec^2 [http://en.wikipedia.org/wiki/Acceleration_due_to_gravity]. We should be, but we aren't, worrying about air resistance.

Let's compute velocity of a falling ball, V, expressed in cm/sec. The V_0 of a falling ball is its initial velocity at time 0. Assume that $V_0 = 0$. Acceleration due to gravity is expressed in cm/sec^2.

Now, how do we compute position?

Discrete time

In discrete time simulation, what happens is that a ball moves, then its velocity changes. A ball's velocity determines how far it moves in each time step (every Δt seconds). So at time 0, it hasn't had any time to move. During the first time step, its velocity hasn't had a chance to increase. So, at Δt, its velocity is $-g\Delta t$, but its position is still 0. Then, when it moves, $V=-g\Delta t$ and distance already moved = 0, so distance becomes $V\Delta t=-g\Delta t^2$. At $2\Delta t$, velocity changes by $-g\Delta t$. Now $V=-2g\Delta t$ and distance already moved $=-g\Delta t^2$, so current distance is $-3\ g\Delta t^2$. It should be plain to see that at every Δ t, V changes by $-g\Delta t$ and d changes by $V\Delta t$.

Computer simulation

I wrote two computer simulations of a ball bouncing. The simulations used discrete time. One of the simulations was written using the 2D programming language Scratch. It "stamps" the ball's location every Δt seconds. A snapshot taken while it was running is shown on the poster. The other simulation was written using the 3D programming language Alice. Since Alice does not have a stamp function, photos from it would only show a ball at one point, and so were not included.

Bouncing ball simulation in Scratch.

Continuous time

So, what if Δt = 1/infinity? It would be too hard to do computations for distance the discrete way. Now, to compute its velocity at any given time, if we replace the Δ t from the velocity formula with t, we get

$$V_t = -gt$$

If we do not assume that V_0=0, then the formula is a bit more complicated:

$$V_t = V_0 - gt$$

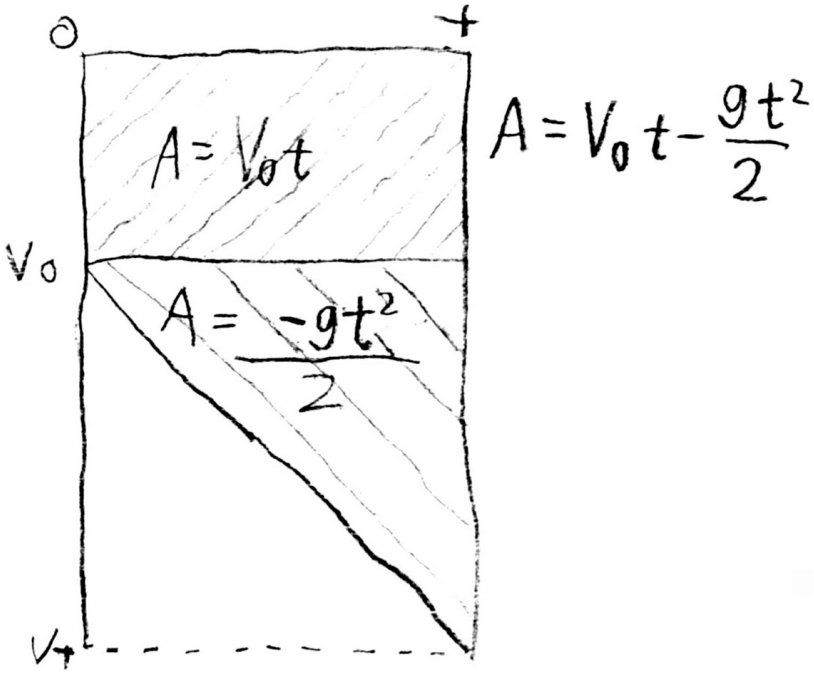
In continuous time, it is much easier to compute position. In discrete time, you had to add up the area of every single rectangle, but here, all you have to do is find the area of the triangle, i.e.,

