



HOFSTRA UNIVERSITY

Center for STEM Research

Computer Science Through Engineering Design

Designing a Roller Coaster Student Edition 2026



GRANT NUMBER – 2341962



*Computer Science through
Engineering Design*

Roller Coaster Design – Student Version

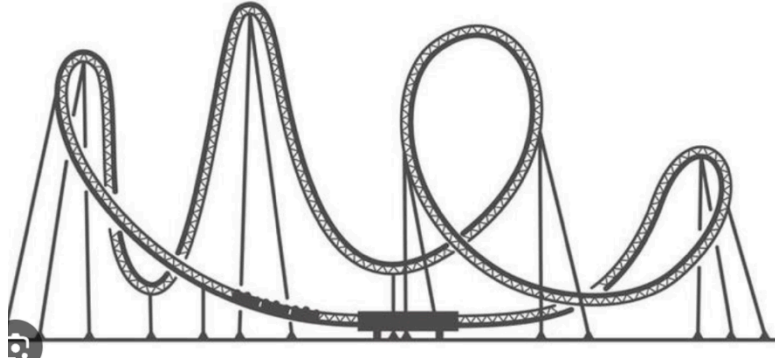
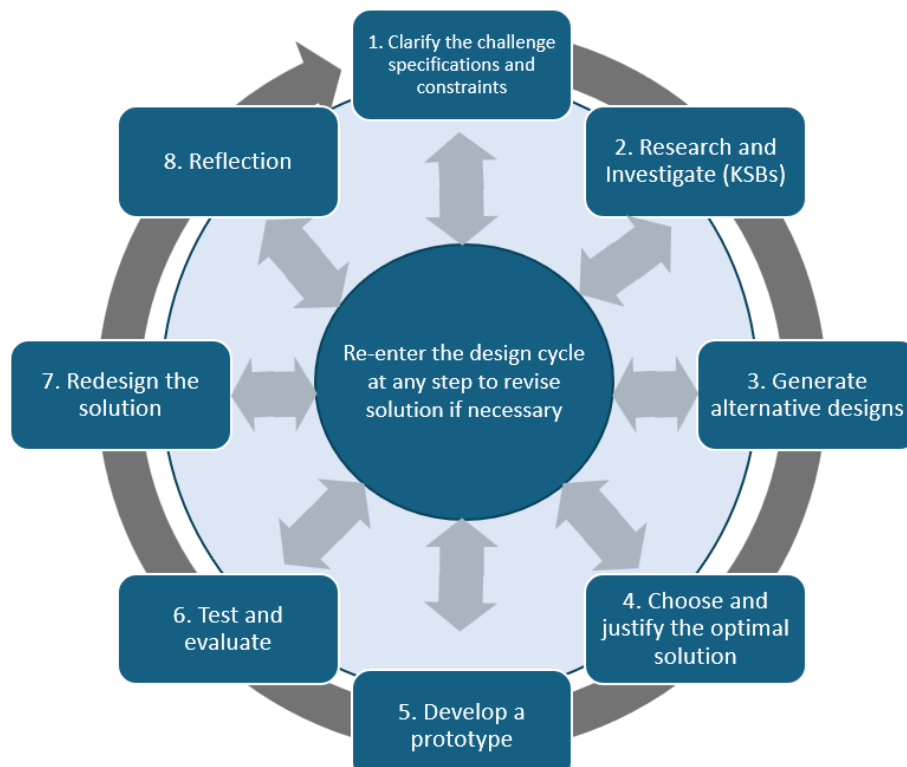


Figure 1. Schematic of a Roller Coaster

Challenge:

Roller coasters can be fun: speeding down a hill, slowing climbing up another one, and looping around curves. In this roller coaster model design, a marble takes the place of a passenger and must safely stay on the track over the first and second hills, a loop and on to the end. You will be using materials provided by your teacher: a marble (passenger), foam insulation (track), and canister (end). After building and testing the roller coaster ride, you will add sensors and code to determine the marble's velocity over the course of the ride.



Specifications (what the design must do)



Work as a class to develop the Specifications for your rollercoaster. Record your specifications in the box below:

Constraints (things that limit your solution)

- 1) The design is limited to materials provided in class.
- 2) The design should include two hills and one loop.
- 3) The marble must stay on the track for the duration of the run and end in the canister, triggering the sound sensor.
- 4) The height of the initial hill may be changed to increase or decrease potential energy.
- 5) The curvature of the foam loop is critical to the marble's ability to stay on the track.

