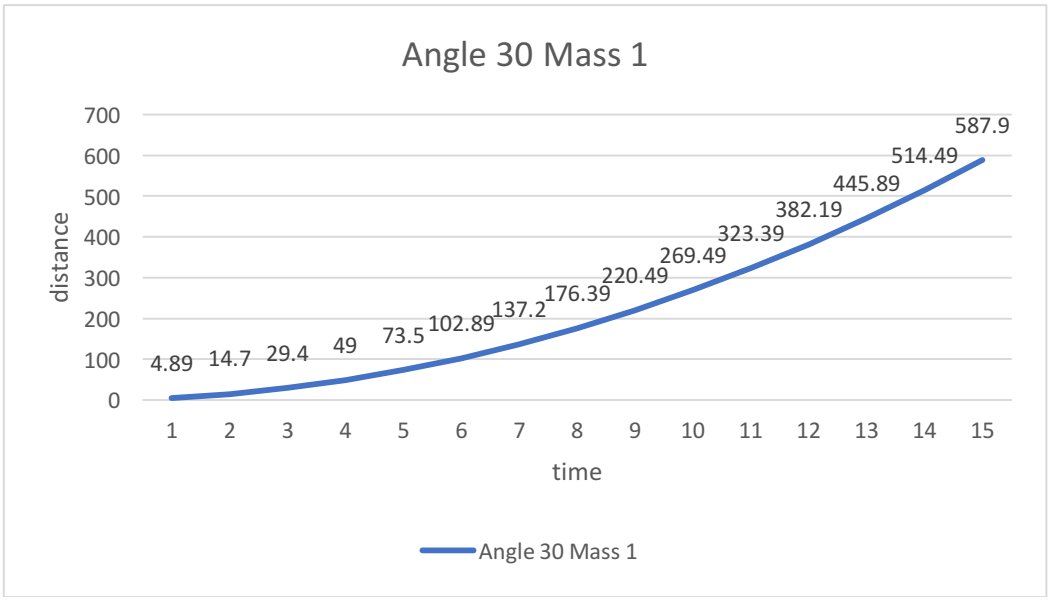
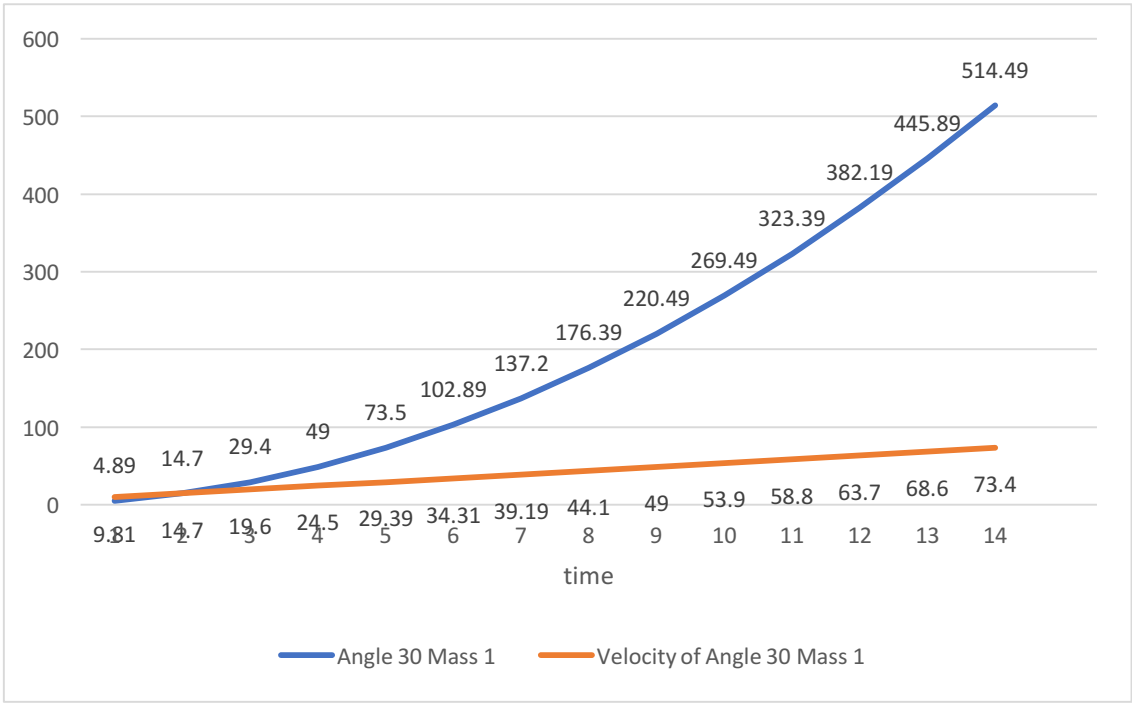