$$-g\ t^2/2,$$

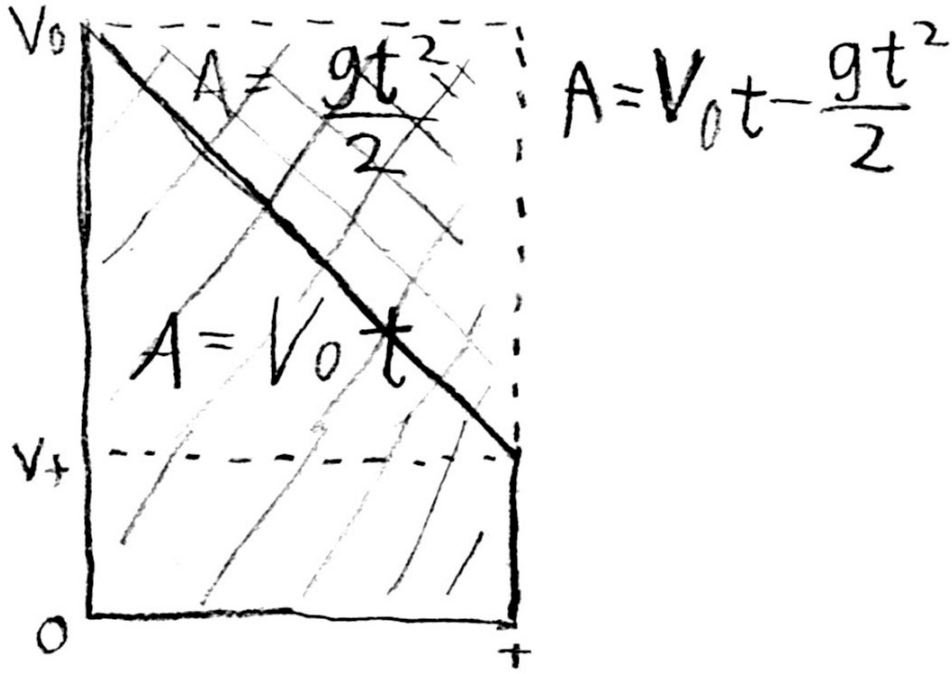
or, if we include V_0, we get a rectangle and a triangle (see drawings for V_0<0 and V_0>0),

$$V_0\ t - g\ t^2$$

Given g, t, and V_0, we can now compute the position and velocity of any falling object (assuming no air resistance).



Velocity as a function of time for $V_0<0$.



Velocity as a function of time for $V_0>0$.

After the bounce

If a ball hits the ground with velocity V_{in}, how high was it dropped from? It took $-V_{in}/g$ seconds to reach this speed, so it must have been dropped from $g\,t^2/2 = V_{in}^2 / (2g)$.

To get the outgoing velocity, multiply the incoming velocity by the coefficient of restitution, r. You also negate it.

$$V_{out} = -r\,V_{in} .$$

Remember how the ball's speed kept increasing while it was falling? Now the velocity will be continually decreasing at the same rate.

The outgoing velocity is its initial velocity, V_{out}, for its upward path. Speed at the top of the bounce is zero, at time $t=V_{out}/g$. So the height of the bounce is $g\,t^2/2 = V_{out}^2/(2g) = r^2\,V_{in}^2\,/(2g)$.

The bounce height divided by the drop height is just r^2.

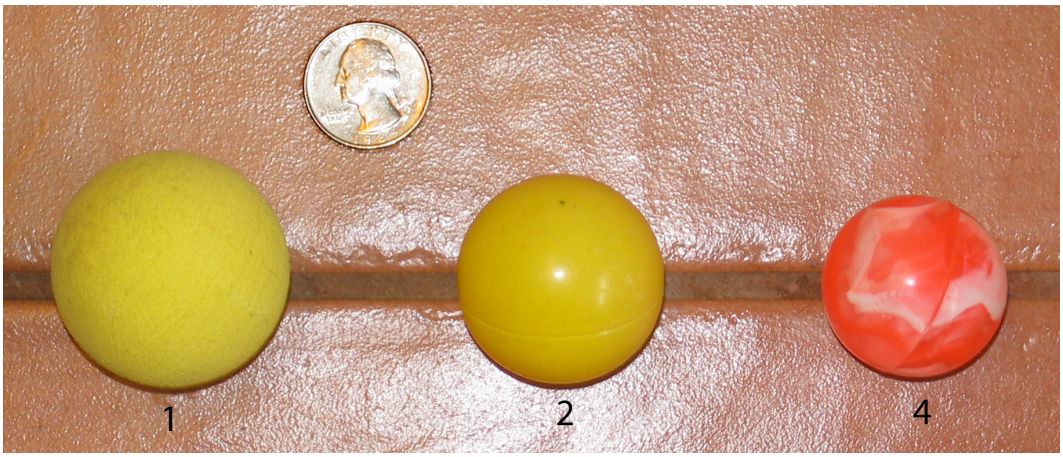
Hypothesis

I predict that a ball will always bounce to the same fraction of its initial drop height, and what fraction it is will be different for each ball.