Materials

Sturdy base for first starting surface

Tube or can with one closed end (ex. oatmeal canister, Pringles™ or juice can)

Foam pipe insulation

Duct tape

Hummingbird Bit controller, sound sensor and connected laptop computer

Hot glue gun and glue sticks

Scissors or Canary shears

Marbles

Meterstick or metric tape measure

Markers/Art Supplies to decorate the roller coaster (optional)

Calculator and stopwatch (optional)

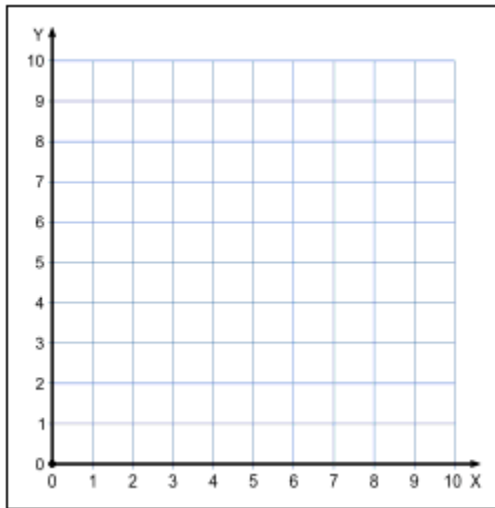
Knowledge and Skills Builders (KSBs)

KSB 1 Preliminary Design



The roller coaster you are designing will not be as complex as an actual one.

Think about where you want to locate the first hill, the loop and the second hill, and how you will secure the foam tubes to the base while you are building and testing. Make a sketch of your first design here using the X-Y coordinate plane concept where Y is the height of your model and X represents the distance travelled (length of track):



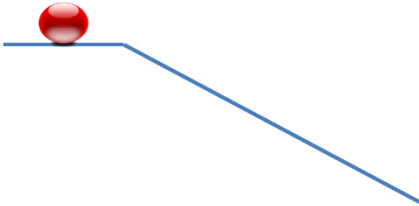
KSB 2 Potential and Kinetic Energy



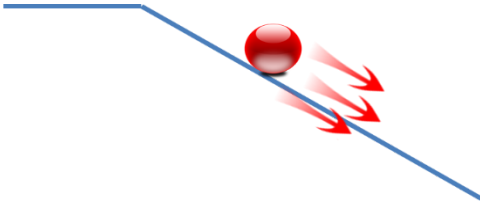
Your design must consider **potential** and **kinetic** energy, and how one changes into the other.

Potential energy is stored energy. The marble at the top of a hill has potential energy based on its mass, height above the surface, and acceleration due to gravity. The value of the initial potential energy is found by the formula **PE = (mass) x (9.8 meters/sec²) x (Height of first hill in meters)**

The only variable you can change to increase the initial potential energy is the height of the first hill.



Kinetic energy is the energy of a moving object.



When traveling uphill, the object slows down due to gravity. Kinetic energy **decreases** and potential energy **increases**.

When traveling downhill, the object accelerates due to gravity. Kinetic energy **increases** and potential energy **decreases**.

In a perfect system, the amount of potential energy becomes the same amount of kinetic energy (“Law of conservation of energy”). However, in the real world, this is not always the case.

What might cause a decrease in the amount of energy that is transferred from potential energy to kinetic energy?

KSB 3 Sound Sensor



A sensor detects something in the outside world and records a value. This value can be read as an input to the computer and used to make decisions or actions.

The sound sensor used with the Hummingbird Bit is a receiver that detects how much sound is around the sensor. The range of this sound sensor's output is a value from 0 to 100.



Here is a tutorial about how to use the sound sensor and code in the Hummingbird Bit environment:

<https://learn.birdbraintechnologies.com/hummingbirdbit/snap/program/16-1>

Get familiar with the sound sensor and how it works by creating and testing code such that:

When the sound sensor records a value greater than 70, an LED light will turn on. Remember that the value needed to trigger the sound sensor will vary based on the amount of ambient noise around the rollercoaster. This value can easily be adjusted to allow the sensor to detect the sound of the marble striking the can.

Based on the values that the sound sensor is reporting what value would you select to trigger the sound sensor?

Take a picture of your sound sensor/LED code and paste it here.

