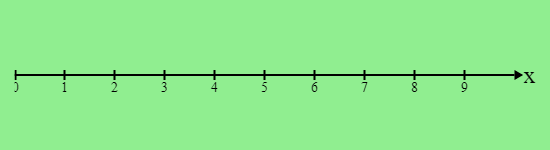
**Unit 2 – Activity 3**

**Motion Maps**

Open the code found here for a simulation of a dog running at a constant velocity: <https://tinyurl.com/y89jru2k>. Complete the next-x function the same way we have in previous simulations. Keeping motion-map = "hide" on line 11 and play around with the velocity and initial position of the dog (NOTE: each tick mark on the axis represents 1 meter). Don’t worry about marker-type and time-between-snapshots for now.



One way we can represent the motion of an object is to visualize its position at different times on the same image. Imagine taking snapshots of the dog’s motion at set intervals of time and superimposing them all on top of each other. We call this representation a ***motion map***. To create a motion map of the dog moving, change line 11 so that it reads motion-map = "show". Then run the program again.

1. Sketch what you see when your simulation ends.
2. Run the simulation again with a different velocity and initial position and sketch what you see when your simulation ends.
3. What things can you tell about the motion of your dog from looking at the motion map? How do these things show up on these motion maps?

To make our lives easier, let’s represent the snapshots of the dog with just a circle. Edit line 13 of your code so that it reads marker-type = “dot” and run your simulation again.

1. Sketch what you see now.
2. Run your simulation with the following conditions and draw the motion maps it produces.

C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png

C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png

1. What do you notice about the two motion maps in question 5? Suggest something we could change about our motion maps to fix this.
2. Edit line 13 of your code so it reads marker = “dot with arrow”. Predict what the motion map will look like for the following initial conditions. Once you’ve made your predictions, test them with your simulation and draw the outcome of each simulation.

|  |  |
| --- | --- |
| **Prediction** | **Outcome** |
| C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png | C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png |
| C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png | C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png |
| C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png | C:\Users\jrutt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\download.png |

1. Determine the velocity and initial position of the dog in each case from the motion map it creates. Use your simulation to check your work.

|  |  |
| --- | --- |
| **Motion Map** | **Initial Conditions** |
|  | \_\_\_\_\_\_\_\_\_\_,  \_\_\_\_\_\_\_\_\_\_ |
|  | \_\_\_\_\_\_\_\_\_\_,  \_\_\_\_\_\_\_\_\_\_ |
|  | \_\_\_\_\_\_\_\_\_\_,  \_\_\_\_\_\_\_\_\_\_ |

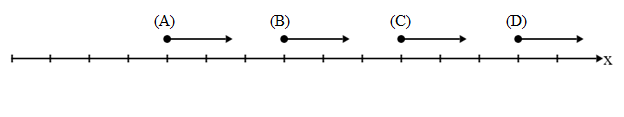
The most important thing to keep in mind when making a motion map is that the amount of time between each “snapshot” or dot we draw. We can only draw comparisons between the motions represented in two different motion maps if they are using the same amount of time between each dot. To see this, play around with the value of time-between-snapshots on line 15.

1. See if you can produce two motion maps where there is more distance between the dots of a slower moving object than there is between the dots of a faster moving object. Draw the motion maps below and record the initial positions, velocities, and time between snapshots for each.

The code below was used to generate a motion map:

|  |
| --- |
| x-initial = 4 #m  v = 1.5 #m/s  delta-t = 2 #s  motion-map = true  **fun** next-x(x):  x + (v \* delta-t)  **end** |

The map produced by this simulation is shown below, where each tick mark on the x-axis represents 1 meter.



1. Identify which of the following calls of next-x would produce each dot.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| next-x(0) |  |  | next-x(1) |  |
| next-x(3) |  |  | next-x(4) |  |
| next-x(5) |  |  | next-x(7) |  |
| next-x(10) |  |  | next-x(13) |  |

1. Fill in the function calls above with the value would have after calling the next-x function with that input.