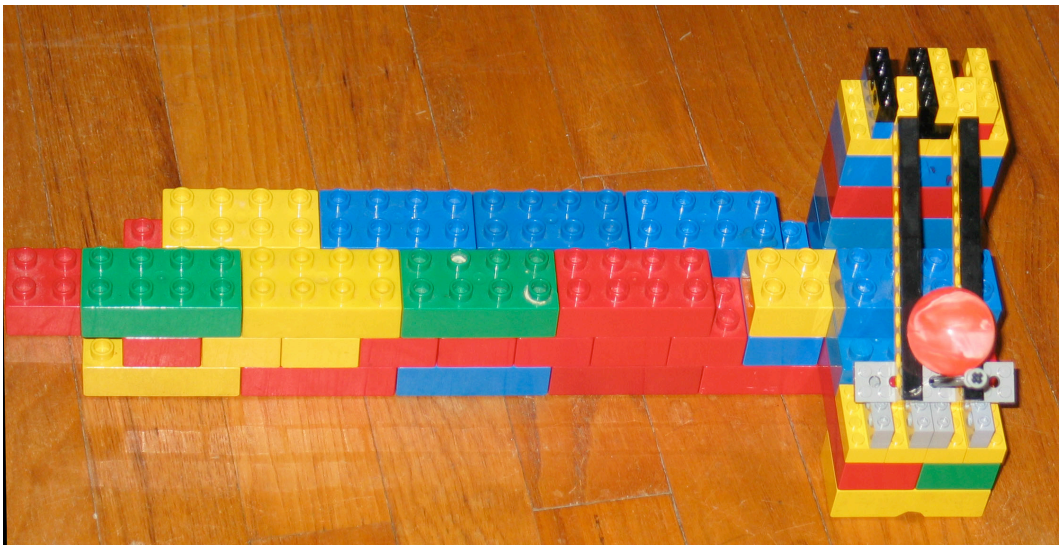
Materials and Methods

Equipment

The equipment I used for my experiment is a digital camera with a long exposure function, a tripod, a black foamcore background with lines marked on it every 20cm, a ball launcher made from Lego and Duplo that allows the balls to be released at always the same height with the same spin, and two lamps because many of the photos were taken at night.



The three balls used. (Ball 3 (another bouncy ball) 's photos were not used due to their blurriness and lack of contrast)



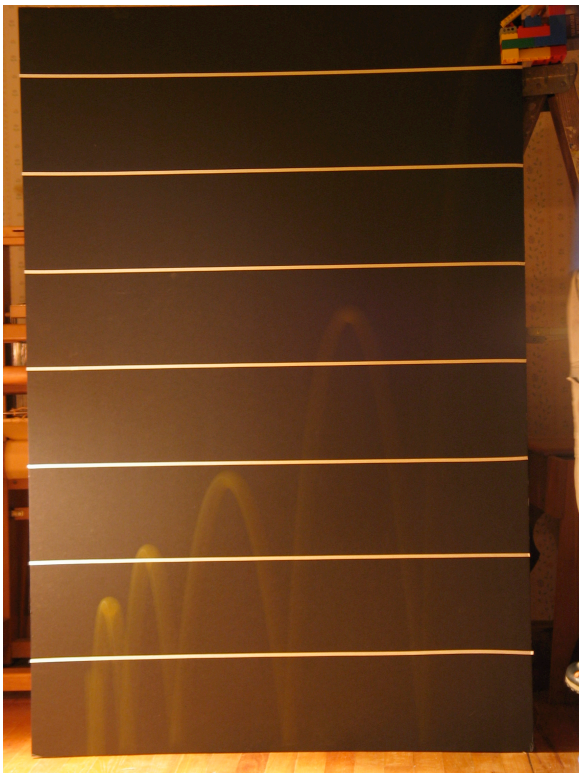
The ball launcher.

Measurements

The two independent variables in my experiment are the height from which the ball is initially dropped and the type of material the ball is made of. The dependent variable is how high it bounces on its first bounce. Usually I also measured the second bounce, third bounce, etc. The controlled variables I attempted to keep constant are the amount of horizontal motion the ball has (it may affect the bounce height), the type of surface onto which I am bouncing the balls, and the amount of spin the balls have, as spin affects their horizontal motion.

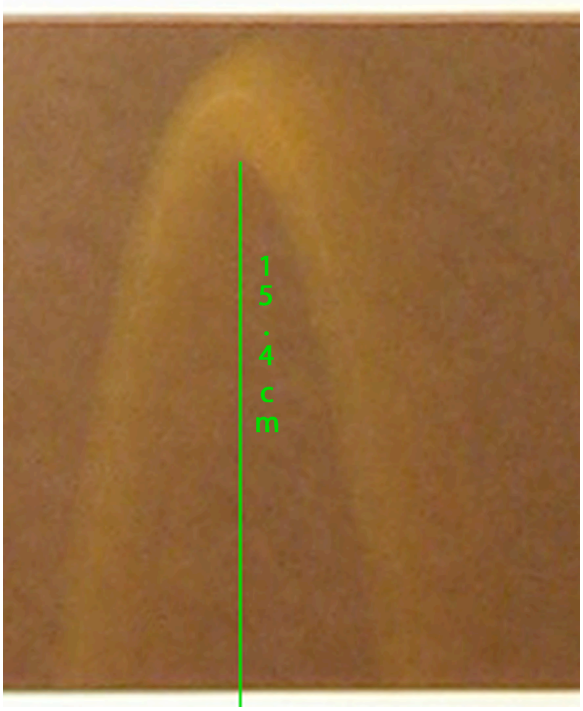
Procedure for taking measurements:

1. Set camera exposure time to 1–4 seconds. Cameraman presses shutter and shouts "go!" Ball-releaser, upon hearing "go!", removes launching pin from the ball launcher, releasing ball.
2. Camera takes photo of ball's trajectory.
3. Photo is downloaded to computer and opened in Photoshop.
4. Photo is cropped around the image of first bounce, so that the top and bottom are in the middle of the lines marked in the backdrop above and below the ball.
5. Photo height is resized to 20cm.
6. Ruler's zero point is set to the bottom of the cropped photo.
7. The height of the bottom of the ball is measured and recorded (adding in 20cm for each extra line below the cropped part).
8. Undo button is pressed twice, to get the original photo back.
9. Steps 4 through 8 are repeated for all legible bounces.