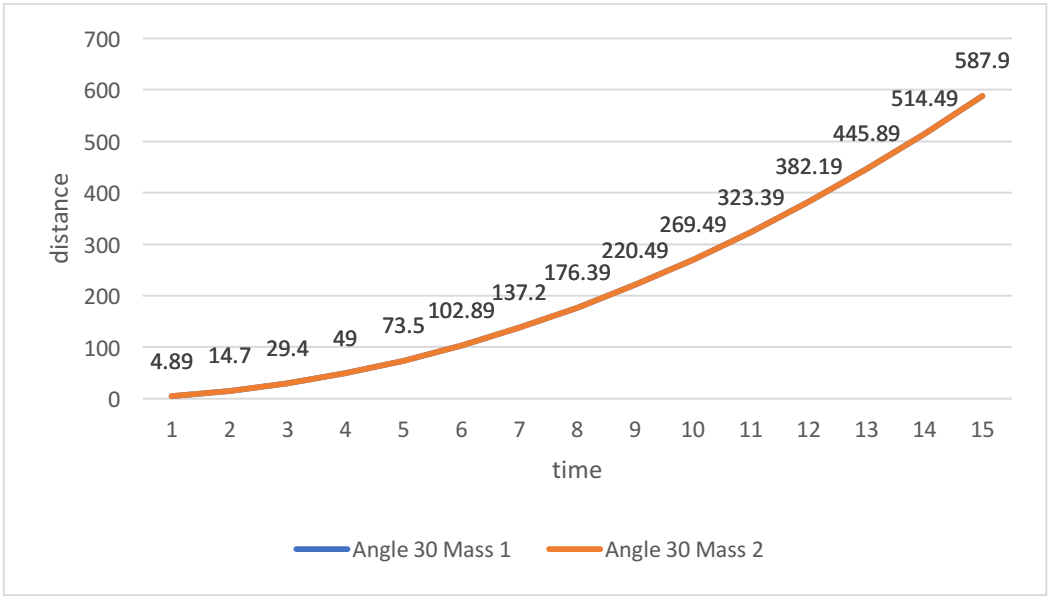
Module 3 Exercises | Joshua Shapiro | 2 March 2017