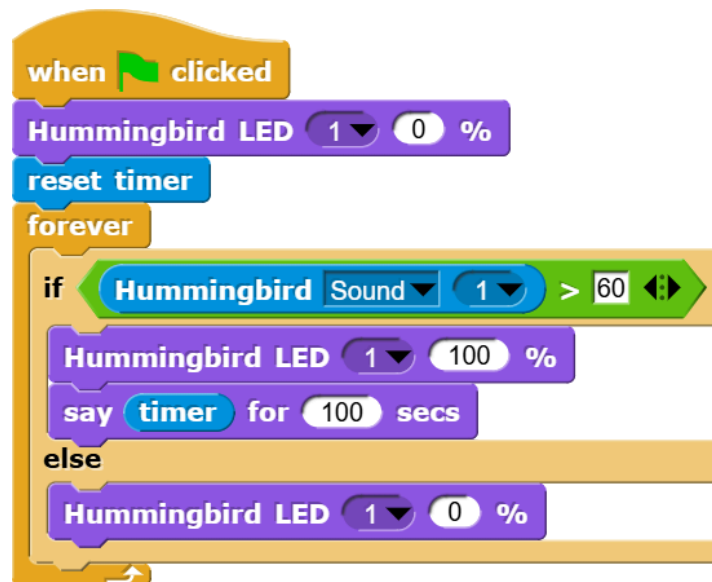
KSB 4 Timer



The timer function in the Hummingbird code needs something to start and end its function. To start the run, one person will release the marble at the top of the first hill while another simultaneously starts the code by pressing the Green Flag. The end occurs when the marble crosses “the finish line” by entering the canister and hitting the canister bottom.

The canister is taped to enclose the end of a flat track section. As the marble hits the canister bottom, it generates a sound in the area where the sound sensor is located. The sound sensor detecting the sound triggers code that uses the value of timer to calculate velocity.

Here is an example of a timer that causes the LED to turn on and captures how long it takes the marble to reach the canister.



In this situation, when the Green Flag is clicked, the timer function resets and begins counting. When the marble hits the container, the sound sensor detects the “hit” as greater than 60 and the LED lights up. The value of the timer will be displayed for 100 seconds on the laptop.

The example code above uses the “reset timer” block and “timer” block to determine how long it takes the marble to travel through the course.

reset timer

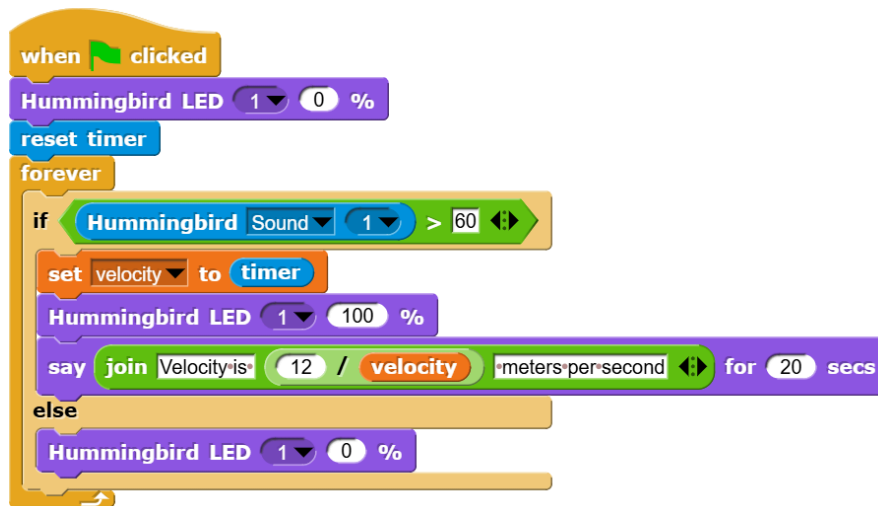
timer

When the marble is released and the Green Flag is clicked the “reset timer” block sets the value of “timer” to zero. The timer acts as a stopwatch by increasing the value over time. When the marble reaches the end of the run and whacks the bottom of the can and the sound sensor detects the loud sound the value in “Timer” will be reported.

Velocity

The velocity of the marble can be calculated by dividing the length of the track, in meters, by the elapsed time.

Here is an example of code that might be used to calculate the velocity of the rollercoaster if the track is 12 meters long.



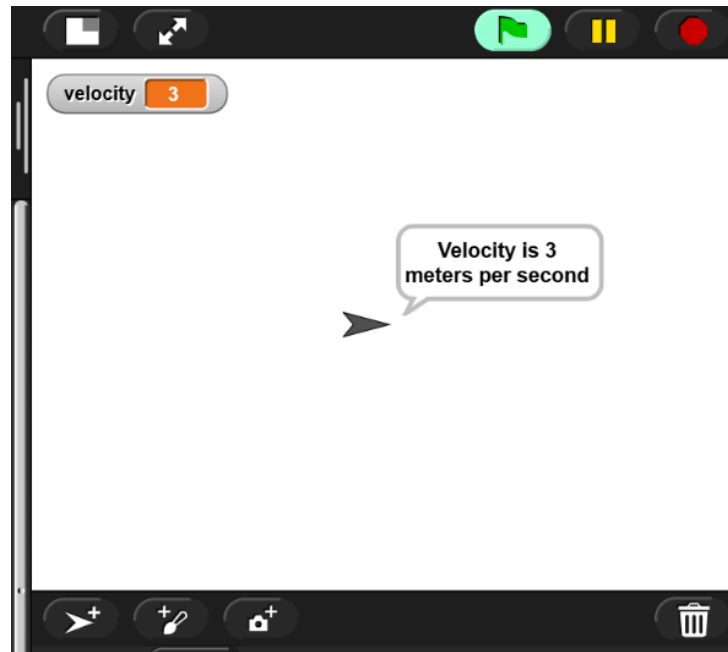
This time can be used to calculate velocity. The distance is the length of foam tubing which should be measured, however, when two six-meter sections are joined the length for the marble to travel is 12 meters. Hence,

$$\text{Velocity} = 12/\text{timer}$$

If you want the velocity to be displayed on the screen, then more programming is needed. The following blocks will be helpful.