An example photo of Ball #2 bouncing.

An example photo of Ball #4 bouncing.



How the measurements are performed.

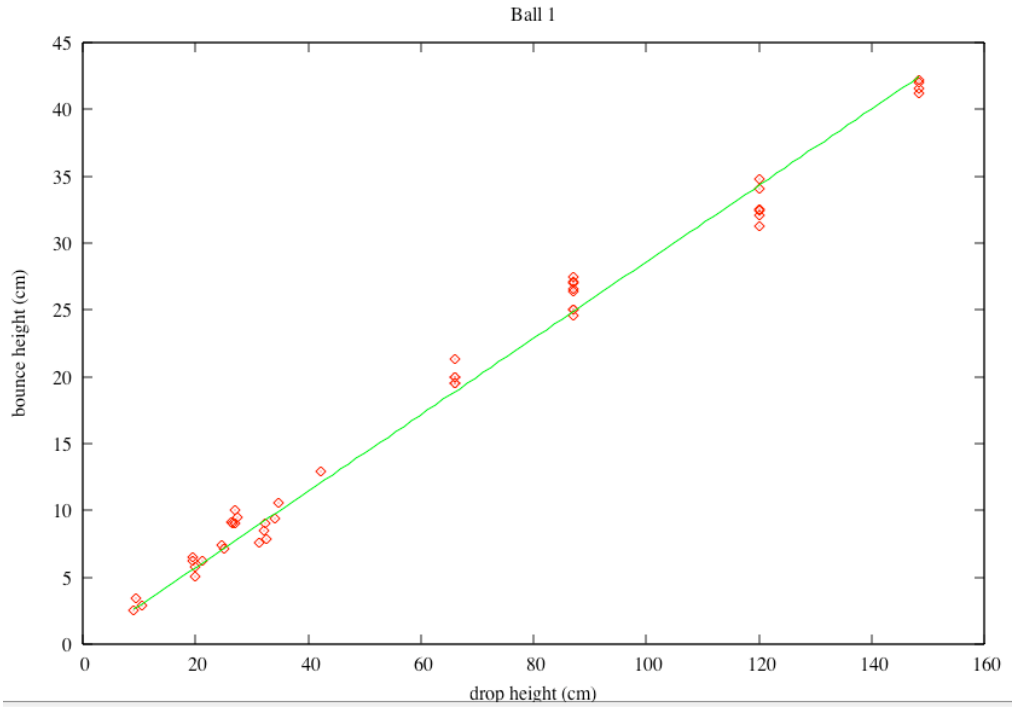
Results

The following plots show bounce height as a function of drop height for balls 1, 2, and 4. The green line represents the best fit for the data using the theory.

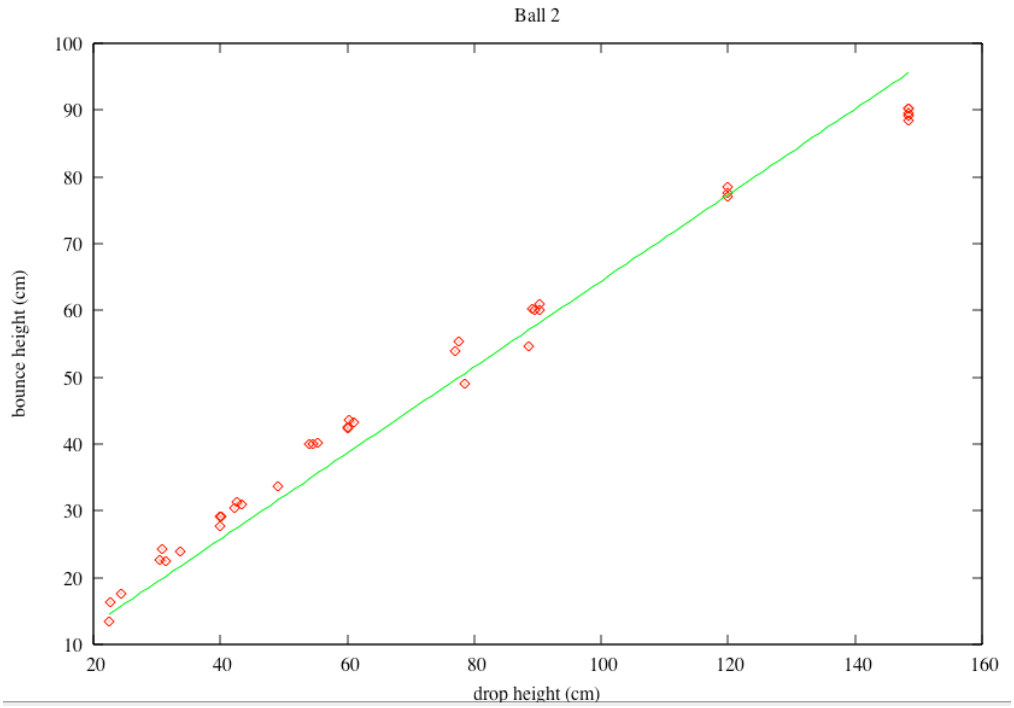
This is the gnuplot script for creating the plots:

```
# plot for ball 1
unset key
set title "Ball 1"
set ylabel "bounce height (cm)"
set xlabel "drop height (cm)"
fit r1*r1*x 'ball1.txt' using 1:2 via r1
plot 'ball1.txt' using 1:2, r1*r1*x
```