1. CODE: Download and execute Incline.java. This is a simulation of a bead falling along an inclined wire with no friction. You can set the incline angle and also themass of the bead.
    a. Estimate the distance traveled in 1 second, 2 seconds, 3 seconds, etc. That is, sketch out a graph with time on the x axis and distance (along the incline) along the y-axis.
        i. 

    b. Estimate the velocity between successive tick marks along the incline. That is, what is the velocity between the first two tick marks, between the second two, etc
        i. 

    c. How do these graphs change if you double the mass?

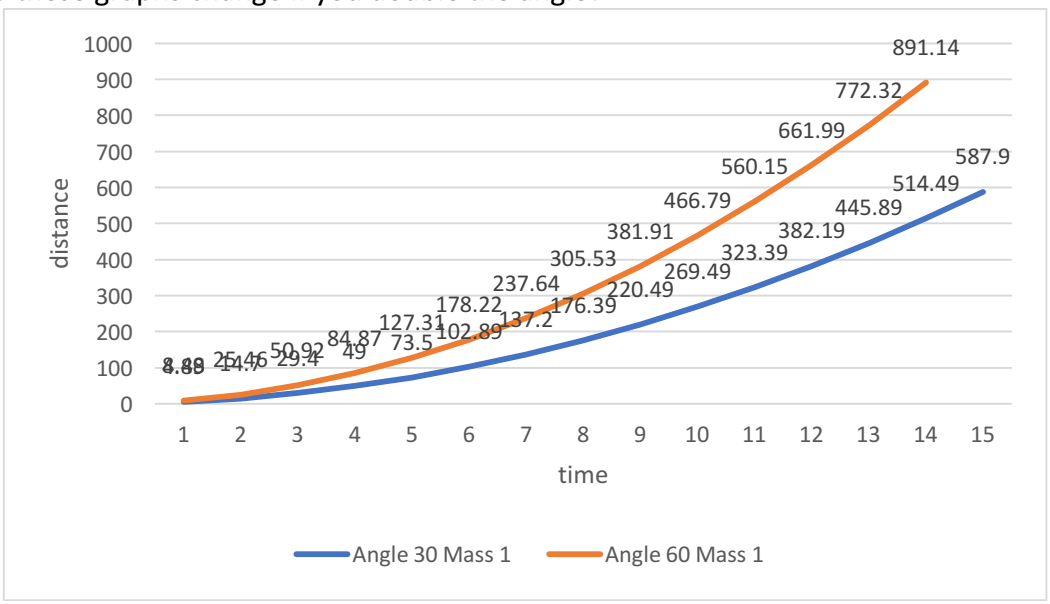
i.



ii. Doubling the mass does not change the distance traveled or the velocity.

d. How do these graphs change if you double the angle?

i.



ii. The distance traveled is greater with a larger angle. Therefore, the velocity is also greater with a larger angle.

e. From the observations only what can you conclude about the relationship between:

i. Distance traveled and time?

1. A greater distance is traveled in the same amount of time when the angle is increased. Mass does not affect distance travelled.

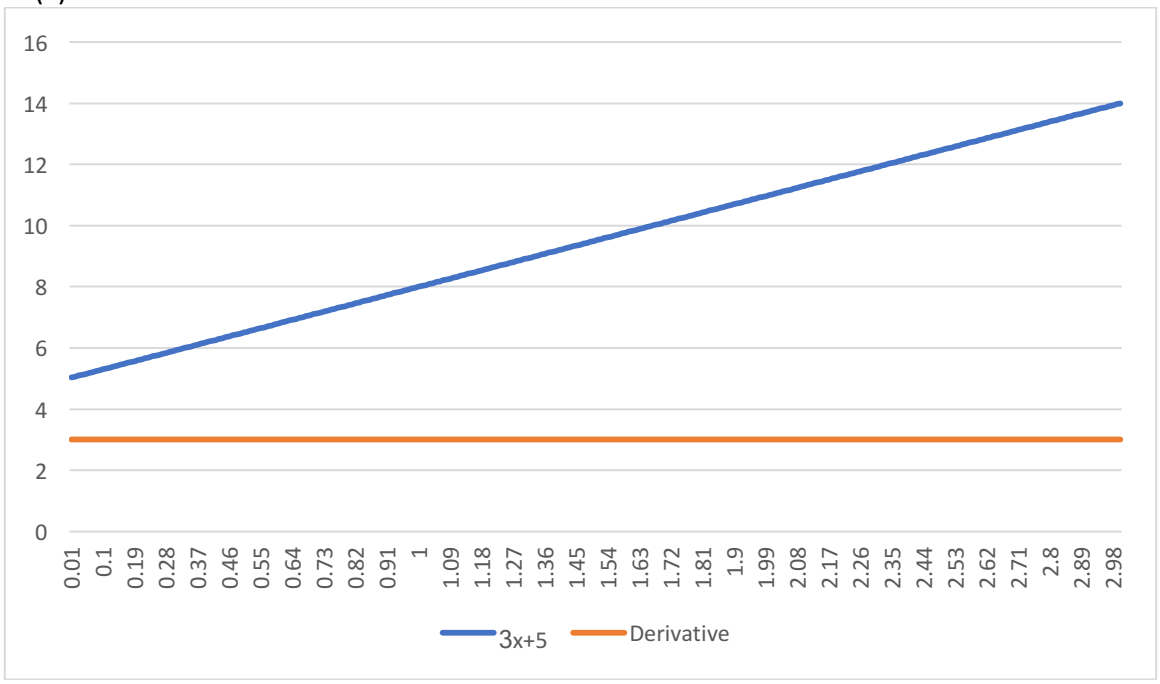
ii. Velocity and time?