The first block defines velocity, the second block allows the information in Hello! to be displayed on the screen for 2 seconds. It can be displayed longer if you wish. The third block allows a phrase such as “Velocity is 3 meters per second” to display on the stage.



Generate Alternative Designs



Consider multiple designs that would be used to create a successful rollercoaster.

What would happen if the height of the first hill changes?
--

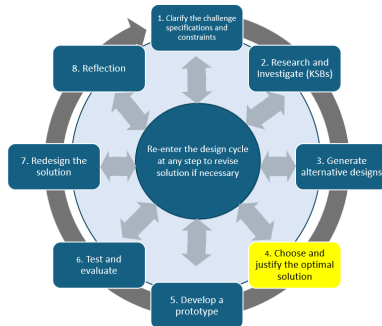
What shape and size would be best for making a loop?
--

Consider a variety of ways you could create code to calculate velocity of the marble.

What variables will we need?

What code will work best to calculate velocity?

Choose the Optimal Solution



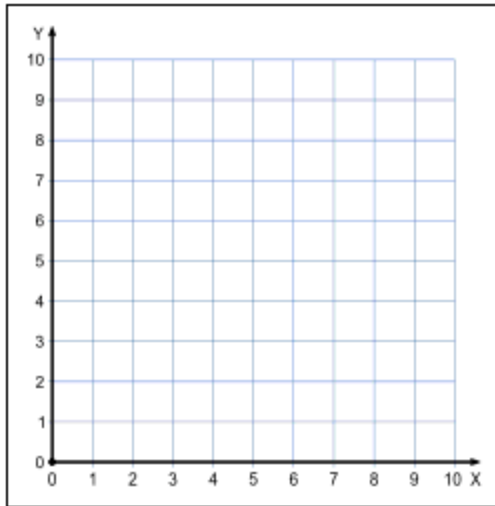
Decide on an optimal solution for your rollercoaster and code that will be used to calculate velocity.

Construct the Rollercoaster



Using the materials provided and your preliminary design, construct your rollercoaster.

Now observe if your marble stays on the track reliably; make any necessary adjustments. Pay particular attention to the curvature of the loop; sometimes it needs to be more like an oval than a circle. Sketch your final design and mark any problem areas here:



Make sure to:

Measure the length of your track in meters.

Measure the height of the initial hill in meters.

You will enter these values in the chart below.

Designing Your Roller Coaster Code



Once you have built a Roller Coaster model that will allow a marble to safely complete a run, you will add the Hummingbird Bit and sound sensor, then write the code on the laptop.

Take a picture of your initial code solution and paste here.

A large empty rectangular box with a black border, intended for pasting a picture of the initial code solution.

Test and Evaluate: Your Roller Coaster



Now you can modify your initial design based on what you have learned from testing the model.

Use this method consistently: When the marble is released, a team member simultaneously clicks the green flag. The timer code runs to determine the time it takes for the marble to reach the canister and trigger the sound sensor.

The distance the marble travels is constant. The velocity in meters/second is calculated from the distance divided by the time it takes to travel to the end (timer value).

Keep track of the distance, height of first hill, and velocity in a table:

Trial Run	Height of first hill (meters) = constant for 3 trials; may be varied in other runs	Distance (meters) = constant	Time (seconds) = experimental	Velocity (meters/second) calculated
1				
2				
3				

Take pictures of the final design of your rollercoaster and include them here:

EXTRA CREDIT:



Try raising or lowering the height of the initial hill and record the data as above:

Trial Run	Height of first hill (meters)	Distance (meters) = constant	Time (seconds) = experimental	Velocity (meters/second) = calculated
4				
5				
6				

What observation can you make by comparing the velocity achieved when a greater initial potential energy is present? Does it make the marble harder or easier to control on the track?

Take a picture of your modified rollercoaster and code. Paste it here. Hand in any extra credit results.

Why did you change the code from what you originally planned?

Test and Evaluate

What changes did you make to your preliminary design to arrive at your final solution?

Did your solution meet the design specifications? Briefly explain why it did or did not.

Reflection



What were two *new* or *unexpected* things you learned about coding from this activity?

What parts of the design were difficult for you to use or understand?

Did your solution meet the design specifications?