# Unit 2 - Constant Velocity

## Table of Contents

# Instructional Goals

1. **Differentiate position, distance and displacement**
   - Develop calculations of distance and displacement
     - Represent displacement as delta-x
2. **Differentiate clock readings and time intervals**
   - Establish *delta-t* to represent the time interval
     - Connect $\Delta t$ and `delta-t`
3. **Understand slope as a rate of change**
   - Use the rate of change to find physical meaning in a graph
4. **Velocity as the rate of change for position over time interval**
   - Calculate average velocity as $\frac{delta-x}{delta-t}$
5. **Using a motion map to pictorially represent the motion of an object**
   - Connect the motion map representation with the computational representation
6. **Representing the motion of an object with a one- or two-argument function**
   - Construct `next-x()` as a function of current position `x`
   - Construct `next-x()` as a function of current position `x` and average velocity `v`
   - Introduce *Boolean* as a Pyret data type (alongside *Number*, *String*, and *Image*)
7. **Using multiple representations to describe the motion of an object**
   - Find relations between state diagrams, motion maps, programming functions, and graphs

# Student Learning Objectives

CV.1 - I can represent the motion of an object using a computational representation in which a differential is added for each interval of time.

CV.2 - I can design and modify functions in Pyret to represent constant velocity motion and use them to make predictions.

CV.3 - I can differentiate between position, distance, and displacement.

CV.4 - I can find the velocity of an object using the slope of the position vs. time graph. I can interpret the initial value of a position vs. time graph. I can draw an average velocity line on the graph and calculate the average velocity.

CV.5 - I can create a mathematical representation (function) relating position, average velocity and time, and use it to solve problems.

CV.6 - I can represent the motion of an object moving with a constant velocity using multiple representations (words, motion maps, position vs time graphs, and velocity vs. time graphs, computational function, etc.) and use them to solve problems.

CV.7 - I can relate displacement to the area between the horizontal axis and the line on a velocity vs. time graph.

## Constant Velocity: An Overview

This unit introduces students to using laboratory tools to make quantitative measurements, specifically of position and time, and many new representations available to them when making quantitative observations (e.g., data tables, motion maps, computer simulations, position vs. time graphs, and velocity vs. time graphs).

The unit begins with a paradigm lab, collecting measurements of the changing position of an object with relatively constant motion at various clock readings. Using this collected data, students create a motion map and a computational simulation of the object, resulting in the development of the relationship between the position at one moment in time and the next. Critically, students first discover the pattern of the object's next position at some future time being the *current position plus the change in position during that time interval*, and later revised to the *current position plus the product of the velocity and the time interval*. This progression leads to the operational definition of velocity as the ratio of the change in position over the change in time.

The ***differential representation*** of motion which is introduced through coding (contrasted with the more common parametric representation of motion) is central to the understanding of motion built in this unit. This idea — that position changes iteratively by some small amount $\Delta x$ over some small time interval $\Delta t$ — is a natural way to think about change and forms the basis of our computational representation of motion. Investigating the relationship between $\Delta x$ and $\Delta t$ naturally leads to the definition of *average velocity* as

$$\bar{\vec{v}} = \frac{\Delta \vec{x}}{\Delta t}.$$

An emphasis is placed early in the unit on collected data (stored in tables), computer simulations, and motion maps. This intense focus on the development of motion maps is intentional and important because motion maps are a more concrete representation of motion than graphs of position and time. By linking tabulated measurements, state diagrams, motion maps, and computational functions together so tightly, students are more ready to make the leap to the higher-order graphical representations later in the unit, and each subsequent unit.

By the end of the Constant Velocity unit, students are able to accurately describe the motion of objects with constant velocity, represent it through multiple representations, and predict the motion of objects with constant motion beyond the time they have observed the motion. They will also be able to identify objects which do *not* behave like they have constant motion - priming the deeper dive into changing motion and uniform acceleration in the next unit.

### I. An Operational Definition of Motion

The combined models of motion (e.g., constant velocity, uniform acceleration, projectile motion, rotation, etc.) fall under the umbrella of "kinematic" models - the models which produce a *description* of an object's motion over a time interval. This is contrasted with "dynamic" models - models which explain the *cause* of an object's motion over a time interval. (i.e., force models).

Motion is defined as *the change in position of an object (or system) between two states*.

# II. Vocabulary of Motion

It is extremely important the vocabulary we use is as clear as possible to make sure that we are not using ambiguous terms that can lead to more student confusion, and even more so we need to ensure that our students mean the word they are using rather than playing darts with the vocabulary and just tossing various words out in the hopes of landing on the right one. It is important to ask your students from time to time to define the term they use, especially early in the unit as they start to use vocabulary to help strengthen the connection to the concepts we are studying, even if they use terms correctly. Students will use the correct term at times while still not understanding *why* it is the correct term in the proper context. It will take extra time early on but lead to a stronger understanding and pay off in taking less time reteaching this vocabulary later.

## Clock Reading and Time Interval

**Clock Reading -** this is the numerical value represented by the clock, such as the position of the second hand on an analog clock, or the numerical output on a digital clock. Clock readings expressed in seconds can be abbreviated as t(s). Clock reading would be the value of the time for a given state of the system. This is a *scalar* quantity as it *does not* require a direction to be defined.

**Time Interval** - it requires two clock readings to define a *time interval*. It is the difference in time from the initial clock reading ($t_i$) to the final clock reading ($t_f$). Time interval can be used interchangeably with 'delta-t.' This is a *scalar* quantity as it *does not* require a direction to be defined.

$$\Delta t = t_f - t_i$$

## Position, Distance, and Displacement

**Origin -** arbitrarily selected reference point from which all positions will be measured. This could be referenced as 'the equator' for latitude, 'prime meridian' for longitude, or 'the ground' for height measurements. Horizontal motion allows for a much more arbitrary definition and can allow for nearly any location to be the origin but once selected, it is locked in for that situation. The object does *not* need to have been present at a location for that location to be a useful origin.

**Distance -** requires two points to be referenced and measure or calculate the length of the path between the two points. Distance can be measured as the path an object takes from one clock reading to another, "counting the steps" colloquially. This is a *scalar* quantity as it *does not* require a direction to be defined.

**Position -** location of an object as measured from the defined origin, including both the 'distance' away and the direction. For example, '3 meters, North' is different from '3 meters, South' or '3 meters, East.' This is a position value for a specific state, just as clock reading is a time value for a specific state. These would be 'state values.' This is a *vector* quantity as it *does* require a direction to be defined.

**Displacement** − as with time interval, displacement requires two clock readings to be defined. Displacement is the difference between the position at an 'initial' clock reading and the position at a 'final' clock reading. Displacement is abbreviated as 'Δx.' This is a *vector* quantity as it *does* require a direction to be defined.

$$\Delta\vec{x} = \overrightarrow{x_f} - \overrightarrow{x_\imath}$$

## Average Speed and Average Velocity

These are both ratios, and as such will be a much more challenging piece to define without a very strong understanding of the previously defined terms.

**Average Speed -** ratio of the distance traveled between two clock readings and the time interval between those same clock readings. This is a *scalar* quantity as it does not require a direction to be defined.

$$avg.speed = \frac{distance\ traveled}{time\ interval}$$

**Average Velocity -** ratio of the displacement traveled between two clock readings and the time interval between those same clock readings. This is a *vector* quantity as it *does* require a direction to be defined.

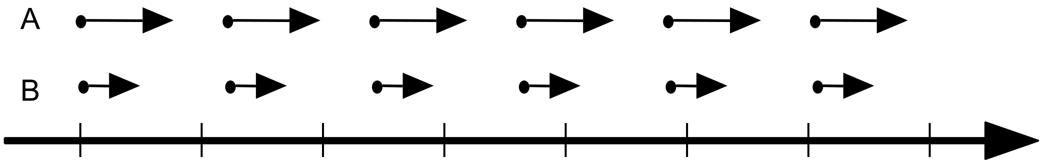
$$avg.velocity = \frac{displacement}{time\ interval}$$

Or using a more algebraic representation:

$$\bar{\vec{v}} = \frac{\Delta\vec{x}}{\Delta t}$$

## III. Representational Tools

Motion maps form the basis for the computational representation of motion in this unit. Emphasize that the difference between dots is the displacement of the object for that particular interval of time. As the time interval decreases the displacement must also decrease if the object is to remain moving with the same rate of change.

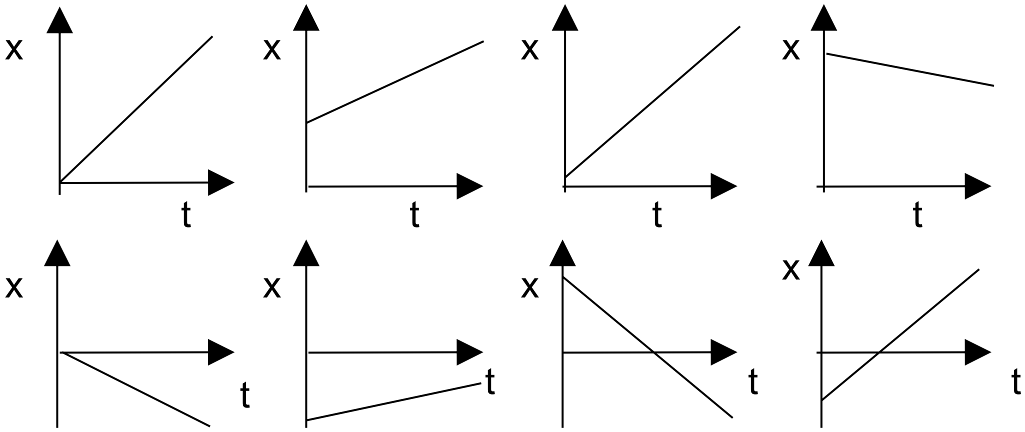


Students have been known to draw the motion map above and state that A was moving faster than B because the velocity vectors were longer. If object A is moving faster than object B, then the space between the dots should also be greater because A's change in position will be larger during each time interval.

Work on making motion maps to represent the position-time behavior of moving objects. Make sure that these semi-quantitative devices are faithful representations depicting where the object is at evenly-spaced clock readings.

1. The definition for *average velocity*, $\bar{\vec{v}} = \frac{\Delta\vec{x}}{\Delta t}$, should be used explicitly in discussing the physical significance of the slope of a position-time graph.

2. When discussing the meaning of the graphs, be sure to use a wide variety of examples.

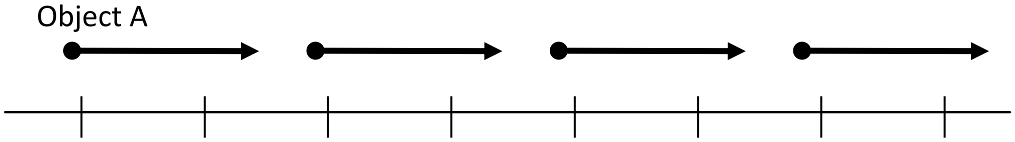


   Induce the students to describe the motion in full detail (e.g., in the fourth graph, the object starts somewhere to the right of the zero position and moves to the left at constant speed).

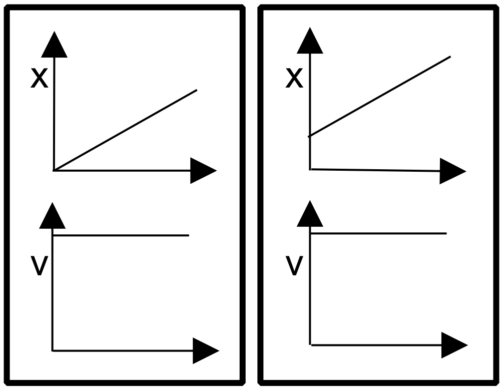
3. Students will have to be taught how to manually produce a graph and do a mathematical analysis of the graph. Students have been conditioned to think of slope only as "rise over run" or Δy over Δx. They need to understand that the slope of any graph describes the rate of change in the physical quantity represented on the vertical axis with respect to the one represented on the horizontal axis.

4. Make sure that they have a thorough grasp of the relationship between slope and velocity. The answer "1's slope is greater than 2's" is not a guarantee of understanding. It would be profitable to have students model the behavior of the object represented by a variety of graphs. If you have an ultrasonic motion detector, this is great fun!

5. Make sure that students can, given a verbal description, an algebraic statement, a function in Pyret, an $\vec{x}$ vs $t$ graph, or a motion map, recreate the other four representations.



Given the motion map above, they should be able to write a verbal description of the motion, express the relationship $\vec{x} = \vec{v}t + \overrightarrow{x_o}$ and draw the graph at right.



6. Be sure to make the connection between $\vec{x}$ vs $t$ graphs and $\vec{v}$ vs $t$ graphs. "Stacking" the curves by placing the $\vec{v}$ vs $t$ graph directly underneath the $\vec{x}$ vs $t$ graph helps to illustrate this relationship.

Make the point that the $\vec{v}$ vs $t$ graph yields no information about starting point. In the two stacks shown, two different $\vec{x}$ vs $t$ graphs can be represented by the same $\vec{v}$ vs $t$ graph.

7. Make the point that the area under a $\vec{v}$ vs $t$ graph represents the displacement, the $\Delta\vec{x}$ of the object. This could be both (+) and (-). Avoid always using the trivial case.

8. Key understandings about computation in physics raised in this unit are:
   A. Computers allow us to do many repeated calculations which would be too time-consuming to do by hand.
   B. Computers can simulate (represent) physical behavior and use those simulations to make predictions about physical phenomena.
   C. Instructions which should be repeated multiple times can be written using functions

## Sequence

1. Lab 1 — Buggy Lab
2. Activity 1 — Simulated Motion
3. Activity 2 — Advanced Simulated Motion
4. Worksheet 1 — Motion Maps
5. Worksheet 2 — Position-Time Graphs
6. Worksheet 3 — Two Bicycles
7. Lab 2 — Graph Matching with Motion Detectors
8. Activity 3 — Multiple Objects
9. Lab 3 — Colliding Buggies
10. Worksheet 4 — Distance vs. Displacement
11. Reading 1 — Vocabulary of Motion
12. Worksheet 5 — Velocity-Time Graphs
13. Activity 4 — Multiple Representations of Motion
14. Activity 5 — Rocket Lander Game
15. The Model So Far

# Lab 1 — Buggy Lab

## Resources

- Unit 2 Lab 1: Buggy Lab

**Note:** The included worksheet includes pre-lab observations and should be given in conjunction with the lab.

## Apparatus

- Any slow-moving battery powered toy vehicle (preferably two—a slow and fast one[1])
- Some type of timer (stopwatch, app on smartphone, metronome - actual or virtual[2])
- Meter sticks or metric tape
- Paper receipt tape or long strip of butcher paper
- Markers to mark the position of the buggy

## Pre-Lab Discussion

- Let the vehicle move across the table and ask students the following:
  - What do you notice/observe?
  - How can you tell if it is moving?
  - What can we change? What are the possible inputs?
  - What is measurable? What are the outputs for our possible inputs?

List observations and then ask which items are measurable. Students are likely to say "speed"; point out that since the buggy doesn't have a speedometer, this is a quantity we cannot directly measure. Lead them to observe that the BPV (battery-powered vehicle) moves at constant speed, i.e., that it travels equal distances in equal time intervals. The pre-lab discussion should lead to a definition of motion along the lines of "something is moving if its position changes" or "its position is different at different times." Based on this definition of motion, the experiment is designed to measure the position of the buggy at set intervals of time, as determined by a metronome (real or on computer) set to 30 ticks/minute.

The variable being measured in this experiment is position ($\vec{x}$). Emphasize the difference between position and distance. (Note: To avoid confusion due to the similarity in terms, we will avoid the use of the word *displacement* until later in this unit.) You can give them a quick description of vector quantities here to explain why there is an arrow in the symbol if you want. Position will be measured in centimeters. Positive and negative positions are on opposite sides of the origin (0). Another important thing to keep in mind in this experiment is that students are not recording time as a variable in their experiment. In discussing their data, the individual ticks of the metronome are referenced rather than some clock reading or absolute time.

Before you begin data collection for this lab, instruct students to create an accurate number line on the receipt tape, including both positive and negative values. It is recommended that this be

[1] https://www.arborsci.com/products/constant-velocity-car
[2] https://www.webmetronome.com/

long enough to allow for the buggy to travel for a minimum time interval of 20 seconds. Assign individual student teams a starting position and direction of travel (away from zero or toward zero) to ensure that post-lab discussion includes students with slow and fast motion in both directions.

Show students how to make a proper data table for this experiment if necessary. In this case, the data collected fits in a single row or column for the position of the buggy at each tick.

Students should end the lab with some description of a pattern they observe in their data.

## Lab Performance Notes

Slow-moving vehicles make it easier for students to mark the position without the use of a high-tech apparatus. The constant speed buggies can be made to move more slowly by wrapping one of the batteries in electrical tape, then in aluminum foil, or by replacing one of the batteries with a wooden dowel wrapped in aluminum foil. If you use a variable speed constant velocity vehicle it is recommended that you cover the adjustment knob so students cannot change the speed themselves.

To obtain position data, students start by placing the vehicle on the paper receipt tape or long strip of butcher paper. They can mark the position of the vehicle on the paper every time the metronome ticks. Younger students especially may not have the coordination needed to get good data from the first few tries. Students should be reminded to perform multiple trials with at least 6 data pairs/trials. Averaging the values of position helps them develop a sense of the precision they should carry through the analysis.

## Post-Lab Discussion

Students should display their data tables on whiteboards and discuss patterns, observations, and predictions and conduct a board meeting. The goal of the board meeting is for students to come to a consensus about the pattern they observed. Not every buggy moves at the same speed, so groups should not agree on the exact amount added to the buggy's position each tick, but they should agree that the buggy's position at the next tick can be found by adding some amount to its current position. Mathematically, this can be expressed as

$$x_{final} = x_{initial} + \Delta x$$

Focus students' attention on how the position of their buggy follows the average change in position between each tick. Then have each group predict the next position value another tick after their last data point and discuss their methods to determine it. If time allows, have students test these predictions. Students added delta-x to x to form a constant rate of change for the position of the buggy. This conclusion is important as it will form the basis for the Pyret simulation reproducing the motion of the buggy. Then move on to Activity 1 where they will use this idea to write a function in Pyret. They should keep their whiteboards or take photos/notes if you need to wait until the next class for the next activity. The receipt tapes from the lab should also be saved as we will be using them again in subsequent lessons.

