MOTION USING A COMPUTER

Once we establish a coordinate system, with a number line that has an origin and a positive and negative direction, we can use numbers to represent motion: position in meters (m), velocity in m/s, and acceleration in m/s^2.

And computers can store numbers.

Background on storing numbers on computers for motion:

- Memory stores 0s and 1s as low or high voltages applied to transistors. Hard disk drives store the 0s and 1s in the magnetic spins of different magnetic domains on the disk. In either case, those 0s and 1s (binary code) can be converted to the 10 decimal digits that we know and love (0-9).

- Each place where a 0 or 1 can go is called a bit. 8 bits make a Byte. 1000 bytes make a kilobyte (kB). 1,000^2 Bytes make a megabyte (MB). 1,000^3 Bytes make a gigabyte (GB), 1,000^4 Bytes make a terabyte (TB), and so on.

- I say this because computers can only store so much information. An irrational number, any number that can't be represented as a ratio of two integers - such as π=3.14159265… or $\sqrt{2}$=1.41421356… or e=2.718281828459045… - would take an infinite number of bytes to store all of its digits, because there are an infinite number of digits. So computers have error in every irrational number it stores.
- Well motion uses all the real numbers, which includes both rational and irrational numbers. So motion will only be approximate.

Motion vectors with glowscript.org:

- Glowscript takes the objects you tell it to create and sticks it on the screen, in the place that you tell it to be put, and with the properties (like color or size or opacity) that you tell it to have, by putting these special keywords inside parentheses which glowscript reads to make that object.

    - e.g. ball = sphere(pos=vec(1,2,3), radius=3.7, color=vec(1,1,1))

    - e.g. ball = sphere(pos=vector(0,0,0), radius=3.7, color=vector(0,1,0))

    - "vec" is shorthand for "vector" and glowscript understands both

- Notice that position and color use vectors, with the keyword "vec". It has 3 numbers, separated by commas, and surrounded by parentheses.

- The color vector is special, the 3 numbers are interpreted as vec(red, green, blue) values, each ranging from 0 to 1. With these, you can make all the colors.

- More importantly, vectors are how you represent position, or velocity, or acceleration in 3-dimensions.
  - A vector is anything with both a direction and a magnitude and position, velocity, acceleration all qualify. In each dimension, they can be positive, or negative, and different absolute values.
  - Something with no direction, just a magnitude, is called a scalar (like time).

- We used one number for position, but this one uses one number for each dimension (x,y,z). Since we live in 3-dimensions, now you have a value for any direction, up/down, left/right, or forward/back. These dimensions are all perpendicular to each other.

- It doesn't HAVE to be this way, you can change the way glowscript does things, but here is how it sets up the coordinates by default.

http://www.glowscript.org/#/user/owendix/folder/Public/program/DefaultGlowscriptAxes

- For position: pos = vec(1,2,3) the first number tells you the x-position, the second the y-position, the third number the z-position. Order matters, they have to be read in this order.

- Glowscript borrowed this term from math. NOT in glowscript, but in physics and math, you can use vectors for position, velocity, acceleration, in 1 or 2 or 3 or 4 or 5 or… dimensions. Any number of dimensions you want. We've mostly just been using 1-dimension, which is just fine.

- These axes don't have to be mysterious. The code is complicated because I wanted to make it very versatile, but you can easily make your own axes to show glowscript's default axes.
  - Have all your arrows start at the origin: vec(0,0,0)
  - Make them different colors: color.red, color.white, color.blue
  - Instead of showing both positive and negative, just show the positive direction. You know negative points opposite of it.
  - Instead of making them different lengths with tick marks, just make them length 1, then you know position vec(2,0,0) is twice as far in the x-direction as your axes.
  - Have the code print to screen what you ended up doing. Here's some basic syntax for that:
    - print('red=+x, white=+y, blue=+z')

Updating motion with time:

- Having an object move is just changing its position as time passes. This means we need to create a variable called t (for time) and assign a value of 0 to it (a good place to start).
    - Look at the Lesson 4 example code. Notice on line 17 he writes t = 0, which does just what we wanted.

    - To understand how glowscript changes this value of t to make the time change, you need to know that glowscript reads the code and (after first checking there's no obvious errors) it executes the code line by line, going down the page.
        - Not all computer languages do this, exactly. Compiled languages compile all the code ahead of time, which turns it into machine instructions, but it still has to execute these instructions sequentially, for the most part.
        - Glowscript, based on VPython, based on Python, is an interpreted language so it does only a tiny bit of checking when you first hit Run program, but then it interprets each line by line.

    - Nevertheless, if you read the code the same way and think about how each line effects what was already written beforehand, you can understand what the code is doing.

    - This lesson first reads the header to set its version, and skips over any comments (#) until it gets to the ball = sphere(… line. This creates a sphere and gives that sphere and all its properties the variable name "ball", so the code can adjust that sphere later.

    - The next important line is creating the ground, which it does by making a box with those certain position, length, width, height, and color and stores that box in the variable name "ground", so the code can adjust that box later too (it never does so we didn't REALLY need to store that box, but it's not a big deal to have a variable you don't do anything with…at least not until you start writing big long programs - which we won't do).

    - Next, the code creates a new property for the ball that it calls v (all of the properties of a saved object can be accessed by putting the period and that property name - its called a method), which it uses to store another vector for the velocity of the ball. THIS IS IMPORTANT.