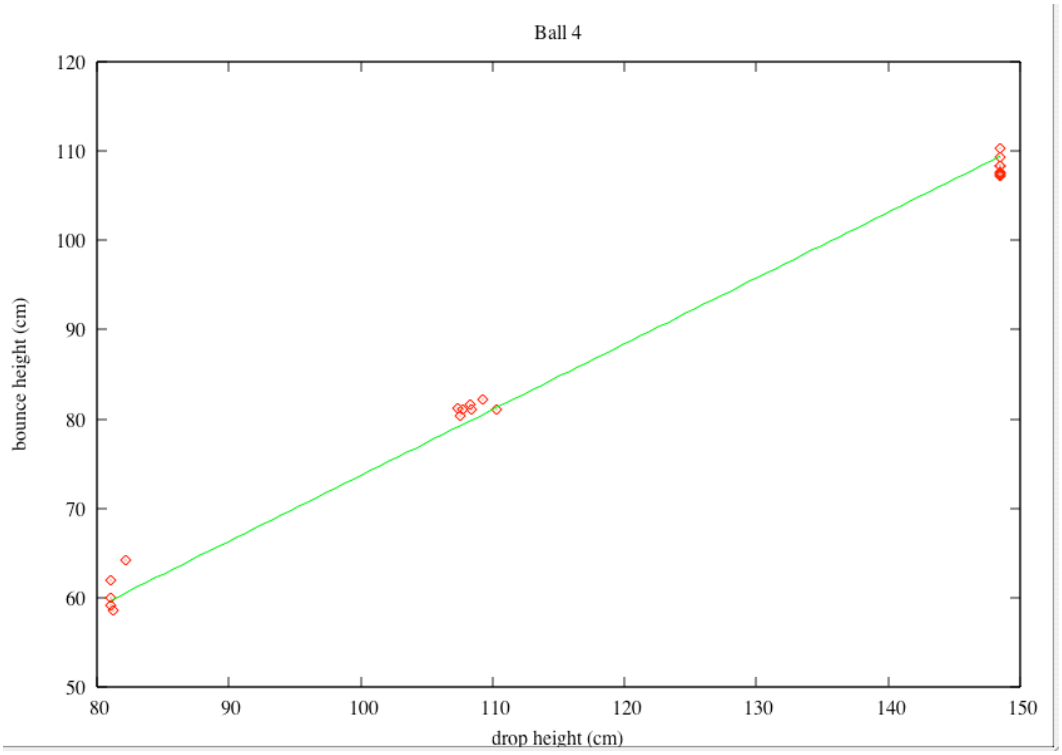
Plots



Ball 1's r value is 0.534713± 0.005



Ball 2's r value is 0.802647± 0.005



Ball 4's r value is 0.858361± 0.005

Interpretation of Results

The straight lines on the graphs are very good fits for the data. This means that a ball always bounces to about the same fraction of its drop height. Also, there are different slopes for different balls. This shows that my hypothesis is correct.

Acknowledgments

I would like to thank my Dad, Kevin, for his help in taking the photos, explaining how to use gnuplot and certain things in Photoshop Elements, teaching me the small amount of calculus I needed for making the conversion from discrete to continuous time, and typing up the poster. I would also like to thank my Mom, Michele, for being a receptive audience for my explanations of the theory of bouncing balls and for helping type the theory.