# Activity 1 — Simulated Motion

## Resources

- Student Code: https://tinyurl.com/U2-Simulated-Motion
- Unit 2 Activity 1: Simulated Motion

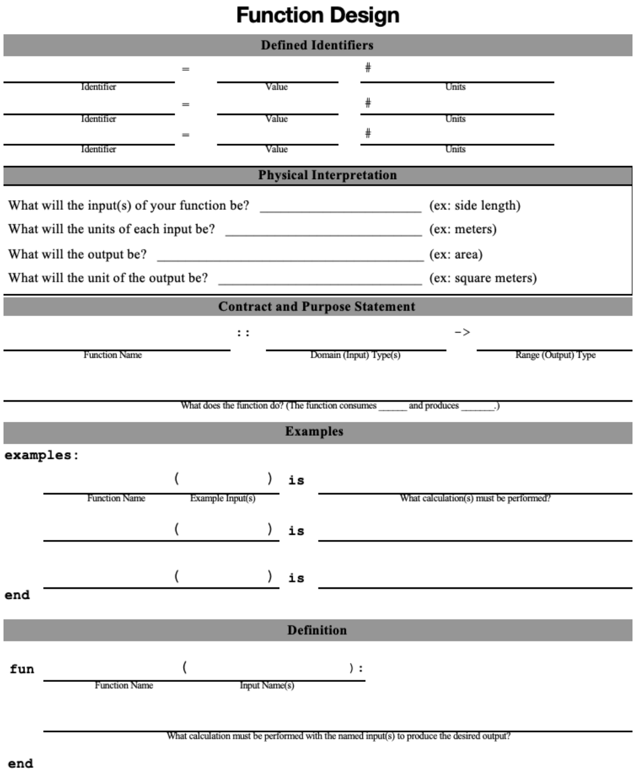
## Introduction

The goal of this activity is for students to learn how to write a differential function to reproduce the motion of the constant velocity buggy. They will learn how to define the function next-x and use it to tweak a simulation of a moving tumble buggy.

Students will use a function design with simple examples in conjunction with the worksheet. The questions in Part I of the worksheet guide students through all the information they need to fill in the function design. *It is strongly recommended that you keep a stack of blank function designs somewhere in your classroom for students to take when they need them.*

Students should reflect on the pattern found in the buggy lab data, that for each tick a constant value (delta-x) was added to each position to form a constant rate of change. Follow the Function Design Performance Notes below to assist students in completing the function design.

After students have completed and defended their function design, they should begin typing their function's design into Pyret. The full URL for the simulation is in the Resources above.



**Function Design**

**Defined Identifiers**

_______ = _______ # _______
Identifier Value Units

_______ = _______ # _______
Identifier Value Units

_______ = _______ # _______
Identifier Value Units

**Physical Interpretation**

What will the input(s) of your function be? _______ (ex: side length)
What will the units of each input be? _______ (ex: meters)
What will the output be? _______ (ex: area)
What will the unit of the output be? _______ (ex: square meters)

**Contract and Purpose Statement**

_______ :: _______ -> _______
Function Name Domain (Input) Type(s) Range (Output) Type

_______
What does the function do? (The function consumes ______ and produces _______.)

**Examples**

```
examples:
    _______(_______) is _______
    Function Name  Example Input(s)  What calculation(s) must be performed?
    _______(_______) is _______
    _______(_______) is _______
end
```

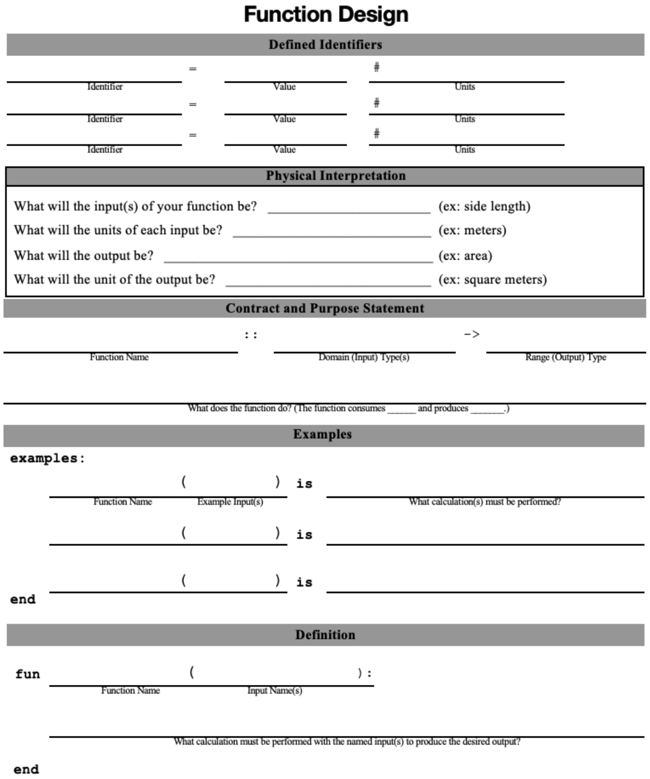
**Definition**

```
fun _______(_______):
    Function Name  Input Name(s)
    _______
    What calculation must be performed with the named input(s) to produce the desired output?
end
```

It is recommended that students use pair programming (one computer for each pair of students) for this exercise. Four eyes are better than two for catching mistakes and bugs, and it likely helps keep students on task on the computer. Students should switch roles periodically, to avoid one student doing all the typing while the other watches.

## The Function Design

A function design is a template students can use to help create custom functions in Pyret. Each design has five sections:



### Function Design

**Defined Identifiers**

________ = ________ # ________
Identifier Value Units

________ = ________ # ________
Identifier Value Units

________ = ________ # ________
Identifier Value Units

**Physical Interpretation**

What will the input(s) of your function be? ____________________ (ex: side length)
What will the units of each input be? _______________________ (ex: meters)
What will the output be? ______________________________ (ex: area)
What will the unit of the output be? _______________________ (ex: square meters)

**Contract and Purpose Statement**

________ :: ________ -> ________
Function Name Domain (Input) Type(s) Range (Output) Type

______________________________
What does the function do? (The function consumes ______ and produces _______.)

**Examples**

```
examples:
    ________(________) is ____________________
    Function Name  Example Input(s)   What calculation(s) must be performed?
    ________(________) is ____________________
    ________(________) is ____________________
end
```

**Definition**

```
fun ________(________________):
    Function Name  Input Name(s)
    ______________________________________
    What calculation must be performed with the named input(s) to produce the desired output?
end
```

### Defined Identifiers

- Students should identify which conditions of the system are necessary to identify for the function to work properly. These are not part of the function itself, but are called during the use of the function, and so should be noted.
- *For Activity 1, we will need to define the metronome tick rate,* `delta-t`.

### Physical Interpretation

- Students should note each input of their function, along with the appropriate units, and the expected output/units. This is the only portion of the function design which will not be translated to Pyret code. It is meant to establish a connection between the programming exercise and the physical phenomenon they are attempting to simulate.
- *For Activity 1, the current position is the input, and the output is the next position of the buggy (i.e., the position of the buggy at the next tick). Both have the units used in Lab 1.*

### Contract and Purpose Statement

- **Function Name** - Brainstorm function names as a class. What do we want the computer to do each time it calculates the position of the buggy? *(Note: The background code requires the function to be named* `next-x`*.)*
- **Domain (Input) Type(s)** - what is/are the datatype(s) of the input parameter(s)? *For this function, the input (current position) is a Number.*
- **Range (Output) Type** - what is the datatype of the output parameter? *For this function, the output (next position) is a Number.*
- **Purpose Statement** - what does the function do? A good purpose statement can fit the format "[function-name] consumes… and produces …". *This function's purpose statement might read:*

```
# next-x consumes the current position and produces the next
  position
```

## Examples

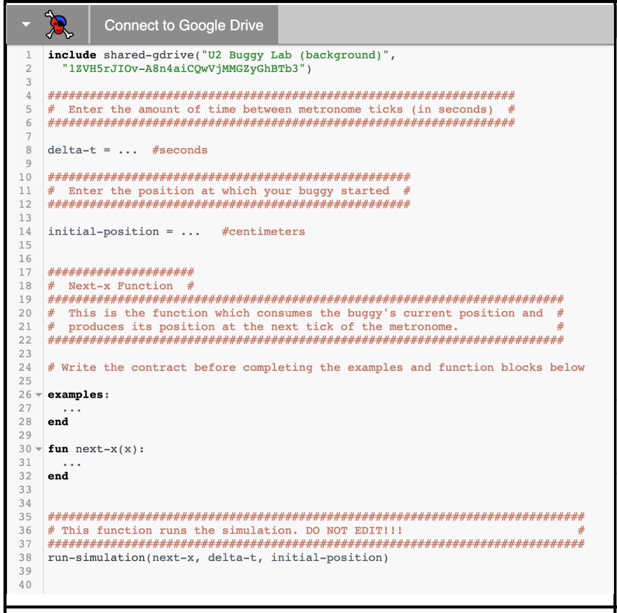
- Examples are concrete and help students design their function. They should be completed *before* the definition of the function, which is more abstract.
- The left side of each example - before the word **`is`** - should be the function name, *immediately followed by* parentheses which include example input arguments, separated by commas (there should be no whitespace between the function name and the parentheses). The number of arguments should match the number of inputs defined in the function contract above.
- The right side of each example - after the word **`is`** - should be written with the *expression that Pyret is evaluating to find the answer.*
- *The example block for this function might read:*

```
examples:
  next-x(45) is 45 + 16
  next-x(77) is 77 + 16
  next-x(109) is 109 + 16
end
```

## Definition

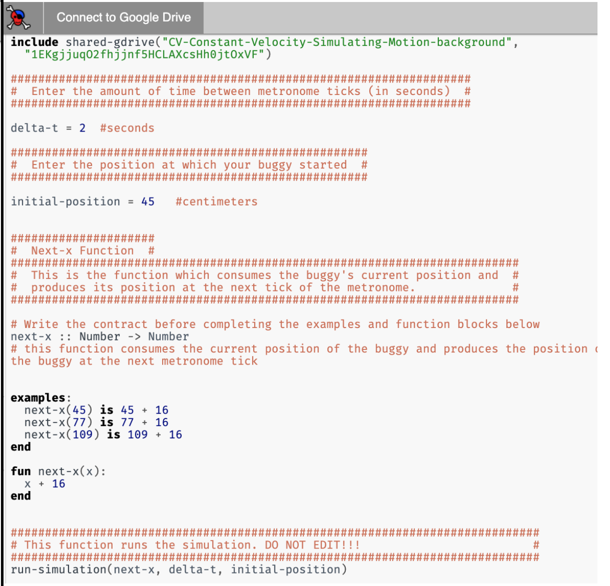
- Have students circle the values in the expression (after **`is`**) that change in each example. These are the arguments of their function; they should name them - the name(s) of the arguments are the function's parameter(s). *In Activity 1, the parameter should be named something like x, position, current-position, x-now, etc. Pyret does not care what the parameter is named, but it should be something that helps the human understand the function and how it is used.*
- Now students write the function with the correct arguments named. Then, write the general expression for the process the function follows, following the pattern of the examples but with the argument(s) named rather than a concrete value.
- *The definition block for this function might read:*

```
fun next-x(x):
  x + 16
end
```



```
include shared-gdrive("U2 Buggy Lab (background)",
  "1ZVH5rJIOv-A8n4aiCQwVjMMGZyGhBTb3")

###################################################################
#  Enter the amount of time between metronome ticks (in seconds)  #
###################################################################

delta-t = ...  #seconds

####################################################
#  Enter the position at which your buggy started  #
####################################################

initial-position = ...   #centimeters


#####################
#  Next-x Function  #
##########################################################################
#  This is the function which consumes the buggy's current position and  #
#  produces its position at the next tick of the metronome.               #
##########################################################################

# Write the contract before completing the examples and function blocks below

examples:
  ...
end

fun next-x(x):
  ...
end


#############################################################################
# This function runs the simulation. DO NOT EDIT!!!                         #
#############################################################################
run-simulation(next-x, delta-t, initial-position)
```

*Unit 2 Activity 1 Starter Code Template*



```
include shared-gdrive("CV-Constant-Velocity-Simulating-Motion-background",
  "1EKgjjuqO2fhjjnf5HCLAXcsHh0jtOxVF")

###################################################################
#  Enter the amount of time between metronome ticks (in seconds)  #
###################################################################

delta-t = 2  #seconds

####################################################
#  Enter the position at which your buggy started  #
####################################################

initial-position = 45   #centimeters


#####################
#  Next-x Function  #
##########################################################################
#  This is the function which consumes the buggy's current position and  #
#  produces its position at the next tick of the metronome.               #
##########################################################################

# Write the contract before completing the examples and function blocks below
next-x :: Number -> Number
# this function consumes the current position of the buggy and produces the position o
the buggy at the next metronome tick


examples:
  next-x(45) is 45 + 16
  next-x(77) is 77 + 16
  next-x(109) is 109 + 16
end

fun next-x(x):
  x + 16
end


#############################################################################
# This function runs the simulation. DO NOT EDIT!!!                         #
#############################################################################
run-simulation(next-x, delta-t, initial-position)
```

*Unit 2 Activity 1 Example Completed Code*

## Function Design Whiteboarding Exercise - “Where’d you get that?”

After students write their function designs, have them defend their work to each other. This exercise encourages students to explain their reasoning in constructing a function and can be used in future situations where they might get stuck and can help each other troubleshoot.

Student A questions Student B:

1. Students turn to a different partner than they worked with on the function design.
2. Student B folds over their function design so that only the bottom two sections are showing. Student A starts at the bottom of Student B’s paper and section by section asks their partner “Where did you get that?” as they point to different parts of the work.
3. The responding student must provide an answer from the previous section of the function design. For example, the first round Student A asks about the Function Definition, and Student B must provide an answer from the Examples.
4. Then Student B reveals the next section upwards, back at the top of their paper, so they can no longer see the functions prompt.
5. Student A now asks, “Where did you get that?” from the Examples section and Student B must provide an answer from the Contract & Purpose Statement.
6. Continue revealing section by section through to the top. Then switch partners.

It is important that students pay attention to the `#`, `:`, `->`, **`is`**, and **`end`**, so that they have the proper syntax. Establish the culture of reading the feedback, thinking about what it is telling you and trying to debug. Don’t be afraid to let students make mistakes in their code. They learn more from getting feedback from Pyret than from getting everything right the first time.

Students need to circle each variable and draw an arrow to where it was defined. This should be in parentheses after the function name. For this function, the input/initial positions should be circled as they will become the `x` in the function.

Some conventions to clarify:

- # indicates a comment – this is recognized as text rather than code (comments appear red in Pyret)
- **`is`** gets used in examples blocks. It is a testing expression which means that two things should be equal.
- `->` can be read as “produces”.
- **`end`** is used to indicate the end of an examples block or function.

Start with students setting delta-t based on what number was used by the metronome such as 2 for 30 bpm or 1 for 60 bpm.

## Activity 2 — Advanced Simulated Motion

### Resources

- Unit 2 Activity 2: Advanced Simulated Motion

### Introduction

### Part I

Using the receipt tape upon which they recorded their data, another visual representation can be formed, that of a motion map.

The points which students marked on the receipt tape represent the position of the buggy for each tick of the metronome. For the motion map, these points will continue to represent the position of the buggy for each tick. Since the term already established for *change* is delta, as in `delta-t`. We will also refer to the change in position between points as *delta-x*. (Note: To avoid confusion due to the similarity in terms, we will avoid the use of the word *displacement* until later in this unit.)

Ask students to identify the starting position of the buggy and the direction in which it moved. How do they know this? The points show the position of the buggy for each tick, but they do not demonstrate starting position nor direction of travel. Ask students to start at the first position marked on the receipt tape. Since this was the position marked when the buggy was released, this is the position at a time of zero ticks. Students can mark this with *tick = 0*. From there, students should mark the direction of the change in position with an arrow. The tail of the arrow begins at x and points to the next x. *It is not necessary that the arrowheads connect to the next point, but rather that the length of each arrow remains the same to demonstrate that the velocity is constant.*

### Part II

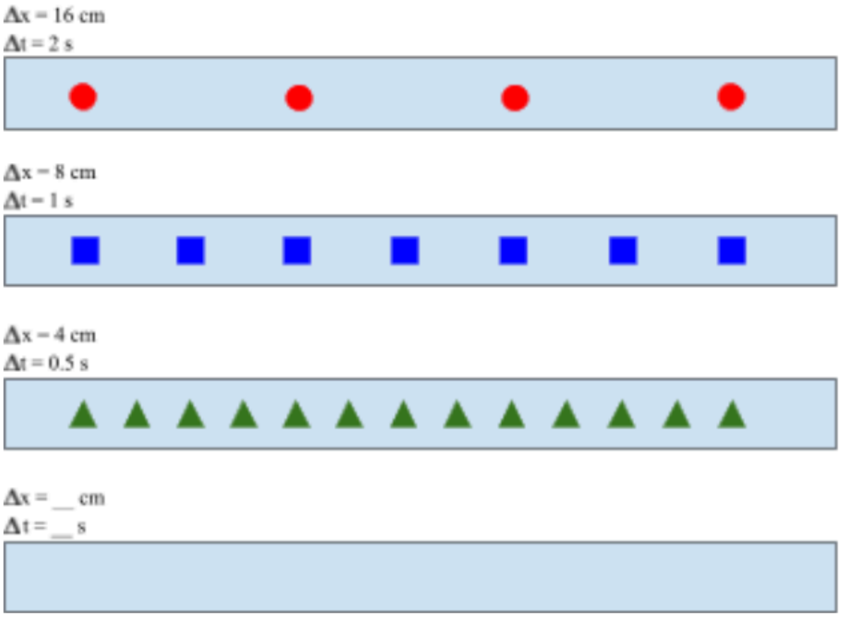
Have students transfer their receipt tape position dots at 2 second intervals onto a whiteboard. Students should identify the ‘change in position’ between dots and record it above their 'motion map'. Then have students identify the 'change in time' between dots (delta-t = tick-rate) and record it above their 'motion map' (as shown below).