1. Velocity is linear in nature. A larger angle means a larger velocity.

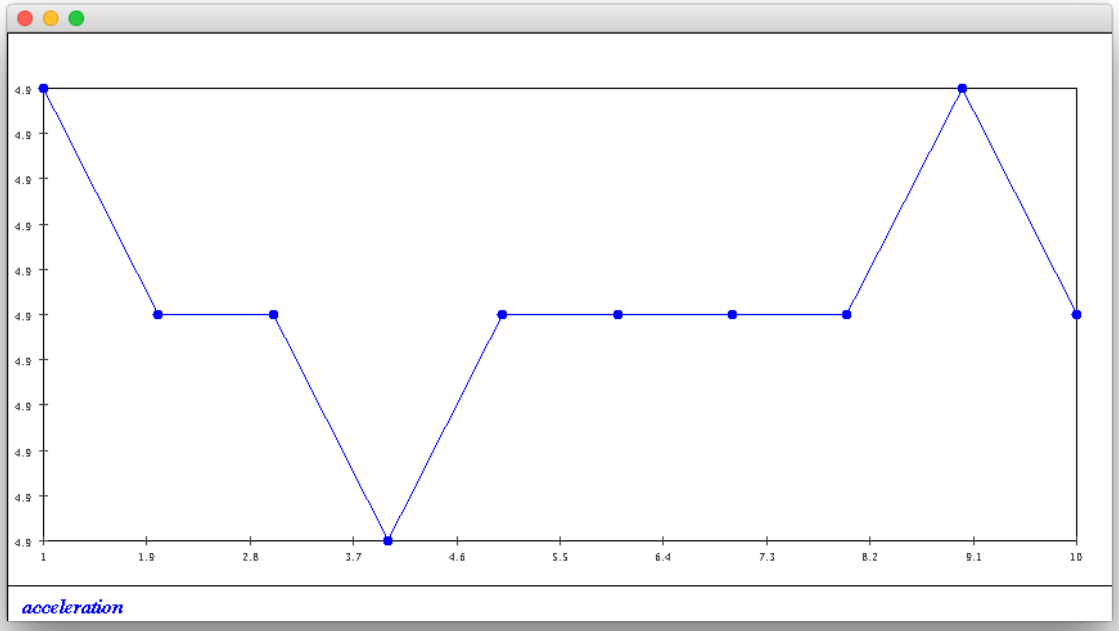
iii. Mass and distance traveled?

1. Mass does not affect distance travelled.

iv. Angle and distance traveled?
    1. A sharper angle makes a larger distance traveled.