Appendix A (Data)

Ball 1 Data

#BALL 1 (yellow hard foam ball)

#A	B
120	32.4
120	32.1
120	34.8
120	32.5
120	34.1
120	31.3
32.4	9
32.1	8.5
34.8	10.6
32.5	7.9
34.1	9.4
31.3	7.6
10.6	2.9
87	27.5
87	25
87	24.6
87	26.6
87	27
87	26.4
87	25
87	27.1
27.5	9.5
24.6	7.4
26.6	9
27	10
26.4	9.1
25	7.1
27.1	9
9.5	3.4
9.1	2.5
66	20
66	19.5
66	20
66	19.5
66	21.3
20	5.1
19.5	6.2
20	5.8
19.5	6.5
21.3	6.2
148.5	41.2
148.5	42.2
148.5	42
148.5	41.6
42.2	12.9

Ball 2 Data

Ball 2 (yellow ping pong ball)

#a	b
148.5	89.5
89.5	60.1
60.1	42.3
42.3	30.4
30.4	22.6
22.6	16.3
148.5	90.3
90.3	60.1
60.1	42.6
42.6	31.4
31.4	22.4
22.4	13.5
148.5	90.2
90.2	61
61	43.3
43.3	30.9
30.9	24.3
24.3	17.6
148.5	89.1
89.1	60.3
60.3	43.6
148.5	88.5
88.5	54.6
54.6	40
40	29.1
120	78.5
78.5	49.1
49.1	33.7
33.7	23.9
120	77
77	54
54	40
40	27.8
120	77.6
77.6	55.4
55.4	40.1
40.1	29.2

Ball 4 Data

Ball 4 (orange bouncy ball)

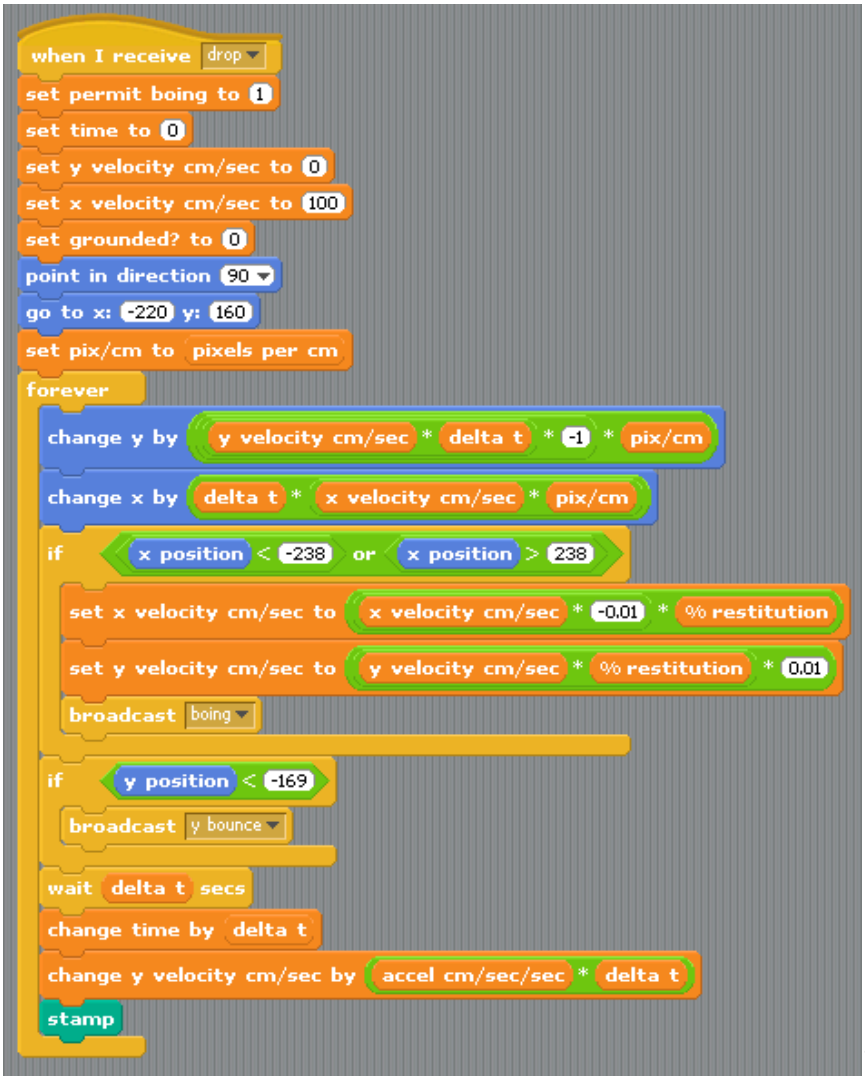
#a	b
148.5	107.5
107.5	80.4
148.5	110.3
110.3	81
81	59.2
148.5	107.7
107.7	81
81	61.9
148.5	107.4
107.4	81.2
81.2	58.6
148.5	108.4
108.4	81
81	60
148.5	108.3
108.3	81.6
148.5	107.2
148.5	109.3
109.3	82.2
82.2	64.2

Appendix B (Program)