Ask students if the diagram would look different if we were to change the metronome to a tick every one second. How so? They need to realize that even though the dot pattern looks different the buggy's motion does not change. Ask students to predict and draw where the dots would be located if the tick rate of the metronome were every 1 second. Repeat the change in position notation on the receipt tape.

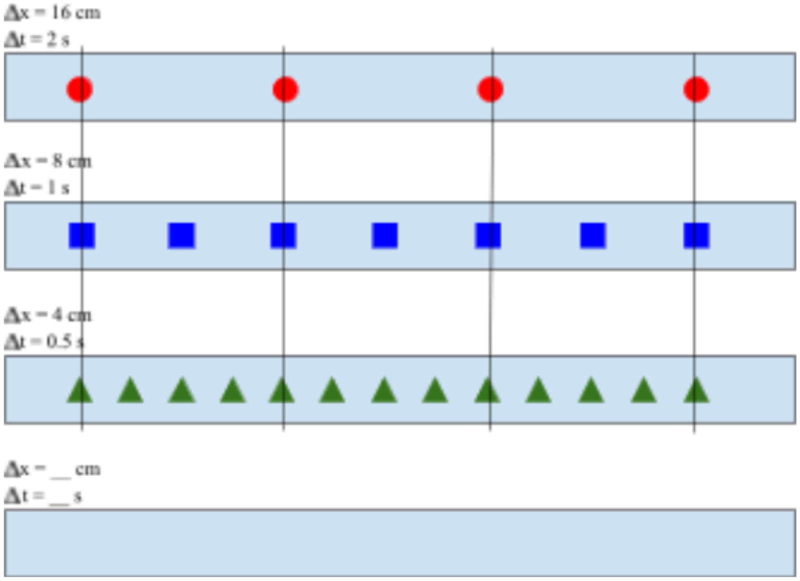
Suggested line of questioning: "What is similar between the new motion map and the first one? What is different? etc."

Ask students to predict and draw where the dots would be located if the tick rate were 0.5 seconds… Repeat above.

The goal is to have students realize multiple times that the speed of the car is the same and the reason the diagrams are different is due to the independent selection of the time between ticks. At this point it is valuable to have the students create a ratio *as a fraction* of the delta-x to the delta-t. Have them do this next to each motion map. They will see that once they simplify the ratio they will get the same values. Question them about the meaning of that ratio with the goal of ultimately getting them to realize that "for every 1 second *change in time,* the buggy *changes position by ____ cm*."



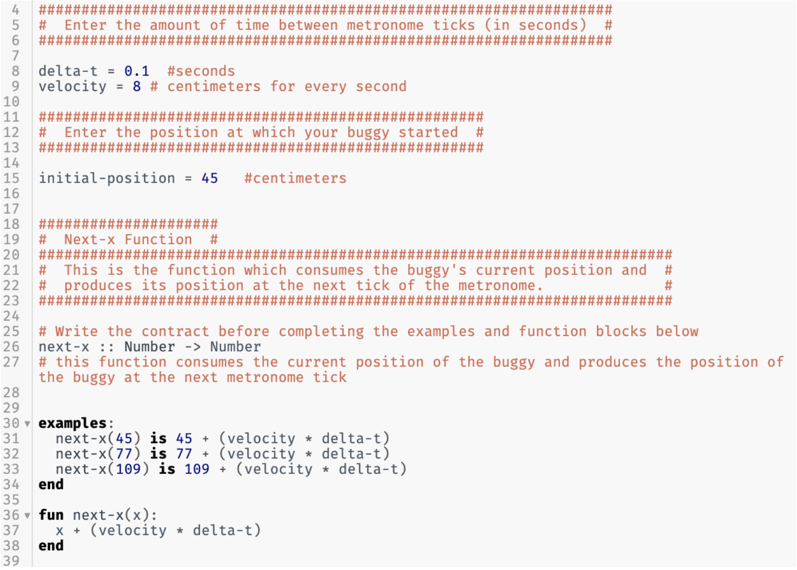
Have students write in the position and clock reading values for the first row of dots and identify *where* the other representations would show the position at those same clock readings. Students should recognize that for all representations, the dots at each 2 second interval are exactly aligned, as shown in the illustration below.



## Part III

Now students are ready to approach this back in the Pyret simulation they used in Activity 1. Have them follow the Activity to see that Pyret gives different positions at the same clock reading if a different delta-t value is given. They need to realize that the delta-t is needed as part of the calculation for change in position. This may be challenging as the kids may do the calculation without even thinking about what they did. The more examples and discussion they do, the more likely they will identify the correct calculation. **Students will need a clean copy of the Function Design to complete this activity.**

The goal of the activity is for them to realize that the change in position can be calculated using the velocity of the buggy and the delta-t.



```
###################################################################
#  Enter the amount of time between metronome ticks (in seconds)  #
###################################################################

delta-t = 0.1  #seconds
velocity = 8 # centimeters for every second

####################################################
#  Enter the position at which your buggy started  #
####################################################

initial-position = 45   #centimeters


#####################
#  Next-x Function  #
###########################################################################
#  This is the function which consumes the buggy's current position and  #
#  produces its position at the next tick of the metronome.               #
###########################################################################

# Write the contract before completing the examples and function blocks below
next-x :: Number -> Number
# this function consumes the current position of the buggy and produces the position of the buggy at the next metronome tick


examples:
  next-x(45) is 45 + (velocity * delta-t)
  next-x(77) is 77 + (velocity * delta-t)
  next-x(109) is 109 + (velocity * delta-t)
end

fun next-x(x):
  x + (velocity * delta-t)
end
```

The final outcome of the function that the students define should be something along the lines of:

Note: this version has defined identifiers of velocity and delta-t, which is a style of programming that we will be using moving forward, so having students recognize that the velocity and delta-t are unique values for EACH RUN of the program, but do not change for a given run, and therefore should be defined to be easily varied if different tick-rates or velocity are simulated.

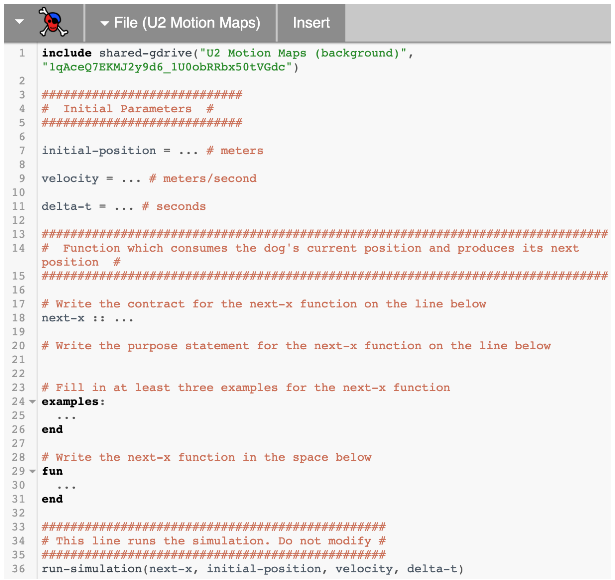
# Worksheet 1 — Motion Maps

## Resources

- The student code can be found at https://tinyurl.com/U2-Motion-Map
- Unit 2 Worksheet 1: Motion Maps

This worksheet provides students with practice creating and interpreting motion maps. Students will use the `next-x(x)` function to create a visual representation of the motion of a dog. The output will include an image of a motion map for the dog. Students can then change the initial conditions to produce a variety of motion maps.

To strengthen the connections between contract, examples, function and motion maps, students should create their motion maps on whiteboards or paper prior to running the program and use the output to check their thinking.



```
include shared-gdrive("U2 Motion Maps (background)",
"1qAceQ7EKMJ2y9d6_1U0obRRbx50tVGdc")

############################
#  Initial Parameters  #
############################

initial-position = ... # meters

velocity = ... # meters/second

delta-t = ... # seconds

###############################################################################
#  Function which consumes the dog's current position and produces its next
position  #
###############################################################################

# Write the contract for the next-x function on the line below
next-x :: ...

# Write the purpose statement for the next-x function on the line below


# Fill in at least three examples for the next-x function
examples:
  ...
end

# Write the next-x function in the space below
fun
  ...
end

################################################
# This line runs the simulation. Do not modify #
################################################
run-simulation(next-x, initial-position, velocity, delta-t)
```

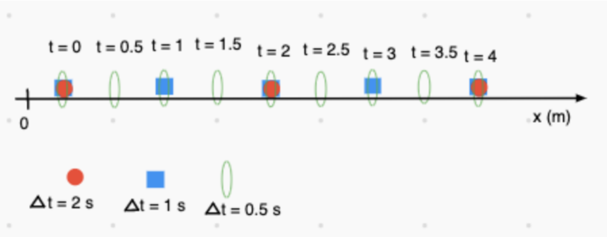
The fraction of each “delta-x” / “delta-t” is the same value, regardless of the tick rate. Emphasize that this iterative nature of calculations is one of the primary values of computational science. Smaller Δt gives smoother motion that is a better approximation of reality but takes more computational time (hence the invention of supercomputers). Pixar uses a Δt that is small enough for your eyes to see fluid motion.

Climate scientists use large Δt often because they have a very complex system to represent, and they want to go many steps into the future.

## Worksheet 2 — Position-Time Graphs

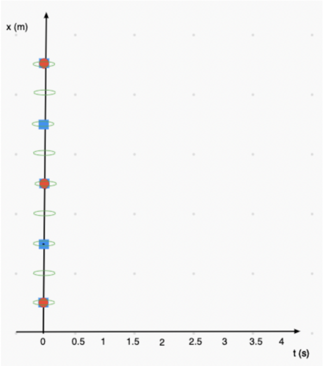
### Resources

- Unit 2 Worksheet 2: Position-Time Graphs

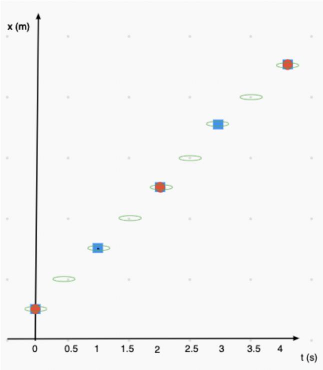


To build the position - time graph use a motion map like the one shown at right with 'post-it' notes to represent the position at various clock readings, based on various tick rates, as shown.

Discuss independent vs dependent variables to lead students to the realization that the output of position data is directly dependent on the delta-t value (aka, tick rate) we choose for the metronome. Rotating the position axis will allow for the time axis to be expanded horizontally as the 'independent variable' and students will then physically move the post-its to their appropriate place based on their clock reading, making position the 'dependent variable'.



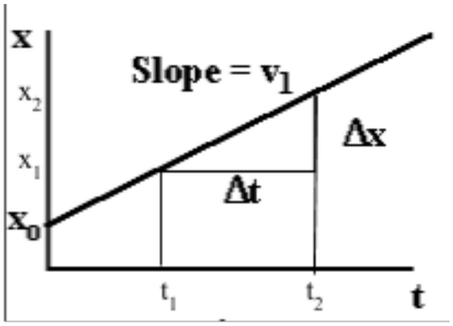
====>
====>
====>
====>
====>
====>
====>
====>
====>



Suggested line of questioning: "What is the same about all the points for each tick rate?"
Goal: All points sit on the same line.

Graphically: Using the graph to the right, we will look at how to build the value for next-*x*.

$$x_2 = x_1 + \Delta x$$



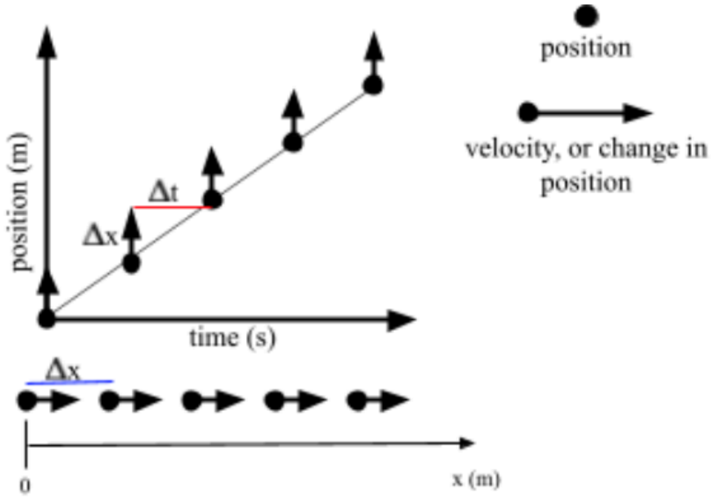
Using the definition of average velocity, we know that:

$$\bar{v} = \frac{\Delta \vec{x}}{\Delta t}$$

$$\Delta \vec{x} = \bar{v} * \Delta t$$

So, $x_2 = x_1 + (v * \Delta t)$

If students think of the final position as the position at the end of the time interval, and the initial position as the current position, it follows that the current position is x, and the final position next-x.



The motion map illustrates the 'change in position' piece with the 'vector', while the tick rate determines the change in time.

The *average* velocity is defined as the change in position divided by the change in time.

This worksheet asks students to apply the learning gained from the whiteboard discussion… specifically the intercept of the graph, as well as the slope meaning. Students should graph both a positive motion buggy and a negative direction buggy.

## Worksheet 3 — Two Bicycles

### Resources

- Unit 2 Worksheet 3: Two Bicycles

This worksheet is quasi-quantitative and forces students to pay attention to other salient attributes of the graph that can help them describe the motion of the cyclists. Some of these attributes are:

1. The intersection of the two lines represents a meeting point, the location of where the cyclists are in the same place at the same time.
2. The steepness of the graph is an indicator of the cyclists' speed.

Use the representations on this worksheet to introduce the construction of velocity-time graphs. Question 1d, for example, asks if the velocities of the two bicycles are equal at any time. While not equal, each bicycle has a constant positive velocity. This idea will be reinforced in the graph matching activity.

# Lab 2 — Graph Matching with Motion Detectors

## Resources

- Unit 2 Lab 2: Graph Matching with Motion Detectors

## Apparatus

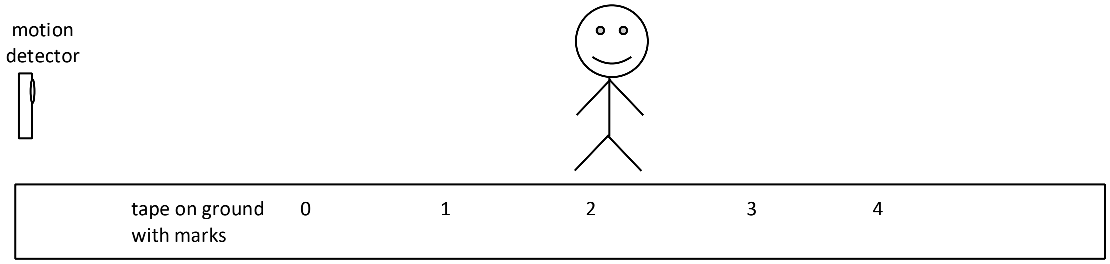
- Motion Detectors - either a wireless Motion Detector connected to a computer or a regular Motion Detector connected to an interface.
- masking tape

## Overview

The purpose of this activity is to give the students a kinesthetic experience in the relationships between verbal, graphical and diagrammatic representations of motion.

## Pre-Lab Discussion

Show students one lab station.



Explain how the motion detector can be used to determine a target object's position and teach students how to zero the detector.

## Lab Performance Notes

1. You will need to clear as much space as possible for the lab. Desks and chairs need to be pushed aside or removed from the room.
2. Smoother graphs can be obtained by:
   a. Having students hold a whiteboard in front of them to use as a target while they move.
   b. Setting the motion sensor for "long range." On Pasco detectors, there is a switch on the detector and the "car" symbol represents short-range, and the "person" represents the long-range setting. For Vernier detectors, the "car" symbol represents the short-range, and the "basketball" represents the long-range setting.
3. One of the biggest mistakes that students make when doing this lab is zeroing the detector from whatever position they are standing when they begin each trial. Make it clear that students only need to zero the detector once—when they are standing on the floor at the 0 mark.

## Post-Lab Discussion

There are sufficient problems for each group to prepare a whiteboard and be able to present the multiple representations of the motion described by the position-time graph.

# Activity 3 — Multiple Objects

## Resources

- Unit 2 Activity 3: Multiple Objects
- Student Code #1: https://tinyurl.com/U2-Multiple-Objects-1
- Student Code #2: https://tinyurl.com/U2-Multiple-Objects-2
- Students will need **THREE** Function Design copies for this activity.

The goal of this activity is to get students to understand that the `next-x` function they have been using in their simulations represents a general rule of motion which applies to all moving objects, rather than a specific rule of motion which moves only a single object.

This change is both an important conceptual step in the understanding of motion and an important milestone in the coding of motion. In order to accomplish this, students must write their `next-x` function differently than they have in the past. Previously, students wrote `next-x` functions which took only one input, the current position `x`.

```
delta-t = ... #s
v = ... #cm for every s

next-x :: Number -> Number
fun next-x(x):
  x + (v * delta-t)
end
```

This worked in the past because there was only one velocity which the program could use to calculate the object's next position. Once you have two objects moving in the simulation, however, the program does not just need to know what the object's current position is, but also which velocity it is supposed to use in order to calculate the next position. As such, the function must now take two inputs, the current position `x` and the velocity to use `v`.

```
delta-t = ... #s

next-x :: Number, Number -> Number
fun next-x(x, v):
  x + (v * delta-t)

end
```

## Reactor Role-Play Exercise – Reactor Play

Students may be confused as to how their program knows which object to move and how to move the object. Engaging students in a role-play exercise where they play the parts of the reactor could help to alleviate some of the confusion.

This exercise could be conducted as the transition from the one-argument to two-argument next-x function, to show students how adding this second argument modifies the function and allows us to simulate the motion of multiple objects using just one next-x function.

**Roles:**

(Multiple roles can be taken by the same student in small groups but keep the `next-x` functions in different people's hands.)

- Programmer
- `draw` function
- initial conditions/parameters
- reactor
- `next-x-black` function
- `next-x-grey` function
- `stop` function

**Before the Reactor Play begins:**

- **Programmer:** Create a whiteboard for each of the following (these can also be written by the teacher to speed up the process). These whiteboards will then be handed to the students playing the matching roles:
    - `next-x-black`
        - i.e., `next-x-black(x): x + (v-black * delta-t)`
    - `next-x-grey`
    - initial conditions
    - stop conditions
        - When should the reactor stop?
- **`draw` function:** draws a background motion map on a whiteboard.
- **Reactor**: Create a whiteboard with the table shown below:

| x-black | x-grey |
|---|---|
|  |  |

    - Reactor asks the initial conditions person how to fill in the reactor table to start.