2. CODE: Download InclineSimulatorExample.java, InclineSimulator.java, Function.java, and SimplePlotPanel.java. Then compile and execute InclineSimulatorExample. What do you observe is the relationship between distance and time?
    a. As time increases, distance increases exponentially.
3. CODE: Modify the code in InclineSimulatorExample.java to compute the derivative at t = 1,2, etc. For example, to estimate the derivative at t=2, you would obtain d. Obtain d', the distance traveled in 2.01 seconds, estimate the derivative at t=2 as (d'-d)/0.01. What do you conclude from estimating the derivative? Repeat the derivative estimation using 0.0001 instead of 0.01. Can you explain what you observe?
    a. As time increases, the derivative increases linearly.
4. Consider the function f(x) = 3x+5. Use 0.1 and compute the instantaneous rate of change at x = 1 x = 2 x = 3. Then use 0.01 and compute the same. What do you conclude? What can you conclude about the instantaneous rate-of-change of any linear function f(x) = ax+b?
    a. 

    b. The instantaneous rate of change for f(x) = ax+b is f(x) = a
5. CODE: Execute the above code and estimate the slope (by hand, looking at the graph). Then modify the code to estimate the derivative of the velocity function at t = 1, t= 2, … t = 10.
    a. 9.8-4.9 = 4.9
    b. 14.7-9.8 = 4.9
    c. 19.6-14.7 = 4.9
    d. The slope is constant between points.



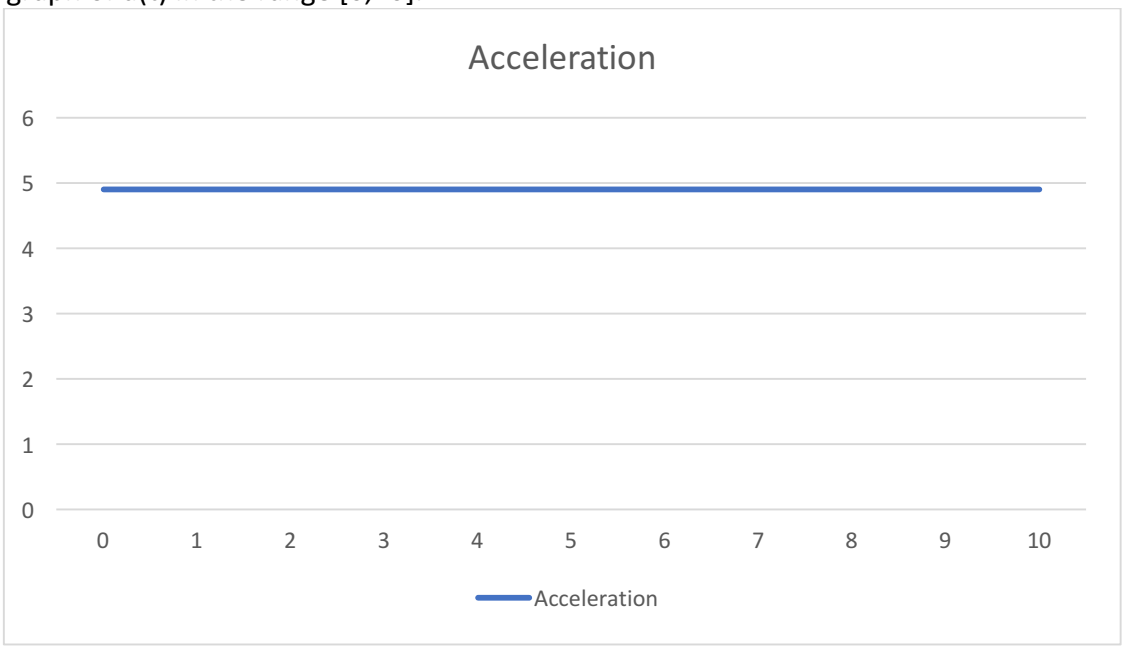
e.

f. This is the graph of acceleration generated by the code. It does not seem constant, but this is due to rounding errors. The y axis is all 4.9. This proves that acceleration is constant.