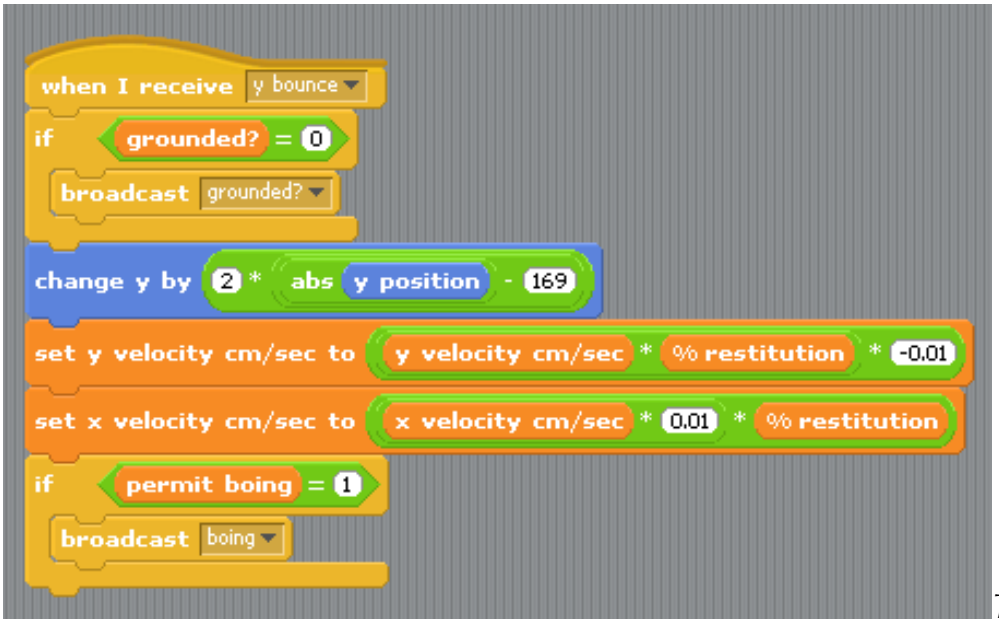
The computer simulation was written in Scratch, a free drag and drop programming language downloadable at: http://llk.media.mit.edu/projects/scratch/download/.

Variables List

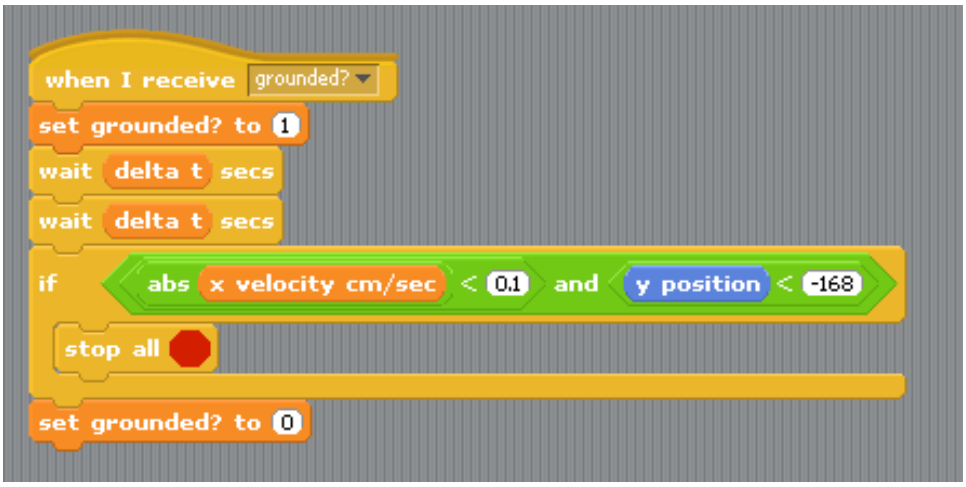
Name	Meaning
% restitution	This is 100*the simulated ball's coefficient of restitution
accel cm/sec/sec	This is acceleration due to gravity
delta t	This is the size of the time step
grounded?	This is a sentinel variable for the grounded script
permit boing	[no longer in use]
pix/cm	This provides a scale by telling the computer how many pixels correspond to a centimeter
pixels per cm	This prevents 'pix/cm' from being changed while the ball is bouncing
time	A record of how long has past since the ball started bouncing
x velocity cm/sec	How fast the ball moves across the screen
y velocity cm/sec	How fast the ball moves up and down



This script controls the falling and x bouncing of the ball.