    - Reactor directs the `draw` function to locate the position of each runner on the background motion map whiteboard. (This could be done at the front of the classroom rather than on a small whiteboard.)

**Rules to update the motion map whiteboard**:

On each tick, the reactor will:

- Ask `next-x-black` function what the new position for x-black is. Erase the current value and enter the next value in table.
- Ask `next-x-grey` function what the new position for x-grey is. Erase the current value and enter the next value in table.
- Direct `draw` function to re-draw the motion map – remove dot from previous location and place a new dot.
- Ask `stop` function if the reactor should stop. If the stop conditions have not been met, then the process repeats.

On each tick, the `next-x-black` and `next-x-grey` functions will:

- use information provided by the reactor and the initial conditions/parameters to calculate the next position using the function definition.

**Round 1:**

This round will repeat the multiple objects simulation with the one-argument function. In this round, the reactor asks one person, the `next-x-black` function, to update the position of the runner in black and a second person, the `next-x-grey` function, to update the position of the runner in grey.

**Initial conditions/parameters:**

```
delta-t = 1 #seconds

x-black-init = 4 #m

x-grey-init = 2 #m

v-black = 2 #m/s

v-grey = 3 #m/s
```

**Reactor:**

| x-black | x-grey |
|---|---|
|  |  |

**`next-x-black` function:**

```
next-x-black(x): x + (v-black * delta-t)
```

**`next-x-grey` function:**

```
next-x-grey(x): x + (v-grey * delta-t)
```

**`stop` function:**

Stop when either runner reaches 20 m or higher, or when either runner passes to the negatives

After completing Round 1, students should be directed to examine what information either `next-x` function needs to run and where it looks for that information. Each function looks to the reactor for the current position of the object, and the initial conditions for `delta-t` and the velocity of the object. Since each function needs to look somewhere for all of the information it needs, could one function be used to complete the calculation? What calculation would this function use? Where would this function look for the information needs to update the position of each runner?

---

**Round 2:**

This round will repeat the multiple objects simulation with the two-argument function. In this round, the reactor asks only one person, the `next-x` function, to update the positions of the runners on each tick. The background reactor needs to know (and it does) to keep the `x-black` and `v-black` information together when asking the `next-x` function to calculate the `next-x` of the runner in black, and the `x-grey` and `v-grey` together when calculating the `next-x` of the runner in grey.

**Initial conditions/parameters:**

```
delta-t = 1 #seconds

x-black-init = 4 #m

x-grey-init = 2 #m

v-black = 2 #m/s

v-grey = 3 #m/s
```

**Reactor:**

| x-black | v-black | x-grey | v-grey |
|---|---|---|---|
|  |  |  |  |

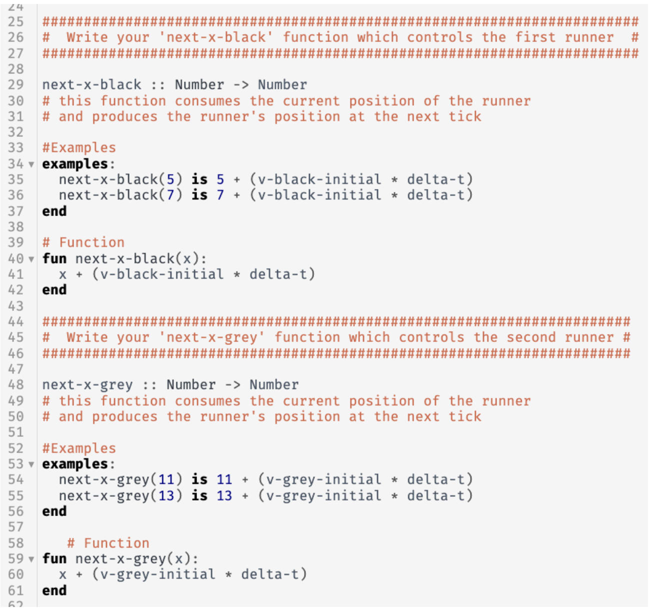
**`next-x` function:**

```
next-x(x, v):  x + (v * delta-t)
```

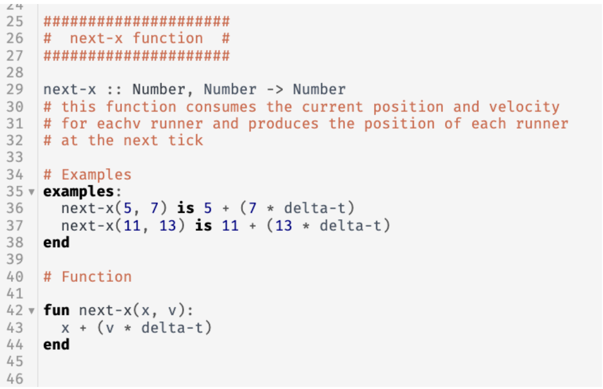
**`stop` function:**

When either runner reaches 20 m or higher, or when either runner passes to the negatives.

The important thing for students to recognize is that the body of the function itself does not change, only the inputs. The phrasing of what the computer “needs to know” in order to calculate the next position as a way of helping to identify the necessary inputs to this new `next-x` function. Thinking of velocity as an additional input to this function will be essential to developing a computational representation of accelerated motion.



```
########################################################################
#  Write your 'next-x-black' function which controls the first runner  #
########################################################################

next-x-black :: Number -> Number
# this function consumes the current position of the runner
# and produces the runner's position at the next tick

#Examples
examples:
  next-x-black(5) is 5 + (v-black-initial * delta-t)
  next-x-black(7) is 7 + (v-black-initial * delta-t)
end

# Function
fun next-x-black(x):
  x + (v-black-initial * delta-t)
end

#######################################################################
#  Write your 'next-x-grey' function which controls the second runner #
#######################################################################

next-x-grey :: Number -> Number
# this function consumes the current position of the runner
# and produces the runner's position at the next tick

#Examples
examples:
  next-x-grey(11) is 11 + (v-grey-initial * delta-t)
  next-x-grey(13) is 13 + (v-grey-initial * delta-t)
end

   # Function
fun next-x-grey(x):
  x + (v-grey-initial * delta-t)
end
```



```
#####################
#  next-x function  #
#####################

next-x :: Number, Number -> Number
# this function consumes the current position and velocity
# for eachv runner and produces the position of each runner
# at the next tick

# Examples
examples:
  next-x(5, 7) is 5 + (7 * delta-t)
  next-x(11, 13) is 11 + (13 * delta-t)
end

# Function

fun next-x(x, v):
  x + (v * delta-t)
end
```

On the left: One-argument function for `next-x-black` and `next-x-grey`

On the right: Two-argument function that generalizes the linear model

# Lab 3 — Colliding Buggies

## Resources

- The student code can be found at https://tinyurl.com/U2-Colliding-Buggies
- Unit 2 Lab 3: Colliding Buggies

*Rationale*: This deployment lab incorporates computational modeling skills into the more "traditional" CVPM representations when students predict the collision point between two cars traveling towards each other or where one car catches up with another slower one. Students will create a computer simulation that will function as a prototype for the phenomena they are trying to model, which is the collision of the two cars.

This is akin to, for example, NASA engineers predicting a landing point on an asteroid for a spacecraft. They have to tweak parameters such as spacecraft velocities, trajectory angles etc. and run these in computer simulations to land the spacecraft safely. Here students will have to know the velocities of each car and write a computer program to simulate the collisions. This is a simpler version of what NASA engineers may do but it emphasizes the role of computational modeling and its uses in scientific endeavors.

In both the physical activity and the simulation, students should be asked to predict the point and time of collision for the two buggies.

## Apparatus

- Fast toy car
- Slow toy car
- Stopwatches or metronome
- approximately 6m of floor space

## Teacher Background

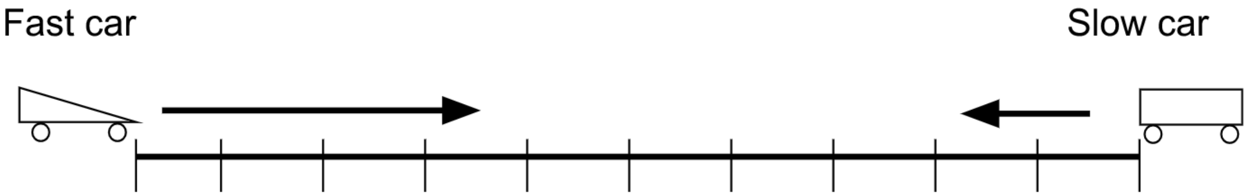
This lab is a more entertaining, more hands-on iteration of the classic "A train leaves from Boston and another leaves from New York" situation. The lab can be done as a whole class lab practicum. The class works as one large group on the problem with little input from the teacher. When the class agrees on an answer, they put the solution (preferably with multiple representations - i.e. graphically, algebraically, and diagrammatically) on the whiteboard. The teacher then calls on one student to present the solution. One part of the entire class grade depends on how well the student presents the solution. The other part of the class grade depends on whether or not the solution works.

Note:

1. If you have enough equipment, instead of having a whole class lab practicum, have different groups of students run the activity separately.
2. You can also have cars play catch up. Different versions of this are:
   a. Slow car starts ahead of fast car and both are released at the same time
   b. Slow car starts ahead of fast car and the slow car is released ahead of time.

## Pre-Lab Discussion

Show the students a diagram of the situation.



1. Explain that the cars will be moving towards each other at two different constant velocities from a known distance apart. The goal is to determine where they will hit (or at least pass each other) by placing a marker on the floor.

2. Explain how the class lab practicum will work. Students will be allowed to work with only one car at a time. Once data has been gathered from one car and students start work on the other, students cannot work with the first car.

3. Students will first predict where the cars will collide using mathematical, graphical, and diagrammatic representations (motion maps). Assign starting positions for each car, and the direction of motion. Though the diagram above shows a head-on collision, another variation can be to have the fast car moving in the same direction as the slow car, resulting in a rear-end collision.

Students might need hints:

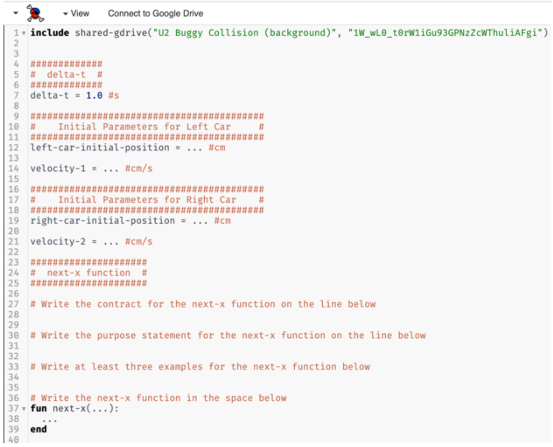
- How could you represent this situation with a graph?
- Is the slope the same for both cars?
- Is the y-intercept the same for both cars?
- Doesn't this look a lot like the bicycle question from worksheet 4?

4. Have students use the function design to write their function. What information will they need to tell the program? The goal is to write:

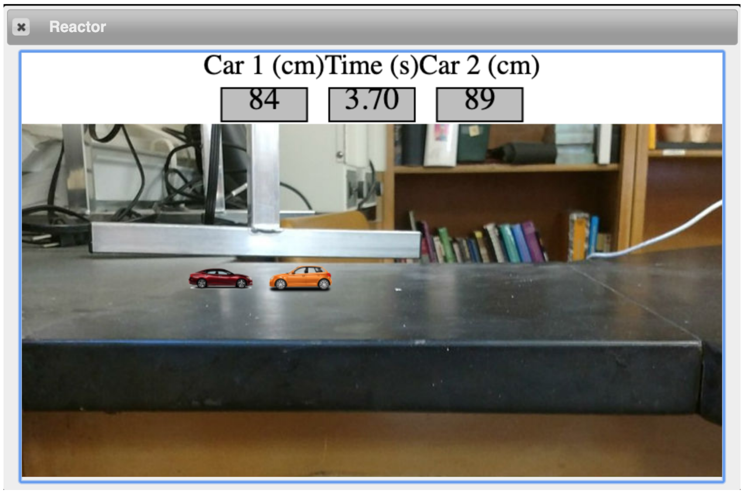
```
fun next-x(x, v):
   x + (v * delta-t)
end
```

with the initial position and velocity of each car set as initial parameters. They will then use Pyret to simulate the collision. This provides students another way to represent their prediction.

This is what students will modify:



```
include shared-gdrive("U2 Buggy Collision (background)", "1W_wL0_t0rW1iGu93GPNzZcWThuliAFgi")


#############
#  delta-t  #
#############
delta-t = 1.0 #s

##########################################
#    Initial Parameters for Left Car     #
##########################################
left-car-initial-position = ... #cm

velocity-1 = ... #cm/s

##########################################
#    Initial Parameters for Right Car    #
##########################################
right-car-initial-position = ... #cm

velocity-2 = ... #cm/s

#####################
#  next-x function  #
#####################

# Write the contract for the next-x function on the line below


# Write the purpose statement for the next-x function on the line below


# Write at least three examples for the next-x function below


# Write the next-x function in the space below
fun next-x(...):
  ...
end

```



Car 1 (cm)Time (s)Car 2 (cm)
84 3.70 89

## Lab Performance Notes

Once predictions have been tested using Pyret, have students run the physical lab. They should use the same starting positions and directions of motion they were assigned.

## Post-Lab Discussion

If students are unable to accurately predict the collision position or clock reading for the two buggies, conduct a post-lab discussion in which they analyze their procedure for possible errors.

# Worksheet 4 — Distance vs. Displacement

## Resources

- Unit 2 Worksheet 4: Distance vs. Displacement

Part 1 is an opportunity for the introduction to the concepts of distance and displacement (these terms may have been introduced during discussion of Worksheet 3 - Two Bicycles). This can be implemented as a class discussion where you project the image on a screen and have students discuss answers. You can also have students work on the worksheet first before discussion.

Have students come to consensus as to who traveled farther. Lead them to think that the answer depends on different definitions of “who traveled farther.” For example, students can say “Dorothy got closer to Oz, but Toto traveled farther.” This provides a mental conflict that highlights the need for two new terms that describe how far something travels: **displacement** and **distance**. Displacement represents the difference between an object’s final position (aka location) and the initial position. Distance represents the “path length” an object traveled from its initial position to its final.

Part 2 provides examples for students to practice these new conceptions. This is suggested as a whiteboard activity. During the board meeting, important points to emphasize are:

- Displacement is either positive or negative
- Distance is always positive

## Reading 1 — Vocabulary of Motion

### Resources

- Unit 2 Reading 1: Vocabulary of Motion

Reading 1 establishes the important differences between scalar (distance and speed) and vector (displacement and velocity) quantities. This common vocabulary will be used throughout the course. This reading can be assigned prior to worksheet 4 or following part 1 of worksheet 4.

This reading can be split into two parts such that number lines, distance and displacement are on one part, average speed and average velocity on the other.

Until now we have avoided the use of the word *displacement* due to its similarity to the word distance. At this point, it is important that students understand the fundamental differences between these two concepts. Displacement will play a larger role in unit 3 but will be introduced as the area between the line and the horizontal axis for the velocity-time graph in Worksheet 5.

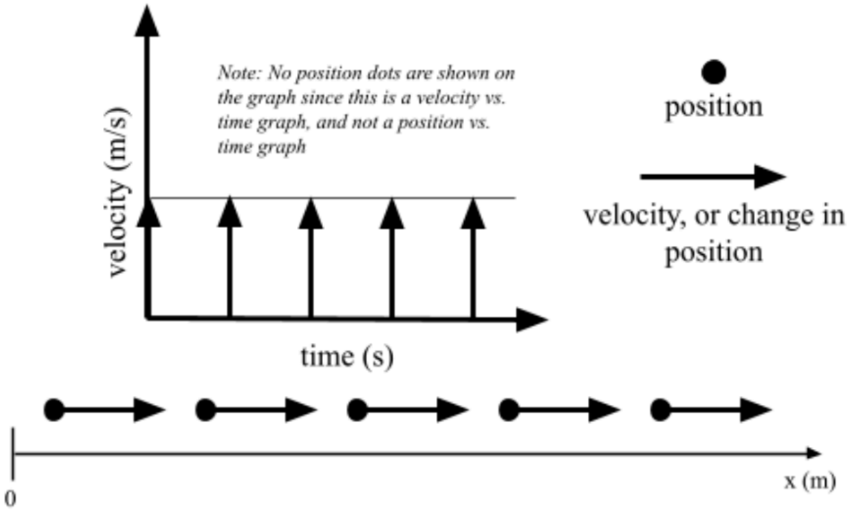
## Worksheet 5 — Velocity-Time Graphs

### Resources

- Unit 2 Worksheet 5: Velocity-Time Graphs

This worksheet introduces how velocity vs. time graphs can be extracted from position vs. time graphs, building upon observations students may have found in the optional motion detector lab.

The connection between motion maps and velocity vs. time graphs can be brought forth during a whiteboard discussion. Using the velocity vs. time graph, arrows can be drawn to show the magnitude (size) of the velocity at each second. Students should note that the length of the arrows remains the same. Since we also indicate velocity with arrows on the motion map, this would mean that the motion map would also have arrows of equal magnitude.


Note: No position dots are shown on the graph since this is a velocity vs. time graph, and not a position vs. time graph
position
velocity, or change in position
velocity (m/s)
time (s)
0
x (m)

## Activity 4 — Multiple Representations of Motion

### Resources

- Unit 2 Activity 4: Multiple Representations of Motion
- Unit 2 Activity 4 Teacher Resource: Multiple Representations of Motion

This activity and worksheet are adapted from 2016 AMTA materials and meant to be whiteboarded. All representations of motion learned thus far are used to construct a multi-dimensional model for the motion of an object. Students should at this point be able to use motion maps, Pyret functions, position-time graphs and velocity-time graphs interchangeably.

A teacher resource is provided in the curricular materials. It is recommended that these or similar data tables be handed to groups, and that they be asked to create the remaining representations (graphs, mathematical, motion map, verbal, contract, examples, function).