6. At this point we might try formulating a general "law" about motion:
    a. How would you state the law?
        i. Acceleration is constant, velocity is linear, and distance is exponential.
    b. How is the acceleration affected by the mass of the bead? Try different mass values
        i. Mass is not changing acceleration
    c. How is acceleration affected by the angle? Try angles of 10, 20, …80. You might have to resize the window.
        i. Acceleration stays constant but its value increases as the angle increases.
    d. What can we say about a universal law at this time?
        i. Acceleration is constant, velocity is linear, and distance is exponential
7. CODE: Execute the above code. You will also need BallDropSimulator.java
    a. Modify the code in main to estimate the velocity curve.
    b. Write code to estimate the slope of the velocity curve
    c. Thought exercise: picture yourself in the times of Newton. How would you perform and experiment to measure d(t) for a falling object? What was the clock technology like at that time?
        i. Use an hour glass and drop something from a very high building.
8. CODE: Execute the above code. You will also need BallTossSimulator.java. Then, modify the code in main to estimate the velocity curve. What can you conclude about your universal law thus far?

a. Velocity is still linear, but it can be negative. The 0 velocity mark is when the distance peaks.

9. CODE: Download ProjectileSimulator.java and then modify ProjectileSimulatorExample.java to compute the distance and velocity functions, d(t) and v(t). Use s = 0.01 and for the derivative of velocity, use v'(t) = (v(t+0.1) – v(t))/0.1. Try this one with angle = 37 and once with angle 70. What do you conclude from observing d(t)? What can you conclude about the rate of change of velocity?
    a. The distance traveled is curved. What is unusual is that the velocity is parabolic, and the acceleration is also curved and increasing. This goes against our previous assumptions that acceleration is always constant and velocity is always linear.
10. Suppose a(t) = 4.9 and that initial velocity is zero.
    a. Draw a graph of a(t) in the range [0,10].
        i.



    b. Use s = 0.1 and compute by hand v(0.1), v(0.2), v(0.3), v(0.4), v(0.5). Show your calculations.
        i. Assuming V(0) = 0, V(0.1) = v(0) + 0.1*a(0) = 0+0.1*4.9 = 0.49
        ii. V(0.2) = v(0.1) + 0.1*a(0.1) = 0.49+0.1*4.9 = 0.98
        iii. V(0.3) = v(0.2) + 0.1*a(0.1) = 0.98+0.1*4.9 = 1.47
        iv. V(0.4) = v(0.3) + 0.1*a(0.1) = 1.47+0.1*4.9 = 1.96
        v. V(0.5) = v(0.4) + 0.1*a(0.1) = 1.96+0.1*4.9 = 2.45
    c. Explain why the value for v(0.5) makes sense.
        i. We have determined before that if acceleration is constant, then velocity should be linear.
    d. What is the connection between the calculations you did and the terms "line" and "slope"? What is the equation of the line in question?
        i. Our calculations converted from the slope of a line to the function the slope was taken from. Since the acceleration is the slope of velocity and the value was 4.9, we know that the equation of the velocity function is y

= 4.9x. If the initial velocity was not zero, the initial velocity would be the b component of y = mx+b.

11. CODE: Execute the above code and compare the results (distance, velocity functions) to the InclineSimulatorExample.java example from earlier. Then, explore the following issues:
    a. There is something wasteful about the way we are creating d(t). Can you see what it is? Can the code be modified to make it more efficient?
        i. If we want to calculate distance at 5, then distance at 6, then distance at 7, … to plot a graph of distance vs time, we recalculate a lot of the same values when calculating 7 as we did for 5. If we stored previous values, we could speed this code up tremendously.
    b. What is the effect of changing the interval size? Try interval sizes of 0.1 and 1.0.
        i. Smaller interval sizes yield more accurate results.
12. What could you do experimentally to determine which of Euler or Euler-Cromer is better?
    a. Take a distance function you can compute, find its second derivative to determine the acceleration. Then use the acceleration to determine the original distance function by integrating with the Euler and Euler-Cromer algorithms. The one that yields a result closer to the original function is the one that is better.
13. CODE: Examine InclineSimulator above and answer these questions:
    a. Is this the Euler or Euler-Cromer Algorithm at work?
        i. Euler-Cromer since velocity is updated first
    b. Is the acceleration changing with time?
        i. No, it is constant but dependent on the initial angle.
14. CODE: Execute Incline.java and observe the shadow objects moving. Are you able to say anything about whether the shadow objects satisfy our “universal law”?
    a. It seems like the x and y components follow the assumptions about distance, velocity, and acceleration we made earlier.
15. CODE: Download and execute the above code. Then modify to compute the velocity of the x-axis shadow. What do you observe?
    a. The x distance seems parabolic, while the velocity seems linear. This is in line with our previous observations.
16. CODE: Execute the above program. What do you notice about the distance function for the x-axis shadow? Can you explain? Modify the above code to estimate and plot velocity. What do you observe?
    a. The distance is linear. That is because with a projectile the arc is caused by moving up and down on the y axis with the x axis motion staying constant. As expected the velocity in the x direction is constant yielding linear distance.
17. CODE: Modify the above code to compute and display the distance and velocity functions for the y-axis shadow. Do the x-axis and y-axis shadows satisfy our universal law?
    a. Yes, the x and y axis shadows satisfy our universal law.
18. Why are these statements true? And why is the second equivalent to the first?