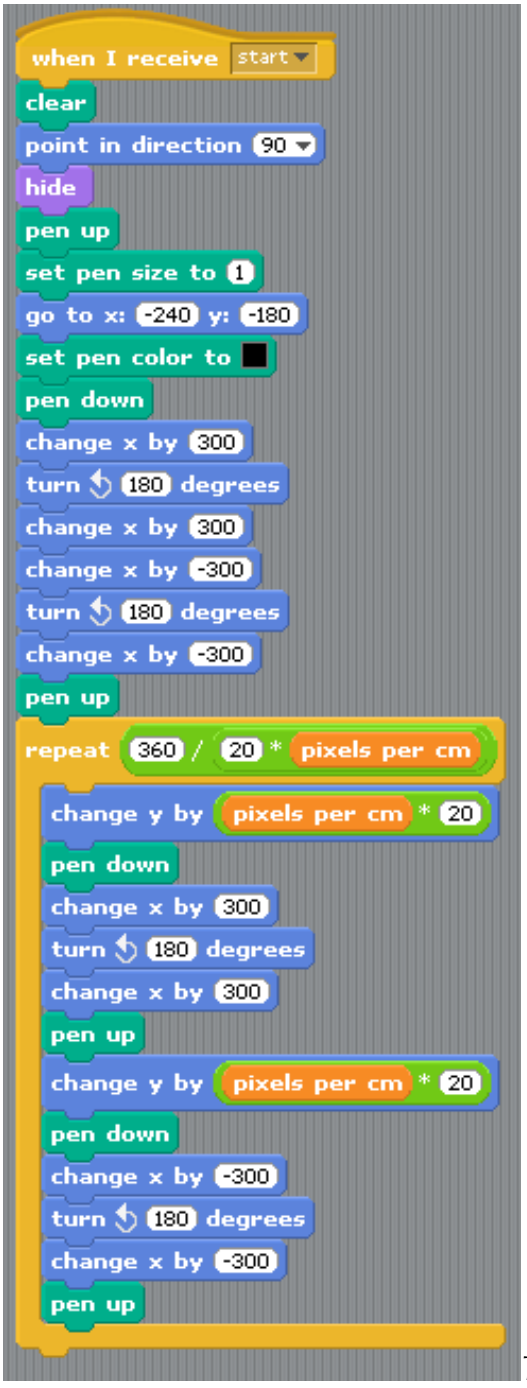


This script controls bouncing off the floor. It multiplies both velocities (x and y) by the coefficient of restitution, as well as negating the y velocity.

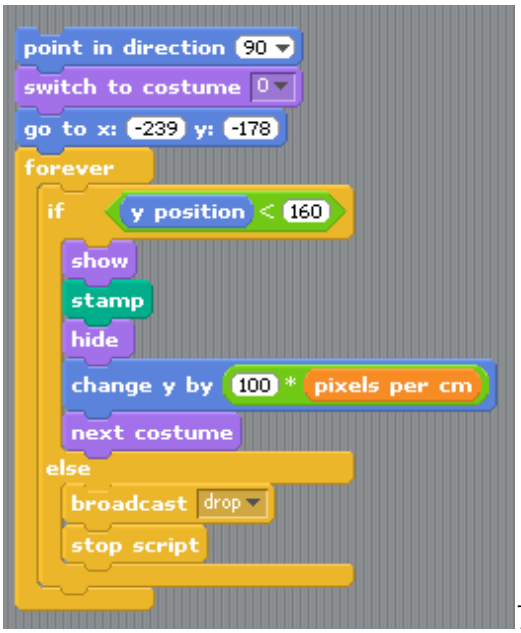


This script checks if the ball has stopped bouncing; if it has, it stops the program.

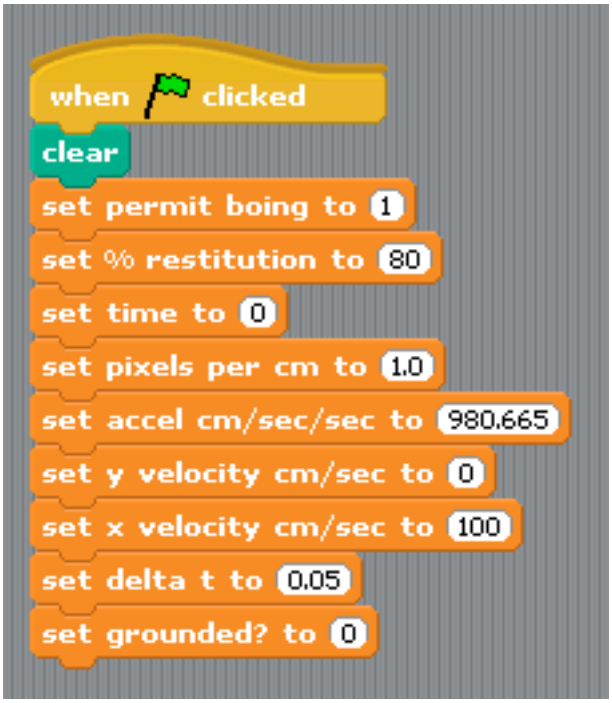
This script makes a bouncing sound.



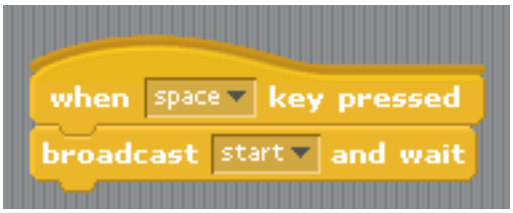
This script is the first part of the grid-drawing script. It draws the lines of the grid every 20 cm.



This script is the second part of the grid-drawing script. It stamps numbers every meter.



This is the variable initialization script.



This script starts the grid-drawing script.