## Activity 5 — Rocket Lander Game

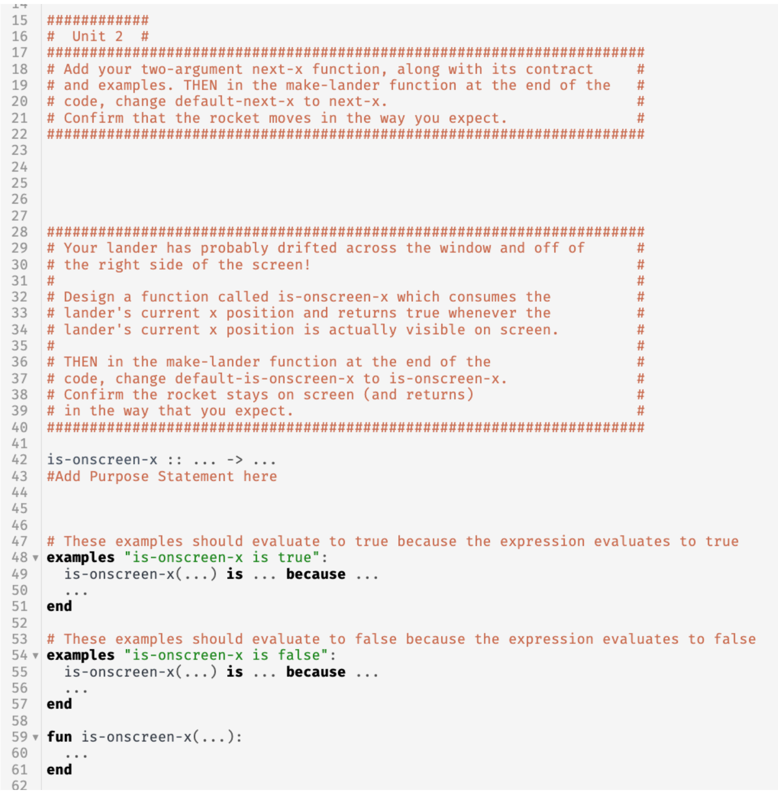
### Resources

- Unit 2 Activity 5: Rocket Lander Game
- The student code can be found at https://tinyurl.com/Rocket-Lander-Game

At the conclusion of this unit students will be adding the first segment of code to their own video game. They previously wrote the program for an object moving with a constant velocity and will use the same format here.

For each unit, students will add only that particular section of the program, replacing it with contracts, purpose statements, examples and functions of the same format as learned earlier in the unit.

For Unit 2, students will create the constant velocity horizontal motion for the rocket by writing the next-x function and keep the rocket onscreen by writing the is-onscreen-x function.



```
############
#  Unit 2  #
######################################################################
# Add your two-argument next-x function, along with its contract     #
# and examples. THEN in the make-lander function at the end of the   #
# code, change default-next-x to next-x.                             #
# Confirm that the rocket moves in the way you expect.               #
######################################################################





######################################################################
# Your lander has probably drifted across the window and off of      #
# the right side of the screen!                                      #
#                                                                    #
# Design a function called is-onscreen-x which consumes the          #
# lander's current x position and returns true whenever the          #
# lander's current x position is actually visible on screen.         #
#                                                                    #
# THEN in the make-lander function at the end of the                 #
# code, change default-is-onscreen-x to is-onscreen-x.               #
# Confirm the rocket stays on screen (and returns)                   #
# in the way that you expect.                                        #
######################################################################

is-onscreen-x :: ... -> ...
#Add Purpose Statement here



# These examples should evaluate to true because the expression evaluates to true
examples "is-onscreen-x is true":
  is-onscreen-x(...) is ... because ...
  ...
end

# These examples should evaluate to false because the expression evaluates to false
examples "is-onscreen-x is false":
  is-onscreen-x(...) is ... because ...
  ...
end

fun is-onscreen-x(...):
  ...
end
```

Once the function for next-x has been written in the appropriate location within the starter program, students need to change the inputs to the `make-lander` function at the end of the program from `default-next-x` to `next-x` so *their* function can be called, rather than the ‘default’.

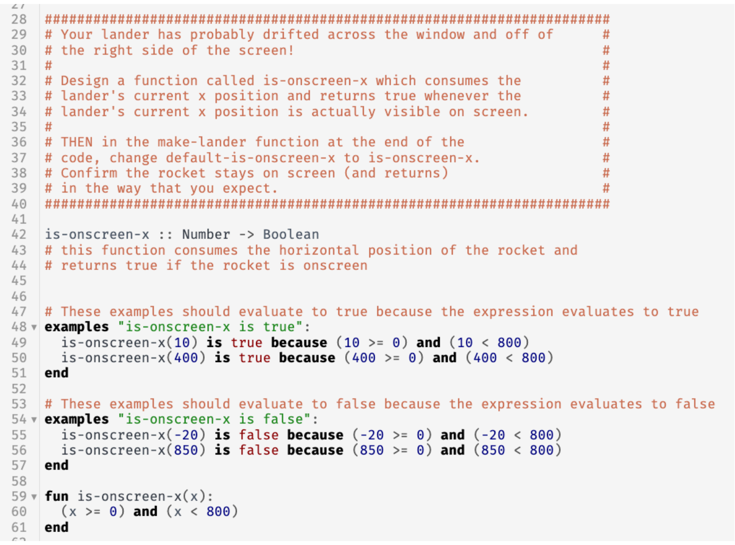
**New Programming Skill: Boolean expressions**

At this point, students can add a function to their rocket lander game to determine if the lander is onscreen.

Students will *revise* the existing `is-onscreen-x` function to test whether the rocket is onscreen (using a Boolean operator… e.g., <, >, ==, >=, <=, <> ). This expression, comparing two sides using a Boolean operator, is a Boolean expression. The expression should evaluate to a value of true when the rocket is onscreen, and to a value of false when the rocket is not onscreen. The examples can be expanded to include *named* examples blocks. These names will be used in any feedback received to make identifying failed examples easier.

When the examples block is checked in comparison to the function block, the check includes a *left side*, a *right side*, and an *explanation*. The *left side* evaluates the function using the example inputs. The *right side* of the example is the Boolean value of `true` or `false` stated after the **`is`**. The *explanation* is the Boolean expression, using the example inputs, written after the **`because`**. To pass all tests the left side, right side, and explanation should all evaluate to the same Boolean value.

The `is-onscreen-x` function should take in the horizontal position of the rocket and compute a Boolean value to answer whether the rocket is onscreen. Since this is a Boolean expression, students should use the Function Design with Booleans to write the `is-onscreen-x` function. As with the `next-x` function, students will need to change `` `default-is-onscreen-x` `` to simply `` `is-onscreen-x` `` before running their function.



```
######################################################################
# Your lander has probably drifted across the window and off of      #
# the right side of the screen!                                      #
#                                                                    #
# Design a function called is-onscreen-x which consumes the          #
# lander's current x position and returns true whenever the          #
# lander's current x position is actually visible on screen.         #
#                                                                    #
# THEN in the make-lander function at the end of the                 #
# code, change default-is-onscreen-x to is-onscreen-x.               #
# Confirm the rocket stays on screen (and returns)                   #
# in the way that you expect.                                        #
######################################################################

is-onscreen-x :: Number -> Boolean
# this function consumes the horizontal position of the rocket and
# returns true if the rocket is onscreen


# These examples should evaluate to true because the expression evaluates to true
examples "is-onscreen-x is true":
  is-onscreen-x(10) is true because (10 >= 0) and (10 < 800)
  is-onscreen-x(400) is true because (400 >= 0) and (400 < 800)
end

# These examples should evaluate to false because the expression evaluates to false
examples "is-onscreen-x is false":
  is-onscreen-x(-20) is false because (-20 >= 0) and (-20 < 800)
  is-onscreen-x(850) is false because (850 >= 0) and (850 < 800)
end

fun is-onscreen-x(x):
  (x >= 0) and (x < 800)
end
```

## The Model So Far

To summarize the unit, students should create “The Model So Far,” the current model for the motion of an object. The goal is to both review representations of motion and assess student understanding of the current model before progressing to uniform acceleration.

## Resource Index

# Unit 2 Lab 1: Buggy Lab

## Part I

Observe the motion of the buggy and answer the following questions:

1. Does the buggy move? How do you know?

2. Draw a series of state diagrams for the buggy. What quantity or quantities change from diagram to diagram?

## Part II

Each time the metronome ticks, mark the position of the buggy on the receipt tape. Do this for a total of ten ticks. Create a table of the results in the space below.

3. Using the above results, predict the position of the buggy on the eleventh, twelfth and thirteenth ticks of the metronome. Perform an experiment to test these predictions.

| Tick | Predicted Position | Measured Position |
|---|---|---|
| 11 | | |
| 12 | | |
| 13 | | |

4. Does the data follow a pattern? If so, how is this pattern represented in the state diagrams?

# Unit 2 Activity 1: Simulated Motion

## Part I

The next representation of the motion of the buggy will be a computer simulation based on the collected data. To do this, we must first represent the motion of the buggy with a ***function***.

The ***purpose statement*** of this function will read:

```
# Consumes the current position of the buggy and
# produces its position at the next metronome tick.
```

1. What does this function need to take as an input? What does this function produce as an output?

2. The name of this function will be `next-x` since it produces the position of the buggy at the next metronome tick. Write a ***contract*** for `next-x`.

`next-x  ::` ______________ -> ______________

Input "Type"   Output "Type"

3. In the left column of the table draw a state diagram for the buggy with its position labeled. In the middle column write the corresponding Pyret example that consumes the position of the buggy at that state. In the right column draw a state diagram for the next position of the buggy, again with its position labeled.

| Current State Diagram | Pyret Examples | Next State Diagram |
|---|---|---|
| | **`examples:`** | |
| | | |
| | **`end`** | |

4. Given the current position of the buggy, how can the next position be calculated? Write this as an algebraic expression.

5. Obtain a ***Function Design*** and fill it out for this `next-x` function.

## Part II

Load the program https://tinyurl.com/U2-Simulated-Motion

The notation … indicates that text must be added before the program will run. In addition to the information written in the function design, there are two initial parameters that must be given values: `delta-t` and `initial-position.`

6. Read the comments for `delta-t` and `initial-position` and enter the values used during the buggy lab.

7. Use the function design from Part I to complete the `next-x` function in the code.

8. Run the simulation and describe what happens. Is this how you expected your simulation to work? Explain.

# Unit 2 Activity 2: Advanced Simulated Motion

Load your saved buggy simulation from Activity 1.

1. Your simulation output from Activity 1 includes a table of values with the headings tick, time, and position. Copy this table below. What do each of these headings indicate?

   Then change your delta-t to 1 second and run your simulation again. Record the new data in the table below.

| Data table for delta-t = 2 sec | | | | Data table for delta-t = 1 sec | | |
|---|---|---|---|---|---|---|
| tick | time | position | | tick | time | position |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

2. What does tick "2" tell you regarding the buggy? Is the information the same for both tables? Explain why or why not.

3. What does time "4 seconds" tell you regarding the buggy? Is the position the same? Is the tick the same? Explain.

4. As we keep changing the delta-t value, the position information is no longer accurate! Where should your buggy be with the given delta-t? **The third column will be used in question 5.** (Add more delta-t examples until you see a pattern!)

| delta-t (s) | expected change in position (cm) | |
|---|---|---|
| 2 | | |
| 1 | | |
| 0.5 | | |
| 10 | | |
| | | |
| | | |

5. Think back to your whiteboard discussion. We saw that the delta-t we used had an effect on how far our buggy traveled in that time interval. What is the pattern that you noticed in the table? What common calculation could you do that uses the delta-t value and gives us the expected position?

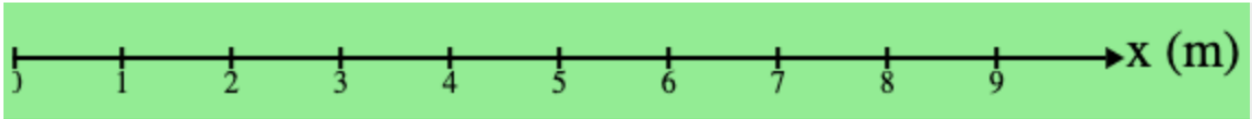
6. We need to change our next-x function so that it returns the correct expected position for each clock reading regardless of the delta-t value. **Our function purpose statement and contract will stay the same as before,** but we need new examples and a new function definition. Complete the new Function Design on paper before you make any changes to your code.

7. At the top of your function design there is a request for defined identifiers. What constant values will you need for your calculation?

8. When you can change your delta-t to any value and the position of the car at a particular clock reading is the same you will know you have successfully modeled the motion of your buggy into a computer simulation. Congratulations!

# Unit 2 Worksheet 1: Motion Maps

Use this url: The student code can be found at https://tinyurl.com/U2-Motion-Map
This simulation shows a dog running at a constant velocity.

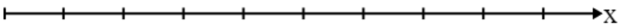
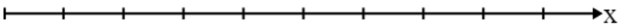
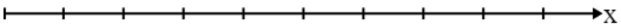
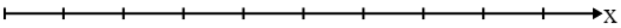
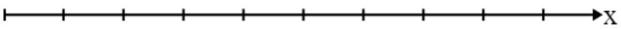
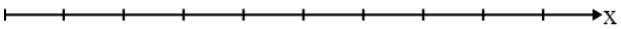


1. Complete the `next-x` function as in previous simulations. Set an initial position and velocity that will make the dog appear on the background above, each mark representing one meter. Sketch the image output shown in the interactions window after the reactor window closes.

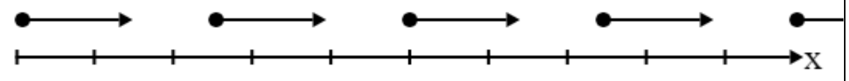
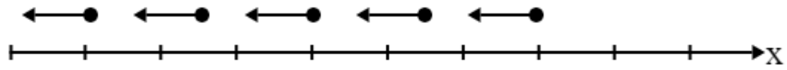
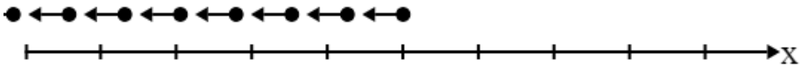
2. Change the velocity and initial position, run the simulation, and sketch the new image output.

3. The images are ***motion maps***. What information about the motion of the dog can be determined from looking at the motion map? How is this information represented?

4. Predict what the motion map will look like for the following initial parameters. Test these predictions with the simulation and draw the outcome in the table below.
   a. $x_o = 1$, $v = 2$ m/s
   b. $x_o = 4.5$, $v = -1.5$ m/s
   c. $x_o = 5$, $v = 1.25$ m/s

| Prediction | Outcome |
|---|---|
| a.<br><br> | <br> |
| b.<br><br> | <br> |
| c.<br><br> | <br> |

5. Determine the initial position and velocity of the dog in each case from the motion maps below. Use the simulation to check your work.

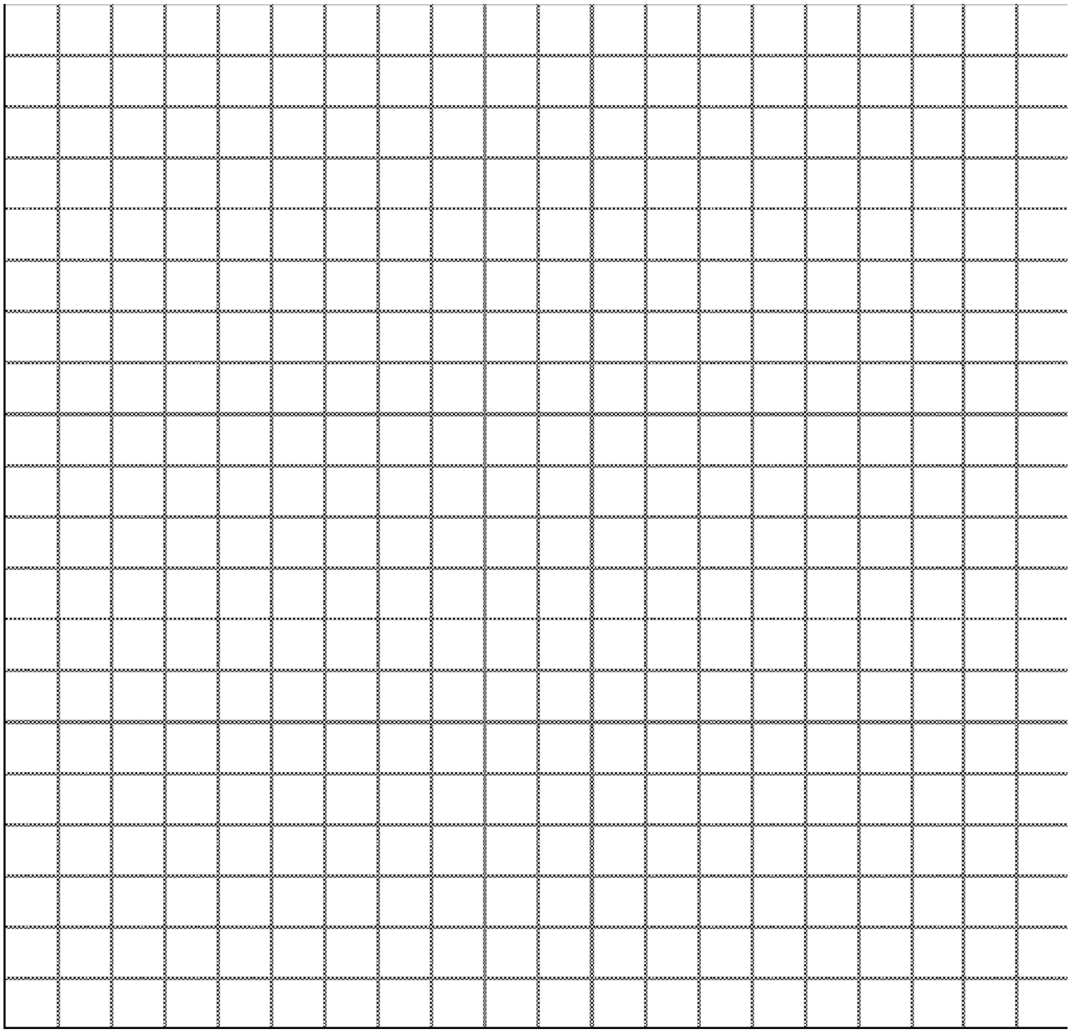
| Motion Map | Initial Conditions |
|---|---|
|  | $x_o$ = ___________,<br>$v$ = ___________ |
|  | $x_o$ = ___________,<br>$v$ = ___________ |
|  | $x_o$ = ___________,<br>$v$ = ___________ |

6. What would change about these motion maps if the position was marked every two seconds, instead of every one second? How do you know?