    - Finally we return to t = 0.

    - Next line: dt = .001, line 19.

    - Why? Because computers can't store an infinite amount of numbers, we need to pick some small-ish amount of time to skip ahead, which we will add to the variable t to make t change. David Weaver called this little change in time, dt and gave it the value 0.001.

    - Note computers don't need units, that's something YOU need to keep track of and make sure you're using all the correct units for. Units are just an agreed-upon scale. The Mars Climate Orbiter crashed into Mars because people messed this up. See Thing Explainer p. 50, How to Count Things, by Randall Munroe, for more information about units.

- Finally, we're at the while loop:
    - "while" is a special keyword for Glowscript and really all computer languages (that I know).

    - It is a loop, so it repeats the stuff that's indented a bunch of times. It goes through all these lines and when it gets to a line that isn't indented as much as this block is, it goes back to the "while"…

    - When does it stop? The while loop keeps going until the condition written after "while" is no longer true. While it is true, the loop continues to go.

    - This says while the time is less than 10, keep looping. That means you should expect (UNLESS SOMEONE MESSED UP) that the time will be increased at one line in this while loop. If you glance down the page, it is, at the very bottom, line 30. t = t + dt. This increases time from what it was just a second ago, to that plus dt (= 0.001).
        - Details: when a computer evaluates a statement with an equals, which is called an "assignment" statement because it assigns a new value to the variable on the left (t), it first evaluates the right hand side of the equals, then it assigns the number to the variable t

        - e.g. t = (0 + 0.001)

        - Now t = 0.001
    - In summary for the while loop, first the while loop's condition is checked to see if its true (is t greater than 10?), if it is, it executes the next line inside the indented block. If it isn't true, it goes past all the indents.

        - Here there is nothing past the indented so the program ends and wherever the objects are located on the last loop, that's where they stay frozen on the screen.
- First line of the while loop: rate(5000), this slows down or speeds up how quickly the screen gets updated (a big rate makes it go faster)

- THE BIGGIE:
    - ball.pos = ball.pos + ball.v*dt

    - This updates the ball's position. It evaluates what's on the right hand side then assigns it to what's on the left hand side (ball.pos).

    - Is this equation correct and why?

    - Yes it is correct for a certain, very simple approximation (called the explicit Euler method named after a mathematician named Leonhard Euler, 1707-1783).
        - One way to be more confident is the units:
        - Left-hand side has units of length (say meters)
        - The right hand side has meters + (meters/sec)*sec = meters + meters = meters.
        - The left and right hand side have the same units so that's good.
    - It comes from the definition of velocity:

        - vel = delta-pos/delta-t = pos(t+dt) - pos(t)/dt

            - By pos(t) I mean the position of the ball AT the time, t

            - By pos(t+dt) I mean the position of the ball AT the time, t + dt

    - If we know where the object is located now (t) and we want to find where the object will be located at the next increment of time (t + dt), and we also know the velocity and the time interval, then we have everything in the equation except pos(t+dt), which is what we want to find, so solve for it:

    - pos(t+dt) = pos(t) + vel*dt

    - Q.E.D. - which is what had to be shown

        - Note that if the velocity changed as time when on too, that is if there was a non-zero acceleration, then velocity would have to be updated every step. This is true for Lesson 9.

    - Now we're back to updating the time, then the while loop loops, continually updating the position of the ball until the time is greater than 10.

    - Last technical note: the equation ball.pos = ball.pos + ball.v*dt has two different kinds of math in here:
        - It multiplies a scalar (time) by a vector. To do this, you simply multiply the scalar (0.001) by every component in the vector:
            - e.g. vec(1,2,3)*(0.001) = vec(0.001, 0.002, 0.003)
            - Note the result is a vector
        - Second, it adds two vectors together. To do this, you simply add each component individually:
            - e.g. vec(1,2,3) + vec(4,5,6) = vec(5,7,9)
            - Note the result is another vector
    - Why should you care?
        - Computers are all around us. You can't escape.

        - This process of specific, clear steps for how to solve some problem is called an algorithm, named after the Persian mathematician Al-Khwarizmi (780-850), as is the name algebra.

        - Algorithmic thinking is a very careful, methodical, logical kind of thinking. It takes patience to go through all the steps and realize what has to be done.

        - If you actually go to make you own code, there is a learning curve to trying to write code so that a very literal thing like a computer will understand it and follow it exactly how you want.

Vectors as arrows:

- The 3 numbers of a vector can represent the measurements for how much that vector points along each of the 3 axes: x, y, and z. For simplicity, though, let's think in just 2 dimensions.

- Imagine we established a coordinate system and a certain vector we care about doesn't point entirely along just the x or just the y axes. We could make a right triangle with the vector as the hypotenuse and the 2 legs of the triangle WOULD lie entirely along the x and y direction.

- If we stick arrowheads on these right-triangle legs, so that they start at the same point as the original vector, they add tip-to-tail to each other, and they end up pointing at the same point as the original vector, then these two leg-vectors are called the components of the original vector.

- More practically for glowscript and graphing, the y-component of a vector you know, like vec(1,2,3), is just another vector with all numbers 0, except that middle y-number, which is the same as the original vector, so vec(0,2,0). You can do something similar for x and z.

- See Vectors with and without Coordinate Systems for examples and more information.