a. Based on the above description of calculating the components of velocity and acceleration, x(t) = d(t)cos(theta). So if we have theta and x(t), we can calculate d(t) with d(t) = x(t)/cos(theta).
b. Using the same original equation for x(t) above if we have x(t) and y(t) we can calculate d(t) with d(t) = sqrt(x(t)^2+y(t)^2). Since using x(t) and y(t) we can also calculate theta, the two statements are equivalent.

19. Explain how d(t) is estimated in the above code? That is why does d = d + distance (x,y,nextX,nextY) work? Why did we need the variables nextX and nextY?
    a. We use the Pythagorean theorem to calculate total distance. So the nextX and nextY values are subtracted from the current x and y values. Then the difference is used as the two sides of a triangle in the Pythagorean theorem to find the magnitude of the new distance.
20. CODE: This clean separation into x and y shadows appears mystifying. Do you believe it works? Download and execute InvlineSimulatorExample3.java to compare the two simulators. Print out the horizontal and vertical accelerations.
    a. Yes, this seems to show that the x and y breakdown does work.
21. Are there more unanswered questions regarding gravity?
    a. Is gravity constant on all planets? Does gravity all pull down?
22. Do you have answers for the above two questions? As it turns out they are quite significant.
    a. What exactly is causing this acceleration? Who's doing the "pushing" in that direction?
        i. The force of gravity pushing down is met by the force from the incline pushing up. This converts some of the acceleration into the x direction.
    b. Why isn't the vertical acceleration simply g?
        i. By the conversion mentioned above, the acceleration that isn't put into the x direction is kept in the y direction. Because some of it was converted by the incline, the y direction acceleration is less than 9.8
23. CODE: How many times is the loop iterated? Can you think of a more efficient algorithm for the same problem?
    a. The loop is iterated 1021 times. Set the height extremely high, and then check the velocity and height at each time step. That way the same simulation doesn't have to be repeated every time whenever we increase the height. For example, if we change the height from 5 to 10, we redo the same calculations from 5 in our simulation to 10.We could simply calculate to 10, and and store the velocity at 5 as well (or all the integers to 10 if needed).
24. CODE: Modify the code to answer the third question: At what time and height is the velocity half the final velocity (When dropped from a height of 1000)?
    a. To get the velocity to be 70 (half of the velocity when dropped from a height of 1000), the ball had to be dropped at a height of 250. It took 7.143 seconds to fall.
25. How would you get a more accurate estimate of the initial velocity needed to reach 1000?
    a. Make the interval between the two stop times smaller. In the code it is set to 1. Perhaps make it 0.01?