7. What would change about these motion maps if the position was marked every 0.5 seconds? How do you know?

# Unit 2 Worksheet 2: Position-Time Graphs

1. Plot points of position and time for two buggies on the graph below (you may need to share information from another lab group). Draw a line of best fit for each data set.

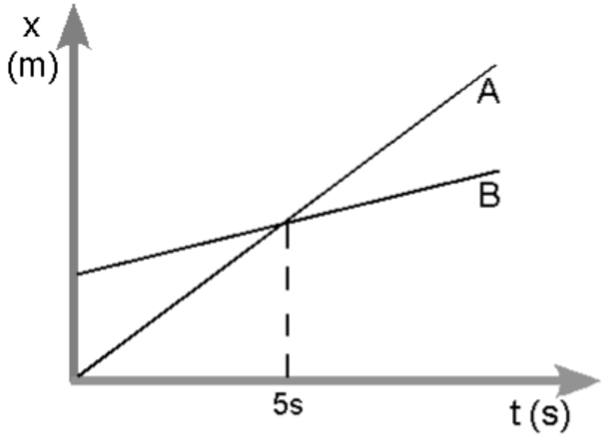


2. What might the vertical intercept of each line represent about the motion of the buggy?

3. What is the slope of each line? What would be the units for this slope? What could it represent for the motion of the buggies?

# Unit 2 Worksheet 3: Two Bicycles

1. Consider the position vs. time graph below for cyclists A and B.



a) Do the cyclists start at the same point? How do you know? If not, which is ahead?

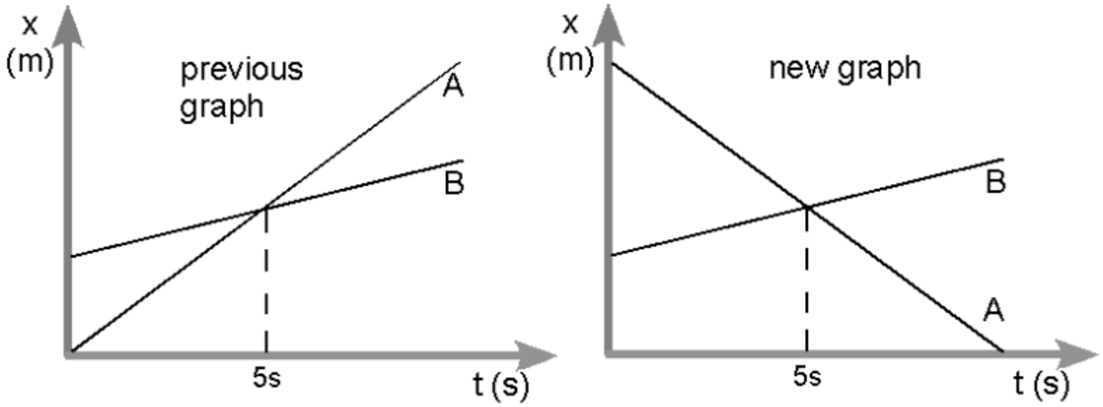
b) At t= 7s, which cyclist is ahead? How do you know?

c) Which cyclist is travelling faster at t = 3s? How do you know?

d) Are their velocities equal at any time? How do you know?

e) What is happening at the intersection of lines A and B?

2. Consider the new position vs. time graph below for cyclists A and B.



a) How does the motion of the cyclist A in the new graph compare to that of A in the previous graph from page one?

b) How does the motion of cyclist B in the new graph compare to that of B in the previous graph?

c) Which cyclist has the greater speed? How do you know?

d) Describe what is happening at the intersection of lines A and B.

e) Which cyclist traveled a greater distance during the first 5 seconds? How do you know?

# Unit 2 Lab 2: Graph Matching with Motion Detectors

For each of the situations listed below, do all of the following:

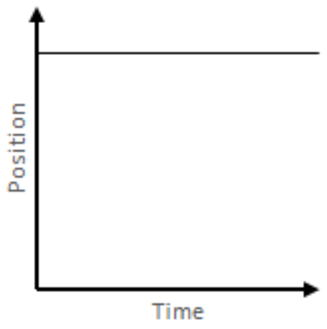
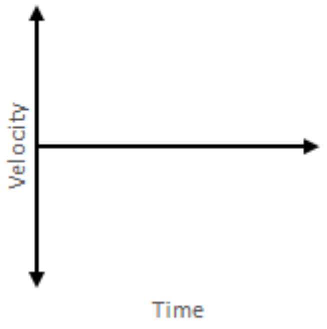
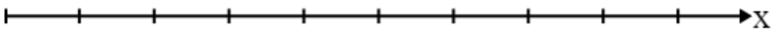
a. Move, relative to the motion detector, so that you produce a position vs. time graph which closely approximates the graph shown.

b. In the space provided, describe how you must move in order to produce the position vs. time graph shown in the space to the right of the velocity vs. time graph.  Be sure to include each of the following in your description: starting position, direction moved, type of motion and relative speed.

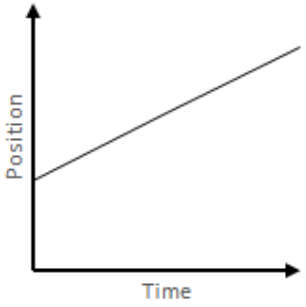
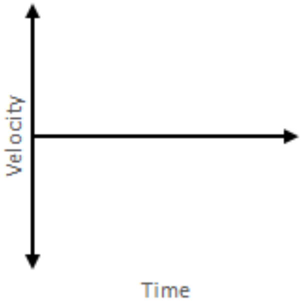
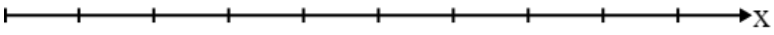
c. On the velocity vs. time axes, sketch the velocity vs. time graph which corresponds to the position vs. time graph shown.

d. In the space provided, sketch the motion map that corresponds to the motion described in the position vs. time graph (Include one dot for every second)

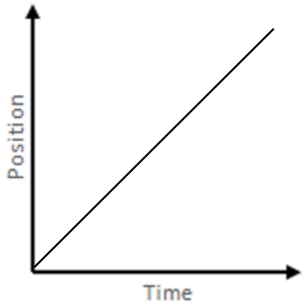
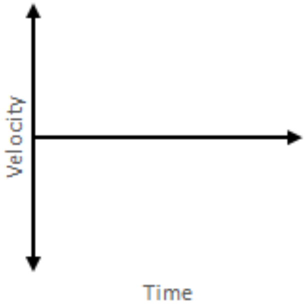
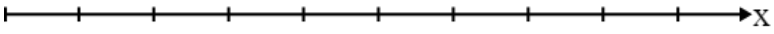
1.

| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

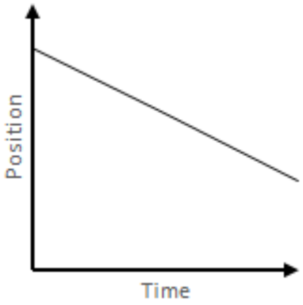
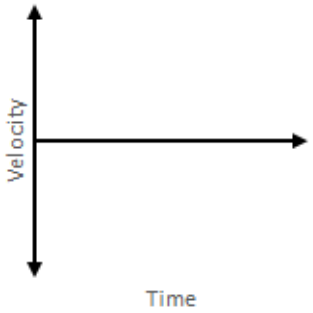
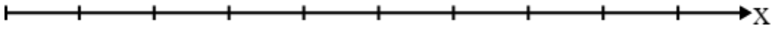
2.

| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

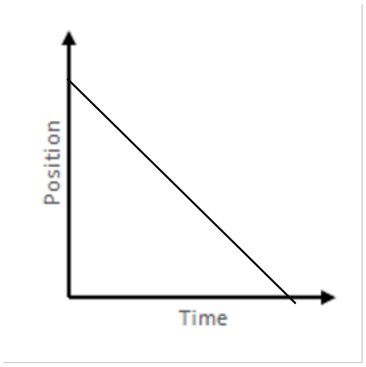
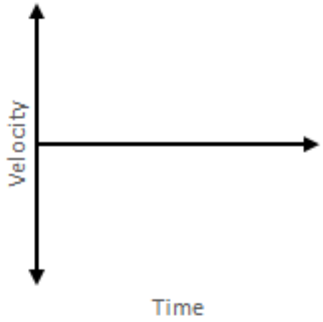
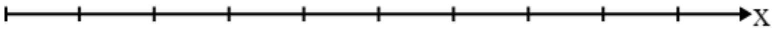
3.

| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

4.

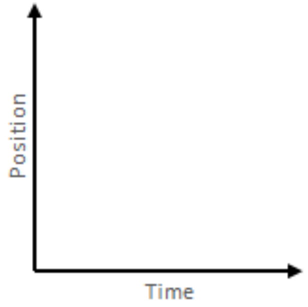
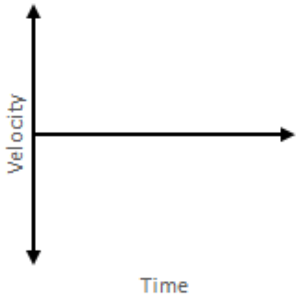
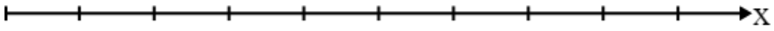
| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

5.

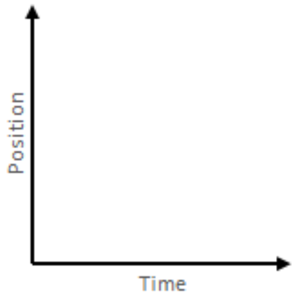
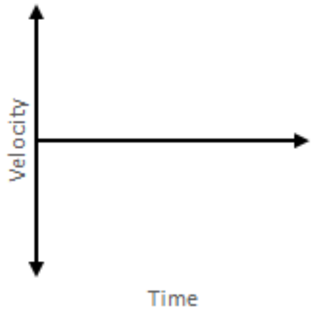
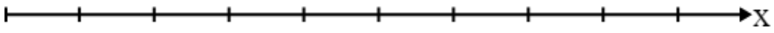
| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

In the following spaces, create a graph to walk and record your results below.

6.

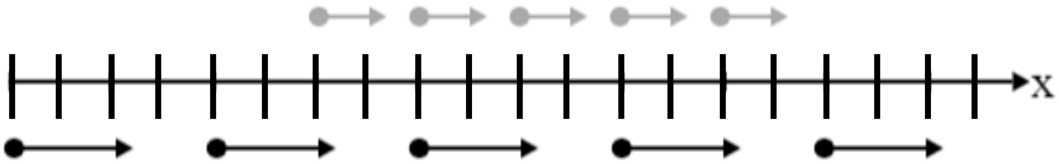
| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

7.

| | |
|---|---|
|  | Written Description |
|  | Motion Map<br> |

# Unit 2 Activity 3: Multiple Objects

The motion map below represents the positions of two runners every 2 seconds. Each mark on the axis represents 1 meter.



1. What information about the runners can be determined from this motion map?

2. Are the two runners ever at the same position at the same time? If so, where? How do you know?

3. Draw a series of state diagrams showing the two runners at the 5 times shown in the motion map.

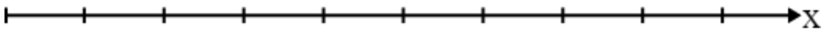
4. What initial conditions and functions would need to be written to simulate the motion of these two runners?

5. Obtain two Function Designs from your teacher. Fill out the function designs for the functions that simulate the motion of each runner.

6. Open the code https://tinyurl.com/U2-Multiple-Objects-1. Fill in the initial conditions and functions for `next-x-black` and `next-x-grey`, then run the simulation. Do the runners meet at the position you determined in #2?

Draw a motion map and identify the necessary initial conditions for each of the situations below, then change the parameters in your simulation and run it to check your predictions.

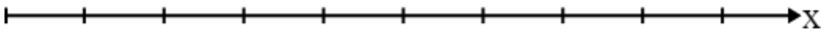
7. The runner in black starts at *x = 2m*, the runner in grey starts at *x = 14m*. The two runners pass each other at *x = 8m*.



$x_{b_i}$ = ______ $x_{g_i}$ = ______

$v_b$ = ______ $v_g$ = ______

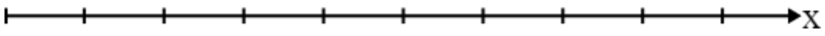
8. The runner in black moves with a velocity of -5 m/s, the runner in grey moves with a velocity of 4 m/s. The two runners pass each other at *x = 9m*.



$x_{b_i}$ = ______ $x_{g_i}$ = ______

$v_b$ = ______ $v_g$ = ______

9. The runner in black moves with a velocity of 4 m/s, the runner the grey starts at *x = 3m*. The runner in black passes the runner in grey at *x = 12m*.



$x_{b_i}$ = ______ $x_{g_i}$ = ______

$v_b$ = ______ $v_g$ = ______

10. Look closely at the functions `next-x-black` and `next-x-grey`. Using a colored pencil, circle the pieces that change from `next-x-black` to `next-x-grey`. What do these pieces have in common?

11. Our goal is to write *one* function `next-x` that simulates the motion of both runners. What information would this function need to consume?

12. What is the contract and purpose statement for this function?

________________ :: ______________________________ -> ___________________

#______________________________________________________________________

13. Using a new Function Design, construct this new-and-improved next-x function. When you have completed the function design, use the following simulation to check your results: https://tinyurl.com/U2-Multiple-Objects-2

# Unit 2 Lab 3: Colliding Buggies

## Part I

In this lab the computational model for constant velocity motion will be used to predict where two cars moving at different speeds will collide. The first step is to collect all the information needed in order to construct a simulation of this scenario.

1. What information about the buggies is needed to predict where they will collide?

2. What procedure will be followed to find the information listed above?

3. Collect data and record it below. Show any calculations.

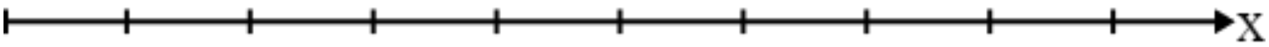
## Part II

4. Record the initial positions of the buggies and the directions they will travel in the table below.

| | Car 1 | Car 2 |
|---|---|---|
| **Starting Position** | | |
| **Direction of Travel** | | |

5. Construct a position vs. time graph for the two buggies below. Use this graph to determine where and when the buggies will collide.

6. Construct a motion map for the two buggies using the axis below. Label the position where they will collide.



## Part III

Open the simulation: https://tinyurl.com/U2-Colliding-Buggies.

7. Complete the next-x function in the starter code so that it can be used to move both cars. Use a function design to help with this process if necessary. Write the contract, purpose statement, examples and function used in the space below.

8. Once the next-x function works, test the simulation for at least three sets of initial conditions. Does it behave as expected? Why or why not?

9. What assumptions does this simulation make? How might those assumptions affect the predictions it makes?

## Part IV

10. Use the simulation to predict where and when the two cars will meet and record these values below.

$t_{predicted} =$ □ $x_{predicted} =$ □

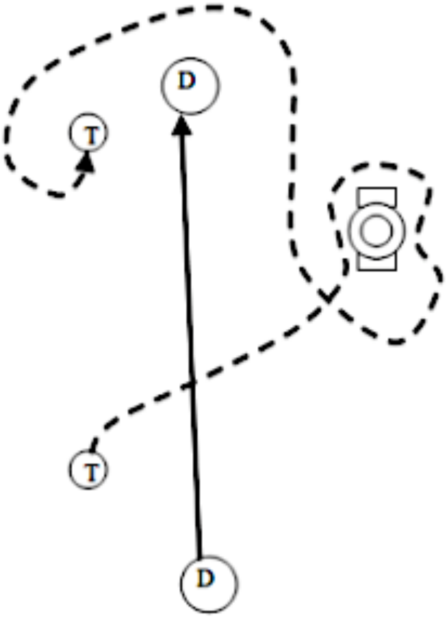
11. Test this prediction using the buggies. Run at least three trials and record all results in the space below.

12. How well does the simulation model the motion of the two buggies? Consider the following points:
    - How reliable were the experimental results?
    - How might the assumptions of the simulation affect the predictions?
    - What assumptions did you make in your initial conditions? How might they have affected your results?

# Unit 2 Worksheet 4: Distance vs. Displacement

## Part I

An overhead view of the paths that Dorothy and Toto take along the yellow brick road is shown below.



From start to finish, who travels farther? Justify your answer.

Find a classmate with a different answer. Why did they choose this answer?

New Terms and Definitions:

More examples using the new terms:

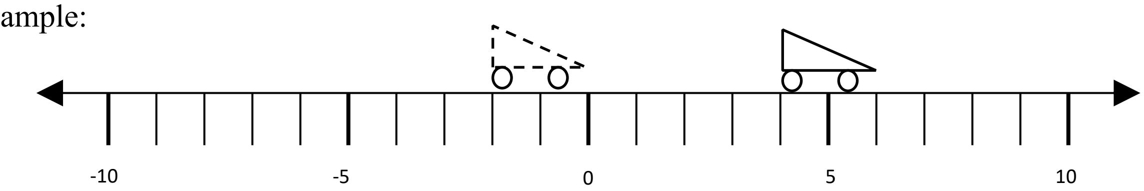
## Part II

**Using Number Lines to Measure Position and Finding Change in Position**

On each number line, there are two cars: an initial car (dashed) and a final car (solid).

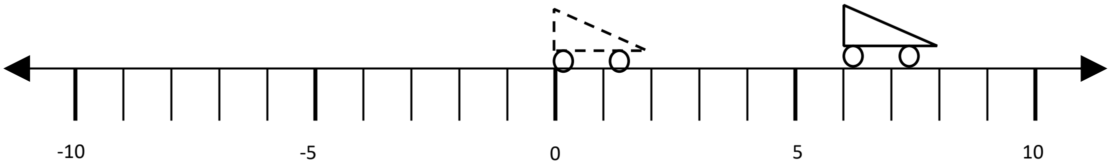
- Record the positions of each car.
- Write a mathematical expression for calculating the change in position of the car
- Record the change in position of the car.
- Measurements are in meters (m)

Example:



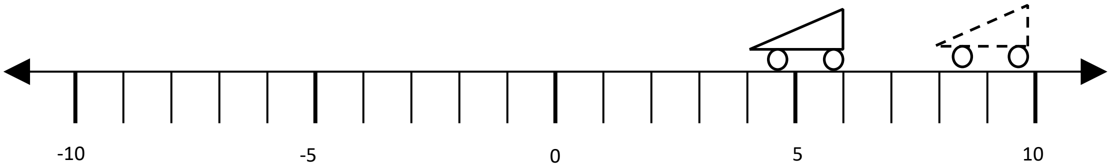
Initial Position = 0 m
Final Position = 6 m
Mathematical Expression: $x_f - x_o$ = 6 m - 0 m
Change in Position = 6 m

1.



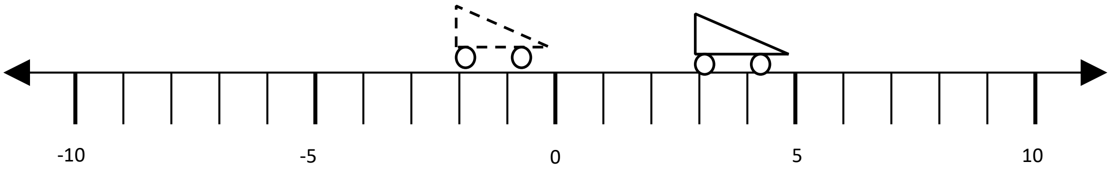
Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

2.



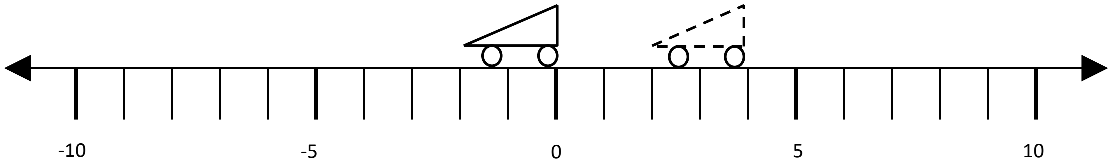
Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

3.



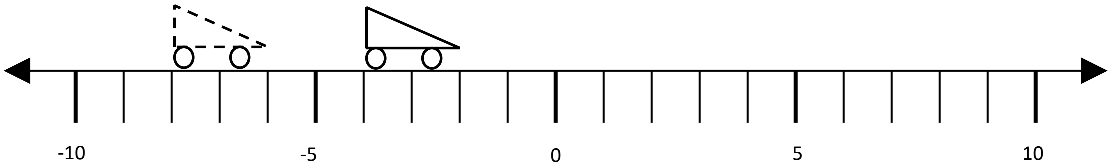
Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

4.



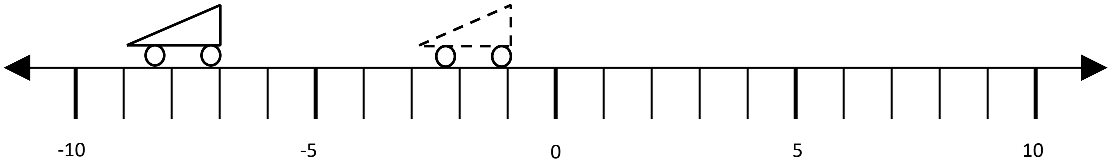
Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

5.



Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

6.



Initial Position =
Final Position =
Mathematical Expression:
Change in Position =

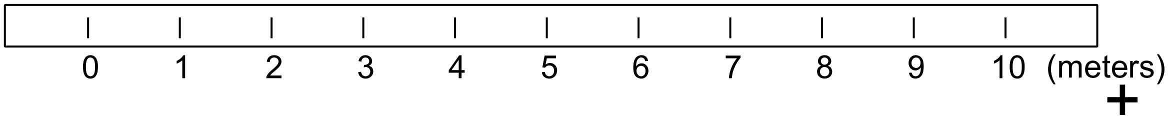
# Unit 2 Reading 1: Vocabulary of Motion

The terms speed and velocity are terms that are commonly used when describing motion. However, we really have no way to measure either of these quantities directly. In order to describe any of these quantities, one must begin with more fundamental quantities. Specifically, we must measure position, which is where an object is located, and time, which is when the object is at a particular location. By considering where something is, when it is there, and tracking the position of an object as time rolls on, one can determine the values of an object's average speed and average velocity.

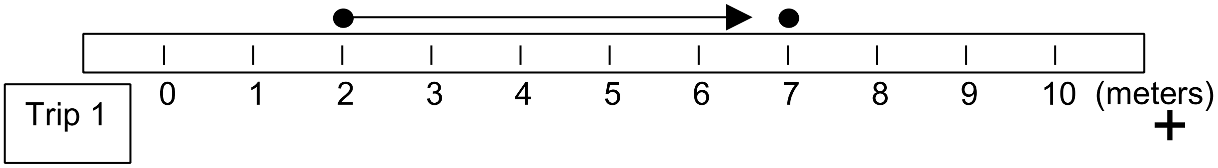
**Distance, Position, and Displacement**

| Name | Description | Symbol |
|---|---|---|
| ***Distance*** | How far something has traveled along some path | d |
| ***Position*** | Where something is located in some reference system. | $\vec{x}$ |
| ***Displacement*** | The difference between an object's starting position and its ending position. | $\Delta\vec{x}$ |

To illustrate the difference between these three quantities, consider a number line calibrated in meters. Assume that each numbered mark on the number line below is 1 meter from the one next to it. Let us also consider that we will call to the right the positive direction and to the left the negative direction.



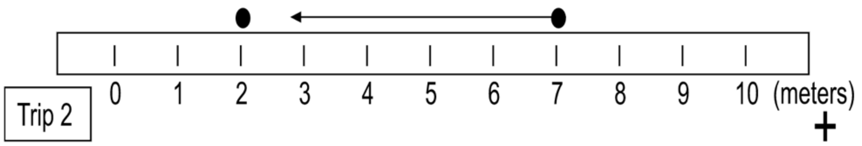
Now consider an object that takes a trip along the number line, beginning at a position of +2 meters, and ending at a position of +7 meters.



How would we describe the distance, position(s), and displacement associated with this trip?

Clearly, we must consider at least two positions—the starting position and the ending position, even though the object occupied many positions during its trip from the +2 meter position to the +7 meter position.

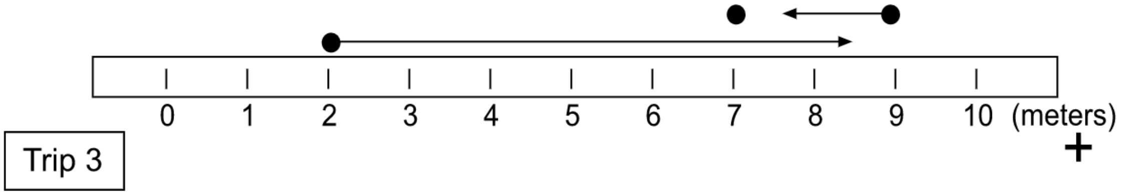
The object has a ***starting position*** of +2 meters.
The object has an ***ending position*** of +7 meters.
The object traveled a ***distance*** along the number line of 5 meters.
The object had a ***displacement*** of 5 meters in the positive direction or +5 meters.
What if the object reversed its trip, starting at a position of +7 meters and ending at a position of +2 meters? Which of the quantities above have changed as a result of this new trip?



The object has a ***starting position*** of +7 meters.
The object has an ***ending position*** of +2 meters.
The object traveled a ***distance*** along the number line of 5 meters.
The object had a ***displacement*** of 5 meters in the <u>negative direction</u> or -5 meters.

So you can see that while distance tells you only how far something has traveled, displacement tells you how far it is from the starting point to the ending point as well as which direction you would have to travel to go from the starting point to the ending point. For the first trip the distance traveled was 5 meters, and the same was true for the second trip. The displacements, however, were quite different, because the object traveled in two different directions for the two trips.

Finally, let us consider a more complicated trip along the number line in order to more fully illustrate the difference between distance and displacement. Consider an object that travels from a position of +2 meters to a position of +9 meters and then turns around and goes back to a position of +7 meters.



Let's consider the positions, distance and displacement for this trip.

The object has a ***starting position*** of +2 meters.
The object has an ***intermediate position*** of +9 meters, which is where it changed directions.
The object has an ***ending position*** of +7 meters.
The object traveled a ***distance*** along the number line of 9 meters.

In this example, the object traveled from a position of +2 meters to a position of +9 meters, a distance of 7 meters. It then traveled from a position of +9 meters to a position of +7 meters, a distance of 2 meters. If it traveled 7 meters and then 2 meters, its total distance traveled would be 9 meters.

The object had a ***displacement*** of 5 meters in the <u>positive direction</u> or +5 meters.

Even though the object traveled a considerably greater distance in this trip compared to the first trip (9 meters compared to 5 meters), the change in position or displacement was exactly the same:  5 meters in the positive direction. This is because the starting and ending positions are exactly the same for this trip as they were for the first trip. So you can see that distance takes into consideration the total amount traveled along a given path, whereas displacement is concerned only with where you started, where you ended, and which direction you traveled.

**Speed and Velocity**

This also brings us to the differences between speed and velocity. Let us assume that each of the three trips required 5 seconds to complete. What would the object's average speed and average velocity be for each trip based on the mathematical definitions of average speed and average velocity.

$$\bar{v} = \frac{d}{\Delta t}$$

The commonsense notion of speed is that it measures how fast something travels. This notion is correct. The average speed of an object is determined by dividing the distance traveled by the object by the time interval over which the object was traveling that distance.

Expressed as an equation,

$$Average\ speed = \frac{distance\ traveled}{elapsed\ time}$$

The symbol that we will use for average speed is $\bar{v}$. Using this symbol for average speed, $d$ for distance traveled, and $\Delta t$ for the time interval,
the equation for average speed would be expressed symbolically by

$$\bar{v} = \frac{d}{\Delta t}$$

Thus, we arrive at a measure, for a given trip by an object, of its average rate of travel (average speed) measured in some distance unit divided by some time unit. In the U.S. most speed limits are measured in the unit miles *per* hour, meaning the number miles that would be traveled *for every* hour of travel. In Mexico, the speed limit signs are in kilometers per hour (kilometers traveled for every hour). But speed can be measured in any distance unit divided by any time unit. Other examples could be meters 'for every' second, feet 'for every' second, or some outrageous unit like furlongs 'for every' fortnight or millimeters 'for every' millennium. Average speed tells us about the rate of travel but tells us nothing about the direction of travel.

To know both the rate of travel and the direction of motion, we rely on a quantity called velocity. In order to specify an object's velocity, we must describe both how fast it is traveling (its speed) and the direction in which it travels. Velocity is defined as the rate at which an object changes its position with respect to time.
Expressed as an equation,

$$Average\ velocity = \frac{change\ in\ position}{elapsed\ time}.$$

The symbol that we will use for average velocity is $\bar{\vec{v}}$.

Using this symbol for average velocity, $\Delta\vec{x}$ for change in position (displacement), and $\Delta t$ for the time interval, the equation for average velocity would be expressed symbolically as:

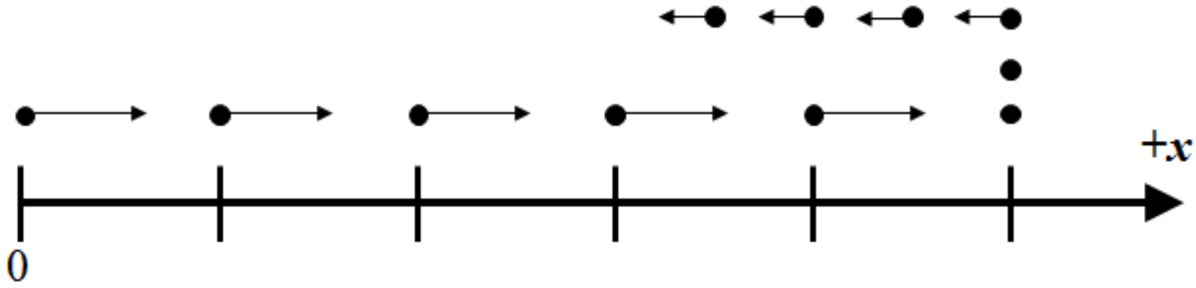
$$\bar{\vec{v}} = \frac{\Delta\vec{x}}{\Delta t}$$

A reasonable question might be: How is average velocity *different* from average speed? The difference lies in the difference between distance and displacement.

The table below shows the calculation of average speed and average velocity for Trips 1, 2 and 3 as shown in the previous pages.

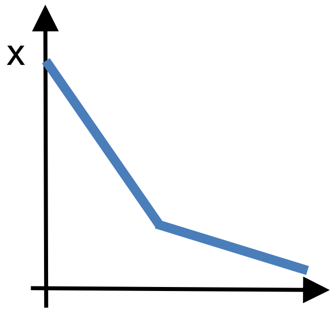
| Trip | Time Interval (s) | Distance Traveled (m) | Change in Position (m) | Average Speed (m/s) | Average Velocity (m/s) |
|---|---|---|---|---|---|
| | | *total path length traveled* | *displacement* | $\frac{\textit{distance traveled}}{\textit{elapsed time}}$ | $\frac{\textit{displacement}}{\textit{elapsed time}}$ |
| 1 | 5 | 5 | + 5 | $\frac{5}{5}=1$ | $\frac{+5}{5}=+1$ |
| 2 | 5 | 5 | - 5 | $\frac{5}{5}=1$ | $\frac{-5}{5}=-1$ |
| 3 | 5 | 9 | + 5 | $\frac{9}{5}=1.8$ | $\frac{+5}{5}=+1$ |

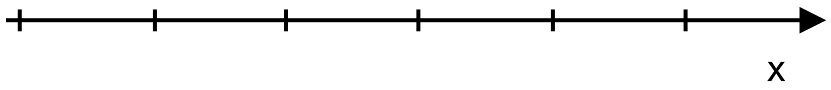
These instances in which distance and displacement are not equal can be represented using a motion map as well. Such an example is represented by the following motion map. Here, an object starts at a position of zero, moves to the right at constant velocity for five seconds, stops and remains in place for two seconds, then moves to the left at a slower constant velocity for three seconds.

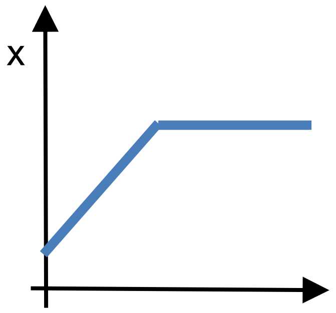


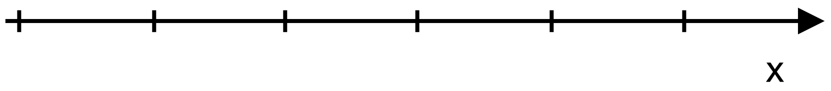
Note that there are eleven dots in this motion map. Those eleven dots represent a total time interval of ten seconds. Why do you think eleven dots, each representing positions at clock readings that are one second apart, are necessary to represent a time interval of ten seconds?

What would the motion map look like for each of the following graphs?

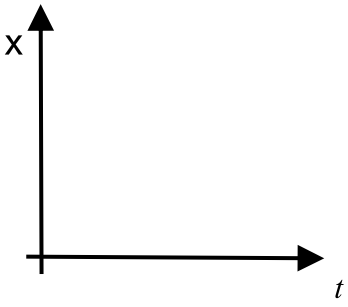


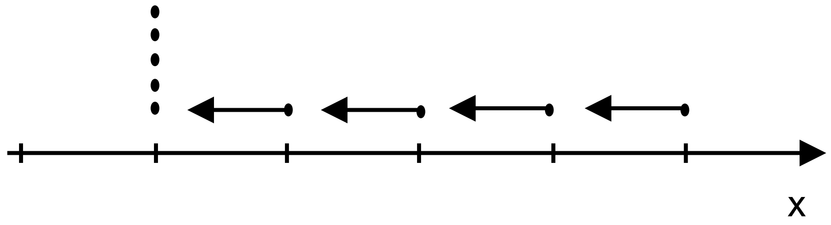






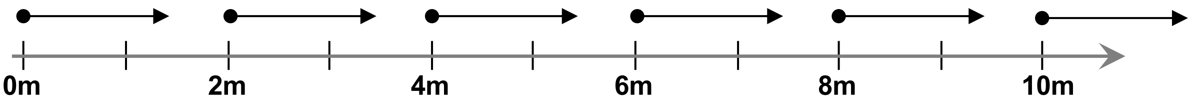
What would the graph look like for the following motion map?





# Unit 2 Worksheet 5: Velocity-Time Graphs

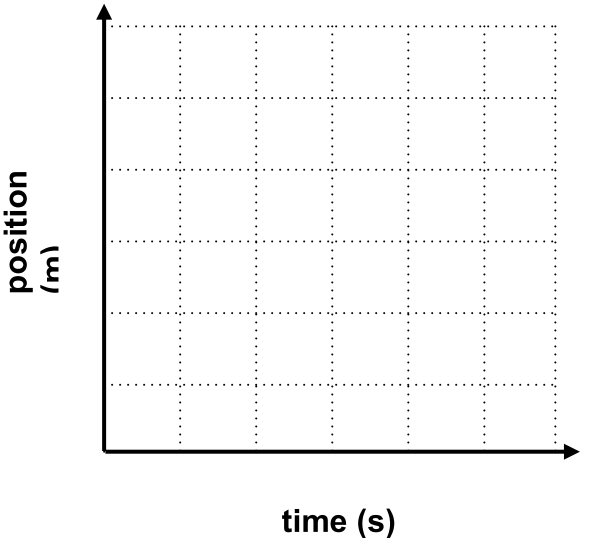
1. This motion map shows the position of an object once every second. From the motion map, answer the following:

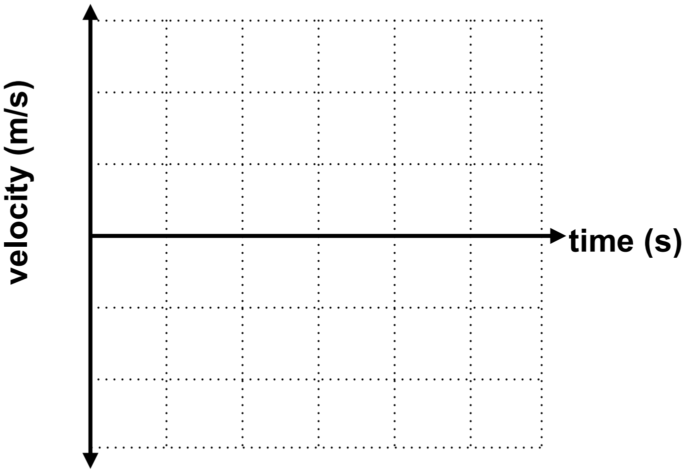


a. Describe the motion of the object.

b. Represent the motion with a quantitative **x** vs. **t** graph.

c. Represent the motion with a quantitative **v** vs. **t** graph.





d. Write a mathematical representation for the relationship between position and time.