26. So, is it true that the two velocities are the same? Is there an intuition for why or why not?
    a. Yes, it is true that the magnitude of the two velocities are the same. That is, on the way up the velocity will be positive, but on the way down the velocity will be negative. This makes sense since acceleration is staying constant. Since there is no change in acceleration, it makes sense that velocity will be linear. If velocity is linear, it makes sense that the velocities up until the peak should mirror the velocities after the peak.
27. CODE: What is the best angle? Run the program and find out. Does it make intuitive sense?
    a. The best angle is 45 degrees. This is not surprising, as we want as much of the initial velocity as possible to go in the x direction while still allowing the projectile to not hit the ground. Therefore, it makes the most sense that splitting the velocity equally (angle of 45) allows for the projectile to move farthest to the right while keeping the projectile off the ground.
28. CODE: Write code to solve the second problem: find two angle/velocity combinations to reach a target at a distance of 200. What is the time-of-flight for each?
    a. Angle = 10, velocity = 75.69, time = 2.682
    b. Angle = 25, velocity = 50.58, time = 4.362
    c. The more shallow the angle, the faster flight time.
29. Is it obvious that the route within each region must be a straight line?
    a. Yes. We want to spend the least amount of time in each patch of land, which implies we should be traveling in straight lines through each patch.
30. CODE: Compute by hand the time taken to go from A to P, and then the time taken to go from P to B in terms of a, b, v1, v2, x, y. Then download and modify TwoVelocityProblem.java and implement your formulas as code. Compare the program's results with your hand-worked results.
    a. Time from A to P = $\frac{\sqrt{P_x^2+(a-P_y)^2}}{V_1}$
    b. Time from P to B = $\frac{\sqrt{(P_x-b)^2+P_y^2}}{V_2}$
    c. The time from the code and by hand = 6.403
31. Suppose v1 is much smaller than v2. Which of the three routes shown below is likely to be best.
    a. The top red route. This is because it minimizes the time of v1 and maximizes the time of v2.
32. CODE: Then download and modify TwoVelocityProblem2.java to try different values of y in a loop.
33. CODE: Then download and modify TwoVelocityProblem3.java to try different values of y in a loop, and also compare with the distances d1, d2, (a-y), y. Notice that we are using different ratios V2/v1.
    a. Is there a relationship between v2/v1 and d2/d1?
        i. There does not seem to be a relationship between these two ratios

b. Between v2/v1 and the ratio of (a-y)/d1 to y/d2?
    i. There is an inverse relationship between these ratios. In fact, v2/v1 = (y/d2)/((a-y)/d1)

34. CODE: Before playing with the simulation, what does your intuition tell you? That we can go faster than a straight line.
    a. Download and execute MultiIncline.java, which lets you define the number of segments and move them around. Where you able to get bead down sooner than using a straight line?
        i. Yes, using a parabolic like shape, we were able to move the bead in 11.6 seconds.
    b. What does the phrase "without loss of generality" mean in the context above? What general situations might we be interested in?
        i. We are not defining the starting point or ending point by absolute coordinates. By doing so, we allow the solution we arrive at to be generalized to any system starting at any points. Some of these general situations may be rolling a ball down a hill or a ramp.
    c. Reffereing to the figure below, write down the sin and cosine of the two angles shown in terms of the point coordinates.
        i. Angle 2 (higher one): sin = (y1-y2)/sqrt((y1-y2)^2+(x1-x2)^2), cos = ((x1-x2)/sqrt((y1-y2)^2+(x1-x2)^2)
        ii. Angle 3 (lower one): sin = (y2-y3)/sqrt((y2-y3)^2+(x2-x3)^2), cos = ((x2-x3)/sqrt((y2-y3)^2+(x2-x3)^2)
35. CODE: Download TwoSegmentIncline.java and InclineSegmentSimulator.java. Fill in the missing code above in InclineSegmentSimulator. What reasoning did you use in the missing code? What can you say about the underlying principle?
    a. Have the starting coordinates of the second segment match the ending coordinates of the first segment. Have the ending coordinates of the second segment be the ending segment of the straight solution. Have the final velocities of the first segment feed in as the initial velocities of the second segment. This can be repeated for as many segments as you want, always using the output of one segment as the input for another.
36. CODE: Fill in the missing code above.