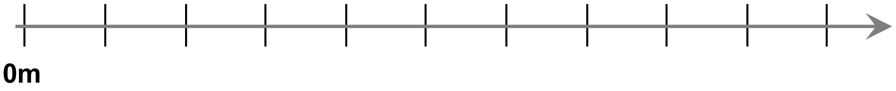
e. From the position-time graph find the displacement from t = 1 s to t = 3 s.

f. Find the area under the velocity-time graph from t = 1 s to t = 3 s. What are the units of this area? What does this area represent?

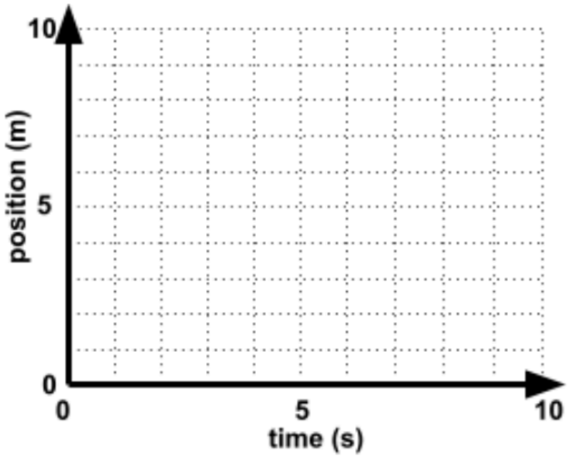
2. From the position vs. time data below, answer the following questions.

| t (s) | x (m) |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 4 |
| 4 | 7 |
| 5 | 10 |
| 6 | 10 |
| 7 | 10 |
| 8 | 5 |
| 9 | 0 |

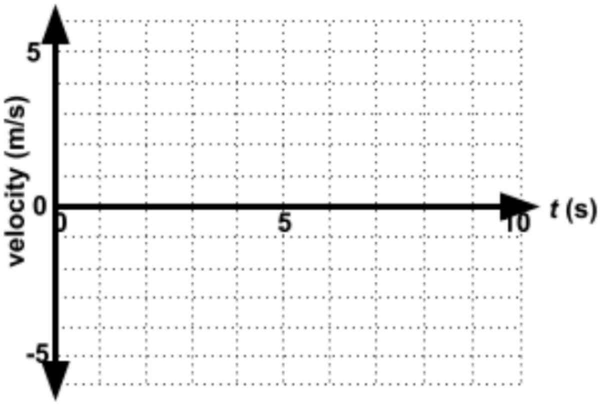
a. Draw a motion map for the object.



b. Construct a graph of position vs. time.



c. Construct a graph of velocity vs. time.



d. Determine the displacement from t = 3.0 s to 5.0 s using the velocity vs. time graph.

e. Determine the displacement from t = 7.0 s to 9.0 s using the velocity vs. time graph.

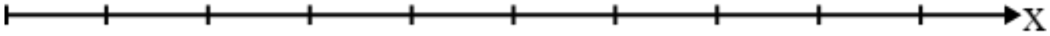
f. Determine the average **velocity** from t = 4 s to 8 s.

g. Determine the average **speed** from t = 4 s to 8 s.

# Unit 2 Activity 4: Multiple Representations of Motion

In each of the boxes below, complete each representation of the motion given in the data tables.
1.

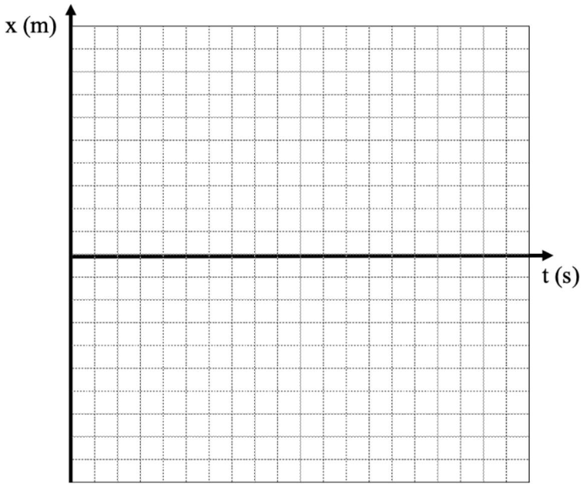
**Motion Map:**

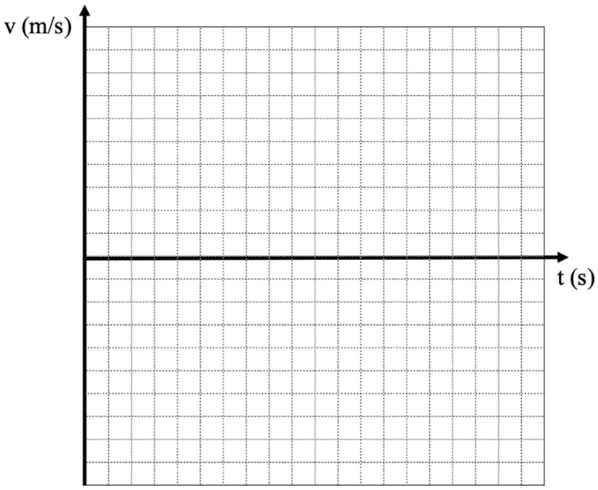


**Computational Representation:**

```
# Contract



examples:







end

fun





end
```

**Graphical Representation:**



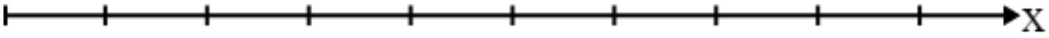


**Verbal Representation:**

**Mathematical Representation:**

2.

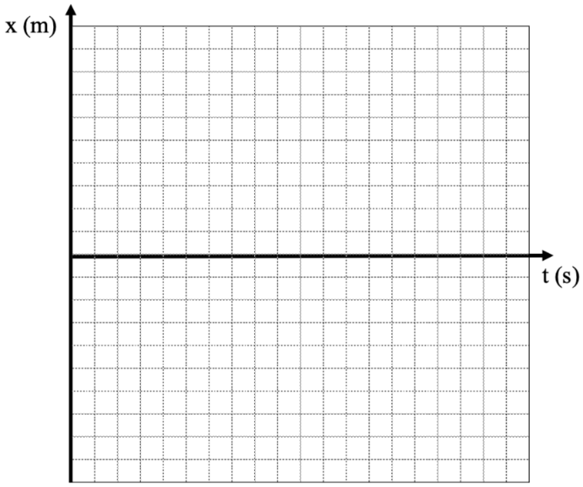
**Motion Map:**

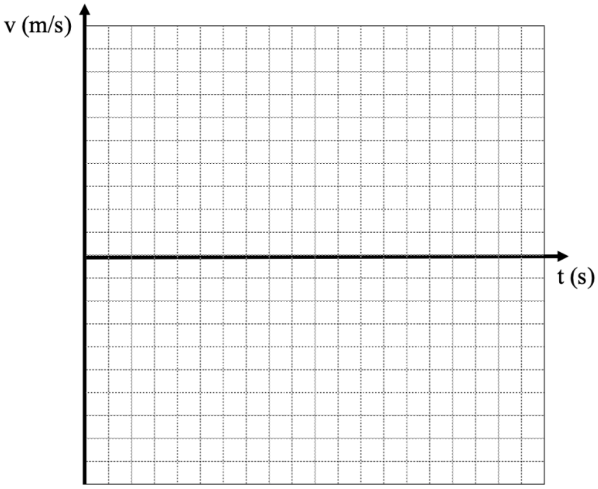


**Computational Representation:**

```
# Contract



examples:









end

fun





end
```

**Graphical Representation:**



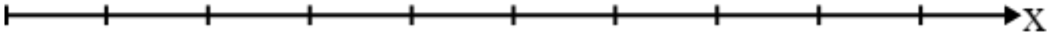


**Verbal Representation:**

**Mathematical Representation:**

3.

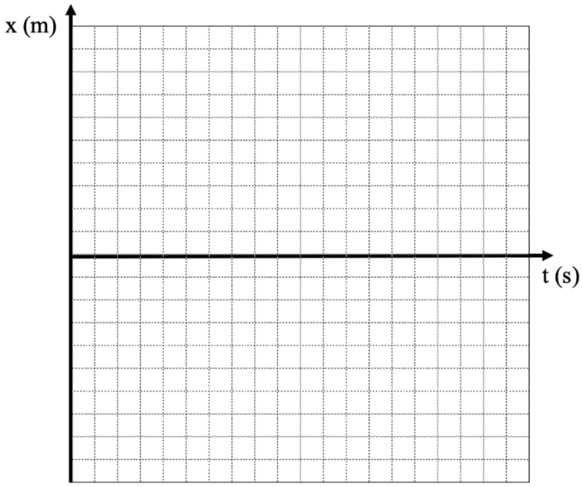
**Motion Map:**

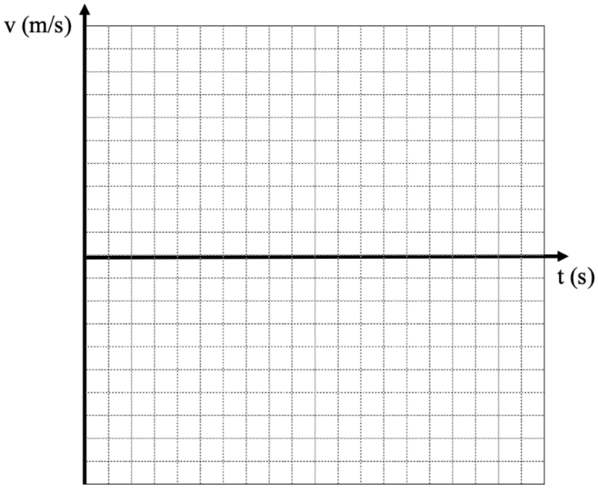


**Computational Representation:**

```
# Contract



examples:








end

fun





end
```

**Graphical Representation:**



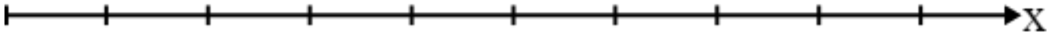


**Verbal Representation:**

**Mathematical Representation:**

4.

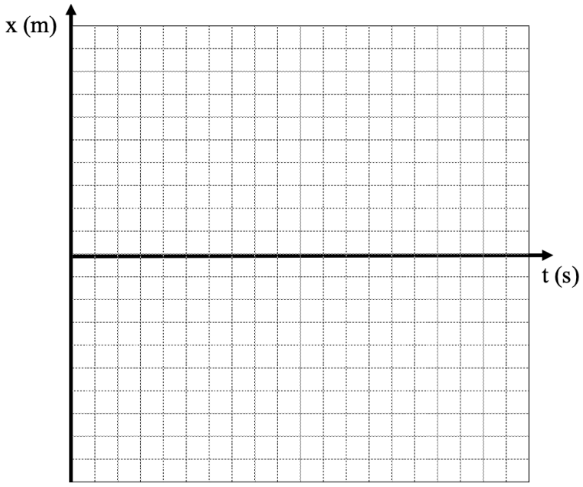
**Motion Map:**

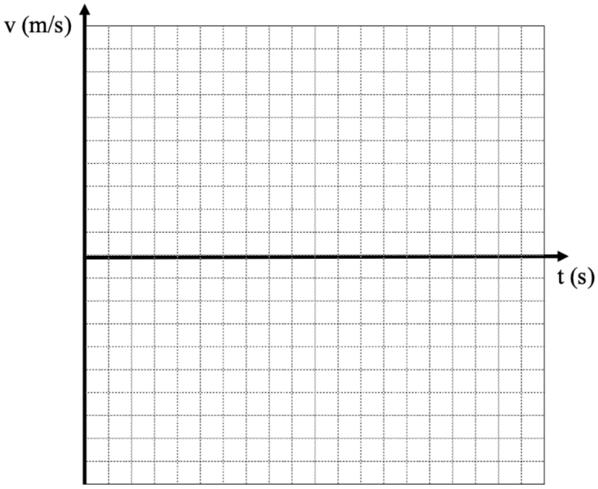


**Computational Representation:**

```
# Contract



examples:









end

fun





end
```

**Graphical Representation:**





**Verbal Representation:**

**Mathematical Representation:**

# Unit 2 Activity 4 Teacher Resource: Multiple Representations of Motion

Use the following four data tables for the four cases of motion students represent in this activity.

**1.**

| *t (s)* | *x (m)* |
|---|---|
| 0 | -14 |
| 3 | -5 |
| 6 | 4 |
| 9 | 13 |
| 12 | 22 |
| 15 | 31 |

**2.**

| *t (s)* | *x (m)* |
|---|---|
| 0 | 12 |
| 2 | 18 |
| 4 | 24 |
| 6 | 30 |
| 8 | 36 |
| 10 | 42 |

**3.**

| *t (s)* | *x (m)* |
|---|---|
| 0 | -4 |
| 5 | -19 |
| 10 | -34 |
| 15 | -49 |
| 20 | -64 |
| 25 | -79 |

**4.**

| *t (s)* | *x (m)* |
|---|---|
| 4 | 8 |
| 8 | 2 |
| 12 | -4 |
| 16 | -10 |
| 20 | -16 |
| 24 | -22 |

# Unit 2 Activity 5: Rocket Lander Game

At the end of this and the next three units you will be building your own video game. In this game, you will program the motion of a rocket as it lands on the moon.

For this unit, your goal is to write the function `next-x` that will allow the rocket to move smoothly across the screen. The rocket should move horizontally and at a constant velocity.

The starter code can be found at https://tinyurl.com/Rocket-Lander-Game. You will see a message stating that this is a shared program and you must save a copy in your own drive to save any changes. Do this first.

1. Looking at the initial parameters in lines 7 through 12 of the starter code, which might affect the horizontal motion of the rocket? How would it affect the motion?

2. Run the program. What do you see?

3. You will see that there are sections of comments in the starter code. For this unit, find

```
############
#  Unit 2  #
######################################################################
# Add your two-argument next-x function, along with its contract     #
# and examples. THEN in the make-lander function at the end of the   #
# code, change default-next-x to next-x.                             #
# Confirm that the rocket moves in the way you expect.               #
######################################################################
```

4. Complete a Function Design, keeping in mind the initial parameters you listed in #1.

5. Add this function to the starter code in its appropriate place. Before you run the program, edit the inputs to the last function to change `default-next-x` to read `next-x`.

6. Did you receive feedback? If so, how do you need to change your written code?

7. Find the following comment block:

```
######################################################################
# Your lander has probably drifted across the window and off of      #
# the right side of the screen!                                      #
#                                                                    #
# Design a function called is-onscreen-x which consumes the          #
# lander's current x position and returns true whenever the          #
# lander's current x position is actually visible on screen.         #
#                                                                    #
# THEN in the make-lander function at the end of the                 #
# code, change default-is-onscreen-x to is-onscreen-x.               #
# Confirm the rocket stays on screen (and returns)                   #
# in the way that you expect.                                        #
######################################################################
```

8. For which horizontal positions is the rocket visible on screen? For which horizontal positions is the rocket not visible on screen?

9. What information would the function `is-onscreen-x` need to consume?

10. Complete a Function Design (with Booleans) for an `is-onscreen-x` function. Once you have a completed Function Design, enter the Contract, Examples, and Definition for the `is-onscreen-x` function below the comment above, and run the program.

11. Did you receive feedback? Did the code highlighted by the feedback message include your mistake? What did you need to do to make the program run as you intended?

# Function Design

## Defined Identifiers

_______________ = _______________ # _______________
Identifier Value Units

_______________ = _______________ # _______________
Identifier Value Units

_______________ = _______________ # _______________
Identifier Value Units

## Physical Interpretation

What will the input(s) of your function be? _______________ (ex: side length)

What will the units of each input be? _______________ (ex: meters)

What will the output be? _______________ (ex: area)

What will the unit of the output be? _______________ (ex: square meters)

## Contract and Purpose Statement

_______________ :: _______________ -> _______________
Function Name Domain (Input) Type(s) Range (Output) Type

_______________
What does the function do? (The function consumes ______ and produces _______.)

## Examples

```
examples:
    _______________(          ) is _______________
    Function Name  Example Input(s)  What calculation(s) must be performed?
    _______________(          ) is _______________
    _______________(          ) is _______________
end
```

## Definition

```
fun _______________(               ):
    _______________
    What calculation must be performed with the named input(s) to produce the desired output?
end
```

# Function Design with Booleans

## Defined Identifiers

________________ = ________________ # ________________

Identifier Value Units

________________ = ________________ # ________________

Identifier Value Units

## Physical Interpretation

What will the input(s) of your function be? ________________ (ex: side length)

What will the units of each input be? ________________ (ex: meters)

What will the output be? ________________ (ex: area)

What will the unit of the output be? ________________ (ex: square meters)

## Contract and Purpose Statement

```
________________ :: ________________ -> ________________
Function Name       Domain (Input) Type(s)   Range (Output) Type
```

________________________________________

What does the function do? (The function consumes ______ and produces _______.)

## Examples

```
examples "________________________________________":
    ________ (        ) is  true   because ________________
    Function Name  Example Input(s)  Expected Output  What calculation must be performed?
    ________ (        ) is  true   because ________________
end
examples "________________________________________":
    ________ (        ) is  false  because ________________
    Function Name  Example Input(s)  Expected Output  What calculation must be performed?
    ________ (        ) is  false  because ________________
end
```

## Definition

```
fun ________ (                ):
    Function Name   Input Name(s)

    ________________________________________
    What calculation must be performed with the named input(s) to produce the desired output?

end
```